

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO
DO RIO GRANDE DO NORTE**

Romulo Fagundes Cantanhede

SBTVD:
COM ÊNFASE NA INTERATIVIDADE

**SBTVD:
COM ÊNFASE NA INTERATIVIDADE**

Orientador:
Ulisses Telesco, M. Sc.

**NATAL
2008**

ROMULO FAGUNDES CANTANHEDE

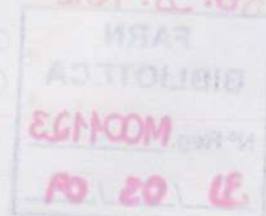
SBTVD:
COM ÊNFASE NA INTERATIVIDADE

SBTVD:
COM ÊNFASE NA INTERATIVIDADE

Monografia apresentada ao Curso de Especialização em Desenvolvimento de softwares corporativos da FARN - Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte como requisito obrigatório à obtenção do título de especialista em desenvolvimento de sistemas corporativos.

Orientador:
Ulisses Telemaco, M. Sc.

Natal
2008



Catálogo na Publicação – Biblioteca da FARN
Setor de Processos Técnicos

Cantanhede, Romulo Fagundes

SBTVD: com ênfase na interatividade / Romulo Fagundes Cantanhede.

– Natal: FARN, 2008.

34f.

Orientador: Ulisses Telemaco (Msc.)

Monografia (Especialização em Desenvolvimento de Sistemas Corporativos) – Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte.

1. Desenvolvimento de Programa – Monografia. 2. Sistema da TV Digital – Monografia. 3. API Java TV – Monografia. 4. SBTVD – Monografia. I. Título.

RN/FARN/BC

CDU 004.43

RO: 31.962



004.43
C23A
2008
ex.01

ROMULO FAGUNDES CANTANHEDE

RESUMO

Este documento tem o intuito de comentar o andamento da evolução televisiva e TV digital e suas possibilidades. Mesmo que o assunto ainda esteja em crescente discussão e sua aplicabilidade ainda seja pouco notória, o mesmo já se faz presente em diversas lugares. A monografia tem como ênfase explicar como funciona o desenvolvimento de um sistema voltado para TV digital, com ênfase na forma em que o mesmo é desenvolvido.

SBTVD:
COM ÊNFASE NA INTERATIVIDADE

Palavras-chave: TV Digital, API Java TV, Desenvolvimento, Interação.

Monografia apresentada ao Curso de Desenvolvimento de software corporativo da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte, aprovada em _____ de _____ de _____, pela banca examinadora composta pelos seguintes membros:

Ulisses Telemaco, M. Sc.
Orientador

Gleydson Azevedo Ferreira Lima, M. Sc.
Membro Examinador

RESUMO

Este documento tem o intuito de comentar o andamento da evolução relativa a TV digital e suas possibilidades. Mesmo que o assunto ainda esteja em crescente discussão, e sua aplicabilidade ainda seja pouco notória, o mesmo já se faz presente em diversos lugares. A monografia tem como ênfase explicar como funciona o desenvolvimento de um sistema voltado para TV digital, com ênfase na forma em que o mesmo é desenvolvido.

Keywords: Digital TV, Java TV API, Development, Interaction.

Palavras-chave: TV Digital, API Java TV, Desenvolvimento, Interação.

LISTA DE FIGURAS

ABSTRACT

This document has the intention of commenting the progress of developments relating to digital TV and its possibilities. Even if the matter is still growing debate, and its applicability is still somewhat remarkable, it is already present in several places. The monograph is to explain how the emphasis on developing a system focused on digital TV, with emphasis on the way in which it is developed.

Keywords: Digital TV, Java TV API, Development, Interaction.

Figura 1: Modelo OSI em relação a forma de transmissão de TV Digital	16
Figura 2: Código do servidor Usatv participante	17
Figura 3: Código do servidor Voto	19
Figura 4: Método responsável por obter os possíveis candidatos	20
Figura 5: Método responsável por enviar o voto do usuário, e em seguida retornar o resultado parcial	21
Figura 6: Tela de exibição, utilizando o JstView para testar a aplicação. Após pressionar OK, é exibida a seguinte tela	22
Figura 7: Caso seja pressionado a TECLA VERMELHA, do controle, o voto irá para o NOME1	23
Figura 8: Caso seja pressionado a TECLA VERDE, do controle, o voto irá para o NOME2	24
Figura 9: Variáveis globais da aplicação	25
Figura 10: Método inicial de aplicação, conforme o ciclo de vida da aplicação Java TV	25
Figura 11: Método executado após a inicialização é quando a aplicação instancia os objetos necessários para exibir e interagir os componentes	26
Figura 12: Método executado no momento em que o sistema é finalizado, no caso, quando é solicitado que o middleware seja finalizado	26
Figura 13: Diagrama de sequência: Usuário interagindo com o Sistema	27

LISTA DE FIGURAS

Figura 1: Modelo OSI em relação a forma de transmissão de TV Digital.	8
Figura 2: Modelo ABNT para decodificação dos canais e fluxo de informação.	11
Figura 3: Modelo ABNT para apresentação de imagem e demais informações.	11
Figura 4: Modelo ABNT para o ambiente de aplicação.	12
Figura 5: Ciclo de vida de uma aplicação JAVA TV.	14
Figura 6: Modelo GINGA de tratamento do sistema. Sendo tratado diretamente pelo hardware.	16
Figura 7: Modelo de referência para tratamento de aplicação.	17
Figura 8: Código do servlet ListaParticipante.	19
Figura 9: Código do servlet Votar.	20
Figura 10: Método responsável por obter os possíveis candidatos.	21
Figura 11: Método responsável enviar o voto do usuário, e em seguida retornar o resultado parcial.	22
Figura 12: Tela de exibição, utilizando o XletView para testar a aplicação. Após pressionar OK, é exibido a seguinte tela.	23
Figura 13: Caso seja pressionado a TECLA VERMELHA, do controle, o voto irá para o NOME1.	24
Figura 14: Caso seja pressionado a TECLA VERDE, do controle, o voto irá para o NOME2.	24
Figura 15: Variáveis globais da aplicação.	25
Figura 16: Método inicial da aplicação, conforme o ciclo de vida da aplicação Java TV.	25
Figura 17: Método executado após a inicialização é quando a aplicação instância os objetos necessários para exibir e tratar os componentes.	26
Figura 18: Método executado no momento em que o sistema é finalizado, no caso, quando é solicitado que o middleware seja finalizado.	26
Figura 19: Diagrama de sequência: Usuário interagindo com o Sistema.	27

SUMÁRIO

1. INTRODUÇÃO

1. INTRODUÇÃO	7
2. SBTVD	10
3. MIDDLEWARE	13
4. ESTUDO DE CASO	19
5. CONCLUSÃO	27
REFERÊNCIA	28
ANEXO	30

Após a década de 60, a televisão se tornou algo mais atrativo e com menor custo. Devido à percepção que se tem sobre a evolução e acessibilidade que se tem de tal aparelho, foi percebido que existia apenas uma via de mão única, na qual o espectador assiste e não tem nenhum tipo de interação com a televisão. Ou seja, televisão funciona apenas como um canal de divulgação de informações, não possibilitando que o usuário envie informações ou receba detalhes sobre o que está se passando no momento.

A principal diferença entre a televisão no formato atual (analógica) e no formato proposto (digital), além da qualidade de imagem, é a possibilidade de enviar outras informações além da imagem e som, ou seja, permitir que os usuários possam ser capazes de receber dados, e com tais dados, enviar retorno de informações. Para tal, se faz necessário o uso da internet, que deverá ser utilizada como canal de retorno para permitir uma total interação com o usuário.

Através da televisão digital é possível ter as seguintes possibilidades:

- Guia de programação;
- Agendamento de eventos;
- Reprodução de vídeos simultâneos;
- Informar sobre a descrição de programas;
- Envio de informação através da internet;
- Sistema de correio eletrônico;
- Maior quantidade de canais;
- Acesso condicional;

1. INTRODUÇÃO

Desde a criação da TV, datada de 1927, a mesma foi se popularizando e evoluindo tecnologicamente. Atualmente a mesma tem o objetivo de divulgar notícias e entreter. No Brasil, se teve a pré-estréia na data de 3 de abril de 1950, atualmente são 60 milhões de televisores, ou seja, 1,4 televisões por domicílio.

Após a década de 60, a televisão se tornou algo mais atrativo e com baixo custo. Devido à percepção que se tem sobre a evolução e acessibilidade que se tem de tal aparelho, foi percebido que existia apenas uma via de mão única, na qual o espectador assiste e não tem nenhum tipo de interação com a televisão. Ou seja televisão funciona apenas como um canal de divulgação de informações, não possibilitando que o usuário envie informações ou receba detalhes sobre o que está se passando no momento.

A principal diferença entre a televisão no formato atual (analógica) e no formato proposto (digital), além da qualidade de imagem, é a possibilidade de enviar outras informações além de imagem e som, ou seja, permitir que os usuários possam ser capazes de receber dados, e com tais dados, enviar retorno de informações. Para tal, se faz necessário o uso da internet, que deverá ser utilizada como canal de retorno para permitir uma total interação com o usuário.

Através da televisão digital é possível ter as seguintes possibilidades:

- Guia de programação;
- Agendamento de eventos;
- Reprodução de vídeos simultâneos;
- Informativos e descrição de programas;
- Envio de informação através da internet;
- Sistema de correio eletrônico;
- Maior quantidade de canais;
- Acesso condicional.

Para transmissão baseada no modelo digital, existe um modelo OSI de referência que especifica a forma como a mesma deverá ser transmitida. Veja esse modelo na figura abaixo:

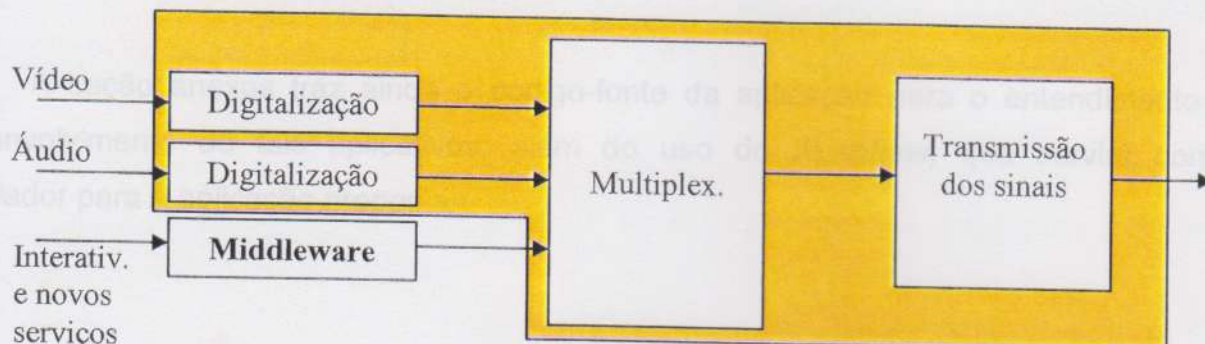


Figura 1: Modelo OSI em relação a forma de transmissão de TV Digital.
Fonte: ABNT NBR 15606.

Para que o sistema digital de TV seja implementado, se faz necessária a adoção/criação de algum padrão, que determine como o mesmo será transmitido. O Brasil optou por criar o seu próprio tipo de transmissão digital, denominado Ginga. Esse sistema permite o acesso tanto de televisores, quanto de dispositivos móveis como celular (essa estratégia permite a recepção de programação, em movimento). Para a criação de aplicativos foram desenvolvidos dois tipos de conjuntos de API, o Ginga-NCL, baseado na linguagem de script LUA, e o Ginga-J, baseado na linguagem Java e a API Java TV. Ambos os padrões permitem o desenvolvimento de aplicações para TV Digital, permitindo assim a interação do usuário com a TV.

"Hoje parece que estamos testemunhando mudanças fundamentais na natureza da comunicação mediada. A troca dos sistemas analógicos pelos sistemas digitais na codificação da informação, combinada com o desenvolvimento de novos sistemas de transmissão, (incluindo os satélites e os cabos de fibra óptica) estão criando um novo cenário técnico no qual a informação e a comunicação podem ser operadas de maneiras mais flexíveis."(Vitória apud Thompson:1998: 31)

O foco desta monografia é o estudo dos padrões brasileiros de TV digital e o desenvolvimento de uma aplicação de referência. Será utilizada a linguagem de programação Java, através da API JAVA TV.

Será realizada uma explanação de como realizar o desenvolvimento de aplicações, e das demais ferramentas que auxiliam no desenvolvimento desse tipo de sistema, incluindo uma explanação sobre o Ginga-J.

A seção anexos traz ainda o código-fonte da aplicação para o entendimento e desenvolvimento de tais aplicativos, além do uso do XLetView, que servirá como emulador para a aplicação proposta.

No padrão criado pelo Brasil, denominado Ginga, foi criado um conjunto de API baseado em Java, denominado Ginga-J, e mesmo se baseia na utilização da API Java TV. Esta monografia tem o intuito de apresentar e especificar a criação de uma aplicação, utilizando essa API. Com a vantagem da portabilidade para qualquer outro padrão que possua esse mesmo tipo de API.

Após a implantação do sistema de TV Digital, o usuário poderá interagir com a televisão, e por consequência com aplicações ricamente detalhadas em informações. Fazendo nessa perspectiva, serão necessários profissionais com entendimento na área de desenvolvimento em tais aplicações.

No passado se imaginava o homem "pegando" algum tipo de objeto através da TV, ou talvez seremos capazes de solicitar algum tipo de produto que estiver sendo produzido, de forma totalmente interativa.

Para que o usuário possa utilizar desse tipo de recurso, ele poderá ter um equipamento capaz de tratar essas informações, denominado set-top-box (no caso dos sistemas que não tiverem esse tipo de recurso ou conversor digital), através a qual a transmissão será tratada, e caso tal transmissão tenha algum tipo de aplicação, o mesmo poderá ser acessado através do set-top-box.

O modo criado pela ABNT para tratamento das imagens e visualização é o chamado modo de transmissão de dados, onde as imagens são transmitidas de forma digitalizada.

2. SBTVD

Com o advento, a crescente utilização e rápida expansão, a TV digital em breve estará em boa parte dos lugares no Brasil. Devido a essa expansão, se faz necessário a crescente mão de obra, que estará apta a criar aplicações que irão “rodar” através da TV digital.

Em 29 de julho de 2006, foi criado o fórum que discute e normatiza o padrão SBTVD, através de normas técnicas.

No padrão criado pelo Brasil, denominado Ginga, foi criado um conjunto de API baseado em Java, denominado Ginga-J, o mesmo se baseia na utilização da API Java TV. Essa monografia tem o intuito de apresentar e especificar a criação de uma aplicação, utilizando essa API. Com a vantagem da portabilidade para qualquer outro padrão que utilize esse mesmo tipo de API.

Após a implantação do sistema de TV Digital, o usuário poderá interagir com a televisão, e por consequência com aplicações ricas em detalhes e em informações. Focando nessa perspectiva, serão necessários profissionais com entendimento na área de desenvolvimento em tais aplicações.

No passado se imaginava o homem “pegando” algum tipo de objeto através da TV, em breve seremos capazes de solicitar algum tipo de produto que estiver sendo realizado propaganda, de forma totalmente interativa.

Para que o usuário possa utilizar desse tipo de recurso, ele poderá ter um equipamento capaz de tratar essas informações, denominado *set-top-box* (no caso dos televisores que não tiverem esse tipo de recurso, ou conversão digital), através dele a transmissão será tratada, e caso tal transmissão tenha algum tipo de aplicação, o mesmo será executado através do *set-top-box*.

O modelo criado pela ABNT para tratamento das camadas e visualização e tratamento de dados seriam os modelos abaixo:

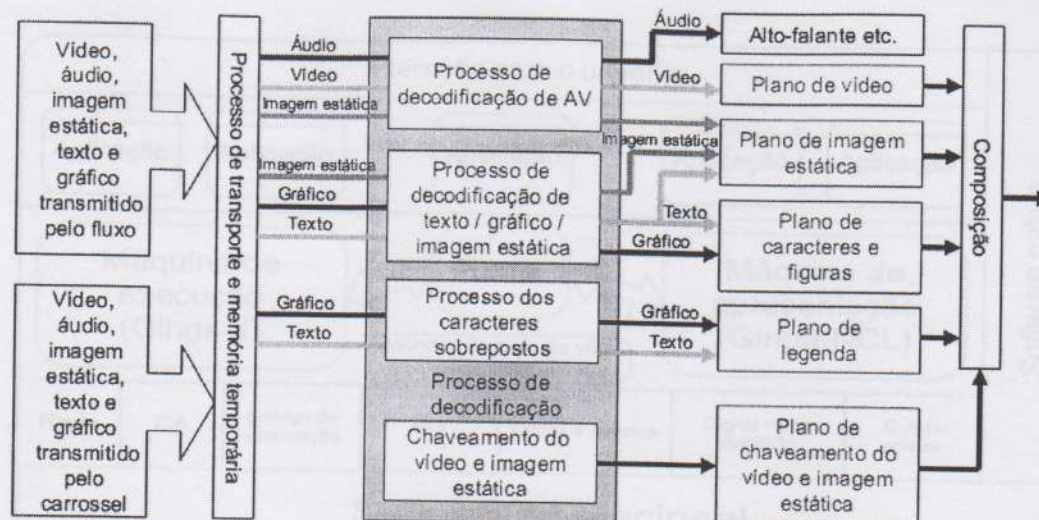


Figura 2: Modelo ABNT para decodificação dos canais e fluxo de informação.
Fonte: ABNT NBR 15606.

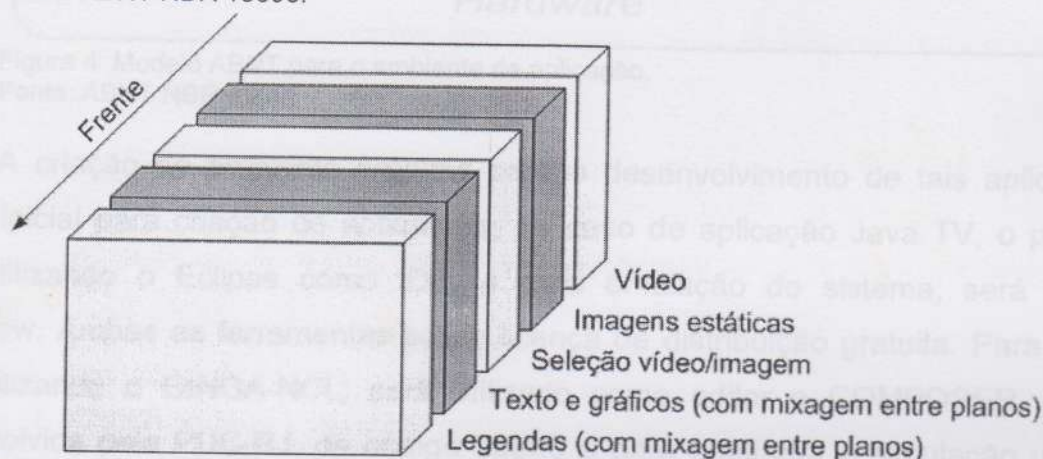


Figura 3: Modelo ABNT para apresentação de imagem e demais informações.
Fonte: ABNT NBR 15606.

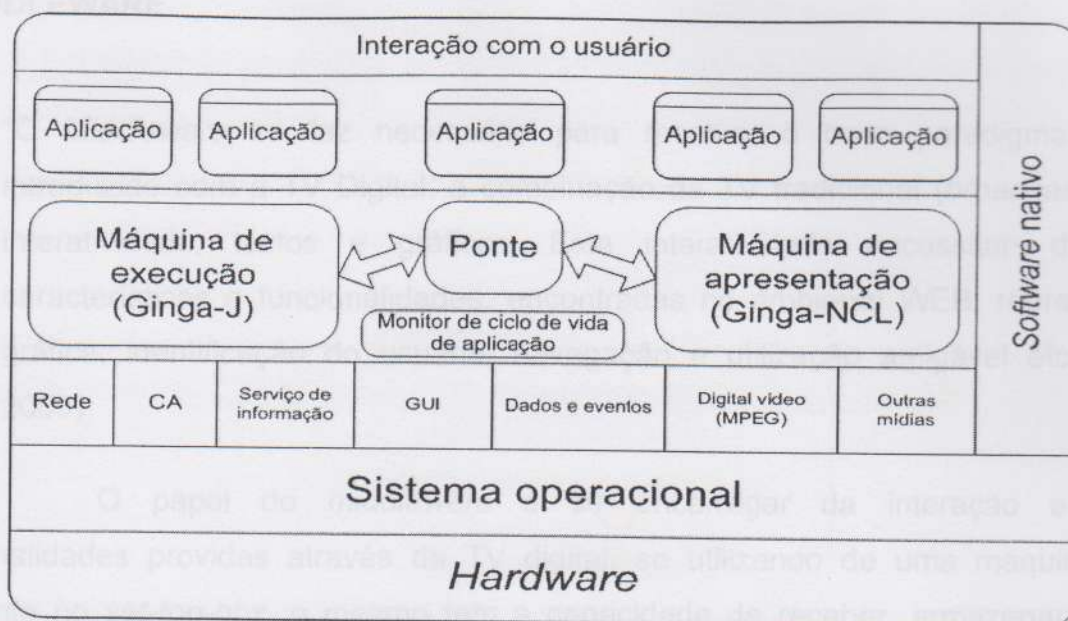


Figura 4: Modelo ABNT para o ambiente de aplicação.
Fonte: ABNT NBR 15606.

A criação de ambiente propício para o desenvolvimento de tais aplicações é o passo inicial para criação de aplicativos, no caso de aplicação Java TV, o projeto será feito utilizando o Eclipse como IDE, e para emulação do sistema, será utilizado o XLetView. Ambas as ferramentas sobre licença de distribuição gratuita. Para o caso de uso utilizando o GINGA-NCL, será utilizado como editor o COMPOSER, ferramenta desenvolvida pela PUC-RJ, de código aberto e para emulador e simulação um sistema específico distribuído no próprio site do GINGA-NCL.

3. MIDDLEWARE

Conforme a descrição abaixo, segue o ciclo de vida de uma aplicação Java TV:

"O *Middleware* se faz necessário para resolver o novo paradigma que foi introduzido com a TV Digital: a combinação da TV tradicional (*broadcast*) com a interatividade, textos e gráficos. Esta interatividade necessitar de várias características e funcionalidades, encontradas no ambiente WEB: representação gráfica; identificação do usuário; navegação e utilização amigável etc." (Paes, 2005)

O papel do *middleware* é se encarregar da interação e demais funcionalidades providas através da TV digital, se utilizando de uma máquina virtual residente no *set-top-box*, o mesmo tem a capacidade de receber, armazenar e enviar informações.

Ainda não existe um padrão de *middleware* adotado por todos os fabricantes, mas existe alguns padrões de transmissões que já adotam alguns tipos comuns e de uma linguagem específica.

O mesmo também tem a capacidade de tratar e exibir outros vídeos em comum ao que está sendo exibido na programação exibida.

3.1 JAVA TV

O padrão JAVA TV é baseado no JAVA ME, voltada para dispositivos com baixo processamento, e baixo recurso de memória. A criação de interfaces, fica por responsabilidade da AWT (*Abstract Window Toolkit*) e para tratamento de mídia o mesmo se utiliza da JMF (*Java Media Framework*), voltada para vídeos e outras multimídias que poderão ser exibidas.

- *javax.tv.net* – permite a comunicação através de IP, para envio e recebimento de informações.
- *javax.tv.service* – é responsável por receber e tratar informações referentes aos canais, tais como tipo de canal, nome do canal, e canais do mesmo.

Conforme é descrito abaixo, segue o ciclo de vida de uma aplicação Java TV:

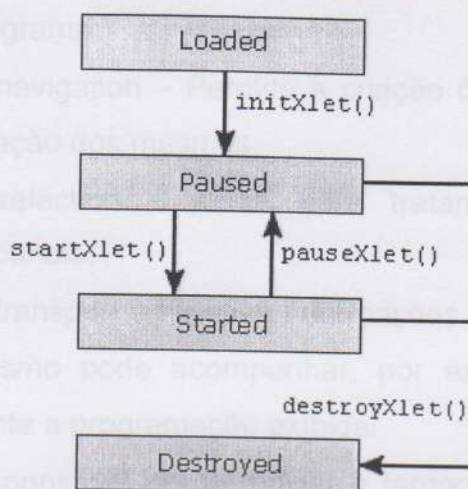


Figura 5: Ciclo de vida de uma aplicação JAVA TV.

Fonte: <http://www.interactivetvweb.org/tutorials/javatv>

A JSR 927 sendo responsável por documentar e tratar as especificações do JAVA TV, a mesma foi dada início em 08 de novembro de 2005.

Essa API tem várias classes para o tratamento das necessidades do sistema, seguem algumas e suas funcionalidades:

- `javax.tv.carousel` – caso a prestadora de serviço envie algum tipo de arquivo em forma de *broadcast*, o mesmo será tratado por essa classe, em conjunto com o pacote `Java.io`.
- `javax.tv.graphics` – responsável por instanciar e descrever o mecanismo principal de tratamento de imagem.
- `javax.tv.locator` – Tem o intuito de facilitar a localização de objetos.
- `javax.tv.media` – serve para tratamento e controle de eventos de mídias em tempo real. O mesmo pode trabalhar em conjunto com o `JMF`.
- `javax.tv.media.protocol` – recepção de mídia de forma assíncrona, para exibição de mídia.
- `javax.tv.net` – permite a comunicação através de IP, para envio e recebimento de informações.
- `javax.tv.service` – é responsável por receber e tratar informações referente aos canais, tais como tipo de canal, nome do canal, e censura do mesmo.

- javax.tv.service.guide – responsável por receber a grade de programação dos canais (EPG). Incluindo informações como tempo de início e fim de determinado programa.
- javax.tv.service.navigation – Permite a criação de grupos de canais, para facilitar a navegação dos mesmos.
- javax.tv.service.selection – serve para tratamento de permissão de navegação dos canais.
- javax.tv.service.transport – repassa informações referente a transporte dos serviços. O mesmo pode acompanhar, por exemplo, qualquer tipo de mudança referente a programação exibida.
- javax.tv.uil – responsável por gerenciar o tempo, podendo assim, realizar programação de eventos.
- javax.tv.xlet – semelhante a um *applet*, o mesmo é responsável por gerenciar e tratar o estado da máquina. No momento em que a máquina virtual é inicializada, a mesma chama os eventos necessário para que a aplicação funcione.

3.2 MHP

O projeto MHP (*Multimedia Home Platform*) é um padrão utilizado pelo padrão de transmissão digital denominado DVB (padrão adotado na Europa). Devido ao tempo de existência do mesmo, ele já tem diversas definições e padrões que foram aperfeiçoados com o tempo, se tornando um projeto “maduro” e bem definido.

A implementação permite que os desenvolvedores criem aplicações e que possam realizar download de tais aplicações. É permitida a criação de conteúdo em padrão HTML (utilizando CSS, Javascript) ou Java.

O MHP criou uma especificação definindo que tipo de aplicação deve ser utilizada, e em que formato o mesmo necessita ser criado, para tal, denominou de GEM (Globally Executable MHP), fazendo com que outros padrões (incluindo o Ginga), possam utilizar o MHP.

3.4 HAVi

O HAVi (*Home Audio Video Interoperability*), é um conjunto de API utilizado para melhorar e permitir mais recursos para aplicações utilizando JAVA TV, permitindo explorar outros componentes além dos componentes AWT da API JAVA.

Outro ponto positivo para o HAVi, é o fato de o mesmo poder gerenciar outros dispositivos externos, tais como vídeos cassetes e câmeras de filmagem.

3.5 GINGA

O projeto GINGA, foi denominado dessa forma, devido a uma questão cultural e artística relativo à atitude e forma que o brasileiro consegue lidar com a vida.

O mesmo foi concebido em formato de código livre, haja vista a possibilidade que todos os brasileiros tenham de gerar conteúdo livre, permitindo a criação de interatividade para serem publicadas em canais comunitários e grandes emissoras.

Um dos fatores importantes para criação de padrão foi proporcionar facilidade no desenvolvimento de aplicações, independente dos tipos de dispositivos receptores, em relação à fabricante e mobilidade. Principalmente focado nos usuários que possuem televisão e que sejam da classe D,E.

O modelo adotado pelo GINGA oferece uma camada entre a seção de visualização e as demais aplicações.

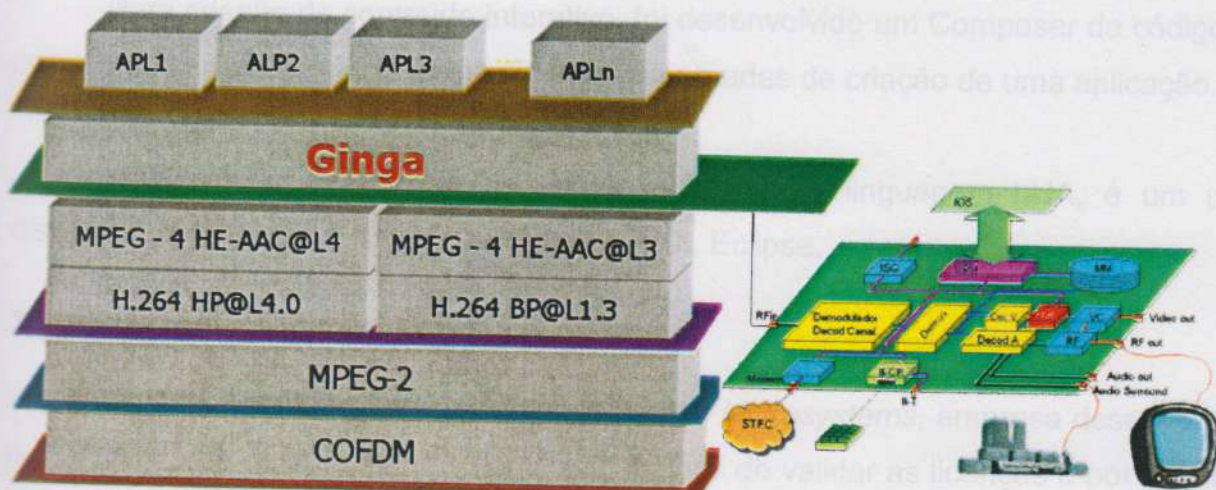


Figura 6: Modelo GINGA de tratamento do sistema. Sendo tratado diretamente pelo hardware.
Fonte: <http://www.ginga.org.br>

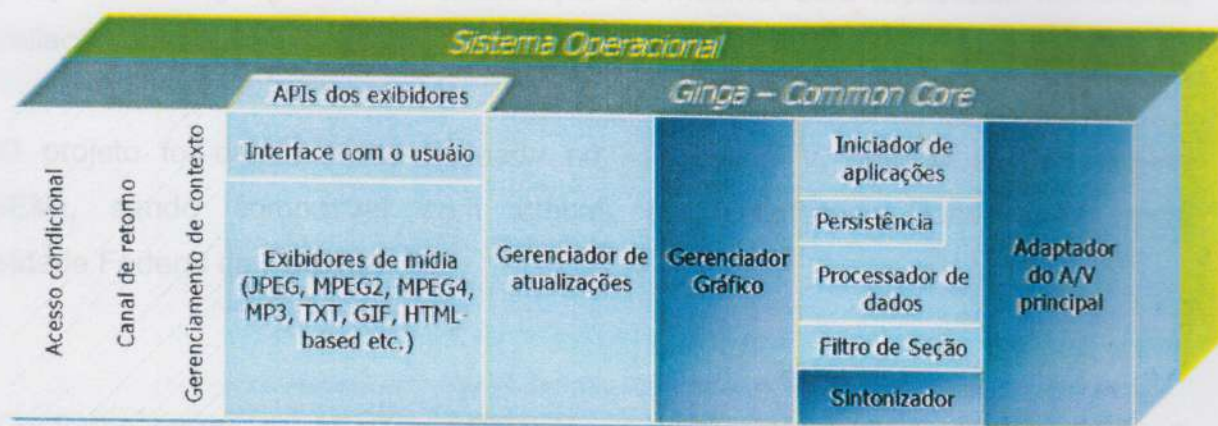


Figura 7: Modelo de referência para tratamento de aplicação.
Fonte: <http://www.ginga.org.br>

3.5.1 GINGA-NCL

O GINGA-NCL (*Nested Context Language*), é feito para criação de conteúdo interativo, linguagem de cunho declarativo, e com foco em sincronismo temporal. O mesmo tem a premissa de permitir a declaração/manipulação de variável através da linguagem de programação LUA.

A linguagem LUA foi adotada pela simplicidade na criação de sistemas e por permitir um altíssimo desempenho.

Ambos foram criados na PUC-RIO, tanto a linguagem LUA, como o padrão GINGA-NCL.

Para criação de conteúdo interativo, foi desenvolvido um Composer de código livre, com o intuito de abstrair de forma simples as camadas de criação de uma aplicação.

Outro modo de criação de scripts utilizando a linguagem LUA, é um plug-in disponibilizado para o Eclipse, denominado NCL Eclipse.

3.5.2 GINGA-J

O GINGA-J ainda não foi liberado pela *Sun Microsystems*, empresa desenvolvedora e mantedora do Java, a mesma ficou responsável de validar as licenças e por publicar as

especificações da linguagem. Após a liberação do mesmo, será repassado novamente para avaliação normativa.

4.1 SISTEMA DE VOTAÇÃO

O projeto foi desenvolvido baseado na API Java TV, com classe do projeto MHP(GEM), sendo compatível com ambas as bibliotecas. Desenvolvido pela Universidade Federal da Paraíba.

Conforme norma da ABNT a SBTv depende de ambos (Ginga-NCL e Ginga-J) para que a mesma possa ser oficializada e colocada em produção.

O desenvolvimento de aplicações utilizando o JAVA-TV pode ser realizado através de qualquer IDE. A simulação pode ser feita através do XletView.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    response.setCharacterEncoding("UTF-8");
    PrintWriter out = response.getWriter();
    out.println("NOME1: NOME2");
}
```

Figura 4: Código do servlet ListaParticipantes.
Fonte: O próprio autor.

4. ESTUDO DE CASO

4.1 SISTEMA DE VOTAÇÃO

O sistema de votação proposto permitirá que um usuário, assistindo a um determinado programa de televisão, possa ao final da programação, escolher ou selecionar algum tipo de opção, e em seguida enviar seu voto, através da internet.

O mesmo é desenvolvido utilizando a API Java TV, em conjunto com a API HAVi e MHP, para melhorar a visualização do sistema. Os dados serão enviados através do pacote java.net para um *servlet*, que irá incrementar um contador usado para registrar o voto.

Foram criados dois *servlet*:

- **ListaParticipante:** Responsável por oferecer a lista de pessoas que estarão em votação, o retorno é algo semelhante a: NOME1|NOME2.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    response.setStatus(response.SC_OK);
    PrintWriter out = response.getWriter();
    out.print("NOME1|NOME2");
}
```

Figura 8: Código do *servlet* ListaParticipante.
Fonte: O próprio autor.

- **Votar:** O mesmo irá receber uma variável através de método GET com o nome do usuário votado. Retornando a quantidade de votos que foram realizados.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String voto = (String)request.getParameter("voto");
    if (voto.equalsIgnoreCase("NOME1")){
        voto1++;
    }else{
        voto2++;
    }
    resultado="Voto para NOME1: "+voto1+" Voto para NOME2: "+voto2;
    System.out.println(resultado);
    PrintWriter out = response.getWriter();
    out.print(resultado);
}
```

Figura 9: Código do servlet Votar.
Fonte: O próprio autor.

Ambos foram criados sem nenhuma implementação de armazenamento de banco de dados, ou algo semelhante, haja visto, que a intenção do sistema é valorizar a aplicação voltada para TV.

Na aplicação Java TV, foram utilizadas algumas API que valorizam a interface e a forma de interação. Não foi utilizado nenhum tipo de recurso de imagem e a interação se dá apenas através do controle.

Para comunicação via Internet, foi utilizado a API java.net, com as classes/interface URL e HttpURLConnection com dois métodos:

- obterVoto – Responsável por receber os dados dos possíveis votantes. O retorno do mesmo se dá através de um array de string.

```
private String[] obterVoto() {
    String resultado[] = new String[2];
    try {
        URL url = new URL(URL + "ListaParticipante");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setConnectTimeout(10);
        con.connect();
        Scanner sc = new Scanner(con.getInputStream());
        if (sc.hasNextLine()) {
            resultado = sc.nextLine().split("\\|");
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return resultado;
}
```

Figura 10: Método responsável por obter os possíveis candidatos.
Fonte: O próprio autor.

- votar – Método que recebe como parâmetro uma String, contendo o nome do candidato e em seguida retorna o total parcial da votação.

```
private String[] obterVoto() {
    String resultado[] = new String[2];
    try {
        URL url = new URL(URL + "ListaParticipante");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setConnectTimeout(10);
        con.connect();
        Scanner sc = new Scanner(con.getInputStream());
        if (sc.hasNextLine()) {
            resultado = sc.nextLine().split("\\|");
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return resultado;
}
```

Figura 11: Método responsável enviar o voto do usuário, e em seguida retornar o resultado parcial.

Fonte: O próprio autor.

Para interação com o usuário, foi utilizado apenas comandos do controle remoto do usuário, ou seja, a aplicação “escuta” as teclas que serão pressionadas pelo usuário. Para tal existe o seguinte mapeamento em códigos:

- 403, 404, 405, 406 – Vermelho, Verde, Amarelo e Azul.
- 27 – Exit.
- 151 - * (asterisco).
- 520 - # (grade).
- Setas:
 - Cima: 38
 - Baixo: 40
 - Esquerda: 37
 - Direita: 39
- 10 –Ok

- Para os números, receber a codificação e subtrair de 48, ou seja, para 2 será recebido o código 50.

Ainda em relação ao teclado, a interface *KeyListener* define a implementação de 3 métodos: *keyTyped*, *keyPressed*, *keyReleased*. Todos recebem como parâmetro um objeto do tipo *KeyEvent*.

Para utilizar o sistema, o usuário deve pressionar a tecla OK, e em seguida as opções de votação. E conforme a cor que fosse visível, o mesmo pressiona a tecla do controle, referente a cor, e o voto é enviado. Em seguida surge o resultado parcial. Como mostra imagem abaixo:

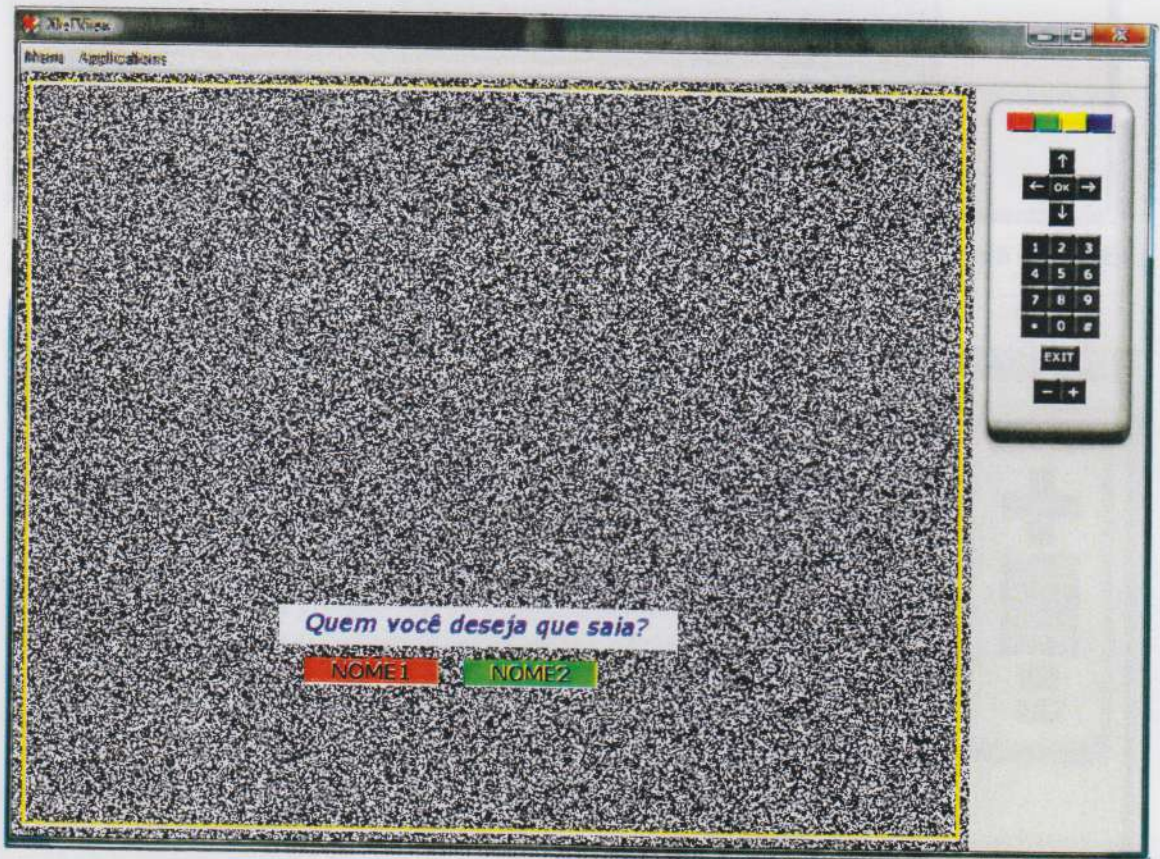


Figura 12: Tela de exibição, utilizando o XletView para testar a aplicação. Após pressionar OK, é exibido a seguinte tela.
Fonte: O próprio autor.

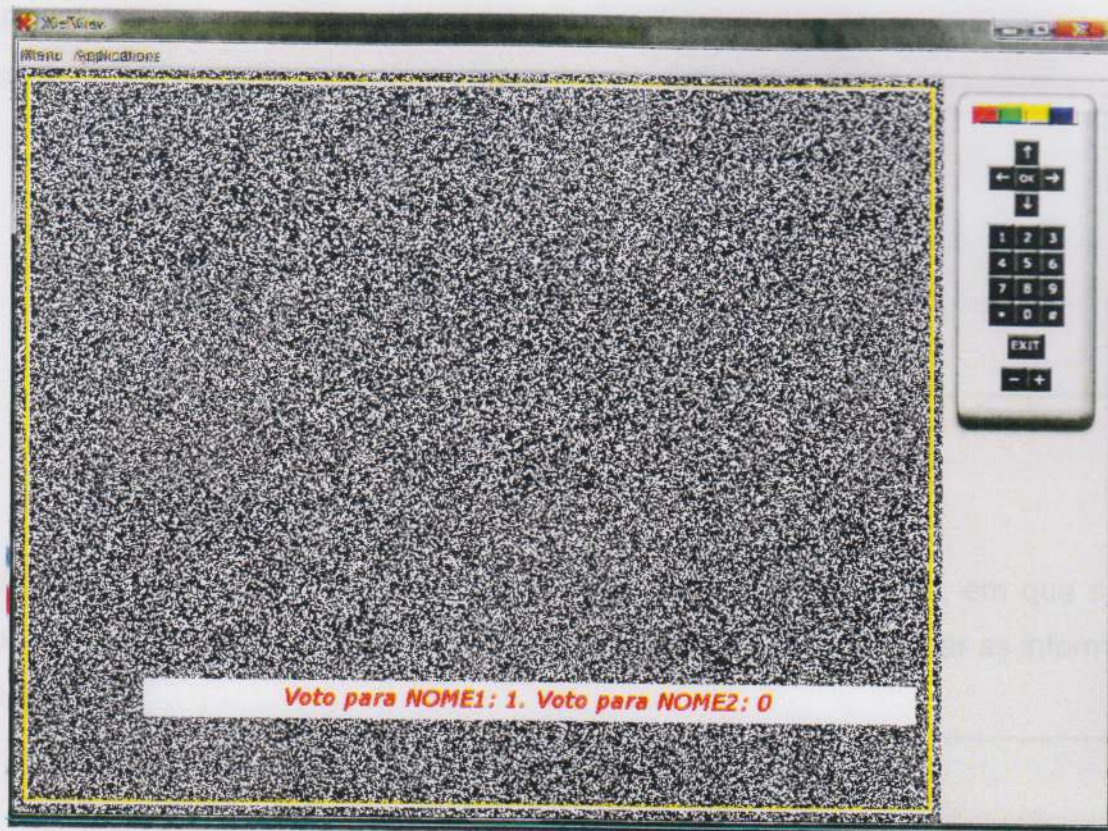


Figura 13: Caso seja pressionado a TECLA VERMELHA, do controle, o voto irá para o NOME1.
Fonte: O próprio autor.

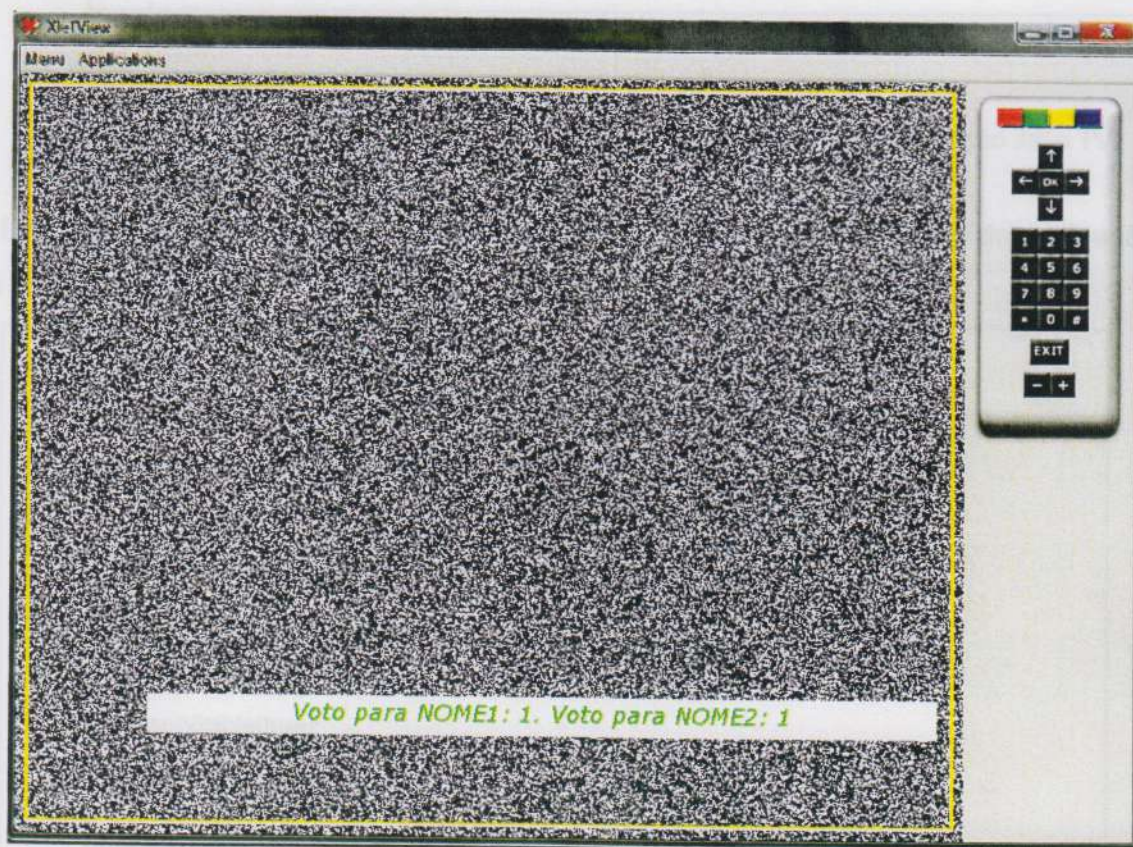


Figura 14: Caso seja pressionado a TECLA VERDE, do controle, o voto irá para o NOME2.
Fonte: O próprio autor.

Para a aplicação criada, foram criadas as seguintes variáveis de uso global do sistema:

```
private XletContext context;

private HScene scene;

private HStaticText label1, btnVermelho, btnVerde, resultado;

private String[] opcoes;

private final static String URL = "http://localhost:8080/MonografiaWeb/";
```

Figura 15: Variáveis globais da aplicação.
Fonte: O próprio autor.

Após a inicialização da classe principal, é carregado o contexto, em que se está trabalhando e em seguida, é utilizado o método de conexão, para carregar as informações de votação:

```
public void initXlet(XletContext context) throws XletStateChangeException {

    this.context = context;

    opcoes = obterVoto();

}
```

Figura 16: Método inicial da aplicação, conforme o ciclo de vida da aplicação Java TV.
Fonte: O próprio autor.

Em seguida é executado o método que irá criar a tela de exibição da aplicação, e que irá definir que a própria aplicação irá controlar os eventos.

```
public void startXlet() throws XletStateChangeException {

    HSceneFactory hsceneFactory = HSceneFactory.getInstance();

    scene = hsceneFactory.getFullScreenScene(

        HScreen.getDefaultHScreen().getDefaultHGraphicsDevice());

    scene.setBounds(0, 0, 800, 600);

    scene.setLayout(null);

    scene.addKeyListener(this);

    scene.setVisible(true);

    scene.requestFocus();

}
```

Figura 17: Método executado após a inicialização é quando a aplicação instancia os objetos necessários para exibir e tratar os componentes.
 Fonte: O próprio autor.

Quando a aplicação for finalizada, o sistema irá remover todos os componentes utilizados e enviará um alerta para a máquina virtual indicando que a aplicação está finalizada.

```
public void destroyXlet(boolean arg0) throws XletStateChangeException {
    if (scene != null) {
        scene.setVisible(false);
        scene.removeAll();
        scene = null;
    }
    context.notifyDestroyed();
}
```

Figura 18: Método executado no momento em que o sistema é finalizado, no caso, quando é solicitado que o middleware seja finalizado.
 Fonte: O próprio autor.

Para a interação com o usuário foi valorizado a implementação de tratamento de eventos, relacionando ao controle remoto, no caso o método KeyPressed, na qual será criado os textos e opções de votação.

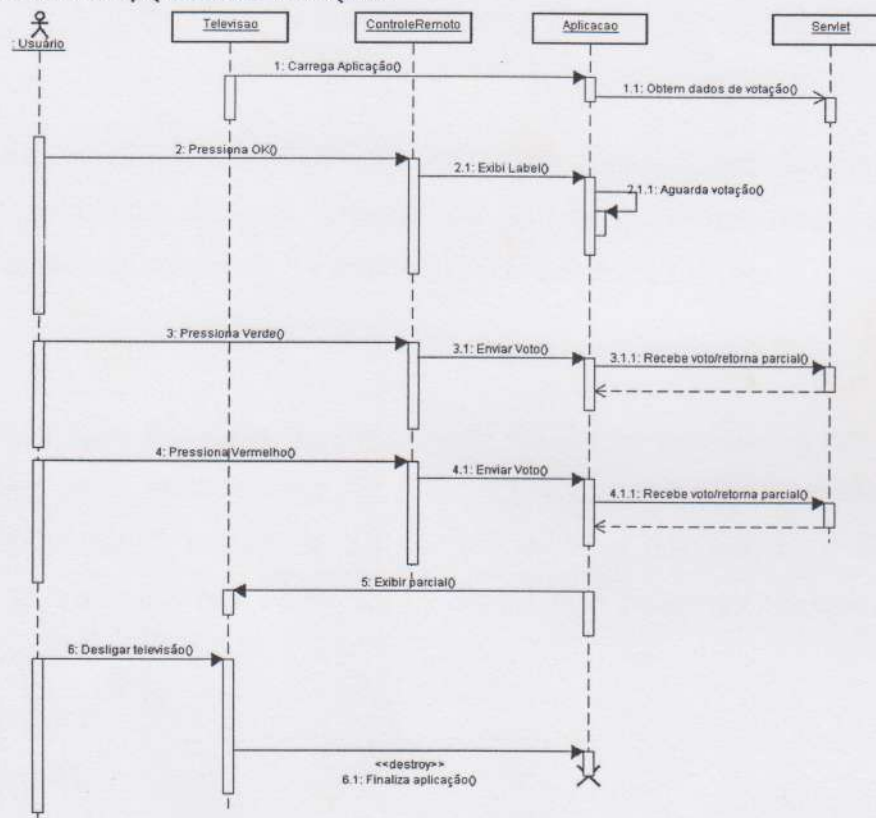


Figura 19: Diagrama de seqüência: Usuário interagindo com o Sistema.
Fonte: O próprio autor.

A televisão digital é algo que estará sendo implementado em todo o Brasil em breve. O estudo desta monografia é validar todo o sistema enquanto estiverem sendo desenvolvidos para futuros desenvolvedores. Com o SBTVD irão surgir dois tipos de tecnologias para permitir a interação com o usuário, no caso, a Ginga-NCL e o Ginga-J.

No realizar estudos em ambas tecnologias, percebeu-se que o material relacionado ao Ginga-NCL é de mais fácil acesso e compreensão. Assim esse padrão foi adotado com o intuito de permitir que qualquer usuário possa criar conteúdo interativo. Além disso, a proposta de utilização da linguagem LUA, é criar conteúdo interativo, rápido e ágil, que não consuma muito processamento do equipamento que está sendo utilizado.

No caso da linguagem Java, com a API voltada para televisão digital, percebeu-se que a mesma ainda tem que amadurecer no que diz respeito ao material didático. Quanto a material necessário para desenvolvimento, a mesma já proporciona algumas facilidades.

Para a criação da aplicação, foi utilizado o conhecimento adquirido. Relacionado a criação de aplicação de cliente, criação de servidor e comunicação de baixo nível, foi possível desenvolver o sistema na maioria dos dispositivos móveis.

Na aplicação proposta, foi perceptível a falta de conteúdo ainda relativo a criação de aplicações Java, voltado para TV. Com o pouco que foi encontrado, foi possível criar uma aplicação que é capaz de se comunicar com um servidor e trocar informações, mostrando assim que o real intuito da TV digital, que é permitir interatividade.

5. CONCLUSÃO

A televisão digital é algo que estará sendo implantado em todo o Brasil em breve. O intuito desta monografia é valorizar, todo o sistema enquanto *middleware*, e dar um norte para futuros desenvolvedores. Com o SBTVD irão surgir dois tipos de tecnologias para permitir a interação com o usuário, no caso, a Ginga-NCL e o Ginga-J.

Ao realizar estudos em ambas tecnologias, percebeu-se que o material relacionado ao Ginga-NCL é de mais fácil acesso e compreensão, afinal esse padrão foi adotado com o intuito de permitir que qualquer usuário possa criar conteúdo interativo. Além disso, a proposta de utilização da linguagem LUA, é criar conteúdo interativo, rápido e ágil, que não consuma muito processamento do equipamento que estará renderizando a mesma.

No caso da linguagem Java, com a API voltada para televisão digital, percebeu-se que a mesma ainda tem que amadurecer no que diz respeito ao matéria didático. Quanto a material necessário para desenvolvimento, a mesma já proporciona algumas facilidades.

Para a criação da aplicação, foi utilizado o conhecimento adquirido. Relacionado a criação de aplicação desktop, criação de *servlet* e comunicação de baixo nível, foi utilizado o conteúdo adquirido na matéria de dispositivos móveis.

Na aplicação proposta, foi perceptível a falta de conteúdo ainda relativo à criação de aplicações Java, voltado para TV. Com o pouco que foi encontrado, foi possível criar uma aplicação que é capaz de se comunicar com um *servlet* e trocar informações, mostrando assim qual o real intuito da TV digital, que é permitir interatividade.

REFERÊNCIA

- ABNT. **TVDIGITAL**. Disponível em: <<http://www.abnt.org.br/tvdigital/TVDIGITAL.html>>. Acesso: 16/11/2008.
- ACME HOW TO.COM. **What is HDTV? EDTV? SDTV?**. Disponível em: <<http://www.acmehowto.com/howto/hometheater/what-is-hdtv.php>>. Acesso: 14/11/2008.
- ANATEL. **Agência Nacional de Telecomunicação**. Disponível em: <<http://www.anatel.gov.br>>. Acesso: 14/11/2008.
- BOCC. **Biblioteca on-line de ciência da comunicação**. Disponível em: <http://www.bocc.ubi.pt/_listas/tematica.php?codtema=19>. Acesso: 14/11/2008.
- FÓRUM DO SISTEMA BRASILEIRO DE TV DIGITAL TERRESTRE. **Fórum SBTVD**. Disponível em: <<http://www.forumsbtvd.org.br/>>. Acesso: 14/11/2008.
- FÓRUM NACIONAL PELA DEMOCRATIZAÇÃO DA COMUNICAÇÃO. **TV DIGITAL: ROYALTIES AMEAÇAM LANÇAMENTO OFICIAL DO GINGA**. Disponível em: <http://www.fndc.org.br/internas.php?p=noticias&cont_key=232655>. Acesso: 18/11/2008.
- IDGNOW!. **A evolução da TV**. Disponível em: <<http://idgnow.uol.com.br/galerias/idgphotoalbum.2006-03-15.5742262351/>>. Acesso: 14/11/2008.
- ISTO É DINHEIRO. **Padrão Brasileiro**. Disponível em: <http://www.terra.com.br/istoedinheiro/especiais/inclusao_digital/padrao_brasileiro.htm>. Acesso: 23/11/2008.
- LABORATÓRIO DE APLICAÇÃO DE VÍDEOS DIGITAIS. **GINGA-J**. Disponível em: <<http://www.lavid.ufpb.br/>>. Acesso: 14/11/2008.
- LAMBDA, Maxwell. **TV DIGITAL**. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/cgi-bin/PRG_0599.EXE/7623_3.PDF?NrOcoSis=22028&CdLinPrg=pt>. Acessado em: 15/11/2008.
- MHP. **MHP 1.2 and GEM 1.2 Especification**. Disponível em: <<http://www.mhp.org/mhpgem12.htm>>. Acesso: 15/11/2008.
- PORTAL DO SOFTWARE PÚBLICO BRASILEIRO. **GINGA-NCL**. Disponível em: <<http://www.softwarepublico.gov.br/>>. Acesso: 17/11/2008.
- REVISTA ÉPOCA. **TV Digital: Relatório da Anatel está disponível ao público**. Disponível em: <<http://revistaepoca.globo.com/Revista/Epoca/0,,EDG49838-6010,00-TV+DIGITAL+RELATORIO+DA+ANATEL+ESTA+DISPONIVEL+AO+PUBLICO.html>>. Acesso: 16/11/2008.
- SOARES, Adriano. Uma especificação de Desenvolvimento de Serviços para Televisão Digital Interativa. Universidade Federal de Pernambuco: 2005.
- SUN MICROSYSTEMS. **API JAVA TV**. Disponível em: <<http://java.sun.com/javame/technology/javatv/>>. Acesso: 14/11/2008.
- TORRES, Gabriel. **TV Digital**. Disponível em: <<http://www.clubedohardware.com.br/artigos/1010>>. Acessado em: 14/11/2008.
- TUDO SOBRE TV. **TELEVISÃO**. Disponível em: <<http://www.tudosobretv.com.br/>>. Acesso: 22/11/2008.
- TV WITHOUT BORDERS. **The Java TV Tutorial**. Disponível em: <<http://www.interactivetvweb.org/tutorials/javatv>>. Acesso: 15/11/2008.
- VITÓRIA, Ana. Programação Educativa destinada à televisão Interativa. Universidade Federal de São Carlos: 2001.

XLETVIEW. **xletView**. Disponível em: <<http://www.xletview.org/>>. Acesso: 15/11/2008.

ANEXO A – CODIFICAÇÃO DO SISTEMA DE VOTAÇÃO ON-LINE, ATRAVÉS DA INERATIVIDADE DA TV DIGITAL

ANEXO

ANEXO A – CODIFICAÇÃO DO SISTEMA DE VOTAÇÃO ON-LINE, ATRAVÉS DA INTERATIVIDADE DA TV DIGITAL.

```
1 import java.awt.Color;
2 import java.awt.Font;
3 import java.awt.event.KeyEvent;
4 import java.awt.event.KeyListener;
5 import java.net.HttpURLConnection;
6 import java.net.URL;
7 import java.util.Scanner;
8 import javax.swing.JList;
9 import javax.swing.JListContext;
10 import javax.swing.JListStateChangeListener;
11 import org.havi.ui.HDefaultTextLayoutManager;
12 import org.havi.ui.HScene;
13 import org.havi.ui.HSceneFactory;
14 import org.havi.ui.HScreen;
15 import org.havi.ui.HStaticText;
```

21 //

22 *

23 * @author Romulo

24 *

```
25
26 public class Main implements JList, KeyListener {
27
28     private JListContext context;
29     private HScene scene;
30     private HStaticText label1, binVermelho, binVerde, resultado;
31     private String[] opcoes;
32     private final static String URL = "http://localhost:8080/MonografiaWeb/";
33
34     public void initJList(JListContext context) throws JListStateChangeListener {
35         this.context = context;
36         opcoes = obterVoto();
37
38     }
39
40     public void startJList() throws JListStateChangeListener {
41         HSceneFactory hsceneFactory = HSceneFactory.getInstance();
42         scene = hsceneFactory.getFullScreenScene(
43             HScreen.getDefaultHScreen().getDefaultHGraphicsDevice());
44         scene.setBounds(0, 0, 800, 900);
45         scene.setLayout(null);
46         scene.addKeyListener(this);
47         scene.setVisible(true);
48         scene.requestFocus();
49     }
50 }
```

```

1
2
3
4
5
6 import java.awt.Color;
7 import java.awt.Font;
8 import java.awt.event.KeyEvent;
9 import java.awt.event.KeyListener;
10 import java.net.HttpURLConnection;
11 import java.net.URL;
12 import java.util.Scanner;
13 import javax.tv.xlet.Xlet;
14 import javax.tv.xlet.XletContext;
15 import javax.tv.xlet.XletStateChangeException;
16 import org.havi.ui.HDefaultTextLayoutManager;
17 import org.havi.ui.HScene;
18 import org.havi.ui.HSceneFactory;
19 import org.havi.ui.HScreen;
20 import org.havi.ui.HStaticText;
21
22 /**
23  *
24  * @author Romulo
25  */
26 public class Main implements Xlet, KeyListener {
27
28     private XletContext context;
29     private HScene scene;
30     private HStaticText label1, btnVermelho, btnVerde, resultado;
31     private String[] opcoes;
32     private final static String URL = "http://localhost:8080/MonografiaWeb/";
33
34     public void initXlet(XletContext context) throws XletStateChangeException {
35         this.context = context;
36         opcoes = obterVoto();
37     }
38 }
39
40 public void startXlet() throws XletStateChangeException {
41     HSceneFactory hsceneFactory = HSceneFactory.getInstance();
42     scene = hsceneFactory.getFullScreenScene(
43         HScreen.getDefaultHScreen().getDefaultHGraphicsDevice());
44     scene.setBounds(0, 0, 800, 600);
45     scene.setLayout(null);
46     scene.addKeyListener(this);
47     scene.setVisible(true);
48     scene.requestFocus();

```

```

49  }
50
51  public void pauseXlet() {
52  }
53
54  public void destroyXlet(boolean arg0) throws XletStateChangeException {
55      if (scene != null) {
56          scene.setVisible(false);
57          scene.removeAll();
58          scene = null;
59      }
60      context.notifyDestroyed();
61  }
62
63  public void keyTyped(KeyEvent e) {
64  }
65
66  public void keyPressed(KeyEvent e) {
67      int codigo = e.getKeyCode();
68      scene.removeAll();
69
70      if (codigo == 10) {
71          label1 = new HStaticText("Quem você deseja que saia?", 200, 400,
72                                  300, 30, new Font("Verdana", Font.ITALIC, 20), Color.blue,
73                                  Color.white, new HDefaultTextLayoutManager());
74          label1.repaint();
75
76          btnVermelho = new HStaticText(opcoes[0], 220, 440,
77                                       100, 20, new Font("Verdana", Font.BOLD, 17), Color.black,
78                                       Color.red, new HDefaultTextLayoutManager());
79          btnVermelho.repaint();
80
81          btnVerde = new HStaticText(opcoes[1], 340, 440,
82                                    100, 20, new Font("Verdana", Font.BOLD, 17), Color.black,
83                                    Color.green, new HDefaultTextLayoutManager());
84          btnVerde.repaint();
85
86          scene.add(label1);
87          scene.add(btnVerde);
88          scene.add(btnVermelho);
89          scene.repaint();
90      } else if (codigo == 403) {
91          String res = votar(opcoes[0]);
92          resultado = new HStaticText(res, 100, 470,
93                                     600, 30, new Font("Verdana", Font.ITALIC, 20), Color.red,
94                                     Color.white, new HDefaultTextLayoutManager());
95          resultado.repaint();
96
97          scene.add(resultado);
98          scene.repaint();

```

```

99     } else if (codigo == 404) {
100         String res = votar(opcoes[1]);
101         resultado = new HStaticText(res, 100, 470,
102             600, 30, new Font("Verdana", Font.ITALIC, 20), Color.green,
103             Color.white, new HDefaultTextLayoutManager());
104         resultado.repaint();
105
106         scene.add(resultado);
107         scene.repaint();
108     } else {
109         scene.repaint();
110     }
111 }
112
113 public void keyReleased(KeyEvent e) {
114 }
115
116 private String[] obterVoto() {
117     String resultado[] = new String[2];
118     try {
119         URL url = new URL(URL + "ListaParticipante");
120         HttpURLConnection con = (HttpURLConnection) url.openConnection();
121         con.setConnectTimeout(10);
122         con.connect();
123         Scanner sc = new Scanner(con.getInputStream());
124         if (sc.hasNextLine()) {
125             resultado = sc.nextLine().split("\\|");
126         }
127     } catch (Exception ex) {
128         ex.printStackTrace();
129     }
130     return resultado;
131 }
132
133 private String votar(String voto) {
134     String resultado = "ERRO";
135     try {
136         URL url = new URL(URL + "Votar?voto=" + voto);
137         HttpURLConnection con = (HttpURLConnection) url.openConnection();
138         con.setConnectTimeout(10);
139         con.connect();
140
141         Scanner sc = new Scanner(con.getInputStream());
142         if (sc.hasNextLine()) {
143             resultado = sc.nextLine();
144         }
145     } catch (Exception ex) {
146         ex.printStackTrace();
147     }
148 }

```

```
149
150     return resultado;
151 }
152 }
153
154
```