

LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O
DESENVOLVIMENTO DO RIO GRANDE DO NORTE
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

BIBLIOTECA DIGITAL — IMPLEMENTAÇÃO DE UM PROTÓTIPO
UTILIZANDO JSP

Monografia apresentada ao
Curso de Graduação em
Sistemas de Informação da
Faculdade Natalense para o
Desenvolvimento do Rio Grande do
Norte como requisito parcial para
obtenção do grau de Bacharel
em Sistemas de Informação

Rodrigo Bruno de Azevedo Melo

NATAL
2002

RODRIGO BRUNO DE AZEVEDO MELO

FACULDADE NATALENSE PARA O
DESENVOLVIMENTO DO RIO GRANDE DO NORTE
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

**BIBLIOTECA DIGITAL — IMPLEMENTAÇÃO DE UM PROTÓTIPO
UTILIZANDO JSP**

Monografia apresentada ao
Curso de Graduação em
Sistemas de Informação da
Faculdade Natalense para o
Desenvolvimento do Rio Grande do
Norte como requisito parcial para
obtenção do grau de Bacharel
em Sistemas de Informação

Orientador:

Fellipe Araújo Aleixo, MsC.

**NATAL
2002**



M
004.4:021.61
M4866
2002

Melo, Rodrigo Bruno de Azevedo

Biblioteca Digital – Implementação de um protótipo utilizando JSP/Rodrigo Bruno de Azevedo Melo. ____ Natal: FARN, 2002.

69 p.

Monografia (Graduação em Sistemas de Informação) – Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte.

Orientador: Fellipe Araújo Aleixo, MsC.

1. Programa de Computador - Bibliotecas – monografia
2. Desenvolvimento de Software - Biblioteca – monografia

RN/FARN/BC

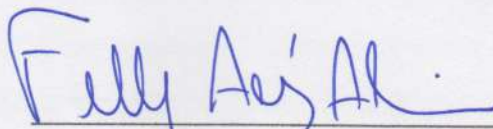
CDU 004.4:021.61

LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O
DESENVOLVIMENTO DO RIO GRANDE DO NORTE
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

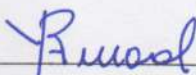
BIBLIOTECA DIGITAL – IMPLEMENTAÇÃO DE UM PROTÓTIPO
UTILIZANDO JSP

Rodrigo Bruno de Azevedo Melo

Monografia apresentada e aprovada em 11 de DEZEMBRO de 2002
pela banca examinadora composta pelos seguintes membros:



Prof. Fellipe Araújo Aleixo
Orientador



Prof. Ytalo Rosendo do Amaral
Examinador



Prof. Carlos Magno de Lima
Examinador

Natal, 11 de DEZEMBRO de 2002.

AGRADECIMENTOS

Agradeço aos meus pais por serem meus portos nos momentos de insegurança, que sempre acreditaram na minha capacidade e por terem me incentivado a realizar tarefas, as quais eu não fazia e devido a isso, mas que até hoje fazem as melhores "escolhas" finais por mim.

A amiga Ana Angélica (Keka) por ter sido a maior companheira e motivadora durante os estudos dos anos da faculdade, por ter me ajudado na conclusão deste trabalho e por ter me ajudado como se um melhor discente.

Pela colaboração, mesmo que indireta, por terem me ajudado e por me suportarem durante todo esse tempo, agradeço a tia Maria José, Antônio Costa, Jorge, Polyanne, Igar e Kika.

Por terem me proporcionado vários momentos de aprendizado e diversão, agradeço a meus grandes amigos Vanessa, Jésson (Snoopy) e Moimane.

Aos professores Lívia Maria, Adriano da Rocha e Marcos César Madruga, que realizaram uma papel com proeza, me proporcionando um vasto aprendizado na área da enfermagem. Um agradecimento especial ao meu orientador, o professor Felipe Alexei, por ter acreditado no meu trabalho, de sua amizade, paciência e pelo apoio durante os meses em que foi meu professor.

E a todos que, de alguma forma, contribuíram na elaboração deste trabalho.

Dedico este trabalho a todos os momentos que vivenciei, em que foi possível descobrir que a vida é como uma escada, e que não importa a altura do degrau, pois o esforço que se faz para subi-lo, será proporcional ao conhecimento adquirido com esta ação.

AGRADECIMENTOS

Agradeço aos meus, pais por serem meus portos nos momentos de insegurança, que sempre acreditaram na minha capacidade e por terem me incentivado a realizar tarefas, as quais eu não dava a devida atenção, mas que até hoje foram as melhores “escolhas” feitas por mim.

A amiga Ana Angélica (Keka) por ter sido a maior companheira e incentivadora durante os últimos dois anos da faculdade, por ter me ajudado na conclusão deste trabalho e por ter me ensinado como ser um melhor discente.

Pela colaboração, mesmo que indireta, por terem me acolhido e por me suportarem durante todo esse tempo, agradeço a tia Maria José, Antônio Costa, Jorge, Pollyanna, Igor e Artur.

Por terem me proporcionado vastos momentos de aprendizado e diversão, agradeço a meus grandes amigos Vanessa, Jailson (Snoopy) e Mounarte.

Aos professores Livia Maria, Aluizio da Rocha e Marcos César Madruga, que realizaram o seu papel com presteza, me propiciando um vasto aprendizado na área de informática. Um agradecimento especial ao meu orientador, o professor Fellipe Aleixo, por ter acreditado no meu trabalho, por sua amizade, paciência e pelo apoio durante os meses em que foi meu professor.

E a todos que, de alguma forma, contribuíram na elaboração deste trabalho.

SUMÁRIO

1. INTRODUÇÃO	10
1.1 APRESENTAÇÃO.....	10
1.2 PROBLEMA DE PESQUISA.....	10
1.3 OBJETIVO GERAL E ESPECÍFICO	11
1.4 METODOLOGIA	11
1.5 ESTRUTURA DO DOCUMENTO.....	12
2. REFERENCIAL TEÓRICO	13
2.1 HTML	13
2.1.1 FORMULÁRIOS HTML	13
2.1.2 TABELAS HTML	19
2.2 JAVA	20
2.3 JDBC (JAVA DATABASE CONNECTIVITY)	22
2.4 SQL (STRUCTURED QUERY LANGUAGE)	23
2.5 JSP (JAVASERVER PAGES).....	25
3. IMPLEMENTAÇÃO DO PROTÓTIPO DO SISTEMA BIBLIOTECA DIGITAL	27
3.1 RESUMO DA ANÁLISE.....	27
3.2 BANCO DE DADOS DO SISTEMA.....	28
3.3 DIAGRAMA DE CLASSES.....	29
3.4 ESCOPO DO SISTEMA.....	31
4. CONCLUSÃO	54
BIBLIOGRAFIA.....	55
ANEXO I – CÓDIGOS DE PÁGINAS HTML.....	58
ANEXO II – CÓDIGOS JSP	63
ANEXO III – CÓDIGOS DOS COMPONENTES JAVA.....	64
ANEXO IV – TELAS DO SISTEMA BIBLIOTECA DIGITAL.....	67

LISTA DE FIGURAS

Figura 1. Exemplo de código HTML.....	13
Figura 2. Estrutura básica de um formulário.....	14
Figura 3. Visualização no browser do campo TEXT.....	15
Figura 4. Visualização no browser do campo PASSWORD.....	15
Figura 5. Visualização no browser do campo RADIO BUTTON.....	16
Figura 6. Visualização no browser do campo CHECK BOX.....	16
Figura 7. Visualização no browser dos botões RESET e SUBMIT.....	17
Figura 8. Visualização no browser do campo TEXTAREA.....	18
Figura 9. Visualização no browser do campo COMBO BOX.....	18
Figura 10. Visualização no browser de uma tabela HTML.....	20
Figura 11. Ambiente de <i>runtime</i> de Java	21
Figura 12. Modelo de interação entre Java e Banco de Dados.....	22
Figura 13. Estrutura básica de uma consulta SQL.....	23
Figura 14. Estrutura básica de uma inserção SQL.....	24
Figura 15. Estrutura básica de uma atualização SQL.....	24
Figura 16. Estrutura básica de uma exclusão SQL.....	24
Figura 17. Separação da Apresentação e implementação em JSP.....	25
Figura 18. Relação entre os diagramas e o desenvolvimento do sistema.....	28
Figura 19. Diagrama de Entidade-Relacionamento do sistema.....	29
Figura 20. Diagrama de classe do sistema	30
Figura 21. Estrutura da aplicação “Cadastrar Usuário”	31
Figura 22 Diagrama de Colaboração – “Cadastrar Usuário”	32
Figura 23. Estrutura da aplicação “Submeter Texto ao Repositório”.....	33
Figura 24. Diagrama de Colaboração – “Submeter Textos ao Repositório”.....	34
Figura 25. Estrutura da aplicação “Consultar Catálogo da biblioteca”.....	35
Figura 26 Diagrama de Colaboração – “Consultar Catálogo da biblioteca”	36
Figura 27. Estrutura da aplicação “Requisitar Textos”.....	37
Figura 28. Diagrama de Colaboração – “Requisitar Textos”.....	37
Figura 29. Estrutura da aplicação “Alterar Dados do Texto”	38

Figura 30. Diagrama de Colaboração – “Alterar Dados do Texto”	39
Figura 31. Estrutura da aplicação “Aprovar Cadastro do Usuário”	40
Figura 32. Diagrama de Colaboração – “Aprovar Cadastro do Usuário”	41
Figura 33. Estrutura da aplicação “Emitir Relatório de Usuários Cadastrados”	41
Figura 34. Diagrama de Colaboração – “Emitir Relatório de Usuários Cadastrados”	42
Figura 35. Estrutura da aplicação “Emitir Relatório de Textos Cadastrados”	43
Figura 36. Diagrama de Colaboração – “Emitir Relatório de Textos Cadastrados”	43
Figura 37. Estrutura da aplicação “Remover Textos”	44
Figura 38. Diagrama de Colaboração – “Remover Textos”	45
Figura 39. Tela do menu Organizar Áreas	45
Figura 40. Estrutura da aplicação “Criar Nova Área”	46
Figura 41. Diagrama de Colaboração – “Criar Nova Área”	47
Figura 42. Estrutura da aplicação “Criar Nova Subárea”	48
Figura 43. Diagrama de Colaboração – “Criar Nova Subárea”	49
Figura 44. Estrutura da aplicação “ExcluirÁrea”	50
Figura 45. Diagrama de Colaboração – “ExcluirÁrea”	50
Figura 46. Estrutura da aplicação “Excluir Subárea”	51
Figura 47. Diagrama de Colaboração – “Excluir Subárea”	52
Figura 48. Estrutura da aplicação “Verificar Áreas e Subáreas Existentes”	52
Figura 49. Diagrama de Colaboração – “Verificar Áreas e Subáreas Existentes”	53

RESUMO

São incontestáveis os benefícios que os recursos de computação e comunicação tem alcançado atualmente. Tais avanços fazem com que as redes de computadores se tornem mais confiáveis e utilizadas pela sociedade de uma forma geral. Com o aumento da aceitação da Internet, surgem diversos serviços que visam suprir as necessidades das pessoas que procuram mais informação, facilidade, comodidade e rapidez. A troca de informação é uma das maiores utilidades da Internet até hoje. Os *sites* de busca começaram a surgir a partir da disseminação de pesquisas e o aumento do intercâmbio de idéias. Estes trabalham armazenando esses materiais e possibilitando buscas por materiais específicos. O referido trabalho trata da implementação de um protótipo de uma biblioteca digital, para armazenamento de informações, utilizando tecnologia JSP. A modelagem do banco de dados está sendo realizada no Access e a interface do sistema está sendo desenvolvida em HTML. Para que o mesmo execute via *web*, o sistema se utilizará de *scripts* JSP (*Java Server Pages*) que integrarão a interface aos componentes JSP *Beans*. Estes componentes são escritos em Java, contendo a funcionalidade da biblioteca digital. Posteriormente ao desenvolvimento, ocorrerão os testes e a validação do protótipo.

Palavras – chave: programação orientada objeto, Java, sistema de biblioteca, *web*.

ABSTRACT

The undisputed the decurrent benefits that the resources of the computation and telecommunications in the current days. These advances make with that the computer networks if the become more trustworthy and used by all the pursings of the society. With the increase of the Internet, new services appear that they to supply the necessities of the people who look more in formations, easiness, comfort and rapidity. The exchange informations are the biggest utilities of the Internet today. The sites of search to appear from research and increase of the dissemination of ideas, the work storing these materials and making possible searches of specific materials. This related work deals the model of implementation for the digital library, for storage information, using JSP. The modeling of the data base is being caused through in the Access and interface of the system is being developed in HTML, so that the same it executes way web, using scripts JSP that compose the interface to components JSP Beans, these writings in java, contend the functionality of the digital library.

Key words: orientation the objects programming, Java, system of library, *web*.

1.1 INTRODUÇÃO

Uma das grandes problemáticas enfrentadas atualmente por comunidades acadêmicas e pela sociedade como um todo ainda é o acesso às informações. Tem se tornado cada vez mais difícil conseguir encontrar informações relacionadas a vários outros assuntos, seja em diferentes comunidades, seja em sites de web. Isto se dá principalmente devido à distância entre as fontes de produção (universidades), centros de acesso às bibliotecas e pela quantidade e falta nos sites de busca e armazenamento de informações.

Essas dificuldades encontradas pelos centros acadêmicos atuais são a barreira que existe no momento de pesquisas entre comunidades e ou alunos, o que tende a diminuir a velocidade de pesquisa, bem como a falta de meios para publicação de artigos.

1 - INTRODUÇÃO

1.1 APRESENTAÇÃO

São incontestáveis os benefícios que os recursos de computação e comunicação tem alcançado atualmente. Tais avanços fazem com que as redes de computadores se tornem mais confiáveis e utilizadas por todos os seguimentos da sociedade. Com o aumento da aceitação da Internet, surgem diversos serviços que visam suprir as necessidades das pessoas que procuram mais informação, facilidade, comodidade e rapidez.

A troca de informação é uma das maiores utilidades da Internet até hoje. Os *sites* de busca começaram a surgir a partir da disseminação de pesquisas e o aumento do intercâmbio de idéias. Estes trabalham armazenando materiais e possibilitando buscas por informações específicas.

As bibliotecas digitais ou *e-library*, funcionam de maneira similar aos *sites* de busca, porém as bibliotecas digitais, armazenam materiais na forma digitalizada, enquanto que *sites* de busca, podem ou não serem os repositórios que contém as informações.

Este trabalho trata da implementação de um sistema de uma biblioteca digital. Esta implementação será baseada na modelagem orientada a objetos deste sistema, realizada em um trabalho desenvolvido pela aluna Ana Angélica Costa de Melo (MELO, 2002). A biblioteca digital funcionará na Internet, e terá inicialmente um subconjunto da funcionalidade especificada no trabalho de modelagem.

1.2 PROBLEMA DE PESQUISA

Um dos grandes problemas enfrentados atualmente nas comunidades acadêmicas e pela sociedade como um todo ainda é o acesso às informações. Tem se tornado cada vez mais difícil conseguir encontrar informações selecionadas e válidas sobre determinado assunto, seja em bibliotecas de universidades, seja em *sites* da *web*. Isto se dá principalmente devido: à distância entre os pólos de produção (universidades), carência de acervo nas bibliotecas e pela inconsistência e falhas nos *sites* de busca e armazenamento de informações.

Outras dificuldades encontradas pelos centros acadêmicos atuais são: a lentidão que existe no intercâmbio de pesquisas entre comunidade e/ou alunos, o que tende a diminuir a velocidade na resolução de pesquisas, bem como a falta de meios para publicação de artigos.

Tendo em vista as dificuldades acima descritas, apresenta-se como proposta para solução a construção de um sistema de biblioteca digital autônomo, em que seja possível concentrar informações de qualidade. Este sistema proverá busca funcional e disponibilidade ininterruptas, além de ser um veículo para publicação de artigos ou textos.

1.3 OBJETIVO GERAL E ESPECÍFICO

Este trabalho trata da implementação de um protótipo de um sistema de biblioteca digital. Como já foi citado, este protótipo será baseada na análise orientada a objetos deste sistema, realizada em um trabalho correlato. Visa-se provar a viabilidade do projeto, bem como a correteude do modelo desenvolvido. Serão experimentados o funcionamento autônomo, bem como a organização e busca de informações por parte do sistema.

Utilizaremos UML (Linguagem de Modelagem Unificada) para trabalharmos com os diagramas de classe, colaboração e atividade.

Será modelado o banco de dados, sendo aplicado em um sistema gerenciador de banco de dados, como o Access da Microsoft.

A interface do sistema será desenvolvida em HTML, para que o mesmo execute via *web*, e *scripts* JSP (*Java Server Pages*) que integrarão a interface aos componentes JSP *Beans*; estes escritos em Java, contendo a funcionalidade da biblioteca digital.

1.4 METODOLOGIA

A seguir todos os passos que serão seguidos para a realização do projeto:

1. Estudo sobre referências bibliográficas que supram as necessidades do projeto;
2. Estudo sobre as ferramentas (JSP, JDBC e SQL) que serão utilizadas no desenvolvimento do trabalho;
3. Modelagem do banco de dados, onde serão apresentados diagramas de entidades-relacionamento;
4. Elaboração do *design* a ser implementado;
5. Desenvolvimento dos componentes de *software* Java a serem utilizados no sistema;
6. Teste e validação do protótipo.

1.5 ESTRUTURA DO DOCUMENTO

Esta sessão apresenta uma breve descrição de cada capítulo que constitui o trabalho.

No capítulo 2 é abordado o referencial teórico usado como fundamentação desse trabalho. Onde são especificadas as ferramentas utilizadas como: HTML, Java, JDBC, SQL e JSP.

No capítulo 3 é apresentada a especificação da implementação do protótipo do sistema biblioteca digital, onde é mostrado um resumo da análise do sistema, e as funcionalidades deste é expressa através de diagramas de colaborações.

Finalmente, o capítulo 4 apresenta uma conclusão sobre o trabalho, crítica aos resultados obtidos e uma previsão para os trabalhos futuros.

Segundo Furgson (2001), HTML não é propriamente uma linguagem de programação, e sim, uma linguagem de marcação de texto que permite especificar como o texto será visualizado no navegador web, definindo os detalhes do documento por meio da utilização de elementos lógicos, conhecidos como tags ou etiquetas.

O exemplo a seguir apresenta a estrutura de um documento HTML.

```
<HTML>
  <HEAD>
    <TITLE> Título da página </TITLE>
  </HEAD>
  <BODY>
    <H1> <H2> <H3> <H4> <H5> <H6> </H1> </H2> </H3> </H4> </H5> </H6>
  </BODY>
</HTML>
```

Figura 1: Exemplo de código HTML

Nos documentos HTML, podemos inserir tags que são palavras ou imagens especiais que servem de cor, diferentes, cuja função é inserir documentos ou páginas armazenadas em arquivos eletrônicos na Internet.

3.1 Ferramentas HTML

As Ferramentas aumentam a interatividade das páginas web e são uma forma de comunicação entre visitantes e administradores de páginas. Servem para que o usuário digite

2 - REFERENCIAL TEÓRICO

Dentre as referências bibliográficas levantadas, convém destacar as seguintes:

2.1 HTML (*Hyper Text Markup Language*)

HTML é uma linguagem que adiciona marcação ao texto. Usada na criação de páginas para *web*, informam ao *browser* (*software* responsável pela visualização ou navegação de páginas *web*) como exibir uma determinada página. Essa linguagem consiste em um conjunto de códigos (chamados de marcas ou *tags*) usados para definir a aparência e funcionalidade das páginas da *web*. Um dos aspectos que tornaram HTML uma linguagem padrão e aceita mundialmente foi à independência de plataforma, esta independência possibilita a interpretação dos documentos HTML em máquinas com sistemas operacionais distintos.

Segundo Furgeri (2001), HTML não é propriamente uma linguagem de programação, e sim, uma linguagem de marcação de texto que permite especificar como o texto será visualizado na tela do navegador *web*, definindo os detalhes do documento por meio da utilização de pequenos códigos, conhecidos como *tags* ou etiquetas.

O exemplo a seguir apresenta a estrutura de um documento HTML:

```
<HTML>
  <HEAD>
    <TITLE> Título da página </TITLE>
  </HEAD>
  <BODY>
    <H1> <I> Conteúdo da página </H1> </I>
  </BODY>
</HTML>
```

Figura 1. Exemplo de código HTML.

Nos documentos HTML podemos inserir *links* que são palavras ou imagens marcadas geralmente de cor diferente, cuja função é interligar documentos ou páginas armazenadas em diferentes máquinas na Internet.

2.1.1 Formulários HTML.

Os Formulários aumentam a interatividade das páginas *web* e são uma forma de comunicação entre visitantes e administradores de páginas. Servem para que o usuário digite

informações ou escolha opções pré definidas, e podem ser usados para cadastro, estatísticas e até mesmo compras via internet.

Elementos Básicos de Formulários:

```
<FORM ACTION= "url" METHOD= "GET | POST " >
```

Campos de entrada de dados e textos simples.

```
</FORM>
```

Figura 2. Estrutura básica de um formulário.

Estes elementos delimitam um formulário no código HTML. Todos os outros elementos do formulário devem ficar entre eles. O atributo *action* deverá conter a *url* do programa que receberá os dados do formulário. O Atributo *method* especifica o método que os dados serão transmitidos, ele pode conter os atributos *get* ou *post*, a diferença é o modo como cada um empacota esses dados para enviar, o método *get* inclui as informações no final da *url*. O método *post*, envia todas as informações sem que possam ser vistas junto a *url*.

Input

Input significa entrada de dados, esse comando determina como será a entrada de dados nos campos do formulário, o atributo *Type* define qual será o elemento, (caixa de texto, botão de escolha ou botão simples). Abaixo será mostrado cada um dos tipos de valores para o atributo *Type* e o que cada um deles faz.

```
<INPUT TYPE=... NAME=... VALUE=... SIZE=... MAXLENGTH=...>
```

a) Texto:

```
<INPUT TYPE=TEXT NAME= "nome do campo" VALUE="Digite a informação"
SIZE=17 MAXLENGTH=30>
```

O atributo *Text* indica que o campo será de texto, ou seja, um campo onde o usuário digita dados, *Value* será usado se quisermos definir um valor pré-determinado que poderá ou não ser mudado pelo visitante, *Size* define o tamanho do campo, se quisermos um campo do tamanho de 100 caracteres digite: *Size=100*, mas o campo não se limitará a 100 caracteres, o limite será determinado pelo atributo *Maxlength*, no caso de um campo com no máximo 40 letras você colocaria *Maxlength=40*. O exemplo a seguir (ver figura 3) apresenta a visualização no browser do código descrito acima.

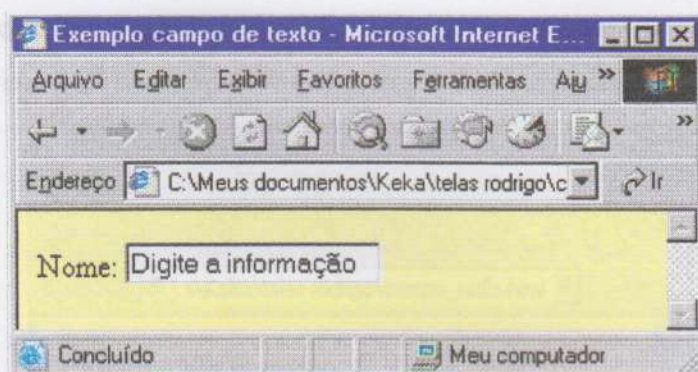


Figura 3. Visualização no browser do campo TEXT.

b) Senha:

`<INPUT TYPE=PASSWORD NAME=... VALUE=... SIZE=... MAXLENGTH=...>`

Com o atributo TYPE recebendo o valor *password* criamos um campo como o TEXT, porém com a diferença de que todo caracter que for digitado aparecerá como um asterisco, da mesma forma que aparece quando você digita sua senha no *dial-up* do windows. A seguir a visualização do campo senha.

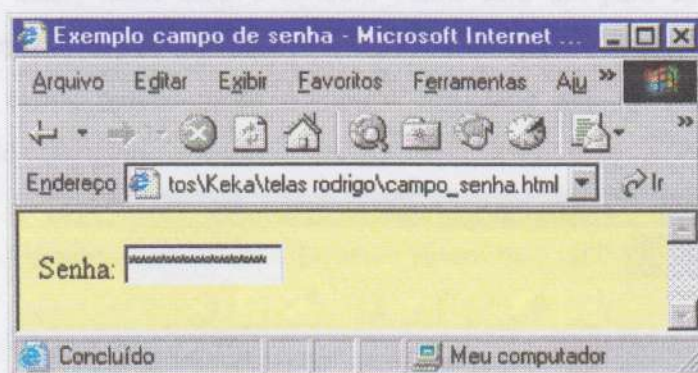


Figura 4. Visualização no browser do campo PASSWORD.

c) Radio Button:

`<INPUT TYPE=RADIO NAME=... VALUE=...CHECKED >`

O RadioButton são aqueles botões que o usuário pode escolher uma das opções mutuamente exclusivas. Podemos inserir na página um RadioButton colocando no atributo TYPE o valor Radio como foi indicado acima, o atributo VALUE deverá conter o valor deste campo, o atributo CHECKED é opcional e será usado para que uma opção seja selecionada como valor inicial podendo ou não ser mudada pelo visitante. E o Atributo NAME deverá ser a identificação

deste conjunto de RadioButtons sendo o mesmo para todos que fazem parte de um conjunto específico. Na figura 5 vemos a visualização de um exemplo de botões de opções.

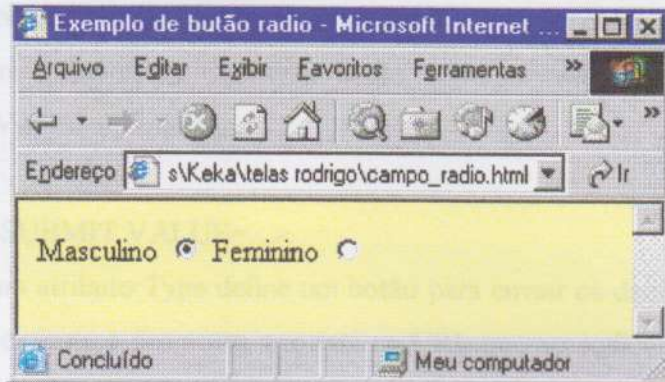


Figura 5. Visualização no *browser* do campo RADIO BUTTON.

d) Check Box:

`<INPUT TYPE=CHECKBOX NAME=... VALUE=... CHECKED>`

CheckBox são botões de checagem, esses botões são como os RadioButton porém podem ser selecionados vários deles, neste caso o atributo Name será um para cada campo, Value também será o valor repassado pelo programa no caso do campo ser assinalado, e o Checked poderá ser usado em vários botões tendo a mesma finalidade que no Radio. Exemplo da visualização de botões de checagem.

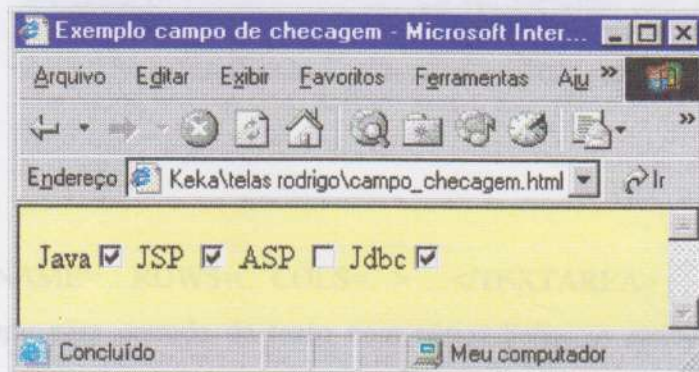


Figura 6. Visualização no *browser* do campo CHECK BOX.

e) Botão Reset:

```
<INPUT TYPE=RESET VALUE=...>
```

Reset no atributo Type define um botão que limpa todos os campos do formulário, no atributo Value você define o que estará escrito no botão, como por exemplo a palavra LIMPAR, o valor *default* é Reset. Ver exemplo na figura 7.

f) Botão Submit:

```
<INPUT TYPE=SUBMIT VALUE=...>
```

O Valor Submit no atributo Type define um botão para enviar os dados do formulário, ao acionar este botão você executa o programa que está na URL que especificou no atributo Action do tag Form, lá no início do formulário. Assim como no Reset o valor Value define o que será escrito no botão, o valor *default* é Submit. Ver exemplo na figura abaixo (figura 7).

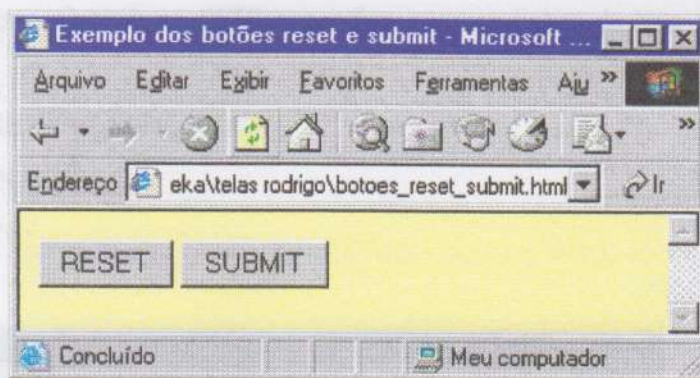


Figura 7. Visualização no *browser* dos botões RESET e SUBMIT.

Textarea:

```
<TEXTAREA NAME=... ROWS=... COLS=...> ... </TEXTAREA>
```

Define um campo para entrada de texto com várias linhas, o atributo Rows define o número de linhas e o atributo Cols o número de caracteres que cada linha possui, o atributo Name define o nome da caixa de texto. Caso queira evitar que uma caixa de texto possua uma barra de rolagem insira também `Wrap="hard"`. Na figura 8 é mostrado como é visualizado o exemplo do código de área de texto que contém mais de uma linha.

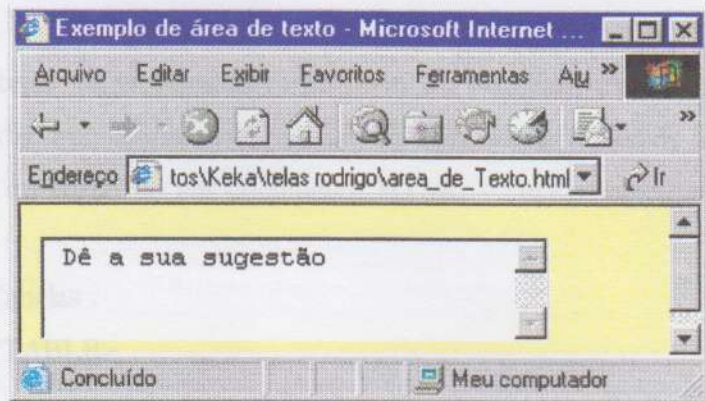


Figura 8. Visualização no *browser* do campo TEXTAREA.

Combo Box:

Este campo é definido com a tag `SELECT` para criar a lista e tags `OPTION` para definir cada opção dentro da lista. O atributo `SIZE` especifica quantas linhas da lista ficarão visíveis de cada vez.

```
<SELECT NAME = "x" SIZE = "z">
```

```
<OPTION>
```

```
<OPTION>
```

```
</SELECT>
```

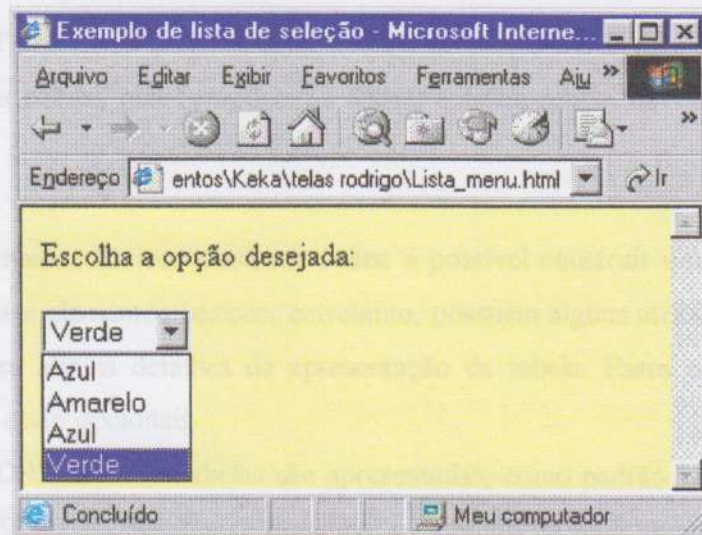


Figura 9. Visualização no *browser* do campo COMBO BOX.

2.1.2 Tabelas HTML.

Assim como tabelas de processadores de texto ou planilhas, as tabelas HTML contêm linhas e colunas formadas por células. Nas células de uma tabela podemos inserir tudo o que normalmente faz parte do corpo de um documento HTML - Textos, links, imagens ou mesmo outras tabelas.

Elementos Básicos de Tabelas :

`<TABLE> ... </TABLE>`

São as marcas que englobam a definição de uma tabela. Todas as demais marcas referentes a tabelas - título, linhas e células - somente serão consideradas se incluídas entre `<table>` e `</table>`.

`<CAPTION> ... </CAPTION>`

Trata-se de um elemento opcional que define o título da tabela, e deve constar entre as marcas que a definem, mas fora das marcas que definem linhas e colunas. Como padrão, o título é apresentado acima da tabela, centralizado.

`<TR> ... </TR>`

Este é o elemento utilizado para definição de linhas. Tabelas são definidas em linhas, sendo as linhas compostas de células. O Número de linhas de uma tabela corresponde ao número de elementos `<TR>`.

`<TD> ... </TD>`

Delimitam as células que compõem as linhas, e assim sendo, devem estar inseridas entre marcas de linhas

Atributos :

Com os elementos básicos descritos acima é possível construir uma tabela e visualizá-la no seu *Browser*. Estes elementos básicos, entretanto, possuem alguns atributos que permitem um maior controle sobre alguns detalhes da apresentação da tabela. Esses atributos são inseridos dentro do comando e são opcionais.

`<TABLE BORDER=...>` Tabelas são apresentadas, como padrão sem bordas. A Presença do atributo `BORDER` indica justamente que deve ser desenhada uma borda fazendo contorno de cada célula da tabela. `BORDER=...` A especificação da espessura da borda deve ser em *Pixels*.

`<TABLE WIDTH=...>` Especifica a largura da tabela, que pode ser definida em *pixels* ou em porcentagem referente a largura da janela. Caso não seja especificado, o próprio *browser* se

encarrega de determinar a largura mais adequada. Na figura 10 podemos observar um exemplo criado a partir dos elementos básicos de uma tabela:

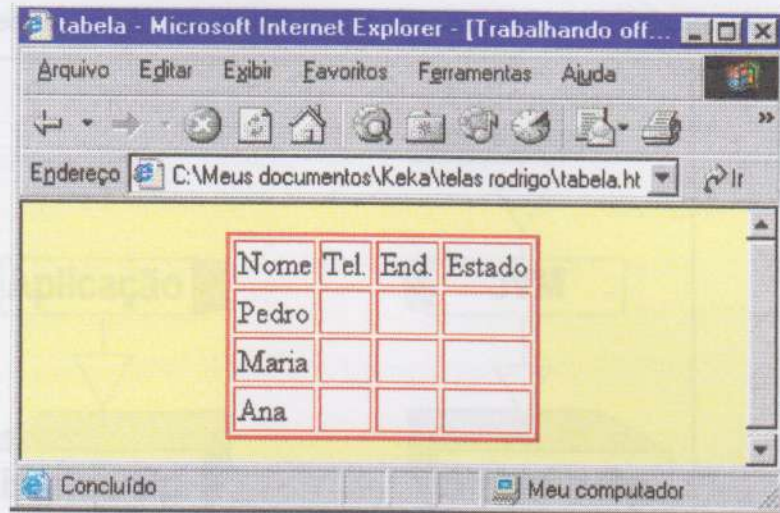


Figura 10. Visualização no *browser* de uma tabela HTML.

HTML descreve documentos estáticos, mas a necessidade leva os desenvolvedores a criar mecanismos que tornem o conteúdo, mostrado em um *browser*, dinâmico.

2.2 JAVA

Os programas escritos nessa linguagem podem ser executados na maioria das plataformas, ou seja, uma mesma aplicação Java pode “rodar” em diferentes sistemas operacionais como *Windows*, *Unix* e *Macintosh*.

A plataforma Java trata-se de um software que executa sobre o hardware de uma plataforma qualquer (Máquina Virtual Java - JVM), e a maioria das plataformas é formada pelo conjunto “*hardware* e sistema operacional”.

Java é uma linguagem compilada e interpretada. O código fonte Java é transformado em instruções binárias simples, no mesmo nível da linguagem dos processadores, *bytecode* – instruções para uma Máquina Virtual Java. Os *bytecodes* de Java, também chamados de *j-code*, são executados por um interpretador Java em *runtime* (tempo real). O sistema de runtime realiza todas as funções de um processador real de uma forma segura, porém virtual.

Desta forma o *bytecode* se torna portátil. A mesma aplicação Java de *bytecode* pode ser executada em qualquer plataforma que possua um ambiente *runtime* (ver figura 11).

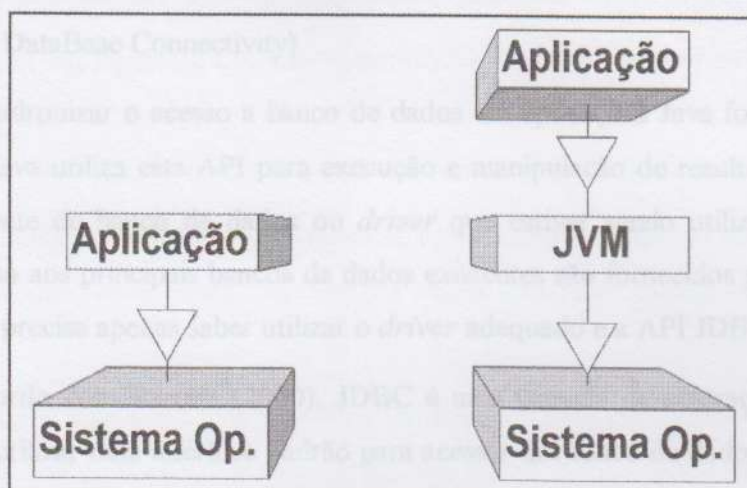


Figura 11. O ambiente de *runtime* de Java.

Segundo Deitel e Deitel (2001), Java é uma linguagem de programação que está revolucionando o desenvolvimento de *software* com código orientado a objetos e uso intensivo de multimídia, independente de plataforma para aplicativos convencionais e *applets* baseados em Internet, Intranet e Extranet.

Niemeyer e Knudsen (2000) afirmam que, Java foi projetada para ser uma linguagem de programação independente de máquina, que seja segura o suficiente para atravessar redes e poderosa o bastante para substituir o código executável nativo; além de ser a plataforma preferida para as modernas aplicações distribuídas.

Java pode executar programas interativos e seguros em rede, além de propiciar conteúdo dinâmico a páginas HTML. Este dinamismo é conseguido com o acréscimo de código Java, que são introduzidos em páginas *web* e processados pelas máquinas clientes. Desta forma, programas Java que funcionam em páginas *web* são chamados *Applets*.

Um programa Java é feito de objetos com certas propriedades e operações que estes podem realizar. A POO concentra-se nas interfaces com os objetos e com o que estes objetos expõem.

Ser totalmente uma linguagem de programação orientada a objetos, propicia uma maximização na reutilização do código e na confiabilidade dos dados, já que cada objeto é o único responsável pelo acesso, processamento e armazenamento de seus dados.

2.3 JDBC (Java DataBase Connectivity)

Para padronizar o acesso a banco de dados em aplicações Java foi criada a API JDBC. Um programa Java utiliza esta API para execução e manipulação de resultados a consultas SQL independentemente de banco de dados ou *driver* que estiver sendo utilizado. Os *drivers* para conexão e acesso aos principais bancos de dados existentes são fornecidos pelos seus fabricantes. O programador precisa apenas saber utilizar o *driver* adequado e a API JDBC.

De acordo com Ramon (2000), JDBC é uma camada de abstração que permite a um programa Java utilizar uma interface padrão para acessar um banco de dados relacional através da linguagem SQL.

As classes JDBC se encontram no pacote `java.sql`, que é importado na classe de java, a qual se deseja fazer a comunicação com o banco.

A *Microsoft* possui uma interface chamada ODBC que dá suporte para sistemas acessarem banco dados relacionais como o Access e SQL Server. Java não interage diretamente com drivers padrões da plataforma *windows* (ODBC), a maneira mais fácil de um programa java se comunicar com um banco de dados que utiliza interface ODBC, é através da ponte JDBC-ODBC.

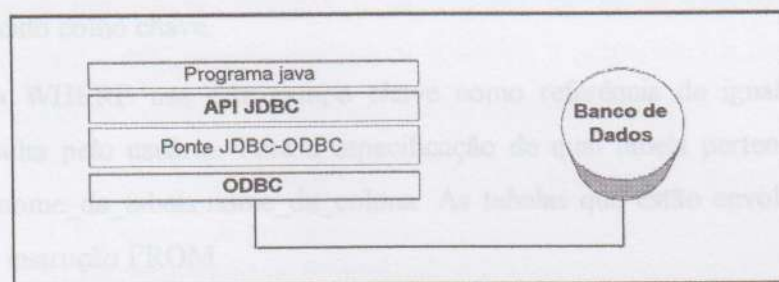


Figura 12. Modelo de interação entre java e banco de dados.

A ponte JDBC-ODBC é um driver que “traduz” as chamadas JDBC para chamadas ODBC, de forma transparente para o programador.

2.4 SQL (*Structured Query Language*)

O grande sucesso dos bancos de dados relacionais atuais se deve principalmente a facilidade que os usuários tem de manipular ou acessar os seus dados.

A SQL (Linguagem de consulta estruturada) é uma linguagem de acesso a dados muito conhecida e amplamente usada; sob ela estão os componentes que permitem a manipulação e a capacidade de recuperação, de dados que estão armazenados, em um banco de dados.

As consultas SQL a dados são feitas de forma simples. Para um melhor entendimento de como os acessos são feitos, serão mostrados a seguir comandos básicos desta linguagem e sua sintaxe:

SELECT é o comando mais usado e fundamental para que se realize uma consulta SQL. Ele é responsável pela seleção das colunas desejadas, e pode ser acrescido de outros comandos:

```
SELECT <campos a serem selecionados>  
[FROM] <lista de tabelas no caso de junção>  
[WHERE] <critérios para a seleção>
```

Figura 13. Estrutura básica de uma consulta SQL.

Para que haja uma maior economia de memória e maior velocidade na busca de informações, os dados são organizados em tabelas distintas e associadas uma as outras através de um campo, conhecido como chave.

A cláusula WHERE usa este campo chave como referência de igualdade aos valores passados na consulta pelo usuário. Para a especificação de qual tabela pertence aquele campo, usa-se o padrão nome_da_tabela.nome_da_coluna. As tabelas que estão envolvidas na pesquisa são descritas pela instrução FROM.

No banco de dados, quando é feita uma consulta, é criada uma nova tabela com a cópia dos dados da(s) outra(s) tabela(s) sem alterações nas originais. Os recursos oferecidos pela linguagem SQL para a inserção, atualização e exclusão de dados, são oferecidos de forma similar as pesquisas.

O comando INSERT geralmente é usado para inserir uma nova tupla na tabela conforme esquema da figura 4:

```
INSERT INTO <nome da tabela> <lista de campos>.  
VALUES <valores dos novos campos>
```

Figura 14. Estrutura básica de uma inserção em SQL

É necessário que se especifique em quais campos serão inseridos os novos dados e qual o valor do seu preenchimento.

A segunda maneira de modificar tabelas, é a alteração de dados que já estão gravados no banco de dados. Para isso usa-se a cláusula UPDATE, que indica qual tabela será manipulada, e que geralmente vem acompanhada de um WHERE ao qual está associado à condição para a escolha das tuplas a serem modificadas. Um outro elemento do UPDATE é o SET, real responsável pela mudança, nesta cláusula é descrito o nome do campo e seu novo valor.

```
UPDATE <nome da tabela modificada>  
SET <nome dos campos e novos valores>  
WHERE <condição para mudança>
```

Figura 15. Estrutura básica de uma atualização em SQL

O comando DELETE, o terceiro dos comandos de modificação de dados, serve para a exclusão de linhas de uma tabela. Como a retirada é feita por linhas inteiras do banco, não é necessária a especificação dos campos; a cláusula WHERE é usado para a escolha das linhas a serem deletadas, e a FROM o nome da tabela em que se encontra a linha.

```
DELETE FROM <nome da tabela>  
WHERE <condição de exclusão da linha>
```

Figura 16. Estrutura básica de uma exclusão em SQL

2.5 JSP (JavaServer Pages)

Usada no desenvolvimento de páginas web de conteúdo dinâmico, JSP tem se mostrado uma tecnologia de grande aceitação, principalmente pelo fato de usufruir todos os benefícios da linguagem Java.

Por ser uma extensão da linguagem Java para *web*, JSP também é portátil. Isto se dá pelo fato de que os bytecodes de java compilados podem ser interpretados em qualquer máquina que suporte uma Máquina Virtual Java, e por possui uma API (Application Programming Interface, Interface de Programação de Aplicativos) padrão aberta – grande biblioteca de programação que consiste em classes prontas que os programadores usam em suas aplicações.

Outra vantagem da JSP, é que ela permite ao desenvolvedor de *sites* produzir aplicações que permitam o acesso à banco de dados, o acesso a arquivos-texto e a captação de informações a partir de formulários.

Como sugerido na figura 9, com JSP é possível manter uma rígida separação entre apresentação de dados – exibição de informações para usuário final – e a implementação do programa – código usado para gerar aquela informação. O benefício proveniente da separação destes dois aspectos, é que as modificações em um podem ser realizadas sem a necessidade de qualquer alteração no outro. Essa tecnologia permite a utilização de *tags* HTML de JSP para o acesso as funcionalidades dos componentes *JavaBeans*, se nenhuma das *tags* disponíveis fornecer a funcionalidade necessária desejada, pode-se escrever uma biblioteca de *tags* específicas de aplicação (*tags* personalizadas), usando o mecanismo de extensão de *tags* JSP.

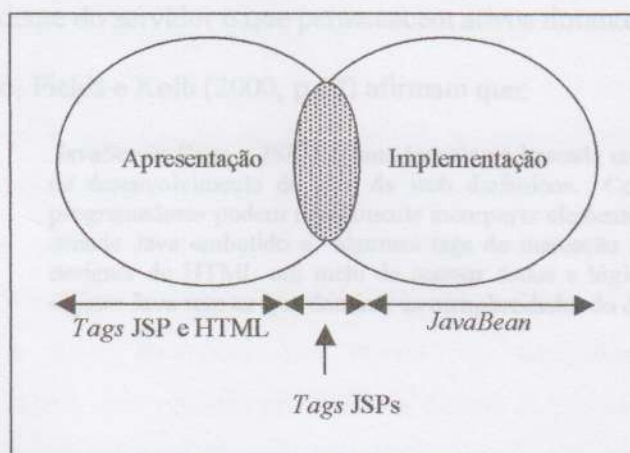


Figura 17. Separação da apresentação e implementação em JSP

3- IMPLEMENTAÇÃO DO PROTÓTIPO DO SISTEMA BIBLIOTECA DIGITAL

O modelo de componentes de JSP está centralizado em objetos chamados *JavaBeans*. Estes são elementos que contêm dados ou executam funções específicas sem revelar detalhes de funcionamento (encapsulamento). Outra característica dos *JavaBeans* é de não estarem vinculados a uso ou aplicações únicas. Isto permite que eles sejam usados como blocos de construção em vários projetos (reusabilidade).

As tecnologias que envolvem a criação modular de aplicações JSP, encontram suas forças nos servlets. Servlets são componentes, pequenos programas na linguagem Java, que geram as páginas visualizadas pelo navegador. Elas funcionam de modo semelhante aos scripts CGI (Common Gateway Interface). Sua grande vantagem está na velocidade depois da primeira execução. Eles podem conter a lógica e a apresentação do conteúdo de uma apresentação ou podem funcionar apenas como componentes da aplicação, dando suporte a outros servlets ou a páginas JSP.

Os servlets JSP podem ser considerados classes Java, e como tais, se apóiam em todos os mecanismos e ferramentas da plataforma Java, como independência de plataforma, mecanismos de segurança que servem a Java bem como acesso aos pacotes e suas ferramentas (classes, interfaces, exceções, métodos, atributos, etc).

Os servlets surgem no cenário da Internet como uma alternativa mais eficiente aos scripts. Isto porque é criado um novo processo para pô-los em funcionamento a cada nova solicitação, o que implica em maior demanda de recursos de memória. Diferentemente dos servlets que rodam em módulo independente do servidor e que permanecem ativos durante várias solicitações.

Neste sentido, Fields e Kolb (2000, p.02) afirmam que:

JavaServer Page – JSP – é uma tecnologia baseada em Java que simplifica o processo de desenvolvimento de sites da web dinâmicos. Com JSP, os designers da web e programadores podem rapidamente incorporar elementos dinâmicos em páginas da web usando Java embutido e algumas tags de marcação simples. Estas tags fornecem ao designer de HTML um meio de acessar dados e lógica de negócios armazenados em objetos Java sem ter que dominar as complexidades do desenvolvimento de aplicações.

3 – IMPLEMENTAÇÃO DO PROTÓTIPO DO SISTEMA BIBLIOTECA DIGITAL

O sistema será desenvolvido de acordo com as especificações da análise orientada a objetos, desenvolvida num trabalho correlato da aluna Ana Angélica Melo, que conta com as seguintes funções básicas:

1. Cadastrar Usuário;
2. Submeter Textos ao Repositório;
3. Consultar Catálogo da Biblioteca;
4. Requisitar Textos;
5. Alterar dados do Texto;
6. Aprovar Cadastro do Usuário;
7. Emitir Relatório de Usuários Cadastrados;
8. Emitir Relatório de Textos Cadastrados;
9. Remover Textos;
10. Organizar Áreas e Subáreas;
11. Criar Nova Área;
12. Criar Nova Subárea;
13. Excluir Área;
14. Excluir Subárea;
15. Verificar Áreas e Subáreas Existentes.

3.1 RESUMO DA ANÁLISE

Esta fase fornece um bom entendimento do sistema. Serão abordados itens que foram necessários para a concepção deste.

Ao iniciarmos a modelagem de um sistema, primeiramente descrevemos o problema, analisamos os seus requisitos e definimos o escopo do sistema.

Descrever o sistema com o máximo de detalhes possíveis, enfatizando os pontos fundamentais para seu funcionamento, durante esta fase o contato com o usuário é de extrema importância; pois é neste momento que vamos nos ambientar melhor em relação ao funcionamento do sistema, seus maiores problemas e discutir as possíveis soluções.

Para a coleta de dados, foram utilizados alguns instrumentos, como aplicação de questionários e entrevistas com os usuários, que foram realizadas em maio do ano corrente. Os

possíveis usuários do sistema são: os alunos, professores e funcionários da instituição de ensino, onde o sistema está sendo desenvolvido.

A figura 18 mostra como foram desenvolvidos os componentes e telas que compõem a Biblioteca Digital; a partir dos diagramas de colaboração foram analisados os requisitos para o desenvolvimento das telas HTML e JSP. Já os diagramas de classes serviram como base para a criação dos componentes Java.

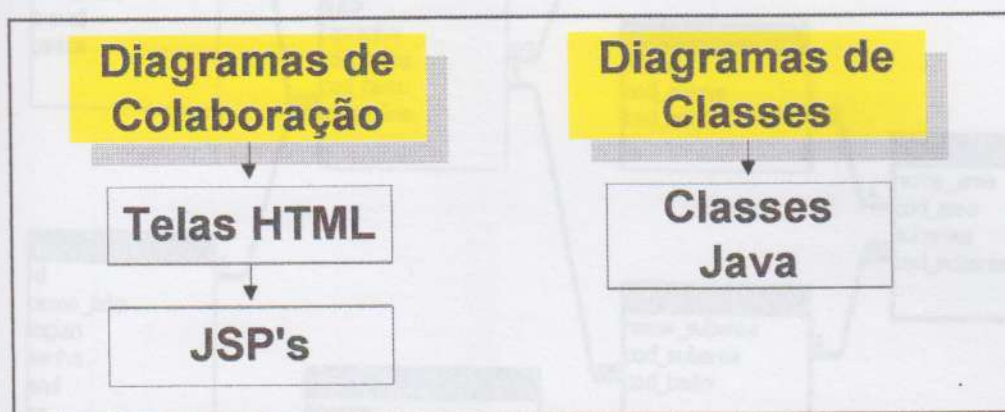


Figura 18. Relação entre os diagramas e o desenvolvimento do sistema

3.2 BANCO DE DADOS DO SISTEMA

O modelo de dados Entidade-Relacionamento (E-R) tem por base a percepção do mundo real, consistindo em um conjunto de objetos básicos chamado entidades e em relacionamento entre estas entidades. Este modelo é desenvolvido para facilitar o projeto do banco de dados e a especificação do sistema. Tal modelo representa toda a estrutura lógica do banco de dados.

Um banco de dados apresentado por um diagrama E-R é representado por uma coleção de tabelas. A conversão da representação do banco de dados de um diagrama E-R para a forma de tabelas é a base para a derivação do diagrama E-R para um banco de dados relacional.

A seguir é apresentado o diagrama de entidade-relacionamento (ver figura 19) referente à modelagem do banco de dados do sistema biblioteca digital.

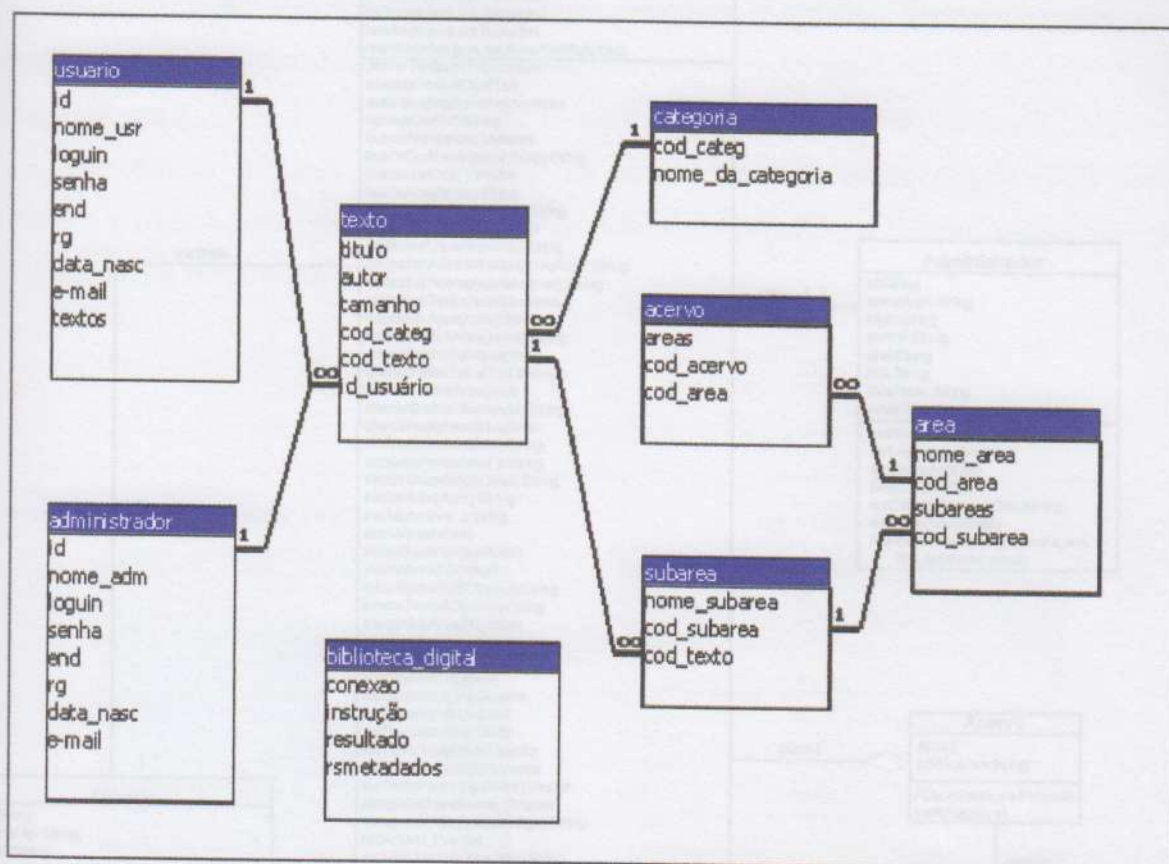


Figura 19. Diagrama de Entidade-Relacionamento do sistema.

3.3 DIAGRAMA DE CLASSES

O diagrama de classes do sistema (ver figura 20) foi gerado na análise orientada a objeto, desenvolvida em um trabalho correlato (MELO,2002), assim como os diagramas de colaboração descritos nas sessões seguintes.

No diagrama de classes estão representadas todas as classes que compõem o sistema da Biblioteca Digital, bem como os seus atributos e métodos.

Figura 20. Diagrama de Classes

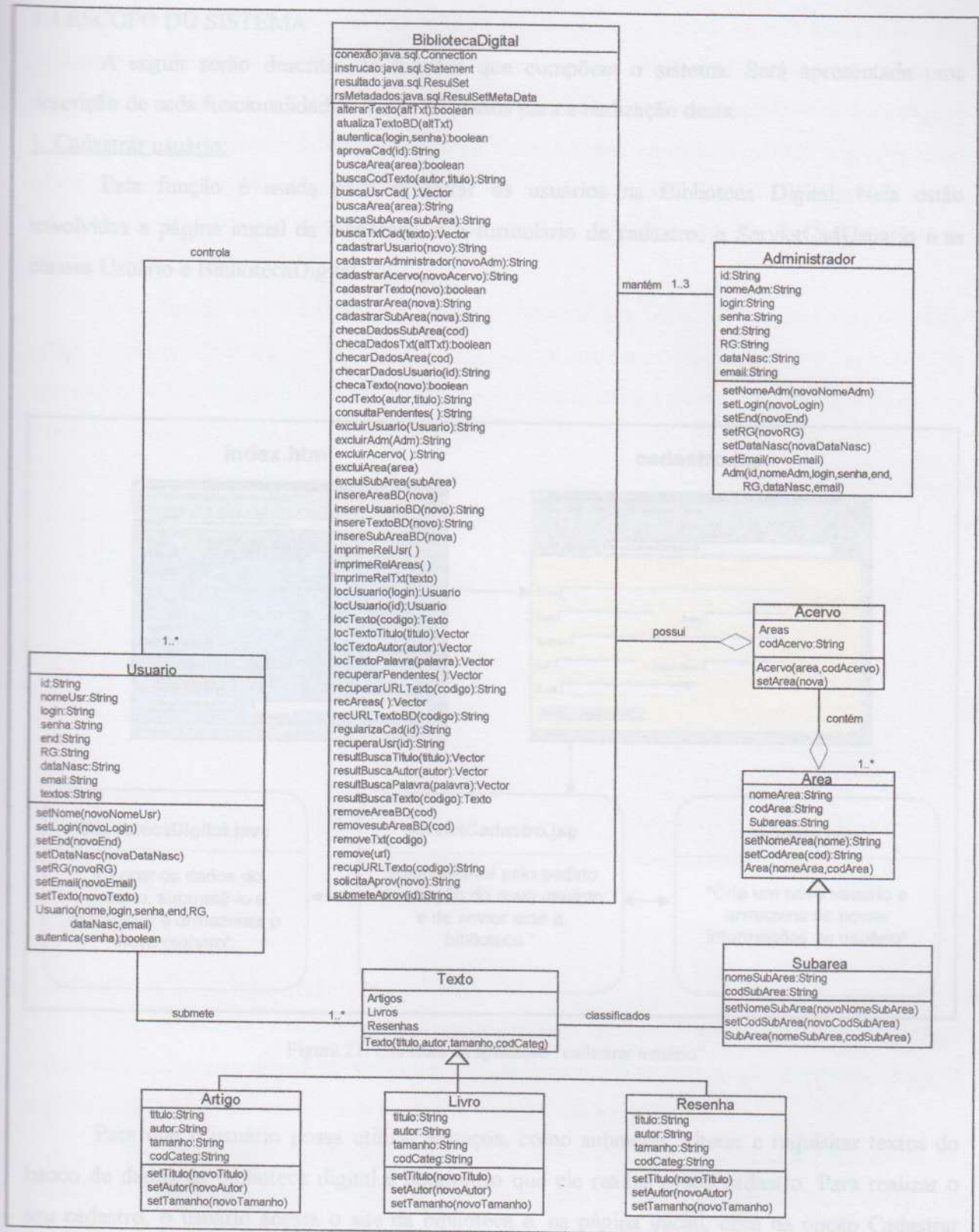


Figura 20. Diagrama de Classes

3.4 ESCOPO DO SISTEMA

A seguir serão descritas as funções que compõem o sistema. Será apresentada uma descrição de cada funcionalidade e todos os passos para a realização desta.

1. Cadastrar usuário:

Esta função é usada para cadastrar os usuários na Biblioteca Digital. Nela estão envolvidas a página inicial da biblioteca, e o formulário de cadastro, o ServletCadUsuario e as classes Usuário e BibliotecaDigital.

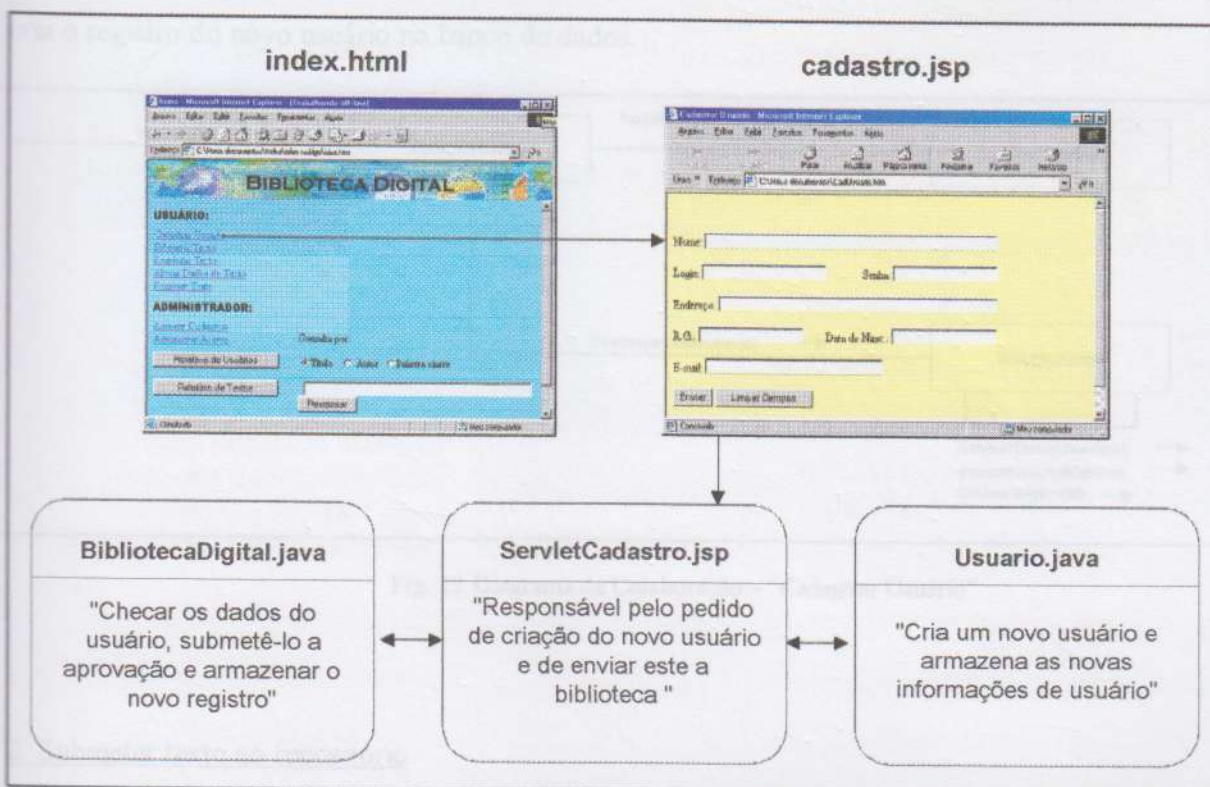


Figura 21. Estrutura da aplicação "cadastrar usuário".

Para que o usuário possa utilizar serviços, como submeter, alterar e requisitar textos do banco de dados da biblioteca digital é necessário que ele realize o seu cadastro. Para realizar o seu cadastro, o usuário acessa o site da biblioteca e, na página inicial, clica na opção Cadastrar Usuário. O sistema exibe um formulário html contendo campos para inserção do nome do

usuário, endereço, número da identidade, data de nascimento, e-mail, login e senha. Com um click no botão Enviar do formulário, os dados são enviados ao ServletCadUsuario.

Este servlet é responsável pelo “pedido” de criação do novo usuário e de inserção no banco de dados. O ServletCadUsuario acessa a classe Usuário através do método criar, passando como parâmetro às informações do usuário. É criado um novo usuário (instância da classe usuário), e através do método cadastrarUsuario o novo usuário é enviado à classe BibliotecaDigital.

A biblioteca checa através do id, se as informações contidas no novo usuário estão corretamente preenchidas, caso as informações estejam corretas, a biblioteca envia um pedido de aprovação de cadastrado ao administrador. Se o administrador aprovar o cadastro, a biblioteca cria o registro do novo usuário no banco de dados.

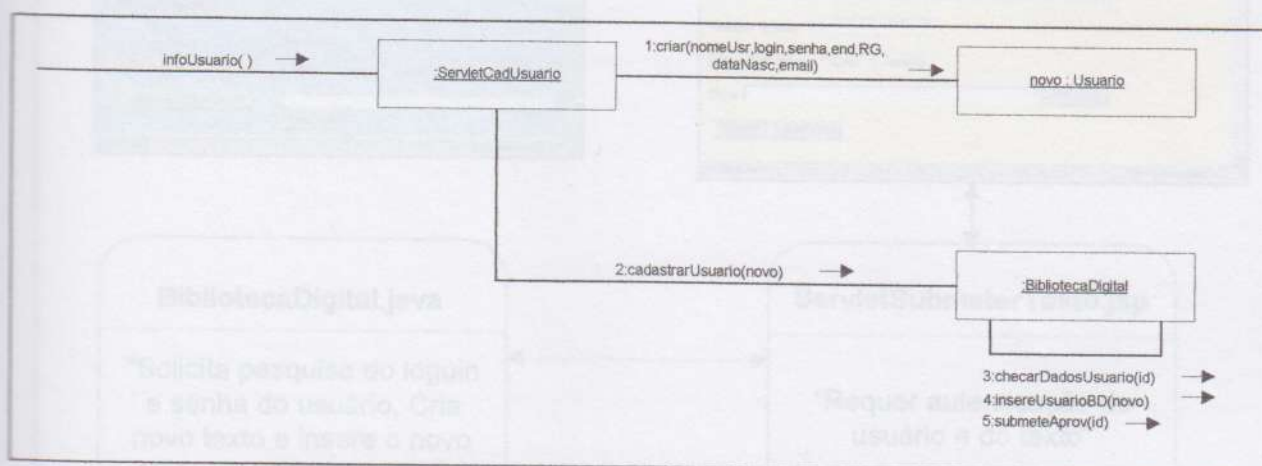


Fig. 22 Diagrama de Colaboração – “Cadastrar Usuário”

2. Submeter texto ao repositório

Para o usuário submeter textos na biblioteca digital, é necessário que ele esteja cadastrado. Na página inicial da biblioteca o usuário deve clicar na opção Submeter texto. O sistema mostrará um formulário para o usuário inserir o seu login e senha.

O ServletSubmeterTextos requisita a classe BibliotecaDigital a verificação dos dados do formulário. A biblioteca localiza nos seus registros o usuário através do método locUsuario(), e se ele existir, a classe Usuário faz uma pesquisa dentre os usuários se existe o login e a senha correspondente.

Caso a resposta da classe Usuário seja bem sucedida, o sistema mostra um formulário para inserção dos dados do texto (nome do autor, login, senha, título do texto, tamanho do texto, categoria do texto - livro/ artigo/ resenha/ documento), e um botão onde o usuário clica para localizar onde o arquivo está, e outro, para após a localização, que anexa o texto ao formulário. Depois de anexado o usuário deve clicar em Submeter.

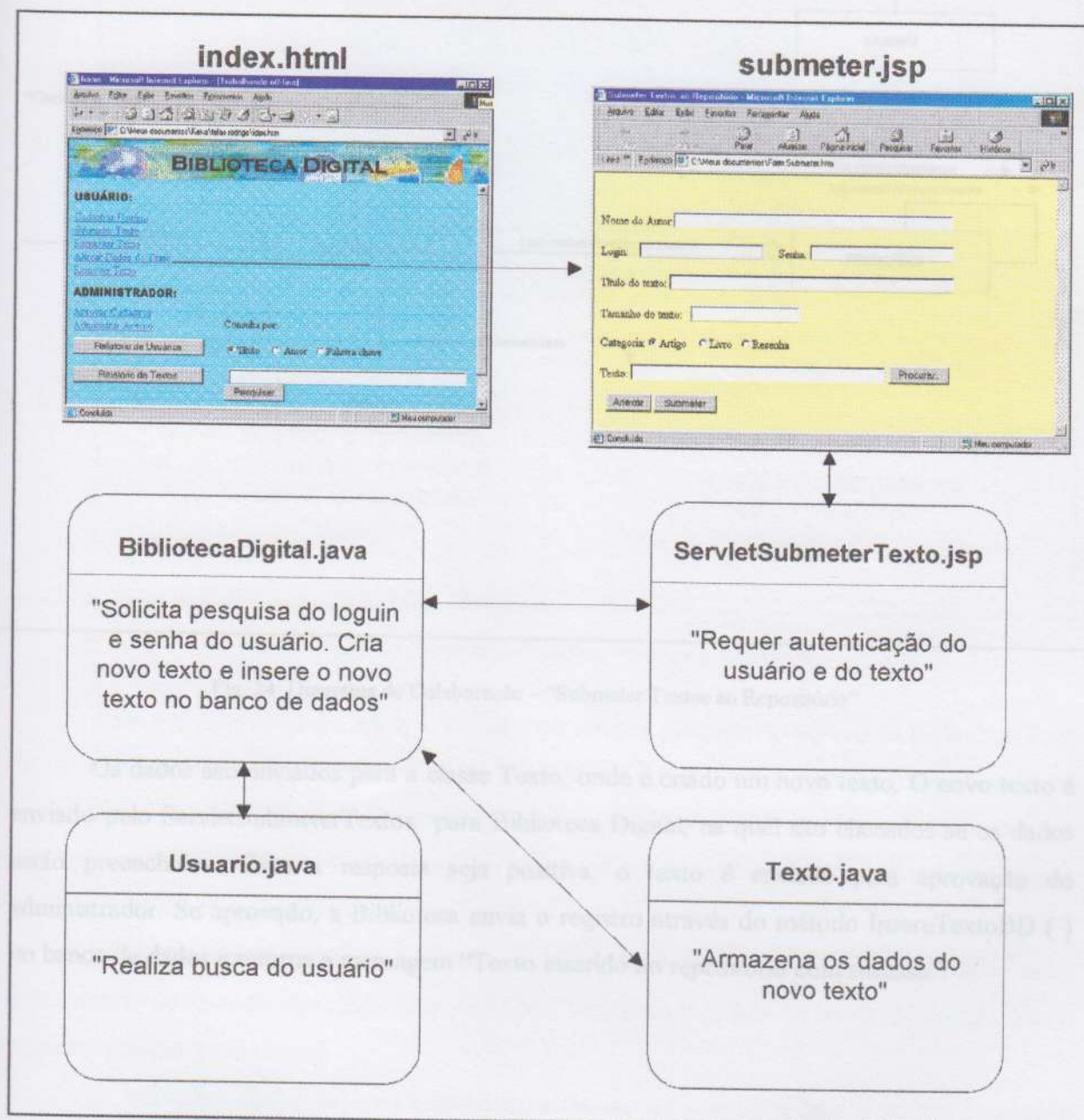


Figura 23. Estrutura da aplicação "Submeter texto ao repositório".

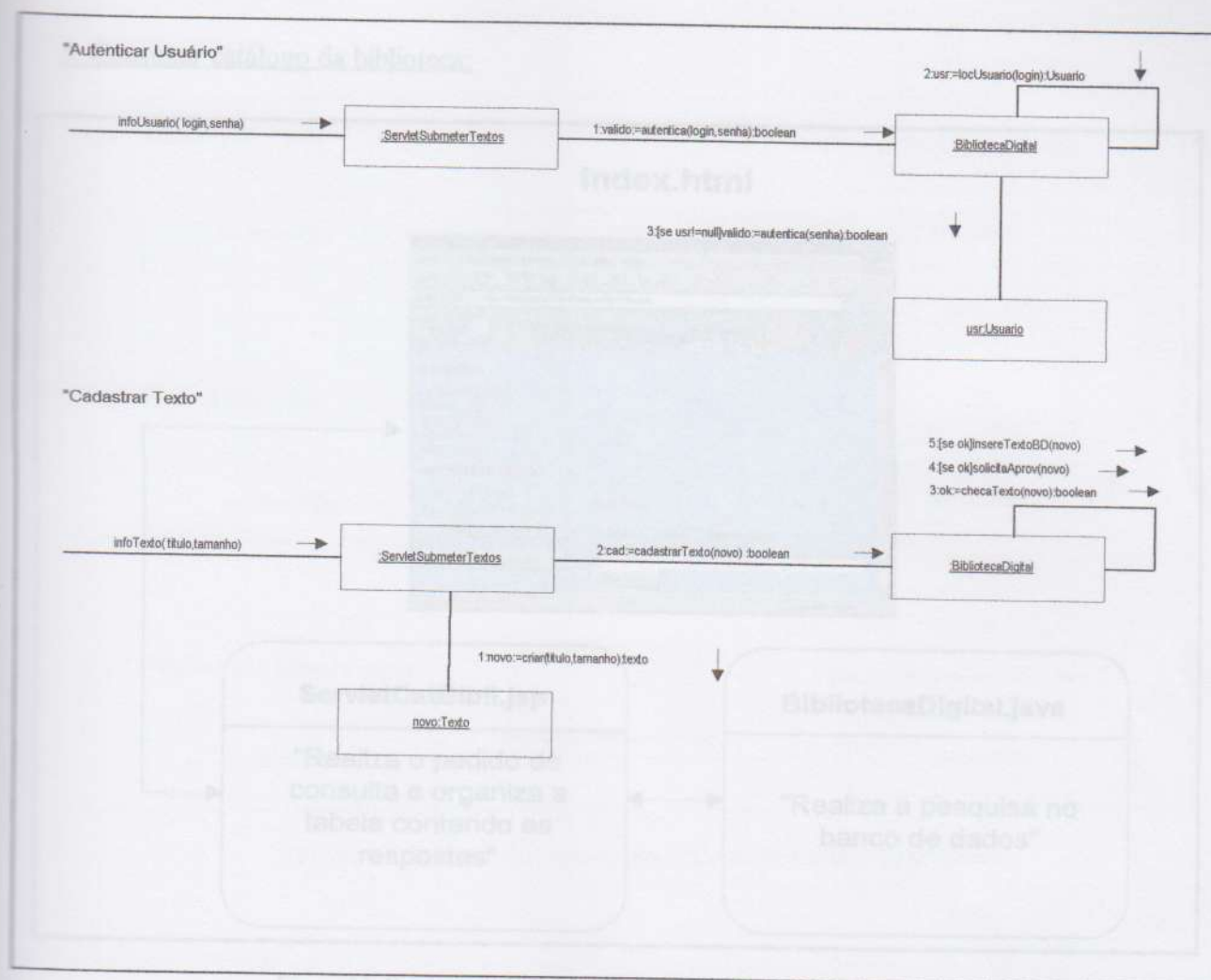


Fig. 24 Diagrama de Colaboração – "Submeter Textos ao Repositório"

Os dados são enviados para a classe `Texto`, onde é criado um novo texto. O novo texto é enviado pelo `ServletSubmeterTextos` para `Biblioteca Digital`, na qual são checados se os dados estão preenchidos. Caso a resposta seja positiva, o texto é enviado para aprovação do administrador. Se aprovado, a `Biblioteca` envia o registro através do método `InsererTextoBD()` no banco de dados e retorna a mensagem "Texto inserido no repositório com sucesso".

A biblioteca digital recebe os parâmetros do servlet e executa uma pesquisa de acordo com a pesquisa escolhida e retorna ao servlet o vetor `locTexto` contendo as possíveis respostas de pesquisa.

3. Consultar catálogo da biblioteca:

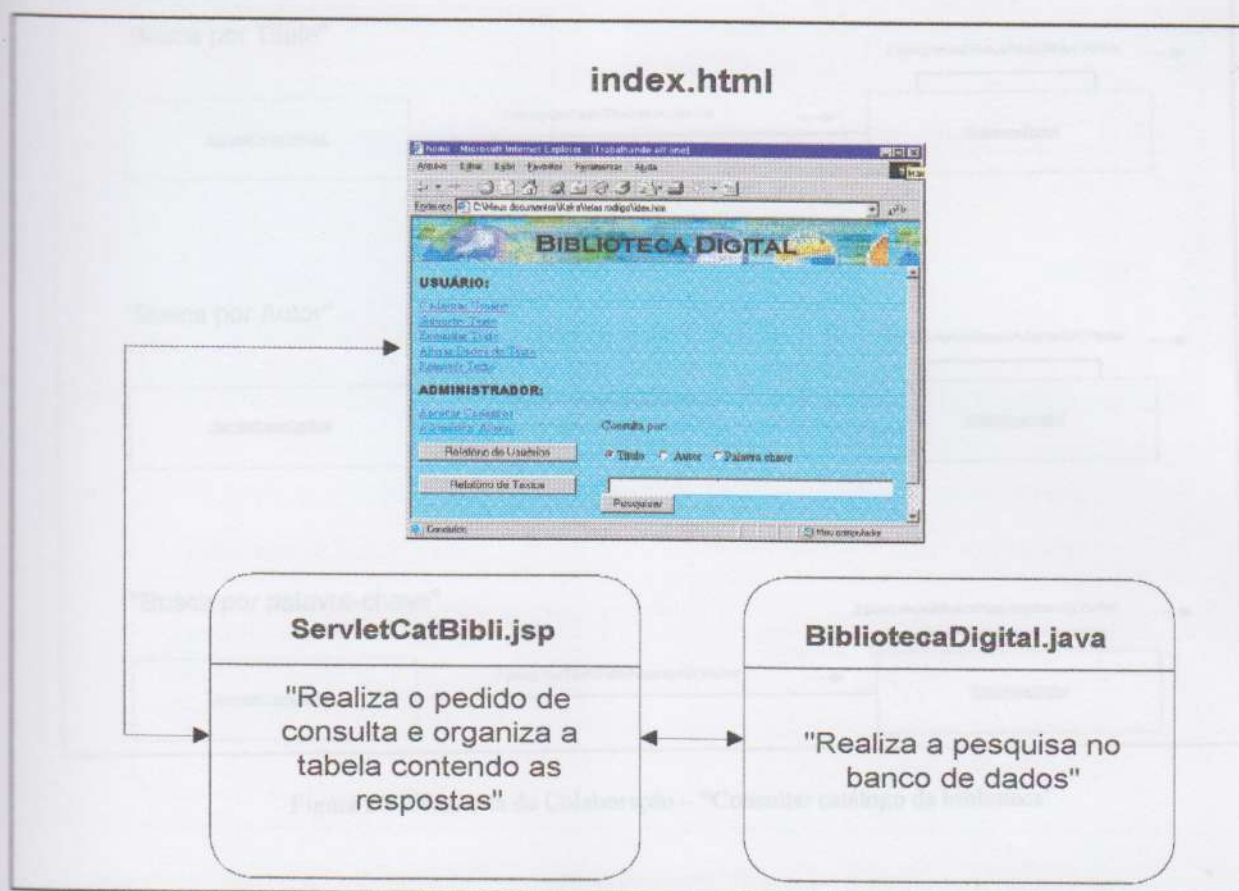


Figura 25. Estrutura da aplicação "consulta catálogo da biblioteca".

Este é o método que o usuário utiliza para realizar pesquisa no acervo da biblioteca. Para este, o usuário escolhe na página inicial da Biblioteca Digital, qual o tipo de pesquisa que ele deseja realizar. Existem três tipos de pesquisa, o usuário ou administrador pode executar buscas por título, autor ou palavra-chave. Após o click no botão Pesquisar, do formulário, o tipo de busca será feita de acordo com o campo preenchido.

Os dados da pesquisa são enviados do formulário para o ServletConsCatBibli. Neste servlet através do método específico, é requisitado à biblioteca o pedido de busca, de acordo com o parâmetro passado pelo usuário.

A biblioteca digital recebe os parâmetros do servlet e executa uma pesquisa de acordo com a pesquisa escolhida e retorna ao servlet o vetor locTexto contendo os possíveis respostas da pesquisa.

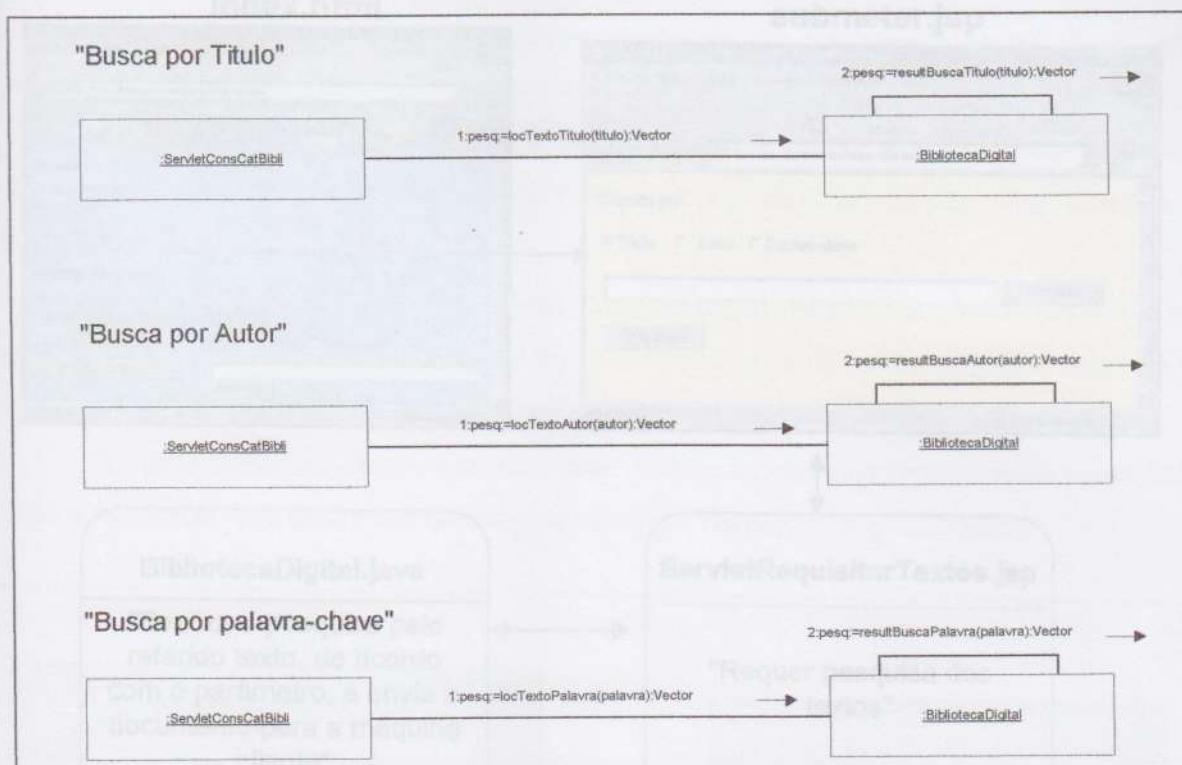


Figura 26. Diagrama de Colaboração – “Consultar catálogo da biblioteca”

4. Requisitar texto:

Para se Ter acesso aos textos do banco de dados, o usuário deve realizar uma pesquisa e fazer o download (baixar) do arquivo, para sua máquina. A pesquisa do texto é feita através do método de busca da biblioteca, que pode ser por título, autor ou palavra-chave, conforme descrito no método “Consultar catálogo da biblioteca”. A pesquisa exibirá um formulário contendo os possíveis textos encontrados e a frente deles um botão com a opção “Download”.

O click neste botão envia o código, que identifica o texto e que é recebido como dado da pesquisa, do texto para o `ServletRequisitarTexto`. Este servlet faz o pedido da url, onde se encontra o texto no banco, à biblioteca através do método `recuperarUrlTexto` passando como parâmetro o código do texto solicitado.

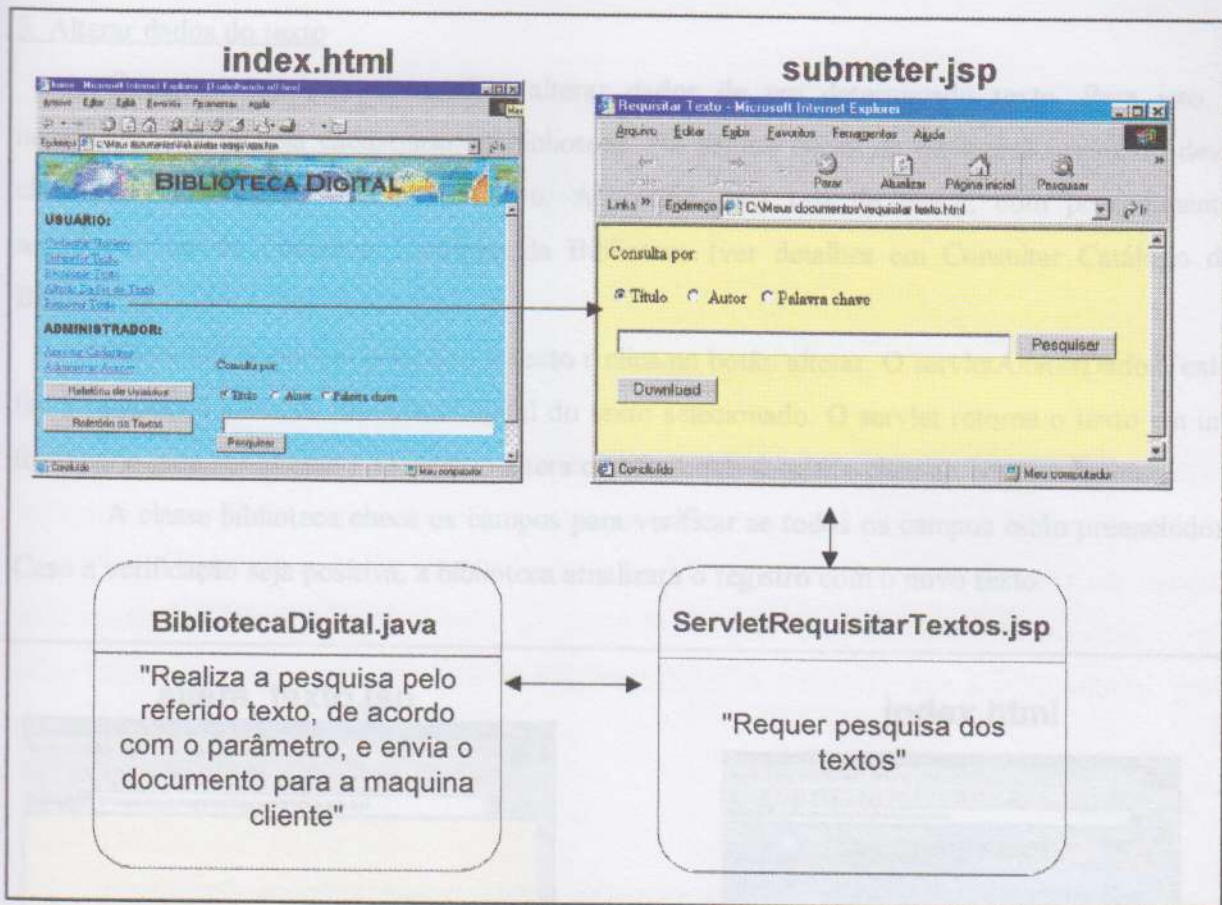


Figura 27. Estrutura da aplicação "Requisitar textos".

A biblioteca recebe o código do texto e realiza uma busca através do método `recuperaURLTextoBd` em seus arquivos, e retorna ao `ServletRequisitarTextos`, uma string contendo a url, endereço, de onde se encontra o texto no banco de dados. O sistema faz a transferência dos dados do bando de dados para a máquina do usuário.

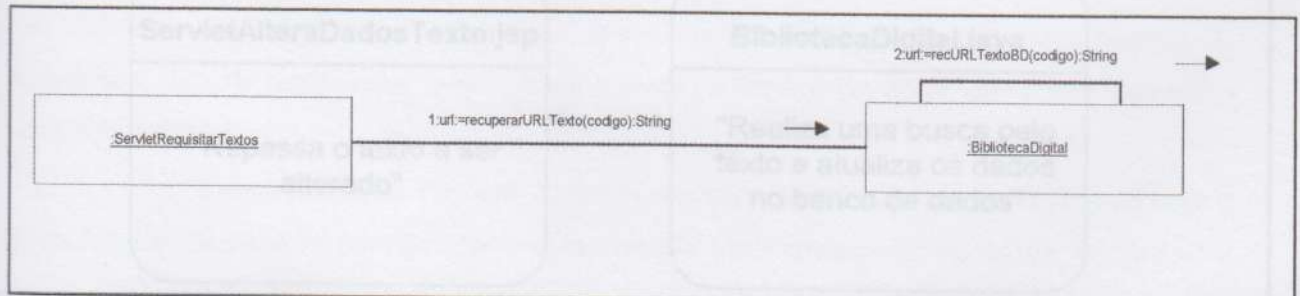


Figura.28. Diagrama de Colaboração – "Requisitar Textos"

5. Alterar dados do texto

O sistema permite ao usuário alterar dados de um determinado texto. Para isso é necessário que ele seja cadastrado na biblioteca. Na página inicial da biblioteca o usuário deve clicar na opção alterar dados do texto. Aparecerá uma tela de busca, com procedimento semelhante ao de Consultar Catálogo da Biblioteca (ver detalhes em Consultar Catálogo da Biblioteca).

Nesta tela o usuário seleciona o texto e clica no botão alterar. O servlet `AlterarDadosTexto` faz a requisição a classe `BibliotecaDigital` do texto selecionado. O servlet retorna o texto em um formulário para ser alterado. O usuário altera o campo que desejar e clica no botão submeter.

A classe biblioteca checa os campos para verificar se todos os campos estão preenchidos. Caso a verificação seja positiva, a biblioteca atualizará o registro com o novo texto.

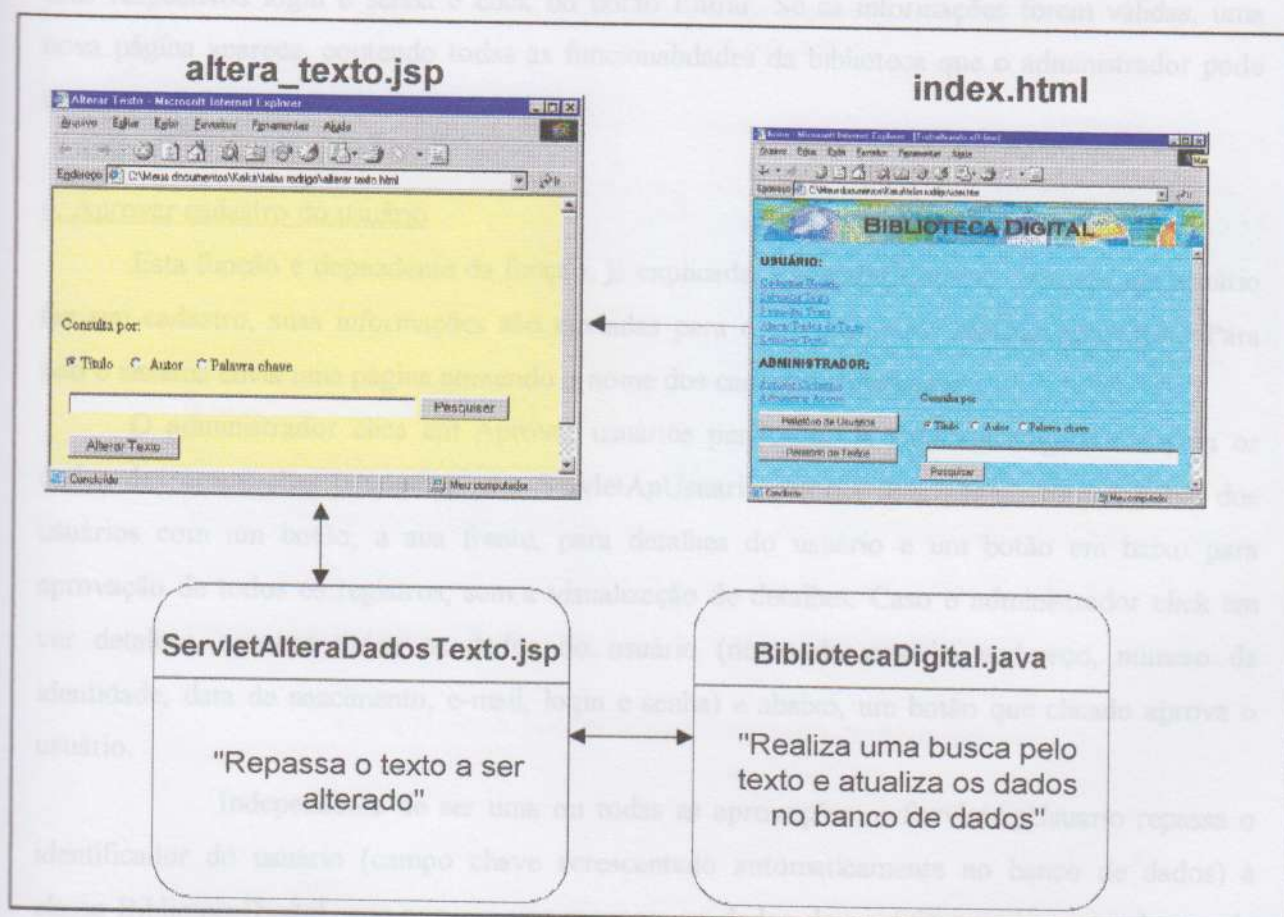


Figura 29. Estrutura da aplicação "Alterar dados do texto".

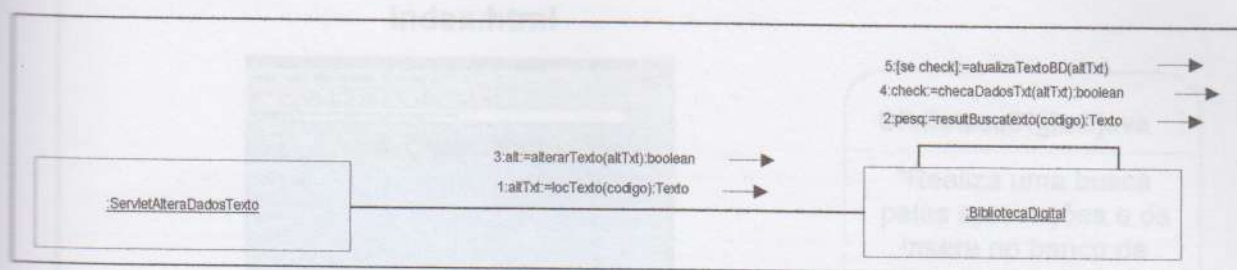


Fig. 30 Diagrama de Colaboração – “Alterar dados do Texto”

As funções descritas abaixo são de uso exclusivo do usuário administrador, nelas encontram-se as funções que administram e organizam o sistema da Biblioteca Digital. Para todas elas, é necessário que o administrador entre na página principal da biblioteca, insira nos campos seus respectivos login e senha e click no botão Entrar. Se as informações forem válidas, uma nova página aparece, contendo todas as funcionalidades da biblioteca que o administrador pode utilizar.

6. Aprovar cadastro do usuário

Esta função é dependente da função, já explicada, Cadastrar Usuário. Quando um usuário faz seu cadastro, suas informações são enviadas para o administrador realizar aprovação. Para isso o sistema envia uma página contendo o nome dos cadastros pendentes.

O administrador clica em Aprovar usuários pendentes. A BibliotecaDigital recupera os dados das aprovações pendentes, e o `ServletApUsuario` retorna uma página com o nome dos usuários com um botão, a sua frente, para detalhes do usuário e um botão em baixo para aprovação de todos os registros, sem a visualização de detalhes. Caso o administrador click em ver detalhes, aparece todos os dados do usuário (nome do usuário, endereço, número da identidade, data de nascimento, e-mail, login e senha) e abaixo, um botão que clicado aprova o usuário.

Independente de ser uma ou todas as aprovações, o `ServletApUsuario` repassa o identificador do usuário (campo chave acrescentado automaticamente no banco de dados) à classe BibliotecaDigital, que por sua vez recupera os dados do usuário e os insere no banco de dados.

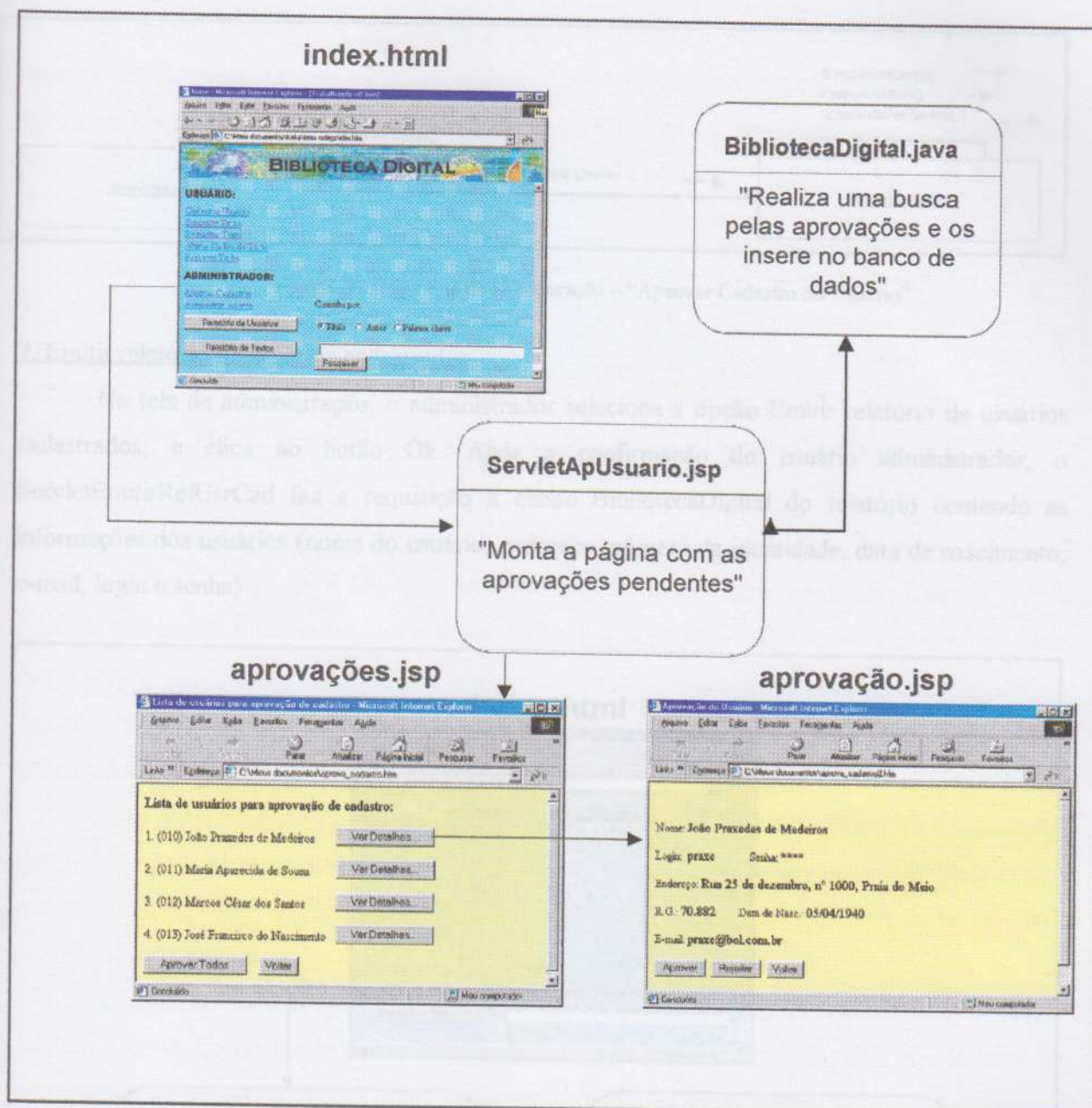


Figura 31. Estrutura da aplicação "Aprovar cadastro do usuário".

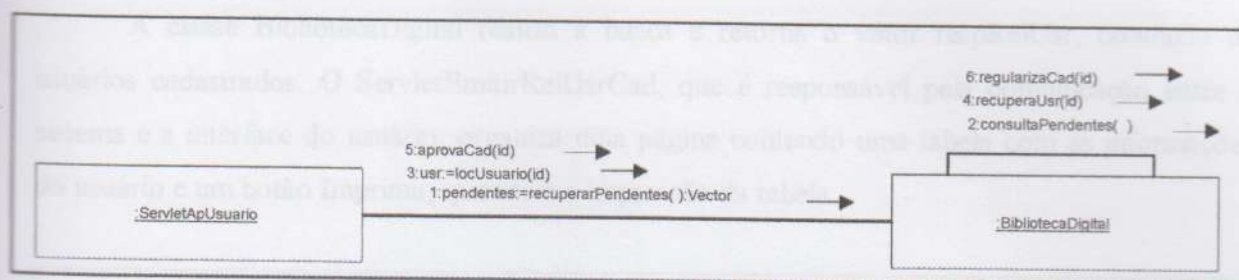


Figura. 32 Diagrama de Colaboração – “Aprovar Cadastro do Usuário”

7. Emitir relatório de usuários cadastrados

Na tela de administração, o administrador seleciona a opção Emitir relatório de usuários cadastrados, e clica no botão Ok. Após a confirmação do usuário administrador, o `ServletEmitirRelUsrCad` faz a requisição a classe `BibliotecaDigital` do relatório contendo as informações dos usuários (nome do usuário, endereço, número da identidade, data de nascimento, e-mail, login e senha).

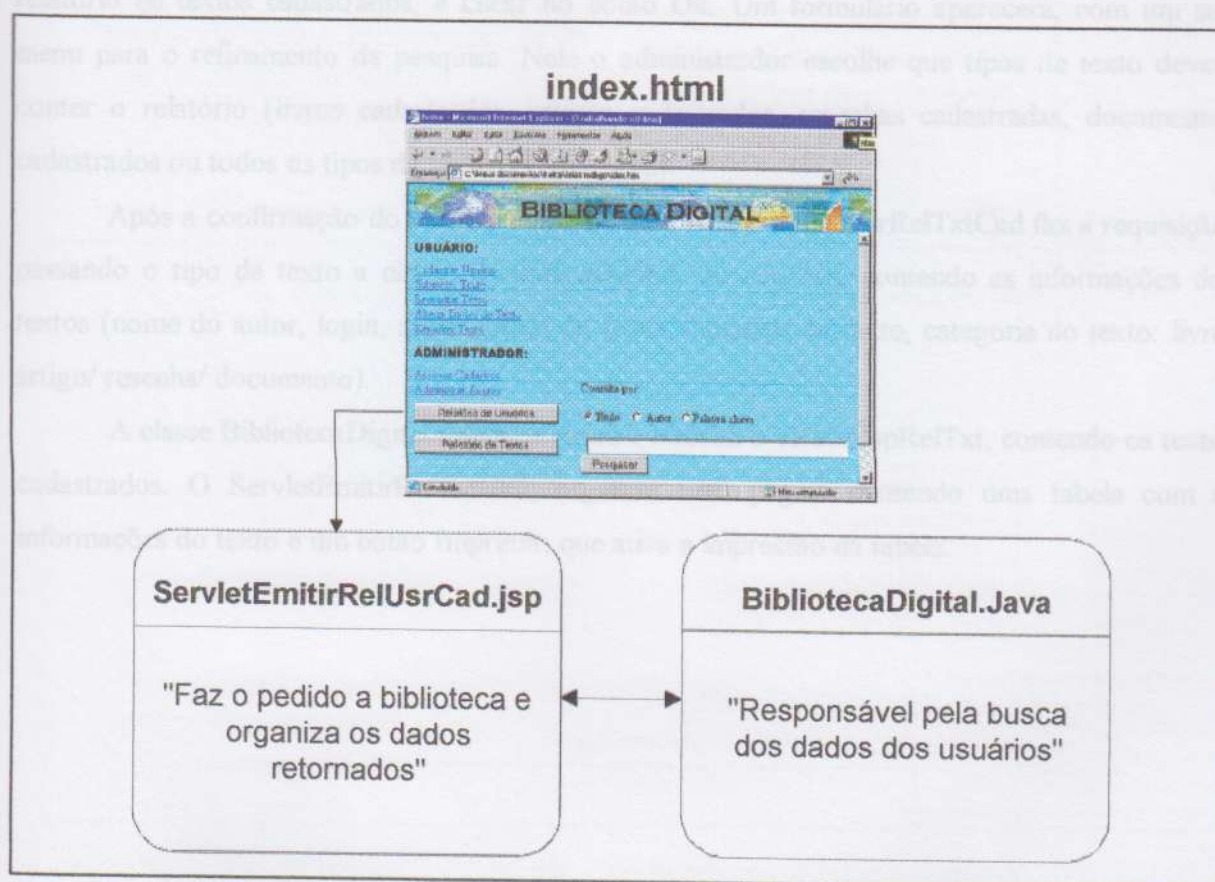


Figura 33. Estrutura da aplicação “Emitir relatório de usuários cadastrados”.

A classe BibliotecaDigital realiza a busca e retorna o vetor respRelUsr, contendo os usuários cadastrados. O ServletEmitirRelUsrCad, que é responsável pela comunicação entre o sistema e a interface do usuário, organiza uma página contendo uma tabela com as informações do usuário e um botão Imprimir, que ativa a impressão da tabela.

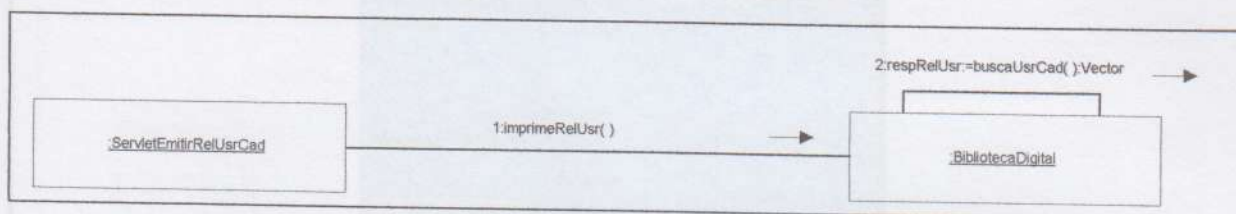


Fig. 34 Diagrama de Colaboração – “Emitir Relatório de Usuários Cadastrados”

8. Emitir relatório de textos cadastrados

Depois de acessada a tela do administrador, o usuário deve selecionar a opção Emitir relatório de textos cadastrados, e clicar no botão Ok. Um formulário aparecerá, com um sub menu para o refinamento da pesquisa. Nele o administrador escolhe que tipos de texto devem conter o relatório (livros cadastrados, artigos cadastrados, resenhas cadastradas, documentos cadastrados ou todos os tipos de textos cadastrados).

Após a confirmação do usuário administrador, o ServletEmitirRelTxtCad faz a requisição, passando o tipo de texto a classe BibliotecaDigital, do relatório contendo as informações dos textos (nome do autor, login, senha, título do texto, tamanho do texto, categoria do texto: livro/ artigo/ resenha/ documento).

A classe BibliotecaDigital realiza a busca e retorna o vetor respRelTxt, contendo os textos cadastrados. O ServletEmitirRelTxtCad, organiza uma página contendo uma tabela com as informações do texto e um botão Imprimir, que ativa a impressão da tabela.

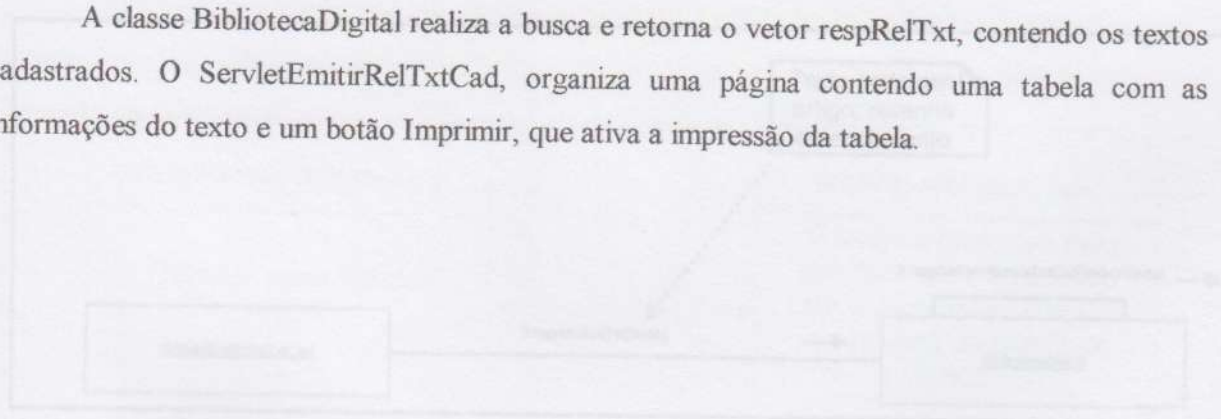


Fig. 36 Diagrama de Colaboração – “Emitir Relatório de Textos Cadastrados”

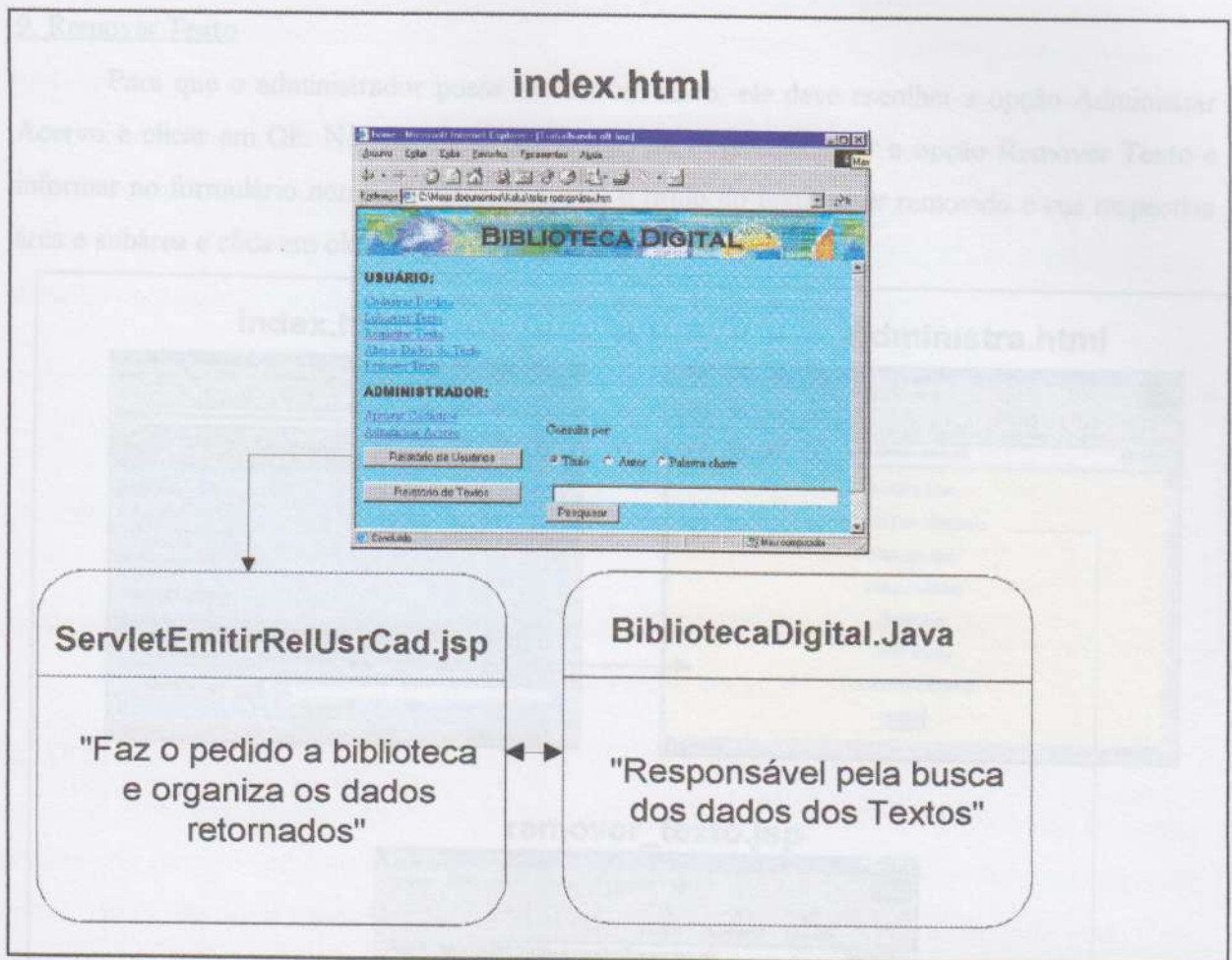


Figura 35. Estrutura da aplicação "Emitir relatório de textos cadastrados".

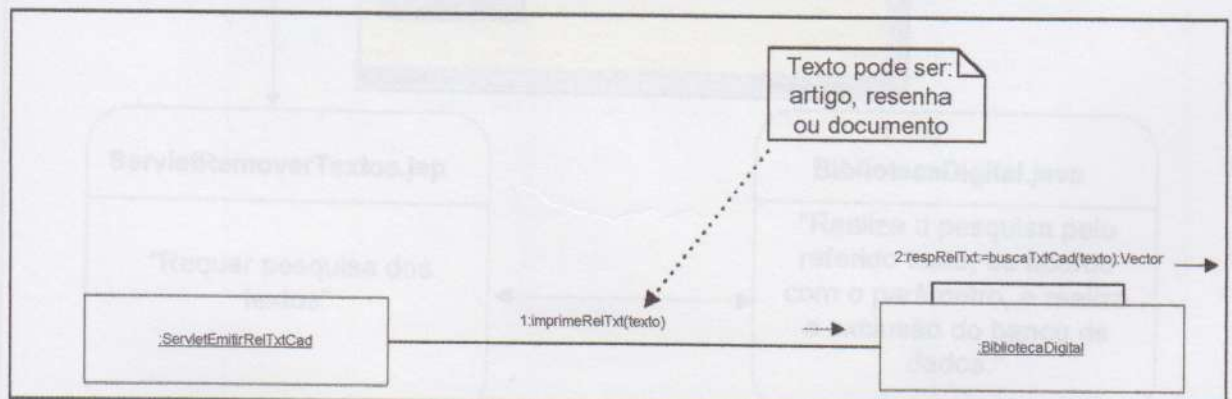


Fig. 36 Diagrama de Colaboração – "Emitir Relatório de Textos Cadastrados"

9. Remover Texto

Para que o administrador possa excluir um texto, ele deve escolher a opção Administrar Acervo e clicar em Ok. No sub menu que surge, ele deve selecionar a opção Remover Texto e informar no formulário nome do autor do texto, o título do texto a ser removido e sua respectiva área e subárea e clica em ok.

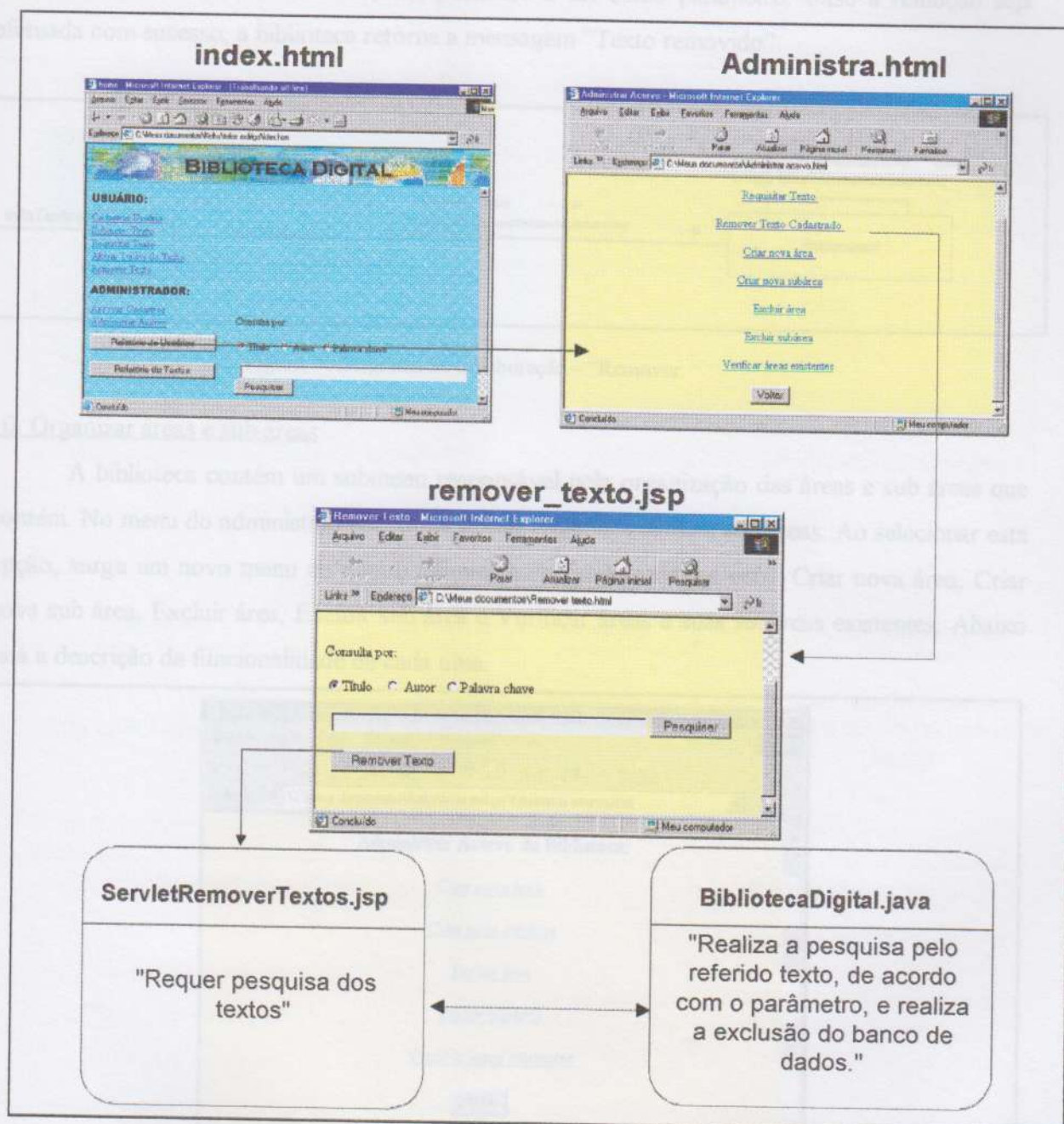


Figura 37. Estrutura da aplicação "Remover textos".

O `ServletRemoverTextos` recebe estas informações, e repassa para a classe `BibliotecaDigital` para ela validar os dados. Depois de encontrado o texto, o servlet mostra uma tela pedindo uma confirmação do administrador, para remoção do texto. Caso a resposta do usuário seja positiva, a Biblioteca fará a busca pela url do texto, e excluirá o registro do banco de dados, através do método `remove(url)`, passando a url como parâmetro. Caso a remoção seja efetuada com sucesso, a biblioteca retorna a mensagem "Texto removido".

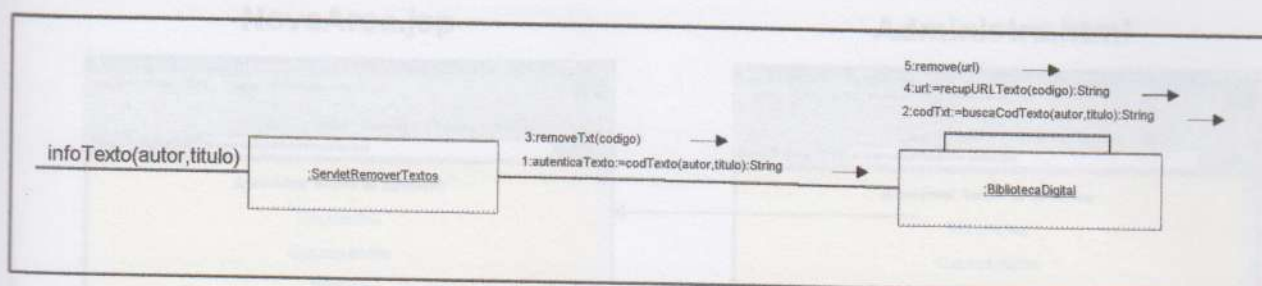


Figura. 38 Diagrama de Colaboração – “Remover

10. Organizar áreas e sub áreas

A biblioteca contém um submenu responsável pela organização das áreas e sub áreas que contém. No menu do administrador existe a opção Organizar áreas e sub áreas. Ao selecionar esta opção, surge um novo menu no qual o administrador pode escolher entre Criar nova área, Criar nova sub área, Excluir área, Excluir sub área e Verificar áreas e suas subáreas existentes. Abaixo está a descrição da funcionalidade de cada uma.

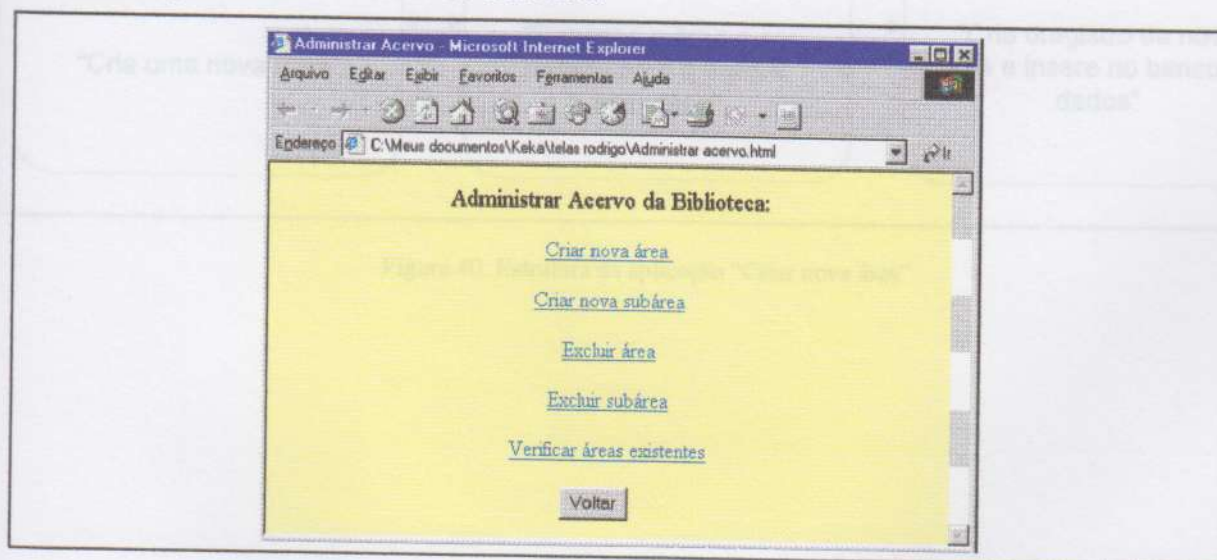


Figura. 39. Tela de menu Organizar Áreas

11. Criar nova área:

Ao clicar na opção Criar nova área, será requisitado num formulário o nome da nova área e os assuntos relacionados a ela. O servlet `CriaNovaArea` envia os dados a classe `Área`, para que ela crie uma nova área através do método `criar()`. Após a criação, a classe `Biblioteca Digital` checa os dados da nova área, cria um novo registro e insere a nova área no banco de dados.

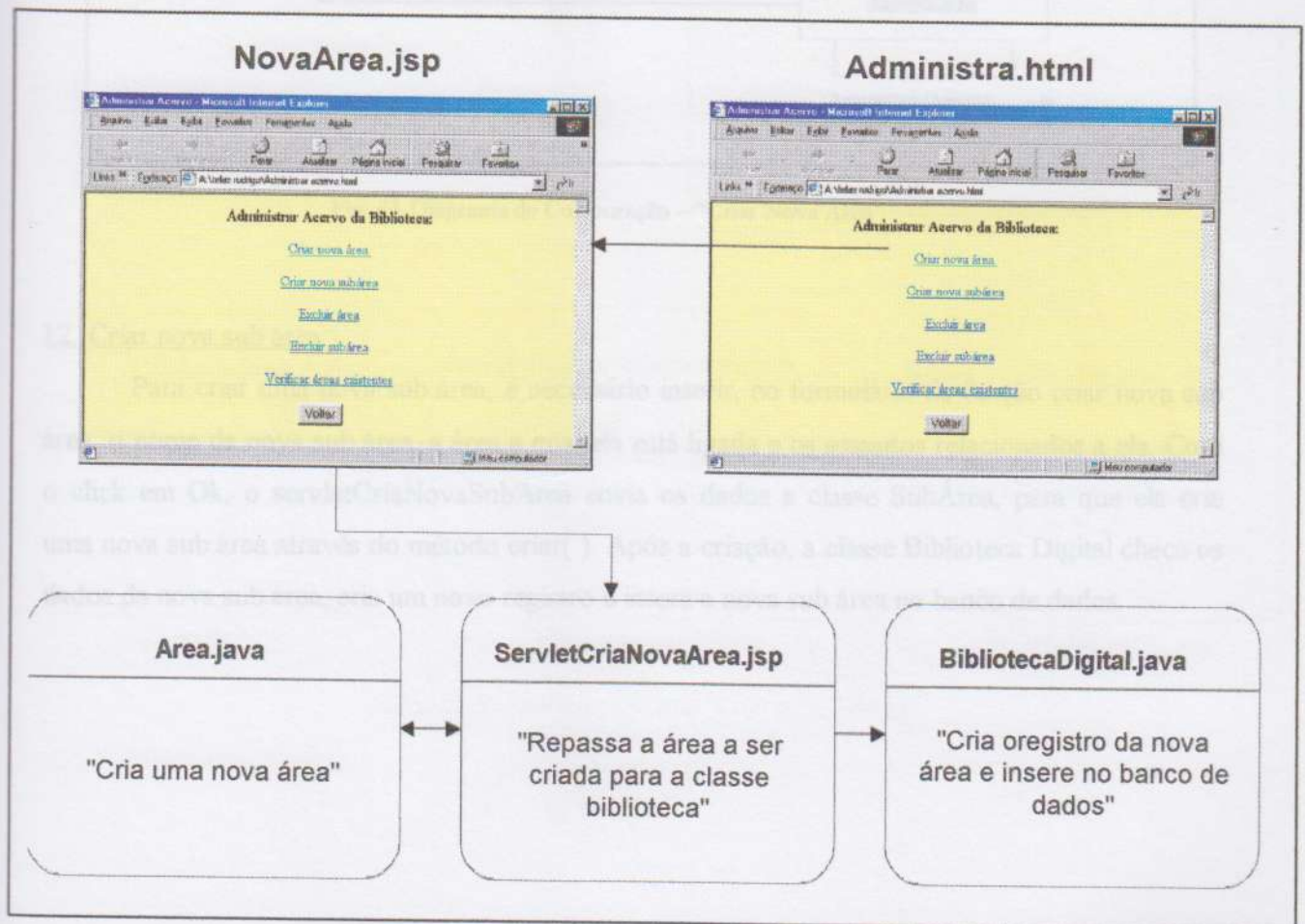


Figura 40. Estrutura da aplicação "Criar nova área".

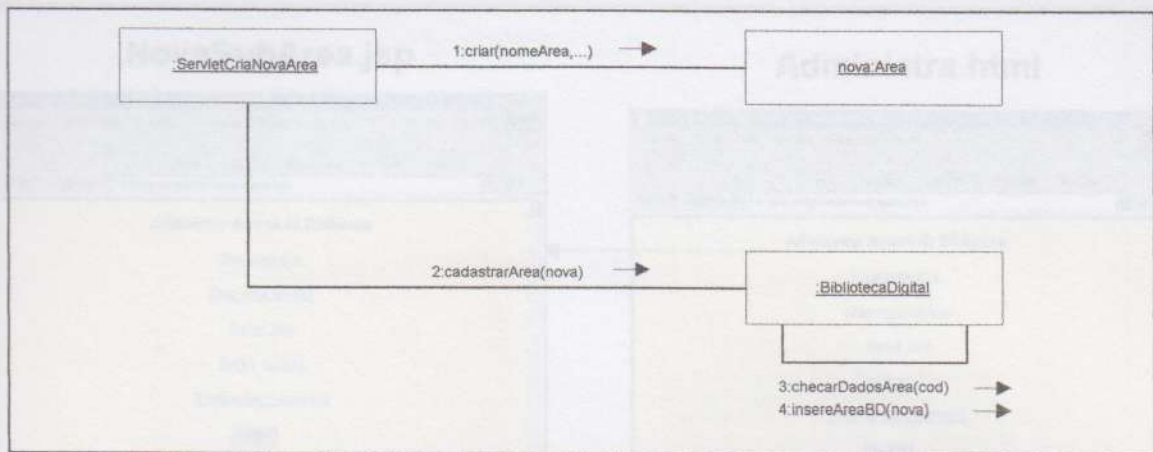


Fig. 41 Diagrama de Colaboração – “Criar Nova Área”

12. Criar nova sub área:

Para criar uma nova sub área, é necessário inserir, no formulário da função criar nova sub área, o nome da nova sub área, a área a qual ela está ligada e os assuntos relacionados a ela. Com o click em Ok, o `servletCriaNovaSubArea` envia os dados a classe `SubÁrea`, para que ela crie uma nova sub área através do método `criar( )`. Após a criação, a classe `Biblioteca Digital` checa os dados da nova sub área, cria um novo registro e insere a nova sub área no banco de dados.

Figura 42, Registro da aplicação “Criar nova Subárea”

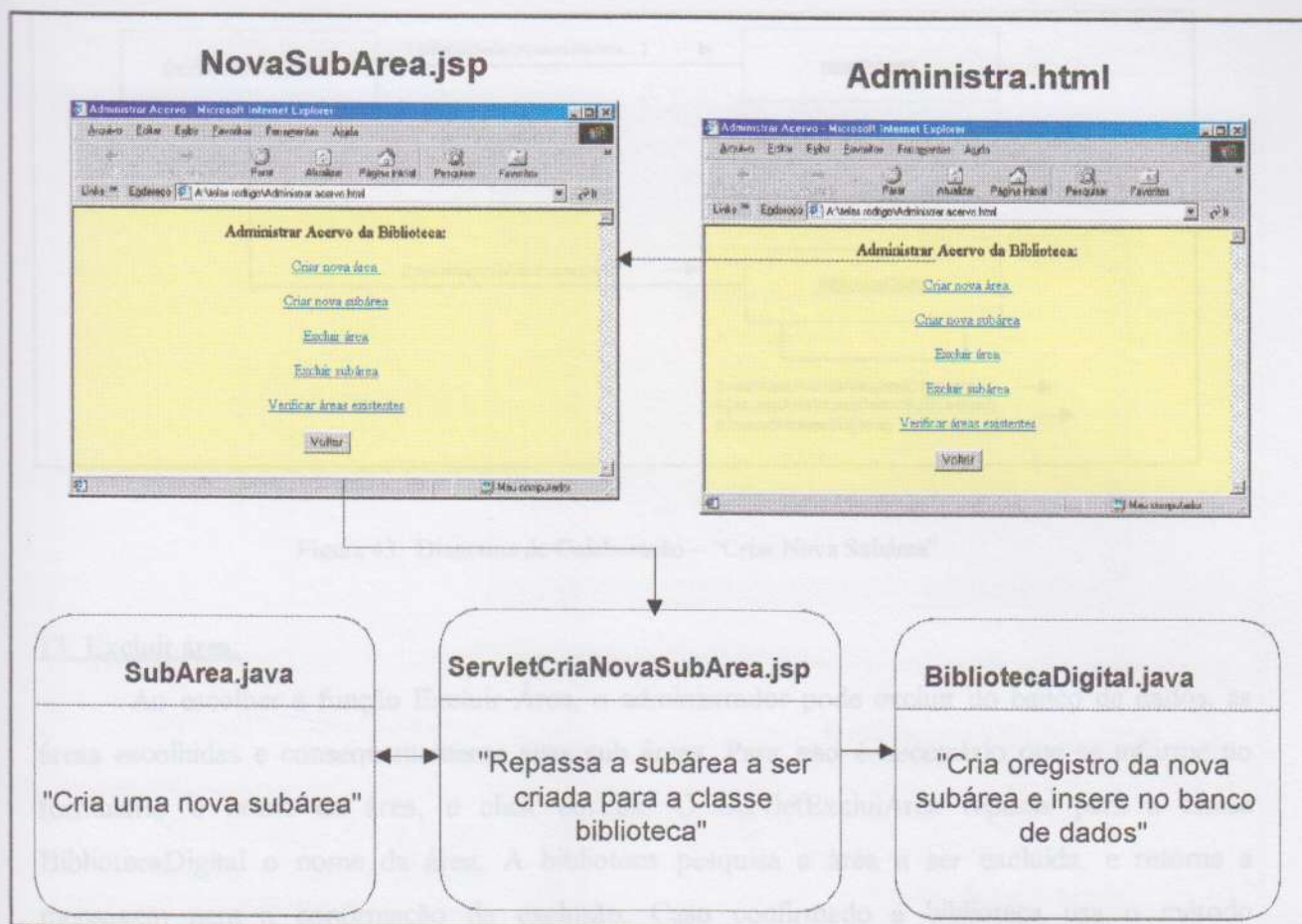


Figura 42. Estrutura da aplicação "Criar nova Subárea".

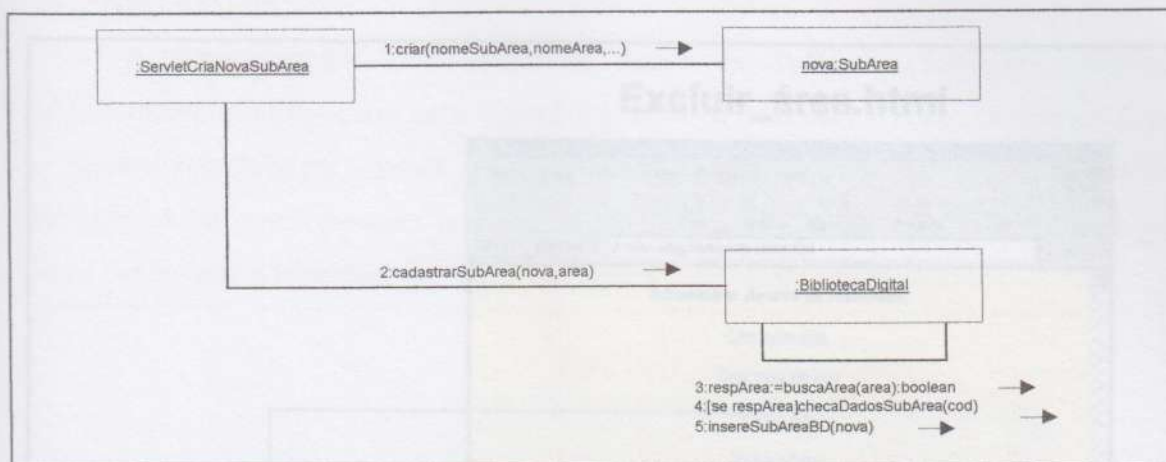


Figura 43. Diagrama de Colaboração – “Criar Nova Subárea”

13. Excluir área:

Ao escolher a função Excluir Área, o administrador pode excluir do banco de dados, as áreas escolhidas e conseqüentemente suas sub áreas. Para isso é necessário que se informe no formulário o nome da área, e click em ok. O ServletExcluiArea repassa para a classe BibliotecaDigital o nome da área. A biblioteca pesquisa a área a ser excluída, e retorna a mensagem para a confirmação da exclusão. Caso confirmado a biblioteca usa o método `removeArea()` para remove o registro do banco de dados.

Figura 44. Esquema da aplicação "Excluir área"



Figura 45. Diagrama de Colaboração – “Excluir Área”

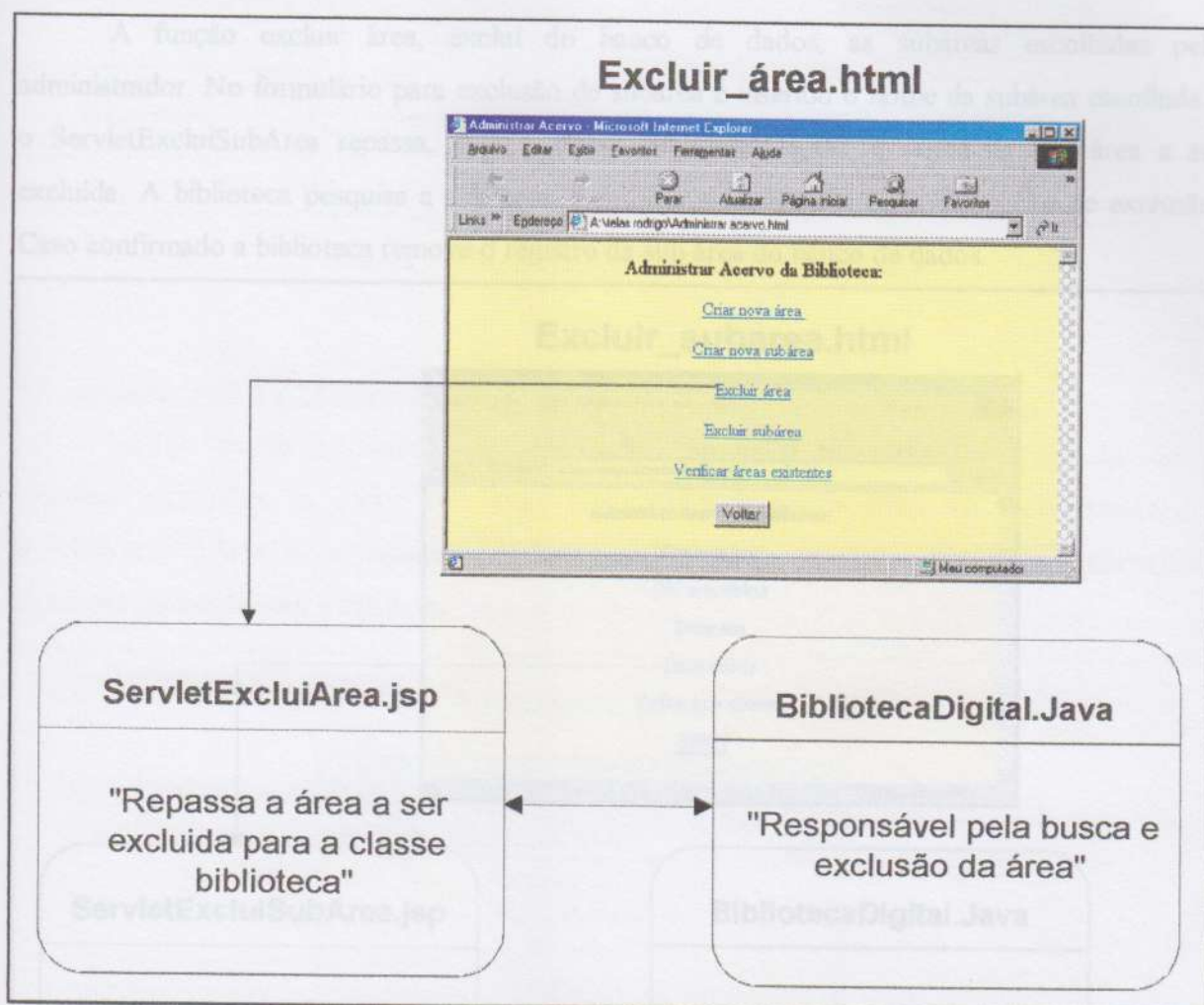


Figura 44. Estrutura da aplicação "Excluir área".

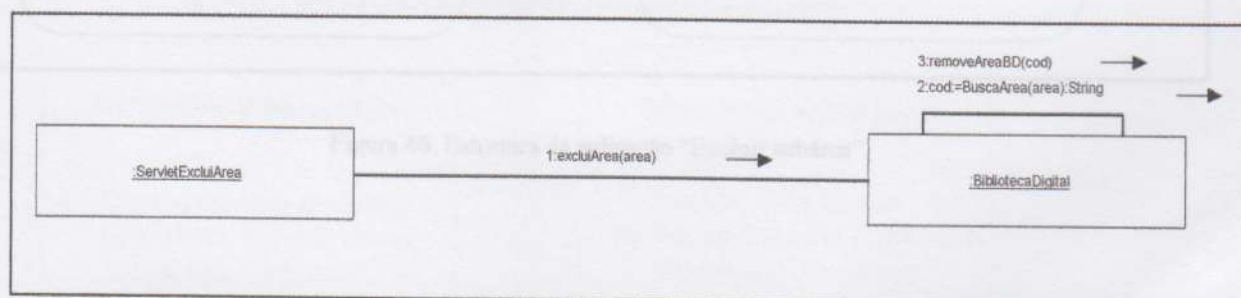


Figura. 45 Diagrama de Colaboração – "Excluir Área"

14. Excluir sub área:

A função excluir área, exclui do banco de dados, as subáreas escolhidas pelo administrador. No formulário para exclusão de subárea é inserido o nome da subárea escolhida e o ServletExcluiSubArea repassa, para a classe BibliotecaDigital, o nome da sub área a ser excluída. A biblioteca pesquisa a sub área, e retorna a mensagem de confirmação de exclusão. Caso confirmado a biblioteca remove o registro da sub área do banco de dados.

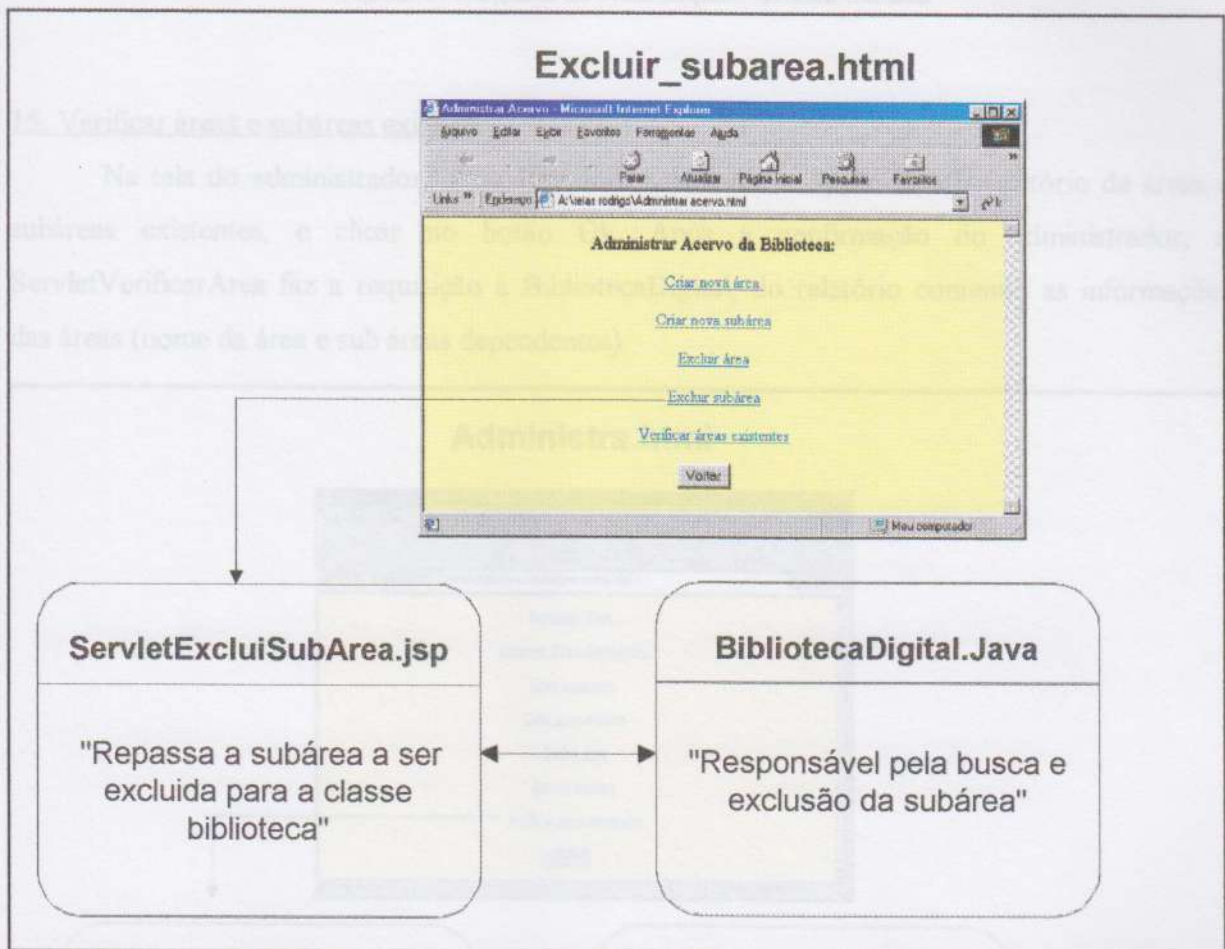


Figura 46. Estrutura da aplicação "Excluir subárea".

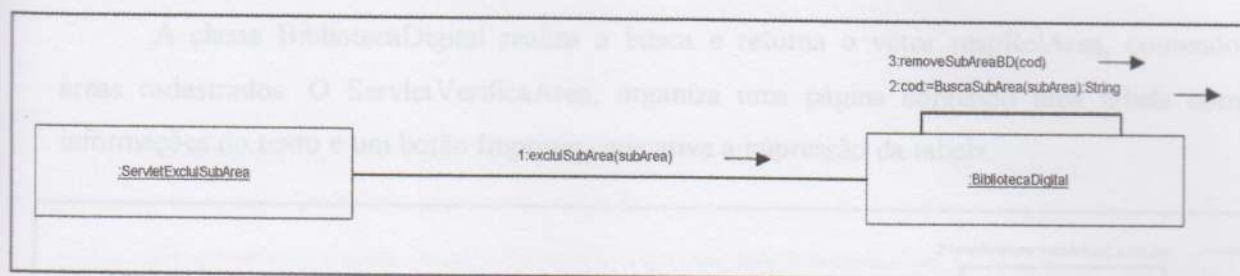


Figura. 47 Diagrama de Colaboração – “Excluir Subárea”

15. Verificar áreas e subáreas existentes:

Na tela do administrador, o usuário deve selecionar a opção Emitir relatório de áreas e subáreas existentes, e clicar no botão Ok. Após a confirmação do administrador, o `ServletVerificarArea` faz a requisição à `BibliotecaDigital`, do relatório contendo as informações das áreas (nome da área e sub áreas dependentes).

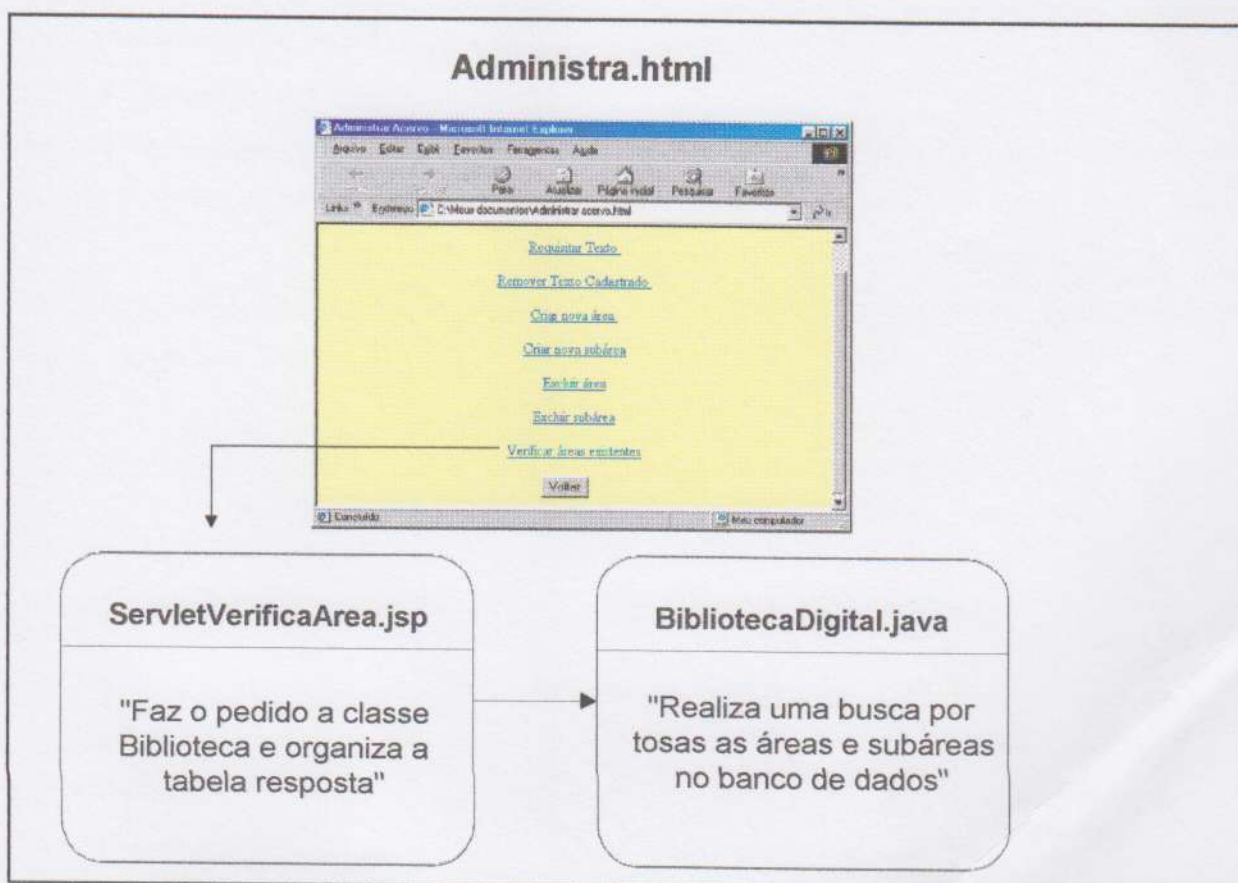


Figura 48. Estrutura da aplicação “Verificar áreas existentes”.

A classe BibliotecaDigital realiza a busca e retorna o vetor respRelArea, contendo as áreas cadastrados. O ServletVerificaArea, organiza uma página contendo uma tabela com as informações do texto e um botão Imprimir, que ativa a impressão da tabela.

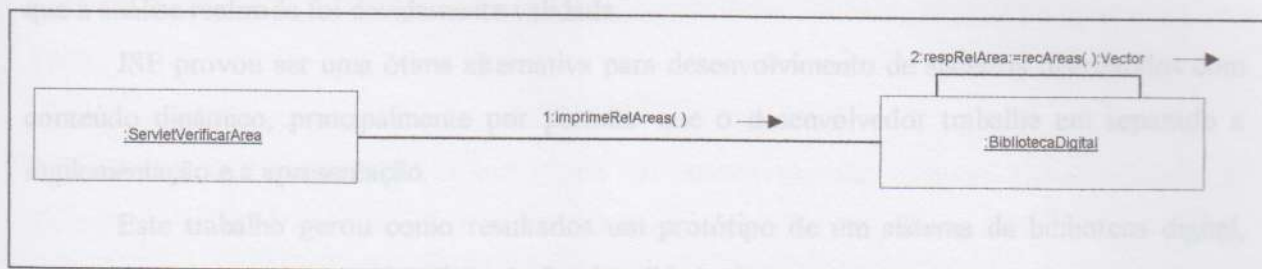


Figura. 49 Diagrama de Colaboração – “Verificar Áreas e Subáreas Existentes”

4. CONCLUSÃO

Ao chegarmos ao fim do trabalho, podemos concluir que este é perfeitamente viável visto que a análise realizada foi devidamente validada.

JSP provou ser uma ótima alternativa para desenvolvimento de sistemas distribuídos com conteúdo dinâmico, principalmente por permitir que o desenvolvedor trabalhe em separado a implementação e a apresentação.

Este trabalho gerou como resultados um protótipo de um sistema de biblioteca digital, utilizando JSP, bem como a descrição da funcionalidade da mesma.

Este trabalho apresenta como pontos fortes; ser uma iniciativa sem fins lucrativos, além de ser academicamente inédito no âmbito da FARN.

Como pontos fracos, o trabalho não apresentou a descrição completa de todas as suas funcionalidades bem como teste e validação destas.

Prevemos como trabalhos futuros:

- Descrição das demais funcionalidades;
- Implementação e validação destas funcionalidades.

DEITEL, H. M.; DEITEL, P. J. Java como programar. Trad. Edson Fomenselewicz. 2. ed. Porto Alegre: Bookman, 2001.

FERREIRA, José Roberto. A Biblioteca Digital. Revista USP, USP - Universidade de São Paulo, n. 35, 1999.

FIELDS, Duane K; KOLD, Mark A. Desenvolvendo na Web com Java Server Pages. Trad. Regina Freitas. Rio de Janeiro: Editora Ciência Moderna, 2000.

FURGERI, Sergio. Business to Business: Aprenda a desenvolver aplicações B2B. São Paulo: Erica, 2001.

BIBLIOGRAFIA

AFONSO, Marcelo. Vida Digital? Rets – Revista do Terceiro Setor, São Paulo, ano 2, n. 109, out. 2000. Disponível em: <http://www.rits.org.br/rets/re_editorial.ctm>. Acesso em: 17 mar 2002.

BROGDEN, Bill; MINNICK, Chris. **Guia do Desenvolvedor Java: Desenvolvendo E-Commerce com JAVA, XML e JSP**. Trad. Mário Moro Fecchio. São Paulo: Makron Books, 2002.

COELHO, Renato; PREVIDELLI, Fátima; DOBBIN, Leonardo. Direito Autoral na Internet. Disponível em: <<http://www.les.int.puc-rio.br/socinfo/dirautinter.htm>> Acesso em: 30 mar. 2002.

CUNHA, Murilo B. **Biblioteca Digital**: bibliografia internacional anotada. Disponível em: <<http://www.unicamp.br/bc/bibvirt/bibvirt3.htm>> Acesso em: 20 mar. 2002.

DATE, C. J. **Introdução a Sistema de Banco de Dados**. Trad. Vandemberg Dantas de Souza. 7 ed. Rio de Janeiro: Campus, 2000.

DEITEL, H. M.; DEITEL, P. J. **Java como programar**. Trad. Edson Furmankiewicz. 3.ed. Porto Alegre: Bookman, 2001.

FERREIRA, José Rincon. **A Biblioteca Digital**. Revista USP. USP – Universidade de São Paulo, n.35, 1989.

FIELDS, Duane K; KOLB, Mark A. **Desenvolvendo na Web com Java Server Pages**. Trad. Rejane Freitas. Rio de Janeiro: Editora Ciência Moderna, 2000.

FURGERI, Sergio. **Busines to Business: Aprenda a desenvolver aplicações B2B**. São Paulo: Érica, 2001.

GEARY, David M. **JavaServer Pages avançado**. Trad. Tereza Queiroz. Rio de Janeiro: Editora Ciência Moderna, 2002.

HAGGAR, Peter. Java: **Guia Prático de Programação**. Trad. Daniel Vieira. Rio de Janeiro: Campus, 2000.

HORSTMANN, Cay S.; CORNELL, Gary. **Core Java 2 vol.1 Fundamentos**. Trad. João E. N. Tortello. São Paulo: Makron Books, 2001.

RAMON, Fabio. **JDBC 2 Guia de consulta rápida**. São Paulo: Novatec Editora, 2001.

JUNIOR, Francisco B. **JSP: A Tecnologia Java na Internet**. São Paulo: Érica, 2002.

SOUKUP, Ron. **Desenvolvendo o Microsoft SQL Server 6.5**. Trad. Vitor Hugo da Paixão. Rio

LEMAY, Laura. **Aprenda em uma semana HTML 4**. Trad. João E. N. Tortello. Rio de Janeiro: Campos, 1998.

SILBERSCHATZ, Abraham; KORTM, Gary F.; SUDARSHAN, S. **Sistema de Banco de**

LEMAY, Laura. **Aprenda em 21 dias Java 2**. Trad. João E. N. Tortello. Rio de Janeiro: Campos, 1999.

MAYER, Roberto Carlos. **Otimizando a performance de Bancos de Dados Relacionais**. Rio de Janeiro: Axcel Books do Brasil Editora, 2001.

MELO, A. A. C.; AZEVEDO, C. L.; MELO, R. B. A. **Uma infra-estrutura de e-business para instituições de ensino superior**. In: I Congresso de Iniciação Científica da FARN, 2001, Natal. Anais do Congresso. Natal, RN, 2001. p. 30.

MELO, Ana Angélica Costa de. **Biblioteca Digital – Uma Análise Orientada a Objeto**. 2002. 56.f. Monografia – Sistemas de Informação. FARN – Faculdade Natalense para o Desenvolvimento do RN – Natal – RN, 2002.

MINK, Carlos; JUNIOR, Aníbal. **Microsoft Access 2000**. São Paulo: Makron Books, 2000.

NIEMEYER, Patrick; KNUDSEN, Jonathan. **Aprendendo Java**. Trad. Daniel Vieira. Rio de Janeiro: Campus, 2000.

PRATES, Rubens; SILVEIRA, Marcelo. **HTML 4 Guia de Consulta Rápida**. São Paulo: Novatec Editora, 2001.

RAMON, Fabio. **Java Guia de consulta rápida**. São Paulo: Novatec Editora, 2000.

RAMON, Fabio. **JDBC 2 Guia de consulta rápida**. São Paulo: Novatec Editora, 2001.

SOUKUP, Ron. **Desenvolvendo o Microsoft SQL Server 6.5**. Trad. Vitor Hugo da Paixão. Rio de Janeiro: Campus, 1998.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de Banco de Dados**. São Paulo: Makron Books, 1999.

ANEXO I – CÓDIGOS DE PÁGINAS HTML

Descrição do código HTML da página index.htm (composta pelos frames: menu.htm, cabeçalho.htm e tela_principal.htm) e da página cadastro.htm.

Código HTML do frame cabeçalho.htm:

```
<html>
<head>
<meta http-equiv="Content-Language" content="pt-br">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>Biblioteca Digital</title>
<base target="conteudo">
<meta name="Microsoft Border" content="none">
</head>
<body background="fig_cabeçalho.jpg" bgproperties="fixed">
<p align="center"><b><sup><font face="Copperplate Gothic Bold" size="7"
color="#000000">Biblioteca
Digital</font></sup></b></p>
</body>
</html>
```

Código HTML do frame menu.htm:

```

<html>
<head>
<meta http-equiv="Content-Language" content="pt-br">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>menu</title>
<base target="principal">
<style fprolloverstyle>A:hover {color: #FF0000; font-weight: bold}
</style>
<meta name="Microsoft Border" content="none">
</head>
<body bgproperties="fixed" bgcolor="#99CCFF">
<p><font face="Arial Black">USUÁRIO:</font></p>
<p style="line-height: 10%"><a href="CadUsuario.htm">Cadastrar
Usuário</a></p>
<p style="line-height: 10%"><a href="Form.Submeter.htm">Submeter
Texto</a></p>
<p style="line-height: 10%"><a href="requisitar%20texto.html">Requisitar Texto</a></p>
<p style="line-height: 10%"><a href="alterar%20texto.html">Alterar Dados do Texto</a></p>
<p style="line-height: 10%"><a href="Remover%20texto.html">Remover Texto</a></p>
<p><font face="Arial Black">ADMINISTRADOR:</font></p>
<p style="line-height: 10%"><a href="aprova_cadastro.htm">
Aprovar Cadastros</a></p>
<p style="line-height: 10%"><a href="Administrar%20acervo.html">
Administrar Acervo</a></p>
<form method="POST" action="--WEBBOT-SELF--" onSubmit="">
<!--webbot bot="SaveResults" startspan
U-File="C:\Meus documentos\Keka\telas rodrigo\_private\form_results.txt"

```

```
S-Format="TEXT/CSV" S-Label-Fields="TRUE" --><input TYPE="hidden" NAME="VTI-
GROUP" VALUE="0"><!--webbot
```

```
bot="SaveResults" endspar -->
```

```
<p style="line-height: 10%"><input type="button" value="Relatório de Usuários"
name="B3"></p>
```

```
<p style="line-height: 10%"><input type="button" value=" Relatório de Textos "
name="B2"></p>
```

```
</form>
```

```
</body>
```

```
</html>
```


ANEXO II – CÓDIGOS JSP COMPONENTES JAVA

Descrição do código JSP do ServletCadastro.jsp

```

<%@page contentType="text/html"%>
<%@page import="BDBiblioteca.*"%>
<html>
<head><title>Servlet para cadastro do usuário</title></head>
<body>

<jsp:useBean id="conexao" class="BDBiblioteca.BeanConexao" />
<CENTER><H2>
    <% String nome = request.getParameter("nome");
        if (nome == null)%> Campo nao preenchido <%;
    String login = request.getParameter("login");
        if (login == null)%> Campo nao preenchido <%;
    String senha = request.getParameter("senha");
        if (senha == null)%> Campo nao preenchido <%;
    String end = request.getParameter("end");
        if (end == null)%> Campo nao preenchido <%;
    String rg = request.getParameter("rg");
        if (rg == null)%> Campo nao preenchido <%;
    String data_nasc = request.getParameter("data_nasc");
        if (data_nasc == null)%> Campo nao preenchido <%;
    String e-mail = request.getParameter("e-mail");
        if (e-mail == null)%> Campo nao preenchido <%;
    String textos = request.getParameter("textos");
        if (textos == null)%> Campo nao preenchido <%;

    <% String sql = "INSERT INTO Usuario (nome_usr, login, senha, end, rg, data_nasc, e-mail,
    textos) VALUES ('" + nome + "','" + login + "','" + senha + "','" + end + "','" + rg + "','" +
    data_nasc + "','" + e-mail + "','" + textos + "')"; %>

    <% int resultado = conexao.executeUpdate(sql);
        if (resultado == 1) { %>
            Cadastro submetido a aprovação pelo administrador
        <% } else { %>
            Cadastro não aprovado
        <% } %>
    </H2></CENTER>
</body>
</html>

```

ANEXO III – CÓDIGOS DOS COMPONENTES JAVA

Descrição do Bean Conexão.Java

```

package BDBiblioteca;

import java.sql.*;
import javax.servlet.http.*;

public class BeanConexao implements HttpSessionBindingListener {
    private Connection conexao;
    private Statement instrucao;

    private static final String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
    private static final String url = "jdbc:odbc:biblioteca_digital";
    private static final String login = "anonymous";
    private static final String senha = "guest";

    public BeanConexao () {
        try {
            Class.forName(driver);
            conexao = DriverManager.getConnection(url,login,senha);
            instrucao = conexao.createStatement();
        } catch (ClassNotFoundException e) {
            System.err.println("Driver ODBC nao disponivel!");
            conexao = null;
        } catch (SQLException ex) {
            System.err.println("Driver ODBC nao carregado!");
            conexao = null;
        }
    }

    public Connection getConnection() {
        return conexao;
    }

    public void commit() throws SQLException {
        conexao.commit();
    }

    public void rollback() throws SQLException {
        conexao.rollback();
    }

    public void setAutoCommit(boolean autoCommit) throws SQLException {
        conexao.setAutoCommit(autoCommit);
    }
}

```

```

public ResultSet executeQuery(String sql) throws SQLException {
    return instrucao.executeQuery(sql);
}

public int executeUpdate(String sql) throws SQLException {
    return instrucao.executeUpdate(sql);
}

protected void finalize() {
    try {
        conexao.close();
    } catch (SQLException ex) {
    }
}

public void valueBound(javax.servlet.http.HttpSessionBindingEvent
httpSessionBindingEvent) {
    try {
        conexao.close();
    } catch (SQLException ex) {
    } finally {
        conexao = null;
    }
}

public void valueUnbound(javax.servlet.http.HttpSessionBindingEvent
httpSessionBindingEvent) {
    System.err.println("BeanConexao: no metodo valueBound");
    try {
        if ((conexao == null) || (conexao.isClosed())) {
            conexao = DriverManager.getConnection(url, login, senha);
            instrucao = conexao.createStatement();
        }
    } catch (SQLException ex) {
        conexao = null;
    }
}
}

```

Descrição da classe Acesso.Java

```
package BDBiblioteca;
import java.sql.*;
import java.util.*;

public class Acesso implements java.io.Serializable {
    public Vector transforma(ResultSet resultado) {
        ResultSetMetaData rsMetaDados;
        Vector tabela = new Vector();
        Vector linha = new Vector();
        try {
            rsMetaDados = resultado.getMetaData();
            while (resultado.next()) {
                linha = new Vector();
                for (int i = 1; i <= rsMetaDados.getColumnCount(); i++) {
                    switch (rsMetaDados.getColumnType(i)) {
                        case Types.VARCHAR :
                            linha.addElement(resultado.getString(i));
                            break;
                        case Types.INTEGER :
                            linha.addElement(new
                                Long(resultado.getLong(i)));
                            break;
                    }
                }
                tabela.addElement(linha);
            }
        } catch (SQLException sqlex) {
            sqlex.printStackTrace();
        }
        return tabela;
    }
}
```

ANEXO IV – TELAS DO SISTEMA BIBLIOTECA DIGITAL

Index.htm

home - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

BIBLIOTECA DIGITAL

USUÁRIO:

- [Cadastrar Usuário](#)
- [Submeter Texto](#)
- [Requisitar Texto](#)
- [Alterar Dados do Texto](#)
- [Remover Texto](#)

ADMINISTRADOR:

- [Aprovar Cadastros](#)
- [Administrar Acervo](#)

Relatório de Usuários

Relatório de Textos

Consulta por:

☒ Título ☐ Autor ☐ Palavra chave

Pesquisar

Concluído

Meu computador

cadastro.htm

The image shows a screenshot of a web browser window titled "Cadastrar Usuário - Microsoft Internet Explorer". The browser's menu bar includes "Arquivo", "Editar", "Exibir", "Favoritos", "Ferramentas", and "Ajuda". The main content area has a yellow background and contains a registration form with the following fields and labels:

- Nome: [text input]
- Login: [text input] Senha: [text input]
- Endereço: [text input]
- R.G.: [text input] Data de Nasc.: [text input]
- E-mail: [text input]

At the bottom of the form are two buttons: "Enviar" and "Limpar Campos". The browser's status bar at the bottom shows "Concluído" on the left and "Meu computador" on the right.

cadastro.htm

Submeter Textos ao Repositório - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Nome do Autor:

Login: Senha:

Título do texto:

Tamanho do texto:

Categoria: ☒ Artigo ☐ Livro ☐ Resenha

Texto:

Concluído

Meu computador