Universidade Católica de Brasília
Pró-Reitoria de Pós-Graduação
Curso de Especialização em Sistemas Orientados a Objetos
Trabalho de Conclusão de Curso

Especialização

Sistema de Pesquisa de Clima Organizacional

Autor: Edmilson Faria Rodrigues
Orientadora: Profª. Msc. Leila de Fátima S. Carvalho.

Brasília 2009

Edmilson Faria Rodrigues

Sistema de Pesquisa de Clima Organizacional

Monografia submetida ao Programa de Pós-Graduação Lato Sensu em Sistemas Orientados a Objetos da Universidade Católica de Brasília para obtenção do título de Especialista.

Orientadora: Profª. Msc. Leila de Fátima de Sousa Carvalho

TERMO DE APROVAÇÃO

Monografia defendida e aprovada como requisito parcial para a obtenção do título de especialista em Sistemas Orientados a Objetos, defendida e aprovada, em 28 de maio de 2009, pela banca examinadora constituída por:

_____

Profª. MSc. Leila de Fátima de Sousa Carvalho
Orientadora

_____

Profº. Dra. Maristela Holanda

_____

Profº. Esp. Osvaldeir Aurélio Elias

Dedicatórias

A Deus, por me guiar e me amparar sempre, e principalmente nos momentos mais difíceis da minha vida.

Aos meus pais por terem, desde muito cedo, ensinado a importância da educação, explicando sempre que era um bem inesgotável, o qual nunca poderia ser subtraído.

À Fabíola, minha namorada, pela dedicação e compreensão com que esteve ao meu lado durante os momentos em que tive que preteri-la em favor do presente trabalho, pelo ambiente sereno e agradável do seu lar, no qual fiz os maiores progressos no desenvolvimento do sistema: os meus sinceros agradecimentos pelo zelo, paciência, alegria e bom humor com os quais me acompanhou nos momentos mais intrincados do processo de desenvolvimento.

Agradecimentos

Ao espírito colaborativo dos colegas do Curso de Sistemas Orientados a Objeto, com quem tive todo o prazer em trocar experiências e saberes díspares; expressar opiniões diversas sobre os assuntos em causa e atualizar conhecimentos no campo da educação.

A todos os professores que contribuíram decisivamente para a minha, e nossa, formação acadêmica, profissional e pessoal.

À Profª MsC. Leila de Fátima de Sousa Carvalho,

Pelo apoio moral e estímulo que me permitiram levar até ao fim este trabalho de projeto de Pós Graduação.

EPÍGRAFE

“Morre lentamente quem abandona um projeto antes de iniciá-lo, não pergunta sobre um assunto que desconhece ou não responde quando lhe indagam sobre algo que sabe. Evitemos a morte em doses suaves, recordando sempre que estar vivo exige um esforço maior que o simples fato de respirar”.

Pablo Neruda

Resumo

O presente trabalho apresenta o processo de desenvolvimento do sistema de Pesquisa de Clima Organizacional no âmbito do Tribunal de Contas da União (TCU) e descreve a análise, o projeto, a implementação e implantação do sistema, bem como discute padrões de projetos, análise da arquitetura e tecnologias utilizadas nesse desenvolvimento, com o fim de propor melhorias tanto ao processo de desenvolvimento de sistemas no TCU, quanto à sua atual arquitetura de sistemas, tornando o desenvolvimento mais ágil, menos sujeito a falhas e mais aderente às mudanças.

Abstract

This paper presents the development process of the system of Organizational Climate under the Brasilian Court of Audit (TCU) and describes the analysis, design, implementation and deployment of the system and discusses design patterns, analysis of architecture and technologies used in this development, in order to propose improvements to both the process of developing systems in the TCU, as to the architecture of its current systems, making development faster, less prone to failure and more adherent to changes.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de Siglas

SIGLA	SIGNIFICADO	TRADUÇÃO
AJAX	*Assincronous JavaScript*	
APEX	*Oracle Application Express*	Aplicação Expressa Oracle
API	*Application Programming Interface*	Interface de Programação de Aplicativos
BSC	*Balanced Scorecard*	Sistema de Medição e Controle Gerencial
DAO	*Data Access Object*	Objeto de Acesso a Dados
DTO	*Data Transfer Object*	Objeto de Transferência de Dados
HTTP	*HiperText Tranfer Protocol*	Protocolo de Transferência de HiperTexto
IDE	*Integrated Development Enviroment*	Ambiente de Desenvolvimento Integrado
J2EE	*Java 2 Enterprise Edition*	Java Edição Empresarial
jBPM	*Java Business Process Modelling*	Modelagem de Processo de Negócio Java
JPA	*Java Persistence API*	API de Persistência Java
JSP	*JAVA Server Pages*	Páginas de Servidor Java
JSF	*Java Server Faces*	
MVC	*Model View Controller*	Modelo de 3 Camadas Modelo Visão Controlador
PET	*Plano Estratégico do TCU*	Plano Estratégico do TCU
POJO	*Plain Java Old Objects*	Objetos Java Puros
RUP	*Rational Unified Process*	Processo Unificado Rational
SVN	*Serviço de Negócio*	
UML	*Unified Modeling Language*	Linguagem de Modelagem Unificada
TCU	*Tribunal de Contas da União*	
TDD	*Test Driven Development*	Desenvolvimento Orientado a Testes
XML	*Extensible Markup Language*	Linguagem de Marcação Extensível
XP	*Extremming Programming*	Programação Extrema

1. Introdução

1.1. Motivação

O Clima Organizacional é a percepção global das pessoas a respeito de seu ambiente de trabalho, o qual influencia o comportamento profissional e pode afetar o desempenho da organização

O modelo de Gestão do Clima organizacional leva em consideração as atuais práticas de gestão do TCU, que consiste em um planejamento estratégico, o qual é formado por um conjunto de documentos que definem a estratégia de atuação do TCU desde o seu objetivo geral e sua missão, passando por seus macro-processos, até o plano diretor de cada subunidade do TCU

O Plano Estratégico (PET) contempla as principais orientações do Tribunal de Contas da União. Constitui importante instrumento gerencial na busca por resultados mais efetivos para a sociedade na medida em que expressa, traduz e comunica a estratégia de atuação institucional. O Plano Estratégico é formado por três novos elementos: o mapa estratégico, os macroprocessos e as competências organizacionais.

O mapa estratégico, construído com base na metodologia do *Balanced Scorecard* (BSC) [Wikipedia-a, 2009], tem por objetivo traduzir e comunicar a estratégia da organização. Constitui importante instrumento de inserção institucional na medida em que possibilita a dirigentes e servidores a real percepção de sua parcela de contribuição no alcance dos resultados desejados e no cumprimento da missão do Tribunal de Contas da União [TCU, 2006]. No nível das unidades essas ações de planejamento se desdobram no plano diretor da unidade, o qual contém as ações a serem implementadas naquele âmbito, o responsável e o prazo para sua execução.

O modelo de gestão do clima tem por objetivo inserir-se dentro dessas práticas de planejamento institucional e alimentar na medida do possível este sistema de indicadores**,** de forma a contribuir na gestão do Tribunal.

Assim, é de fundamental importância alinhar as ações dos diversos servidores do TCU com aquelas apontadas no Mapa Estratégico e no Plano Estratégico, pois sem esse alinhamento e sem a participação e as condições favoráveis ao comprometimento de cada servidor com o objetivo, a missão e o negócio do TCU, não é possível atingir as metas estabelecidas.

Além disso, a gestão do clima organizacional:

- Produz alto desempenho organizacional e pessoal sustentável;
- Está interligada sistemicamente a outras ações de gestão de pessoas;
- Alimenta o sistema de planejamento e gestão;
- Cria e mantém canal de comunicação com os servidores;
- Gera indicadores diversas unidades:
    - Estrutura logística;
    - Práticas de gestão de pessoas;
    - Treinamento oferecido;
    - Comunicação;
    - Soluções de TI;
    - Avaliação da gestão
- Oferece subsídio para a tomada de decisão gerencial
    - Alocação de recursos;
    - Planejamento de ações
- Aponta oportunidades de melhoria

Para o sucesso da gestão de pesquisa de clima, é fundamental dispor de uma ferramenta que possibilite:

- O preenchimento *on-line* da pesquisa de clima organizacional, estimulando assim a participação dos servidores através da facilidade de acesso;
- O preenchimento anônimo do questionário, uma vez que a identificação do servidor pode deixá-lo constrangido e inibir sua participação;
- A emissão de relatórios que permita a consolidação das respostas na forma de indicadores do grau de satisfação dos servidores de acordo com as diversas dimensões ou fatores de pesquisa (construção de relacionamento, gestão por resultados, engajamento, etc..), com a unidade do TCU a que pertence o servidor ou com outros critérios que vierem a ser definidos

1.2. Objetivo Geral

Desenvolvimento de um sistema de pesquisa de clima organizacional, em plataforma J2EE, e segundo a arquitetura de sistemas atual do TCU, que atenda aos requisitos dispostos na seção anterior.

1.3. Objetivos Específicos.

Os objetivos específicos apresentam, com maior grau de detalhamento, aquilo que se busca alcançar com o presente trabalho e, portanto, norteiam todo o processo de desenvolvimento. No capítulo 3, esses objetivos são retomados de modo a apresentar o resultado do trabalho no que tange a cada um desses objetivos:

a) Testar e adquirir familiaridade com a aplicação do *framework* de camada de aplicação JSF;

b) Testar e adquirir familiaridade com o mecanismo de mapeamento objeto relacional Hibernate com Anotações;

c) Testar a nova versão do Eclipse, versão 3.4 (“Ganymede”), com WTP nativo e o *plugin* “Dali” para desenvolvimento de mapeamento objeto relacional;

d) Testar a integração do conjunto de ferramentas no ambiente web;

e) Empregar boas práticas e padrões de projeto*: Data Transfer Object* (DTO), *Application Controller, Singleton, Data Access Object* (DAO), *Façade, Business Object, Model-View-Controller* (MVC), Fábrica Abstrata e Método Fábrica.

f) Validar um modelo arquitetural de 5 camadas J2EE;

g) Implementar uma suíte de testes unitários, a fim de proporcionar uma base para Integração Contínua e futura migração para o Desenvolvimento Orientado a Testes (TDD);

h) Validar o emprego de um processo ágil de desenvolvimento na implementação do protótipo, customizando práticas do XP e do RUP.

1.4. Organização da Monografia

Este documento está organizado conforme a seguir:

- O Capítulo 1, Introdução, fornece uma visão geral do que será estudado neste trabalho. Permite ao leitor ter um primeiro contato com o assunto, mostrando a importância e expondo as motivações e objetivos que levaram ao desenvolvimento do mesmo.

- O Capítulo 2 trata do projeto do Sistema de Pesquisa de Clima Organizacional, descreve e especifica o sistema e apresenta o Protótipo. Apresenta as necessidades do usuário, descreve o processo de pesquisa de clima organizacional, apresenta a especificação dos casos de uso críticos para validação da arquitetura, bem como apresenta o protótipo para esses mesmos casos de uso.
- O Capítulo 3, Arquitetura de Software, descreve a arquitetura de sistemas do TCU apresentando seus principais componentes e a sua distribuição segundo a divisão de cinco camadas J2EE. A seguir, este capítulo detalha a aplicação de alguns dos padrões de projeto dentro do presente trabalho, trazendo uma breve descrição e discussão do padrão, bem como uma apresentação da forma pela qual ele foi implementado no presente sistema.
- O Capítulo 4, Conclusão e Trabalhos Futuros, descreve os pontos positivos e negativos das tecnologias e processos empregados no desenvolvimento do Protótipo, bem como as dificuldades enfrentadas e aprendizados absorvidos no decorrer de seu desenvolvimento. A seguir, este capítulo apresenta uma esperada continuação do presente trabalho. Os trabalhos futuros descritos são o resultado da experiência obtida visando corrigir erros cometidos, ou visando o aprofundamento em algumas das tecnologias com a qual se teve contato durante o presente trabalho de pesquisa.

2. Sistema de Pesquisa de Clima Organizacional

2.1. Documento de Visão

2.1.1. Escopo

O escopo deste projeto é a criação de um sistema informatizado que permita ao usuário gestor o cadastramento de pesquisa(s) de clima, bem como seus grupos e subgrupos de perguntas. Uma vez cadastrada a pesquisa, o usuário comum pode responder às questões de maneira anônima, autenticando-se por meio de fichas de autenticação geradas aleatoriamente pelo sistema.

2.1.2. Não Escopo

Não faz parte do escopo deste projeto:

- a criação de relatórios gerenciais não apresentados no presente documento;
- a distribuição das fichas de autenticação ao usuário, a qual deverá ser feita manualmente;
- o suporte à definição do nível de segurança das informações em tempo de execução, pelo usuário gestor.

Tabela 2.1 – Atores potenciais do Sistema.

Ator	Representa	Papel
Gestor	Servidor lotado no Serviço de Admissão e de Avaliação de Desempenho (SEDEM)	Responder pela gestão do sistema, providenciando o cadastramento das pesquisas, criação de grupo e/ou subgrupo e suas questões correspondentes, bem como a atualização desses dados e a emissão de relatórios analíticos e sintéticos.
Responsável Unidade	Servidor designado para chefe de alguma das secretarias do Tribunal de Contas da União.	Obter informações relativas ao grau de participação dos servidores da sua unidade no evento de pesquisa, bem como emitir relatórios que mostrem a

		distribuição das respostas dos servidores lotados na sua unidade.
Servidor Comum	Servidor respondente da pesquisa de clima organizacional.	Responder ao questionário e enviá-lo dentro do prazo de participação na pesquisa de clima.

Tabela 2.2 – Posicionamento do Produto.

Para	Área de gestão de pessoas, e secretários do TCU.
Que	Necessitam aferir o clima organizacional, que é percepção global das pessoas a respeito de seu ambiente de trabalho capaz de influenciar o comportamento profissional e afetar o desempenho da organização.
O Sistema de Pesquisa	É um aplicativo para automatização do processo de criação de pesquisa de clima, preenchimento de questionários e verificação dos resultados.
Que	Gera indicadores para diversas unidades referentes à estrutura logística, práticas de gestão de pessoas, treinamento oferecido, comunicação, soluções de TI e avaliação da gestão; além de oferecer subsídio para a tomada de decisão gerencial no que tange à alocação de recursos e ao Planejamento de ações
Diferentemente	Do sistema utilizado na última pesquisa que era uma solução que fazia parte de um serviço de consultoria contratado pela Secretaria de Gestão de Pessoas, e que foi descontinuado com o fim deste serviço.

Tabela 2.3 – Sentença do Problema.

O problema de	Tornar a pesquisa de clima organizacional disponível a todos os servidores do TCU, inclusive nas suas subunidades e nas demais unidades regionais da federação
Afeta (quem)	Todos os servidores.
O impacto disto é	Se poucos servidores participam da pesquisa, então a tomada de decisões e a avaliação do clima organizacional ficam prejudicadas, pela falta de representatividade dos dados
Uma solução de sucesso permitirá	Criar um sistema disponível na internet, acessível a qualquer servidor do TCU.

Tabela 2.4 – Sentença do Problema.

O problema de	Falta de confiança no caráter anônimo da pesquisa
Afeta	Todo o TCU
O impacto dele é	Baixa participação no evento de pesquisa de clima organizacional
Uma solução de sucesso pode ser	Criar um módulo de geração e validação de fichas de acesso, que serão distribuídas aleatoriamente aos servidores

Tabela 2.5 – Sentença do Problema.

O problema de	Falta da percepção pelos secretários do clima organizacional de sua unidade
Afeta	Secretários.
O impacto dele é	Erros na tomada de decisões referentes à alocação de pessoas, gerenciamento de recursos de TI, treinamento, etc.
Uma solução de sucesso pode ser	Ter um mecanismo de geração de relatórios acessível aos secretários e agrupados segundo os critérios mais relevantes para sua unidade

Tabela 2.6 – Sentença do Problema.

O problema de	Falta de estímulo para se responder uma quantidade enorme de perguntas de uma só vez
Afeta	Todo o TCU
O impacto dele é	Baixa participação no evento de pesquisa de clima organizacional
Uma solução de sucesso pode ser	Possibilitar o preenchimento parcial do questionário para que seja concluído posteriormente, e possibilitar a visão geral da quantidade de questões respondidas e a responder

Tabela 2.7 – Lista de Riscos.

Risco	Impacto	Situação	Materialização	Plano de Contingência
Utilização da tecnologia JSF com o qual a equipe de desenvolvimento não tem experiência	Alto	Resolvido	Dificuldade no desenvolvimento e atraso na entrega.	Caso não seja possível o desenvolvimento do sistema até o mês de setembro/2009 (data da realização da pesquisa), uma solução provisória utilizando a tecnologia APEX será utilizada.
Dificuldade em conciliar a agenda de analista de requisitos e gestores do sistema.	Médio	Resolvido	Demora na aprovação de artefatos e atraso na entrega.	Desenvolvimento fortemente voltado à prototipação. A tecnologia JSF e o uso de IDEs favorecerá o desenvolvimento de protótipos na própria reunião e as próprias atas de reunião servirão como base para elaboração dos casos de uso, cuja implementação prescinde da assinatura deste, já que a ata de reunião subsidia este trabalho.

Tabela 2.8 – Funcionalidades do Sistema.

Número	Nome	Descrição	Prioridade
1	Validar ficha de autenticação	Este caso de uso tem como finalidade validar as informações contidas na ficha de autenticação do usuário autorizando o seu acesso ao sistema.	Alta
2	Responder Questionário	Este caso de uso tem como objetivo: • Apresentar o questionário da pesquisa de clima organizacional • Atualizar as respostas do usuário • Fornecer mecanismo de preenchimento parcial do questionário.	Alta
3	Gerar fichas de autenticação	Este caso de uso tem como objetivo descrever o processo de geração de fichas individuais impressas contendo *login* e senha que servirão para que o usuário acesse o sistema de maneira anônima.	Média

4	Consultar número de Questões Respondidas	Este caso de uso tem como finalidade permitir o acompanhamento do nível de participação no evento de pesquisa de clima, bem como do número de questões respondidas até o momento por cada participante.	Média
5	Emitir Relatório de Distribuição das Respostas	Este caso de uso tem como finalidade apresentar consolidação das respostas dos participantes por critério da pesquisa, por unidade pesquisada, bem apresentar o resultado dos cálculos do índice de respostas favoráveis, do índice geral de percepção da gestão na unidade analisada e do índice de satisfação na unidade analisada	Média
6	Verificar questões respondidas	Este caso de uso tem como finalidade apresentar ao usuário um mapa das questões agrupadas por dimensão com a indicação se a pergunta foi respondida, a fim de possibilitar o usuário reiniciar o preenchimento do questionário a partir de qualquer pergunta ou dimensão que lhe for mais conveniente	Baixa

2.2. Diagrama de Casos de Uso

A figura a seguir mostra o diagrama de casos de uso do sistema. No presente trabalho foi feita a implementação e implantação dos casos de uso “Responder Questionário” e “Validar Ficha de Autenticação”, por serem os casos de uso de maior risco e maior importância para o projeto, e que serão usados para validação da arquitetura.

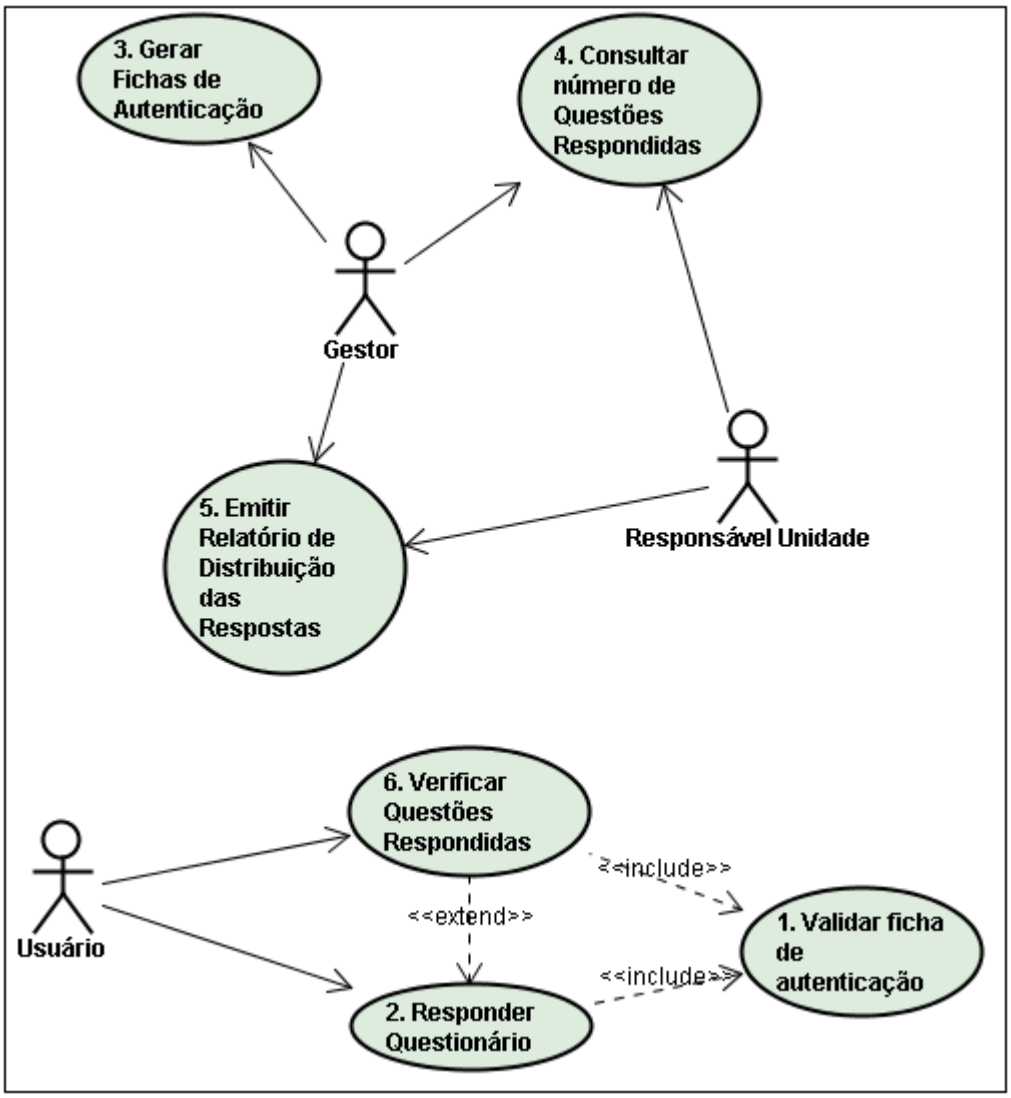

Figura 2.1 – Diagrama de Caso de Uso do Negócio.

2.2.1. Detalhamento de Casos de Uso

O detalhamento dos casos de uso foi feito segundo o documento padrão do TCU, portanto os seus campos e sua estrutura atendem a esse padrão estabelecido pelo Serviço de Qualidade e Padronização de Sistemas, por essa razão, estão apresentados na forma de anexos no fim do presente trabalho: o Anexo I apresentada o detalhamento do caso de uso "Responder Questionário" e o anexo II apresenta o detalhamento do caso de uso "Validar Ficha de Autenticação".

2.2.2. Diagrama de atividades

A figura 2.2 demonstra o fluxo de atividade do processo “Realizar Pesquisa de Clima Organizacional”. Após o cadastramento da pesquisa, o usuário cadastra as questões e o seu respectivo grupo (ou dimensão). A seguir, a partir da data definida para início da pesquisa é possível:

- ao gestor: emitir relatórios de distribuição de perguntas e acompanhar o número de perguntas respondidas;
- ao usuário: responder o questionário e verificar a distribuição e número de questões respondidas/ a responder;

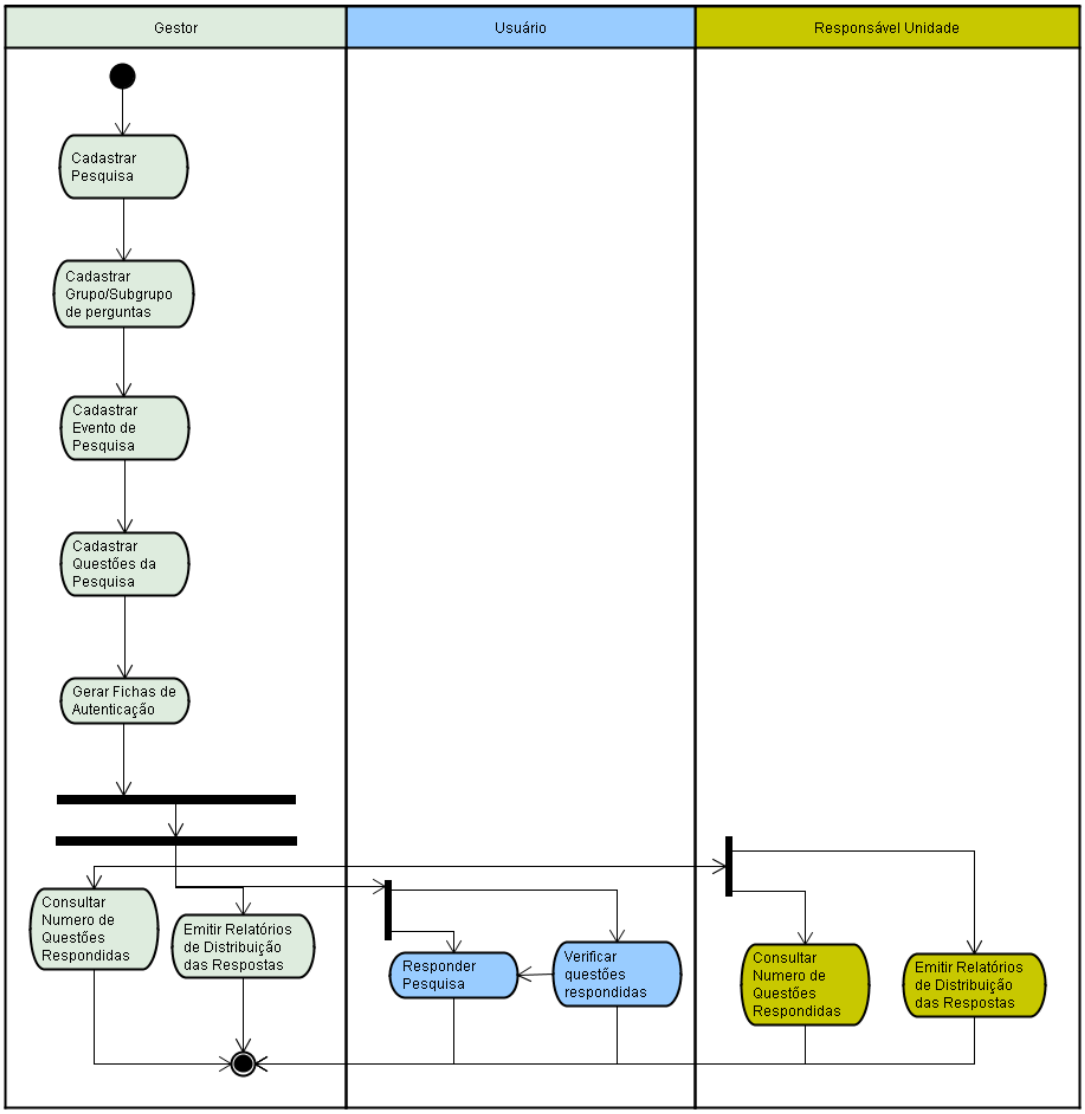

Figura 2.2 – Diagrama de Atividades do Processo.

2.2.3. Diagramas de Sequência

A figura 2.3 demonstra a sequência de chamadas que ocorre durante uma autenticação mal-sucedida do usuário. A classe Segurança, estende a classe *Filter*, da API Servlet, de modo que toda requisição passa pelo método *doFilter*() dessa classe, a qual verifica se o usuário está logado (possui uma sessão ativa), em caso positivo passa controle para um eventual próximo filtro [Marty Hall & Larry Brown, 2003] .

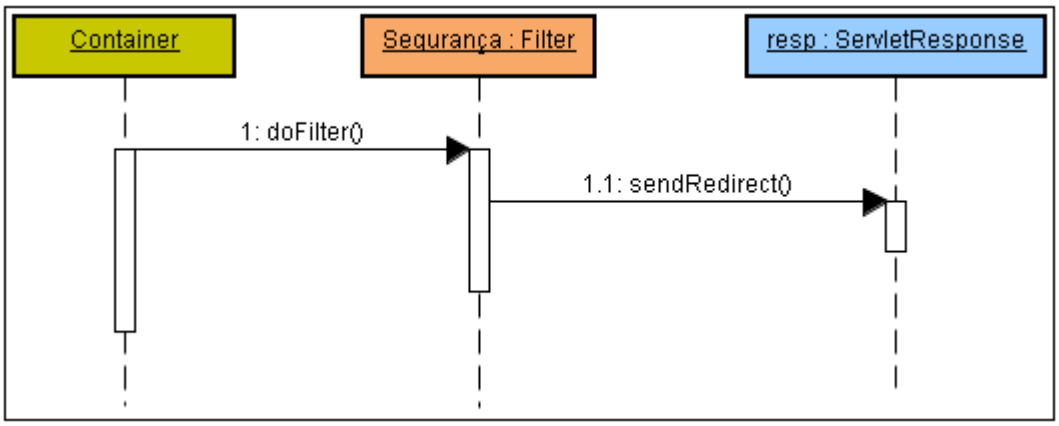

Figura 2.3 – Diagrama de Sequência de Validação Mal-Sucedida do Usuário.

A figura 2.4 demonstra a sequência de chamadas que ocorre durante uma autenticação bem-sucedida do usuário. A implementação JSF, por meio da FacesServlet encaminha a requisição para o método validarUsuário do bean ActSegurança vinculado ao botão na tela de login do usuário. A classe ActSegurança encaminha o pedido a CtlPesquisa, a qual faz o tratamento de geração hash a partir da senha informada e encaminha para a Svn-Façade apropriada. A SvnPesquisa solicita um novo UsuárioDAO para a DAOFactory de acordo com o tipo de persistência utilizada. Após obtida a referência a implementação apropriada da interface UsuárioDAO, a SvnPesquisa solicita a validação do usuário. A classe que implementa UsuárioDAO faz os devidos acessos à persistência utilizada para fazer a validação do usuário.

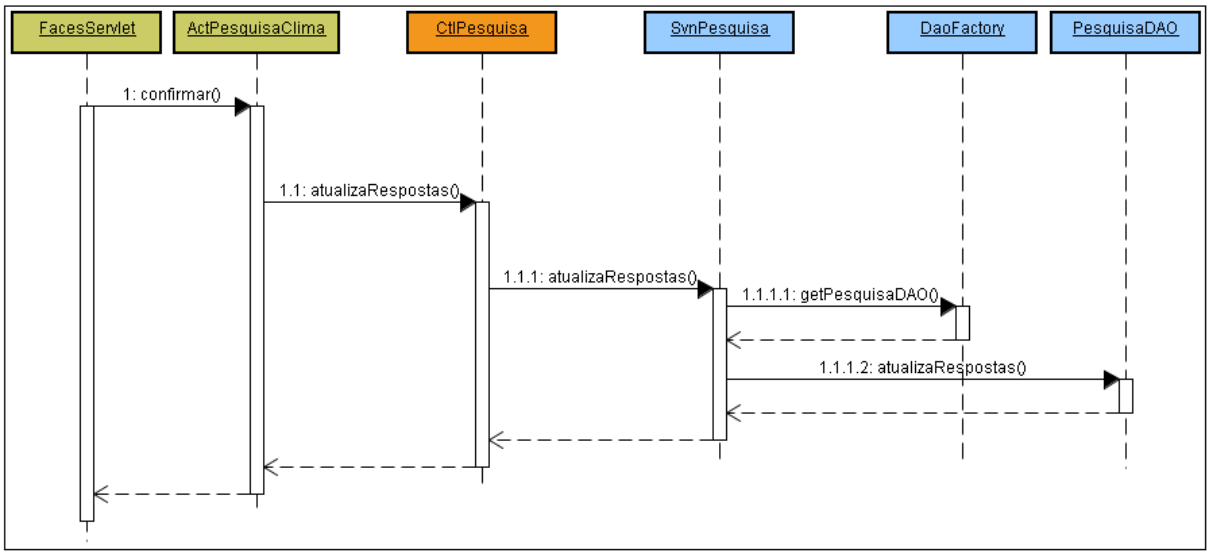

Figura 2.4 – Diagrama de Sequência de Validação Bem-Sucedida do Usuário.

A figura 2.5 demonstra a sequência de chamadas que ocorre durante uma atualização de resposta ou criação de nova resposta. A implementação JSF, por meio da *FacesServlet* encaminha a requisição para o método atualizaResposta do *bean* ActPesquisaClima vinculado ao botão de navegação na tabela que apresenta as questões ao usuário. A classe ActPesquisaClima encaminha o pedido a CtlPesquisa, a qual o encaminha para a Svn apropriada. A SvnPesquisa solicita um novo objeto que implementa a interface PesquisaDAO para a DAOFactory de acordo com o tipo de persistência utilizada. Depois de obtida a referência a implementação apropriada da interface PesquisaDAO, a SvnPesquisa solicita a atualização da resposta. A classe que implementa PesquisaDAO faz os devidos acessos à persistência utilizada para fazer a atualização da resposta do usuário.

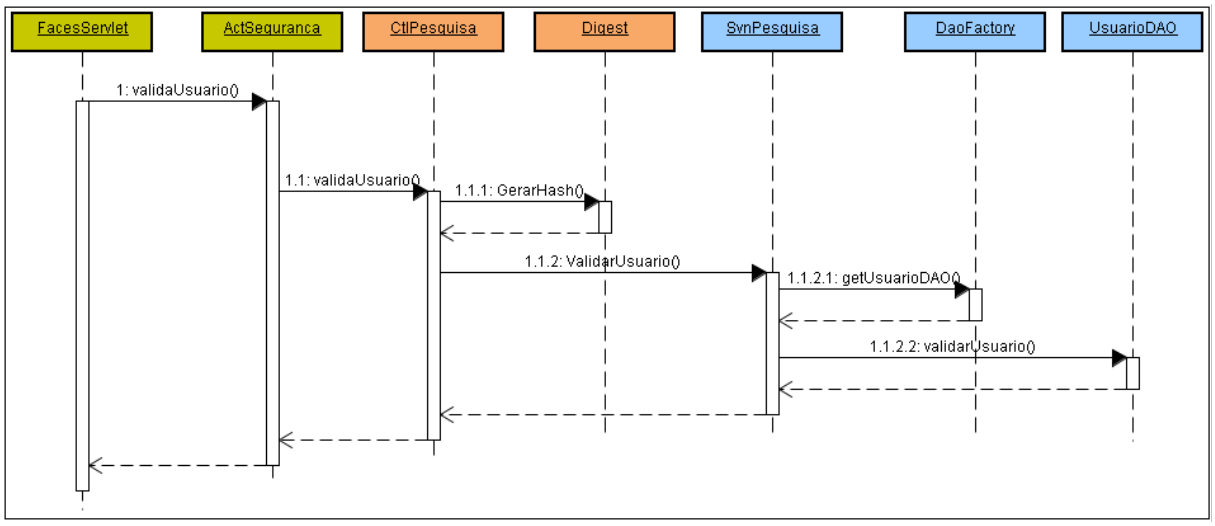

Figura 2.5 – Diagrama de Sequência de Atualização de Resposta do Usuário.

2.2.4. Apresentação do Protótipo

A figura 2.6 mostra a tela de acesso do usuário ao sistema de Pesquisa de Clima Organizacional. Esta tela contém algumas orientações sobre a unidade gestora do sistema e informações gerais sobre a pesquisa de Clima Organizacional. O usuário deve informar o *login* de usuário e a senha conforme descrito na sua ficha de Autenticação.

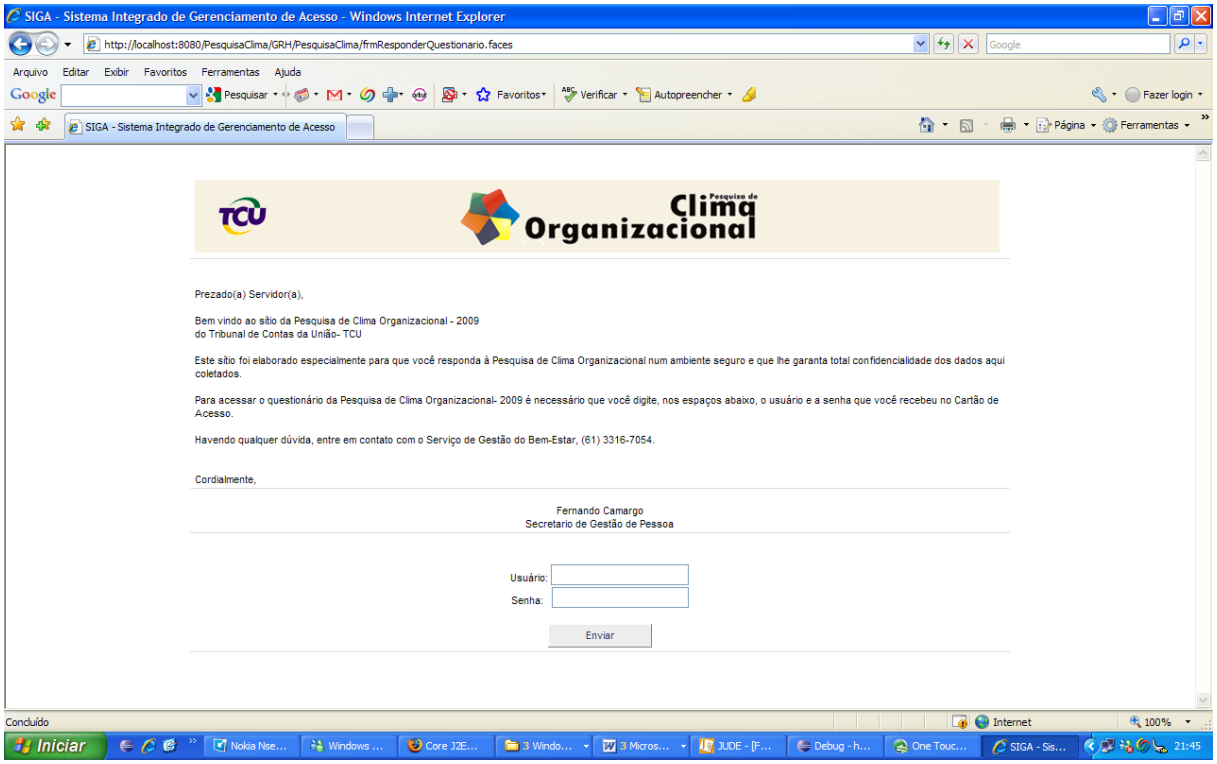

Figura 2.6 – Autenticação do Usuário

A figura 2.7 é a tela de boas vindas, ela contém instruções sobre o preenchimento da pesquisa.

Figura 2.7 – Tela de Boas Vindas

A figura 2.8 é a tela apresentada na sequência, onde o usuário escolhe um número de 1 a 6 para indicar o grau de concordância com a assertiva feita no *caput* de cada item. Os itens indicados com fundo avermelhado são aqueles que ainda não foram respondidos. De igual sorte, os demais já tiveram sua resposta cadastrada. O usuário pode, em qualquer tempo antes do envio do questionário, responder os itens em vermelho, ou alterar a resposta dos demais itens. Ao navegar para a próxima página de itens, ou ao clicar em "Encerrar" a resposta do usuário é automaticamente salva. O usuário tem a visualização também da quantidade total de perguntas, bem como da quantidade de perguntas a serem respondidas. O paginador utilizado também indica o número total de páginas e número da página atualmente selecionada. Outra funcionalidade é o ajuste da quantidade de questões a serem apresentadas por tela navegada, uma vez que há usuário que desejam ter uma visão geral das perguntas e outros que conseguem ter uma visão melhor e menos poluída com a apresentação de poucos itens, ou ainda preferem não utilizar barra de rolagem.

Figura 2.8 – Responder Pesquisa

2.2.5. Diagrama de classes

A figura 2.9 mostra o diagrama de classes do sistema. O relacionamento n:m entre Pergunta e Evento_Pesquisa é a indicação do questionário daquele evento de pesquisa e permite o reaproveitamento de uma pergunta em mais de um evento de pesquisa, evitando assim redundância no banco de dados. Este relacionamento e o relacionamento entre Resposta e Pergunta foram os principais usados para validação da arquitetura utilizada, onde foi aplicado na camada de Integração a implementação JPA Hibernate com Anotações. Cada Pergunta só pode existir se pertencer a um SubGrupo e cada Subgrupo só pode existir se pertencer a um Grupo, por isso o relacionamento de composição, o mesmo valendo para o relacionamento entre Evento_Pesquisa e Pesquisa.

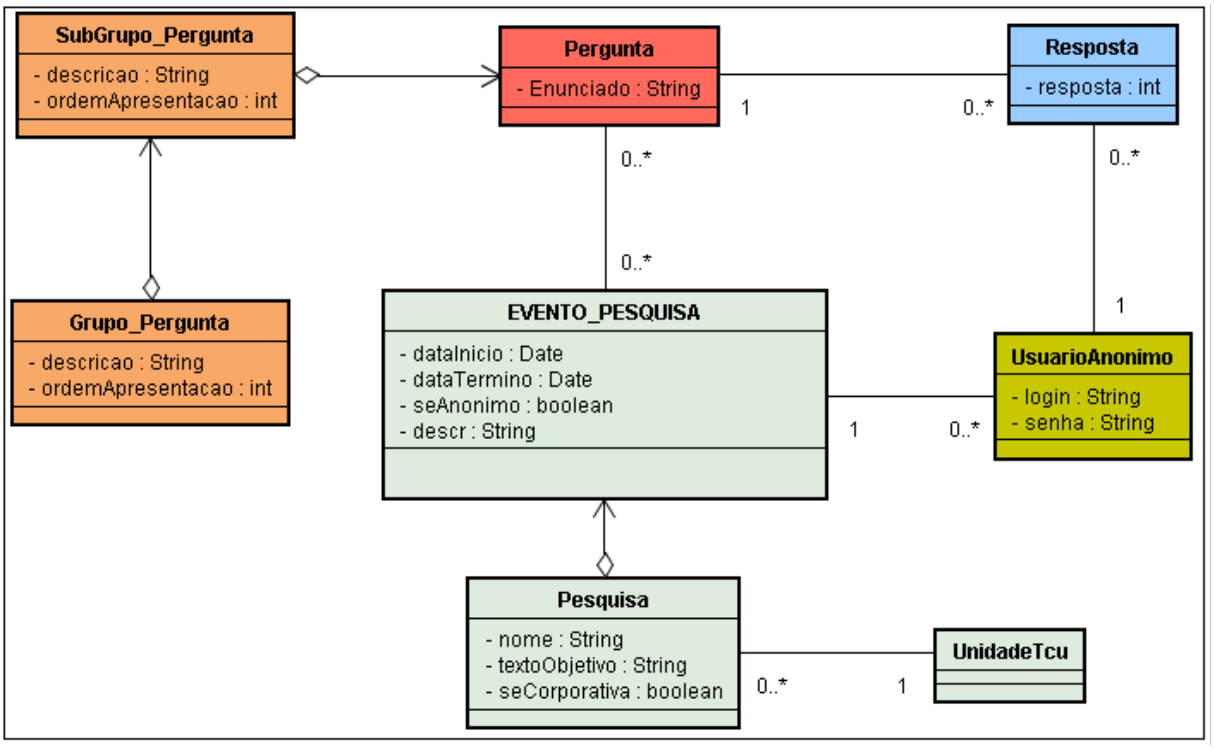

Figura 2.9 – Diagrama de Classes

3. Arquitetura de Software

A arquitetura do TCU é totalmente aderente a divisão em cinco camadas proposta pelo catálogo de padrões J2EE [Core, 2001], segundo o qual uma aplicação J2EE é dividida de acordo com as seguintes camadas:

- Camada cliente: interface do usuário ou de serviços. Tipicamente representa uma aplicação independente ou *browser* rodando applets ou páginas HTML;
- Camada Web ou de Apresentação: consiste de servlets e páginas JSP com o objetivo de capturar requisições e processar respostas para a camada do cliente;
- Camada de Negócio: contém toda a lógica da aplicação e representa o modelo de negócio;
- Camada de integração: contém lógica de acesso à camada de dados;
- Camada de dados: consiste de sistemas de bancos de dados, transações e outros recursos legados;

Segundo o documento de arquitetura do TCU [TCU-a, 2008], na camada de apresentação estão as *servlets*, folhas de estilos, *templates HTML*, páginas *JSP*, *backing beans*, *action classes* e *scripts javascript*.

As *Servlets* são usada para geração dinâmica das interfaces *WEB*, principalmente em sistemas legados, uma vez que atualmente novos sistemas são desenvolvidos utilizando o *framework JSF*.

As *Action* classes da tecnologia *JSF* são usadas para implementar um "evento" disparado por uma página Web. Para tratar os eventos da tela, a *Action Class* faz chamadas a Controladora, pertencente à camada de negócio. Os tratamentos de segurança, autenticação e autorização do usuário são tratados pela ActTcuAbstrata, que verifica se o usuário está autorizado e em caso positivo o coloca em um objeto do contexto *J2EE*, chamado *ContextObject*. O *Context Object*, assim como os *DTOs* (*Data Transfer Objects*), pode ser acessado em qualquer das camadas de apresentação, negócio ou integração.

A classe *backing bean* é usada na tecnologia *JSF* para guardar todos os dados de uma tela *JSF*. A classe *backing bean* é opcional pois a *Action Class* pode utilizar um *DTO* para guardar os dados da tela, isto é, o *DTO* pode atuar como *backing bean*.

A figura 3.1 mostra a distribuição dos componentes na arquitetura do TCU:

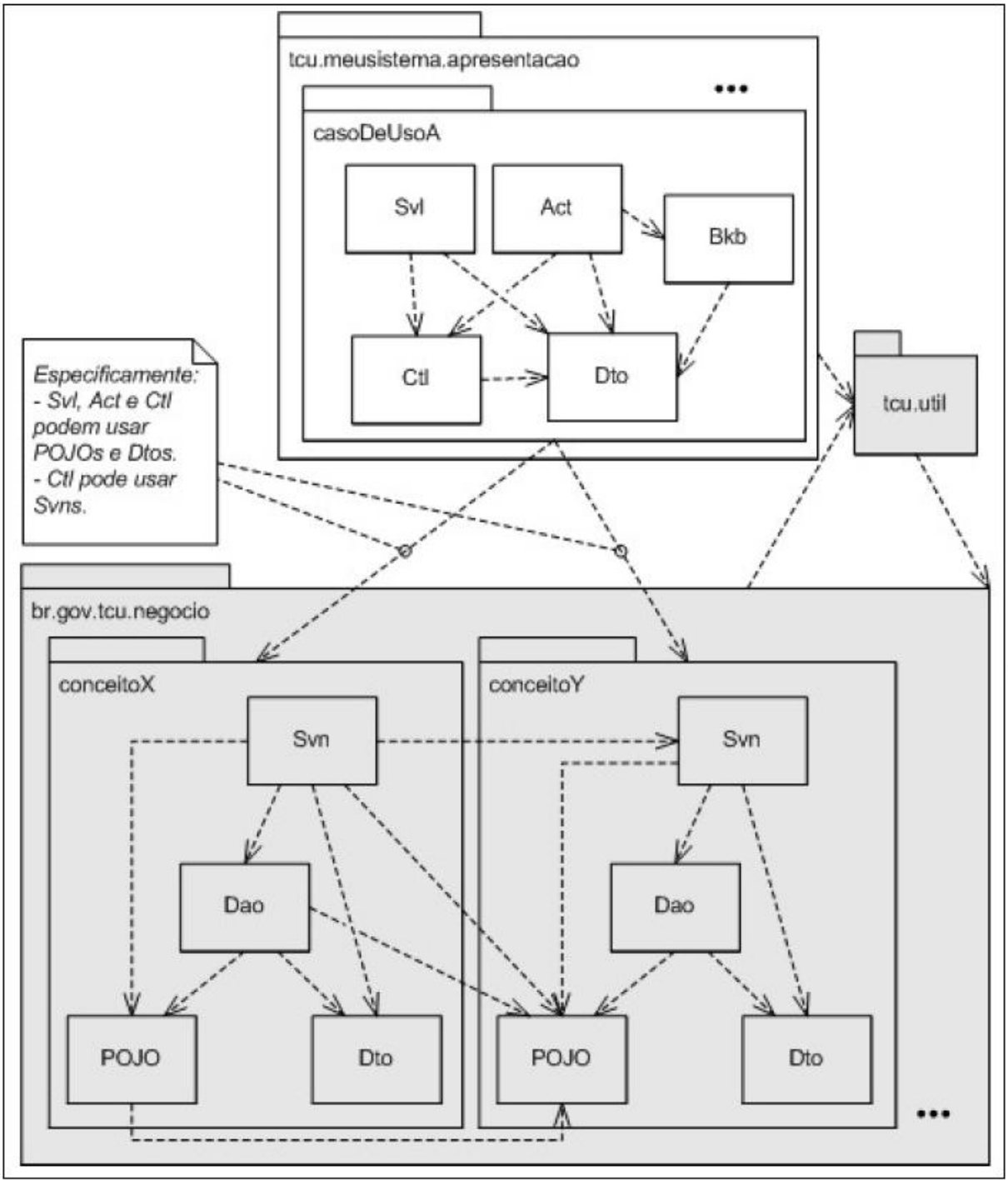

Figura 3.1 – Arquitetura de camada de apresentação do TCU [TCU-a, 2008]

A controladora representa um ponto de acesso único à camada de negócio por caso de uso. Os métodos definidos em uma controladora representam as necessidades de um caso de uso em relação ao negócio. São instanciadas por *Servlets* (Svl) e *Action Classes* JSF (Act) e não mantêm estado conversacional entre uma requisição (*http request*) do usuário e outra.

As Controladoras não demarcam transação. A demarcação é implícita: a transação inicia quando a primeira operação de banco é realizada e é fechada no *ContextFilter*, que efetua um *commit* caso não tenha ocorrido ou exceção ou *rollback* caso tenha ocorrido exceção. Em casos excepcionais, o código da aplicação pode efetuar o *commit* antes do fim do processamento do *request* chamando *ContextObject.getPersistencia().efetivarTransacao()*.

A Controladora no pacote A pode chamar a Controladora no pacote B dentro do mesmo sistema. Porém, uma Controladora não pode chamar outra que esteja em outro sistema. O motivo principal é que o *deploy* da camada de apresentação de cada sistema é feito por meio de arquivos war/ear separados e, portanto uma Controladora não enxerga uma classe Controladora de outro sistema. Contudo, essa restrição não impede que a camada de apresentação de um sistema interaja com outro sistema por meio de redirecionamento de requisições *HTTP* ou *Web Services*.

As Controladoras recebem os dados do Svn (Serviço de Negócio) normalmente na forma de *Data Transfer Objects*(*DTO*), objetos tipicamente usado para o transporte de dados entre classes de diferentes camadas, ou *Plain Java Old Objects* (*POJOs*), que podem ou não ser convertidos para *DTOs* de apresentação que são então repassados para a *Servlet* ou *Action Class* chamadora.

Dentro do pacote br.gov.tcu.negocio, o código está dividido em pacotes que correspondem normalmente a conceitos da modelagem de negócios. Exemplos reais: avaliacaodesempenho, orgaoentidade, papeltramitavel.

Dentro destes pacotes pode haver uma ou mais classes Svn, DAO (*Data Access Object*), *DTO* e *POJO*.

Um Svn ou Serviço de negócio representa um comportamento (regra, ação, etc.) de negócio e, geralmente, seus métodos são responsáveis por regras de negócio que manipulam mais de um Objeto de Negócio (*POJO*). Os parâmetros recebidos são, com freqüência, Objetos de Negócio envolvidos na regra de negócio, mas podem ser também tipos primitivos ou estruturas nativas do Java.

São criados e chamados pelas Controladoras e retornam tipos primitivos e estruturas da linguagem Java; *DTOs* que são utilizados para agregar os dados de retorno; ou objetos de negócio (*POJOs*) quando o retorno representa um conceito de negócio.

Os métodos representam operações de negócio corporativas, ou seja, utilizadas por várias aplicações. O Svn é, por definição, corporativo, ou seja, assim como os objetos de negócio, podem ser utilizados por várias aplicações.

Os Svns não estão mapeados para o banco de dados para fins de persistência; devem ser utilizados quando um comportamento de negócio não possa ser alocado (atribuição de responsabilidade) a um único Objeto de Negócio POJO, ou seja, quando um processo ou transformação significante no domínio (conceitos de negócio) não é uma responsabilidade natural de um único Objeto de Negócio POJO; não executam lógica atrelada a uma determinada interface ou Caso de Uso específico já que este é o papel da Controladora; e criam *DAOs* dentro dos métodos de negócio quando necessário.

Os *DAOs* são objetos especializados em acesso aos dados, ou seja, todo acesso ao banco de dados deve ser feito por este elemento arquitetural, para tal estes objetos recuperam a interface IPersistencia através do *ContextObject*, por meio do qual fazem acesso ao tipo de persistência apropriado e devolvem tipos já existentes na linguagem Java, *DTOs* quando o retorno não representar um conceito de negócio, ou *POJOs* quando o retorno for um conceito de negócio.

A persistência é acessada pelo *DAO* das seguintes formas:

- Através de Objetos de Negócio *POJO* mapeados via mapeamento objeto relacional (ex.: Hibernate);
- Diretamente via *JDBC*, através de cláusulas *SQL* e procedimentos armazenados (*stored procedures*);

Os POJOS são objetos que representam conceitos no domínio TCU. Os objetos *POJO* são objetos "puros" Java, contrapondo-se a outros tipos de objetos, que são dependentes ou vinculados a um *framework*:

- Não possuem dependências a recursos de infra-estrutura ou outros elementos da arquitetura;
- Não implementam interfaces de infra-estrutura e não derivam de classe de infra-estrutura;

Podem possuir métodos de negócio, desde que sejam capazes de resolver cada chamada de forma auto-suficiente, ou seja, os dados necessários ao cumprimento do método de negócio devem estar disponíveis no próprio objeto de negócio ou em objetos de negócio associados.

Quando mapeados através do mapeamento objeto relacional (ex.:Hibernate), estes objetos podem estar associados a outros objetos mapeados, formando um “grafo” navegável de objetos de negócio.

3.1. Padrões de Projeto Utilizados

3.1.1. Padrão MVC

A finalidade do padrão *Model-View-Controller* (MVC) [Reenskaug, 1979] é isolar a lógica de negócios da interface com o usuário, resultando em uma aplicação onde é mais fácil modificar seja a aparência da aplicação ou as regras de negócio subjacentes. No padrão MVC, o modelo representa a informação (os dados) da aplicação; a visão corresponde aos elementos da interface com o usuário tais como rótulos, *checkboxes*, campos de formulários, etc; e o controlador gerencia a comunicação entre essas camadas, gerenciando as regras de negócio usadas para transmitir dados de e para o modelo. Na arquitetura do TCU encontramos as *Servlets*, as *ACTs* e o *BackingBeans* na camada de visão, as controladoras na camada de controle, e as *SVNs* e os *DAOs* na camada de modelo. A figura 2.11 mostra a divisão dos objetos da presente aplicação segundo o padrão *MVC*.

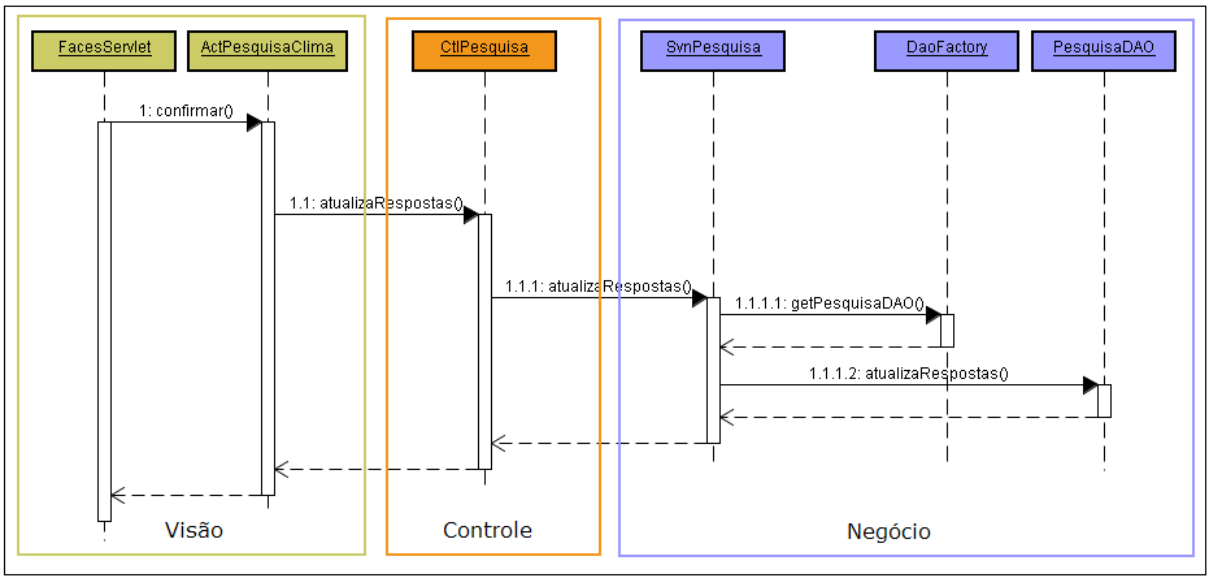

Figura 3.2 – Padrão MVC aplicado no presente projeto

3.1.2. Padrão Data Access Object (DAO)

Há atualmente diversos mecanismos de persistência, e mesmo para um tipo específico de persistência, há peculiaridades que variam até mesmo de um fornecedor para outro. Os mecanismos de acesso, as APIs e as características variam entre os diferentes tipos tais como Sistemas de Gerenciamento de Banco de Dados (SGBDs), banco de dados orientados a objetos, arquivos de texto puro, e assim por diante. As aplicação que precisam ter acesso a dados de um sistema legado frequentemente precisam acessa APIs que podem ser proprietárias. Tais fontes de dados oferecem desafios ao desenvolvimento da aplicação e podem causar uma dependência entre o código da aplicação e o código de acesso aos dados. Quando *entity beans, session beans,* ou mesmo componentes da camada de apresentação como servlets e *helper objects* para páginas *JavaServer Pages* (JSP), precisam de acesso a uma fonte de dados, elas podem usar a API apropriada para obter conectividade e manipular a fonte de dados. Mas incluir código de conectividade e de acesso a dados dentro desses componentes introduz um alto acoplamento entre esses componentes e a implementação da fonte de dados. Tais dependências de código nesses componentes torna difícil migrar uma aplicação de uma fonte de dados para outra. Quando a fonte de dados muda, os componentes precisam ser modificados para manipular o novo tipo de fonte de dados.

Segundo o padrão DAO, os códigos de acesso a persistência devem ficar concentrados nesse objeto [Joshi, 2005]. No presente trabalho a aplicação do padrão se faz por meio da concentração dos códigos específicos de acesso ao banco de dados nos objetos que implementam as interfaces PesquisaDAO e UsuarioDAO, para acesso aos dados da pesquisa e do usuário respectivamente.

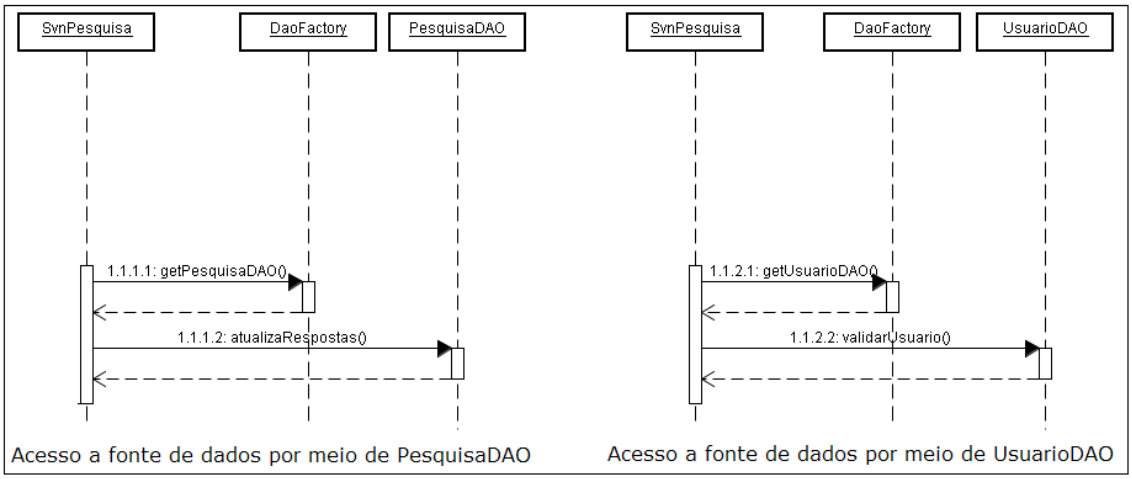

Figura 3.3 – Acesso a fonte de dados e aos objetos de negócio

3.1.3. Padrão Façade

Numa aplicação J2EE, a camada de apresentação (serlets, jsp, etc) precisa de acesso aos objetos de negócio para cumprir suas funcionalidades. Esses clientes podem diretamente interagir com estes objetos de negócio, no entanto, quando há exposição dos objetos de negócio ao cliente, o cliente precisa entender e ser responsável pelos relacionamento entre os objetos, e deve ser capaz de manipular o fluxo do processo de negócio.

Entretanto, interação direta entre o cliente e os objetos de negócio leva a um alto acoplamento entre os dois, e tal alto acoplamento faz com o que o cliente seja diretamente dependente da implementação dos objetos de negócio. Dependência direta significa que o cliente precisa representar e implementar as complexas interação que dizem respeito à localização do objeto e sua criação, e deve gerenciar os relacionamentos entre os objetos de negócio participantes.

Também ocorre um alto acoplamento, agora entre objetos de negócio, quando estes próprios gerenciam seus relacionamentos. Frequentemente, não está claro onde o relacionamento é gerenciado. Isso leva a relacionamentos complexos entre os objetos de negócio e a uma falta de flexibilidade da aplicação, o que torna mais difícil implementar as modificações demandadas.

Portanto é necessária uma estratégia de acesso uniforme aos objetos de negócio. Na arquitetura do TCU essa estratégia é implementada nos *SVNs*, ou Serviços de Negócio, que concentram o acesso a esses objetos, conforme preconizado pelo padrão Fachada (*Façade*) [Freeeman & Freeman, 2007] recebendo os pedidos da camada de Controle e os distribuindo adequadamente aos objetos de negócio apropriados

3.1.4. Padrões Fábrica Abstrata e Método Fábrica

O padrão Fábrica oferece um meio de encapsular a instanciação de objetos concretos. Toda instanciação desses objetos é feita pela Fábrica, que decide qual o tipo de objeto a ser instanciado baseado nos parâmetros enviados. Isso permite maior flexibilidade e menor acoplamento entre o cliente e o objeto a ser instanciado.

Na arquitetura utilizada no presente projeto os padrões Fábrica Abstrata e método Fábrica foram utilizados para instanciação dos *DAOs* [Joshi, 2005], bastante útil quando o

mecanismo de persistência de dados está sujeito a mudanças de uma implementação para a outra, como sugerido em "Design Patterns: Elements of Reusable Object-Oriented Software" [Gamma et. al, 1994]. Neste caso, esta estratégia fornece um objeto Fábrica de DAO abstrata que pode construir vários tipos de Fábricas de *DAO* concretas, cada fábrica suportando diferentes mecanismos de persistência. Uma vez obtida uma Fábrica concreta para um mecanismo de persistência específico, esta é usada para fornecer *DAOs* implementados segundo este mecanismo específico.

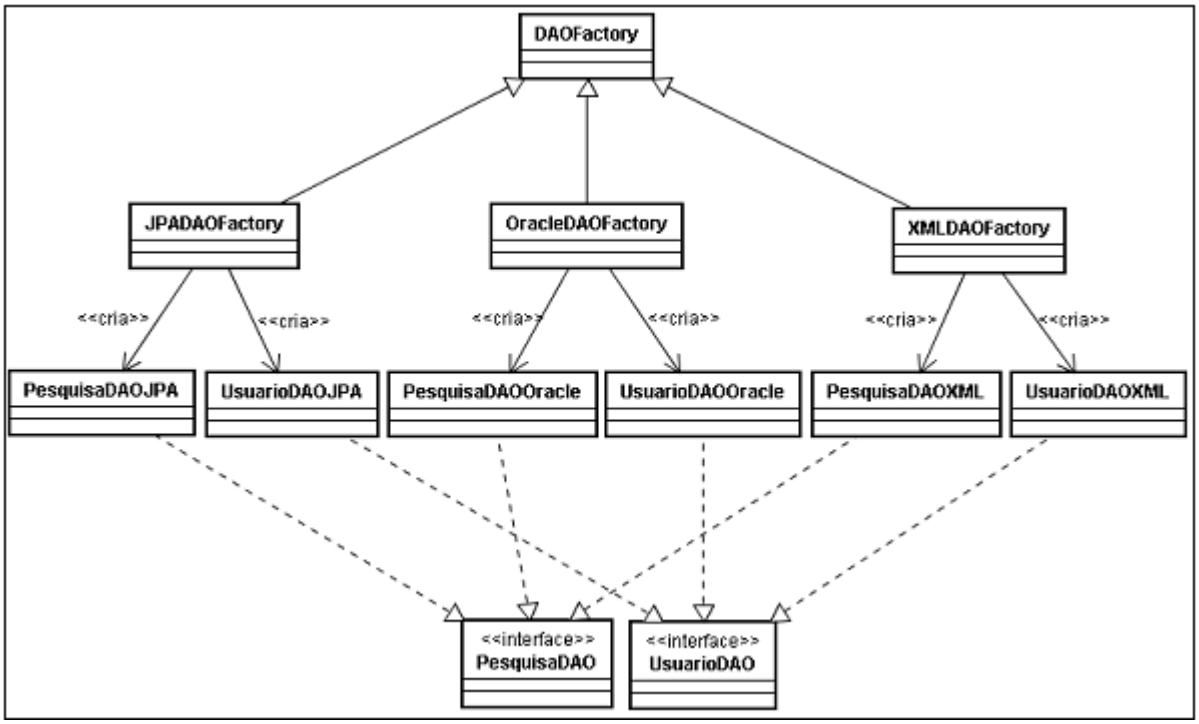

Figura 3.4 – Padrões Fábrica Abstrata e Método Fábrica aplicados no presente projeto

A figura 3.4 mostra um diagrama de classes dessa estratégia. Este diagrama de classes mostra uma Fábrica de *DAOs*, que é uma classe abstrata herdada e implementada por diferentes Fábricas de *DAOs* concretas que suportam acesso aos mecanismos de persistência específicos. O cliente pode obter uma implementação tal como *JPADAOFactory* e usá-la para obter *DAOs* concretos que trabalham com um mecanismo de persistência específico. Por exemplo, o cliente pode obter uma *JPADAOFactory* e usá-la para obter *DAOs* específicos tais como *PesquisaDAOJPA* e *UsuarioDAOJPA*. Esses *DAOs* podem estender e implementar uma classe base ou uma interface (*PesquisaDAO* e *UsuarioDAO* respectivamente) que contém os serviços do DAO para o objeto de negócio que o mesmo suporta.

3.1.5. Padrão Singleton

A finalidade do padrão *Singleton* é garantir que apenas uma instância de uma classe é carregada, e que essa instância fornece um ponto de acesso global e não mantém informações de estado [Freeman & Freeman, 2007].

Isso é útil quando exatamente um objeto é necessário para coordenar as ações em todo o sistema. Na arquitetura do TCU é esse exatamente o papel da Controladora, a qual deve ser única e sem informações de estado. Assim este é justamente o local onde é aplicado o padrão *Singleton*.

A figura 3.5 demonstra a implementação desse padrão dentro do sistema de Pesquisa de Clima. A classe CtlPesquisa não possui um construtor público, apenas um construtor privado, desse modo nenhuma outra classe que senão ela própria pode obter uma instância de CtlPesquisa. Para obter uma instância de CtlPesquisa, a ActSegurança chama o método *getInstance*(), este método verifica se a sua referência ao objeto CtlPesquisa é nula, se for ele cria um novo objeto e o retorna, caso contrário ele retorna a referência já existente.

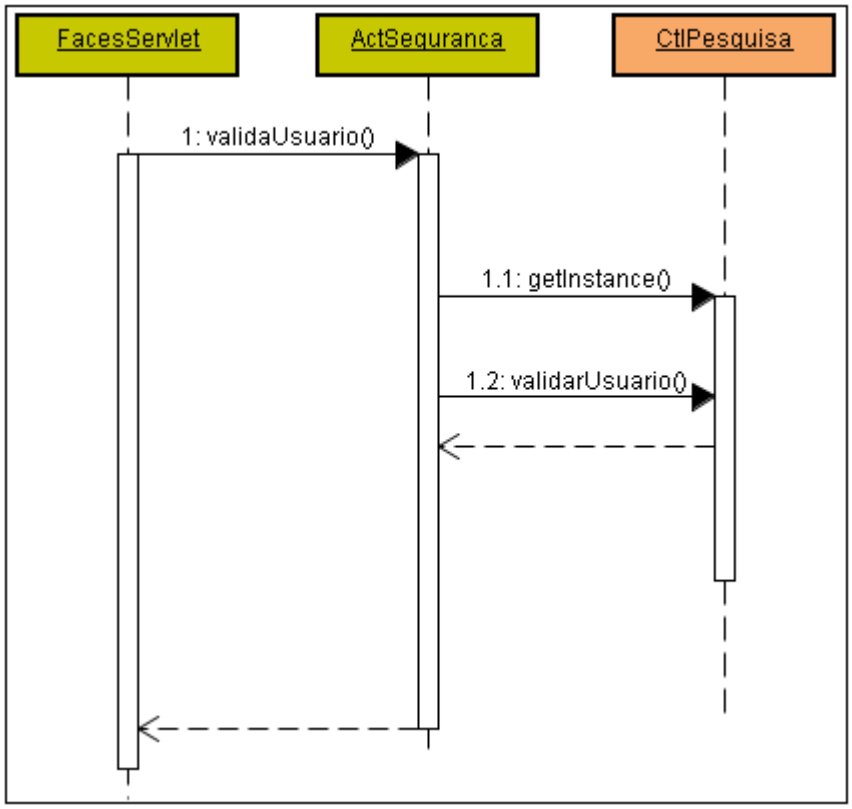

Figura 3.5 – Padrão Singleton

3.2. Configuração do ambiente de desenvolvimento

O ambiente de desenvolvimento está configurado para utilizar as seguintes ferramentas de apoio ao desenvolvimento:

- Eclipse Ganymede: Eclipse é uma plataforma de desenvolvimento para várias linguagens de programação, e é composto por Ambiente de Desenvolvimento Integrado (*IDE*) e um sistema de *plugins* para estendê-lo. No presente trabalho, dentre os *plugins* ou conjunto de *plugins* nativos da versão Ganymede do Eclipse (lançada em 2008), destaca-se principalmente o WTP (*Web Tools Plataform*). A versão Ganymede do Eclipse pode ser baixada em http://www.eclipse.org/downloads/packages/
- O *WTP* é um *plugin* do Eclipse que auxilia os usuários no desenvolvimento de aplicações *J2EE*. Este *plugin* inclui múltiplos editores de texto, editores gráficos, *wizards*, mecanismos de acesso e consulta a banco de dados, dentre outros componentes. O WTP é dividido em dois sub-projetos: O *Web Standart Tools* (*WST*) e o J2EE *Stantdart Tools*. O projeto *WST* possibilita a publicação de recursos no servidor de dentro do Eclipse, bem como sua execução no servidor, dentre outras funcionalidades. O *J2EE* Standart Tools (JST) fornece ferramentas que simplificam o desenvolvimento para *APIs J2EE* incluindo *EJBs*, *Servlet*, *JSP*, *JDBC*, *Web Services*, dentre outros. Dentre essas ferramentas, merece especial destaque o *plugin* Dali, para o desenvolvimento de mapeamentos objeto-relacional. O *WTP* é nativo das versões Europa, Ganymede e Galileo (lançada neste ano) do Eclipse. Para outras versões é necessário fazer o *download* do *plugin* em http://download.eclipse.org/webtools/downloads/
- Dali é um subprojeto do projeto *WTP*, cuja finalidade é construir um amplo conjunto de *frameworks* extensíveis e ferramentas de auxílio na definição e edição de mapeamentos objeto-relacional para entidades JPA (*Java Persistence API*). O suporte ao mapeamento *JPA* foca na minimização da complexidade da construção desses mapeamentos ao fornecer *wizards* para sua criação automática a partir de uma conexão com uma base de banco de dados.

- JUDE é uma ferramenta de modelagem UML criada pela empresa japonesa ChangeVision. A versão Community é livre e oferece as seguintes funcionalidades: suporte à UML 2.0; diagrama de classes, diagrama de casos de uso, diagrama de seqüência, diagrama de colaboração, diagrama de estados, diagrama de atividades, diagrama de implantação, diagrama de componentes; além da geração de código fonte Java 5 a partir do modelo e a importação de código para geração de diagrama de classes, dentre outras funcionalidades. Como existe uma versão livre do JUDE, o número de usuários no mundo inteiro ultrapassava 120000 em outubro de 2006 e recebeu o prêmio "Produto de Software do Ano 2006" instituído pelo IPA (*Information-Technology Promotion Agency*, Japão) [Wikipedia-b, 2009]. A versão 5.2.1 foi a utilizada no presente projeto. Ela está disponível para download em https://jude.change-vision.com/jude-web/download/index.html
- Apache Tomcat é um container *servlet* desenvolvido pela Apache Software Foundation (ASF). Tomcat implementa a especificação de *Servlets* Java e da *JavaServer Pages* (JSP) da Sun Microsystems, e proporciona um servidor web HTTP em ambiente "puro Java" para execução de código Java [Wikipedia-c, 2009]. A versão 6.0.16 foi a utilizada no presente projeto. Ela está disponível para download em http://archive.apache.org/dist/tomcat/tomcat-6/v6.0.16/
- O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (*Structured Query Language* - Linguagem de Consulta Estruturada) como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo [Wikipedia-e, 2009]. A versão 5.0.51-a-comunity-nt foi a utilizada no presente projeto. Ela está disponível para download em http://dev.mysql.com/downloads/mysql/5.0.html

O sistema desenvolvido utiliza as seguintes bibliotecas/*frameworks*:

- *JavaServer Faces (JSF):* é um framework para criação de interfaces para Web. JSF fornece um gerenciamento do ciclo de vida da requisição Web através de *servlet* controladora, bem como um rico modelo de componentes com manipulação de eventos e renderização de componentes. Em suma, o JSF facilita o desenvolvimento Web porque, dentre outras vantagens:

permite a criação de interfaces a partir de um conjunto de componentes padrão reutilizáveis; fornece um conjunto de *tags* JSP para acesso a esses componente; salva informações de estado e as reapresenta nos formulários quando estes são reexibidos e fornece um *framework* para implementação de componentes customizáveis. Há várias implementações do framework JSF (myfaces, richfaces, ADF, icefaces). No presente trabalho foi utilizada a biblioteca myfaces 1.1.5 em conjunto com o tomahawk 1.1.6, a primeira é uma implementação JSF, e a segunda é uma extensão que contém uma série de componentes adicionais. Ambas são fornecidas pela da Apache Software Foundation e podem ser baixadas em http://myfaces.apache.org/download.html

- Hibernate/JPA: a Java Persistence API é uma interface padrão para mapeamento objeto-relacional e gerenciamento de persistência para a platarforma JAVA EE. Como parte do esforço de especificação EJB 3.0, ela é suportada pela maior parte dos fornecedores da indústria Java. Hibernate implementa JPA com os módulos Hibernate Annotations e Hibernate Entity Manager, sobre o tradicional Hibernate Core. Hibernate, como outras ferramentas de mapeamento objeto-relacional, requer metadados que determinam a transformação dos dados de uma representação para outra (e vice-versa), para este fim é possível utilizar anotações JDK 5.0 em conjunto ou em substituição aos mapeamentos XML. No presente projeto foi utilizado Hibernate-Annotation 3.4.0 GA, Hibernate Entity-Manager 3.4.0 GA e Hibernate Core 3.3.0.SP1. O Hibernate pode ser baixado em https://www.hibernate.org/6.html

A aplicação utiliza JRE 6 e roda em um servidor de aplicação Tomcat versão 6.0, o qual já traz embutido a biblioteca servlet-api e jsp-api, necessárias respectivamente para execução de servlets e interpretação de páginas JSP. A JRE 6 pode ser baixada em http://java.sun.com/javase/downloads/index.jsp e o servidor de aplicação Tomcat pode ser baixado em http://tomcat.apache.org/download-60.cgi

A figura a seguir mostra a IDE Eclipse com o detalhamento da pasta WEB-INF/lib:

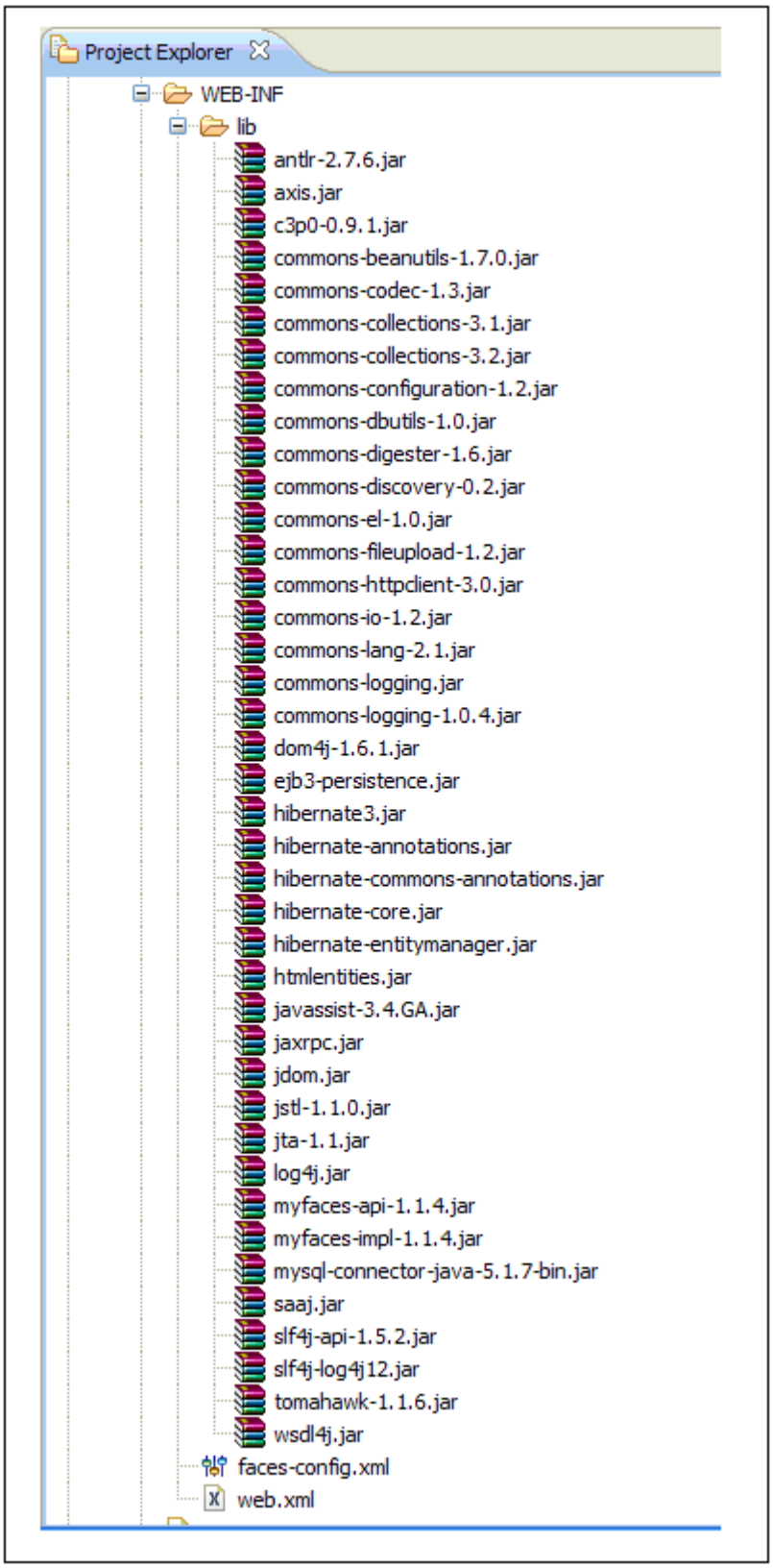

Figura 3.6 – Bibliotecas utilizadas no projeto

4. Conclusão e Trabalhos Futuros

O desenvolvimento do um sistema de pesquisa de clima organizacional, em plataforma J2EE, foi bem sucedido. Os casos de uso escolhidos foram implementados dentro do prazo estabelecido obedecendo à arquitetura de sistemas atual do TCU.

A aplicação do *framework JSF* permitiu um rápido desenvolvimento da camada de aplicação, requisitos posteriormente apresentados pelo usuário, como a possibilidade de indicar o número de itens a serem exibidos puderam ser prontamente atendidos graças a certa familiaridade desenvolvida com os componentes JSF. Nesse sentido, foram fundamentais os exemplos de código myfaces e tomahawk disponíveis em http://www.irian.at/myfaces, neste sítio é possível baixar o arquivo war dos exemplos, sua importação para a IDE eclipse ou o seu *deploy* e teste no servidor de aplicação é realizado então de maneira muito rápida, permitindo ao desenvolvedor maior rapidez no entendimento do funcionamento do componente por meio de um código funcional.

Entretanto, verificou-se que é de fundamental importância entender as nuances do funcionamento interno do JSF, principalmente no que tange ao seu ciclo de vida, pois por vezes a solução de um problema demorou muito mais do que seria esperado, quando finalmente se verificava que determinada ação ocorria em fase diversa da imaginada ou simplesmente determinada fase não estava sendo executada. Não obstante, o fato de o JSF ser fortemente voltado a utilização de componentes permite que os prazos de desenvolvimento sejam cada vez mais curtos, em razão da experiência adquirida na utilização do componente e da forma que seus eventos são tratados durante o ciclo de vida do JSF.

O fórum de discussão do JSF no sítio da Sun já possui mais de um milhão de visitas, o que além de mostrar a ampla disseminação dessa tecnologia, mostra o grau de dificuldade que os desenvolvedores têm para, partindo de um framework Web do modelo 2 MVC, como o *Struts*, se adaptarem a um novo paradigma fortemente orientado a componentes e eventos [Barcia, 2004], daí a necessidade de se considerar o médio e o longo prazo para analisar o retorno do investimento humano no estudo desse framework de camada Web.

O framework Hibernate ofereceu desafios ainda maiores. A utilização do plugin "Dali" permitiu a geração automática dos mapeamentos, a partir da configuração da fonte de dados. No entanto, o *plugin* ainda não é "inteligente" o suficiente para resolver alguns conflitos, como por exemplo, a definição dos atributos "*insertable*" e "*updatable*" como *false*,

no caso do mapeamento de chaves estrangeiras que são ao mesmo tempo chave primária no banco relacional.

Uma vez resolvidos problemas dessa natureza, a realização das consultas e das atualizações no banco de dados foi possível tão somente com a leitura da documentação e dos inúmeros exemplos disponíveis na internet.

No entanto, para utilização do Hibernate, o que trouxe maiores dificuldades foi a combinação das bibliotecas *hibernate-core, hibernate-annotations e hibernate-entity-manager* e suas respectivas dependências. Nesse sentido em projeto futuro sugere-se a utilização da ferramenta MAVEN [Apache Maven], que resolve de maneira automática a dependência entre as bibliotecas.

O mecanismo de anotações facilitou bastante a criação dos mapeamentos, já que é feita diretamente no atributo/classe mapeada, não havendo a necessidade de múltiplas *tags* XML aninhadas e do gerenciamento do arquivo hibernate.cfg.xml, o qual ainda é utilizado no âmbito do TCU. Como essas duas formas mapeamento podem coexistir, do presente trabalho nasce a oportunidade de apresentar essa alternativa para o Serviço de Padronização e Qualidade de Software a fim migrar paulatinamente para o uso de Hibernate com Anotações.

O desenvolvimento do presente Sistema de Pesquisa de Clima foi uma excelente oportunidade para aplicação de padrões de projeto já largamente aplicados no âmbito do TCU, como DTO, DAO, Façade, *Application Controler* e MVC; mas também de padrões não utilizados ou pouco utilizados como a combinação dos padrões *Abstract Factory*, *Factory Method* e *DAO*. Essa combinação proporciona uma grande flexibilidade para troca do mecanismo de persistência e um baixo acoplamento entre as camadas de negócio e de integração. Aqui, novamente se apresenta uma oportunidade de discussão de melhoria da arquitetura do Tribunal de Contas da União. A classe *DAOFactory* é uma classe abstrata que define uma interface para criação das fábricas para cada tipo de persistência. Assim, a camada de negócio solicita um novo DAOUsuário, sem conhecer qual mecanismo de persistência está sendo utilizado, isso permite uma grande flexibilização na troca do mecanismo de persistência.

As classes do presente projeto foram desenvolvidas e distribuídas pela arquitetura segundo o modelo J2EE de cinco camadas, já que a arquitetura do TCU é totalmente aderente a esse modelo.

Os testes unitários foram desenvolvidos utilizando a ferramenta JUnit, com ampla cobertura da parte do sistema que foi desenvolvida. Assim foram realizados testes de validação do usuário, recuperação das perguntas de um evento de pesquisa de clima e atualização das respostas do usuário. A utilização de testes unitários representa um importante passo rumo ao desenvolvimento orientado a testes, o qual não foi empregado no presente projeto, em razão da pouca familiaridade dos desenvolvedores com essa metodologia, mas que deverá ser empregada em projetos futuros.

A metodologia de desenvolvimento foi resultado de uma customização ora envolvendo práticas preconizadas pelo XP (*eXtreme Programming*) ora com prática do RUP (*Rational Unified Process*). Assim, desde o princípio o presente projeto tinha quatro marcos claramente definidos: análise e projeto, implementação, monografia e defesa; foi escolhido um conjunto essencial de artefatos do RUP a serem implementados, quais sejam o diagrama de classes de negócio, diagrama de casos de uso, documento de especificação de casos de uso, protótipos, diagramas de atividade do processo e diagramas de seqüência dos casos de uso críticos para validação da arquitetura. Outra prática sempre presente que veio a ratificar esse processo foi a colaboração sempre próxima do usuário, com reuniões semanais e validação dos protótipos. Nesse sentido, uma vez definido o protótipo, a prioridade era a implementação, a geração do código, a qual em momento algum atrasou em razão da falta de algum ou alguns dos artefatos ora mencionados.

O desenvolvimento dos demais casos de uso do uso presente sistema se dará como continuidade ao presente projeto, uma vez que se trata de ferramenta institucional no âmbito do Tribunal de Contas de alto valor para seu planejamento estratégico conforme apresentado nos primeiro capítulo do presente trabalho.

Para facilitar a configuração do ambiente no caso utilização de novas tecnologias que não as já mencionadas no presente trabalho, propõe-se a utilização da ferramenta Maven, que é uma ferramenta para gerenciamento e automação de projetos em Java, similar à ferramenta Ant, mas possui um modelo de configuração mais simples, baseado no formato XML. Uma característica chave do Maven é que ele é construído para trabalhar em rede. O núcleo da ferramenta pode baixar *plugins* de um repositório. (o mesmo repositório utilizado pelos outros projetos Java do Apache e outras organizações). O Maven disponibiliza suporte nativo para a recuperação de arquivos deste repositório, e para a inclusão dos artefatos resultantes no final do processo [Wikipedia-d, 2009]. Com isso, pretende-se automatizar o processo de

preparação do ambiente de desenvolvimento, uma das principais dificuldades experimentadas no presente trabalho.

Outro trabalho futuro que merece destaque é fomentar a discussão, no âmbito do TCU, da discussão sobre Fábrica de DAOs, utilizando os padrões aqui expostos, a fim de reduzir o acoplamento entre as camadas de negócio e de integração e da atualização do Hibernate para se trabalhar com anotações, a fim de facilitar o processo de mapeamento e reduzir o impacto de uma manutenção nos fontes do sistema, uma vez que a alteração do mapeamento passa a ocorrer em um único ponto.

Finalmente, nas pesquisas realizadas no presente projeto, chamou a atenção o poder do framework JBoss Seam [JBoss Seam] pela sua capacidade de integrar JSF e Hibernate, de gerenciar o contexto, de lidar com fluxo de trabalho, de trabalhar com AJAX e de gerenciamento de tarefas via jBPM [JBoss jBPM], sendo que, este último é uma demanda há muito discutida no âmbito do Tribunal de Contas, num esforço para, antes da implementação do software, fazer a modelagem do processo de negócio e tornar o software aderente àquele e não o contrário, como ocorre muitas vezes. Diante dessas facilidades, é importante discutir a implementação desse framework no âmbito do TCU, analisando o impacto dessa adoção por meio de projetos pilotos implantados em servidores exclusivamente dedicados a esse fim.

REFERÊNCIAS BIBLIOGRÁFICAS

[Apache Maven] http://maven.apache.org/

[Barcia, 2004] "JavaServer Faces (JSF) vs Struts: A brief comparison", http://websphere.sys-con.com/node/46516

[Core, 2001] Deepak Alur, John Crupi, Dan Malks. Core J2EE Patterns: Best Practices and Design Strategies. Prentice-Hall, 2001. http://java.sun.com/blueprints/corej2eepatterns/index.html

[Gamma et. al, 1994] Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides- Design Patterns: Elements of Reusable Object-Oriented Software, 1994 .

[Freeeman & Freeman, 2007] Eric Freeman e Elisabeth Freeman- Use a cabeça! Padrões de Projeto (Design Patterns) 2. Ed, 2007

[Joshi, 2005] Joshi, Anand Prakash- Design with the JSF architecture- http://www.ibm.com/developerworks/web/library/wa-dsgnpatjsf.html

[JBoss Seam] http://www.jboss.com/products/seam/

[JBoss jBPM] http://jboss.com/products/jbpm

[Marty Hall & Larry Brown, 2003] "Core Servlets e Java Server Pages". Editora Ciência Moderna

[Reenskaug, 1979]- Trygve M. H. Reenskaug- THING-MODEL-VIEW-EDITOR http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

[TCU, 2006] Plano Estratégico do Tribunal de Contas da União- Período 2006-2010

[TCU-a, 2008] Norma de Arquitetura – DAS da Arquitetura de Referência 7.0

[Wikipedia-a, 2009] Balanced Score Card- http://pt.wikipedia.org/wiki/BSC

[Wikipedia-b, 2009] JUDE UML Tool- http://en.wikipedia.org/wiki/JUDE_(UML_Tool)

[Wikipedia-c, 2009] Apache Tomcat- http://en.wikipedia.org/wiki/Apache_Tomcat

[Wikipedia-d, 2009] Apache Maven- http://pt.wikipedia.org/wiki/Apache_Maven

[Wikipedia-e, 2009] MySQL SGBD http://pt.wikipedia.org/wiki/MySQL

ANEXO I

Especificação de Caso de Uso

Responder Questionário

1. INTRODUÇÃO

A proposta deste Documento é detalhar um caso de uso. Esse detalhamento descreve as instâncias (cenários) de casos de uso, em que cada instância representa um conjunto de passos que devem ser executados pelo sistema, com o objetivo de produzir algo significativo para o(s) ator(es).

Seu escopo abrange a descrição dos atores envolvidos na funcionalidade, a seqüência de ações a serem realizadas pelo sistema e pelo ator e a série de características específicas do caso de uso em questão.

Termos e abreviaturas específicos podem ser encontrados no Glossário do respectivo sistema.

2. BREVE DESCRIÇÃO

A proposta deste caso de uso é permitir que o usuário responda questionário em pesquisas institucionais realizadas no âmbito do Tribunal de Contas da União, promovendo a inserção ou alteração de respostas ao questionário.

3. ATOR

O ator envolvido na execução deste caso de uso é o "**Servidor Ativo do TCU**".

4. FLUXOS DE EVENTOS

4.1. Fluxo Básico

Este caso de uso inicia quando o cliente deseja inserir, alterar ou enviar as respostas de um questionário de um determinado evento de pesquisa de clima.

4.1.1. O Sistema valida a ficha do usuário. Neste ponto, incluir o caso de uso "Validar Cartão de Acesso" para calcular a gratificação de desempenho de servidores avaliáveis

4.1.2. Caso o cliente não seja validado com sucesso o caso de uso é encerrado.

4.1.3. Caso o cliente seja validado com sucesso o sistema apresenta ao usuário uma tela contendo a primeira questão ainda não respondida, diferenciando por cores às questões não respondidas daquelas que já foram respondidas. As questões não respondidas são apresentadas em fundo esverdeado e as questões não respondidas são apresentadas em fundo avermelhado.

4.1.4. Caso o usuário solicite navegar para a próxima página de questões o sistema salva as questões respondidas na página atual e exibe a próxima página de questões.

4.1.5. Caso o usuário solicite salvar as questões já respondidas ou solicite sair do sistema, o sistema salva a respostas do usuário até o momento e retorna ao passo **Erro! Fonte de referência não encontrada.**

4.1.6. Caso o usuário solicite o envio do questionário o sistema executa o subfluxo Enviar Questionário

4.2. Subfluxos

4.2.1. Subfluxo Enviar Questionário

4.2.1.1. O sistema apresenta a última página de questões com a opção para o usuário enviar o questionário

4.2.1.2. Caso o usuário solicite salvar as questões já respondidas ou solicite sair do sistema ou ainda feche a janela do navegador, o sistema salva a respostas do usuário até o momento e retorna ao passo **Erro! Fonte de referência não encontrada.**

4.2.1.3. Caso o usuário solicite o envio do questionário, o sistema salva o questionário, muda seu estado para finalizado e apresenta uma mensagem ao usuário confirmando o envio do questionário e retorna ao passo **Erro! Fonte de referência não encontrada.** do fluxo básico.

Fluxo(s) Alternativo(s):

Existe(m) questão(ões) não respondida(s)

4.3. Fluxos Alternativos

4.3.1. Existe(m) questão(ões) não respondida(s)

No passo 4.2.1.3, caso haja alguma questão ainda não respondida pelo usuário, o sistema exibe a mensagem "***Existe questão que ainda não foi respondida. É necessário que todas as questões estejam respondidas para que o questionário possa ser enviado.***" e retorna ao passo 4.2.1.1

5. REQUISITOS ESPECIAIS

Esta seção não se aplica ao caso de uso em questão.

6. PRÉ-CONDIÇÕES

Esta seção não se aplica ao caso de uso em questão.

7. PÓS-CONDIÇÕES

As questões do questionário respondidas até o momento são salvas, ou todas as questões são salvas e o questionário muda para o estado finalizado.

8. PONTOS DE EXTENSÃO

Esta seção não se aplica ao caso de uso em questão.

9. OBSERVAÇÕES

Esta seção não se aplica ao caso de uso em questão.

10. REFERÊNCIAS

Esta seção não se aplica ao caso de uso em questão.

ANEXO II

Especificação de Caso de Uso
Validar Cartão de Acesso

1. INTRODUÇÃO

A proposta deste Documento é detalhar um caso de uso. Esse detalhamento descreve as instâncias (cenários) de casos de uso, em que cada instância representa um conjunto de passos que devem ser executados pelo sistema, com o objetivo de produzir algo significativo para o(s) ator(es).

Seu escopo abrange a descrição dos atores envolvidos na funcionalidade, a seqüência de ações a serem realizadas pelo sistema e pelo ator e a série de características específicas do caso de uso em questão.

Termos e abreviaturas específicos podem ser encontrados no Glossário do respectivo sistema.

2. BREVE DESCRIÇÃO

Este caso de uso inclui um comportamento adicional e obrigatório no caso de uso Responder Questionário.

3. ATOR(ES)

O(s) ator(es) envolvido(s) na execução do caso de uso está(ão) listado(s) abaixo:

- Servidor Ativo do TCU

4. FLUXOS DE EVENTOS

4.1. Fluxo Básico

Este caso de uso inclui um comportamento adicional e obrigatório no caso de uso Responder Questionário.

4.1.1. O usuário informa o número de usuário e a senha de seu Cartão de Acesso

4.1.2. O sistema verifica se o par número de usuário/senha informados corresponde a um Cartão de Acesso válido. Para verificar a validade do cartão o sistema verifica se existe em sua base o par informado e se a data atual é posterior à data inicial de validade e a anterior à data final de validade

4.1.3. Caso a data de validade do cartão esteja expirada o sistema apresenta uma mensagem ao usuário informando que seu Cartão de Acesso corresponde a um evento de pesquisa em estado criado (ainda não aberto) ou encerrado, conforme a data atual seja anterior ou posterior, respectivamente, ao período de validade do Cartão de Acesso apresentado

4.1.4. Caso o sistema não encontre na base de consulta um par número de usuário/ senha correspondente ao Cartão de Acesso apresentado, o sistema apresenta uma mensagem informando que o par número de usuário/ senha informado não corresponde a um cartão válido, sugerindo ao mesmo que tente digitar novamente.

4.1.5. Caso o sistema encontre na base de consulta um par número de usuário/ senha correspondente ao Cartão de Acesso apresentado, o sistema valida o usuário.

4.1.6. O caso de uso é finalizado

4.2. Fluxos Alternativos

Não há fluxos alternativos para esse caso de uso

5. REQUISITOS ESPECIAIS

Esta seção não se aplica ao caso de uso em questão.

6. PRÉ-CONDIÇÕES

6.1. Cartões de autenticação devem estar gerados

Os Cartões de Acesso para o evento de pesquisa de clima corrente já devem ter sido anteriormente gerados pelo sistema e distribuído aos usuários

7. PÓS-CONDIÇÕES

7.1. Usuário validado/Validação mal-sucedida

O usuário é validado e autorizado a proceder nos passos seguintes do caso de uso base do presente caso de uso, ou, em caso de insucesso, o usuário não é autorizado a proceder nos passos seguintes.

8. PONTOS DE EXTENSÃO

Esta seção não se aplica ao caso de uso em questão.

9. OBSERVAÇÕES

Esta seção não se aplica ao caso de uso em questão.

10. REFERÊNCIAS

1. RESOLUÇÃO TCU Nº 187, DE 5 DE ABRIL DE 2006