Universidade Federal de Pernambuco

Centro de Informática

TVision – Ferramenta Gráfica para Desenvolvimento de Aplicações Para TV Digital no Formato GINGA-NCL

Trabalho de Graduação em Ciência da Computação

Caio César Neves de Oliveira

Recife, 03/03/2009

Universidade Federal de Pernambuco

Centro de Informática

TVision – Ferramenta Gráfica para Desenvolvimento de Aplicações Para TV Digital no Formato GINGA-NCL

Trabalho de Graduação em Ciência da Computação

Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: *Prof. Ph.D.* Carlos André Guimarães Ferraz

Folha de Aprovação

TVision – Ferramenta Gráfica para Desenvolvimento de Aplicações Para TV Digital no Formato GINGA-NCL

Trabalho de Graduação em Ciência da Computação

Banca Examinadora:

_____

Carlos André Guimarães Ferraz – Orientador

_____

Nelson Souto Rosa – Avaliador

“It is best not to swap horses while crossing the river.”

Abraham Lincoln

Agradecimentos

Em primeiro lugar, a Deus por me ceder a oportunidade de poder concluir o curso de que escolhi.

Em seguida, a minha família por sempre acreditar em mim, especialmente minha mãe e meu pai e não menos especial minha querida irmã.

A Rebecka e família pelo amor, compreensão e ajuda em toda minha caminhada na universidade.

Aos meus amigos de universidade que compartilhei alegrias e tristezas, em especial a João Paulo, Bruno, Mário, Tiago, João Rocha, Henrique, Rafael e minha grande pequena amiga Leila.

Ao meu ex-chefe e grande amigo Ricardo Lessa que confiou em meu potencial no início de minha carreira e a todos que compõe a “velha guarda” da Educandus que me ajudaram no meu crescimento profissional.

A todos que compõe a Midias Educativas por confiar em meu trabalho e pela ajuda e compreensão nesta fase final de minha graduação.

Enfim, a todos que me ajudaram nessa jornada e contribuíram para a conquista de minha Vitória.

Resumo

A digitalização do sinal brasileiro de televisão traz muitos benefícios tanto na qualidade de som e imagem quanto à possibilidade do telespectador (agora tratado como usuário) interagir com uma aplicação que leva informações adicionais.

Com a quantidade e qualidade de conteúdo programático existente hoje na televisão brasileira, serão necessárias ferramentas que possam tornar o processo de desenvolvimento de software para TV Digital baseado no SBTVD – Sistema Brasileiro de TV Digital - mais ágil e sem perda de qualidade, tanto no produto quanto no processo, para suprir a demanda existente.

As técnicas ferramentais encontradas hoje são artesanais onde a escrita dos programas é feitas a mão e no melhor dos casos possuem IDEs (*Integrated Development Environment*) que agilizam alguns processos.

Este trabalho fará um estudo das ferramentas existentes identificando as necessidades do desenvolvedor no processo de implementação das aplicações e irá propor uma ferramenta de desenvolvimento de aplicações interativas para TV Digital no modelo brasileiro, mais especificamente para o GINGA-NCL (aplicações escritas em NCL e com o suporte de uma linguagem de script Lua) que atenda essas necessidades.

Essa ferramenta estará disponível na internet, ou seja, será executada em *browsers* da web e utilizando tecnologias do tipo RIA (*Rich Internet Application*), com uma interface que possa ser amigável e intuitiva.

Palavras-chave: TV digital, SBTVD, interatividade, aplicações, ferramenta de desenvolvimento.

Abstract

The digitalization of the Brazilian television sign brings many benefits, both in sound and image quality as the possibility of the viewer (now as user) to interact with the application, which carries additional information.

With the quantity and quality of program content in the Brazilian television, will be necessary tools to become the Digital TV development process based on SBTVD – Brazilian System of Digital TV – more agile and without quality loss, both in the product and in the process, to meet the existent demand.

With the quantity and quality of program content existents in the Brazilian television, will be necessary tools to become the Digital TV development process based on SBTVD – Brazilian System of Digital TV – more agile and without quality loss, both in the product and in the process, to meet the existent demand.

The tools found nowadays are craft, where the program writing is handmade and in the best case have IDEs (Integrated Development Environment) which accelerate some processes.

This project will do a study of the existent tools indentifying the developer necessities in the application implementation process and will propose a development tool to interactive applications for Digital TV in the Brazilian model, more specifically to GINGA-NCL (applications written in NCL, with support to LUA language scripts) which attends these necessities.

That tool will be available in the internet, running in web browsers and using RIA (Rich Internet Application) technologies, with an interface which can be friendly and intuitive.

Keywords: digital TV, SBTVD, interactivity, applications, development tool.

Índice

Folha de Aprovação 3

Agradecimentos 5

Resumo 6

Abstract 7

Lista de Figuras 10

Lista de Quadros 12

1. Introdução 13

1.1. Estrutura da Monografia 14

2. Referencial Teórico 15

2.1 GINGA-NCL 15

2.1.1. NCL – *Nested Context Language* 15

2.1.2. Lua 16

2.2 Tecnologias RIA 17

2.3 Trabalhos Relacionados 19

2.3.1 Composer 19

2.3.2 Gingaway 22

2.3.3 TVDesigner 28

2.3.4 Conclusão 33

3. Implementação 33

3.1. Requisitos 33

3.2. Arquitetura 34

3.3. Detalhes de Desenvolvimento 36

4. Estudo de Caso 38

4.1. Definição 38

4.2. Utilizando o TVision 39

4.2.1. Entrar no sistema 39

4.2.2. Criar Projeto 40
4.2.3. Criar Telas 41
4.2.4. Gerar Código do Projeto 42
4.2.5. Executar o Projeto no Simulador 43
5. Conclusões 46
5.1. Trabalhos Futuros 47
Referências 48
Apêndice I – Código NCL Gerado 51
Apêndice II – Código Lua Gerado 53

Lista de Figuras

Figura 19: Quadro comparativo entre tecnologias RIA [13]. 18
Figura 1: Visão completa da ferramenta Composer [10]. 19
Figura 2: Visão estrutural da ferramenta Composer [10]. 20
Figura 3: Visão de leiaute da ferramenta Composer [10]. 20
Figura 4: Visão temporal da ferramenta Composer [10]. 21
Figura 5: Visão textual da ferramenta Composer [10]. 21
Figura 6: Ginga-NCL Emulator. 22
Figura 7: Assistente de criação de projetos Ginga-NCL [11]. 23
Figura 8: Assistente de criação de documentos NCL [11]. 24
Figura 9: Assistente de criação de documentos Lua [11]. 24
Figura 10: Editor de código NCL do Gingaway [11]. 25
Figura 11: Editor de código Lua do Gingaway [11]. 26
Figura 12: Assistente de configuração do Ginga-NCL Virtual Set-top Box [11]. 27
Figura 13: Ginga-NCL Virtual Set-top Box [11]. 28
Figura 14: Visão geral do TVDesigner [18]. 29
Figura 15: Assistente de criação de projeto no TVDesigner [18]. 30
Figura 16: Criação de telas no TVDesigner [18]. 30
Figura 17: Tela de seleção de tela principal [18]. 31
Figura 18: Tela de diagrama de telas e geração de código [18]. 31
Figura 20: Diagrama da arquitetura em camadas. 35
Figura 21: Estrutura do banco de dados do TVision. 35
Figura 22: Diagrama de conexão com o servidor. 36
Figura 23: Flex Builder IDE. 37
Figura 24: Fluxo de navegação das telas do PropCel. 38
Figura 25: Visão geral do TVision. 39
Figura 26: Tela de *login* do TVision. 40
Figura 27: Tela de Criação de projeto no TVision. 40
Figura 28: Primeiro passo na criação de telas no TVision. 41
Figura 29: Segundo passo na criação de tela no TVision. 41
Figura 30: Descrição da área de edição do TVision. 42
Figura 31: Definição de tela principal no TVision. 43
Figura 32: Exportar projeto no TVision. 43

Figura 33: TelaPrincipal do PropCel..... 44
Figura 34: Tela1 do PropCel. 44
Figura 35: Tela2 do PropCel. 45
Figura 36: Tela3 do PropCel. 45

Lista de Quadros

Quadro 1: Quadro comparativo das ferramentas analisadas. 33
Quadro 2: Lista de requisitos e situação. 46

1. Introdução

A digitalização do sinal de televisão afetará diretamente o modo como os brasileiros irão assisti-la, principalmente quando a interatividade estiver sendo utilizada em plenitude, pois estará se tratando de uma nova plataforma de comunicação baseada numa tecnologia de transmissão e processamento de dados digitais. Pesquisas mostram que 94,8% dos domicílios no Brasil possuem pelo menos uma televisão [1] e que os brasileiros passam em média 5 horas por dia assistindo-a [2], ilustrando o impacto dessa mudança e o poder da televisão nos lares brasileiros.

O sinal digital já chegou às principais cidades do Brasil são elas: São Paulo (dezembro 2007), Rio de Janeiro, Belo Horizonte, Uberlândia, Goiânia, Curitiba, Brasília (TV Justiça), Porto Alegre, Florianópolis, Salvador, Campinas, Teresina, Recife e as demais capitais até o final de 2009 [3].

O padrão brasileiro de TV digital (SBTVD – Sistema Brasileiro de TV Digital), baseado no modelo de transmissão japonês (ISDB-T – *Integrated Services Digital Broadcasting Terrestrial*) é marcado pelo alto nível de inovação principalmente na plataforma, GINGA (também conhecida como *middleware* que é uma camada de *software* que provê uma série de serviços que as aplicações podem utilizar [4]) onde as aplicações são executadas.

O *middleware* é dividido em dois subsistemas: Ginga-NCL, baseado no NCL que é uma linguagem declarativa e com isso perde um pouco de poder no desenvolvimento de aplicações, e Ginga-J, baseado na linguagem Java que é procedural e por isso tem um poder maior no desenvolvimento de aplicações em geral.

Baseado na quantidade de conteúdo programático existente na televisão brasileira, levando também em consideração as propagandas comerciais que movimentou mais de 21 bilhões de reais em 2005 [16], existirá uma grande demanda de desenvolvimentos de aplicações, com isso exigirá ferramentas que tornem o processo de criação mais otimizado e rápido.

Este trabalho tem como objetivo criar uma ferramenta de desenvolvimento de aplicações interativas, no formato *drag and drop* – arrastar e soltar – que consiste no ato de selecionar componentes pré estabelecidos e arrumar manualmente (através do *mouse*) no leiaute da aplicação, para TV

Digital no modelo brasileiro, mais especificamente para o GINGA-NCL (aplicações escritas em NCL e com o suporte de uma linguagem de script Lua), a ser executada em *browsers* da web e utilizando tecnologias do tipo RIA (*Rich Internet Application*), com uma interface que possa ser amigável e intuitiva.

Serão estudadas as principais necessidades que podem tornar o desenvolvimento de aplicações para TV digital mais otimizado.

1.1. Estrutura da Monografia

Este trabalho está dividido nas seções a seguir:

- No capítulo 2 estudarei as tecnologias envolvida no projeto e os trabalhos relacionados com o TVision (ferramentas que auxiliem o desenvolvimento de aplicações para TV digital).
- No capítulo 3 elicitarei todas as necessidades identificadas para o desenvolvimento e explicitarei detalhes de implementação da ferramenta que estou a propor.
- No capítulo 4 darei um exemplo de como utilizar o TVision para desenvolver uma aplicação para TV digital no padrão GINGA-NCL.
- No quinto e último capítulo descreverei as conclusões que obtive no desenvolvimento deste trabalho e listarei algumas propostas de trabalhos futuros.

2. Referencial Teórico

Nesta sessão irei levantar e analisar tecnologias necessárias para o desenvolvimento de aplicações digitais e algumas ferramentas já existentes que auxiliam no desenvolvimento de aplicações para TV Digital, principalmente as ferramentas que geram aplicação no formato Ginga-NCL e Ginga-NCLua.

2.1 GINGA-NCL

Ginga-NCL é o subsistema Ginga para exibição de documentos NCL, com o suporte de uma linguagem de script Lua, e foi desenvolvido pela PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro) visando prover uma infra-estrutura de apresentação para aplicações declarativas escritas na linguagem NCL [5].

2.1.1. NCL – *Nested Context Language*

NCL é uma linguagem declarativa para desenvolvimento de documentos hipermídia baseado no modelo conceitual NCM *Nested Context Model* [6]. NCL prove facilidades para a especificação de aspectos de interatividade, sincronismo espaço-temporal entre objetos de mídia, adaptabilidade, suporte a múltiplos dispositivos e suporte à produção ao vivo de programas interativos não-lineares [5].

O desenvolvimento de aplicações NCL se baseia em algumas perguntas essências: qual mídia tocar? Onde tocar a mídia? Como tocar a mídia? E quando tocar a mídia? [7].

- Qual mídia tocar? Refere-se diretamente ao arquivo que será apresentado, para aplicações Ginga-NCL os tipos e formatos de mídias previstos são: imagem (GIF, JPEG etc.), de vídeo (MPEG, MOV etc.), de áudio (MP3, WMA etc.), de texto (TXT, PDF etc.), de execução (Xlet, Lua etc.), entre outros, como objetos de mídia NCL.

- Onde tocar a mídia? Refere-se ao local onde a mídia será apresentada. Essa região pode ser definida por valores absolutos ou relativos.
- Como tocar a mídia? Refere-se ao comportamento da mídia, exemplo: nível de transparência de uma imagem, o congelamento do último frame de um vídeo e etc.
- Quando tocar a mídia? Refere-se a sincronia com o documento NCL como um todo ou com outras mídias que estão sendo apresentadas no mesmo documento.

Desse modo como o documento é estruturado facilita o aprendizado de um desenvolvedor que nunca teve o contato com a linguagem.

2.1.2. Lua

Lua uma linguagem de programação inteiramente projetada, implementada e desenvolvida no Brasil, no laboratório Tecgraf - Grupo de Tecnologia em Computação Gráfica da PUC-Rio [8].

Lua é uma linguagem de programação procedural com construção para descrição de dados baseado em tabelas associativas e apesar de não ser orientada a objetos, que torna a linguagem mais leve, ela permite a criação de mecanismos que permite a implementação de classes e herança. Outros pontos positivos da linguagem que foram decisivos para a adesão dela ao Ginga-NCL são: rapidez de processamento (apesar de ser interpretada, Lua é uma das mais rápidas linguagens de script [8]), portabilidade e facilidade de ser embutida [8].

2.2 Tecnologias RIA

A internet foi originalmente concebida para a simples transferências de arquivos e informação [13], mas com o passar dos anos, a necessidade de formas mais interativas para prender a atenção do usuário proveu o desenvolvimento de aplicações com uma maior sofisticação gráfica e uma série de inovações no tocante a interação com o usuário.

Para um melhor entendimento do propósito das tecnologias *RIA*, basta perceber uma grande diferença: as aplicações tradicionais necessitam recarregar a página no *browser* para cada interação do usuário (com isso, torna a navegação da aplicação mais travada e é acentuada principalmente se a conexão com a internet não for banda larga) e as aplicações desenvolvidas com tecnologia RIA não recarregam a página (isso produz uma experiência de interatividade mais rica).

Nessa sessão irei mostrar alguns resultados de análises já feitas entre as mais famosas tecnologias *RIA*, mostrarei uma tabela comparativa e decidirei qual tecnologia será desenvolvida a ferramenta proposta nesse trabalho.

Na tabela abaixo segue as comparações entre três das mais famosas tecnologias RIA: AJAX, Flash (Flex) e Java.

	AJAX	Macromedia Flash	Java
Graphical Richness	Average (Same as HTML)	Very Rich	Rich
Container/Engine Footprint	Very Light (browser built-in)	Light	Heavy
Application Download	Fast	Slow	Slow
Audio/Video Support	Poor (unless use ActiveX)	Excellent	OK
Consistency on Different Computing Environments	Varies	Very consistent	Relatively consistent
Server Requirements	None or very minimal (TIBCO General Interface)	Yes (Flex or Open Laszlo)	Yes or No (Nexaweb, Java Web Start)
Plug-in/Runtime Requirement on Client	No	Flash (Player)	Java Runtime (JRE)
Development Challenge	Very complex without tools such as TIBCO, and high skills required (JavaScript, CSS, XML, XSLT, DOM, ActiveX...)	Relatively easy with tools such as Flex or Open Laszlo (XML, DOM, JavaScript, Flash, ActionScript)	Relatively easy with tools such as Nexaweb (XML, JavaScript, Java)
Security Concerns	JavaScript codes are open to public. Everybody can see source codes if desire	Flash files (compressed binary) are created. Flash Player becomes a sandbox	Class/Jar compressed binary files are created. JVM (Java Runtime) becomes a sandbox
Cost	Custom Build - Free TIBCO - Unknown	Open Laszlo - Free Flex - $15,000 per CPU	Java Web Start - Free Nexaweb - Unknown

Good Average Poor

Figura 1: Quadro comparativo entre tecnologias RIA [13].

De acordo com a tabela acima a melhor tecnologia a ser desenvolvida é Macromidia Flash.

Para esse trabalho caso irei utilizar a ferramenta Flex que utiliza toda base da tecnologia Flash e com um adicional muito importante que é a manipulação de componentes gráficos através da linguagem própria MXML baseada em XML [14].

2.3 Trabalhos Relacionados

Nesta sessão irei analisar as ferramentas: Composer, Gingaway e TVDesigner.

2.3.1 Composer

O Composer é um ambiente de autoria de NCL para TV Digital interativa onde são definidos alguns tipos de visões (estrutural, temporal, leiaute e textual) que permite dar um maior poder de edição, de modo que essas visões são sincronizadas [9].

Com o Composer é possível construir programas áudio visuais interativos com pouco conhecimento da linguagem NCL. A figura abaixo mostra uma visão completa do Composer [10].

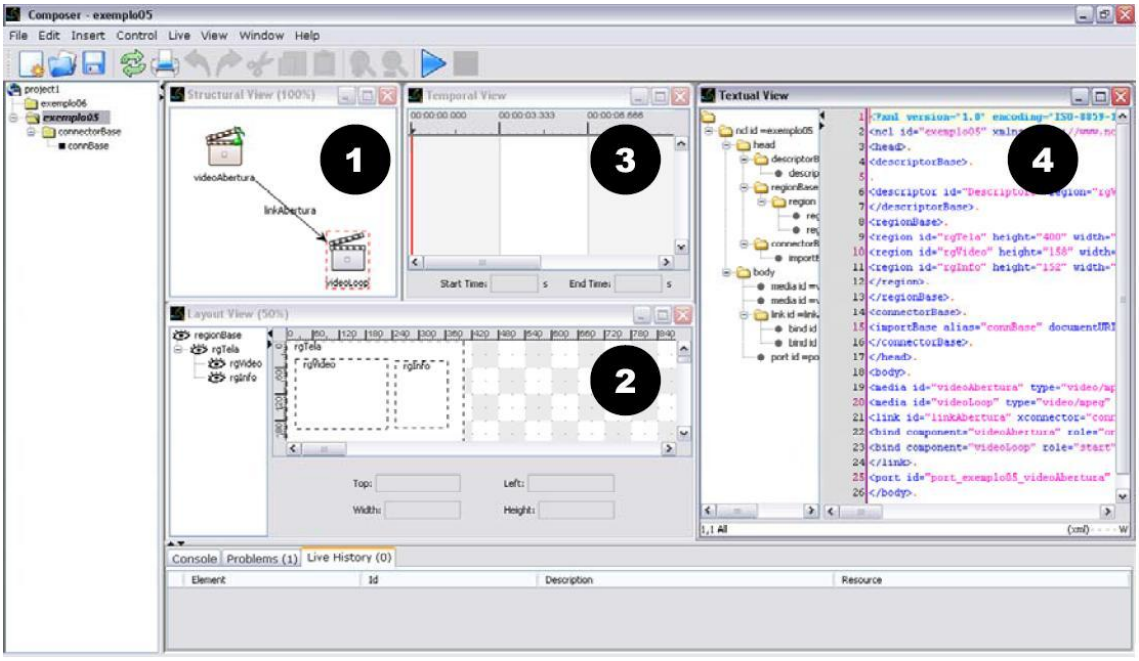

Figura 2: Visão completa da ferramenta Composer [10].

De acordo com a figura acima temos os quatro tipos de visão que a ferramenta dispõe:

- Na janela 1 temos a visão estrutural onde é apresentado os nós e os elos entre os nós, nela também pode-se alterar as propriedades de um nó de mídia.

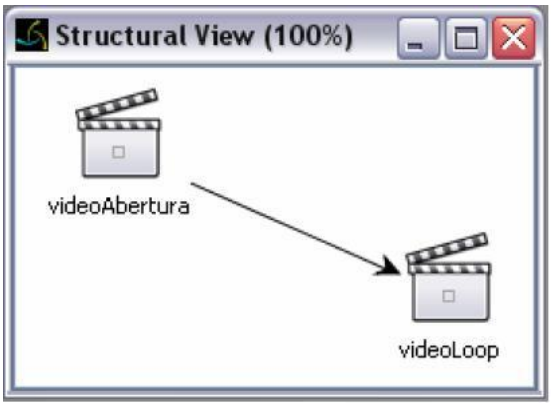

Figura 3: Visão estrutural da ferramenta Composer [10].

- Na janela 2 temos a visão de leiaute onde são apresentadas as regiões onde as mídias serão apresentadas.

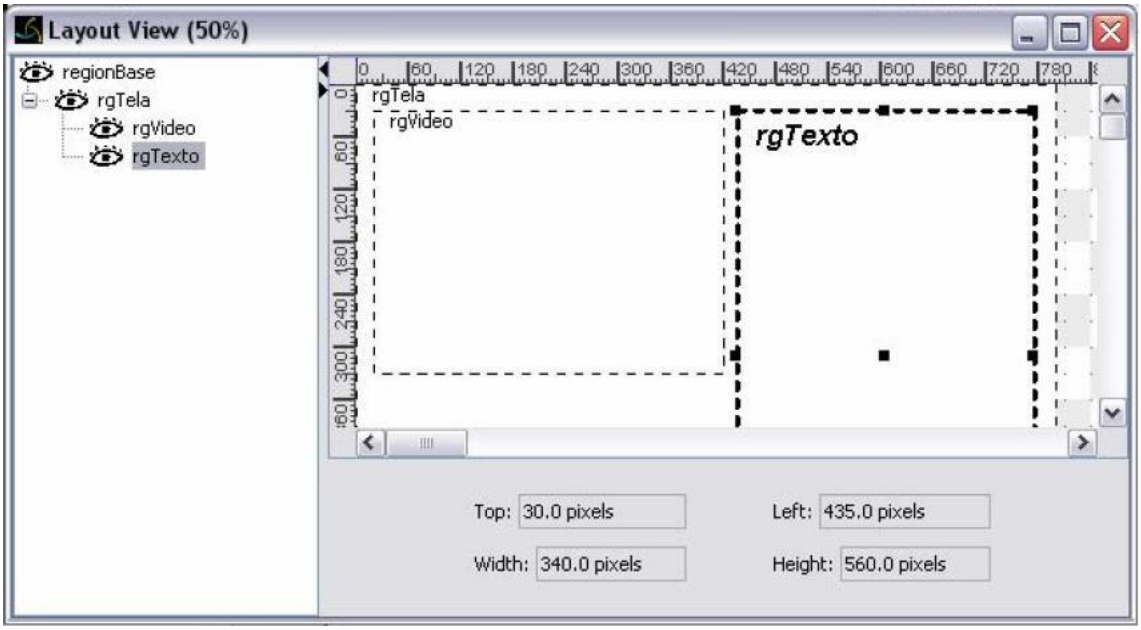

Figura 4: Visão de leiaute da ferramenta Composer [10].

- Na janela 3 temos a visão temporal onde pode-se fazer a sincronia das mídias da aplicação e também as oportunidades de interatividade.

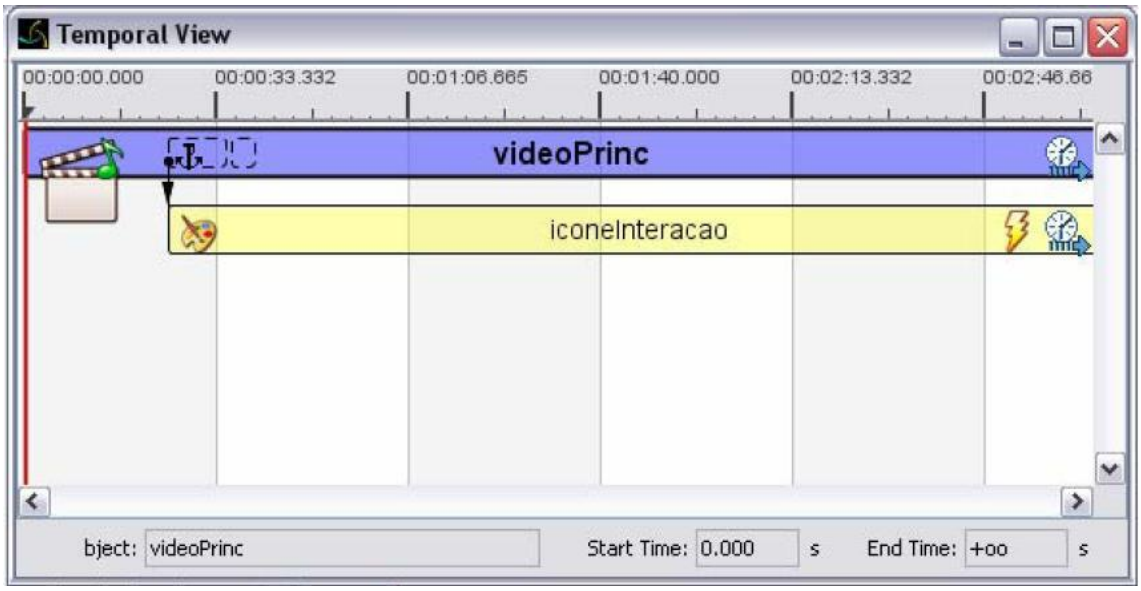

Figura 5: Visão temporal da ferramenta Composer [10].

- Na janela 4 temos a visão textual onde é apresentado o código NCL do documento que está sendo desenvolvido pela ferramenta e nessa visão pode-se alterar o código NCL.

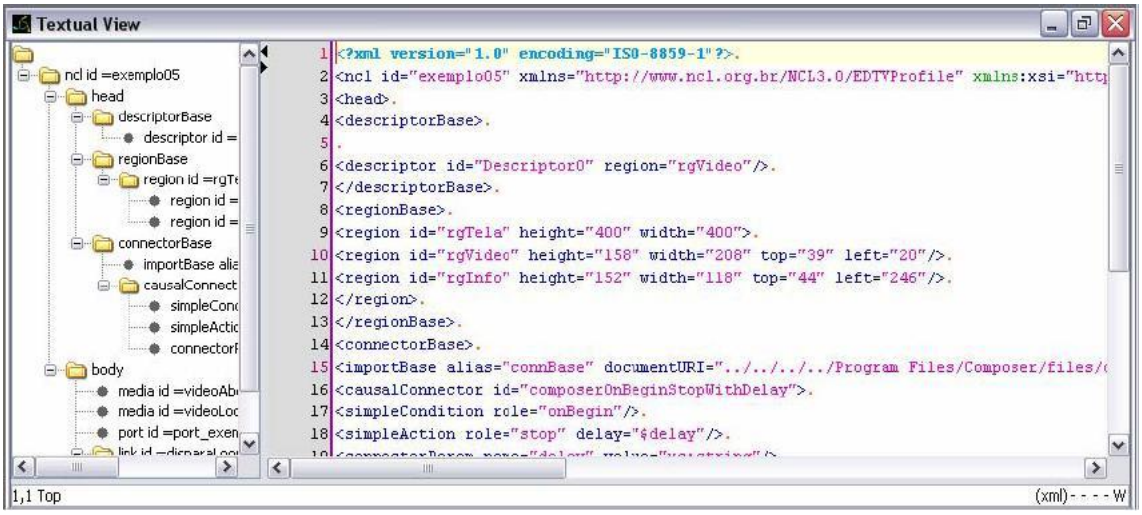

Figura 6: Visão textual da ferramenta Composer [10].

- A integração com a ferramenta de teste Ginga-NCL Emulator facilita a depuração da aplicação aumentando o poder da ferramenta em desenvolver aplicações com um maior nível de interatividade.

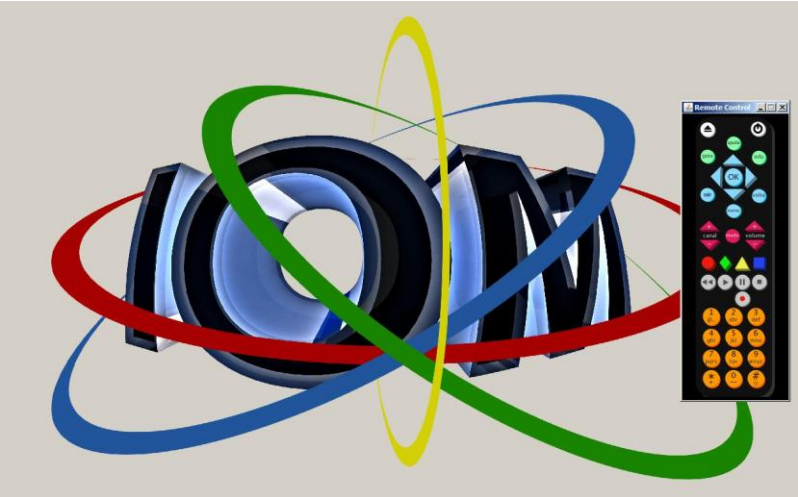

Figura 7: Ginga-NCL Emulator.

Apesar da pluralidade de visões, que facilita na melhor manipulação do documento NCL, a sincronização entre elas não muda o fato de algumas deficiências na ferramenta como: problemas com a disposições dos componentes na interface gráfica, pouca especificação dos erros encontrados pela ferramenta, pouco poder de manipulação dos arquivos do projeto e o baixo desempenho [11].

2.3.2 Gingaway

Gingaway é uma ferramenta especializada no desenvolvimento de aplicações interativas para a TV digital brasileira. O Gingaway é uma extensão para a plataforma Eclipse (Eclipse é uma comunidade *open source*, cujos projetos estão focados em construir uma plataforma de desenvolvimento composta de *frameworks* extensíveis e ferramentas de implementação e gestão de software em todo seu ciclo de vida [12]) que facilita na codificação de aplicações em Ginga-NCL com suporte de scripts em Lua.

Abaixo segue alguns pontos analisados dessa ferramenta:

- Assistente de criação de projetos Ginga-NCL.

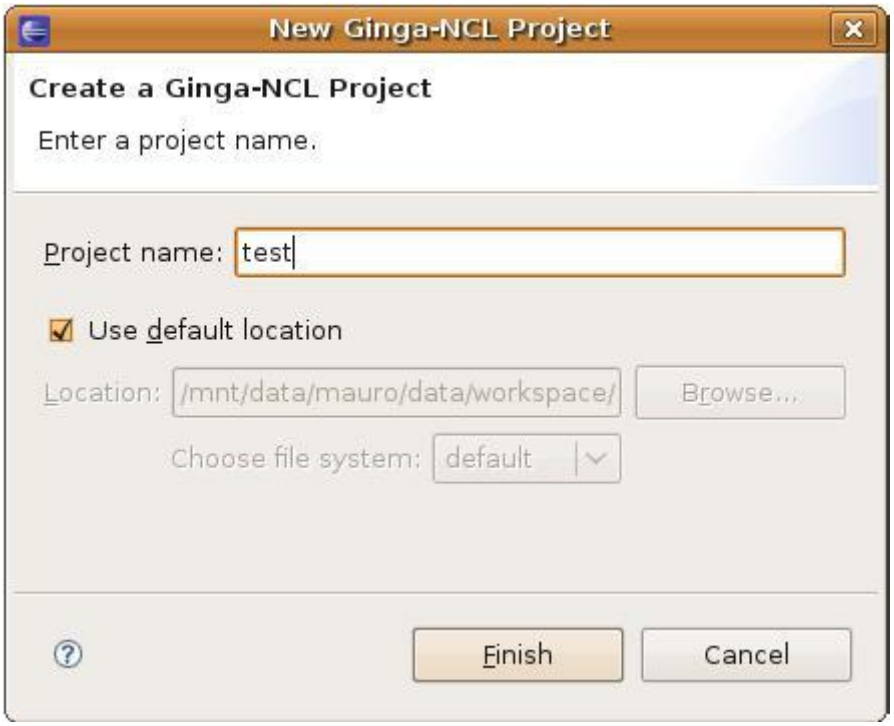

Figura 8: Assistente de criação de projetos Ginga-NCL [11].

Nessa tela pode-se também selecionar o diretório onde o projeto será criado.

- Assistente de criação de documentos NCL.

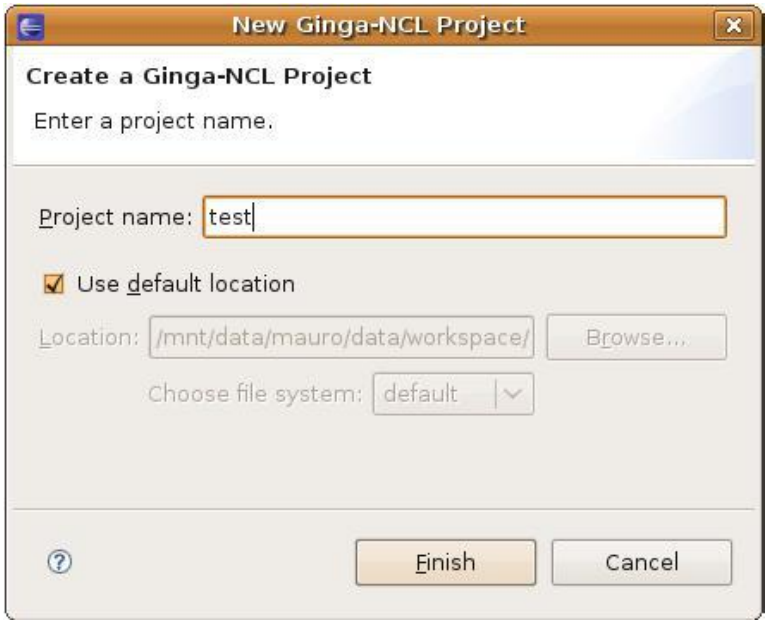

Figura 9: Assistente de criação de documentos NCL [11].

O documento NCL criado é adicionado ao referenciado projeto.

- Assistente de criação de documentos Lua.

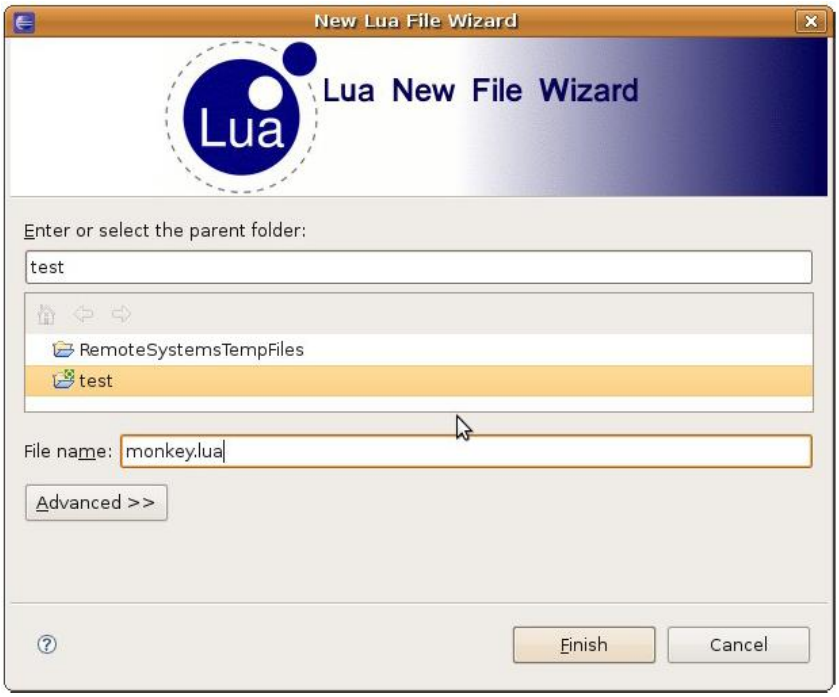

Figura 10: Assistente de criação de documentos Lua [11].

O documento Lua é criado no projeto selecionado.

- Editores de código NCL e Lua, e marcadores de erros que indicam a linha de código onde o erro ocorreu e a natureza do erro.

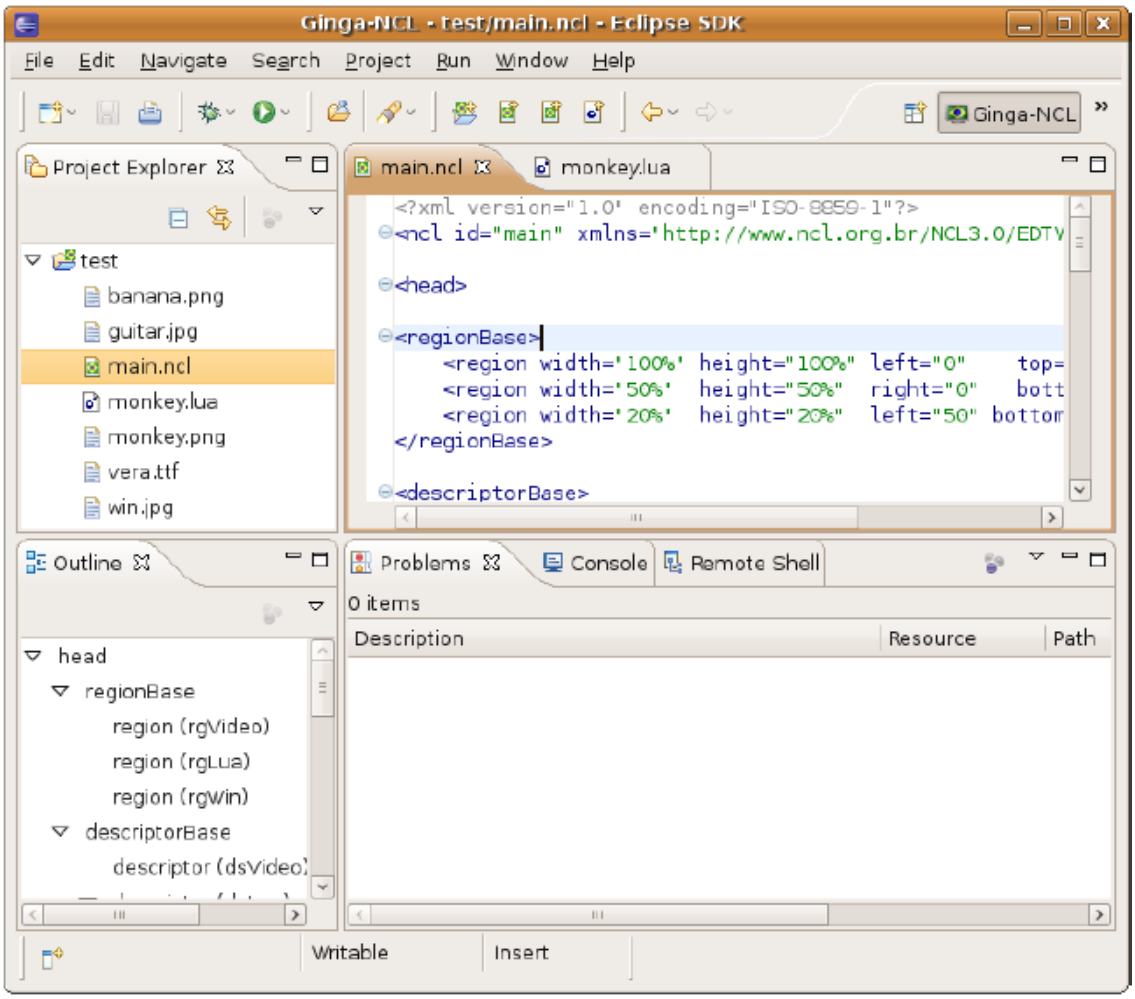

Figura 11: Editor de código NCL do Gingaway [11].

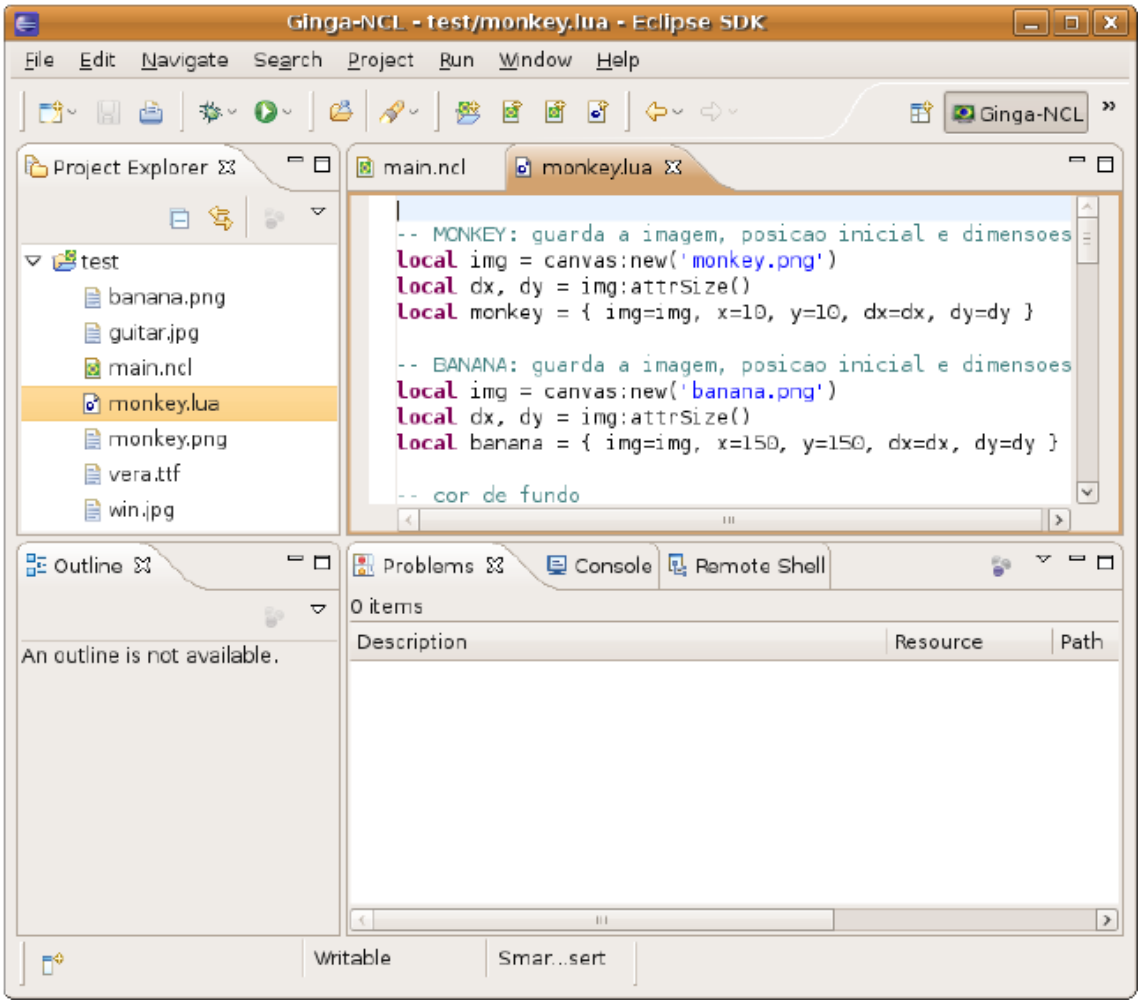

Figura 12: Editor de código Lua do Gingaway [11].

- Assistente de configuração do emulador, onde a aplicação pode ser testada em ambiente simulado ao real, Ginga-NCL Virtual Set-top Box que consiste em uma máquina virtual (esses tipos de simuladores são os que mais se aproximam dos dispositivos reais) executada em linux e implementada em C++.

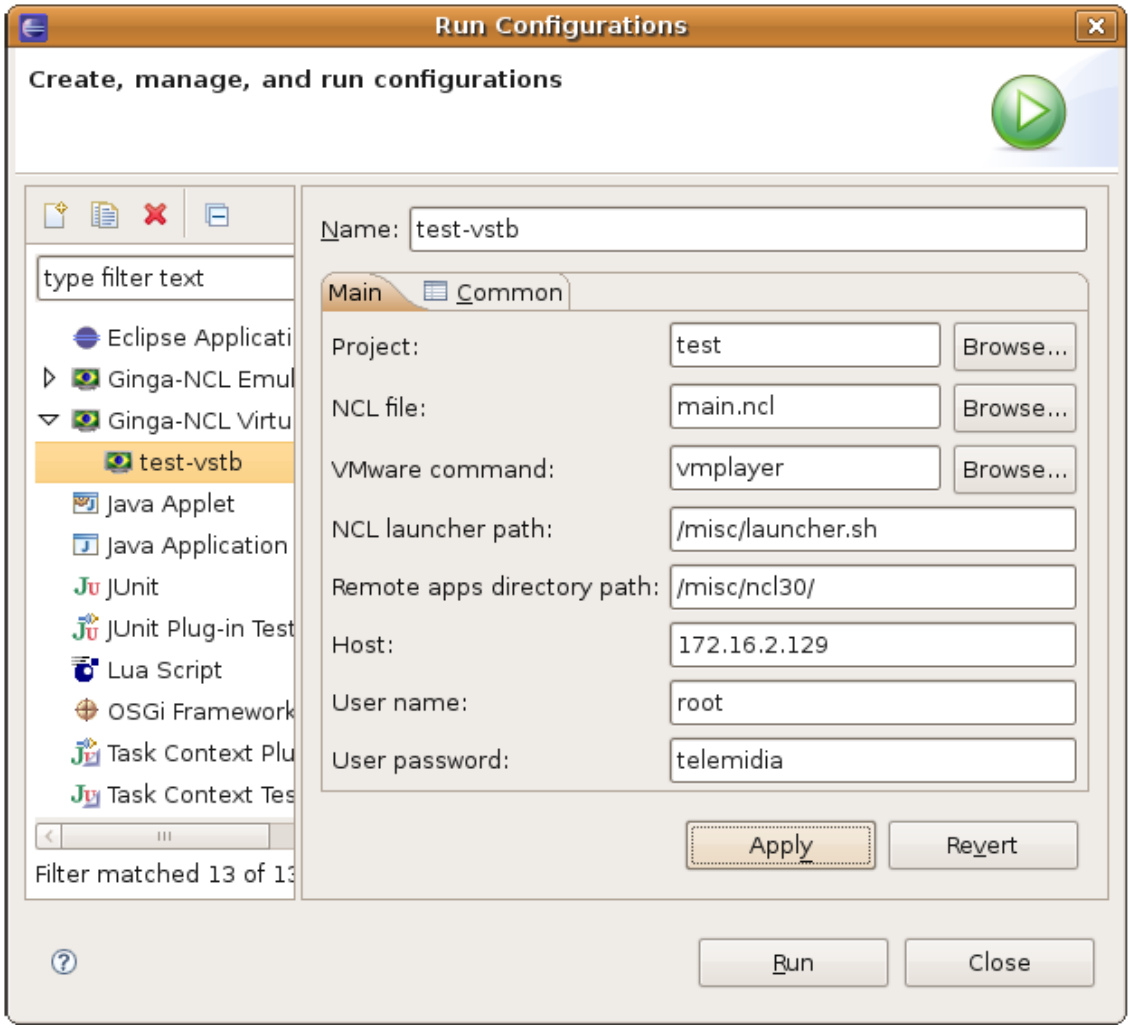

Figura 13: Assistente de configuração do Ginga-NCL Virtual Set-top Box [11].

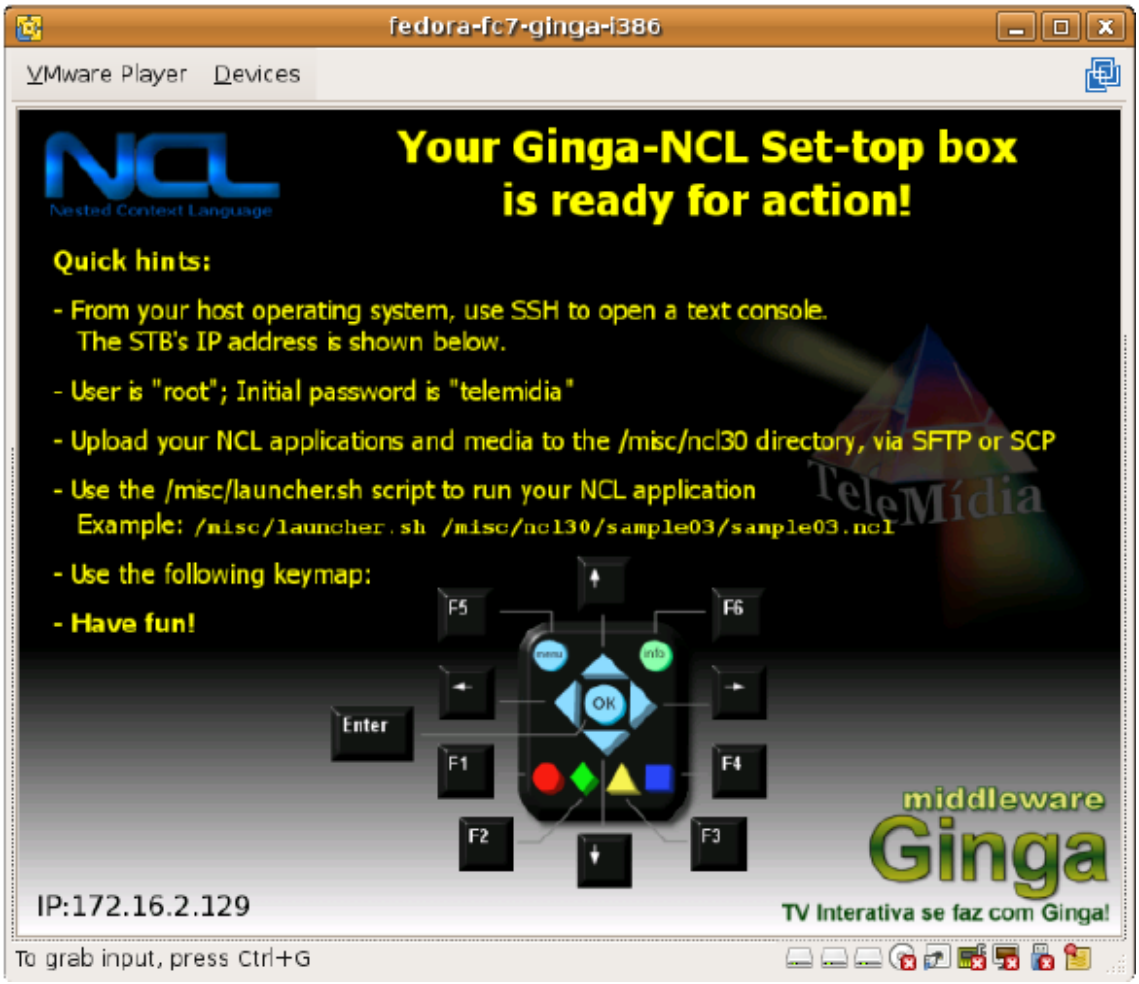

Figura 14: Ginga-NCL Virtual Set-top Box [11].

2.3.3 TVDesigner

O TVDesigner é uma ferramenta que auxilia o processo de desenvolvimento de aplicações de TV digital no padrão MHP – *Multimedia Home Platform*[17]. Essa ferramenta utiliza-se de componentes gráficos e geração de código para otimizar o processo.

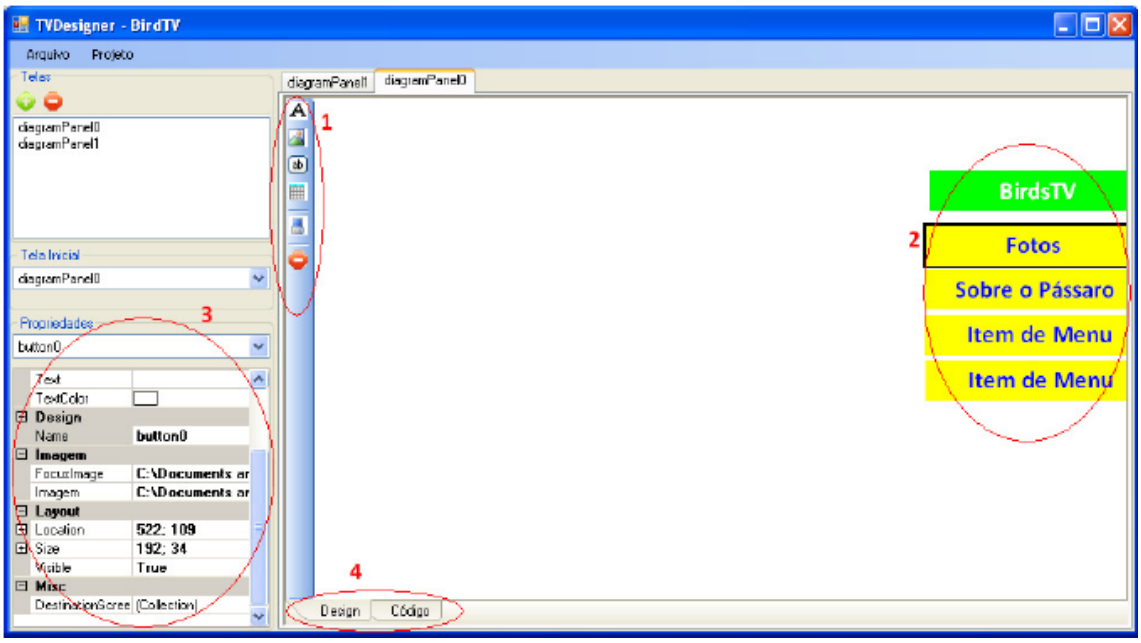

Figura 15: Visão geral do TVDesigner [18].

De acordo com a figura acima, temos uma visão geral da ferramenta. Na região 1 estão dispostos os componentes gráficos que podem ser utilizados no projeto, na região 2 tem-se os componentes adicionados no projeto e que podem ser arrastados para uma melhor posição no leiaute da aplicação, na região 3 são listados os atributos do componente selecionado e que podem ser modificados, e na região 4 estão as abas onde o usuário pode escolher se visualiza o leiaute ou o código gerado pela ferramenta (na visão de código não é possível a edição de código, pois não foi implementado).

Algumas características chaves foram identificadas abaixo:

- Assistente de criação de projeto.

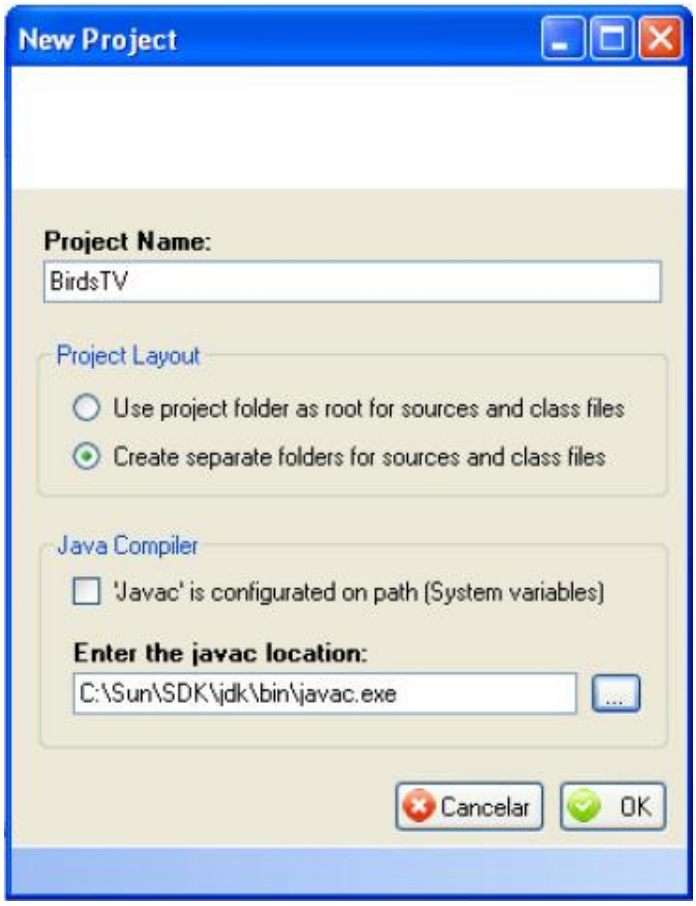

Figura 16: Assistente de criação de projeto no TVDesigner [18].

- Criação e remoção de telas.

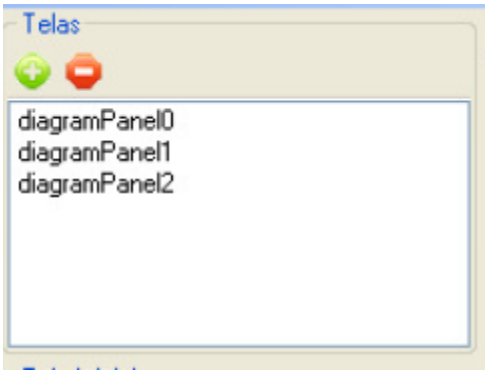

Figura 17: Criação de telas no TVDesigner [18].

- Seleção da tela principal.

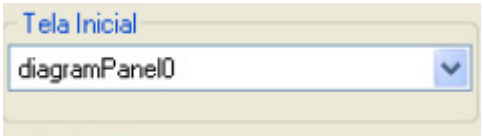

Figura 18: Tela de seleção de tela principal [18].

Nesta tela será selecionada a tela que será apresentada inicialmente quando a aplicação for executada.

- Tela de diagrama de telas e geração de código.

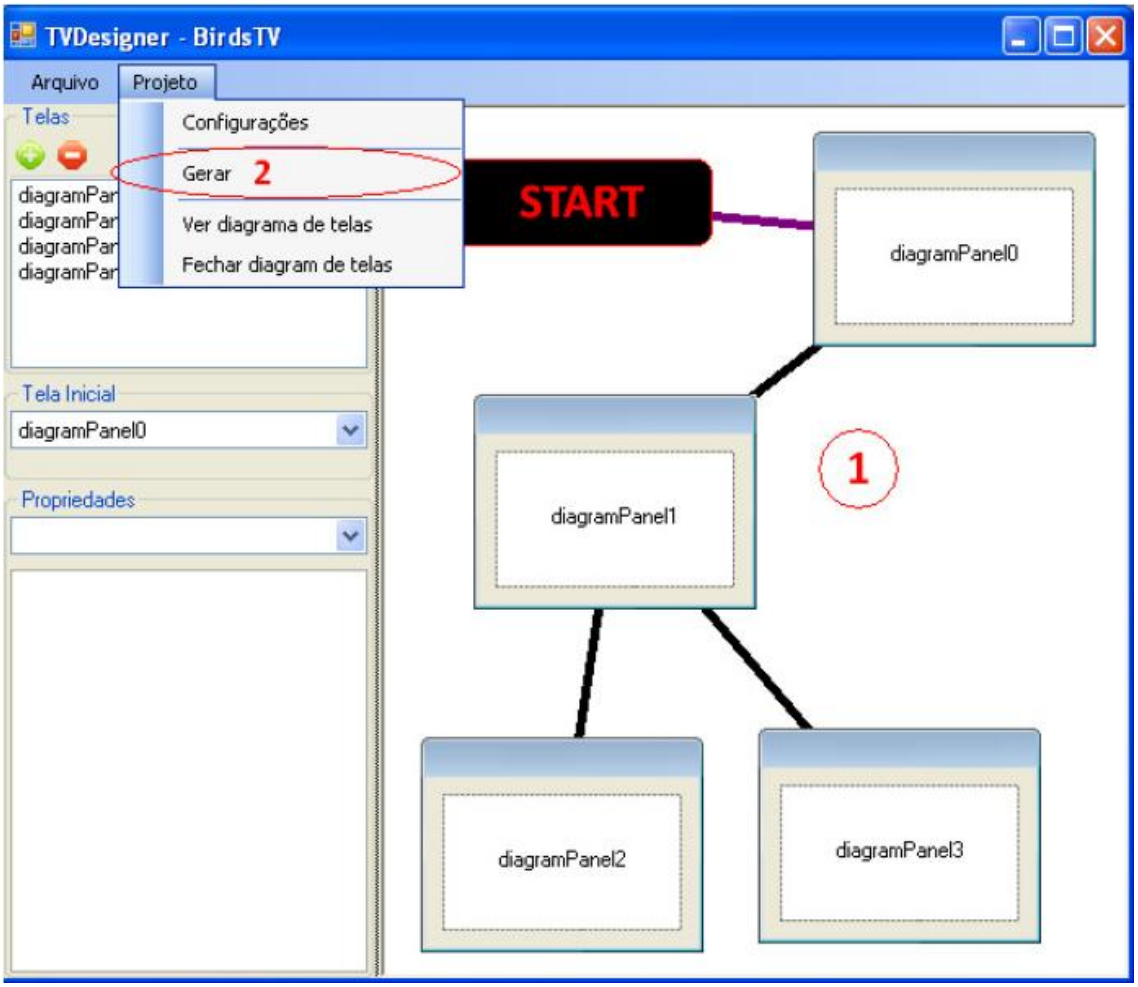

Figura 19: Tela de diagrama de telas e geração de código [18].

Na tela acima é possível observar que na região 1 descreve o diagrama de telas com relação aos componentes do tipo botão que estão dispostos em cada tela e qual tela é referenciada na ação de clique (definida na lista de atributos do componente do tipo botão), e na região 2 está a opção de gerar

aplicação (geração de código e a compilação do código gerado), nesta etapa é quando a aplicação é propriamente criada e logo após pode ser testada em um simulador e executada em um *set-top-box*.

Abaixo segue um quadro comparativo das ferramentas analisadas:

	Pontos Fortes	Pontos Fracos
Composer	• Criação de projetos; • Múltiplas visões; • Edição de documentos NCL; • Integração com ferramentas de teste; • Manipulação gráfica de componentes. • Multiplataforma.	• Não possibilidade de manipulação de documentos Lua; • Interface pouco intuitiva; • Falta de especificação de erros.
Gingaway	• Criação de projetos; • Poder de manipulação e edição de documentos NCL e Lua; • Integração com ferramenta de teste; • Especificação de erros tanto em documentos NCL quanto documentos Lua; • Multiplataforma.	• Falta de manipulação gráfica de componentes.
TVDesigner	• Criação de projetos; • Exportação de projetos; • Manipulação de	• Não segue o padrão do *middleware* brasileiro, GINGA;

componentes gráficos; • Visualização de diagrama de fluxo entre telas; • Visualização de código.	• Poucos componentes pré-definidos; • Não é multiplataforma.

Quadro 1: Quadro comparativo das ferramentas analisadas.

2.3.4 Conclusão

Baseado na análise feita nas sessões anteriores observou-se que as ferramentas têm muitas características importantes que o TVision pode incorporar e algumas falhas também importantes que o TVision pode corrigir.

3. Implementação

Nesta sessão será apresentados os requisitos quais o TVision irá atender, a arquitetura base do sistema e detalhes de implementação da ferramenta.

3.1. Requisitos

Esta sessão tem o objetivo de listar as necessidades que o TVision vai suprir baseadas na experiência que o autor deste teve em participar de um projeto de desenvolvimento de uma aplicação Ginga-NCLua e no teste de outra.

Essas experiências demonstraram que no desenvolvimento de aplicações simples (é definida aplicação simples como uma aplicação tem poucos elementos gráficos e pouca interação com o usuário) é gasto muito tempo com codificação por falta de ferramentas que auxiliem o desenvolvimento do leiaute e da navegação da aplicação.

Com essa necessidade reconhecida e com base nas ferramentas analisadas segue os Requisitos Funcionais que o TVision proporá em suprir:

- [RF01] – O sistema permitirá que o usuário se cadastre;
- [RF02] – O sistema permitirá a criação de projetos NCLua;

- [RF03] – O sistema proverá uma área onde serão apresentados os componentes que o usuário poderá manipular;
- [RF04] – O sistema portará uma região onde o usuário poderá arrastar os componentes, facilitando a arrumação dos componentes no leiaute da aplicação;
- [RF05] – O sistema deverá dar suporte a *upload* de imagens para serem utilizadas nos componentes gráficos;
- [RF06] – O sistema deverá dar suporte a *upload* de vídeos no formato mp4 para serem utilizados como *background* (plano de fundo) para testes no simulador;
- [RF07] – O sistema terá que ter um painel de configurações das propriedades dos componentes gráficos;
- [RF08] – O sistema terá que prever a configuração da ação de clique do componente gráfico do tipo botão;
- [RF09] – O sistema terá que gerar código NCLua para testes da aplicação;
- [RF10] – O sistema proverá o download do projeto codificado gerado pela aplicação;
- [RF11] – O sistema proverá o download da solução do projeto (um código baseado em XML que conterá as características do projeto, mas não é o projeto codificado).

Além dos requisitos funcionais citados acima também foi elicitado os Requisitos Não Funcionais abaixo:

- [RNF01] – O sistema possuirá uma interface gráfica de fácil utilização;
- [RNF02] – O sistema será integrado com a ferramenta de testes;
- [RNF03] – O sistema exibirá o assistente de ajuda ao usuário quando este requisitar.

3.2. Arquitetura

A arquitetura do projeto será em camadas, que permite a cada camada uma interface de serviços por onde as camadas superiores acessar as funcionalidades, com algumas abordagens de tratamento de eventos para

alguns procedimentos que são assíncronos, com isso tornando o sistema modulado e facilitando futuras modificações.

Figura 20: Diagrama da arquitetura em camadas.

O banco de dados do sistema basear-se-á na arquitetura relacional e sua estrutura será descrita na figura abaixo:

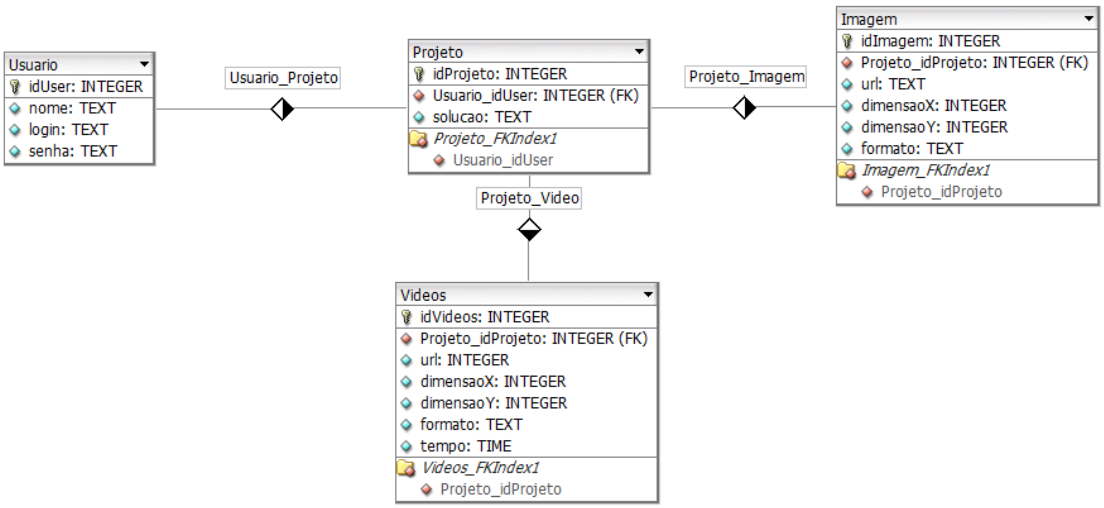

Figura 21: Estrutura do banco de dados do TVision.

De acordo com o diagrama das tabelas acima mostra que o usuário pode ser registrado no sistema através de *login* e senha. O usuário pode criar um ou quantos projetos quiser, nesse projeto também o usuário poderá importar imagens e vídeos que ficarão relacionados unicamente ao projeto que foi importado, e o projeto terá a sua solução armazenada no banco de dados para futuras modificações.

O diagrama de abaixo demonstra o fluxo de dados entre a aplicação e o servidor web:

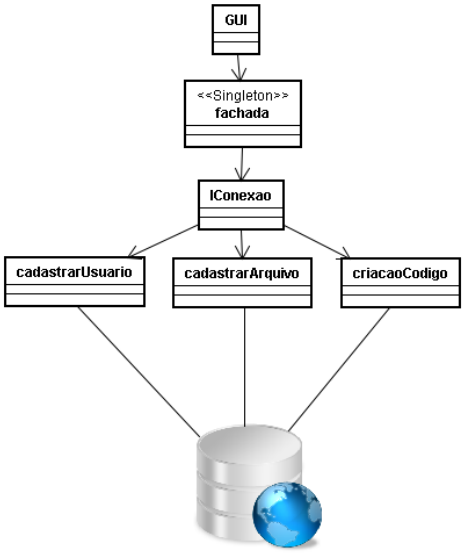

Figura 22: Diagrama de conexão com o servidor.

O diagrama acima demonstra como será feita a conexão com o servidor web (nesse servidor web estarão códigos em php que auxiliarão a execução dos requisitos, já que as aplicações em Flash possuem limitações técnicas). Neste exemplo notar-se que na camada de comunicação existe uma interface IConexao que lista as funcionalidades que a camada de superior irá utilizar, tornando o projeto assim fiel a arquitetura definida e torna o código modularizado.

Com essas descrições fica definida a estrutura base da ferramenta.

3.3. Detalhes de Desenvolvimento

Um dos principais detalhes no desenvolvimento do TVision começa pela IDE Flex Builder que foi construída baseada no Eclipse, isso permitiu uma maior rapidez na implementação tendo visto que o autor já possui grande experiência tanto com o Eclipse quanto com o próprio Flex Builder.

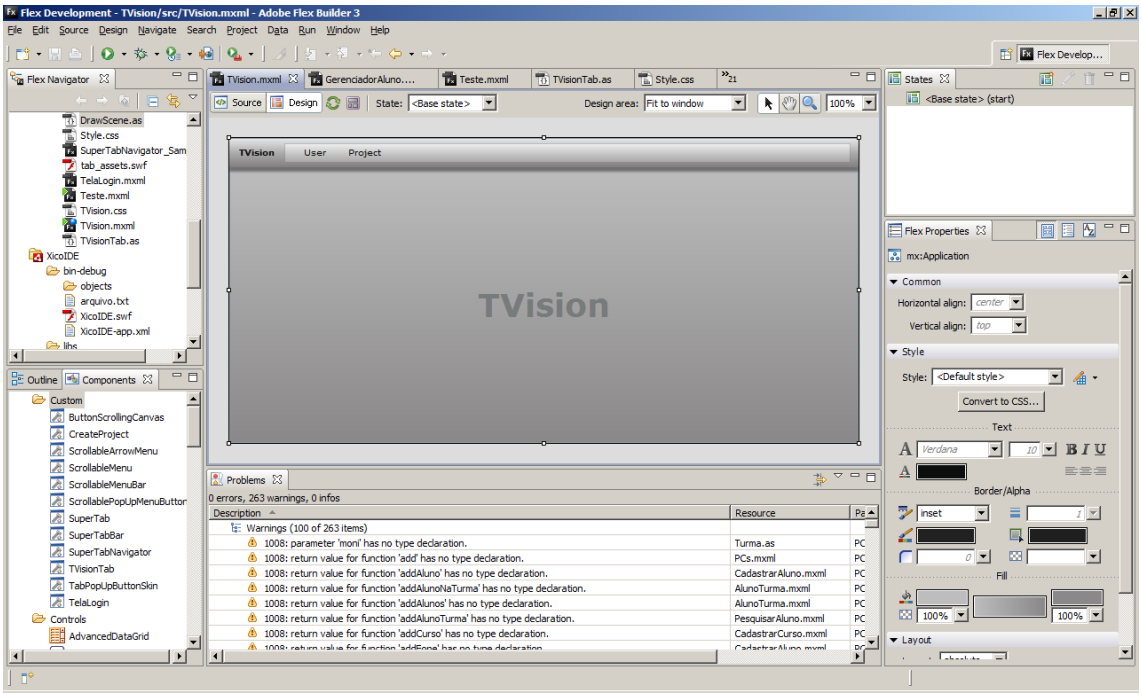

Figura 23: Flex Builder IDE.

No decorrer do projeto tive que definir alguns componentes que fosse possível manipular suas propriedades e que pudessem ser implementados no Ginga-NCLua. Baseado nisso defini os três componentes abaixo:

- Campo de Texto: Componente em que o usuário poderá escrever o texto em que será apresentado na aplicação. O usuário pode configurar a posição do componente e o tipo da fonte;
- Botão: Componente em que o usuário pode configurar a sua posição na aplicação, a imagem que comporá o componente e a ação do clique do botão (essa ação indica qual tela será selecionada);
- Imagem: Componente em que o usuário pode configurar a sua posição na aplicação e adicionar a imagem associada.

A conexão com o banco de dados se foi feita através de códigos em PHP (linguagem de script de internet que é incorporada no documento HTML) [19] utilizando a implementação da biblioteca AMFPHP (conjunto de classes em PHP que faz a comunicação de aplicações Flash com o servidor) [20].

Além da conexão com o banco de dados o PHP foi responsável também pela criação dos arquivos do projeto (o documento NCL e o documento Lua) e pelo *download* dos recursos (imagens e vídeos necessários para o funcionamento do projeto) e arquivos do projeto.

No decorrer do projeto ocorreram alguns problemas que serviram de aprendizado. Um desses problemas foi a falta de capacidade de aplicações em Flash de não poder gerar arquivos para *download* e com isso foi necessário a utilização de scripts em php para prover esse serviço, outro grande problema encontrado foi a falta de suporte técnico para o desenvolvimento, por exemplo, o servidor utilizado, o servidor do Centro de Informática da Universidade Federal de Pernambuco, não dava permissão de criação de arquivos via script php e não oferecia a possibilidade de comandos de compactação de arquivos (compactação em arquivos .zip, .tar e etc., que nesse caso prejudicou o *download* dos arquivos do projeto para o computador do usuário), que forçou o autor a utilização de formas alternativas de implementação para atender os requisitos solicitados.

Uma das facilidades foi o desenvolvimento com o Flex Builder IDE, pois essa ferramenta possui um editor de visual muito poderoso que permite a fácil manipulação de componentes padrões (já definidos no Flex Builder) e componentes customizados, o que contribui para que a tecnologia tenha a interface gráfica rica.

4. Estudo de Caso

Neste capítulo definirei um protótipo de aplicação para TV Digital no qual será implementado utilizando o TVision.

4.1. Definição

A aplicação será uma propaganda de celulares em que terá imagens de celulares e a descrição dos mesmos e será chamada de PropCel. Abaixo segue o fluxo de navegação das telas da aplicação:

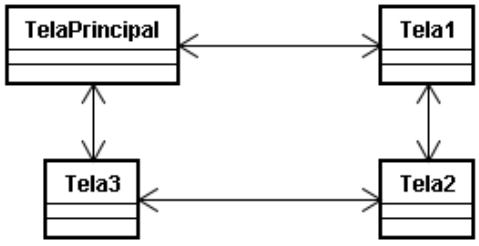

Figura 24: Fluxo de navegação das telas do PropCel.

Como mostrado na Figura 24 o PropCel terá uma tela principal que conterá a descrição de um produto com dois botões que indicam que o usuário poderá percorrer a aplicação em dois sentidos e em cada tela selecionada existirá sempre dois sentidos de percorrer a aplicação.

4.2. Utilizando o TVision

Com a definição do protótipo PropCel feita, o próximo passo é a implementação utilizando o TVision. Segue abaixo uma visão geral da ferramenta:

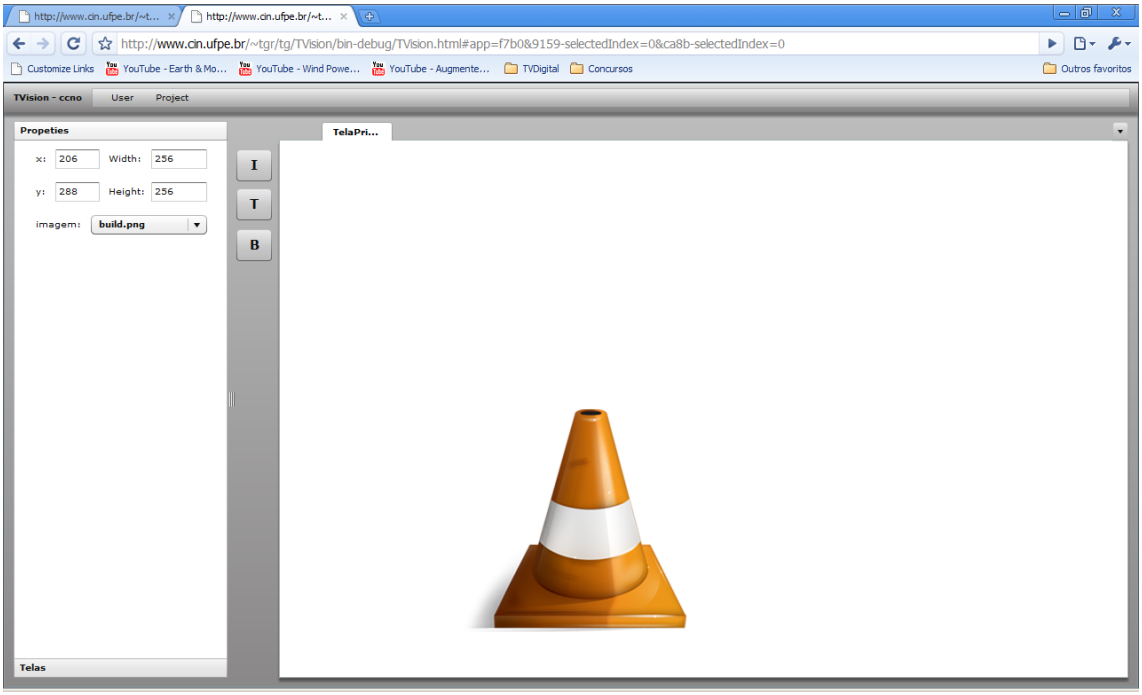

Figura 25: Visão geral do TVision.

4.2.1. Entrar no sistema

Inicialmente é necessário que o usuário inicie algum programa que possibilite a navegação em *sites* na internet (*browsers*) e acessar o esse endereço de internet: <http://www.cin.ufpe.br/~tgr/tg/TVision/bin-debug/TVision.html>. Logo que entrar no sistema é necessário que o usuário faça o *login* com mostrado na imagem abaixo:

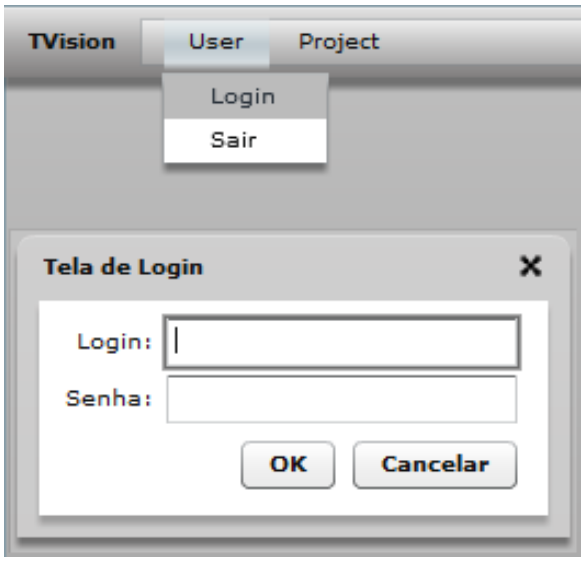

Figura 26: Tela de *login* do TVision.

Nesta tela é necessário que o usuário insira nos campos de texto o seu login e a senha previamente cadastrados pelo administrador do sistema.

4.2.2. Criar Projeto

Logo após entrar no sistema é necessário que o usuário crie um projeto como mostrado na figura a seguir:

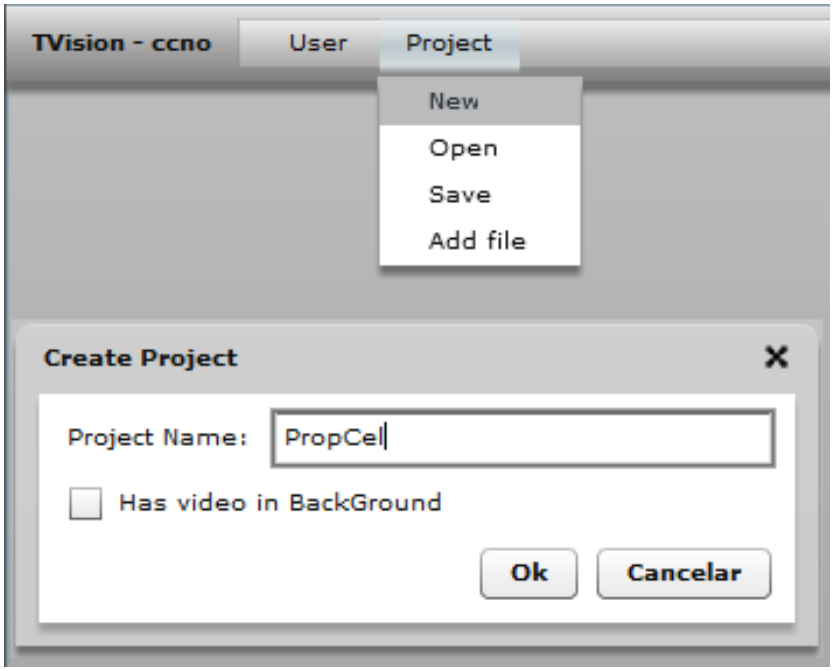

Figura 27: Tela de Criação de projeto no TVision.

Na tela de criação de projeto é necessário que o usuário insira o nome do projeto, neste caso específico será PropCel, e também existe a possibilidade de inserir um vídeo de plano de fundo para testar questões de transparências de algumas imagens que serão inseridas no projeto.

4.2.3. Criar Telas

O próximo passo seria a criação das telas que se dá através de um botão na região de configuração de telas. Abaixo segue as imagens que detalham o processo de criação de telas:

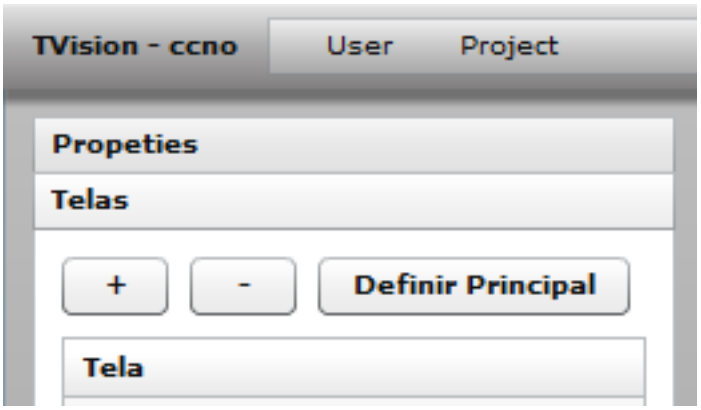

Figura 28: Primeiro passo na criação de telas no TVision.

Neste primeiro passo é necessário que na região de configuração de telas o usuário adicione uma tela ao projeto clicando no botão “+”, como mostrado acima.

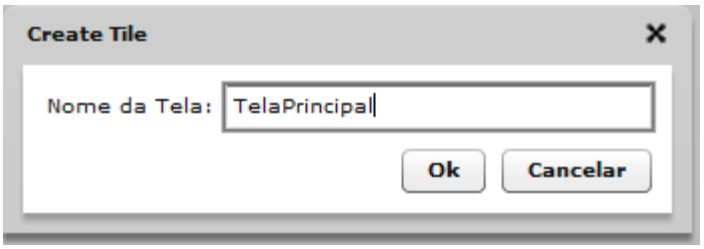

Figura 29: Segundo passo na criação de tela no TVision.

Neste passo é necessário que o usuário insira no campo de texto o nome da tela para que possa ser editada.

Depois da tela criada, agora basta configurar os componentes na área de edição. A imagem abaixo mostra a região de edição junto com os componentes a serem utilizados:

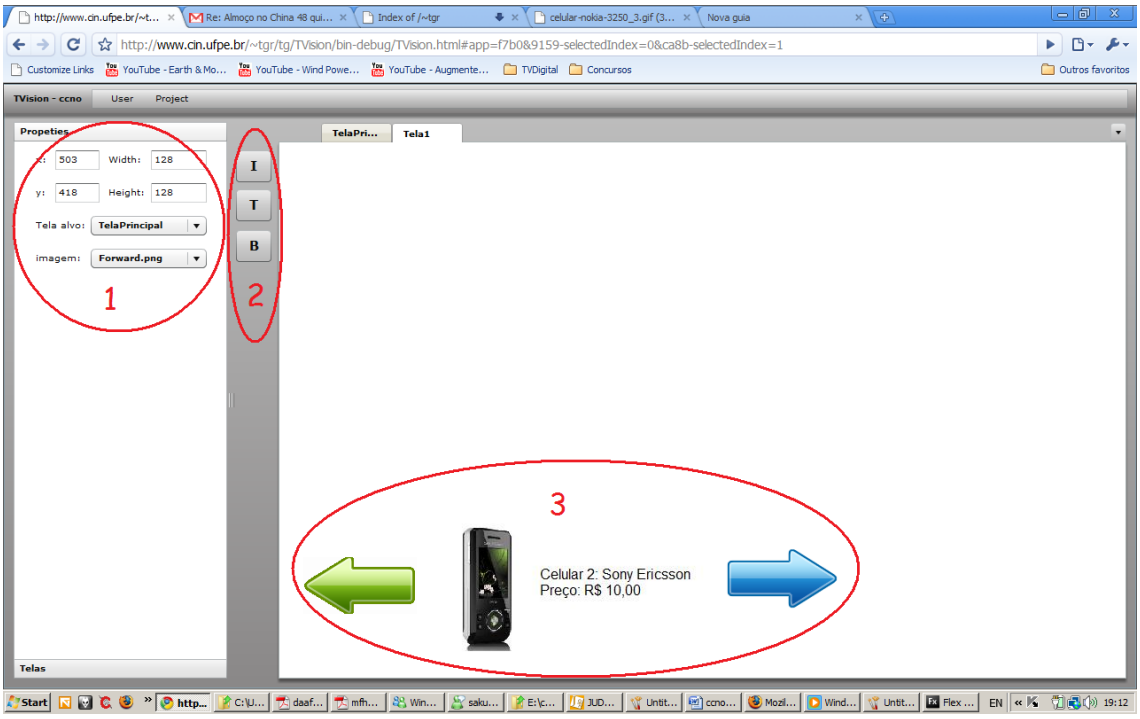

Figura 30: Descrição da área de edição do TVision.

Na região de numero 1 está a lista de atributos do componente selecionado, nesses atributos estão, por exemplo, a posição nos eixos x e y, a imagem que será carregada no componente e no caso de botão, para qual tela será redirecionada a aplicação quando a ação de clique for executada.

Na região de numero 2 estão os componentes (botão “I” é o componente imagem, botão “T” é o componente campo de texto e o botão “B” é o componente botão, definidos na sessão 4.1) que podem ser carregados na região 3 onde os componentes são carregados.

4.2.4. Gerar Código do Projeto

Com o projeto já estruturado é necessário definir qual será a tela principal, e para isso, na tela de configuração de telas o usuário terá que selecionar a tela e clicar no botão “Definir Principal” como mostra a figura abaixo:

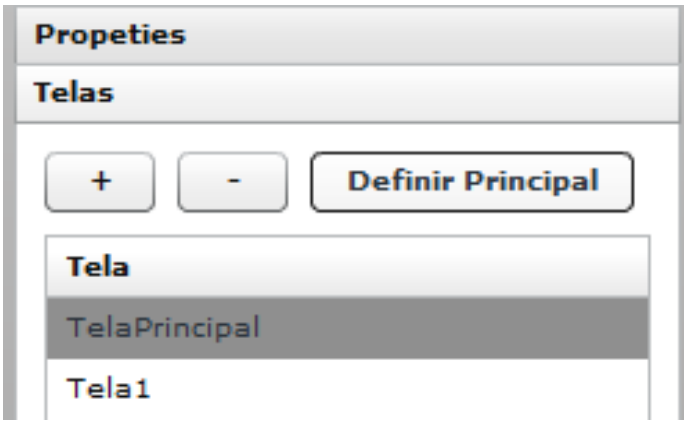

Figura 31: Definição de tela principal no TVision.

Após essa definição é necessário que o usuário selecione a opção de exportar no menu principal como na imagem abaixo:

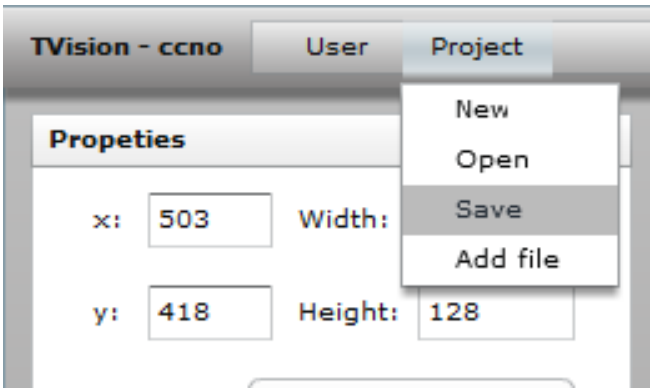

Figura 32: Exportar projeto no TVision.

Após esta ação todos os arquivos serão baixados no computador do usuário. E assim ele poderá executar em um simulador para testes.

É notável que o menu não é intuitivo, mas esse pequeno erro foi ocasionado por falta de revisão e falta de tempo para corrigir o projeto.

Os códigos gerados estarão disponíveis nos apêndices desse trabalho.

4.2.5. Executar o Projeto no Simulador

Nesta etapa com o projeto baixado no computador do usuário é necessário a execução no simulador para testar as funcionalidades da aplicação. Neste caso iremos testar a navegação entre as telas criadas. Os passos abaixo mostram as passagens entre as telas:

- TelaPrincipal:

Figura 33: TelaPrincipal do PropCel.

- Tela1:

Figura 34: Tela1 do PropCel.

- Tela2:

Figura 35: Tela2 do PropCel.

- Tela3:

Figura 36: Tela3 do PropCel.

O projeto foi executado no simulador Ginga virtual *set-top-box* e o fluxo entre as telas ocorreu como previsto confirmando assim a eficiência do TVision.

5. Conclusões

Neste trabalho foram apresentadas conceitos de TV digital, necessidades de mercado a serem atendidas e por final foi apresentada a ferramenta de desenvolvimento de aplicações para TV digital, baseada padrão GINGA, TVision. E com isso cumprindo com o seu objetivo.

A implementação foi marcada por problemas na comunicação da aplicação com o servidor web, problemas com permissões em manipular arquivos no servidor e problemas com compactação de arquivos viam código.

Apesar de todos os problemas encontrados, que tardou o desenvolvimento da ferramenta e impossibilitaram o a realização de alguns requisitos, a experiência de trabalhar em um projeto que envolve TV digital e tecnologia web serviram como aprendizado.

O Quadro 2 relaciona quais requisitos foram, não foram ou foram parcialmente atendidos:

Requisito	Situação
[RF01]	Atendido
[RF02]	Atendido
[RF03]	Atendido
[RF04]	Atendido
[RF05]	Atendido
[RF06]	Atendido
[RF07]	Atendido
[RF08]	Atendido
[RF09]	Atendido
[RF10]	Atendido
[RF11]	Não Atendido
[RNF01]	Parcialmente Atendido
[RNF02]	Não Atendido
[RNF03]	Não Atendido

Quadro 2: Lista de requisitos e situação.

Como mostrado no Quadro 2 o TVision atendeu a maioria dos requisitos elicitados. O TVision não atendeu aos requisitos [RF11], [RNF02] e [RNF03]

por falta de tempo para o desenvolvimento da ferramenta pois as viabilidades técnicas foram estudadas e os requisitos são implementáveis. E o requisito [RNF01] foi parcialmente atendido também por falta de tempo para correções mais refinadas na ferramenta.

5.1. Trabalhos Futuros

Como sugestões proponho continuidade deste trabalho com a implementação dos requisitos que foram parcialmente atendidos e não atendidos, a melhoria da estrutura e do código.

Umas das principais melhorias deste trabalho seriam: a aquisição de um servidor que facilite a comunicação com a ferramenta, uma revisão na usabilidade do sistema e a integração com a ferramenta de teste, ginga virtual *set-top-box*.

Sugiro também algumas funcionalidades a serem adicionadas em trabalhos futuros como:

- Adicionar o padrão Ginga-J (padrão GINGA com programação em Java);
- Adicionar uma visão de edição de código e a cada modificação do código o editor visual também ser modificado;
- Adicionar mais componentes à ferramenta para aumentar a complexidade das aplicações geradas pelo TVision;
- Maior suporte na manipulação dos arquivos de recurso do projeto;
- Estudar a viabilidade de componentes que acessam o canal de retorno.

Referências

[1] – IBGE – Percentual de domicílios com alguns bens duráveis e serviços de acesso a comunicação. Disponível em <http://www.ibge.gov.br/home/presidencia/noticias/noticia_visualiza.php?id_noticia=1230&id_pagina=>. Acessado em: Abril de 2009.

[2] – Midiativa – Brasileiro vê TV mais de 5 horas por dia. Disponível em: <http://www.midiativa.org.br/index.php/midiativa/content/view/full/2717>. Acessado em: Maio de 2009.

[3] – Dataprev – Empresa de Tecnologia e Informacao da Previdencia Social, Cidades que já possuem sinal digital de TV. Disponível em: <http://www.softwarelivre.serpro.gov.br/recife/download-plaestras/Edson%20Luiz%20e%20Marco%20Munhoz%20-%20TV%20Digital%20-%20Social.pdf>. Acessado em: Junho de 2009.

[4] – RNP, GT Middleware. Disponível em: <http://www.rnp.br/pd/gts2004-2005/middleware.html>. Acessado em: Maio de 2009.

[5] – Ginga. Disponível em: <http://www.ginga.org.br/>. Acessado em: Maio de 2009.

[6] – NCL - *Nexted Context Language*. Disponível em: <http://www.ncl.org.br/>. Acessado em: Maio de 2009.

[7] – Norma padrão ABNT - Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 2: Ginga-NCL para receptores fixos e móveis - Linguagem de aplicação XML para codificação de aplicações. Disponível em: < http://www.abnt.org.br/imagens/Normalizacao_TV_Digital/ABNTNBR15606-5_2008Ed1.pdf>. Acessado em: Maio de 2009.

[8] – A Linguagem de Programação Lua. Disponível em: <http://www.lua.org/portugues.html>. Acessado em: Maio de 2009.

[9] – GUIMARAES, R. L., Composer: um ambiente de autoria de documentos NCL para TV digital interativa, dissertação de mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2007. P. 17.

[10] – Telemídia – PUC-Rio. Construindo Programas Audiovisuais Interativos Utilizando a NCL 3.0 e a Ferramenta Composer. Disponível em: <www.ncl.org.br/documentos/TutorialNCL3.0-2ed.pdf>. Acessado em Maio de 2009.

[11] – BEUTRAO FILHO, M. F. H., Gingaway – Uma ferramenta para criação de aplicações GINGA-NCL interativas para TV Digital, trabalho de graduação, Centro de Informática – Universidade Federal de Pernambuco.

[12] – Eclipse.org home. Disponível em: <http://www.eclipse.org/>. Acessado em: Abril de 2009.

[13] – NADA, T. e Helwig S., Rich Internet Applications - Technical Comparison and Case Studies of AJAX, Flash, and Java based RIA, University of Wisconsin-Madison. Disponível em: <http://www.uwebc.org/newsletter/downloads/11_16_2005/RIA.pdf>. Acessado em: Abril de 2009.

[14] – Adobe Developer Connection - Flex Quick Start: Getting Started. Disponível em: <http://www.adobe.com/devnet/flex/quickstart/coding_with_mxml_and_actionscript/>. Acessado em Maio de 2009.

[15] – Ginga-NCL Virtual Set-top Box. Disponível em: < http://www.gingancl.org.br/ferramentas.html>. Acessado em: Maio de 2009.

[16] – ABAP – Associação Brasileira de Agências de Publicidade / IBGE, Números Oficias da Indústria da Comunicação e seu Impacto na Economia Brasileira. Disponível em: <http://webserver.4me.com.br/wwwroot/abap/ibge.pdf>. Acessado em Abril de 2009.

[17] – MHP - Multimedia Home Platform Specification. Disponível em: <http://www.mhp.org/>. Acessado em: Maio de 2009.

[18] – ARAUJO FILHO, D. A., TVDesigner: Uma Ferramenta para Criação de Aplicações MHP Interativas para TV Digital, trabalho de graduação, Centro de Informática – Universidade Federal de Pernambuco.

[19] – PHP – Hypertext Preprocessor. Disponível em: <http://www.php.net/>. Acessado em: Junho de 2009.

[20] – AMFPHP – Action Message Format. Disponível em: <http://www.amfphp.org/>. Acessado em: Junho de 2009.

Apêndice I – Código NCL Gerado

O código abaixo foi retirado do arquivo propcel.ncl que foi gerado pelo TVision e baixado no computador do autor.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="PropCel" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
<head>
<regionBase>
	<region width="100%" height="100%" left="0"    top="0"     id="rgVideo"/>
	<region width="100%"  height="30%"  left="5%"   bottom="0"
id="rgLua"/>
</regionBase>
<descriptorBase>
	<descriptor id="dsVideo" region="rgVideo"/>
	<descriptor id="dsLua" region="rgLua" focusIndex="luaIdx">
		<descriptorParam name="transparency" value="0.6"/>
	</descriptor>
</descriptorBase>
<connectorBase>
	<causalConnector id="onBeginStart">
		<simpleCondition role="onBegin"/>
		<simpleAction role="start"/>
	</causalConnector>
</connectorBase>
</head>
<body>
	<port id="entryPoint" component="video"/>
	<media type="application/x-ginga-settings" id="programSettings">
		<property name="currentKeyMaster" value="luaIdx"/>
	</media>
	<media id="video" src="/misc/video-samples/corrego1.mp4"
descriptor="dsVideo"/>
```

```
        <media id="lua" src="propcel.lua" descriptor="dsLua"/>
        <link xconnector="onBeginStart">
            <bind role="onBegin" component="video"/>
            <bind role="start"    component="lua"/>
        </link>
</body>
</ncl>
```

Apêndice II – Código Lua Gerado

O código abaixo foi retirado do arquivo propcel.lua que foi gerado pelo TVision e baixado no computador do autor.

```
tela = 0
focus = 0
enter = 0

local img_b1 = canvas:new("Back.png");
local dx, dy = img_b1:attrSize();
local button1 =  { img=img_b1, x=10, y=50, dx=dx, dy=dy, focus=0, enter=3 }

local img_b2 = canvas:new("Forward.png")
local dx, dy = img_b2:attrSize()
local button2 =  { img=img_b2, x=500, y=50, dx=dx, dy=dy, focus=1, enter=1 }

local img_i1 = canvas:new("celular1.jpg")
local dx, dy = img_i1:attrSize()
local img1 =  { img=img_i1, x=170, y=10, dx=dx, dy=dy }
------------------------------------------------------------------------
local img_b3 = canvas:new("Back.png");
local dx, dy = img_b3:attrSize();
local button3 =  { img=img_b3, x=10, y=50, dx=dx, dy=dy, focus=0, enter=0 }

local img_b4 = canvas:new("Forward.png")
local dx, dy = img_b4:attrSize()
local button4 =  { img=img_b4, x=500, y=50, dx=dx, dy=dy, focus=1, enter=2 }

local img_i2 = canvas:new("celular2.jpg")
local dx, dy = img_i2:attrSize()
local img2 =  { img=img_i2, x=170, y=10, dx=dx, dy=dy }
------------------------------------------------------------------------
```

```
local img_b5 = canvas:new("Back.png");
local dx, dy = img_b5:attrSize();
local button5 =  { img=img_b5, x=10, y=50, dx=dx, dy=dy, focus=0, enter=1 }

local img_b6 = canvas:new("Forward.png")
local dx, dy = img_b6:attrSize()
local button6 =  { img=img_b6, x=500, y=50, dx=dx, dy=dy, focus=1, enter=3 }

local img_i3 = canvas:new("celular3.jpg")
local dx, dy = img_i3:attrSize()
local img3 =  { img=img_i3, x=170, y=10, dx=dx, dy=dy }
--------------------------------------------------------------------------
local img_b7 = canvas:new("Back.png");
local dx, dy = img_b7:attrSize();
local button7 =  { img=img_b7, x=10, y=50, dx=dx, dy=dy, focus=0, enter=2 }

local img_b8 = canvas:new("Forward.png")
local dx, dy = img_b8:attrSize()
local button8 =  { img=img_b8, x=500, y=50, dx=dx, dy=dy, focus=1, enter=0 }

local img_i4 = canvas:new("celular4.gif")
local dx, dy = img_i4:attrSize()
local img4 =  { img=img_i4, x=170, y=10, dx=dx, dy=dy }
--------------------------------------------------------------------------
local tela1 = { button1, button2, foco=0, lenght=2 }
local tela2 = { button3, button4, foco=0, lenght=2 }
local tela3 = { button4, button6, foco=0, lenght=2 }
local tela4 = { button7, button8, foco=0, lenght=2 }

local telas = {}
telas[0] = tela1
telas[1] = tela2
telas[2] = tela3
```

```
telas[3] = tela4

function refreshScreen ()

	canvas:attrColor('white')
	canvas:drawRect("fill", 0, 0, canvas:attrSize());
	if tela == 0 then
		canvas:compose(button1.x, button1.y, button1.img);
		canvas:compose(img1.x, img1.y, img1.img);
		canvas:compose(button2.x, button2.y, button2.img);
		tela1.foco = focus
		canvas:attrColor('black');
		if tela1.foco == 0 then
			canvas:drawRect("frame", button1.x, button1.y, button1.dx,
button1.dy);
		elseif tela1.foco == 1 then
			canvas:drawRect("frame", button2.x, button2.y, button2.dx,
button2.dy);
		end
		if tela1.foco == 0 and enter == 1 then
			enter = 0
			tela = button1.enter
			refreshScreen ()
		elseif tela1.foco == 1 and enter == 1 then
			enter = 0
			tela = button2.enter
			refreshScreen ()
		end
	elseif tela == 1 then
		canvas:compose(button3.x, button3.y, button3.img);
		canvas:compose(img2.x, img2.y, img2.img);
		canvas:compose(button4.x, button4.y, button4.img);
		tela2.foco = focus
```

```
                canvas:attrColor('black');
                if tela2.foco == 0 then
                        canvas:drawRect("frame", button3.x, button3.y, button3.dx,
button3.dy);
                elseif tela2.foco == 1 then
                        canvas:drawRect("frame", button4.x, button4.y, button4.dx,
button4.dy);
                end
                if tela2.foco == 0 and enter == 1 then
                        enter = 0
                        tela = button3.enter
                        refreshScreen ()
                elseif tela2.foco == 1 and enter == 1 then
                        enter = 0
                        tela = button4.enter
                        refreshScreen ()
                end
        elseif tela == 2 then
                canvas:compose(button5.x, button5.y, button5.img);
                canvas:compose(img3.x, img3.y, img3.img);
                canvas:compose(button6.x, button6.y, button6.img);
                tela3.foco = focus
                canvas:attrColor('black');
                if tela3.foco == 0 then
                        canvas:drawRect("frame", button5.x, button5.y, button5.dx,
button5.dy);
                elseif tela3.foco == 1 then
                        canvas:drawRect("frame", button6.x, button6.y, button6.dx,
button6.dy);
                end
                if tela3.foco == 0 and enter == 1 then
                        enter = 0
                        tela = button5.enter
```

```
                refreshScreen ()
            elseif tela3.foco == 1 and enter == 1 then
                enter = 0
                tela = button6.enter
                refreshScreen ()
            end
        elseif tela == 3 then
            canvas:compose(button7.x, button7.y, button7.img);
            canvas:compose(img4.x, img4.y, img4.img);
            canvas:compose(button8.x, button8.y, button8.img);
            tela4.foco = focus
            canvas:attrColor('black');
            if tela4.foco == 0 then
                canvas:drawRect("frame", button7.x, button7.y, button7.dx,
button7.dy);
            elseif tela4.foco == 1 then
                canvas:drawRect("frame", button8.x, button8.y, button8.dx,
button8.dy);
            end
            if tela4.foco == 0 and enter == 1 then
                enter = 0
                tela = button7.enter
                refreshScreen ()
            elseif tela4.foco == 1 and enter == 1 then
                enter = 0
                tela = button8.enter
                refreshScreen ()
            end
        end
        canvas:flush();
end

function handler (evt)
```

```
    if evt.class == "key" and evt.type == "press" then
        if evt.key == 'CURSOR_UP' then
                focus = focus + 1
                if focus > telas[tela].lenght-1 then
                        focus = 0
                end
        elseif evt.key == 'CURSOR_DOWN' then
                focus = focus - 1
                if focus < 0 then
                        focus = telas[tela].lenght - 1
                end
        elseif evt.key == 'CURSOR_LEFT' then
                focus = focus - 1
                if focus < 0 then
                        focus = telas[tela].lenght - 1
                end
        elseif evt.key == 'CURSOR_RIGHT' then
                focus = focus + 1
                if focus > telas[tela].lenght-1 then
                        focus = 0
                end
        elseif evt.key == 'ENTER' then
                enter = 1
        end
        refreshScreen ()
    end
end
refreshScreen ();
event.register(handler);
```