

UTILIZAÇÃO DA API JAVATV PARA O DESENVOLVIMENTO DE
APLICAÇÕES PARA A TV DIGITAL INTERATIVA

ELI CANDIDO JUNIOR

UTILIZAÇÃO DA API JAVATV PARA O DESENVOLVIMENTO DE
APLICAÇÕES PARA A TV DIGITAL INTERATIVA

ELI CANDIDO JUNIOR

Monografia apresentada a Pós-Reitoria de Pesquisa e Pós-Graduação, da Universidade do Oeste Paulista, como parte dos requisitos para obtenção do título de Especialista em Desenvolvimento de Sistemas para Ambientes Web baseados em Tecnologia Java.

Orientador: MSc. Francisco Assis da Silva

004

Candido Junior, Eli.

Utilização da API JavaTV para o desenvolvimento de aplicações para TV Digital Interativa / Eli Candido Junior. – Presidente Prudente: [s. n.], 2008.
54 f. : il.

Monografia (Especialização em Desenvolvimento de Sistemas para Ambientes baseados em Tecnologia Java) – Universidade do Oeste Paulista – UNOESTE: Presidente Prudente – SP, 2008.

Bibliografia

1. Televisão Digital Interativa. 2. JavaTV. 3. Xlet. I. Autor. II. Título.

AGRADECIMENTOS

Agradeço principalmente a Deus e a todas as pessoas que me apoiaram para concluir mais essa etapa em minha vida.

DEDICATÓRIA

Dedico este trabalho para minha família, minha namorada, meu orientador e demais professores, os quais sempre acreditaram em meus esforços.

“[...] Use os obstáculos para abrir as janelas da inteligência [...]”

RESUMO

UTILIZAÇÃO DA API JAVATV PARA O DESENVOLVIMENTO DE APLICAÇÕES PARA A TV DIGITAL INTERATIVA

A televisão é uma das principais fontes de informação, entretenimento e cultura. A mais nova mudança é a digitalização do sinal. Com a transmissão digital além de uma melhora na qualidade da imagem e do som, possibilita ao telespectador interagir com o conteúdo transmitido. Especificações como API JavaTV fornecem suporte para o desenvolvimento de aplicações para a Televisão Digital Interativa. Este projeto apresenta os conceitos e definições da tecnologia da TV Digital, bem como introduz os aspectos relevantes ao desenvolvimento de uma aplicação para a TV Digital, os Xlets. Para isso, foi desenvolvida uma aplicação para TV Digital, onde o telespectador manifesta sua opinião sobre um determinado tema apresentado em um programa de televisão, utilizando o controle remoto do televisor. Para simular uma rede de transmissão de TV Digital e um aparelho televisor digital em um micro-computador convencional foi utilizado um software que emula o *middleware* do *set-top-box*, o XleTView. Durante o desenvolvimento desse projeto, além de descrever os métodos da API JavaTV e o conceito de um Xlet, também são utilizados recursos XML (para a gravação do voto) e *sockets* (que por intermédio do canal de retorno realizam a comunicação entre a TV Digital e a Emissora de Televisão).

Palavras-Chave: Televisão Digital Interativa, JavaTV, Xlet.

ABSTRACT

USING JAVATV API TO DEVELOPMENT APPLICATIONS TO INTERACTIVE DIGITAL TV

The television is one of the source main of information, entertainment and culture. The newest change is the digital signal. With the digital transmission besides an improvement image and sound quality, it makes possible the viewer to interact with the transmitted content. Specifications as JavaTV API provide support development of applications for Interactive Digital Television. This project presents concepts and definitions of the Digital TV technology, as well introduces the important aspects to development application for Digital TV, the Xlets. An application was developed for Digital TV, where the viewer manifests your opinion on a certain theme presented in a television program, using the remote control. To simulate Digital TV transmission and digital television in a conventional personal computer was used a software that emulates the set-top-box middleware, the XleTView. During the development this project, besides describing JavaTV API's methods and the Xlet concept, it is also used resources XML (for recording the vote) and sockets (that through the return channel make the communication between the Digital TV and the television channel).

Key-Words: Interactive Digital Television, JavaTV, Xlet.

LISTA DE FIGURAS

Figura 1 - Comparação entre a transmissão do sinal digital x analógico.	14
Figura 2 - Modelo de um sistema de televisão digital interativa.	15
Figura 3 - Componentes que definem um sistema de televisão digital.	16
Figura 4 - Modelo de um Set-top-box.	17
Figura 5 - Opções de padrões para um sistema de televisão digital interativa.	18
Figura 6 - Interatividade entre o telespectador e a Televisão Digital.	22
Figura 7 - Arquitetura de um <i>set-top-box</i> genérico e um set-top-box utilizando a tecnologia Java.	25
Figura 8 - Ciclo de vida de um Xlet.	29
Figura 9 - Interface javax.tv.Xlet.	30
Figura 10 - Interface javax.xlet.XletContext.	31
Figura 11 - A arquitetura da API JavaTV executada em um <i>set-top-box</i> e em um PC.	32
Figura 12 - Interface do XletView.	34
Figura 13 - Arquivo de lote (xletview.bat).	35
Figura 14 - Execução do Console pelo XleTView.	36
Figura 15 - Diagrama de funcionamento em um ambiente real.	38
Figura 16 - Interface da aplicação TVoto.	39
Figura 17 - Aplicação iniciada.	40
Figura 18 - Telespectador escolhe sua alternativa.	41
Figura 19 - Momento em que o telespectador finaliza a votação.	42
Figura 20 - Estrutura do Projeto.	43
Figura 21 – Diagrama de Classes.	46
Figura 22 - Classe xleTVoto.	47
Figura 23 - Classe ContainerVoto.	47

Figura 24 - Classe ContainerFim.....	48
Figura 25 - Classe beanVoto.....	49
Figura 26 - Classe GravaVoto.....	49
Figura 27 - Classe enviaVoto.	50
Figura 28 - Arquivo XML que é transmitido.	51
Figura 29 - Servidor aguardando envio do arquivo XML.	51

LISTA DE SIGLAS

ADSL - *Asymmetric Digital Subscriber Line*
API - *Application Programming Interface*
ARIB - *Association of Radio Industries and Businesses*
ATSC - *Advanced Television Systems Committee*
AWT - *Abstract Window Toolkit*
BML - *Broadcast Markup Language*
DASE - *Digital Applications Software Environment*
DVB - *Digital Video Broadcasting*
GPL - *General Public License*
GPRS - *General Packet Radio Service*
HDTV - *High Definition Television*
IDE - *Integrated Development Environment*
ISDB - *Integrated Services Digital Broadcasting*
ISDTV - *International System for Digital TV*
IP - *Internet Protocol*
JMF - *Java Media Framework*
JVM - *Java Virtual Machine*
MHP - *Multimedia Home Platform*
MPEG - *Moving Picture Experts Group*
NCL - *Nested Context Language*
TCP - *Transmission Control Protocol*
TV - *Televisão*
TVDI - *TV Digital Interativa*
UHF - *Ultra High Frequency*
VHF - *Very High Frequency*
XML - *Extensible Markup Language*

SUMÁRIO

1	INTRODUÇÃO.....	12
2	SISTEMA DE TV DIGITAL	14
2.1	Componentes da TV Digital Interativa	15
2.2	Padrões para TV digital interativa	18
2.2.1	Digital Video Broadcasting – DVB	18
2.2.2	Advanced Television Systems Committee – ATSC	19
2.2.3	Integrated Services Digital Broadcasting – ISDB	19
2.2.4	International System for Digital TV – ISDTV	20
2.3	Tipos de Interatividade.....	21
2.4	TV Digital no Brasil	22
3	JAVATV.....	24
3.1	Conceitos.....	25
3.2	Pacotes.....	27
3.3	Xlet.....	28
4	EMULADOR	32
4.1	XletView	33
5	APLICAÇÃO PARA TV DIGITAL DESENVOLVIDA NESTE TRABALHO	37
5.1	TVoto	38
5.2	Configuração da aplicação	42
5.3	Implementação do Xlet	44
5.4	Construção da interface gráfica	45
5.5	Classes Java.....	45
5.6	XStream (XML)	50
5.7	Socket.....	51
6	CONSIDERAÇÕES FINAIS.....	52
	REFERÊNCIAS BIBLIOGRÁFICAS.....	53

1 INTRODUÇÃO

A transmissão digital traz de imediato aos telespectadores uma imagem e som de maior qualidade, sem chuviscos e sombras, com a possibilidade de tela larga (*widescreen*) e recursos de *zoom*. Em um único canal de TV há a possibilidade de ter vários sub-canais sendo transmitidos ao mesmo tempo (BITTENCOURT, 2006).

O surgimento da televisão digital além de uma melhora significativa na qualidade do vídeo e áudio, trouxe consigo a possibilidade de interação do telespectador com conteúdo transmitido através da Televisão Digital Interativa. O telespectador não será mais tratado como uma figura passiva a partir de então.

Especificações como a API (*Application Programming Interface*) JavaTV da *Sun Microsystems*, surgiram com o intuito de fornecer suporte aos desenvolvedores desse ramo. Essa API introduziu o conceito de Xlet, uma nova abstração para as televisões digitais, equivalentes ao de um *applet* Java para um *browser* na Internet.

O objetivo deste projeto é apresentar uma sinopse da tecnologia de TV Digital e também introduzir aspectos do desenvolvimento de aplicações para a TV Digital, os Xlets. Sendo sua implementação na linguagem Java.

Este projeto se destina a demonstrar como se desenvolve aplicações para a TV Digital Interativa (TVDI), se especializando na API JavaTV desenvolvida pela *Sun Microsystems* e no emulador para simular uma TV Digital em um *desktop*, chamado de XleTView. Serão apresentadas diversas características e ferramentas de desenvolvimento baseadas em software livre voltadas para as especificações de *middleware* para TVDI adotadas na Europa (MHP – *Multimedia Home Platform*), nos Estados Unidos (DASE – *Digital Applications Software Environment*), no Japão (ARIB – *Association of Radio Industries and Businesses*) e focando o padrão adotado no Brasil (Ginga).

Neste projeto foi criada uma aplicação para a TVDI denominada de TVoto, foi desenvolvida em Java utilizando a API JavaTV, e pode ser executada no *middleware* da TV Digital. A TVoto mostra o que a TVDI poderá proporcionar ao telespectador, onde diretamente do aparelho de TV poderá manifestar seu voto, no caso um programa de TV é exibido, e o telespectador é questionado qual é a sua

opinião em determinado assunto. Para o desenvolvimento do sistema, além de utilizar a API JavaTV, foi utilizada a API XStream, que manipula XML (*Extensible Markup Language*); recurso XML foi utilizado para o envio dos dados. Para simular a conexão do canal de retorno entre o TVDI e a Emissora e estabelecer essa comunicação foi utilizado o recurso de *sockets* (seção 5.6).

2 SISTEMA DE TV DIGITAL

Um sistema de TV digital interativa pode ser decomposto em três partes principais: (i) um difusor, responsável por prover o conteúdo a ser transmitido e dar suporte às interações dos telespectadores; (ii) um receptor, que recebe o conteúdo e oferece a possibilidade do telespectador reagir ou interagir com o difusor; e (iii) um meio de difusão, que habilita a comunicação entre o difusor e o receptor (BECKER, 2004).

É um conjunto de definições que viabilizam a construção de dispositivos para a transmissão e recepção dos sinais da TV Digital. Devido à transmissão do vídeo ser efetuada como um fluxo de *bits*, é possível transmitir uma maior quantidade de dados multiplexados (comparado com o sistema analógico), e existindo um canal de retorno (conexão à Internet, por exemplo), o receptor pode enviar dados para a difusora, caracterizando assim a interatividade.

A figura 1 ilustra a principal diferença entre a transmissão do sinal digital e a transmissão do sinal analógico.



Figura 1 - Comparação entre a transmissão do sinal digital x analógico.

A maior vantagem da transmissão em sistema digital é a conservação da qualidade do sinal. O modo de modulação e compressão digital para enviar vídeo, áudio e sinais de dados aos aparelhos proporciona uma transmissão e

recepção de maior quantidade de conteúdo em uma mesma frequência (canal). A figura 2 ilustra um modelo de um sistema de televisão digital interativa.

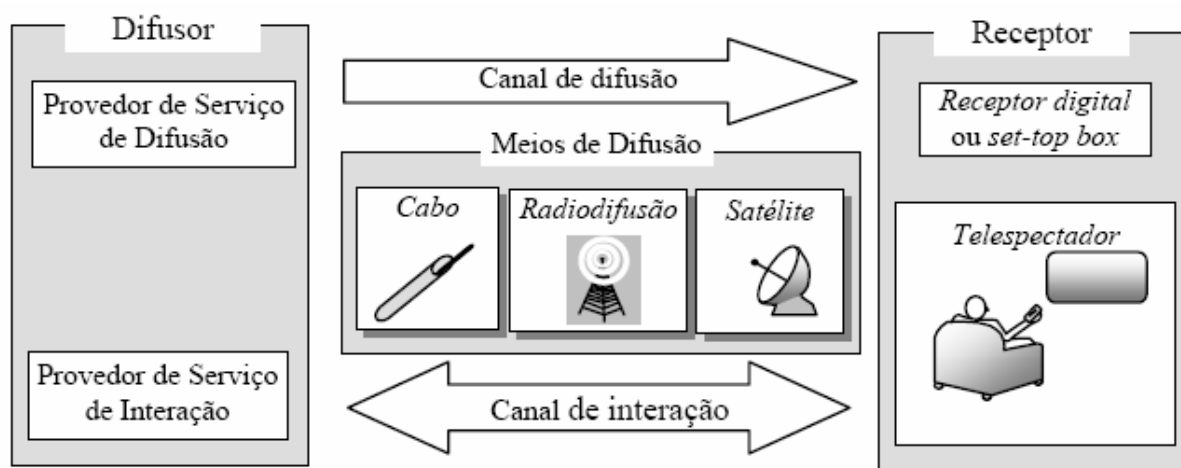


Figura 2 - Modelo de um sistema de televisão digital interativa.

2.1 Componentes da TV Digital Interativa

Um conjunto de tecnologias foi criado e adaptado para possibilitar as execuções de aplicações que permitem interatividade com o uso da televisão. Para tanto, um grupo de componentes atuam em conjunto para viabilizar a transmissão, recepção e apresentação dos conteúdos audiovisuais e a execução dos programas interativos em aparelhos de televisão.

Os componentes de um sistema de TV digital podem ser classificados em cinco conjuntos de padrões: modulação, codificação, transporte, *middleware* e aplicação. Basicamente, um padrão é definido pelos esquemas de compressão e codificação de áudio e vídeo, pela camada de abstração de software do *middleware* e o esquema de multiplexação e modulação de dados (POZZER, 2003). Esses esquemas são escolhidos para atender a determinadas características de um padrão adotado, onde pode privilegiar certos quesitos, como alta definição, interatividade, recepção móvel, dentre outros.

A transmissão de uma aplicação de TV Digital é feita da seguinte forma: cada conteúdo (áudio, vídeo e aplicação) é codificado e gera um fluxo

chamado elementar. No caso das aplicações, o mecanismo de transmissão é conhecido como carrossel de dados. Nele o conteúdo das aplicações é dividido em módulos que são constantemente enviados, em ciclo, permitindo que mude para o canal de cada módulo. Os fluxos elementares são agrupados de forma a criar programas de TV que também são agrupados em canais de TV lógicos (JUCÁ, 2006). Quando o sinal chega ao receptor, o processo inverso acontece. O Sinal é demodulado e demultiplexado para separar todos os fluxos. O aparelho multiplexador da figura 3 realiza a modulação do sinal de TV e é mais conhecido como o *set-top-box*.

A figura 3 ilustra os componentes que definem um sistema de televisão digital.

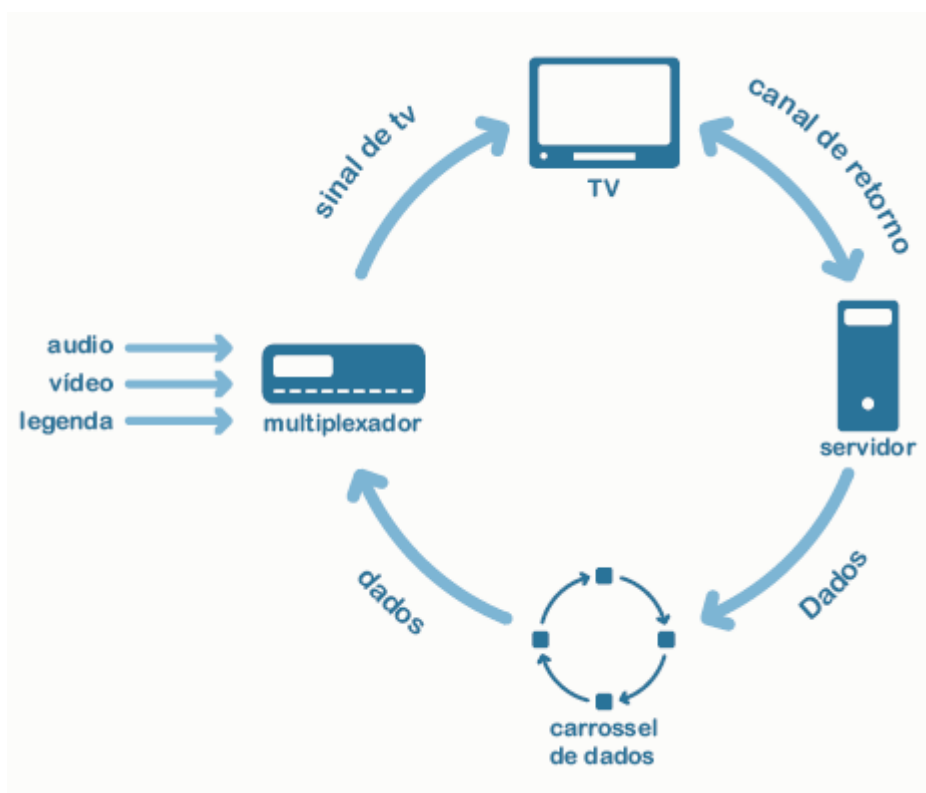


Figura 3 - Componentes que definem um sistema de televisão digital.
Fonte: (JUCÁ, 2006).

O componente mais importante desse conjunto tendo em vista uma aplicação para a TV Digital Interativa (TVDI) é a camada de software denominada *middleware*. Um *middleware* é de forma simplificada, uma camada de software que

une dois sistemas separados. O *middleware* estabelece a comunicação entre o hardware e o sistema operacional do *set-top-box* às aplicações.

O *set-top-box* (figura 4) é um equipamento responsável pela conexão entre um televisor e uma fonte externa de sinal (cabo *ethernet*, antena de satélite, cabo coaxial, linha telefônica ou conexão com uma antena VHF/UHF), e transforma este sinal em conteúdo (vídeo, áudio, internet, interatividade, jogos, entre outros), que é apresentado na tela.



Figura 4 - Modelo de um *set-top-box*.

O *set-top-box* pode ser entendido como um computador adaptado para as necessidades do ambiente televisivo, possuindo processador, memória, sistema operacional, etc. Nele, é encontrado o *middleware*, que é uma instância de software de suma importância para o desenvolvimento em TV Digital (BATISTA, 2007).

Os aparelhos convencionais de TV necessitam da instalação de um *set-top-box* para receber o sinal digital, já os mais modernos, já vêm com o *set-top-box* embutidos no próprio aparelho.

2.2 Padrões para TV digital interativa

Atualmente existem três grandes sistemas de televisão digital aberta. O Sistema Europeu: DVB (*Digital Video Broadcasting*) com o *middleware* MHP (*Multimedia Home Platform*); O Sistema Americano: ATSC (*Advanced Television Systems Committee*) com o seu *middleware* DASE (*Digital Applications Software Environment*); O Sistema Japonês: ISBD (*Integrated Services Digital Broadcasting*) com o *middleware* ARIB (*Association of Radio Industries and Businesses*) (BECKER, 2004).

A figura 5 mostra as opções de padrões para um sistema de televisão digital interativa.

No entanto, o Brasil praticamente passa a definir um novo padrão, baseado no sistema japonês, porém acrescida de tecnologias desenvolvidas nas pesquisas das universidades brasileiras, o ISDB-TB (*Integrated Services Digital Broadcasting - Terrestrial*) com o *middleware* Ginga.

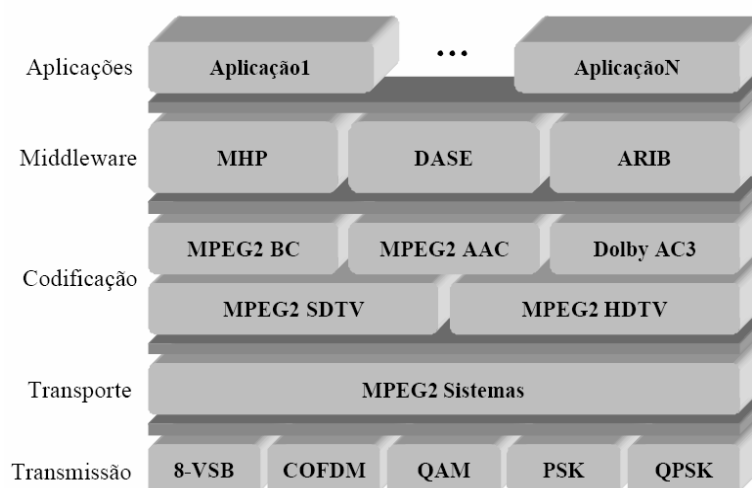


Figura 5 - Opções de padrões para um sistema de televisão digital interativa.

2.2.1 Digital Video Broadcasting – DVB

Digital Video Broadcasting (DVB) o padrão de televisão digital europeu que definiu três padrões para a tecnologia de transmissão: via rádio-difusão terrestre

(DVB-T), via satélite (DVB-S) e via cabo (DVB-C). O padrão DVB privilegia a multi-programação e interatividade.

O *middleware* do padrão DVB é o *Multimedia Home Platform* (MHP), ele é baseado em tecnologias como JavaTV, DAViC e HAVi, o qual foi completamente desenvolvido em Java (MHP, 2007). Ele está voltado para interatividade, além do suporte à qualidade de imagem de alta definição: HDTV (*High Definition Television*).

O MHP permite a transmissão para dispositivos móveis, mas para isso é necessário uma banda extra, o DVB-H.

2.2.2 Advanced Television Systems Committee – ATSC

O *Advanced Television Systems Committee* (ATSC) é o padrão de transmissão digital norte-americano, o qual definiu dois padrões técnicos de transmissão: via cabo (ATSC-C) e via rádio-difusão (ATSC-T), onde o modelo de transmissão mais difundido é o via cabo.

Esse padrão privilegia a transmissão de alta qualidade (HDTV), no entanto não oferece suporte a transmissão de televisão digital para dispositivos móveis.

O *Digital Applications Software Environment* (DASE) foi desenvolvido como um padrão norte-americano para a camada de *middleware* em *set-top-boxes* de TVs digitais. O DASE adota uma Máquina Virtual Java como mecanismo que facilita a execução de aplicações interativas baseadas em Java, mas também permite o uso de linguagens declarativas, usadas na web, como HTML e *JavaScript* (ATSC, 2007).

2.2.3 Integrated Services Digital Broadcasting – ISDB

O *Integrated Services Digital Broadcasting* (ISDB) é o padrão japonês para transmissão de televisão digital, oferece suporte às transmissões terrestres

(ISDB-T e ISDB-TSB ou *Terrestrial Sound Broadcasting*), via satélite (ISDB-S) e via cabo (ISDB-C).

O grande diferencial deste padrão é a possibilidade de transmitir sinal de televisão para aparelhos móveis como celulares e PDAs dentro da faixa usada para televisão terrestre sem a necessidade da operadora de telefonia, porém, isso só é possível em *broadcast*.

O *middleware* definido para o padrão japonês é padronizado pela organização ARIB (*Association of Radio Industries and Businesses*). Esse *middleware* é formado por alguns padrões, como o ARIB STD-B24 - *Data Coding and Transmission Specification for Digital Broadcasting*, que define uma linguagem declarativa (BML – *Broadcast Markup Language*) baseada em XML (*Extensible Markup Language*), e usada para especificação de serviços multimídia para TV digital. Outra especificação do *middleware* é o ARIB-STDB23 (*Application Execution Engine Platform for Digital Broadcasting*), baseada no padrão europeu (MHP), e que permite a execução de aplicações interativas baseadas em Java (ARIB, 2007).

2.2.4 International System for Digital TV – ISDTV

O sistema de televisão digital especificado no Brasil possui três características que o diferencia dos demais. Ele é uma modificação da modulação japonesa (ISBD), adotou o padrão MPEG4 para codificação de vídeo, o qual é uma evolução dos demais e possui um *middleware* desenvolvido no Brasil.

O *middleware* para ser utilizado no sistema brasileiro de TV Digital é o Ginga. Oferece uma série de facilidades para o desenvolvimento de conteúdo e aplicativos para TV Digital, entre elas a possibilidade desses conteúdos serem exibidos nos mais diferentes sistemas de recepção, independente do fabricante e tipo de receptor (TV, celular, PDAs, etc.) (GINGA, 2007).

Existem duas variações para o desenvolvimento de aplicações: declarativo (Ginga-NCL) e procedural (Ginga-J).

2.2.4.1 Ginga-NCL

As linguagens declarativas são mais intuitivas (de mais alto nível) e, por isso, mais fáceis de serem usadas, normalmente não exigindo um perito programador. Contudo, as linguagens declarativas têm de ser definidas com um foco específico. Quando o foco da aplicação não casa com o da linguagem, o uso de uma linguagem procedural não é apenas vantajoso, mas se faz necessário.

Para implementação utilizando linguagens declarativas, o Ginga oferece suporte à linguagem NCL (*Nested Context Language*).

2.2.4.2 Ginga-J

O uso de linguagens procedurais geralmente requer um perito em programação. No entanto, oferece um maior poder de implementação, uma vasta gama de possibilidades de implementações.

O Ginga oferece suporte ao desenvolvimento de aplicações em Java, por meio do Ginga-J, utilizando API JavaTV.

O Ginga-J é compatível com o *middleware* de outros sistemas de TV Digital como o MHP, pois se baseia no padrão DVB.

2.3 Tipos de Interatividade

Um dos aspectos mais empolgantes da TV Digital é a interatividade (exemplo na figura 6), no entanto, para que isso seja possível, é necessário a existência de um canal de interação (Canal de Retorno), que é um meio onde o aparelho de TV se comunica com a Emissora do Sinal Digital. Os canais de retorno podem ser via cabo, linhas telefônicas fixas (ADSL) e celulares (GPRS), onde na maioria das vezes é utilizada a comunicação TCP/IP.

Os tipos de interatividade:

- **Interatividade local:** não existe canal de retorno, ou seja, nenhuma informação pode ser enviada pelo receptor. Além da tradicional exibição de um canal televisivo, é possível lidar com

gráficos, imagens e textos transmitidos junto com a difusão do sinal;

- **Interatividade intermitente:** além das capacidades na interatividade local, existe um canal de retorno, porém este é unidirecional (respostas simples, mensagens SMS, votação);
- **Interatividade permanente:** tem todas as capacidades anteriores e o canal de retorno é bidirecional (navegação na Internet, *download* e *upload* de softwares, serviços de e-mails).



Figura 6 - Interatividade entre o telespectador e a Televisão Digital.

2.4 TV Digital no Brasil

A televisão digital no Brasil remete à implantação do sistema de transmissão digital que foi definida entre 2005 e 2007 de maneira significativa, apesar de muitas polêmicas quanto ao padrão adotado e alguns impasses que estão pendentes até os dias atuais.

A primeira transmissão oficial de sinal de TV Digital no Brasil aconteceu no dia 2 de dezembro de 2007 na cidade de São Paulo. A previsão do governo é que em 2009 todas as capitais do país já recebam o sinal digital. Em 2013 o sinal se

estenderá a todo território nacional. O governo deseja em 2016 encerrar definitivamente as transmissões de televisão por sinal analógico (DTV, 2007).

O sistema de televisão digital adotado no Brasil possui três características que o torna único no mundo. Foi realizada uma alteração da modulação japonesa. A modulação escolhida possibilita a transmissão simultânea de sinal digital tanto para aparelhos de TVs e *set-top-boxes* quanto para aparelhos móveis (celulares e PDAs). Isso é possível, pois a banda é dividida em 13 (treze) segmentos, sendo 12 (doze) reservados para transmissão “fixa” e 1 (um) segmento de banda destinado à transmissão móvel. A segunda característica é a utilização do padrão MPEG4 de codificação de vídeo, o qual é uma evolução dos demais sistemas de televisão do mundo. E finalmente, a utilização de um *middleware* nacional para o sistema brasileiro de televisão digital.

Porém essas definições foram apenas a primeira etapa da especificação do sistema brasileiro de televisão digital. Diversas instituições prosseguiram na finalização das especificações, dando origem a uma primeira versão do que ficou conhecido como ISDTV (*International System for Digital TV*) (JUCÁ, 2006).

3 JAVATV

A API (*Application Programming Interface*) JavaTV é uma extensão da plataforma Java criada pela *Sun Microsystems* com parceria da indústria de televisão digital. Ela está sendo amplamente adotada como um padrão para aplicações de televisão digital no mundo. Esta API foi projetada para oferecer acesso à funcionalidades para definição do modelo das aplicações, acesso a fluxo de áudio e vídeo, acesso condicional, acesso à informações de serviço, controle de troca do canal do receptor, controle dos gráficos na tela, dentre outras (SUN, 2007).

A API se situa entre a camada do sistema operacional e as aplicações interativas, pode-se definir JavaTV como um *middleware* de TV digital. Com ela, é possível o desenvolvimento de aplicações interativas independentes da tecnologia utilizada nos protocolos de transmissão, do sistema operacional dos *set-top-boxes* e da camada de hardware das redes de difusão, assim a API JavaTV funciona independente do padrão do sistema de televisão digital adotado, sendo ele ATSC, DVB, até mesmo o padrão Brasileiro (ISDTV), desde que, esse padrão suporte JavaTV.

As aplicações para a televisão digital interativa são geradas uma única vez, sendo compatível com todos os aparelhos que usam a Máquina Virtual Java (JVM – *Java Virtual Machine*), tornando assim independente do processador ou sistema operacional do receptor, isso é uma característica da linguagem Java, a portabilidade.

O ambiente de software de um *set-top-box* com a tecnologia JavaTV consiste basicamente de um ambiente de aplicações Java, a API JavaTV e outras APIs, e suas aplicações, além de um sistema operacional de tempo real (MONTEZ, 2004). Na figura 7 há uma comparação entre a arquitetura de um *set-top-box* genérico com uma utilizando a tecnologia JavaTV.

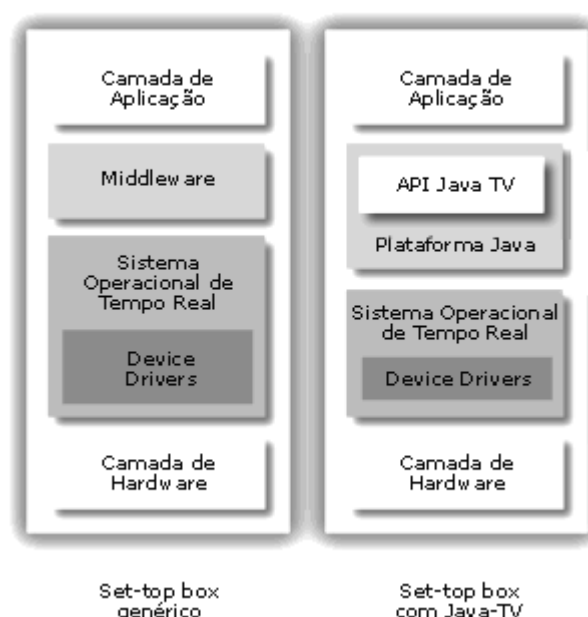


Figura 7 - Arquitetura de um *set-top-box* genérico e um *set-top-box* utilizando a tecnologia Java.

Fonte: (MONTEZ, 2004).

A camada de aplicação utiliza a API JavaTV e os pacotes Java da camada abaixo, onde todas as aplicações em Java são executadas sobre a Máquina Virtual Java (*Java Virtual Machine* - JVM).

Com a adoção dessa arquitetura, a JavaTV, é possível o programador abstrair-se dos detalhes de hardware e protocolos de comunicação dos *set-top-boxes*.

O Sistema Operacional de Tempo Real oferece um suporte em nível de sistema necessário para a camada Java, além de controlar o receptor através de uma coleção de *Device Drivers*.

3.1 Conceitos

Para uma melhor compreensão a respeito da API JavaTV, esta seção apresenta a definição de alguns conceitos fundamentais, que fazem parte da lógica de funcionamento de uma aplicação para a televisão digital interativa utilizando a API JavaTV.

O **Canal de Retorno** representa a possibilidade de criação de aplicações interativas e o acesso à internet. A conexão através do canal pode ocorrer de várias formas, via modens a cabo, Ethernet, etc.

Um **Serviço** pode ser definido como um canal de televisão composto por vários sub-componentes tais como *streams* de áudio, vídeo ou dados. Cada serviço possui um SI (*Service Information*) associado a ele e cada entrada do SI define uma propriedade do serviço.

Os **Componentes do Serviço** são *streams* elementares como de áudio e vídeo, uma aplicação Java ou outro tipo de dado que pode ser apresentado sem uma informação descritiva adicional. Um componente pode ser compartilhado por mais de um serviço.

Service Information (SI) descreve a estrutura do fluxo, além de conter informações relativas ao serviço, como por exemplo, um conteúdo descritivo, dados referentes aos horários de apresentação dos programas, a frequência do canal para possibilitar a sintonização no devido serviço, informações de demultiplexação, a linguagem utilizada, informações da rede, além de meta-informações como uma lista dos atores ou categorização do serviço.

Service Locator é a informação necessária que possibilita a sintonização do canal, análogo a uma URL. **SI Database** representa a base de dados que armazena o SI. **SI Manager** é um objeto *singleton*¹, pois só existe um no receptor de televisão digital. *SI Manager* é o ponto de acesso para a base de dados do SI e ele reporta qualquer alteração que ocorra na base. **SI Factory** é responsável em criar objetos do tipo *SI Manager*.

¹ *singleton*: é um padrão de projeto de software (*Design Pattern*). Este padrão garante a existência de apenas uma instância de uma classe, mantendo um ponto global de acesso ao seu objeto.

3.2 Pacotes

JavaTV necessita de um ambiente *PersonalJava* e usa um subconjunto do *Abstract Window Toolkit* (AWT) para construir as interfaces do usuário e *Java Media Framework* (JMF) para o controle da mídia.

A API está organizada nos seguintes pacotes:

- `javax.tv.xlet` – modelo de ciclo de vida das aplicações e classes de apoio;
- `javax.tv.locator` – provê mecanismos para referências em formato de URL para serviços de *broadcast* e clipes de mídia *broadcast*;
- `javax.tv.net` – provê mecanismo para acessar datagramas IP contidos em um fluxo de *broadcast*;
- `javax.tv.carousel` – acesso a arquivos de *broadcast* agregados em um sistema de arquivos no fluxo de vídeo;
- `javax.tv.graphics` – adiciona um suporte mínimo à biblioteca gráfica AWT, para solução de questões específicas em TV (sobreposição de imagens em vídeo, etc.);
- `javax.tv.util` – classes utilitárias para aplicações JavaTV, incluindo gerência de temporizadores e eventos temporizados;
- `javax.tv.media` – extensões para suporte à integração JMF;
- `javax.tv.media.protocol` – suporte JMF para protocolos de *streaming broadcast*;
- `javax.tv.service` – descrição alto nível de serviços de TV Digital, incluindo também mecanismos básicos para a coleta de informações sobre serviço do fluxo *broadcast*;
- `javax.tv.service.guide` – suporte para aplicações do tipo EPG (*Electronic Program Guide*, Guia Eletrônico de Programação) e seus conceitos associados (horário, classificação etária, etc.);

- `javax.tv.service.navigation` – suporte à navegação em serviços de TV Digital. Isso inclui suporte à lista de serviços favoritos, componentes agregados a serviços, etc.;
- `javax.tv.service.transport` – conceitos descrevendo os mecanismos de transporte para um serviço de TV Digital;
- `javax.tv.service.selection` – conceitos descrevendo como os serviços são apresentados ao usuário e como um novo serviço pode ser selecionado. Há também a possibilidade de apresentação de múltiplos serviços de uma só vez (*picture-in-picture*).

3.3 Xlet

Um Xlet é uma aplicação Java que proporciona interatividade para a TV Digital. É similar a um *Applet*, que é adicionado em páginas HTML, o Xlet é incluído em um serviço de TV Digital. O *middleware* identifica o ponto de entrada da aplicação (classe que implementa a interface `javax.tv.xlet.Xlet`) e executa a aplicação utilizando a máquina virtual Java. O *middleware* geralmente possui um componente que é responsável por gerenciar o ciclo de vida das aplicações, que instancia a classe principal (*main class*) e invoca os métodos responsáveis pela mudança de estados (BATISTA, 2007).

Um Xlet possui quatro estados principais: Carregado (*Loaded*); Paralisado (*Paused*); Iniciado (*Started*); e Destruído (*Destroyed*). A Figura 8 ilustra os estados que um Xlet pode possuir.

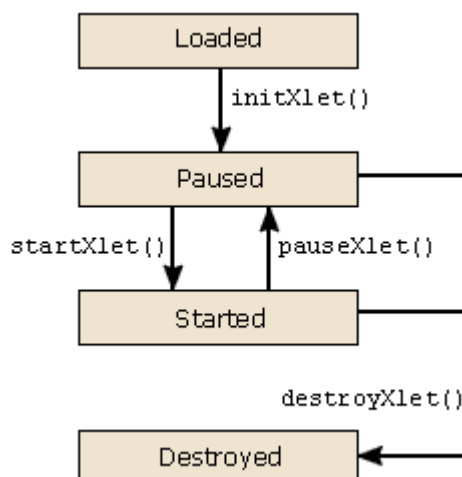


Figura 8 - Ciclo de vida de um Xlet.

Quando o gerenciador de aplicações carrega a classe principal do Xlet ele cria uma instância do Xlet chamando o construtor. Isso pode ocorrer em qualquer momento depois que a aplicação foi transmitida, nesse momento o Xlet encontra-se no estado **Loaded**.

No momento em que o telespectador escolher iniciar um determinado serviço que contenha o Xlet, ou quando alguma aplicação invoca um Xlet o gerenciador de aplicações chama o método de inicialização do Xlet (`initXlet()`), passando para o Xlet um objeto chamado `XletContext`. O Xlet utiliza esse objeto para se auto-inicializar e pré-carregar qualquer recurso que detenha maior processamento (como imagens, por exemplo). Após essa ação o Xlet entra no estado de **Paused**, no qual está pronto para ser iniciado imediatamente.

Quando o gerenciador de aplicações invocar o método `startXlet()` já poderá iniciar a interação ao telespectador, ou seja, a aplicação está em execução, **Started**.

Durante a execução de um Xlet, o gerenciador de aplicações pode invocar o método `pauseXlet()`, Movendo a aplicação do estado **Started** para **Paused**. E posteriormente pode chamar novamente o método `startXlet()` para voltar para o estado **Started**.

Ao finalizar o ciclo de vida de um Xlet o gerenciador de aplicações invoca o método `destroyXlet()`, que moverá o Xlet para o estado **Destroyed**

liberando todos os recursos utilizados pela aplicação, e o Xlet não pode voltar a ser iniciado. A figura 9 mostra o código-fonte da Interface Xlet.

```
public interface Xlet {  
  
    public void initXlet(XletContext ctx)  
        throws XletStateChangeException;  
  
    public void startXlet()  
        throws XletStateChangeException;  
  
    public void pauseXlet();  
    public void destroyXlet(boolean unconditional)  
        throws XletStateChangeException;  
}
```

Figura 9 - Interface `javax.tv.Xlet`.

O objeto `XletContext` realiza a comunicação entre o gerenciador de aplicações e a aplicação.

O método `notifyDestroyed()` notifica o *middleware* de que o Xlet está pronto para ser destruído. Quando o Xlet está pronto para ser paralisado é chamado o método `notifyPaused()`. Para fazer uma requisição ao *middleware* para ativar o Xlet é invocado o método `resumeRequest()`, em alguns casos o Xlet pausa a si próprio, de acordo com algum evento, requer ativação. Para acessar as propriedades definidas para o Xlet utiliza-se o método `getXletProperty()`.

A figura 10 mostra o código-fonte da Interface `XLetContext`.

```
public interface XletContext {  
    public static final String ARGS = "javax.tv.xlet.args"  
  
    public void notifyDestroyed();  
    public void notifyPaused();  
    public void resumeRequest();  
  
    public Object getXletProperty(String key);  
}
```

Figura 10 - Interface javax.xlet.XletContext.

4 EMULADOR

Como normalmente é difícil um desenvolvedor possuir uma rede de TV Digital experimental disponível, na maior parte das vezes o ambiente é simulado com a utilização de estações testes ou com emuladores em software.

Em uma estação teste são disponibilizados equipamentos e softwares que visam reproduzir as características relevantes em um ambiente de TV Digital real. Já o emulador faz o papel do componente do *middleware* que gerencia as aplicações. Um emulador é, então, interessante para testes preliminares em aplicações de TV Digital, visto que o ambiente possui vários elementos envolvidos, além do *middleware* (MONTEZ, 2004).

Já foi desenvolvida pelo Laboratório de Aplicações Vídeo Digital (LAVID) da Universidade Federal da Paraíba (UFPB) uma estação (software e hardware) para testes, chamada Xtation (LAVID, 2007).

Para desenvolvimento de aplicações existem diversos emuladores, que simulam o papel do *middleware*, dentre eles, o mais utilizado é o XleTVView.

A figura 11 ilustra a arquitetura da API JavaTV executada em um *set-top-box* e em um PC convencional.

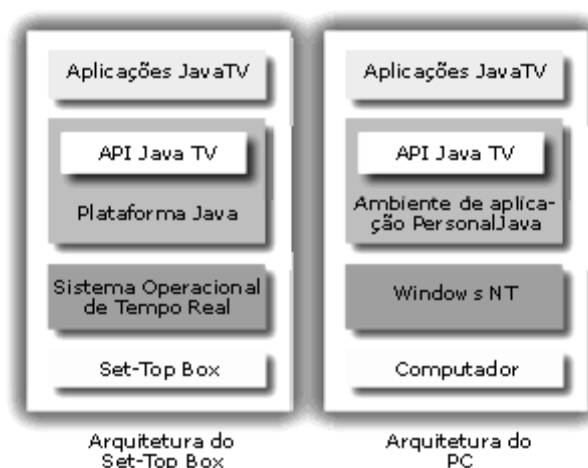


Figura 11 - A arquitetura da API JavaTV executada em um *set-top-box* e em um PC.

Um ambiente de televisão interativa possui recursos de memória e processamento mais escassos que um PC convencional, um *set-top-box* recebe o conteúdo através de difusão de um sinal que contém uma seqüência de transporte MPEG-2, enquanto um PC pode utilizar as mais diversas fontes de mídias (locais ou remotas). O *set-top-box* possui vários dispositivos de hardware específicos que não são encontrados normalmente em um computador, tais como decodificadores MPEG e demodularizadores, dentre outras particularidades. Para isso os emuladores foram desenvolvidos para superar essas diferenças.

4.1 XletView

O XletView é um emulador usado para executar Xlets em um PC. Ele possui código aberto sob a licença para software livre (*General Public License - GPL*), e além de uma implementação de referência da API JavaTV, traz consigo implementações de outras APIs especificadas no padrão MHP (*Multimedia Home Platform*), como HAVI (*Home Audio Video Interoperability*), DAVIC (*Digital Audio Video Council*) e implementações especificadas pela própria DVB (*Digital Video Broadcasting*), além das bibliotecas do *PersonalJava* que o mesmo padrão faz uso (MONTEZ, 2004).

Como a versão para desenvolvimento em Java do *middleware* brasileiro (Ginga-J) ainda não foi liberada, todos os testes de aplicações voltadas para a televisão digital interativa do Brasil, também são feitas utilizando o XletView, pois todas os padrões que baseiam na tecnologia Java seguem os mesmos princípios e requisitos, garantindo assim, uma portabilidade entre os padrões de transmissão de TV Digital.

O XletView é desenvolvido em Java e para sua execução independente do sistema operacional, é necessário a utilização do Java 2 *Standard Development Kit* para compilar Xlets e executar o XletView. Esse emulador utiliza o JMF (*Java Media Framework*) 2.1.1, porém com várias deficiências, como a incapacidade de exibir vídeo MPEG (*Moving Picture Experts Group*) relacionado ou controlado por uma Xlet (MONTEZ, 2004).

Existe no XletView um simulador de controle remoto, baseado nos botões padrões MHP, onde os eventos são capturados utilizando a classe AWT ou HAVI (que são extensões AWT). A figura 12 mostra a interface do XletView.

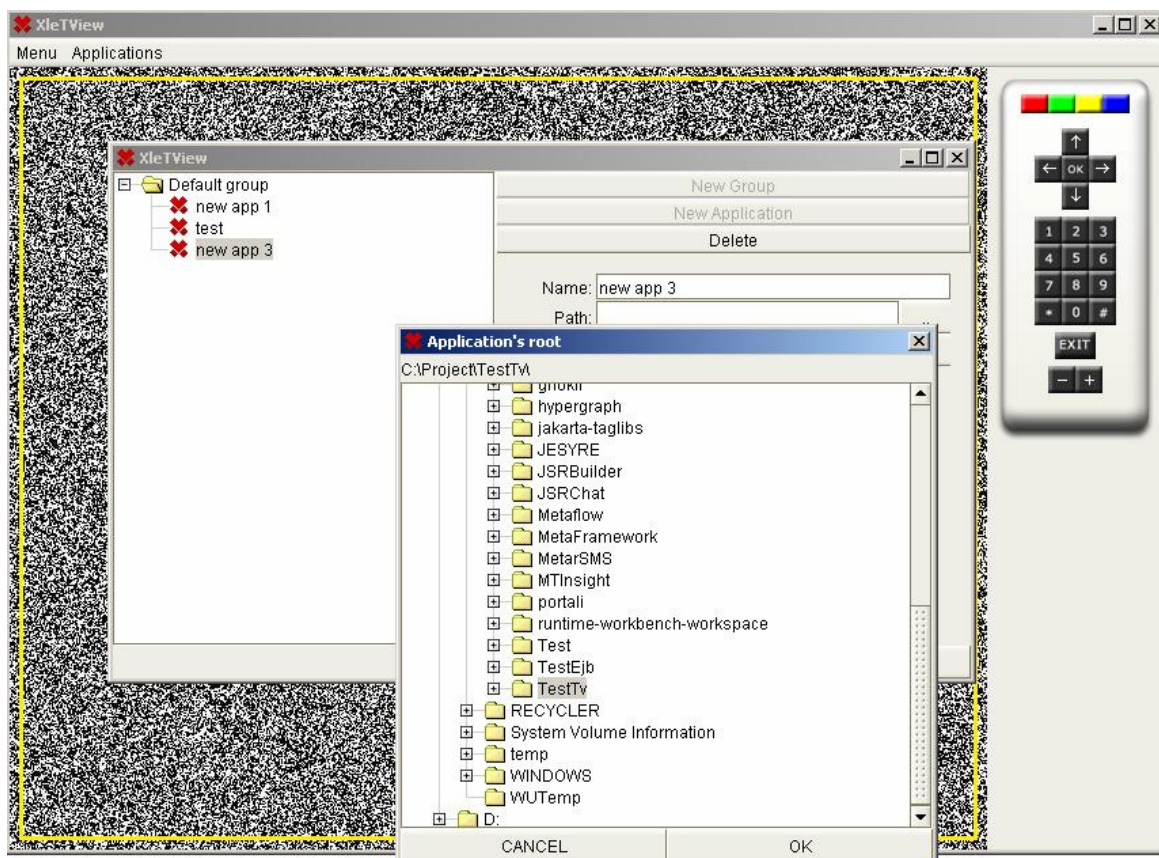


Figura 12 - Interface do XletView.

Para executar o XletView é necessário instalar os seguintes programas e bibliotecas:

- Java 2 *Standard Development Kit* (jdk-6u2-windows-i586-p);
- JavaTV API (api_jtv-1_0-src);
- XletView (xletview-0.3.6).

Para que uma aplicação seja executada faz-se necessário definir as seguintes bibliotecas no *CLASSPATH* onde a JRE (*Java Runtime Environment*) está

instalada. Por exemplo, se a JRE está na pasta “C:\Arquivos de programas\Java”, deve-se ter a seguinte estrutura de sub-pastas:

- “C:\Arquivos de programas\Java\xletview-0.3.6\xletview.jar”;
- “C:\Arquivos de programas\Java\javatv_fcs\javatv.jar”;
- “C:\Arquivos de programas\Java\javatv_fcs\lib”.

Para executar o XleTView faz-se necessário abri-lo via console, pois se executar diretamente o arquivo “xletview.jar” não é possível visualizar os resultados de comandos como `System.out.println()` e mensagens de erro. Para isso, foi criado um arquivo de lote (Figura 13) para facilitar a execução do programa.

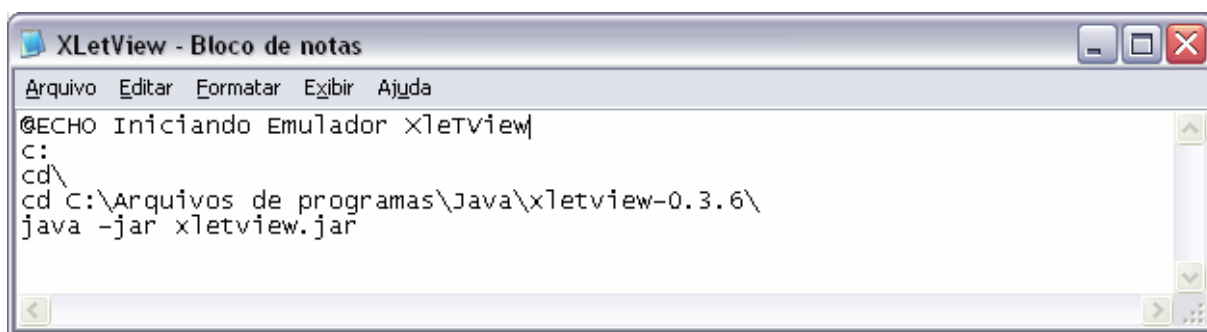
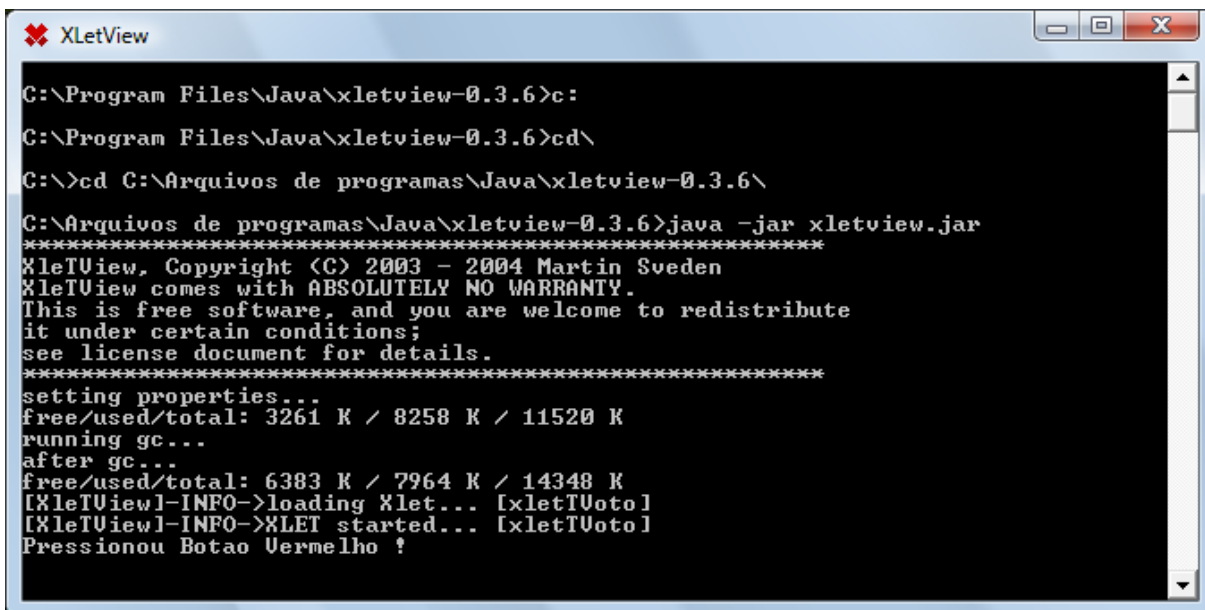


Figura 13 - Arquivo de lote (xletview.bat).

Durante a execução de um Xlet, são exibidas diversas informações no console (Figura 14), além de permitir ao desenvolvedor uma interface para inserir informações para acompanhamento da etapa de execução do Xlet e também informações referentes à execução da aplicação pelo emulador, inclusive mensagens referentes a exceções causadas em tempo de execução.

The image shows a screenshot of a Windows application window titled "XleTView". The window contains a black console area with white text. The text shows a series of commands being entered and executed in a command prompt. The commands include navigating to the directory "C:\Program Files\Java\xletview-0.3.6\" and running "java -jar xletview.jar". The output of the command shows copyright information for Martin Sveden (2003-2004), a license notice stating "ABSOLUTELY NO WARRANTY", and memory usage statistics before and after a garbage collection (gc) process. The final output line is "Pressiou Botao Vermelho !".

```
C:\Program Files\Java\xletview-0.3.6>c:
C:\Program Files\Java\xletview-0.3.6>cd\
C:\>cd C:\Arquivos de programas\Java\xletview-0.3.6\
C:\Arquivos de programas\Java\xletview-0.3.6>java -jar xletview.jar
*****
XleTView, Copyright (C) 2003 - 2004 Martin Sveden
XleTView comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute
it under certain conditions;
see license document for details.
*****
setting properties...
free/used/total: 3261 K / 8258 K / 11520 K
running gc...
after gc...
free/used/total: 6383 K / 7964 K / 14348 K
[XleTView]-INFO->loading Xlet... [xletTUoto]
[XleTView]-INFO->XLET started... [xletTUoto]
Pressiou Botao Vermelho !
```

Figura 14 - Execução do Console pelo XleTView.

5 APLICAÇÃO PARA TV DIGITAL DESENVOLVIDA NESTE TRABALHO

Desenvolver uma aplicação para a televisão digital é muito diferente de uma aplicação para *desktop*. As restrições de tamanho e resolução de tela, quantidade de cores, distância entre o telespectador e o aparelho de TV faz com que as fontes usadas sejam maiores, com isso diminui a área útil do aplicativo. Além disso, a limitação de interação imposta pelo controle remoto, baixo processamento e memória, são fatores que fazem do desenvolvimento de aplicações de TV digital um negócio especializado (JUCÁ, 2007).

Na realidade brasileira, a baixa qualidade das conexões de dados, representa uma restrição adicional, pois muitos usuários não possuem bons *links* de internet e por restrições de custos não conectam suas linhas telefônicas (JUCÁ, 2007). Sem esse canal de retorno não é possível a interatividade total das aplicações.

Em aplicações para a TV digital o processo de desenvolvimento é um pouco diferenciado, pois existe uma grande dependência das etapas de *design* e testes. As aplicações de TV geralmente são pequenas, devido às limitações de memória e processamento.

Cada aplicação terá particularidades específicas as quais dependem do *middleware* escolhido. No caso do Ginga, o *middleware* para o sistema de televisão digital adotado no Brasil, a aplicação pode ser declarativa (Ginga NCL) ou procedural (Ginga-J). Os declarativos permitem fazer aplicações mais rapidamente, mas geralmente aplicações bem mais simples, já os procedurais podem ter como linguagem de desenvolvimento Java ou C.

Neste projeto, foi desenvolvida uma aplicação para ser executada em um *middleware* procedural, utilizando a linguagem Java (JDK versão 6.0) e a API JavaTV (versão 1.0). Essa aplicação foi nomeada de TVoto, para sua codificação e compilação foi utilizada a IDE (*Integrated Development Environment*) NetBeans (versão 6.0). Para executar a aplicação foi utilizado o emulador XleTView (versão 0.3.6).

5.1 TVoto

TVoto é uma aplicação (um Xlet) que pode ser executada no *middleware* da TV Digital. Foi desenvolvido na linguagem Java utilizando a API JavaTV, a ferramenta para implementação do projeto (IDE) adotada foi o NetBeans e para simular um *middleware* de TV Digital foi utilizando o XletView.

O TVoto é um aplicativo que proporciona ao telespectador interatividade com o programa que esta sendo transmitido. Durante a exibição de um programa de televisão é solicitado ao telespectador manifestar sua opinião através de um voto. Como é feito nos programas de *Reality Shows*, onde é realizada uma votação para eliminar um participante do programa, ou uma enquete que é feita durante a apresentação de um programa. O telespectador manifesta sua opinião diretamente do controle remoto, onde ele informa qual é sua opção e envia seu voto para a emissora de TV.

Uma estação de TV (Servidor de Geração de Conteúdo) transmitirá (via satélite, terrestre ou cabo) o sinal de vídeo, para os receptores de televisão digital interativa, durante esse momento o telespectador enquanto assiste o programa, poderá executar a aplicação (Xlet do TVoto) para manifestar seu voto, e por intermédio de um canal de retorno (conexão internet) poderá registrar seu voto nos Servidores de Aplicações Interativas, conforme ilustra o diagrama de funcionamento na figura 15, o qual detalha as características de um ambiente real.

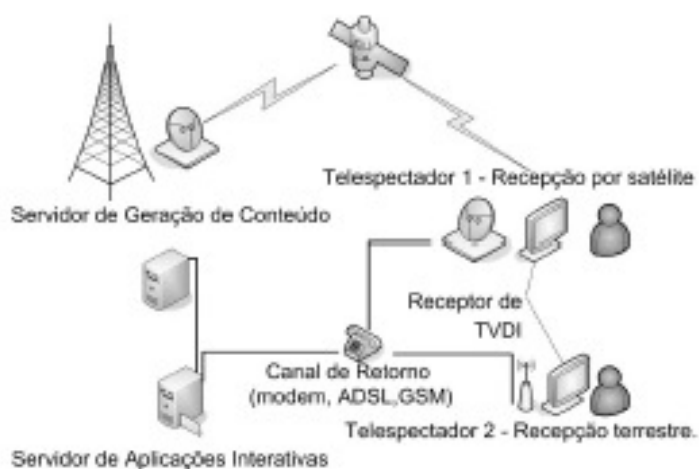


Figura 15 - Diagrama de funcionamento em um ambiente real.

Para implementação desse projeto foi adotado um ambiente simulado, onde foi focado apenas na execução da aplicação no *middleware*, no caso, tal função foi realizada pelo emulador XleTView. O Xlet do TVoto é executado enquanto é mostrado um vídeo de fundo, simulando a transmissão de um canal de TV e uma aplicação simulando o Servidor de Aplicações Interativas fica rodando aguardando uma conexão via *sockets*, para receber o arquivo XML (*Extensible Markup Language*) que contém o voto do telespectador.

A figura 16 ilustra a execução do Xlet da aplicação TVoto.



Figura 16 - Interface da aplicação TVoto.

No momento que o XleTView inicia sua execução é exibido um vídeo de fundo, o qual simula a transmissão de um programa de televisão. Esse vídeo pode ser configurado no arquivo “channels.xml”.

Ao iniciar o Xlet TVoto é exibido na parte inferior da tela a mensagem: “Pressione o botão vermelho para iniciar a interatividade.” Quando o telespectador pressiona o botão “Vermelho” a aplicação é iniciada, exibindo a questão ou tema relacionado com o assunto da programação que está sendo transmitida no momento e as opções existentes para o telespectador manifestar sua opinião (votar). Conforme pode ser observado na figura 17.



Figura 17 - Aplicação iniciada.

O telespectador escolhe sua opção e pressiona o número correspondente no controle remoto, em seguida pressiona o botão “OK” (figura 18).

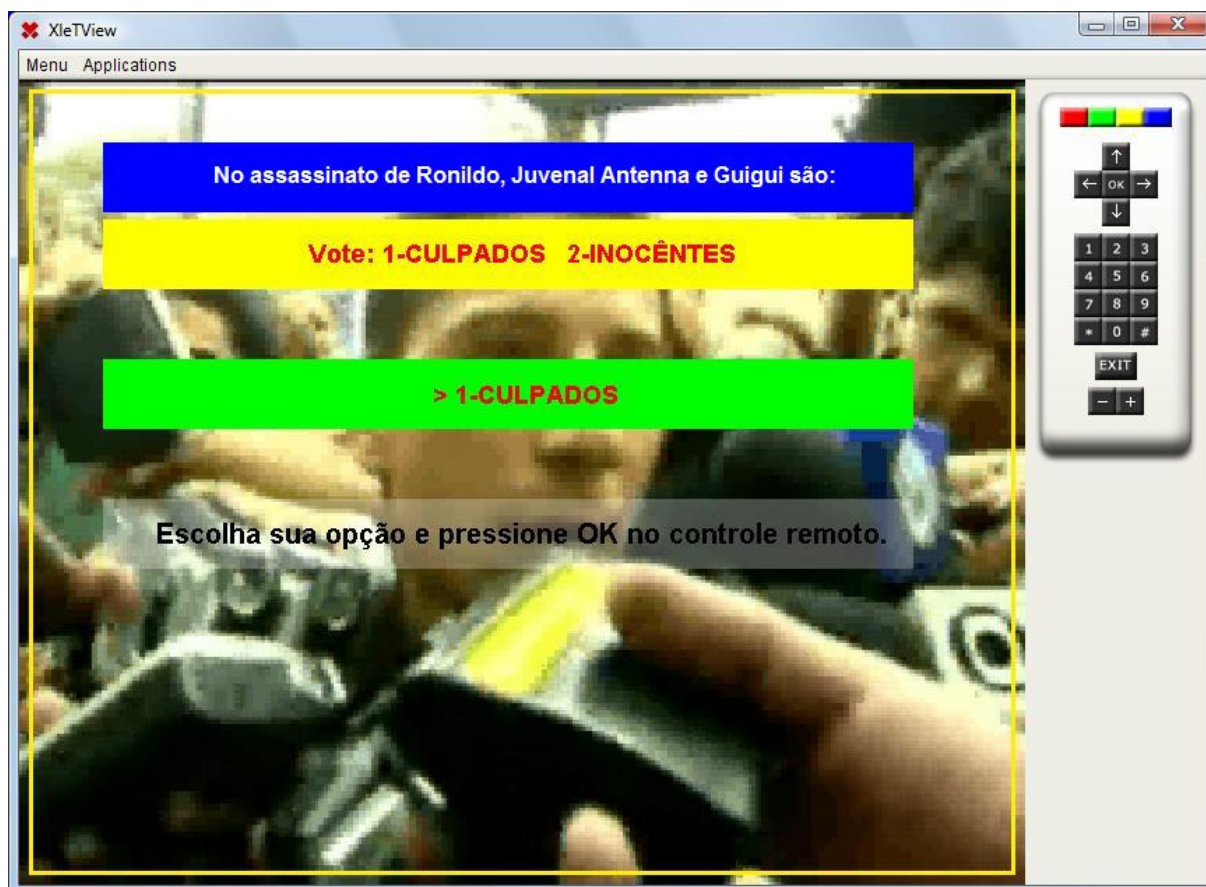


Figura 18 - Telespectador escolhe sua alternativa.

No momento em que o telespectador pressiona o botão "OK" é utilizado o canal de retorno para enviar o voto para emissora. É gerado um arquivo XML contendo todas as informações pertinentes à opção escolhida, e informações referentes ao aparelho que enviou o voto. Após a criação do arquivo XML, o mesmo é enviado via *socktes* para o aplicativo Servidor (que simula uma aplicação executada na emissora de TV).

Após a transmissão do voto, a aplicação é encerrada e é exibida ao telespectador a mensagem: "Voto Registrado com sucesso!" (figura 19).

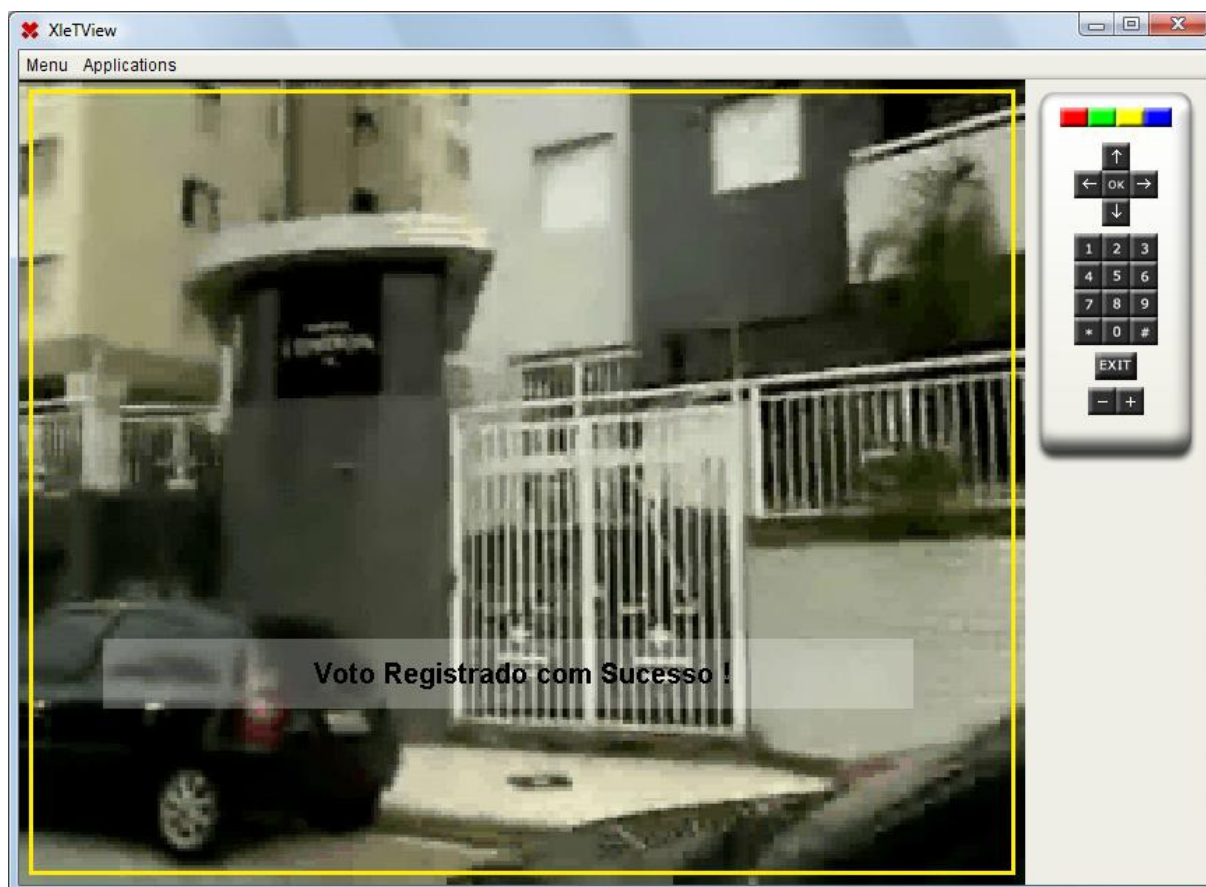


Figura 19 - Momento em que o telespectador finaliza a votação.

5.2 Configuração da aplicação

Na IDE NetBeans, para que seja possível compilar o xlet TVoto é necessário adicionar as seguintes bibliotecas :

- “javatv.jar”;
- “xletview.jar”;
- “xstream-1.3.jar”.

A estrutura do projeto ficará conforme mostra a figura 20, contendo as classes do projeto e as bibliotecas utilizadas para sua compilação.

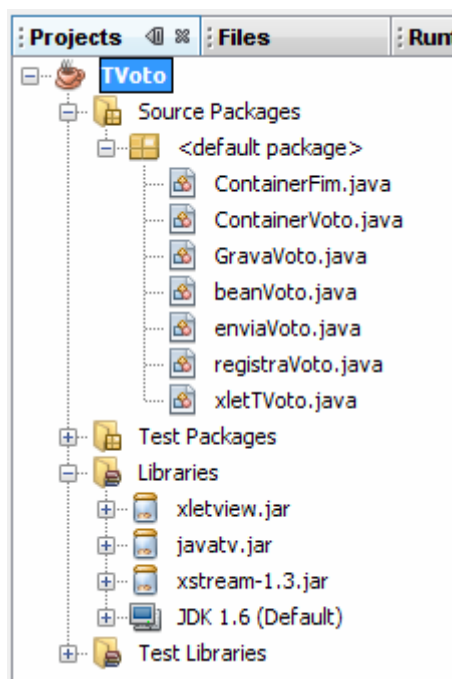


Figura 20 - Estrutura do Projeto.

Para que um vídeo seja executado no XleTView é necessário que seja um vídeo no formato “.avi” codificado com o codec Cinepak.

Para obter os vídeos para executar juntamente com o TVoto foram realizados os seguintes passos:

- Obter o vídeo da internet, no formato “.flv” (utilizando o site <http://www.youtubecatcher.com/>);
- Converter para o formato “.avi” (utilizando o software “FLV to AVI Video Converter”);
- Converter vídeo para o formato “.avi” com o codec *Cinepak* (utilizando o software “RAD Video Tools”).

Para executar o vídeo basta configurar o arquivo “channels.xml” que se encontra na pasta “config” no diretório do XleTView. A *tag* que define qual vídeo que será executado é <MEDIA>.

No XleTView faz necessário informar qual Xlet deve ser executado, indicando qual pasta se encontra os arquivos “.class”. Para configurar deve-se

acessar o menu "Applications", selecionar "Manage applications", na janela que se abre, deve-se escolher "New Application" e salvar. Para executar, deve-se abrir o menu "Applications" e selecionar o nome dado ao Xlet (`xletTVoto`).

5.3 Implementação do Xlet

Uma aplicação para TV Digital (um Xlet) é composta por uma classe principal (que implementa a interface `Xlet` provida pela API JavaTV) e demais classes que compõem o projeto.

A classe `xletTVoto` é a classe principal que implementa a interface `Xlet`, por isso nela são reescritos os métodos: `initXlet()`, `startXlet()`, `pauseXlet()`, e `destroyXlet()`. A classe principal possui uma propriedade que é do tipo `XletContext`, ele que é responsável por notificar o *middleware* as mudanças de estado, no momento em que o Xlet é iniciado (`initXlet()`) esse contexto é referenciado.

No método `startXlet()`, é definido a cena (*scene*) onde serão inseridos os elementos gráficos da aplicação, bem como a definição da resolução da tela. Nesse método já é criado os primeiros elementos gráficos da aplicação, os quais serão visíveis ao usuário, a partir daí a aplicação é iniciada, e por meio da interface `KeyListener` que também é implementada na classe principal é possível monitorar a interação do usuário através das teclas pressionadas (as quais simulam a utilização do controle remoto).

Dependendo da opção selecionada pelo usuário são instanciadas as demais classes que são *Containers*, os quais são adicionados à cena do Xlet, assim sendo exibidas ao usuário. Cada *container* é como se fosse um formulário da aplicação, que contém elementos gráficos que são apresentados ao usuário e também executam métodos característicos de cada classe, como gerar um arquivo XML ou enviar via *sockets* um documento.

5.4 Construção da interface gráfica

O principal elemento na construção da interface gráfica é representado pela classe `HScene`. Essa classe é semelhante ao *Frame*, ou *Window* utilizado como *container* principal durante o desenvolvimento de interfaces gráficas em Java.

Uma `HScene` apresenta algumas restrições como por exemplo a exibição de uma única instância dessa classe durante qualquer momento da existência da aplicação. Quando a aplicação for finalizada, deverá ser chamado o método `HScene.dispose()` para remover a instância daquela classe.

Para criação de um objeto `HScene`, basta declarar o objeto e obter uma instância através da classe auxiliar `SceneFactory` (POZZO, 2006).

Os componentes inseridos sobre a instância da `HScene` são similares aos componentes AWT ou *Swing* do Java. Foi utilizando o componente `HStaticText` para exibir as informações na tela, no entanto existem diversos outros componente que podem ser utilizados, o conjunto completo de componentes disponíveis pode ser obtido da própria documentação HAVi (*Home Audio Video Interoperability*) (HAVI, 2007).

Para o tratamento de eventos relacionados aos componentes da interface gráfica (como pressionar um botão do controle remoto) as classes implementam a interface `KeyListener` (`java.awt.event.*`).

5.5 Classes Java

O Xlet TVoto é composto por várias classes, todas inicializadas através da classe principal `xletTVoto`, como ilustra o diagrama de classes da figura 21.

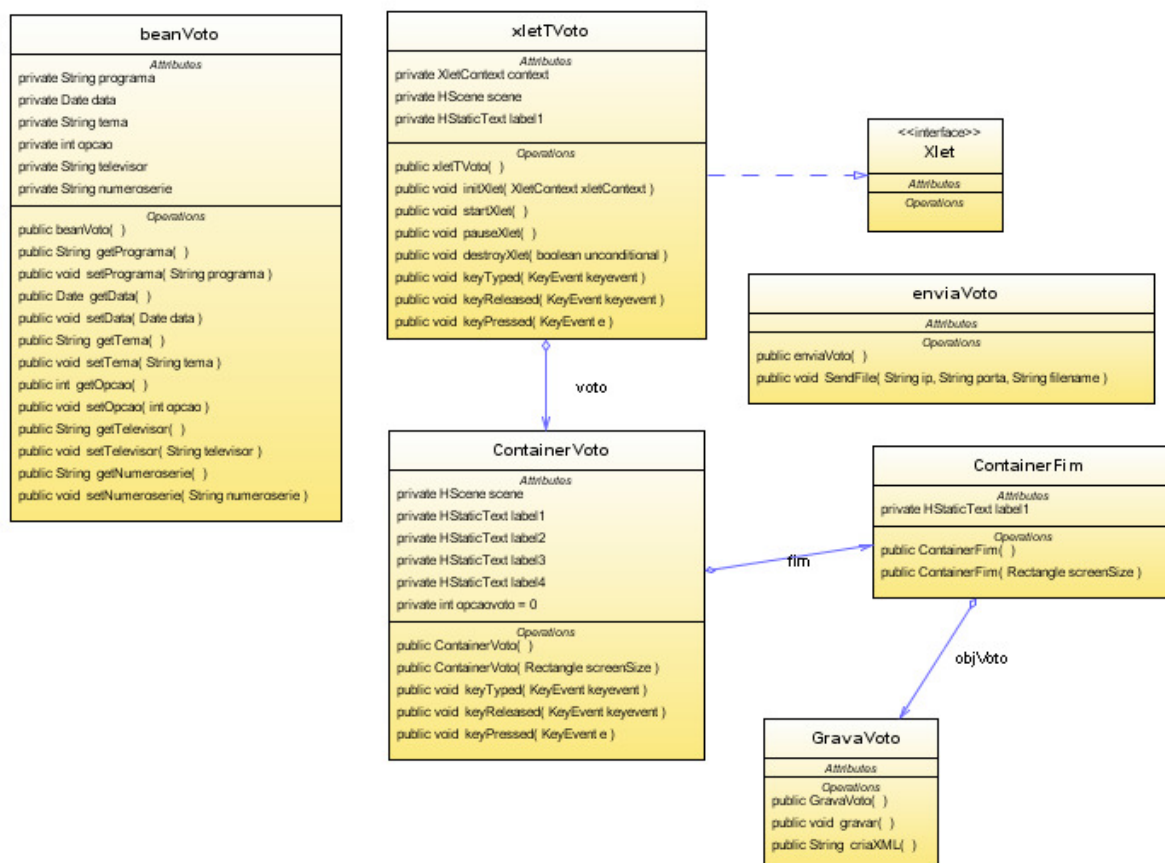


Figura 21 - Diagrama de Classes.

A classe `xletTVoto` (figura 22) é a classe principal do Xlet. Ela implementa a interface `Xlet` com todos os métodos que controlam o ciclo de vida de um Xlet e implementa a interface `KeyListener` para tratamento dos eventos.

xletTVoto
<i>Attributes</i> private XletContext context private HScene scene private HStaticText label1
<i>Operations</i> public xletTVoto() public void initXlet(XletContext xletContext) public void startXlet() public void pauseXlet() public void destroyXlet(boolean unconditional) public void keyTyped(KeyEvent keyevent) public void keyReleased(KeyEvent keyevent) public void keyPressed(KeyEvent e)

Figura 22 - Classe xletTVoto.

A classe `ContainerVoto` (figura 23) é a classe responsável por exibir e controlar as ações do telespectador, é pela sua instância que o usuário da aplicação informa qual é a opção escolhida.

ContainerVoto
<i>Attributes</i> private HScene scene private HStaticText label1 private HStaticText label2 private HStaticText label3 private HStaticText label4 private int opcaoovoto = 0
<i>Operations</i> public ContainerVoto() public ContainerVoto(Rectangle screenSize) public void keyTyped(KeyEvent keyevent) public void keyReleased(KeyEvent keyevent) public void keyPressed(KeyEvent e)

Figura 23 - Classe ContainerVoto.

`ContainerFim` (figura 24) é a classe responsável por finalizar a votação, nela são instanciados os objetos das classes responsáveis por gerar o arquivo XML e transmitir o voto pelo canal de retorno.

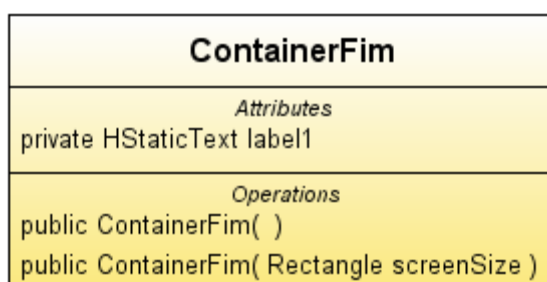


Figura 24 - Classe `ContainerFim`.

A classe `beanVoto` como o próprio nome já indica é um *Bean* que define conceitualmente um voto, com todas as propriedades suficientes para que a emissora possa identificar e contabilizar um voto. Com a instância de um objeto da classe `beanVoto` a partir da biblioteca XStream é gerado um arquivo XML.

A figura 25 exibe a classe `beanVoto` juntamente com todas suas propriedades e métodos *Getters* e *Setter*, os quais as definem como um *JavaBean*.

beanVoto
<i>Attributes</i>
private String programa
private Date data
private String tema
private int opcao
private String televisor
private String numeroserie
<i>Operations</i>
public beanVoto()
public String getPrograma()
public void setPrograma(String programa)
public Date getData()
public void setData(Date data)
public String getTema()
public void setTema(String tema)
public int getOpcao()
public void setOpcao(int opcao)
public String getTelevisor()
public void setTelevisor(String televisor)
public String getNumeroserie()
public void setNumeroserie(String numeroserie)

Figura 25 - Classe beanVoto.

GravaVoto (figura 26) é a classe responsável por gerar e armazenar o arquivo XML. Nela são utilizados os métodos da biblioteca XStream e o objeto da classe beanVoto.

GravaVoto
<i>Attributes</i>
<i>Operations</i>
public GravaVoto()
public void gravar()
public String criaXML()

Figura 26 - Classe GravaVoto.

A classe `enviaVoto` (figura 27) é responsável pela transmissão do voto do telespectador, ela utiliza *sockets* para estabelecer a conexão entre o *set-top-box* e a aplicação (que é executada na emissora).

enviaVoto
<i>Attributes</i>
<i>Operations</i>
<pre>public enviaVoto() public void SendFile(String ip, String porta, String filename)</pre>

Figura 27 - Classe `enviaVoto`.

5.6 XStream (XML)

Para registrar o voto e enviá-lo para os Servidores de Aplicações Interativas foi criado um arquivo XML (figura 28) contendo todas as informações sobre o aparelho que enviou (telespectador) e qual a opção escolhida (voto).

O padrão XML foi adotado devido a sua grande popularidade na comunicação entre aplicações.

Foi utilizada a API XStream para agilizar o trabalho com XML, que possibilita uma maneira simples pra manipular e gerar os arquivos XML (XSTREAM, 2007).

Para que a API gerasse um arquivo XML foi criado um *Bean* chamado `voto` que tem os valores de suas propriedades preenchidos pela aplicação, a API XStream cria um arquivo XML baseado nas propriedades do *Bean* `voto`, com seus respectivos valores.

Foi desenvolvida também uma interface servidor, que irá receber e processar esses votos, que também trabalha com a API XStream para manipular os arquivos XML.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <voto>
3      <programa>JN</>
4      <data>01/02/2008</data>
5      <tema>001</tema>
6      <opcao>1</opcao>
7      <televisor>XleTView Emulator</televisor>
8      <numeroserie>001000001XPT0</numeroserie>
9  </voto>

```

Figura 28 - Arquivo XML que é transmitido.

5.7 Socket

A comunicação por intermédio do canal de retorno entre a aplicação executada no *middleware* e o aplicativo que esta sendo executado no Servidor de Aplicações Interativas que irá receber e processar os dados, baseia-se na comunicação utilizando *Sockets*.

Através das funções da interface *sockets* pode-se estabelecer uma comunicação, processos cliente e servidor conseguem especificar endereços IP e portas de protocolos locais e remotos. Com essas funções pode-se determinar o protocolo utilizado (no caso da televisão digital será TCP), se está em modo ativo para o processo cliente que inicia a comunicação, ou modo passivo, para o processo em que o servidor aguarda por uma requisição, entre outras particularidades da comunicação utilizando *Sockets* (CANDIDO JUNIOR, 2005). A figura 29 exibe a aplicação que é executada no servidor.

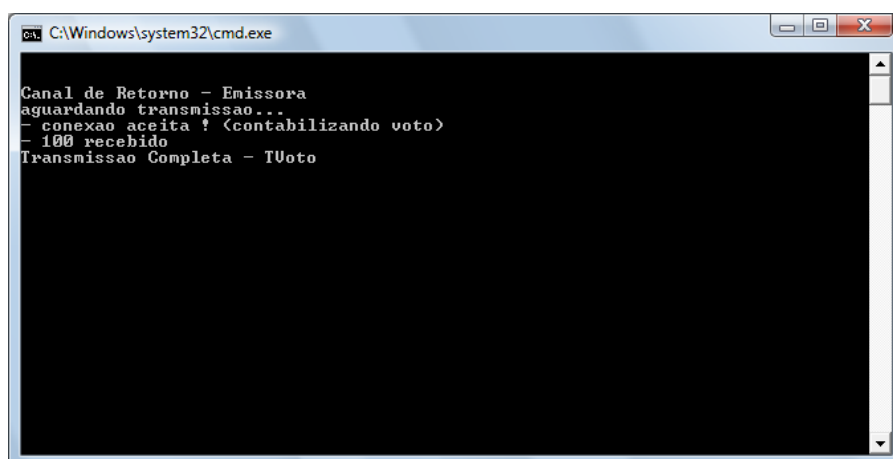


Figura 29 - Servidor aguardando envio do arquivo XML.

6 CONSIDERAÇÕES FINAIS

Devido ao avanço das tecnologias digitais, a evolução do ambiente televisivo, e principalmente ao início das transmissões do Sistema Brasileiro de Televisão Digital, surge um grande mercado para os desenvolvedores nessa área, principalmente no que se refere à televisão interativa.

A especificação API JavaTV para a televisão digital, além de uma implementação de referência para a mesma, introduziu o conceito de Xlets, tornando-o padrão nessa área, além de diversos recursos presentes na especificação.

Este projeto apresentou um estudo sobre o conceito de desenvolvimento de aplicações para a televisão digital interativa, bem como, foi definido o que é o sistema de televisão digital. Por fim, foi desenvolvida uma aplicação a qual foi executada por intermédio de um emulador do *middleware* da televisão digital para demonstrar o funcionamento da interatividade.

A partir desse projeto poderão ser desenvolvidas novas aplicações para a televisão digital interativa, além de proporcionar uma base para interessados em iniciar pesquisas e ingressar nas novas tecnologias relacionadas ao sistema de transmissão de televisão digital.

Ainda existe uma grande dificuldade no acesso às informações relacionadas ao desenvolvimento de aplicações para TV Digital, onde apenas um seleto grupo tem acesso e participa efetivamente nas decisões do padrão adotado no Brasil, além das barreiras tecnológicas existem os interesses financeiros e políticos que insistem em retardar a efetivação do projeto que está sendo implantado no país.

No momento em que finalizar as definições do *middleware* Ginga, será possível executar as aplicações Java. O módulo responsável por executar tais aplicações é o Ginga-J.

REFERÊNCIAS BIBLIOGRÁFICAS

ARIB - **Association of Radio Industries and Businesses**. Disponível em < <http://www.arib.or.jp/english/> >. Acesso 20/05/2007.

ATSC - **Advanced Television Systems Committee**, Inc. Disponível em < <http://www.atsc.org> >. Acesso 20/05/2007.

BATISTA, Carlos Eduardo. **TV Digital - Java na sala de estar**. 2007. Revista Mundo Java, número 17, ano III - Editora Mundo.

BECKER, Valdecir e MONTEZ, Carlos. **TV digital interativa: conceitos, desafios e perspectivas para o Brasil**. 2004. I2TV, Florianópolis.

BITTENCOURT, Mariana Mello e ARAUJO, Thiago Pereira. **Infra-estrutura para o Desenvolvimento de Aplicações para TV Digital Interativa**. 2006. Universidade Federal Fluminense, Rio de Janeiro.

CANDIDO JUNIOR, Eli. **Uma extensão multiplataforma para o subsistema Descritor MediSeek**. 2005. Trabalho de Conclusão de Curso (Bach. Em Ciência da Computação) – Faculdade de Informática de Presidente Prudente, Universidade do Oeste Paulista, Presidente Prudente.

DTV - **TV digital Brasileira**. Disponível em < <http://www.dtv.org.br/> >. Acesso 29/12/2007.

GINGA – **Portal do Software Público Brasileiro**. Disponível em < <http://www.softwarepublico.gov.br/> >. Acesso 20/11/2007.

HAVI - **HAVi Level 2 graphical user interface API**. Disponível em < <http://www.havi.org/> >. Acesso 20/05/2007.

JUCÁ, Paulyne Matthews. White Paper: **Aplicações para TV Digital**. 2006. Centro de Estudos e Sistemas Avançados do Recife.

LAVID - **Laboratório de Aplicações de Vídeo Digital**. Disponível em < <http://www.lavid.ufpb.br/> >. Acesso 20/06/2007.

MHP – **Multimedia Home Platform**. Disponível em < <http://www.mhp-interactive.org/> >. Acesso 20/05/2007.

MONTEZ, Carlos e PICCIONI, Carlos. **Um Estudo sobre Emuladores de Aplicações para a Televisão Digital Interativa**. 2004. Universidade Federal de Santa Catarina, Florianópolis.

POZZER, Cesar T. e FEIJÓ, Bruno. **Proposição de um novo paradigma de conteúdo para tv interativa**. 2003. Série Monografias em Ciência da Computação (MCC38/03), DI/PUC-Rio, Rio de Janeiro.

POZZO, Douglas Del. **Aplicativos para Televisão Digital Interativa**. 2006. Universidade Federal de Santa Catarina, Florianópolis.

SUN Microsystems. **JavaTV(TM) API: Technical Overview**. [S.l sn], 2000. disponível em < <http://java.sun.com/products/javatv/> >. Acesso 01/05/2007.

XLETVIEW - **XleTView is an emulator for testing MHP Xlets on a PC**. Disponível em < <http://sourceforge.net/projects/xletview/> >. Acesso 01/05/2007.

XSTREAM – **API XStream**. Disponível em < <http://xstream.codehaus.org/> >. Acesso 10/05/2007.