UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UMA INTERFACE WEB PARA INTEGRAÇÃO E CONFIGURAÇÃO DA REDE DE SENSORES-ATUADORES DO PROJETO CONVERGE UFSM

TRABALHO DE GRADUAÇÃO

Cristiano Cortez da Rocha

Santa Maria, RS, Brasil

2008

DESENVOLVIMENTO DE UMA INTERFACE WEB PARA INTEGRAÇÃO E CONFIGURAÇÃO DA REDE DE SENSORES-ATUADORES DO PROJETO CONVERGE UFSM

por

Cristiano Cortez da Rocha

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Bacharel em Ciência da Computação

Orientador: Prof. Msc. João Carlos Damasceno Lima

Trabalho de Graduação Nº 244
Santa Maria, RS, Brasil

2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

DESENVOLVIMENTO DE UMA INTERFACE WEB PARA INTEGRAÇÃO E CONFIGURAÇÃO DA REDE DE SENSORES-ATUADORES DO PROJETO CONVERGE UFSM

elaborado por
Cristiano Cortez da Rocha

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Prof. Msc. João Carlos Damasceno Lima
(Presidente/Orientador)

Prof. Dr. Benhur de Oliveira Stein (UFSM)

Profª Drª Iara Augustin (UFSM)

Santa Maria, 31 de janeiro de 2008.

AGRADECIMENTOS

Para que fosse possível chegar a este momento de escrever os agradecimentos, muitas pessoas me ajudaram durante essa caminhada de quatro anos de graduação, e este momento é dedicado a todas elas.

Primeiramente, agradeço aos meus pais Fernando e Rossana por sempre terem me proporcionado as melhores condições para que eu pudesse ingressar em uma instituição renomada como a Universidade Federal de Santa Maria e que quando estive dentro desta me apoiaram em todas as escolhas que tomei e nunca deixaram que nada me faltasse. Agradeço ao meu pai por sempre ter me aconselhado e indicado as melhores direções a serem seguidas durante o curso, pelas conversas sobre assuntos do curso, que muita gente não tem a oportunidade de conversar em casa. Agradeço à minha mãe e à minha irmã Fernanda por terem me proporcionado um ótimo ambiente em casa para que eu pudesse me formar tanto no aspecto pessoal, quanto profissional.

Agradeço aos meus avós Moacyr e Doroty por serem o modelo de pureza, honestidade e caráter em que eu me espelho. Sou muito grato à eles por sempre terem me apoiado e dado suporte a tudo que eu fiz durante a minha vida e também por terem construído um dos meus lugares preferidos: a casa no Lermen, local de inúmeros churrascos e feijoadas realizados pela turma.

Agradeço à minha avó Iracema por ser a pessoa mais bem humorada que eu conheço, que contagia qualquer pessoa com suas frases quase sempre "pervertidas". Agradeço à ela por também nunca ter me negado nenhuma ajuda e por sempre ter sido parceira para uma cervejinha.

Agradeço ao meu tio Reinaldo por ser a pessoa mais organizada, prestativa e tranqüila que eu conheço e em quem eu me espelho por isso. Sou grato à ele por sempre ter uma solução para tudo, desde assuntos automobilísticos até assuntos sobre a melhor forma de desmontar um roupeiro gigante. Agradeço também a minha tia Eliana e à minha prima Giana por sempre terem torcido por mim, me apoiando em todas as decisões que tomei.

Para que fosse possível a realização desde trabalho não posso deixar de agradecer à professora Iara e ao professor Caio por terem confiado em mim e terem me convidado para ingressar no GMob, grupo do qual tiro muito proveito e conhecimento para as próximas caminhadas. Agradeço especialmente ao Caio, que foi muito mais do que um orientador durante esses anos, foi um grande amigo com quem aprendi muito e que sempre acreditou no meu rendimento. Agradeço à ele e ao professor Benhur pelas reuniões do projeto CONVERGE que sempre rendiam boas risadas, mas também foram muito importantes para minha formação acadêmica.

Agradeço também aos meus amigos mais antigos (Visco, Lucas, Mathias, Ricardo, Marcelo e Henrique) com quem tenho orgulho de ter uma amizade tão duradoura, que sempre rende muita gargalhada, churrasco e jogos do Grêmio e que estão quase chegando

neste momento tão merecido de fazer agradecimentos. Agradeço também à gurizada do Lermen (Lelo, Guima, Juquiiinha e Tomas) pelos vários salsipães, gengiskans e Buds consumidos no Lermen.

Não poderia acabar os agradecimentos sem deixar de agradecer aos meus colegas de curso, em especial aos meus irmãos Matheus, Rodolfo, Leandro, Eduardo, Márcio e Tiago por terem sido fundamentais nessa caminhada, pelo conhecimento trocado, pelo apoio que nunca foi negado e principalmente por terem sido parceiros na degustação de diversos vinhos, cervejas, caipirinhas, cachaças, tequilas, absintos, etc.

Por fim, agradeço à Deus por colocar todas essas pessoas no meu caminho e por ter me ensinado a escolhe-las.

"Vim, vi, venci." — JÚLIO CÉSAR

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UMA INTERFACE WEB PARA INTEGRAÇÃO E CONFIGURAÇÃO DA REDE DE SENSORES-ATUADORES DO PROJETO CONVERGE UFSM

Autor: Cristiano Cortez da Rocha
Orientador: Prof. Msc. João Carlos Damasceno Lima
Local e data da defesa: Santa Maria, 31 de janeiro de 2008.

Este trabalho visa o desenvolvimento de uma interface *web* para centralizar as tarefas de gerenciamento, integração e configuração da rede de sensores-atuadores que fazem parte do sistema proposto pelo projeto CONVERGE UFSM. Este projeto propõe uma aplicação de convergência das tecnologias voz sobre IP (*VoIP*), redes sem fio e PABX para a construção de um sistema de telefonia digital, que também ofereça controle de acesso aos laboratórios nas instituições.

Com este trabalho, construiu-se uma aplicação *web* que diminui a complexidade de gerenciamento e operação de diferentes tecnologias, facilitando a visualização dos eventos capturados pelos diversos dispositivos pertencentes ao sistema e tornando possível a configuração de determinadas regras que definem os eventos e as ações realizadas pelo sistema, apresentando essas funcionalidades através de um ambiente amigável e eficaz para os usuários com as devidas permissões. Para a implementação deste trabalho, utilizou-se a arquitetura de *software Model-View-Controller* (MVC) implementada pelo *framework Google Web Toolkit* (GWT), para desenvolvimento de aplicações AJAX.

Palavras-chave: Sensores-atuadores, interface *web*, Google Web Toolkit, AJAX, MVC, convergência tecnológica.

ABSTRACT

Graduation Work
Undergraduate Program in Computer Science
Federal University of Santa Maria

DEVELOPMENT OF A WEB INTERFACE FOR INTEGRATION AND CONFIGURATION OF THE NET OF ACTUATOR-SENSORS OF THE PROJECT CONVERGE UFSM

Author: Cristiano Cortez da Rocha
Advisor: Prof. Msc. João Carlos Damasceno Lima

This work aims at the development of a web interface to centralize the management, integration and configuration tasks from the net of actuator-sensors that are part of the system proposed by the project CONVERGE UFSM. This project proposes a convergence application of the technologies voice over IP (VoIP), wireless and PABX to build a system of digital telephony, and it also provides control the access to the laboratories in institutions.

With this work, built up a web application that decreases the complexity of management and operation of different technologies, facilitating the visualization of the events captured by the several devices belonging to the system and making possible the configuration of some rule that define the events and actions realized by the system, presenting these functionalities to the authenticated users through a friendly and effective environment. In the implementation of this work, it was used the software architecture Model-View-Controller (MVC) implemented by the framework Google Web Toolkit (GWT) for the development of AJAX applications.

Keywords: Actuator-sensors, web interface, Google Web Toolkit, AJAX, MVC, technological convergence.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous Javascript And XML
API	Application Programming Interface
ASP	Active Server Pages
CSS	Cascading StyleSheets
DAO	Data Access Object
DHTML	Dynamic HyperText Markup Language
DOM	Document Object Model
FXS	Foreign eXchange Station
GUI	Graphical User Interface
GWT	Google Web Toolkit
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JDK	Java Development Kit
JRE	Java Runtime Environment
JSP	Java Server Pages
JVM	Java Virtual Machine
MVC	Model-View-Controller
ORM	Object Relational Mapping
PABX	Private Automatic Branch eXchange
PHP	PHP: Hypertext Processor
RPC	Remote Procedure Call
SQL	Structured Query Language
XHTML	eXtensible HyperText Markut Language
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language Transformations

SUMÁRIO

1 INTRODUÇÃO

Com o desenvolvimento científico das últimas décadas, novas tecnologias foram desenvolvidas e, a cada dia, muitas outras estão surgindo. Juntamente com os benefícios que essas tecnologias nos fornecem, observa-se o aumento da complexidade de gerenciamento e operação destas. Para contornar estas dificuldades, alguns conceitos dentro da Computação têm se destacado, como a computação Ubíqua e Pervasiva (*Ubiquitous/Pervasive Computing*) (SATYANARAYANAN, 2001), que prevê um ambiente impregnado de computação, de forma invisível ao usuário final, onde diversos dispositivos digitais estejam integrados, trabalhando de forma colaborativa e pró-ativa, no auxílio às atividades do dia-a-dia dos usuários.

Para atingir a Computação Pervasiva/Ubíqua desejada, várias áreas de estudos estão fazendo suas contribuições, como a Computação Sensível ao Contexto (*Context-Aware Computing*) (SCHILIT et al., 1994) que cria redes de sensores-atuadores para desempenho de funções específicas e atendimento pró-ativo aos usuários. Essas tecnologias podem ser utilizadas em áreas como segurança, onde se fazem indispensáveis os sistemas de computação devido a estes oferecerem a possibilidade de automação e redução de custos.

Nesse sentido, o projeto CONVERGE UFSM, em desenvolvimento no Grupo de Sistemas de Computação Móvel (GMob) e no Laboratório de Sistemas de Computação (LSC) tem por objetivo promover uma convergência das tecnologias voz sobre IP (*VoIP*) (GOODE, 2002), redes sem sio (RAPPAPORT, 1996) e PABX (RYU et al., 1995) na implementação de um sistema de telefonia digital, que tem também funcionalidade de sistema de controle de acesso e central de alarme.

Com base neste cenário, o principal objetivo deste trabalho é o desenvolvimento de uma interface *web* para gerenciamento e configuração da rede de sensores-atuadores que

faz parte do sistema proposto pelo projeto CONVERGE UFSM. Esta interface visa facilitar as tarefas de visualização dos eventos capturados pelo sistema, além de tornar possível a configuração de determinadas regras que definem os eventos e as ações realizadas pelo sistema, apresentando essas funcionalidades através de um ambiente amigável e eficaz para os usuários que detêm as devidas permissões.

O restante deste texto está organizado da seguinte forma. O capítulo 2 apresenta uma revisão da literatura sobre implementação de aplicações web, apresentando a arquitetura MVC, além da técnica de desenvolvimento AJAX e o *framework* GWT. O capítulo 3 apresenta a arquitetura e os objetivos do projeto CONVERGE UFSM. No capítulo 4, descreve-se a interface web proposta, detalhando as funcionalidades a que se propõe sua utilização, juntamente com uma descrição de sua implementação. O capítulo 5 apresenta uma avaliação da aplicação desenvolvida. Por fim, no capítulo 6, apresentam-se a conclusão do trabalho e algumas sugestões de atividades futuras.

2 FERRAMENTAS PARA IMPLEMENTAÇÃO DE APLICAÇÕES WEB

Neste capítulo, apresenta-se uma revisão de literatura sobre padrões, técnicas e ferramentas de implementação de aplicações *web*. Primeiramente, uma revisão sobre o padrão de arquitetura de *software* MVC, destacando-se pontos referentes ao modelo. Posteriormente, é revisada uma técnica de desenvolvimento de aplicações *web* interativas conhecida como AJAX. Por fim, é revisada a estrutura do GWT, um *framework* para implementação de aplicações AJAX e que incorpora o conceito MVC.

2.1 Arquitetura MVC

MVC (*Model-View-Controller*) consiste em um padrão de arquitetura de *software* que tem por objetivo prover um design claro para separar diferentes responsabilidades em uma aplicação interativa (VEIT; HERRMANN, 2003). Aplicar o paradigma MVC torna possível o desacoplamento de três componentes de uma aplicação: acesso aos dados, lógica de negócio e interação com o usuário. Essa separação em camadas é capaz de realizar a separação física dos componentes de *software*. Desta maneira, a organização em camadas é a chave para a independência entre os componentes, objetivando agrupar componentes por responsabilidades em comum (FRAGMENTAL, 2007).

A fundamentação da divisão das funcionalidades de um sistema em camadas surgiu como alternativa para solucionar alguns problemas existentes nas aplicações monolíticas. Nessas aplicações, dados e códigos eram armazenados em uma mesma máquina, na qual todas as funcionalidades eram definidas em um único módulo contendo uma grande quantidade de linhas de código e de difícil manutenção (MACORATTI, 2007).

Uma aplicação complexa, que apresenta uma grande quantidade de dados ao usuário, pode ser usualmente separada em três camadas: interface com usuário (*view*), domínio

lógico (*controller*) e acesso aos dados (*model*)(LEFF; RAYFIELD, 2001). Essa divisão pode ser observada na figura 2.1.

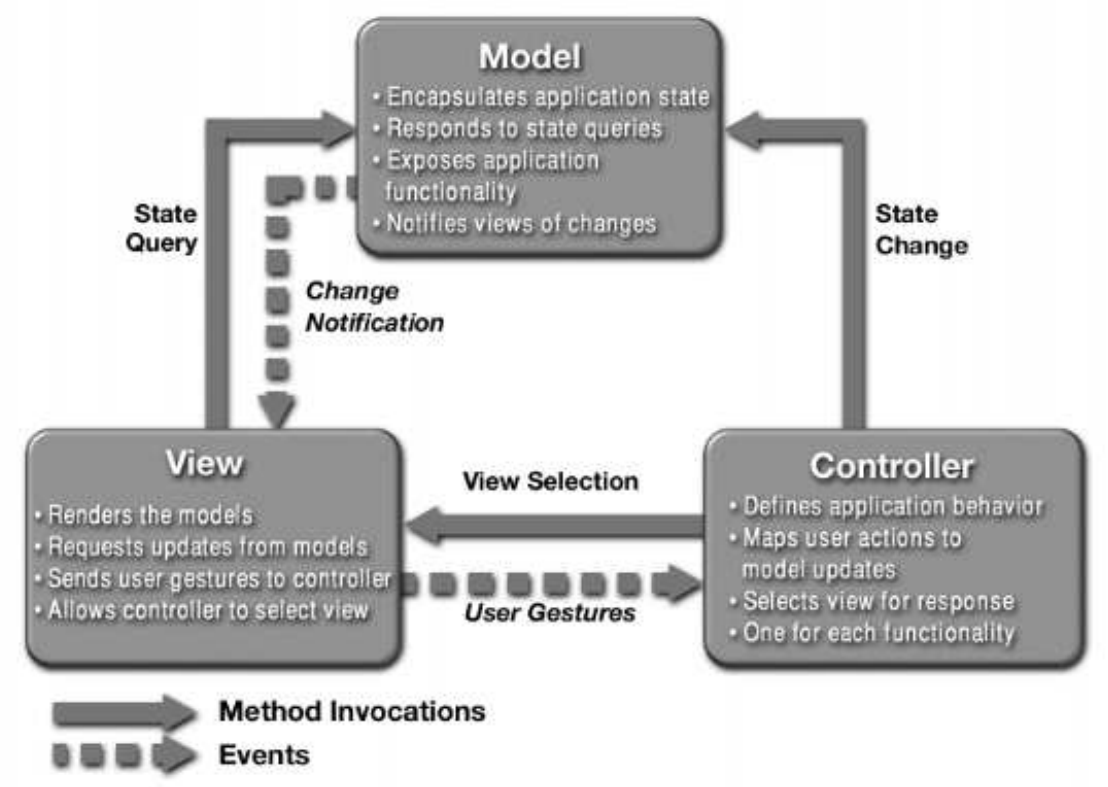

Figura 2.1: O modelo MVC original (ECKSTEIN, 2007)

Porém, nas aplicações *web*, o padrão MVC sofre algumas alterações em relação ao padrão original descrito por Gamma (GAMMA et al., 1995). No padrão original, a camada *view* é notificada quando modificações na camada *model* ocorrem. Além disso, todas as camadas podem se comunicar entre si, o que não acontece nas aplicações *web* devido ao desacoplamento entre cliente e servidor. O padrão MVC modificado para utilização no desenvolvimento de aplicações *web* e a interação entre as camadas são apresentados na figura 2.2, onde ocorrem as seguintes ações:

1. Um navegador envia uma requisição ao *controller*;
2. O *controller*, então, interage com o *model* para obter os dados necessários;
3. O *model* consulta essas informações na base de dados (3) e, então, fornece-as para o *controller* (4);
4. O *controller*, então, envia os dados para serem renderizados pelo *view* (5);
5. O *view*, finalmente, gera a saída e a envia para o navegador (6).

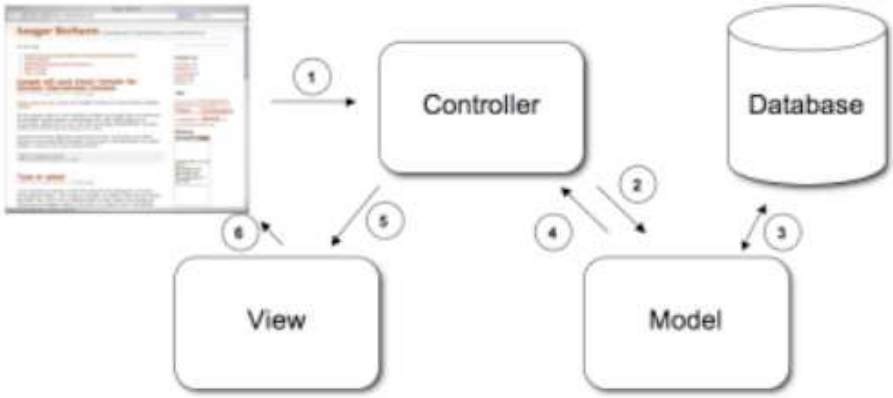

Figura 2.2: Arquitetura MVC aplicada ao ambiente *web* (BERHORN, 2007)

2.1.1 Camada *Model*

A camada *model* representa o estado da aplicação. É responsável por fazer a interface da aplicação com a fonte dos dados, geralmente um banco de dados. Quando é necessário salvar o estado da aplicação, é através dessa camada que as informações manipuladas pelo sistema podem ser armazenadas na base de dados.

Os objetos da camada *model* são responsáveis pela representação do domínio da informação em que a aplicação opera. O domínio lógico acrescenta significado aos dados. Possíveis modificações na fonte dos dados, implicarão em um menor impacto sobre a aplicação, visto que apenas o objeto envolvido com essa fonte necessita ser modificado, ficando evidenciadas as vantagens da adoção de tal padrão. O paradigma MVC não cita, especificamente, a camada para acesso aos dados, porque subentende-se que estes métodos estariam encapsulados pelo *model*.

2.1.2 Camada *View*

A camada *view* é, normalmente, composta pela GUI (*Graphical User Inteface* - Interface Gráfica do Usuário) e tem a responsabilidade de apresentar a aplicação ao usuário. Esta camada pode ser implementada de diversas formas, desde focada para a *web* até voltada para o *desktop*.

Em aplicações web, a camada *view* é responsável pela transformação da camada *model* em uma forma apropriada para interação, que na maioria dos sistemas *web* é construída através de páginas HTML para conteúdo estático, ou de páginas com conteúdo dinâmico, como páginas JSP, ASP, PHP.

2.1.3 Camada *Controller*

A camada *controller* é responsável pelo processamento e resposta a eventos, geralmente ações do usuário, e, ainda, pode invocar alterações no *model*, dependendo do fluxo determinado pela camada. Logo após o processamento, o *controller* redireciona as informações para a camada *view*, possivelmente com os resultados deste processamento, de modo que o usuário possa seguir utilizando o *software*.

Em aplicações voltadas para a *web*, a camada *controller* é responsável por atuar como intermediária entre o componente *view* e o componente *model*. Além disso, nessas aplicações, essa camada costuma ser implementada através de *Servlets* (COWARD, 2001).

Nos sistemas *web*, freqüentemente, opta-se por definir o fluxo da aplicação em um arquivo XML. Dessa maneira, a camada *controller* não sofre alterações caso, por exemplo, uma *view* sofra uma alteração no nome, e, então, as classes não precisam ser recompiladas.

2.2 AJAX

AJAX é um acrônimo consagrado muito recentemente por Jesse James Garret, da Adaptive Path, e significa *Asynchronous Javascript And XML* (JavaScript e XML assíncrono), porém o que existe é mais que a junção de JavaScript com XML, é todo um conceito de navegação e atualização de páginas *web*. Algumas partes descritas na definição de AJAX não são novas, as quais muitas vezes foram denominadas de DHTML (*Dynamic HTML* - HTML Dinâmico) e Script Remoto (SOARES, 2006).

O AJAX surgiu para resolver um problema que ocorre desde o surgimento da Internet, que é o de que a interação é feita de forma síncrona, ou seja, exige-se que, para cada solicitação em uma página *web*, atualiza-se a página inteira no navegador. Não importava se a atualização era apenas de uma pequena parte da página, toda a página era recebida pelo servidor e a página era inteiramente redesenhada e retornada para o navegador.

2.2.1 Estrutura

Com o AJAX, pode-se trafegar apenas os dados que realmente foram atualizados em uma página *web* e, sendo assim, ganhar em desempenho e interatividade com o usuário. O seu uso é recomendado, principalmente, para páginas que utilizam validações e preenchimento de formulários.

Praticamente todos os navegadores, em suas últimas versões, são compatíveis com as técnicas utilizadas pelo AJAX, viabilizando a sua utilização em qualquer tipo de aplicação *web*. Na figura 2.3, pode-se visualizar a estrutura tradicional da *web* (parte superior), com o fluxo das informações, sendo transferida toda a página *web* entre o servidor e o navegador cada vez que é requisitada qualquer informação da página. Já na parte inferior desta figura, observa-se um fluxo de informações utilizando AJAX. Neste caso, o navegador interage com o servidor, passando apenas as informações requisitadas e apenas nos momentos que realmente são necessárias, evitando, assim, o tráfego de toda a página, a cada requisição.

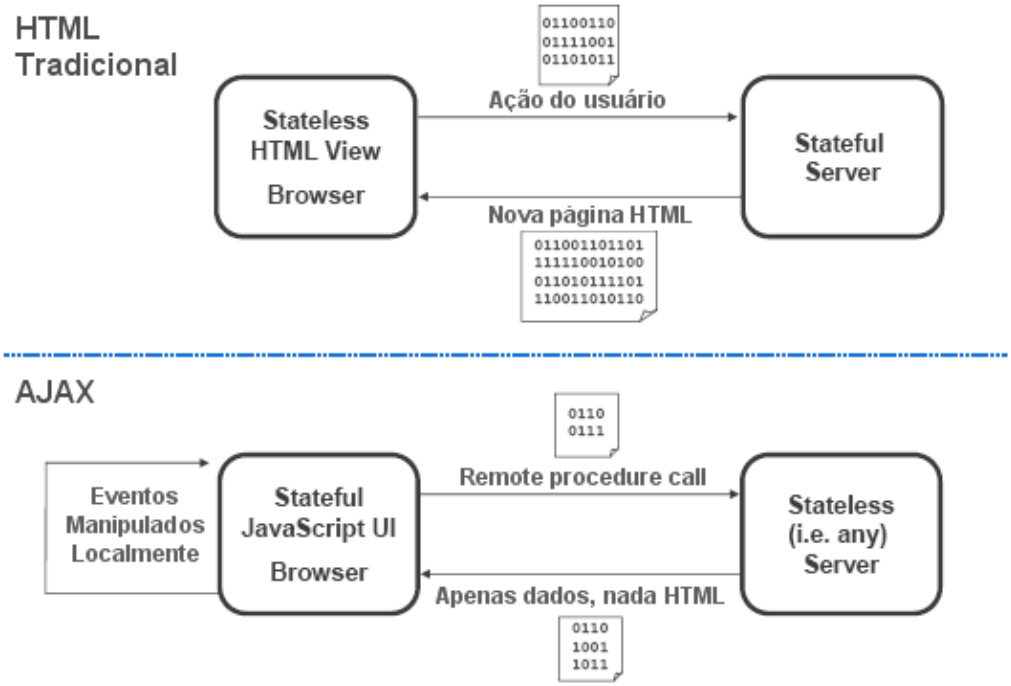

Figura 2.3: Estrutura de uma aplicação *web* tradicional comparada à estrutura de uma aplicação AJAX (JOHNSON, 2006)

Na figura 2.4, percebe-se bem a diferença entre os dois modelos e os módulos envolvidos. À esquerda, no modelo tradicional, tem-se a interface do usuário interagindo diretamente com o servidor *web* para enviar a página inteira para o servidor e depois receber uma nova página para ser apresentada ao usuário, que fica aguardando o retorno sem conseguir fazer qualquer outra operação na página. À direita, no modelo AJAX, a interface do usuário interage com um mecanismo AJAX que atua como intermediário entre as solicitações do cliente e do servidor, apenas passando as informações realmente necessárias de ambos os lados, permitindo, assim, que o usuário, quando requisitar algo

da página, possa continuar trabalhando nela enquanto o resultado ainda não é apresentado.

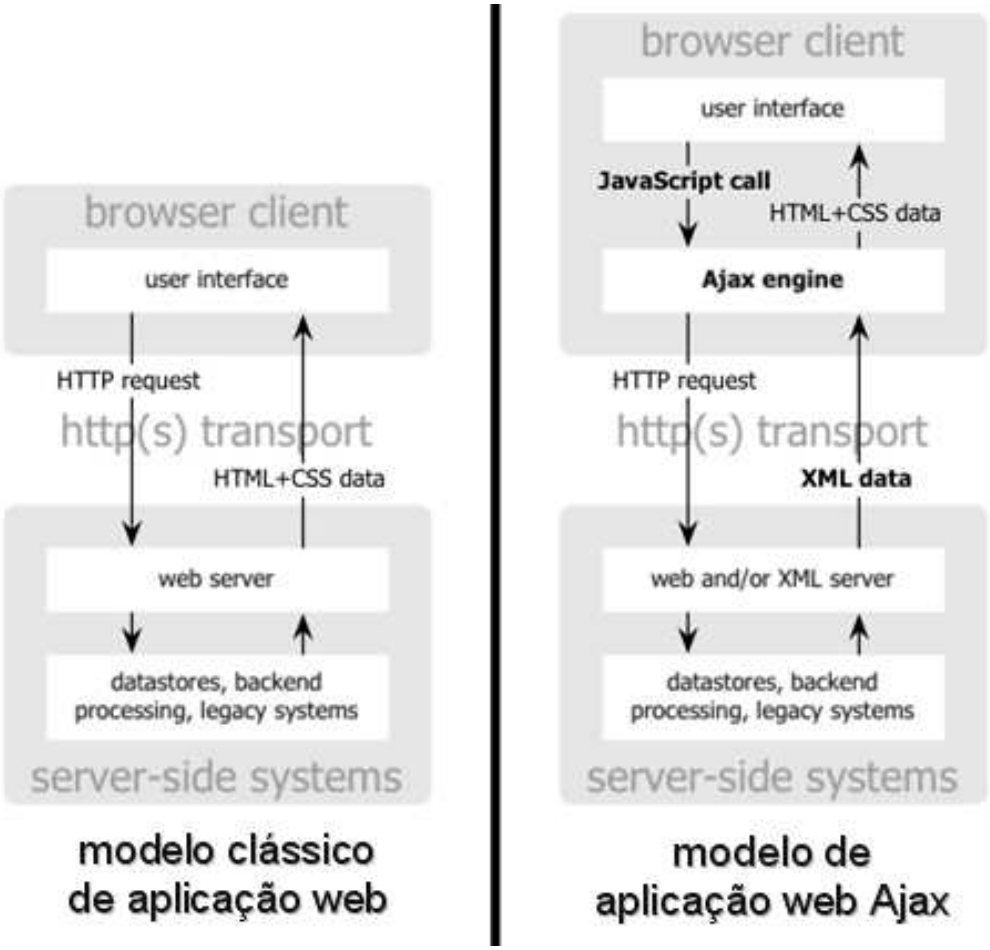

Figura 2.4: O modelo tradicional para aplicações *web* comparado ao modelo de uma aplicação AJAX (GARRETT, 2005)

2.2.2 Tecnologias envolvidas

AJAX não é uma tecnologia por si só, na verdade, é um termo utilizado para a reunião de diversas tecnologias. Ele incorpora e utiliza as tecnologias abaixo descritas:

- padrões de apresentação utilizando XHTML e CSS;
- apresentações dinâmicas e interativas utilizando DOM;
- manipulação de elementos de uma página utilizando XML e XSLT;
- recuperação de dados assíncrona usando XMLHttpRequest;
- e JavaScript, que agrega todos esses elementos. (GARRETT, 2005)

2.3 *Framework* GWT

A arquitetura cliente-servidor tem sofrido uma grande quantidade de mudanças em um curto período de tempo. Anteriormente, cada aplicação possuía um *software* cliente diferente, com o *software* servindo como uma interface com o usuário. Esse *software* deveria ser instalado individualmente em cada cliente, e necessitava ser atualizado cada vez que a aplicação sofria alterações (PRABHAKAR, 2007).

Atualmente, com o desenvolvimento de aplicações para a Internet, é possível utilizar navegadores para acessar nossas aplicações de qualquer lugar. Essa foi uma enorme mudança, mas ainda surgem assuntos sobre performance e aplicações que não tem a mesma receptividade que as aplicações *desktop*. Além disso, escrever aplicações *web* dinâmicas é um processo desgastante e propício a uma série de erros. Muito tempo de desenvolvimento é perdido em problemas de incompatibilidade entre navegadores e plataformas (GOOGLE, 2007).

Com o surgimento da técnica de desenvolvimento AJAX, torna-se possível a construção de páginas *web* que podem competir com aplicações *desktop* em receptividade e aparência. O *Google Web Toolkit* (GWT) é uma ferramenta de código aberto que torna mais fácil o projeto e desenvolvimento de aplicações AJAX, utilizando-se apenas a linguagem de programação Java.

2.3.1 Desenvolvimento

Ao utilizar o *framework* GWT, os desenvolvedores podem rapidamente desenvolver e depurar aplicações AJAX na linguagem Java, usando a IDE (*Integrated Development Environment* - Ambiente Integrado de Desenvolvimento) de sua preferência. Essas aplicações podem ser executadas em dois modos:

- *Hosted mode*: A aplicação é executada como Java *bytecode* pela JVM (*Java Virtual Machine* - Máquina Virtual Java). Este modo é, geralmente, usado para desenvolvimento;
- *Web mode*: A aplicação é executada como JavaScript puro e HTML, compilada a partir de um código Java. Modo utilizado para implantação.

2.3.2 Arquitetura

O *framework* GWT é constituído de quatro grandes componentes: um compilador Java-to-JavaScript, um navegador e duas bibliotecas Java, como pode ser visualizado na figura 2.5. As funcionalidades desses componentes são:

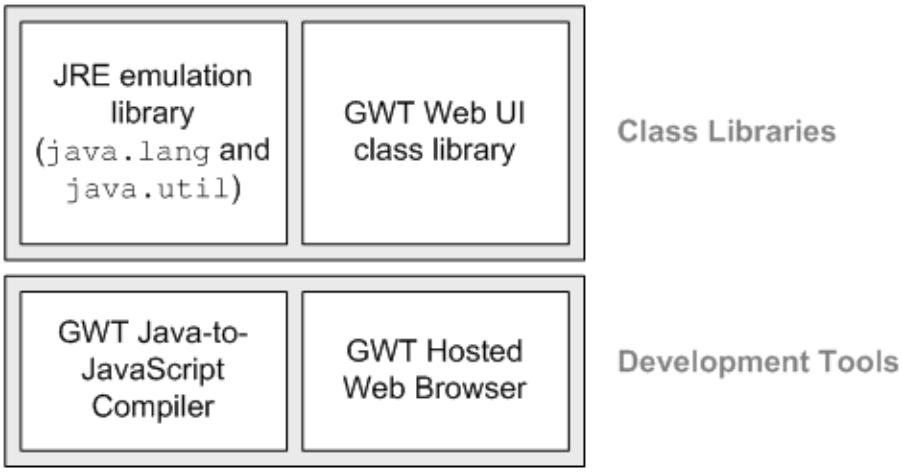

Figura 2.5: Arquitetura do GWT (GOOGLE, 2007)

- Compilador GWT Java-to-JavaScript: converte a aplicação escrita na linguagem de programação Java para a linguagem de programação JavaScript. O compilador GWT é usado para compilar as aplicações que são executadas em *web mode*;
- Navegador GWT: Permite a execução das aplicações GWT no modo *hosted mode*, onde o código é executado pela Máquina Virtual Java (JVM) sem a necessidade de compilar o código para JavaScript;
- Biblioteca de emulação JRE: O *framework* GWT contém implementações em JavaScript das classes mais usadas pelas bibliotecas do JDK (*Java Development Kit* - Kit de desenvolvimento Java) versão 1.4, incluindo a maioria das classes do pacote *java.lang* e um subconjunto de classes do pacote *java.util*. O restante das bibliotecas presentes no JDK 1.4, ou nas versões mais recentes, não são suportadas nativamente pelo GWT;
- Biblioteca GWT para interface com usuário: possui uma série de interfaces e classes que possibilitam a criação de *widgets*, como botões, imagens e texto. Essa biblioteca contém o núcleo da interface com o usuário e é usada para criar as aplicações GWT.

2.3.3 GWT e o padrão MVC

Como pode ser visualizado na figura 2.6, toda a parte de interface com o usuário (camada *view* do modelo MVC) situa-se na parte cliente, e é implementada utilizando-se as bibliotecas de desenvolvimento do GWT, onde estas possibilitam a utilização de uma série de componentes visuais. Quando o cliente solicita alguma informação ao servidor, essa requisição é efetuada através de RPC (*Remote Procedure Call* - Chamada Remota de Procedimento) de forma assíncrona. A API GWT *Servlet* oferece diversos métodos e interfaces que possibilitam essa comunicação, como por exemplo, interface de serialização de objetos, interface para manipulação de requisições assíncronas, por exemplo.

No lado servidor da aplicação AJAX, com a utilização da API GWT *Servlet*, é possível, então, utilizar as mais diversas ferramentas e APIs disponíveis para a linguagem de programação Java, como, por exemplo, *frameworks* para manipular informações contidas em banco de dados, estabelecer comunicação com outros sistemas, por exemplo. Neste lado servidor, o GWT permite, claramente, a separação das camadas *controller*, que irá responder a eventos do usuário, e *model*, que será requisitada para operar com as informações em alguma fonte de dados, e, então, fornecê-las para a camada *controller*.

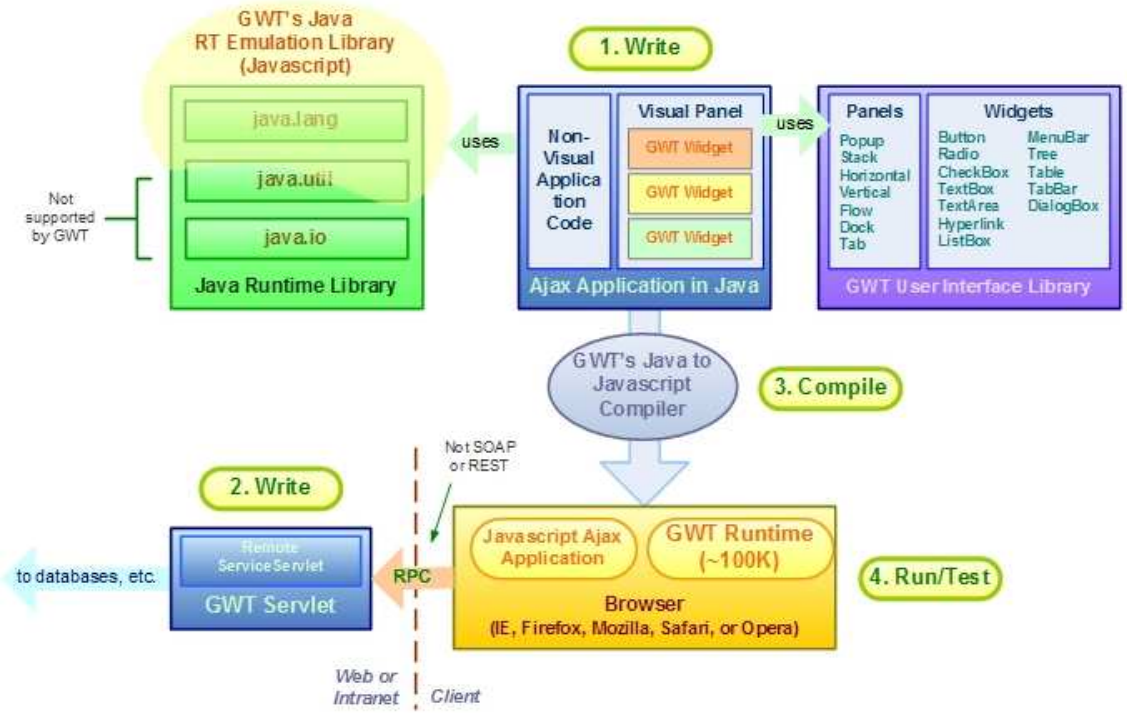

Figura 2.6: Interação entre os componentes do GWT (DESK, 2006)

3 PROJETO CONVERGE UFSM

O projeto CONVERGE UFSM, com período de execução de março/2007 a março/2008, está sendo desenvolvido pelos grupos de pesquisa GMob (Grupo de Sistemas de Computação Móvel) e LSC (Laboratório de Sistemas de Computação). Este projeto propõe uma aplicação de convergência de tecnologias para controle de acesso aos laboratórios nas instituições, com o intuito de melhorar o sistema de segurança atual da UFSM - Campus, que não prevê câmeras ou outro mecanismo para segurança dos vários laboratórios da instituição, que contêm recursos que devem ser resguardados. Acredita-se que esta seja também a realidade em muitas outras instituições.

3.1 Objetivos

O principal objetivo do CONVERGE é o desenvolvimento de uma solução de baixo custo para a segurança dos laboratórios da Universidade Federal de Santa Maria através da integração da central telefônica analógica com *VoIP*, e destes com uma rede sem fio, uma central de alarme e um sistema de controle de acesso. Esta solução também demonstra que a tecnologia *VoIP* pode ser utilizada como plataforma base para o desenvolvimento de outras aplicações que anexam novas funcionalidades aos aparelhos telefônicos. Deve-se observar que este sistema de segurança poderia ser utilizado, também, em residências e outros estabelecimentos. A grande proliferação dos computadores pessoais fez com que a maioria das casas possua, pelo menos, um computador, hoje em dia. Assim, esta solução de segurança poderia ser facilmente expandida para residências, onde ao invés de se desligar o computador ao sair, por exemplo, este poderia ser alocado para a execução dos programas de controle de acesso e alarme; servindo, assim, para proteção do patrimônio físico do local, a um baixo custo.

3.2 Arquitetura

O modelo proposto para segurança dos laboratórios parte da utilização da estrutura telefônica já existente como dispositivo de acesso ao local. Esta abordagem diminui o custo de implantação do sistema e possibilita uma integração entre diversos dispositivos, tanto sensores como atuadores, um dos objetivos do projeto.

A figura 3.1 esquematiza a solução de integração das tecnologias de *PABX* - *VoIP* - dispositivos sem fio sob o ponto de vista físico (*hardware*). O sistema é formado por uma rede de sensores-atuadores que executam as funcionalidades programadas. Nessa rede, sensores podem ser de dois tipos: (i) sensores de *hardware*, como dispositivos de detecção de presença, por exemplo; (ii) sensores de *software*, como um *software* de detecção de movimento de uma câmera de vídeo. Atuadores também podem ser de dois tipos: (i) atuadores de *hardware*, com um mecanismo que abre uma porta por exemplo; (ii) e atuadores de *software*, como uma mensagem por e-mail, uma mensagem por telefone ou programas específicos desenvolvidos por terceiros.

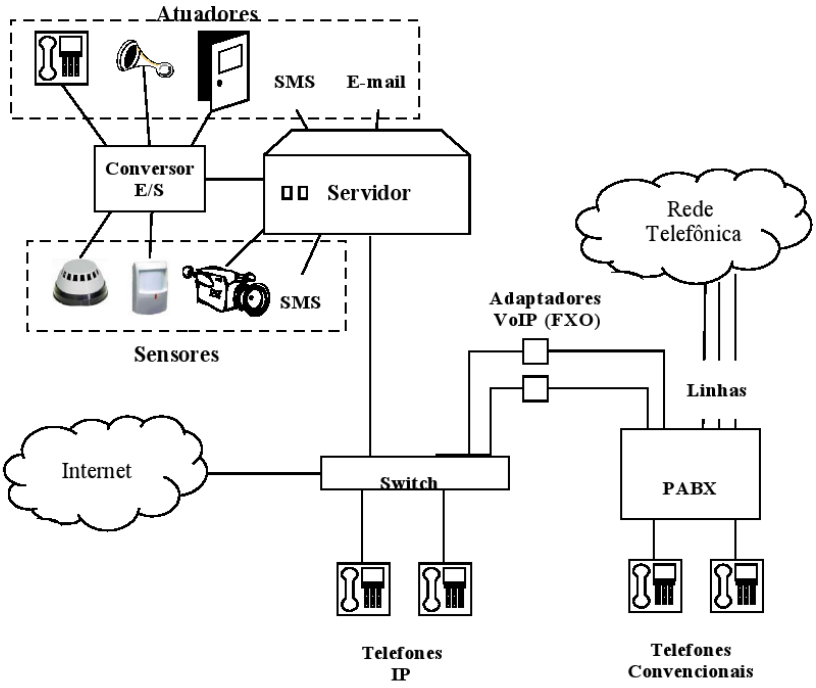

Figura 3.1: Projeto CONVERGE - Solução integrada - Aspectos físicos

O *software* para implantação de *VoIP*, ASTERISK (SPENCER, 1999), pode atuar como sensor ou atuador de *software*, dependendo da função ser geradora de informação ou consumidora de informação (ação). Com a adição de adaptadores *VoIP* do tipo *FXS*, que são adaptadores que fornecem uma interface entre um telefone analógico e o padrão

VoIP à proposta da figura 3.1, é possível também configurar o sistema como uma ponte (*gateway*) entre *VoIP* e telefonia convencional.

Sob o ponto de vista de *software*, como é apresentado na figura 3.2, sensores e atuadores são reunidos através de regras de interpretação, que contêm a inteligência do sistema, e usam o banco de dados com informações sobre os usuários, como papéis que os usuários exercem (perfil), dispositivos, regras de interpretação. Os papéis que um usuário pode exercer estabelecem as permissões/restrições de acesso que este tem e as ações que pode executar. Como pode ser observado na figura 3.2, a camada de mais alto nível do sistema consiste em uma interface *web*, objeto deste trabalho, que será descrita no próximo capítulo.

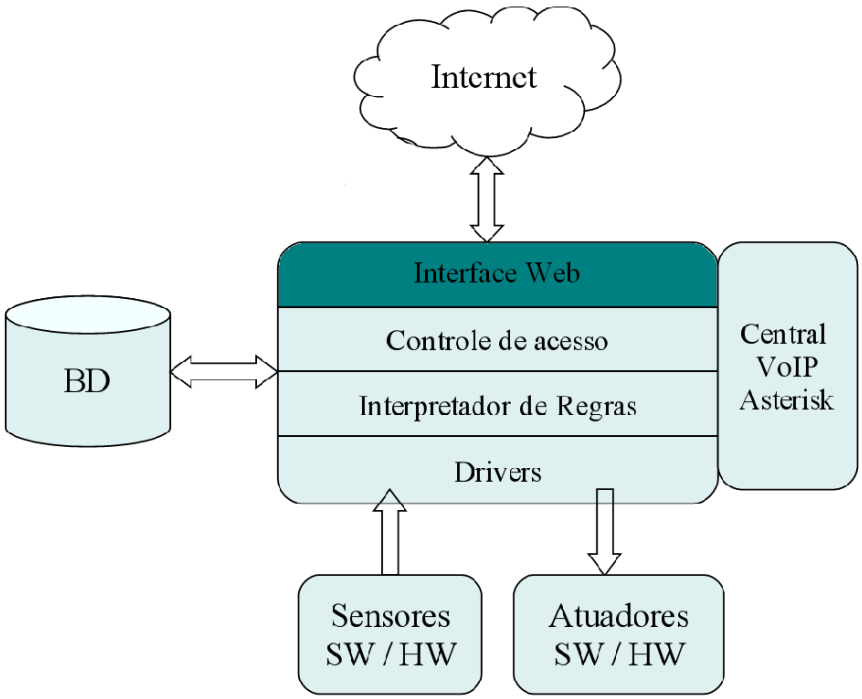

Figura 3.2: Projeto CONVERGE - Aspectos lógicos

3.2.1 Núcleo interpretador de regras

O núcleo interpretador de regras, em desenvolvimento em outro trabalho de graduação (OLIVEIRA, 2007), tem como objetivo analisar eventos gerados por sensores e a partir destas informações comandar a ativação de atuadores. As regras registram quais eventos ou combinação destes define uma situação a ser reconhecida, e quais ações devem ser realizadas frente a esta. A estrutura proposta para núcleo interpretador de regras pode ser visualizada na figura 3.3.

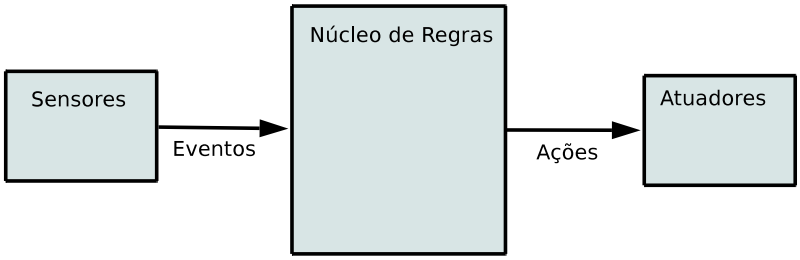

Figura 3.3: Máquina de Regras

A fim de padronizar e facilitar a comunicação dos sensores e atuadores com a máquina de regras, existem dois módulos auxiliares encarregados de fazer a tradução dos eventos gerados pelos sensores, e a tradução das ações a serem submetidas para os atuadores. Os modelos propostos para os módulos auxiliares descritos são apresentados nas subseções seguintes.

3.2.2 Tradutor de eventos

O tradutor de eventos tem por função receber os eventos gerados pelos sensores, fazer a tradução para o formato de eventos padrão, e enviar estes para a máquina de regras, como pode ser visualizado na figura 3.4. O arquivo de configuração que a figura apresenta contém as informações sobre os sensores, os associando à qual evento representam. Assim, quando existir uma necessidade de alteração na configuração dos sensores pelo administrador do sistema, esta pode ser facilmente realizada através da edição deste arquivo.

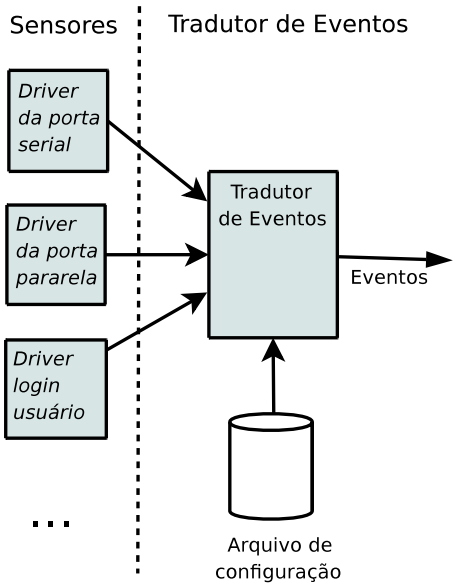

Figura 3.4: Tradutor de Eventos

3.2.3 Tradutor de ações

O tradutor de ações tem uma funcionalidade semelhante ao de eventos, traduzindo informações através de uma arquivo de configuração pré-definido. Essas informações, no entanto, são ações que a máquina de regras deseja efetuar, e o trabalho é mapear qual atuador, seja ele de *software* ou de *hardware*, corresponde a cada ação a ser executada. Esta estrutura pode ser observada na figura 3.5.

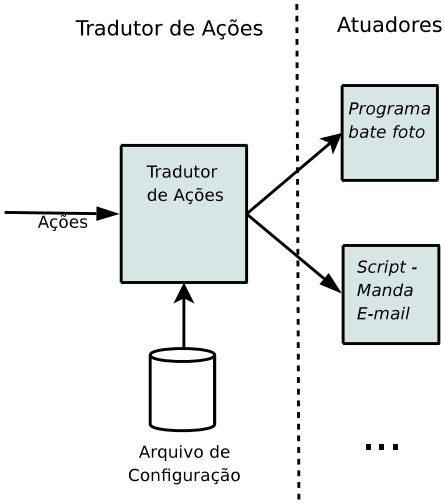

Figura 3.5: Tradutor de Ações

4 IMPLEMENTAÇÃO DO MÓDULO DE INTERFACE WEB DO PROJETO CONVERGE UFSM

Este capítulo descreve o módulo de interface *web* do projeto CONVERGE UFSM desenvolvido neste trabalho. Inicialmente, apresenta-se uma visão geral deste módulo, com suas funcionalidades. Por fim, apresentam-se as principais etapas e decisões do processo de implementação.

4.1 Funcionalidades da Aplicação

Todas as funcionalidades previstas para a interface *web* visam centralizar e facilitar as tarefas de abstração de tecnologias exigidas pelo projeto CONVERGE UFSM. Tais funcionalidades podem ser divididas em operações básicas (executadas por todos os usuários cadastrados no sistema) e operações avançadas de gerenciamento (executadas apenas por usuários administradores).

Como operações básicas, entende-se as tarefas que compreendem pequenas alterações nas informações do usuário, como, por exemplo, a senha de acesso ao sistema. Ainda como procedimentos básicos, existem as funcionalidades de visualização de vídeo on-line e abertura da porta de acesso ao laboratório por parte do usuário autenticado no sistema.

Como operações avançadas, tem-se a administração do sistema, como, por exemplo, cadastro de novos usuários com permissões básicas ou avançadas, além do gerenciamento e configuração de todas as regras de interpretação que determinam a inteligência do sistema. Além dessas funcionalidades, entende-se como operações avançadas, também, a possibilidade do administrador visualizar todos os eventos que ocorrem no sistema, armazenados na forma de fotos e vídeos ou de registros gerados pela aplicação (*logs*).

Além desses recursos oferecidos pela aplicação, objetiva-se agregar uma boa acessibilidade à interface, dispondo de uma forma clara e objetiva os componentes para o

usuário.

4.2 Arquitetura

Atualmente, o *framework* GWT encontra-se na versão 1.4 para as plataformas *Windows*, *Mac OS X* e *Linux x86*. Além das APIs que compõem o GWT, para a implementação está-se utilizando bibliotecas adicionais, como *Hibernate* (BAUER; KING, 2005), *log4j* (APACHE, 2008) e *dom4j* (METASTAFF, 2001).

O processo de desenvolvimento da aplicação foi dividido em quatro etapas. Primeiramente, implementou-se o módulo de visualização de fotos e vídeos capturados pelo sistema. Posteriormente, implementou-se o módulo responsável pela autenticação de usuários no sistema, onde esses usuários possuem papéis que podem ser exercidos no sistema. Depois, desenvolveu-se o módulo de configurações, responsável por permitir a visualização e edição das regras de interpretação que alimentam a máquina de regras (OLIVEIRA, 2007) do sistema. Por fim, implementou-se o módulo de consulta do sistema, responsável por apresentar os registros de ponto e *logs* do sistema para o administrador. A figura 4.1 apresenta uma visão geral da interface *web*, juntamente com os módulos que a integram. As seções seguintes apresentam maiores detalhes sobre essas etapas de desenvolvimento.

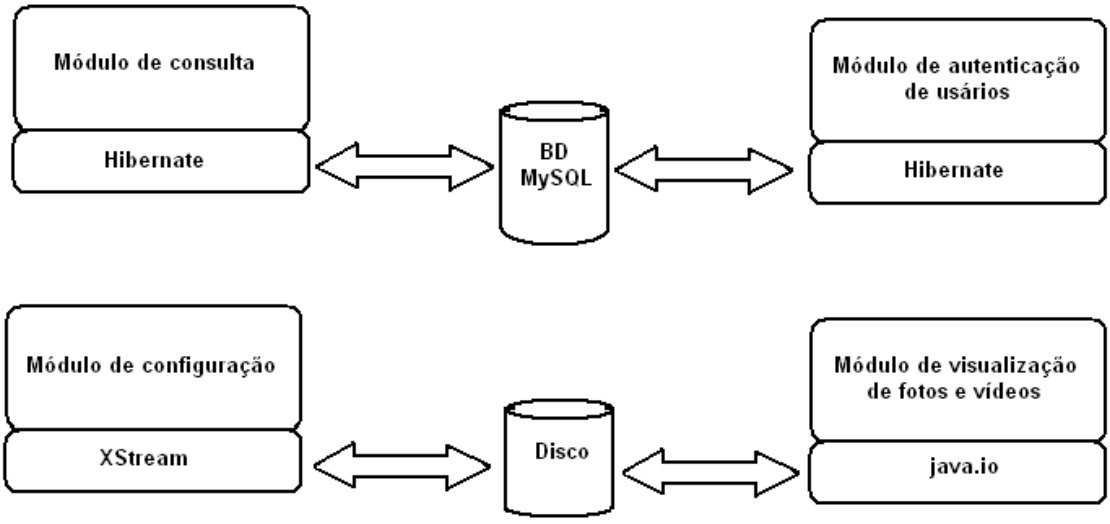

Figura 4.1: Módulo de interface *web* - Visão geral

4.2.1 Módulo de visualização de fotos e vídeos

As funcionalidades oferecidas por este módulo são: possibilitar a visualização das fotos e vídeos capturados pelo *software Motion* (VREEKEN, 1999) de uma forma ordenada

(exibindo dos eventos mais recentes aos eventos mais antigos), além de informar a data e horário em que o evento registrado ocorreu. Além disso, implementou-se um modo de fácil navegação, possibilitando ir diretamente aos eventos mais recentes ou aos eventos mais antigos registrados através de botões, como pode ser visualizado na figura 4.2.

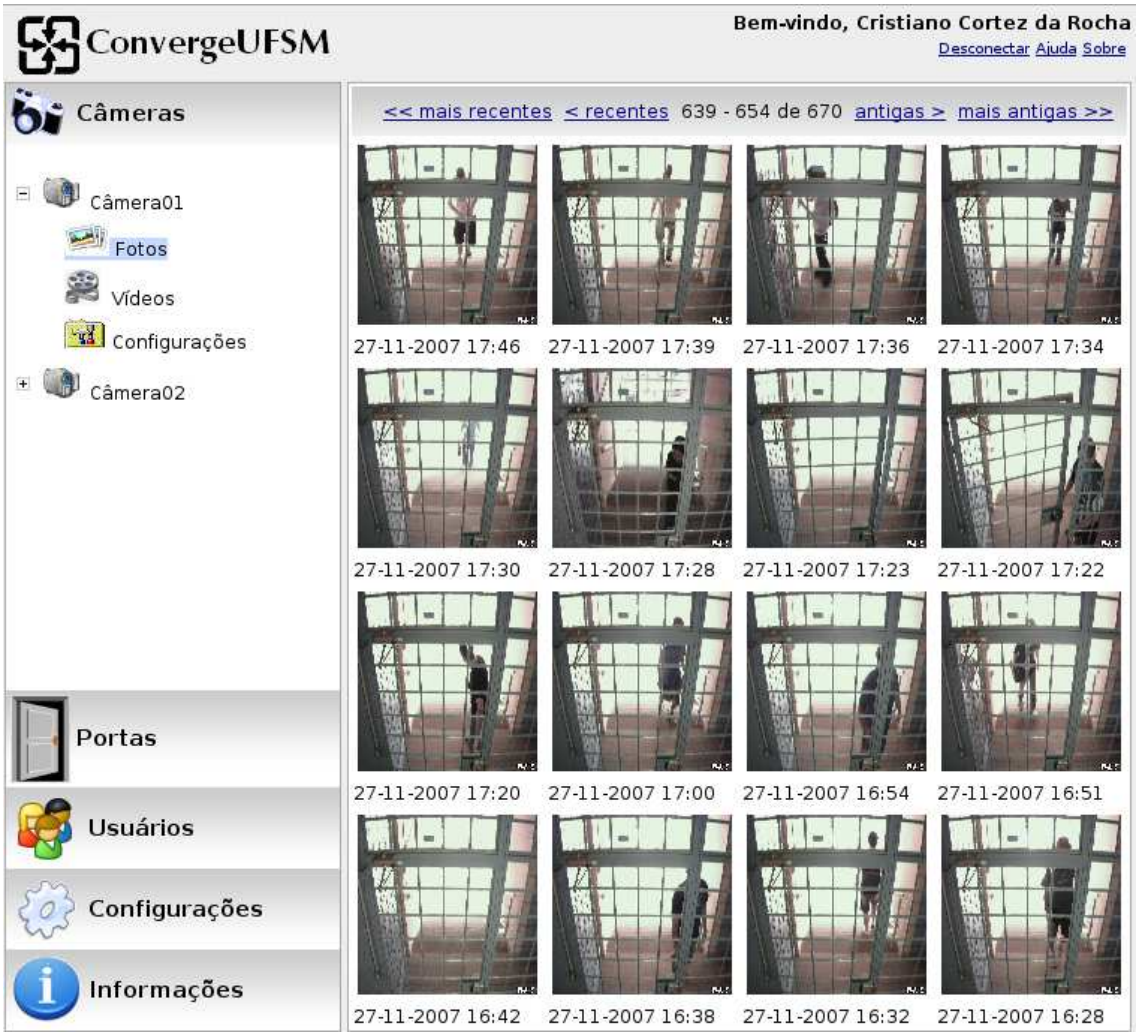

Figura 4.2: Interface *web* - Visualização de fotos

No módulo de visualização de eventos, fez-se necessário a utilização da biblioteca *GWT-Servlet*, pois as classes compatíveis com a parte cliente não oferecem suporte à manipulação de diretórios e arquivos, o qual é necessário para localização das fotos e vídeos solicitados. Com isso, criou-se um serviço responsável por responder a todas as requisições de fotos e vídeos. As classes e interfaces envolvidas na implementação do serviço de imagem podem ser observadas na figura 4.3.

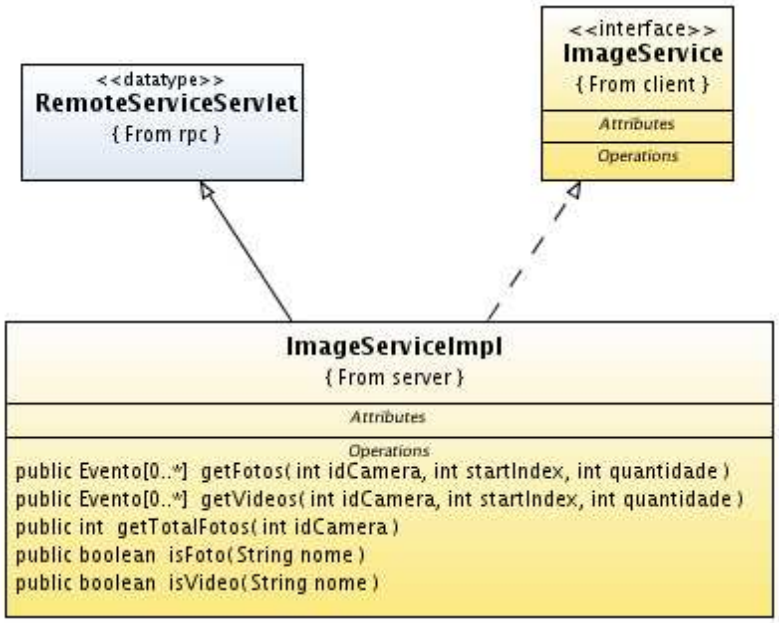

Figura 4.3: Diagrama de dependência - Serviço de imagem

4.2.2 Módulo de autenticação baseado em papéis

Primeiramente, estipularam-se os diferentes tipos de usuários que haveriam no sistema e que ações estes usuários poderiam exercer no ambiente. O sistema proposto possui, atualmente, dois tipos de usuários: usuários administradores e usuários comuns. A tabela 4.1 apresenta os tipos de usuários e as permissões que estes possuem dentro do sistema.

Tabela 4.1: Tipos de usuários e suas permissões no sistema

Permissões	Usuário Administrador	Usuário Comum
Abertura de porta	Sim	Sim
Edição de configuração de eventos/ações	Sim	Não
Visualização de eventos (fotos e vídeos)	Sim	Não
Visualização de *logs*	Sim	Não
Visualização on-line de vídeo	Sim	Sim
Troca de senha de acesso	Sim	Sim

Posteriormente, optou-se pela utilização de banco de dados para guardar as informações sobre os usuários, pois o sistema possui um sistema de registro de ponto. Então, ao utilizar-se um banco de dados, consegue-se alcançar diversas facilidades, como velocidade e simplicidade para realização de consultas para geração de relatórios, como por exemplo, consulta aos registros de entrada e/ou saída de usuários em um determinado período de tempo.

Optou-se pelo banco de dados *MySQL* (AXMARK; LARSSON, 1995) versão 5.0.45

pelo fato de ser um *software* gratuito, possuir uma documentação bem acessível e exigir um pequeno poder de processamento, comparado a outros banco de dados. Visto que, o *MySQL* é uma base de dados relacional e a aplicação *web* descrita neste trabalho é implementada com uma linguagem de programação orientada a objetos, faz-se necessário um mapeamento entre as tabelas do banco de dados e os objetos da aplicação. Com a finalidade de tornar esses dois paradigmas compatíveis, optou-se pela utilização da ferramenta de Mapeamento Objeto Relacional (ORM - *Object Relational Mapping*) *Hibernate* versão 3.2.5. Através dessa ferramenta, é possível a abstração do conceito de tabelas do banco de dados e, então, manipular apenas objetos. Neste caso, esse objeto que acessa os dados segue o padrão DAO (*Data Access Object* - Objeto de Acesso aos Dados).

Após a criação das tabelas na base de dados, iniciou-se o processo de implementação do serviço de usuário (*UsuarioService*), responsável por responder a todas as requisições assíncronas de *login*, cadastros de novos usuários e alterações de informações de usuários já existentes no sistema. Visto que a classe que implementa o serviço realiza diversas operações com banco de dados, novamente, fez-se necessária a utilização da API *GWT-Servlet* para tornar possível o uso da biblioteca do *framework Hibernate*. Após concluído o desenvolvimento desta classe, partiu-se para a implementação das classes *CadastraUsuario*, *TrocaSenha* e *AutenticaUsuario*, que possuem os componentes de interface com o usuário necessários para obter os dados a serem enviados ao serviço de usuário para realizar a operação desejada. As classes e interfaces envolvidas na implementação do serviço descrito podem ser observadas na figura 4.4.

4.2.3 Módulo de configuração

O módulo de configuração foi desenvolvido buscando dois objetivos principais: possibilitar o cadastro de novas regras que alimentarão a máquina de regras e tornar possível o processo de manutenção dessas regras (atividades de exclusão e edição). Essas regras consistem em um conjunto de condições que determinam certos eventos e um conjunto de ações individuais a serem realizadas na detecção dos eventos ocorridos. Pode-se citar, como objetivos secundários, a possibilidade de cadastro de novas variáveis globais, possibilitando também a manutenção destas.

Inicialmente, estudou-se a modelagem proposta para a máquina de regras, analisando a estrutura das ações e condições para poder iniciar o processo de projeto da GUI. Essa

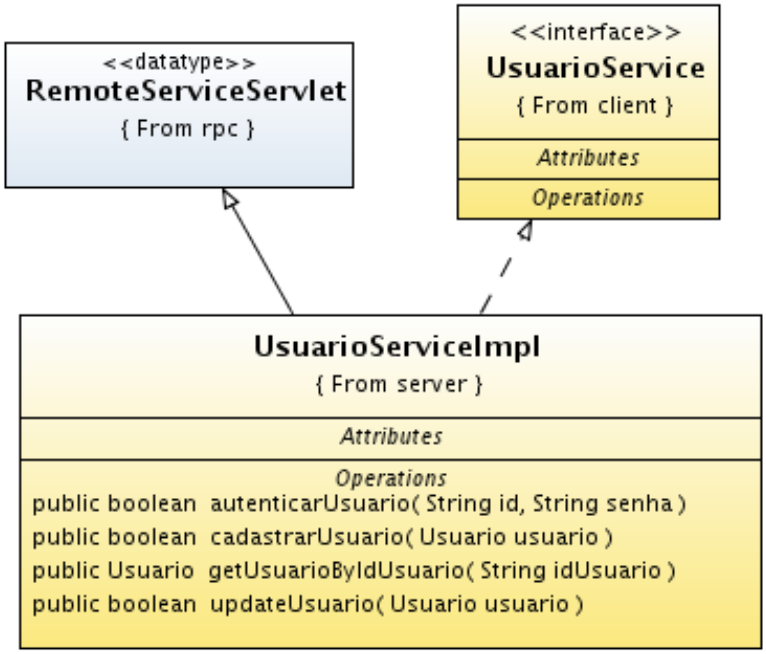

Figura 4.4: Diagrama de dependência - Serviço de usuário

interface de configuração modelada, como pode ser visualizado na configuração exemplo da figura 4.5, possibilita a inserção de quantas expressões o usuário achar necessário para que a condição seja adequada, tornando possível, também, a exclusão de uma expressão que não achar conveniente, sem necessidade de um recarregamento da página de configurações. Visto que a ordem de execução das ações pode interferir no resultado desejado, a interface torna possível a inserção de ações individuais na posição desejada.

Com a finalidade de facilitar a tarefa de configuração do sistema por parte do administrador, é possível este efetuar o cadastro de uma nova variável que seja necessária para a regra em questão, possibilitando, ainda, a determinação de um valor inicial para esta variável, sem a necessidade do usuário sair e perder os campos já editados da regra para efetuar o cadastro da variável desejada. Além disso, conforme a modelagem realizada para a máquina de regras, existem apenas dois tipos de ações possíveis: alterar variável e efetuar a chamada de um programa ou *script*. Então, quando é selecionada a opção de alterar variável, pelo dinamismo oferecido pelo AJAX, pode-se mostrar uma lista com todas as variáveis existentes no sistema, e, então, selecionar a variável desejada e atribuir o seu valor no campo ao lado. Porém, quando a outra opção é selecionada, um campo de texto é mostrado no lugar das variáveis possíveis para que o administrador possa escrever o nome do programa ou *script* a ser invocado, passando, no campo de texto ao lado, os parâmetros necessários.

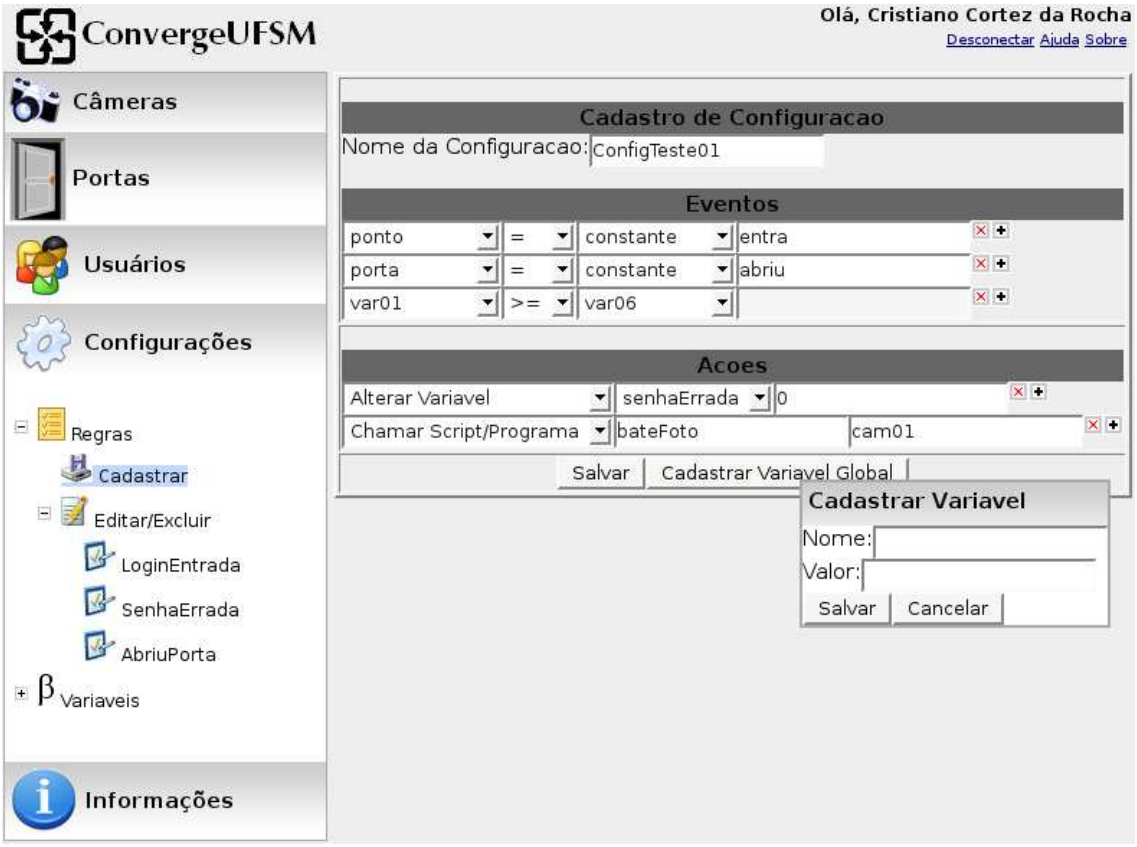

Figura 4.5: Interface *web* - Configuração de condições/ações

Para a persistência das regras, utilizou-se a ferramenta *XStream* (WALNES, 2003), versão 1.2.2, para persistir o objeto de configuração (conjunto de regras) através de um arquivo XML. Optou-se por este *framework*, pois além de oferecer diversos métodos para manipulação de arquivos XML, possibilitando a persistência e o carregamento de objetos através de poucas linhas de código, ele facilita o trabalho de salvar um objeto de configuração em um meio de armazenamento, visto que esta classe que representa as configurações possui diversas classes aninhadas, dificultando a elaboração de um padrão próprio para a persistência.

Porém, para que esta ferramenta possa ser utilizada, fez-se novamente necessário o uso da API *GWT-Servlet*, onde criou-se um serviço responsável por atender atividades referentes às configurações (*ConfiguracaoService*). Além de responder a requisições de cadastro, manutenção e carregamento de configurações, este serviço também é responsável por salvar, editar, excluir e carregar variáveis globais do sistema, que, assim como as configurações, são persistidas utilizando-se a ferramenta *XStream*.

No desenvolvimento do componente gráfico responsável pela exclusão e edição das variáveis do sistema, como pode ser visualizado na figura 4.6, é possível escolher a var-

iável que se deseja editar ou excluir, além de tornar possível a inserção de novas variáveis. Ao escolher pela opção de exclusão de uma variável, a fim de manter a consistência do sistema, o serviço de configuração procura pela variável em todas as regras cadastradas, caso seja localizada alguma ocorrência, a aplicação impede a remoção desta. As classes e interface envolvidas na implementação do serviço descrito podem ser observadas na figura 4.7.

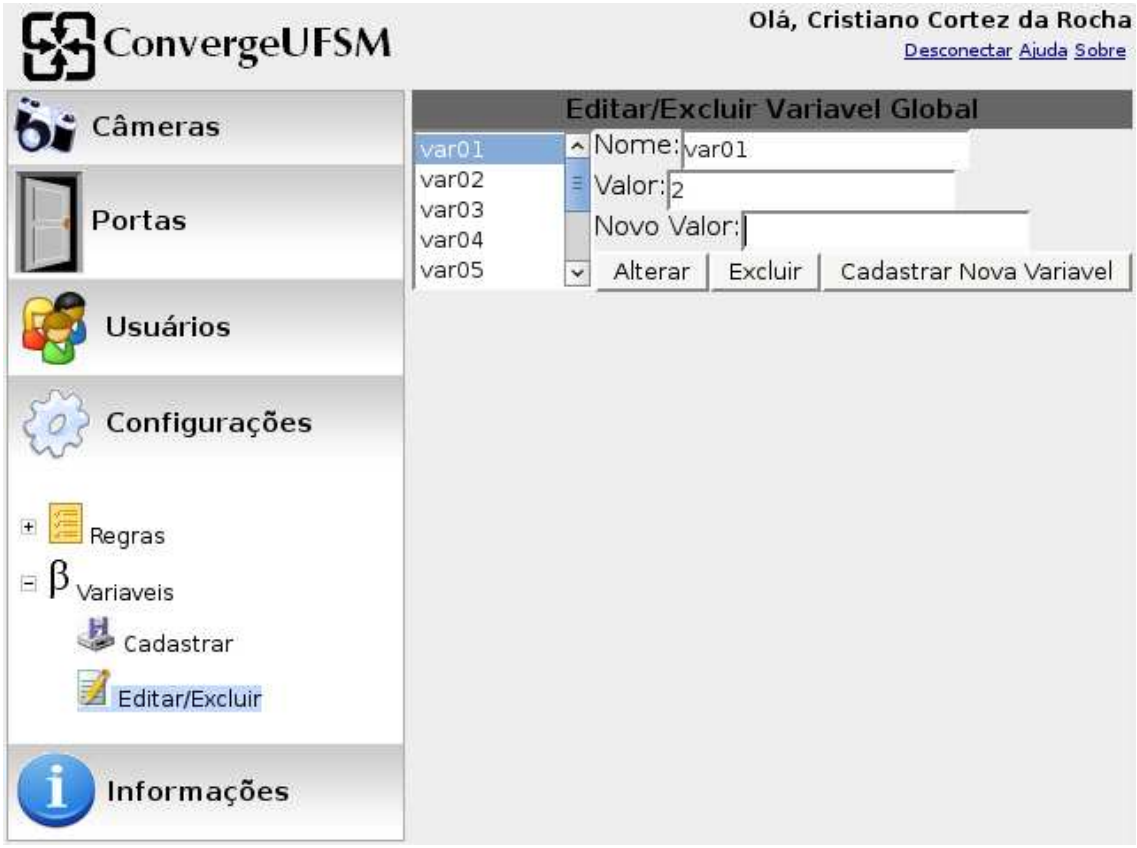

Figura 4.6: Interface *web* - Edição/exclusão de variável global

4.2.4 Módulo de consulta

A última etapa de desenvolvimento consistiu no desenvolvimento do módulo responsável por apresentar informações sobre o sistema, como relatórios de registros de ponto e *logs*.

Inicialmente, projetou-se um mecanismo genérico para pesquisar tanto registros de ponto, quanto *logs* de eventos ocorridos no sistema, visto que na modelagem definiu-se que um registro de entrada e saída de usuários também é considerado um evento do sistema. O componente visual para a busca descrita é apresentado na figura 4.8.

Como pode ser visualizado na figura 4.8, o sistema de busca permite restringir diversas

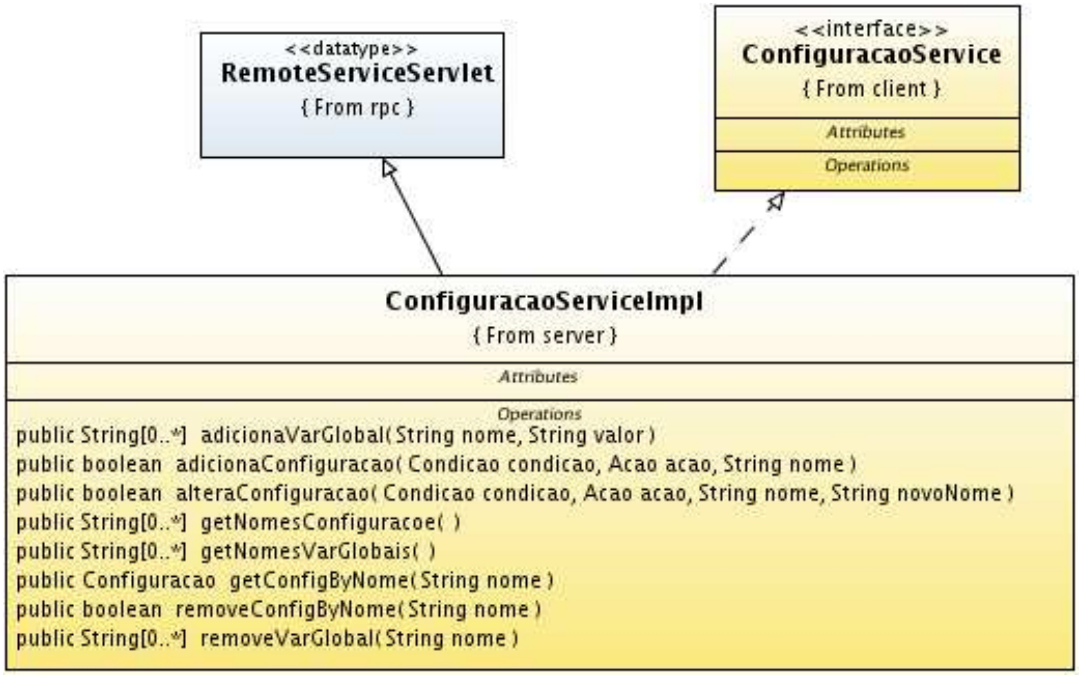

Figura 4.7: Diagrama de dependência - Serviço de configuração

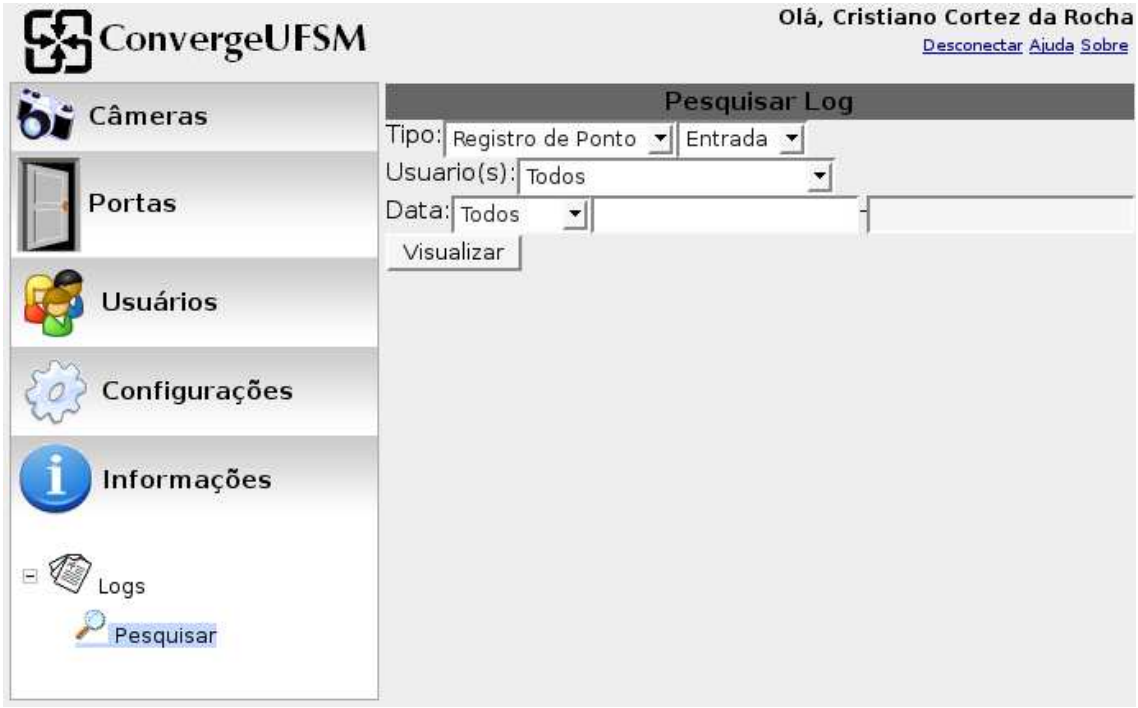

Figura 4.8: Interface *web* - Pesquisa de *logs*

opções, refinando a pesquisa de acordo com a necessidade do administrador. Entre as opções permitidas para a filtragem da busca estão:

- tipo de evento: registro de ponto (entrada e/ou saída) ou *logs* gerados pelo sistema;
- usuários: todos os usuários, administradores, usuários comuns ou um usuário em

particular;

- data: todas as ocorrências, intervalo inferior/superior a uma determinada data ou intervalo entre duas datas.

Após a implementação do mecanismo de busca descrito, partiu-se para o desenvolvimento do serviço de consultas, responsável por realizar a busca de registros a partir das opções citadas. Visto que os eventos do sistema são armazenadas em uma base de dados, fez-se necessário o uso da API *GWT-Servlet* para que as consultas possam ser realizadas com o auxílio do *framework Hibernate*.

Posteriormente, implementou-se as classes responsáveis pela apresentação do resultado da pesquisa realizada (*VisualizadorRegistroPonto* e *VisualizadorLogSistema*). A fim de facilitar a navegação entre os registros localizados, estes são apresentados ordenados por data de ocorrência (exibindo dos registros mais recentes aos mais antigos), além de possibilitar ir diretamente aos eventos mais recentes ou aos eventos mais antigos registrados através de botões de navegação, como pode ser visualizado na figura 4.9.

Figura 4.9: Interface *web* - Visualização de registros de ponto

Além disso, pelo dinamismo oferecido pela técnica de desenvolvimento AJAX, quando algum usuário administrador incluir um novo tipo de evento, este é automaticamente incluído na lista de opções de eventos do sistema a serem consultados. Sendo assim, o sistema permanece consistente sem a necessidade de um recarregamento de toda a aplicação.

As classes envolvidas na implementação do serviço descrito podem ser observadas na figura 4.10.

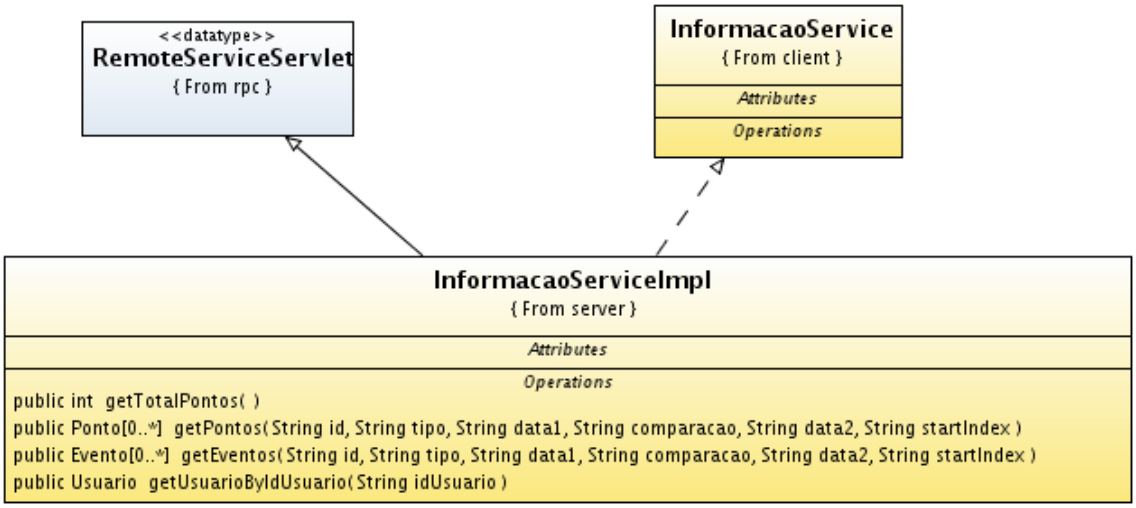

Figura 4.10: Diagrama de dependência - Serviço de informação

5 AVALIAÇÃO DO MÓDULO DE INTERFACE WEB

Este capítulo apresenta uma avaliação das funcionalidades oferecidas pela interface *web* desenvolvida. São realizadas comparações entre operações básicas e avançadas com e sem a utilização da interface, destacando a forma de realização destas operações, e como estas atendem às necessidades do projeto CONVERGE UFSM.

A aplicação desenvolvida foi testada e está em uso pelos membros dos grupos de pesquisa GMob e LSC desde maio/2007. Sendo assim, esta seção apresenta as principais vantagens adquiridas com a utilização desta interface *web*, tanto por parte dos usuários comuns, quanto pelos administradores do sistema.

5.1 Operações Básicas

Inicialmente, a comparação entre a realização de operações básicas com e sem a utilização da aplicação desenvolvida, permite observar as facilidades obtidas na possibilidade de visualização de vídeo on-line da porta de acesso ao laboratório, possibilitando a abertura da mesma. Sendo assim, o usuário não precisa se deslocar até a porta para abri-la, além deste evento poder ser registrado, oferecendo um maior controle de acesso ao laboratório para os administradores, um dos principais objetivos buscados pelo projeto CONVERGE UFSM.

Outra facilidade oferecida pela utilização da interface é o fato do usuário poder trocar sua senha de acesso ao laboratório, e por sua vez, à aplicação, sem a necessidade de contatar algum dos administradores. Assim, o usuário pode trocar sua senha sempre que achar necessário, mantendo sua privacidade sem a intervenção de outra pessoa.

5.2 Operações Avançadas

Este ponto da avaliação descreve a realização de operações avançadas, disponibilizadas apenas para os usuários administradores. As operações avançadas permitem a percepção da maior parte das vantagens oferecidas pela aplicação. Utilizando a interface *web* desenvolvida, o administrador possui uma visão geral e centralizada do sistema, abstraindo a manipulação das particularidades de cada tecnologia utilizada.

Uma das atividades que integra as operações avançadas é a visualização de fotos e vídeos capturados quando as câmeras instaladas detectam a ocorrência de movimento através do *software Motion*. Sem a utilização da interface, o administrador teria que obter acesso ao computador onde as imagens são armazenadas. Além disso, a todo momento que desejasse ver as fotos, teria que percorrer toda a estrutura de diretórios para poder visualizar as imagens. Com a utilização da aplicação desenvolvida neste trabalho, o usuário com permissão de administrador pode navegar rapidamente entre os eventos capturados que são identificados com a data e horário de ocorrência, além de poder visualizar uma imagem pequena que auxilia na identificação da seqüência desejada. Ao selecionar o evento, a imagem é ampliada e a seqüência de fotos pode simplesmente ser visualizada através de botões de navegação, ou, então, ser visualizada na forma de vídeo, como é apresentado na figura 5.1.

Figura 5.1: Interface *web* - Visualização de fotos (à esquerda) e vídeos (à direita)

Outra operação que é permitida para os administradores é a visualização de registros de ponto de entrada e/ou saída do laboratório ou de *logs* dos demais eventos ocorridos no sistema. Sem a utilização da interface, este usuário, toda vez que desejasse visualizar

estes registros teria que abrir uma conexão com o banco de dados através de um *software* cliente, e, então, com os conhecimentos necessário de SQL (*Structured Query Language* - Linguagem de Consulta Estruturada) realizar a consulta. Visto que inúmeros eventos ocorrem no sistema, mesmo com as diversas opções para refinamento da pesquisa, o administrador, dependendo da consulta, levaria mais tempo para encontrar os registros desejados. Já com a utilização da interface, este administrador encontra um mecanismo de pesquisa já estruturado, contendo a lista de todos os usuários cadastrados no sistema, além da possibilidade de consultar pelo grupo de usuários comuns ou administradores, além de possuir a lista com todos os tipos de eventos possíveis no sistema. Sendo assim, o processo de consulta não se torna desgastante e demorado para este usuário.

Uma das atividades em que os benefícios oferecidos pela interface ficam mais evidentes é o processo de configuração do sistema. Sem a utilização da aplicação, o administrador teria que entender profundamente os detalhes da modelagem da máquina interpretadora de regras, além disso, este usuário teria que manualmente configurar o sistema, obrigando-se a conhecer a linguagem de programação em que a máquina de regras foi implementada, neste caso, a linguagem Java, para que seja possível a inserção, remoção ou edição de regras. Além disso, o administrador teria que criar um padrão para persistir a máquina de regras em algum meio de armazenamento. Ao utilizar a interface *web*, esta abstrai todo o funcionamento e implementação da máquina interpretadora de regras, compreendendo ao usuário administrador apenas a manipulação das regras e das variáveis do sistema de forma forma simples e eficaz.

6 CONCLUSÃO E TRABALHOS FUTUROS

É perceptível a crescente necessidade por segurança que se tem na sociedade atual (SCHWARTZ, 2004). Cada vez mais se fazem necessários sistemas computacionais que ofereçam segurança, pois os custos de aquisição, instalação e manutenção destes muitas vezes são elevados. Além disso, sistemas convencionais de segurança frequentemente necessitam de interação humana, o que prejudica sua automação e adiciona à eles possíveis pontos de falha.

Este trabalho descreveu o processo de desenvolvimento de uma interface *web* para tornar possível a utilização e administração das diversas tecnologias usadas na construção de um sistema de telefonia digital, com funções de controle de acesso e alarme para os laboratórios nas instituições, descrevendo suas funcionalidades e apresentando uma avaliação de sua utilização. Para o processo de implementação desta interface utilizou-se o *framework* GWT, que possibilita o desenvolvimento de aplicações AJAX utilizando a linguagem de programação Java. Desta forma, a interface oferece para o usuário devidamente autenticado um ambiente dinâmico e interativo para visualização dos diversos eventos ocorridos no sistema e possibilitando a configuração da máquina interpretadora de regras, que contém a inteligência do sistema.

Durante o processo de desenvolvimento foram descritas as operações básicas e avançadas possíveis na aplicação. O processo de avaliação comparou a utilização com e sem a interface *web* descrita, apontando as facilidades obtidas pelos usuários na execução de operações básicas.

Quanto as operações avançadas, apresentou-se uma descrição de suas funcionalidades, apresentando através de um estudo de caso os diversos pontos onde a interface auxilia o administrador do sistema, identificando as diversas facilidades fornecidas pela utilização da aplicação. Assim, pôde-se perceber que em muitos pontos a interface *web* alcançou

os objetivos do projeto CONVERGE UFSM, através da abstração de gerenciamento e manipulação das diversas tecnologias que integram o sistema.

O processo de desenvolvimento da interface proposta permitiu a percepção de outros pontos que podem ser abordados para aumentar sua funcionalidade e usabilidade. Dentre eles a possibilidade de ampliar o escopo de configuração do sistema por parte da interface, facilitando o processo de manutenção do sistema inserindo a possibilidade de exclusão de registros antigos de eventos, como fotos, vídeos e *logs*. Este processo de manutenção de registros pode ser estendido a fim de oferecer a possibilidade de gravação destes registros em um outro local de armazenamento.

Outra funcionalidade que poderia ser agregada à interface seria a possibilidade de configuração dos demais arquivos pertencentes ao sistema, como, por exemplo, os arquivos referentes ao tradutor de eventos, que traduz as saídas dos *drivers* dos sensores em eventos, e ao tradutor de ações, que traduz as ações provenientes da máquina de regras para os atuadores. Sendo assim, todo o processo de configuração do sistema seria centralizado e realizado através da interface.

REFERÊNCIAS

APACHE. **Apache log4j**. Disponível em: <http://logging.apache.org/log4j>. Acesso em: janeiro de 2008.

AXMARK, D.; LARSSON, A. **MySQL**. Disponível em: <http://www.mysql.com>. Acesso em: novembro de 2007.

BAUER, C.; KING, G. **Hibernate in action**. [S.l.]: Manning, 2005.

BERHORN, A. **An Introduction to the Development of Web Applications using Ruby o Rails with Ajax**. 2007.

COWARD, D. Java Servlet Specification Version 2.3. **Sun Microsystems**, [S.l.], v.9, 2001.

DESK, A. N. **Google's Innovative Yet Limited AJAX Environment: gwt**. Disponível em: <http://web2.sys-con.com/read/225045.htm>. Acesso em: novembro de 2007.

ECKSTEIN, R. **Java SE Application Design With MVC**. Disponível em: <http://java.sun.com/developer/technicalArticles/javase/mvc>. Acesso em: novembro de 2007.

FRAGMENTAL, T. **MVC e Camadas**. Disponível em: <http://fragmental.com.br/wiki/index.php/MVC_e_Camadas>. Acesso em: novembro de 2007.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design patterns**: elements of reusable object-oriented software. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

GARRETT, J. Ajax: a new approach to web applications. **Adaptive Path**, [S.l.], v.18, 2005.

GOODE, B. Voice over Internet protocol(VoIP). **Proceedings of the IEEE**, [S.l.], v.90, n.9, p.1495–1517, 2002.

GOOGLE. **Google Web Toolkit (GWT)**. Disponível em: <http://code.google.com/webtoolkit>. Acesso em: novembro de 2007.

JOHNSON, B. **Google Web Toolkit: what, why, and how**. Disponível em: <http://code.google.com/webtoolkit/presentations.html>. Acesso em: novembro de 2007.

LEFF, A.; RAYFIELD, J. T. Web-Application Development Using the Model/View/Controller Design Pattern. In: EDOC '01: PROCEEDINGS OF THE 5TH IEEE INTERNATIONAL CONFERENCE ON ENTERPRISE DISTRIBUTED OBJECT COMPUTING, 2001, Washington, DC, USA. **Anais...** IEEE Computer Society, 2001. p.118.

MACORATTI, J. C. **Padrões de Projeto - O modelo MVC - Model View Controller**. Disponível em: <http://www.macoratti.net/vbn_mvc.htm>. Acesso em: novembro de 2007.

METASTAFF, L. **dom4j: the flexible xml framework for java**. 2001.

OLIVEIRA, R. L. d. **Implementação de um núcleo interpretador de regras e eventos para gerenciamento de ações de controle de acesso e alarme**. Trabalho de Graduação - Curso de Ciência da Computação - UFSM.

PRABHAKAR, C. **Google Web Toolkit: gwt java ajax programming**. [S.l.]: Packt Publishing, 2007.

RAPPAPORT, T. **Wireless Communications: principles and practice**. [S.l.]: IEEE Press Piscataway, NJ, USA, 1996.

RYU, K. et al. **Private automatic branch exchange**. US Patent 5,400,397.

SATYANARAYANAN, M. Pervasive computing: vision and challenges. **Personal Communications, IEEE [see also IEEE Wireless Communications]**, [S.l.], v.8, n.4, p.10–17, 2001.

SCHILIT, B.; ADAMS, N.; WANT, R. et al. **Context-aware Computing Applications**. [S.l.]: Xerox Corp., Palo Alto Research Center, 1994.

SCHWARTZ, S. Are there universal aspects in the structure and contents of human values. **Journal of Social Issues**, [S.l.], v.50, n.4, p.19–45, 2004.

SOARES, W. **AJAX - Guia Prático para Windows**. [S.l.]: Editora Érica Ltda., 2006.

SPENCER, M. **Asterisk**. Disponível em: <http://www.asterisk.org>. Acesso em: novembro de 2007.

VEIT, M.; HERRMANN, S. Model-view-controller and object teams: a perfect match of paradigms. In: AOSD '03: PROCEEDINGS OF THE 2ND INTERNATIONAL CONFERENCE ON ASPECT-ORIENTED SOFTWARE DEVELOPMENT, 2003, New York, NY, USA. **Anais...** ACM Press, 2003. p.140–149.

VREEKEN, J. **Motion**. Disponível em: <http://www.lavrsen.dk/twiki/bin/view/Motion/WebHome>. Acesso em: novembro de 2007.

WALNES, J. **XStream**. Disponível em: <http://xstream.codehaus.org>. Acesso em: novembro de 2007.