UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

Desafios ao Portar uma Aplicação Web para TV Digital:
um estudo de caso com o ambiente Moodle

Aluna: Melissa Cheau Chen Liang
Supervisor: Prof. Dr. Marco Aurélio Gerosa

Trabalho de Formatura do
Bacharelado em Ciência da Computação do
Instituto de Matemática e Estatística

São Paulo
2009

Índice

Capítulo I – Introdução

Motivação, decisões de abordagem do problema e aplicações do dia-a-dia.

Este trabalho objetiva explorar os desafios envolvidos no desenvolvimento de aplicativos interativos para a TV digital. O estudo envolveu os diferentes padrões mundiais e brasileiro de *middleware* para receptores de TV digital, o uso de APIs JavaTV, o emprego de tecnologias adicionais Web, assim como questões de usabilidade e design. Foi desenvolvido um aplicativo que faz acesso ao fórum de discussões de um sistema de educação a distância [26], bem como o envio de notificações de programação da TV a outros usuários conectados ao sistema.

Ao longo dos últimos cinquenta anos, somaram-se à TV novas formas de uso, como gravação e reprodução do conteúdo transmitido, acréscimo de diferentes formas de entretenimento, como vídeo-games, reprodutores de arquivos de imagem e som, compras por telefone, e mais recentemente, acesso a Internet. A TV também se tornou presente em todas as classes sociais e em vários locais não restritos aos lares, como em dispositivos móveis, elevadores e meios de transporte.

A disseminação da TV na sociedade somada às novas funções que ela comporta tem sido utilizada cada vez mais para trazer informações e dados à população. Tanto serviços governamentais, de propaganda, de entretenimento e de educação são transmitidos por esta mídia. Percebe-se este fenômeno pelo surgimento de diversos canais de TV especializados em cada um desses serviços. No entanto, com a tecnologia analógica, a experiência de assistir TV é ainda bastante passiva, com pouca interatividade, ou seja, pouco envolvimento do telespectador em relação ao conteúdo apresentado em determinado canal e no *feedback* que este possa oferecer às emissoras. Atualmente, a forma mais comum de gerar este *feedback* é pedir ao telespectador acessar mais conteúdo relacionado pela Internet, limitando o número e o perfil de usuários que possam realizar este tipo de atividade.

Com a digitalização do sinal de transmissão, possibilita-se a transmissão de dados e mídias junto ao áudio e vídeo. Como estes dados serão submetidos à descompressão e decodificação por um receptor, pode-se aproveitar o processador e a memória deste para renderizar elementos gráficos e exibir aplicativos similares a programas de computador. Dados podem ser fornecidos a esses programas tanto pelo próprio receptor (informações de canal, dados de região, etc.) quanto pelo usuário por meio do controle remoto, teclados, mouse e outros dispositivos USB. Com estes recursos, possibilita-se a interatividade na TV digital, navegação por páginas com dados adicionais, exibição de outras mídias e jogos.

Este trabalho teve como foco caracterizar as similaridades e as diferenças na interatividade e usabilidade de aplicativos e sistemas para computador, web e TV digital. Como no desenvolvimento da TV digital houve muito reaproveitamento de conceitos e tecnologias oriundas dos computadores desktop, algumas funcionalidades tipicamente de computadores foram portadas para a TV. Especificamente, um sistema web de educação a distância foi escolhida como base do trabalho visto que espera-se que a TV digital

promova a educação e a inclusão digital e os sistemas de ensino-aprendizagem são largamente difundidas no ambiente web.

Para realizar este trabalho, foi necessário uma pesquisa aprofundada por se tratar de uma tecnologia ainda em desenvolvimento no Brasil. Há poucos resultados concretos nesta área, ferramentas prontas para uso e muita indefinição das normas do padrão que o Brasil seguirá para implementar esta digitalização. Como foi realizado um trabalho que mesclou conceitos web com emuladores de TV digital com APIs próprios, muitos testes, erros e soluções alternativas foram buscadas ao longo do desenvolvimento. Um aplicativo interativo foi desenvolvido, e novas linhas de pesquisa foram contempladas nesta área não só em expansão, mas também muito promissora em termos de permeabilidade e difusão na sociedade.

Capítulo II – Conceitos e tecnologias estudadas

Tecnologias envolvidas no desenvolvimento de aplicações para a TV digital, tecnologias web.

2.1 TV Digital e interatividade

A transição entre TV analógica e digital vem a ser, do ponto de vista técnico, a substituição do sinal analógico, sinal com valores contínuos, recebido pela televisão pelo sinal digital, sinal com valores discretos. Esta digitalização do sinal analógico é feita por meio da codificação e compressão dos dados de vídeo e áudio, ainda na emissora. A recepção do sinal é feita por um aparelho específico chamado *set-top box*. O sinal é então decodificado e após descompressão, é exibido como outrora, de forma compatível com a televisão instalada. Algumas observações podem ser feitas desta digitalização do sinal. A compressão permite maior fluxo de dados, melhorando a qualidade de vídeo e áudio. Alem disso, a digitalização por si só garante a integridade dos dados transmitidos. Finalmente, a digitalização permite que diversos outros tipos de dados, não limitados a áudio e vídeo, sejam transmitidos junto ao sinal digital.

Com a digitalização da TV, os dados transmitidos podem conter outros tipos de conteúdo para o telespectador, como textos informativos, imagens, animações, jogos etc. Estes tipos de conteúdo, antes mais comumente encontrados em computadores, possibilita ao telespectador novos tipos de interação e experiência com o meio. O telespectador, outrora passivo em relação ao conteúdo transmitindo, torna-se mais ativo, podendo escolher quais dados e aplicativos utilizar, assim como é o usual nos computadores pessoais. Para reforçar ainda mais esta integração, planeja-se com o desenvolvimento da nova mídia, que o telespectador possa retornar dados às emissoras utilizando um canal de retorno. Este canal, apesar de estar em fase de estudo para melhor adequação à realidade dos usuários, poderá abranger várias tecnologias, desde telefone discado a telefone celular, de Internet banda larga a pulsos na rede elétrica. Espera-se, que com maior integração entre as mídias, e, munido de um canal de retorno de dados às emissoras, o telespectador possa responder questionários, enviar perguntas, participar de programas ao vivo e aos poucos tenha um papel mais decisivo no conteúdo exibido. Especula-se que ele até se torne um produtor de conteúdo, transformando-se em um pequeno emissor, como vem acontecendo na Internet atualmente.

Segundo Becker e Montez [13], “a interatividade não se resume no simples aumento da comodidade das partes envolvidas na transmissão televisiva. Envolve também aspectos financeiros, ao aumentar a quantidade e a qualidade dos serviços, oferecidos.” À primeira vista, esta afirmação parece ter pouca relevância para o cenário da TV digital no Brasil. Mas, ao analisá-la sob a ótica dos programas de inclusão social do governo, percebe-se que a frase engloba uma amplitude de sentidos. A interatividade e as novas possibilidades de mídia podem ser aproveitadas para levar serviços governamentais e informativos para uma parcela maior da população. Segundo dados do IBGE [29], a TV aberta é o principal meio de cultura no Brasil, presente em 95,2% dos municípios brasileiros. Em contrapartida, provedores de acesso à Internet estão presentes em 45,6% dos municípios

enquanto 31,2% dos lares possuem computadores [30]. Com esses números, é possível compreender o interesse do governo brasileiro em desenvolver um sistema nacional de TV digital aberta e acessível. Serviços antes oferecidos como *e-governo* (governo eletrônico), como por exemplo, páginas de cartórios e prefeituras na Internet, poderão ser adaptadas para a forma de *t-governo* (governo pela televisão). Dessa forma, a TV Digital possui o potencial de tornar o conhecimento mais acessível e reduzir burocracias.

No entanto, para que se possa entender como a TV digital e a interatividade evoluem e se fundem, é preciso listar quais são as características da interatividade, e como esta aparece na TV. O conceito da interatividade pode ser classificada em três níveis: reativo, coativo e pró-ativo [13]. No nível reativo, o usuário possui pouco controle sobre a estrutura do conteúdo apresentado pela mídia e todas as opções e entradas de dados são controladas pelo programa. No nível coativo, o usuário tem a possibilidade de controlar a sequência, o ritmo e o estilo do conteúdo apresentado. Finalmente, no nível pró-ativo, o usuário controla não só a estrutura, mas também o conteúdo apresentado.

Para levar o telespectador do nível reativo para o pró-ativo na sua experiência com a televisão, a interação passou e ainda passará por cinco níveis distintos. Quando a TV foi criada, ela exibia imagens preto-e-branco e poucos canais eram usados pelas emissoras. Para o telespectador havia pouco que controlar, ligava ou desligava o aparelho, regulava o volume, brilho e contraste, e trocava de um canal para outro. Ao passar dos anos, a televisão ganhou cores, vários canais e controle remoto. Ao telespectador possibilitou-se certa escolha sobre conteúdo. Logo, acessórios periféricos foram acrescentados à TV, como o videocassete, a câmera portátil e o jogo eletrônico, e o telespectador tornou-se coativo neste nível de desenvolvimento. Passados mais alguns anos, os programas de TV tornam-se verdadeiramente interativos, possibilitando ao espectador interferir em alguns aspectos do conteúdo apresentado, por meio de telefonemas, fax ou correio eletrônico. No entanto, neste caso, a interatividade ainda não é necessariamente exibida em tempo real. No nível mais alto de interatividade, pressupõe-se que o controle pró-ativo do conteúdo será feita de forma mais completa, possibilitando a escolha de ângulos de câmera, encaminhamentos de informações exibidas e outros.

Para realizar a transição da TV analógica à digital, diversos países concentraram esforços em pesquisa de tecnologias de codificação, transmissão e recepção de sinal. Estes estudos criaram padrões e normas específicos para diferentes regiões do planeta.

2.2 Padrão americano, europeu, japonês e brasileiro

Como há diversas tecnologias envolvidas em áudio, vídeo e programação de aplicativos interativos, é necessário adotar certos padrões de hardware e software para criar um sistema de televisão digital [12]. Há basicamente três grandes padrões de soluções interativas no mundo, o americano ATSC (Advanced Television Systems Committee), o europeu DVB (Digital Video Broadcasting) e o japonês ISDB (Integrated Services Digital Broadcasting). Cada padrão determina desde a modulação do sinal de difusão, do transporte e codificação dos dados até o middleware que permite a execução dos programas interativos. Estes padrões surgiram como fruto de pesquisas na área de TV digital realizadas nas décadas de 1980 e 1990, nos respectivos países. O padrão brasileiro SBTVD (Sistema Brasileiro de Televisão Digital) é uma adaptação do padrão japonês pela

indústria nacional. A escolha do padrão japonês pelo fórum (com membros do governo, indústrias e fabricantes) como base de desenvolvimento do SBTVD veio após testes de desempenho para a realidade brasileira (ambientes geralmente fechados com áreas de sombra, pouca transmissão a cabo e uso de receptores móveis) [13].

Cada padrão de transmissão também define o middleware a ser utilizado para executar programas interativos. O ATSC adota o middleware DASE (DTV Application Software Environment) que adota uma máquina virtual Java e permite o uso de linguagens declarativas como HTML e Javascript. O DVB especificou o middleware MHP (Multimedia Home Platform) que utiliza APIs Java, a máquina virtual Java e a uma linguagem similar ao HTML. Programas escritos em Java são denominadas aplicações DVB-J enquanto as escritas em HTML são denominadas DVB-HTML. E o middleware do ISDB é definido pela organização ARIB (Association of Radio Industries and Businesses) e adota a linguagem declarativa BML (Broadcast Markup Language) baseada em XML. Num esforço de estabelecer um núcleo comum entre estes middlewares, outro middleware, GEM (Globally Executable MHP) foi definido [13]. Para o SBTVD, o middleware Ginga foi definido, cujo núcleo Ginga-CC (Common Core) foi estabelecido em Java. Acima desta camada, foram definidas duas especificações, uma em Java e uma em linguagem declarativa NCL (baseada na linguagem de script Lua). Estas especificações são denominadas Ginga-J e Ginga-NCL e são resultado do desenvolvimento das universidades UFPB e PUC-Rio, respectivamente.

O aplicativo que executa sobre o middleware foi batizado de Xlet.

2.3 Xlets

Um Xlet é definido como “um programa Java que executa em *set-top box* em conformidade com a biblioteca Java-TV (uma API Java para prover serviços específicos para a TV digital)” [13]. Para que o programa Java execute da forma definida pelo Java-TV, ele deve obedecer a um padrão de comportamento. O Xlet deve ser controlado por chamadas a métodos para sua inicialização, execução, pausa e destruição, que definem os *estados* ou o *ciclo de vida* do Xlet. Para que este comportamento esteja disponível em todos programas escritas para a TV, o programa Xlet deve implementar os métodos descritos pela classe abstrata *Xlet* definida na API. Estes estados são necessários para que os aplicativos sejam coordenados de forma eficiente no *set-top box*, liberando recursos de memória e processamento quando já não estejam mais em uso, e para que estes aplicativos não atrapalhem a principal experiência do telespectador: assistir o programa da TV.

Porém, até a realização deste projeto, ainda não foi lançado em nosso país, *set-top boxes* funcionais com o middleware em Java para realizar testes. Tendo esta dificuldade à vista, o estudo prosseguiu utilizando emuladores para computador.

2.4 Emuladores de aplicativos: XleTView e LWUIT

O XleTView, cuja tela é exibida na Figura 1, é um software de código aberto desenvolvido para visualizar xlets escritos para o executar no middleware europeu (*MHP*), cujo ambiente de execução é a máquina virtual Java. O software possibilita a inserção de objetos de interface gráfica e recursos multimídia especificadas pelo padrão *HAVI* (organização formada por grandes companhias de produtos eletrônicos). Estes objetos

abrangem ícones, botões, imagens, fontes, animações, textos, sons, vídeos, entre outros, que são utilizados pelo programador de aplicativos interativos. O software emulador possui ainda um controle remoto virtual, similar a de televisão, desenhado sobre a tela que pode ser utilizado para enviar comandos ao aplicativo, simulando a interação com o telespectador.

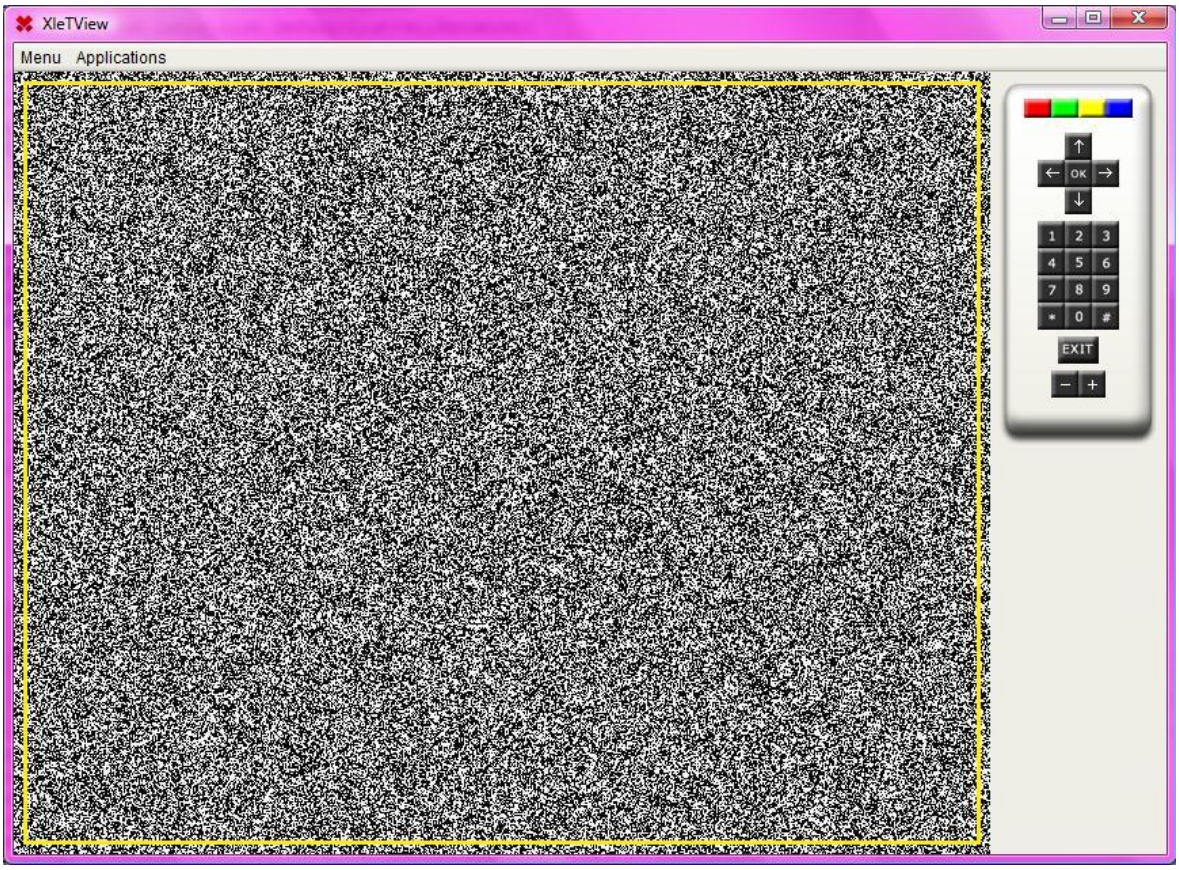

Figura 1. Tela inicial do emulador XleTView com controle remoto virtual

O LWUIT (LightWeight User Interface Kit) é uma API desenvolvida para tratar da interface gráfica do JavaME, inicialmente utilizado em dispositivos móveis (Figura 3). Ele tem sua construção como o Swing, mas como seu nome indica, deve utilizar menos recursos, já que os dispositivos móveis dispõem de memória e processamento menores que os computadores pessoais. Como os receptores de TV digital, os *set-top boxes*, também possuem menos recursos de capacidade de processamento e memória, e muitas vezes nem possuem memória vindo de fábrica. A API foi adaptada para tratar dos elementos gráficos e simular um ambiente de execução de aplicativos próprios para a TV. Este emulador permite a exibição do aplicativo formatado para *SDTV* (*Standard Definition TV*, padrão que prevê 483 linhas ativas por quadro, 30 quadros por segundo e relação de aspecto 16:9) ou *HDTV* (*High Definition TV*, com relação de aspecto 16:9, 1080 linhas ativas por quadro a 30 quadros por segundo, ou 720 linhas ativas por quadro a 60 quadros por segundo). É possível transmitir quatro programas *SDTV* no lugar de um programa *HDTV* utilizando a mesma taxa de transmissão[31].

O LWUIT possibilita a inserção de temas (agrupamento de fontes de texto, cores e tamanhos de itens), imagens de fundo, caixas de texto, ícones, menus de seleção, botões, entre outros, na execução emulada do aplicativo escrito para a TV. Como existe muita demanda para a sua utilização em dispositivos móveis, esta API encontra-se com vários recursos e documentação disponíveis, bem como um editor gráfico de temas (Figura 2).

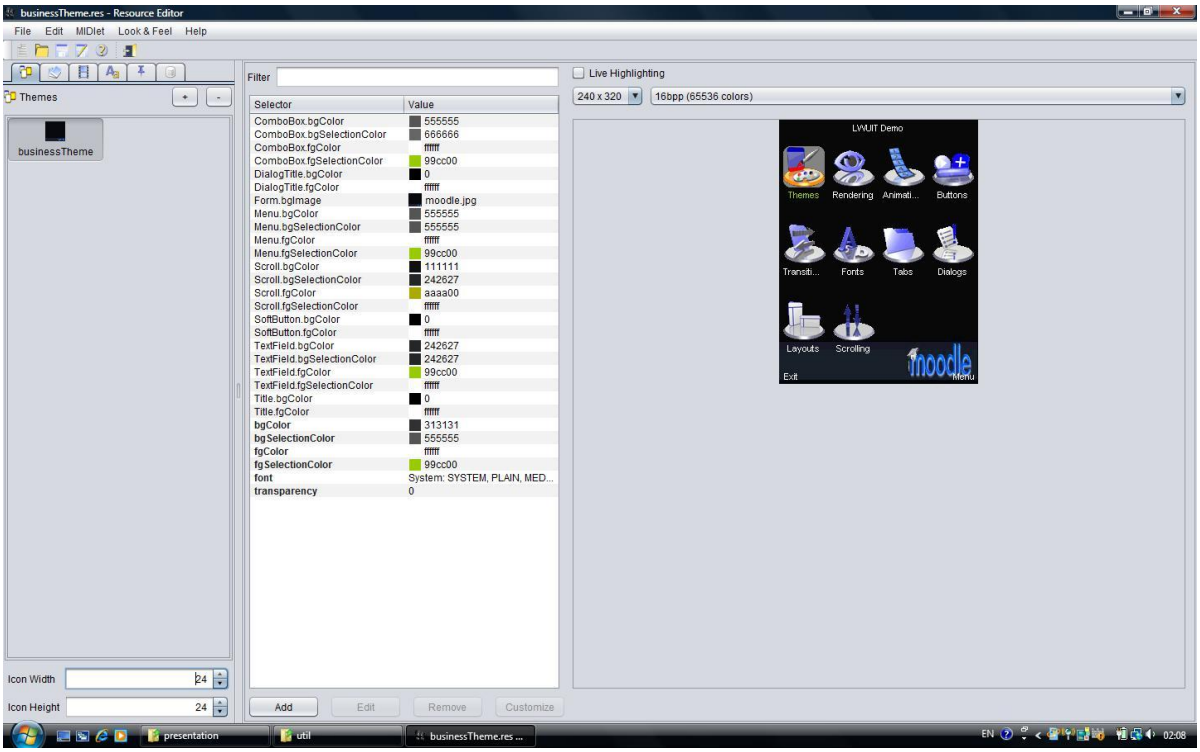

Figura 2. Editor de temas LWUIT

Figura 3. Emulador LWUIT para dispositivos móveis

No entanto, como o LWUIT para TV ainda se encontra em fase de adaptação, a entrada de dados não é feita por um teclado virtual, mas sim pelo teclado e mouse convencionais. O programa para TV não precisa seguir a forma do Xlet (implementar o ciclo de vida do Xlet).

Estes emuladores, por serem baseados em várias APIs Java, possibilitam ao programador utilizar bibliotecas conhecidas de rotinas de entrada e saída, conexões de rede, estruturas de dados, e outros. Como o projeto objetiva estudar os desafios de portar um sistema web para a TV digital, o sistema de apoio à aprendizagem Moodle, largamente utilizado (em mais de quarenta mil sites e por mais de trinta milhões de usuários [40]) no ensino a distância e como apoio ao ensino presencial, foi escolhido para ser portado para TV digital.

2.5 Moodle

O Moodle, gratuito e open-source, é um sistema web de apoio à aprendizagem baseada em PHP e um banco de dados MySQL. Após sua instalação em um servidor, o administrador do sistema cadastra docentes, alunos e cursos, através da própria interface web. Uma vez feito o cadastro inicial, os docentes criam fóruns de discussões, tarefas e outros recursos úteis para seus cursos. Atividades são realizadas, discussões ocorrem nos fóruns e médias de notas são geradas. A Figura 4 exibe uma mensagem em um fórum do Moodle.

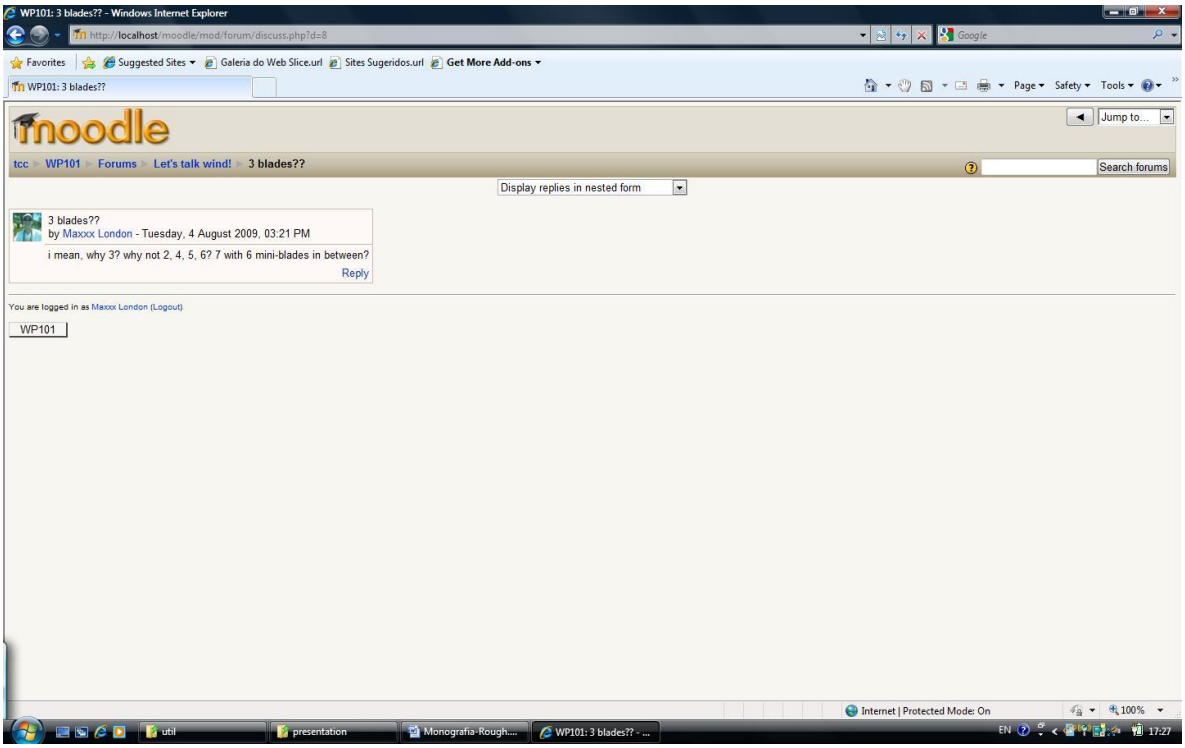

Figura 4. Mensagem em fórum de discussões no Moodle

2.6 RESTful Web Services e Comet

REST (Representational State Transfer) é um estilo arquitetural baseada em servidores e clientes. Uma aplicação que segue os padrões REST é denominada RESTful. RESTful Web Services utiliza métodos HTTP (post, get, delete e put) para fornecer recursos ou coleções de recursos em algum tipo MIME (JSON, XML, etc.). Coleções de recursos ficam disponíveis em um endereço URI (Uniform Resource Identifier) base, e cada recurso pode ser acessado por uma URI própria. Coleções específicas são acessadas por meio de instruções passadas na própria URI. Utilizando essa tecnologia, é possível expor tabelas de bancos de dados em uma web service, onde cada item de tabela corresponde a um recurso com URI própria. Coleções de itens são então acessadas com instruções similares a sintaxe de acesso a banco de dados na própria URI.

Geralmente, requisições são iniciadas por clientes, ao pedirem recursos aos servidores. Comet é uma forma de efetuar trocas de mensagens assíncronas entre cliente e servidor a partir do próprio servidor. Dessa forma, o servidor notifica clientes de atualizações de conteúdo, sem que o cliente tenha que fazer a requisição. Para tanto, sempre que um cliente se conecta ao servidor, este mantém a conexão viva, seja respondendo em pacotes que indicam mais pacotes a receber, seja pedindo para o cliente reiniciar a conexão a cada resposta dada.

Capítulo III – Atividades realizadas

O processo de aprendizado dos conceitos necessários ao trabalho e de implementação do projeto.

3.1 Revisão da literatura

As primeiras pesquisas foram realizadas em artigos, websites e dissertações, em especial, a tese do Juliano Costa [14], que implementou um *Quiz* no MHP. Este trabalho do Juliano foi a implementação de um aplicativo que gera perguntas e respostas em formato de teste a partir de um arquivo XML. Este aplicativo foi escrito para o emulador XleTView e foi bastante interessante para situar o contexto dos estudos realizados.

Logo, foi realizada a leitura do livro *TV Digital Interativa*, de Becker e Montez [13], que faz um panorama da TV digital no mundo e os principais desafios para o Brasil, que faz uma introdução à história da TV e das telecomunicações, um resumo dos principais padrões adotados no mundo, e das tecnologias envolvidas. Também foi estudada e implementada uma aplicação interativa oriunda de um artigo sobre Xlets na revista Java Magazine Edição 66 [10].

Também foi realizada uma busca por trabalhos de formatura e trabalhos de mestrado relacionados à TV digital, tanto da área de engenharia quanto de computação. Alguns trabalhos foram desenvolvidos escrevendo aplicações de voto, gerenciadores de perfis de usuários, leitores de RSS, fórum, quiz, acessibilidade, e jogos. E foram encontrados trabalhos que simulavam a emissora de televisão no computador utilizando dispositivos USB e sistemas de arquivos (*file system*) em *broadcast*.

Direcionando o estudo em termos da aplicação a ser desenvolvida, foram examinadas as especificações do Ginga e do OpenGinga (middleware de código aberto em desenvolvimento).

3.2 Estudo das tecnologias

Foram testados exemplos em NCL no emulador para Linux, numa máquina virtual e no Ginga Live CD. A escrita de aplicações em NCL foi logo iniciada, a partir do próprio código e da ferramenta Composer [11]. Tanto a elaboração manual dos programas quanto pela ferramenta provaram ser bastante inviável, por ter de criar várias referências a objetos de mídias em lugares distintos de código, pela falta de documentação de como escrever os programas e por falhas de execução da ferramenta. Ao final, chegou-se à conclusão de que programar em NCL não seria uma boa opção para o projeto, pois ele apenas oferece a possibilidade de reproduzir mídias acionadas por botões do controle remoto, não possuindo recursos para realizar acesso às conexões de rede, dados, etc. A linguagem de script Lua, que oferece mais recursos de baixo nível não foi testada, e logo optou-se o Java e as APIs JavaTV pela familiaridade e documentação que ele oferece.

Em termos de hardware, buscou se conhecer os *set-top boxes* que possuem a implementação do middleware Ginga. Encontrei alguns: Proview XPS-1000 [33] e [34], o Zinwell ZBT-620 e um da marca Visiontec ainda para lançar [35] e [36]. No entanto, estes receptores rodam ou rodarão NCL. Então, foi comprado e testado o *set-top box* da Proview. Este apresentou diversas falhas ao carregar imagens e aplicativos NCL por USB. Por outro lado, ele foi capaz de acessar programas interativos em fase de teste vindos das emissoras de televisão (pelo carrossel de dados).

Mais adiante, foi realizada uma busca por APIs que tratassem do canal de retorno do padrão DVB (org.dvb.net.rc.RCInterface). Todo este estudo teve como objetivo entender como é feita a transmissão e recepção de dados na TV digital, e como simular estes fenômenos caso não se tenha acesso a equipamento de modulação, codificação e recepção no computador.

Tendo em mente o tipo de aplicação a ser desenvolvida, as tecnologias compatíveis, e definição de arquitetura, iniciaram-se os testes de aplicações para o emulador XleTView (xlets em Java).

3.3 Definição da arquitetura

Aplicativos foram escritos utilizando um parser e browser de HTML para Java chamado Lobo [41] (que utiliza conexões HTTP). Estas APIs foram testadas sozinhas (programa Java) e carregadas dentro de um Xlet executando sobre o emulador XleTView. Diversas páginas da Internet puderam ser acessadas dentro do XleTView, porém, nenhuma do Groupware workbench, o sistema colaborativo que foi testado inicialmente para realizar o porte. No entanto este era acessado pelo aplicativo Java (não o Xlet). Estes testes tinham como objetivo iniciar o acesso de algum sistema web dentro do aplicativo para TV digital. As figuras a seguir demonstram os estes testes realizados com o XleTView e o Lobo.

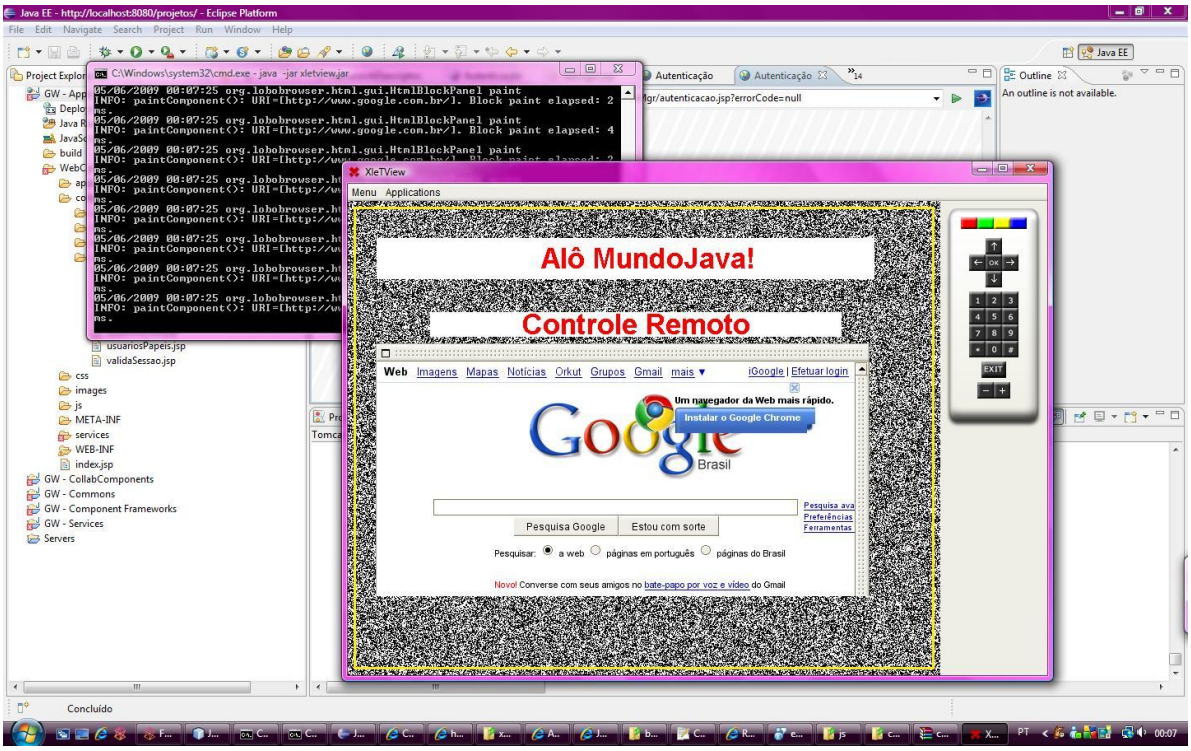

Figura 5. Abrindo páginas da Internet dentro do XleTView

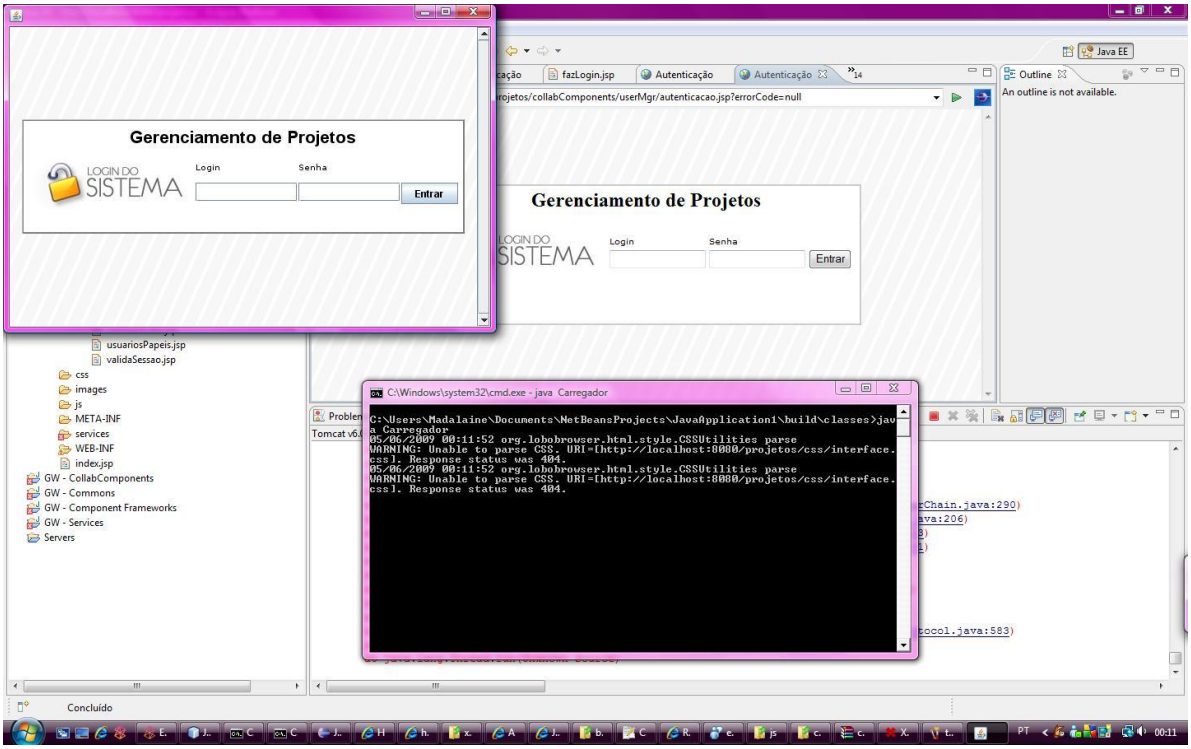

Figura 6. Abrindo página do Groupware workbench em programa Java

Realizando mais testes com o browser Lobo, percebe-se que esta abordagem seria bastante inviável. O browser era bastante simples, sem possuir muitos mecanismos de renderização e acesso a arquivos importantes que nem cookies. O teste foi realizado com páginas do sistema Moodle do servidor Paca do Instituto. Outro problema encontrado foi de limitação na entrada de dados dentro da janela das páginas da Internet (as entradas do controle remoto virtual não eram reconhecidas dentro da página, apenas na área fora da janela do browser). Logo, este caminho foi abandonado e testes com demonstrações do LWUIT foram realizadas.

O XleTView é deixado em segundo plano e logo o emulador do LWUIT apresenta melhores resultados. A API tem documentação disponível, e se torna mais fácil inserir objetos gráficos nos aplicativos. Ao estudar também a implementação do Moodle, percebi que ele tem a funcionalidade de gerar notícias em RSS das mensagens dos fóruns. Com as mensagens em formato XML e com um parser deste no emulador, foi possível escrever um primeiro protótipo do programa: um leitor de mensagens, organizados em um menu, e exibidos em uma caixa de texto. Este protótipo foi baseado nas demonstrações do LWUIT e em aplicativos para dispositivos móveis (Midlets).

Terminada a implementação da leitura de mensagens dos fóruns, começou a ser pensada como enviar mensagens a outros usuários, e que tipos de mensagens poderiam ser criadas com a limitação na entrada de dados. Chegou-se a ideia de enviar recomendações de programas de televisão que estejam passando no momento. Mas, como a mensagem deveria ser instantânea para não perder o programa, não haveria como postar no fórum e buscar a mensagem com atraso. Logo, foi pensado no padrão Comet. Como mencionado em outro capítulo, há várias formas de implementar a funcionalidade, mas neste projeto, junto com APIs Grizzly Comet, foi feita o envio de mensagens assíncronas a diversos usuários. No lado do servidor, implementa-se os métodos da interface CometHandler (explicado em detalhes na próxima seção) com as modificações nos parâmetros de requisição e nas mensagens devolvidas. No lado do cliente, sempre se inicia uma requisição ao perceber uma resposta do servidor.

Terminada a implementação, foram estudados alguns exemplos e arquivos da API JavaTV para descobrir se há como sintonizar canais ou receber informações do canal sintonizado a partir do programa. Apesar da API prover classes que realizam esta tarefa, por ser uma emulação, estas informações aparecem fixas em um arquivo texto estático e armazenado no computador. A idéia por trás deste estudo foi, apesar de termos informações de EPG vindas de um web service, tentar utilizar as próprias funcionalidades que devem fazer parte do middleware para gerar as mensagens de recomendação de programação.

Finalmente, realizou-se a leitura do manual do OpenGinga para tentar entender como realizar a sintonia dos canais e acessar o EPG. Da mesma forma que o JavaTV, ele apenas faz a simulação destas funcionalidades a partir de alguns arquivos de texto. Em conclusão, ainda há grandes diferenças entre a emulação e o teste real que não tem como ser solucionadas até este momento.

3.4 Desenvolvimento do protótipo

Para realizar o protótipo com o emulador LWUIT, uma abordagem mais profunda de programação foi adotada ao se comparar com os testes com o browser Lobo. Ficou a cargo do programa realizar a conexão HTTP, a abertura e leitura do arquivo RSS, e o envio do conteúdo trabalhado para ser renderizado pela interface gráfica. Houve mais trabalho a ser feito, porém, este esforço é compensado em termos de controle do comportamento do sistema. Como explicado na seção anterior, este protótipo foi realizado já com o sistema Moodle. A figura 7 exibe o arquivo RSS gerado pelo fórum de discussões do sistema. Em seguida, a figura 8 apresenta a leitura do conteúdo do arquivo RSS no emulador LWUIT.

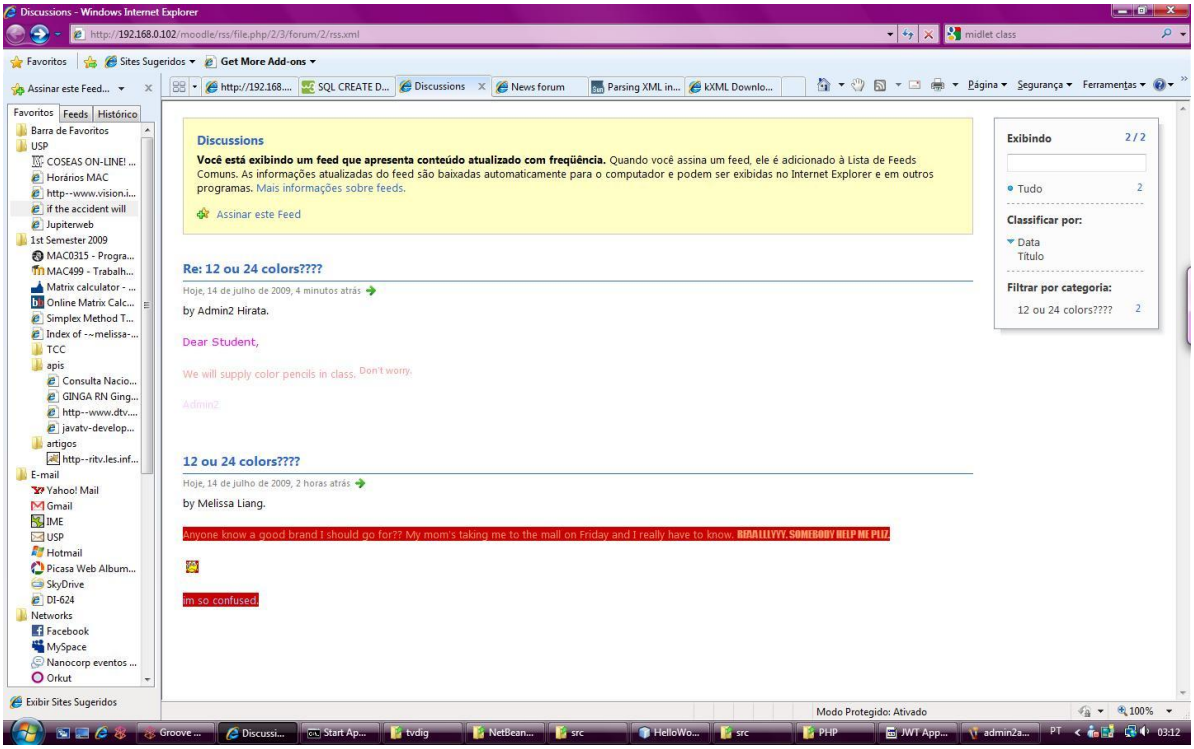

Figura 7. Mensagens dos fóruns do Moodle em formato RSS

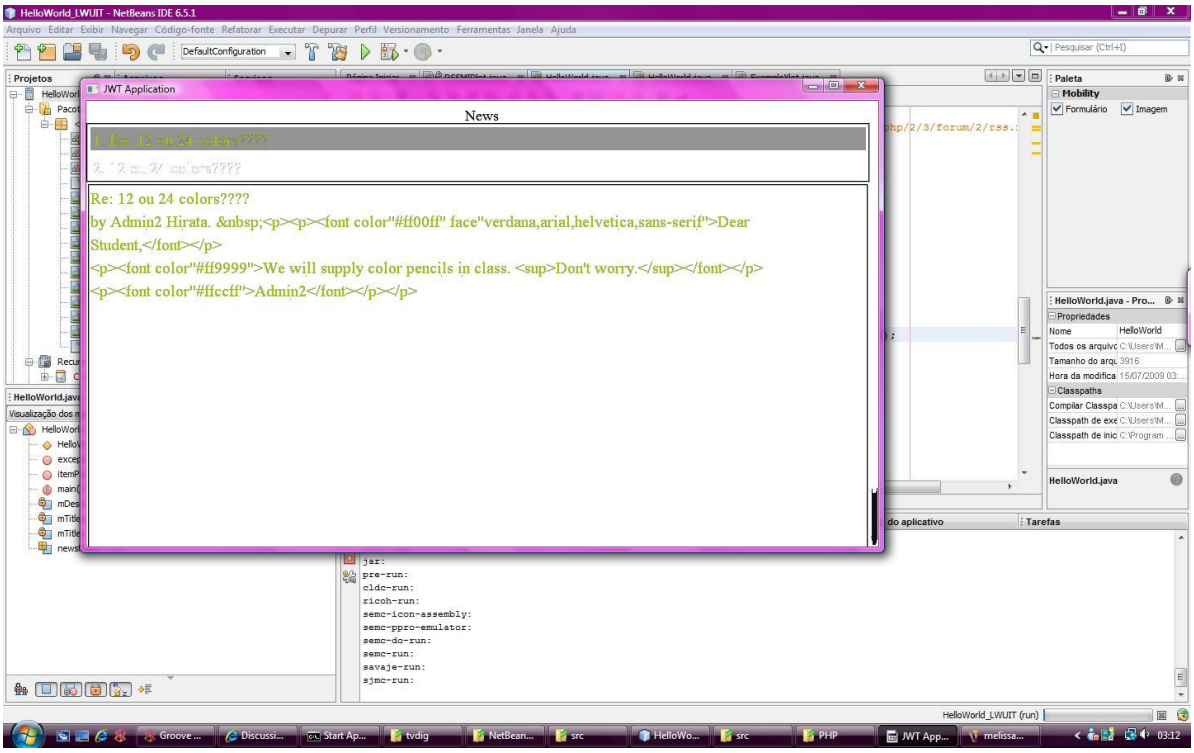

Figura 8. Exibindo mensagens de RSS no LWUIT

A abordagem por RSS ainda não é satisfatória. Neste arquivo não há informações de mensagens lidas e o arquivo é sempre gerado contendo todas as mensagens do fórum. Como ao longo da utilização do sistema durante um semestre de curso muitas mensagens seriam enviadas, o arquivo RSS ficaria muito pesado para um *set-top box* real. Além do mais, a maioria das mensagens seriam desnecessárias uma vez lidas. Pensou-se então em como utilizar um web service para buscar apenas as mensagens relevantes e não lidas.

Felizmente, no NetBeans, há um plugin que cria o web service RESTful automaticamente [2]. Primeiro, configurou-se o JDBC para acessar o banco de dados MySQL do Moodle. Após sincronizar os dados utilizados no banco de dados com o plugin, este gera um web service com os métodos HTTP a serem acessados, configura um servidor e uma porta de acesso, e cria classes que representam uma linha de tabela e uma tabela inteira que são acessadas por classes que as devolvem em formato XML e JSON. Ou seja, o plugin cria um programa que faz a conexão com o banco de dados em questão, cria objetos que representam as tabelas de dados, e os retorna em XML ou JSON (dependendo do cabeçalho da requisição) em uma URI.

Utilizando as ferramentas do LWUIT como caixas de texto, botões, menus, rótulos e ícones a partir de imagens foi possível criar uma interface organizada de modo similar ao Moodle com as informações provenientes dos bancos de dados.

O trecho de código a seguir mostra como é realizada o acesso ao web service pela URI. Em destaque, a sintaxe de busca por recursos ordenados por um campo da tabela em questão.

```
f = mdlCourse.getMdlForums().elements();
while (f.hasMoreElements()) {
      mdlForum = (MdlForum) f.nextElement();

      mdlForumDiscussionsUrl =
      "http://localhost:8081/ServidorMoodle/resources/mdlForumDiscu
      ssionss/?start=0&max=10000&expandLevel=1&query=SELECT%20m%20F
      ROM%20MdlForumDiscussions%20m%20WHERE%2m.forum%20%3D%20" +
      mdlForum.getId().toString() +
      "%20order%20by%20m.timemodified%20desc";

      mdlForumDiscussionsParser.parse(mdlForumDiscussionsUrl);
}
```

As figuras a seguir mostram as mensagens dos fóruns em uma interface similar ao Moodle.

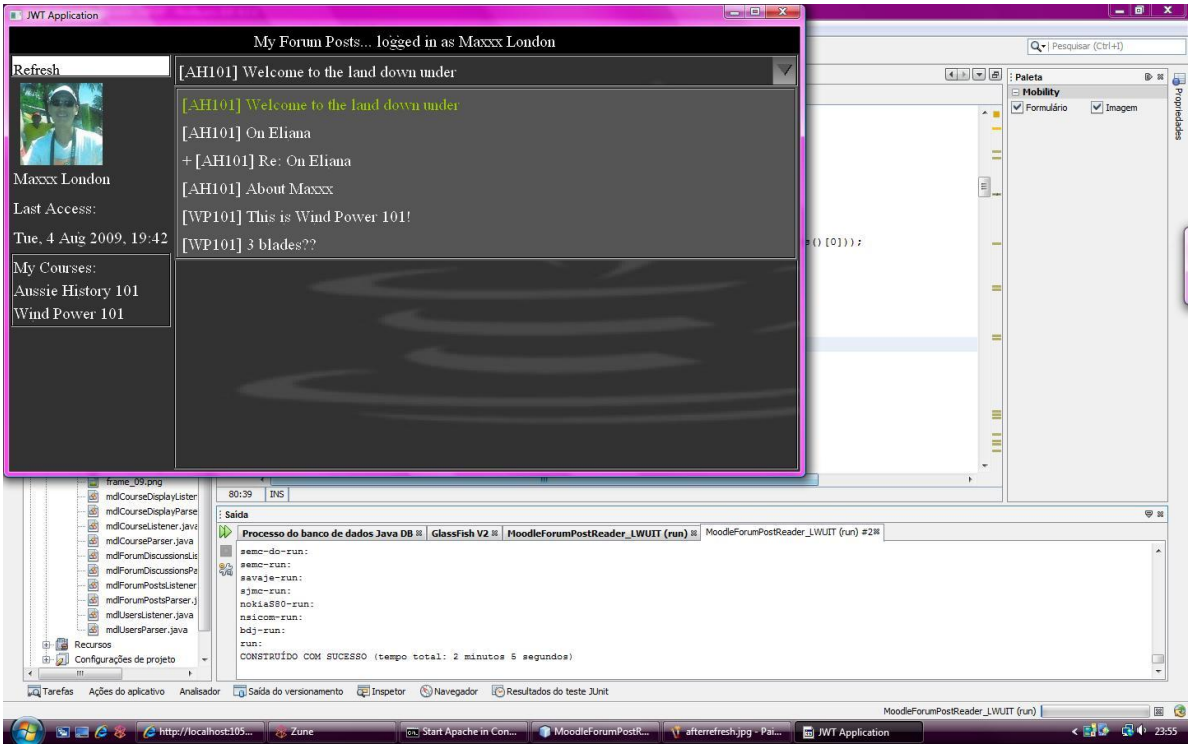

Figura 9. Escolhendo mensagens a partir de um único menu. Obs. O sinal de "+" indica uma resposta a uma mensagem

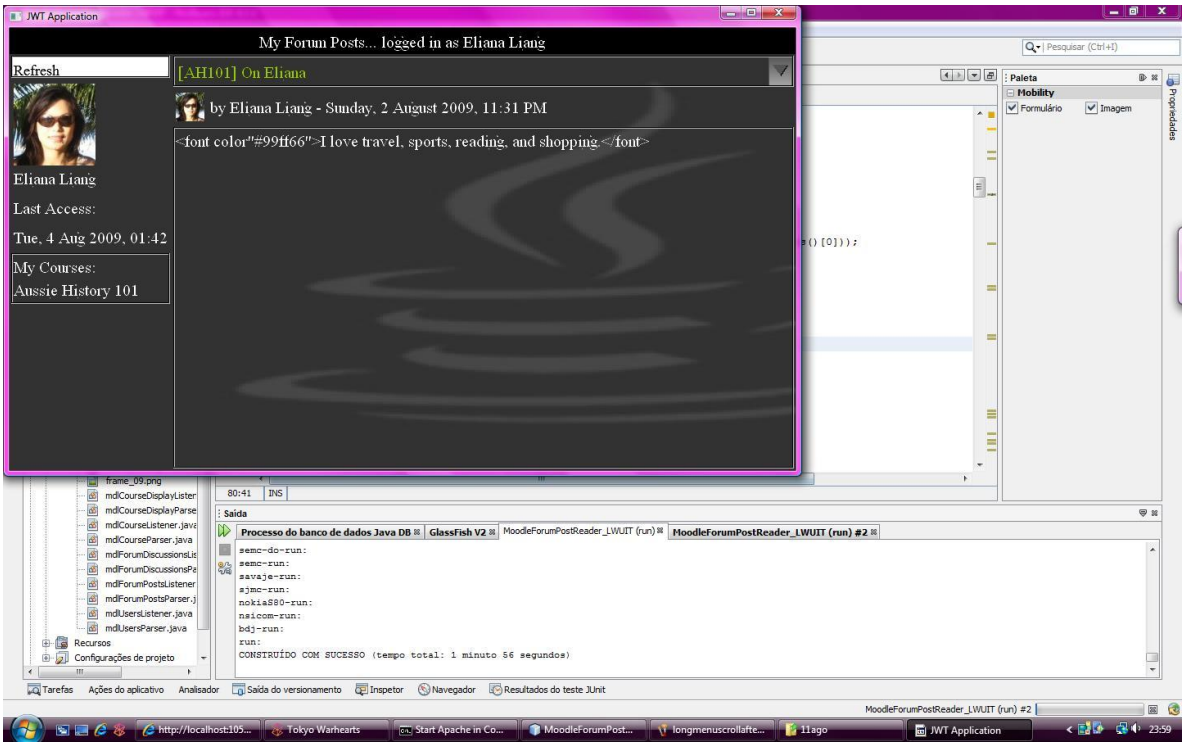

Figura 10. Imagem e nome do autor da mensagem similar ao Moodle

Logo, com mais testes no LWUIT, algumas melhorias foram implementadas em relação às mensagens, como um painel de mensagens não lidas e menus que organizam estas mensagens por curso, fórum, tópico de discussão e mensagem.

Para realizar o acesso e a autenticação dos usuários, foi realizado um estudo na área de criptografia (algoritmo MD5). Com uma caixa de diálogo, foi possível receber dados pelo teclado e enviar junto à requisição do web service para serem conferidos. Haviadois problemas: a senha criptografada hexadecimal no banco de dados, e os dados alfanuméricos que não poderiam ser digitados a partir de um controle remoto de televisão. Ao estudar um pouco sobre criptografia, aprendi que seria impossível realizar a operação inversa para comparar com a senha real do usuário. A comparação teria de ser feita com a senha criptografada. Mas realizar a criptografia de uma senha alfanumérica que não poderia ser digitada no controle remoto também seria impossível. A solução implementada foi digitar a parte numérica da senha criptografada e realizar a comparação. Para tanto, modifiquei no web service as classes que listam os usuários de forma a não exibir senhas nas requisições e devolver requisições vazias caso houvesse falha na autenticação.

O trecho de código a seguir utiliza expressões regulares para formatar identificação digitada e realizar a comparação contra o banco de dados. Se o usuário foi autorizado, o programa gera os dados pro web service, caso contrário, o web service retornará dados vazios.

```
if (!userid.replaceAll("[\\D]", "").equals("")) {
      MdlUserResource m = new MdlUserResource();
      m.setId(Long.parseLong(userid));
      MdlUser user = m.getEntity();
      if (user.getPassword().replaceAll("[\\D]",
          "").equals(password))
            return new MdlUsersConverter(getEntities(start, max,
query), uriInfo.getAbsolutePath(), expandLevel);
}
return new MdlUsersConverter();
```

Alguns detalhes de navegação pelos painéis foram melhoradas como carregar sub-menus automaticamente quando um menu possui apenas um item, e mostrar mensagens a partir do painel de mensagens novas ao passar com as setas do teclado. Mensagens com arquivos anexos passaram a receber um rótulo com o nome do documento. Finalmente, um fundo com o tema do Moodle foi criado no editor de temas e uma fonte (de "Monospace" para "System") mais nítida foi utilizada para exibição do programa.

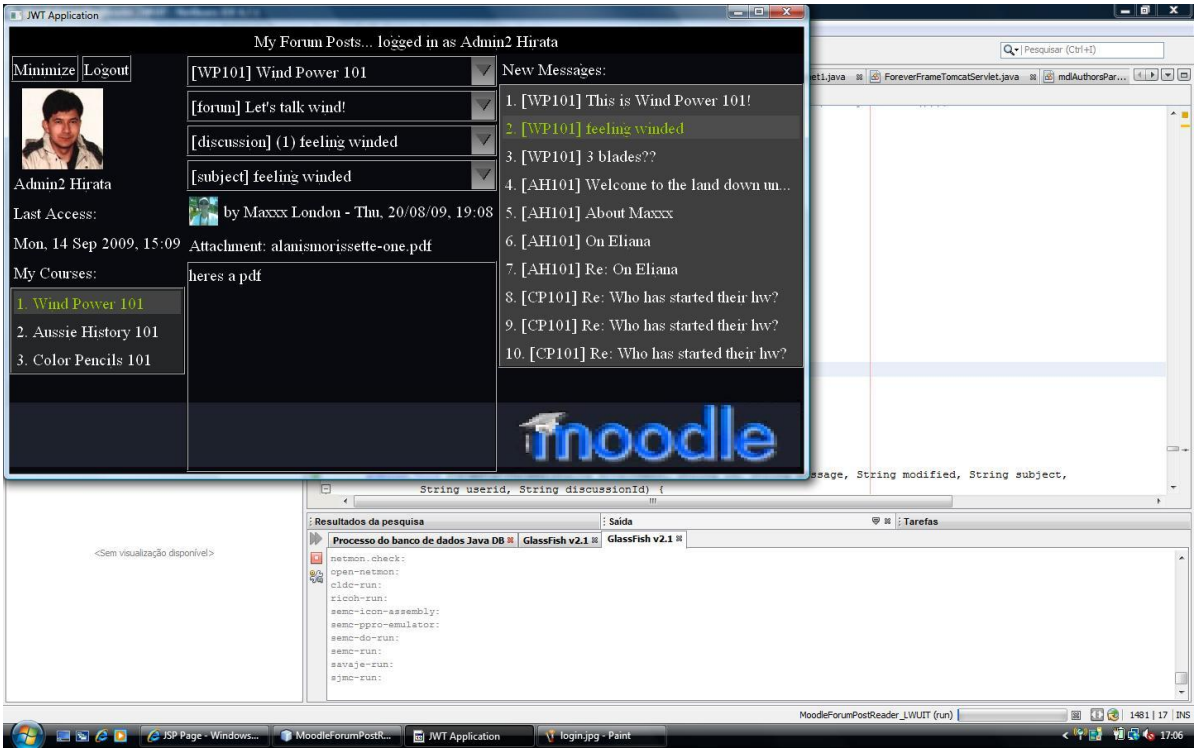

Figura 11. Rótulo indicando anexo de documentos na mensagem

Outra melhoria implementada (mostrada acima) é indicar ao leitor se o autor da mensagem incluiu um arquivo em anexo.

Finalmente, foi implementado a busca de informações da programação de TV a partir de um web service existente na Internet para serem enviadas via Comet para o aplicativo. O usuário escolhe um canal de televisão a partir de um menu, que consulta o web service e devolve o programa que está sendo exibido naquele instante. Os dados são enviados como

parâmetros da requisição HTTP ao servidor Comet, que por sua vez, transmite a mensagem aos usuários conectados ao sistema.

O trecho de código abaixo demonstra o envio (post) e recebimento (resposta) de dados pelo Comet (lado cliente):

```
// Send data
URL url = new URL("http://localhost:21816/ServidorComet/Comet");
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter wr = new
OutputStreamWriter(conn.getOutputStream());
wr.write(data);
wr.flush();

// Get the response
BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
String line;
while ((line = rd.readLine()) != null) {
      // Process line...
      System.out.println(line);
}
wr.close();
rd.close();
```

Para implementar o Comet no lado do cliente, sempre se reinicia a conexão depois que o servidor retorna uma mensagem:

```
while (true) {
      String requestUrl =
          "http://localhost:21816/ServidorComet/Comet";
      try {
            URL url = new URL(requestUrl.toString());
            BufferedReader in = new BufferedReader(new
                InputStreamReader(url.openStream()));
            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                  cometMsg.setText(inputLine);
                  forumPosts.repaint();
            }
            in.close();
      }
```

E pelo lado do servidor, deve-se implementar as seguintes classes com o comportamento desejado (enviar a mensagem, receber os parâmetros enviados pelo cliente):

```
public void onEvent(CometEvent event) throws IOException {
      if (CometEvent.NOTIFY == event.getType()) {
            PrintWriter writer = response.getWriter();
```

```
                writer.write(viewer + " recommends " + title + " on " +
                channelname + " from " + starttime + " to " + endtime +
                ". [Description] " + description + ". Don't miss it!");
                writer.flush();
                event.getCometContext().resumeCometHandler(this);
        }
}

protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException
{
        title = request.getParameter("title");
        description = request.getParameter("description");
        starttime = request.getParameter("starttime");
        endtime = request.getParameter("endtime");
        channelname = request.getParameter("channelname");
        viewer = request.getParameter("viewer");
        PrintWriter writer = response.getWriter();
        writer.println(viewer + title + description + starttime +
            endtime + channelname);


        CometEngine engine = CometEngine.getEngine();
        CometContext<?> context =
            engine.getCometContext(contextPath);
        context.notify(null);
}
```

3.5 Análise dos desafios encontrados

Para realizar a análise dos desafios encontrados, buscou-se os momentos de teste e implementação que tiveram as maiores dificuldades, por conta de falta de documentação, indefinição de normas, diferenças de compatibilidade entre emuladores e APIs, etc. Depois de compilada a listagem desses, buscou-se entender se eram dificuldades específicas ao porte deste projeto ou se eram dificuldades em relação ao desenvolvimento para a TV digital. A análise destes desafios permitiu entender os pontos fortes e fracos dos sistemas utilizados e realizar uma análise crítica da abordagem tomada.

Capítulo IV – Resultados e produtos obtidos

Os resultados finais e o aplicativo.

Como produto deste estudo, foi desenvolvido um aplicativo interativo para TV digital no IDE NetBeans 6.7.1 com o Java SE Development Kit 6u16 e o Java EE SDK 5.07. O aplicativo interativo utiliza os pacotes do LWUIT (emulador Sun CDC Java Media Client Emulator) e de um parser XML (kxml-min). O servidor que trata do web service RESTful que busca os dados do sistema Moodle é o GlassFish v2.1 e o servidor das mensagens Comet fica por conta do GlassFish v3 Prelude. O sistema de educação a distância Moodle (versão semanal 19) utiliza o banco de dados MySQL 5.1.37, o servidor HTTP Apache 2.2.11 (configurado para ser acessado manualmente) e o PHP 5.3.0. O web service externo consultado para obter a programação da TV pertence ao endereço http://sapo.pt

A seguir algumas imagens do aplicativo (organizadas por funcionalidades):

Figura 12. Tela de transição com plano de fundo com logotipo do Moodle

- O LWUIT possui animações de transição entre as telas exibidas (Figura 12). Entre a tela inicial e a tela de autenticação do usuário foi inserida um efeito de esmaecimento das imagens.

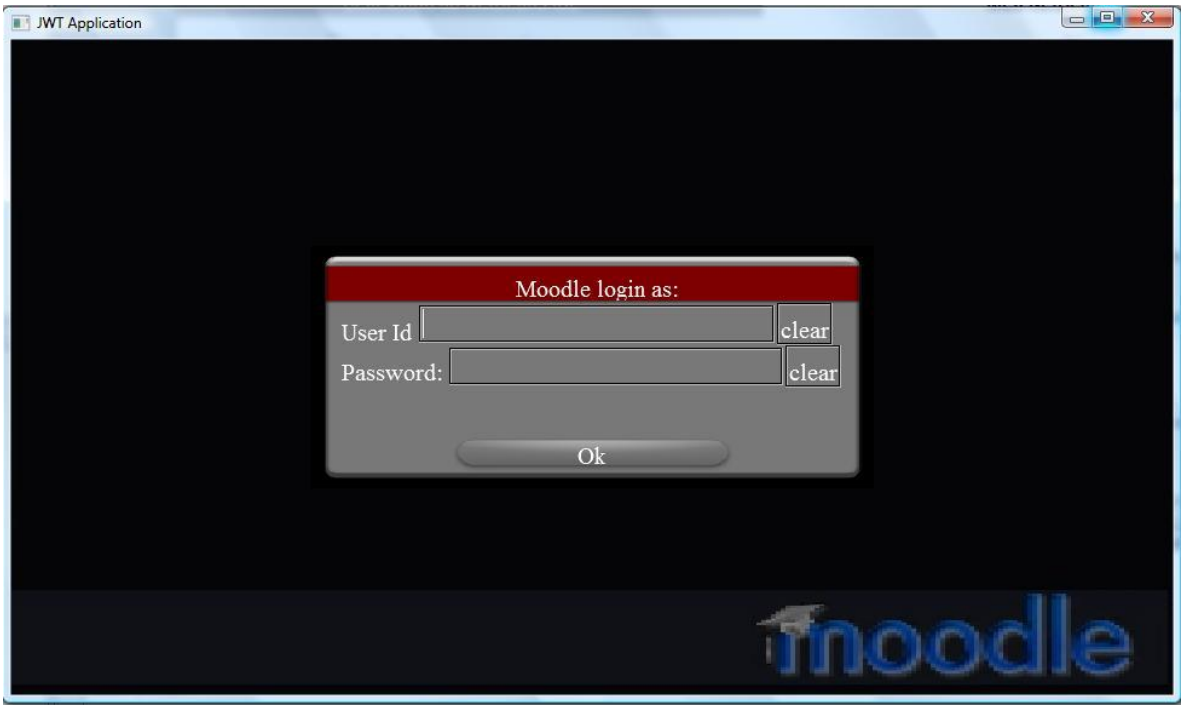

Figura 13. Caixa de diálogo da tela de login do aplicativo (Obs. A caixa de diálogo ganha destaque em relação ao plano de fundo)

- A Figura 13 exibe a tela de autenticação do usuário e senha. Em uma caixa de diálogo, foram criadas caixas de entrada de texto, botões de correção do texto e botão de envio dos dados. Caso haja algum erro, após o efeito de transição, novamente é exibida esta tela.

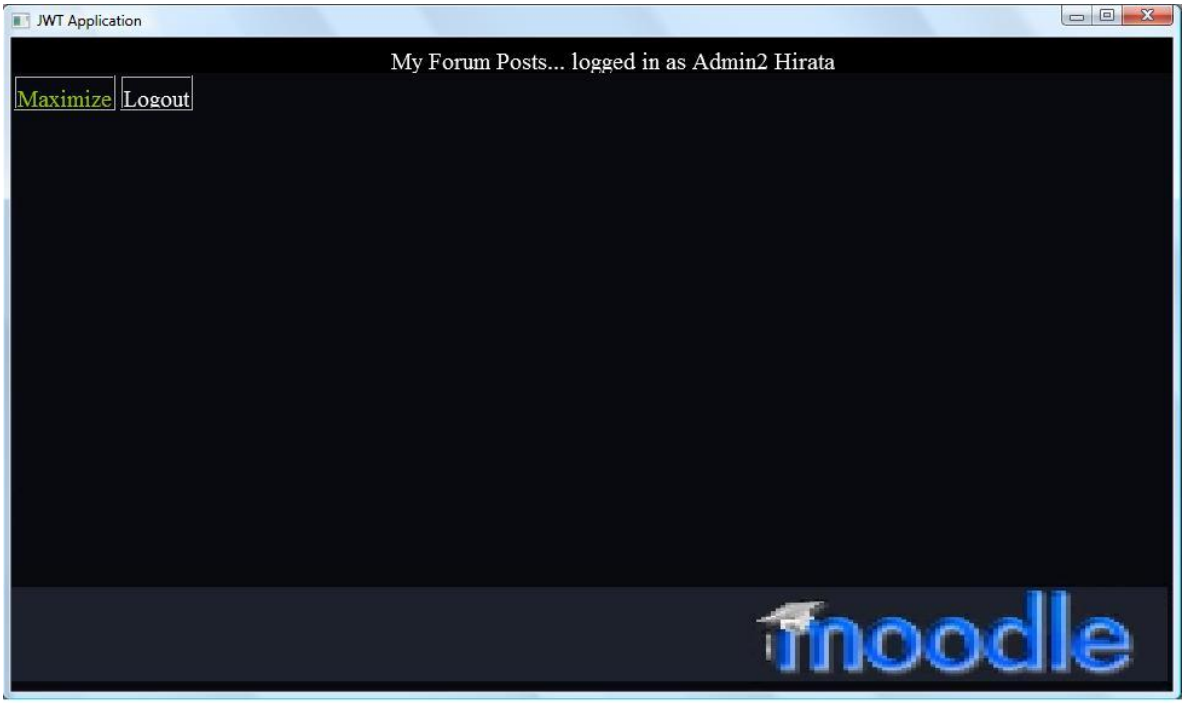

Figura 14. Os painéis de exibição de mensagens são ocultados ao pressionar o botão “Minimize” (que é substituído por “Maximize”)

- No painel de apresentação há botões de minimização e maximização da tela, botão de saída do sistema, textos informando nome do usuário, último acesso ao sistema web (Figura 14), e foto do usuário caso este possua um (caso contrário, o sistema busca a imagem de usuário sem foto do Moodle) (Figura 15).

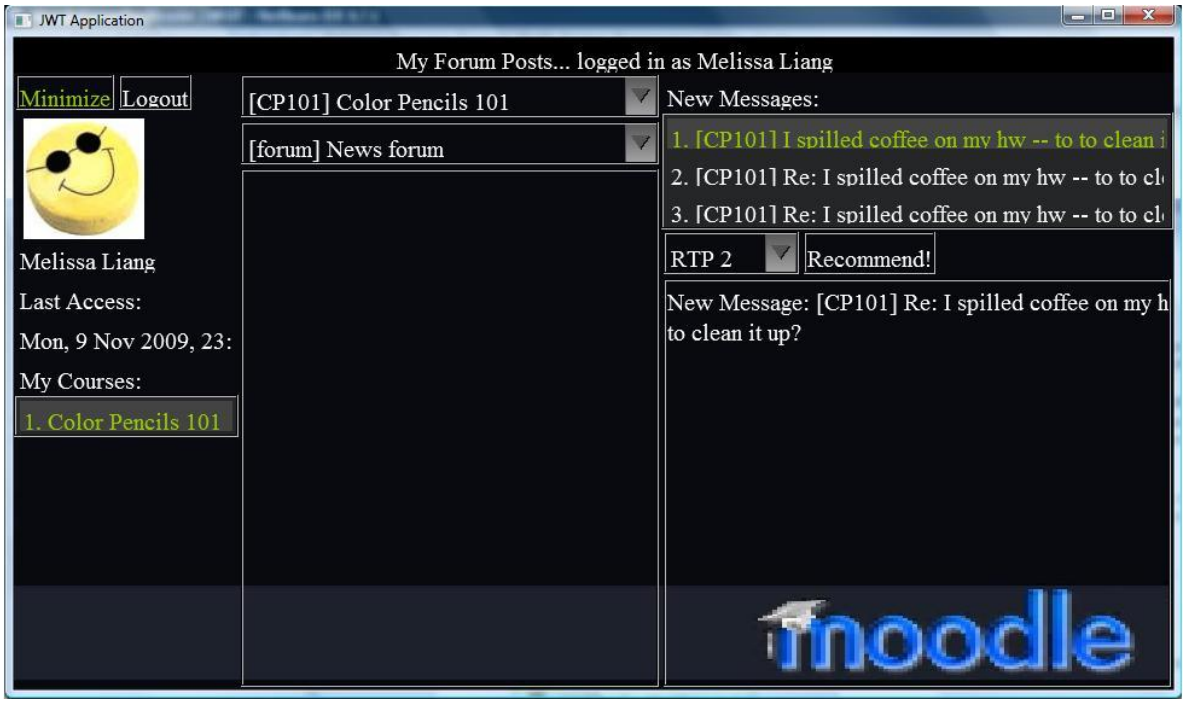

Figura 15. Imagem de usuário sem foto, nome do usuário e último acesso

- As figuras 15 e 16 exibem também a busca e notificação de novas mensagens: de tempos em tempos, o programa verifica novas mensagens enviadas no banco de dados, atualiza os menus e painel de mensagens não-lidas e exibe um recado na caixa de notificações.

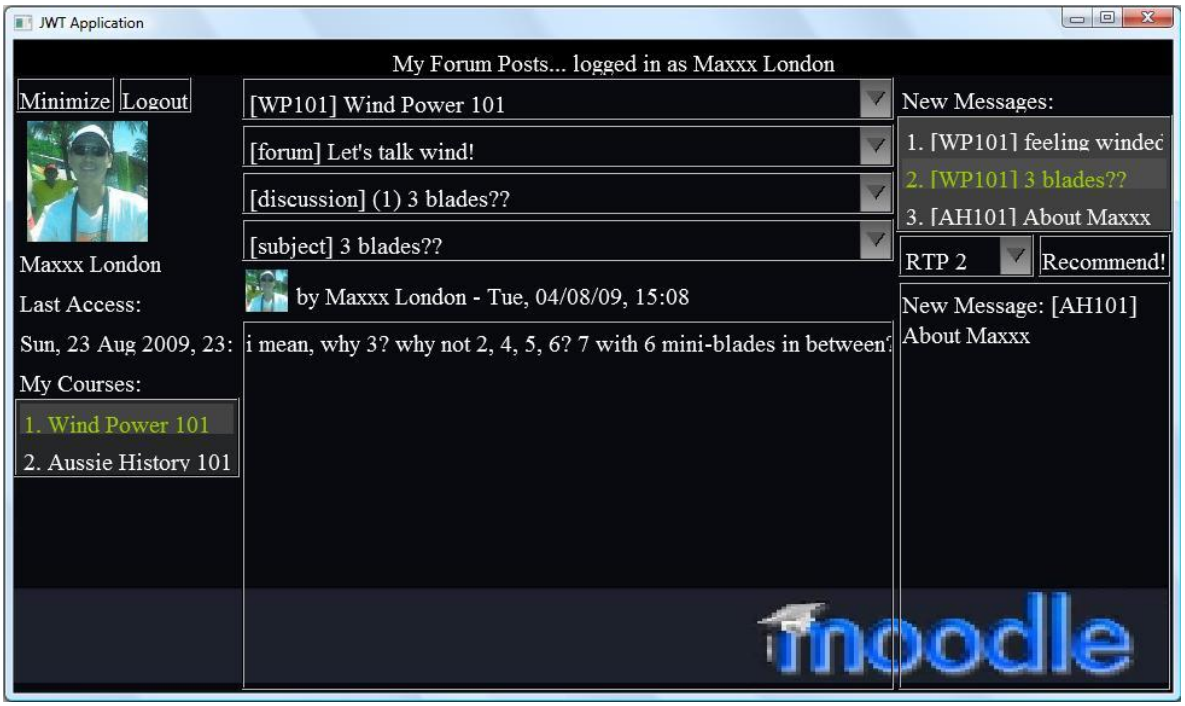

Figura 16. Mensagem do fórum de discussões no Moodle acessado pelo aplicativo de TV Digital

- A Figura 16 exibe a busca de mensagens dos fóruns dos cursos do usuário implementada com menus que organizam a navegação por curso, fórum de discussão, tópico de discussão, e mensagens enviadas. A quantidade de mensagens é exibida ao lado do nome do tópico de discussão. Caso haja apenas um item em algum dos menus, ele é automaticamente selecionado (listando os itens do menu abaixo) de forma a exigir menos movimentos entre os menus.

Há também um painel à esquerda, listando os cursos do usuário, também utilizável para filtrar mensagens de determinado curso, similar ao sistema web. No lado direito, também familiar ao usuário do sistema web, um painel com as mensagens não-lidas. Este painel lista as mensagens, organizadas por curso, e são automaticamente exibidas ao navegar sobre elas com as setas do teclado (sem ter que selecionar utilizando a tecla “enter”). As mensagens são exibidas em uma caixa de texto com a foto e nome do autor e horário de envio, seguindo o formato do Moodle. Finalmente, se alguma mensagem possuir um anexo (documento enviado ao fórum), exibe-se uma notificação sob o autor da mensagem com o nome do arquivo enviado.

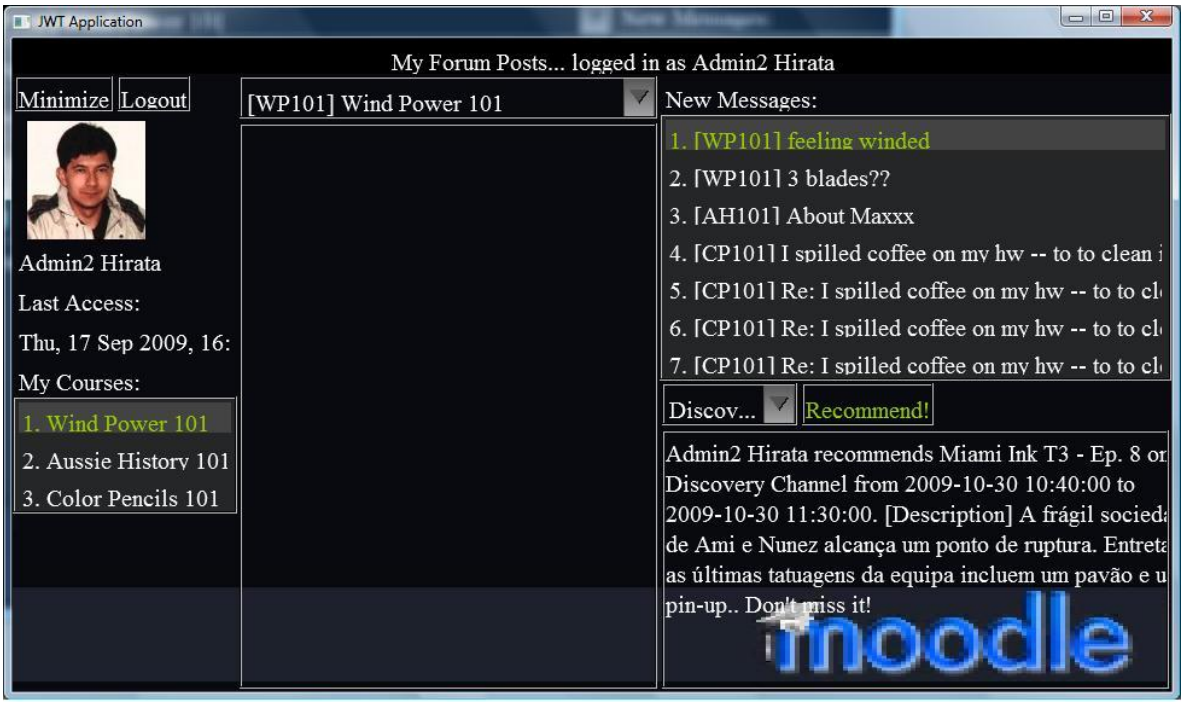

Figura 17. Mensagem recebida de recomendação de programação de TV

- O Envio de recomendações da programação de TV a outros usuários foi organizado em um menu de canais existentes do web service EPG e um botão de envio da recomendação (Figura 17). As notificações de recomendação são exibidas na caixa de notificações.

Caso haja menus ou caixas de notificação que tenham muito texto, o emulador automaticamente formata estes itens para exibirem barras de rolagem e ficarem com uma largura um pouco maior que o tamanho padrão dos itens. Estas barras de rolagem podem ser utilizadas tanto pelo mouse quanto pelo teclado.

Foi desenvolvido então um aplicativo que realiza tanto o envio quanto o recebimento de mensagens. Por pensar na usabilidade do sistema numa televisão com controle remoto numérico e não num emulador, a mensagem enviada está pré-definida e o usuário não precisa escrever um texto (apesar do emulador permitir a inserção de textos com teclado do tipo QWERTY ou de celulares - T9).

Capítulo V – Conclusões

O aprendizado e os maiores desafios.

Neste projeto, foi implementado um aplicativo para TV digital para acessar informações do ambiente Moodle. Especificamente, o aplicativo realiza o acesso por usuário e senha às mensagens do fórum, separados por curso, fórum de discussões, tópico de discussão e mensagem. Há também a possibilidade de acessar as mensagens não lidas pelo painel de mensagens novas, assim como receber notificações sobre mensagens e recomendações de programas de TV vindas de outros usuários.

Alguns desafios foram encontrados, por tratar de uma tecnologia ainda em desenvolvimento em nosso país. Destacam-se as seguintes:

- Falta de referências, indefinição de normas. As normas para a parte Java do middleware (Ginga-J) ainda não foram finalizadas até o momento da escrita deste texto. Esta dificuldade impede os fabricantes de lançarem um *set-top box* com o middleware completo, impossibilitando testes com sistemas reais. Além disso, a questão de quais APIs e quais versões destas deviam ser utilizadas ficava sem resposta. Também estavam indefinidas quais canais de retorno serão considerados oficiais para a TV digital. Estas indefinições impedem o desenvolvedor de escrever programas inteiramente compatíveis com o middleware Ginga.

- A diferença da implementação do software entre emuladores e *set-top boxes* comerciais tanto para aplicativos Java quanto NCL dificultaram o desenvolvimento de aplicativos robustos universalmente. Algumas implementações dos APIs, como o de sintonia de canal e de EPG não eram iguais nos emuladores e nos *set-top boxes*.

- A necessidade de canal de retorno para receber dados ao invés do carrossel de dados (não pode ser emulada) distorce a verdadeira função deste canal. O carrossel de dados é fornecido pela emissora, ou seja, transmite dados que podem ser acessados por qualquer usuário. Como aplicativos interativos deveriam ser transmitidos para todos os usuários, eles deveriam ser recebidos, decodificadas e interpretadas da mesma forma que o sinal digital. O canal de retorno deveria apenas enviar dados específicos do usuário ou buscar mais informações de acordo com o perfil do usuário.

- As questões de usabilidade e design, como resolução de tela (pouco texto permitido, uso de fontes legíveis a distância) exigiam muito estudo e planejamento do espaço de exibição para reproduzir as funcionalidades essenciais sem prejudicar a experiência e o design do sistema web. No sistema web é permitido bastante informação visual e textual. Por outro lado, na TV, recomenda-se pouca informação textual. O desafio foi portar o sistema com destaque nos recursos visuais.

- A falta de privacidade (ambiente familiar com várias pessoas) vai em direção oposta ao ambiente web. Esta questão faz com que o sistema seja de uso limitado a certos

perfis de usuários (que convivem com poucas pessoas) ou ambientes (quartos, escritórios, etc.)

- O problema de autenticar os usuários e as senhas identificadas por números (pelo uso do controle remoto numérico para entrar senhas alfanuméricas) contra as senhas criptografadas no banco de dados foi um desafio que exigiu um estudo na área de criptografia e autenticação. Este tipo de problema decorreu das diferenças entre web e TV e deve ser tratada com cuidado para manter qualquer sistema destes seguro.

- O problema com dispositivos diversos de entrada de dados (tanto numa TV ou no emulador) como teclado, mouse, controle remoto, gera a necessidade de realizar vários testes de uso e navegação de modo a comportar estas diferentes formas de interação. Especialmente para teclados e controles remoto, é preciso definir sequências de objetos selecionáveis de forma a manter a média de cliques ou pressionamento de botões para realizar determinada tarefa.

- Manter o design da interface do aplicativo similar ao sistema é um desafio para facilitar a navegação, melhorar a curva de aprendizado do novo sistema, e melhorar a experiência do usuário com os dois sistemas [28].

Estes desafios foram os encontrados neste projeto. Como o objetivo foi realizar o porte de determinadas funcionalidades de um sistema web para um sistema de TV digital e observar como lidar com as incompatibilidades entre estes, foram listados os tópicos com os principais pontos que um programador deverá se atentar. Naturalmente, não é possível prever todas as situações em que o desenvolvedor de aplicativos interativos deparará pois foi utilizado como base um sistema específico de educação a distância com características próprias e o trabalho foi desenvolvido em um emulador específico para Java. Além disso, como as normas para o middleware Ginga ainda estão em estudo pelo Fórum SBTVD, ainda há a possibilidade do padrão mudar antes de se oficializar. No entanto, o trabalho realizado documenta estas e outras dificuldades, relatando as principais características de aplicações interativas e uma abordagem que foi utilizada para realizar a experiência. O estudo foi bastante proveitoso para realizar testes com diferentes tecnologias web que puderam ser aproveitadas para a TV digital. Justamente ao usar uma linguagem já difundida na web, Java, tanto no middleware como no aplicativo, muito pode ser portado sem grandes mudanças de paradigma. Esta convergência, bem vista pela comunidade de programadores, abre muitas oportunidades e auxilia o aprendizado do desenvolvedor iniciante nesta área em expansão. Ao final, foi possível trabalhar as dificuldades, e elaborar um aplicativo interativo para TV digital que ilustra seu potencial de serviços diferenciados, como inclusão digital e apoio ao ensino.

Capítulo VI – Referências Bibliográficas

Artigos relacionados.

[1] Asynchronous HTTP and Comet architectures, http://www.javaworld.com/javaworld/jw-03-2008/jw-03-asynchhttp.html
[2] Exposing a MySQL Database with RESTful Web Services, http://www.netbeans.org/kb/60/websvc/rest-mysql
[3] MHP Tutorial, http://www.scribd.com/doc/3910826/MHP-Tutorial
[4] Ginga-J, http://gingarn.wikidot.com/gingaj
[5] TV Digital Brasileira, http://www.dtv.org.br/
[6] JavaTV, https://javatv-developers.dev.java.net/
[7] kXML, http://kxml.objectweb.org/
[8] Parsing XML in J2ME, http://developers.sun.com/mobility/midp/articles/parsingxml/
[9] Tutorial de Instalação do JavaTV e XletView, http://itvd.files.wordpress.com/2009/04/tutorial_javatv_xletview.pdf
[10] Java Magazine, Edição 66, http://www.linux.ime.usp.br/~melissa/mac499/tvdig/2009-04-17%20articletvdig/artigo3.pdf
[11] NETO, C. S. S., SOARES, L. F. G., RODRIGUES, R. F., BARBOSA, S. D. J., Construindo Programas Audiovisuais Interativos Utilizando a NCL 3.0 e a Ferramenta Composer, 2ª edição, 2007
[12] FERNANDES, J., LEMOS, G., SILVEIRA, G., Introdução à Televisão Digital Interativa: Arquitetura, Protocolos, Padrões e Práticas, Anais do JAI—SBC, 2004. Disponível em http://www.cic.unb.br/docentes/jhcf/MyBooks/itvdi/texto/itvdi.pdf
[13] MONTEZ, C., BECKER, V., TV Digital Interativa, Conceitos, Desafios e Perspectivas para o Brasil, 2ª edição, Editora da UFSC, Florianópolis, SC, 2005
[14] COSTA, J., EMTV – Extensão de Middleware para TV Digital com Componentes de Software para Educação, Manaus, AM, 2008
[15] TV Digital: a realidade do sistema no Brasil, http://olhardigital.uol.com.br/links/video_wide.php?id_conteudo=7558
[16] Clube NCL, http://clube.ncl.org.br/
[17] Aplicações Interativas no Mundo, http://b4it.blogspot.com/2007/11/aplicaes-interativas-no-mundo.html
[18] Interactive TV Today, http://www.itvt.com
[19] Introdução a TV Digital, http://www.linux.ime.usp.br/~melissa/mac499/tvdig/intTVdig.ppt
[20] Java Web, http://www.caelum.com.br/downloads/apostila/caelum-java-web-fj21.pdf
[21] Ginga no Sistema Brasileiro de TV Digital: Concepção de Programas Interativos, http://www.linux.ime.usp.br/~melissa/mac499/tvdig/Palestra-Encasoft-vf.ppt
[22] GTTV – Grupo de Trabalho de Televisão Digital, http://www.linux.ime.usp.br/~melissa/mac499/tvdig/GT-TV-apresentacao.pdf
[23] COSTA, J., LUCENA Jr., V., EMTV – A Component-Based DTC Middleware Extension for Educational Purposes, EuroITV 2008, LNCS 5066, pp. 219 – 228, 2008.

[24] Montez, Carlos; Becker, Valdecir. TV Digital Interativa: Conceitos e Tecnologias. In: WebMidia e LA-Web 2004 – Joint Conference. Ribeirão Preto, SP, Outubro de 2004. http://www.das.ufsc.br/~montez/publications/2004%20MinicursoWebMidia.pdf
[25] OpenGinga, http://www.openginga.org
[26] De Franco, B. B., Oliveira, H. C.. "Proposal for Convergence of E-learning Systems for T-learning". Conference ICL2007 September 26-28, 2007 Villach, Austria. http://online-journals.org/i-jet/article/viewArticle/192
[27] PAZOS-ARIAS, J. J., et al, Provision of distance learning services over Interactive Digital TV with MHP, Computers & Education 50 (2008) 927–949
[28] PICCOLO, L. S. G. ; BARANAUSKAS, M. C. C. . Desafios de Design para a TV Digital Interativa. In: VII Simpósio de Fatores Humanos em Sistemas Computacionais, IHC 2006.
[29] http://web.infomoney.com.br/templates/news/view.asp?codigo=818749&path=/suasfinancas/
[30] http://idgnow.uol.com.br/internet/2009/09/18/ibge-23-8-dos-domicilios-brasileiros-tem-acesso-a-internet/
[31] http://www.teleco.com.br/tutoriais/tutorialtvd/pagina_2.asp
[32] Web Service de programação de TV http://developers.blogs.sapo.pt/4736.html
[33] Proview XPS-1000: http://planetech.uol.com.br/?p=4186
[34] Proview XPS-1000: http://www.convergenciadigital.com.br/cgi/cgilua.exe/sys/start.htm?infoid=14979&sid=54
[35] Set-top boxes com ginga / JavaTV: http://www.inf.ufrgs.br/~tpbiazus/pelotas/12-benchmarking_de_settopboxes_para_tv_digital.pdf ----e----
[36] http://www.tvdi.inf.br/upload/artigos/tvdigital-tchelinux.pdf
[37] CCSL http://ccsl.ime.usp.br/
[38] Java@TV Digital
[39] Parsing XML in J2ME http://developers.sun.com/mobility/midp/articles/parsingxml/
[40] http://moodle.org/stats
[41] The Lobo Project http://lobobrowser.org/

Capítulo VII – Parte Subjetiva

O projeto e a experiência no IME.

Para realizar este projeto, encontrei vários desafios e dificuldades diretamente envolvidos ou não à pesquisa que teve de ser feita para a implementação. Em relação ao curso do BCC em si, as maiores dificuldades ficaram por conta da pouca convivência que tive com a área de pesquisa. Isto ocorreu pois meu supervisor é relativamente novo no Instituto e não tive a chance de cursar as matérias optativas que ele ministrou. No entanto, acredito que tive certa afinidade com a área após realizar leituras de artigos relacionados e ouvido palestras de apresentação das pesquisas de área dos alunos de mestrado. Também tive a oportunidade de conversar com vários destes e outros alunos para entender, do ponto de vista dos alunos, que tipo de trabalho e abordagem deveriam ser estudados. Este tipo de aprendizado vindo de colegas é raramente reconhecido ou lembrado por nós. Cito talvez a natureza dos próprios alunos do Instituto como razão, que não deixa de ser mais uma das dificuldades enfrentadas durante a graduação.

Entre as dificuldades técnicas encontradas, a maior delas foi indubitavelmente a falta de referências encontradas. Isto é natural por se tratar de uma área ainda em desenvolvimento no país de uma tecnologia ainda não estabelecida nem padronizada. As normas previstas para o middleware Ginga ainda se encontram incompletas e com maiores atrasos, devido a disputas políticas entre as grandes entidades envolvidas. Além da falta de referências, houve também a falta de conhecimento e trocas de informações entre os interessados da área. Apesar da participação no evento de TV digital e em listas de discussão, percebi que a empolgação dos envolvidos logo se perdia pela falta de respostas para as dúvidas. Além disso, fomos tomar conhecimento do grupo de pesquisas de TV digital na Escola Politécnica apenas no último mês de trabalho. Perdeu-se a oportunidade de juntar os grupos de estudos para realizar um trabalho mais avançado.

Todos estas dificuldades, no entanto, não são vistas como impedimentos, pois por outro lado, percebe-se como a área ainda permite muito desenvolvimento e certa liberdade de pesquisa e experiência. O lado positivo de trabalhar com uma área nova é que ela é muito promissora e estimulante. Neste projeto foi possível realizar diversas experiências com conceitos e APIs diversos de modo a chegar a resultados concretos. Para levar o projeto de adaptar recursos de sistemas web a TV digital, poderia-se realizar um estudo mais profundo de design de sistemas e interfaces, além de firmar a parceria com a Escola Politécnica com o objetivo de convergir as pesquisas e trabalhar com receptores de sinal fabricados de acordo com o que será lançado no mercado futuramente. O estudo de design de sistemas e interfaces facilitaria o porte de outros recursos web requerendo menos esforço e maior reaproveitamento de recursos (lembrando que a capacidade de processamento e memória dos receptores são limitados). E ao realizar testes em receptores reais, seria possível avaliar a robustez e eficiência dos aplicativos desenvolvidos. Por fim, seria interessante cursar algumas optativas relacionadas a sistemas para ter uma base mais sólida de conhecimentos de padrões e técnicas de programação.

Algumas disciplinas cujo conteúdo foi aproveitado para este trabalho de formatura são:

- Laboratório de Programação I e II: pela introdução a orientação a objetos, utilização de bibliotecas e APIs Java
- Programação Orientada a Objetos: para entender e utilizar linguagens orientadas a objetos
- Programação Concorrente: para coordenar diferentes partes do projeto que rodam em paralelo

Ao final, é bastante prazeroso ter finalizado um projeto em uma área ainda em desenvolvimento no país e ter conseguido enfrentar os desafios encontrados. Foi bastante interessante também atuar em uma área não muito familiar para mim e conseguir resultados interessantes. Sentia grande dificuldade em trabalhar com sistemas grandes e não em exercícios-programas pequenos e de escopo limitado. Este projeto, como a minha experiência na graduação em geral, me ajudou a perceber minhas capacidades e deficiências, e mais importante, buscar como superá-las.