

## **WEBSITE PARA ACOMPANHAMENTO DA ANÁLISE DE PROJETOS DE COMBATE A INCÊNDIO**

**Dr. George SILVA; Dr. Gilbert SILVA; Misael BARRETO;  
Rafael SILVA;**

Centro Federal de Educação Tecnológica do Rio Grande do Norte – CEFET-RN, Av. Sen. Salgado Filho, 1559,  
Tirol, Natal-RN, Fone/Fax: (84) 4005-2600 / 4005-2637, e-mail: george@cefetrn.br, gilbert@cefetrn.br

### **RESUMO**

Este trabalho de pesquisa e desenvolvimento apresenta um Website para o Acompanhamento da Análise de Projetos de Combate a Incêndio desenvolvido por alunos e professores-orientadores do Núcleo de Desenvolvimento de Softwares do Centro Federal de Educação Tecnológica do Rio Grande do Norte (CEFET-RN) junto ao Corpo de Bombeiros Militar do Rio Grande do Norte (CBMRN). O objetivo deste website é fornecer ao proprietário do imóvel ou engenheiro responsável um mecanismo de fácil acesso para o acompanhamento da análise do respectivo projeto, análise essa realizada pelo Serviço Técnico de Engenharia (SERTEN), órgão do CBMRN. Através desse acompanhamento, os interessados poderão ficar cientes se determinado projeto já está sendo analisado (em tramite), se está parado por apresentar alguma irregularidade (não-conformidade), que irregularidade é essa e como corrigi-la, entre outras informações. Para isso se faz necessário apenas ter acesso a um computador conectado à Internet, um browser (navegador) e algumas informações relativas ao projeto. Desta forma, espera-se tornar mais ágil o processo de aprovação dos projetos de combate a incêndio e, conseqüentemente, causar um desafogamento no atendimento ao público, fazendo com que o CMBRN mantenha a máxima eficiência e eficácia em suas atribuições constitucionais.

**Palavras-chave:** Desenvolvimento de Software, Projeto de Combate a Incêndio, Corpo de Bombeiros, Internet, Java, JSF, JPA.

## 1. INTRODUÇÃO

Fundado oficialmente em 21 de setembro de 1955, o Corpo de Bombeiros do Estado do RN (CBMRN) é a corporação responsável pelas ações de defesa civil no combate a incêndios (urbanos e florestais) e calamidades públicas, através da realização de buscas e salvamentos, atuando também em atendimentos de socorro pré-hospitalar em via pública, com ambulâncias e motocicletas. Dentre as atividades realizadas pela corporação, destacam-se às ações prevenção de acidentes, que tem por objetivo evitar com que sinistros ocorram, como por exemplo, incêndios e desabamentos.

Todas as edificações do estado do RN, que desenvolvam atividades comerciais ou industriais, incluindo ainda edifícios residenciais, necessitam apresentar ao Corpo de Bombeiros um projeto de combate a incêndio. Neste projeto, baseado em normas estaduais e federais de segurança e prevenção contra incêndio e pânico, são dimensionados equipamentos de prevenção e listados todos os pré-requisitos de segurança necessários ao estabelecimento com o intuito de garantir os meios necessários ao combate a incêndio, evitar ou minimizar a propagação do fogo, facilitar as ações de socorro e assegurar a evacuação segura dos ocupantes das edificações.

No Rio Grande do Norte, os projetos de combate a incêndio são apresentados ao Serviço Técnico de Engenharia (SERTEN) do Departamento de Engenharia Operacional do Corpo de Bombeiros Militar do estado (CBMRN) pelos proprietários das edificações ou pelos engenheiros projetistas para que uma avaliação técnica do mesmo seja realizada. Um projeto para ser considerado seguro tem de ter sido avaliado e aprovado pelo SERTEN, caso contrário, a obra pode ser embargada ou interditada.

Atualmente, para dar entrada na análise do projeto de combate a incêndio de uma edificação, se faz necessário o comparecimento ao Corpo de Bombeiros Militar do RN, com sede em Natal, ou em uma das unidades do interior, Caicó ou Mossoró, de posse da seguinte documentação:

- Preenchimento da guia de entrada do projeto;
- Projeto arquitetônico;
- Projeto de proteção contra incêndio;
- ART do Engenheiro Civil ou de Segurança responsável pelo projeto;
- Memorial descritivo de construção (2 vias);
- Memorial descritivo de proteção contra incêndio (2 vias);
- Guia com comprovante de recolhimento da taxa.

Feito isto, o projeto está apto a ser analisado. O processo de análise pode resultar na aprovação do projeto ou na requisição de ajustes caso alguma não-conformidade (irregularidade) seja encontrada. Neste último caso, o projeto é devolvido ao engenheiro projetista ou ao proprietário da edificação para a realização de tais ajustes.

Visando fornecer ao proprietário do imóvel ou engenheiro responsável um mecanismo de fácil acesso para o acompanhamento da análise do respectivo projeto, está sendo desenvolvido um website. Através desse website, os interessados poderão ficar cientes se determinado projeto já está sendo analisado (em tramite), se apresenta alguma irregularidade (não-conformidade), que irregularidade é essa e como corrigi-la, entre outras informações.

## 2. FUNDAMENTACAO TEÓRICA

O website em desenvolvimento é parte integrante de uma solução computacional integrada realizada para o Departamento de Engenharia Operacional do CBMRN pelo Núcleo de Desenvolvimento de Softwares do CEFET-RN. O sistema computacional em desenvolvimento utiliza várias tecnologias nas áreas de projeto e desenvolvimento de sistema de informação. Algumas tecnologias utilizadas na realização do referido website serão apresentadas a seguir.

## 2.1. Arquitetura MVC

A arquitetura MVC (ver Figura 1) é um padrão de arquitetura de software. Sua finalidade é separar os dados (lógica de negócios) da interface do usuário. Dentro desse padrão, a aplicação é dividida em três camadas básicas:

- Visão: Trata-se da interface da aplicação, através do qual o usuário interage com o Sistema.
- Controle: É onde serão processadas todas as requisições feitas através da interface (*Visão*). O *Controle* acessa o *Modelo* a fim de obter determinadas informações. Toda lógica da aplicação - validações, atribuições, etc - é feita no *Controle*.
- Modelo: Representa os dados da aplicação e as regras do negócio que governam o acesso e a modificação dos dados. O *Modelo* mantém o estado persistente do negócio e fornece ao *Controle* a capacidade de acessar as funcionalidades da aplicação encapsuladas pelo próprio *Modelo*.

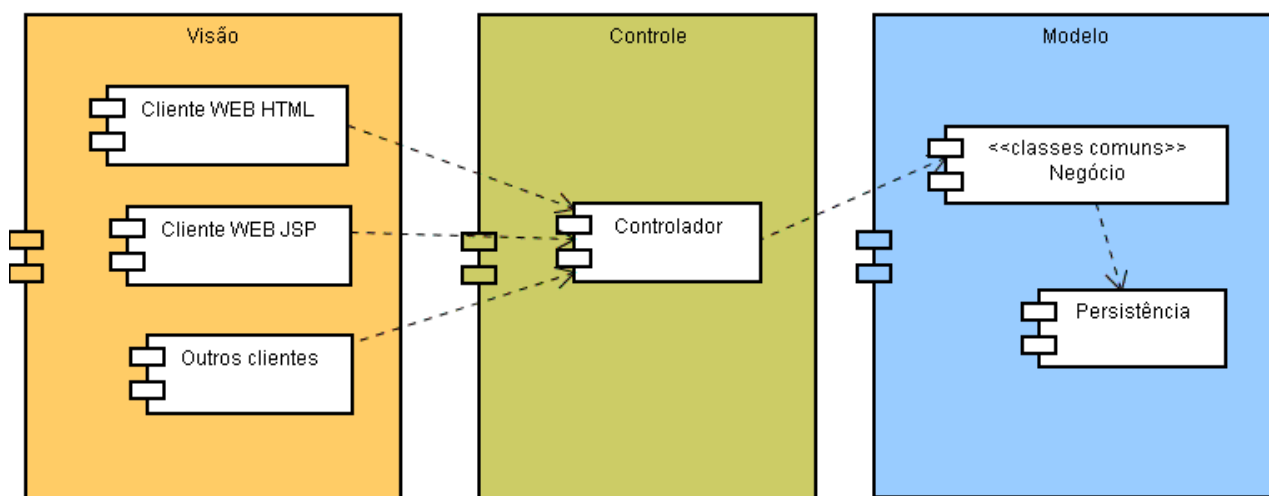


Figura 1 – Representação da arquitetura MVC

A adoção da arquitetura MVC torna a aplicação escapável e de fácil manutenção, além de propiciar o desenvolvimento em paralelo para cada camada, pois todas são independentes (MACORATTI, 2007).

## 2.2. UML

A UML (Unified Modeling Language) é uma linguagem para especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos. Sintetiza os principais métodos existentes, sendo considerada uma das linguagens mais expressivas para modelagem de sistemas orientados a objetos. Por meio de seus diagramas é possível representar sistemas de softwares sob diversas perspectivas de visualização. Facilita a comunicação de todas as pessoas envolvidas no processo de desenvolvimento de um sistema - gerentes, coordenadores, analistas, desenvolvedores - por apresentar um vocabulário de fácil entendimento (SILVA, 2001).

### 2.2.1. Diagrama de Casos de Uso

Um caso de uso descreve um objetivo que um ator externo ao sistema tem com o sistema. Um ator pode ser um elemento humano ou não que interage com o sistema. O ator se encontra fora do escopo de atuação do sistema, enquanto o conjunto de casos de uso forma o escopo do sistema. A linha que separa os atores dos casos de uso é a fronteira do sistema.

O diagrama de caso de uso representa graficamente esta interação e define o contexto do sistema. Os atores são representados por representações simplificadas de uma figura humana, enquanto os casos de uso são elipses contendo cada uma o nome de um caso de uso. Os atores se comunicam com os casos de uso, que é representado por uma linha unindo os dois elementos. Uma seta pode, opcionalmente, representar o fluxo principal de informação nesta interação e ajudar a leitura do caso de uso.

### 2.2.2. Diagrama de Classes

Os diagramas de classe descrevem as classes que formam a estrutura do sistema e suas relações. As relações entre as classes podem ser associações, agregações ou heranças. As classes possuem além de um nome, os atributos e as operações que desempenham para o sistema. Uma relação indica um tipo de dependência entre as classes. Essa dependência pode ser forte como no caso da herança ou da agregação ou mais fraca, como no caso da associação, mas indicam que as classes relacionadas cooperam de alguma forma para cumprir um objetivo para o sistema.

Sendo uma linguagem de descrição, a UML permite diferentes níveis de abstração (perspectivas) aos diagramas, dependendo da etapa do desenvolvimento do sistema em que se encontram. Assim, os diagramas de classe podem exibir nas fases iniciais da análise apenas o nome das classes (diagrama de classes conceitual), e em uma fase seguinte os atributos e operações (diagrama de classes de especificação). Finalmente, em uma fase avançada do projeto pode exibir os tipos dos atributos, a visibilidade, a multiplicidade das relações e diversas restrições (diagrama de classes de implementação). Existem elementos na UML para todas estas representações.

### 2.2.3. Diagrama de Componentes

Os diagramas de componentes mostram os elementos reutilizáveis de software e sua interdependência. Um componente é formado por um conjunto de classes que são implementadas no componente. Um componente, assim como as classes que ele possui, depende funcionalmente das classes de outro componente. O diagrama de componentes mostra esta dependência, inclusive entre os diversos arquivos que compõem o sistema.

## 2.3. JSF

O JSF (JavaServer Faces) é uma tecnologia Java EE (Enterprise Edition) utilizada na criação de interfaces gráficas de aplicações web. Seu objetivo é simplificar o desenvolvimento desse tipo de aplicação. Ele é composto por um conjunto de APIs para representação de componentes de interface com o usuário. A utilização do JSF implica em utilizar a arquitetura MVC, ou semelhante, na aplicação, de tal forma que as camadas de Visão e Controle ficam sob responsabilidade do JSF, enquanto que o Modelo fica sob total responsabilidade do desenvolvedor.

O JavaServer Faces foi desenvolvido segundo a especificação JSR-127. Além da implementação da Sun (<http://java.sun.com/javaee/jaserverfaces/overview.html>), existem várias outras, inclusive de código aberto, como, por exemplo, a implementação Tomahawk, do Apache MyFaces Project (<http://myfaces.apache.org/tomahawk/index.html>).

## 2.4. JPA

A JPA (Java Persistence API) é um conjunto de interfaces, especificadas na JSR-220, responsáveis pela padronização do mapeamento objeto-relacional na plataforma Java. A API fornece uma solução completa para o mapeamento e persistência de objetos (ROCHA; KUBOTA, 2007), através de:

- Um modo declarativo de descrever o mapeamento O/R (objeto relacional);
- Uma linguagem de consulta;
- Um conjunto de ferramentas para manipular entidades.

Desta forma, ao utilizar JPA, todas as consultas e operações de persistência tornam-se independentes do banco de dados que está sendo utilizado. Com isso, uma eventual mudança de banco de dados passa a ter um impacto bem menor na aplicação.

Existem várias implementações dessa API, os chamados provedores JPA. A implementação de referência é a TopLink JPA (<http://www.oracle.com/technology/products/ias/toplink/jpa/index.html>), desenvolvida pela Oracle.

## 2.5. AJAX

AJAX (Asynchronous Javascript And XML) é o uso sistemático de tecnologias providas por navegadores, como Javascript e XML, para tornar páginas web mais interativas, dinâmicas e criativas, utilizando-se de solicitações assíncronas de informações (ASLESON; SCHUTTA, 2006).

As tecnologias envolvidas no AJAX são:

- Javascript - A linguagem de script dos navegadores;
- DOM - A árvore de estrutura do HTML e CSS, acessada via JavaScript;
- XmlHttpRequest - A peça principal., responsável por estabelecer conexão com o servidor da aplicação sem precisar recarregar a página do cliente.

## 3. METODOLOGIA

Para o desenvolvimento do website foi utilizada a linguagem de programação JAVA com o objetivo de utilizar e avaliar as tecnologias citadas previamente. Foram utilizados também diagramas da UML para criação de modelos de abstração para análise e projeto do sistema. Dentre eles, estão o diagrama de casos de uso, de classes e de componentes.

Conforme descrito anteriormente, o sistema consiste em facilitar a entrada de informações por parte do engenheiro (profissional responsável pelo projeto) e oferecer aos proprietários e engenheiros a atual situação da análise do projeto de combate a incêndio (processo).

O diagrama de casos de uso abaixo mostra justamente como ocorre a interação entre os usuários e as partes funcionais do sistema. Como atores dos casos de uso, temos: o Atendente do CBMRN, responsável por validar os dados do Engenheiro (profissional) recém cadastrado e enviar senha de acesso ao sistema para o Engenheiro. Após a validação, o Profissional que é o Engenheiro responsável pelo projeto, pode fazer seu cadastro e manutenção de seus dados, cadastrar o pré-projeto (que é o início do cadastro do projeto de combate a incêndio com dados básicos, como os dados do imóvel e do proprietário) e acompanhar a situação da análise do projeto. O Proprietário interage com o sistema, fazendo o acompanhamento da situação do projeto. A Figura 2, abaixo, representa o diagrama de casos de uso construído a partir das funcionalidades requeridas pelo sistema.

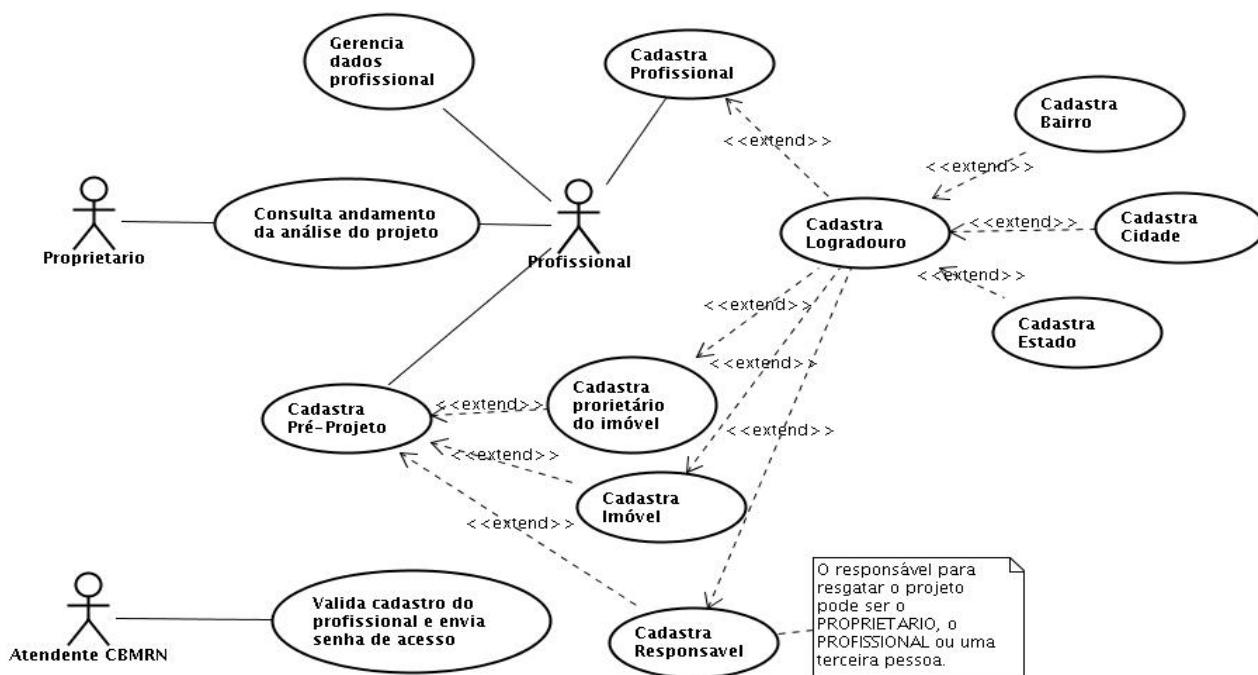
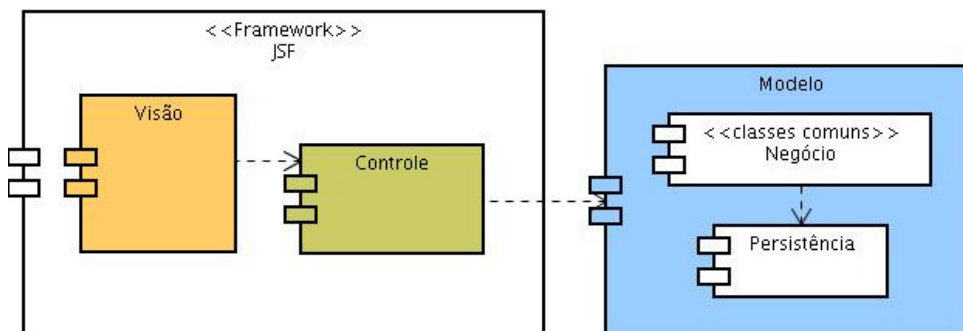


Figura 2 – Diagrama de casos de uso

O diagrama de componentes (ver Figura 3) foi utilizado para a modelagem da arquitetura do sistema, mostrando a interação existente entre os componentes da aplicação. A arquitetura adotada segue o padrão MVC (modelo – visão – controle).



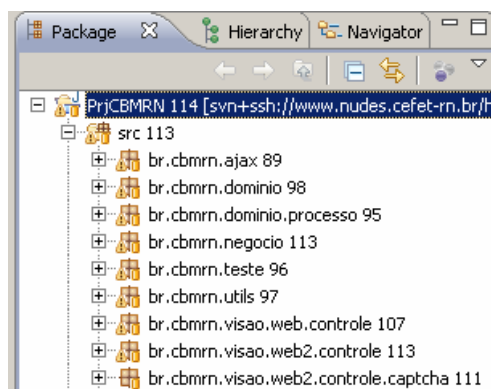
**Figura 3 – Diagrama de componentes (a arquitetura base do sistema)**

Para o desenvolvimento da interface com o usuário (páginas web) foi escolhida a tecnologia JSF (JavaServer Faces). Ela foi adotada por facilitar o processo de desenvolvimento da interface com o usuário e também por estar plenamente de acordo com a idéia da arquitetura MVC, atuando nas camadas de visão e controle. A implementação JSF utilizada foi a MyFaces Tomahawk (versão 1.1.6) por ser estável, bastante difundida e apresentar componentes mais sofisticados do que a implementação de referência da Sun, a JSF 1.1.

Como mecanismo de acesso a dados e persistência foi utilizado a tecnologia JPA, por facilitar o mapeamento objeto-relacional. A implementação utilizada foi a TopLink JPA (versão 2 build 41), que por sinal é a implementação de referência JPA.

Visando facilitar o processo de preenchimento de dados de endereço, o AJAX foi utilizado na aplicação, fazendo o auto-preenchimento de dados como bairro, logradouro e cidade, com base no CEP fornecido pelo usuário. Ele foi implementado através de Servlet e JavaScript. O Servlet é o responsável pela geração do XML que contem os dados do endereço, e o JavaScript pela chamada ao servlet e pelo preenchimento dos dados, obtidos via XML, na página de exibição.

O sistema foi basicamente subdividido em cinco pacotes principais (ver figura 4): domínio, negócio, controle, util e ajax.



**Figura 4 – Pacotes**

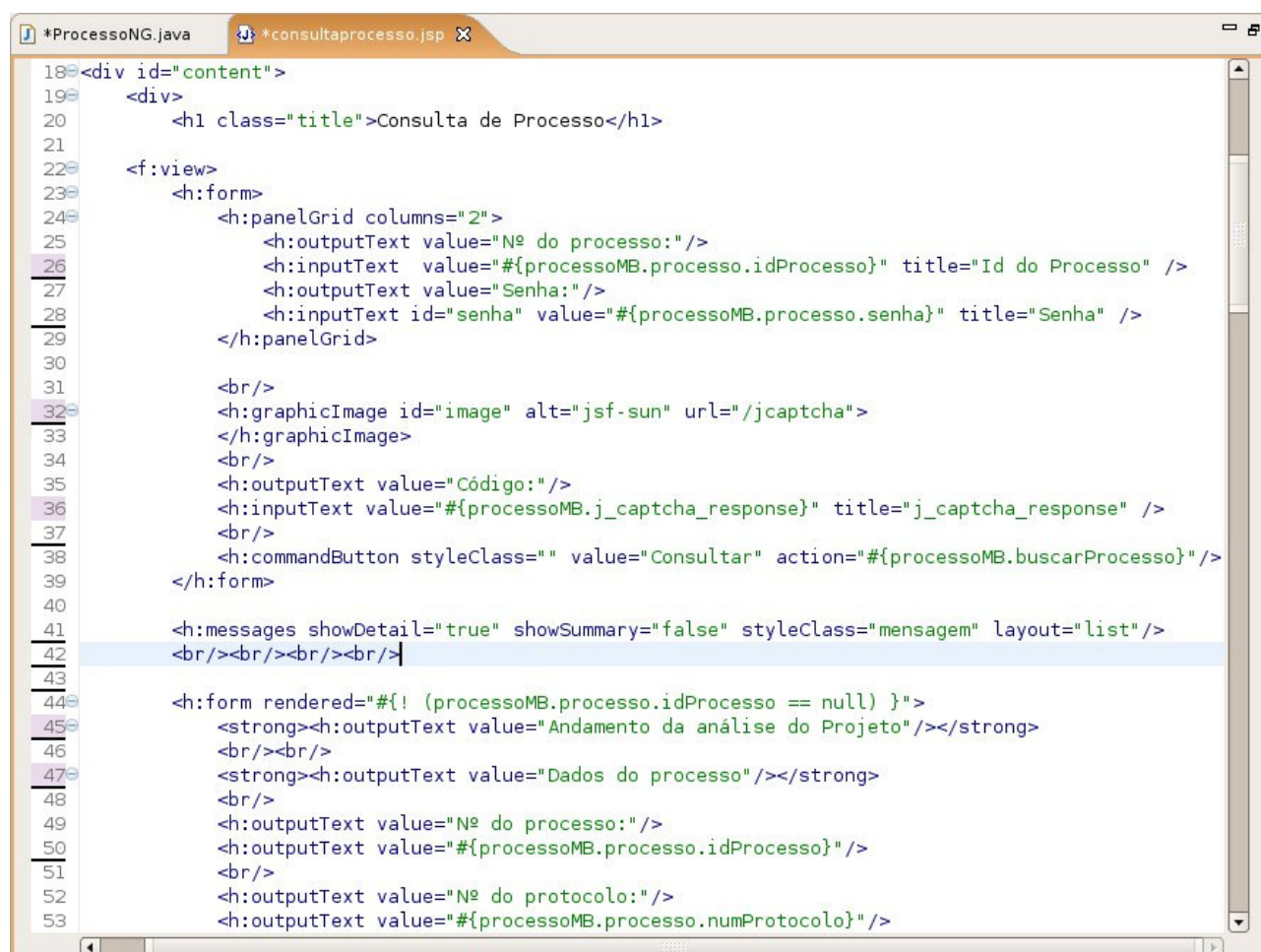
A seguir uma breve descrição de cada pacote.

- **domínio:** representa as entidades persistentes e alvos de processamento.
- **negócio:** representa todas as classes que possuem o núcleo funcional do sistema. Ele juntamente com o pacote de domínio e mais algumas classes do pacote útil formam o componente modelo da arquitetura MVC nesse sistema.
- **controle:** possui as classes que interagem com as páginas JSF e o negócio. São responsáveis por empacotar os dados das páginas JSF nas classes de domínio e chamarem a classe de negócio. São conhecidos no JSF como os ManagedBeans. Nesse sistema representa o componente de controle.

- **util:** possui classes utilitárias do sistema, como classes responsáveis pela conexão, classes responsável por envio de e-mail e classes responsáveis por geração de identificadores(ID) para as entidades.
- **ajax:** possui servlets responsáveis pela geração de XML.

Em seguida, serão apresentados alguns trechos de códigos construídos para a solução do caso de uso de "Consulta Andamento da Análise do Projeto".

O arquivo "consultaprocessos.jsp" (ver Figura 5), na arquitetura MVC, está dentro da camada de visão. Neste arquivo, na linha 26, é utilizada uma instância da classe ProcessoMB, que possui como atributo um objeto da classe Processo. Nesta mesma linha o atributo "idProcesso" da classe Processo é mapeado para um componente "h:inputText". Este componente quando renderizado (montado e exibido) em um navegador será equivalente a uma caixa de texto, e implicará na alteração do valor do atributo "idProcesso" com o valor informado pelo usuário, assim que a página for submetida.



```
18<div id="content">
19  <div>
20    <h1 class="title">Consulta de Processo</h1>
21
22    <f:view>
23      <h:form>
24        <h:panelGrid columns="2">
25          <h:outputText value="Nº do processo:"/>
26          <h:inputText value="#{processoMB.processo.idProcesso}" title="Id do Processo" />
27          <h:outputText value="Senha:"/>
28          <h:inputText id="senha" value="#{processoMB.processo.senha}" title="Senha" />
29        </h:panelGrid>
30
31        <br/>
32        <h:graphicImage id="image" alt="jsf-sun" url="/jcaptcha">
33        </h:graphicImage>
34        <br/>
35        <h:outputText value="Código:"/>
36        <h:inputText value="#{processoMB.j_captcha_response}" title="j_captcha_response" />
37        <br/>
38        <h:commandButton styleClass="" value="Consultar" action="#{processoMB.buscarProcesso}" />
39      </h:form>
40
41      <h:messages showDetail="true" showSummary="false" styleClass="mensagem" layout="list"/>
42      <br/><br/><br/><br/>
43
44      <h:form rendered="#{!(processoMB.processo.idProcesso == null)}">
45        <strong><h:outputText value="Andamento da análise do Projeto"/></strong>
46        <br/><br/>
47        <strong><h:outputText value="Dados do processo"/></strong>
48        <br/>
49        <h:outputText value="Nº do processo:"/>
50        <h:outputText value="#{processoMB.processo.idProcesso}" />
51        <br/>
52        <h:outputText value="Nº do protocolo:"/>
53        <h:outputText value="#{processoMB.processo.numProtocolo}" />

```

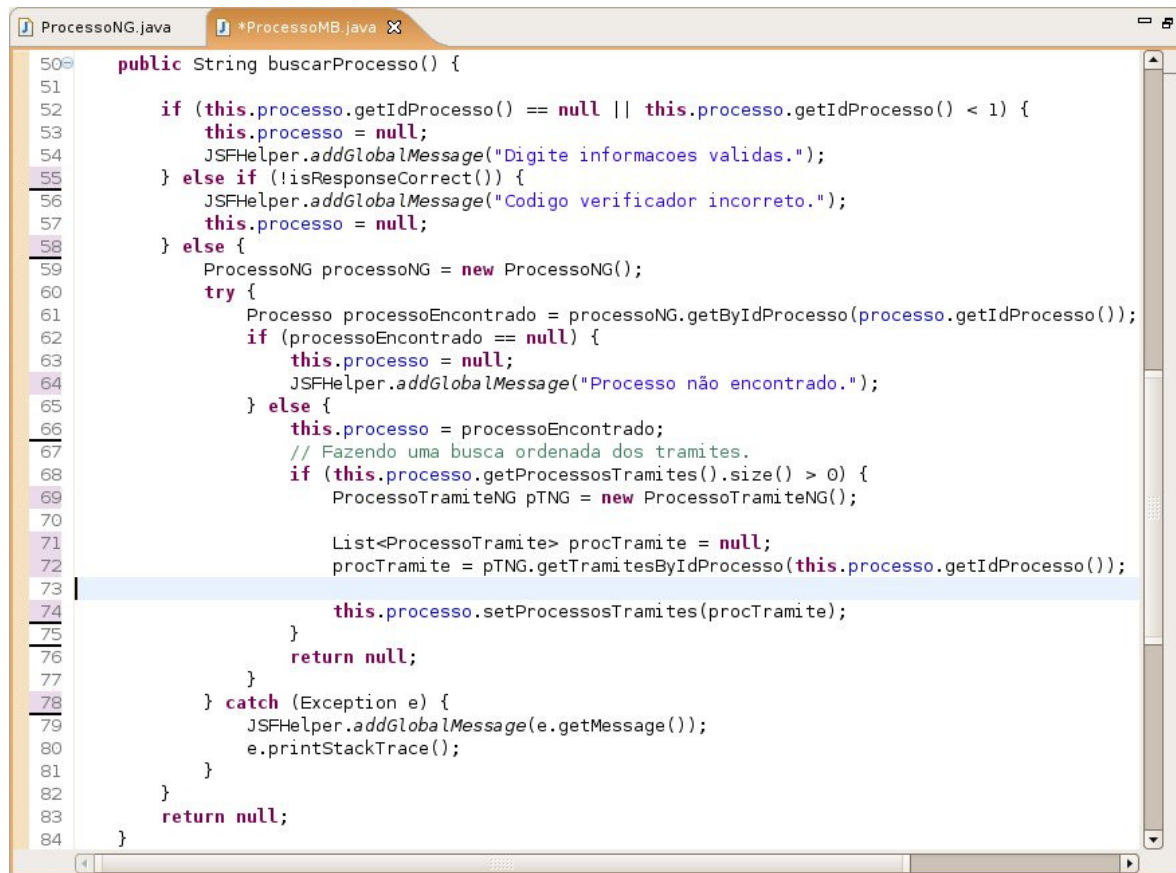
Figura 5 – Trecho de código do arquivo consultaprocessos.jsp (MVC – Visão)

Ainda observando o trecho de código do arquivo "consultaprocessos.jsp", na linha 38, o componente "h:commandButton" é mapeado para o método "buscarProcesso" da classe ProcessoMB. Este componente ao ser renderizado será equivalente a um botão, e ao ser clicado irá disparar o método "buscarProcesso". A instância da classe ProcessoMB está ligada a página JSF, logo todos os valores exibidos ao usuário na página, como nº do processo, nome do proprietário entre outros, correspondem exatamente aos valores contidos nas propriedades de ProcessoMB.

A classe ProcessoMB (ver figura 6) encontra-se no pacote de controle, e como já informado de forma genérica, esse pacote é o elo de ligação entre as páginas da aplicação e o modelo. Na arquitetura MVC, está dentro da camada de controle. O método explícito, na linha 61, mostra que dentro da classe ProcessoMB, no



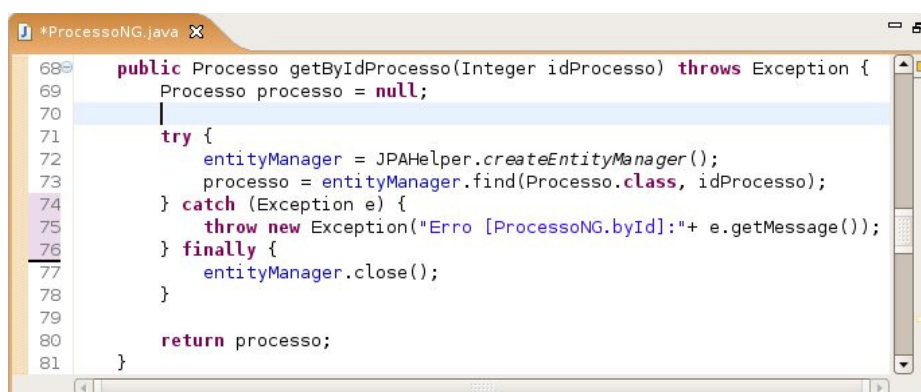
método “buscarProcesso”, ocorre uma chamada a classe ProcessoNG. Esta última classe é responsável pelo núcleo funcional da aplicação e de todas as operações que envolvem a entidade processo.



```
50 public String buscarProcesso() {
51
52     if (this.processo.getIdProcesso() == null || this.processo.getIdProcesso() < 1) {
53         this.processo = null;
54         JSFHelper.addGlobalMessage("Digite informacoes validas.");
55     } else if (!isResponseCorrect()) {
56         JSFHelper.addGlobalMessage("Codigo verificador incorreto.");
57         this.processo = null;
58     } else {
59         ProcessoNG processoNG = new ProcessoNG();
60         try {
61             Processo processoEncontrado = processoNG.getByIdProcesso(this.processo.getIdProcesso());
62             if (processoEncontrado == null) {
63                 this.processo = null;
64                 JSFHelper.addGlobalMessage("Processo não encontrado.");
65             } else {
66                 this.processo = processoEncontrado;
67                 // Fazendo uma busca ordenada dos tramites.
68                 if (this.processo.getProcessosTramites().size() > 0) {
69                     ProcessoTramiteNG pTNG = new ProcessoTramiteNG();
70
71                     List<ProcessoTramite> procTramite = null;
72                     procTramite = pTNG.getTramitesByIdProcesso(this.processo.getIdProcesso());
73
74                     this.processo.setProcessosTramites(procTramite);
75                 }
76                 return null;
77             }
78         } catch (Exception e) {
79             JSFHelper.addGlobalMessage(e.getMessage());
80             e.printStackTrace();
81         }
82     }
83     return null;
84 }
```

Figura 6 – Trecho de código da classe ProcessoMB (MVC - Controle)

A classe ProcessoNG (ver Figura 7) faz parte do componente de Modelo. No caso, o trecho de código exibido na figura abaixo é o método “getByIdProcesso”, que tem como finalidade efetuar a busca de um processo no banco de dados com base no idProcesso (código do processo). A busca é feita utilizando a tecnologia JPA.



```
68 public Processo getByIdProcesso(Integer idProcesso) throws Exception {
69     Processo processo = null;
70
71     try {
72         entityManager = JPAHelper.createEntityManager();
73         processo = entityManager.find(Processo.class, idProcesso);
74     } catch (Exception e) {
75         throw new Exception("Erro [ProcessoNG.byId]: "+ e.getMessage());
76     } finally {
77         entityManager.close();
78     }
79
80     return processo;
81 }
```

Figura 7 – Trecho de código da classe ProcessoNG (MVC - Modelo)

#### 4. WEBSITE

Conforme citado anteriormente, uma das funcionalidades do sistema é o acompanhamento da análise do projeto de combate a incêndio. O acesso a essa informação se dá através da guia “consulta processo” (ver Figura 8). Para isso o usuário não precisa estar logado no sistema, apenas terá que informar o número do processo e a senha recebida do protocolo do CBMRN. Um detalhe nessa consulta é a utilização de um



gerador de imagem com caracteres aleatórios, com base no qual o usuário deverá fornecer um código. Este gerador foi utilizado com o objetivo de evitar que algoritmos de força bruta tivessem acesso aos dados do processo por tentativa e erro do preenchimento da senha.

**cbmrn**  
natal - rn

bem vindo login cadastro **consulta processo**

**Consulta de Processo**

Nº do processo: 19  
Senha:

Código: feeted

**Andamento da análise do Projeto**

**Dados do processo**  
Nº do processo: 19  
Nº do protocolo: 19  
Situação: ENTREGUE  
Usuário solicitante: Cláudio Luis Régis de Menezes  
Data Entrada: 12/03/07 13:09

**Dados do projeto**  
Proprietário: Natur Administradora de Bens LTDA  
CPF/CNPJ:  
Engenheiro: Cláudio Luis Régis de Menezes  
Nº CREA:  
Imóvel: Edifício Ponta Negra Brasil II  
Área construída:(m2) 1930.33  
Endereço: Travessa Alto da Boa Vista  
Número: S/N  
Bairro: Ponta Negra  
Cidade: Natal

**Rastreamento do Processo**

Data Entrada	Entrada	Data saída	Destino	Situação
12/03/07 13:09	Atendimento	12/03/07 13:09	Vistoria	CONCLUÍDO
12/03/07 13:43	Vistoria	12/03/07 14:37	Atendimento	CONCLUÍDO
12/03/07 14:37	Atendimento	12/03/07 14:39		ENTREGUE

**Figura 8 - Tela de consulta de processo**

Ao realizar a consulta, podemos observar as seguintes informações: os dados do processo, como o usuário que solicitou a análise e a data de entrega do projeto; os dados do projeto, como o nome do imóvel, a área construída, a localização do imóvel, e o engenheiro responsável; e o rastreamento do local onde o processo de encontra informando quando ele entrou em um setor e para onde foi encaminhado.

Outro caso de uso do sistema é o Cadastro do Profissional (ver Figura 9), através da guia “cadastro”. O AJAX foi utilizado nesse cadastro, entrando em ação no momento em que o CEP é fornecido. Se na base de dados já existir um endereço com o mesmo CEP, dados como cidade, bairro, tipo de logradouro e logradouro são preenchidos automaticamente pela aplicação. Também faz parte desse caso de uso o envio de e-mail para o Engenheiro informando do sucesso do cadastro e explicando que a conta dele só será ativada após a verificação dos dados fornecidos, feito pelo pessoal do CBMRN. Logo abaixo, a Figura 9 mostra a tela de cadastro profissional.

bem vindo login cadastro consulta processo

**Cadastro de Profissional (NOVO)**

**Dados Pessoais**

Nome:

RG:

RG (Orgão):

CPF:

CREA:

E-mail:

Fone(1):

Fone(2):

Fone(3):

Fone(4):

**Endereco**

CEP:

Cidade:

Bairro:

Tipo de Logradouro:

Logradouro:

Nº:

Complemento:

**Endereco**

CEP:

Cidade:

Bairro:

Tipo de Logradouro:

Logradouro:

Nº:

Complemento:

**Cadastrar**

Figura 9 – Tela do cadastro profissional e apresentação da funcionalidade do AJAX

## 5. CONCLUSÃO

Durante o desenvolvimento deste projeto muitas tecnologias foram utilizadas, com destaque para o JSF (JavaServer Faces) e o JPA (Java Persistence API). Utilizando o JSF percebe-se que realmente essa ferramenta reduz muito o trabalho do desenvolvedor no momento de montar a interface web da sua aplicação, além de estimulá-lo de maneira natural a trabalhar a sua aplicação em camadas. Quanto ao JPA no início ocorreram alguns problemas devido a tradição antiga de fazer o mapeamento objeto-relacional de maneira direta no banco, através de instruções SQL. Utilizando JPA tem-se de lembrar que as instruções (SELECT, INSERT, DELETE etc) não são mais feitas em dados de tabelas, e sim em objetos, que por sua vez representam os dados do banco. Sem dúvida JPA também é uma poderosa ferramenta, e impressiona pela capacidade de tornar a aplicação independente do banco de dados utilizado. Dependendo do contexto em que se encontra a aplicação a independência de banco de dados pode ser um fator importante.

Quanto ao resultado da solução desenvolvida (website) espera-se que todo o processo de análise de um projeto de combate a incêndio ocorra de maneira mais eficiente e rápida, já que os interessados na aprovação do projeto terão um meio de fácil acesso para saber como anda a análise, quais os problemas encontrados e como corrigi-los. Isso fará com que o CMBRN mantenha a máxima eficiência e eficácia em suas atividades constitucionais, mantendo o seu reconhecimento nacional como referência de qualidade no atendimento.

## REFERÊNCIAS

ASLESON, R.; SCHUTTA, N.T. **Fundamentos do Ajax**. 1. ed. Rio de Janeiro: Editora Alta Books, 2006.

DEBONI, JOSÉ EDUARDO ZINDEL **Diagramas da UML**

Disponível em: <<http://www.voxxel.com.br/pages/introdiauml.html>> Acesso em: 25 ago 2007.

MACORATTI, JOSÉ CARLOS **Padrões de Projeto : O modelo MVC - Model View Controller**.

Disponível em: <[http://www.macoratti.net/vbn\\_mvc.htm](http://www.macoratti.net/vbn_mvc.htm)> Acesso em: 1 set 2007.

ROCHA, A.D.; KUBOTA, S.O. **Persistência no Java EE**. Java Mazagine, Rio de Janeiro, ano 5, n. 39, p.28-37, 2006.

SILVA, DOUGLAS MARCOS DA. **UML – Guia de Consulta Rápida** 1. ed. São Paulo: Novatec, 2001.

## AGRADECIMENTOS

Ao Corpo de Bombeiros Militar do Rio Grande do Norte pela parceria e financiamento indispensáveis ao desenvolvimento deste trabalho.