A APLICAÇÃO DE JAVASERVER FACES E TECNOLOGIAS ASSOCIADAS NA CONSTRUÇÃO DE UM SISTEMA DE CONTROLE DE SOLICITAÇÕES

Wellington Wagner Rodrigues SILVA
Centro de Ensino Superior de Juiz de Fora, Juiz de Fora, MG
Marco Antônio Pereira ARAÚJO

Resumo: O Sistema de gestão de Solicitações foi desenvolvido visando atender aos processos de um setor de Tecnologia de Informação (TI) de empresas em qualquer área de mercado, onde a atividade fim não é TI. Na construção do sistema foram utilizadas ferramentas gratuitas como NetBeans, linguagem Java e SGBD SQLServer *Express* 2008. Adotou-se o padrão *Model-View-Controller* para facilitar o reuso da aplicação, inclusive visando manutenções futuras do sistema e permitindo flexibilidade na adaptação ao ambiente da empresa, além de proteger as regras de negócio.

Palavras-chave: Controle de solicitação; Controle de atividades; Manutenção de Computadores.

1 INTRODUÇÃO

O setor de Tecnologia da Informação do Centro de Ensino Superior de Juiz de Fora (CES/JF) é responsável por atender todas as solicitações de manutenção dos computadores da instituição. Atualmente, a instituição possui um número significativo de computadores, de uso administrativo e acadêmico separados em três *campi*, com isso o volume de computadores que precisam de manutenção é grande, e o setor tem apenas dois funcionários em cada *campus* para atender essas solicitações. Com este volume de solicitações é difícil ter o controle sobre quais foram atendidas e às quais ainda não foram.

Tendo em vista esses problemas, foi desenvolvido um sistema que pudesse gerenciar esse volume de solicitações. O sistema foi construído com a linguagem Java para a plataforma Web utilizando JavaServer Faces com a suíte de componentes Primefaces, Hibernate para persistência de objetos e iReport para a confecção de relatórios. O JavaServer Faces foi utilizado, pois é uma especificação oficial da linguagem Java, o que torna as atualizações mais constantes, e a disponibilização da especificação não fica nas mãos de empresas terceiras. A suíte de componentes do Primefaces foi escolhida pois é uma das mais completas, possui ampla documentação e tem muito conteúdo explicativo na Internet. O Hibernate foi selecionado pela facilidade de utilização, por ser o *framework* de mapeamento objeto/relacional mais utilizado e conhecido na plataforma Java. O Hibernate é também um dos pioneiros

desse tipo de solução de persistência e foi utilizado como base para a definição do padrão Java *Persistence* API 2.0 (SOUZA, 2013). O iReport foi utilizado por ser uma ferramenta *open source* que oferece uma IDE para gerar relatórios de forma simples e flexível, e que pode utilizar dados vindos de diversas fontes.

Este artigo está divido em mais três sessões, além desta introdução. A Sessão 2 aborda as metodologias utilizadas para desenvolver o sistema. Na Sessão 3 é feita uma introdução ao JavaServer Faces, Primefaces, Hibernate e iReport. A Sessão 4 aborda o sistema em si, mostrando algumas janelas e funcionalidades. Na Sessão 5 é feita a conclusão do artigo com as contribuições, dificuldades e os trabalhos futuros.

2 METODOLOGIA

A metodologia utilizada na realização deste trabalho consiste num conjunto de atividades, desenvolvidos da seguinte forma:

Atividade 1: Foi realizada uma entrevista com o gerente do setor para fazer o levantamento das funcionalidades a serem disponibilizadas pelo sistema, assim também foi possível saber quais seriam os dados que o sistema deveria coletar;

Atividade 2: Acompanhar o dia a dia dos funcionários para saber quais outras informações poderiam ser incluídas no sistema;

Atividade 3: Elaboração dos casos de uso, requisitos funcionais e não funcionais, além da definição das regras de negócio;

Atividade 4: Implementação do sistema, utilizando as tecnologias descritas anteriormente;

Atividade 5: Teste no sistema com os usuários do setor, afim de obter o *feedback* e realizar melhorias;

Atividade 6: Implementação das melhorias sugeridas pelos usuários do setor.

3 AS TECNOLOGIAS UTILIZADAS

Nesta sessão é feita uma introdução sobre as tecnologias e ferramentas utilizadas para o desenvolvimento do Sistema de Gestão de Solicitações, será dado um enfoque maior para o JavaServer Faces. Haverá também uma breve apresentação do Hibernate e do iReport.

3.1 JAVASERVER FACES

O JavaServer Faces (JSF) (ORACLE, 2014d) é a especificação oficial do Java EE que define uma abordagem para a criação de aplicações Web de modo similar com o utilizado na criação de sistemas *desktop* em Java, sendo a base de um conjunto de componentes visuais e não visuais, extensíveis, em sua maioria. Existem diversas suítes de componentes de terceiros que enriquecem o JSF por meio de componentes mais inteligentes e visualmente mais elegantes, como caixas de texto com *drag and drop*, diálogos modais, *autocomplete*, suporte a AJAX, entre outros. Também é oferecido um *framework* para realizar validações, conversões, navegação, aparência, internacionalização e acessibilidade, dentre outros aspectos de um sistema Web. Dessa forma, é possível criar aplicações Web com interfaces ricas sem muito esforço.

Em uma aplicação JSF, a interface gráfica é escrita em HTML, com o uso de *tags* especiais, definidas pelas *taglibs* da especificação (como <h:form/>). Para abstrair a complexidade da comunicação entre a interface gráfica e as classes de negócio, que são classes Java comuns, o JSF utiliza o conceito de *ManagedBeans*. Esses representam os objetos cujo ciclo de vida é gerenciado pelo JSF. Tais objetos são acessíveis tanto a partir das páginas HTML quanto de outras classes da aplicação. Para caracterizar uma classe como um *ManagedBean*, não precisa herdar de nenhuma classe pré-definida ou mesmo implementar uma interface, basta marcá-la com a anotação *@ManagedBean* e o JSF se encarrega do restante (COUTINHO, 2013).

Um *ManagedBean* tem quatro principais tipos de escopo, que são:

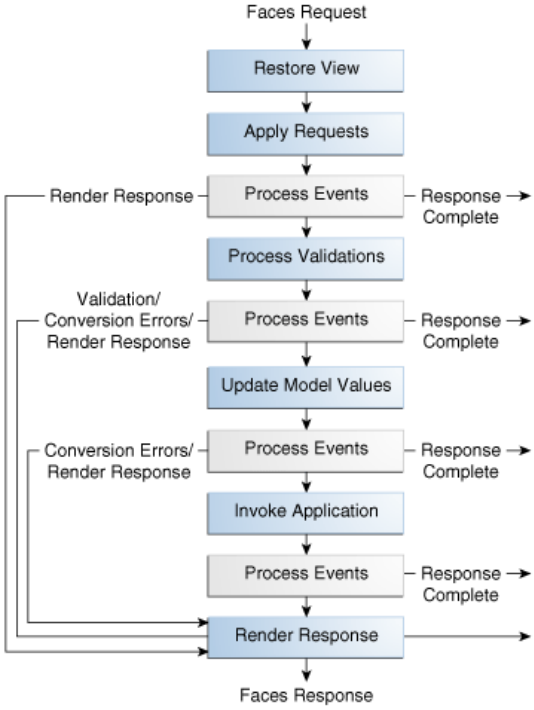
- *RequestScoped* - o *ManagedBean* fica ativo enquanto uma requisição HTTP é feita a uma página do JSF que está relacionada com um *bean* deste escopo, e é destruído quando a resposta da requisição é enviada ao usuário;
- *ViewScoped* - permanece ativo enquanto o usuário estiver interagindo com a mesma página JSF que está relacionada com um *ManagedBean* deste escopo, e é destruído assim que o usuário faz uma requisição de uma URL diferente;
- *SessionScoped* - permanece ativo enquanto a sessão do usuário estiver ativa e é destruído quando a sessão do usuário é invalidada;
- *ApplicationScoped* – o *ManagedBean* é criado quando ocorre a primeira requisição HTTP, sendo feita ao *bean* relacionado que esse tipo de escopo e permanece ativo enquanto a aplicação estiver ativa.

O ciclo de vida de uma aplicação JSF consiste em seis fases, conforme a Figura 1, onde cada fase é descrita a seguir.

3.1.1 Restore View

No servidor existe a árvore de componentes, que é a representação da página exibida para o usuário em forma de componentes e organizada hierarquicamente. Nessa fase, uma *view* (representação da página) é criada ou restaurada caso a mesma já exista no servidor.

Figura 1
Ciclo de vida padrão de uma requisição JSF



Fonte: Jendrock *et al.*, 2014.

3.1.2 Apply Request Values

Depois que a árvore de componentes é criada/restaurada, cada componente da árvore usa o método *decode* para extrair o novo valor dos parâmetros da requisição. O componente armazena esse valor. Se houver falha na conversão, uma mensagem de erro será gerada, sendo exibida durante a fase *Render Response*, junto com qualquer outro erro de validação.

3.1.3 Process Validations

Durante esta fase é realizado o trabalho de processar os pares de nome e valor das requisições que chegam. Então os valores da página do usuário são atribuídos aos componentes da *view* no servidor, isto é efetuado chamando o método *decode()* de cada componente, seguindo a hierarquia da *view*.

Se ocorrer algum erro durante a atribuição de valores uma mensagem de erro, é gerada e enfileirada para ser exibida ao usuário.

3.1.4 Update Model Values

Nesta fase, as propriedades dos *ManagedBeans*, entidades (classes Java), são atualizadas com os novos valores vindos dos UI *components* (*input text*, por exemplo) das páginas. A atualização é iniciada chamando o método *updateModel()* de cada componente *UIInput* na hierarquia da *view*.

O valor digitado em um campo pelo usuário neste ponto já foi convertido e validado, então eles são atribuídos aos campos das classes Java correspondentes.

3.1.5 Invoke Application

Durante esta fase qualquer "*action handler*" e "*event listener*" limitados aos componentes UI são invocados, ou seja, a ação dos botões e *links* correspondem a um método em uma classe Java, é nesse momento que esse método é invocado. A classe Java que contém esse método é comumente chamada de *ManagedBean*.

Esta também é a fase onde talvez a navegação de página ocorra, isso significa que o usuário poderá ser redirecionado para outra página ou permanecer na página corrente.

3.1.6 Render Response

Durante esta fase, o JSF delega ao *container* JSP a responsabilidade de renderizar a página se a aplicação estiver usando páginas JSP. Se for uma requisição inicial, os componentes representados na página serão adicionados à arvore de componentes. Se não for uma requisição inicial, os componentes já estarão na árvore

de componentes e não será necessário adicioná-los novamente. Em qualquer um dos casos, os componentes serão renderizados como em um container JSP (JENDROCK et al, 2014).

3.2 HIBERNATE

Hibernate (2014) é um projeto que procura ter uma completa solução para o problema de gerenciamento de objetos persistentes em Java. O Hibernate é um *framework* que se relaciona com o banco de dados, onde esse relacionamento é conhecido como mapeamento objeto/relacional, deixando o desenvolvedor livre para se concentrar em problemas da lógica do negócio. Seu objetivo é aliviar o desenvolvedor de tarefas comuns de persistência, eliminando a necessidade de manipulação manual de SQL (*Structured Query Language*). No entanto, ao contrário de muitas outras soluções de persistência, o Hibernate não esconde o poder do SQL e garante que seu investimento em tecnologia relacional e conhecimento continuem sendo válidos. Sua simplicidade em configuração fornece ao desenvolvedor algumas regras para que sejam seguidas como padrões de desenvolvimento ao escrever a lógica de negócios e classes persistentes. De resto, o Hibernate se integra a um sistema se comunicando com o banco de dados como se fosse diretamente feito pela aplicação. A figura 2 mostra a arquitetura do Hibernate abstraindo aplicação dos processos de comunicação com o banco de dados (GONÇALVES, 2013).

Figura 2
Arquitetura do Hibernate

Fonte: Adaptado de HIBERNATE, 2014.

O Hibernate é suficientemente flexível para ser usado de várias formas e em várias arquiteturas. Por isso, na sua representação não há como detalhar a visão de todas as arquiteturas possíveis de se trabalhar com o *framework*.

3.3 IREPORT

O *JasperReports* (2014) foi criado a partir da necessidade de avaliação de ferramentas de relatórios para um projeto de software. As soluções existentes eram muito caras para o orçamento do projeto e foi escrita uma ferramenta de relatórios de utilização livre para uso pela comunidade.

O *JasperReports* é uma biblioteca escrita em Java, de código fonte *open source*, projetada para ajudar o desenvolvedor com a tarefa de criar relatórios para aplicações, tanto *desktop* como Web, fornecendo uma API (*Application Program Interface*) que facilita sua geração.

Embora o *JasperReports* tenha simplificado o desenvolvimento de relatórios, o desenvolvedor, além da necessidade de conhecer seu formato XML (*eXtensible Markup Language*) utilizado para criar os relatórios, também dependia de cálculos para determinar a posição de cada componente no relatório de forma harmônica.

Em 2002, foi criada uma ferramenta para gerar relatórios visuais, conhecida por *iReport* (2014). Sua característica era de desenvolver relatórios gerando o formato XML no padrão *JasperReports*. Isso tornou mais acessível e intuitivo o uso dos relatórios escritos nessas tecnologias. Em 2005, com a popularidade do *iReport*, a *JasperSoft* (empresa mantenedora da ferramenta) tornou-a oficial na construção de relatórios para o *JasperReports* (GONÇALVES, 2013).

4 O SISTEMA DE CONTROLE DE SOLICITAÇÕES

Nesta sessão será abordado o desenvolvimento do Sistema de Controle de Solicitações em si, detalhando seu funcionamento e implementação, dando ênfase a utilização da especificação JavaServer Faces.

4.1 CONTEXTUALIZAÇÃO DETALHADA

O setor de Tecnologia da Informação é responsável por atender todas as solicitações relacionadas a hardware, software e infraestrutura de rede e servidores, dos três *campi* do Centro de Ensino Superior de Juiz de Fora (CES/JF).

As solicitações atendidas pelos funcionários referentes a software e infraestrutura são atendimento para sanar dúvidas dos usuários da área administrativa da instituição em relação à utilização de softwares, configuração e mapeamento da rede administrativa e acadêmica, cabeamento de redes, troca de cartuchos de tinta de impressoras, auditorias para levantamento de softwares instalados indevidamente, instalação e, se necessário, ativação de softwares utilizados nas áreas administrativas e nos laboratórios de informática (como AutoCad, Adobe Premier, Matlab, NetBeans, MySQL, PHP), entre outras atividades. As solicitações relacionadas a hardware são formatação, instalação de sistemas operacionais, instalação e configuração de impressoras, substituição de peças com problemas e manutenções preventivas.

Os funcionários responsáveis por essas solicitações trabalham seis horas por dia, em diferentes turnos. Quando o funcionário que trabalha nos turnos da manhã e da tarde inicia uma manutenção, muitas vezes não consegue concluí-la no mesmo dia, e o funcionário que trabalha no turno seguinte, nem sempre se encontra com o funcionário que iniciou a manutenção, portanto não sabe o que aconteceu com o computador, nem o que já foi feito para evitar problemas e retrabalho. Acaba, então, não dando continuidade à manutenção iniciada pelo outro funcionário. Além desse problema, não é possível saber quais as solicitações que os funcionários estão atendendo ou já atenderam.

Todo final de mês os funcionários do setor têm que elaborar um relatório de atividades para enviar ao gerente, para que possa avaliar o desempenho, as atividades dos funcionários e detectar problemas. Como os funcionários, muitas vezes, esquecem ou não anotam as atividades que realizaram durante o expediente, acaba tornando o relatório de atividades incoerente e incompleto.

Para resolver esse problema, foi construído um sistema onde todas as atividades são registradas e ficam acessíveis aos funcionários do setor. Ao cadastrar a atividade, o funcionário descreve o problema e, se necessário, adiciona anotações pertinentes ao andamento da atividade. Caso não consiga finalizá-la, outro funcionário pode acessar o sistema, visualizar as anotações e, baseado nelas, dar continuidade

à atividade. O sistema também permitirá ao gerente gerar relatórios de atividades por funcionário, selecionando um período de tempo, tornando mais fácil a avaliação de desempenho dos funcionários.

4.2 LEVANTAMENTO DE REQUISITOS

Nesta seção serão listados os requisitos funcionais, não funcionais e regras de negócio levantadas para a elaboração do sistema, obtidos através de entrevistas com o gerente de TI.

4.2.1 Requisitos funcionais

RF01: O sistema deverá permitir ao gerente do setor incluir, alterar, excluir e consultar unidades (*campus*/local), contendo os dados nome e descrição.
RF02: O sistema deverá permitir ao gerente do setor incluir, alterar, excluir e consultar setores, contendo os dados nome e descrição.
RF03: O sistema deverá permitir ao gerente do setor incluir, alterar, excluir e consultar categorias, contendo os dados nome e descrição.
RF04: O sistema deverá permitir ao gerente do setor incluir, alterar, excluir e consultar status, contendo os dados nome e descrição.
RF05: O sistema deverá permitir ao gerente do setor incluir, alterar, ativar e desativar solucionadores (funcionários do setor de Tecnologia de Informação), contendo os dados usuário, unidades que o solucionador irá atender, quais as categorias de solicitações o solucionador irá atender e status (ativo ou inativo).
RF06: O sistema deverá permitir ao gerente do setor incluir, alterar e consultar parâmetros, contendo os dados status inicial, um ou mais status finais.
RF07: O sistema deverá permitir aos usuários do sistema abrir solicitações, informando unidade, setor, assunto e descrição do problema/solicitação.
RF07: O sistema deverá permitir aos usuários do sistema consultar e adicionar anotações às solicitações abertas por ele.
RF08: O sistema deverá permitir aos solucionadores inserir, alterar, assumir e adicionar andamentos às solicitações no qual ele tenha assumido.
RF09: O sistema deverá permitir ao gerente emitir o relatório informando o solucionador, data de inicial e data final.

4.2.2 Requisitos Não Funcionais

RNF01: O sistema deverá operar em plataforma Web.
RNF02: O sistema deverá possibilitar que todos os relatórios sejam visualizados antes do envio para a impressora.

4.2.3 Regras de Negócio

RN01: Todos os usuários do sistema podem abrir solicitações e visualizá-las para acompanhar seu andamento.
RN02: Somente o gerente ou solucionadores podem alterar a unidade, categoria e o status de uma solicitação.
RN03: Somente o gerente do setor pode acessar os cadastros básicos, de solucionadores e parâmetros.
RN04: Tanto os solucionadores quanto o gerente podem associar um ou mais solucionadores a uma solicitação.
RN05: Ao finalizar uma solicitação, o solucionador terá que obrigatoriamente preencher o campo de descrição, informando a solução para a solicitação.

4.3 MODELAGEM DO SISTEMA

A modelagem do Sistema de Gestão de Solicitações foi feita utilizando os principais diagramas da UML (*Unified Modeling Language*), incluindo o Diagrama de Casos de Uso, Diagrama de Classes e Diagramas de Seqüência. A Figura 3 representa o Diagrama de Casos de Uso referente aos requisitos funcionais descritos anteriormente.

A Figura 4 representa o Diagrama de Classes referente ao Sistema de Gestão de Solicitações. É possível perceber que a classe **SSTISolicitacao** é uma das classes mais complexas e com maior número de relacionamentos com outras classes do sistema.

A classe **SSTISolicitacao** será utilizada posteriormente para demonstrar o processo de abertura de solicitação que é um dos processos mais complexos do

sistema. A Figura 5 representa o Diagrama de Sequência desse processo de abertura de uma solicitação por um cliente.

Figura 3
Diagrama de casos de uso

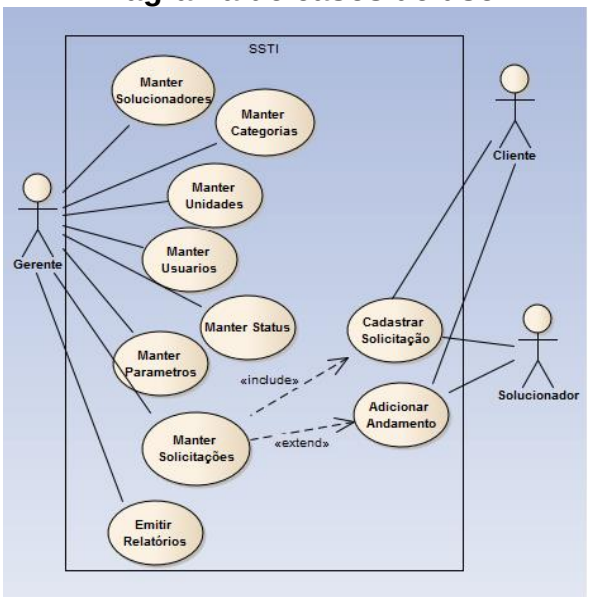


Figura 4
Diagrama de classes

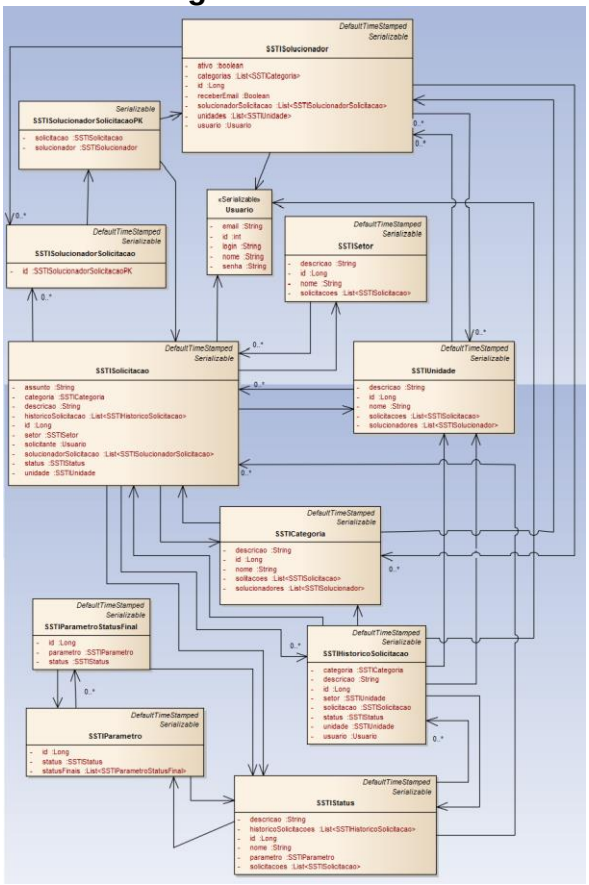
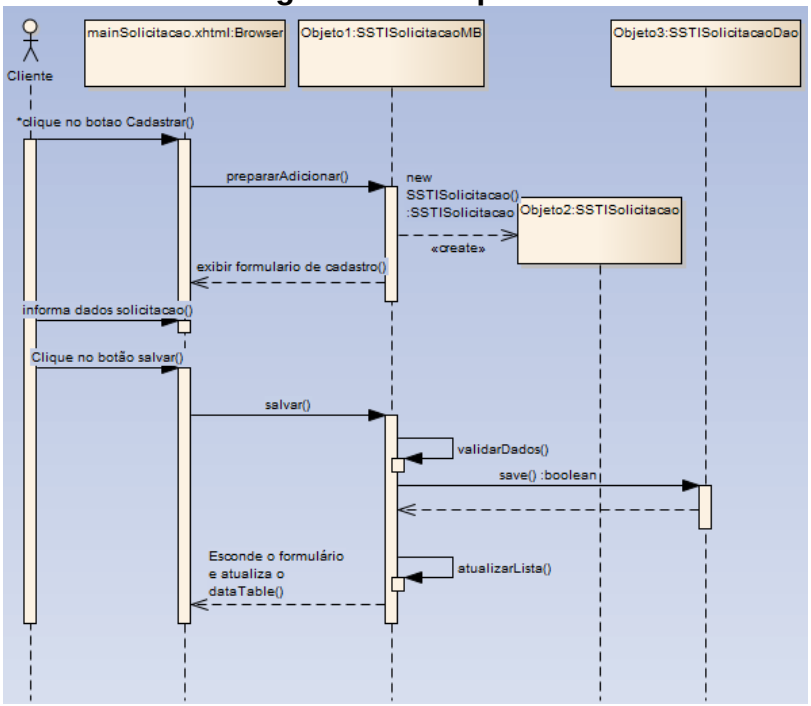


Figura 5
Diagrama de Sequência



4.4 IMPLEMENTAÇÃO DO SISTEMA DE GESTÃO DE SOLICITAÇÃO

O sistema em questão é divido em três partes, a primeira está relacionada com os cadastros básicos para o funcionamento e gerenciamento das solicitações, que são: cadastro de *status* das atividades, setores, unidades/campus, categorias, parâmetros e solucionadores.

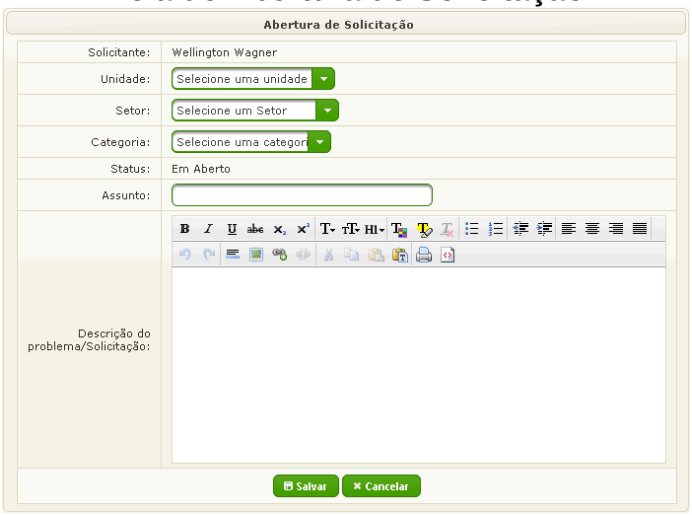
A primeira parte é a responsável pelos cadastros básicos para o funcionamento e gerenciamento das solicitações, estando representada no menu pela opção de cadastros, e todos os seus subitens (Figura 6). Nas opções Setores, Unidades, Categorias e *Status*, são encontrados os cadastros simples (CRUD – *Create, Retrieve, Update, Delete*) dessas opções. Na opção Parâmetros é possível informar ao sistema quais dos *status* cadastrados é o inicial e quais são os *status* finais, ou seja, quando um usuário abrir uma solicitação, o sistema irá verificar nos parâmetros qual o *status* deve ser incluído primeiramente, e quando um funcionário do TI alterar o *status* da solicitação para um dos *status* cadastrados nos parâmetros como final, o sistema irá entender que a solicitação foi concluída. Na opção Solucionadores é informado quais os usuários do sistema são solucionadores, e também qual unidade/*campus* e categorias de solicitações o funcionário poderá atender.

Figura 6
Menu do Sistema

A segunda parte está representada no menu pelas opções Minhas Solicitações e Abrir Solicitação, disponíveis para todos os usuários do sistema. Na opção Minhas Solicitações, o usuário pode visualizar todas as suas solicitações, acompanhar seu andamento, visualizar o histórico de alterações e anotações de forma cronológica, sendo possível identificar o usuário que realizou a alteração. Na opção Abrir Solicitação o usuário pode abrir uma nova solicitação, informando a unidade/*campus*, setor, assunto e descrição da solicitação (Figura 7).

Para o escopo deste artigo, será apresentada detalhadamente a funcionalidade de Abertura de Solicitação, por ser a mais complexa do sistema, apesar de todas as funcionalidades descritas nos requisitos funcionais da aplicação terem sido implementadas.

Figura 7
Tela de Abertura de Solicitação



A terceira parte que representada no menu inclui as opções ***Dashboard*** e **Relatórios**, estando disponíveis somente para funcionários do setor de TI. No ***Dashboard***, os funcionários podem fazer alterações de *status*, solucionador, unidade/*campus*, setor, categoria ou adicionar anotações referentes às atividades solicitadas/cadastradas por eles ou por outros funcionários.

4.4.1 Implementação da Abertura de Solicitações

A Listagem 1 representa um trecho do *template* JSF que será utilizado como base para as outras páginas do sistema. As *tags* <*p:layout*> e <*p:layoutUnit*> são componentes do Primefaces utilizados para dividir uma página em regiões. Essa página é separada em duas áreas, o cabeçalho e a área central. A *tag* <*ui:insert*> é utilizada para definir áreas de substituição na página. A área do cabeçalho está definida com o nome **cabecalho** e já possui um conteúdo padrão. A área central está defina com o nome de **principal** e nessa área será exibido o conteúdo das páginas.

Listagem 1
template.xhtml

```
01:         <p:layout fullPage="true">
02:             <p:layoutUnit position="north" size="100"
header="Cabeçalho">
03:                 <ui:insert name="cabecalho">
04:                     <ui:include src=”/menu_superior.xhtml”/>
05:                 </ui:insert>
06:             </p:layoutUnit>
07:             <p:layoutUnit position="center">
08:                 <ui:insert name="principal"> </ui:insert>
09:             </p:layoutUnit>
10:         </p:layout>
```

No cabeçalho do *template* a *tag* ***ui:include****,* juntamente com o atributo ***src****,* é utilizada para definir o conteúdo do arquivo **menu_superior.xhtml** como padrão para a área de substituição **cabecalho**.

O trecho do arquivo cadastro.xhtml representado na listagem 2, referente ao formulário de abertura de solicitação, usa como base o arquivo **template.xhtml**. A *tag ui:composition* define o relacionamento de composição entre a tela e o *template*. No arquivo cadastro.xhtml (listagem 2), a única área sobrescrita do *template* é a área **principal**. A *tag ui:include* agora é utilizada para sobrepor a definição de *ui:insert*. No resto da tela, as definições são mantidas.

Listagem 2
Trecho do arquivo cadastro.xhtml

```
01: <ui:composition template="/template.xhtml">
02:    <ui:define name="principal"><h:form id="formAberturaSolicitacao" >
03:        <p:panel header="Abertura de Solicitação">
04:            <p:panelGrid>
05:                <p:row><p:column>
06:                    Solicitante:
07:                    </p:column><p:column>
08:                        #{ SSTIAberturaSolicMB.solicitante.nome}
09:                    </p:column></p:row>
10:                        <!—Trecho omitido -->
11:                <p:commandButton action="#{SSTIAberturaSolicMB.save()}"
icon="ui-icon-disk" value="Salvar" ajax=”false”/>
12:     </h:form></ui:define>
13: </ui:composition>
```

O JSF faz uso da *Expression Language* (EL), que permite vincular valores de atributos ou métodos do *Managed Bean* aos componentes visuais do *JavaServer Faces* contidos nas páginas Web da aplicação.

O *ManagedBean* SSTIAberturaSolicitacaoMB (listagem 3) é responsável por salvar, obter as informações necessárias para exibir o formulário de cadastro e persistir a solicitação no banco de dados. Nessa listagem, os atributos utilizados nos métodos do *ManagedBean* serão associados e utilizados na *view* com o uso da *EL*. Os atributos das linhas de 07 a 10, referem-se aos objetos das classes DAO (*Data Access Objects*), que são utilizados pelos métodos do *ManagedBean* para obter e persistir objetos no banco de dados.

Listagem 3
***ManagedBean* SSTIAberturaSolicitacaoMB.java - atributos**

```
01: @ManagedBean(name="SSTIAberturaSolicMB")
02: @RequestScope
03: public class SSTIAberturaSolicMB {
04:    private SSTISolicitacao current;
05:    private Usuario solicitante;
06:    //DAO's
07:    @Autowired
08:    private SSTISolicitacaoDao solicitacaoDao;
09:    @Autowired
10:    private SSTISetorDao setorDao;
11:    /* métodos omitidos*/ }
```

A Listagem 4 apresenta os métodos do *ManagedBean* SSTIAberturaSolicMB, que são utilizados para realizar a obtenção dos dados iniciais necessários para o preenchimento do formulário de abertura de solicitação. Essa listagem mostra que o método init() está marcado com a anotação @PostConstruct, utilizada no método que precisa ser executado logo após as injeções de dependência terem sido feitas para

executar qualquer inicialização necessária do *ManagedBean* (similar ao método *construct*). Quando o usuário acessar a *view* de Abertura de Solicitação, o JSF irá carregar o arquivo cadastro.xhtml (listagem 2) e também irá instanciar o *ManagedBean* SSTIAberturaSolicMB, executando o método que instancia um novo objeto da classe *SSTISolicitacao* (listagem 4, linha 03), obtendo o usuário que logado no sistema que é o solicitante (linhas de 09 a 15). Em seguida, é obtido o parâmetro de *status* inicial da solicitação (linhas de 06 a 08) e preenche as listas de unidades, setores e categorias (linhas 16 a 19).

Listagem 4
***ManagedBean* SSTIAberturaSolicitacaoMB.java - métodos de inicialização**

```
01: @PostConstruct
02: public void init(){
03:     current = new SSTISolicitacao();
04:     getUsuario(); getParametros(); atualizarLista();
05: }
06: public void getParametros(){
07:     current.setStatus(parametroDao.getParametro());
08: }
09: public void getUsuario(){
10:     ExternalContext ec =
FacesContext.getCurrentInstance().getExternalContext();
11:     Map sessionMap = ec.getSessionMap();
12:     if(sessionMap.containsKey("idUsuario")){
13:          Long id =
Long.parseLong(String.valueOf(sessionMap.get("idUsuario")));
14:          current.setSolicitante(usuarioDao.getRegistro(id));}
15: }
16: public void atualizarLista(){
17:     unidades = unidadeDao.getRegistros(); setores =
setorDao.getRegistros();
18:     categorias = categoriaDao.getRegistros();
19: }
```

A Listagem 5 apresenta os métodos executados após o usuário clicar no botão **Salvar**, gerando um *postback* para o JSF. Ao executar a EL **#{SSTIAberturaSolicMB.save()}** do cadastro de Abertura de Solicitação, o atributo *action* associa o botão **Salvar** ao método **save()** do *ManagedBean* SSTIAberturaSolicitacaoMB (linha 09). Esse método faz a validação dos dados inseridos pelo usuário, salva os dados da solicitação no banco de dados e, caso haja algum problema de validação, o método não faz a inserção, exibindo mensagens com informações sobre os erros encontrados, para que o usuário possa corrigi-las.

Logo após a solicitação ter sido registrada, o sistema informa ao usuário que os dados foram salvos e apresentando o formulário de abertura de solicitação vazio, para que possa ser cadastrada uma nova solicitação.

Listagem 5
ManagedBean* SSTIAberturaSolicMB – Métodos de *postback

```
01: public boolean validarDados(){
02:       if(current.getAssunto() == null || current.getAssunto().trim().isEmpty()){
03:          fc.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
"Informe o assunto", null));
04              return false;
05:       }
06:       // Outras verificações omitidas //
07:       return true;
08:   }
09:   public String save(){
10:       if(validarDados()==false){
11          return null;
12:       }
14:       if(currentDao.insert(current)){
15:          fc.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
"Sua solicitação foi enviada ao setor de TI." , null));
16:       } else {
17:          fc.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
"Ocorreu um erro interno na aplicação, por favor entre em contato com o setor de TI.",  null));
18:       }
19:       Return “/restrito/sistemaSolicitacao/solicitação/mainSolicitacao”;
20:   }
```

Como dito anteriormente, essa implementação representa a funcionalidade de Abertura de Solicitações, utilizada aqui para representar como a implementação do sistema foi feita utilizando as tecnologias envolvidas. As demais funcionalidades do sistema foram realizadas utilizando a mesma abordagem.

5 CONCLUSÃO

Foi apresentado o Sistema de Gestão de Solicitações desenvolvido para auxiliar no acompanhamento, avaliação de desempenho, organização e na documentação das atividades desempenhadas pelos funcionários do setor TI do Centro de Ensino Superior de Juiz de Fora (CES/JF). Com o sistema, a perspectiva da instituição é conseguir diminuir o tempo em que as máquinas em manutenção ficavam paradas no setor de TI. A principal dificuldade a ser enfrentada será mudar a cultura dos funcionários do setor, que não estão acostumados a cadastrar as atividades e solicitações realizadas.

Uma das contribuições deste trabalho refere-se à utilização de tecnologias não exploradas nas disciplinas do curso de Bacharelado em Sistemas de Informação do CES/JF, muito em virtude de serem tecnologias novas e ainda não abordadas

plenamente nos cursos de graduação. Isso se reflete na pouca bibliografia técnica encontrada sobre os assuntos abordados neste trabalho. Desta forma, este trabalho utilizou-se de diversos *sites* e tutoriais disponíveis na Web.

As principais dificuldades encontradas no desenvolvimento deste trabalho foram justamente a falta de documentação e de literatura técnica sobre as tecnologias utilizadas e, principalmente, na utilização conjunta dessas tecnologias na construção de um sistema de informação.

Como trabalhos futuros, e a partir de uma avaliação inicial das funcionalidades oferecidas pelo sistema, sente-se a necessidade de implementar uma funcionalidade de envio por e-mail quando alguma alteração ocorrer nas atividades e solicitações associadas aos solucionadores.

Ainda, adição de outros filtros, como por unidade, categoria, setor e solicitante, auxiliará na identificação solicitações reincidentes, que podem ser consequência de outros fatores, indicando a real causa da reincidência. Resolvendo o problema de solicitações reincidentes, os funcionários terão mais tempo para focar em melhorias que podem e precisam ser realizadas.

REFERÊNCIAS

BAUER, Christian; KING, Gavin. Hibernate in Action. Greenwich: Manning, 2005.

BRIGATTO, Pedro E. Cunha. A renovação do JSF – Parte 1: Conheça as principais novidades trazidas com a versão 2.2 desta especificação. DevMedia, Java Magazine, Edição 119, 2013, p. 18-33.

COUTINHO, Paulo César M. N. A.. Help desk com JavaServerFaces 2: Uma Aplicação de atendimento virtual com JSF 2 e Primefaces. DevMedia, Java magazine, Edição 118, 2013, p. 14-30.

ÇIVICI Çağatay. PRIMEFACES: User Guide 5.0. Disponível em: http://www.primefaces.org/docs/guide/primefaces_user_guide_5_0.pdf. Acesso em 25 de maio de 2014.

GONÇALVES, Edson. Desenvolvendo aplicacoes WEB com JSP, Servlets, Javaserver, Faces, Hibernate, EJB 3 persistence e AJAX. Rio de Janeiro: Ciência Moderna, 2007.

GONÇALVES, Edson. Desenvolvendo Relatórios Profissionais com IReport para NetBeans IDE. Rio de Janeiro: Ciência Moderna, 2009.

HIBERNATE. Hibernate: Relational Persistence for idiomatic Java. Disponível em: http://docs.jboss.org/hibernate/orm/4.3/manual/en-US/html_single. acesso em: 02 de junho de 2014.

IREPORT. IReport Designer: The Report Development Tool for JasperReports and JasperReports Server. Disponível em: http://community.jaspersoft.com/project/ireport-designer. Acesso em: 26 de maio de 2014.

JASPERREPORT. JavaReports Library: Open Source Java Reporting Library. Disponível em: http://community.jaspersoft.com/project/jasperreports-library. Acesso em: 26 de maio de 2014.

JENDROCK, Erick; CERVERA-NAVARRO, Ricardo; EVANS, Ian; HAASE, Kim;

MARKITO, William. The Java EE 7 Tutorial, 2014. Disponível em: http://docs.oracle.com/javaee/7/tutorial/doc/home.htm. Acesso em 11 de maio de 2014.

ORACLE b. Introdução ao JavaServer Faces 2.x. Disponível em: https://netbeans.org/kb/docs/web/jsf20-intro_pt_BR.html. Acesso em: 11 de maio de 2014.

ORACLE d. The Java EE 7 Tutorial: Release 7 for Java EE Platform, maio de 2014. Disponível em: doc/javaeetutorial7.pdf, p. 407 Acesso em 16 de maio de 2014.

ORACLE e. Java Server Faces. Disponível em: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html. Acesso em: 16 de maio de 2014.

SOUZA, Vitor E. Silva. Novidades do Hibernate 4: Registro de serviços, bancos de dados multi-inquilinos e IDs naturais. DevMedia, Java Magazine, Edição 116, 2013, p. 30-39.