• PLATAFORMA PARA A INTEGRAÇÃO DE WEBLABS EM ENSINO E PESQUISA COLABORATIVA

Ariadne Arrais Cruz – ariadne.cruz@sj.unisal.br
Centro Universitário Salesiano de São Paulo (UNISAL)
Av. Almeida Garret, 267 – Jd. Ns. Sra. Auxiliadora
13087-209 – Campinas – SP

Fábio A. Lisboa Gomes – adrianol@decom.fee.unicamp.br
Dalton Soares Arantes – dalton@decom.fee.unicamp.br
Departamento de Comunicação – Faculdade de Engenharia Elétrica e Computação
Universidade Estadual de Campina (UNICAMP)
Av. Albert Einsten, 400
13083-852 – Campinas - SP

Resumo: *O objetivo deste trabalho é apresentar uma plataforma colaborativa virtual genérica e extensível para a integração de WebLabs e outros serviços via Web. Esta integração visa promover uma colaboração acadêmica em diferentes áreas da ciência, oferecendo aos desenvolvedores uma infra-estrutura básica de serviços. Tal infra-estrutura viabilizará o compartilhamento de experimentos reais e remotos. A plataforma proposta é baseada em uma arquitetura modular e utiliza o conceito de plugins pré-instalados. Esta abordagem é bastante flexível, permitindo aos administradores da plataforma, adicionar, remover ou atualizar seus serviços WebLabs sem a necessidade de recompilar o núcleo da plataforma.*

Palavras-chave: *WebLab, Plataforma Colaborativa, padrões de projeto.*

1. INTRODUÇÃO

A demanda por aplicações e serviços baseados em *Web* (Arsanjani, 2003)(Sun, 2009) têm se expandido continuamente. Um *WebLab* é um típico exemplo de um serviço para Web, em que o objetivo é prover acesso remoto para experimentos de laboratórios, ou seja, esses serviços oferecem acesso comum a dados geograficamente distribuídos, permitindo que os usuários possam utilizar todos os recursos de um equipamento remoto, sem a necessidade de investir grande quantidade de recursos na aquisição de equipamentos laboratoriais.

O desenvolvimento de uma plataforma para integração de *WebLabs* provê uma implementação rápida e de baixo custo, resultando em um estrutura básica eficiente para integração de equipamentos remotos, sensores e qualquer tipo de dispositivo eletrônico. Atualmente, há um grande número de propostas para *WebLabs* (Le Roux, 2009)(Moreira,

2008) (Szpigel, 2007) na rede Kyatera da FAPESP, e a plataforma discutida neste artigo visa uma integração eficiente desses *WebLabs*. O acesso comum a recursos geograficamente distribuídos é de fundamental importância para contribuir com o crescimento de trabalhos colaborativos em áreas estratégicas da ciência. O objetivo deste artigo é o desenvolvimento de uma plataforma para prover colaboração acadêmica em diferentes disciplinas e oferecer aos desenvolvedores uma infraestrutura básica de serviços, tais como comunicação cliente-servidor e serviços persistentes, além de facilitar o desenvolvimento das interfaces gráficas do usuário.

A plataforma aqui apresentada é baseada em metodologias do estado-da-arte e tecnologias avançadas, tais como Padrões de Projeto e o paradigma Ajax (Gross 2006)(Crane 2005). Os Padrões DIP (*Dependency Inversion Principle*) e EO (*Extension Object*) são usados para a criação da arquitetura baseada em *plug-ins*. Ajax é uma tecnologia de desenvolvimento *Web* para criação de aplicações ricas de Internet, e surgiu para enfatizar a comunicação assíncrona entre o cliente e o servidor. Ela garante que o cliente poderá utilizar a aplicação de forma contínua enquanto os dados são enviados ou recebidos em segundo plano. Somente os dados realmente necessários são baixados para a máquina cliente, o que propicia uma interação melhor e mais rápida.

2. OBJETIVOS E MOTIVAÇÃO

O objetivo deste artigo consiste em propor a implementação de uma plataforma colaborativa virtual, tornando possível utilizar uma única linguagem (Java) para a implementação tanto no lado cliente quanto no lado servidor. Esta característica viabiliza a reutilização de um mesmo código em ambos os lados, cliente e servidor. Isto é uma grande vantagem, quando comparado a técnicas clássicas de desenvolvimento *Web* baseadas em Java, que utilizam JSP (*JavaServer Pages*), Java, *JavaScript*, HTML (*HyperText Markup Language*), onde o reaproveitamento de código é mínimo, diminuindo a complexidade de manutenção.

Além disso, este trabalho também tem como objetivo criar uma plataforma para *Web* totalmente baseada no conceito de *plug-ins*. A ideia é poder acoplar diferentes tipos de *plug-ins* (serviços) à plataforma, sem que para isso haja a necessidade de interromper a aplicação, ou seja, recompilar a plataforma inteira toda vez que um *plug-in* for adicionado. Dessa forma, nem a plataforma e nem os serviços precisam ser interrompidos quando ocorrer uma atualização.

A plataforma também oferece ganhos significativos em relação a sua interface, pela utilização do paradigma Ajax, que utiliza técnicas já conhecidas, como *JavaScript* e XML (*Extensible Markup Language*). O Ajax utiliza essas tecnologias de maneira eficiente, provendo interfaces ricas e amigáveis, que se parece com aplicações *desktop*. Este trabalho foi desenvolvido de acordo com os princípios de extensibilidade, robustez e usabilidade (Cruz, 2007).

3. AJAX

Devido à popularização da tecnologia Ajax, há uma grande diversidade de plataformas sendo desenvolvidas. A cada dia uma nova plataforma é lançada, e a escolha de qual plataforma usar depende de muitos fatores, como o tipo de linguagem e se o desenvolvedor da

aplicação é um programador ou um *Web Designer*. Para o desenvolvimento da plataforma aqui proposta, foi utilizada a plataforma fornecida pelo *Google*, chamada *Google Web Toolkit* (GWT) (Hanson, 2007), descrita a seguir.

3.1. *Google Web Toolkit* (GWT)

É uma plataforma Java para o desenvolvimento de aplicações *Web* dinâmicas e está na sua versão 2.4. O GWT permite desenvolver *sites* como o *Gmail* e o *Google Maps*. É bastante promissor, e tem vantagens significativas para a construção de aplicações Ajax. Talvez, uma das grandes vantagens do GWT seja a possibilidade de implementar toda a aplicação em Java, tanto no lado cliente, quanto no lado servidor.

Diferente dos demais frameworks Ajax existentes que constroem uma camada de abstração, o GWT provê um código otimizado para os diferentes tipos de navegadores existentes, o que resulta em códigos menores e mais rápidos que a maioria das outras plataformas. Esta característica, que produz um código otimizado para cada navegador, é obtida gerando-se tantos *scripts* quanto forem o número de navegadores suportados pelo GWT. Além disso, o GWT provê suporte à programação *multi-touch*, o que torna possível o seu uso em *tablets* e *smartphones*.

Cada novo *plug-in* adicionado à plataforma necessita de uma interface por meio da qual os usuários possam acessá-lo. Isto requer que essas interfaces sejam construídas em tempo de execução, ou seja, em tempo real, à medida que os *plug-ins* forem sendo adicionados à plataforma. Como o GWT não é capaz de gerar código em tempo real, foi necessário uma abordagem alternativa, que consistiu na implementação de uma linguagem intermediária, CIL (ComLab Intermediate Language), e respectivo interpretador, responsáveis então por construir essas interfaces dinamicamente (descrita na Seção 5).

4. PADRÃO DE PROJETO PLUGIN

Foram utilizados diversos Padrões de Projeto durante a implementação da plataforma aqui proposta. Contudo, iremos focar nesta seção os padrões DIP e EO, que são os padrões responsáveis pela criação de uma arquitetura baseada em *plug-ins*.

4.1. Princípio da Inversão de Dependência (DIP)

Também chamado de Inversão de Controle (IoC) (Fowler, 2004), o objetivo desse padrão é remover o controle de dependências de dentro das classes. Ele simplesmente recebe a referência, ao invés de instanciar cada objeto internamente. Os objetos necessários são passados a classe, por exemplo, como parâmetro de construtor.

Neste padrão, o programador não precisa criar seus objetos, mas apenas descrever como eles devem ser criados. Não há necessidade de conectar os componentes e serviços diretamente no código, basta descrever que serviços são requeridos por cada componente. O *container* é responsável por realizar as tarefas restantes.

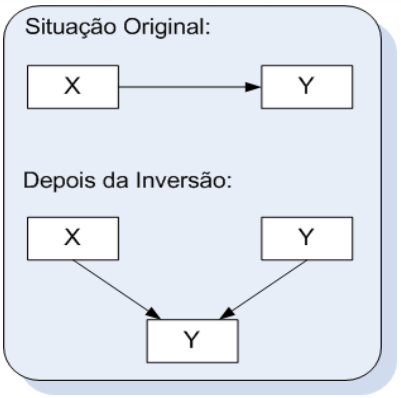

Figura 1 – Quebra de Dependência com a Inversão de Controle

4.2. Objetos de Extensão (OE)

Também conhecido como Interface de Extensão, este padrão permite que um componente possa exportar múltiplas interfaces, prevenindo a quebra do código do cliente e o "inchaço" de interfaces quando desenvolvedores estendem ou modificam a funcionalidade do componente. Múltiplas extensões podem ser anexadas ao mesmo componente (Gamma, 2004). O padrão EO é necessário quando os usuários desejam modificar ou estender as funcionalidades dos clientes.

Em conjunto com o DIP, esses dois padrões formam o núcleo da plataforma proposta, pois são os responsáveis por criar uma arquitetura baseada em *plug-ins*. Essa arquitetura torna possível inserir, remover e atualizar serviços, sem a necessidade de recompilar o núcleo da plataforma.

5. LINGUAGEM INTERMEDIÁRIA DO COMLAB

Para que fosse possível a incluir, alterar ou remover serviços, além da arquitetura baseada em *plugins*, precisaria ser possível incluir códigos para a criação da interface dos serviços do usuário, sem que fosse necessário recompilar a plataforma inteira. O framework utilizado para a criação da *interface* da plataforma, GWT, não permite a criação de códigos em tempo de execução. Devido a isso, se fez necessário a criação de uma linguagem intermediária para a criação desses códigos.

A Linguagem Intermediária do ComLab (CIL) é uma linguagem de *script* que foi concebida para ser compacta e permitir uma rápida e fácil interpretação de seus comandos. Por meio dessa linguagem, códigos de programa podem ser facilmente enviados de um servidor para diversos tipos de clientes, tais como navegadores, *tablets*, *smartphones*, dentre outros. Outra característica da CIL, que a torna diferente de outras linguagens, é que ela foi projetada tendo-se em mente a possibilidade de integrá-la à plataforma proposta neste trabalho.

Na prática, essa forte integração com uma plataforma de serviços permite que operações complexas como envio de *streaming* de vídeos, som ou mesmo de arquivos simples possam ser executadas com uma instrução bem simples. Nenhuma pré-configuração ou preparação é necessária para executar essas operações, pois elas já fazem parte da plataforma.

O objetivo principal da CIL é permitir o envio do código que criará a interface de usuário

nas máquinas clientes. Mais especificamente, ao se adicionar um novo serviço à plataforma (um *WebLab*, por exemplo), é necessário que a interface, por meio da qual os usuários poderão interagir com o novo serviço, seja enviada para a máquina onde o usuário se encontra. É nesse contexto que a CIL está inserida.

6. ARQUITETURA PROPOSTA

O protótipo desenvolvido neste trabalho visa integrar os diversos serviços que estão sendo desenvolvidos na rede Kyatera à plataforma, sem a necessidade de qualquer interrupção nos serviços ou na plataforma. Esta integração permite não apenas a agregação de novas funcionalidades aos serviços já instalados, mas também possibilita que os serviços utilizem recursos uns dos outros.

O núcleo da aplicação é implementado em Java, sendo dividido em sete módulos, como os Serviços de Interface do Usuário (UI – User Interface), o Gerenciador de Usuários, Serviço de Persistência, Modelo de Comunicação, de Monitoramento e a Camada de Segurança (Figura 2). Os serviços que integram a plataforma são desenvolvidos em Java e *Labview*, que é um ambiente de desenvolvimento da *National Instruments*.

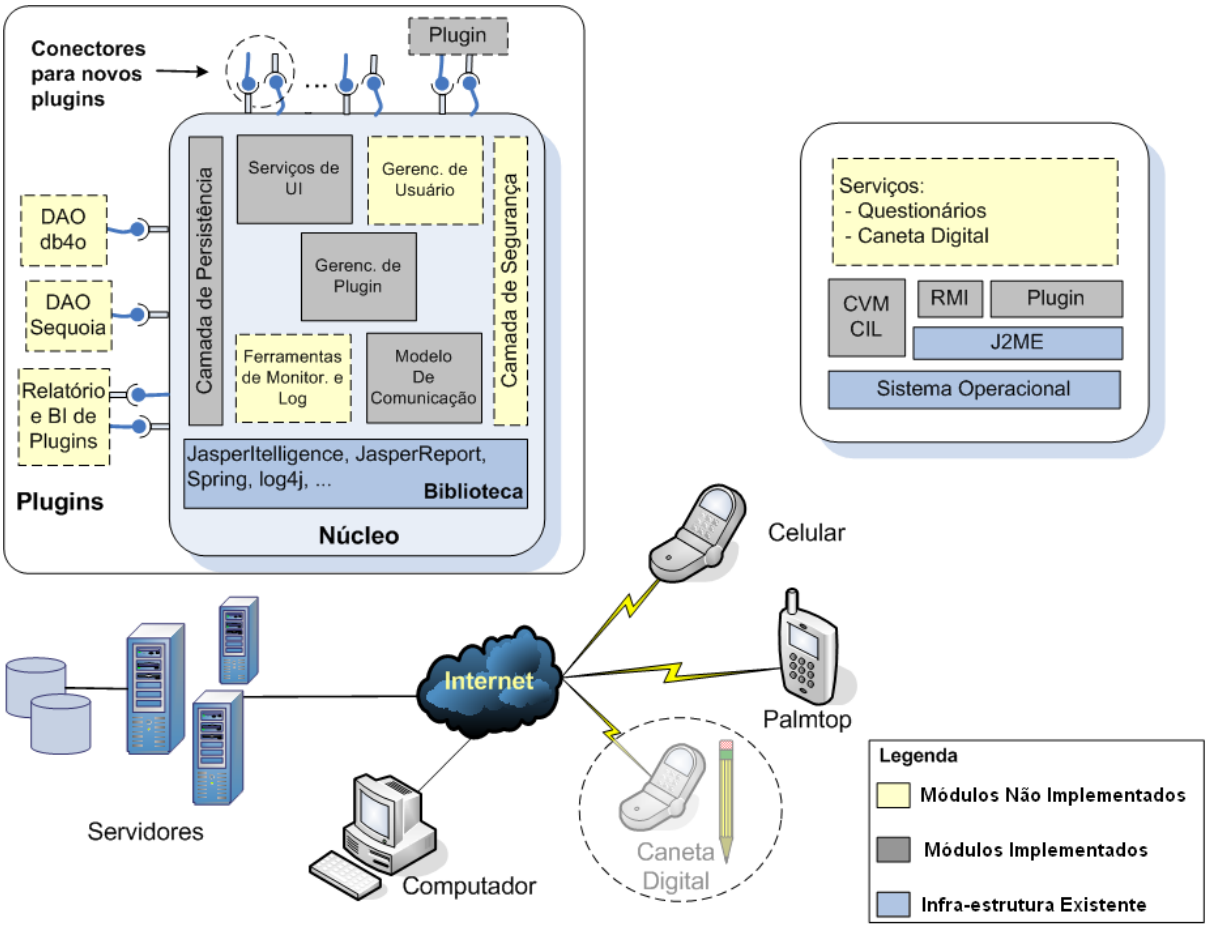

Figura 2 – Arquitetura da Plataforma Proposta

Cada tipo de dispositivo pode instalar uma máquina virtual responsável pela interpretação dos comandos CIL, permitindo que o serviço instalado na plataforma possa, automaticamente, ser executado em vários tipos de dispositivos diferentes. A máquina virtual também poderá se integrar a diferentes tecnologias existentes. Esta facilidade de integração com diferentes tecnologias mostra que a plataforma está preparada para evoluir conforme novas tecnologias vão surgindo. A seguir, será feita a descrição dos módulos atualmente implementados.

6.1. Serviço de Interface do Usuário

Esses serviços são uma aplicação Ajax que provê avançadas funcionalidades para o usuário. É responsável por oferecer serviços aos *plug-ins*. As interfaces dos serviços serão construídas em tempo de execução por meio da CIL. Quando executada em um navegador, a interface gerada pela CIL utilizará os objetos gráficos disponibilizados pelo GWT.

O módulo *Serviço de UI* se comunica com a camada Modelo de Comunicação para fazer as atualizações necessárias a sua interface, que explora toda a interatividade oferecida pelas aplicações Ajax, e foi desenvolvida usando GWT. Todos os serviços integrados à plataforma têm sua interface desenvolvida com o auxílio da CIL, o que possibilita a geração de código em tempo de execução.

6.2. Gerenciamento de *Plugin*

Esta camada é responsável pelo controle de todos os *plugins* instalados. A arquitetura é baseada nos padrões DIP e EO. Os *plugins* podem ser inseridos na plataforma, sem a necessidade de recompilá-la. Os *plugins* e o núcleo da plataforma foram desenvolvidos baseados numa arquitetura em camadas, provendo um baixo acoplamento e uma alta coesão. Os *plug-ins* podem utilizar uma interface já existente ou alguma outra específica para serem integrados à plataforma.

6.3. Gerenciamento de *Plugin*

Os Serviços de Persistência provêem a habilidade de armazenar objetos em um banco de dados relacional. Foi utilizado o Padrão de Projeto DAO (*Data Access Object*) [Crupi 2003], responsável por encapsular a lógica de acesso a dados. Este padrão oferece uma interface comum de acesso a dados e esconde as características de uma implementação específica, permitindo uma maior flexibilidade ao projeto, praticidade de programação e facilidade de manutenção.

No trabalho proposto, os dados são armazenados na forma de arquivos XML. Contudo, caso haja a necessidade de trocar a base de dados, bastará acrescentar a implementação da nova base de dados, sem precisar fazer outras alterações na plataforma, já que o padrão DAO encapsula esses dados. Os dados armazenados são acessados por meio de seus DAOs, que por sua vez são acessadas através da camada Modelo de Comunicação (descrita na seção seguinte).

6.4. Modelo de Comunicação

Esta camada provê tipos para estrutura de dados dos elementos armazenados, sendo responsável pela comunicação entre o cliente *JavaScript* e o servidor, como mostra a Figura

3. A comunicação é feita usando o mecanismo primário de RPC que está embutido no *kit* de ferramentas do GWT (GWT-RPC).

Um conceito importante a ser entendido é a maneira assíncrona da comunicação GWT-RPC. Uma vez que o método do serviço remoto é chamado, o código continuará a executar. A comunicação assíncrona trabalha como um evento *handler*, que fornece uma rotina de *callback*, que é executada quando o evento ocorre. Neste caso, o evento é o retorno da chamada do serviço. A natureza assíncrona da comunicação traz algumas vantagens. A primeira é que a latência da rede juntamente com a execução de serviços complexos pode retardar a comunicação. Ao permitir que a comunicação ocorra de forma assíncrona, a chamada não ficará esperando a execução da aplicação terminar. Outra vantagem é poder emular um "*server-push*", que é um mecanismo onde o servidor envia dados para o cliente, sem que este tenha requisitado.

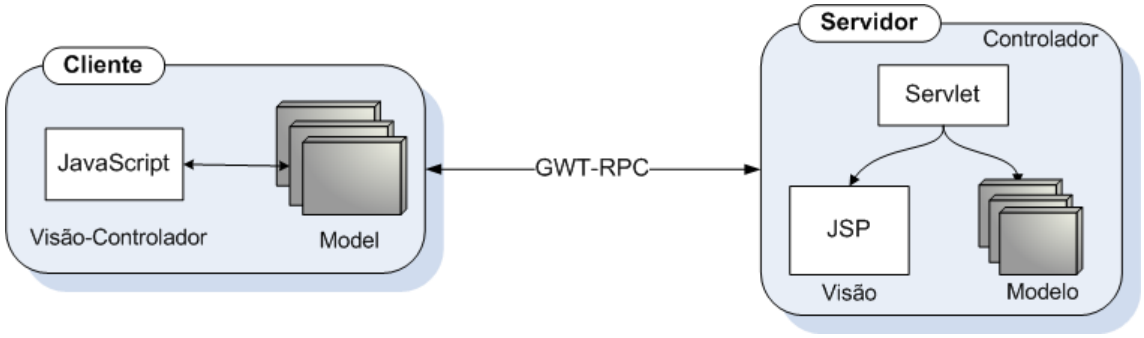

Figura 3 – Comunicação Cliente-Servidor (Hanson, 2007)

7. UTILIZAÇÃO DA PLAFORMA PROPOSTA

Com a grande diversidade de aplicações laboratoriais existentes, há uma necessidade real de uma ferramenta para auxiliar a integração dessas aplicações (*Weblabs*). A plataforma aqui proposta visa auxiliar no processo de ensino de aulas práticas de laboratórios nas mais diferentes áreas da ciência, possibilitando que os alunos possam utilizar os experimentos em suas casas e num horário que lhes seja mais oportuno. Isso possibilita uma maior flexibilidade de horário, quando comparamos as tradicionais aulas ministradas em sala de aula. Além disso, a plataforma propõe uma disseminação do conhecimento, permitindo que instituições que não possuam recursos para adquirir tais dispositivos eletrônicos para a execução das atividades ministradas em aulas práticas, possam fazer uso dos laboratórios remotos.

O acesso aos laboratórios remotos se dará por meio de interfaces de fácil uso e com recursos para a disponibilização dos experimentos e o seu agendamento, já que algumas aplicações precisarão ser pré-agendadas antes da sua execução. As aplicações que deverão ser pré-agendadas são as aplicações que só poderão ser executadas por um único aluno, num determinado momento, enquanto os demais ficarão apenas observando, não terão o controle do equipamento.

A Figura 4 mostra a Interface da Plataforma, onde podemos observar alguns dos serviços que foram utilizados para a prova de conceito. A plataforma pode ser utilizada para aplicar na prática os conceitos aprendidos nas aulas teóricas, deixando o aprendizado mais dinâmico e os alunos mais motivados. A plataforma proposta é bastante flexível e robusta, permitindo que experimentos possam ser incluídos, excluídos ou alterados, sem a necessidade

de que plataforma precise ser recompilada. Após a integração do experimento à plataforma, os alunos poderão ter acesso a dados reais, por exemplo, e com isso poderão aferir medidas, analisar resultados e executar os experimentos sem restrição de tempo ou espaço físico.

A execução de experimentos de laboratórios remotos, muitas vezes necessita de redes de alta velocidade, que possam garantir uma largura de banda satisfatória e uma alta confiabilidade. No Brasil, já estão disponíveis algumas redes de alta velocidade para ensino e pesquisa, como a Rede Kyatera/FAPESP e a Rede GIGA/RNP. A plataforma proposta visa a integração de WebLabs para a rede Kyatera, que é um projeto cooperativo consistindo de uma rede de fibras ópticas projetada para a pesquisa e desenvolvimento de conexões de alta velocidade, integrando laboratórios de pesquisa que focam o estudo, desenvolvimento e demonstração de tecnologia e aplicações na Internet avançada (Projeto Kyatera).

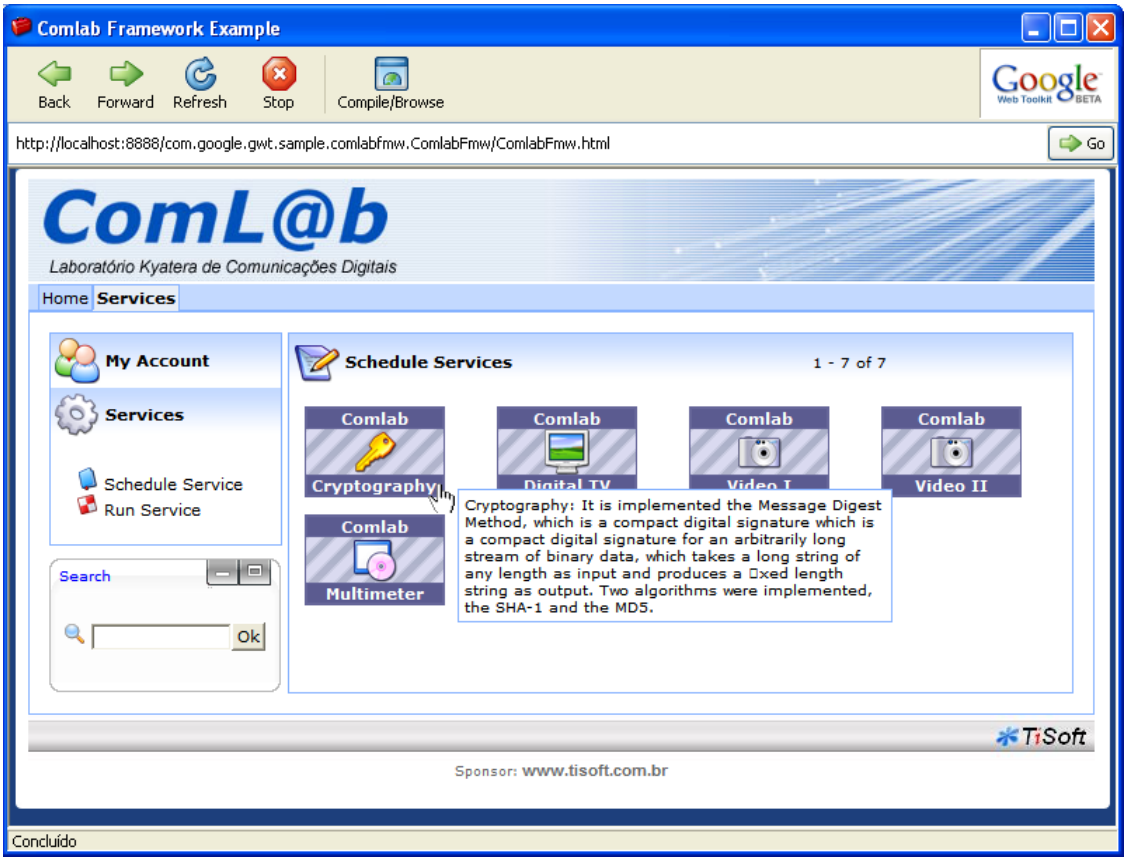

Figura 4 – Interface da Plataforma Proposta

8. CONSIDERAÇÕES FINAIS

Neste trabalho foram descritos os princípios, metodologias e tecnologias usadas no desenvolvimento de uma plataforma de integração de *WebLabs*. Esta plataforma estará disponível para integração de *WebLabs* na rede KyaTera da FAPESP, o que permite acesso remoto a equipamentos laboratoriais. Como não é possível prever todos os serviços, foi adotada uma solução baseada em *plug-ins*, o que torna possível inserir, remover e atualizar os serviços sem recompilar à plataforma. Essas alterações, todavia, só terão efeito após a reinicialização da plataforma.

Outro importante ponto considerado no desenvolvimento da plataforma foi à questão da usabilidade das aplicações *Web* tradicionais. Por esta razão, o paradigma Ajax foi utilizado, permitindo o desenvolvimento de interfaces ricas. Estas interfaces se aproximam bastante das interfaces de aplicações *desktop*, porém sem a necessidade de qualquer instalação no navegador ou na máquina cliente. A plataforma foi implementada usando a plataforma GWT, que se mostrou a mais adequada entre as plataformas Ajax estudadas. O GWT permite a implementação inteira da aplicação em Java, em ambos os lados cliente e servidor, possibilitando um maior reuso do código.

Como trabalhos futuro propomos a migração da base de dados, a implementação de novas versões da CIL e o desenvolvimento de um tradutor, que converterá o código Java para CIL. Essa abordagem proporcionará uma maior reutilização de código e a simplificação do desenvolvimento e a manutenção do sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

Arsanjani, A. et al. (2003) "Web Services: Promisses and Compromisses. ACM Queue – Vol 1, No. 1.

Crane, D. (2005) "Ajax in Action". Manning Publications, 1st Edition.

Crupi, J. et al. (2003) "Core J2EE Patterns". Pretice Hall PTR.

Cruz, A. A., Gomes, F. A. L., Cardoso, F. A. C. M., Martin, E. B. e Arantes, D. A. (2007) "Development of a Robust and Flexible Weblab Framework based on Ajax and Design Patterns". In Proceedings of Seventh IEEE International Sysposium on Cluster Computing and the Grid (CCGrid 07), pp. 811-816. Brasil.

Fowler, M. (2004) "Inversion of Control Containers and the Dependendy Injection Pattern", http://www.itu.dk/courses/VOP/F2006/9_injection.pdf, acessado em 12/07/2010.

Gamma, E. and Beck, K. (2004) "Contributing to Eclipse Principles, Patterns and Plug-ins". Addison-Wesley Professional, 2004.

Gross, C. (2006) "Ajax Patterns and Best Practices". Apress, 1st Edition.

Hanson, R. And Tacy, A. (2007) "GWT in Action: Easy Ajax with the Google Web Toolkit". Manning Publication, 1st. Edition.

Le Roux, G. et. al. (2009). "Cooperative WebLab: A tool for cooperative Learning in Chemical Engineering in a Global Environment". In 10th International Symposium on Process Systems Engineering: Part A. Brazil, October, 2009, pp. 2139-2144.

Moreira, V. R., Cardoso, F. A. C. M. e Arantes, D. S. (2008) "Plataforma Reconfigurável para Ensino e Pesquisa em Laboratório de Sistemas Digitais a Distância". Simpósio Brasileiro de Informática na Educação, Anais do SBIE. Brasil, 2008.

Projeto Kyatera. http://kyatera.incubadora.fapesp.br/portal, acessado em 08/05/2012.

Sun, Z., Dong, D. And Han, J. (2009) "A demand driven Web Service Lifecycle". In New Trends in Information and Service Science. Beijing, Septemper, 2009, pp. 8-14.

Szpigel, S., Souza, E. A., Paschoal, F., Antonio, E. A. And Filho, J. P. (2007) "WebLab for Measuring the Attenuation Coefficient of an Optical Fiber". In Education and Training in Optics and Photonics, OSA Technical Digest Series (Optical Society of America, 2007),

paper EMB5.

INSTRUCTIONS FOR THE PREPARATION AND SUBMISSION OF PAPERS TO BE PUBLISHED IN THE PROCEEDINGS OF THE XL BRAZILIAN CONGRESS ON ENGINEERING EDUCATION

Abstract: *The objective of this paper is to present a virtual collaborative framework architecture for WebLab integration. In this framework each WebLab becomes accessible by means of a preinstalled plug-in. This modular approach makes it possible to add, remove or update a plug-in, and its corresponding WebLab, without framework recompilation.*

Key-words: *WebLab, collaborative learning, design pattern*