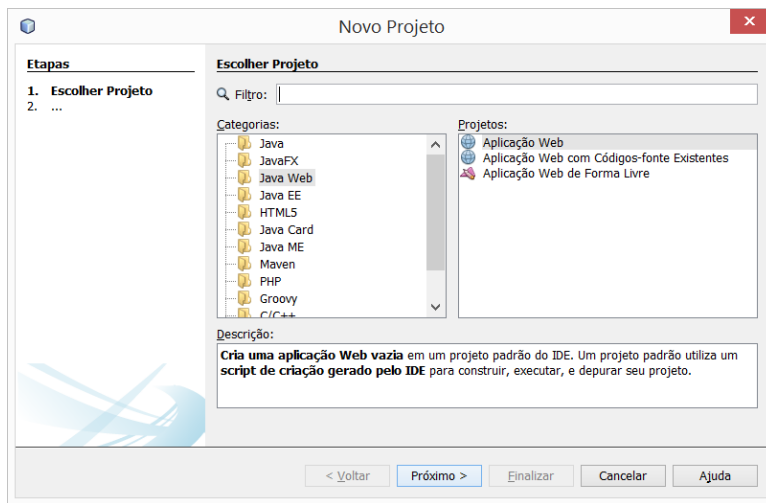
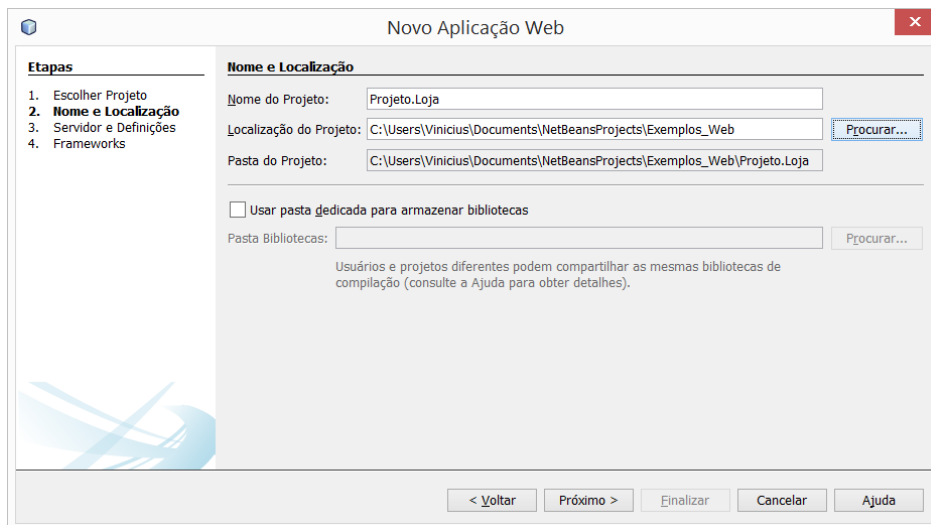


1) Iniciando um novo projeto Java Web na IDE NetBeans.

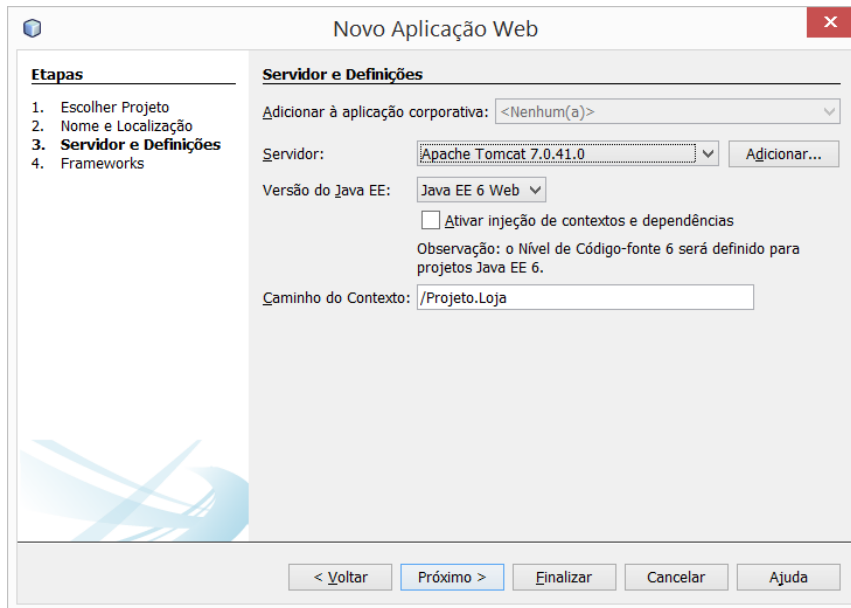
Arquivo > Novo Projeto > Java Web: Selecione a opção Aplicação Web, clique em próximo.



Informe o Nome do Projeto e o caminho onde será armazenado, e clique em próximo:



Informe o Servidor Web e a versão do Java EE e Clique em Finalizar:



Novo Aplicação Web

Etapas

1. Escolher Projeto
2. Nome e Localização
3. **Servidor e Definições**
4. Frameworks

Servidor e Definições

Adicionar à aplicação corporativa: <Nenhum(a)>

Servidor: Apache Tomcat 7.0.41.0 Adicionar...

Versão do Java EE: Java EE 6 Web

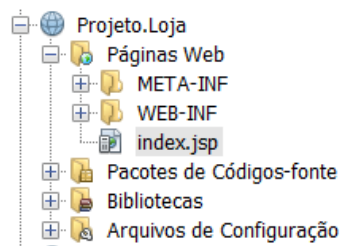
☐ Ativar injeção de contextos e dependências

Observação: o Nível de Código-fonte 6 será definido para projetos Java EE 6.

Caminho do Contexto: /Projeto.Loja

< Voltar Próximo > Finalizar Cancelar Ajuda

Então a seguinte estrutura será criada:

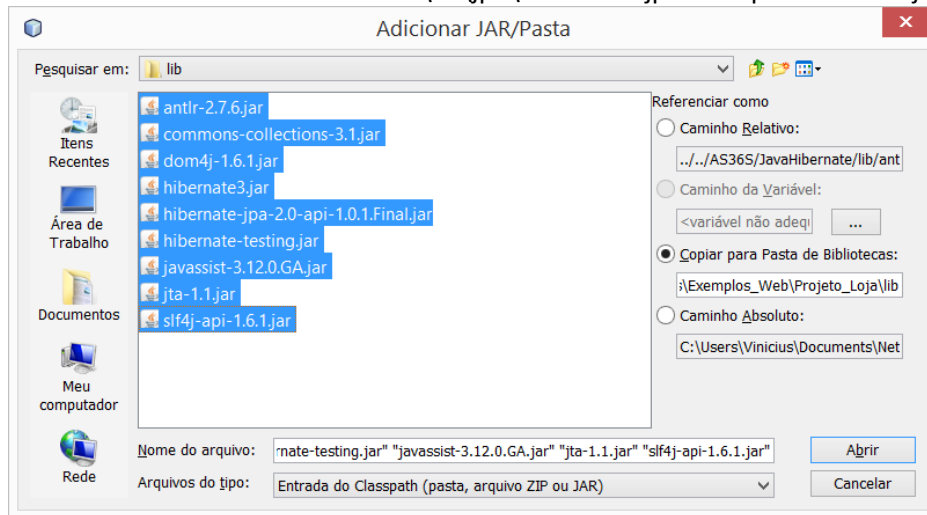


2) Adicionando as bibliotecas necessárias:

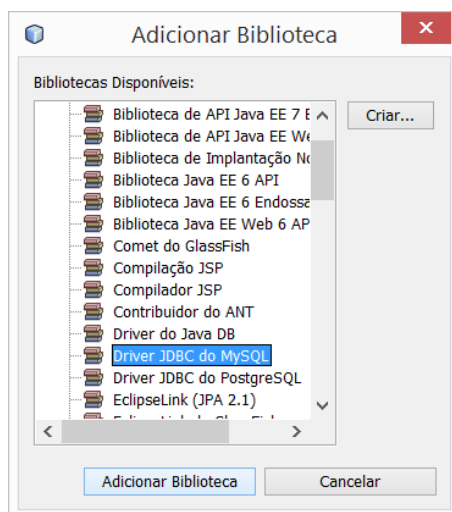
Baixe a versão 3.6.10 do Hibernate em:
<http://sourceforge.net/projects/hibernate/files/hibernate3/3.6.10.Final/>

Descompacte em um diretório. No NetBeans clique com o botão direito no projeto, selecione Propriedades, Bibliotecas e adicione os arquivos .jar abaixo:

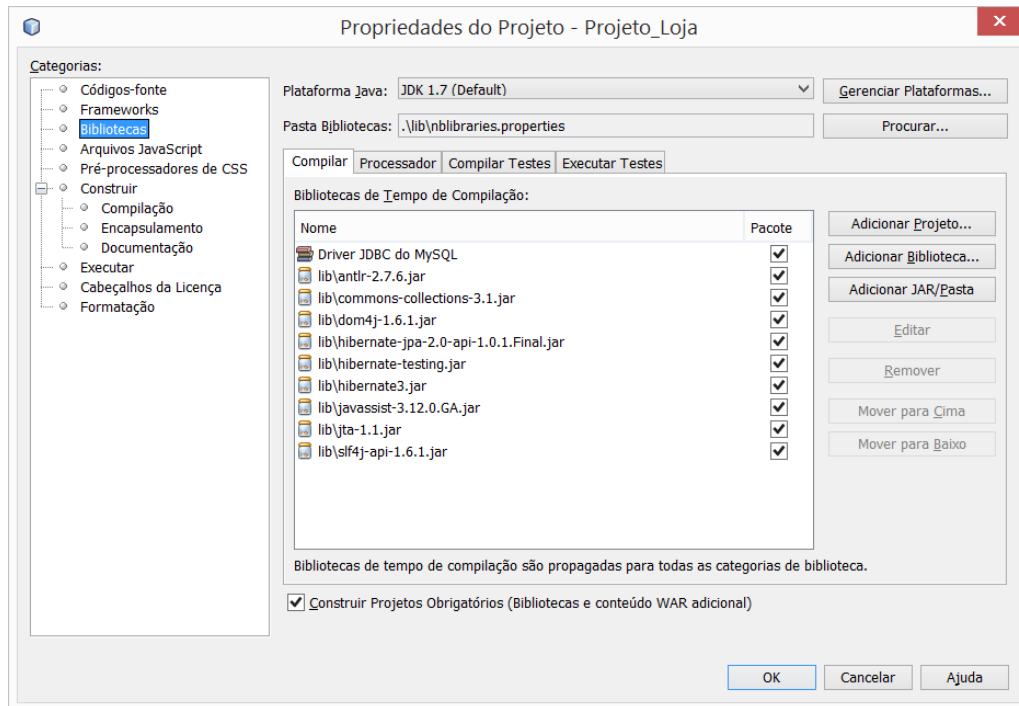
hibernate-distribution-3.6.10.Final\hibernate3.jar
hibernate-distribution-3.6.10.Final\hibernate-testing.jar
hibernate-distribution-3.6.10.Final\lib\required\antlr-2.7.6.jar
hibernate-distribution-3.6.10.Final\lib\required\commons-collections-3.1.jar
hibernate-distribution-3.6.10.Final\lib\required\dom4j-1.6.1.jar
hibernate-distribution-3.6.10.Final\lib\required\javassist-3.12.0.GA.jar
hibernate-distribution-3.6.10.Final\lib\required\jta-1.1.jar
hibernate-distribution-3.6.10.Final\lib\required\slf4j-api-1.6.1.jar
hibernate-distribution-3.6.10.Final\lib\jpa\hibernate-jpa-2.0-api-1.0.1.Final.jar



Adicione também a biblioteca de driver de conexão JDBC com o MySQL.



O resultado final com as bibliotecas adicionadas será esse:



3) Adicionando classes para trabalhar com JSON e Criptografia.

JSON

As informações sobre o uso das classes JSON com Java podem ser encontradas em:

<http://json.org/java/>

As classes estão disponíveis para download em:

<https://github.com/douglascrockford/JSON-java>

Serão utilizadas as classes: JSONArray.java; JSONException.java; JSONObject.java; JSONString.java; JSONStringer.java; JSNTokener.java; JSONWriter.java. Essas classes devem ser adicionadas dentro de pacotes de código fonte.

Criptografia

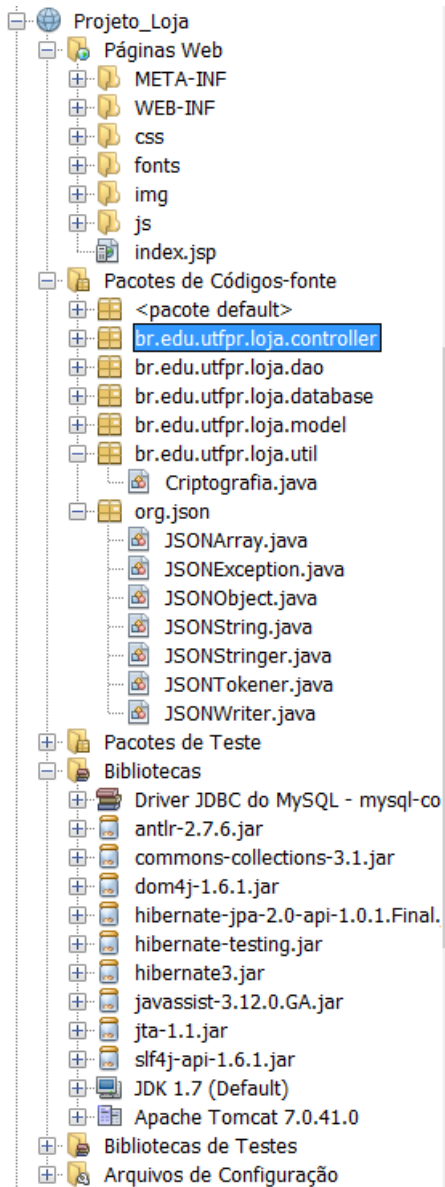
Será utilizada a classe Criptografia que implementa os algoritmos: MD5, SHA-1, SHA-256. A classe Criptografia.java deve ser criada no pacote util, com o código abaixo:

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.logging.Level;
import java.util.logging.Logger;
import sun.misc.BASE64Encoder;

public class Criptografia {

    public static String encripta(String senha) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-1");
            digest.update(senha.getBytes());
            BASE64Encoder encoder = new BASE64Encoder();
            return encoder.encode(digest.digest());
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(Criptografia.class.getName()).log(Level.SEVERE, null, ex);
            return null;
        }
    }
}
```

Abaixo está a estrutura do projeto com as classes adicionadas:

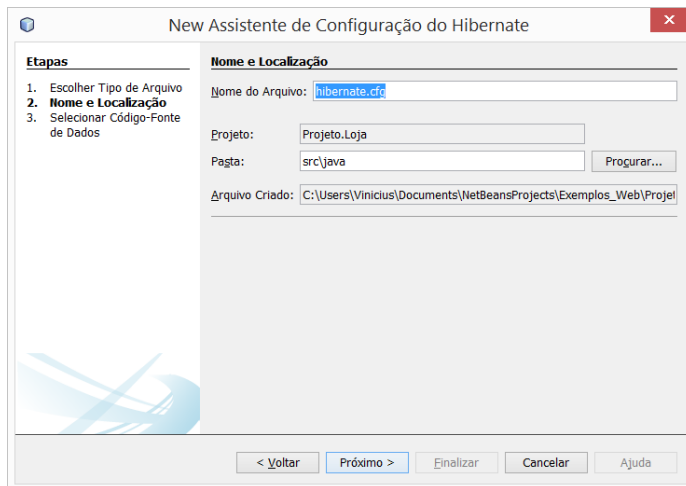


4) Utilizando Hibernate para persistir os dados.

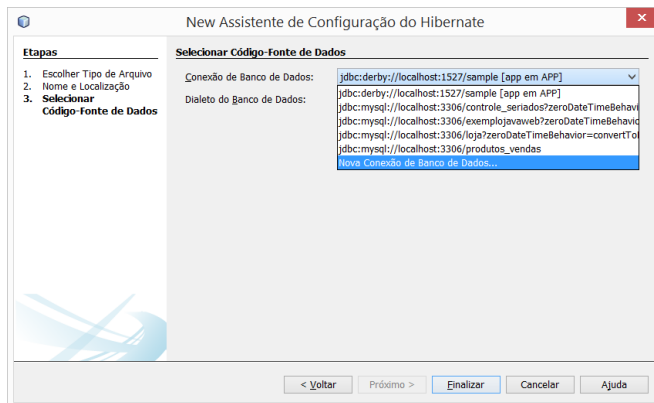
Criar o arquivo de configuração do Hibernate (Hibernate.cfg.xml):

Botão direito em Pacotes de Código Fonte> Novo> Outros> Hibernate> Assistente de Configuração do Hibernate, Próximo.

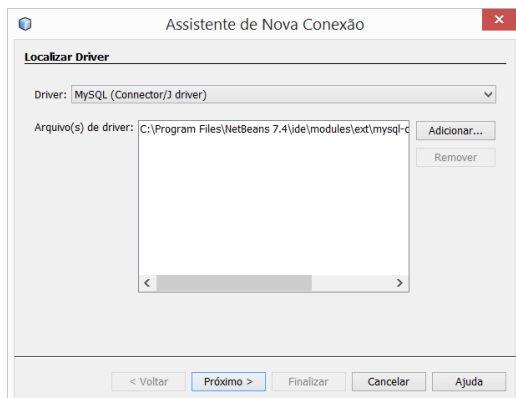
Informar o nome do arquivo: “hibernate.cfg”, clicar em próximo:



Na próxima tela, clicar em Nova Conexão de Banco de Dados..., isso vai permitir que seja configurada a conexão para o banco MySQL.



Na tela seguinte, selecione MySQL (Connector/J driver), clique em próximo



Informe os dados para conexão com o banco, host, porta, nome do banco de dados, usuário e senha, clique em Testar Conexão. Se a conexão for efetuada aparecerá a mensagem “Conexão Bem Sucedida” (clique em Finalizar), caso contrário verifique os dados informados.

Com tudo funcionando clique em finalizar na última tela e será criado o arquivo de configuração do Hibernate, contendo os dados para conexão com o banco.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/loja?zeroDateTimeBehavior=convertToNull</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">root</property>
    <property name="hibernate.show_sql">true</property>

  </session-factory>
</hibernate-configuration>
```

Criar a classe HibernateUtil, que é responsável por instanciar a SessionFactory do Hibernate:

Dentro do pacote **database**, botão direito do mouse, novo> hibernate> HibernateUtil. Em nome da classe informe: HibernateUtil e clique em Finalizar. A classe deve ficar com o seguinte código:


```
import org.hibernate.cfg.Configuration;
import org.hibernate.SessionFactory;
public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from standard (hibernate.cfg.xml)
            // config file.
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

5) Utilizando Hibernate para persistir os dados.

Criando a classe Usuarios.java

```
package br.edu.utfpr.loja.model;

import java.io.Serializable;
import javax.persistence.*;

@Entity
@Table(name = "usuarios")
public class Usuarios implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue
    @Column(name = "id")
    private Integer id;

    @Column(name = "nome")
    private String nome;

    @Column(name = "email")
    private String email;

    @Column(name = "login")
    private String login;

    @Column(name = "senha")
    private String senha;

    @Column(name = "admin")
    private int admin;

    //Metodo Construtor sem parâmetros
    public Usuarios() {
    }

    //Criar getters() e setters()

    //Criar hashCode() e equals()

}
```

Adicionar o mapeamento da classe no arquivo de configuração do Hibernate (Hibernate.cfg.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>

    <!-- Após as informações de configuração -->
    <mapping class="br.edu.utfpr.loja.model.Usuarios"/>
  </session-factory>
</hibernate-configuration>
```

Criando a classe **UsuariosDao.java**:

```
package br.edu.utfpr.loja.dao;

import br.edu.utfpr.loja.database.HibernateUtil;
import br.edu.utfpr.loja.model.Usuarios;
import java.util.List;
import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

public class UsuariosDao {

    public int salvar(Usuarios usuario) {
        Session sessao = null;
        Transaction transacao = null;

        try {
            sessao = HibernateUtil.getSessionFactory().openSession();
            transacao = sessao.beginTransaction();
            sessao.save(usuario);
            transacao.commit();
        } catch (HibernateException ex) {
            System.out.println("Erro - erro ao salvar. " + ex.getMessage());
        } finally {
            try {
                sessao.close();
            } finally {
                return usuario.getId();
            }
        }
    }
}

//Fim método salvar
```

```
public void atualizar(Usuarios usuario) {
    Session sessao = null;
    Transaction transacao = null;

    try {
        sessao = HibernateUtil.getSessionFactory().openSession();
        transacao = sessao.beginTransaction();
        sessao.update(usuario);
        transacao.commit();
    } catch (HibernateException e) {
        System.out.println("Não foi possível alterar o usuário. Erro: " + e.getMessage());
    } finally {
        try {
            sessao.close();
        } catch (Throwable e) {
            System.out.println("Erro ao fechar operação de atualização. Mensagem: " + e.getMessage());
        }
    }
} //Fim método atualizar

public void excluir(Usuarios usuario) {
    Session sessao = null;
    Transaction transacao = null;

    try {
        sessao = HibernateUtil.getSessionFactory().openSession();
        transacao = sessao.beginTransaction();
        sessao.delete(usuario);
        transacao.commit();
    } catch (HibernateException e) {
        System.out.println("Não foi possível excluir o usuario. Erro: " + e.getMessage());
    } finally {
        try {
            sessao.close();
        } catch (Throwable e) {
            System.out.println("Erro ao fechar operação de exclusão. Mensagem: " + e.getMessage());
        }
    }
} //Fim do método excluir
```

```

public List<Usuarios> listar() {
    Session sessao = null;
    Transaction transacao = null;
    Query consulta = null;
    List<Usuarios> resultado = null;

    try {
        sessao = HibernateUtil.getSessionFactory().openSession();
        transacao = sessao.beginTransaction();
        consulta = sessao.createQuery("from Usuarios");
        resultado = consulta.list();
        transacao.commit();
        return resultado;
    } catch (HibernateException e) {
        System.out.println("Não foi possível selecionar usuarios. Erro: " + e.getMessage());
        throw new HibernateException(e);
    } finally {
        try {
            sessao.close();
        } catch (Throwable e) {
            System.out.println("Erro ao fechar operação de consulta. Mensagem: " + e.getMessage());
        }
    }
} //Fim do método listar

public Usuarios buscaUsuario(int valor) {
    Usuarios usuario = null;
    Session sessao = null;
    Transaction transacao = null;
    Query consulta = null;

    try {
        sessao = HibernateUtil.getSessionFactory().openSession();
        transacao = sessao.beginTransaction();
        consulta = sessao.createQuery("from Usuarios where id = :parametro");
        consulta.setInteger("parametro", valor);
        usuario = (Usuarios) consulta.uniqueResult();
        transacao.commit();
        return usuario;
    } catch (HibernateException e) {
        System.out.println("Não foi possível buscar categoria. Erro: " + e.getMessage());
    } finally {
        try {
            sessao.close();
        } catch (Throwable e) {
            System.out.println("Erro ao fechar operação de buscar. Mensagem: " + e.getMessage());
        }
    }
    return usuario;
} //Fim do método buscaUsuario
} //Fim da Classe

```

6) Utilizando Hibernate para persistir os dados.

Criando o arquivo UsuariosController.java, que será um Servlet responsável por receber as requisições das páginas e devolver os dados no formato JSON.

Dentro do arquivo UsuarioController.java alterar o urlPatterns para:

urlPatterns = {"/admin/UsuarioController"}, uma vez que as páginas para controle de usuário ficarão dentro da pasta admin da aplicação.

O método processRequest() da classe ficará com o seguinte código:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    String acao = request.getParameter("acao");
    String pagina = "";
    String msg = "";
    try {

        if (acao.equalsIgnoreCase("listar")) {
            JSONObject result = listarUsuarios();
            out.print(result);
        } else if (acao.equalsIgnoreCase("salvar")) {
            JSONObject result = salvarUsuario(request);
            out.print(result);
        } else if (acao.equalsIgnoreCase("carregar")) {
            JSONObject result = carregarUsuario(request);
            out.print(result);
        } else if (acao.equalsIgnoreCase("excluir")) {
            JSONObject result = excluirUsuario(request);
            out.print(result);
        }

    } finally {
        out.close();
    }
} // FIM processRequest
```

Logo após o método processRequest() criar os demais métodos necessários:

```
private JSONObject listarUsuarios() {
    UsuariosDao usuarioDao = new UsuariosDao();
    List<Usuarios> lista = usuarioDao.listar();
    JSONObject result = new JSONObject();
    JSONArray array = new JSONArray();
    for (Usuarios usuario : lista) {
        JSONArray ja = new JSONArray();
        ja.put(usuario.getId());
        ja.put(usuario.getNome());
        ja.put(usuario.getLogin());
        ja.put(usuario.getEmail());
        array.put(ja);
    }
    //O resultado será colocado em um array com o nome de aaData,
    //pois é o formato esperado pelo DataTable utilizado para
    //exibir os dados
    result.put("aaData", array);

    return result;
} // Fim ListarUsuarios

private JSONArray validarUsuario(HttpServletRequest request) {
    JSONArray array = new JSONArray();
    JSONArray ja;

    String nome = request.getParameter("nome");
    if (nome == null || nome.trim().isEmpty()) {
        ja = new JSONArray();
        ja.put("Informe o nome do Usuário.");
        array.put(ja);
    }

    return array;
} //Fim ValidarUsuario

private JSONObject carregarUsuario(HttpServletRequest request) {
    int id = 0;
    if (!request.getParameter("usuarioid").isEmpty()) {
        id = Integer.parseInt(request.getParameter("usuarioid"));
    }
    UsuariosDao usuarioDao = new UsuariosDao();
    Usuarios usuario = usuarioDao.buscaUsuario(id);
    JSONObject result = new JSONObject();
    result.put("usuarioid", id);
    result.put("nome", usuario.getNome());
    result.put("email", usuario.getEmail());
    result.put("login", usuario.getLogin());

    return result;
} //Fim CarregarUsuario
```

```

private JSONObject salvarUsuario(HttpServletRequest request) {
    int id = 0;
    if (!request.getParameter("usuarioid").isEmpty()) {
        id = Integer.parseInt(request.getParameter("usuarioid"));
    }

    JSONObject result = new JSONObject();
    JSONArray erros = validarUsuario(request);
    if (erros.length() == 0) {
        Usuarios usuario = new Usuarios();
        usuario.setId(id);
        usuario.setNome(request.getParameter("nome"));
        usuario.setLogin(request.getParameter("login"));
        usuario.setSenha(Criptografia.encripta(request.getParameter("senha")));
        usuario.setEmail(request.getParameter("email"));
        usuario.setAdmin(0);
        UsuariosDao dao = new UsuariosDao();
        if (id == 0) {
            usuario.setId(dao.salvar(usuario));
        } else {
            dao.atualizar(usuario);
        }
        result.put("situacao", "OK");
        result.put("id", usuario.getId());
        result.put("mensagem", "Os dados foram gravados com sucesso!");

    } else {
        result.put("situacao", "ERRO");
        result.put("mensagem", erros);
    }
    return result;
}

//fim do método salvarUsuário

private JSONObject excluirUsuario(HttpServletRequest request) {
    int id = 0;
    if (!request.getParameter("usuarioid").isEmpty()) {
        id = Integer.parseInt(request.getParameter("usuarioid"));
    }
    UsuariosDao produtoDao = new UsuariosDao();
    Usuarios usuario = produtoDao.buscaUsuario(id);
    produtoDao.excluir(usuario);
    JSONObject result = new JSONObject();
    result.put("situacao", "OK");
    result.put("mensagem", "Registro removido com sucesso!");

    return result;
}

//Fim do método excluirUsuario

```


Tela de Cadastro de usuários (usuarioscadastro.jsp), deve ficar dentro do diretório admin:

Em destaque estão os links aos arquivos CSS.

A chamada ao método javascript: "loadData()" que é responsável por carregar os dados quando um usuário é editado.

E a inclusão do arquivo "menu.jsp", que contém o menu padrão da aplicação.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" href="../img/favicon.ico">
    <title>Cadastro de Usuários</title>
    <link href="../css/bootstrap.css" rel="stylesheet">
    <link href="../css/geral.css" rel="stylesheet">
  </head>
  <body onload="loadData()">
    <%@include file="menu.jsp" %>
```

Após esses dados temos o formulário com os campos para preenchimento dos dados.

```
<div class="container">
  <div class="row">
    <div class="col-md-1 col-md-offset-3">
      <fieldset>
        <form id="frm" name="frm" class="well span6" role="form" method="POST"
          action="UsuarioController?acao=salvar" >
          <legend>Cadastro de Usuários</legend>

          <div class="form-group">
            <label for="id">Código</label>
            <input type="text" class="form-control" readonly id="usuarioid"
name="usuarioid"
              value="{param.id}" />
          </div>
          ... Continua
```

Abaixo estão os links com os arquivos javascript utilizados (jQuery e Bootstrap). E script executado ao clicar no botão submit do formulário.

```
<script type="text/javascript" src="../js/jquery-2.0.3.js"></script>
<script type="text/javascript" src="../js/bootstrap.js"></script>
```

```
<script type="text/javascript">
var form = $('#frm');
//No submit do form salva os dados
form.submit(function() {
    //Os dados do formulário fazem comunicação através de Ajax
    //Com o servlet de usuários
    $.ajax({
        type: form.attr('method'),
        url: form.attr('action'),
        data: form.serialize(),
        //Caso a comunicação ocorra com sucesso, a variável data
        //possui o retorno do servlet
        success: function(data) {

            //a variável retorno vai conter as propriedades vindas
            //do UsuarioController na ação Salvar.
            var retorno = $.parseJSON(data);
            $("#divErro").hide("fast");
            $("#divSucesso").hide("fast");
            if (retorno.situacao === "OK") {
                $("#usuarioid").attr('value', retorno.id);
                $("#divSucesso").html(retorno.mensagem);
                $("#divSucesso").show("slow");
            } else {
                var msg = "";
                $.each(retorno.mensagem, function() {
                    $.each(this, function(index, value) {
                        msg += value + '<br />';
                    });
                });
                $("#divErro").html(msg);
                $("#divErro").show("slow");
            }
        }
    });
    return false;
}); //FIM SALVAR
```

```
// .. continua
```

Abaixo o script para carregar os dados do usuário para edição.

```
//Carrega dados do usuário
function loadData() {
    //verifica se foi passado um id
    var id = $("#usuarioid").val();
    if (id > 0) {
        //caso tenha sido passado um id, é chamado o Servlet e passado a ação carregar
        juntamente com o id
        var destino = 'UsuarioController?acao=carregar&usuarioid=' + id;
        $.ajax({
            type: 'POST',
            url: destino,
            success: function(data) {
                //no caso de sucesso os dados são transformados em JSON
                var json = $.parseJSON(data);
                //então são preenchidos os inputs com os dados vindos do servlet
                $("#usuarioid").val(json.usuarioid);
                $("#nome").val(json.nome);
                $("#email").val(json.email);
                $("#login").val(json.login);
            }
        });
    }
}
//fim loadData

</script>
</body>
</html>
```

Tela para listagem dos usuários (usuarios.jsp), deve ficar dentro do diretório admin:

Link aos arquivos css utilizados na página, aqui foi adicionado o arquivo **dataTables.bootstrap.css**, que vai fazer a formatação da tabela que vai exibir os dados.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" href="../img/favicon.png">
    <title>Lista de Usuários</title>
    <link href="../css/bootstrap.css" rel="stylesheet">
    <link href="../css/dataTables.bootstrap.css" rel="stylesheet">
    <link href="../css/geral.css" rel="stylesheet">
  </head>
```

Após temos o início do documento, com a inclusão do menu padrão, e os botões de ação para o cadastro, edição e exclusão de um usuário. E na sequência temos dois elementos div, para exibição das mensagens de erro e sucesso.

```
<body>
  <%@include file="menu.jsp" %>
  <div class="container">
    <div class="starter-template">
      <h1>Cadastro de Usuários</h1>
    </div>
    <!-- Botões de ação -->
    <a class="btn btn-primary" id="novo" href="usuarioscadastro.jsp"
role="button">Novo</a>
    <a class="btn btn-success" id="editar" role="button">Editar</a>
    <a class="btn btn-danger" id="excluir" role="button">Excluir</a>

    <!-- Mensagens de Erro -->
    <div class="alert alert-success" id="div_OK" style="display: none">
    </div>
    <div class="alert alert-danger" id="div_ERRO" style="display: none">
    </div>

    <div class="spacer"><br /></div>
```

Após é criada a tabela para exibição dos dados, que é baseada no componente dataTables, sua documentação e exemplos podem ser acessados em: <http://datatables.net/>

```
<!-- TABELA -->
<table class="table table-striped table-bordered display"
  id="exemplo" width="100%">
  <thead>
    <tr>
      <th width="20%">Código</th>
      <th width="40%">Nome</th>
      <th width="20%">Email</th>
      <th width="20%">Login</th>
    </tr>
  </thead>

  <tbody>
  </tbody>

  <tfoot>
    <tr>
      <th width="20%">Código</th>
      <th width="40%">Nome</th>
      <th width="20%">Email</th>
      <th width="20%">Login</th>
    </tr>
  </tfoot>
</table>
<div class="spacer"></div>
</div><!-- /.container -->
```

Então são importados os arquivos javascript necessários:

```
<script type="text/javascript" src="../js/jquery-2.0.3.js"></script>
<script type="text/javascript" src="../js/jquery.dataTables.js"></script>
<script type="text/javascript" charset="utf-8" src="../js/bootstrap.js" ></script>
<script type="text/javascript" charset="utf-8" src="../js/dataTables.bootstrap.js" ></script>
<script type="text/javascript" charset="utf-8" src="../js/bootbox.js" ></script>
```

Nessa página são adicionados dois arquivos javascript adicionais: o dataTables.bootstrap.js, responsável pela formatação e funcionalidades da tabela. E o bootbox.js responsável pelas mensagens de confirmação ao clicar no botão excluir.

Abaixo está o script para carregar a tabela e executar os botões: editar e excluir.

```
<script type="text/javascript" charset="UTF-8" id="init-code">
    var oTable;

    $(document).ready(function(){
        //Popula a tabela dos os dados do servlet
        oTable = $('#exemplo').dataTable({
            "bProcessing": true,
            "sAjaxSource": 'UsuarioController?acao=listar'
        });

        /* Código para selecionar as linhas da tabela, ao clicar
        * em uma linha
        */
        $("#exemplo tbody").click(function(event) {
            $(oTable.fnSettings().aoData).each(function() {
                $(this.nTr).removeClass('row_selected');
            });
            $(event.target.parentNode).addClass('row_selected');
        });

        //Seta a linguagem da message box para br;
        bootbox.setDefaults({
            locale: "br"
        });

        //Evento ao clicar no botão editar
        $('#editar').click(function() {
            var anSelected = fnGetRSelected(oTable);
            if (anSelected.length !== 0) {
                var colunas = $(anSelected).eq(0).find('td');
                var id = colunas.eq(0).text();
                var destino = 'usuarioscadastro.jsp?id=' + id;
                $(window.document.location).attr('href', destino);
            } else {
                exibeMensagem('ERRO','Selecione um registro para editar.');
```

```
/* Evento ao clicar no botão excluir */
```

```
$('#excluir').click(function() {  
    var anSelected = fnGetRSelected(oTable);  
    if (anSelected.length !== 0) {  
        bootbox.confirm('Deseja excluir o registro?', function(result) {  
            if (result) {  
                var colunas = $(anSelected).eq(0).find('td');  
                var id = colunas.eq(0).text();  
                var destino = 'UsuarioController?acao=excluir&usuarioid=' + id;  
                $.ajax({  
                    type: 'POST',  
                    url: destino,  
                    success: function(data) {  
                        var json = $.parseJSON(data);  
                        oTable.fnDeleteRow(anSelected[0]);  
                        if (json.situacao === 'OK') {  
                            exibeMensagem('ERRO', 'Registro removido com sucesso.');                        } else {  
                            exibeMensagem('ERRO', 'Falha ao remover o registro.');                        }  
                    }  
                });  
            }  
        });  
    } else {  
        exibeMensagem('ERRO', 'Selecione um registro para remover.');    }  
}); //FIM EXCLUIR
```

```
}); //READY
```

```
//Exibe as mensagens na página
```

```
function exibeMensagem(situacao, mensagem){  
    if (situacao === 'OK'){  
        $('#div_OK').html(mensagem);  
        $('#div_OK').fadeIn('fast');  
        $('#div_OK').fadeOut(2000);  
    } else {  
        $('#div_ERRO').html(mensagem);  
        $('#div_ERRO').fadeIn('fast');  
        $('#div_ERRO').fadeOut(2000);  
    }  
}
```

```
/* Retorna as linhas selecionadas para editar ou excluir */
```

```
function fnGetRSelected(oTableLocal) {  
    return oTableLocal.$('tr.row_selected');
```

```
</script>  
</body>  
</html>
```