Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Serviço de Aplicações: Proposta de Arquitetura para o Sistema Brasileiro de Televisão Digital

Welington da Veiga Silva

JUIZ DE FORA
DEZEMBRO, 2011

Serviço de Aplicações: Proposta de Arquitetura para o Sistema Brasileiro de Televisão Digital

Welington da Veiga Silva

Universidade Federal de Juiz de Fora
Institudo de Ciências Exatas
Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Eduardo Barrere

JUIZ DE FORA
DEZEMBRO, 2011

Serviço de Aplicações: Proposta de Arquitetura para o Sistema Brasileiro de Televisão Digital

Welington da Veiga Silva

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUDO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Eduardo Barrere
D. Sc.

Marcelo Lobosco
D. Sc.

Eduardo Pagani
M. Sc.

JUIZ DE FORA

1 DE DEZEMBRO, 2011

Aos meu pais, pelo apoio e confiança

Resumo

Neste trabalho propomos uma Arquitetura para Serviço de Aplicações para Televisão Digital que busca facilitar o acesso e a distribuição de aplicativos no Sistema Brasileiro de Televisão Digital, criando uma Central de Software similar às existentes em outras plataformas, como o Android Market, a *Apple App Store* e o *Ubuntu Software Center*. Através dessa central o usuário pode obter e executar facilmente aplicativos multimídia, os desenvolvedores podem utilizar um padrão de empacotamento que facilite sua distribuição em vários serviços, e os provedores de serviços passam a poder oferecer como diferencial um rico ambiente interativo a seus clientes.

Palavras-chave: SBTVD, Central de Software, Lojas de Aplicativos, Widgets, Televisão Digital.

Abstract

In this paper we propose an architecture for Application Service for Digital TV that seeks to facilitate access and distribution of applications in the Brazilian System of Digital Television by creating a Software Center similar to those existing in other platforms such as *Android Market*, *Apple App Store* and *Ubuntu Software Center*. Through this center the user can easily obtain and run multimedia applications, developers can use a standard packaging to facilitate their distribution in various services and service providers can now offer as differential a rich interactive environment to theirs customers.

Keywords: SBTVD, Software Center, Applications Stores, widgets, Digital TV.

Agradecimentos

Agradeço ao meu orientador, pelo auxílio na pesquisa,

Aos meus pais, mestres para muito além do que está nos livros,

À Priscila, pelo carinho, apoio, por entender minha ausência todo fim de período e ainda assim aceitar se casar comigo,

Ao amigo Claudson, pela filosofia despreocupadamente discutida na cozinha entre um plano e outro para dominar o mundo.

E aos demais companheiros e aos professores pelo apoio, a atenção e o bom humor.

És livre, escolhe, ou seja: inventa.

Jean-Paul Sartre

Sumário

Lista de Figuras

Lista de Abreviações

API	Interface de Programação de Aplicativos
HTTP	HyperText Transfer Protocol
JSR	Java Specification Request
PDA	Personal Digital Assistants
PSP	PlayStation Portable
RESTful	Representational State Transfer
RSS	Really Simple Syndication
SOAP	Simple Object Access Protocol
XML	EXtensible Markup Language
XMPP	EXtensible Messaging and Presence Protocol

1 Introdução

A disponibilidade de aplicativos é um fator cada vez mais relevante para o sucesso de um dispositivo ou plataforma à medida que celulares, *smartphones*, *tablets*, consoles e outros dispositivos capazes de executar aplicativos e acessar à internet tornam-se populares.

Nesse contexto, as lojas de aplicativos como a *Apple App Store*[1] e o *Android Market*[2] atuam como Centrais de Software, tanto facilitando o acesso a aplicativos gratuitos quanto a sua comercialização, apresentando-se como um modelo de negócio interessante para usuários, desenvolvedores de aplicativos, e fornecedores de dispositivos.

A *Apple App Store*, por exemplo, foi lançada em 2007 e em 2011 atingiu a marca de 15 bilhões de downloads entre os mais de 425 mil aplicativos disponíveis aos cerca de 200 milhões de dispositivos, em mais de 90 países(Wire, 2011).

Com o sucesso da *Apple App Store*, acessível apenas pelos dispositivos da empresa, viu-se a expansão de serviços similares. A própria Apple inaugurou em 2010 a *Mac App Store*[3], uma loja de aplicativos no mesmo modelo disponível para seu Sistema Operacional. Antes disso, em 2009, o Ubuntu[4] já trazia embarcado o *Ubuntu Software Center*[5], uma Central de Software com aplicações gratuitas que podem ser encontradas e instaladas facilmente, só ganhando suporte à aplicações comerciais em 2011. Na plataforma Android, o *Android Market*, lançado em 2008, cresce rapidamente e já ultrapassa o número de *downloads* da *Apple App Store* e provavelmente irá superá-la também no total de aplicações disponíveis em 2012 (Idgnow, 2011). Também estão disponíveis lojas de aplicativos para consoles, como o *Xbox Live Marketplace* [6] do XBox 360 e a *PlayStation Store* [7] do PlayStation e do PSP.

Centrais de Software também podem ser um serviço disponível em sistemas de televisão Digital capazes de executar aplicações e ter acesso à rede, como o *Internet*

[1] http://www.apple.com/app-store/
[2] https://market.android.com/
[3] http://www.apple.com/mac/app-store/
[4] http://www.ubuntu.com
[5] http://www.ubuntu.com/ubuntu/features/ubuntu-software-centre
[6] http://marketplace.xbox.com/en-US/
[7] http://us.playstation.com/psn/playstation-store/

Protocol TeleVision (IPTV) e o Sistema Brasileiro de Televisão Digital (SBTVD). No IPTV, aplicações estão disponíveis e são frequentemente chamadas de *widgets*. Inicialmente *widgets* eram pequenas aplicações restritas a uma funcionalidade, como gráficos de temperatura e leitores de RSS[8], mas posteriormente o termo passou a ser aplicado a qualquer aplicação para IPTV(Shin, 2008).

Há um esforço para padronização de *widgets* e uma recomendação está sendo redigida[9] pela *Internet Protocol Television Global Standards Initiative* (ITU-T IPTVGSI)[10], propondo um empacotamento de *widgets*.

O Ginga, *middleware* especificado no SBTVD(ABNT, 2008), é uma plataforma completa de execução de aplicações interativas(Borges, 2007) com ambientes de desenvolvimento imperativo e declarativo e uma rica API, capaz oferecer Centrais de Software tão completas quanto as existentes em outras plataformas.

Uma Central de Software de fácil acesso, capaz de oferecer de forma transparente uma ampla variedade de aplicações, pode trazer maior interesse de desenvolvedores de software, companhias de publicidade, emissoras, fabricantes de hardware e investimento em tecnologias que demonstrem o real potencial do SBTVD enquanto plataforma de execução de aplicações.

Neste trabalho propõe-se uma Arquitetura de Serviços de Aplicações para o Ginga no capítulo 4, o Serviço de *Widgets* Ginga (SWG), aplicando o modelo de negócio estabelecido nas Centrais de Software de outras plataformas, definido no capítulo 2, e adaptando-o à arquitetura do SBTVD, detalhada no capítulo 3, porém mantendo a compatibilidade com a especificação de *widgets* para IPTV na versão atual da recomendação ITU-T IPTV-GSI. No capítulo 5 discutimos a implementação de referência da arquitetura definida e no capítulo 6 abordamos os resultados obtidos.

[8] http://cyber.law.harvard.edu/rss/rss.html
[9] http://www.itu.int/md/T09-IPTV.GSI-110516-TD-GEN-0423/en
[10] http://www.itu.int/en/ITU-T/gsi/iptv/Pages/default.aspx

2 Trabalhos Relacionados

Lojas de aplicativos tem chamado a atenção da indústria, dos usuários de vários dispositivos, dos desenvolvedores de aplicações e da comunidade acadêmica. A razão é bastante simples: um dos princípios base da *Apple App Store*, por exemplo, é atrair recursos de desenvolvedores de aplicações externos, e não restringir-se à própria capacidade de criação, aumentando o número de aplicações disponíveis e sua variedade(Kim, 2010), gerando um diferencial competitivo para a plataforma.

Os fatores de sucesso das principais Lojas de Aplicativos são tema de vários estudos que comparam sua arquitetura com o modelo de obtenção de aplicações existente anteriormente, evidenciando a importância de uma Central de Software, mas distinguindo seus possíveis graus de abertura(Holzer, 2009) na aceitação de aplicações de terceiros. Esta é uma diferença entre o Android, que não obriga o usuário a utilizar o Market como único portal nem possui uma rígida política para aceitação de aplicações e o iOS, que acessa exclusivamente a *Apple App Store* e possui uma criteriosa avaliação de todas as aplicações submetidas por terceiros(Mcdaniel, 2010).

O *Ubuntu Software Center*, por sua vez, permite que o usuário adicione Canais de Software, ou Repositórios, de terceiros que passam a poder disponibilizar aplicações à Central de Software sob controle do usuário, mas sem qualquer necessidade de aprovação centralizada e nem garantias de que as aplicações funcionem como o desejado e não possuam códigos maliciosos. Este é outro aspecto que concentra discussões em torno de Centrais de Software e Lojas de Aplicativos: a segurança das aplicações obtidas através desses serviços e seu respeito à privacidade dos usuários.

É claro que é impossível testar cada uma das aplicações submetidas a esses serviços(Mcdaniel, 2010) avaliando vulnerabilidades, violações dos termos de privacidade ou presença de códigos maliciosos. Cabe às plataformas criar uma camada de segurança entre aplicações e os recursos e dados dos dispositivos a fim de minimizar os riscos de se instalar um aplicativo malicioso. Outra recurso disponível às lojas é disponibilizar formas dos próprios usuários inibirem o avanço na adoção dessas aplicações maliciosas(Mcdaniel,

2010), seja avaliando negativamente ou denunciando as violações das políticas estabelecidas pela Central de Software.

A convergência entre televisão e internet com os Sistemas de Televisão Digital traz novas possibilidades, oportunidades e desafios(Savarese, 2009) que, se comparados com os serviços *web* oferecidos nos computadores pessoais, ainda estão em sua infância, oferecendo uma oportunidade para desenvolvedores de conteúdo, designers, emissoras, publicitários e fabricantes(Shin, 2008), além de vários desafios, como a adaptação dos serviços web para as características de tela, conteúdo, dispositivos de entrada de dados e público da televisão.

Provedores de Serviços IPTV podem adicionar aplicações multimídia a seu produto que além de vídeo, áudio, imagens e texto podem oferecer interatividade, serviços web, jogos e etc. No entanto, Serviços IPTV são conhecidos por possuir múltiplos fornecedores(Mikoczy, 2008) que desenvolvem produtos baseados nos próprios software e hardware(Moreno, 2010). Neste cenário, nos últimos anos ganharam força várias iniciativas de padronização de serviços IPTV(Mikoczy, 2008). Entre elas as recomendações da série ITU-T H.760(ITU-T H.760, 2009) buscam identificar *frameworks* de aplicações multimídia relevantes e cuja primeira recomendação é o *Nested Context Language*[1] (NCL) através da recomendação ITU-T H.761(ITU-T H.761, 2009).

O *Internet Protocol Television Global Standards Initiative* (ITU-T IPTV-GSI) tem em desenvolvimento uma recomendação de serviços de *widgets*, que inclui a padronização de seu empacotamento, em um arquivo ZIP[2] com uma estrutura de diretórios padronizada e um arquivo descritor indicando metadados como o ícone, o nome, a versão e o arquivo principal onde a aplicação deve ser iniciada, além disso, a recomendação inclui as linguagens em que esses *widgets* devem ser escritos, entre as quais esta o NCL.

O Ginga-NCL[3] é dos ambientes do Ginga e possui entre os principais componentes o Formatador NCL(Damasceno, 2008) capaz de interpretar aplicativos escritos nesta linguagem, o que permite a adoção do padrão de empacotamento de *widgets* a ser recomendado pelo ITU-T IPTV-GSI para IPTV também no SBTVD.

[1] http://www.ncl.org.br/
[2] http://www.pkware.com/support/zip-app-note
[3] http://www.gingancl.org.br/

Seguindo esta convergência existem vários trabalhos trazendo Serviços Web para o SBTVD, como redes sociais(Ghisi, 2010), educação à distância(Mendes, 2010) (Silva, 2010)(Barrere, 2010), saúde(Becker, 2006) e t-commerce(Almeida, 2010) e desenvolvendo soluções tanto para a adequação técnica às particularidades do SBTVD quando na proposição de modelos de negócio, juntamente com esse serviços.

Outros trabalhos sugerem arquiteturas que criem soluções para o acesso a serviços web no SBTVD, como para a utilização do protocolo XMPP[4] como alternativa à HTTP para comunicação em tempo real(Reis, 2011), para acesso a redes sociais(Ghisi, 2010) e para descobertas de serviços(Borges, 2007), trazendo soluções exploradas em outras plataformas e adaptando-os à arquitetura do SBTVD. Esses trabalhos demonstram que o Ginga é uma plataforma de execução poderosa, e que o acesso à serviços em rede é uma realidade que pode ser explorada para trazer à televisão novas possibilidades de interação.

[4]http://xmpp.org/

3 O Sistema Brasileiro de Televisão Digital

O O Sistema Brasileiro de Televisão Digital (SBTVD) é o padrão de televisão digital terrestre brasileiro, instituído pelo decreto Nº. 4.901, de 26 de Novembro de 2003, que traz além de significativa melhora na qualidade de áudio e vídeo em relação à transmissão analógica, a possibilidade de aplicações interativas serem enviadas junto com programação, podendo ser executadas diretamente na televisão(Fernandes, 2004).

Dada a grande penetração das televisões nos lares brasileiros, 95,7% em 2010 segundo o Instituto Brasileiro de Geografia e Estatísticas (IBGE), uma das principais motivações por trás da especificação do SBTVD é, com essa possibilidade, criar formas de inclusão social(Castro, 2005)(Mendes, 2010) e digital(Becker, 2006).

A figura 3.1 ilustra a arquitetura do SBTVD, onde áudio, vídeo e dados são multiplexados em um único fluxo na emissora e transmitidos via broadcast para os terminais de acesso dos usuários, que demultiplexam o fluxo recebido, exibem o conteúdo de áudio e vídeo e executam as aplicações enviadas.O Middlware Ginga, o canal de retorno, e o carrossel de dados, estratégia de envio de dados de aplicações via broadcast, são conceitos do SBTVD importantes para o desenvolvimento do SWG e serão discutidos nas sub-sessões seguintes.

3.1 *Middleware* Ginga

Ginga é o *middleware* padrão do SBTVD(ABNT, 2008), implementado sob código aberto e especificado segundo norma ABNT em conformidade com a norma ITU J.200(ITU-T J.200, 2001). Um *middleware* se apresenta como a camada de software entre o hardware e o Sistema Operacional(as Aplicações), concentrando serviços como identificação, autenticação, autorização, diretórios, certificados digitais e outras ferramentas para segurança(Damasceno, 2008).

O Ginga possui dois ambientes, o Ginga-NCL e o Ginga-J, ambos utilizando uma infraestrutura comum chamada Ginga-*Core*, e podendo comunicar-se entre si através de

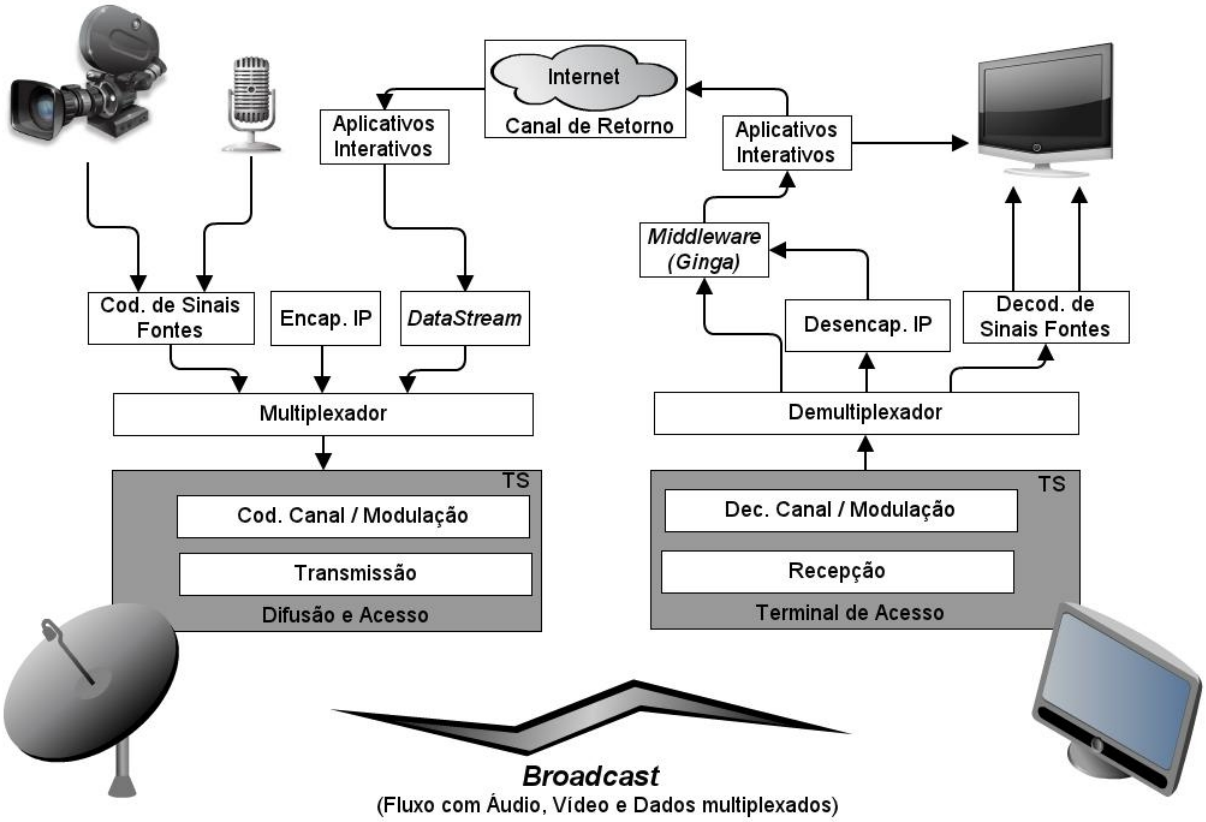

Figura 3.1: O Sistema Brasileiro de Telvisão Digital (SBTVD) (Adaptado de BARBOSA 2006)

uma 'ponte' (*bridge*) entre os dois subsistemas(Ghisi, 2010).

O Ginga-NCL é o ambiente de apresentação, declarativo e obrigatório(Damasceno, 2008). Utiliza NCL como linguagem de programação e possui um formatador para executar aplicações declarativas nesta linguagem, além de um exibidor XHTML e uma máquina de apresentação Lua[1](Ghisi, 2010), capaz de executar *scripts*imperativos.

O Ginga-J é o ambiente de execução, imperativo e opcional. Trata-se de uma plataforma distribuída(Damasceno, 2008) que executa uma Máquina Virtual Java[2] (JVM) podendo tirar proveito do grande número de bibliotecas, vasta documentação e possibilidade de utilizá-la para integrar-se com dispositivos móveis, como PDAs, *tablets* e *smartphones* além de uma gama de outros dispositivos que também possuam uma JVM.

[1]http://www.lua.org/
[2]http://www.java.com/en/

3.2 Carrossel de Dados

Os dados necessários para a execução de aplicações interativas no Ginga são transmitidos via broadcast juntamente com os fluxos de áudio e vídeo, um após o outro, havendo retransmissão do primeiro pacote de dados após a transmissão do último(Pessoa, 2008), como em um "Carrossel de Dados", como mostra a figura 3.2.

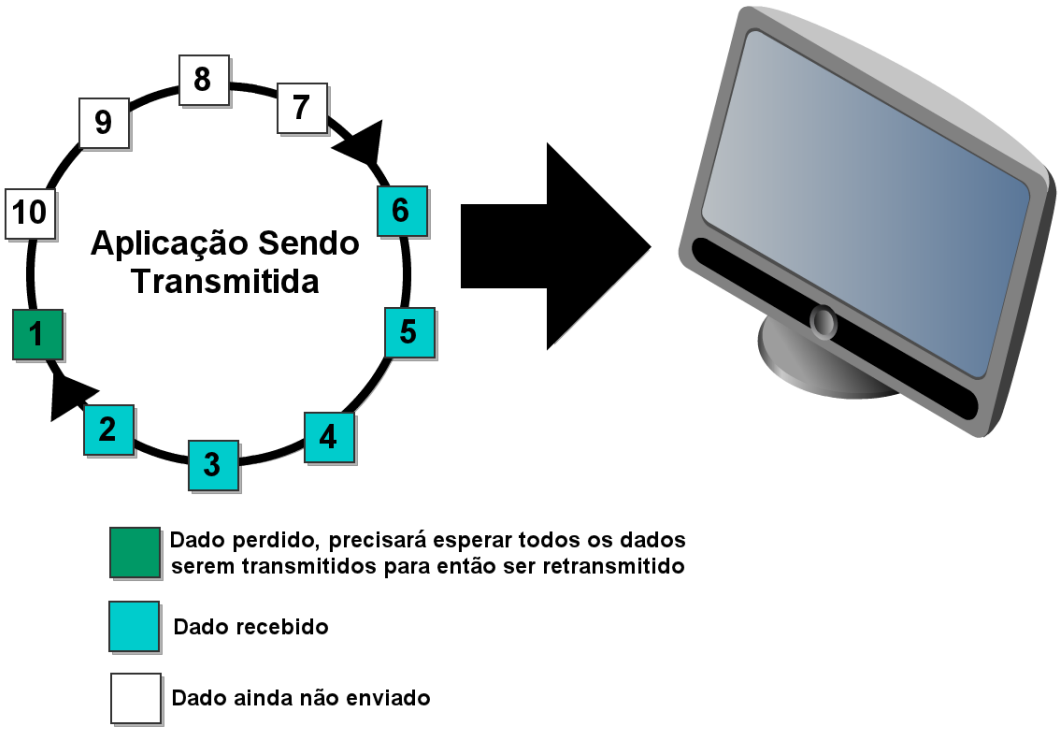

Figura 3.2: O Carrossel de Dados transmitindo uma aplicação

A retransmissão ocorre porque caso algum pacote de dados seja perdido durante uma mudança de canal, ou logo após a televisão ser ligada, por exemplo, a aplicação deve esperar que os dados anteriores, 'perdidos', seja retransmitido para então estar apta à execução da aplicação relacionada à programação em exibição.

Embora seja necessária, a retransmissão não é desejada por entrar em conflito com o modelo de negócio das emissoras(Pessoa, 2008), centrado na venda de intervalos de tempo a anunciantes que podem ter suas aplicações exibidas com atraso, ou até mesmo dessincronizadas de seu anúncio, enquanto a sua aplicação espera o tratamento da aplicação anterior a ela.

Essa limitação também sugere que as aplicações devem ser pequenas para min-

imizar o período de retransmissão, porém aplicações interativas podem ser hipermídias compostas por áudio, vídeo, imagens, e até mesmo bancos de dados (Pessoa, 2008).

3.3 Canal de Retorno

A proposta do governo para Televisão Digital tem por objetivo permitir a inclusão digital utilizando o acesso à Internet através da televisão. Com isso, torna-se necessário a implantação do canal de retorno ou interatividade(Mendes, 2010) entre o receptor e um servidor remoto, embora a forma da conexão do canal de interatividade não seja definida nas especificações do Ginga, que também não obriga a existência da mesma(Pessoa, 2008).

O canal de retorno pode conectar a televisão à uma rede doméstica, à uma rede privada fornecida pela emissora, à um provedor de serviço, ou à Internet. Conectar-se à uma rede expande as possibilidades de interação, permitindo que as aplicações interativas disponibilizem muito mais conteúdo do que seria possível via Carrossel de Dados, oferecendo uma experiência de Televisão Digital realmente interativa, onde o telespectador pode responder à uma enquete, enviar perguntas, realizar uma compras, e acessar vários serviços disponíveis na internet.

4 Arquitetura Proposta

Um Serviço de Aplicações para Televisão Digital deve ser genérico, escalável, transparente na obtenção e execução de aplicações, possuir uma Central de Software como as existentes nas plataformas móveis, permitir que o seu fornecedor tenha controle sobre o conteúdo oferecido e fornecer um padrão de empacotamento de aplicações para que desenvolvedores possam distribuílas entre vários fornecedores que utilizem essa arquitetura.

O Serviço de *Widgets* Ginga (SWG), proposto neste trabalho, busca atender a esses requisitos através de uma arquitetura distribuída, que separa a Central de Software onde as aplicações são acessadas pelo cliente dos repositórios de aplicações, onde elas são realmente oferecidas, estabelecendo, assim, dois atores principais: O Provedor de Serviço que distribui uma Central de Software onde pode adicionar e remover repositóriose o Provedor de Conteúdo, que disponibiliza seu repositório de aplicações.

Provedores de Serviço podem ser fabricantes de receptores, embarcando sua Central de Software diretamente no receptor, ou emissoras e seus parceiros, disponibilizando-a através do Canal de Retorno. Produtores de Conteúdo podem ser empresas de publicidade, estúdios de jogos, agências de notícias, prestadores de serviços, desenvolvedores independentes ou até mesmo o próprio Provedor de Serviço.

As aplicações disponibilizadas no SWG estão empacotadas em um ZIP, onde um arquivo XML (o config.xml) contém metadados do aplicativo e indica o primeiro arquivo que deve ser executado, além de uma estrutura de diretórios padronizada, adequando-se assim ao padrão descrito na recomendação em desenvolvimento pela ITU-T IPTV-GSI para Serviços de *Widgets* para IPTV. Seguir esta recomendação permite que um Provedor de Conteúdo distribua aplicações entre vários Provedores de Serviço sem necessidade de alterações e alinha o SWG aos serviços de *widgets* para IPTV, permitindo que os dois sistemas possam inclusive compartilhar aplicações NCL. Por isso sempre é utilizado o termo *widget* para às aplicações distribuídas pelo SWG.

Os três componentes principais do SWG são o Cliente-SWG, o Servidor-SWG e o Repositório-SWG. O Cliente-SWG é executado no receptor do cliente e corresponde

à interface da Central de Software e possibilita a execução de *widgets* localmente. O Servidor-SWG atua como um *backend* para a Central de Software. O Repositório-SWG disponibiliza os *widgets*. Os componentes são apresentados na figura 4.1.

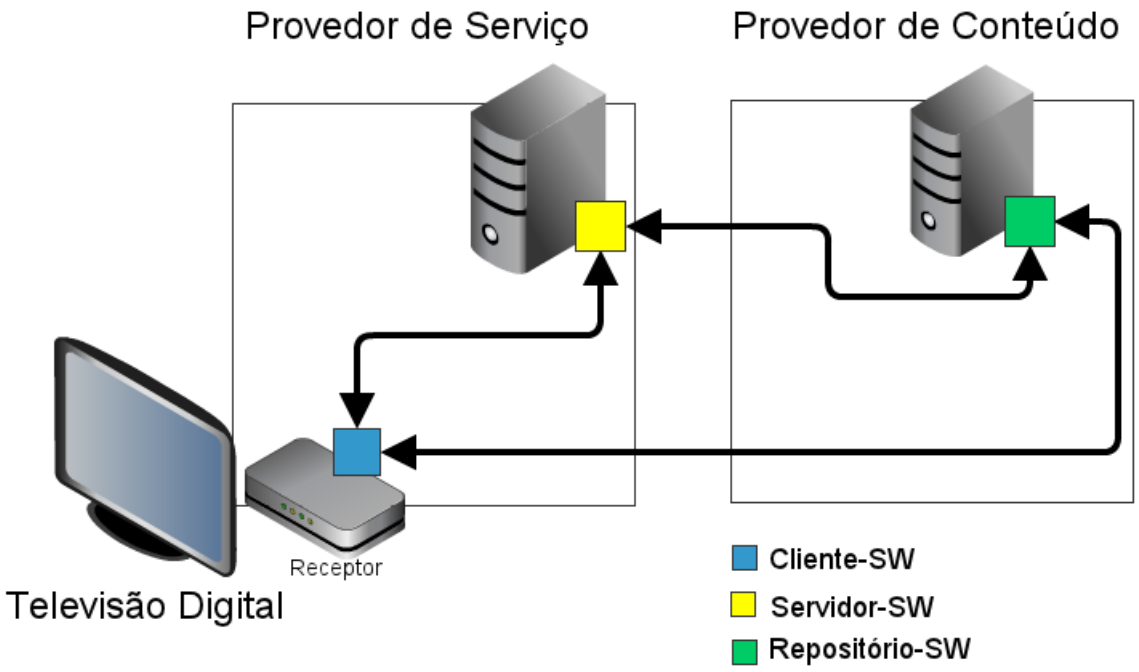

Figura 4.1: Componentes do Serviço de Widgets Ginga

A seguir cada um dos componentes do SWG é descrito juntamente com a forma com que se relacionam, em seguida é detalhada a comunicação entre eles, questões relacionadas à sua segurança e a possibilidade de criar extensões sobre a arquitetura proposta.

4.1 Cliente-SWG

O Cliente-SWG é a parte da Central de Software executada no receptor, responsável pela navegação entre os *widgets* disponibilizados pelo serviço, e por sua execução local. Embora o cliente seja o responsável pelo acesso ao serviço, ele deve ser pequeno, para que possa ser disponibilizado através do Carrossel de Dados, e leve, para que possa ser executado nos variados tipos de hardware para receptores, com limitações de processamento e memória.

Internamente, o Cliente-SWG é composto por 3 camadas principais, a Camada de Segurança, a Camanda de Comunicação, e o Núcleo. A Camada de Segurança, no topo

da camada de comunicação, intercepta toda a comunicação com os demais componentes podendo auutenticá-los, assinando, criptografando e decriptografando as mensagem. A camada de comunicação transforma as interações do usuário em mensagens e as mensagens recebidas em ações de resposta. O Núcleo é quem interage com o Ginga, trocando eventos com o mesmo e interagindo com vídeo, áudio, e exibindo informações na tela do televisor. As duas últimas camadas consomem dados de configuração transmitidos via broadcast juntamente com o próprio Cliente-SWG e que não podem ser alterados, restringindo o seu comportamento ao definido pelo Provedor de Serviço, como ilustra a figura 4.2.

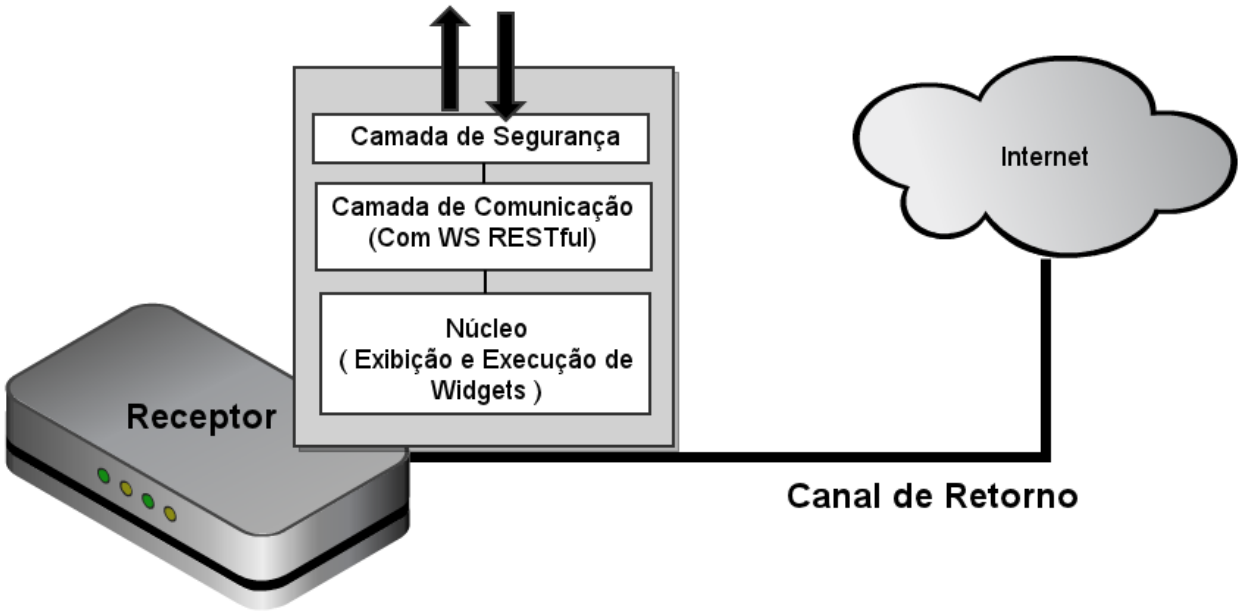

Figura 4.2: Cliente-SWG em camadas

Durante todo o seu tempo de vida, um Cliente-SWG se comunica exclusivamente com um Servidor-SWG, em geral propriedade do mesmo Provedor de Serviço. Esta é uma das características chave do SWG, pois dá ao Provedor de Serviço controle sobre o serviço oferecido a todos os usuários aos quais disponibiliza seu Cliente-SWG.

4.2 Servidor-SWG

O Servido-SWG é o componente central desta arquitetura, respondendo como *beckend* da Central de Software, mantendo atualizados os metadados dos *widgets* disponíveis nos repositórios, e processando as regras para sua exibição no cliente.

O Servidor-SWG é composto por uma camada de acesso ao Banco de Dados, abstraindo o acesso a ele, uma camada de regra de negócio, onde o Provedor de Serviço pode personalizar o comportamento de sua Central de Software, duas camadas para abstração da comunicação entre o Servidor-SWG e os demais componentes, de acordo com o protocolo utilizado e uma camada de segurança, que intercepta toda a comunicação e pode autenticá-la, assiná-la e criptografá-la. A figura 4.3 ilustra as camadas do Servidor-SWG
.

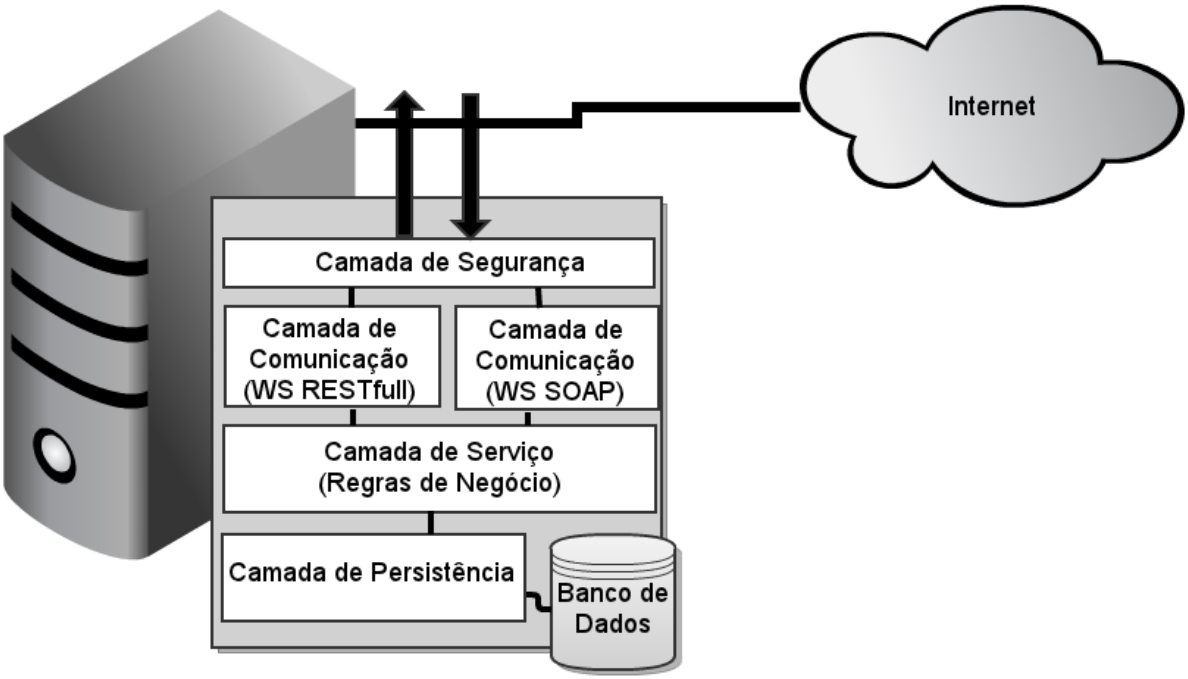

Figura 4.3: Servidor-SWG em camadas

Devido a essa posição central, o Servidor-SWG recebe requisições do Cliente-SWG derivadas de interações do usuário com a Central de Software, e devolve respostas com dados pré-formatados, prontos para exibição. Isso é possível pois a cada requisição o Cliente-SWG envia um cabeçalho de metadados como versão, idioma preferencial, código

postal, modelo de receptor, entre outros dados através dos quais o Servidor-SWG pode personalizar a resposta gerada.

Além de facilitar que correções e melhorias possam ser feitas no Servidor-SWG, sem que seja necessário atualizar os Clientes-SWG já disponibilizados, essa centralização das regras de negócio permite que o Provedor de Conteúdo tenha um rígido controle do serviço oferecido. Na seção 4.4 Comunicação temos tem-se um exemplo de requisição e resposta na comunicação entre Cliente-SWG e Servidor-SWG.

Embora um Cliente-SWG se comunique com apenas um Servidor-SWG, o número de comunicações entre Servidores-SWG e os demais componentes é livre, cabendo ao Servidor-SWG atender aos múltiplos clientes e a responsabilidade de gerenciar quais repositórios estarão disponíveis aos esses clientes, e de manter atualizados os dados locais de *widgets* à medida que eles são adicionados, alterados e excluídos do repositório. Na seção 4.4 há detalhes do processo de atualização desses metadados.

4.3 Repositório-SWG

O Repositório-SWG é o componente responsável por oferecer informações sobre os *widgets* aos Servidores-SWG, e disponibilizá-los para *download* e execução ao Cliente-SWG.

Sua estrutura interna é dividida em camadas, conforme ilustrado pela figura 4.4. A Camada de Persistência faz acesso ao Banco de Dados onde estão os metadados dos *widgets* e ao sistema de arquivos quando um Cliente-SWG tenta realizar um *download*. A Camada de Serviço tem as regras de negócio dos processos de acesso às inforações e *widgets*. Duas camadas de comunicação são responsáveis pelo envio e recebimento de mensagens aos demais componentes, cada uma responsável por um protocolo. No topo das camadas de comunicação a Camada de Segurança é responsável por interceptar todas as mensagens, podendo autenticá-las, assiná-las, e criptografá-las. .

Um repositório pode possuir apenas um *widget*, uma pequena coleção deles relacionados a uma campanha de publicidade, ou mesmo *widgets* de toda a programação de uma emissora. *Widgets* podem ser adicionadas e removidas de um repositório e seus metadados como versão, descrição e avaliação alterados. Cabe ao repositório notificar aos Servidores-SWG interessados que essas alterações ocorreram.

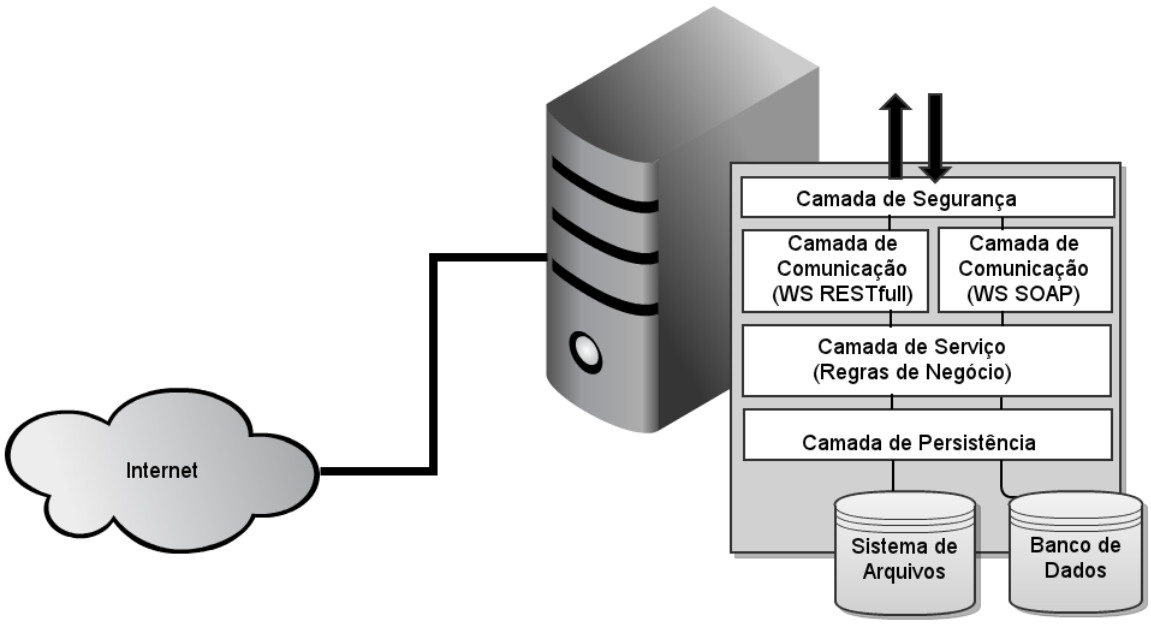

Figura 4.4: Repositório-SWG em camadas

Essa notificação se baseia no Padrão de Projetos Observer[1], adaptado-o para uma arquitetura distribuída. Este padrão é caracterizado por uma relação de um para muitos, estabelecendo dois papéis. No primeiro o Sujeito possui uma lista de interessados em suas atualizações e cabe a ele informá-los quando elas ocorrerem, e no segundo o Observador possui uma interface para receber as atualizações nas quais está interessado. No SWG os Observadores são Servidores-SWG e os Sujeitos são Repositórios-SWG, cabendo ao Servidor-SWG dar o primeiro passo e registrar-se no Repositório-SWG, que caso aceite o novo Observador irá notificá-lo sempre que uma atualização ocorrer.

Publicar apenas as alterações nos Servidores-SWG visa diminuir o tempo de atendimento aos clientes, evitando um acesso ao Repositório-SWG a cada requisição dos Clientes-SWG. Mantendo todos os dados no Servidor-SWG, o SWG reduz o custo da infraestrutura necessária para manter um Repositório-SWG, facilitando que produtores de conteúdo independentes disponibilizem seus *widgets*. O acesso direto do Cliente-SWG ao Repositório-SWG acontece apenas no momento do *download* do *widget*, muito menos frequente que o acesso aos dados dos *widgets* oferecidos.

[1]http://c2.com/cgi/wiki?ObserverPattern

4.4 Comunicação

Toda a comunicação entre os componentes do SWG é feita via Serviço Web (WS) utilizando dois protocolos diferentes, WS RESTful na comunicação entre o Cliente-SWG e os demais componentes, e WS SOAP[2] na comunicação entre Servidor-SWG e Repositório-SWG conforme ilustrado na figura 4.5.

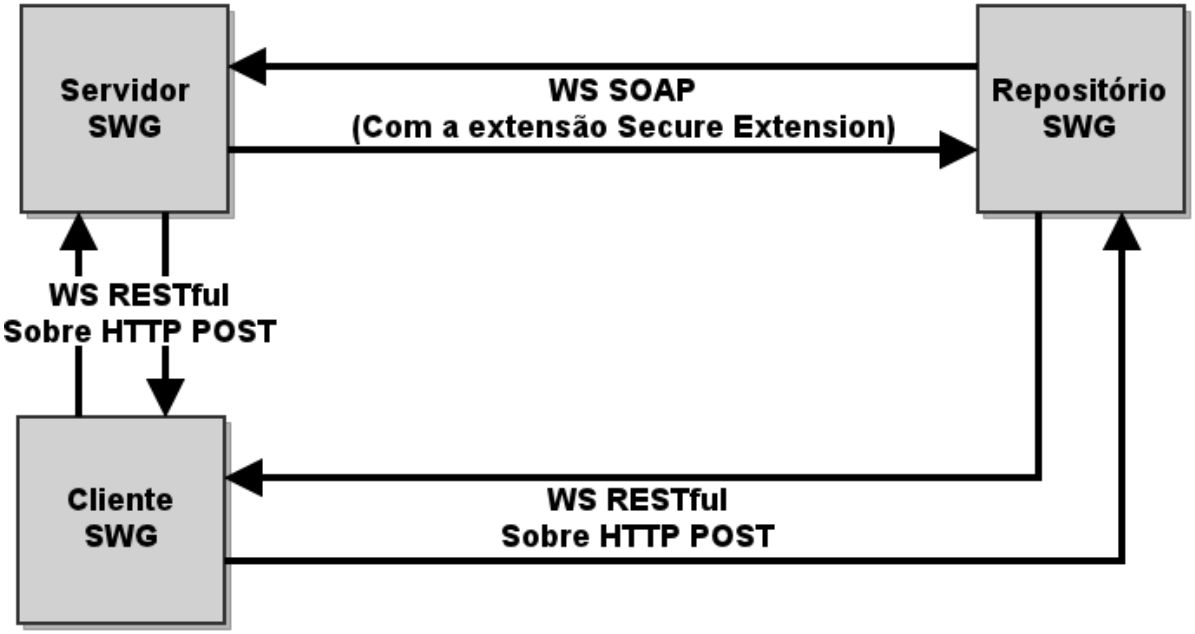

Figura 4.5: Comunicação entre os componentes do SWG

WS RESTful são construídos sobre os métodos existentes no protocolo HTTP, transferindo dados em formato XML[3] sem estabelecer conexão e, por tanto, sem manter estado. Esta característica oferece vantagens como simplicidade e escalabilidade. Simplicidade por não se tratar de um protocolo com regras rígidas para a comunicação, e escalabilidade por retirar do servidor várias custos relacionados à criação de 'sessões' para armazenar e consultar dados dos múltiplos usuários conectados ao serviço, liberando recursos e diminuindo o tempo de resposta, permitindo maior número clientes atendidos simultaneamente, além de ser um protocolo de acesso a WS particularmente interessante em dispositivos móveis(Hatem, 2010) ou com recursos limitados, como podem ser os re-

[2]http://www.w3.org/TR/soap/
[3]http://www.w3.org/XML/

ceptores.

No arquivo XML exemplificado a seguir há um exemplo de requisição utilizando WS RESTful, onde um Cliente-SWG solicita a um Servidor-SWG a segunda página da listagem de *widgets* da categoria Documentários utilizando o método HTTP POST.

```
1
2  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
3  <request>
4       <head>
5            <deviceid>9BWZZZ377ST000011</deviceid>
6            <language>pt-BR</language>
7            <timezone>GMT -3</timezone>
8            <swgversion>1</swgversion>
9            <zipcode>20912</zipcode>
10      </head>
11      <page>2</page>
12      <list>apps</list>
13      <categoryid>documentarios</categoryid>
14 </request>
```

O Servidor-SWG então processa a requisição e retorna *widgets* com base nas informações recebidas no cabeçalho da requisição, podendo por exemplo restringir aplicações cujos requisitos não sejam suportados pelo receptor, estejam em um idioma diferente do utilizado pelo usuário ou seja específico para outra região. No XML abaixo exemplificamos uma resposta para a requisição anterior.

```
1
2  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
3  <response>
4       <head version="1.0.0">
5            <name>Serviço de Widgets Ginga</name>
6            <provider>Lapic UFJF</provider>
7            <contact>welington.veiga@gmail.com</contact>
8            <language>pt_BR</language>
9            <total>118</total>
10      </head>
11      <category id="documentarios" logosrc="static/noticias.png">
12           <title>Notícias</title>
13           <description>Notícias na TV, informe-se!</description>
14           <numitens>3</numitens>
15           </category>
16      <apps total="9">
17           <app id="jornal-ufjf-1.0.2-BETA" version="1.0.2 BETA">
18           <name>Jornal da UFJF</name>
19           <description>
20                Aqui voce encontra as novidades da UFJF: noticias,
```

```
                    palestras,
21                calendário e muito mais.
22            </description>
23            <author href="http://lapic.ufjf.br"
24                     email="lapic@ufjf.br"
25                    >
26                  Welington Veiga
27            </author>
28            <license>free</license>
29            <language>pt_BR</language>
30            <icon src="static/app/icon/ufjfnews.png"/>
31            <content src="static/app/ufjfnews/install.xml"
32            type="application/x-ginga-ncl"/>
33            <rating>2</rating>
34            <downloads>54</downloads>
35            <added_at>GMT-2 2011-00-15 21:19</added_at>
36        </app>
37            ? <!-- Mais 8 aplicações aqui, completando 1 pagina-->
38        </apps>
39  </response>
```

Entre o Servidor-SWG e o Repositório-SWG a comunicação é feita usando WS SOAP, protocolo para WS que permite que informações sejam trocadas entre sistemas com ambientes computacionais heterogêneos de forma transparente, baseada em transferência de arquivos XML(Hatem, 2010). WS SOAP oferecem maior formalismo na comunicação entre serviços que os WS RESTful(Geeknizer, 2009).

Esse formalismo trazido pela utilização do SOAP foi introduzido entre Servidor-SWG e Repositório-SWG porque há uma dependência maior da autenticidade dos componentes envolvidos nessa comunicação, para que Repositório-SWG disponibilize suas aplicações para Servidores-SWG parceiros, e vice versa. A extensão SOAP *Security Extensions*[4] adiciona ao WS SOAP a possibilidade de identificar através de assinaturas digitais a autenticidade das entidades envolvidas. Além disso, diferentemente da comunicação altamente acoplada entre Cliente-SWG e Servidor-SWG distribuídos pelo Provedor de Serviço, a comunicação entre Servidor-SWG e Repositório-SWG é feita entre provedores diferentes, em uma relação de muitos para muitos.

[4]http://www.w3.org/TR/SOAP-dsig/

4.5 Segurança

Embora haja uma camada de segurança em cada componente da arquitetura, esta camada de segurança não é obrigatória no SWG, e não há especificações de como esta camada deva ser implementada; apenas é descrito como uma camada de segurança pode ser adicionada ao serviço, ficando a cargo dos Provedores de Serviço e Provedores de Conteúdo implementá-las.

Um dos principais problemas para a implantação de comunicação segura no Ginga é a falta de APIs para tratar de métodos de criptografia, criando uma dependência do receptor, caso esse serviço seja oferecido por um fabricante específico, ou criando overhead no serviço, caso essa camada seja implementada em uma linguagem de alto nível como as disponibilizadas nos ambientes Ginga-NCl e Ginga-J.

A utilização de um WS RESTful nas comunicações com o Cliente-SWG como alternativa ao WS SOAP, que é reconhecido por sua robustez e segurança, se deve a questões de desempenho e escalabilidade do SWG. No entanto WS RESTful são baseados em HTTP(Renner, 2011) e podem utilizar todas as soluções encontradas para comunicação segura sobre esse protocolo nos navegadores, esbarrando mais uma vez nos custos que estas soluções podem representar se implementadas nos ambientes de alto nível do Ginga.

O Canal de Retorno liga o *middleware* Ginga à uma rede. Essa rede pode ser a Internet, mas esta não é uma restrição, já que o Provedor de Serviço pode oferecer uma rede privada e disponibilizar o SWG dentro dessa infraestrutura, acessando, se necessário, apenas a Repositórios-SWG externos, permitindo que o provedor de serviço crie uma infraestrutura de rede segura, fora da internet.

Novos serviços para prover segurança podem ser adicionados ao SWG utilizando extensões, conforme descrito na Seção 4.6.

4.6 Extensões

Entre os objetivos chave do SWG está a generalidade, no entanto os Provedores de Serviço não podem ficar restritos a um comportamento padronizado de suas Centrais de Software. Para utilizá-las como um serviço lucrativo é preciso que promoções, informações extras de

um programa de vantagens, ou até mesmo um modelo de negócio de loja similar às lojas *Apple App Store* e *Android Market* sejam permitidos, sem quebrar a compatibilidade com repositórios distribuídos por terceiros.

Na arquitetura proposta, é forte o acoplamento entre Servidor-SWG e Cliente-SWG, distribuídos geralmente pelo mesmo Provedor de Serviço, e fraco o acoplamento entre eles e o Repositórios-SWG. Como o Cliente-SWG pode acessar a apenas um Servidor-SWG, que por sua vez são acessados apenas pelos Clientes-SWG distribuídos para acessá-lo, novos recursos podem ser incorporados nessa comunicação sem perder a compatibilidade com os Repositórios-SWG.

Esses recursos devem ser adicionados aos XMLs trocados na comunicação WS RESTful entre Cliente-SWG e Servidor-SWG através de *namespaces* específicos, que podem adicionar informações sem alterar as mensagens obrigatórias, expandindo os serviços que o Servidor-SWG pode oferecer e criando a possibilidade de Provedores de Serviço desenvolverem novos negócios exclusivos sobre a arquitetura.

A restrição nas extensões do SWG se dá justamente na comunicação com o Repositório-SWG. Primeiro porque a arquitetura prevê que um Repositório-SWG distribua aplicações para vários Servidores-SWG, que devem seguir o padrão de comunicação estabelecido, e depois porque é essa parte da arquitetura que garante a distribuição de aplicações no padrão de *widgets*, conforme recomendação para IPTV.

5 Implementação de Referência

Em paralelo à definição de um serviço genérico de distribuição de aplicações dentro do SBTVD, foi desenvolvida uma implementação de referência, como laboratório das soluções adotadas, aproximando a arquitetura desenvolvida à realidade das ferramentas, bibliotecas, e recursos do ambiente de desenvolvimento existente para Televisão Digital.

A arquitetura definida no Capítulo 4 é muito ampla, de modo que apenas um subconjunto da arquitetura foi implementado. Esta implementação é completa o bastante para contemplar cada um dos componentes do SWG, toda a comunicação entre Servidor-SWG e Cliente-SWG, entre Repositório-SWG e Cliente-SWG, e permitir a navegação pela Central de Software via Máquina Virtual Ginga[1], a figura 5.1 mostra as tecnologias utilizadas.

A navegação pela Central de Software disponibilizada pela Implementação de Referência é simples inspirada nas Lojas de Aplicativos *Apple App Store* e *Android Market* e na Central de Software do Ubuntu. A navegação é feita pelo controle remoto, intuitivamente. A figura 5.2 apresenta o Menu Principal, e destaca algumas das soluções de usabilidade encontradas: 1) a barra de navegação que mantém sempre acessível a informação de onde o usuário se encontra. 2) Informações do Provedor de Serviço, 3) ícones que identificam facilmente os botões do controle remoto correspondentes, e 4) a paginação através dos botões direcionais do controle remoto..

Na figura 5.3 temos a tela para navegação pelos *widgets* dentro de uma categoria, identificados por ícones e uma breve descrição. Para direcionar o usuário para os melhores *widgets* exibimos também a avaliações dos outros usuários, indicada por estrelas. Em qualquer momento a partir do menu principal, a tela anterior está sempre disponível pelo clique do botão verde do controle remoto, e caso haja mais de nove *widgets* em uma mesma categoria eles serão paginados, da mesma forma que as categorias, simplificando o acesso pelos números do controle remoto. Ao selecionar um *widget* exibimos detalhes como versão, autor, tamanho, descrição completa e uma captura de tela antes de executá-lo,

[1] http://www.gingancl.or g.br/en/ferramentas

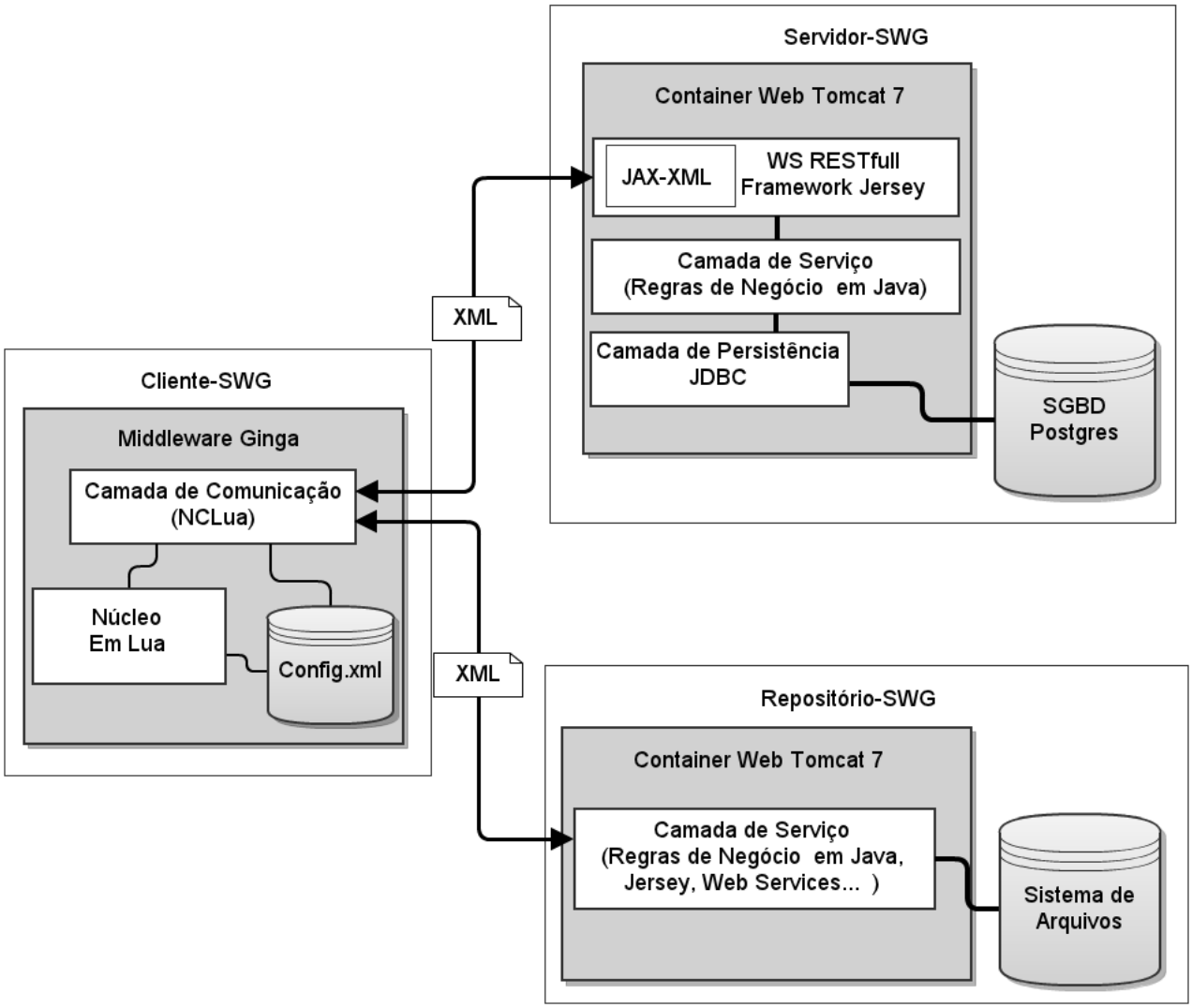

Figura 5.1: Implementação de referência do SWG com destaque para as tecnologias utilizadas

como mostra a figura 5.4.

Internamente, para gerenciar de forma genérica as diferentes visões com múltiplas opções, foi utilizada uma estratégia baseada em eventos, em que cada ação do usuário gera uma série de eventos para os quais podem haver tratadores responsáveis pela resposta; caso não haja nenhum tratador para um evento, nada é feito. Tratadores podem ser adicionados ou removidos da lista de interessados em um evento interno da aplicação do módulo de eventos do NCLua. Os eventos dessa estratégia são diferentes dos eventos gerados pelo módulo NCLua Event[2], sendo estes convertidos naqueles, desacoplando o funcionamento interno do Cliente-SWG dos eventos gerados pelo NClua.

[2]http://www.lua.inf.puc-rio.br/ francisco/nclua/referencia/event.html

Figura 5.2: Menu principal da Central de Software da Implementação de Referência

O Servidor-SWG foi desenvolvido na Linguagem Java[3], executando sobre o container web Tomcat 7[4]. Embora não seja livre, a Linguagem Java goza de uma implementação livre de sua Máquina Virtual[5] e vários projetos livres disponíveis em seu ecossistema, como o próprio Tomcat mantido pela Fundação Apache[6], e a implementação da especificação JSR-311[7] que utilizamos para facilitar a construção da camada de WS RESTful, o Jersey[8]. Para persistência dos metadados dos *widgets* disponíveis, utilizamos o Sistema de Gerenciamento de Banco de Dados (SGBD) Postgres[9], um projeto livre, conhecido por sua robustez e escalabilidade.

Excetuando-se a camada WS SOAP, todas as camadas do Servidor-SWG foram definidas através de interfaces que identificam os serviços que oferecem. Utilizamos o Padrão de Projetos *Factory*[10] estático(Bloch, 2008) para construir as implementações dessas camadas, de modo que novas implementações possam ser criadas sem necessidade de alterações nas demais camadas, permitindo que Provedores de Conteúdo adicionem suas

[3]http://www.java.com/
[4]http://tomcat.apache.org/
[5]http://openjdk.java.net
[6]http://www.apache.org/
[7]http://jcp.org/en/jsr/detail?id=311
[8]http://jersey.java.net/
[9]http://www.postgresql.org/
[10]http://c2.com/cgi/wiki?FactoryMethodPattern

Figura 5.3: Navegação por *widgets* de uma categoria

próprias implementações de uma delas, como a de Persistência, por exemplo, e continue utilizando as implementações de referência das demais camadas.

O Repositório-SWG na implementação de referência apenas disponibiliza a aplicação para o Cliente-SWG através de um WS RESTful, já que ainda não há comunicação com o Servidor-SWG. Utilizamos as mesmas tecnologias que no Servidor-SWG, Java, sobre o Apache Tomcat 7, utilizando Jersey para prover o WS. No entanto, não possui uma implementação em camadas tão desacoplada quanto no Servidor-SWG, também não utiliza SGBD, obtendo a aplicação diretamente do Sistema de Arquivos.

Na implementação de referência a compatibilidade com a recomendação de empacotamento ITU-T IPTV-GSI é mantida, sendo utilizado um arquivo descritor com informações da aplicação, uma estrutura de diretórios similar à recomendada mas não é utilizado formato ZIP para o empacotamento, por não ser um recurso padrão da especificação do Ginga. Contornamos esta limitação acrescentando ao descritor todos os arquivos que compõem o *widget*, incluído arquivos NCL, Lua e multimídia em seus respectivos diretórios.

Figura 5.4: Detalhes de um *widget* disponível para execução

6 Resultados Obtidos

A implementação de referência demonstra que um serviço de aplicações sobre o SBTVD pode ser genérico, com uma comunicação bem definida entre componentes de fornecedores diferentes, flexível para permitir controle dos mesmos sobre o serviço oferecido, escalável minimizando o custo da comunicação entre os componentes e transparente na execução de aplicações, utilizando um empacotamento padrão.

O modelo de empacotamento padrão utilizado, a recomendação de *widgets* para IPTV, permite que Provedores de Conteúdo possam gerar apenas um empacotamento e disponibilizá-lo em vários serviços, facilitando novas iniciativas para a criação de aplicações graças à segurança de compatibilidade com serviços de Centrais de Software para SBTVD e até mesmo para outros Sistemas de Televisão Digital compatíveis com o sistema.

Através da implementação de referência do SWG demonstramos ainda a viabilidade da parte crítica do serviço, que incluí o acesso à Central de Software, a obtenção do *widget* em seu repositório, e sua execução. Tudo utilizando apenas os recursos da implementação de referência do middleware Ginga, requisito para que serviço possa ser disponibilizado totalmente independente dos receptores utilizados pelos clientes.

7 Trabalhos Futuros

Este trabalho define uma arquitetura genérica para facilitar o acesso a aplicativos no SBTVD. Após sua conclusão, novas metas podem ser estabelecidas para nos aproximar desse objetivo. A primeira delas é desenvolver uma implementação do Repositório-SWG para acessar, empacotar, e disponibilizar as aplicações para SBTVD existentes no Clube-NCL[1], repositório colaborativo de aplicações NCL.

Outro trabalho importante é a inclusão de uma solução para o problema de segurança na comunicação entre o Cliente-SWG e os demais componentes ao SWG, que hoje não oferece nenhuma garantia de que quem envia as respostas é confiável e está entregando o recurso solicitado, sem necessidade de extensões proprietárias ou dependente de implementações de fabricantes específicos.

Estabelecida uma forma confiável de identificação dos componentes do serviço, outra linha importante para o prosseguimento do trabalho é a extensão do SWG para incorporar um modelo de Loja de Aplicativos, similar ao *Android Market* ou a *Apple App Store*, criando um mercado para *widgets* e serviços vendidos e consumidos diretamente pela Televisão Digital.

Outro ponto interessante é a criação de um Serviço de Aplicações que reaproveite os repositórios do SWG para prover aplicações para IPTV ou qualquer outro Sistema de Televisão Digital capaz de executar aplicações que seguem a recomendação de *widgets* para aquela plataforma, endossando a padronização do empacotamento de *widgets* para Televisão Digital.

[1] http://clube.ncl.org.br/

8 Conclusão

Entre as principais metas na definição de um Serviço de Aplicações para SBTVD está aumentar a disponibilidade de aplicações, reconhecendo tratar-se de uma plataforma de execução comparável aos dispositivos móveis que gozam de grande interesse de desenvolvedores e usuários pela facilidade de acesso e grande disponibilidade de aplicações existente em Centrais de Software para essas plataformas.

A Arquitetura proposta neste trabalho é uma forma de facilitar o acesso e a distribuição de aplicações para Televisão Digital adaptando o modelo de Central de Software existente nessas outras plataformas para o SBTVD, estendendo o acesso à aplicações para além do Carrossel de Dados, onde é decisivo que um serviço de aplicações esteja alinhado a um modelo de negócio que atraia interesse de usuários, provedores de conteúdo, e provedores de serviço, que podem usar sua Central de Software como um produto competitivo ou um canal para distribuição de conteúdo patrocinado.

Por fim, ressaltamos que os repositórios podem tanto oferecer conteúdos publicitários, esportivos, e informativos quanto de interesse público, com conteúdo educativo como vídeo-aulas, atividades, testes, material didático, ou voltados para programas de saúde do Governo como informativos regionalizados com datas de vacinação, informações preventivas, palestras em vídeo, entre uma infinidade de outras possibilidades. Através das Centrais de Software, esses aplicativos estarão disponíveis não apenas durante um comercial ou um programa específico, mas sempre que o usuário tiver um serviço de widgets acessível em sua televisão.

Referências Bibliográficas

Associação Brasileira de Normas Técnicas. **Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 1: Codificação de dados**. ABNT, 2008.

Almeida, W. F.; de Carvalho, J. M. T-commerce e modelo de negócio: o comércio eletrônico televisivo nos canais do sistema brasileiro do agronegócio . **CELACOM - XIV Colóquio Internacional da Escola Latino-Americana de Comunicação**, 2010.

Barrere, E.; e Silva, C. K. P. Acesso ao Moodle via Aplicação para Tv Digital . **Moodle-Moot Brasil**, 2010.

Becker, V.; Filho, G. H. H. ; Montez, C. Inclusão Digital via Serviços de Saúde para o Sistema Brasileiro de TV Digital . **XXXII CONFERENCIA LATINOAMERICANA DE INFORMÁTICA CLEI**, 2006.

Bloch, J. **Creating and Destroying Java Objects: Part 1**. Disponível em: http://drdobbs.com/java/208403883, Último acesso em Novembro 2011, 2008.

Borges, R. C. **Uma Arquitetura para Descoberta de Serviços em Ambientes de TV Digital Interativa** . 2007. Dissertação de Mestrado - UFG.

Castro, C.; Tome, T. ; Filho, A. B. **Mídias digitais - convergencia tecnológica e inclusão social**. 1. ed., Paulinas, 2005, 368p.

Damasceno, J. R. **Middleware Ginga**. Disponível em: http://drdobbs.com/java/208403883, Último acesso em Novembro 2011, 2008.

Fernandes, J.; Lemos, G. ; Silveira, G. E. Introdução à Televisão Digital Interativa: Arquitetura, Protocolos, Padrões e Práticas. **XXIII Jornada de Atualização em Informática do XXIV Congresso da Sociedade Brasileira de Computação JAI-SBC**, 2004.

Geeknizer. **REST vs. SOAP - The Right WebService**. Disponível em: http://geeknizer.com/rest-vs-soap-using-http-choosing-the-right-webservice-protocol/, Último acesso em Novembro 2011, 2009.

Ghisi, B. C.; Lopes, G. F. ; Siqueira, F. Integração de Aplicações para TV Digital Interativa com Redes Sociais. **WebMedia 2010 - WebMedia - Simpósio Brasileiro de Sistemas Multimídia - Workshop de TV DIgital Interativa**, 2010.

Hamad, H.; Saad, M. ; Abed, R. Performance Evaluation of RESTful Web Services for Mobile Devices. **International Arab Journal of e-Technolog**, v.1, n.3, 2010.

Holzer, A.; Ondrus, J. Trends In Mobile Development. **Mobile Wireless Middleware, Operating Systems, and Applications - Workshops**, v.12, p. 55–60, 2009.

Recommendation ITU-T J.200. **Worldwide common core ? Application environment for digital interactive television services**. ITU-T, 2001.

Recommendation ITU-T H.760. **Overview of Multimedia Application Frameworks for IPTV Services**. ITU-T, 2009.

Recommendation ITU-T H.761. **Nested Context Language (NCL) and Ginga-NCL for IPTV Services**. ITU-T, 2009.

Idgnow. **Android Market supera App Store, da Apple, em número de downloads**. Disponível em: http://idgnow.uol.com.br/mobilidade/2011/10/25/android-market-supera-app-store-da-apple-em-numero-de-downloads/, Último acesso em Novembro 2011, 2011.

Kim, H. J.; Kim, I. ; Lee, H. G. PThe Success Factors for App Store-Like Platform Businesses from the Perspective of Third-Party Developers: An Empirical Study Based on A Dual Model Framework. **PACIS 2010 Proceedings**, 2010.

Mcdaniel, P.; Enck, W. Not So Great Expectations: Why Application Markets Haven't Failed Security. **Security & Privacy - IEEE 2010**, v.8, p. 76–78, 2010.

Mendes, C. O. S. **Arquitetura e Serviços para EAD no SBTVD com Escalabilidade**. PESC/COPPE/UFRJ, Rio de Janeiro, Agosto 2010. Tese de Doutorado - UFRJ.

Mikoczy, E.; Sivchenko, D.; Bangnan, X. ; Moreno, J. I. IPTV Systems, Standards and Architectures: Part II - IPTV Services over IMS: Architecture and Standardization. **Communications Magazine, IEEE**, v.60, p. 128–135, 2008.

Moreno, M. F.; Batista, C. ; Soares, L. F. G. NCL and ITU-T Standardization Effort on Multimedia Application Frameworks for IPTV. **WebMedia 2010 - WebMedia - Simpósio Brasileiro de Sistemas Multimídia - Workshop de TV DIgital Interativa**, 2010.

Pessoa, B. J.; Souza, G. L. S. ; Cabral, L. A. Metaheurística Aplicada à Geração de Carrossel no Sistema Brasileiro de TV Digital. **WebMedia 2008**, 2008.

Reis, R. L. B.; Teixeira., M. M. Uma Arquitetura para Interoperablidade entre Aplicativos NCLua e Serviços da Internet. **O Simpósio Brasileiro de Sistemas Multimídia e Web - WebMedia 2011**, 2011.

Renner, A. **RESTful Web Services**. Disponível em: http://faculty.fortlewis.edu/ADAMS_E/CLASSES/CS496SENIORSEMINAR/CS496W08/StudentPapers/Aaron_Renner_-_RESTful_Web_Services.pdf, Último acesso em Novembro 2011, 2011.

Savarese, E. **Digital Television in the Environment of Convergence**. Disponível em: http://www.eett.gr/conference2009/pdf/Savarese.pdf, Último acesso em Novembro 2011, 2009).

Shin, S. Mu: channel ui to optimize the widget control in internet tv. **ACM International Conference Proceeding Series Vol 291**, p. 41–44, 2008.

da Silva, E.; Nunes, V. B. Uso da TV Digital na Educação a Distância. **WIE - Workshop de Informática na Escola, 2010, Belo Horizonte. XXX Congresso da SBC Computação verde: desafios científicos e tecnológicos**, p. 1167–1176, 2010.

Wire, B. **Apple's App Store Downloads Top 15 Billion**. Disponível em: http://www.businesswire.com/news/home/20110707005466/en, Último acesso em Novembro 2011, 2011.