

OFICINA WEB

Uso de Relatórios e Hibernate com SPRING

Sobre o Hibernate

Hibernate é o framework para persistência de dados mais utilizado em projetos Java. Sendo uma das primeiras opções a implementar o conceito de mapeamento objeto-relacional (ORM), em pouco tempo se tornou referência entre os desenvolvedores.

Sobre o Spring

Spring é um framework que inicialmente não foi criado para o desenvolvimento web. Na essência o Spring é um container leve que visa fornecer serviços para sua aplicação como por exemplo o gerenciamento de objetos ou transação. Mas com o tempo a comunidade Spring entendeu que o Struts era ultrapassado e começou criar um framework MVC próprio. O Spring MVC é um framework moderno que usa os recursos atuais da linguagem além de usar todo poder do container Spring.

Hibernate com SPRING

O Spring é um framework muito usado em projetos Java, fornecendo de maneira simples o processo de injeção de dependências e inversão de controle. Além disso, ele disponibiliza várias implementações de apoio distribuídas entre diversas APIs.

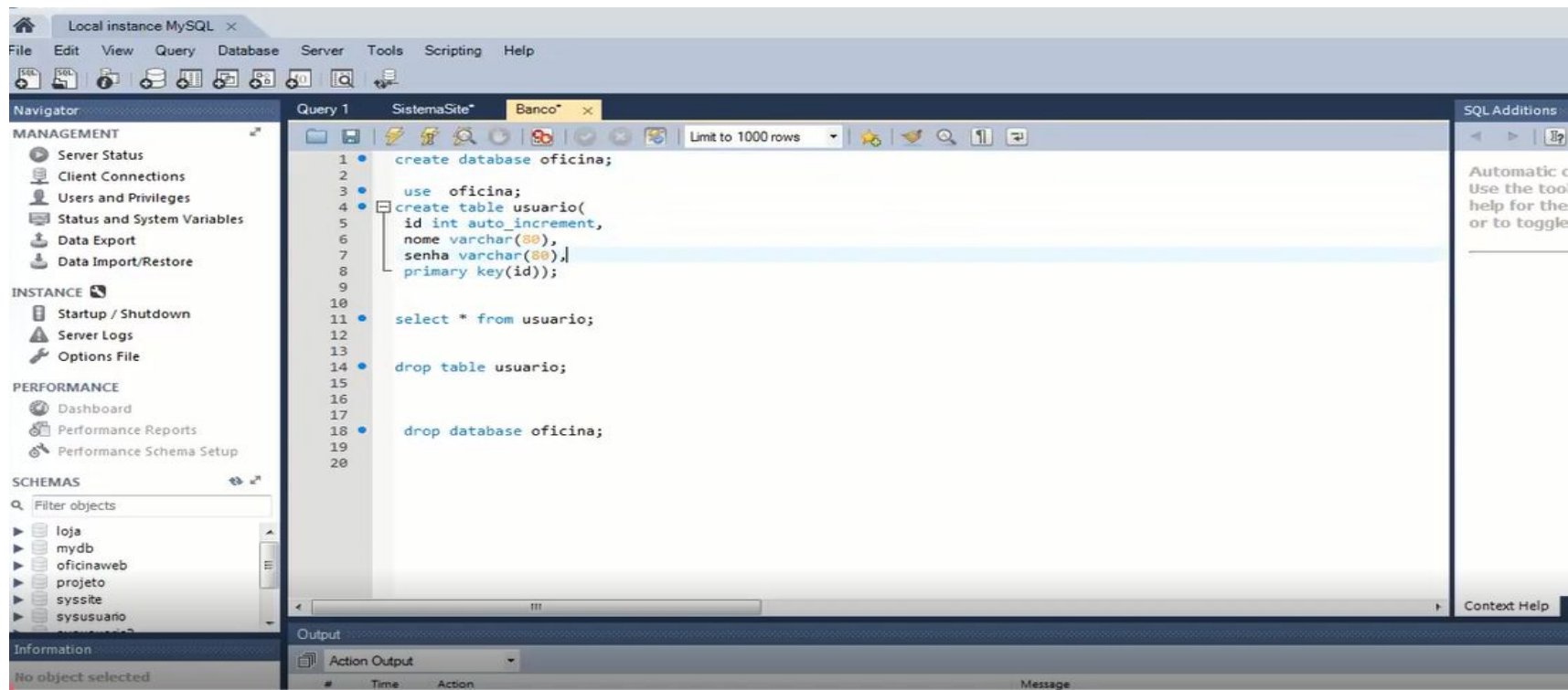
O Hibernate Framework, por outro lado, é uma poderosa ferramenta de persistência que possibilita ao desenvolvedor trabalhar com objetos e deixar de lado o uso da linguagem SQL para consultas e armazenamento de dados.

Hibernate com SPRING

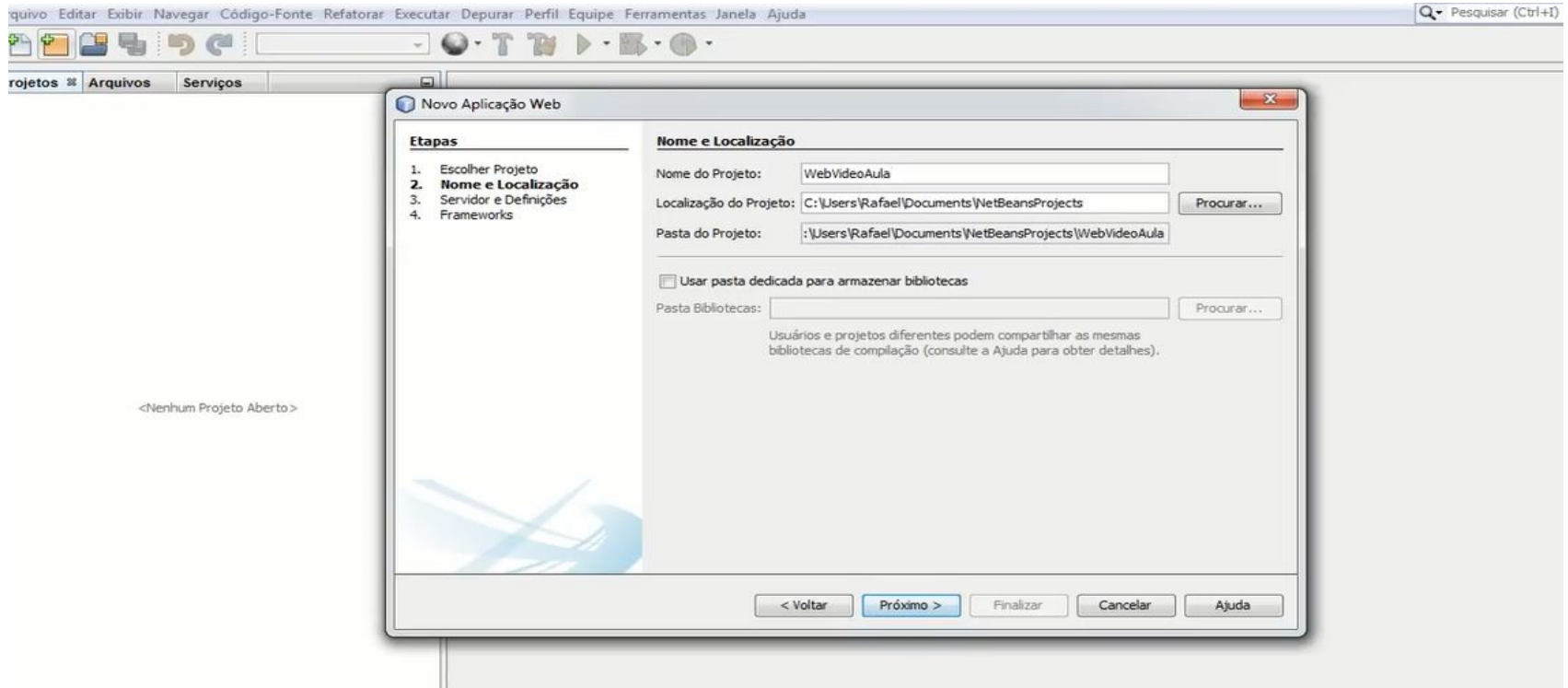
Um dos fatores positivos do Spring Framework é a facilidade de integração entre ele e outros frameworks Java, como o próprio Hibernate. Esta é uma parceria de muito sucesso em projetos Java, sendo eles de pequeno, médio ou grande porte.

Criando uma Aplicação Web usando Spring com Hibernate

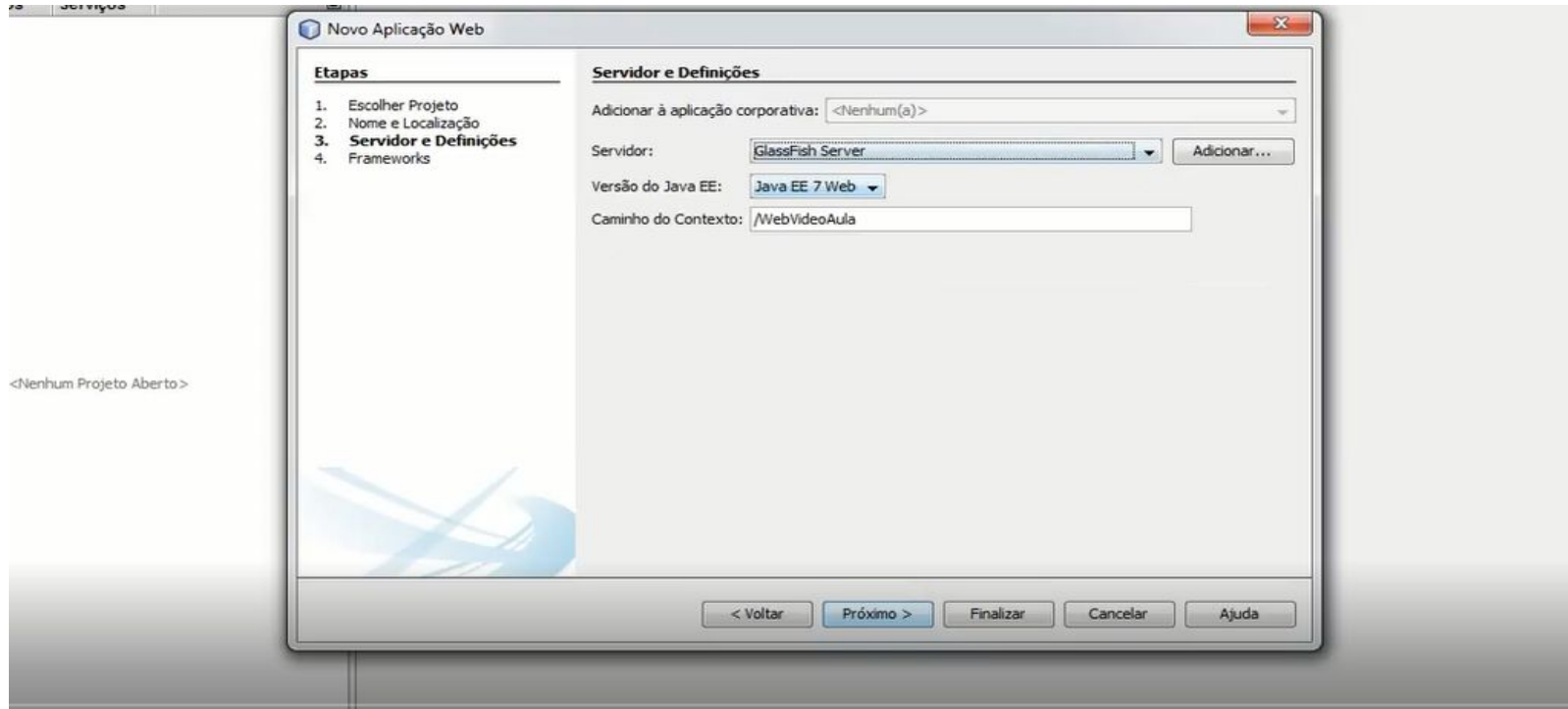
Primeiramente vamos criar o banco



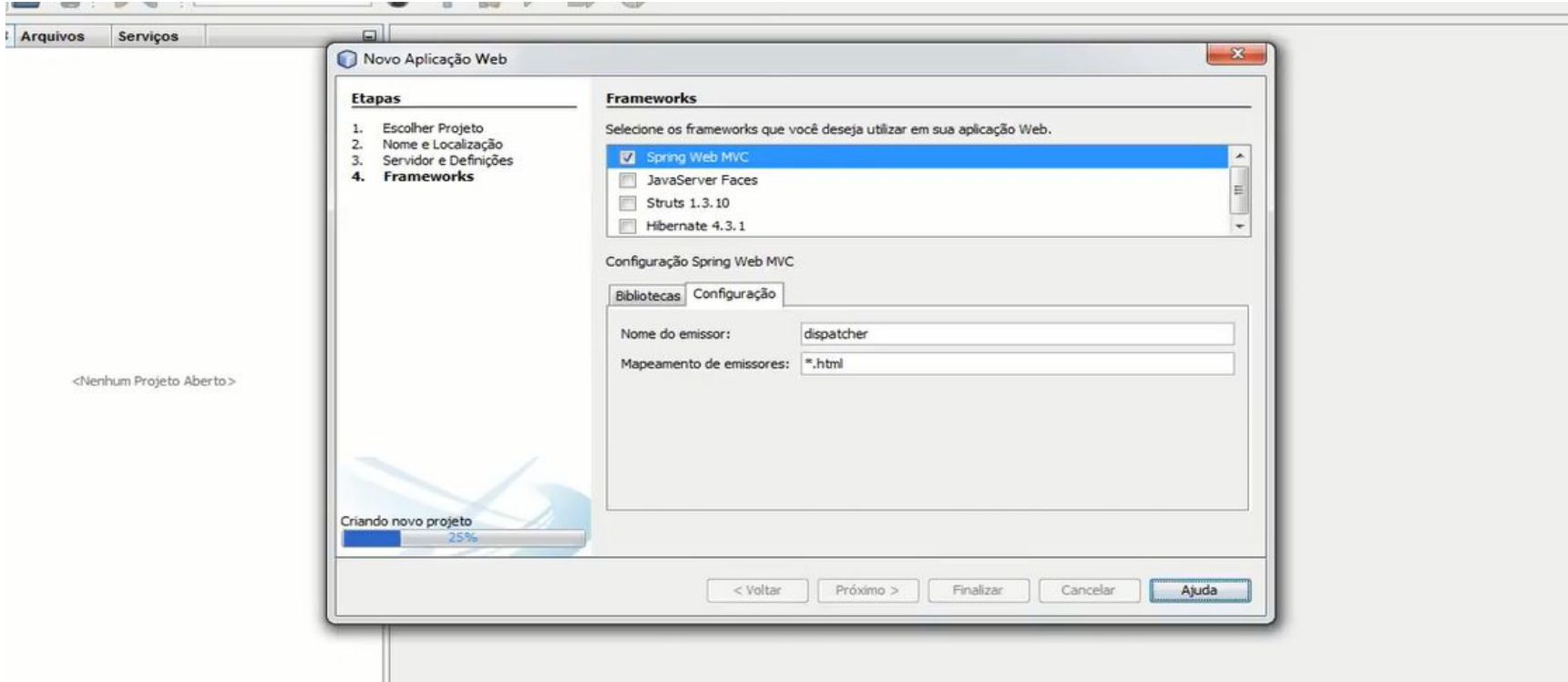
Agora podemos dar início a criação do projeto, será criado uma aplicação web com nome “WebVideoAula”



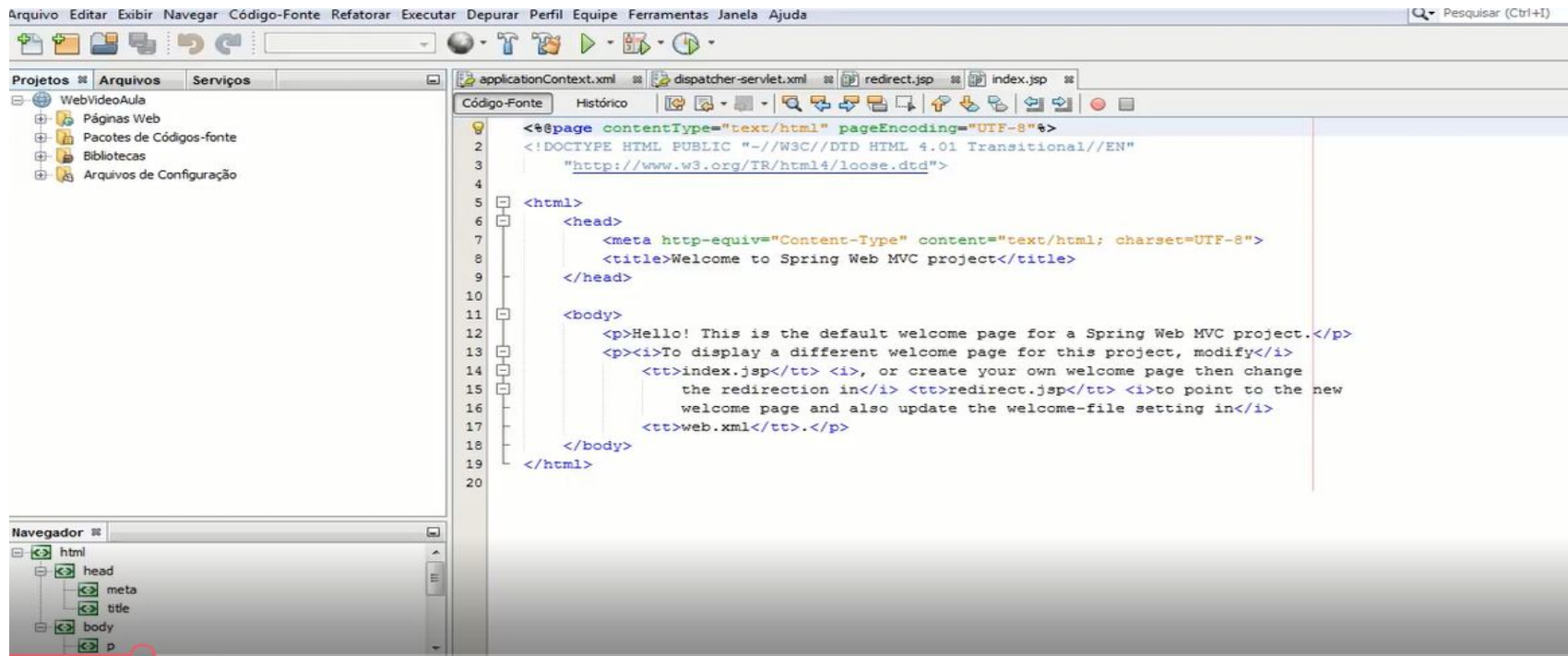
Em seguida essas configurações são mantidas



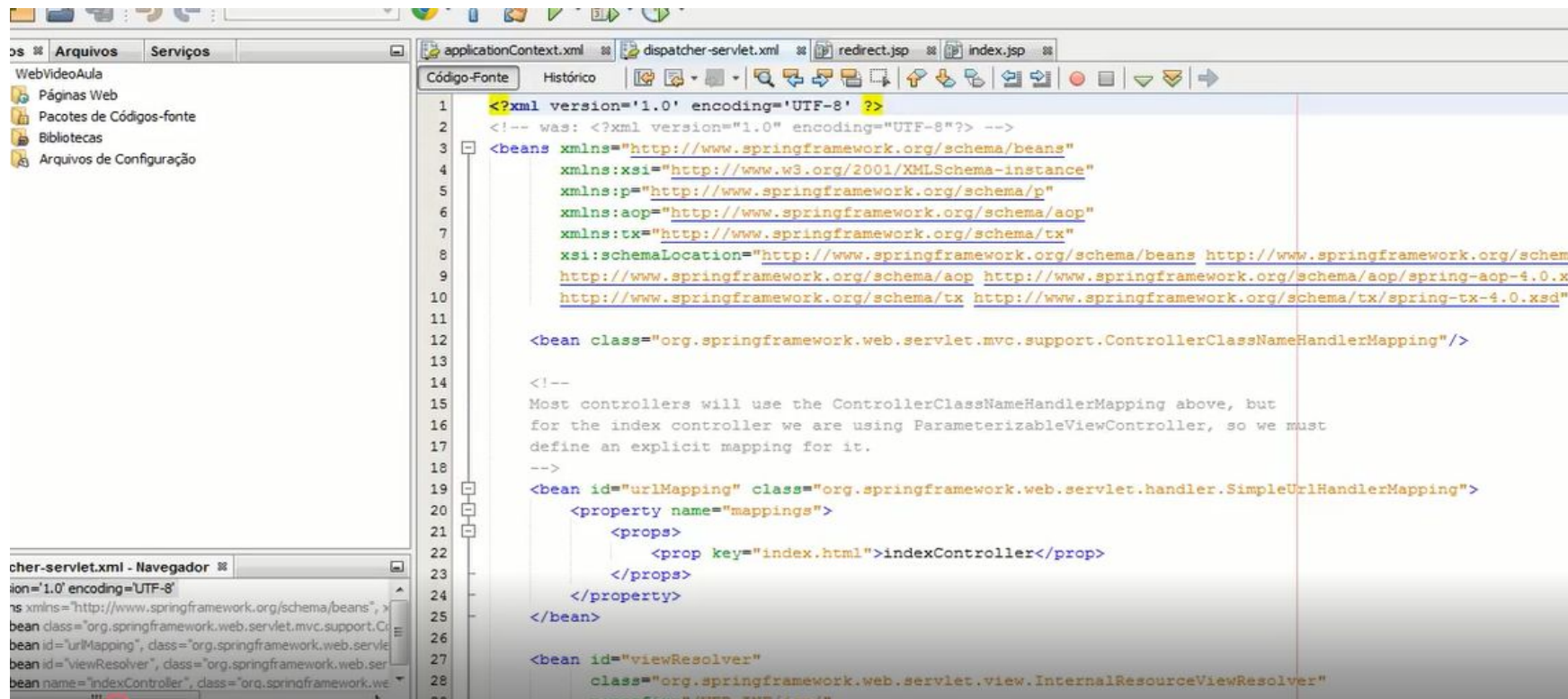
É escolhido o Framework Spring, e no campo Mapeamento de emissores é acrescentado o “|”



Após a criação do projeto vamos fazer uma adaptação utilizando o Spring com Hibernate



No dispatcher-servlet.xml, será alterado este código pelo código que será mostrado a seguir

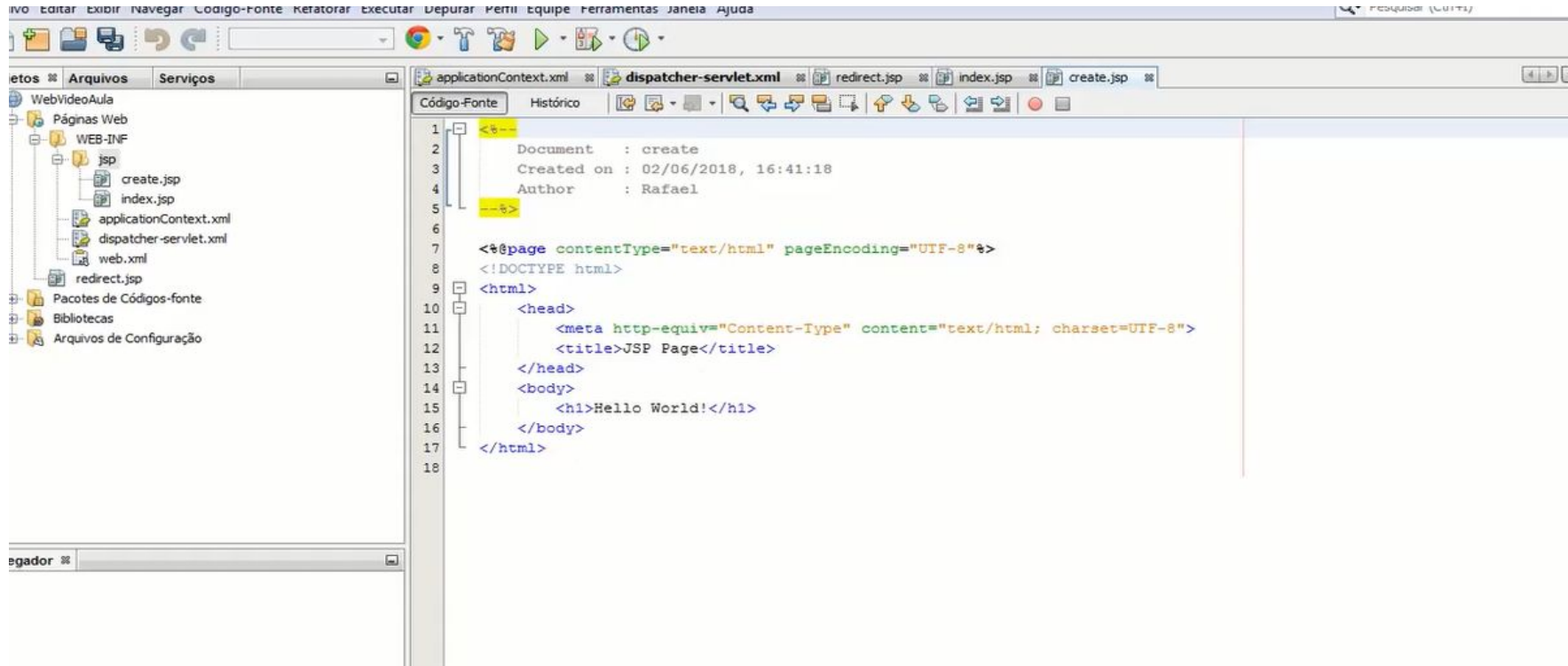


```
1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!-- was: <?xml version="1.0" encoding="UTF-8"? -->
3 <beans xmlns="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:aop="http://www.springframework.org/schema/aop"
7       xmlns:tx="http://www.springframework.org/schema/tx"
8       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schem
9       http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.x
10      http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd"
11
12   <bean class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"/>
13
14   <!--
15   Most controllers will use the ControllerClassNameHandlerMapping above, but
16   for the index controller we are using ParameterizableViewController, so we must
17   define an explicit mapping for it.
18   -->
19   <bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
20     <property name="mappings">
21       <props>
22         <prop key="index.html">indexController</prop>
23       </props>
24     </property>
25   </bean>
26
27   <bean id="viewResolver"
28       class="org.springframework.web.servlet.view.InternalResourceViewResolver"
```

Este é o Dispatcher-servlet recomendável para evitar problema de implantação em nossa aplicação

```
1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!-- was: <?xml version="1.0" encoding="UTF-8"? -->
3 <beans xmlns="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:context="http://www.springframework.org/schema/context"
7       xmlns:mvc="http://www.springframework.org/schema/mvc"
8       xmlns:tx="http://www.springframework.org/schema/tx"
9       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
10      http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
11      http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
12      http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">
13
14     <bean class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"/>
15     <context:component-scan base-package="controller" />
16     <mvc:annotation-driven />
17     <bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"/>
18     <bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"/>
19     <mvc:view-controller path="/create.html" view-name="create"/>
20     <!--bean class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"/-->
21 <!--
22 Most controllers will use the ControllerClassNameHandlerMapping above, but
23 for the index controller we are using ParameterizableViewController, so we must
24 define an explicit mapping for it.
25 -->
26 <bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
27     <property name="mappings">
28         <props>
29             <prop key="index.html">indexController</prop>
30         </props>
31     </property>
32 </bean>
33 <bean id="viewResolver"
34       class="org.springframework.web.servlet.view.InternalResourceViewResolver"
35       p:prefix="/WEB-INF/jsp/"
36       p:suffix=".jsp" />
37 <!--
38 The index controller.
39 -->
40 <bean name="indexController"
41       class="org.springframework.web.servlet.mvc.ParameterizableViewController"
42       p:viewName="index" />
43 </beans>
```

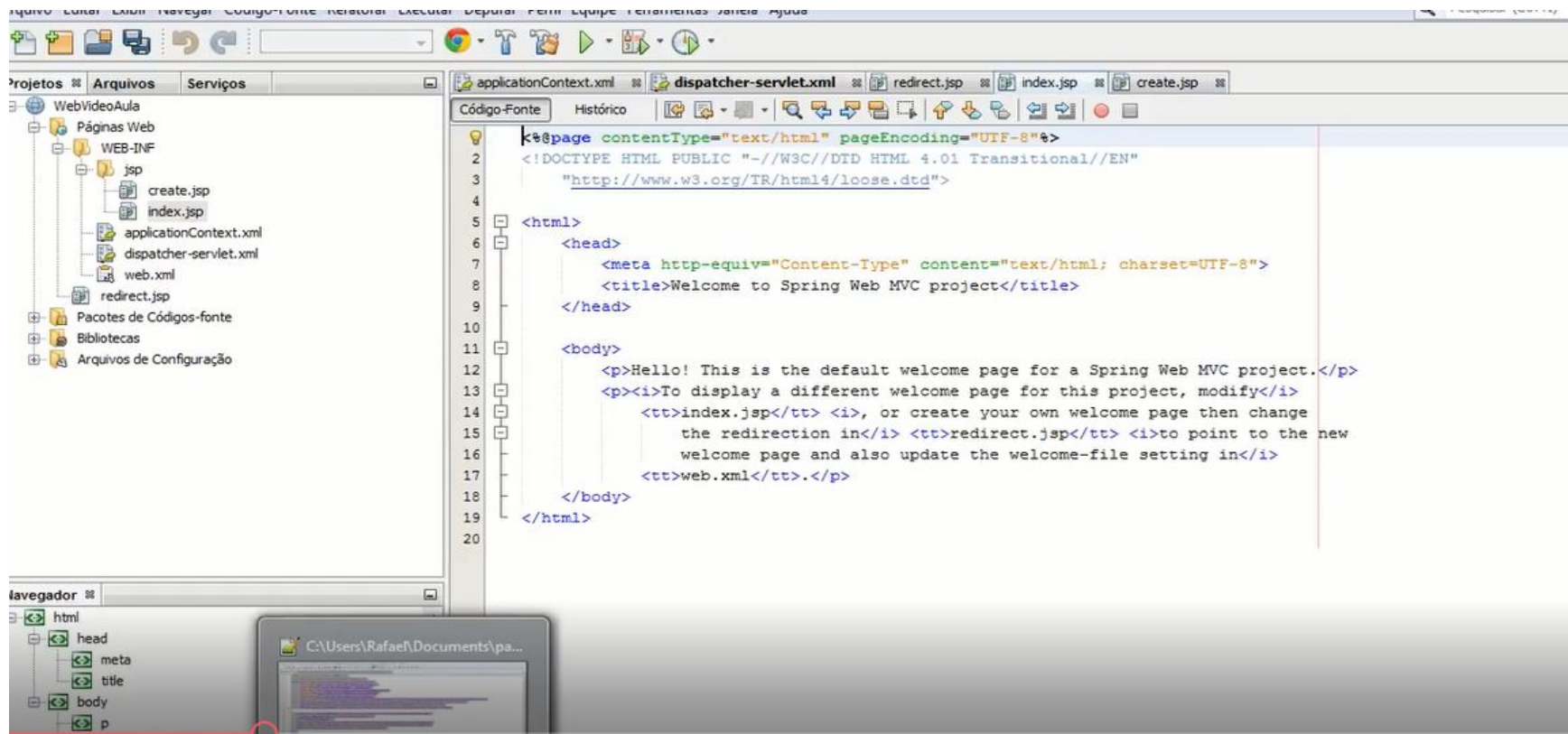
Agora vamos criar uma página jsp 'create', que serve para armazenar dados ao banco



Será colocado no 'create' este código para adicionar os usuários ao banco

```
1 |<%--
2 |   Document    : create
3 |   Created on  : 31/05/2018, 00:29:28
4 |   Author     : Rafael
5 | --%>
6 |
7 |<%@page contentType="text/html" pageEncoding="UTF-8"%>
8 |<%@ taglib prefix="f" uri="http://www.springframework.org/tags/form" %>
9 |<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
10 |<!DOCTYPE html>
11 |<html>
12 |   <head>
13 |     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14 |     <title>Adicionar Usuario</title>
15 |   </head>
16 |   <body>
17 |     <h1>Formulario de Adicao de Usuarios</h1>
18 |     <f:form action="usuario/tudo.html" modelAttribute="usuario">
19 |       Nome: <input type="text" name="nome" /></br>
20 |       <br><br>
21 |       Senha: <input type="password" name="senha"/></br>
22 |       <br><br>
23 |       <input type="submit" value="Adicionar"/></br>
24 |     </f:form>
25 |     <br><br>
26 |     <a href="${pageContext.request.contextPath}/index.html">
27 |       Voltar
28 |     </a>
29 |
30 |   </body>
31 | </html>
32 |
33 |
```

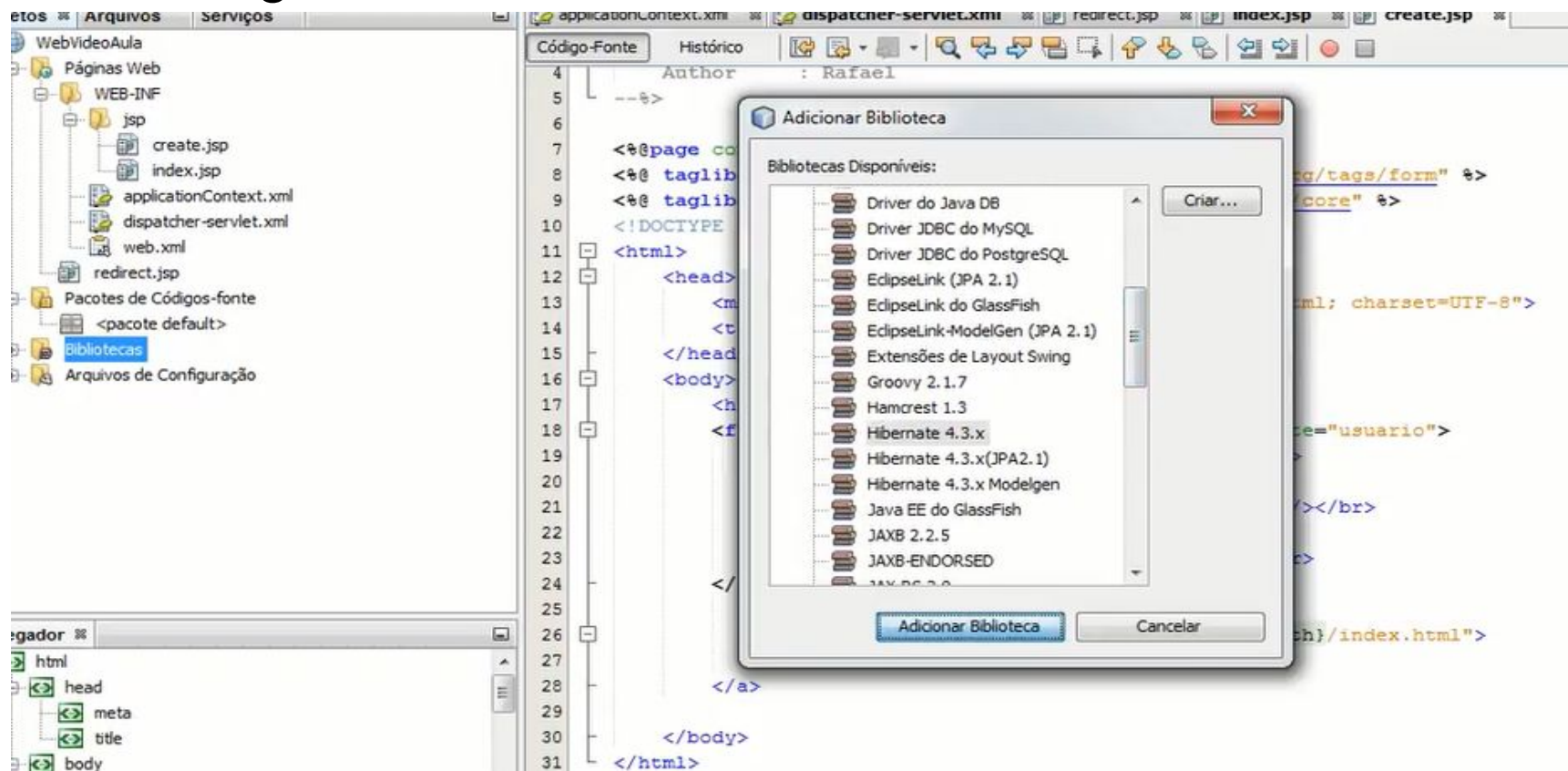
Esta é pagina do 'index', onde vai está a listagem e os links



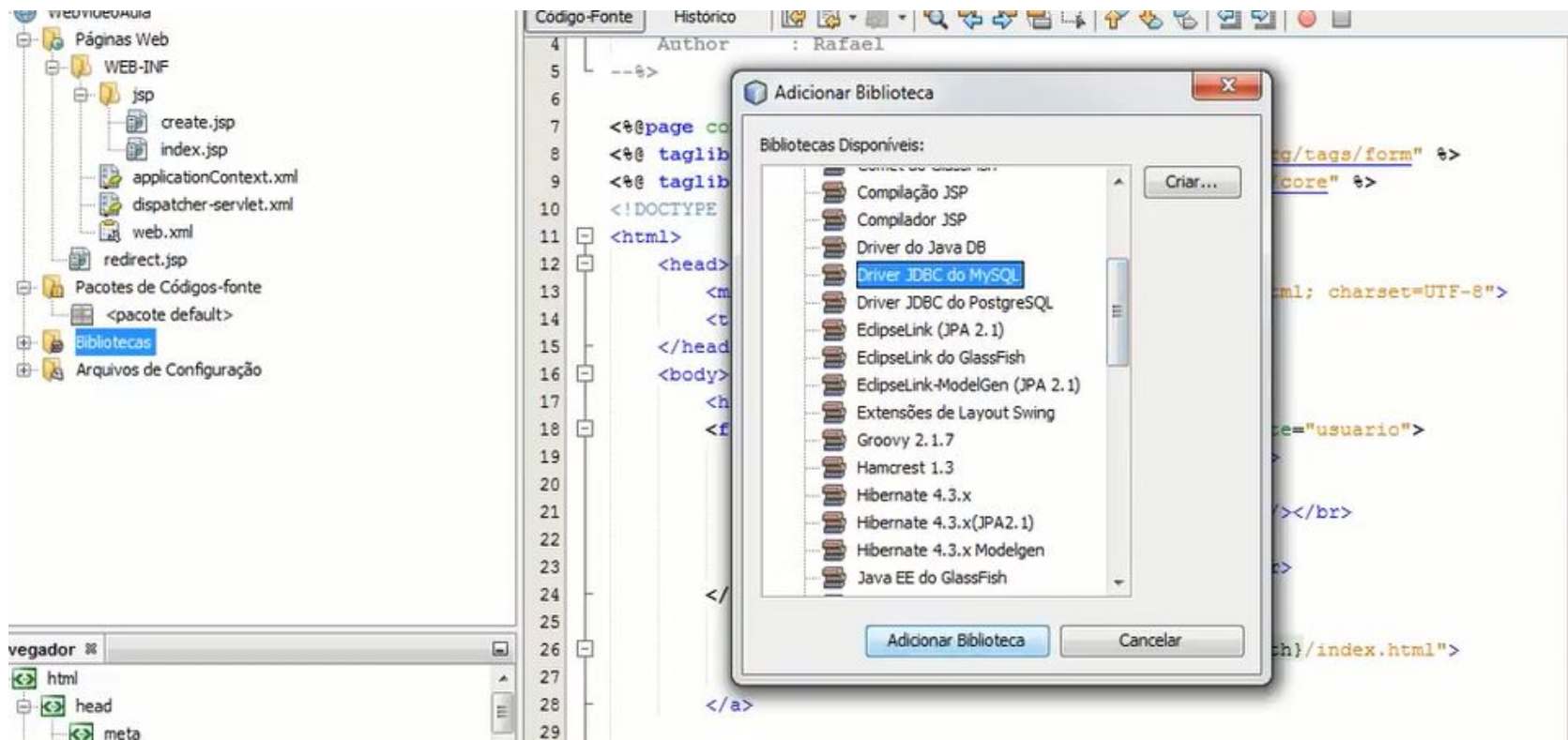
Será colocado no 'index' este código para listagem dos usuários que estão armazenados no banco de dados

```
1 |<%@page contentType="text/html" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <%@ taglib prefix="f" uri="http://www.springframework.org/tags/form" %>
4
5
6
7 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
8 "http://www.w3.org/TR/html4/loose.dtd">
9
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13 <title>Oficina Spring com Hibernate</title>
14 </head>
15
16 <body>
17 <center><h1> Bemvindo a Oficina Spring com Hibernate </h1></center>
18 <a href="usuario/tudo.html">Atualizar Lista</a><hr>
19 <table style="border-collapse: collapse" cellpadding="7px" border="1">
20 <tr>
21 <th>Id</th>
22 <th>Nome</th>
23 <th>Senha</th>
24 </tr>
25 <c:forEach items="${lst}" var="em">
26 <tr>
27 <td>${em.id}</td>
28 <td>${em.nome}</td>
29 <td>${em.senha}</td>
30 </tr>
31 </c:forEach>
32 </table>
33 <a href="create.html">Adicionar Usuario</a> <br><br>
34 <a href="${pageContext.request.contextPath}/index.html">
35 OK
36 </a>
37 </body>
38 </html>
39
```

Vamos agora adicionar a biblioteca do Hibernate



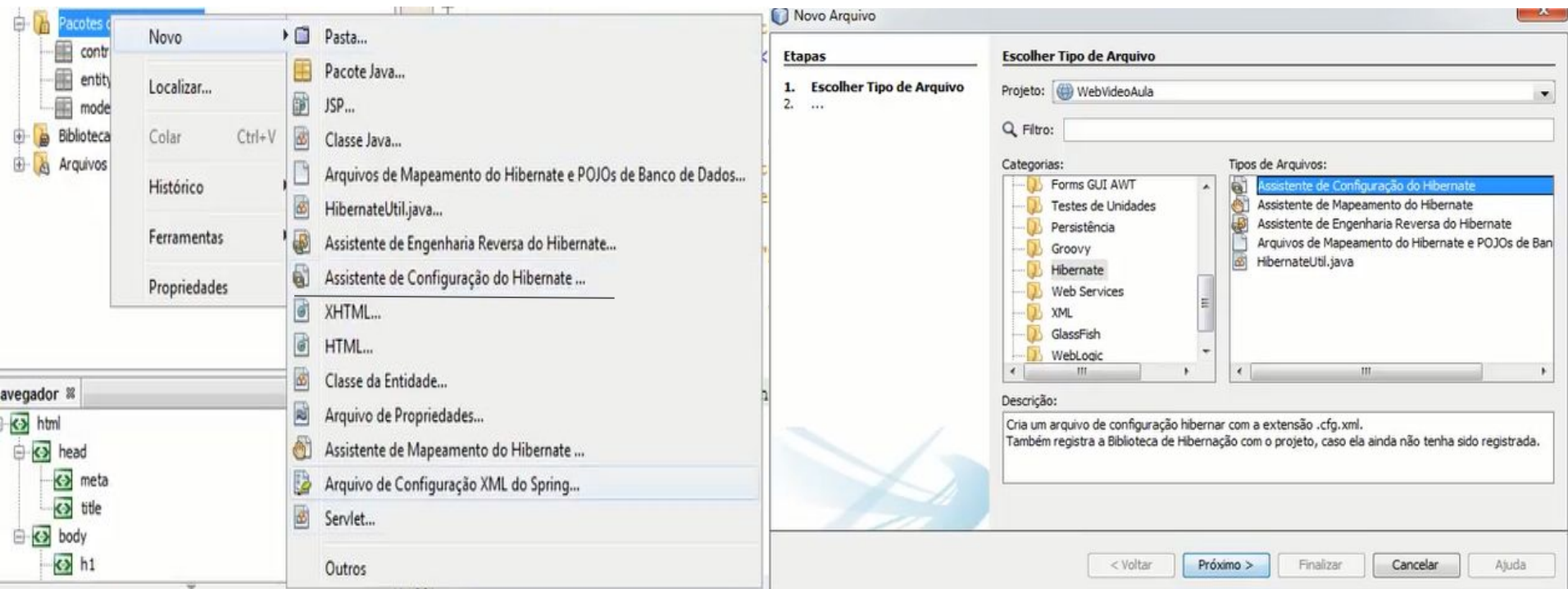
Vamos adicionar também a biblioteca do Driver JDBC



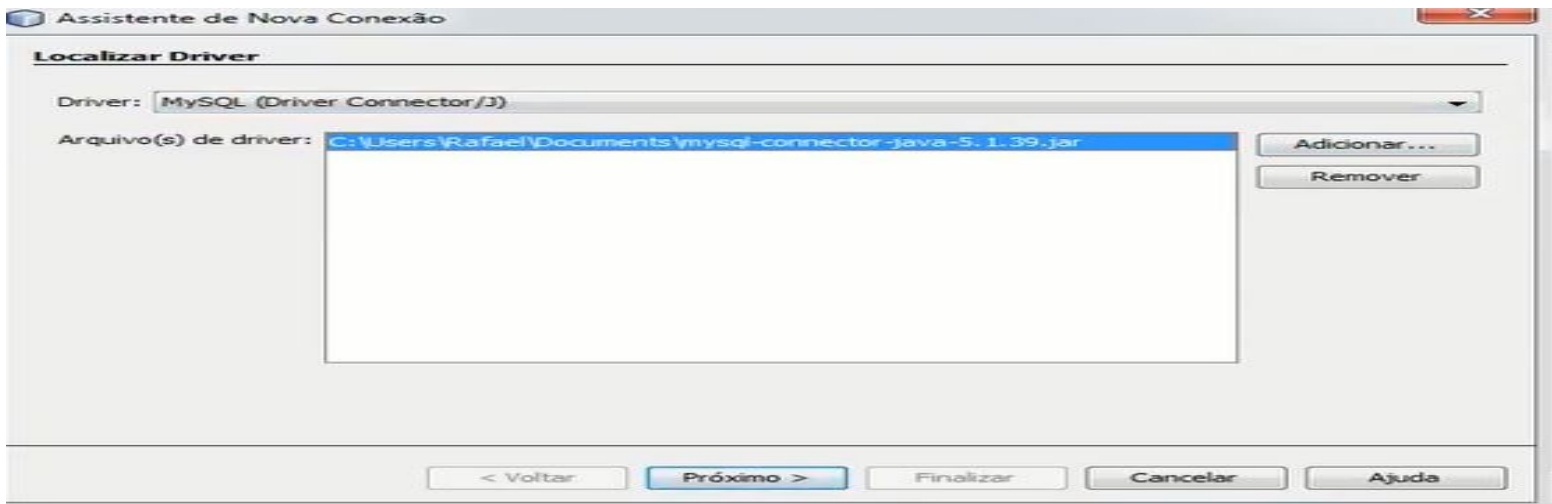
Em seguida serão criados os pacotes: controller, entity e model



Criando um assistente de configuração do hibernate



Fazendo a conexão com o banco de dados



Agora vamos inserir o nome do banco de dados

Assistente de Nova Conexão

Personalizar Conexão

Nome do Driver: MySQL (Driver Connector/J)

Host: localhost Porta: 3306

Banco de dados: oficina

Nome do Usuário: root

Senha: ●●●●

☒ Lembrar senha

Propriedades da Conexão Testar Conexão

JDBC URL: jdbc:mysql://localhost:3306/oficina?zeroDateTimeBehavior=convertToNull

< Voltar Próximo > Finalizar Cancelar Ajuda

E podemos finalizar a conexão

Assistente de Nova Conexão

Escolher nome para conexão

Substituir o nome default da conexão. O nome deverá ser descritivo sobre a conexão que você está criando.

Entrar nome da conexão:

`jdbc:mysql://localhost:3306/oficina?zeroDateTimeBehavior=convertToNull`

< Voltar Próximo > **Finalizar** Cancelar Ajuda

Na página do 'hibernate' em Propriedades diversas vamos alterar o campo valor da propriedade, inserindo thread

The screenshot shows an IDE with the following components:

- Project Explorer (Left):** Displays the project structure for 'WebVideoAula'. It includes 'Páginas Web' (Web Pages) with files like 'create.jsp', 'index.jsp', 'applicationContext.xml', 'dispatcher-servlet.xml', 'web.xml', and 'redirect.jsp'. It also shows 'Pacotes de Códigos-fonte' (Source Code Packages) with a default package containing 'hibernate.cfg.xml', 'controller', 'entity', and 'model'. There are also 'Bibliotecas' (Libraries) and 'Arquivos de Configuração' (Configuration Files).
- Editor (Right):** Shows the 'Propriedades Diversas' (Various Properties) tab for the 'hibernate.cfg.xml' file. It contains a table of properties:

| Nome | Valor |
|-----------------------------------|---|
| hibernate.connection.driver_class | com.mysql.jdbc.Driver |
| hibernate.connection.url | jdbc:mysql://localhost:3306/oficina?zeroDateTimeBehavior... |
| hibernate.connection.username | root |
| hibernate.connection.password | root |

Below the table are buttons: 'Adicionar...' (Add), 'Editar...' (Edit), and 'Remover' (Remove).

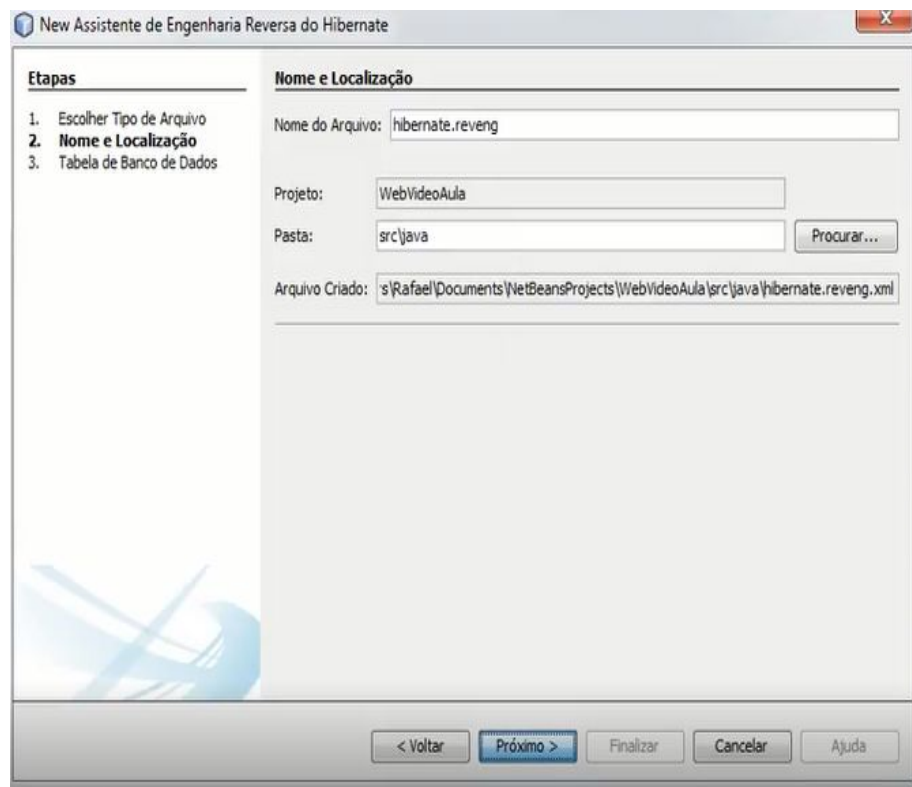
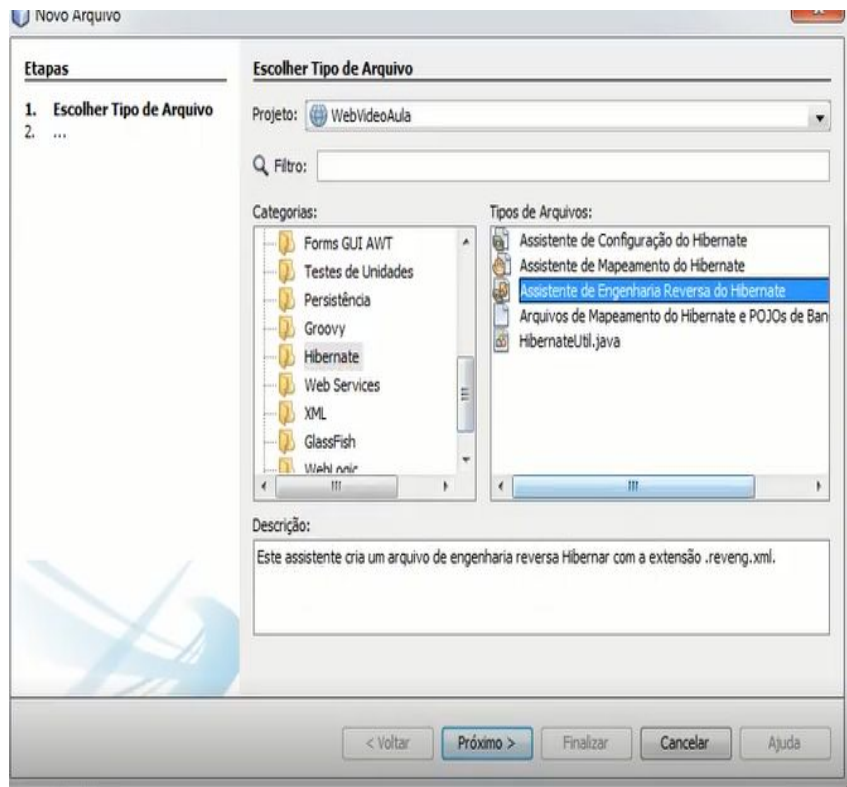
A dialog box titled 'Adicionar Propriedade do Hibernate' (Add Hibernate Property) is open in the foreground. It contains:

- Nome da Propriedade:** hibernate.current_session_context_class
- Valor da Propriedade:** thread
- Buttons: 'OK' and 'Cancelar' (Cancel).

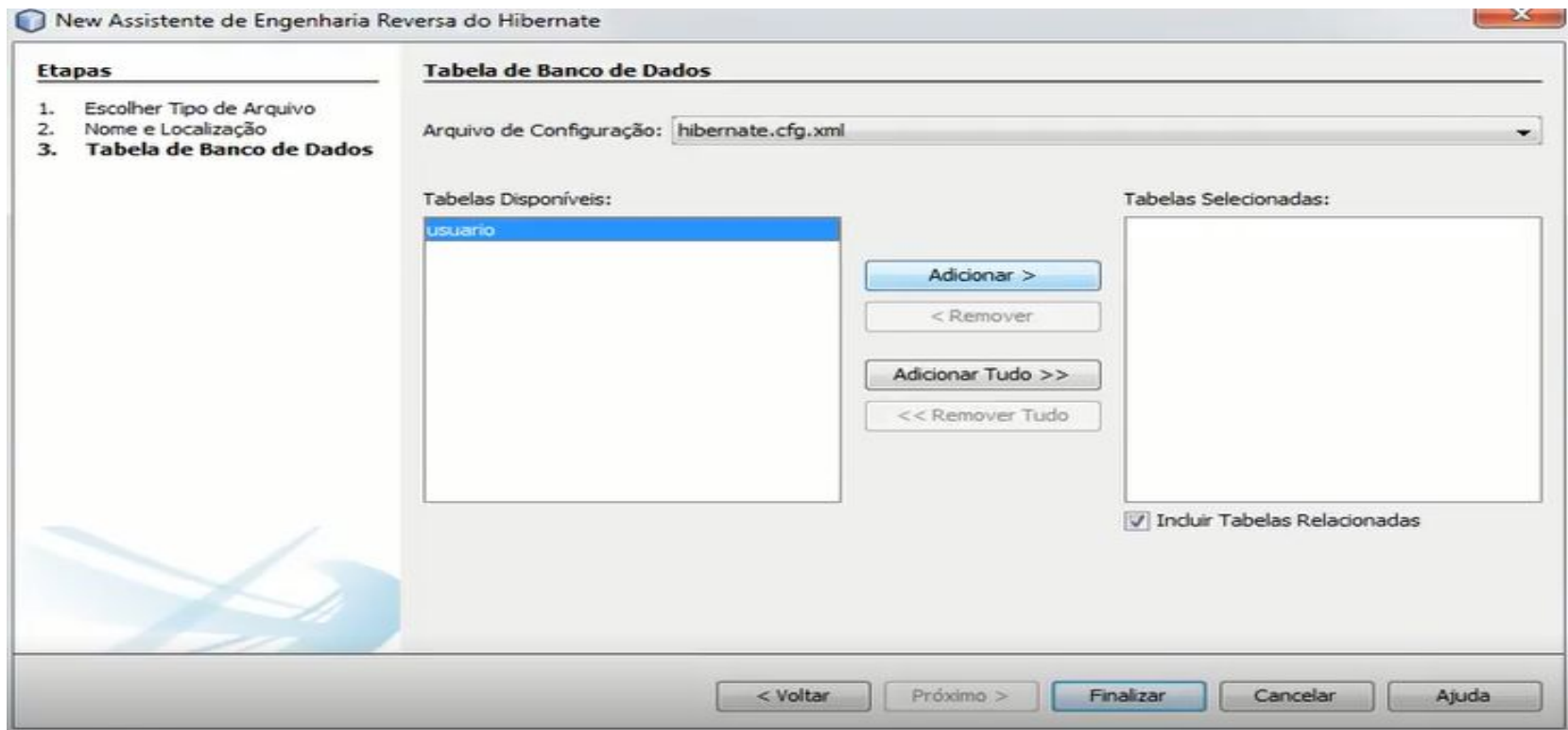
At the bottom of the IDE, a 'hibernate.cfg.xml - Navegador' (hibernate.cfg.xml - Navigator) pane shows the XML structure:

```
<?xml version="1.0" encoding="UTF-8" ...>
<PUBLIC "-//Hibernate/Hibernate Configuration DTD 3...
<hibernate-configuration>
  <session-factory>
```

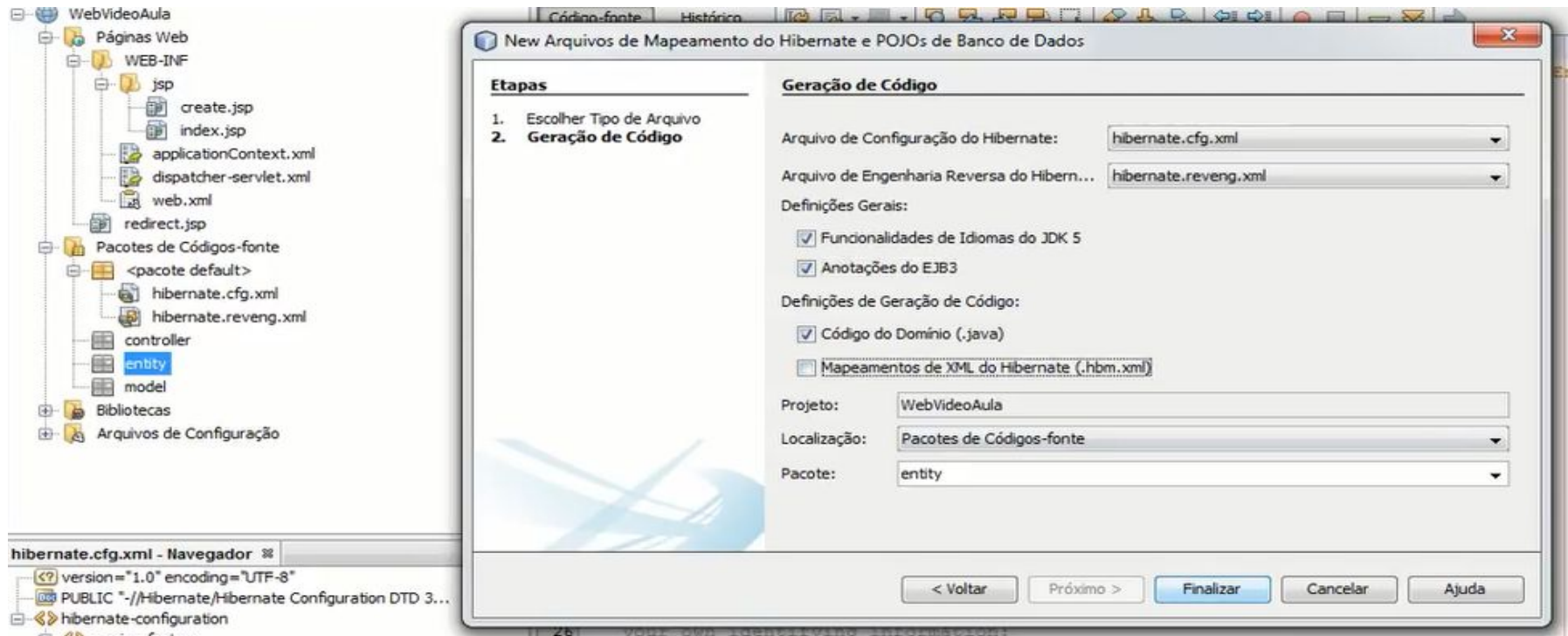
Criando o Assistente de Engenharia Reversa do Hibernate



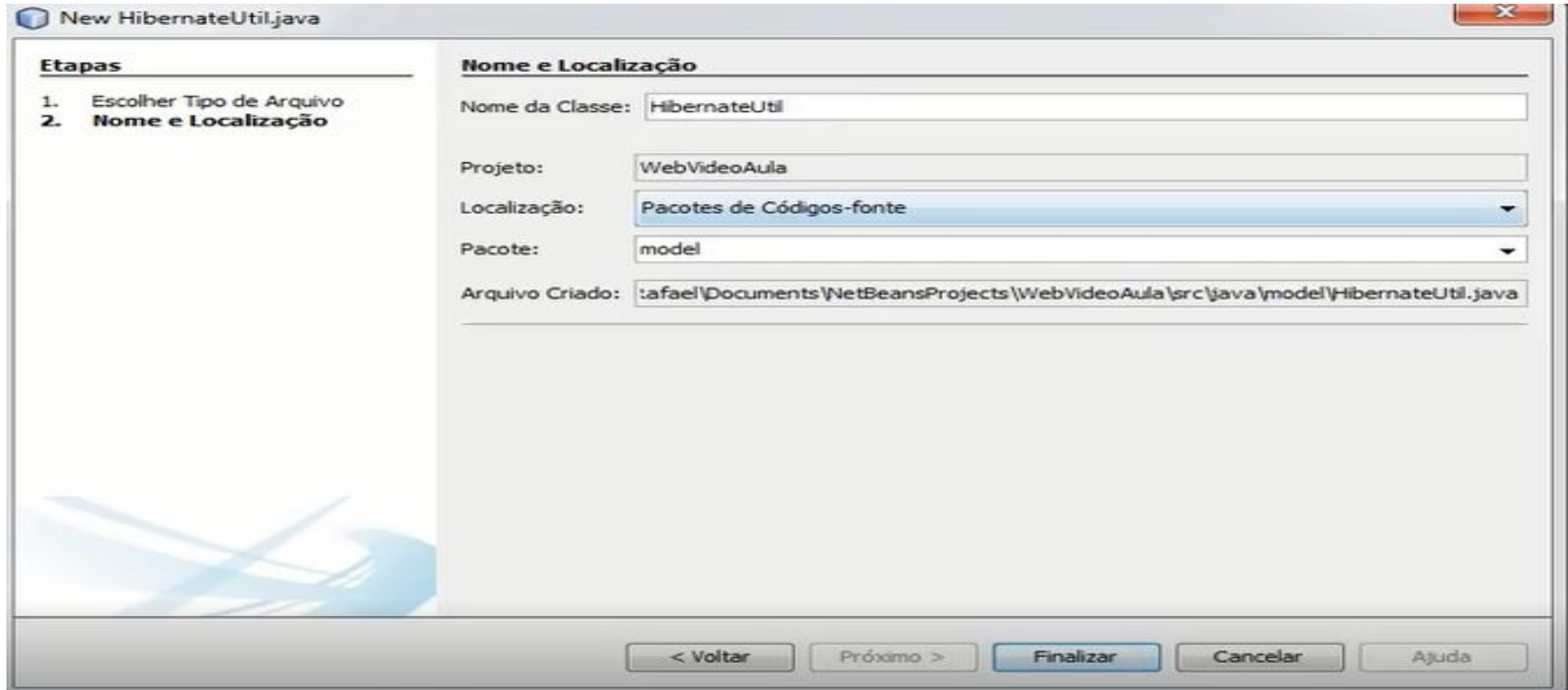
Adicionar as tabelas disponíveis e Finalizar



No pacote entity vamos adicionar o Arquivo de mapeamento do Hibernate, marcar as duas primeiras opções e desmarcar a última e Finalizar



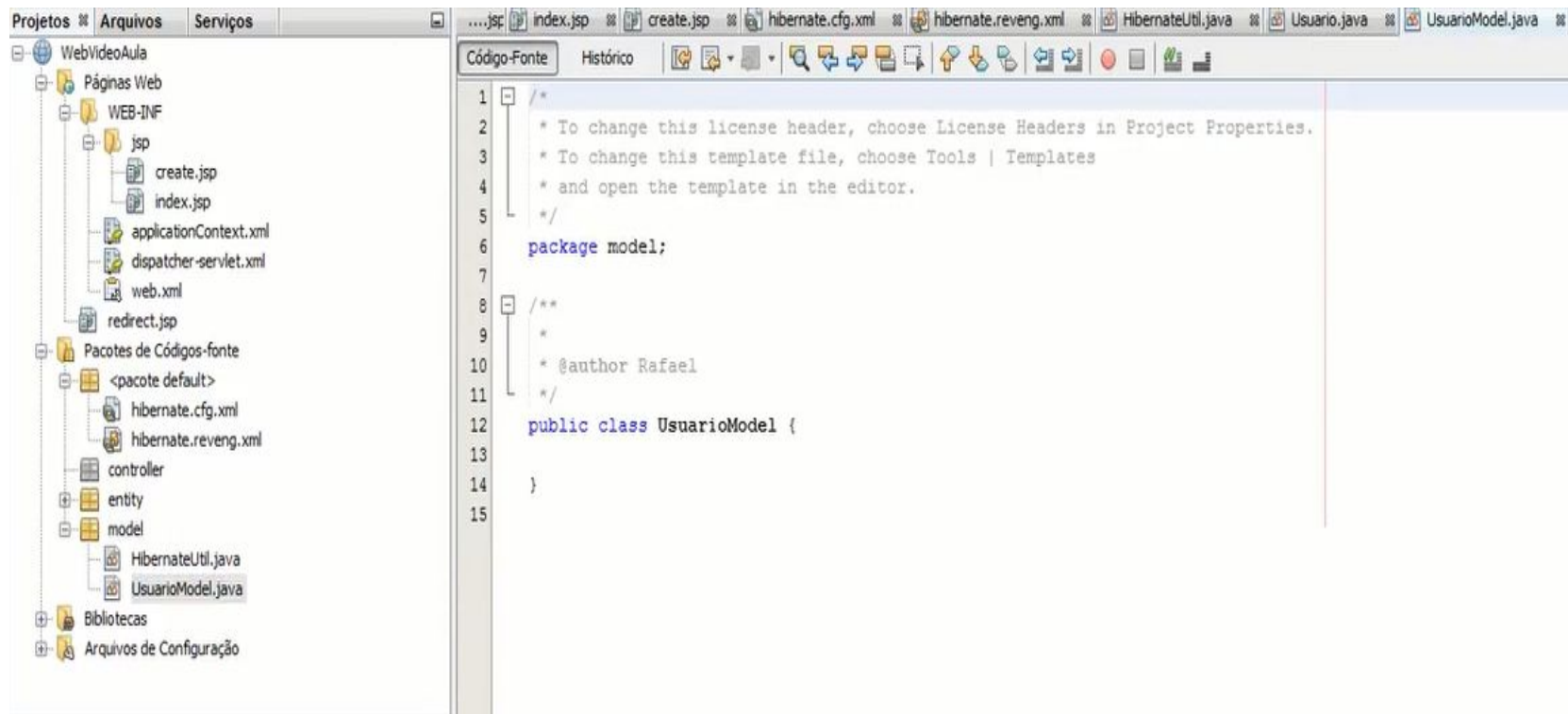
Vamos criar o 'HibernateUtil' que também será gerado automático pelo Netbeans



Em seguida vamos inserir este código no 'hibernateUtil', o que permitirá os usuários serem inseridos no banco

```
1  ▾ /*
2    * To change this license header, choose License Headers in Project Properties.
3    * To change this template file, choose Tools | Templates
4    * and open the template in the editor.
5    */
6    package model;
7
8  ▾ import org.hibernate.cfg.AnnotationConfiguration;
9    import org.hibernate.SessionFactory;
10   import org.hibernate.cfg.Configuration;
11
12  ▾ /**
13   * Hibernate Utility class with a convenient method to get Session Factory
14   * object.
15   *
16   * @author Rafael
17   */
18  ▾ public class HibernateUtil {
19      private static final SessionFactory sessionFactory = buildSessionFactory();
20
21      private static SessionFactory buildSessionFactory() {
22          ▾ try {
23              // Create the SessionFactory from hibernate.cfg.xml
24              return new Configuration().configure().buildSessionFactory();
25          ▾ } catch (Throwable ex) {
26              // Make sure you log the exception, as it might be swallowed
27              System.err.println("Initial SessionFactory creation failed." + ex);
28              throw new ExceptionInInitializerError(ex);
29          }
30      }
31
32  ▾ public static SessionFactory getSessionFactory() {
33      return sessionFactory;
34  }
35 }
36
37
```

Agora vamos criar o 'UsuarioModel'



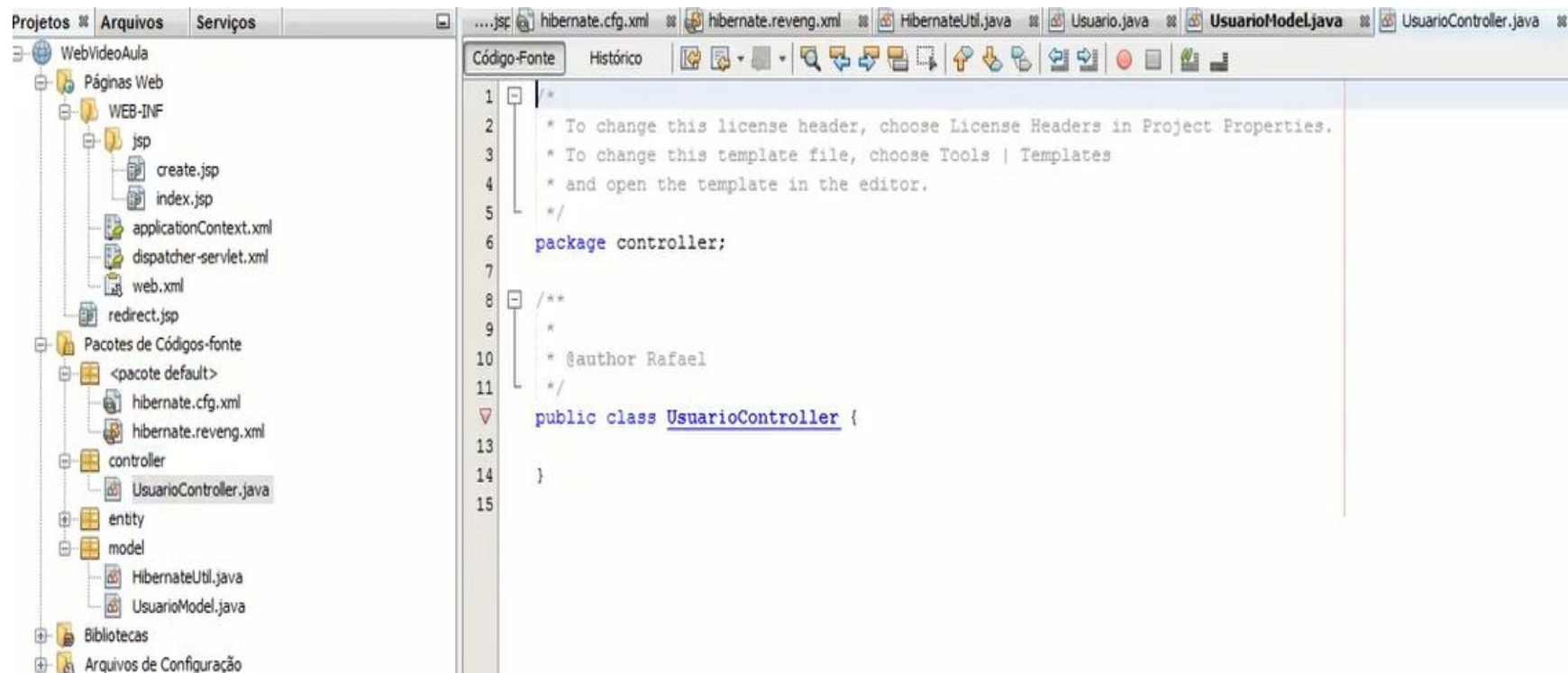
Este código será adicionado na classe 'UsuarioModel' para que seja permitido listar e inserir os usuários ao banco

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package model;
7  import entity.*;
8  import java.io.Serializable;
9  import java.util.*;
10 import javax.faces.bean.ManagedBean;
11 import javax.faces.bean.SessionScoped;
12 import org.hibernate.*;
13 import relatorio.Relatorio;
14 /**
15  *
16  * @author Rafael
17  */
18
19 @ManagedBean
20 @SessionScoped
21 public class UsuarioModel{
22
23     public List<Usuario> VerTudo(){
24         Session s = HibernateUtil.getSessionFactory().getCurrentSession();
25         List<Usuario> lst = new ArrayList<Usuario>();
26         try{
27             s.beginTransaction();
28             lst = s.createCriteria(Usuario.class).list();
29             s.getTransaction().commit();
30
31         }catch (Exception e){
32
33             e.printStackTrace();
34         }
35         return lst;
36     }
37 }
38
```


código cont.

```
39
40 ▼ public void create(Usuario user) {
41
42     Transaction trns = null;
43     Session session = HibernateUtil.getSessionFactory().openSession();
44 ▼     try {
45         trns = session.beginTransaction();
46         session.save(user);
47         session.getTransaction().commit();
48 ▼     } catch (RuntimeException e) {
49 ▼         if (trns != null) {
50             trns.rollback();
51         }
52         e.printStackTrace();
53 ▼     } finally {
54         session.flush();
55         session.close();
56     }
57 }
58
59
60
61
62
63
64 }]
```

Agora vamos criar o 'UsuarioController'



Este código será adicionado na classe 'UserController', para que seja realizado o mapeamento, que serve para adicionar novos usuários, atualização e listagem

```
7 package controller;
8 import entity.Usuario;
9 import model.*;
10 import java.util.*;
11 import org.springframework.stereotype.Controller;
12 import org.springframework.ui.Model;
13 import org.springframework.web.bind.annotation.ModelAttribute;
14 import org.springframework.web.bind.annotation.RequestMapping;
15 import org.springframework.web.bind.annotation.RequestMethod;
16
17 /**
18  *
19  * @author Rafael
20  */
21
22 @Controller
23 @RequestMapping(value="/usuario")
24 public class UsuarioController {
25
26     @RequestMapping(value = "/tudo", method = RequestMethod.GET)
27     public String VerTudo(Model m){
28         UsuarioModel model = new UsuarioModel();
29         m.addAttribute("lst", model.VerTudo());
30
31         return "index";
32     }
33
34 }
35
36 /**
37  *
38  * @param a
39  * @return
40  */
41
42 @RequestMapping(value = "/tudo", method = RequestMethod.POST )
43 public String adiciona(@ModelAttribute(value = "/usuario") Usuario a){
44     UsuarioModel model = new UsuarioModel();
45     model.create(a);
46
47     return "create";
48 }
49 }
```

Agora vamos mostrar parte da aplicação executada

Bemvindo a Oficina Spring com Hibernate

[Atualizar Lista](#)

| Id | Nome | Senha |
|----|------|-------|
|----|------|-------|

[Adicionar Usuario](#)

[OK](#)

Bemvindo a Oficina Spring com Hibernate

[Atualizar Lista](#)

| Id | Nome | Senha |
|----|----------|----------|
| 1 | Patricia | aluna123 |
| 2 | Rafael | aluno123 |

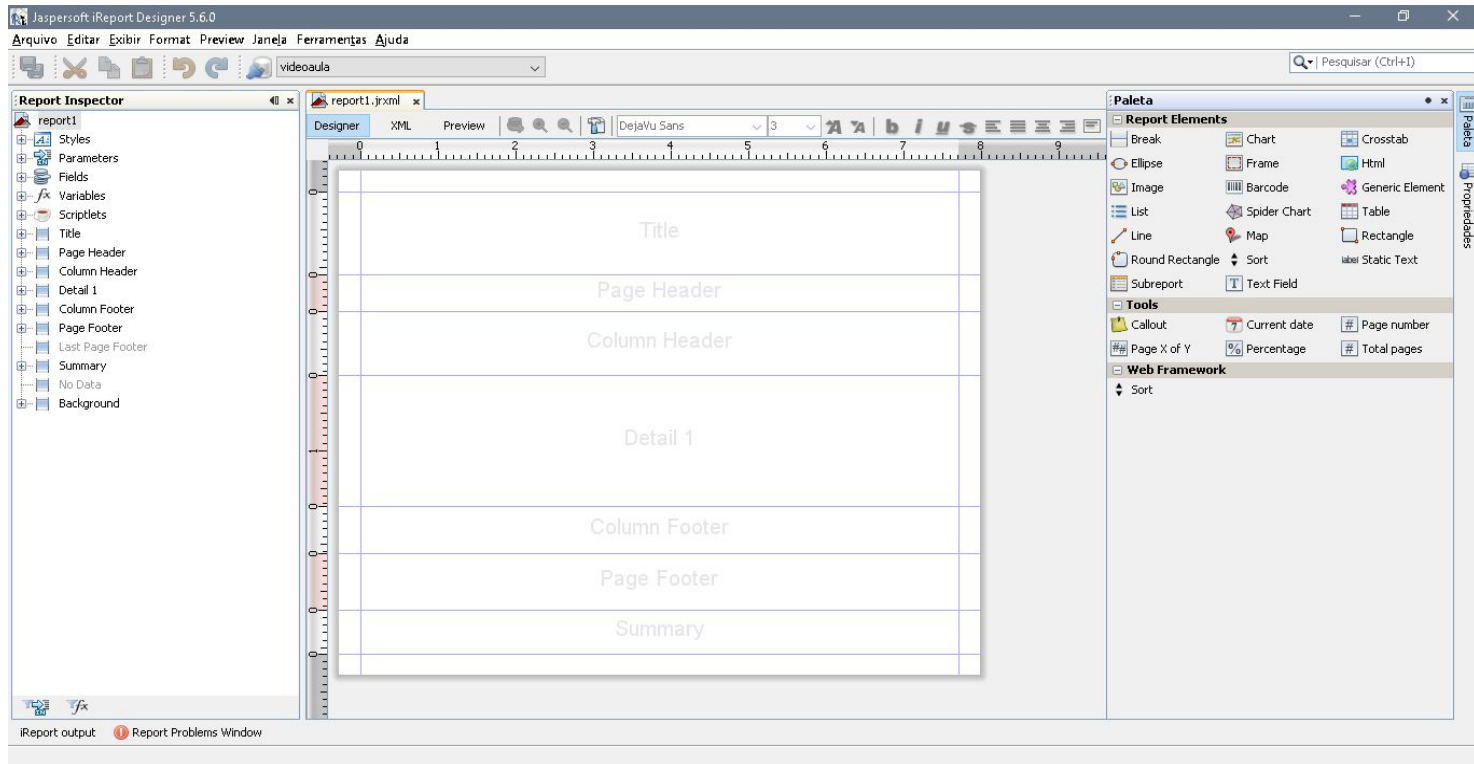
[Adicionar Usuario](#)

[OK](#)

Geração de Relatórios

- JasperReports (iReport)
- É um framework open source, gratuito e mais usado para geração de relatórios, capaz de criar os mais complexos relatórios para aplicações Java;
- É um framework multiplataforma;
- Utiliza uma interface gráfica e intuitiva;

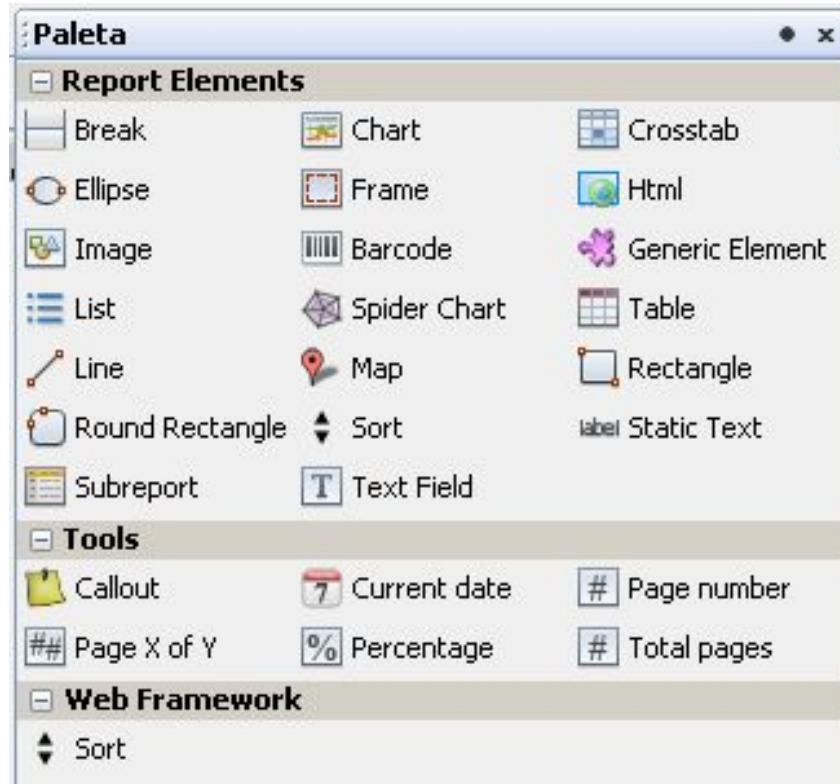
Ambiente - contém todos os recursos necessários para a elaboração de relatórios



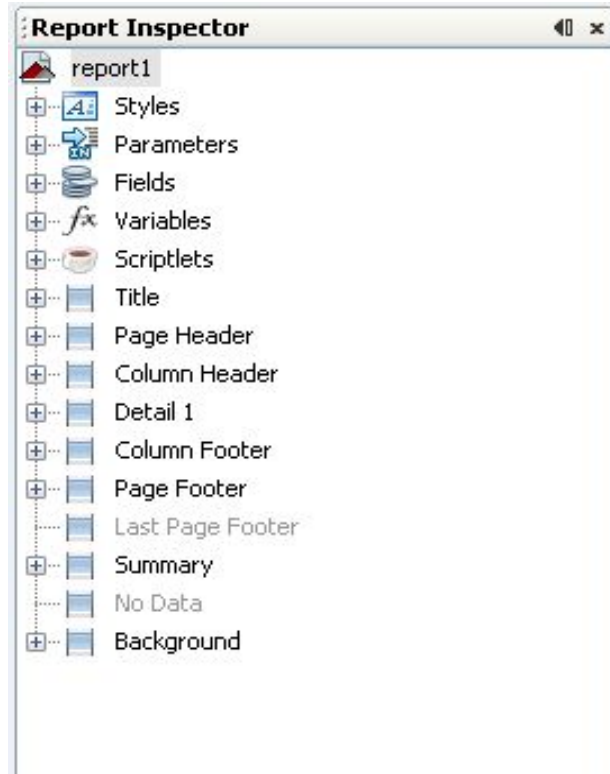
Estrutura do relatório - está dividido em 7 seções

| | | |
|--|---------------|--|
| | Title | |
| | Page Header | |
| | Column Header | |
| | Detail 1 | |
| | Column Footer | |
| | Page Footer | |
| | Summary | |

Menu Paleta – possui todos os components que pode ser utilizado na elaboração dos relatórios



Menu Report Inspector



Menu Propriedades

report1 - Propriedades

| | | |
|------------------------------------|--------------------------|-----|
| Report name | report1 | ... |
| Page size | | |
| Page width | 595 | |
| Page height | 842 | |
| Orientation | Portrait | ▼ |
| Margins | | |
| Left margin | 20 | |
| Right margin | 20 | |
| Top margin | 20 | |
| Bottom margin | 20 | |
| Columns | | |
| Columns | 1 | |
| Column Width | 555 | |
| Column space | 0 | |
| Print order | Vertical | ▼ |
| More... | | |
| Scriptlet class | | ... |
| Resource bundle | | ... |
| When Resource Missing Type | Type Null | ▼ |
| Query Text | | ... |
| The language for the dataset query | SQL | ▼ |
| Filter Expression | | ... |
| Properties | No properties set | ... |
| Title on a new page | <input type="checkbox"/> | |
| Summary on a new page | <input type="checkbox"/> | |

Referências

Framework Gerador de Relatórios / Wesley Daniel Silva Eliziario. Fundação Educacional do Município de Assis – FEMA – Assis,2014.

Ireport : Crie relatórios práticos e elegantes. Casa do Código . Autor: Maurício Morais

INTRODUÇÃO ao Hibernate. Devmedia. Disponível em:

<<https://www.devmedia.com.br/guia/hibernate/38312>>. Acesso em: 05 jun. 2018

<<https://www.devmedia.com.br/como-integrar-os-frameworks-spring-e-hibernate/23414>> Acesso em: 05 jun. 2018

<<https://www.caelum.com.br/apostila-java-web/spring-mvc/#um-pouco-de-histria>> Acesso em: 05 jun. 2018