Universidade Federal de Pernambuco

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

2010.1

# Uma Comparação Entre o Desenvolvimento de Aplicações Ginga-J e OpenTV®

## Trabalho de Graduação



| | | |
|---|---|---|
| **Aluno** | Victor Hazin da Rocha | {vhr@cin.ufpe.br} |
| **Orientador** | Carlos André Guimarães Ferraz | {cagf@cin.ufpe.br} |
| **Co-Orientador** | Felipe Silva Ferraz | {fsf3@cin.ufpe.br} |



9 de Julho de 2010

Universidade Federal de Pernambuco

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

2010.1

# Uma Comparação Entre o Desenvolvimento de Aplicações Ginga-J e OpenTV®

## Trabalho de Graduação

*Trabalho de graduação apresentado no Centro de Informática da Universidade Federal de Pernambuco por Victor Hazin da Rocha, orientado por Carlos A. G. Ferraz, como requisito para a obtenção do Grau de Bacharel em Ciência da Computação.*

| | | |
|---|---|---|
| **Orientador** | Carlos André Guimarães Ferraz | {cagf @cin.ufpe.br} |
| **Co-Orientador** | Felipe Silva Ferraz | {fsf 3@cin.ufpe.br} |

9 de Julho de 2010

**Folha de Aprovação**

# Uma Comparação Entre o Desenvolvimento de Aplicações Ginga-J e OpenTV®

Victor Hazin da Rocha

Aprovado em 9 de julho para ser apresentado.

Ciente:

_____

Profº. Carlos André Guimarães Ferraz, PhD – UFPE (Orientador)

**“**É preciso força pra sonhar e perceber que a estrada vai além do que se vê.**”**

**Marcelo Camelo**

# Agradecimentos

Não seria possível começar os agradecimentos de outra forma que não fosse agradecendo a Deus por ter conseguido concluir mais uma etapa da minha vida. Afinal ele se mostra presente nela desde o meu primeiro dia de vida até hoje sempre, me dando forças para superar as dificuldades e atingir meus objetivos. Gostaria de agradecer a Deus também por durante esse tempo ter colocado tantas pessoas maravilhosas na minha vida, pessoas essas que foram fundamentais me dando suporte durante essa caminhada.

Espero que entendam que é impossível citar o nome de todas as pessoas que foram importantes para mim durante essa caminhada, portanto se seu nome não estiver aqui não quer dizer que não foi importante para mim, nem que não me ajudou. Outra informação extremamente importante é o fato de que a ordem dos agradecimentos não implica em importância.

Em especial gostaria de agradecer a toda minha família, em especial a minha mãe Carla, que sem o apoio e dedicação dela tantas conquistas na minha vida nunca seriam possíveis. Não posso deixar de dar um muito obrigado especial também a Roberto que desde que entrou na minha vida tem sido muito mais que um pai. E jamais deixaria de citar meus amados irmãos Lucas e Vinícius, que apesar de todas as interrupções feitas enquanto eu escrevia esse trabalho contribuíram sempre trazendo alegria com suas presenças. É impossível não citar minha avó Júlia e meus avôs de coração Fernando e Lúcia, sempre incentivando meus estudos. Infelizmente não dar pra citar todos os familiares que me ajudaram e me apoiaram, mas saibam que todos foram muito importantes.

Gostaria de agradecer a meus avôs Elias e Rocha, minha avó Alba e meu pai Inácio que já não se encontram mais aqui, mas que eu sei que onde quer que estejam estão olhando por mim e felizes com essa conquista.

Gostaria também de agradecer aos meus orientadores o professor Carlos Ferraz e Felipe Ferraz, sempre disponíveis para me tirar minhas dúvidas e fazer as revisões desse trabalho. Não poderia deixar de agradecer aos meus orientadores de monitoria durante todos esses anos: a professora Renata e o professor Fernando Fonseca. Pela ajuda no desenvolvimento desse trabalho devo muitos

agradecimentos a Artur Tenório, Jancleidsson e toda equipe de TV Digital do C.E.S.A.R.

Não tem como não agradecer ao iTeam (Luiz, Marcio, Petrônio e Thiago), muito mais que parceiros de projetos e sim verdadeiros amigos. Não tem como não lembrar as nossas eternas e intermináveis discussões no almoço, contando com o reforço do membro honorário Hugo. Impossível esquecer também o CInBolando (o time com o "i" vermelho, o Campeão), todos que fizeram parte da monitoria de estatística comigo e Natália, sim, por incrível que pareça eu fiz uma amiga de verdade no CIn.

Aos sempre amigos Amanda Gondim, Amanda Souza, Gabi, Júlio, Manu, Mariana, Mirella, Newton, Rafael e Ramon não quero apenas agradecer, mas sim pedir desculpa por tantas ausências durante esses anos, sempre causadas pelos intermináveis projetos e provas. Obrigado de verdade por ter vibrado com cada projeto que eu entregava mesmo sem entender muito bem o que eu falava. Eu sempre serei grato a vocês pela amizade verdadeira.

Se mesmo depois de todos estes nomes você não foi citado e se sente injustiçado, só me resta pedir desculpas. E sinceramente eu espero que saiba que o verdadeiro agradecimento não está em pouco mais de uma página escrita, mas sim no sentimento de gratidão que eu carrego comigo que é muito maior que essas poucas palavras.

A todos vocês, muito obrigado!

# Resumo

Por ter sido lançado oficialmente muito recentemente a literatura atual é muito carente em relação à comparação do padrão brasileiro para TV Digital Interativa com outros que já existam no mercado. Este trabalho compara o desenvolvimento de aplicações de televisão digital usando o padrão aberto brasileiro, Ginga-J, e o padrão fechado OpenTV®. Para atingir esse objetivo, uma aplicação (Tabela Dinâmica da Copa do Mundo) foi desenvolvida usando ambas as plataformas. Os resultados obtidos são detalhados nesse artigo.

**Palavras-chave**: Televisão Digital, Ginga, OpenTV, SBTVD, Ginga-J.

# Índice

# Índice de Figuras

# Índice de Tabelas

# 1. Introdução

Segundo pesquisas do IBGE o percentual de domicílios com aparelhos de televisão cresce desde 1992, ano em que a primeira pesquisa foi feita. O percentual de domicílios com pelo menos um aparelho de televisão analógico deu um salto de 74% em 1992 para 94,8% em 2007[1]. Já no dia 2 de dezembro de 2007 foram iniciadas as primeiras transmissões de TV Digital no Brasil, na cidade de São Paulo [2].A Figura 1 mostra o cronograma de implantação.

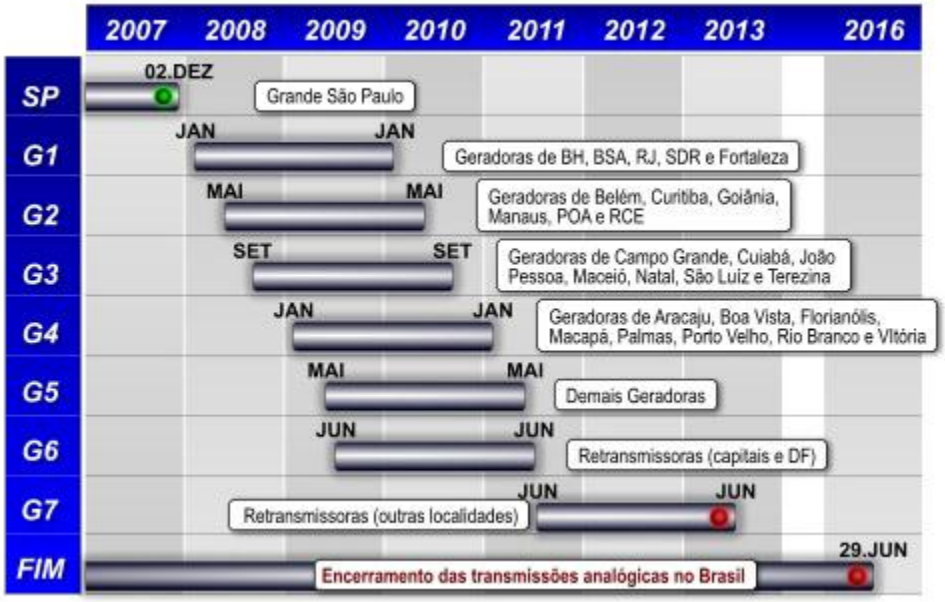


**Figura 1 - Cronograma de implantação da TV Digital no Brasil**
**Fonte: http://dtv.org.br/materias.asp?menuid=3&id=11**

Apesar de alguns atrasos no cumprimento do cronograma, o sinal digital já está presente em quase todas as capitais do Sul, Sudeste e Centro-Oeste, além de algumas cidades do interior paulista e capitais do Norte-Nordeste, como Recife, João Pessoa, Fortaleza, Aracaju e Teresina [2].

O processo de digitalização do sinal aberto brasileiro teve inicio recentemente. Apesar disso já é grande a quantidade de informações que circulam nos meios de comunicação a respeito da TV Digital. A existência dessas informações são decorrentes da digitalização do sistema de TV já ter sido bastante explorado por empresas de televisão por assinatura.

Com a recente digitalização do sinal brasileiro é importante que existam pesquisas voltadas para o desenvolvimento de aplicações para TV digital, focados no *middleware*, que é um programa de computador que faz a mediação entre outros *softwares*. Contudo, a literatura ainda é muito carente em relação à comparação do padrão brasileiro com outros já consolidados. E é essa lacuna que esse trabalho visa ajudar a preencher.

O principal objetivo do trabalho é realizar uma comparação focada no desenvolvimento de uma aplicação utilizando o *middleware* OpenTV [3] e uma utilizando Ginga-J[4]. É importante perceber que não faz parte do escopo dizer qual das plataformas é melhor, mas apenas ressaltar as possíveis semelhanças e diferenças entre elas. Para uma melhor compreensão e organização, essa pesquisa foi subdividida nas seguintes partes: introdução, referencial teórico, estudo de caso, desenvolvimento e resultados e conclusão.

# 2. Referencial Teórico

O processo de digitalização do sinal de televisão trás junto com a melhoria na qualidade de imagem um novo modo de assistir televisão, onde o usuário deixará de apenas receber o sinal e passará interagir com aplicações transmitidas e/ou enviar informações de seu interesse como e-mail, requisição a uma pagina web, responder a enquetes, entre várias novas possibilidades que começam a surgir. Ao tornar possível a interação do telespectador com a emissora e/ou com outros telespectadores, a TV Digital Interativa (TVDI) atende a uma necessidade inerente ao ser humano, que é a de participar, se pronunciar e se sentir mais inserido em seu contexto social [5].

O mercado de TV Digital Brasileiro inicialmente foi controlado pelos padrões fechados, as chamadas TVs por assinatura, tendo o OpenTV como seu principal representante, já que este detém mais de 70% deste mercado [6]. Porém o Brasil decidiu criar o *middleware* Ginga [4], estabelecendo um novo padrão aberto e livre de *royalties*. O Sistema Brasileiro de TV Digital, também chamado de *Integrated Services Digital Broadcasting-Terrestrial* (ISDB-TB), já foi adotado como padrão em outros países como Argentina, Chile, Peru, Equador, Venezuela e Costa Rica [2], além de outros que ainda estão em fase de estudo para sua possível adoção ou não.

Nas próximas seções serão descritas com mais detalhes as duas plataformas que serão comparadas nesse trabalho.

## 2.1. OpenTV®

OpenTV é resultado da cooperação entre a *Thomson Multimedia* e a *Sun Microsystems*, que em 1994 se juntaram para desenvolver uma solução para televisão digital. Em 1996 essa união criou uma empresa à parte chamada *Thomson Sun Interactive* LLC e lançou a primeira versão do *middleware* OpenTV. Um ano depois a companhia foi rebatizada para *OpenTV Incorporated*[3].

O *middleware* OpenTV é um ambiente operacional para TV Digital interativa, que roda aplicações num receptor, um pequeno computador com CPU, memória, um pequeno espaço para armazenar as aplicações e portas de entrada/saída. O receptor recebe as aplicações, o áudio e o vídeo de um transmissor e deve processar essas informações de modo a enviar para televisão as informações corretas a serem exibidas na mesma.

O processo de transmissão da aplicação para o receptor começa com a organização do áudio, do vídeo e dos dados para a difusão. Os sinais de áudio e vídeo precisam ser codificados e encapsulados em pacotes de transporte *MPEG2-TS* por um multiplexador, antes de serem transmitidos. Os dados/aplicações também precisam ser inseridos no multiplexador, através de um injetor de dados. Após a multiplexação, o próximo passo é modular o sinal digital para ser difundido pelos meios convencionais. Cabe ao modulador essa tarefa, que gera um sinal em baixa freqüência. Esse sinal precisa ser convertido em um sinal de freqüência maior para poder ser difundido pelos diversos meios. O equipamento responsável por essa conversão é o *UpConverter* [7]. Esse processo é ilustrado na Figura 2.

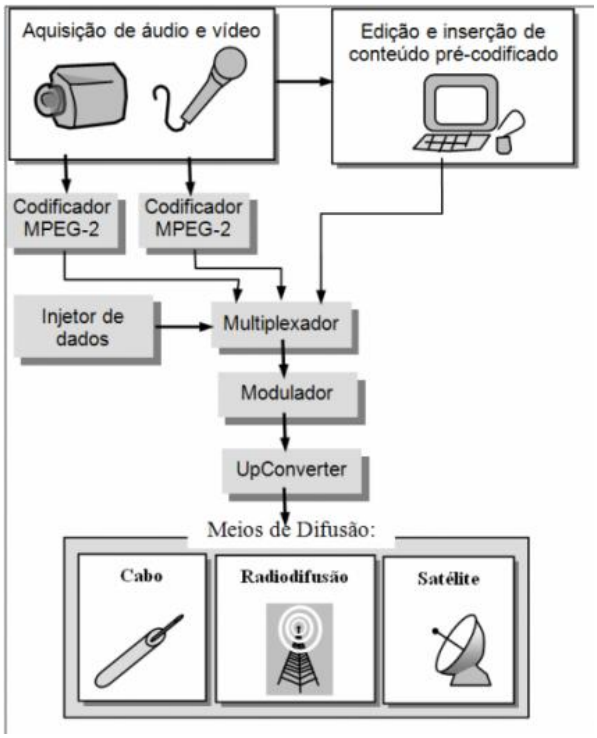


**Figura 2 – Etapas na Geração do Sinal de TV**
**Fonte: http://blog.itvproducoesinterativas.com.br/up/i/it/blog.itvd.com.br/img/.resized_img2.png**

No receptor o primeiro elemento que processa (capta) o sinal difundido é o sintonizador digital. A seguir, o sinal passa pelo demodulador, que extrai o fluxo de transporte *MPEG-2*, passando-o para o demultiplexador, responsável por extrair todos os fluxos elementares. Esses, por sua vez, são então encaminhados para o decodificador, que os converterá para o formato apropriado de exibição utilizado pelo equipamento televisivo. Já as aplicações são executadas pelo processador conforme o estado determinado no injetor de dados, podendo iniciar automaticamente ou ficar a espera de uma ação do telespectador [7]. A Figura 3 esboça esse processo.

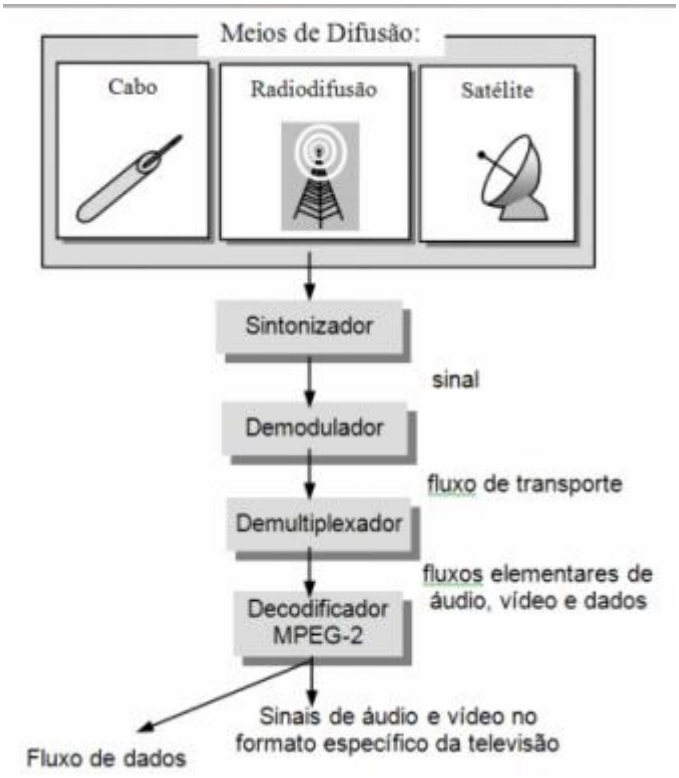


**Figura 3 - Etapas na Recepção do Sinal**
**Fonte: http://blog.itvproducoesinterativas.com.br/up/i/it/blog.itvd.com.br/img/.resized_img3.png**

Os processos para transmissão e recepção que foram explicados até o momento são indiferentes em relação à implementação do *middleware* em si, ou seja, servem tanto para transmissão de aplicações em Ginga quanto OpenTV. A diferenciação do funcionamento dos *middlewares* começa quando a aplicação chega ao processador para ser executada pelo OpenTV. Ele possui uma pilha de *software* característica, semelhante ao apresentado na Figura 4. No *set-top box* a aplicação faz uso de uma API de aplicações. Esta provê uma biblioteca de funções, através da

qual as aplicações obtêm acesso a serviços, como: áudio e vídeo, *display* gráfico e, recepção da interação dos usuários.

Esse acesso às funções é provido pelo interpretador, o qual realiza a tradução da aplicação, perfazendo a conversão das funções das aplicações para chamadas legíveis às camadas de mais baixo nível na pilha de *software*. Para realizar essa conversão, o interpretador utiliza uma série de bibliotecas providas pelo *middleware*. Bibliotecas estas, que podem ser tanto fornecidas pelo OpenTV, como pelo fabricante do terminal de acesso [6].


**Figura 4 - Visão Geral da Arquitetura do OpenTV**

As aplicações devem ser codificadas em *ANSI C*, seguindo o paradigma imperativo e utilizando a OpenTV IDE, que atualmente é baseada no Eclipse (http://www.eclipse.org/). Após compilar o código da aplicação a IDE/compilador gera código objeto; só então é que o código objeto pode ser executado na máquina virtual OpenTV (OpenTV *Virtual Machine* – OVM), que encontra-se instalada nos *set-top box* compatíveis. O processo de execução da aplicação já foi explicado acima.

O acesso à documentação, às APIs e à arquitetura do OpenTV é feito através da compra do produto. A empresa adota uma política de restrição da divulgação de seus componentes, política essa que dificulta a obtenção de artigos relacionados e de dados sobre a plataforma. Contudo, ainda há alguns poucos trabalhos acerca do assunto, como [8]: uma dissertação de mestrado onde é feito um estudo sobre a migração de aplicações de OpenTV para MHP.

## 2.2. Ginga

Ginga® é o nome do *middleware* aberto do Sistema Brasileiro de TV Digital (SBTVD). Ele é constituído por um conjunto de tecnologias padronizadas e inovações brasileiras que o tornam a especificação de *middleware* mais avançada e a melhor solução para os requisitos do país, segundo [4].

O Ginga foi inicialmente desenvolvido por grupos de pesquisa sob a liderança da Pontifícia Universidade Católica do Rio de Janeiro (PUC - Rio) em conjunto com a Universidade Federal da Paraíba (UFPB). O sistema Ginga é subdividido em dois subsistemas principais interligados (Ginga-J e Ginga-NCL), e permite o desenvolvimento de aplicações seguindo dois paradigmas de programação diferentes – procedural/imperativo e declarativo. A utilização de um paradigma ou outro depende das necessidades e funcionalidades de cada aplicação [4].

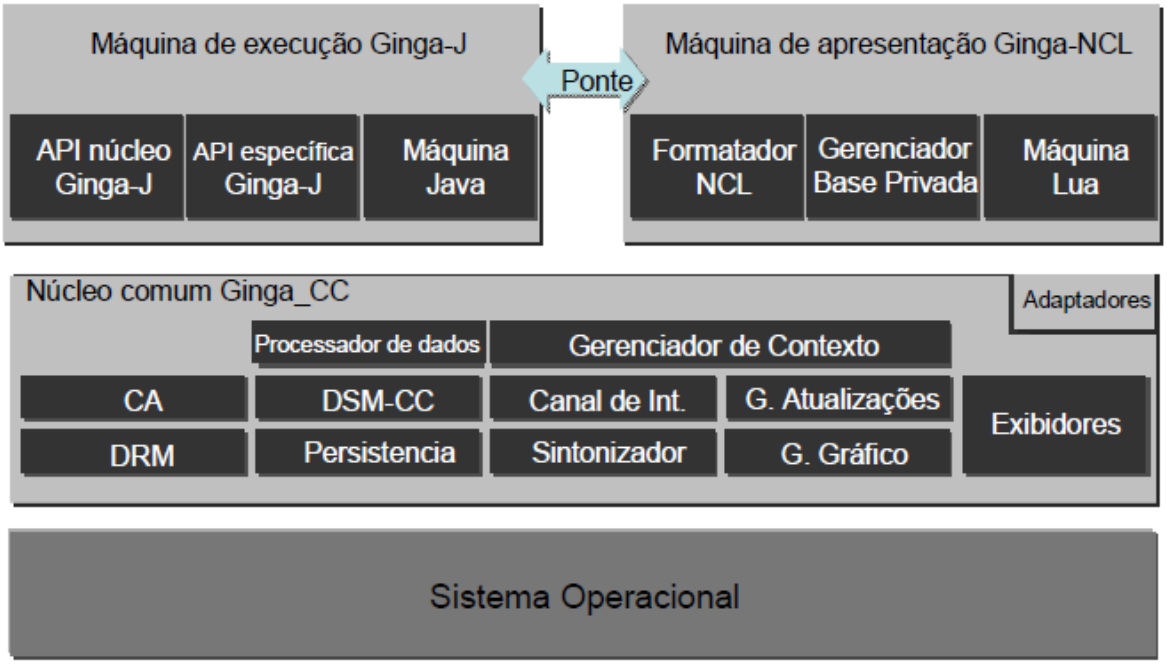


**Figura 5 - Arquitetura de Referência do *Middleware* Ginga**

A arquitetura do Ginga pode ser dividida em três módulos principais: Ginga-CC, Ginga-NCL e Ginga-J, como mostra a Figura 5 . Os dois últimos módulos compõem a camada de Serviços Específicos do Ginga.

Ginga-CC (Ginga *Common Core*) é o módulo lógico que provê todas as funcionalidades comuns ao suporte dos ambientes declarativo, Ginga-NCL, e

imperativo, Ginga-J. A arquitetura do sistema garante que apenas o módulo Ginga-CC deva ser adaptado à plataforma onde o Ginga será embarcado. Ginga-CC provê, assim, um nível de abstração da plataforma de *hardware* e sistema operacional, acessível através de APIs (*Application Program Interfaces*) bem definidas [9]. Um conjunto de exibidores monomídia comuns faz parte dos componentes do Ginga-CC. As características de tais exibidores são definidas em [10]. Os outros dois módulos serão explicados nas seções seguintes.

### 2.2.1. Ginga-NCL

Ginga-NCL é "[...] o subsistema lógico do *middleware* Ginga responsável pelo processamento de aplicações declarativas NCL" [11]. Segundo [12], o formatador NCL recebe um documento NCL e controla a sua apresentação, sincronizando os objetos de mídia, fazendo com que elas sejam apresentadas no momento programado.

A linguagem de programação NCL (*Nested Context Language*) foi desenvolvida pela PUC - Rio com o objetivo de facilitar as especificações de interatividade, sincronismo espaço-tempo entre os objetos de mídia, adaptabilidade, suportar múltiplos dispositivos e suportar programas ao vivo interativos não-lineares [2]. Ginga-NCL é a inovação totalmente brasileira do SBTVD [13].

Outra importante característica de NCL "é a facilidade de reuso das especificações, ou seja, todos os documentos podem importar elementos já declarados em outros documentos. NCL também oferece um nível mais alto de abstração para a autoria de programas. Além disso, é adequada no desenvolvimento de aplicações não-lineares por se tratar de uma linguagem para integração e sincronização de mídias, ou seja, facilita a sincronização temporal e espacial das mídias, dispensando muitas vezes a programação por scripts" [14].

Para tornar possível que a aplicação possa "tomar decisões em tempo real" a linguagem Lua é utilizada. "Lua foi planejada para ser utilizada por qualquer aplicação que necessite de uma linguagem de script leve e poderosa" [15]. Por isso é ideal para executar as partes procedurais dessa aplicação [16].

Neste trabalho a linguagem NCL não foi utilizada e é citada apenas para contextualizar e descrever melhor o *middleware* brasileiro.

### 2.2.2. Ginga-J

Segundo [2] "O Ginga-J foi desenvolvido pela UFPB para prover uma infra-estrutura de execução de aplicações baseadas na linguagem Java, com facilidades especificamente voltadas para o ambiente de TV digital".

O Ginga-J, como o próprio nome deixa claro, da suporte a linguagem procedural Java. De acordo com [11] "é o subsistema lógico do *middleware* Ginga responsável pelo processamento de aplicações imperativas escritas utilizando a linguagem Java". Assim é possível a manipulação de vídeos, áudios e textos.

A arquitetura do Ginga-J foi recentemente publicada oficialmente em [10]. O modelo Ginga-J distingue entre as entidades e recursos de hardware, software do sistema e aplicativos conforme descrito na Figura 6.

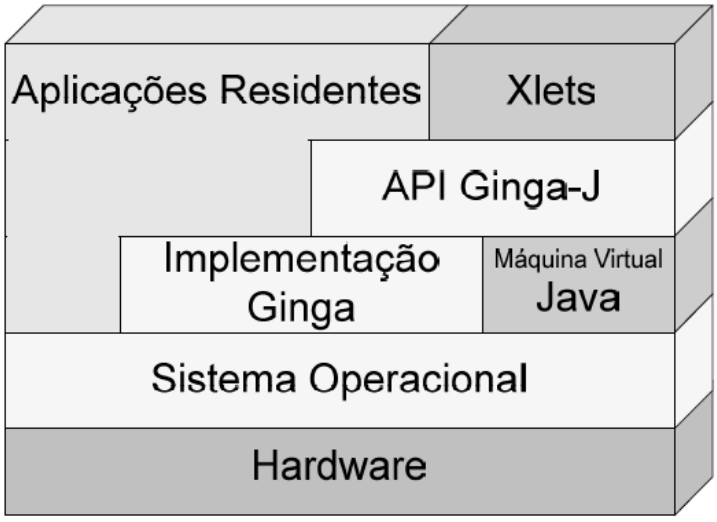


**Figura 6 - Arquitetura Ginga-J e ambiente de execução**
**Fonte: Norma ABNT NBr 15606-4**

Ginga-J é dividido em três módulos, conforme ilustrado na Figura 5, na Figura 6 e Figura 7: a máquina virtual Java; o núcleo e suas APIs, também chamadas APIs verde do Ginga-J; e o módulo responsável pelo suporte às APIs específicas do Ginga-J, chamadas de APIs amarela e vermelha do Ginga-J [13].

As APIs específicas que podem ser exportadas para outros sistemas são chamadas de amarelas. E oferecem suporte aos múltiplos usuários, a múltiplos dispositivos e a múltiplas redes. Também oferecem suporte para aplicações que poderão ser recebidas, armazenadas e executadas no futuro.

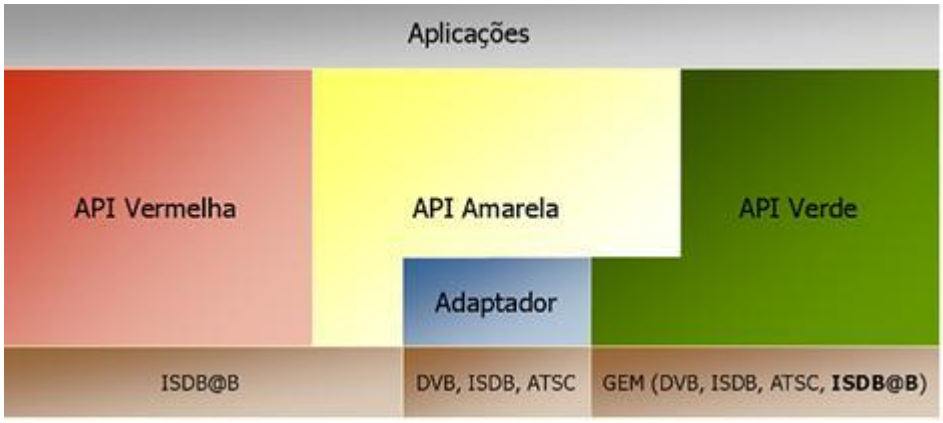


**Figura 7 - APIs Verde, Amarela e Vermelha do Ginga-J**
**Fonte: http://www.forumsbtvd.org.br**

As APIs verdes, do núcleo, são específicas e responsáveis por manter o máximo possível de compatibilidade com os sistemas: europeu e americano. Inclui o pacote Java DTV;

As APIs vermelhas dão suporte às aplicações voltadas para o Brasil, especialmente para a interatividade, promovendo a inclusão social. Permitem também a integração do conteúdo declarativo e procedural na mesma aplicação.

Alguns trabalhos como [12] e [5] comparam o desenvolvimento em Ginga-NCL e Ginga-J. Contudo esse trabalho visa comparar o desenvolvimento e o resultado final de duas aplicações similares feitas usando o padrão brasileiro e OpenTV. A aplicação feita seguindo o padrão do SBTVD será implementada usando Ginga-J.

# 3. Estudo de Caso

Este trabalho propõe um estudo comparativo entre duas das plataformas que tornam possível a TV Digital Interativa. Para realizar esse estudo foram desenvolvidas duas aplicações idênticas utilizando Ginga-J e OpenTV.

A escolha por esses padrões foi feita em virtude de Ginga-J fazer parte do middleware adotado no Brasil como padrão para a TV Digital "aberta". Já OpenTV foi indicado por ser um padrão já bem conceituado, utilizado por empresas de TV "fechada" (por assinatura) para prover a interatividade para seus clientes.

É importante ressaltar que ambas as plataformas nunca haviam sido utilizadas pelo desenvolvedor, assim como ele também não possuía conhecimento na criação de softwares para TV digital. Isso se faz necessário para garantir que a preferência por determinada linguagem não influencie no resultado das comparações.

Com a ocorrência da Copa do Mundo de Futebol neste ano de 2010, o aplicativo escolhido para o desenvolvimento foi uma versão simplificada de um simulador de classificação e resultados. Foi tomado como modelo inicial o simulador da globo.com, que pode ser visto na Figura 8.


| Grupo G | P | J | V | E | D | GP | GC | SG | GV | (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 Brasil | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 Coreia do Norte | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 Costa do Marfim | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 Portugal | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figura 8 - Simulador de Classificação**
**Fonte: globoesporte.com**

Além da motivação dada pela proximidade da Copa, a escolha em fazer uma tabela de classificação dinâmica se mostra interessante por já ser um formato de aplicação bastante utilizado e conceituado, existente na Internet e sua migração

para televisão tende a trazer junto os usuários desse tipo de aplicação web para TV. Além disso, a área esportiva já conta com uma grande participação dos telespectadores no sentido de participar da transmissão dos jogos dando sua opinião ou comentando em áreas específicas dos sites das emissoras.

O aplicativo será simplificado e contará apenas com a classificação para um grupo dos grupos da copa, contudo a arquitetura deverá permitir que trocando apenas alguns parâmetros sejam exibidos os jogos e classificação de outros grupos. Além disso, para facilitar a visualização na televisão, só será mostrada na tabela de classificação a pontuação e o número de jogos de cada equipe (as duas informações consideradas mais importantes numa tabela de classificação). Também serão mostrados no aplicativo todos os jogos do grupo.

## 3.1. Funcionalidades

Como já foi dito, serão construídos dois aplicativos idênticos, com as mesmas funcionalidades. A descrição das funcionalidades será independente das linguagens que serão utilizadas na implementação, se limitando apenas a descrever as funcionalidades em alto nível.

Primeiramente, a aplicação proverá apenas a interatividade local, ou seja, o canal de retorno não será utilizado. E o usuário não enviará informações de volta para o servidor, interagindo apenas com a aplicação que já foi carregada localmente no seu *set-top box*.

Quanto aos requisitos em si da aplicação ela deverá conter uma tabela com todos os 6 jogos de um dos grupos da Copa do Mundo. Cada jogo deverá permitir que o usuário entre com o placar que desejar para aquele jogo (Figura 9). O dispositivo que deverá ser utilizado pelo usuário para entrar com os placares deverá ser o controle remoto. Apenas números devem ser aceitos no preenchimento do placar dos jogos.


**Figura 9 - Tabela de Jogos**
**Fonte: http://copadomundo.uol.com.br/2010/simulador/**

Além de exibir a tabela de jogos do grupo a aplicação também deverá exibir a classificação do grupo similar a Figura 10. Algumas adaptações devem ser feitas para exibição da classificação na área disponível nas televisões. Por isso que serão exibidos apenas os nomes das seleções, o número de jogos e a pontuação de cada uma.

| | SELEÇÕES | PG | J | V | E | D | GP | GC | SG |
|---|---|---|---|---|---|---|---|---|---|
| 1º | Brasil | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2º | Coreia do Norte | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3º | Costa do Marfim | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4º | Portugal | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figura 10 - Classificação do Grupo**
**Fonte: http://copadomundo.uol.com.br/2010/simulador**

Após o preenchimento do placar de todos os jogos, a aplicação deverá com base nesses resultados inseridos pelo usuário atualizar o número de jogos de cada seleção e calcular a pontuação de cada uma, que seguirá o modelo padrão do futebol.

## 3.2. Arquitetura

Essa seção objetiva descrever a arquitetura das aplicações que serão desenvolvidas. É importante perceber que embora as aplicações possuem as mesmas funcionalidades, elas não possuem exatamente a mesma arquitetura. Isso ocorre em virtude dos *middleware* suportarem o desenvolvimento em linguagens de paradigmas de programação diferentes. Nas próximas sub-seções a arquitetura será descrita para cada padrão.

### 3.2.1. Arquitetura da Aplicação em OpenTV

Como foi explicado seção 2.1, OpenTV suporta o desenvolvimento em C, uma linguagem imperativa e procedural. Tal abordagem impacta diretamente na arquitetura, pois não permite uma maior abstração como é o caso de uma linguagem orientada a objetos.

No caso do desenvolvimento do sistema de tabela em OpenTV temos a seguinte visão em alto nível do sistema presente na Figura 11.

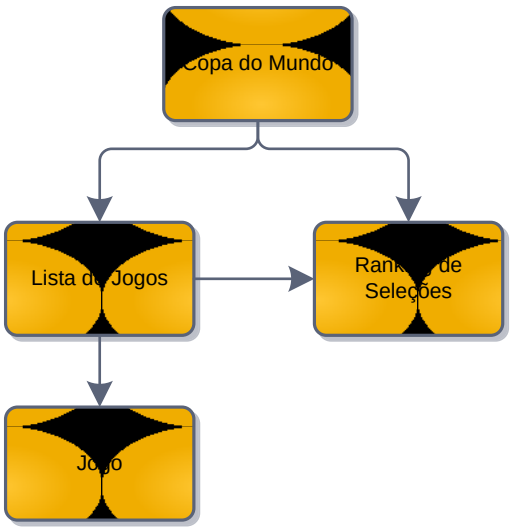


**Figura 11 - Arquitetura da Aplicação em OpenTV**

Uma vez entendida a visão macro, vamos a uma melhor descrição de seus componentes mostrados na Figura 11:

- **Tabela Dinâmica:** Essa é o módulo da aplicação responsável por gerenciar os componentes de Tabela de Jogos e Classificação Grupo. Além disso, é a aqui que está a função principal da aplicação.

- **Tabela de Jogos**: É o componente responsável por unir e exibir os jogos do grupo. Representa, como o próprio nome diz, a tabela de jogos do grupo.

- **Classificação Grupo**: Módulo responsável por calcular, atualizar e exibir a classificação do grupo de acordo com os resultados dos jogos.

- **Jogo**: Componente que representa a abstração de um jogo. Possui informações como o nome das duas seleções/times que disputarão aquela partida e o placar da mesma. Também contem os componentes gráficos que "representam" o jogo na tela

Após a breve explicação da função de cada componente, vamos à descrição da forma como eles interagem entre si e com o usuário. O diagrama que ilustra essas interações pode ser visto na Figura 12.

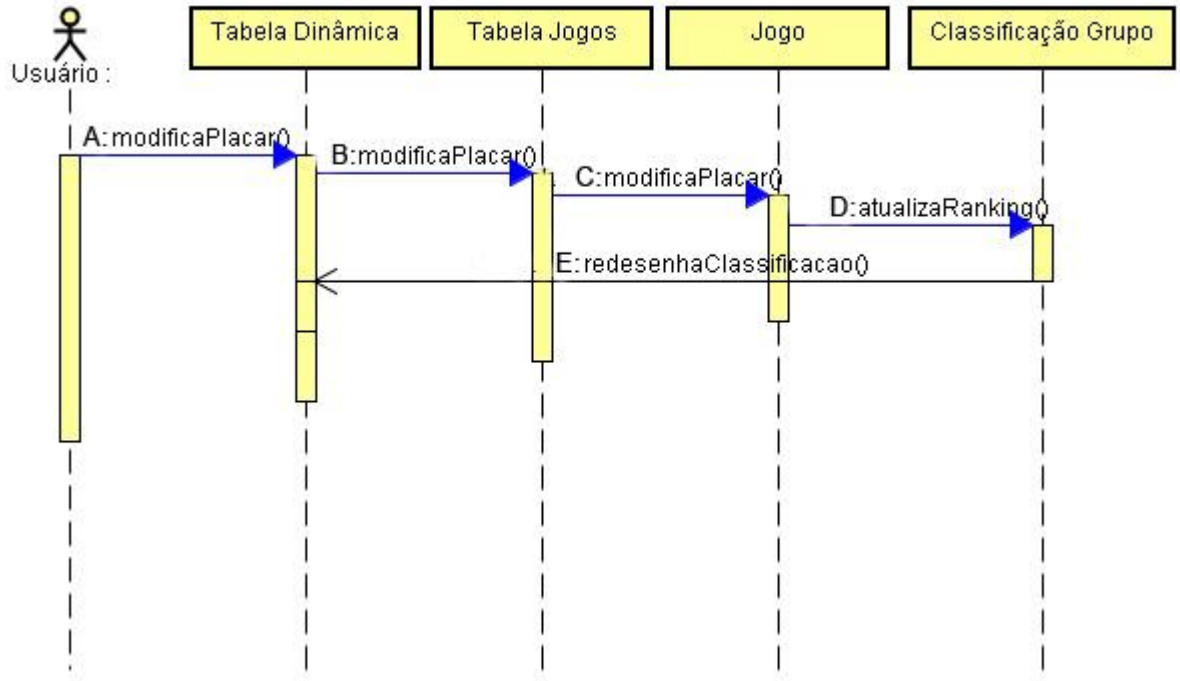


**Figura 12 - Diagrama de Seqüência da Aplicação em OpenTV**

O usuário inicialmente solicita à aplicação a modificação/atualização de um placar de determinado jogo (Figura 12, item A). Essa solicitação é recebida pela Tabela Dinâmica e repassada para Tabela Jogos (item B) que identifica qual jogo é o jogo que o seu resultado está sendo alterado e modifica o placar do mesmo (item C). Após a atualização de seu placar o componente Jogo envia a solicitação de atualização da classificação para Classificação Grupo (item D). Esse após calcular a nova classificação a envia para ser redesenhada pela Tabela Dinâmica(item E).

Como o objetivo do desenvolvimento da aplicação seria o de ganhar familiaridade com o *middleware* e com a linguagem suportada por ele, a arquitetura foi feita de forma simples e sem utilizar padrões de codificação.

### 3.2.2. Arquitetura da Aplicação em Ginga-J

Ginga-J, conforme explicado na seção 2.2.2, oferece Java como linguagem de desenvolvimento. Permitindo assim que se trabalhe em um nível mais alto. Essa maior abstração, desde que bem utilizada e respeitando conceitos como o de encapsulamento e de coesão, favorece a criação componentes reutilizáveis.

A visão em alto nível da arquitetura da tabela interativa em Ginga-J pode ser vista na Figura 13.

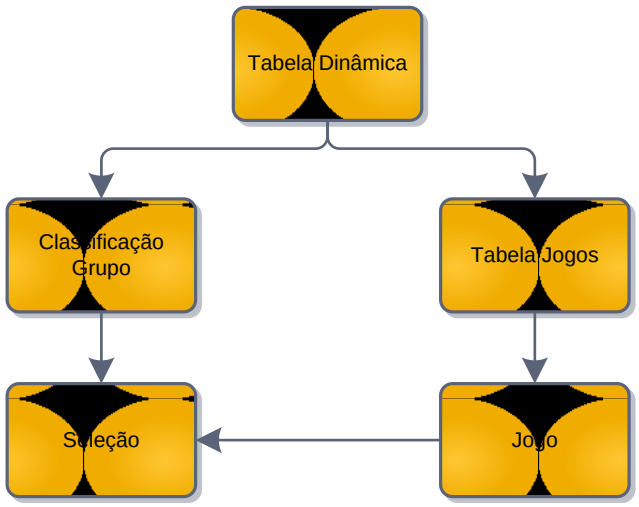


**Figura 13 - Arquitetura Aplicação em Ginga-J**

Uma vez entendida a visão geral vamos a um detalhamento de cada componente do sistema, é importante lembrar que nesse caso, por usar Java e a mesma ser orientada a objetos, cada componente representa uma classe:

- **Tabela Dinâmica**: É a classe responsável por gerenciar a Classificação Grupo e a Tabela Jogos. É a classe inicializada pelo *Xlet*.

- **Tabela de Jogos**: É a classe que agrupa e gerencia os jogos do grupo Representa a abstração da tabela do grupo.

- **Classificação Grupo**: É o componente que representa a classificação dos times/seleções no grupo. Responsável por calcular, atualizar e exibir a classificação do grupo de acordo com os resultados dos jogos.

- **Jogo**: Componente que representa a abstração de um jogo. Também é contem os componentes gráficos que "representam" o jogo na tela.

- **Seleção**: É a classe que agrupa e abstrai as informações de uma seleção/time, como o nome e a sigla. É utilizada tanto no componente Jogo quanto na Classificação Grupo.

Após a explicação de cada componente vamos à descrição da forma como eles interagem entre si e com o usuário. O diagrama que ilustra isso pode ser visto na Figura 14.

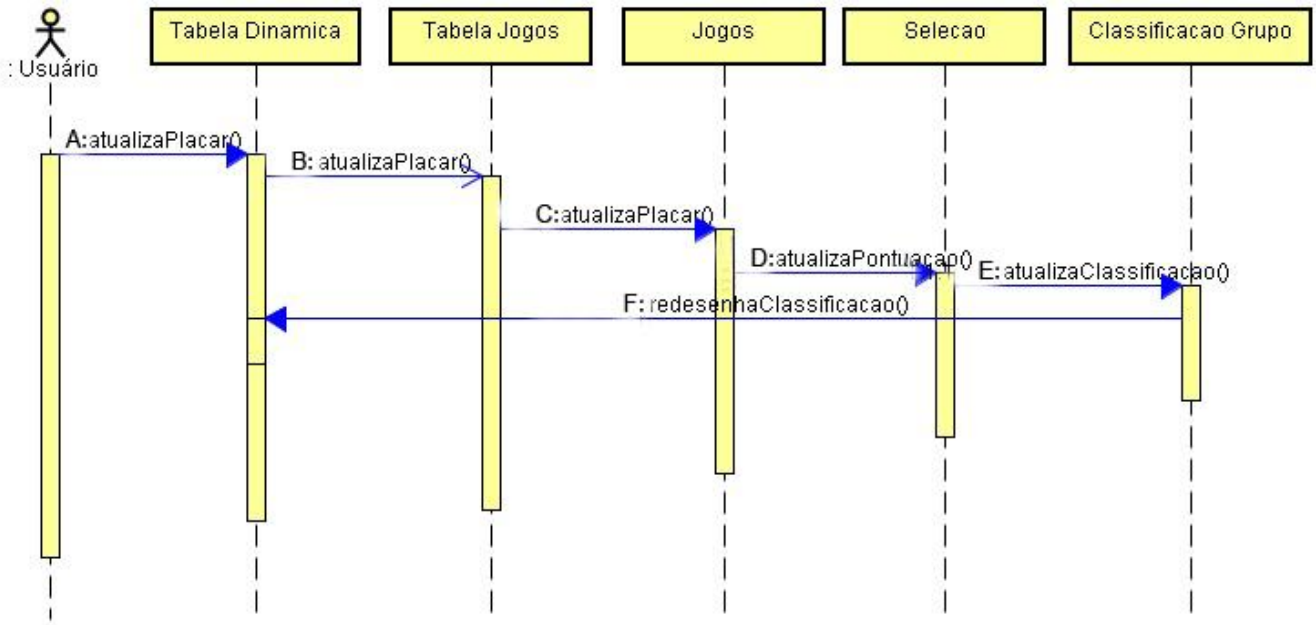


**Figura 14 - Diagrama de Seqüência da Aplicação em Ginga-J**

Como é mostrado no diagrama de classes, o usuário solicita à aplicação a atualização de um placar de determinado jogo (Figura 14, A). Essa solicitação é recebida pela Tabela Dinâmica e repassada para Tabela Jogos (item B), que identifica qual jogo que teve o seu resultado alterado e atualiza o placar do mesmo (item C). Após a atualização de seu placar o componente Jogo envia a solicitação de atualização da pontuação para as seleções que faziam parte dele (item D). Após atualizar a sua pontuação internamente a Seleção informa para Classificação Grupo que teve sua pontuação alterada e solicita a atualização da classificação (item E). Após o cálculo a nova classificação é enviada para ser redesenhada pela Tabela Dinâmica (item F).

Pela aplicação desenvolvida ser simples, a arquitetura não utilizou boa parte da infra-estrutura suportada por Java. Não foi feito o uso de padrões de projetos, afinal a intenção desse desenvolvimento, como já foi dito, é apenas de ambientação com a plataforma. Mesmo assim é importante lembrar que a utilização de uma linguagem orientada a objetos torna possível que também sejam usados inúmeros padrões de projeto.

## 3.3. Comparativo das arquiteturas

Após analisar as visões macro das duas arquiteturas pode-se imaginar que não existe muita diferença entre elas. Porém é importante ressaltar mais uma vez que a similaridade existe em grande parte pela simplicidade das aplicações. Um comparativo mostrando os componentes que existem em cada implementação pode ser visto na Tabela 1.

| Componente | OpenTV | Ginga-J |
|---|---|---|
| **Tabela Dinâmica** | ✓ | ✓ |
| **Tabela Jogos** | ✓ | ✓ |
| **Classificação Grupo** | ✓ | ✓ |
| **Jogo** | ✓ | ✓ |
| **Seleção** | | ✓ |

**Tabela 1 - Componentes Aplicação em OpenTv x Ginga-J**

A análise da Tabela 1 não é suficiente para visualizar as diferenças existentes entre as duas arquiteturas, pois as funções de alguns componentes diferem um pouco de uma arquitetura para outra apesar de possuírem o mesmo nome. As principais diferenças são mostradas no quadro abaixo.

| **Principais Diferenças da Arquitetura da Aplicação em OpenTV x em Ginga-J** |
|---|
| Em OpenTV, apesar de existir componentes, eles não são independentes e facilmente reusáveis como em Ginga-J. |
| Em Ginga-J existe o componente Seleção, enquanto em OpenTV não. |
| Em Ginga-J é o componente Seleção que guarda a pontuação de cada seleção/time, enquanto o Classificação Grupo apenas agrupa e exibe as pontuações dos times. Já em OpenTV o componente Classificação grupo é responsável por ambas tarefas. |

Apesar das diferenças de linguagens e paradigmas utilizados pelos *middleware* ainda foi possível manter a arquitetura similar entre as duas aplicações. Isso foi feito para que as implementações das aplicações pudessem ser feitas de forma mais

parecidas possíveis. Afinal se uma arquitetura fosse muito mais complexa poderia influenciar nas impressões sobre a linguagem ou sobre o *middleware*.

# 4. Desenvolvimento e Resultados

Esse capítulo irá explicar como foram desenvolvidas as aplicações, expor os resultados obtidos e comparar vários aspectos relacionados ao desenvolvimento das aplicações em ambas as plataformas.

## 4.1. Implementação

Como já foi explicado no capítulo 3, foram implementadas duas aplicações com os mesmos requisitos, uma utilizando OpenTV e a outra Ginga-J. Objetivo do trabalho é a comparação entre os dois *middlewares* e as seguintes métricas foram escolhidas para análise: IDE (*Integrated Development Environment*), API (*Application Programming Interface*), quantidade de linhas de código e documentação. Devido ao pouco tempo, alguns outros pontos de avaliação, como análise de desempenho das aplicações e aspectos relacionados à segurança, não puderam ser implementados nem avaliados.

As próximas sub-seções elucidam melhor cada métrica e a comparação entre as aplicações.

### 4.1.1. IDE (*Integrated Development Environment)*

O objetivo dessa métrica é comparar as IDE utilizadas para a implementação das duas aplicações.

O termo IDE vem do inglês *Integrated Development Environment*, ou Ambiente Integrado de Desenvolvimento, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo [17].

Os ambientes de desenvolvimento utilizados durante o desenvolvimento das aplicações foram: Eclipse [18] para a aplicação em Ginga-J e OpenTV IDE [3] que foi utilizado para a implementação no padrão homônimo.

O ambiente de desenvolvimento do OpenTV foi recentemente reformulado passando a ser baseado no Eclipse, sua versão atual pode ser vista na Figura 15. Além da IDE o ambiente possui um simulador, o OpenTV VSTB, usado para emular e visualizar as aplicações no computador.

O processo para criação de uma nova aplicação utilizando a OpenTV é bem simples, basta ir no menu "*File*" e clicar em "*Create New*" depois escolher a opção "OpenTV *Project*". Todos os arquivos de configurações são gerados bastando definir apenas algumas configurações que a própria IDE pergunta ao desenvolvedor. Já para rodar e emular a aplicação seleciona-se o projeto "*Flow*", clicar em "*Run As*" e escolher a opção "*Run flow at local target*".

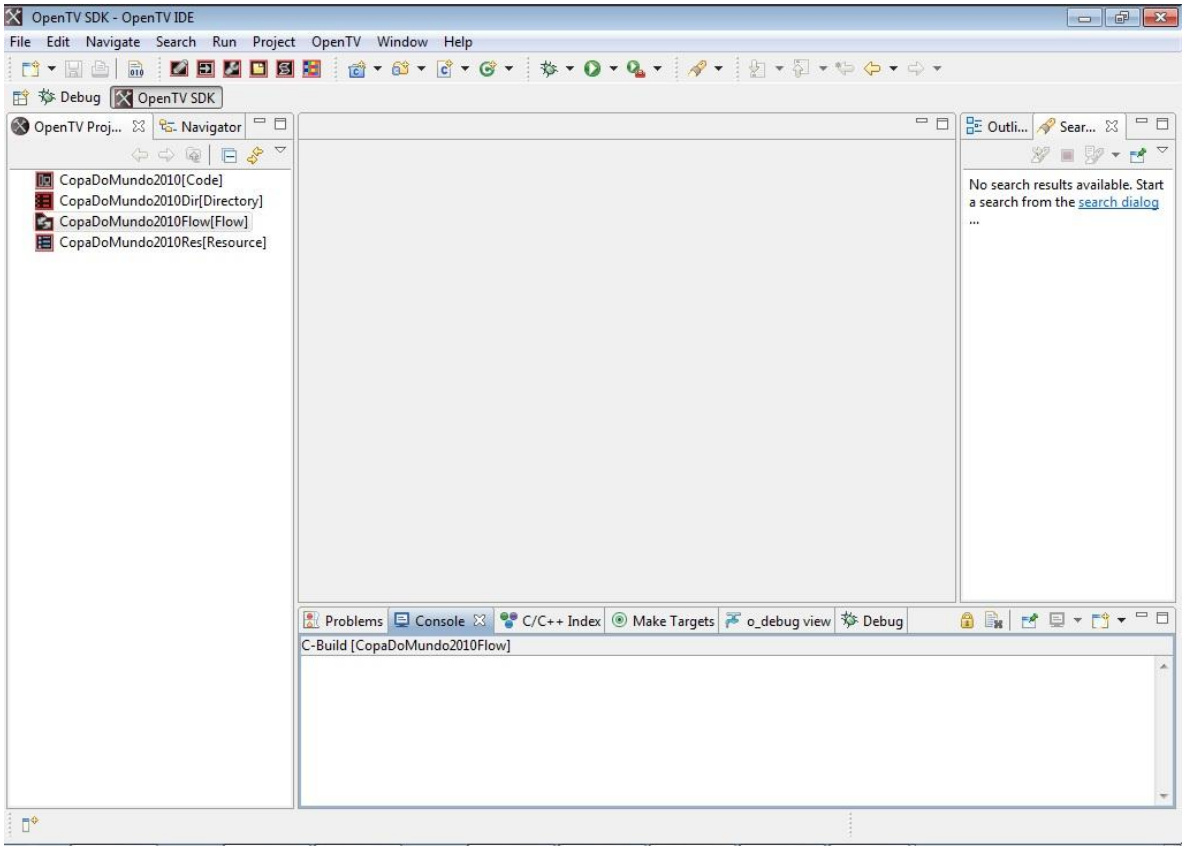


**Figura 15 - OpenTV IDE C2.2**

Já o ambiente utilizado para a implementação em Ginga-J é o já mundialmente conhecido Eclipse. Todo o processo de criação de projeto é igual ao de um projeto

Java normal e pode ser facilmente encontrado em tutorias na internet, como o que pode ser visto em [19]. A interface dessa IDE pode ser visualizada na Figura 16. Diferente de OpenTV não é fornecido nenhum emulador junto com o ambiente de desenvolvimento, por isso foi utilizado uma versão experimental e não oficial para a emulação da aplicação.

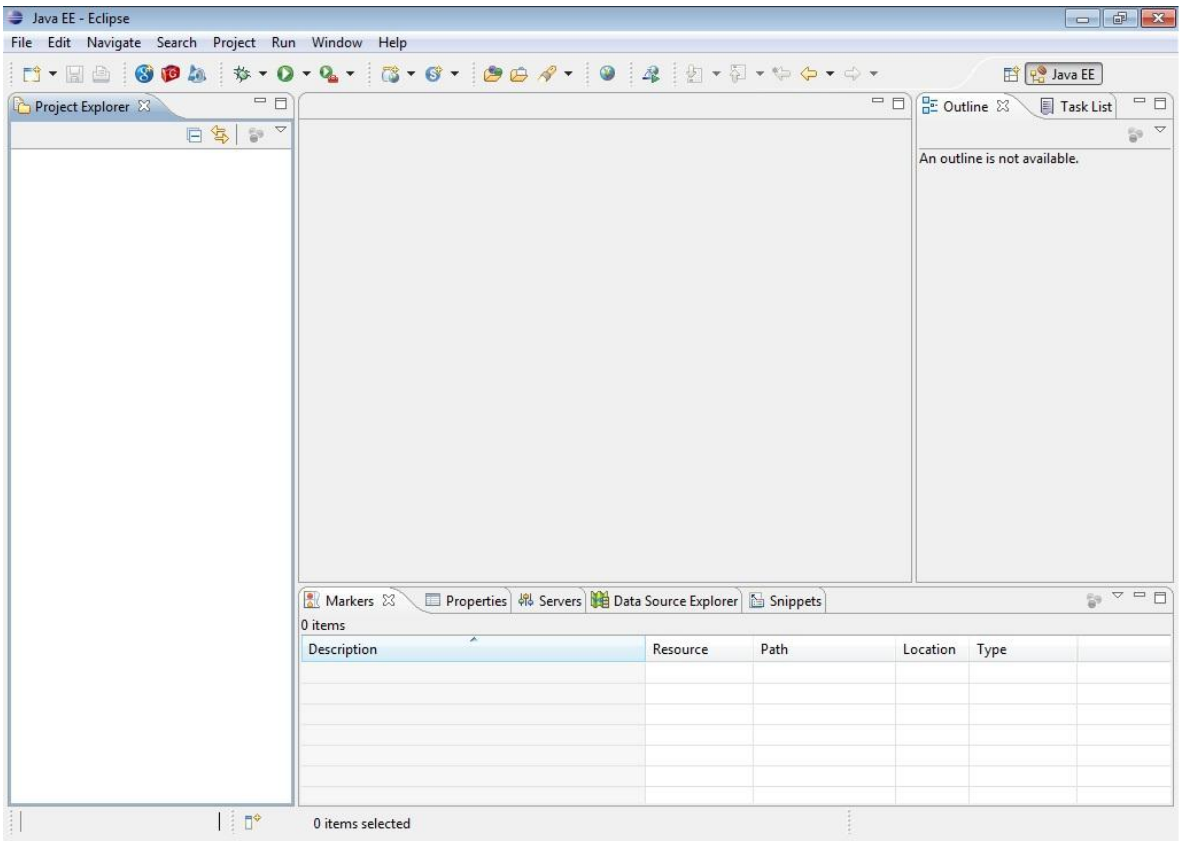


**Figura 16 - Eclipse IDE**

Se analisarmos as duas *Integrated Development Environment* veremos que as interfaces são muito similares. Fazendo uma comparação superficial baseada na Figura 15 e na Figura 16 vemos apenas 6 botões a mais na OpenTV IDE, que servem apenas de atalhos para ações como a criação de um novo projeto OpenTV. Tal semelhança entre as IDEs acontece pelo fato da OpenTV IDE ter sido construída em cima da plataforma Eclipse. Esse fato torna as duas muito semelhantes também quanto à usabilidade e não apenas no visual. A grande diferença fica por conta da ausência de um emulador integrado com a IDE para Ginga-J.

### 4.1.2. API (*Application Programming Interface*)

API, de *Application Programming Interface* (ou Interface de Programação de Aplicativos) é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por programas aplicativos que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços [20].

Esse ponto de avaliação foi escolhido em virtude da importância das APIs para o desenvolvimento de uma aplicação, afinal são elas que facilitam ou inviabilizam o trabalho do programador dando suporte às funções pré-definidas.

De forma geral o desenvolvimento de aplicações para set-top boxes por meio do Ginga-J é feito de forma idêntica às aplicações feitas para *desktop* no Java SE. Tanto o paradigma quanto as possibilidades de estruturação do código e abstração são preservados da versão padrão de Java. A diferença é que a máquina virtual utilizada é uma variante reduzida, pois os set-top boxes tem menor capacidade de processamento e armazenamento. Por conta disso o middleware suporta menos bibliotecas da linguagem do que é suportado por JSE. Apenas um subconjunto das APIs de Java estão disponíveis para uso no Ginga.

Um aspecto importante a ser ressaltado é que foi utilizada uma implementação ainda em estado experimental para o desenvolvimento da aplicação em Ginga-J. Em virtude disso as APIs dessa implementação ainda possuem alguns erros, o que levou a uma pequena dificuldade na utilização das mesmas. O grande ponto positivo encontrado foi o fato de a utilização ser idêntica a da versão padrão de Java.

Enquanto Ginga-J oferece suporte a um número reduzido de bibliotecas, o OpenTV fornece ao desenvolvedor um conjunto bastante extenso de funções em suas APIs. Existem funções que fazem parte do núcleo (core) da linguagem e outras que fazem parte de extensões opcionais do middleware.

Essas funções oferecidas servem para os mais diversos propósitos, tais como criação de componentes gráficos (chamados de *gadgets*), acesso a propriedades do sistema ou tratamento de eventos do usuário. Contudo a grande maioria dessas funções é de baixo nível, ou seja, para que o desenvolvedor possa de fato criar algo,

é necessário um esforço razoável de codificação, mesmo com o uso das bibliotecas oferecidas. Um exemplo é a biblioteca gráfica, que oferece apenas recursos para se desenhar textos e formas geométricas, em vez de oferecer componentes de mais alto nível, como botões, rótulos e caixas de texto. No entanto, apesar da biblioteca gráfica padrão ser muito limitada, isso pode ser resolvido com a adição de APIs pagas que oferecem suporte a Adobe Flash.

Sendo assim, temos OpenTV com um número extenso de APIs para os mais diversos fins, porém grande parte delas são pagas e suas funções são de baixo nível, exigindo muita codificação para sua utilização. Por sua vez, Ginga-J oferece opções mais restritas, porém com funções em alto nível e de fácil utilização, além de ser totalmente livre de *royalties*.

### 4.1.3. Quantidade de linhas de Código

Essa métrica foi escolhida para dar uma noção da complexidade da aplicação. É sabido que uma aplicação com 500 linhas de código não é obrigatoriamente mais complexa que outra com 200 linhas, pois olhar apenas a quantidade de código pode não avaliar a funcionalidade e complexidade do mesmo. Porém uma diferença muito grande no número de linhas de código de duas aplicações idênticas, desenvolvidas pela mesma pessoa e com a única diferença de utilizar linguagens diferentes, pode sim indicar que o desenvolvimento de uma foi mais trabalhoso que outra.

Para começar, vamos analisar as informações do código em Ginga-J que, como foi explicado na seção 4.1.2, é feito de forma idêntica às aplicações feitas para *desktop* no Java SE. Para a aplicação da tabela interativa foram necessárias 6 classes e no total 410 linhas de código. È importante citar que não foi direcionado esforço para que o desenvolvimento seguisse padrões de codificação.

Já o código da aplicação em OpenTV, que como explicado na seção 3.2.1 foi desenvolvido em C, possui 20 arquivos totalizando aproximadamente 2500 linhas de código. É relevante lembrar que os componentes gráficos utilizados pela a

aplicação tiveram que ser implementados, pois as APIs não possuíam artefatos similares.

Para uma melhor visualização da comparação foi construída a Tabela 2. Nela podemos observar que a aplicação em OpenTV exigiu praticamente o quíntuplo de linhas de código que a outra. Além de ter utilizado 20 arquivos contra 6 classes da aplicação em Ginga-J. Contudo, apesar da aplicação em OpenTV possuir muito mais arquivos que a outra aplicação, a metade deles foi implementação de *gadgets,* arquivos responsáveis por componentes visuais, e poderiam ser reaproveitados para a construção de outras aplicações.

Sendo assim verifica-se uma tendência que o primeiro desenvolvimento de uma aplicação em OpenTV tende a ser um pouco mais complexo que em Ginga. Porém nos próximas implementações o esforço tende a ser menor uma vez que um conjunto de componentes gráficos já teriam sido implementados. Já o esforço para o desenvolvimento em Ginga-J tende a ser o mesmo, visto que todos os componentes gráficos já estão prontos e disponíveis para uso através das APIs.

| | **Aplicação em Ginga-J** | **Aplicação em OpenTV** |
|---|---|---|
| **Número de Classes/Arquivos** | 6 | 20 |
| **Número de Linhas** | 410 | 2500 |

**Tabela 2- Comparativo Quantidade Linhas de Código**

### 4.1.4. Documentação

Essa seção tem o propósito de comparar a documentação disponível acerca dos *middleware* e o suporte ao desenvolvimento dado por cada um. Quando falamos em documentação é uma referência aos manuais que explicam cada função das APIs.

Por ser um middleware fechado, não é permitida a existência de fóruns de discussão sobre OpenTV, centralizando-se todo o conhecimento sobre a plataforma nas mãos da empresa fabricante. O acesso a manuais e a documentação da linguagem é feita através de compra dos mesmos junto à empresa o que dificulta o desenvolvimento para essa plataforma, contudo o ponto positivo é a existência de um suporte oficial dado pela companhia responsável pelo *middleware*.

O Ginga-J, por sua vez, não exige qualquer forma de pagamento para seu uso e sua documentação. No entanto, é uma tecnologia muito nova, só foi oficialmente regulamentada muito recentemente, e em virtude disso são praticamente inexistentes fóruns que discutam aspectos relacionados ao desenvolvimento nessa tecnologia.

## 4.2. Resultados

Como não podia deixar de ser, o resultado obtido contempla as duas aplicações implementadas para o middleware OpenTV e para Ginga-J, as quais podem ser vistas, respectivamente, na Figura 17 e na Figura 18. Analisando as imagens, é possível ver que visualmente existem diferenças na interface, porém as funcionalidades e requisitos foram preservados nas duas.

As diferenças visuais na interface ocorreram em virtude do uso de APIs diferentes. Além da tabela interativa em OpenTV ter tido a necessidade de implementação de todos os componentes gráficos durante o desenvolvimento. Isso aconteceu em virtude do problema apresentado na seção 4.1.2, onde vimos que as bibliotecas gráficas fornecidas com a implementação básica do *middleware* permitem apenas o desenho de textos e formas geométricas. Já em Ginga-J as APIs padrão foram utilizadas. Também é importante ter em mente que não foi feito um esforço adicional de design para que as aplicações ficassem iguais.


**Figura 17 - Aplicação em OpenTV**

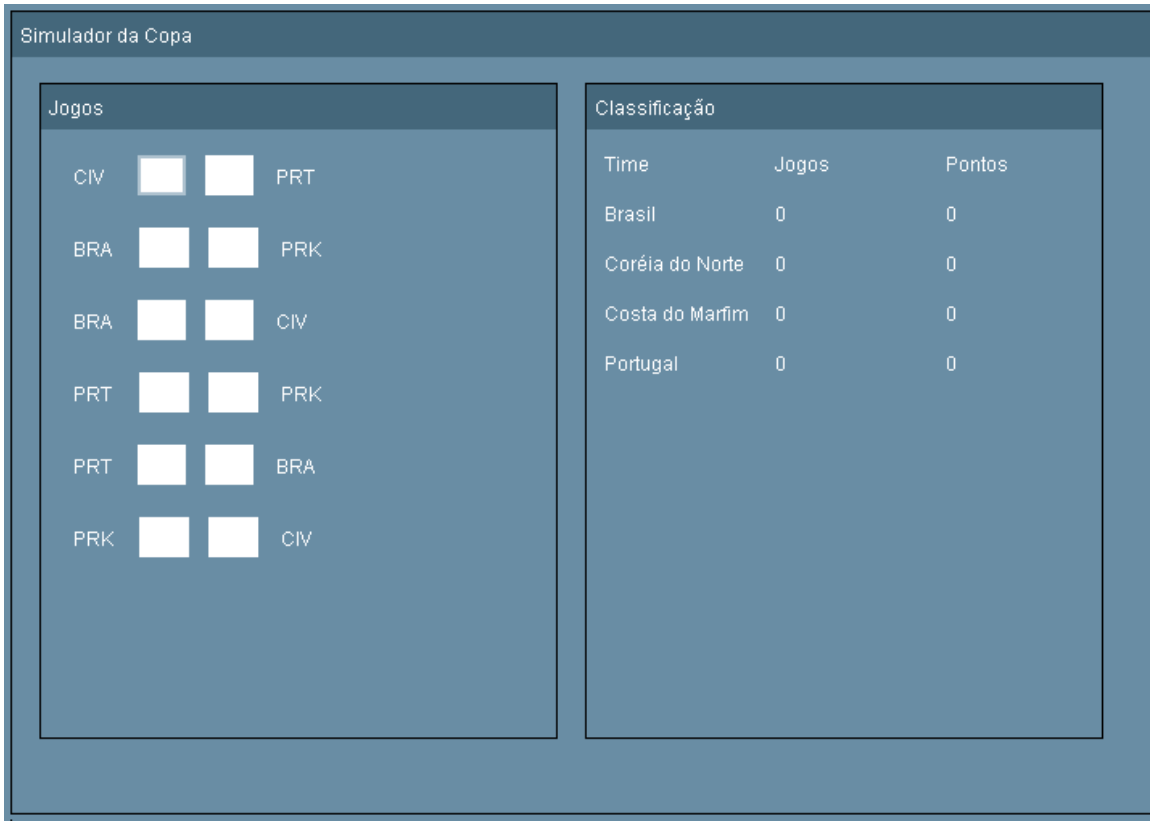


**Figura 18 - Aplicação em Ginga-J**

## 4.3. Comparação

Após uma análise separada nas seções anteriores, essa seção visa reunir e consolidar as informações expostas nelas. Para começar, é importante lembrar que OpenTV é um *middleware* proprietário e seu uso exige pagamento, enquanto Ginga-J é aberto e livre de *royalties*

Já os ambientes de programação, como explicado na seção 4.1.1 são bastante similares, afinal a OpenTV IDE foi construída sobre o ambiente do Eclipse. Já no que se refere ao modelo de construção da aplicação e as principais APIs, Ginga-J segue o modelo de construção baseado no JavaTV utilizando *Xlets,* e utiliza um subconjunto das bibliotecas da versão padrão de Java. OpenTV usa como modelo de desenvolvimento o padrão de C, utilizando *main* e *callback*, além de suportar um subconjunto das bibliotecas de C *Standard* e possibilitar a adição de APIs adicionais.

Para finalizar a comparação foram analisados aspectos como a documentação, o suporte ao desenvolvimento e a curva de aprendizado. Nesses pontos o Ginga-J possui poucos exemplos e não oferece suporte oficial para os desenvolvedores. Porém, o tempo para o aprendizado é curto, principalmente para programadores Java, ao passo que OpenTV possui apenas os exemplos fornecidos com a ferramenta e oferece um suporte oficial pago para o seu desenvolvimento. Contudo, o grau de facilidade para o seu aprendizado pode ser classificado como 'médio' para quem já lidou com C e 'longo' para quem nunca teve contato com essa linguagem. Para uma melhor visualização e percepção das diferenças foi construída a Tabela 3 reunindo os aspectos discutidos nesse capítulo.

| Critério | Ginga-J | OpenTV |
|---|---|---|
| **Ambiente de Programação** | Eclipse | OpenTV IDE |
| **Especificação Fechada vs. Aberta** | Aberta | Fechada (Proprietária) |
| **Modelo de Aplicação** | Xlet (JavaTV) | Padrão C (*main* e *callback*) |
| **Principais Bibliotecas** | Subconjunto de Java SE | Subconjunto de C *Standard* (com por exemplo extensões como suporte a Adobe Flash) |
| **Facilidade de Aprendizado** | Curto – especialmente para programadores Java | Médio para programadores C e Longo para programadores de outras linguagens |
| **Facilidade de Exemplos** | Pouca variedade de códigos | Baixa disponibilidade de exemplos. Apenas os fornecidos com as ferramentas OpenTV. |
| **Suporte Oficial para Desenvolvedores** | Não | Sim |

**Tabela 3 - Comparação Ginga-J x OpenTV**

# 5. Conclusão e Trabalhos Futuros

Este trabalho apresentou um estudo comparativo entre o desenvolvimento de aplicações nas plataformas OpenTV e Ginga-J. Este estudo não pretendeu apontar a melhor plataforma de desenvolvimento de aplicações, nosso principal foco foi tentar mostrar as forças de cada uma das plataformas mencionadas, de forma a possibilitar uma escolha mais embasada para futuros trabalhos. Apresentamos no Capítulo 4 os pontos positivos e negativos de cada plataforma.

A principal contribuição desse trabalho foi divulgar a experiência adquirida em ambas as plataformas, servindo como guia para desenvolvedores que desejem migrar entre plataformas e contribuir para ampliar o conhecimento da comunidade de desenvolvedores e pesquisadores na área de TVDi.

Em virtude do curto tempo para desenvolver esse trabalho, alguns pontos de comparação tiveram que ficar para trabalhos futuros como análise da segurança dos *middleware* e o desempenho das aplicações em cada um deles. Não foi possível analisar esses aspectos nesse trabalho, pois para que isso fosse possível seria necessário o desenvolvimento de uma aplicação mais elaborada, o que fugia da idéia inicialmente proposta.

# 6. Referências


[1] "Instituto Brasileiro de Geografia e Estatística. Disponível em: http://www.ibge.gov.br. Acessado em: 25/05/2010," 2010.

[2] "Site Oficial da TV Digital Brasileira. Disponível em: http://dtv.org.br/. Acessado em: 29/05/2010."

[3] "OpenTV. Disponível em: http://www.opentv.com/. Acessado em: 05/06/2010."

[4] "Ginga Digital TV Middleware Specification. Disponível em: http://www.ginga.org. Acessado em: 29/05/2010."

[5] B.J. Oliveira, "Um estudo de caso entre Ginga-J e Ginga-NCL no âmbito de aplicações interativas residentes.," 2010.

[6] J. Paulyne, A. Coêlho, R. Duarte, and C. Ferraz, "Comparação entre o Desenvolvimento de Aplicações MHP e OpenTV," 2006.

[7] "Alguns conceitos de TV Digital. Disponível em: http://imasters.uol.com.br/artigo/11545/tvdigital/alguns_conceitos_de_tv_digital/. Acessado em: 31/05/2010."

[8] M. Fagerqvist and A. Marcussen, "Application and System Migration from OpenTV to DVB-MHP," *Studies in Higher Education*, vol. 26, 2000, pp. 21-34.

[9] S.D. Barbosa and L.F. Soares, "TV digital interativa no Brasil se faz com Ginga - Fundamentos, Padrões, Autoria Declarativa e Usabilidade," Rio de Janeiro/RJ: PUC-Rio, 2008, pp. 105-174.

[10] "ABNT NBR 15606-1 (2010) , Associação Brasileira de Normas Técnicas "Televisão digital terrestre" - Codificação de dados e especificações de transmissão para radiodifusão digital."

[11] L.F. Soares, "Ambiente para desenvolvimento de aplicações declarativas para a TV digital brasileira. TV digital: qualidade e interatividade," 2007.

[12] A. ZANCARO, P.M. SANTOS, and J.L. TODESCO, "Ginga-J ou Ginga-NCL: características das linguagens de desenvolvimento de recursos," 2009.

[13] L.F. Soares and P.H. Castro, "As múltiplas possibilidades do middleware Ginga," 2008.

[14] T.M. Prota, "Moondo: Um Framework para Desenvolvimento De Aplicações Declarativas no SBTVD," 2009.

[15] "Lua – The Programming Language. Disponível em: http://www.lua.org. Acessado em: 31/05/2010."

[16] B. Hildegard, T.A. Tavares, J.C. Silva, and B.J. Oliveira, "Um estudo entre Ginga-J e Ginga-NCL no âmbito de aplicações interativas residentes," 2009.

[17] "Ambiente de desenvolvimento Integrado. Disponível em: http://pt.wikipedia.org/wiki/Ide_(software). Acessado em: 17/06/2010."

[18] "Eclipse.org.home. Disponível em: http://www.eclipse.org/. Acessado em: 17/06/2010."

[19] "Eclipse Tutorial. Disponível em: http://www.scribd.com/Eclipse-Tutorial/d/2490807. Acessado em: 15/06/2010."

[20] "API. Diposnível em: http://pt.wikipedia.org/wiki/API. Acessado em: 17/06/2010."