Pós-Graduação em Ciência da Computação

"*Contextual Ginga*: Uma Ferramenta de Autoria de Aplicações Interativas Sensíveis a Contexto de TV Digital para Ginga-NCL"

Por

Ana Paula Bezerra Alves de Carvalho

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, JUNHO/2010

Universidade Federal de Pernambuco
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ana Paula Bezerra Alves de Carvalho

"Contextual Ginga: Uma Ferramenta de Autoria de Aplicações Interativas Sensíveis a Contexto de TV Digital para Ginga-NCL"

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: ***Prof. Carlos André Guimarães Ferraz***

RECIFE, JUNHO/2010

Carvalho, Ana Paula Bezerra Alves de

Contextual ginga: uma ferramenta de autoria de aplicações interativas sensíveis a contexto de TV digital para Ginga-NCL / Ana Paula Bezerra Alves de Carvalho. - Recife: O Autor, 2010.

xx, 169 folhas : il., fig., tab.

Dissertação (mestrado) – Universidade Federal de Pernambuco. Cln. Ciência da Computação, 2010.

Inclui bibliografia e apêndice.

1. Sistemas distribuídos. 2. Engenharia de software. I. Título.

004.36 CDD (22. ed.) MEI2010 – 0106

Dissertação de Mestrado apresentada por **Ana Paula Bezerra Alves de Carvalho** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Contextual Ginga: Uma Ferramenta de Autoria de Aplicações Interativas Sensíveis a Contexto de TV Digital para Ginga-NCL**", orientada pelo **Prof. Carlos André Guimarães Ferraz** e aprovada pela Banca Examinadora formada pelos professores:

Profa. Patricia Cabral de Azevedo Restelli Tedesco
Centro de Informática / UFPE

Prof. Fernando Antonio Mota Trinta
Mestrado em Informática Aplicada / UNIFOR

Prof. Carlos André Guimarães Ferraz
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 30 de junho de 2010.

Prof. Nelson Souto Rosa
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

A Gustavo, Rosário, Edson e Laís

Agradecimentos

A Deus, sobretudo, e a Nossa Senhora, que me deram forças quando mais eu precisava para conseguir terminar este trabalho, mesmo trabalhando oito horas por dia e com os preparativos do casamento pelo caminho.

Ao meu amado namorado e marido, a minha mãe, ao meu pai e a minha irmã, que sempre estiveram a meu lado para me apoiar e incentivar na busca de alcançar objetivos mais altos. As palavras de pensamento positivo da minha mãe foram requisitos essenciais.

Aos meus sogrinhos queridos, por estarem na torcida pela conquista de mais uma vitória. Em especial a minha sogrinha, que nem de perto é uma sogra como a maioria.

Ao Professor Carlos Ferraz, por em todas as reuniões sempre estar de bom humor e me orientando com idéias. Consegui acertar no tema e no orientador.

Ao Centro de Informática, pelo excelente nível de qualidade proporcionado a seus alunos, e à Universidade Federal de Pernambuco, pelo acesso ao Portal de Periódicos da CAPES que permitiu as pesquisas necessárias para a realização deste trabalho.

Aos participantes desta pesquisa, por permitirem a realização do experimento de avaliação da ferramenta *Contextual Ginga*.

A todas as outras pessoas, que me ajudaram a atingir mais este objetivo, meus sinceros agradecimentos.

Resumo

Em muitos casos, como no telejornalismo, o conteúdo televisivo precisa estar disponível para transmissão num curto intervalo de tempo. A estas notícias, pode-se desejar adicionar uma aplicação interativa de TV digital (TVDi), que precisará estar pronta rapidamente para ser transmitida em conjunto com o conteúdo. Dessa forma, a aplicação precisará apresentar um ciclo de desenvolvimento rápido, sendo, até mesmo, possível de ser feito por pessoas que não são diretamente da área de informática.

Com o crescente número de aplicações desenvolvidas para a televisão digital e a interatividade proporcionada por elas, os usuários podem encontrar-se em situações em que estejam fornecendo uma informação para uma aplicação e que esta informação já tenha sido fornecida anteriormente para outra aplicação ou que poderia ser capturada automaticamente, sem uma resposta direta do usuário. Uma alternativa para este problema é a utilização automática de informações referentes ao contexto do usuário.

A construção de aplicações de TVDi sensíveis ao contexto do usuário precisa seguir a mesma idéia de um curto ciclo de desenvolvimento devido à rapidez que estas aplicações devem estar disponíveis juntamente com o conteúdo de televisão. Além disso, quanto menor o conhecimento de codificação que a aplicação exigir, no seu desenvolvimento, para estar disponível, maior o número de pessoas que poderão estar envolvidas na produção de conteúdo para a televisão digital.

Desta forma, o principal objetivo deste trabalho é a construção de uma ferramenta de autoria, denominada *Contextual Ginga*, que permite a produção de aplicações interativas sensíveis a contexto para TV digital. Além disto, estas aplicações devem ser executadas no ambiente Ginga-NCL do *middleware* pertencente ao Sistema Brasileiro de TV Digital.

A ferramenta foi avaliada através de uma abordagem hipotética-dedutiva que contou com a realização de um experimento em duas fases. Nestas, os participantes tiveram que desenvolver em 5 passos uma aplicação de TVDi sensível a contexto através da ferramenta *Contextual Ginga*. Os dados coletados, que foram analisados qualitativamente, indicaram que a ferramenta atingiu o seu objetivo.

Palavras-chave: televisão digital, interatividade, sensibilidade a contexto, ferramenta de autoria.

Abstract

In many cases, such as TV journalism, the television content needs to be available for transmission in a short period of time. To these news, one may want to add interactive applications for digital TV that need to be ready quickly to be transmitted together with the content. Thus, the application will need to present a rapid development cycle, which should be done by people who are not from computer science.

Considering that more and more applications have been developed for digital television and the interactivity allowed by them, these applications' users may need to provide many applications with repeated information. Moreover, some information could be automatically gathered without user intervention. An alternative to this problem is the automatic use of information regarding the user's context.

The production of context-sensitive interactive digital TV applications should also have a short development cycle like others applications due to the necessity to deliver them with television shows and advertisements. Furthermore, the lower the codification knowledge needed to have an application available, the greater the number of involved people in production of digital television content.

In this scenario, this work's main objective is to develop an authoring tool, called Contextual Ginga, that allows producing context-sensitive interactive digital TV applications. Moreover, these applications must be executed in the Ginga-NCL environment, part of Brazilian System Digital Television middleware.

The tool was evaluated using a hypothetical-deductive approach which included one experiment in two phases. Accordingly, participants had to develop in five steps a context-sensitive interactive digital TV application through Contextual Ginga. The data collected, which were analyzed qualitatively, indicated that the tool reached its objective.

Keywords: digital television, interactivity, context sensitivity, authoring tool.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Códigos

Lista de Abreviaturas e Siglas

API	*Application Programming Interface*
DOM	*Document Object Model*
EPG	*Eletronic Program Guide*
GEM	*Globally Executable MHP*
GUI	*Graphical User Interface*
ISDTV-T	*International Standard for Digital Television - Terrestrial*
MDE	*Model-Driven Engineering*
MHP	*Multimedia Home Plataform*
NCL	*Nested Context Language*
NCM	*Nested Context Model*
OCAP	*OpenCable Application Platform*
PDA	*Personal Digital Assistant*
SAX	*Simple API for XML*
SBTVD	Sistema Brasileiro de TV Digital
SMS	*Short Message System*
STB	*Set-top-box*
TVDi	Televisão Digital Interativa
W3C	*World Wide Web Consortium*
XML	*EXtensible Markup Language*

1 Introdução

Neste capítulo, este trabalho será apresentado através de sua motivação, como também através da descrição do problema e hipótese considerados, além dos objetivos a serem alcançados. Por fim, será explicado como este documento encontra-se estruturado.

1.1 Motivação

A evolução da televisão analógica para a digital confere vários benefícios para as pessoas. Os principais benefícios são: melhor qualidade de imagem e de áudio, multiprogramação e interatividade. Em particular, elas deixam de ser telespectadores passivos para se tornarem usuários ativos que interagem com aplicações.

A capacidade de multiprogramação e de processamento da TV digital proporciona para os seus usuários um aumento no número de programas, informações e serviços disponíveis. Com esse aumento, os usuários podem não encontrar um programa que seja mais adequado a sua vontade em um dado momento do tempo, um serviço útil do qual esteja precisando ou ainda receber em excesso propagandas que não sejam do seu interesse. Para tentar resolver esse problema, algumas soluções foram propostas a partir do uso de sistemas de recomendação que se baseiam no contexto do usuário de televisão, construído sobre perfis e históricos do usuário; desta forma, permitindo a personalização de conteúdo.

Uma dessas soluções é para tornar sensíveis a contexto os serviços multimídia baseados em IMS (*IP-based Multimedia Subsystem*) (THAWANI; GOPALAN; SRIDHAR, 2004). Como outro exemplo de solução, tem-se a proposta de: um gerador de perfis que automaticamente cria perfis de usuário para representar as suas preferências; e um algoritmo de recomendação que estima o interesse do usuário em conteúdo não conhecido por ele, através de uma comparação do perfil com meta-dados do conteúdo (WEISS et al., 2008). Existe uma revisão do estado da arte para aplicações de TV digital personalizadas

e móveis, trazendo os algoritmos de personalização das referências então pesquisadas (CHORIANOPOULOS, 2008).

Por sua vez, a capacidade de interatividade na TV digital permite que seus usuários interajam com aplicações executadas localmente em seu domicílio e que se comuniquem diretamente com os provedores de conteúdo ou mesmo entre si. Este formato de interatividade integrada no aparelho de TV representa um avanço em relação a outros mais comuns na televisão atualmente, como telefone, Internet e mensagens SMS − *Short Messaging Service* (JENSEN, 2005). Vários tipos de serviços podem fazer uso dessa capacidade de interatividade na TV digital, como comércio (KIM; AHN; KO, 2007), governo (SCHIBELSKY et al., 2008) e aprendizado televisivo (REY-LÓPEZ et al., 2008).

Com o aumento da interação entre usuário e aplicação de TV digital, os usuários podem se encontrar em situações que estejam fornecendo uma informação para uma aplicação e que esta informação já tenha sido fornecida anteriormente para outra aplicação ou que poderia ser capturada automaticamente, sem uma resposta direta do usuário. Este excesso de informações dadas e/ou recebidas pelos usuários pode diminuir o nível de interesse destes pela aplicação.

Através do design interativo, podem ser tomados cuidados na criação de aplicações de TV digital para não diminuir este interesse. Como cuidados, destacam-se: a definição do perfil do usuário público-alvo; a projeção dos contextos em que a aplicação será utilizada; e a observação de requisitos de usabilidade para TV digital (GOMES, 2008). Estes cuidados baseiam-se nas Leis da Simplicidade (MAEDA, 2006).

Um exemplo de aplicação de TV digital interativa é a *Pan Americano*, desenvolvida para os jogos de 2007, que aconteceram no Rio de Janeiro. Esta aplicação que foi transmitida na Rede Globo de São Paulo e ela exibia, por exemplo, notícias associadas ao evento esportivo e o quadro de medalhas atualizado (COÊLHO, 2008).

Esta aplicação foi modificada para permitir a captura de forma natural de características contextuais relevantes na interação do usuário com a aplicação interativa. Esta captura é realizada através de uma infra-estrutura de personalização da interação que utiliza informações do perfil do usuário. Através do uso dessa infra-estrutura, foi possível, com poucas modificações na aplicação, que esta fosse capaz de identificar a tela que o usuário passava mais tempo e executá-la inicialmente na próxima vez que o usuário acessasse a aplicação. Dessa forma, a aplicação passava a utilizar o histórico do usuário (COÊLHO, 2008).

Através da inserção da característica de sensibilidade a contexto do usuário, que está presente na maioria das referências citadas até o momento, os sistemas se tornam mais atrativos, adaptáveis e pró-ativos. Para o desenvolvimento destes sistemas sensíveis a contexto, foi definido um processo para o detalhamento das principais atividades de especificação e projeto. Este processo é denominado CEManTIKA – *Contextual Elements Modeling and Management through Incremental Knowledge Acquisition* (VIEIRA; TEDESCO; SALGADO, 2009).

A forma de interação do usuário com a televisão também deve ser destacada. Como o controle remoto ainda é o principal dispositivo de entrada, pode-se tornar cansativo o uso freqüente dele, principalmente na digitação de textos. Esse fato pode inclusive desmotivar o usuário. Pois, mesmo que exista a opção de rejeitar ou não utilizar uma aplicação, ainda assim pode ser preciso que se pressione algum botão (SILVA; TAVARES; FILHO, 2008).

Em muitos casos, o conteúdo televisivo precisa estar disponível para transmissão em um curto intervalo de tempo (GOMES, 2008). Exemplos destes casos são acontecimentos que ocorrem durante o dia e que devem ser reportados como notícias em telejornais à noite. A estas notícias, pode-se desejar adicionar uma aplicação interativa de TV digital (TVDi), que precisará estar pronta rapidamente para ser transmitida em conjunto com o conteúdo. Dessa forma, a aplicação deverá apresentar um ciclo de desenvolvimento rápido.

Como é improvável que os produtores de TV possuam equipes de informática suficientes, considera-se que o desenvolvimento das aplicações TVDi deveria poder ser feito por pessoas que não são diretamente da área de informática. Deve ser ressaltado que o desenvolvimento destas aplicações é diferente do desenvolvimento de aplicações para *desktop* e para celular.

Os recursos de hardware disponíveis na televisão ainda são inferiores aos dos computadores pessoais, bem como a interação com a televisão é mais simples e menos intensa do que nos computadores. Comparando a televisão com o celular, este segundo dispositivo também possui um hardware limitado em relação ao primeiro. Mas, pode ser realizada também uma comparação entre as dimensões de tela dos aparelhos de televisão convencional e dos celulares, ou seja, não considerando a transmissão de TV digital recebida pelos celulares. Enquanto as televisões possuem uma tela que permite que as pessoas assistam e interajam a um distância medida em metros, os celulares possuem um tamanho de tela reduzido, exibida na própria mão da pessoa.

Assim como existem ferramentas de autoria para o desenvolvimento de aplicações

desktop e *web*, por exemplo, também já existem ferramentas para auxiliar o desenvolvimento de aplicações para TV digital, como: Ginga-NCL Composer (GUIMARÃES, 2007), Icareus iTV Suite Author (ICAREUS, 2009), InteracTV (ALMEIDA; SOUZA; NETO, 2007), iTV Project (OLIVEIRA; FILHO; SILVA, 2008), JAME Author (IMK, 2005), NCL Eclipse (LAWS, 2008) e SCO Creator Tool (REY-LÓPEZ et al., 2008). Cada uma destas ferramentas será detalhada nesta dissertação.

Estas ferramentas de autoria permitem, na sua maioria, que o trabalho de codificação seja substituído pela construção de modelos e abstrações. Estes passam posteriormente por um processo de transformação em código. Esta mudança de foco de código para modelo contribui para o aumento de produtividade, facilitando e diminuindo o tempo de desenvolvimento das aplicações. Este conceito faz parte do *Model-Driven Engineering - MDE* (BORDIN; PANUNZIO; VARDANEGA, 2008).

É importante antecipar que nenhuma das ferramentas de autoria consideradas como trabalhos relacionados é capaz de produzir aplicações interativas sensíveis a contexto do usuário para o *middleware* Ginga. Este *middleware* é pertencente ao Sistema Brasileiro de TV Digital – SBTVD (ABNT, 2007b). O Ginga é composto, principalmente, por duas máquinas, uma para execução de aplicações procedurais Java (Ginga-J) e outra para apresentação de documentos NCL (Ginga-NCL). A máquina Ginga-J ainda se encontra em desenvolvimento, mas a máquina Ginga-NCL já possui versões disponíveis (PORTAL, 2009).

Considerando a implantação da TV digital no cenário do nosso país, o *middleware* Ginga deve em breve estar presente em larga escala nos *set-top-boxes* (conversor digital) das residências brasileiras. E isso deve favorecer o uso de aplicações interativas de TV digital no país, aumentando o volume de informações envolvidas na interação do brasileiro com a televisão e motivando também a necessidade do uso de informações contextuais.

1.2 Problema e Hipótese

O problema desta pesquisa é baseado no projeto "Desenvolvimento de Metodologia, Técnicas e Ferramentas de Engenharia de Software para Aplicações Interativas de TV Digital", financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

Como método científico, esta pesquisa utiliza o hipotético-dedutivo, no qual hipóteses são formuladas e em seguida submetidas a testes de falseamento. Dessa forma, a partir

do projeto mencionado e da motivação levantada na seção anterior, é proposto o seguinte problema de pesquisa:

- **Problema:** Do ponto de vista computacional, o desenvolvimento de aplicações interativas de TV digital, particularmente adaptáveis (sensíveis) a contexto do usuário, é difícil e demorado, em especial se feito por profissional não especialista em informática.

Para este problema, formulou-se a seguinte hipótese:

- **Hipótese:** Se for construída uma ferramenta de autoria, que permita a produção de aplicações interativas de TV digital sensíveis a contexto do usuário para a plataforma Ginga-NCL, então uma pessoa, que entende dos conceitos de TV e de sensibilidade a contexto, poderá construir tal aplicação sem a necessidade de codificação, ou seja, de maneira mais fácil e rapidamente.

Com base nesta hipótese, pode-se considerar como variável dependente o tempo de construção de aplicações interativas de TV digital sensíveis a contexto. E como variáveis independentes, tem-se: o auxílio de uma ferramenta de autoria e o conhecimento dos conceitos de TV e sensibilidade a contexto por parte do desenvolvedor das aplicações.

Com o objetivo de evitar ambiguidade, nesta dissertação, o termo usuário será sempre referente à pessoa que assiste TV e interage com as aplicações. Enquanto o usuário da ferramenta proposta é referenciado como desenvolvedor, por ser o responsável pela construção das aplicações para TV digital.

1.3 Objetivos

Um dos objetivos do projeto citado na seção anterior é a proposta de novas ferramentas e bibliotecas que auxiliem diretamente o desenvolvimento de aplicações de TV digital. Dessa forma, partindo do projeto e da hipótese especificada, o principal objetivo desta dissertação consiste em:

Construir uma ferramenta de autoria, denominada Contextual Ginga, que permita a produção de aplicações interativas de TV digital sensíveis a contexto e que sejam executadas na plataforma Ginga-NCL.

Sendo destacado que, a escolha dessa máquina foi determinada pelo maior estágio de maturidade e disponibilidade desta em relação à Ginga-J.

Desmembrando o objetivo principal, são determinados os seguintes objetivos específicos deste trabalho:

1. Gerar automaticamente código para a máquina Ginga-NCL, através da ferramenta de autoria *Contextual Ginga* a ser desenvolvida;
2. Permitir que códigos sejam reusados entre aplicações geradas, desenvolvendo uma biblioteca denominada *Contextual Ginga Lua*;
3. Comparar as principais ferramentas de autoria disponíveis para TV digital e compará-las com a proposta por este trabalho;
4. Incorporar na ferramenta *Contextual Ginga* o conceito de sensibilidade a contexto, de forma a enriquecer a interação do usuário com as aplicações interativas de TV digital produzidas através da ferramenta;
5. Permitir que pessoas, que não saibam codificar nas principais linguagens de programação para Ginga-NCL, produzam aplicações interativas de TV digital sensíveis a contexto.

1.4 Método de Pesquisa

Sendo utilizada uma abordagem hipotética-dedutiva, esta pesquisa se origina no problema detectado e, considerando a hipótese levantada, propõe como solução a ferramenta de autoria *Contextual Ginga*. A capacidade da ferramenta proposta, de satisfazer a hipótese desta pesquisa, foi verificada através da eliminação de erros. Esta tentativa de eliminação de erros foi realizada através de um experimento.

Os dados obtidos através do experimento foram analisados a partir de uma abordagem qualitativa. Esta destaca-se principalmente em cenários nos quais o objeto de estudo é tão complexo que não é suficiente apenas medir e quantificar fenômenos. É preciso também compreender as diferentes perspectivas de indivíduos e grupos de indivíduos sobre o objeto estudado, e como estes constroem significado e conhecimento (FLICK, 2008). Tal tipo de pesquisa foi utilizada neste trabalho justamente devido à complexidade do objeto de estudo que envolve a produção, através de uma ferramenta de autoria, de aplicações sensíveis a contexto para TV digital.

Esta pesquisa encontra-se organizada em seis etapas (Figura 1.1), cada uma com o seu objetivo:

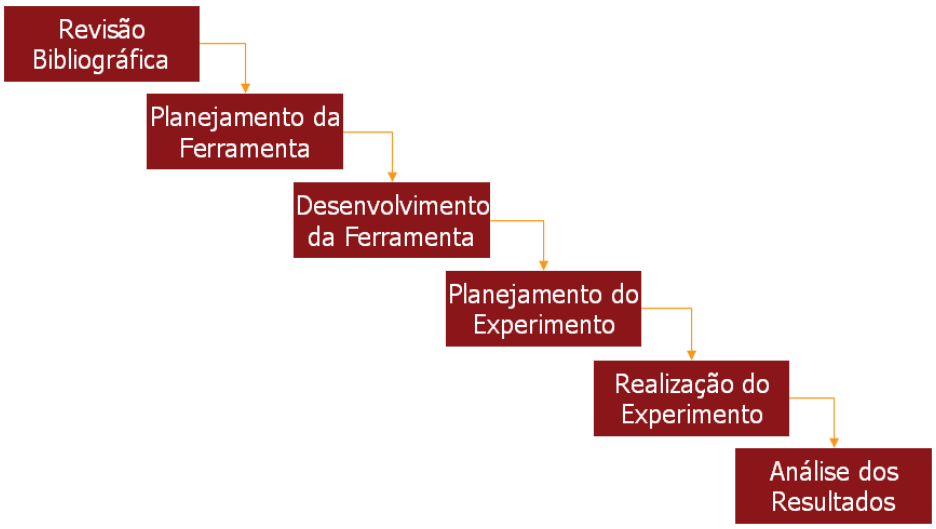


Figura 1.1: Etapas da Pesquisa.

1. **Revisão Bibliográfica**: Realizar uma revisão bibliográfica para levantar e, em seguida, analisar os principais trabalhos relacionados;

2. **Planejamento da Ferramenta**: Definir macro-requisitos com base nos trabalhos relacionados e no que este trabalho se diferencia;

3. **Desenvolvimento da Ferramenta**: Documentar e codificar a ferramenta;

4. **Planejamento do Experimento**: Definir os participantes e por quais razões foram selecionados; e definir como o experimento deve ser executado e quais instrumentos devem ser utilizados;

5. **Realização do Experimento**: Realizar o experimento de acordo com o planejado;

6. **Análise dos Resultados**: Analisar os dados coletados no experimento com o intuito de corroborar ou refutar a hipótese desta pesquisa. Sendo considerado que a hipótese foi corroborada, caso os participantes tenham conseguido executar uma aplicação interativa sensível a contexto produzida através da ferramenta proposta.

1.5 Estrutura da Dissertação

Este trabalho é composto por mais cinco capítulos:

- **Capítulo 2**: Apresenta o referencial teórico analisado e as principais ferramentas de autoria para TV digital;
- **Capítulo 3**: Descreve a ferramenta de autoria *Contextual Ginga* e apresenta a biblioteca *Contextual Ginga Lua*, ambas propostas por este trabalho. Além disto, realiza uma comparação entre a ferramenta proposta e as apresentadas no Capítulo 2;
- **Capítulo 4**: Apresenta os resultados obtidos no experimento e os analisa;
- **Capítulo 5**: Sintetiza as principais contribuições desta dissertação, expondo as limitações da ferramenta *Contextual Ginga* e propondo trabalhos futuros pertinentes.

2 Referencial Teórico

Neste capítulo, será descrito o referencial teórico que serve de fundamentação para este trabalho: interatividade na TV digital, o *middleware* Ginga, sensibilidade a contexto e *personas*. Além disto, serão descritos os principais trabalhos relacionados que foram encontrados no decorrer desta pesquisa.

2.1 Interatividade na TV Digital

Interatividade é um conceito relacionado diretamente à comunicação, seja entre pessoas, entre estas e dispositivos computacionais ou entre estes dispositivos. No segundo caso, as pessoas, denominadas usuários, fornecem entradas e os dispositivos fornecem respostas ou mudam seu comportamento para se adaptar às entradas. Esta adaptação também é uma forma de resposta do dispositivo para o usuário. Como dispositivos computacionais que permitem interação, podemos citar o computador, o celular, o PDA (*Personal Digital Assistant*) e agora até mesmo as televisões digitais.

A interatividade na TV digital pode ser categorizada em 3 níveis (JENSEN, 2005). No primeiro nível, chamado interatividade local, não existe um canal de retorno através do qual o usuário possa enviar informações através da própria televisão. A interatividade é baseada no fato do usuário poder escolher, através de uma aplicação, quais textos e imagens ele prefere visualizar dentre aqueles que já são enviados no fluxo *broadcast*.

Exemplos de aplicações neste nível de interatividade são aquelas que envolvem notícias, informações sobre o clima e esportes. A Figura 2.1 esquematiza este nível de interatividade, no qual o conteúdo produzido é transmitido via antena para a televisão do usuário e não há canal de retorno.

No segundo nível, chamado interatividade parcial ou unidirecional (*one-way*), o canal de retorno possui poucos recursos e permite, por exemplo, que o usuário envie dados simples, como a opção preferida dele em uma dada enquete, e que o *set-top-box* realize

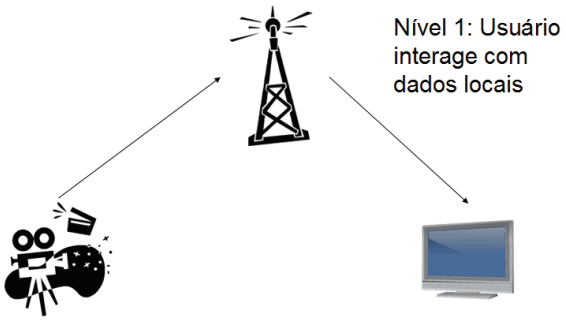


Figura 2.1: Nível 1: Interatividade local.

chamadas telefônicas com o objetivo de realizar pagamentos (JENSEN, 2005).

No terceiro nível, chamado interatividade plena, o usuário dispõe de um canal de retorno que permite enviar vários tipos de informações tanto para o *broadcaster* quanto para outros usuários (JENSEN, 2005). Podem ser citadas como exemplos, aplicações multimídia, vídeo sob demanda e aplicações de comunicação instantânea.

Um esquema deste nível de interatividade está representado na Figura 2.2, no qual o conteúdo produzido é transmitido via antena para a televisão do usuário e existe um canal de retorno (interatividade) entre o produtor de conteúdo e a televisão do usuário, que permite a comunicação bidirecional.

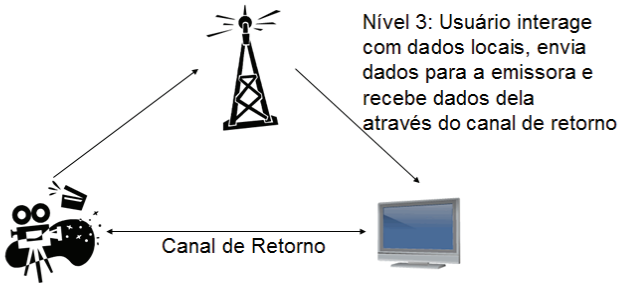


Figura 2.2: Nível 3: Interatividade plena.

Quanto aos dispositivos de entrada de dados que os usuários podem utilizar para interagir com a televisão digital, o principal deles ainda é o controle remoto, mas já se vem estudando outros tipos, como os dispositivos móveis, que podem se comunicar com o *set-top-box* através de *Bluetooth* (SILVA; TAVARES; FILHO, 2008).

2.2 Ginga

Ginga é o middleware especificado pelo Fórum SBTVD (Sistema Brasileiro de TV Digital) para o *International Standard for Digital Television - Terrestrial* ISDTV-T, que sobre ele são executadas aplicações interativas de TV digital independentemente do hardware do *set-top-box* que elas estejam.

O Ginga possui duas máquinas como elementos principais, uma para execução de aplicações procedurais Java (ambiente procedural Ginga-J) e outra para a apresentação de documentos NCL (ambiente declarativo Ginga-NCL). Nesta máquina, as principais linguagens de codificação são NCL (*Nested Context Language*), padronizada pelo *International Telecommunication Union* ITU (2009), e Lua (IERUSALIMSCHY; FIGUEIREDO; CELES, 2006). Esses dois ambientes podem se comunicar através de uma ponte, como pode ser visto na Figura 2.3 (ABNT, 2007b).

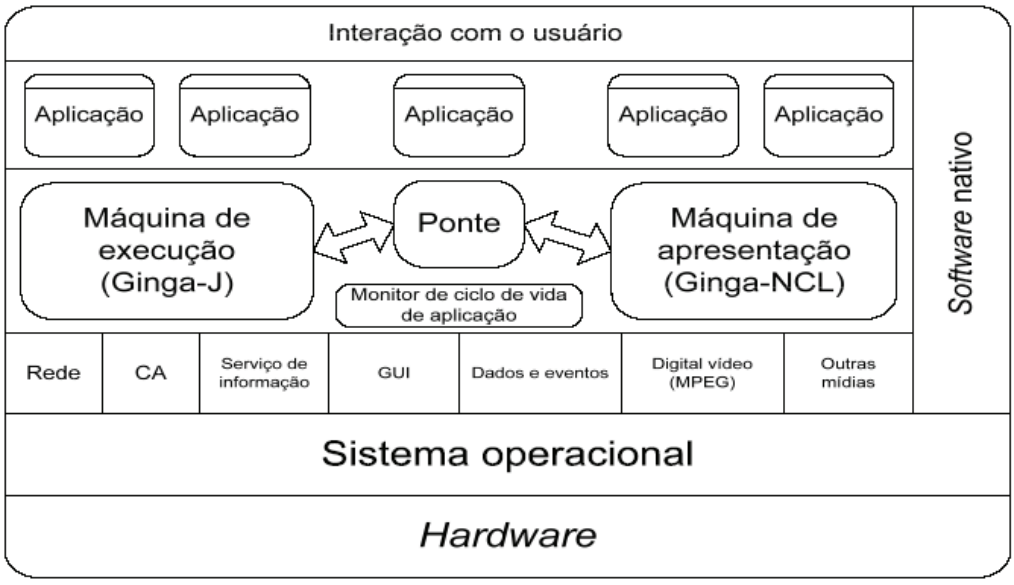


Figura 2.3: Arquitetura do Middleware Ginga.
[Fonte: ABNT (2007b)]

O Ginga-NCL é responsável por processar documentos NCL (*Nested Context Language*), possuindo um módulo, denominado formatador NCL, para interpretar este conteúdo e outros módulos secundários como o exibidor XHTML e a máquina de apresentação de *scripts* Lua. Já o Ginga-J possui como principal módulo a máquina virtual Java para código procedural escrito nesta linguagem (ABNT, 2007a).

Nesta dissertação, será dado maior enfoque à máquina Ginga-NCL, uma vez que ela apresenta maior estágio de maturidade e disponibilidade em relação à máquina Ginga-J.

As duas próximas seções descrevem brevemente as linguagens NCL e Lua.

2.2.1 *Nested Context Language - NCL*

A *Nested Context Language* (NCL) é uma linguagem XML (*EXtensible Markup Language*), baseada no modelo *Nested Context Model* (NCM), para a autoria de documentos hipermídia. A linguagem NCL possibilita separação entre conteúdo e estrutura, sincronização espaço-temporal e adaptabilidade através de um comando de controle de fluxo. As aplicações declarativas NCL podem utilizar *scripts* escritos na linguagem Lua, descrita a seguir (ABNT, 2007a).

A linguagem NCL apresenta um conjunto de conceitos: nó, porta, região, descritor, elo, conector, papel, associação e âncora. Um nó pode representar um conteúdo ou mídia ou pode representar uma composição ou contexto. Na Figura 2.4, os nós são representados por círculos. Os nós de conteúdo ou mídia são definidos dentro de nós de composição ou contexto, ou seja, estes fazem a composição das mídias ou outros contextos.

O contexto raiz, ou seja, aquele que agrupa todas as mídias e contextos do documento NCL, é denominado *body*. O conteúdo do contexto (por exemplo, *video2*, *imagem1* e *audio1* na Figura 2.4) é acessado através de uma porta (*portaContexto*), que atua como um ponto de interface (entre os contextos *body* e *contexto1*) (GUIMARÃES, 2007). Estes contextos e o ponto de acesso a um contexto interno podem ser visualizados também na Figura 2.4. Estes relacionamentos podem ser vistos, escritos em linguagem NCL, na Figura 2.5.

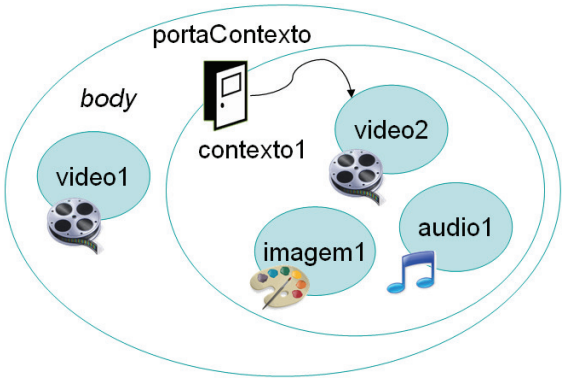


Figura 2.4: Contexto para a linguagem NCL.
[Fonte: Neto et al. (2007)]

Deve ficar claro que este conceito de contexto para a linguagem NCL não é o mesmo de contexto usado nesta dissertação. Aqui, contexto representa uma situação, que pode

ser do usuário (por exemplo, idade), do dispositivo (por exemplo, tamanho da tela) ou do ambiente (por exemplo, temperatura). Lembrando que o foco deste trabalho está direcionado para o contexto do usuário da aplicação de TVDi.

Voltando aos conceitos da linguagem NCL, uma região representa uma área que exibe um nó de mídia. Para que o nó de mídia seja exibido numa determinada posição do dispositivo de saída, é preciso relacionar um nó a um descritor, que é a forma que se define como o nó será apresentado (GUIMARÃES, 2007).

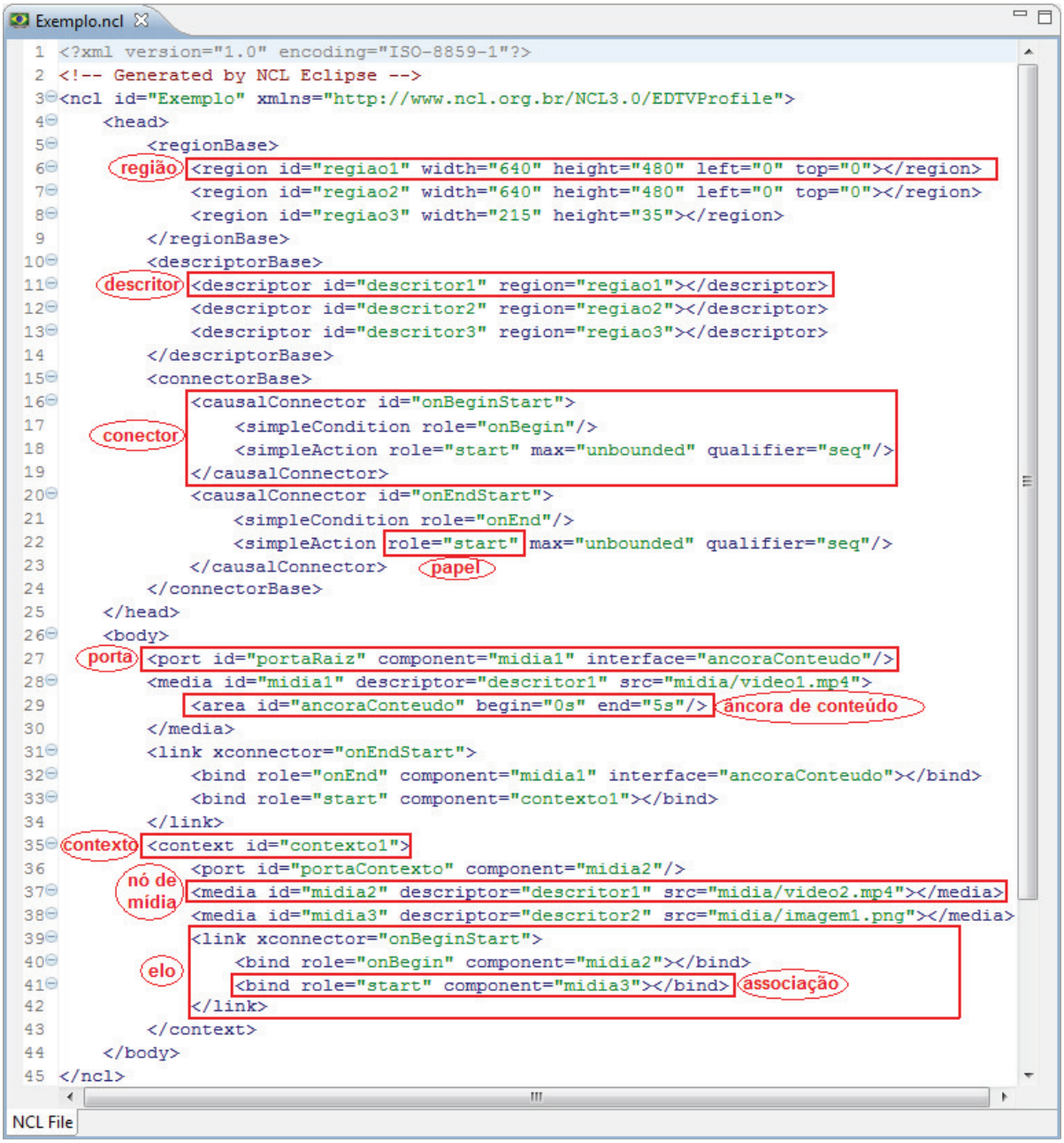
O elo define relações de sincronização entre os nós, onde o tipo da relação está definido num conector. O conector estabelece uma relação de causalidade para que em função de uma condição seja executada uma ação. Papéis indicam qual a condição e qual a ação do conector. São definidas associações para criar o relacionamento entre um nó e um papel. Uma âncora pode ser utilizada para segmentar uma mídia de forma que apenas o segmento escolhido seja apresentado (âncora de conteúdo com elemento *area*) ou para definir uma propriedade para um nó de mídia (âncora de propriedade com elemento *property*) (PORTAL, 2009).

Todos estes conceitos da linguagem NCL podem ser visualizados no exemplo de código, editado no *plug-in* NCL Eclipse (LAWS, 2008), presente na Figura 2.5: região (linha 6), descritor (linha 11), conector (da linha 16 à linha 19), papel (linha 22), porta (linha 27), âncora de conteúdo (linha 29), nó de composição ou contexto (linha 35), nó de conteúdo ou mídia (linha 37), elo (da linha 39 à linha 42) e associação (linha 41).

2.2.2 Lua

Desenvolvida no laboratório Tecgraf da PUC-Rio e criada em 1993, Lua é uma linguagem de programação de extensão que dá suporte à programação procedural. Ela oferece apoio para outras linguagens nos paradigmas funcional, orientado a objetos e orientada a dados. É uma linguagem de *script* leve e poderosa, escrita como biblioteca na linguagem C (IERUSALIMSCHY; FIGUEIREDO; CELES, 2006). Os *scripts* Lua são principalmente utilizados em aplicações de jogos e interfaces gráficas pelo seu alto desempenho computacional.

Sendo uma linguagem de extensão, ela precisa que um programa escrito em outra linguagem faça a invocação a suas funções. Dessa forma, o formatador NCL é o responsável por controlar a comunicação entre o objeto NCLua e outros objetos NCL (ABNT, 2007a).




```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!-- Generated by NCL Eclipse -->
3  <ncl id="Exemplo" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
4      <head>
5          <regionBase>
6              <region id="regiao1" width="640" height="480" left="0" top="0"></region>
7              <region id="regiao2" width="640" height="480" left="0" top="0"></region>
8              <region id="regiao3" width="215" height="35"></region>
9          </regionBase>
10         <descriptorBase>
11             <descriptor id="descritor1" region="regiao1"></descriptor>
12             <descriptor id="descritor2" region="regiao2"></descriptor>
13             <descriptor id="descritor3" region="regiao3"></descriptor>
14         </descriptorBase>
15         <connectorBase>
16             <causalConnector id="onBeginStart">
17                 <simpleCondition role="onBegin"/>
18                 <simpleAction role="start" max="unbounded" qualifier="seq"/>
19             </causalConnector>
20             <causalConnector id="onEndStart">
21                 <simpleCondition role="onEnd"/>
22                 <simpleAction role="start" max="unbounded" qualifier="seq"/>
23             </causalConnector>
24         </connectorBase>
25     </head>
26     <body>
27         <port id="portaRaiz" component="midia1" interface="ancoraConteudo"/>
28         <media id="midia1" descriptor="descritor1" src="midia/video1.mp4">
29             <area id="ancoraConteudo" begin="0s" end="5s"/>
30         </media>
31         <link xconnector="onEndStart">
32             <bind role="onEnd" component="midia1" interface="ancoraConteudo"></bind>
33             <bind role="start" component="contexto1"></bind>
34         </link>
35         <context id="contexto1">
36             <port id="portaContexto" component="midia2"/>
37             <media id="midia2" descriptor="descritor1" src="midia/video2.mp4"></media>
38             <media id="midia3" descriptor="descritor2" src="midia/imagem1.png"></media>
39             <link xconnector="onBeginStart">
40                 <bind role="onBegin" component="midia2"></bind>
41                 <bind role="start" component="midia3"></bind>
42             </link>
43         </context>
44     </body>
45 </ncl>
```

Figura 2.5: Exemplo de código NCL.

No Código 2.1, é mostrado um exemplo de código Lua, através da função *fatorial* que calcula o fatorial de um número *n*, passado como argumento para a função (linha 2). Nas linhas 3, 5 e 9, a sintaxe da estrutura de controle *if*. Na linha 6, a atribuição à variável local *res*, do valor da multiplicação de *n* pelo fatorial de *n-1*, representando uma função recursiva. O valor da variável *res* é impresso pelo comando da linha 7, que chama a função *print* de Lua.

Código 2.1: Exemplo código Lua.

```
1  -- Função que calcula o fatorial de um número n qualquer.
2  function fatorial(n)
3          if n == 0 then
4                  return 1;
5          else
6                  res = n * fatorial(n - 1);
7                  print('O fatorial de ' .. n .. ' é ' .. res .. '.');
8                  return res;
9          end
10 end
```

2.3 Sensibilidade a Contexto

O uso do contexto na comunicação entre humanos é bastante comum. Quando as pessoas falam e escutam elas já possuem um conjunto de informações implícitas de forma que nem tudo precisa ser repetido para que haja entendimento. Uma das definições de contexto mais referenciadas na área de computação é a de que:

> "Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Essa entidade pode ser uma pessoa, lugar ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação" (DEY; ABOWD, 2000).

O conceito de contexto encontra-se também relacionado com o de computação ubíqua, que possui como objetivo fazer com que a tecnologia seja invisível para os usuários, tornando a interação homem-máquina mais simples (NIEUWDORP, 2007).

A sensibilidade a contexto em aplicações permite que as mesmas se adaptem aos seus usuários, sejam estes pessoas ou outras aplicações. Elas coletam dados julgados relevantes para uma determinada tarefa e usam estes dados, em geral, para: fornecer informações que os usuários achariam interessante receber; comportarem-se de maneira mais específica para os usuários; ou para tomarem uma decisão que provavelmente os usuários tomariam, prevendo o comportamento destes e diminuindo seu esforço.

Tratando do cenário de TV digital e suas aplicações interativas, a sensibilidade a contexto pode tornar a interação mais agradável. Informações contextuais, como tempo de visualização de um determinado conteúdo e quais conteúdos são visualizados, já estão sendo utilizadas na TV digital, principalmente para personalização de conteúdo oferecendo recomendação de programas (CHORIANOPOULOS, 2008; WEISS et al., 2008) e exibição de propagandas específicas para o usuário (THAWANI; GOPALAN; SRIDHAR, 2004).

Os principais dados que podem ser utilizados como contexto em TV digital, segundo as categorias *Who-When-Where-What-Why-How* (MORSE; ARMSTRONG; DEY, 2000), são: quem são os usuários que estão interagindo com a televisão, ou seja, os perfis destas pessoas e suas preferências (*Who*); informações temporais, como dia da semana, hora do dia e estação do ano (*When*); localização do usuário com maior ou menor granularidade, por exemplo bairro e sala, respectivamente (*Where*); o que o usuário está fazendo ao interagir com a televisão (*What*); por que o usuário está fazendo isto (*Why*); e como as informações contextuais estão sendo coletadas (*How*).

2.4 Persona

Persona é um conceito utilizado principalmente no desenvolvimento de produtos com design centrado no usuário. Ele representa, de forma imaginária, o usuário através de suas principais características, devendo ser construído a partir de dados especificados sobre pessoas reais (PRUITT; ADLIN, 2006).

Do ponto de vista do produto final, são melhorados a sua usabilidade, a sua utilidade e o seu apelo geral com a participação figurativa de *personas* no desenvolvimento. Do lado da organização que usa este conceito, o processo do time de desenvolvimento se torna mais eficiente pela exclusão de partes não necessárias; os indivíduos do time trabalham de forma mais integrada; e as decisões de negócio são feitas de forma mais fundamentada (PRUITT; ADLIN, 2006).

Relacionando o conceito de *persona* com o apresentado na subseção anterior, pode-se dizer que a informação contextual *Who* compreende a caracterização do usuário e as demais definem o cenário do momento que o usuário interage com uma aplicação de TV digital (GOMES, 2008). Dessa forma, esta aplicação se tornará mais simples por causa da sua adaptabilidade de acordo com o contexto do usuário.

2.5 Ferramentas de Autoria para TV Digital

Esta seção lista por ordem alfabética os principais trabalhos relacionados referentes a ferramentas de autoria para TV digital.

2.5.1 Ginga-NCL Composer

Ginga-NCL Composer (GUIMARÃES, 2007) é uma ferramenta de autoria gráfica, com ambiente de execução *desktop*, desenvolvida pelo Laboratório TeleMídia da PUC-Rio para criação de aplicações NCL interativas de TV digital. Seu objetivo é diminuir o trabalho do autor (desenvolvedor) da aplicação através da diminuição da complexidade da programação em NCL.

Esta ferramenta, gratuita, possui quatro tipos de visões, sincronizadas entre si, para edição: estrutural (1), *layout* (2), temporal (3) e textual (4). A Figura 2.6 mostra estas visões com o mesmo exemplo de código presente na Figura 2.5.

Na visão estrutural (1), são exibidos os nós de mídia (*midia1*, *midia2* e *midia3*) que formam um documento NCL e o agrupamento entre estas mídias, formando nós de composição (*contexto1*). Na visão *layout* (2), são exibidas visualmente as regiões utilizadas para exposição dos nós. Na visão temporal (3), é exibida uma linha do tempo com indicações dos intervalos, nos quais cada mídia será apresentada, bem como o sincronismo entre as mídias do documento. Na visão textual (4), é possível editar o código gerado através da utilização das outras visões.

Para a criação de uma aplicação através desta ferramenta, mesmo sem ser através da visão textual, é necessário possuir conhecimento dos conceitos presentes na linguagem NCL (*Nested Context Language*), descritos na Subseção 2.2.1.

O Ginga-NCL Composer utiliza o conceito de contexto, presente na linguagem NCL, para estruturar as partes da aplicação, ou seja, os conteúdos e recursivamente outros contextos. Mas, apesar disto, não existe raciocínio sobre este contexto ou o contexto do usuário da aplicação. Esta ferramenta encontra-se atualmente descontinuada.

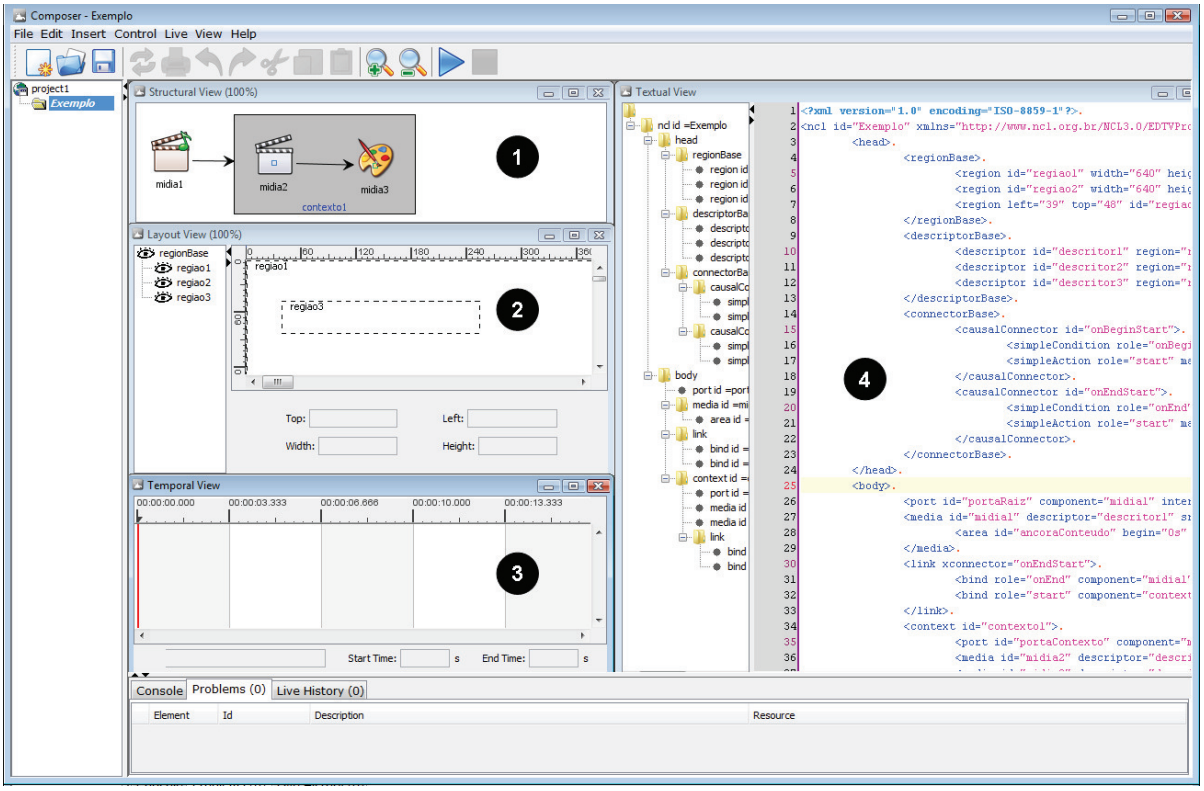


Figura 2.6: Tela da ferramenta Ginga-NCL Composer.
[Fonte: Portal (2009)]

2.5.2 Icareus iTV Suite Author

O Icareus iTV Suite Author (ICAREUS, 2009) é ferramenta gráfica de autoria, proprietária, com ambiente de execução *desktop*, e voltada para os padrões MHP[1] (DVB Project, 2002), OCAP/tru2way (CABLELABS, 2010), IDway-J (IRDETO, 2008) e Blu-ray (BDA, 2006). Atualmente a Icareus está desenvolvendo uma versão para o padrão brasileiro Ginga-J.

Na Figura 2.7, podem ser visualizados: os tipos de componentes que podem ser adicionados às cenas (1), que são as telas da aplicação; a área de edição da cena (2); a lista de cenas (3); uma lista de recursos (imagens) presentes para adição na cena (4); e a possibilidade de alterar as propriedades dos componentes (5). O Icareus também possui um emulador que pode ser utilizado para pré-visualizar como a aplicação interativa irá se comportar.

Focando na área (5) da Figura 2.7, mais especificamente nas propriedades referentes

[1]Multimedia Home Platform (MHP) é um *middleware* aberto para televisão digital interativa, no qual são executadas aplicações baseadas em Java. Ele é especificado pelo projeto DVB (*Digital Video Broadcasting*) (DVB Project, 2002).

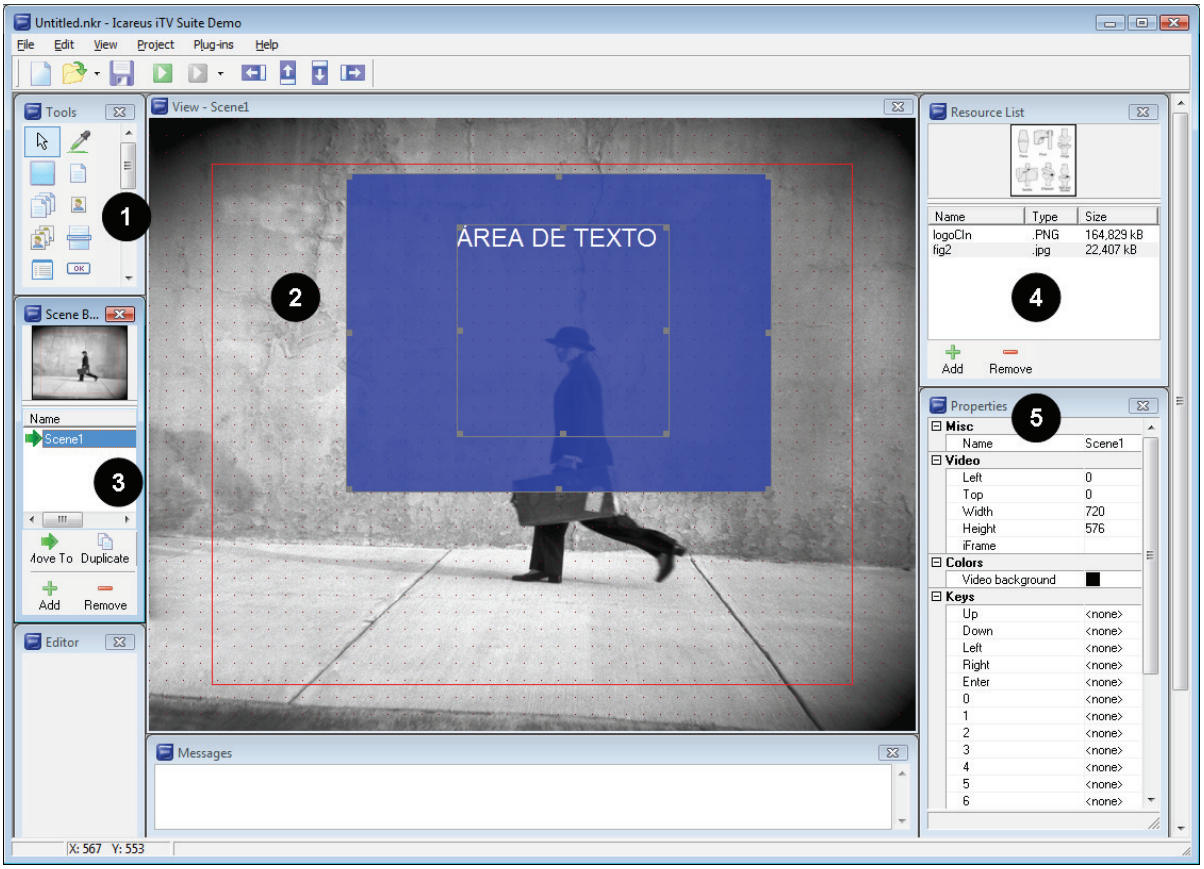


Figura 2.7: Tela da ferramenta Icareus iTV Suite Author.
[Fonte: Icareus (2009)]

a teclas, pode-se visualizar na Figura 2.8, na área (1), que é possível configurar qual ação será disparada ao usuário apertar uma tecla do controle remoto. A área (2), desta figura, mostra ações de mudar de cena, sair da aplicação e mudar o canal da TV, por exemplo. Esta ferramenta não trata o contexto do usuário da aplicação gerada.

2.5.3 InteracTV

A ferramenta gráfica de autoria InteracTV (ALMEIDA; SOUZA; NETO, 2007) é destinada ao middleware MHP e também possui como objetivo facilitar a criação de aplicações interativas para TV digital. A princípio, ela permite a criação de dois tipos de aplicações: jogo de perguntas e respostas e informativos em geral, por exemplo sobre esportes e clima.

Os documentos XML gerados pelo InteracTV são recursos de entrada para o *framework* AppTV, que é o responsável por se comunicar com a API JavaTV e por gerar os componentes gráficos. A visualização da aplicação construída é possível através da invocação do *software* XleTView (SVEDEN, 2003).

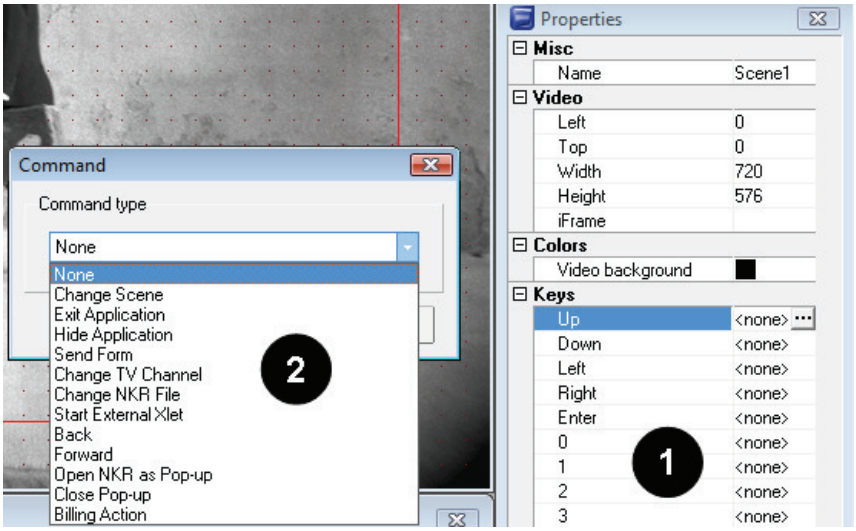


Figura 2.8: Tela de ações possíveis (Icareus iTV Suite Author).
[Fonte: Icareus (2009)]

Na interface gráfica da ferramenta (Figura 2.9) existem: duas abas - *Info* e *Quiz* (1), uma para cada tipo de aplicação que pode ser criada; uma área que exibe a tela da aplicação (2); e outra área com as propriedades do componente que está selecionado (3) (ALMEIDA; SOUZA; NETO, 2007).

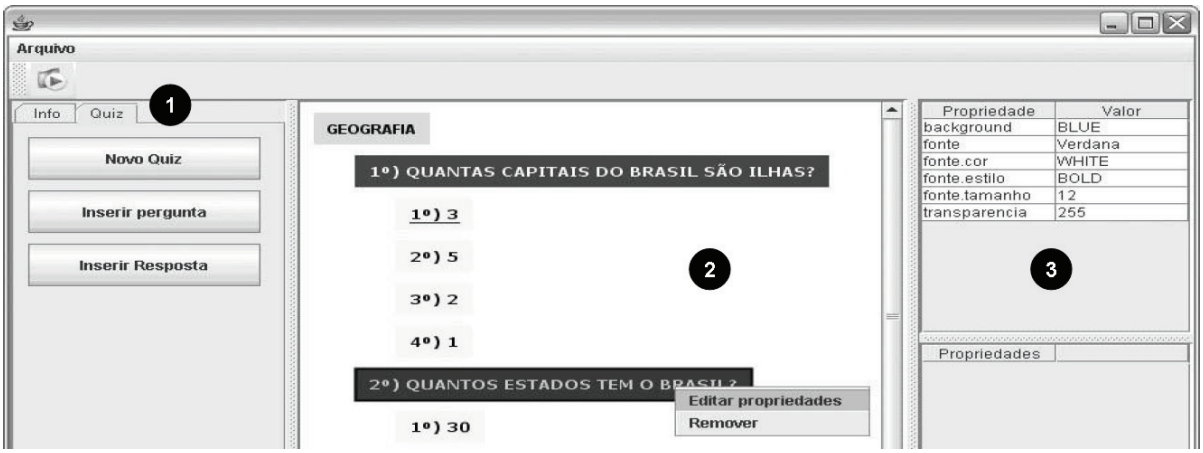


Figura 2.9: Tela da ferramenta InteracTV.
[Fonte: Almeida, Souza e Neto (2007)]

2.5.4 iTV Project

O iTV Project (OLIVEIRA; FILHO; SILVA, 2008) é uma ferramenta de autoria *web* para o desenvolvimento rápido de aplicações interativas de TV digital baseadas na especificação GEM (*Globally Executable MHP*). Além disso, ela também permite a realização de testes

e distribuição de aplicativos.

A ferramenta permite adição de componentes gráficos à tela, controle do fluxo das telas e gerenciamento de eventos. O fato de ser uma ferramenta *web*, diferentemente das demais aqui listadas, justifica-se pela maior acessibilidade da ferramenta, sendo suficiente que seus usuários estejam conectados à Internet. Outro elemento que justifica este ambiente de execução é a criação de um repositório *online* com os arquivos gerados pela ferramenta, possibilitando o compartilhamento de aplicações, por exemplo.

A ferramenta fornece sete tipos de componentes gráficos que podem ser adicionados à interface: campo de texto, área de texto, elipse, botão, imagem, rótulo, painéis que contêm outros componentes gráficos e imagem de fundo que pode ser colocada na tela da aplicação (OLIVEIRA; FILHO; SILVA, 2008).

Na Figura 2.10 (OLIVEIRA; FILHO; SILVA, 2008) podem ser vistos: (1) os componentes gráficos que podem ser adicionados à aplicação; (2) a área em que os componentes adicionados podem ser pré-visualizados; (3) opções para se alterar as propriedades dos componentes, como cor e fonte; (4) e as opções de navegação para a aplicação.

O iTV Project em si é apenas um editor visual que gera arquivos XML. Esses arquivos XML são interpretados pelo *framework* TUIG (Television User Interface Generator), que é o responsável por gerar GUI's (*Graphical User Interfaces*), isto é, classes Java, dinamicamente para o MHP (SILVA; OLIVEIRA, 2006).

2.5.5 JAME Author

O JAME Author (IMK, 2005) é uma ferramenta de autoria, proprietária, visual e com ambiente de execução *desktop*, que permite criar aplicações interativas de TV digital para a plataforma MHP. Ela permite definir o *design* e o comportamento da aplicação de forma separada. Esta ferramenta possui um emulador próprio para MHP, o que permite integração mais rápida para testes.

A interface gráfica do JAME Author, que pode ser vista na Figura 2.11, possui em destaque: uma lista com as telas (1); uma área que exibe a tela e permite sua edição visual (2); uma área para alterar as propriedades dos componentes presentes na tela (3); e uma área para gerenciamento das camadas gráficas da tela (4).

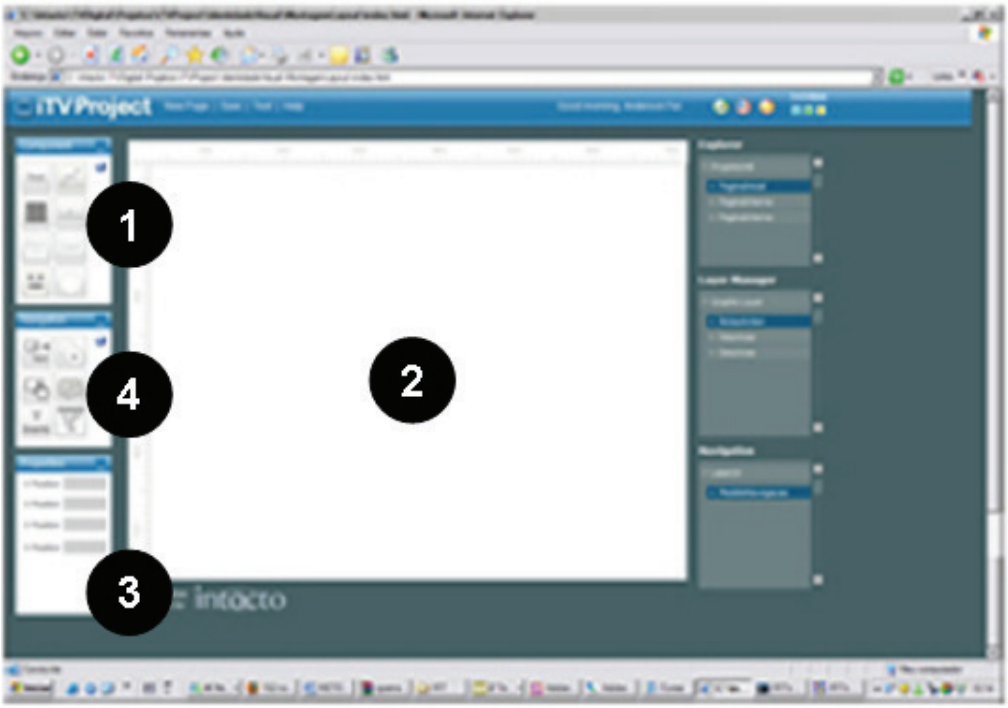


Figura 2.10: Tela da ferramenta iTV Project.
[Fonte: Oliveira, Filho e Silva (2008)]

2.5.6 NCL Eclipse

O *plug-in* NCL Eclipse, que está exibido na Figura 2.12 é um editor textual, gratuito, para documentos XML/NCL para a TV digital. Seu objetivo é agilizar o desenvolvimento de aplicações interativas em NCL, bem como tornar o processo de criação menos propenso a erros. Seus princípios são: aumento de produtividade, facilidade e integração no desenvolvimento (LAWS, 2008).

As características deste *plug-in* para o Eclipse são: coloração sintática das tags, atributos e comentários XML (1); *Outline View* (2), possibilitando navegar no documento NCL; sugestão de código (*autocomplete*) dinâmica e contextual (3); validação automática de documentos NCL e marcação do erro no documento (4), facilitando a sua localização; mecanismo para esconder/revelar nós XML; formatação automática de código XML; e execução do documento NCL usando o Ginga NCL Emulator.

2.5.7 SCO Creator Tool

A ferramenta SCO Creator Tool faz parte do T-MAESTRO (*T-Learning Multimedia Adaptative Educational SysTem based on Reassembling TV Objects*) (REY-LÓPEZ et al.,

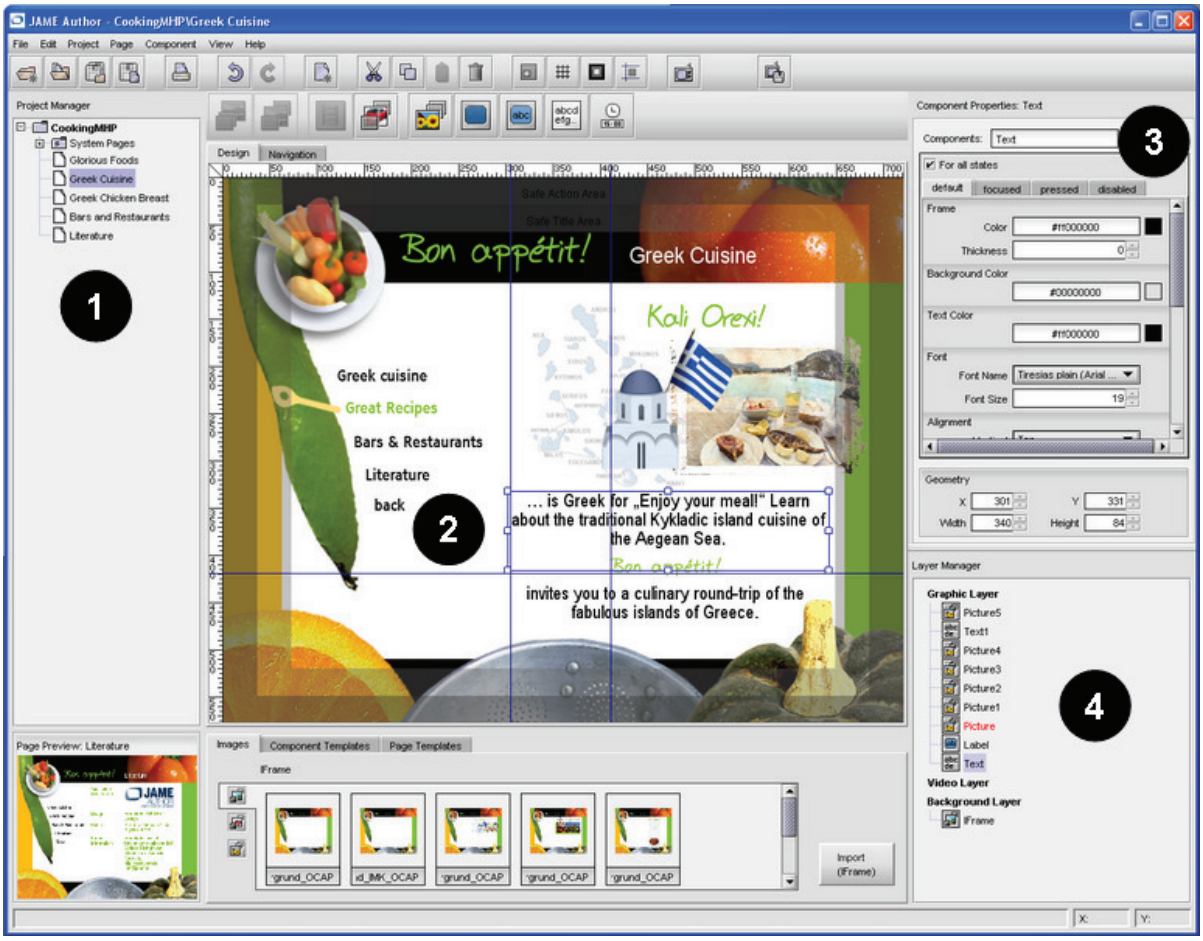


Figura 2.11: Tela da ferramenta JAME Author.
[Fonte: IMK (2005)]

2008). Este é um sistema de tutoria inteligente que utiliza TV digital para conceber experiências de aprendizado. SCO significa *Sharable Content Object*, ou seja, uma unidade de recurso de aprendizado.

Esta ferramenta de autoria permite que educadores construam cursos adaptados ao usuário e destinados ao middleware MHP. Dentre as ferramentas relacionadas com este trabalho, esta é a única que permite a adição de regras de adaptação, bem como a validação para encontrar inconsistências entre as mesmas, mas esta ferramenta é específica para aprendizado. Os educadores que utilizarem a ferramenta de autoria não precisam possuir conhecimento técnico sobre o desenvolvimento de aplicações interativas de TV digital (REY-LÓPEZ et al., 2008).

Na Figura 2.13 é exibida a interface da ferramenta com a lista de regras contextuais (1), a expressão lógica de uma regra (2) e a hierarquia de componentes da tela (3).

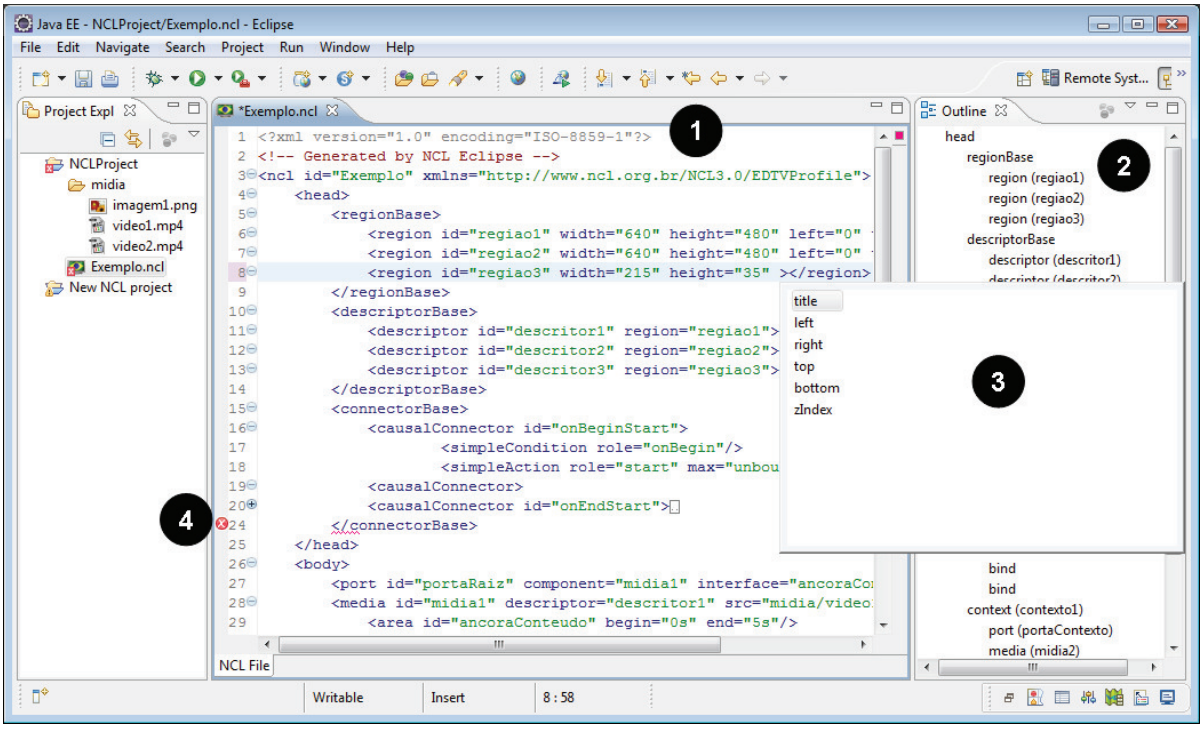


Figura 2.12: Tela do plug-in NCL Eclipse.
[Fonte: LAWS (2008)]

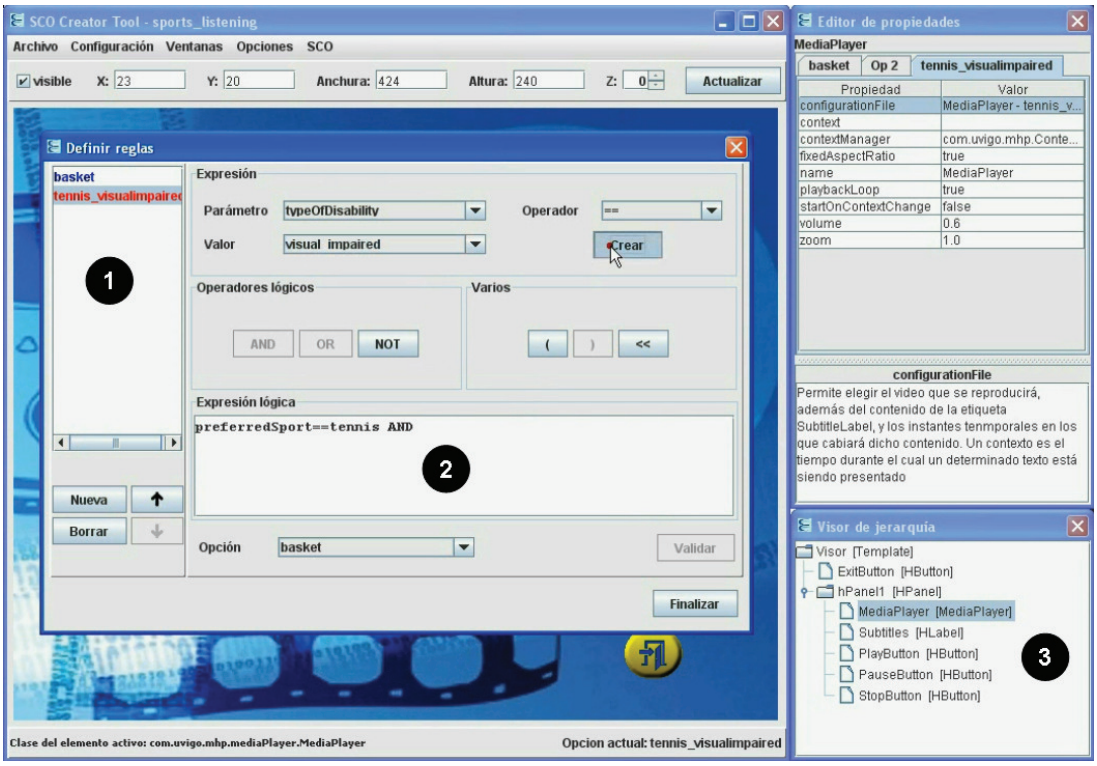


Figura 2.13: Tela da ferramenta SCO Creator Tool.
[Fonte: REY-LÓPEZ et al. (2008)]

2.6 Considerações Finais

Neste capítulo foi apresentado o referencial teórico no qual se baseia esta pesquisa. Foi explicado o conceito de interatividade na TV digital e como ela pode ser categorizada, segundo os dados que são enviados/recebidos através do canal de retorno disponível na televisão. Outros dois conceitos explicados, que são inter-relacionados, foram os de sensibilidade a contexto e *personas*. Estes dois últimos são a principal motivação para este trabalho sobre TV digital.

Sendo a ferramenta proposta por este trabalho voltada para o Ginga-NCL, o *middleware* Ginga foi descrito, mas com foco neste ambiente e em suas linguagens NCL e Lua. Elas foram detalhadas, pois o código gerado pela ferramenta proposta será escrito nestas duas linguagens.

Depois dessa introdução, foram descritas as principais ferramentas de autoria para TV digital interativa, através de suas características e funcionalidades. No Capítulo 3, será realizada uma comparação entre elas e a ferramenta aqui proposta.

3 *Contextual Ginga*

Neste capítulo, será apresentada a ferramenta *Contextual Ginga*, através de sua descrição inicial, suas funcionalidades, sua interface gráfica, e da descrição de sua arquitetura e implementação. Em seguida, será abordado como a geração de código ocorre. Ao final deste capítulo, será realizada uma comparação do *Contextual Ginga* com as ferramentas de autoria presentes no Capítulo 2, Seção 2.5.

3.1 Introdução

A ferramenta de autoria desenvolvida por este trabalho, denominada *Contextual Ginga* (CARVALHO; FERRAZ, 2010), possui como objetivo principal permitir a construção de aplicações interativas sensíveis a contexto para TV digital e que sejam executadas na plataforma Ginga-NCL. Estas aplicações são geradas nas linguagens NCL e Lua de forma automática, ou seja, sem a necessidade de codificação.

O conjunto de elementos gráficos que formam uma aplicação devem estar representados na ferramenta dentro de um mesmo projeto. Estes elementos são as telas que reúnem uma coleção de componentes que são exibidos por vez, mas que existem em todos os *personas*. A ferramenta permite adicionar três tipos de componentes gráficos: texto, imagem e vídeo. Uma primeira imagem da ferramenta pode ser vista na Figura 3.1.

As propriedades de um componente do tipo texto são: nome, texto, posição, dimensões, cor da fonte, cor do fundo e estilo da fonte. Um componente deste tipo pode conter texto em uma ou mais linhas. As propriedades de um componente do tipo imagem são: nome, posição, dimensões, imagem e imagem com foco. Com esta possibilidade de permitir que um componente possua duas imagens, juntamente com as transições, pode-se fazer que ele apresente o comportamento de um botão. Se o componente representar sempre a mesma imagem, é suficiente atribuir valor para a imagem sem foco.

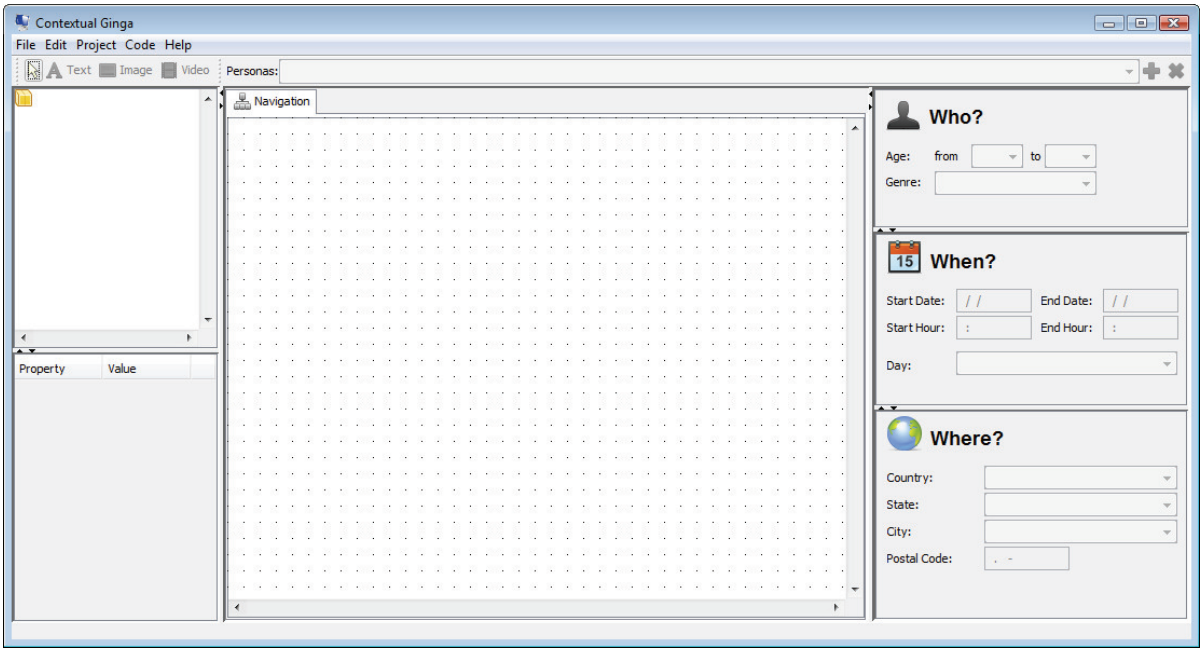


Figura 3.1: Imagem do *Contextual Ginga*.

O componente do tipo vídeo representa o vídeo principal que o documento NCL carrega. Para um projeto, é permitido adicionar apenas um vídeo.

As telas são interligadas através de transições (Figura 3.2). Uma transição representa a mudança de foco de um componente para outro (do componente C1 para C2), podendo estes componentes estarem na mesma tela (do componente C1 para C2, na Tela 1 para o *persona* X) ou em telas diferentes (do componente C1 na Tela 1 para C3 na Tela 2 para o *persona* Y). Com essa mudança de foco, são construídos os possíveis fluxos de navegação da aplicação. Uma transição é composta pelos elementos: tela de origem, componente de origem, tela de destino, componente de destino, *persona* e um botão que quando acionado dispara a mudança de foco de componente.

Os possíveis fluxos dependem do contexto do usuário da aplicação. Este contexto é formado pelas categorias *Who*, *When* e *Where*, que em conjunto definem um *persona*. A categoria *Who* do *persona* possui propriedades que definem a faixa etária e o gênero/sexo do usuário. A categoria *When* possui propriedades que definem uma data e uma hora inicial e final, e um dia da semana. A *Where* possui como propriedades o país, o estado, a cidade e o código postal do usuário.

As demais categorias contextuais não estão disponíveis; pois, para a categoria *What*, foi considerada a própria atividade de interagir com a televisão. A categoria *Why* não foi considerada uma vez que representa uma grande dificuldade devido ao entendimento

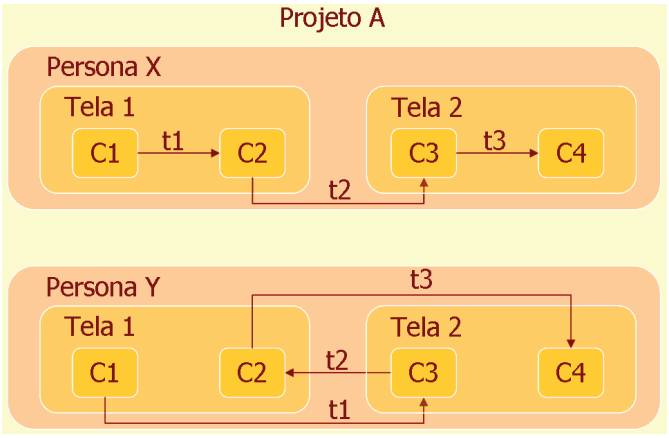


Figura 3.2: Conceitos envolvidos no *Contextual Ginga*.

necessário sobre o raciocínio de um ser humano e esta categoria normalmente é construída sobre as outras. Por fim, para a categoria *How*, no momento a ferramenta coleta informações contextuais exclusivamente vindo de um arquivo com as características do usuário e da data e da hora do sistema, ou seja, a aplicação não busca por exemplo preferências do usuário que estejam armazenadas num dispositivo móvel, como o seu celular, ou em um servidor remoto (COÊLHO, 2008).

As funcionalidades foram descritas e implementadas através de um processo iterativo e incremental (Figura 3.3). Neste processo, em cada iteração, foram realizadas as seguintes atividades:

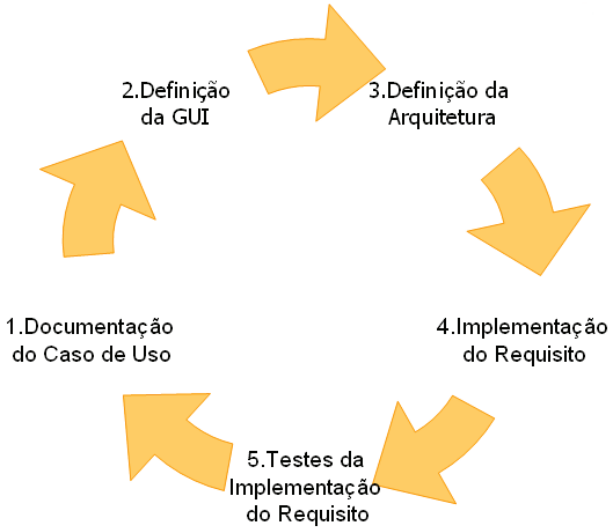


Figura 3.3: Processo de desenvolvimento.

1. **Documentação dos casos de uso** (Apêndice A), para descrever sobretudo os

fluxos de eventos que precisariam ser implementados;

2. **Definição da interface gráfica com o usuário**, buscando manter semelhança com outras ferramentas, sobretudo as de autoria em geral, para facilitar a aprendizagem de utilização da ferramenta por parte dos usuários;

3. **Definição da arquitetura**, com suas classes e relacionamentos entre elas. A arquitetura possui como elementos fundamentais as classes que representam a interface gráfica e a geração de código;

4. **Implementação dos requisitos**, buscando documentar o código para facilitar futuras atualizações;

5. **Testes da implementação dos requisitos**, realizados a cada ciclo de implementação e também durante a execução do experimento.

Apesar de não estarem presentes em todos os ciclos do desenvolvimento, devem ser destacadas as atividades de definição e implementação de como seria gerado o código da aplicação de TV digital. Estas atividades foram fundamentais para atingir o objetivo específico de geração automática de código.

3.2 Funcionalidades

As funcionalidades do *Contextual Ginga*, em número de 24, foram baseadas nas ferramentas relacionadas a este trabalho, adicionando-se principalmente novas funcionalidades para tratar do contexto do usuário. Elas estão descritas nas subseções seguintes, que agrupam os conceitos de projeto, de tela, de componente, de *persona*, de transição e de geração de código NCLua (NCL e Lua).

As funcionalidades diretamente relacionadas a contexto são as dos conceitos de transição e de *persona*. Todas as funcionalidades estão detalhadas nos casos de uso do Apêndice A. Além disso, no Apêndice D, encontra-se o manual do usuário da ferramenta, com instruções de instalação e utilização do *Contextual Ginga*.

As funcionalidades da ferramenta estão classificadas como: essencial, importante ou desejável. As essenciais são aquelas que não podem estar ausentes na ferramenta. Se uma destas estiver faltando, este projeto não terá a utilidade visada. As importantes podem não estar presentes na ferramenta, mas os seus usuários não a utilizarão de forma satisfatória. Por fim, as desejáveis podem ser incorporadas à ferramenta em versões

posteriores porque mesmo sem elas, a ferramenta possuirá suas funcionalidades suficientes. Apesar desta categorização das funcionalidades, todas elas foram implementadas.

As funcionalidades foram implementadas visando atingir também os seguintes requisitos não-funcionais de usabilidade: facilidade de aprendizado e utilização; facilidade de memorização; e diminuição da taxa de erros. O primeiro requisito se refere à facilidade que o desenvolvedor experimenta ao utilizar a ferramenta pela primeira vez e conseguir realizar tarefas básicas. O segundo se refere à propriedade da ferramenta permitir que, mesmo depois de um tempo sem utilizá-la, o desenvolvedor consiga voltar a interagir com ela sem precisar aprendê-la novamente. O terceiro se refere à ferramenta permitir que os seus usuários cometam um baixo número de erros e que eles consigam se recuperar facilmente deles. Ou seja, a ferramenta deve ter a capacidade de prevenir que os usuários se encontrem em situações de erro e, caso se encontrem, que as mensagens de erro sejam claras e sugiram uma solução (NIELSEN, 1994).

Para satisfazer os dois primeiros requisitos, a interface da ferramenta foi projetada para ser simples, permitindo que a maior parte das funcionalidades gráficas esteja disponível diretamente na interface principal. Para satisfazer o terceiro, na maior parte dos campos que o usuário precisa fornecer algum valor existem restrições e validações de valores que podem ser informados.

Visando estimular a inserção de sensibilidade a contexto nas aplicações geradas, os campos que os usuários utilizam para construir as características de um *persona* e o cenário do momento da interação estão sempre visíveis enquanto a ferramenta está sendo executada. Mesmo assim, o usuário pode criar, através da ferramenta, uma aplicação que não seja sensível a contexto, sendo suficiente utilizar um *persona Default*, ou seja, um *persona* com este nome que não possuirá valores para as características contextuais.

3.2.1 Projeto

A seguir, são descritas as funcionalidades referentes ao conceito de projeto. A primeira funcionalidade deste conceito é **Criar Projeto**. Classificada como **Essencial**, ela permite agrupar um conjunto de telas para formar uma aplicação de TV digital numa determinada resolução.

A Figura 3.4 mostra onde está disponível esta funcionalidade na ferramenta. Por sua vez, a Figura 3.5 mostra onde o desenvolvedor deve informar o diretório e o nome do projeto, tendo como exemplo o projeto *Canal* para representar um canal de informações

para o usuário.

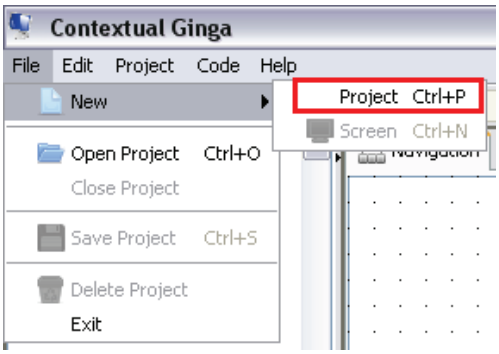


Figura 3.4: Criação de um novo projeto.

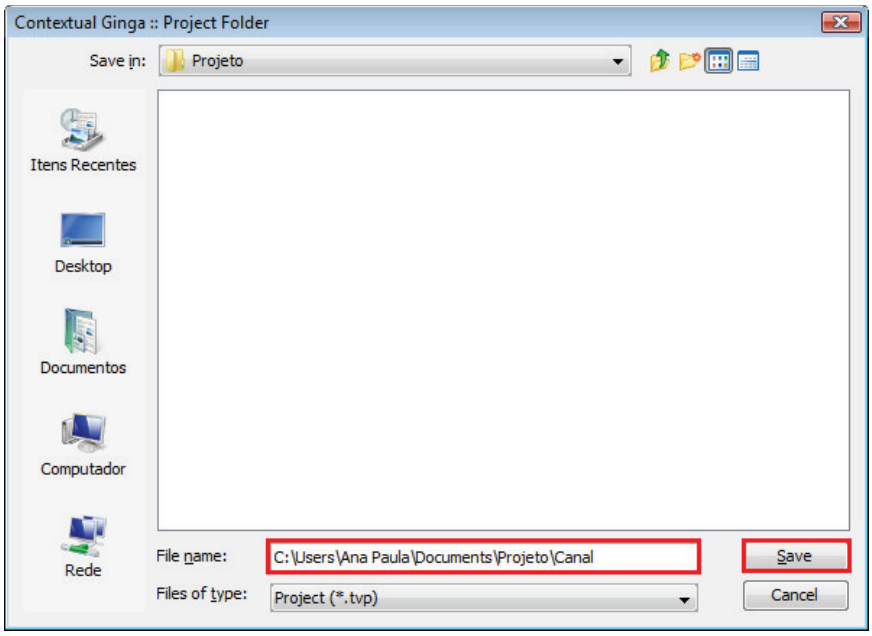


Figura 3.5: Diretório e nome do projeto.

A segunda funcionalidade é **Abrir Projeto**. Classificada como **Essencial**, permite que se visualize todos os dados de um projeto e que se trabalhe nele. Apenas um projeto pode ser visualizado por vez.

A terceira funcionalidade é **Salvar Projeto**. Classificada como **Essencial**, permite que sejam salvos todos os dados do projeto, ou seja, todas as telas com seus componentes e transições, bem como os *personas*.

A quarta funcionalidade é **Fechar Projeto**. Classificada como **Importante**, permite que não mais se visualize os dados de um projeto.

A quinta funcionalidade é **Excluir Projeto**. Classificada como **Desejável**, permite excluir os arquivos com informações sobre o projeto em que se está trabalhando.

3.2.2 Tela

A seguir, são descritas as funcionalidades referentes ao conceito de tela. A primeira funcionalidade deste conceito é **Criar Tela**. Classificada como **Essencial**, ela permite adicionar uma tela ao projeto. Para a criação é necessário apenas informar um nome. Ao criar uma tela, ela já é exibida em formato gráfico para que se possa iniciar a adição de componentes.

Continuando o exemplo presente nas Figuras 3.4 e 3.5, a criação de uma tela de notícias gerais, denominada *Noticia*, é mostrada na Figura 3.6, que mostra onde a funcionalidade está disponível, e na Figura 3.7, que mostra onde o desenvolvedor deve informar o nome da tela.

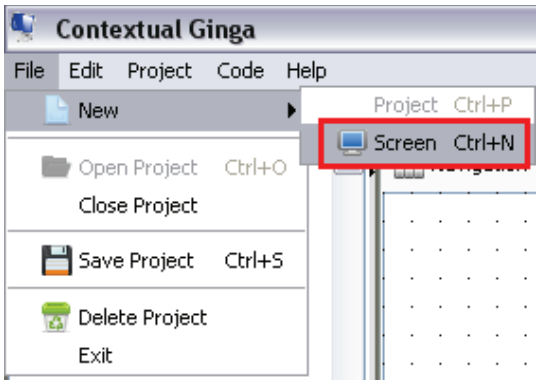


Figura 3.6: Criar tela.

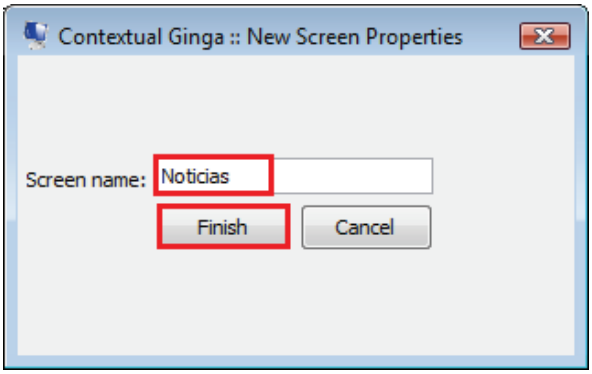


Figura 3.7: Nome da tela.

A segunda funcionalidade é **Listar Telas**. Classificada como **Essencial**, permite que sejam exibidos os nomes de todas as telas pertencentes ao projeto e, dessa forma, poder escolher uma para visualizar graficamente.

A terceira funcionalidade é **Exibir Tela em Editor Gráfico**. Classificada como **Essencial**, em função do objetivo principal desta ferramenta, ela permite que as telas e seus componentes sejam exibidos em formato gráfico. Apenas uma tela pode ser visualizada por vez.

A quarta funcionalidade é **Renomear Tela**. Classificada como **Desejável**, ela permite alterar o nome de uma tela.

A quinta funcionalidade é **Excluir Tela**. Classificada como **Desejável**, ela permite excluir uma tela de um projeto. Com esta exclusão, também são excluídas as transições

que se originam desta tela e que se destinam a ela. Além disso, caso existam, os pontos iniciais (conjunto de componentes iniciais e suas respectivas telas) formados por esta tela também serão excluídos.

3.2.3 Componente

A primeira funcionalidade deste conceito é **Adicionar Componente**. Classificada como **Essencial**, ela permite que o usuário, depois de selecionar um dos tipos de componente, adicione um novo componente deste tipo à tela que está sendo mostrada na ferramenta. Na Figura 3.8, é mostrada em destaque a barra com os tipos de componentes disponíveis e, na Figura 3.9, é mostrado onde o desenvolvedor deve informar o nome do componente que está sendo adicionado.

Figura 3.8: Tipos de componente.

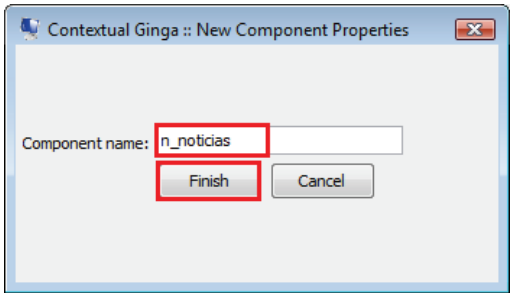


Figura 3.9: Nome do componente.

A segunda funcionalidade é **Excluir Componente**. Classificada como **Essencial**, ela permite excluir um componente de uma tela. Quando o componente é excluído: são excluídas as transições associadas; o componente deixa de existir na visão de todos os *personas* adicionados ao projeto; e se este componente com sua tela eram o ponto inicial de um *persona*, este passa a ter ponto inicial não definido.

A terceira funcionalidade é **Mover Componente**. Classificada como **Essencial**, ela permite mover graficamente um componente através de *drag-and-drop*. A posição de um componente pode ser alterada também ao se mudar a propriedade de posição do elemento.

A quarta funcionalidade é **Definir Componente e Tela Iniciais**. Classificada como **Essencial**, ela permite determinar para cada *persona* do projeto, qual a primeira tela a ser exibida e qual o componente desta tela que receberá foco inicialmente, ou seja, dependendo das características do usuário que interage com a aplicação, esta começa a ser executada de maneira diferente. Esta funcionalidade deve ser executada mesmo quando o projeto não considerar o contexto do usuário. Pois, mesmo em uma aplicação que não utilize informações contextuais, é preciso definir qual componente será o primeiro a receber o

foco. Na Figura 3.10, pode ser visualizada a execução da funcionalidade de definição.

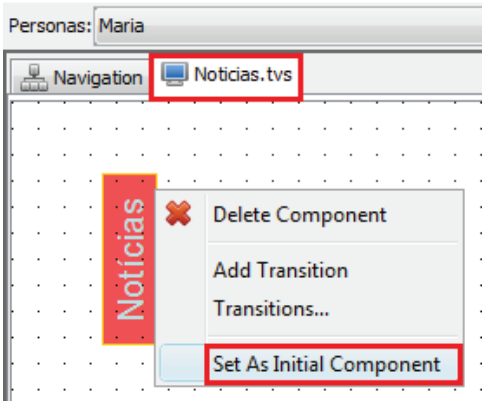


Figura 3.10: Definir componente e tela iniciais.

A quinta funcionalidade é **Alterar Propriedade de um Componente**. Classificada como **Essencial**, ela permite alterar o valor de uma propriedade de um componente a partir de uma lista de propriedades que variam de acordo com o tipo do componente. Na Figura 3.11, são exibidas as propriedades de componente tipo texto e, na Figura 3.12, as propriedades de componente tipo imagem. O tipo vídeo possui, como propriedade, apenas o seu nome. Ao clicar no tipo vídeo (Figura 3.8), uma janela para alteração é exibida.

Property	Value
Name	texto
Text	Text
Location	53, 80
Size	38, 16
Foreground	0, 0, 0
Background	224, 223, 227
Font Style	Dialog, Plain, 12

Figura 3.11: Propriedades de componente tipo texto.

Property	Value
Name	n_noticias
Location	52, 41
Size	32, 99
Image	C:\Users\Ana...
Image on Focus	C:\Users\Ana...

Figura 3.12: Propriedades de componente tipo imagem.

3.2.4 Persona

A seguir, são descritas as funcionalidades referentes ao conceito de *persona*. A primeira funcionalidade deste conceito é **Adicionar Persona**. Classificada como **Essencial**, ela permite adicionar um *persona* ao projeto. Para a criação do *persona*, deve ser informado um nome para ele. Na Figura 3.13, é exibida a adição de um *persona* de nome *Maria*.

A segunda funcionalidade é **Excluir Persona**. Classificada como **Essencial**, ela

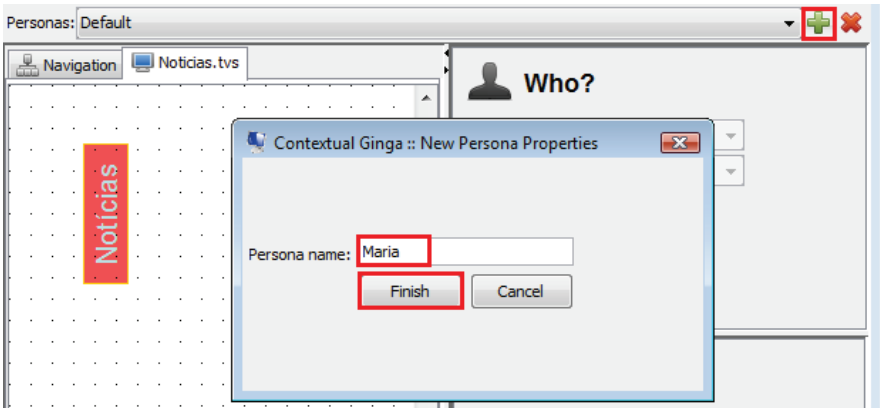


Figura 3.13: Adicionar *persona*.

permite excluir um *persona* do projeto. Quando um *persona* é excluído, as transições associadas a ele são excluídas também.

A terceira funcionalidade é **Alterar Contexto do Persona**. Classificada como **Essencial**, ela permite alterar as características contextuais de um *persona* para as categorias: *Who* (Figura 3.14), *When* (Figura 3.15) e *Where* (Figura 3.16).

Figura 3.14: Característica *who*.

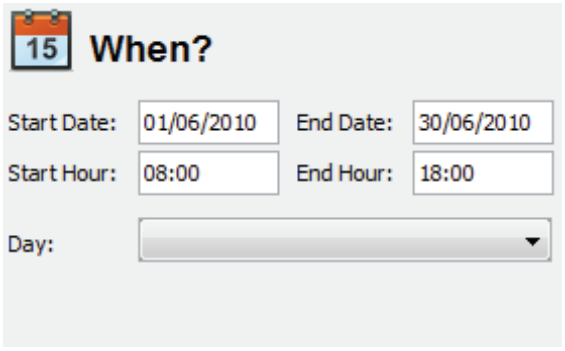


Figura 3.15: Característica *when*.

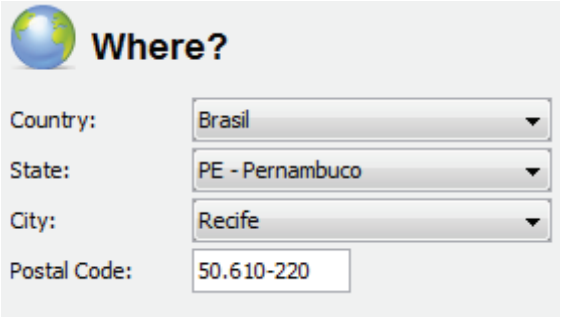


Figura 3.16: Característica *where*.

3.2.5 Transição

A seguir, são listadas as funcionalidades referentes ao conceito de transição. A primeira funcionalidade deste conceito é **Adicionar Transição**. Classificada como **Essencial**, ela permite que se crie uma transição entre dois componentes, estando eles na mesma tela ou em telas distintas.

Na Figura 3.17, é mostrado onde o desenvolvedor pode encontrar a funcionalidade de adicionar transição. E, na Figura 3.18, são exibidas as informações necessárias para adicioná-la, exemplificando uma transição do componente *n_noticias*, presente na tela de origem *Noticias.tvs*, para o componente *e_esportes*, presente na tela de destino *Esportes.tvs*, quando o usuário da aplicação apertar a seta de navegação para baixo. As informações de tela e componente de origem já são obtidas a partir do componente selecionado pelo desenvolvedor. A extensão *.tvs* significa *TV Screen*.

O prefixo *n*, no componente *n_noticias*, e o prefixo *e*, no componente *e_esportes*, citados como exemplo, referenciam a letra inicial do nome da tela que cada componente se encontra. Este padrão foi adotado apenas neste exemplo com o objetivo de nomear cada componente diferentemente. Pois, por uma restrição da ferramenta, ainda não é possível adicionar dois componentes com o mesmo nome em telas diferentes.

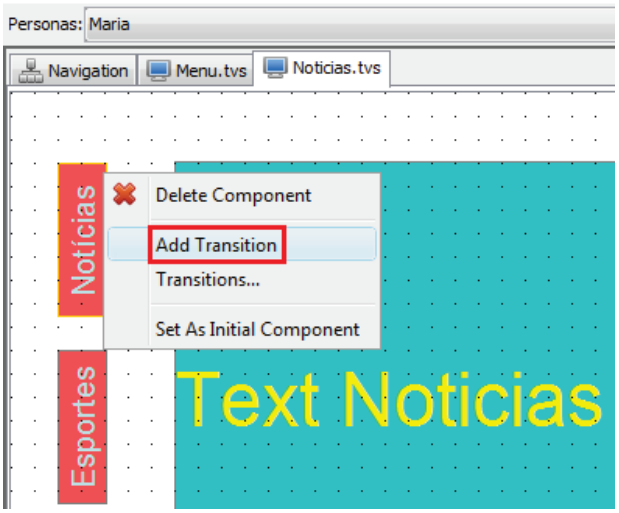


Figura 3.17: Adicionar transição.

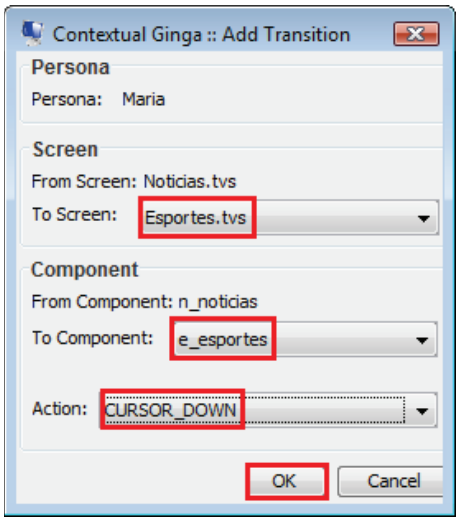


Figura 3.18: Valores da transição.

A segunda funcionalidade é **Visualizar Transições por Componente**. Classificada como **Essencial**, ela permite visualizar as transições que são originadas em um componente ou tela para certo *persona*.

A terceira funcionalidade é **Excluir Transição**. Classificada como **Essencial**, ela

permite que se exclua uma transição entre dois componentes.

A quarta funcionalidade é **Alterar Transição**. Classificada como **Essencial**, ela permite que se altere as propriedades de uma transição.

A quinta funcionalidade é **Visualizar Transições Graficamente**. Classificada como **Importante**, ela permite que sejam visualizadas, para um dado *persona*, as transições existentes entre as telas de maneira similar a um diagrama de estados. Mesmo que não existam transições, as telas são representadas graficamente. Esta funcionalidade também permite que o usuário da ferramenta altere a exibição da navegação entre as telas.

Uma linha entre os nomes de duas telas indica que existe uma transição entre elas e um arco indica o seu sentido da origem para o destino. Se entre duas telas existem transições nos dois sentidos, os dois arcos, formam um círculo. Um círculo, no canto superior esquerdo do nome da tela, representa uma transição entre componentes da mesma tela.

Na Figura 3.19, pode ser visto um fluxo de navegação entre as telas para o *persona Maria*, citado como exemplo. No *persona Maria*, da tela *Menu.tvs*, alcançam-se as telas *Noticias.tvs* e *Esportes.tvs*. Da tela *Noticias.tvs*, alcança-se as duas demais. A tela *Esportes.tvs* possui apenas transição para a tela *Noticias.tvs*. Além das transições entre telas, a tela *Menu.tvs* possui uma auto-transição, ou seja, uma transição entre dois componentes da mesma tela.

Dessa forma, entre as telas *Menu.tvs* e *Noticias.tvs*, existem transições em ambos os sentidos, bem como entre as telas *Noticias.tvs* e *Esportes.tvs*. Entre as telas *Menu.tvs* e *Esportes.tvs*, existe transição apenas da primeira tela para a segunda. Apenas a tela *Menu.tvs* possui auto-transição.

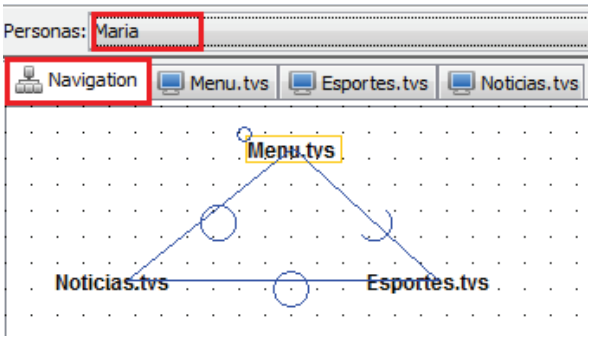


Figura 3.19: Navegação entre telas.

3.2.6 Código NCLua

A funcionalidade **Gerar Código NCLua**, classificada como **Essencial**, permite produzir o código da aplicação para a plataforma Ginga-NCL a partir dos dados presentes em um projeto construído na ferramenta, ou seja, os dados do projeto ficam armazenados em arquivos específicos da ferramenta e apenas quando se executa a operação para gerar código, é que o código NCLua é gerado. A Figura 3.20 exibe onde, na ferramenta, o código pode ser gerado. Exemplo e explicação do código serão fornecidos na Seção 3.5.

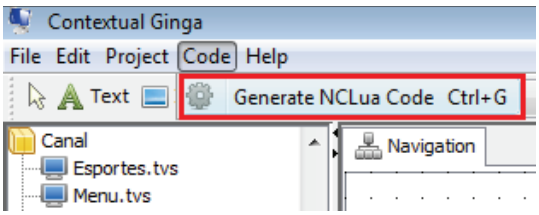


Figura 3.20: Gerar código NCLua.

3.3 Interface Gráfica

Na Figura 3.21, pode ser vista a interface gráfica do *Contextual Ginga* dividida por áreas. A área (1) exibe o nome do projeto *Canal*, dado como exemplo, aberto ao lado de uma imagem de pacote e abaixo os nomes das telas associadas ao projeto (*Esportes.tvs*, *Menu.tvs* e *Noticias.tvs*).

A área (2) contém uma aba (*Navigation*) para exibir a representação dos fluxos entre as telas e abas para exibir a representação gráfica de cada tela com seus componentes. Os dois tipos de abas permitem movimentação através de *drag-and-drop* do nome da tela, no caso da navegação, ou do componente, no caso da tela. A representação dos fluxos varia de acordo com o nome do *persona* selecionado na área (5).

A área (3) exibe os tipos de componentes que podem ser adicionados. Apenas o tipo vídeo não pode ser adicionado à representação gráfica da tela, uma vez que não representa uma imagem estática. Ele é exibido apenas em tempo de execução. A área (4) exibe o valor corrente de cada propriedade de um componente selecionado na área (2). As propriedades exibidas variam de acordo com o tipo do componente.

A área (5) exibe a lista de *personas* associados ao projeto e permite adicionar e excluir um determinado *persona*. A área (6) exibe as características do *persona* selecionado na área (5), divididas nas três categorias consideradas.

Figura 3.21: Imagem da tela principal do Contextual Ginga.

3.4 Arquitetura e Implementação

O *Contextual Ginga* é uma ferramenta para ser executada em ambiente *desktop* e foi desenvolvida na linguagem Java, versão 1.6, sem adição de bibliotecas. A sua arquitetura está estruturada seguindo o padrão de camadas com: interface, negócio, acesso a dados e geração de código de aplicações de TVDi (Figura 3.22). Esta última será detalhada na próxima seção.

A **camada de interface** gráfica da ferramenta foi desenvolvida utilizando a API (*Application Programming Interface*) Swing e a API AWT (*Abstract Window Toolkit*), ambas nativas do próprio Java (MICROSYSTEMS, 2006). Como ferramenta de desenvolvimento do *Contextual Ginga* foi utilizada a IDE (*Integrated Development Environment*) Eclipse.

A **camada de negócio** possui: os cadastros de projeto, tela, *persona* e transição; suas entidades; e a fachada da ferramenta (Figura 3.22). Das entidades presentes na arquitetura, um projeto possui uma resolução, para o dimensionamento da aplicação, e uma coleção de telas. Cada tela possui uma coleção de transições que partem da mesma. Um projeto também possui uma coleção de *personas*, cada um deles com um ponto (tela e componente) inicial e as categorias de contexto *Who*, *Where* e *When*. Esta última categoria possui tempo inicial e final. O relacionamento entre estas entidades pode ser

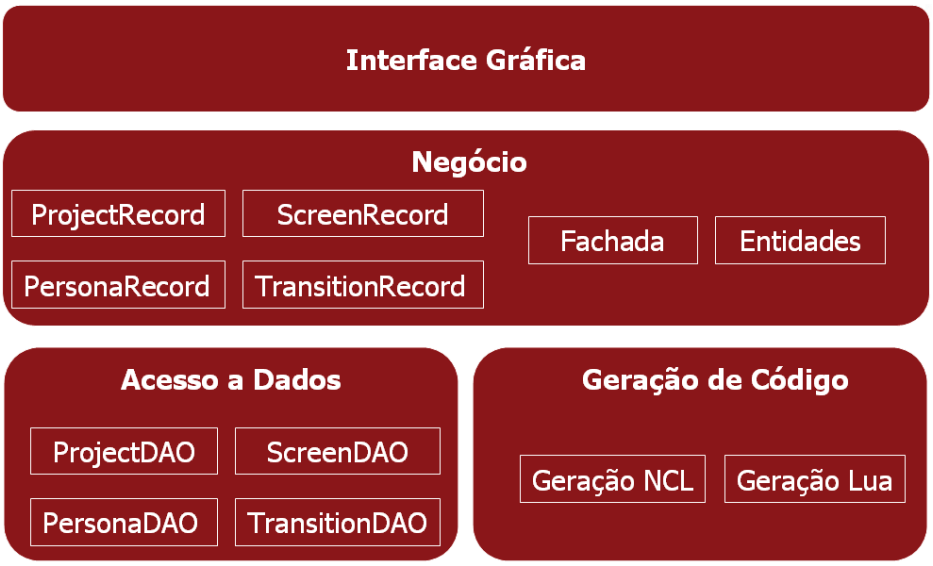


Figura 3.22: Arquitetura do *Contextual Ginga*.

visto no diagrama de classes da Figura 3.23. Neste diagrama não consta a entidade componente, pois foi utilizada a representação que existe na API AWT.

Na **camada de acesso a dados** encontram-se a leitura e a escrita dos arquivos salvos pela ferramenta. Os dados de um projeto estão armazenados em arquivos XML e, portanto, foi necessária a utilização de uma API para o processamento destes arquivos. A API utilizada foi a *javax.xml.parsers*, também nativa do Java, através do analisador baseado em SAX (*Simple API for XML*). Esta abordagem foi escolhida em relação ao DOM (*Document Object Model*) (W3C, 2005) pelo fato do processamento do arquivo ser realizado de forma seqüencial, não sendo necessários acessos aleatórios ao conteúdo do arquivo.

Os arquivos XML criados pelo *Contextual Ginga* para uma projeto dividem-se em quatro tipos: um de características do projeto, um para os componentes de uma tela, um para os *personas* e um último para as transições. O arquivo de projeto possui extensão *.tvp* (*TV Project*). Já os arquivos de componentes de tela possuem extensão *.tvs* (*TV Screen*). Por fim, os arquivos de transições e *personas* possuem extensão *.ctx* (*context*), que armazenam informações relacionadas ao uso de contexto na aplicação. Exemplos completos destes arquivos estão presentes no Apêndice E.

As características do projeto armazenadas no primeiro tipo de arquivo são: nome do projeto (elemento *projectName*); resolução para a aplicação (elementos *resolution, width* e *height*); o nome do vídeo principal a ser exibido (elemento *video*); e para cada nome de tela

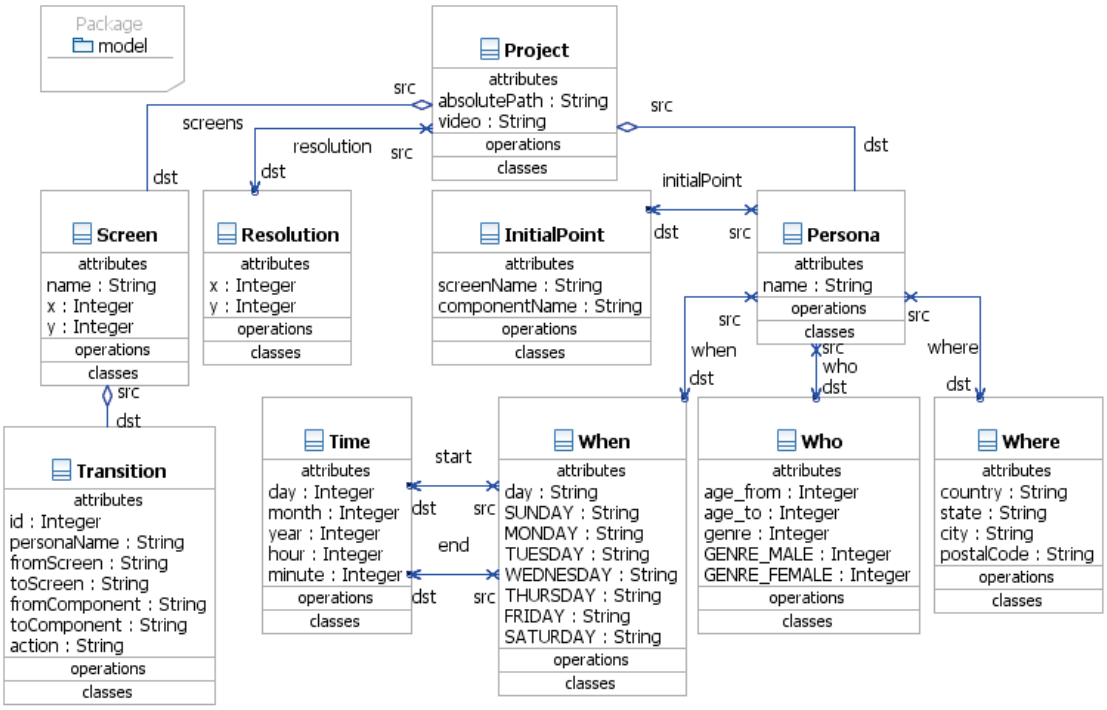


Figura 3.23: Diagrama de classes das entidades.

do projeto (atributo *name*), a posição (elementos x e y) que este nome está localizado na aba *Navigation*. O esquema deste XML pode ser visualizado na Figura 3.24 e um exemplo de arquivo do projeto *Canal*, nos Códigos 3.1 e 3.2.

Código 3.1: Arquivo Canal.tvp (Parte 1).

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <projectDescription>
3      <projectName>Canal</projectName>
4      <resolution>
5          <width>640</width>
6          <height>480</height>
7      </resolution>
8      <video>corrego1.mp4</video>
9      <screens>
10         <screen name="Noticias.tvs">
11             <x>78</x>
12             <y>265</y>
13         </screen>
14         <screen name="Esportes.tvs">
15             <x>362</x>
16             <y>265</y>
17         </screen>
18         <screen name="Menu.tvs">
19             <x>223</x>
20             <y>168</y>
21         </screen>
```

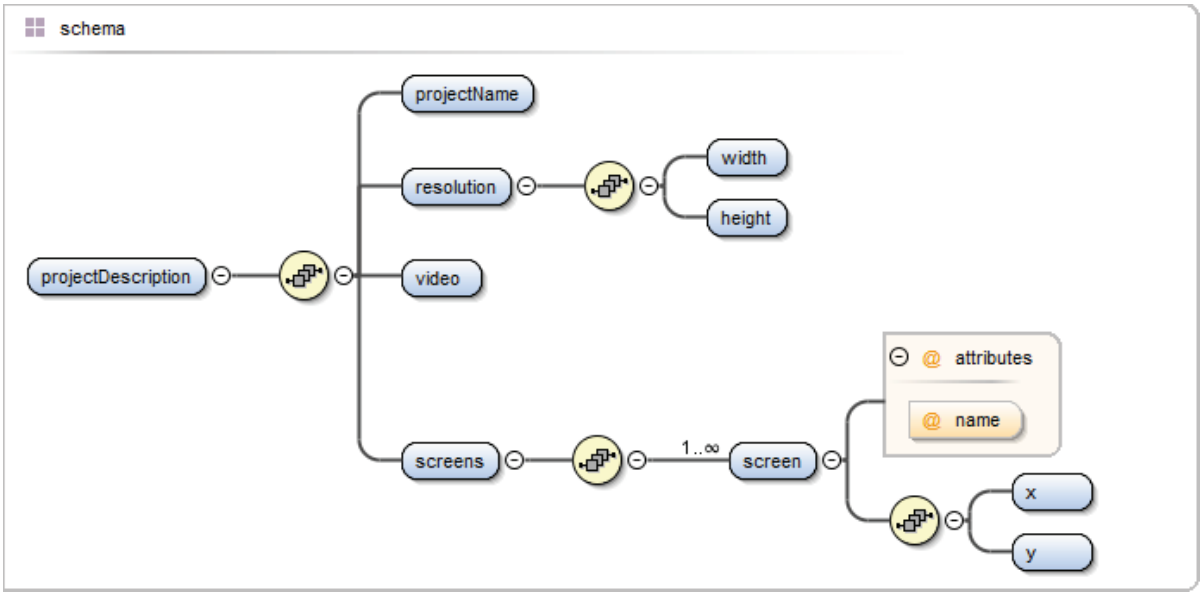


Figura 3.24: Esquema XML para arquivo com características do projeto.

Código 3.2: Arquivo Canal.tvp (Parte 2).

```
22      </screens>
23  </projectDescription>
```

Uma tela possui um nome (atributo *name*) e um conjunto de componentes. Cada componente possui obrigatoriamente os elementos XML: tipo do componente (elemento *type*); posição x (elemento *x*); posição y (elemento *y*); largura (elemento *width*); e altura (elemento *height*). Os elementos texto (*text*), fonte (*font*), cor de fundo (*background*) e cor de fonte (*foreground*) apenas existem quando o componente é do tipo texto. Os elementos imagem (*image*) e imagem com foco (*imageOnFocus*) apenas existem quando o componente é do tipo imagem. O esquema deste XML pode ser visualizado na Figura 3.25, e um exemplo de parte do arquivo da tela *Noticias.tvs*, nos Códigos 3.3 e 3.4.

Código 3.3: Arquivo Noticias.tvs (Parte 1).

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <screen name="Noticias.tvs">
3      <component name="n_noticias">
4          <type>javax.swing.JLabel</type>
5          <x>30</x>
6          <y>45</y>
7          <width>32</width>
8          <height>99</height>
9          <image>C:\img\Noticias.png</image>
10         <imageOnFocus>C:\img\NoticiasSelecionado.png</imageOnFocus>
11     </component>
12     <component name="texto">
```

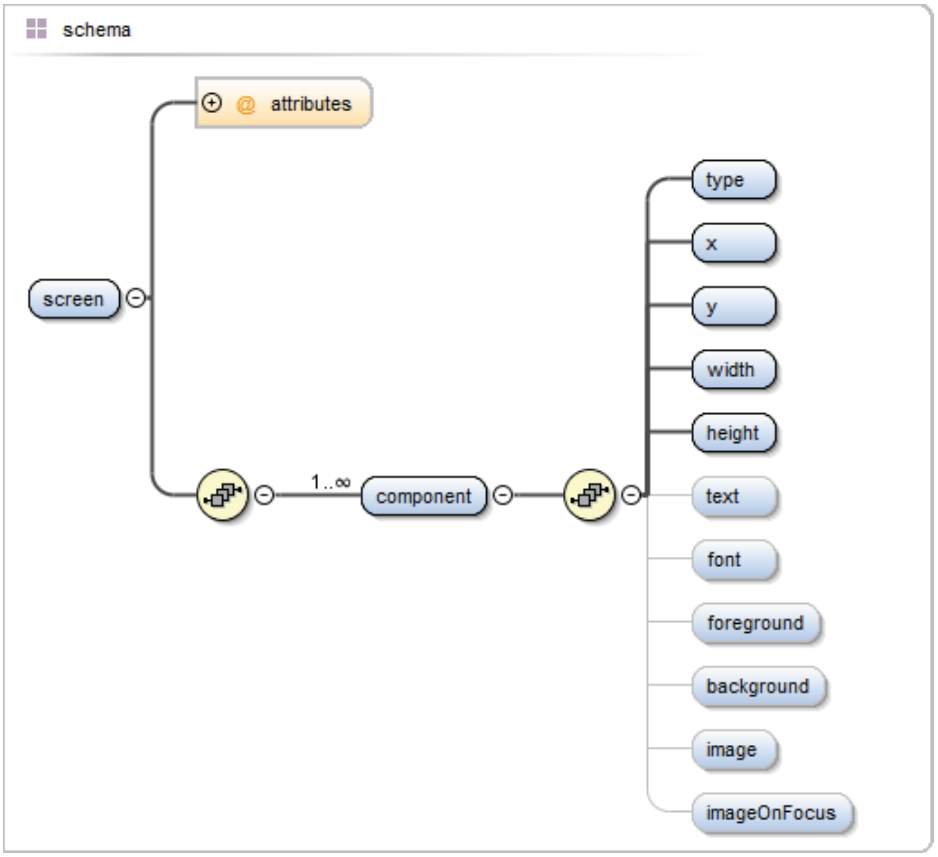


Figura 3.25: Esquema XML para arquivo de tela.

Código 3.4: Arquivo Noticias.tvs (Parte 2).

```
13          <type>javax.swing.JLabel</type>
14          <x>105</x>
15          <y>44</y>
16          <width>500</width>
17          <height>300</height>
18          <text>Text Noticias</text>
19  (...)
```

No terceiro tipo de arquivo XML, são armazenados os *personas*. Cada um deles possui um nome (atributo *name*), uma tela inicial (atributo *initialScreen*), um componente inicial (atributo *initialComponent*) e três categorias contextuais. A categoria *Who* possui os elementos idade inicial (*ageFrom*) e idade final (*ageTo*) da faixa etária e o elemento sexo (*genre*). A categoria *When* possui os elementos data e hora inicial (*start*), data e hora final (*end*) e o dia da semana (*dayOfWeek*). A categoria *Where* possui os elementos país (*country*), estado (*state*), cidade (*city*) e código postal (*postalCode*). O esquema deste XML pode ser visualizado na Figura 3.26, e um exemplo de parte do arquivo dos *personas* do projeto *Canal*, nos Códigos 3.5 e 3.6.

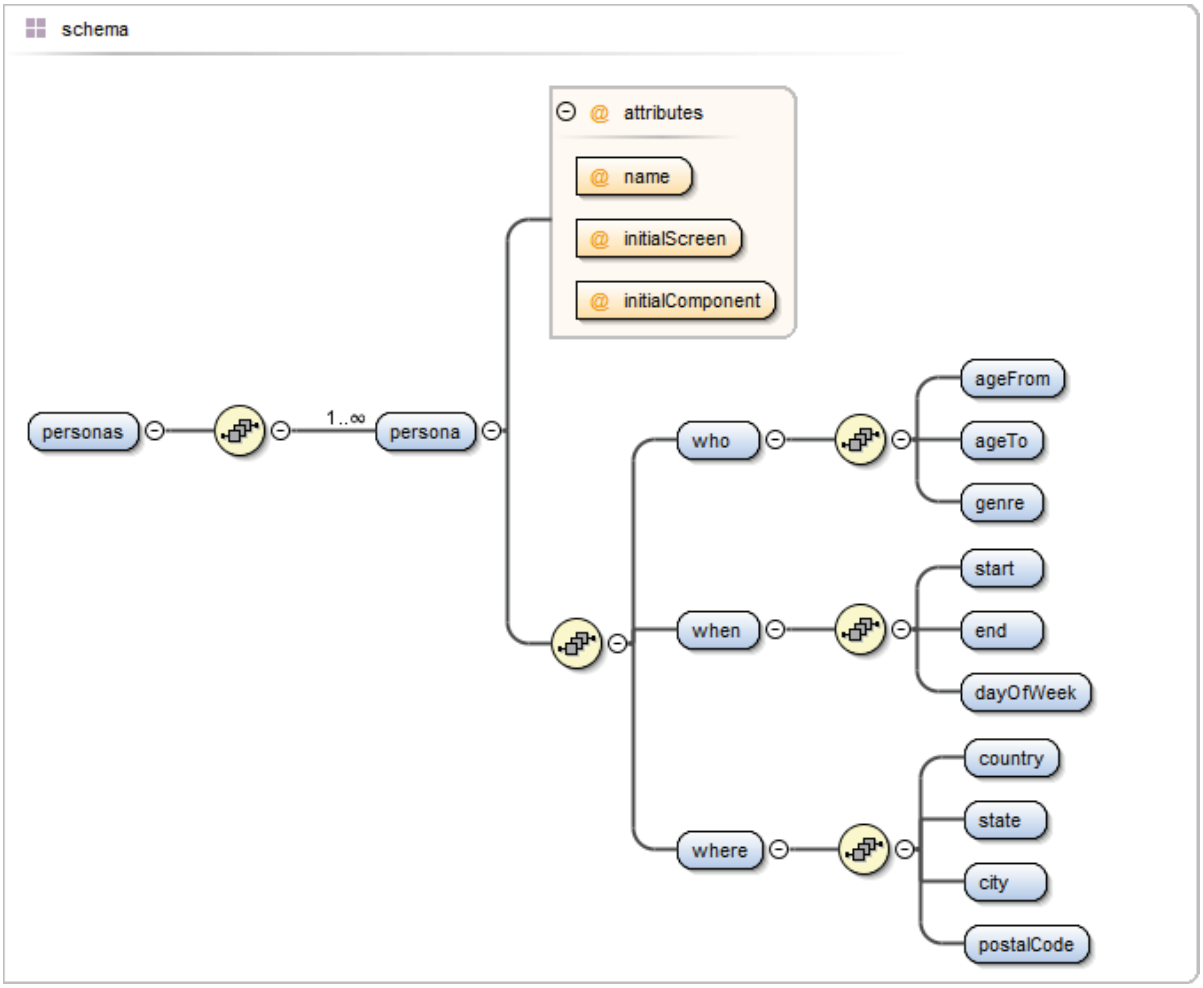


Figura 3.26: Esquema XML para os *personas*.

Código 3.5: Arquivo Personas.ctx (Parte 1).

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <personas>
3      <persona initialComponent="n_noticias" initialScreen="Noticias.tvs"
4          name="Maria">
5          <who>
6              <ageFrom>20</ageFrom>
7              <ageTo>-1</ageTo>
8              <genre>1</genre>
9          </who>
10         <when>
11             <start>01/06/2010 08:00</start>
12             <end>30/06/2010 18:00</end>
13             <dayOfWeek/>
14         </when>
15         <where>
16             <country>Brasil</country>
17             <state>PE - Pernambuco</state>
18             <city>Recife</city>
```

Código 3.6: Arquivo Personas.ctx (Parte 2).

```
19              <postalCode>50.610-220</postalCode>
20          </where>
21      </persona>
22  (...)
```

No último tipo de arquivo XML, são armazenados as transições. Cada uma delas possui um identificador (atributo *id*) e os elementos: nome do *persona* (*personaName*), nome da tela de origem (*fromScreen*), nome da tela de destino (*toScreen*), nome do componente de origem (*fromComponent*), nome do componente de destino (*toComponent*) e a ação que dispara a transição (*action*). O esquema deste XML pode ser visualizado na Figura 3.27, e um exemplo de parte do arquivo das transições do projeto *Canal*, no Código 3.7.

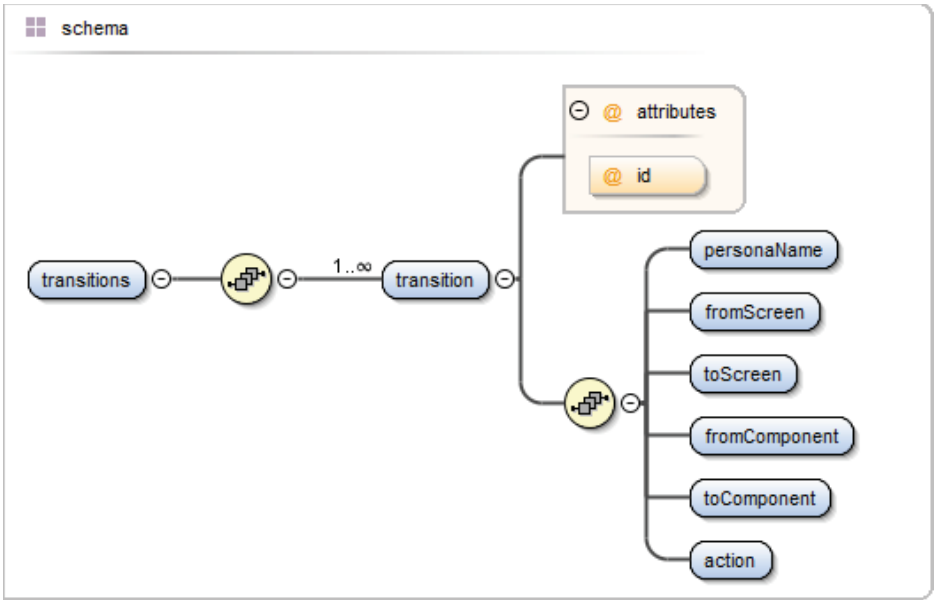


Figura 3.27: Esquema XML para as transições.

Código 3.7: Arquivo Transitions.ctx.

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <transitions>
3      <transition id="1">
4          <personaName>Default</personaName>
5          <fromScreen>Menu.tvs</fromScreen>
6          <toScreen>Menu.tvs</toScreen>
7          <fromComponent>noticias</fromComponent>
8          <toComponent>esportes</toComponent>
9          <action>CURSOR_DOWN</action>
10     </transition>
11     <transition id="2">
```

As 8050 linhas de código da aplicação estão distribuídas em 48 classes: 986 linhas em 12 classes de entidades; 5974 linhas em 28 classes de interface gráfica; 477 linhas em 4 classes de leitura e escrita dos arquivos salvos; e 613 linhas em 4 classes de geração de código.

3.5 Geração de Código

A geração de código ocorre com base no conteúdo dos arquivos XML do projeto, descritos na seção anterior. Na geração, são criados os arquivos abaixo, que fazem uso da *Biblioteca Contextual Ginga Lua* (Apêndice C), desenvolvida neste trabalho. Exemplos completos de códigos, gerados pela aplicação, estão presentes no Apêndice E.

- Um arquivo NCL;
- Um arquivo Lua principal;
- Para cada arquivo de tela do projeto, um arquivo Lua.

O arquivo NCL (trecho no Código 3.8) é responsável em sua essência por executar o vídeo principal (linha 1), especificado no arquivo das características do projeto, e por invocar a execução do arquivo Lua principal como uma mídia (linha 2).

Código 3.8: Exemplo de código NCL principal (Canal.ncl).

```
1 <media id="video" src="media/video.mp4" descriptor="dsVideo"/>
2 <media id="lua" src="Main.lua" descriptor="dsLua"/>
```

No arquivo Lua principal (trecho nos Códigos 3.9 e 3.10), chamado pelo código NCL acima, são coletadas a data e a hora do sistema (linha 1) e as informações do usuário que está interagindo com a televisão (da linha 6 à linha 26, na função *getUserContext*). As informações do usuário são coletadas a partir do arquivo *Context.properties*, que possui a idade (*age*), o sexo (*genre*), o país (*country*), o estado (*state*), a cidade (*city*) e o código postal (*postalCode*) do usuário.

Código 3.9: Coleta de informações contextuais no Lua principal (Main.lua, Parte 1).

```
1 currentTime = os.date('*t');
2 ...
3 userContext = {age = '', genre='',
4         country = '', city = '', state = '', postalCode = ''};
```

Código 3.10: Coleta de informações contextuais no Lua principal (Main.lua, Parte 2).

```
5
6  function getUserContext ()
7          for line in io.lines("Context.properties") do
8                  property = string.sub(line, 1, string.find(line, '=')-1);
9                  value = string.sub(line, string.find(line, '=')+1,
10                                 string.find(line, ';')-1);
11                 if property == 'age' then
12                         userContext.age = tonumber(value);
13                 elseif property == 'genre' then
14                         userContext.genre = value;
15                 elseif property == 'country' then
16                         userContext.country = value;
17                 elseif property == 'state' then
18                         userContext.state = value;
19                 elseif property == 'city' then
20                         userContext.city = value;
21                 elseif property == 'postalCode' then
22                         userContext.postalCode = value;
23                 end
24         end
25 end
26 getUserContext();
```

Foi escolhido capturar as informações contextuais dessa maneira devido à Gerência de Contexto ser um ponto em aberto na implementação de referência do Ginga-NCL (C++) (PORTAL, 2009). Esta gerência encontra-se implementada apenas no Ginga Live CD.

Ainda no arquivo Lua principal (trecho no Código 3.11), com as informações contextuais coletadas, pode-se determinar a qual *persona* o usuário deve ser associado (da linha 6 à linha 30, na função *getPersona*). Dependendo do *persona*, são exibidos a tela inicial e o primeiro componente a receber foco. As linhas 16 e 26 invocam as funções de desenho da tela de acordo com o *persona*.

Se o usuário não se encaixar em nenhum *persona* e existir um *Default*, serão exibidos a tela inicial e o primeiro componente a receber foco deste *persona Default*. Se este *persona* não existir, a aplicação não será exibida. Pois, a aplicação não tem como escolher atualmente um ponto inicial de um *persona* de forma aleatória, já que ela não tem elementos para decidir o *persona* correto.

As regras contextuais não são armazenadas em estruturas especiais. Elas estão diretamente implementadas em código através de comandos condicionais. Sendo um ponto de melhoria para a representação do contexto.

Código 3.11: Definição do *persona* no Lua principal (Main.lua).

```
1  persona = 'Default';
2  screen = 'Menu.tvs';
3  componentWithFocus = 'noticias';
4  personaSetted = false;
5
6  function getPersona ()
7          if persona_evaluateWho(15, -1, 'm')
8          and persona_evaluateWhen('01/06/2010', '08:00',
9                  '30/06/2010', '18:00', nil, currentTime)
10         and persona_evaluateWhere('Brasil', 'PE - Pernambuco',
11                 'Recife', '50.610-220') then
12
13                 persona = 'João';
14                 screen = 'Esportes.tvs';
15                 componentWithFocus = 'e_esportes';
16                 drawScreen_Esportes();
17                 personaSetted = true;
18
19         elseif persona_evaluateWho(20, -1, 'f')
20         and persona_evaluateWhen(nil, nil, nil, nil, nil, currentTime)
21         and persona_evaluateWhere(nil, nil, nil, nil) then
22
23                 persona = 'Maria';
24                 screen = 'Noticias.tvs';
25                 componentWithFocus = 'n_noticias';
26                 drawScreen_Noticias();
27                 personaSetted = true;
28         end
29 end
30 getPersona();
31
32 if personaSetted == false and persona == 'Default' and screen ~= '' then
33         drawScreen_Menu();
34 end
```

A avaliação do *persona* envolve uma função para cada categoria contextual: *persona_evaluateWho* (linhas 7 e 19 no Código 3.11), *persona_evaluateWhen* (linhas 8 e 20) e *persona_evalute Where* (linhas 9 e 21). Estas funções estão presentes na *Biblioteca Contextual Ginga Lua* (Apêndice C).

Os arquivos Lua relacionados a cada tela estão organizados em três partes cada um: a inicialização dos componentes a serem exibidos (Códigos 3.12); a função que invoca o desenho dos componentes, estando um destes com foco (Códigos 3.13 e 3.14); e as funções de manipulação de eventos (Códigos 3.15 e 3.16). Para cada transição, é gerada uma função que trata eventos.

Os componentes são inicializados de acordo com o seu tipo (Código 3.12, da linha 3 à linha 19). Se for um texto, será criada uma instância de *Text* (da linha 8 à linha 14).

Se for uma imagem, será criada uma instância de *Image* (da linha 3 à linha 6 e da linha 16 à linha 19). *Text* e *Image* são representações de um objeto presentes na *Biblioteca Contextual Ginga Lua*.

Código 3.12: Inicialização dos componentes da tela (Noticias.lua, Parte 1).

```
1  -- Components.
2
3  n_noticias = Image:new(
4          {name = 'n_noticias', x = 30, y = 45,
5          location = 'media/Noticias.png',
6          locationOnFocus = 'media/NoticiasSelecionado.png'});
7
8  texto = Text:new(
9          {text = 'Text Noticias',
10         font = Font:new({name = 'vera', style = 'nil', size = 48,
11                 color = Color:new({r = 255, g = 255, b = 0, alpha = 255})}),
12         x = 105, y = 44,
13         width = 500, height = 300,
14         background = Color:new({r = 0, g = 204, b = 204, alpha = 255})});
15
16 n_esportes = Image:new(
17         {name = 'n_esportes', x = 30, y = 165,
18         location = 'media/Esportes.png',
19         locationOnFocus = 'media/EsportesSelecionado.png'});
```

A função que realiza o desenho dos componentes da tela é *drawScreen_<nomeDaTela>* (Códigos 3.13 e 3.14). Se o componente a ser desenhado for do tipo texto, é chamada diretamente a função *draw* (linha 11) presente na Biblioteca. Se o componente for do tipo imagem, é preciso avaliar se o componente deve ser desenhado com ou sem foco. Se o componente for um vídeo, ele é invocado no arquivo NCL. Depois da avaliação, a função *draw* é invocada (linhas 5 e 13), passando-se o booleano *true*, caso seja com foco (linhas 6 e 14), ou *false*, caso seja sem foco (linhas 8 e 16).

Código 3.13: Desenho dos componentes da tela (Noticias.lua).

```
1  -- Draw Screen.
2
3  function drawScreen_Noticias()
4          if componentWithFocus ~= nil then
5                  if n_noticias.name == componentWithFocus then
6                          n_noticias:draw(true);
7                  else
8                          n_noticias:draw(false);
9                  end
10
11                 texto:draw();
12
13                 if n_esportes.name == componentWithFocus then
```

Código 3.14: Desenho dos componentes da tela (Noticias.lua).

```
14                          n_esportes:draw(true);
15                  else
16                          n_esportes:draw(false);
17                  end
18          end
19          canvas:flush();
20  end
```

Para cada transição, é gerado código para manipulação de um evento (Códigos 3.15 e 3.16). O nome da função de manipulação segue o padrão *handler_<numero>*, onde o número é o identificador da transição gerado pelo *Contextual Ginga*. Dependendo do *persona* (linha 6) identificado para o usuário que está interagindo com a televisão, da tela que está sendo exibida (linha 7), do componente que está com foco no momento (linha 8) e da tecla pressionada pelo usuário (linha 9), a transição é realizada para uma nova tela (linha 14) e para outro componente a receber foco (linha 15).

Cada transição realizada pelo usuário é armazenada num arquivo para posterior análise do fluxo de tarefas. Este registro é realizado através das funções *log_registerTransition1* (linhas 12 e 13) e *log_registerTransition2* (linhas 17 e 18) presentes na Biblioteca. A primeira função registra o identificador da transição, o *persona*, a tela e o componente de origem. A segunda registra a tela e o componente de destino e a ação que disparou a transição. Antes do desenho da nova tela, é chamada a função *cleanScreen* (linha 19) para limpar a tela atual. Esta função também se encontra na Biblioteca.

Código 3.15: Tratamento das transições/eventos (Noticias.lua).

```
1   -- Events.
2
3   function handler_5(evt)
4           if evt.class == 'key' and evt.type == 'press' then
5
6                   if persona == 'Default' and
7                   screen == 'Noticias.tvs' and
8                   componentWithFocus == 'n_noticias' and
9                   evt.key == 'CURSOR_DOWN' then
10
11                          transitionId = 5;
12                          log_registerTransition1(file, transitionId, persona,
13                                  screen, componentWithFocus);
14                          screen = 'Esportes.tvs';
15                          componentWithFocus = 'e_esportes';
16
17                          log_registerTransition2(file, screen,
18                                  componentWithFocus, evt.key);
19                          cleanScreen(0, 0, 640, 480);
```

Código 3.16: Tratamento das transições/eventos (Noticias.lua).

```
20                              drawScreen_Esportes();
21                      end
22              end
23      end
24      event.register(handler_5);
```

O código Lua específico da aplicação é gerado pela ferramenta. O código independente da aplicação foi separado em uma biblioteca que implementa funcionalidades comuns às aplicações geradas pela ferramenta *Contextual Ginga*. A Biblioteca simplifica o código das aplicações através de reuso de representações de objetos e funções em Lua.

Como visto nesta seção, as aplicações geradas invocam a Biblioteca para a funcionalidade de desenho dos componentes gráficos e funções auxiliares para o tratamento do contexto do usuário, registro das ações tomadas pelo usuário e gerenciamento dos *scripts* Lua. Por isto, os arquivos da biblioteca estão divididos em dois grupos: um para tratamento de interface gráfica, e outro para funções auxiliares.

Resumindo graficamente todos os arquivos envolvidos na produção de uma aplicação de TV digital, através da ferramenta *Contextual Ginga*, tem-se a Figura 3.28. Nela, os arquivos específicos da ferramenta (*<NomeProjeto>.tvp*, *<NomeTela>.tvs*, *Personas.ctx* e *Transitions.ctx*) são utilizados na geração de código para escrever automaticamente os arquivos *<NomeProjeto>.ncl*, *Main.lua* e *<NomeTela.lua>*. A esses arquivos gerados, são adicionados o arquivo com as informações contextuais do usuário, a *Biblioteca Contextual Ginga Lua* e as mídias (imagens e vídeos dos componentes das telas).

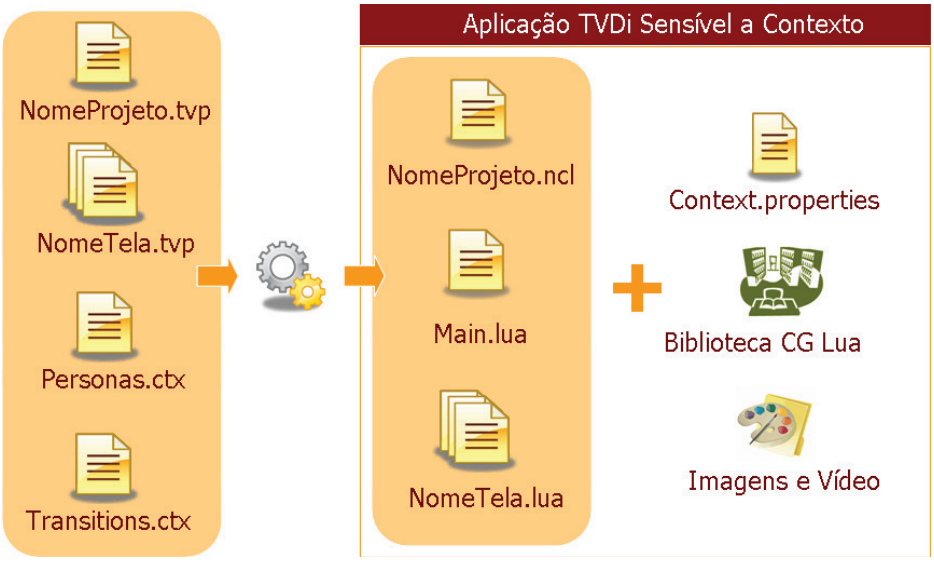


Figura 3.28: Arquivos para aplicação.

3.6 Comparação com Trabalhos Relacionados

A Figura 3.29 destaca as principais diferenças entre o *Contextual Ginga* e as ferramentas de autoria descritas no Capítulo 2, Seção 2.5, através das seguintes características: se é voltada para a plataforma MHP, se é voltada para a plataforma Ginga-J; se é voltada para a plataforma Ginga-NCL; se permite edição visual; se permite edição textual; se é uma ferramenta que gera aplicações sensíveis a contexto; se a execução da ferramenta é *desktop*; e se a execução da ferramenta é *web*. Um "x" identifica se a característica está presente na ferramenta.

Analisando a Figura 3.29, pode-se concluir que a maioria das ferramentas é para ambiente *desktop*, com exceção do iTV Project (OLIVEIRA; FILHO; SILVA, 2008). Além disso, a maioria também é voltada para a plataforma MHP: Icareus iTV Suite Author (ICAREUS, 2009), InteracTV (ALMEIDA; SOUZA; NETO, 2007), iTV Project (OLIVEIRA; FILHO; SILVA, 2008), JAME Author (IMK, 2005) e SCO Creator Tool (REY-LÓPEZ et al., 2008).

Apenas o *plug-in* NCL Eclipse (LAWS, 2008) não oferece edição visual. A maioria oferece este tipo de edição com o propósito de diminuir ou até mesmo eliminar o trabalho de codificação. Além disso, apenas o SCO Creator Tool (REY-LÓPEZ et al., 2008) permite a inserção de regras de contexto, mas ela é específica para aplicações de *t-learning* de TV digital e voltada para o MHP. Das ferramentas encontradas, o NCL Eclipse (LAWS, 2008) e o Ginga-NCL Composer (GUIMARÃES, 2007) são voltados para Ginga-NCL, mas não existe tratamento de contexto nas aplicações produzidas.

Característica / Ferramenta	MHP	Ginga-J	Ginga-NCL	Ed. Visual	Ed. Textual	Aplicação SC	Desktop	Web
Ginga-NCL Composer			X	X	X		X	
Icareus iTV Suite Author	X	X		X			X	
InteracTV	X			X			X	
iTV Project	X	X		X	X			X
JAME Author	X			X			X	
NCL Eclipse			X		X		X	
SCO Creator Tool	X			X		X	X	
Contextual Ginga			X	X		X	X	

Figura 3.29: Comparação com trabalhos relacionados.

3.7 Considerações Finais

Neste capítulo foi apresentada a ferramenta de autoria *Contextual Ginga*, desenvolvida nesta pesquisa. Na introdução foi explicado como os elementos gráficos que constituirão a aplicação de TVDi devem ser organizados num projeto da ferramenta. Além disso, foram descritos quais elementos contextuais podem ser adicionados nas aplicações. Em seguida, foram descritas as funcionalidades presentes na ferramenta e, em paralelo, foram sendo exibidas imagens de um exemplo de projeto.

Na seqüência, foi detalhada a interface gráfica da ferramenta e como se encontra estruturada a sua arquitetura. Depois, foram fornecidos detalhes de implementação do *Contextual Ginga*, focando em como as informações do projeto são armazenadas em arquivos específicos. Continuando, foi explicado como a geração automática de código ocorre e detalhes dos arquivos gerados. Para concluir este capítulo, foi realizada uma comparação do *Contextual Ginga* com as ferramentas de autoria presentes no Capítulo 2, Seção 2.5.

4 Experimento

Neste capítulo, serão apresentados os resultados obtidos através das duas fases do experimento realizado para a avaliação do *Contextual Ginga*. Além disso, será apresentada a análise destes resultados. Em ambas, a versão do Ginga-NCL Virtual STB utilizada foi a versão 0.11.2 revisão 23.

Os participantes de ambas as fases foram divididos em grupos. Cada grupo será referenciado através de um número: de 1 a 7 para os alunos da graduação, na primeira fase; e de 1 a 2 para os alunos da pós-graduação, na segunda fase. Dentro de cada grupo, cada participante será referenciado através da letra "D", de desenvolvedor da aplicação de TV digital, seguida de um número seqüencial. Todos os grupos da primeira fase possuíram 4 participantes, com exceção do grupo 6 que possuiu 3. Os participantes da segunda fase foram agrupados em duas duplas.

4.1 Planejamento do Experimento

O experimento foi planejado para ser realizado em duas fases diferentes em abril do período letivo 2010.1 com: alunos da disciplina IF745 - Sistemas Multimídia Distribuídos (TV Digital) das graduações em Ciência da Computação e Engenharia da Computação; e alunos de pós-graduação a nível de Mestrado em Ciência da Computação, com temas referentes à TV digital. Todos os alunos são do Centro de Informática da Universidade Federal de Pernambuco.

Os participantes da graduação foram selecionados para a primeira fase por: (1) possuírem conhecimento sobre TV digital e sensibilidade a contexto, recebidos na introdução da disciplina; (2) à princípio, pelo menos a maioria deles não possuir conhecimento sobre as linguagens NCL e Lua; e (3) estarem de fácil acesso a esta pesquisa, uma vez que o docente responsável pela disciplina é o orientador deste trabalho.

Os participantes da pós-graduação foram selecionados para a segunda fase por: (1)

possuírem conhecimento mais aprofundado sobre TV digital, incluindo experiências em desenvolvimento de aplicações; e (2) estarem também de fácil acesso à esta pesquisa uma vez que o docente orientador do trabalho deles é o mesmo deste trabalho.

O início do experimento contou com uma contextualização deste trabalho através da exposição do problema e da hipótese que seria verificada. Em seguida, foi apresentada a ferramenta proposta, através de suas características principais, e como a sensibilidade a contexto pode ser utilizada. Continuando, foi explicado como se relacionam as entidades disponíveis na ferramenta e foram demonstradas as funcionalidades através de vídeos tutoriais. Finalizando esta introdução, foi fornecida uma breve descrição de como ocorre a geração de código.

Depois destes esclarecimentos, foi fornecido um cenário da aplicação que deveria ser construída através da ferramenta, bem como os protótipos das telas e as transições que deveriam existir entre elas. No cenário da aplicação, estavam presentes indicações sobre a variabilidade dos fluxos de telas de acordo com informações contextuais do usuário da televisão.

Para participar do experimento, foi necessário que se possuísse conhecimento sobre TV digital e sensibilidade a contexto. Os participantes realizaram o experimento conforme os cinco passos a seguir, para os quais foram registrados os tempos utilizados em cada um deles. Apenas o passo 1 e a execução do código não são realizados através do *Contextual Ginga*.

- **Passo 1**: Com base no cenário da aplicação, reconhecer quais elementos fornecem sensibilidade a contexto para a aplicação;
- **Passo 2**: Criar o projeto, as telas da aplicação e os seus componentes;
- **Passo 3**: Criar os *personas* e inserir as suas características contextuais;
- **Passo 4**: Definir qual o ponto inicial (tela e componente iniciais) de cada *persona* e informar suas transições;
- **Passo 5**: Gerar o código da aplicação e executá-la no Ginga-NCL Virtual STB.

Além da aula expositiva como instrumento desta pesquisa, para auxiliar a coleta de dados no experimento, foi aplicado um questionário (Apêndice B) composto por três partes: a primeira para obter informações sobre os participantes e suas experiências passadas; a segunda sobre a ferramenta proposta por este trabalho; e a terceira sobre

a aplicação gerada pela ferramenta. Este questionário serviu principalmente para a verificação da hipótese e para a identificação de potenciais trabalhos futuros relevantes.

Em função dos passos, o questionário foi aplicado em dois momentos: a parte 1 antes da realização do primeiro passo; e as partes 2 e 3 depois da utilização da ferramenta e execução da aplicação gerada.

4.2 Fase 1

Nesta seção, serão apresentadas as informações fornecidas para que os alunos construíssem a aplicação para a fase 1 deste experimento, seguindo os passos definidos na Seção 4.1. Depois disso, serão apresentados os resultados, juntamente com a análise deles.

Esta fase do experimento foi realizada com 27 alunos ao todo, distribuídos em 6 grupos de 4 alunos e 1 grupo de 3 alunos. O início do experimento, ou seja, as exposições iniciais, aconteceu no dia 13/04/2010 (Aula 1) das 16h às 18h. Por questões de tempo, os passos planejados de 1 a 3 foram realizados na aula do dia 15/04/2010 (Aula 2) das 14h às 16h, o passo 4 e o início do passo 5 foram realizados na aula do dia 20/04/2010 (Aula 3) das 16h às 18h, e o final do passo 5 foi realizado na aula do dia 22/04/2010 (Aula 4) das 14h às 16h, todas no laboratório G1 do Centro de Informática da UFPE.

Na Aula 1, foram formados 6 grupos. Na Aula 2 surgiram mais 2 grupos, contudo apenas o resultado coletado com um deles foi considerado nesta pesquisa porque esse grupo conseguiu executar os cinco passos na Aula 3 e na Aula 4 restantes, ou seja, conseguiu realizar todas as atividades necessárias para a verificação da hipótese dentro do período planejado. O grupo desconsiderado não atingiu o objetivo de realizar todos os passos porque não conseguiu acompanhar o andamento da execução da fase.

Para o passo 1, o cenário utilizado nesta fase foi:

"A aplicação de TV digital é um canal de informações sobre o Recife e que possui conteúdo de notícias, esportes, jogos e novelas. O conteúdo de notícias é mais acessado por homens com mais de 20 anos quando estão no centro da cidade. Quando eles estão em outros lugares, eles acessam mais o conteúdo de esportes. Já as mulheres com mais de 30 anos, preferem acessar conteúdo referente a novelas. Jovens do sexo masculino com idade entre 10 e 18 acessam mais informações sobre jogos atuais. Por fim, para os que preferem acessar conteúdo referente a novelas, na sexta-feira são exibidos os filmes em

estréia. Se o usuário não se encaixar em nenhum dos personas, deve ser exibida uma lista com os tipos de conteúdo para o usuário selecionar qual prefere visualizar. Independente do persona o usuário pode navegar entre os diferentes conteúdos."

Para o passo 2, ou seja, a criação das telas e dos componentes, foram passadas para os participantes as imagens de protótipos (Figura 4.1 e da Figura 4.2). O fundo representa o vídeo principal que seria exibido na execução da aplicação.

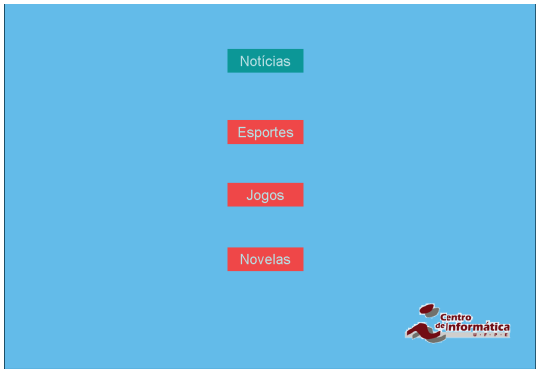


Figura 4.1: Protótipo do menu.

Figura 4.2: Protótipo da tela de notícias.

Com base nesses protótipos, os participantes deveriam criar seis telas: uma para o menu de seleção de conteúdo (Figura 4.1), uma para cada tipo de conteúdo (Figura 4.2 para o conteúdo de notícias), e mais uma para que na sexta-feira sejam exibidas imagens com propagandas de filmes.

Na Figura 4.3, pode ser visualizada a tela de notícias construída na ferramenta, na qual os componentes do tipo imagem, que são das opções de tela, estão todos iguais (com mesma cor de fundo), significando ausência de foco. A imagem com foco apenas é exibida em tempo de execução (Figura 4.4), quando é realizada uma transição para este componente ou ele é o componente inicial a receber foco. O vídeo exibido na Figura 4.4 vem como parte do exemplo de aplicação presente no Ginga-NCL Virtual STB (PORTAL, 2009).

Para o passo 3, considerando o cenário elaborado, deveriam ser identificados 6 *personas*, com os seus elementos contextuais divididos nas 3 categorias consideradas na ferramenta. Para simplificar esta fase, escolheu-se não utilizar os elementos de data e hora inicial e final. Pois, os usuários a cada vez que utilizassem o Ginga-NCL Virtual STB, eles precisariam também configurar a data e a hora da máquina virtual, inserindo mais um elemento de configuração que eles não conheciam. Nenhum elemento precisa ter valor especificado para o *persona Default*. Estas informações encontram-se na Tabela 4.1.

Para o passo 4, o ponto inicial de cada *persona* também pôde ser extraído a partir do

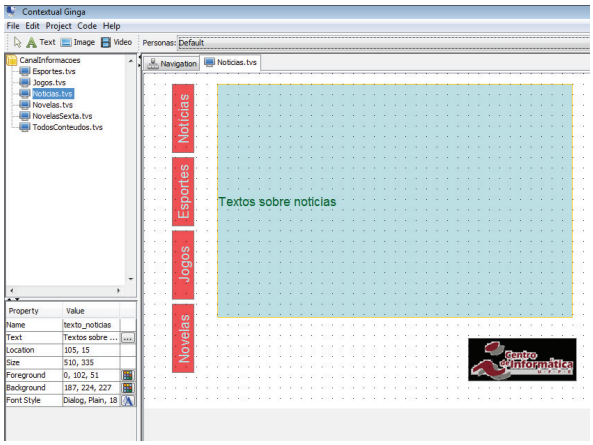


Figura 4.3: Tela de notícias na ferramenta.

Figura 4.4: Tela de notícias no STB.

Tabela 4.1: Identificação dos *personas* da fase 1.

Elemento Contextual	João	Pedro	Maria	José	Ana	*Default*
Idade Inicial	20	20	30	10	30	-
Idade Final	-	-	-	18	-	-
Sexo	M	M	F	M	F	-
Data/Hora Inicial	-	-	-	-	-	-
Data/Hora Final	-	-	-	-	-	-
Dia da Semana	-	-	-	-	Sexta-feira	-
País	Brasil	Brasil	Brasil	Brasil	Brasil	-
Estado	PE	PE	PE	PE	PE	-
Cidade	Recife	Recife	Recife	Recife	Recife	-
Código Postal	50.000-000	-	-	-	-	-

cenário. Exemplificando, se o usuário se encaixar no *persona* João, que prefere assistir conteúdo de notícias em geral, então a tela de notícias deve ser a primeira a ser exibida e o componente de notícias deve estar com foco. Na Tabela 4.2, estão as telas e os componentes iniciais de cada *persona*.

Tabela 4.2: Telas e componentes iniciais dos *personas* da fase 1.

Persona	Tela Inicial	Componente Inicial
João	Notícias	Notícias
Pedro	Esportes	Esportes
Maria	Novelas	Novelas
José	Jogos	Jogos
Ana	Novelas Sexta	Novelas
Default	Menu	Notícias

Para a inserção das transições, deveria ser considerado que da tela com o menu principal seria possível navegar para todas as outras telas, apertando-se o botão Enter com

o componente do conteúdo selecionado. As transições diretas entre as telas de conteúdo poderiam ser realizadas através das setas de navegação para cima e para baixo. Da tela de notícias, se o usuário apertasse o botão de seta para esquerda, retornaria para o menu principal (Figura 4.1).

Para este cenário, todos os *personas* possuem fluxos de navegação semelhantes, alterando-se a tela e o componente iniciais e, no caso da Ana, se for sexta-feira, a tela padrão de novelas que não é exibida. É exibida, então, sua variante com as imagens de propagandas de filmes. Na Figura 4.5, pode-se ver o fluxo de navegação para os *personas*, com exceção do fluxo do *persona* Ana.

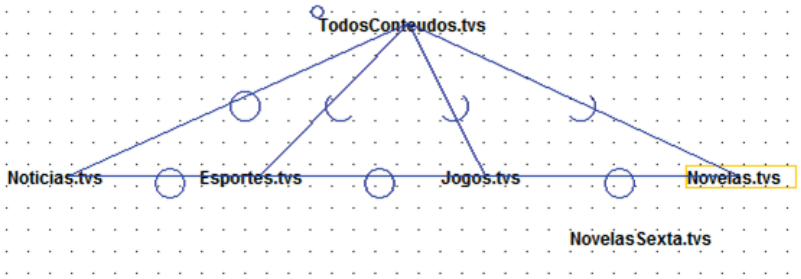


Figura 4.5: Navegação entre as telas.

Para o passo 5, procedeu-se com a geração de código e a utilização do Ginga-NCL Virtual STB, versão 0.11.2, revisão 23 (PORTAL, 2009). O código foi enviado para a máquina virtual através do WinSCP e executado por uma aplicação de console remoto.

4.2.1 Informações dos Participantes

Nesta subseção serão apresentadas as respostas para as questões da parte 1 do questionário, referente às informações dos participantes.

Questão 1: Vocês já utilizaram aplicações de TV digital? Se sim, quais? (Detalhar se a aplicação era sensível a contexto.)

- Grupo 1: todos os participantes já utilizaram pelo menos uma aplicação de TV digital. As aplicações foram:

  1. "Guia de Programação", por todos os participantes;
  2. "Aplicação interativa da Sky (Multishow, Globo News)", pelos participantes D1 e D2;
  3. "Jogos", também pelos participantes D1 e D2.

- Grupo 2: dois dos quatro participantes já utilizaram uma aplicação de TV digital:
  1. "Globo News Sky", pelo participante D2;
  2. "Grade de Programação Sky", pelos participantes D1 e D2;
  3. "Compra de Filmes Sky", também pelos participantes D1 e D2.
- Grupo 3: nenhum participante utilizou uma aplicação de TV digital.
- Grupo 4: todos os participantes já utilizaram a aplicação de TV digital "Sky TV" e o participante D1 utilizou também o "EPG".
- Grupo 5: dois dos quatro participantes já utilizaram uma aplicação de TV digital:
  1. "Compras de Filme em TV Fechada", pelos participantes D3 e D4;
  2. "Tabela do campeonato Brasileiro", apenas pelo participante D3;
  3. "Tabela de Programação", pelos participantes D3 e D4.
- Grupo 6: dois dos três participantes já utilizaram uma aplicação de TV digital:
  1. "EPG (Sky)", pelos participantes D1 e D2;
  2. "Interatividade Brasileirão (Sky)", apenas pelo participante D1;
  3. "Sinopse dos Programas (Sky)", pelos participantes D1 e D2.
- Grupo 7: apenas o participante D1 utilizou a aplicação de TV digital "Guia de Programação da Globo".

Em nenhuma das aplicações citadas pelos participantes, foi descrito que a aplicação era sensível a contexto.

Questão 2: Vocês já desenvolveram aplicações para TV digital? Se sim, quais aplicações, para qual *middleware* e quais ferramentas de desenvolvimento foram utilizadas?

Nenhum participante dos grupos já desenvolveu aplicação para TV digital.

Questão 3: Qual o tempo de experiência de cada participante nas linguagens NCL e Lua?

Apenas o participante D4 do grupo 1 disse que possuía 1 mês de experiência em Lua. Nenhum outro participante afirmou possuir experiência nestas linguagens.

Questão 4: Através de quais meios, vocês já tomaram conhecimento do conceito de sensibilidade a contexto?

- Grupo 1: todos os participantes deste grupo possuíam conhecimento sobre sensibilidade a contexto pelo menos através da "aula da disciplina de TV digital". Além disso, o participante D4 tomou conhecimento deste conceito também através de "Internet", "televisão" e "aula da disciplina Interface Usuário Máquina" da graduação;

- Grupo 2: todos os participantes deste grupo possuíam conhecimento sobre sensibilidade a contexto através de "sala de aula" e de "palestra/seminário de TV digital";

- Grupo 3: com exceção do participante D3, todos deste grupo possuíam conhecimento sobre sensibilidade a contexto através de "aula". O participante D3 tomou conhecimento do conceito através de "amigos";

- Grupo 4: todos os participantes deste grupo possuíam conhecimento sobre sensibilidade a contexto apenas através da "disciplina de TV digital";

- Grupo 5: todos os participantes deste grupo possuíam conhecimento sobre sensibilidade a contexto pelo menos através de "aula". Além disso, o participante D1 tomou conhecimento deste conceito também através de "seminário de TV digital";

- Grupo 6: dois dos três participantes deste grupo possuíam conhecimento sobre sensibilidade a contexto através da "disciplina de História e Futuro da Computação" da graduação;

- Grupo 7: todos os participantes deste grupo possuíam conhecimento sobre sensibilidade a contexto pelo menos através da "disciplina de TV digital". Além disso, o participante D1 tomou conhecimento deste conceito também através de pesquisas sobre "computação pervasiva para o seu trabalho de graduação".

Questão 5: Vocês já utilizaram quais aplicações sensíveis a contexto?

Os grupos 1, 4, 5 e 6 afirmaram não ter utilizado uma aplicação sensível a contexto. Os demais grupos responderam que:

- Grupo 2: todos os participantes deste grupo citaram ter utilizado como aplicação sensível a contexto: o "Gmail", que possui as "propagandas" baseadas no contexto do usuário; aplicações de "comércio eletrônico"; e o "Orkut", que possui a "sugestão de amigos" também baseada no contexto do usuário;

- Grupo 3: todos os participantes deste grupo já utilizaram "aplicações *Web* (*e-commerce*)";
- Grupo 7: todos os participantes deste grupo já utilizaram: o "Gmail", que possui propagandas baseadas no contexto do usuário; o "Spotify", que permite "fazer lista de músicas ou escutá-las de acordo com sugestões dadas pelo *site* ou de acordo com o seu perfil e amigos"; e o *site* "Amazon" de comércio eletrônico.

Análise das Respostas

Sobre os participantes da pesquisa, pode-se afirmar que dos 27 alunos, 15 (55%) já utilizaram pelo menos uma aplicação de TV digital, e que apenas de 1 grupo, nenhum aluno utilizou uma aplicação deste tipo anteriormente. Nenhum aluno desenvolveu alguma aplicação de TV digital e apenas 1 aluno (4%) disse ter experiência com a linguagem Lua, por 1 mês. Do total, apenas 1 aluno (4%) não tomou conhecimento do conceito de sensibilidade a contexto; mas, apenas 12 alunos (44%) afirmaram ter utilizado uma aplicação sensível a contexto. Dos sete grupos, apenas 3 (11%) disseram ter utilizado uma aplicação sensível a contexto.

4.2.2 Ferramenta *Contextual Ginga*

Nesta subseção serão apresentadas as respostas para as questões da parte 2 do questionário, referente à avaliação do *Contextual Ginga*. A análise das respostas desta fase será apresentada por questão.

O questionário aplicado nesta fase possuiu, inicialmente, as duas perguntas a seguir. Em função da semelhança das perguntas, as respostas dadas foram parecidas. Desta forma, a lista dos pontos positivos e das facilidades será reportada como uma única resposta.

Questão 1: Quais os pontos positivos da ferramenta?

Questão 2: Quais facilidades foram encontradas na utilização da ferramenta?

- Grupo 1:
  - Questão 1: "Abstrai o código";
  - Questão 1: "Boa usabilidade";
  - Questão 2: "Criação das telas utilizando *drag-and-drop*";

  - Questão 2: "Visualização do fluxo de transições".

- Grupo 2:
  - Questão 1: "Abstração de código";
  - Questão 1: "Facilidade de uso";
  - Questão 2: "Criação de telas simples".

- Grupo 3:
  - Questão 1: "Geração automática de código";
  - Questão 1: "Criação intuitiva de *personas*";
  - Questão 2: "Todo trabalho foi realizado através da GUI";
  - Questão 2: "Não necessidade de conhecimento da linguagem".

- Grupo 4:
  - Questão 1: "Fácil de utilizar";
  - Questão 1: "Geração automática do grafo das transições";
  - Questão 1: "Não requer conhecimentos de uma linguagem de programação";
  - Questão 2: "Criação de *personas*";
  - Questão 2: "Criação das transições";
  - Questão 2: "Geração do código".

- Grupo 5:
  - Questão 1: "A geração automática de código. Possibilidade de fazer uma aplicação sem escrever uma linha de código";
  - Questão 1: "Integração com sensibilidade a contexto com a fácil criação de *personas*";
  - Questão 2: "Definição de *personas*";
  - Questão 2: "Facilidade de inserir mídias, porém dificuldade de manipulação das mesmas";
  - Questão 2: "Fácil geração de código. Intuitiva".

- Grupo 6:

  - Questão 1: "É um sistema visual";
  - Questão 1: "É de simples manuseio";
  - Questão 2: "A simplicidade".

- Grupo 7:
  - Questão 1: "Simplicidade e objetividade (fácil utilização)";
  - Questão 1: "Aplicação Leve";
  - Questão 2: "Abstração do código".

Análise das Respostas para as Questões 1 e 2

Dos sete grupos, sete (100%) listaram como ponto positivo ou facilidade a geração automática de código, que trouxe a abstração do código para os desenvolvedores e possibilitou que não fossem exigidos conhecimentos de uma linguagem de programação, sendo a aplicação construída de forma visual; cinco (71%) consideraram também como ponto positivo a facilidade de uso; três (43%) trouxeram como ponto positivo que a criação de *personas* era fácil ou intuitiva, permitindo a integração com sensibilidade a contexto; e dois (29%) listaram a visualização do fluxo de transições.

Um resumo destes benefícios trazidos pela ferramenta *Contextual Ginga* pode ser visualizado no gráfico da Figura 4.6. Outros pontos positivos ou facilidades destacados foram: criação das telas utilizando *drag-and-drop*; criação das transições; facilidade de inserir mídias, porém dificuldade de manipulação das mesmas; e o fato da aplicação ser leve.

Questão 3: A ferramenta permitiu que fossem cometidos erros na construção da aplicação? Quais?

- Grupo 1:
  1. "Não foi possível visualizar o foco nos botões do menu";
  2. "Não foi direto para a tela do contexto do *persona*".

- Grupo 2:
  1. "Nomes de botões";
  2. "Telas com grafia não aceita".

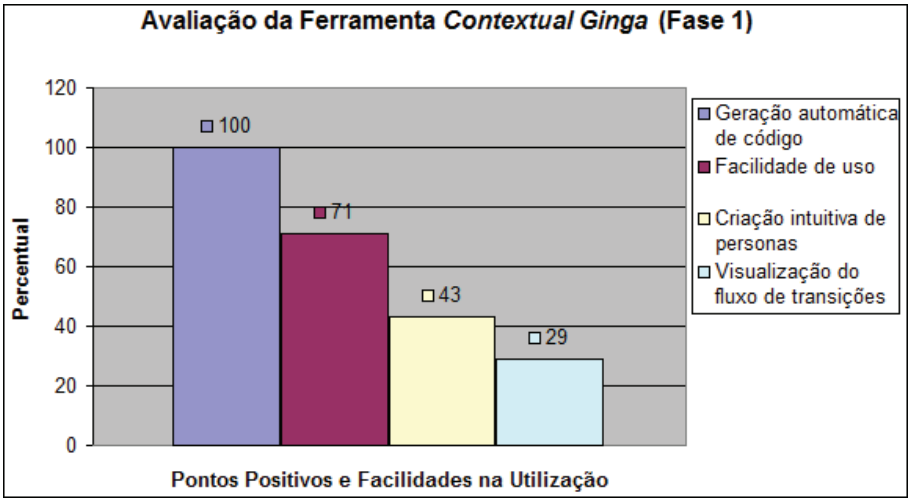


Figura 4.6: Respostas para as questões 1 e 2.

- Grupo 3: Inicialmente não conseguimos executar a aplicação.

Os grupos 4, 5, 6 e 7 responderam que a ferramenta não permitiu que fossem cometidos erros na construção da aplicação. O grupo 7 afirmou ainda que "os erros encontrados foram relacionados à interface".

Análise das Respostas para a Questão 3

O grupo 1 afirmou que não foi possível visualizar o foco nos botões da aplicação, mas não se pode analisar o problema uma vez que o código gerado não foi disponibilizado pelos alunos. O grupo 2 afirmou que a aplicação permitiu cometer erro no nome dos botões, mas foi avisado antes do início do experimento que não poderiam ser criados dois componentes com o mesmo nome, mesmo que em telas distintas. O grupo 2 afirmou ainda que existiram telas com grafia não aceita. Este problema ocorreu em função da fonte não suportar acento para os componentes do tipo texto. O grupo 3 afirmou que inicialmente não conseguiu executar a aplicação, mas não indicou a razão.

Questão 4: A adição de quais elementos contextuais poderia tornar as aplicações geradas mais interessantes?

Os grupos 1 e 2 não informaram qualquer elemento contextual. As respostas dos demais grupos estão informadas a seguir.

- Grupo 3: "Atividade paralela - o usuário pudesse informar uma atividade que estivesse executando em paralelo";
- Grupo 4: "Ocupação dos *personas*";

- Grupo 5: "Achamos que o código postal é muito específico para determinar o local. Sugerimos colocar Bairro";
- Grupo 6: "Padrões pré-definidos de tela";
- Grupo 7: "Log de telas acessados pelo usuário e aprendizagem de máquina".

Análise das Respostas para a Questão 4

O grupo 3 indicou que poderia ser utilizada como informação contextual uma atividade que o usuário estivesse executando em paralelo, mas a categoria *What* foi considerada apenas como a atividade de se interagir com a televisão, por isso não foi utilizada. O grupo 4 sugeriu que fosse utilizada a ocupação das pessoas, podendo ser adicionada na categoria *Who*. O grupo 5 sugeriu a utilização do bairro no lugar do código postal, mas este foi selecionado por ser um atributo do usuário no Ginga Live CD (PORTAL, 2009).

O grupo 6 sugeriu utilizar padrões pré-definidos de telas. Dessa forma, seria possível que fossem criados modelos de telas e que as telas, criadas a partir destes modelos, possuiriam os mesmos componentes presentes no modelo, mas com outros nomes. Portanto, o desenvolvedor da aplicação de TV digital precisaria apenas adicionar os componentes que não estivessem no modelo, sendo reduzido o esforço de construção de telas semelhantes. Apesar de ser uma contribuição a ser adicionada na ferramenta *Contextual Ginga*, esta resposta não caracteriza um elemento contextual para as aplicações geradas.

Por fim, o grupo 7 sugeriu que fossem registradas as telas acessadas pelos usuários, mas esta funcionalidade já existe através do registro das transições realizadas pelos usuários (diagrama de atividades) e que são salvas em um diretório dentro do qual os arquivos da aplicação se encontram. Sendo estas informações armazenadas localmente, seria necessário que elas fossem transmitidas, através de um canal de retorno, para que os produtores de conteúdo pudessem analisar o diagrama de atividades realizado, para melhoria da usabilidade e da interação em novas versões da aplicação de TV digital.

Como a data e a hora da transição realizada pelo usuário também estão presentes no arquivo de registro das interações, poderia-se calcular o tempo que o usuário interage com cada tela. Este tempo poderia enriquecer a sensibilidade a contexto para, por exemplo, se o usuário (o seu *persona* associado) passa pouco tempo na primeira tela apresentada, que a aplicação mudasse automaticamente para que a tela inicial fosse a que o *persona* passa mais tempo. Desta forma, a ferramenta *Contextual Ginga* poderia fazer uso de

aprendizagem de máquina para assim poder evoluir automaticamente os contextos dos usuários e se adaptar a estas mudanças, como também sugerido pelo grupo 7.

O questionário aplicado nesta fase possuiu as duas perguntas a seguir. Em função da semelhança das respostas, elas serão reportadas como uma única resposta.

Questão 5: Quais funcionalidades podem ser adicionadas na ferramenta? Questão 6: Quais funcionalidades podem ser melhoradas na ferramenta? E como?

O grupo 6 não respondeu estas perguntas. As demais respostas informadas foram:

- Grupo 1:
  - Questão 5: "Copiar e colar";
  - Questão 5: "Inserir imagem no tamanho original";
  - Questão 5: "Armazenar último caminho do diretório usado";
  - Questão 5: "As imagens poderiam receber uma posição (x ou y) única, para se alinharem";
  - Questão 6: "Criar telas: copiar e colar, pois as telas podem ser semelhantes".
- Grupo 2:
  - Questão 5: "Imagem ser adicionada no tamanho certo automaticamente";
  - Questão 5: "Função desfazer (Ctrl + Z)";
  - Questão 5: "Adicionar imagens arrastando-soltando";
  - Questão 5: "Botões de *Copy* e *Paste* não funcionam";
  - Questão 5: "Poder selecionar mais de um componente";
  - Questão 6: "Adicionar imagem: ser adicionada no tamanho da imagem";
  - Questão 6: "Poder deletar através do botão *Delete*";
  - Questão 6: "Ao criar um componente e adicionar o nome, teclar Enter, fazendo o mesmo efeito de clicar em *Finish*".
- Grupo 3:
  - Questão 5: "Realizar transições pelo mouse";
  - Questão 5: "Copiar e colar objetos";

  - Questão 5: "Selecionar mais de um objeto";
  - Questão 6: "Adição de transição entre os componentes: realizar a funcionalidade diretamente pelo mouse";
  - Questão 6: "Selecionar objetos: permitir a seleção de vários objetos para movimentos, copiar, excluir, etc";
  - Questão 6: "Botões de Adicionar e Remover *personas* ficam sumindo";
  - Questão 6: "Redimensionamento direto no componente".

- Grupo 4:
  - Questão 5: "Copiar e colar (componentes, telas, etc.)";
  - Questão 5: "Desfazer a última modificação";
  - Questão 5: "Alinhar os componentes à grade";
  - Questão 6: "Remoção de *personas*: o botão de remoção não está aparecendo";
  - Questão 6: "Criação de componentes: poder arrastar para definir o tamanho";
  - Questão 6: "Renomear componente: ao renomear componentes atualizar as transições".

- Grupo 5:
  - Questão 5: "Funcionalidade para facilitar a criação de telas";
  - Questão 5: "Possibilidade de criar uma tela padrão que contém *layout* comum. As outras telas podem estender dela";
  - Questão 6: "Criação das telas. Melhorar a usabilidade geral para facilitar e agilizar a criação das telas";
  - Questão 6: "Transição: a criação de transições poderia ser feita tanto nas telas (como já é), como no *Navigation*".
  - Questão 6: "Redimensionar as imagens diretamente na tela";
  - Questão 6: "Poderia ter uma opção que demonstrasse as transições das telas, sem necessariamente ter que rodar a aplicação no Ginga".

- Grupo 7:
  - Questão 5: "*Copy and paste*";
  - Questão 5: "Importar telas";

– Questão 5: “Ctrl + Z”;

– Questão 6: “Deletar componentes: não precisar reiniciar o programa”.

Análise das Respostas para as Questões 5 e 6

Sintetizando as respostas fornecidas nesta primeira fase do experimento, foram detectados os três problemas presentes na Tabela 4.3. Esta tabela também indica a quantidade de grupos que reportou cada problema.

Tabela 4.3: Problemas encontrados na fase 1 do experimento.

Problema	Qtd. de Grupos
P1 - Botões de adicionar e remover *personas* desaparecem, devido ao fato da aplicação ter sido feita para uma resolução mais alta do que a dos monitores em que o experimento foi realizado e que não poderia ser alterada devido a restrições administrativas do ambiente.	2
P2 - Ao renomear um componente, atualizar as transições para este novo nome. As transições estão sendo apagadas.	1
P3 - Em algumas situações, o botão de excluir componente não está funcionando.	1

E foram sugeridas doze melhorias para a ferramenta, presentes na Tabela 4.4.

Em paralelo à realização desta fase do experimento, foram realizadas duas correções de problemas encontrados na ferramenta e na biblioteca que poderiam impedir a execução da aplicação gerada. A primeira correção foi realizada na ferramenta para que quando um *persona* não possuísse um ponto inicial, que o código gerado não fizesse a avaliação dele. A segunda correção foi realizada para adicionar uma validação na biblioteca caso não fosse informada a idade do usuário que estava interagindo com a aplicação.

4.2.3 Aplicação Gerada

Nesta subseção serão apresentadas as respostas para as questões da parte 3 do questionário, referente à avaliação da aplicação gerada pelo *Contextual Ginga*.

Questão 1: Compare as semelhanças e diferenças entre a representação da tela na ferramenta e a exibida no Ginga-NCL Virtual STB.

As respostas informadas foram:

- Grupo 1: “Idêntica se não contar com o vídeo”;

Tabela 4.4: Melhorias sugeridas na fase 1 do experimento.

Melhoria	Qtd. de Grupos
M1 - Permitir copiar e colar telas e componentes.	6
M2 - Permitir o redimensionamento do componente através da representação gráfica da tela.	3
M3 - Permitir selecionar mais de um componente ao mesmo tempo para facilitar as operações de copiar, colar e excluir.	2
M4 - Inserir o componente do tipo imagem já na dimensão do arquivo da imagem.	2
M5 - Permitir desfazer últimas alterações.	2
M6 - Realizar transições através do mouse na aba *Navigation*.	2
M7 - Permitir o alinhamento dos componentes a uma determinada posição.	2
M8 - Quando for adicionar outra imagem, já abrir a janela de seleção de arquivo no último diretório utilizado.	1
M9 - Adicionar imagens arrastando e soltando.	1
M10 - Poder excluir um componente através do botão *Delete*.	1
M11 - Ao informar o nome de uma tela, componente ou *persona*, que a tecla Enter tenha o efeito do botão *Finish*.	1
M12 - Criar visualização das transições antes de rodar a aplicação no Ginga, ou seja, simular a aplicação rodando.	1

- Grupo 2: "Única diferença perceptível é a transparência ligeiramente maior na execução no Virtual STB (talvez devido ao fundo azul do vídeo)";
- Grupo 3: "Foram praticamente idênticas";
- Grupo 4: "O *persona* selecionado estava incorreto. O resultado final corresponde ao mostrado na ferramenta";
- Grupo 5: "Não vimos nenhuma diferença";
- Grupo 6: "As representações são semelhantes, porém alguns componentes da ferramenta não são exibidos no Virtual STB (por erro do operador/programador)";
- Grupo 7: "Foram bastante semelhantes".

Análise das Respostas para a Questão 1

Todos consideraram a representação da tela na ferramenta bastante semelhante a que foi exibida no Ginga-NCL Virtual STB (PORTAL, 2009). As diferenças citadas foram o vídeo, que não é exibido na ferramenta, e a transparência das imagens.

Questão 2: O grupo considera que conseguiu atingir o objetivo de produzir uma aplicação de TV digital sem precisar codificar e sem ter conhecimento das linguagens NCL e Lua? Por quê?

- Grupo 1: "Não. Porque não conseguimos utilizar a aplicação, ou seja, não conseguimos navegar nas telas";
- Grupo 2: "Sim. Não houve necessidade de criação de código para ter uma aplicação em mãos";
- Grupo 3: "Sim. Porém há muitos procedimentos a serem realizados antes de executar o projeto";
- Grupo 4: "Sim. Apesar dos problemas (relacionados aos *personas* e às transições). Foi possível executar a aplicação criada com as funcionalidades desejadas";
- Grupo 5: "Sim";
- Grupo 6: "Sim";
- Grupo 7: "Sim. Não foi possível estabelecer todas as transições".

Análise das Respostas para a Questão 2

Dos sete grupos, apenas um (15%) afirmou não ter conseguido atingir o objetivo de produzir uma aplicação de TV digital. Apenas o grupo 5 (15%) conseguiu com sucesso testar os *personas* e as transições entre as telas. Dos grupos que enviaram o código do projeto e da aplicação (grupos 2, 5, 6 e 7), foi possível constatar que, com exceção do 5, estavam faltando transições e imagens com foco. As transições haviam sido inseridas, contudo quando tiveram que renomear os componentes, porque existiam componentes em telas diferentes com o mesmo nome, a aplicação apagou as transições. No início da primeira aula no laboratório, foi avisado que não poderiam ser inseridos dois componentes com o mesmo nome.

Através da análise dos códigos disponibilizados pelos grupos 2, 5, 6 e 7, pode-se verificar porque alguns problemas ocorreram. O código do grupo 2 não tinha qualquer transição, nem qualquer componente com imagem de foco. O código do grupo 5 funcionou corretamente. O código do grupo 6 não estava com o vídeo informado, por isso que a aplicação apareceu com um fundo preto. Além disso, como nenhum componente teve um valor atribuído à propriedade "imagem com foco", na execução da aplicação nenhum

componente recebeu foco. O código do grupo 7 não funcionou com transições porque possuía apenas duas transições para um *persona* que não era o que o grupo estava tentando usar e, além disso, as imagens com foco não estavam informadas.

Como apenas 15% dos grupos desta fase conseguiu produzir e executar uma aplicação de TV digital sensível a contexto de acordo com a especificação, considerou-se que o objetivo da dissertação foi atingido parcialmente. Foi detectado, como obstáculo para atingir o objetivo nesta fase, o fato de haver muitos grupos realizando o experimento ao mesmo tempo. Seus participantes tinham dúvidas ou encontravam problemas que não podiam ser respondidos e corrigidos imediatamente, uma vez que havia apenas uma pessoa disponível para isto. Isto motivou a execução da fase 2, com menos grupos, para ser possível um andamento contínuo do experimento e para que erros em passos iniciais não prejudicassem a execução final da aplicação.

Tempos de Execução dos Passos

Os sete grupos gastaram em cada passo os tempos especificados na Tabela 4.5. Não se pode estabelecer qualquer comparação com estes números, uma vez que cada grupo conseguiu criar um número diferente de telas e *personas* com as transições necessárias.

Tabela 4.5: Tempos dos passos executados na fase 1.

Grupo	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5	Total
1	7min	41min	7min	1h 20min	15min	2h 30min
2	4min	33min	9min	34min	1h 27min	2h 47min
3	3min	38min	3min	27min	40min	1h 51min
4	2min	50min	3min	40min	1h 20min	2h 55min
5	4min	36min	13min	1h 30min	30min	2h 53min
6	1min	45min	5min	30min	10min	1h 31min
7	5min	50min	10min	5min	1h 30min	2h 40min

Com base nas Tabelas 4.3 e 4.4, os problemas de código a seguir foram corrigidos, bem como foram implementadas algumas das melhorias sugeridas. A escolha destas implementações se baseou principalmente nos problemas e na melhoria da usabilidade da ferramenta, considerando-se o tempo disponível para esta atividade. O problema P3 não foi corrigido porque não se conseguiu reproduzir o erro.

1. P1 - A ferramenta foi corrigida para que os botões de adicionar e excluir *persona* apareçam independentemente da resolução do monitor;

2. P2 - A ferramenta foi corrigida para que, ao se renomear um componente, as transições sejam mantidas para o componente que agora assume um novo nome;

3. M4 - A ferramenta foi melhorada para que, ao se atribuir um valor para a propriedade imagem, o componente seja redimensionado para o tamanho da imagem;

4. M8 - A ferramenta foi melhorada para que, ao adicionar outra imagem, a janela de seleção de arquivo seja aberta no último diretório utilizado;

5. M10 - A ferramenta foi melhorada para que se possa excluir um componente através do botão *Delete*;

6. M11 - A ferramenta foi melhorada para que, ao ser informado o nome de uma tela, componente ou *persona*, a tecla Enter tenha o efeito do botão *Finish*.

4.3 Fase 2

Depois das correções e melhorias realizadas na ferramenta *Contextual Ginga*, deu-se início à fase 2 do experimento. Esta fase completa aconteceu no dia 27/04/2010 das 9h 30min às 11h 30min no laboratório para alunos da pós-graduação.

O cenário aplicado na fase anterior foi simplificado com o objetivo de diminuir o tempo do experimento já que os alunos da pós-graduação não teriam o mesmo tempo disponível que os da graduação.

Para o passo 1, o novo cenário utilizado foi:

"*A aplicação de TV digital é um canal de informações sobre o Recife e que possui conteúdo de notícias em geral e esportes. O conteúdo de notícias é mais acessado por mulheres com mais de 20 anos quando estão no centro da cidade. Já o conteúdo de esportes é mais acessado por homens com mais de 15 anos. Se o usuário não se encaixar nestes perfis, é exibida uma lista com os tipos de conteúdo para o usuário selecionar qual prefere visualizar. Independente do seu perfil, o usuário pode navegar entre os diferentes conteúdos.*"

Para o passo 2, ou seja, a criação das telas e dos componentes, foram passadas para os participantes as imagens da Figura 4.7 e da Figura 4.8.

Com base nessas imagens, os participantes deveriam criar três telas: uma para o conteúdo de notícias (Figura 4.9), uma para o conteúdo de esportes (Figura 4.10) e outra para o menu de seleção de conteúdo (Figura 4.11).

Para o passo 3, considerando o cenário elaborado, deveriam ser identificados 3 *personas*, com os seus elementos contextuais divididos nas 3 categorias consideradas na

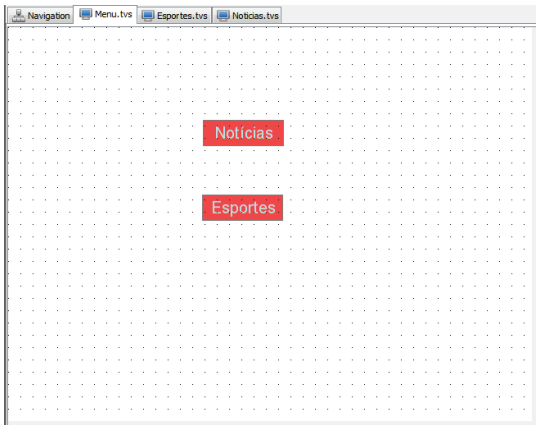


Figura 4.7: Tela do menu.

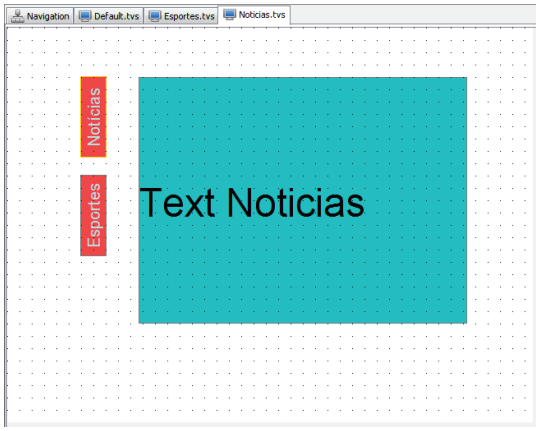


Figura 4.8: Tela de notícias.

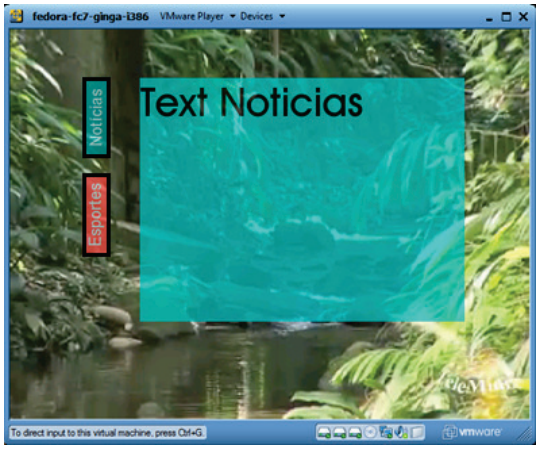


Figura 4.9: Tela de notícias.

Figura 4.10: Tela de esportes.

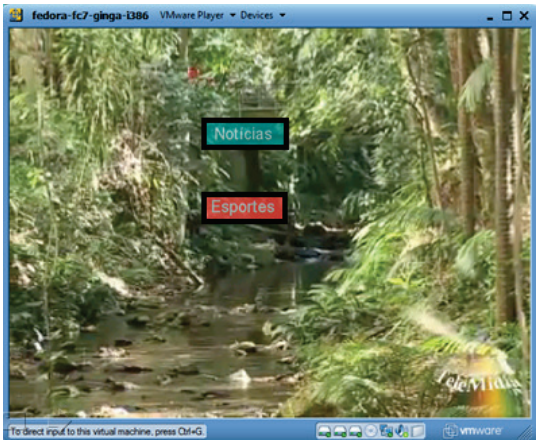


Figura 4.11: Tela do menu.

ferramenta. Estas informações encontram-se na Tabela 4.6.

Tabela 4.6: Identificação dos *personas* da fase 2.

Elemento Contextual	**Maria**	**João**	***Default***
Idade Inicial	20	15	-
Idade Final	-	-	-
Sexo	F	M	-
Data/Hora Inicial	-	-	-
Data/Hora Final	-	-	-
Dia da Semana	-	-	-
País	Brasil	Brasil	-
Estado	PE	PE	-
Cidade	Recife	Recife	-
Código Postal	50.000-000	-	-

Para o passo 4, o ponto inicial de cada *persona* também pode ser extraído a partir do cenário. Exemplificando, se o usuário se encaixar no *persona* Maria, que prefere assistir conteúdo de notícias em geral, então a tela de notícias deve ser a primeira a ser exibida e o componente de notícias deve estar com foco. Na Tabela 4.7, estão as telas e os componentes iniciais de cada *persona*.

Tabela 4.7: Telas e componentes iniciais dos *personas* da fase 2.

Persona	**Tela Inicial**	**Componente Inicial**
Maria	Notícias	Notícias
João	Esportes	Esportes
Default	Menu	Notícias

Para a inserção das transições, deveria ser considerado que da tela com o menu principal seria possível navegar para todas as outras telas, apertando-se o botão Enter no componente do conteúdo selecionado. As transições diretas entre as telas de conteúdo poderiam ser realizadas através das setas de navegação para cima e para baixo. Da tela de notícias, se o usuário apertasse o botão de seta para esquerda, a aplicação retornaria para o menu principal.

Para exemplificar as transições em execução, considerando que o usuário é representado através do *persona* João, da tela de esportes (Figura 4.10), apertando-se a seta para cima, a tela de notícias (Figura 4.9) é exibida. Da tela de notícias, apertando-se a seta para esquerda, a tela com o menu é exibida (Figura 4.11). Do menu, apertando-se Enter no componente de notícias, a tela de notícias é exibida novamente. Este fluxo pode ser visualizado na máquina de estados da Figura 4.12.

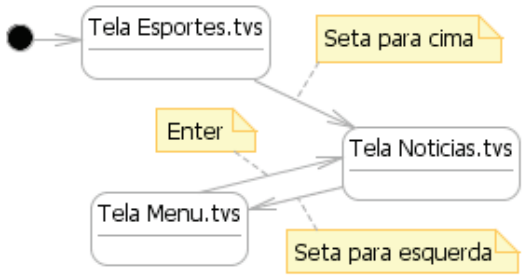


Figura 4.12: Máquina de estados com exemplo de transições.

Para o passo 5, procedeu-se de forma idêntica ao que foi descrito na fase anterior.

Para este cenário, todos os *personas* possuem os mesmos fluxos de navegação. Na Figura 4.13, é exibido o fluxo de navegação para os *personas*.

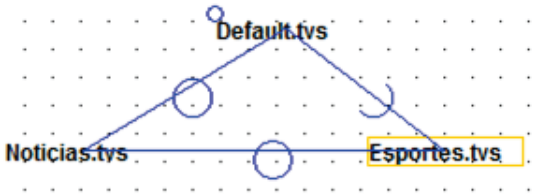


Figura 4.13: Navegação entre as telas.

4.3.1 Informações dos Participantes

Nesta subseção serão apresentadas as respostas para as questões da parte 1 do questionário, referente às informações dos participantes.

Questão 1: Vocês já utilizaram aplicações de TV digital? Se sim, quais? (Detalhar se a aplicação era sensível a contexto.)

Para esta questão, foi respondido que:

- Grupo 1:
  1. "Informações da Copa 2010 (com contexto)", pelo participante D2;
  2. "TBolão", pelos dois participantes;
  3. "Brasileirão 2008/9 (CESAR)", pelo participante D1;
  4. "Copa 2006 (CESAR)", pelo participante D1.
- Grupo 2:

1. "T-Cup (projeto da cadeira de contexto)", pelo participante D2;
2. "Campeonato Brasileiro 2009 (Sky, sem contexto)", pelo participante D1.

Questão 2: Vocês já desenvolveram aplicações para TV digital? Se sim, quais aplicações, para qual *middleware* e quais ferramentas de desenvolvimento foram utilizadas?

Para esta questão, foi respondido que:

- Grupo 1:
  1. "Informações da Copa 2010", pelo participante D2 para o *middleware* Ginga;
  2. "TBolão", pelos participante D1 para o *middleware* Ginga;
  3. "Brasileirão 2008/9 (CESAR)", pelo participante D1 para o *middleware* Open TV[1];
  4. "Visualizador de Propagandas", pelo participante D1 para o *middleware* MHP.
- Grupo 2:
  1. "T-Cup (projeto da cadeira de contexto)", pelo participante D2 para o *middleware* Ginga-NCL;
  2. "Playout para MHP", pelo participante D1 para o *middleware* MHP.

Do grupo 1, o participante D1 utilizou para o desenvolvimento das aplicações as ferramentas "TVision (Visual)" e "Context", que é uma ferramenta de edição textual. O participante D2 utilizou o "Eclipse", mas não especificou se era associado a algum *plug-in*.

Do grupo 2, o participante D1 utilizou as ferramentas "Eclipse" para desenvolvimento e "XletView" para a execução da aplicação. O participante D2 utilizou o "Composer" e o "Eclipse com *plug-in* para NCL".

Questão 3: Qual o tempo de experiência de cada participante nas linguagens NCL e Lua?

Do grupo 1, o participante D1 possui 1 ano e 5 meses de experiência com as linguagens NCL e Lua. Enquanto o participante D2 possui 6 meses também em ambas as linguagens. Do grupo 2, o participante D1 não possui experiência com as linguagens,

[1]O *middleware* Open TV permite a execução de aplicações de TV digital escritas na linguagem C (GROUP, 1994).

mas o participante D2 possui 6 meses com a linguagem NCL e 3 meses com a linguagem Lua.

Questão 4: Através de quais meios, vocês já tomaram conhecimento do conceito de sensibilidade a contexto?

Para esta questão, foi respondido pelo grupo 1, que ambos os participantes tiveram conhecimento do conceito de sensibilidade a contexto através da "universidade". Ambos os participantes do grupo 2 afirmaram que obtiveram através de "aulas" e "*sites web*".

Questão 5: Vocês já utilizaram quais aplicações sensíveis a contexto?

Os dois participantes do grupo 1 afirmaram ter utilizado como aplicação sensível a contexto o "Google" e "*sites* de compras *e-commerce*". Do grupo 2, os dois participantes citaram o "Amazon (*site*)". O participante D1 citou ainda o "Movielens", de recomendação de filmes, e o participante D2 citou o "Musicovery", de recomendação de músicas baseada no humor do usuário.

Análise das Respostas

Pode-se observar que os participantes desta fase do experimento possuem maior conhecimento sobre TV digital, dado o tempo de experiência com as linguagens NCL e Lua, as aplicações de TV digital desenvolvidas e as ferramentas de desenvolvimento utilizadas. Como eles possuem conhecimento também sobre a utilização do Ginga-NCL Virtual STB, ou seja, do que é necessário para executar uma aplicação nele, a avaliação pôde ser focada mais na ferramenta e na aplicação gerada do que nos passos necessários para executar a aplicação. Mas, isto não alterou os passos definidos para o experimento.

4.3.2 Ferramenta *Contextual Ginga*

Nesta subseção serão apresentadas as respostas para as questões da parte 2 do questionário, referente à avaliação do *Contextual Ginga*. A análise das respostas desta fase será apresentada apenas no final da subseção.

Devido à semelhança das respostas 1 e 2 no questionário aplicado na primeira fase, foi descartada a questão 2 sobre quais facilidades foram encontradas na utilização da ferramenta. A questão 1 não sofreu alterações.

Questão 1: Quais os pontos positivos da ferramenta?

As respostas informadas foram:

- Grupo 1:
  - "Gerar código sem precisar alterar";
  - "Inclusão de elementos contextuais automaticamente".
- Grupo 2: "Livra o seu usuário de lidar com códigos relacionados ao tratamento do contexto do telespectador".

Questão 2: A ferramenta permitiu que fossem cometidos erros na construção da aplicação? Quais?

O grupo 1 respondeu "não" e o grupo 2 respondeu que a ferramenta "permitiu a criação de transições impossíveis", mas não especificou o que seria uma transição impossível.

Questão 3: A adição de quais elementos contextuais poderia tornar as aplicações geradas mais interessantes?

O grupo 1 respondeu que seria interessante adicionar como elemento contextual "de onde está vindo a interação (celular, Palm...)". Por sua vez, o grupo 2 respondeu que seria interessante adicionar "informações sobre o tamanho da tela".

As questões 5 e 6 da fase anterior, analisadas em conjunto devido à semelhança das respostas, foram separadas. A questão 5 (questão 4, nesta fase) não foi alterada, mas a 6 (questão 5, nesta fase) mudou para quais funcionalidades podem ser corrigidas.

Questão 4: Quais funcionalidades podem ser adicionadas na ferramenta?

As respostas informadas foram:

- Grupo 1:
  - "*Drag and drop*";
  - "Redimensionar utilizando o mouse".
- Grupo 2:
  - "Adição de componentes mais complexos (ex. menu)";
  - "Uso de transições globais".

Questão 5: Quais funcionalidades podem ser corrigidas na ferramenta?

As respostas informadas foram:

- Grupo 1:
  - Para a funcionalidade "Alterar o nome de um componente", foi detectado o problema: "era alterado mas a ferramenta não salvava".
- Grupo 2:
  - Para a funcionalidade "Edições dos componentes (ex. redimensiomento)", foi detectado o problema: "dificuldade para editar componentes através do painel de propriedades";
  - Para a funcionalidade "Criação de telas", foi detectado o problema: "localização do comando para efetuar a ação" e foi sugerida a adição de opção no "menu *pop-up* na árvore do projeto (lado esquerdo)".

Análise das Respostas

Apesar das melhorias propostas e dos problemas ainda encontrados, pelos pontos positivos destacados, pode-se concluir que os participantes mencionaram essencialmente o objetivo visado por este trabalho.

4.3.3 Aplicação Gerada

Nesta subseção serão apresentadas as respostas para as questões da parte 3 do questionário, referente à avaliação da aplicação gerada pelo *Contextual Ginga*.

Questão 1: Compare as semelhanças e diferenças entre a representação da tela na ferramenta e a exibida no Ginga-NCL Virtual STB.

A resposta informada pelo grupo 1 foi que a aplicação "foi exibida de acordo como estava na ferramenta". O grupo 2 respondeu que a representação da tela na ferramenta e a exibida no Ginga-NCL Virtual STB eram "muito parecidas".

Questão 2: O grupo considera que conseguiu atingir o objetivo de produzir uma aplicação de TV digital sem precisar codificar e sem ter conhecimento das linguagens NCL e Lua? Por quê?

As respostas informadas foram:

- Grupo 1: "Sim -> sem precisar codificar";
- Grupo 2: "Sim. Toda a aplicação foi gerada visualmente através da ferramenta e atendeu ao seu objetivo considerando suas transições".

Nesta fase, 100% dos grupos conseguiu produzir e executar uma aplicação de TV digital sensível a contexto de acordo com a especificação. Desta forma, considera-se que, nesta fase, o objetivo da dissertação foi atingido completamente. O problema de haver muitos grupos executando o experimento ao mesmo tempo não foi reproduzido nesta segunda fase, pois o segundo grupo apenas começou o experimento após o primeiro ter concluído. Isto permitiu um maior acompanhamento do experimento.

Tempos de Execução dos Passos

Os dois grupos gastaram em cada passo os tempos especificados na Tabela 4.8. Ao contrário da fase 1, o número menor de grupos favoreceu um maior controle sobre o experimento. Dessa forma, as dúvidas puderam ser resolvidas rapidamente. Com isso, os dois grupos desta fase conseguiram construir todo o cenário proposto através da ferramenta e executar a aplicação de TV digital sensível a contexto. Como ambos os grupos construíram exatamente a mesma aplicação, ou seja, mesmas telas, componentes, transições e *personas*, pode-se realizar uma média entre os tempos utilizados em cada um dos passos. Esta média está informada na última linha da Tabela 4.8.

Tabela 4.8: Tempos das realizações dos passos na fase 2.

Grupo	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5	Total
1	7min	13min	4min	12min	20min	56min
2	1min	11min	2min	14min	33min	1h 1min
Média	4min	12min	3min	13min	26,5min	58,5 min

4.4 Considerações Finais

Este capítulo apresentou os resultados obtidos nas duas fases do experimento realizado para a avaliação da ferramenta *Contextual Ginga*. Em paralelo a esta apresentação, os resultados foram analisados.

Como análise final do experimento, ou seja, sendo consideradas as suas duas fases, conclui-se que esta pesquisa atingiu, através da ferramenta de autoria *Contextual Ginga*, o seu objetivo de permitir a produção de aplicações interativas de TV digital sensíveis a contexto. Esta análise é obtida através dos resultados do experimento, principalmente da fase 2, na qual os grupos conseguiram produzir a aplicação conforme especificada e que foi executada com sucesso.

Apesar de todos os participantes possuírem conhecimento em informática, buscou-se, na fase 1, minimizar esta influência com participantes sem experiência de desenvolvimento de aplicações interativas de TV digital. Mesmo os participantes da fase 2 possuindo esta

experiência, o impacto desta influência também foi minimizado com o fato deles abstraírem a codificação para o desenvolvimento da aplicação.

Quanto aos requisitos não-funcionais, pode ser constatado que o de facilidade de aprendizado e utilização foi alcançado principalmente na fase 2 do experimento, uma vez que os grupos conseguiram utilizar as principais funcionalidades da ferramenta na primeira vez. O segundo requisito não-funcional descrito, sobre facilidade de memorização, foi alcançado, pois os grupos que continuaram o mesmo passo entre duas aulas (fase 1) conseguiram continuar utilizando a funcionalidade da aula anterior sem precisar perguntar novamente como realizá-la. O terceiro requisito, sobre a taxa de erros permitidos pela ferramenta, pode ser considerado como satisfatório, já que do total dos 9 grupos das duas fases, apenas 4 citaram problemas.

5 Conclusões

Neste capítulo, serão destacadas as principais contribuições deste trabalho, visionadas a partir da hipótese levantada inicialmente e dos objetivos alcançados. Em seguida, serão apresentadas as necessidades, identificadas a partir do experimento, para trabalhos futuros.

5.1 Principais Contribuições

O problema definido para esta pesquisa foi: "Do ponto de vista computacional, o desenvolvimento de aplicações interativas de TV digital, particularmente adaptável (sensível) a contexto, é difícil e demorado, em especial se feito por profissional não especialista em informática."

E a partir deste problema, foi considerada a hipótese: "Se for construída uma ferramenta de autoria, que permita a produção gráfica de aplicações interativas sensíveis a contexto de TV digital para a plataforma Ginga-NCL, então uma pessoa, que entende dos conceitos de TV e de sensibilidade a contexto, poderá construir tal aplicação sem a necessidade de codificação, ou seja, mais fácil e rapidamente."

A hipótese foi submetida à tentativa de falseamento através do experimento, mas continuou válida, sendo respeitada a premissa de que as pessoas que participariam do experimento possuiriam conhecimento sobre TV e sensibilidade a contexto. A validade da hipótese pode ser inferida a partir das respostas fornecidas pelos participantes, principalmente da segunda fase, que afirmaram ter conseguido atingir o objetivo de produzir uma aplicação de TV digital através da ferramenta *Contextual Ginga*, sem codificar e sem ter conhecimento das linguagens NCL e Lua. As aplicações desenvolvidas por todos os grupos foram sensíveis a contexto.

Para as características de facilidade e rapidez no desenvolvimento presentes na hipótese, assume-se que como o público-alvo da ferramenta não sabe codificar nas

linguagens NCL e Lua, eles teriam que primeiro aprender a fazer isto para depois pôr em prática esse conhecimento. Além disso, mesmo que a pessoa saiba NCL e Lua, ferramentas de autoria já possuem intrinsecamente a função de tornar o desenvolvimento mais rápido.

A principal contribuição deste trabalho é a criação de uma ferramenta de autoria que permite o desenvolvimento de aplicações interativas sensíveis a contexto de TV digital, que tendem a aumentar a motivação do usuário ao interagir com elas. Outra contribuição é que mais facilmente e mais rapidamente são desenvolvidas aplicações mais complexas, uma vez que uma aplicação sensível a contexto é mais complexa (possui mais regras) do que uma aplicação que não realiza esse tratamento. Subdividindo estas contribuições através dos objetivos específicos traçados e alcançados ao longo desta pesquisa, tem-se:

1. *Comparação entre ferramentas de autoria*: realização de uma comparação entre as principais ferramentas de autoria voltadas para TV digital e a ferramenta proposta neste trabalho. Foram comparadas características do *middleware* em que as aplicações produzidas são executadas, a forma de edição das aplicações, o ambiente de execução da ferramenta, e se as aplicações produzidas são sensíveis a contexto;

2. *Geração automática de código*: através da ferramenta *Contextual Ginga*, o código é gerado para a máquina Ginga-NCL, nas linguagens NCL e Lua. A maior parte do código gerado é na linguagem Lua, por esta ser eficiente e permitir maior poder de expressão para a construção de aplicações mais complexas;

3. *Reuso de código*: através da biblioteca *Contextual Ginga Lua*, o código comum entre as aplicações pode ser reusado, simplificando a geração de código. A biblioteca possui funções de desenho dos componentes gráficos, de tratamento do contexto do usuário, de registro das ações tomadas pelo usuário e de gerenciamento dos *scripts* Lua;

4. *Criação de aplicações sensíveis a contexto sem codificação*: possibilidade de criação de aplicações interativas de TV digital para Ginga-NCL e, sobretudo, que sejam sensíveis a contexto, sem a necessidade de codificação por parte do desenvolvedor;

5. *Criação de aplicações sem conhecimento de NCL e Lua*: possibilidade de criação de aplicações interativas de TV digital sensíveis a contexto para Ginga-NCL por pessoas que não detêm conhecimento das linguagens de programação NCL e Lua. Ou seja, além de *Contextual Ginga NCLua* ser uma ferramenta de autoria gráfica,

o desenvolvedor não precisa saber, por exemplo, o que é um nó, uma âncora, um descritor.

5.2 Limitações da Pesquisa

São visualizadas três limitações para esta pesquisa:

- *Tipos das Aplicações*: como primeira limitação, tem-se que, por enquanto, a ferramenta é capaz de construir, sem a necessidade de codificação, apenas aplicações mais simples, que exijam a movimentação de foco entre componentes gráficos da tela. Isto consiste numa limitação, uma vez que para permitir o desenvolvimento mais rápido de aplicações interativas de TV digital, o cenário ideal é que se possa construir vários tipos de aplicações. Apesar desta limitação, o seu impacto é minimizado pelo fato que o foco principal deste trabalho ser a inserção da sensibilidade a contexto nestas aplicações.

- *Participantes do Experimento*: como segunda limitação, tem-se que todas as pessoas selecionadas para participar da avaliação do *Contextual Ginga* possuíam um alto nível de conhecimento em informática. Pois, um dos motivos de terem sido selecionados foi o fato de estarem de fácil acesso a esta pesquisa. Isto consiste numa limitação, uma vez que o problema e a hipótese de pesquisa são voltados para pessoas não especialistas em informática, mas que entendem dos conceitos de televisão e sensibilidade a contexto. Apesar que não é exclusivamente voltada para estas pessoas.

  Para contornar esta limitação, as pessoas selecionadas para a primeira fase do experimento possuíam conhecimento sobre estes conceitos, mas não possuíam sobre o desenvolvimento de aplicações de TV digital. Em trabalhos futuros, deverão se realizados experimentos com pessoas que atendam aos requisitos da hipótese e que sejam diretamente do ambiente televisivo, como jornalistas.

- *Subjetividade da Pesquisa Qualitativa*: como terceira limitação, tem-se a subjetividade da análise qualitativa e o pequeno número de avaliações permitidas por este tipo de análise em função do tempo necessário de envolvimento dos participantes na pesquisa. Com o objetivo de diminuir o impacto desta limitação, buscou-se definir, com o maior nível de detalhes possível, quais atividades deveriam ser executadas pelos participantes em cada passo do experimento. Dessa forma, todos eles estariam seguindo o mesmo processo. Além disso, no Capítulo 4, são descritas

todas as análises das informações coletadas, podendo ser comparadas aos resultados de experimentos realizados no futuro.

5.3 Trabalhos Futuros

Os trabalhos que podem dar continuidade a esta pesquisa são propostos a partir das limitações da ferramenta *Contextual Ginga* e dos resultados obtidos com o experimento realizado, que possibilitou encontrar pontos de melhoria e problemas que ainda precisam ser corrigidos.

Sobre os passos do experimento, é proposto que a ordem dos passos a serem executados seja alterada, permutando-se o passo 2 com o passo 3 de forma a existir uma melhor seqüência para reconhecimento e criação dos *personas* na ferramenta.

Para evolução da ferramenta e de sua usabilidade, o *Contextual Ginga* precisa adicionar principalmente funcionalidades listadas a seguir. Recomenda-se também que a ferramenta passe por um processo de avaliação de usabilidade.

1. Selecionar mais de um componente ao mesmo tempo, para permitir mover os componentes selecionados e para permitir também a funcionalidade listada no item a seguir;

2. Copiar e colar telas, componentes e transições com o objetivo de facilitar a criação de telas semelhantes, diminuindo o tempo gasto na criação. Ou, como alternativa, criar modelos de telas, que possam ser reutilizados;

3. Redimensionar um componente através da edição visual e não apenas através da informação dos valores na área de propriedades;

4. Desfazer as últimas alterações realizadas pelo desenvolvedor da aplicação;

5. Organizar as telas em sub-diretórios;

6. Indicar que existem alterações não salvas numa tela, através de um asterístico ao lado do nome da tela, por exemplo;

7. Permitir adicionar transições entre os componentes a partir da aba de navegação;

8. Permitir que um projeto seja portável de um diretório para outro, considerando os caminhos relativos dos arquivos de imagem e vídeo. Pois, atualmente, a ferramenta

considera o caminho absoluto e ao se copiar o projeto de um diretório para outro, as imagens deixam de ser exibidas na ferramenta, sendo necessário que o desenvolvedor reconfigure os caminhos;

9. Na aba de navegação, alterar a representação gráfica das transições entre telas diferentes, para que sejam indicadas por uma seta ou losango em vez de um círculo ou semi-círculo. Dessa forma, a representação deve se tornar mais intuitiva. Além disso, indicar qual a tela inicial do *persona* selecionado.

As aplicações geradas pela ferramenta precisam ter um nível maior de elaboração gráfica, para permitir:

1. Possibilidade de sobreposição de componentes para criar camadas numa mesma tela;
2. Suporte a outros tipos de fonte, para os componentes do tipo texto, além do tipo *Vera*;
3. Redimensionamento do vídeo do fluxo principal para ocupar uma dada fração da tela;
4. Sincronismo entre as mídias da aplicação para que, por exemplo, a aplicação seja iniciada em determinado instante do vídeo principal;
5. Permitir adicionar mais de um vídeo à aplicação.

Em relação à inserção de sensibilidade a contexto nas aplicações geradas, recomenda-se que sejam realizados experimentos com pessoas com maior conhecimento sobre este assunto, para que sejam encontradas outras melhorias para a ferramenta. Mas, já se propõe que a ferramenta possibilite:

1. Adição de outras categorias de informações contextuais para enriquecimento da sensibilidade a contexto, permitindo, por exemplo: que o desenvolvedor selecione de qual fonte (servidor remoto, dispositivo móvel) serão recuperadas as informações contextuais do usuário, sendo possível, inclusive, uma combinação de dados de múltiplas fontes (*How*); ou que possam ser consideradas outras atividades que o usuário esteja realizando ao interagir com a televisão (*What*);
2. Expansibilidade das informações consideradas em cada categoria contextual de um *persona*, permitindo que o desenvolvedor possa: adicionar elementos contextuais,

como o fuso horário ou múltiplos intervalos de tempo, dependendo da aplicação que está sendo criada; ou definir um maior ou menor nível de granularidade para a categoria *Where*. Estes elementos devem estar presentes num repositório comum para reuso;

3. Utilização de estrutura específica para armazenamento das regras contextuais que fazem parte da aplicação de TVDi gerada e melhoria da forma de inferência do contexto do usuário da aplicação;

4. Utilização de histórico do usuário ao interagir com aplicações;

5. Descoberta automática das informações do usuário que está interagindo com a televisão, podendo este usuário representar uma pessoa ou um grupo delas. Ela deve ser automática com o objetivo de diminuir ao máximo o nível de intrusão que representaria uma pergunta da aplicação sobre quem é o usuário.

Por fim, não se pode afirmar, pelo experimento realizado, que a ferramenta *Contextual Ginga* seja capaz de produzir aplicações complexas. No entanto, por limitação dos recursos computacionais dos atuais aparelhos de TV digital, as aplicações interativas ainda são relativamente simples. Isto torna viável a utilização prática da ferramenta na vida real, inclusive por programadores NCL que podem fazer melhorias no código gerado, mas que este representaria a maior parte do trabalho que ele precisaria realizar.

Referências

ABNT. *Associação Brasileira de Normas Técnicas. 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*. 2007. ABNT NBR 15606-2:2007. Disponível em: <http://www.forumsbtvd.org.br/>.

ABNT. *Associação Brasileira de Normas Técnicas. Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital Parte 1: Codificação de dados*. 2007. ABNT NBR 15606-1:2007. Disponível em: <http://www.forumsbtvd.org.br/>.

ALMEIDA, F. B. M. d.; SOUZA, R. B.; NETO, M. C. M. Ferramenta de Autoria Gráfica para a Construção de Aplicações Utilizando o Framework AppTV. In: *Sétima Escola Regional de Computação Bahia-Sergipe*. Vitória da Conquista: [s.n.], 2007.

BDA. *Blu-ray Disc Association. Blu-ray Disc*. Junho 2006. Disponível em: <http://www.blu-raydisc.com>.

BORDIN, M.; PANUNZIO, M.; VARDANEGA, T. Fitting schedulability analysis theory into model-driven engineering. In: *ECRTS '08: Proceedings of the 2008 Euromicro Conference on Real-Time Systems*. Washington, DC, USA: IEEE Computer Society, 2008. p. 135–144. ISBN 978-0-7695-3298-1.

CABLELABS. *Cable Television Laboratories. Tru2Way*. 2010. Disponível em: <http://www.tru2way.com/>.

CARVALHO, A. P. B. A. d.; FERRAZ, C. A. G. Contextual Ginga: Uma Ferramenta de Autoria de Aplicações Interativas Sensíveis ao Contexto de TV digital para Ginga-NCL. In: *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Gramado, RS: [s.n.], 2010. Salão de Ferramentas.

CHORIANOPOULOS, K. Personalized and mobile digital TV application. *Multimedia Tools and Applications*, v. 36, n. 1-2, p. 1–10, Janeiro 2008. Disponível em: <http://www.springerlink.com/content/7878817pj6106hx1/>.

COÊLHO, A. S. d. S. *ProfileTV: Um Sistema de Gerenciamento de Perfis em TVDi*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, Recife, Agosto 2008.

DEY, A. K.; ABOWD, G. D. Towards a Better Understanding of Context and Context-Awareness. In: *Proceedings of the CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*. Netherlands: [s.n.], 2000.

DVB Project. *Multimedia Home Platform (MHP)*. 2002. Disponível em: <http://www.mhp.org/>.

FLICK, U. *Uma introdução à Pesquisa Qualitativa*. 3. ed. [S.l.]: Artmed, 2008. 408 p.

GOMES, P. C. *Simplicidade aplicada ao design interativo na TV digital: o desenvolvimento de aplicações com interfaces orientadas ao usuário e seu contexto de uso*. Dissertação (Especialização em Produção para TV Digital) — Faculdade de Comunicação Multimídia da Universidade Metodista de São Paulo, São Bernardo do Campo, 2008. Disponível em: <http://biblioteca.metodista.br/>.

GROUP, O. T. K. *Open TV Kudelski Group. Open TV*. 1994. Disponível em: <http://www.opentv.com/>.

GUIMARÃES, R. L. *Composer: um ambiente de autoria de documentos NCL para TV digital interativa*. Dissertação (Mestrado em Informática) — PUC, Rio de Janeiro, Junho 2007. Disponível em: <http://www2.dbd.puc-rio.br>.

ICAREUS. *Icareus iTV Suite Author*. 2009. Disponível em: <http://www.icareus.com/>.

IERUSALIMSCHY, R.; FIGUEIREDO, L. H. d.; CELES, W. *Lua 5.1 Reference Manual*. Lua.org, 2006. Disponível em: <http://www.lua.org>.

IMK. *Fraunhofer Institute for Media Communication. JAME Author*. 2005. Disponível em: <http://jame.tv/>.

IRDETO. *Java-based Irdeto IDway-J middleware*. 2008. Disponível em: <http://www.irdeto.com>.

ITU. *International Telecommunication Union. Nested context language (NCL) and Ginga-NCL for IPTV services*. Abril 2009. H SERIES: AUDIOVISUAL AND MULTIMEDIA SYSTEMS. IPTV multimedia services and applications for IPTV - IPTV multimedia application frameworks. Disponível em: <http://www.itu.int/itu-t/recommendations/rec.aspx?id=9715>.

JENSEN, J. F. Interactive television: new genres, new format, new content. In: *IE2005: Proceedings of the second Australasian conference on Interactive entertainment*. Sydney, Australia: Creativity & Cognition Studios Press, 2005. p. 89–96. ISBN 0-9751533-2-3.

KIM, E.; AHN, J.; KO, S. Understanding the Adoption of Telecommunications-Broadcasting Convergence Services: Focusing on t-Commerce. In: *ICCIT '07: Proceedings of the 2007 International Conference on Convergence Information Technology*. Washington, DC, USA: IEEE Computer Society, 2007. p. 2368–2373. ISBN 0-7695-3038-9.

LAWS. *Laboratory of Advanced Web Systems. NCL Eclipse*. 2008. Universidade Federal do Maranhão. Disponível em: <http://laws.deinf.ufma.br/ ncleclipse/>.

MAEDA, J. *The Laws of Simplicity*. Massachusetts, MA, United States of America: MIT Press, 2006. 127 p. (Simplicity: Design, Technology, Business, Life).

MICROSYSTEMS, S. *Java Runtime Environment*. 2006. Disponível em: <http://java.com>.

MORSE, D. R.; ARMSTRONG, S.; DEY, A. K. The What, Who, Where, When, and How of Context-Awareness. In: *Proceedings of the CHI 2000 Workshop*. Georgia Institute of Technology, 2000. Disponível em: <http://www.cc.gatech.edu/fce/contexttoolkit/pubs/CHI2000-workshop.pdf>.

NETO, C. d. S. S. et al. *Construindo Programas Audiovisuais Interativos Utilizando a NCL3.0 e a Ferramenta Composer*. 2ª. ed. Rio de Janeiro, Julho 2007.

NIELSEN, J. *Usability Engineering*. [S.l.]: Academic Press, 1994. 362 p.

NIEUWDORP, E. The pervasive discourse: an analysis. *Comput. Entertain.*, ACM, New York, NY, USA, v. 5, n. 2, p. 13, 2007. ISSN 1544-3574.

OLIVEIRA, M. R.; FILHO, C. B.; SILVA, A. F. iTV project: an authoring tool for mhp and Ginga-J based on a web environment. In: *UXTV '08: Proceeding of the 1st international conference on Designing interactive user experiences for TV and video*. New York, NY, USA: ACM, 2008. p. 179–182. ISBN 978-1-60558-100-2.

PORTAL. *Portal do Software Público Brasileiro. Ginga*. 2009. Disponível em: <http://www.softwarepublico.gov.br/>.

PRUITT, J.; ADLIN, T. *The Persona Lifecycle: Keeping People in Mind Throughout Product Design*. [S.l.]: Morgan Kaufmann Publishers, 2006. 724 p.

REY-LÓPEZ, M. et al. T-MAESTRO and its authoring tool: using adaptation to integrate entertainment into personalized t-learning. *Multimedia Tools and Applications*, v. 40, n. 3, p. 409–451, Dezembro 2008.

SCHIBELSKY, L. et al. Understanding iDTV in a developing country and designing a T-gov application prototype. In: *DIS '08: Proceedings of the 7th ACM conference on Designing interactive systems*. New York, NY, USA: ACM, 2008. p. 379–385. ISBN 978-1-60558-002-9.

SILVA, A. F. R.; OLIVEIRA, M. R. M. Projeto de um Framework de Desenvolvimento de Interfaces Gráficas para TV Digital. Monografia da Universidade de Brasília UnB. 2006.

SILVA, L. D. N. e.; TAVARES, T. A.; FILHO, G. L. d. S. Desenvolvimento de programas de TVDI explorando as funções inovadoras do GINGA-J. In: *WebMedia '08: Proceedings of the 14th Brazilian Symposium on Multimedia and the Web*. New York, NY, USA: ACM, 2008. p. 75–82.

SVEDEN, M. XletView. Emulador do middleware MHP. 2003. Disponível em: <http://xletview.sourceforge.net>.

THAWANI, A.; GOPALAN, S.; SRIDHAR, V. Context Aware Personalized Ad Insertion in an Interactive TV Environment. In: *TV'04: the 4th Workshop on Personalization in Future TV - Methods, Technologies, Applications for Personalized TV*. Eindhoven, The Netherlands: [s.n.], 2004.

TøNDERING, C. *Frequently Asked Questions about Calendars*. Abril 2008. Disponível em: <http://www.tondering.dk/claus/cal/calendar29.html>.

VIEIRA, V.; TEDESCO, P.; SALGADO, A. A process for the design of Context-Sensitive Systems. In: *Proceedings of the 2009 13th International Conference on Computer Supported Cooperative Work in Design*. Santiago, Chile: IEEE Computer Society, 2009. p. 143 –148.

W3C. *World Wide Web Consortium. Document Object Model (DOM)*. Janeiro 2005. Disponível em: <http://www.w3.org/DOM/>.

WEISS, D. et al. A user profile-based personalization system for digital multimedia content. In: *DIMEA '08: Proceedings of the 3rd international Conference on Digital interactive Media in Entertainment and Arts*. ACM, New York, NY: ACM, 2008. v. 349, p. 281–288.

APÊNDICE A – Casos de Uso

Neste apêndice estão descritos os casos de uso da ferramenta de autoria *Contextual Ginga*. Por convenção estes casos de uso serão identificados e referenciados pelo padrão [UCx], onde x representa o número do caso de uso. Além do identificador, o caso de uso possui descrição, prioridade, pré e pós-condições e os fluxos de eventos principal, alternativo e de exceção. Todos os casos de uso possuem o mesmo ator, que é o desenvolvedor que constrói aplicações interativas sensíveis a contexto para TV digital.

Os casos de uso a seguir estão organizados por cada conceito principal da ferramenta, assim como os requisitos foram. Para cada conceito, é apresentado um diagrama de casos de uso.

A.1 Projeto

A Figura A.1 exibe o diagrama dos casos de uso relacionados ao conceito Projeto e estes são descritos da Tabela A.1 à Tabela A.6.

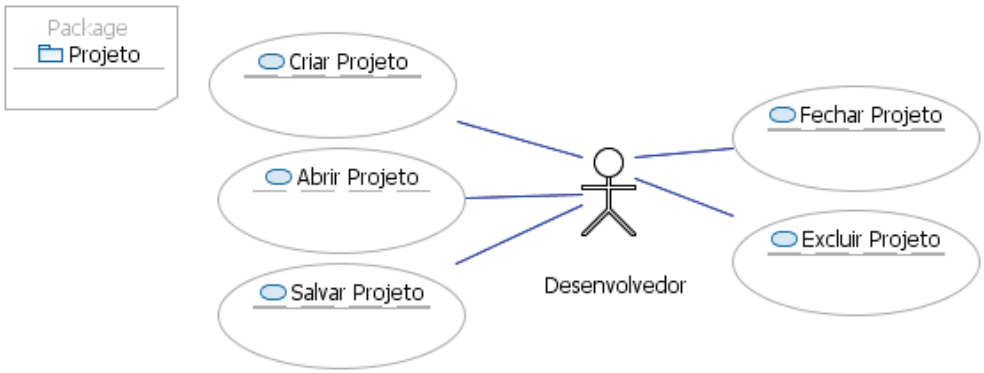


Figura A.1: Diagrama de Casos de Uso de Projeto.

Tabela A.1: Caso de Uso UC1 Criar Projeto.

Identificação:	**UC1 Criar Projeto**
Descrição:	Permite agrupar um conjunto de telas para formar uma aplicação de TV digital numa determinada resolução.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com a ferramenta aberta.
Pós-condições:	Um novo projeto é criado.
Fluxos de Eventos	
Principal:	1.O usuário clica no botão Criar Projeto; 2.Na janela aberta, o usuário informa um diretório vazio onde os arquivos do projeto devem ser armazenados e clica no botão OK; 3.Na janela aberta, o usuário seleciona a resolução que as telas da aplicação devem possuir e clica no botão Finish; 4.A ferramenta cria os dados iniciais do projeto.
Alternativo:	No passo 2 do fluxo principal, o usuário cancela a criação do projeto.
De Exceção:	No passo 2 do fluxo principal, se o diretório selecionado já possuir um projeto armazenado, a ferramenta exibe uma mensagem com esta informação.

Tabela A.2: Caso de Uso UC2 Abrir Projeto.

Identificação:	**UC2 Abrir Projeto**
Descrição:	Permite que se visualize todos os dados de um projeto e que se trabalhe nele.
Prioridade:	Essencial
Pré-condições:	Deve existir um projeto criado.
Pós-condições:	Os dados do projeto selecionado são exibidos.
Fluxos de Eventos	
Principal:	1.O usuário clica no botão Abrir Projeto; 2.Na janela aberta, o usuário informa o diretório onde os arquivos do projeto estão armazenados; 3.O usuário clica no botão OK; 4.A ferramenta exibe os dados do projeto.
Alternativo:	No passo 3 do fluxo principal, o usuário cancela a escolha de um diretório que contenha um projeto.
De Exceção:	Não se aplica.

Tabela A.3: Caso de Uso UC3 Salvar Projeto.

Identificação:	**UC3 Salvar Projeto**
Descrição:	Permite que sejam salvos todos os dados do projeto, ou seja, todas as telas com seus componentes e transições, bem como os *personas*.
Prioridade:	Essencial
Pré-condições:	Deve existir um projeto aberto.
Pós-condições:	Os dados do projeto selecionado são salvos.
Fluxos de Eventos	
Principal:	1.O usuário clica no botão Salvar Projeto; 2.A ferramenta salva os dados do projeto.
Alternativo:	Não se aplica.
De Exceção:	Não se aplica.

Tabela A.4: Caso de Uso UC4 Fechar Projeto.

Identificação:	**UC4 Fechar Projeto**
Descrição:	Permite que não mais se visualize os dados de um projeto.
Prioridade:	Importante
Pré-condições:	Deve existir um projeto aberto.
Pós-condições:	Os dados do projeto que estava aberto deixam de ser exibidos.
Fluxos de Eventos	
Principal:	1.O usuário clica no botão Fechar Projeto; 2.A ferramenta deixa de exibir os dados do projeto.
Alternativo:	Não se aplica.
De Exceção:	Não se aplica.

Tabela A.5: Caso de Uso UC5 Excluir Projeto.

Identificação:	**UC5 Excluir Projeto**
Descrição:	Permite excluir os arquivos com informações sobre o projeto em que se está trabalhando.
Prioridade:	Desejável
Pré-condições:	Deve existir um projeto aberto.
Pós-condições:	Os dados do projeto aberto são excluídos.
Fluxos de Eventos	
Principal:	1.O usuário clica no botão Excluir Projeto; 2.A ferramenta abre uma janela para perguntar se o usuário tem certeza que deseja excluir o projeto;

Tabela A.6: Caso de Uso UC5 Excluir Projeto.

Identificação:	**UC5 Excluir Projeto**
	Fluxos de Eventos
Principal:	3.O usuário seleciona a opção para excluir o projeto; 4.A ferramenta exclui os dados do projeto.
Alternativo:	1.No passo 3 do fluxo principal, se o usuário não escolher a opção de confirmar a exclusão do projeto, o evento do passo 4 não será executado.
De Exceção:	Não se aplica.

A.2 Tela

A Figura A.2 exibe o diagrama dos casos de uso relacionados ao conceito Tela e estes são descritos da Tabela A.7 à Tabela A.13.

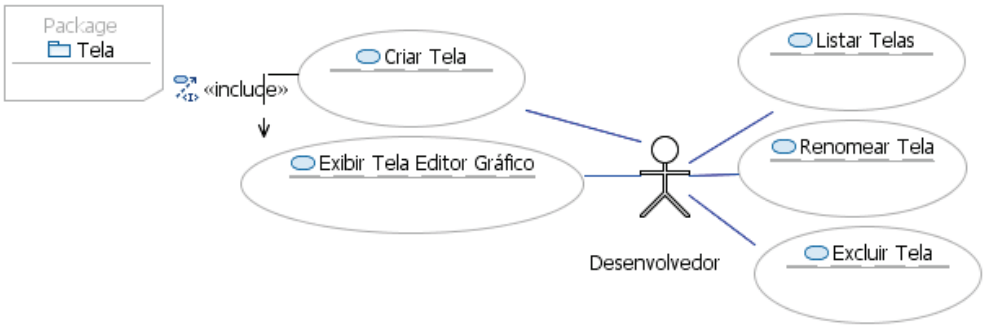


Figura A.2: Diagrama de Casos de Uso de Tela.

Tabela A.7: Caso de Uso UC6 Criar Tela.

Identificação:	**UC6 Criar Tela**
Descrição:	Permite adicionar uma tela ao projeto.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	Uma nova tela é criada no projeto aberto.

Tabela A.8: Caso de Uso UC6 Criar Tela.

Identificação:	**UC6 Criar Tela**
	Fluxos de Eventos
Principal:	1.O usuário clica no botão Criar Tela; 2.A ferramenta exibe uma janela para o usuário informar o nome para a tela; 3.O usuário informa o nome para a tela e clica no botão OK; 4.A ferramenta adiciona uma nova tela ao projeto; 5.A nova tela é exibida no editor gráfico.
Alternativo 1:	No passo 3 do fluxo principal, o usuário desiste de criar uma tela e clica no botão Cancelar.
De Exceção 1:	No passo 3 do fluxo principal, se o usuário informar um nome de tela vazio, é exibida uma mensagem dizendo que um nome precisa ser informado.
De Exceção 2:	No passo 3 do fluxo principal, se o usuário informar um nome que já existe em outra tela do projeto, é exibida uma mensagem com esta exceção e a ferramenta continua com a tela aberta para o usuário informar novamente o nome para a mesma.

Tabela A.9: Caso de Uso UC7 Listar Telas.

Identificação:	**UC7 Listar Telas**
Descrição:	Permite que sejam exibidos os nomes de todas as telas pertencentes ao projeto.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	As telas do projeto são listadas.
	Fluxos de Eventos
Principal:	Sempre que o usuário está com um projeto aberto, são listados os nomes das telas deste projeto.
Alternativo:	Não se aplica.
De Exceção:	Não se aplica.

Tabela A.10: Caso de Uso UC8 Exibir Tela em Editor Gráfico

Identificação:	**UC8 Exibir Tela em Editor Gráfico**
Descrição:	Permite que as telas e seus componentes sejam exibidos em formato gráfico.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	Uma tela é exibida no editor visual.

Tabela A.11: Caso de Uso UC8 Exibir Tela em Editor Gráfico

Identificação:	**UC8 Exibir Tela em Editor Gráfico**
Fluxos de Eventos	
Principal:	1.Dentre as telas listadas na janela principal da ferramenta, o usuário clica no nome de uma delas. 2.A ferramenta exibe graficamente a representação da tela.
Alternativo:	Não se aplica.
De Exceção:	Não se aplica.

Tabela A.12: Caso de Uso UC9 Renomear Tela.

Identificação:	**UC9 Renomear Tela**
Descrição:	Permite alterar o nome de uma tela.
Prioridade:	Desejável.
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	O nome de uma tela do projeto é alterado.
Fluxos de Eventos	
Principal:	1.Dentre as telas listadas na janela principal da ferramenta, o usuário clica com o botão esquerdo para selecionar a tela que quer renomear; 2.O usuário clica com o botão direito no nome da tela selecionada; 3.O usuário clica na opção Renomear Tela; 4.A ferramenta abre uma janela para o usuário informar o novo nome para a tela; 5.O usuário clica no botão OK; 6.A ferramenta faz as alterações necessárias no projeto para trocar o nome da tela.
Alternativo:	No passo 5 do fluxo principal, o usuário desiste de renomear a tela e clica no botão Cancelar.
De Exceção 1:	No passo 5 do fluxo principal, se o usuário informar um nome de tela vazio, é exibida uma mensagem dizendo que um nome precisa ser informado.
De Exceção 2:	No passo 5 do fluxo principal, se o usuário informar um nome que já existe em outra tela do projeto, a ferramenta exibe uma mensagem com essa informação e pede outro nome.

Tabela A.13: Caso de Uso UC10 Excluir Tela.

Identificação:	**UC10 Excluir Tela**
Descrição:	Permite excluir uma tela de um projeto. Com esta exclusão, também são excluídas as transições que se originam da tela e que se destinam a ela. Caso existam, os pontos iniciais formados por esta tela também serão excluídos.
Prioridade:	Desejável.
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	Uma tela é excluída do projeto.
Fluxos de Eventos	
Principal:	1.Dentre as telas listadas na janela principal da ferramenta, o usuário clica com o botão esquerdo para selecionar a tela que quer excluir; 2.O usuário clica com o botão direito no nome da tela selecionada e clica na opção Excluir Tela; 3.Na janela aberta, o usuário clica no botão Sim para confirmar a exclusão; 4.A ferramenta exclui a tela do projeto.
Alternativo:	No passo 4 do fluxo principal, o usuário desiste de excluir a tela e clica no botão Não ou no botão Cancelar.
De Exceção:	Não se aplica.

A.3 Componente

A Figura A.3 exibe o diagrama dos casos de uso relacionados ao conceito Componente e estes são descritos da Tabela A.14 à Tabela A.19.

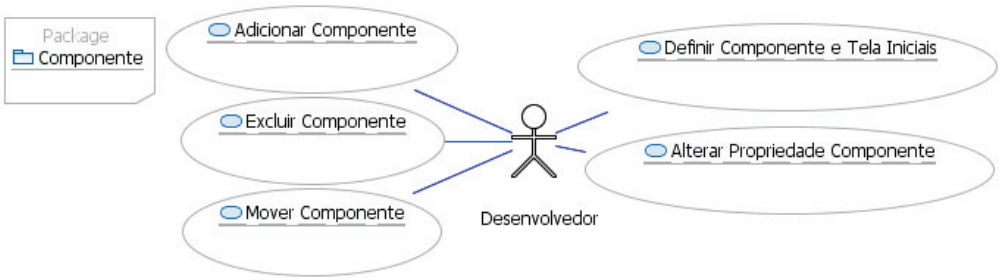


Figura A.3: Diagrama de Casos de Uso de Componente.

Tabela A.14: Caso de Uso UC11 Adicionar Componente.

Identificação:	**UC11 Adicionar Componente**
Descrição:	Permite que o usuário adicione um novo componente à tela que está sendo mostrada na ferramenta.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto e com a representação de uma tela sendo exibida.
Pós-condições:	Um novo componente é adicionado a uma tela do projeto.
Fluxos de Eventos	
Principal:	1.O usuário clica em um dos tipos disponíveis de componente; 2.O usuário clica no local da tela que prefere colocar o novo componente; 3.A ferramenta abre uma janela para o usuário informar o nome do componente; 4.O usuário informa o nome e clica no botão OK; 5.A ferramenta pode pedir mais alguma informação dependendo do tipo do componente. Exemplo: se for uma imagem, o endereço onde o arquivo está armazenado; 6.A ferramenta adiciona um novo componente à tela.
Alternativo 1:	No passo 1 do fluxo principal, se o usuário desistir de adicionar um componente, ele clica no botão com uma imagem de apontador.
Alternativo 2:	No passo 3 do fluxo principal, se o usuário desistir de adicionar um componente, ele clica no botão Cancelar.
De Exceção 1:	No passo 4 do fluxo principal, se o usuário informar um nome de componente vazio, é exibida uma mensagem dizendo que um nome precisa ser informado.
De Exceção 2:	No passo 4 do fluxo principal, se o usuário informa um nome que já existe em outro componente da mesma tela do projeto, a ferramenta exibe uma mensagem com essa informação e pede outro nome.

Tabela A.15: Caso de Uso UC12 Excluir Componente.

Identificação:	**UC12 Excluir Componente**
Descrição:	Permite excluir um componente de uma tela. Quando o componente é excluído, ele deixa de existir na visão de todos os *personas* adicionados ao projeto.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto e com a representação de uma tela sendo exibida.
Pós-condições:	Um componente é excluído de uma tela do projeto.

Tabela A.16: Caso de Uso UC12 Excluir Componente.

Identificação:	**UC12 Excluir Componente**
Fluxos de Eventos	
Principal:	1.O usuário clica com o botão direito em cima do componente que deseja excluir; 2.O usuário clica na opção Excluir Componente; 3.A ferramenta abre uma janela, pedindo confirmação para a exclusão; 4.O usuário clica no botão Sim; 5.A ferramenta exclui o componente.
Alternativo:	No passo 4 do fluxo principal, se o usuário desistir de excluir um componente, ele clica no botão Não ou no botão Cancelar.
De Exceção:	Não se aplica.

Tabela A.17: Caso de Uso UC13 Mover Componente.

Identificação:	**UC13 Mover Componente**
Descrição:	Permite mover graficamente um componente através de *drag-and-drop*. O componente não pode ser movido para fora da área da tela.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto e com a representação de uma tela sendo exibida.
Pós-condições:	Um componente da tela tem sua posição alterada.
Fluxos de Eventos	
Principal:	1.O usuário clica com o botão esquerdo em cima do componente que se deseja mover; 2.O usuário, sem deixar de pressionar o botão esquerdo, move o componente até a posição desejada; 3.O usuário deixa de pressionar o botão esquerdo.
Alternativo:	Através da lista de propriedades do componente, o usuário altera os valores x e y da posição do componente e tecla Enter.
De Exceção:	No passo 2 do fluxo principal, se o usuário tentar mover o componente para fora dos limites da tela, a ferramenta reposiciona o componente dentro da tela.

Tabela A.18: Caso de Uso UC14 Definir Componente e Tela Iniciais.

Identificação:	**UC14 Definir Componente e Tela Iniciais**
Descrição:	Permite determinar para cada *persona* do projeto, qual a primeira tela a ser exibida e qual o componente desta tela que receberá foco inicialmente.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	São atribuídos uma tela e um componente iniciais ao *persona* selecionado.
Fluxos de Eventos	
Principal:	1.O usuário clica com o botão esquerdo em cima de um componente; 2.O usuário clica na opção Definir como Componente Inicial; 3.A ferramenta atribui a tela e o componente selecionados como o ponto inicial para o *persona* que está selecionado.
Alternativo:	1.O usuário clica na aba de navegação; 2.A ferramenta mostra os nomes das telas do projeto, juntamente com as transições entre as mesmas. 3.O usuário clica com o botão direito no nome da tela para ser a inicial; 4.O usuário clica na opção Definir como Tela Inicial; 5.A ferramenta mostra uma janela para o usuário selecionar o primeiro componente a ter foco na tela; 6.O usuário seleciona um componente pertencente à tela; 7.O usuário clica no botão OK; 8.A ferramenta atribui a tela e o componente selecionados como o ponto inicial para o *persona* que está selecionado.
De Exceção:	No passo 2 do fluxo principal, se para o *persona* selecionado já existir uma tela e seu componente inicial definidos, a ferramenta diz qual é e pede uma confirmação do usuário para a alteração.

Tabela A.19: Caso de Uso UC15 Alterar Propriedade de um Componente.

Identificação:	**UC15 Alterar Propriedade de um Componente**
Descrição:	Permite alterar o valor de uma propriedade de um componente a partir de uma lista de propriedades que variam de acordo com o tipo do componente.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto e com a representação de uma tela sendo exibida.
Pós-condições:	É alterado o valor de uma propriedade de um componente.
Fluxos de Eventos	
Principal:	1.O usuário clica com o botão esquerdo em cima de um componente; 2.A ferramenta exibe na barra lateral os valores de todas as propriedades para o componente selecionado, de acordo com o seu tipo; 3.O usuário clica no valor atual da propriedade; 4.O usuário altera o valor da propriedade; 5.O usuário tecla Enter.
Alternativo:	No passo 3 do fluxo principal, caso o valor seja um texto, o usuário tecla F2 para renomear o valor da propriedade. Caso seja outro tipo, é aberta uma janela para o usuário alterar uma cor ou o endereço de um arquivo.
De Exceção:	No caso da propriedade nome do componente, mesmos fluxos de exceção do UC11 Adicionar Componente.

A.4 Persona

A Figura A.4 exibe o diagrama dos casos de uso relacionados ao conceito *Persona* e estes são descritos da Tabela A.20 à Tabela A.23.

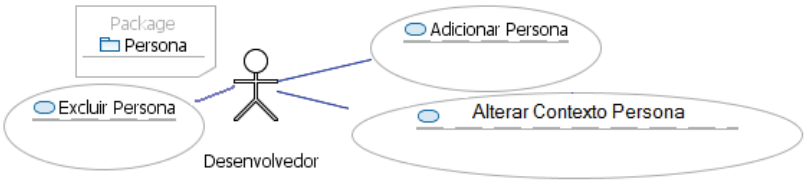


Figura A.4: Diagrama de Casos de Uso de *Persona*.

Tabela A.20: Caso de Uso UC16 Adicionar *Persona*.

Identificação:	**UC16 Adicionar *Persona***
Descrição:	Permite adicionar um *persona* ao projeto.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	É adicionado um *persona* ao projeto.
Fluxos de Eventos	
Principal:	1.O usuário clica no botão com imagem "+"; 2.Na janela aberta, o usuário digita o nome do *persona* e clica no botão OK; 3.A ferramenta adiciona o novo *persona* ao projeto.
Alternativo:	Não se aplica.
De Exceção 1:	No passo 3 do fluxo principal, se o usuário informar um nome de *persona* vazio, é exibida uma mensagem dizendo que um nome precisa ser informado.
De Exceção 2:	No passo 3 do fluxo principal, se o usuário informa um nome que já existe em outro *persona* do projeto, a ferramenta exibe uma mensagem com essa informação e pede outro nome.

Tabela A.21: Caso de Uso UC17 Excluir *Persona*.

Identificação:	**UC17 Excluir *Persona***
Descrição:	Permite excluir um *persona* do projeto. Quando um *persona* é excluído, as transições associadas a ele são excluídas também.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	É excluído um *persona* do projeto e as transições associadas.
Fluxos de Eventos	
Principal:	1.O usuário seleciona o nome do *persona* que deseja excluir; 2.O usuário clica no botão com imagem "-";

Tabela A.22: Caso de Uso UC17 Excluir *Persona*.

Identificação:	**UC17 Excluir *Persona***
Fluxos de Eventos	
Principal:	3.A ferramenta exclui do projeto o *persona* selecionado.
Alternativo:	1.No passo 2 do fluxo principal, caso o *persona* possua transições associadas, a ferramenta exibe uma mensagem com esta informação e a descrição de até 3 transições, e pede uma confirmação para excluir o *persona*; 2.O usuário clica no botão Sim.
De Exceção:	Não se aplica.

Tabela A.23: Caso de Uso UC18 Alterar Contexto do *Persona*.

Identificação:	**UC18 Alterar Contexto do *Persona***
Descrição:	Permite alterar as características contextuais de um *persona* para as categorias: *Who*, *When* e *Where*.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	Um *persona* possui novos valores para os elementos contextuais.
Fluxos de Eventos	
Principal:	1.O usuário seleciona o nome do *persona* que deseja alterar o elemento contextual; 2.O usuário altera o valor desejado e aperta Tab ou tira o foco do campo.
Alternativo:	Não se aplica.
De Exceção:	Não se aplica.

A.5 Transição

A Figura A.5 exibe o diagrama dos casos de uso relacionados ao conceito Transição e estes são descritos da Tabela A.24 à Tabela A.29.

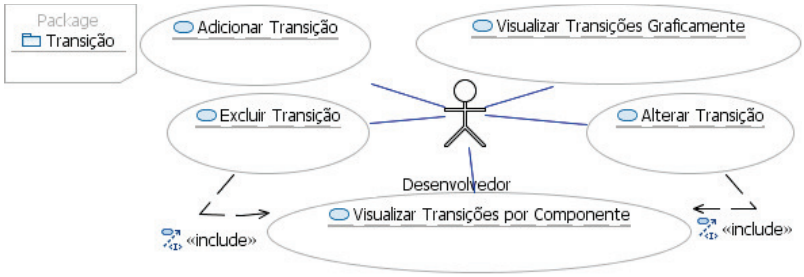


Figura A.5: Diagrama de Casos de Uso de Transição.

Tabela A.24: Caso de Uso UC19 Adicionar Transição.

Identificação:	**UC19 Adicionar Transição**
Descrição:	Permite que se crie uma transição entre dois componentes, estando eles na mesma tela ou em telas distintas.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto e com a representação de uma tela sendo exibida.
Pós-condições:	Uma nova transição entre componentes é adicionada ao projeto.
Fluxos de Eventos	
Principal:	1.O usuário clica com o botão direito em cima de um componente e clica na opção Adicionar Transição; 2.A ferramenta abre uma janela para o usuário informar qual a tela e o componente de destino e a ação que dispara a transição. Os outros valores de uma transição são obtidos do *persona* selecionado, da tela e do componente que o usuário clicou; 3.O usuário clica no botão OK; 4.A ferramenta adiciona a nova transição.
Alternativo:	Não se aplica.
De Exceção:	No passo 4 do fluxo principal, se já existir uma transição com os mesmos valores, a ferramenta exibe mensagem com essa informação.

Tabela A.25: Caso de Uso UC20 Visualizar Transições.

Identificação:	**UC20 Visualizar Transições**
Descrição:	Permite visualizar as transições que são originadas em um componente ou tela por *persona*.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto e com a representação de uma tela ou a aba de navegação do projeto sendo exibida.
Pós-condições:	O usuário visualiza as transições que partem de um componente ou de uma tela por *persona*.
Fluxos de Eventos	
Principal:	1.Na representação de uma tela, o usuário clica com o botão direito em cima de um componente; 2.O usuário clica na opção Transições; 3.A ferramenta abre uma janela que lista todas as transições que partem deste componente.
Alternativo:	No passo 1 do fluxo principal, o usuário clica com o botão direito em cima do nome de uma tela na aba de navegação do projeto.
De Exceção:	Não se aplica.

Tabela A.26: Caso de Uso UC21 Excluir Transição.

Identificação:	**UC21 Excluir Transição**
Descrição:	Permite que se exclua uma transição entre dois componentes.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto e com a representação de uma tela sendo exibida.
Pós-condições:	Uma transição entre componentes é excluída do projeto.
Fluxos de Eventos	
Principal:	1.O usuário seleciona uma transição;

Tabela A.27: Caso de Uso UC21 Excluir Transição.

Identificação:	**UC21 Excluir Transição**
	Fluxos de Eventos
Principal:	3.O usuário clica no botão para Excluir; 4.A ferramenta exclui a transição do projeto.
Alternativo:	Não se aplica.
De Exceção:	Não se aplica.

Tabela A.28: Caso de Uso UC22 Alterar Transição.

Identificação:	**UC22 Alterar Transição**
Descrição:	Permite que se altere as propriedades de uma transição.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto e com a representação de uma tela sendo exibida.
Pós-condições:	As propriedades de uma transição são alteradas.
	Fluxos de Eventos
Principal:	1.O usuário seleciona uma transição; 2.O usuário clica no botão para Alterar; 3.A ferramenta abre uma janela que exibe os dados atuais da transição; 4.O usuário altera os dados desejados; 5.O usuário clica no botão OK; 6.A ferramenta altera a transição do projeto.
Alternativo:	Não se aplica.
De Exceção:	No passo 5 do fluxo principal, se já existir uma transição com os mesmos valores, a ferramenta exibe mensagem com essa informação.

Tabela A.29: Caso de Uso UC23 Visualizar Transições Graficamente.

Identificação:	**UC23 Visualizar Transições Graficamente**
Descrição:	Permite que sejam visualizadas, para um dado *persona*, as transições existentes entre as telas de maneira similar a um diagrama de estados. Mesmo que não existam transições, as telas são representadas graficamente. Esta funcionalidade também permite que o usuário da ferramenta altere a exibição da navegação entre as telas.
Prioridade:	Importante
Pré-condições:	O usuário deve estar com um projeto aberto e com a aba de navegação do projeto sendo exibida.
Pós-condições:	Os fluxos entre as telas do projeto são exibidos.
Fluxos de Eventos	
Principal:	1.O usuário clica na aba de navegação. 2.A ferramenta exibe os nomes de todas as telas do projeto e uma seta para cada transição entre telas. Para transições dentro da mesma tela, é exibido um círculo no canto superior esquerdo do nome da tela. 3.O usuário pode arrastar os nomes das telas.
Alternativo:	Não se aplica.
De Exceção:	Não se aplica.

A.6 Código NCLua

A Figura A.6 exibe o caso de uso relacionado à geração de código NCLua para a aplicação interativa sensível a contexto e este está descrito na Tabela A.30.

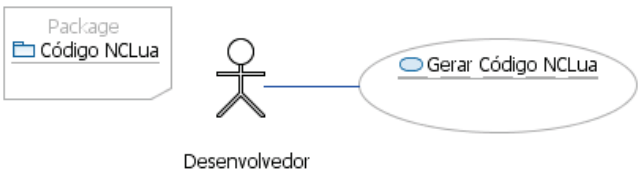


Figura A.6: Diagrama de Casos de Uso de Código NCLua.

Tabela A.30: Caso de Uso UC24 Gerar Código NCLua.

Identificação:	**UC24 Gerar Código NCLua**
Descrição:	Permite produzir a aplicação para a plataforma Ginga-NCL a partir dos dados presentes em um projeto construído na ferramenta.
Prioridade:	Essencial
Pré-condições:	O usuário deve estar com um projeto aberto.
Pós-condições:	O código NCLua de uma aplicação de TV digital é gerado para o middleware Ginga.
Fluxos de Eventos	
Principal:	1.No menu Código, o usuário clica no item Gerar Código NCLua; 2.Se o projeto não está salvo, a ferramenta diz que irá salvar antes de gerar código e pede uma confirmação do usuário para prosseguir; 3.O usuário clica no botão Sim; 4.A ferramenta gera o código NCLua na pasta de saída dentro do diretório do projeto; 5.A ferramenta exibe uma mensagem de sucesso sobre a geração do código.
Alternativo:	Não se aplica.
De Exceção 1:	Se o projeto não tiver telas, é exibida uma mensagem com essa informação.
De Exceção 2:	Se os *personas* do projeto não tiverem tela e componente iniciais definidos, é exibida uma mensagem com essa informação.

APÊNDICE B – Questionário

Contextual Ginga é uma ferramenta de autoria, proposta por este trabalho de dissertação de mestrado, que permite a produção de aplicações interativas sensíveis a contexto de TV digital para a plataforma Ginga-NCL.

Para realizar a avaliação da ferramenta, será aplicado este questionário com o objetivo de obter: informações sobre as experiências dos participantes com TV digital e sensibilidade a contexto; e elementos que avaliem o estágio atual da ferramenta e melhorias necessárias para a sua evolução. Com base nestes objetivos, este questionário está dividido em três partes:

- **A1**: Obtenção de informações sobre as suas experiências dos participantes;
- **A2**: Avaliação da ferramenta *Contextual Ginga*;
- **A3**: Avaliação da aplicação gerada pela ferramenta.

Passos da Avaliação

Para a avaliação, é fornecido um cenário de aplicação de TV digital e os protótipos de suas telas. A produção da aplicação deve seguir os passos abaixo, devendo cada passo ter registrado o tempo necessário para a sua execução (Tabela B.1, no final da Parte A1). Apenas o passo 1 e a execução do código do passo 5 não são realizados através do *Contextual Ginga*. A parte A1 deste questionário deve ser preenchida antes da realização do Passo 1 e as partes A2 e A3, depois do Passo 5.

- **Passo 1**: Com base no cenário da aplicação, reconhecer quais elementos fornecerão sensibilidade a contexto para a aplicação;
- **Passo 2**: Criar o projeto, as telas da aplicação e os seus componentes;
- **Passo 3**: Criar os *personas* e inserir as suas características contextuais;

- **Passo 4**: Definir qual o ponto inicial (tela e componente iniciais) de cada *persona* e informar suas transições;

- **Passo 5**: Gerar o código da aplicação e executá-la no Ginga-NCL Virtual STB.

Observações

Principalmente por ainda estar em fase de desenvolvimento, Contextual Ginga possui limitações que devem ser observadas durante sua utilização:

1. Não permite a sobreposição de componentes;

2. Não devem ser criados componentes com nomes iguais no mesmo projeto, mesmo que em telas diferentes.

Sigilo das Informações

Com exceção dos nomes das pessoas, as informações providas neste questionário serão transcritas para a dissertação deste trabalho de mestrado, bem como a análise das respostas. Estas informações também poderão ser utilizadas em possíveis artigos futuros. Os nomes dos participantes são informações sigilosas que não serão divulgadas em nenhum momento.

Agradecimentos

O orientador da pesquisa, Prof. Carlos André Guimarães Ferraz (cagf@cin.ufpe.br) e a aluna de mestrado Ana Paula Bezerra Alves de Carvalho (apba@cin.ufpe.br) agradecem antecipadamente a contribuição dos participantes desta pesquisa.

B.1 Informações dos Participantes

Cada participante será identificado pela letra P de participante seguido de um número que o representará dentro do grupo. Para as questões com tabelas que possuem colunas variando de P1 a P4, devem ser informados com um "X"na coluna dos participantes associados à informação solicitada.

Número do grupo: ____

1.Dados pessoais

Nome (P1): _____ Email: _____

Nome (P2): _____ Email: _____

Nome (P3): _____ Email: _____

Nome (P4): _____ Email: _____

2.Vocês já utilizaram aplicações de TV digital? ( ) Sim ( ) Não

Se sim, quais? (Detalhar se a aplicação era sensível a contexto.)

	Aplicação	P1	P2	P3	P4
A1					
A2					
A3					
A4					

3.Vocês já desenvolveram aplicações para TV digital? □ Sim □ Não

Se sim, quais? (Detalhar se a aplicação era sensível a contexto.)

	Aplicação	Middleware	P1	P2	P3	P4
A1						
A2						
A3						
A4						

Se sim, utilizaram quais ferramentas de desenvolvimento?

	Ferramenta	P1	P2	P3	P4
F1					
F2					
F3					
F4					

4.Qual o tempo de experiência de cada participante nas linguagens NCL e Lua?

P1: _____ anos e _____ meses para NCL e _____ anos e _____ meses para Lua

P2: _____ anos e _____ meses para NCL e _____ anos e _____ meses para Lua

P3: _____ anos e _____ meses para NCL e _____ anos e _____ meses para Lua

P4: _____ anos e _____ meses para NCL e _____ anos e _____ meses para Lua

5.Através de quais meios, vocês já tomaram conhecimento do conceito de sensibilidade a contexto?

	Meio	P1	P2	P3	P4
M1					
M2					
M3					
M4					

6.Vocês já utilizaram quais aplicações sensíveis a contexto?

	Aplicação	P1	P2	P3	P4
A1					
A2					
A3					
A4					

Tempos dos Passos

Tabela B.1: Tempos dos Passos.

Passo	**Tempo**
1. Com base no cenário da aplicação, reconhecer quais elementos fornecerão sensibilidade a contexto para a aplicação.	__ h __ min
2. Criar projeto, telas da aplicação e os seus componentes.	__ h __ min
3. Criar os *personas* e inserir as suas características contextuais.	__ h __ min
4. Definir qual o ponto inicial (tela e componente iniciais) de cada *persona* e informar suas transições.	__ h __ min
5. Gerar o código da aplicação e executá-la no Ginga-NCL Virtual STB.	__ h __ min

B.2 Ferramenta Contextual Ginga

1.Quais os pontos positivos da ferramenta?

(a) _____

(b) _____

(c) _____

(d) _____

(e) _____

2.A ferramenta permitiu que fossem cometidos erros na construção da aplicação?

□ Não □ Sim. Quais?

(a) _____

(b) _____

(c) _____

3.A adição de quais elementos contextuais poderia tornar as aplicações geradas mais interessantes?

(a) _____

(b) _____

(c) _____

4.Quais funcionalidades podem ser adicionadas na ferramenta?

(a) _____

(b) _____

(c) _____

5.Quais funcionalidades precisam ser corrigidas na ferramenta?

(a) Funcionalidade: _____

Problema: _____

(b) Funcionalidade: _____

Problema: _____

(c) Funcionalidade: _____

Problema: _____

B.3 Aplicação Gerada

1. Compare as semelhanças e diferenças entre a representação da tela na ferramenta e a exibida no Ginga-NCL Virtual STB.

2. O grupo considera que conseguiu atingir o objetivo de produzir uma aplicação de TV digital sem precisar codificar e sem ter conhecimento das linguagens NCL e Lua? □ Sim □ Não. Por quê (em ambas as respostas)?

APÊNDICE C – Biblioteca Contextual Ginga Lua

Neste apêndice são apresentados os arquivos da *Biblioteca Contextual Ginga Lua* desenvolvida para implementar funcionalidades comuns às aplicações geradas pela ferramenta Contextual Ginga. A biblioteca simplifica o código das aplicações através do reuso de representações de objetos e funções em Lua.

As aplicações geradas invocam a biblioteca para a funcionalidade de desenho dos componentes gráficos, e para funções auxiliares de tratamento do contexto do usuário, de registro das ações tomadas pelo usuário e de gerenciamento dos *scripts* Lua. Por isto, os arquivos da biblioteca estão divididos em dois grupos: um para o tratamento de interface gráfica, e outro que contém as funções auxiliares. As seções a seguir descrevem cada um destes dois grupos.

C.1 Interface Gráfica

Neste pacote estão presentes os *scripts* Lua responsáveis pelo desenho de telas e de seus componentes. Com exceção do script *Screen*, os demais simulam o conceito de classe de linguagem orientada a objeto.

Nos Códigos C.1 e C.2, pode ser visto o código que trata do desenho de telas. Da linha 10 à linha 13, a função que é chamada para limpar uma tela antes de desenhar outra.

Código C.1: Arquivo Screen.lua (Parte 1).

```
1 --============================================================
2 -- Universidade Federal de Pernambuco
3 -- Centro de Informática
4 --============================================================
5
6 ------------------------------------------------------------
7 --[=[ This function cleans the screen and it makes
8 possible to draw another screen. ]=]
```

Código C.2: Arquivo Screen.lua (Parte 2).

```
9
10 function cleanScreen (x, y, width, height)
11         canvas:attrColor(0, 0, 0, 0);
12         canvas:drawRect('fill', x, y, width, height);
13 end
```

No Código C.3, pode ser visto o código que representa a classe *Component*. Na linha 6, a invocação para o gerenciador de scripts executar o script *Color*. Na linha 8, a tabela que armazena os valores dos atributos de um componente. Da linha 11 à linha 16, os atributos que são comuns a todos os tipos de componentes. Da linha 20 à linha 23, a função que representa o construtor da classe. Na linha 25, a criação do módulo Lua *Component*.

Código C.3: Arquivo Component.lua.

```
1  --=============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  --=============================================================
5
6  ScriptManager:execute('framework.gui.Color');
7
8  Component = {};
9  mt_Component = {};
10
11 Component.name = '';
12 Component.x = 0;
13 Component.y = 0;
14 Component.width = 0;
15 Component.height = 0;
16 Component.background = nil;
17
18 mt_Component.__index = Component;
19
20 function Component:new()
21         local ret = {};
22         return setmetatable(ret, mt_Component);
23 end
24
25 module("Component", package.seeall);
```

Nos Códigos C.4 e C.5, pode ser visto o código que representa a classe *Text*, que herda os atributos da classe *Component*. Nas linhas 6 e 7, a invocação para o gerenciador de scripts executar os scripts *Component* e *Font*. Na linha 9, a atribuição de um novo *Component* à *Text*, criando a relação de herança entre as duas classes. Nas linhas 12 e 13, os atributos que são específicos de um componente do tipo *Text*. Da linha 17 à linha 20, a função que representa o construtor da classe.

A forma de desenhar componentes do tipo *Text* depende da quantidade de linhas presentes no texto (linhas 24 e 25). Para cada linha, é chamada a sequencia de operações a seguir. Da linha 22 à linha 47 a função que desenha um *Text*, ou seja, atribui o tipo de fonte (linhas 30 e 31), atribui a cor do *background* (linhas 32 e 33), atribui as dimensões e a posição (linhas 34 e 35), atribui a cor da fonte (linhas 36 e 37), atribui o texto (linhas 38 e 39) e atualiza o *canvas* depois das operações de desenho (linha 45).

A quantidade de linhas presentes no texto é obtida através da função *getTextLines* (da linha 49 à 68). Ela divide o parâmetro *text* através do caracter de escape "\n" e retorna uma tabela com cada linha do texto.

Código C.4: Arquivo Text.lua (Parte 1).

```
1  --=============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  --=============================================================
5
6  ScriptManager:execute('framework.gui.Component');
7  ScriptManager:execute('framework.gui.Font');
8
9  Text = Component.new();
10 mt_Text = {}
11
12 Text.text = '';
13 Text.font = nil;
14
15 mt_Text.__index = Text;
16
17 function Text:new(o)
18         local ret = o or {};
19         return setmetatable(ret, mt_Text);
20 end
21
22 function Text:draw()
23
24         lines = getTextLines (self.text);
25         height = math.floor(self.height / #lines);
26         i = 1;
27         line = lines[i];
28
29         while line ~= nil do
30                 canvas:attrFont(self.font.name, self.font.size,
31                         self.font.style);
32                 canvas:attrColor(self.background.r, self.background.g,
33                         self.background.b, self.background.alpha);
34                 canvas:drawRect('fill', self.x, self.y + height * (i - 1),
35                         self.width, height);
36                 canvas:attrColor(self.font.color.r, self.font.color.g,
37                         self.font.color.b, self.font.color.alpha);
```

Código C.5: Arquivo Text.lua (Parte 2).

```
38                  canvas:drawText(self.x, self.y + height * (i - 1),
39                          lines[i]);
40
41                  i = i + 1;
42                  line = lines[i];
43          end
44
45          canvas:flush();
46
47  end
48
49  function getTextLines (text)
50
51          lines = {};
52          count = 1;
53          s = text;
54          i = string.find(s, '\n');
55
56          while i ~= nil do
57                  line = string.sub(s, 0, i-1);
58                  s = string.sub(s, i+1, string.len(s));
59                  i = string.find(s, '\n');
60
61                  lines[count] = line;
62                  count = count + 1;
63          end
64
65          lines[count] = s;
66
67          return lines;
68  end
```

Nos Códigos C.6 e C.7, pode ser visto o código que representa a classe *Image*, que herda os atributos da classe *Component*. Na linha 6, a invocação para o gerenciador de scripts executar o script *Component*. Na linha 8, a atribuição de um novo *Component* à *Image*, criando a relação de herança entre as duas classes. Nas linhas 11 e 12, os atributos que são específicos de um componente do tipo *Image*. Da linha 16 à linha 19, a função que representa o construtor da classe. Da linha 21 à linha 32 a função que desenha um *Image*, dependendo de se a imagem tem foco (linha 25) ou não (linha 27). Esta função também atribui à imagem uma posição (linha 30) e atualiza o *canvas* depois das operações de desenho (linha 31).

Código C.6: Arquivo Image.lua (Parte 1).

```
1  --=============================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
```

Código C.7: Arquivo Image.lua (Parte 2).

```
4  --=============================================================
5
6  ScriptManager:execute('framework.gui.Component');
7
8  Image = Component.new();
9  mt_Image = {}
10
11 Image.location = '';
12 Image.locationOnFocus = '';
13
14 mt_Image.__index = Image;
15
16 function Image:new(o)
17         local ret = o or {};
18         return setmetatable(ret, mt_Image);
19 end
20
21 function Image:draw(onFocus)
22         image = nil;
23
24         if(onFocus == true) then
25                 image = canvas:new(self.locationOnFocus);
26         else
27                 image = canvas:new(self.location);
28         end
29
30         canvas:compose(self.x, self.y, image);
31         canvas:flush();
32 end
```

No Código C.8 pode ser visto o código que representa a classe *Color*. Na linha 6, a tabela que armazena os valores dos atributos de uma cor. Nas linhas 8 e 9, os atributos de uma cor. Da linha 12 à linha 15, a função que representa o construtor da classe.

Código C.8: Arquivo Color.lua.

```
1  --=============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  --=============================================================
5
6  Color = {};
7  mt_Color = {};
8  Color.r = 0; Color.g = 0; Color.b = 0;
9  Color.alpha = 255;
10 mt_Color.__index = Color;
11
12 function Color:new(o)
13         local ret = o or {};
14         return setmetatable(ret, mt_Color);
15 end
```

No Código C.9 pode ser visto o código que representa a classe *Font*. Na linha 6, a tabela que armazena os valores dos atributos de uma fonte. Nas linhas 8 e 9, os atributos de uma fonte. Da linha 12 à linha 15, a função que representa o construtor da classe.

Código C.9: Arquivo Font.lua.

```
1  --============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  --============================================================
5
6  Font = {};
7  mt_Font = {};
8  Font.name = ''; Font.style = 0;
9  Font.size = 0; Font.color = nil;
10 mt_Font.__index = Font;
11
12 function Font:new(o)
13         local ret = o or {};
14         return setmetatable(ret, mt_Font);
15 end
```

C.2 Útil

Neste pacote estão presentes os *scripts* Lua responsáveis por auxiliar o código principal da aplicação NCLua. Nos Códigos C.10 e C.11, pode ser visto o código do gerenciador de *scripts* Lua. Na linha 7, a tabela que armazena os nomes dos arquivos Lua que já foram executados. Da linha 9 à linha 17, a função que é chamada para executar um *script* Lua, que chama a função *require* para carregar um módulo apenas se ele já não tiver sido carregado.

Código C.10: Arquivo ScriptManager.lua (Parte 1).

```
1  --============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  --============================================================
5
6  ScriptManager = {};
7  ScriptManager.executedScripts = {};
8
9  function ScriptManager:execute (fileName)
10
11         if fileName ~= nil and
12                 ScriptManager.executedScripts[fileName] == nil then
13
14                 require(fileName);
```

Código C.11: Arquivo ScriptManager.lua (Parte 2).

```
15                  ScriptManager.executedScripts[fileName] = true;
16          end
17  end
18
19  module("ScriptManager", package.seeall);
```

Nos Códigos C.12, C.13, C.14 e C.15, pode ser visto o código que trata da avaliação de se as informações contextuais do usuário correspondem aos valores passados como argumentos para cada função.

No Código C.12, da linha 10 à 26, é exibida a função de avaliação para a categoria *Who*, comparando-se a faixa etária e o sexo do usuário. Se o persona não tem uma destas informações, a comparação com o contexto do usuário não é executada.

Código C.12: Arquivo Persona.lua (Parte 1).

```
1  --=============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  --=============================================================
5
6  ---------------------------------------------------------------
7  --[=[ This function evaluates if user context variables
8  fit in persona's who. ]=]
9
10 function persona_evaluateWho (ageFrom, ageTo, genre)
11         who = true;
12
13         if ageFrom ~= -1 and userContext.age ~= nil then
14                 who = who and (userContext.age >= ageFrom);
15         end
16
17         if ageTo ~= -1 and who ~= false and userContext.age ~= nil then
18                 who = who and (userContext.age <= ageTo);
19         end
20
21         if genre ~= -1 and who ~= false and userContext.genre ~= nil then
22                 who = who and (userContext.genre == genre);
23         end
24
25         return who;
26 end
```

No Código C.13, da linha 31 à 61, é exibida a função de avaliação para a categoria "quando", comparando-se a data e a hora do parâmetro *time* com os parâmetros *startDate*, *startHour*, *endDate* e *endHour*, e o dia da semana.

Código C.13: Arquivo Persona.lua (Parte 2).

```
27 ---------------------------------------------------------
28 --[=[ This function evaluates if user context variables
29 fit in persona's when. ]=]
30
31 function persona_evaluateWhen (startDate, startHour,
32         endDate, endHour, day, time)
33
34         when = true;
35
36         if startDate ~= nil then
37                 when = when and time_isTime1AfterEqualsTime2(time,
38                         time_getTimeTable(startDate, true), true);
39         end
40
41         if startHour ~= nil and when ~= false then
42                 when = when and time_isTime1AfterEqualsTime2(time,
43                         time_getTimeTable(startHour, false), false);
44         end
45
46         if endDate ~= nil and when ~= false then
47                 when = when and time_isTime1BeforeEqualsTime2(time,
48                         time_getTimeTable(endDate, true), true);
49         end
50
51         if endHour ~= nil and when ~= false then
52                 when = when and time_isTime1BeforeEqualsTime2(time,
53                         time_getTimeTable(endHour, false), false);
54         end
55
56         if day ~= nil then
57                 when = when and ((time_getDayOfWeek(time)) == day);
58         end
59
60         return when;
61 end
```

Nos Códigos C.14 e C.15, da linha 67 à 88, é exibida a função de avaliação para a categoria "onde", comparando-se o país, o estado, a cidade e o código postal do usuário. Se o persona não tem uma destas informações, a comparação com o contexto do usuário não é executada.

Código C.14: Arquivo Persona.lua (Parte 3).

```
62
63 ---------------------------------------------------------
64 --[=[ This function evaluates if user context variables
65 fit in persona's where. ]=]
66
67 function persona_evaluateWhere (country, state, city, postalCode)
68         where = true;
```

Código C.15: Arquivo Persona.lua (Parte 4).

```
69
70          if country ~= nil and userContext.country ~= nil then
71                  where = where and (userContext.country == country);
72          end
73
74          if state ~= nil and where ~= nil and userContext.state ~= nil then
75                  where = where and (userContext.state == state);
76          end
77
78          if city ~= nil and where ~= nil and userContext.city ~= nil then
79                  where = where and (userContext.city == city);
80          end
81
82          if postalCode ~= nil and where ~= nil and
83                  userContext.postalCode ~= nil then
84                  where = where and (userContext.postalCode == postalCode);
85          end
86
87          return where;
88  end
```

Nos Códigos C.16, C.17, C.18 e C.19 pode ser visto o código que trata de funções de manipulação de data e hora. No Código C.16, da linha 11 à 19, a função *time_completeWithZeros* que completa um número com zeros à esquerda até que a cadeia de caracteres atinja um determinado tamanho. Esta função é utilizada na função *time_getTimeString*, auxiliando na formatação da data no tipo *string*.

Código C.16: Arquivo Time.lua (Parte 1).

```
1  --=============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  --=============================================================
5
6  -------------------------------------------------------------
7  --[=[ This function returns a string with size equals to
8  length argument and with value equals to number
9  argument. ]=]
10
11 function time_completeWithZeros(number, lenght)
12          s = number;
13          if number >= 0 then
14                  while string.len(s) < lenght do
15                          s = '0' .. s;
16                  end
17          end
18          return s;
19 end
```

No Código C.17, da linha 25 à 34, a função *time_getTimeString* que recebe como parâmetro uma data na representação de tabela e retorna a mesma data no tipo *string*. Da linha 42 à 58, a função *time_getTimeTable* que recebe uma data ou hora através do parâmetro *timeString* e um booleano para indicar se esta *string* é uma data ou não. Neste caso, a *string* é uma hora. Esta função retorna uma tabela com os campos dia, mês, ano, hora e minuto. Se *timeString* for uma data, os campos hora e minuto são valores *default*. Se não for, os campos dia, mês e ano são valores *default*.

Código C.17: Arquivo Time.lua (Parte 2).

```
20
21 ---------------------------------------------------
22 --[=[ This function gets a time in table structure and
23 returns a string with fields yearMonthDay_hourMinSec. ]=]
24
25 function time_getTimeString(timeTable)
26         time = time_completeWithZeros(timeTable.year, 4) ..
27                 time_completeWithZeros(timeTable.month, 2) ..
28                 time_completeWithZeros(timeTable.day, 2) .. '_' ..
29                 time_completeWithZeros(timeTable.hour, 2) ..
30                 time_completeWithZeros(timeTable.min, 2) ..
31                 time_completeWithZeros(timeTable.sec, 2);
32
33         return time;
34 end
35
36 ---------------------------------------------------
37 --[=[ This function gets a time in string format and returns a table with
38 fields year, month, day, hour and min. If isDate is true, hour and minute
39 are default values. If isDate is false, day, month and year are default
40 values. ]=]
41
42 function time_getTimeTable(timeString, isDate)
43         time = {year = 2000, month = 1, day = 1, hour = 0, min = 0};
44
45         if timeString ~= nil then
46
47                 if isDate == true then
48                         time.day   = string.sub(timeString, 1, 2);
49                         time.month = string.sub(timeString, 4, 5);
50                         time.year  = string.sub(timeString, 7, 10);
51                 else
52                         time.hour  = string.sub(timeString, 1, 2);
53                         time.min   = string.sub(timeString, 4, 5);
54                 end
55         end
56
57         return time;
58 end
```

No Código C.18, da linha 63 à 76, a função *time_getDayOfWeek* que calcula o dia da semana de uma data de acordo com a fórmula apresentada por Tøndering (2008). Da linha 84 (Código C.18) à 112 (Código C.19), a função *time_isTime1AfterEqualsTime2* que compara duas datas ou duas horas, dependendo do booleano *isDate*, para verificar se a primeira é posterior ou igual à segunda.

No Código C.19, da linha 119 à 148, a função *time_isTime1BeforeEqualsTime2* que compara duas datas ou duas horas, dependendo do booleano *isDate*, para verificar se a primeira é anterior ou igual à segunda.

Código C.18: Arquivo Time.lua (Parte 3).

```
59 ------------------------------------------------------
60 --[=[ This function returns the day of week of timeTable:
61 Sunday, Monday, Tuesday, Wednesday, Thursday, Friday.
62
63 function time_getDayOfWeek (timeTable)
64
65         days = {'Sunday', 'Monday', 'Tuesday', 'Wednesday',
66                 'Thursday', 'Friday', 'Saturday'};
67
68         a = math.floor((14 - timeTable.month) / 12);
69         y = timeTable.year - a;
70         m = timeTable.month + 12*a - 2;
71         q = timeTable.day + math.floor(31*m/12) + y + math.floor(y/4) -
72                 math.floor(y/100) + math.floor(y/400);
73         d = math.fmod(q, 7);
74
75         return days[d+1];
76 end
77
78 ------------------------------------------------------
79 --[=[ This function evaluates if timeTable1 is after timeTable2 or if timeTable1
80 is equals to timeTable2. If isDate is true, hour and minute are not compared
81 because they are setted to be equals. If isDate is false, day, month and year
82 are not compared because they are setted to be equals. ]=]
83
84 function time_isTime1AfterEqualsTime2(timeTable1, timeTable2, isDate)
85         isAfter = false;
86         time = {};
87
88         if isDate == false then
89                 time.day = 1;
90                 time.month = 1;
91                 time.year = 2000;
92                 time.hour = timeTable1.hour;
93                 time.min = timeTable1.min;
94         else
95                 time.day = timeTable1.day;
96                 time.month = timeTable1.month;
```

Código C.19: Arquivo Time.lua (Parte 4).

```
97                  time.year = timeTable1.year;
98                  time.hour = 0;
99                  time.min = 0;
100         end
101
102         time.sec = 0;
103         timeTable2.sec = 0;
104
105         diff = os.difftime (os.time(time), os.time(timeTable2));
106
107         if(diff > 0 or diff == 0) then
108                 isAfter = true;
109         end
110
111         return isAfter;
112 end
113
114 --[=[ This function evaluates if timeTable1 is before timeTable2 or if
115 timeTable1 is equals to timeTable2. If isDate is true, hour and minute are not
116 compared because they are setted to be equals. If isDate is false, day, month
117 and year are not compared because they are setted to be equals.]=]
118
119 function time_isTime1BeforeEqualsTime2(timeTable1, timeTable2, isDate)
120
121         isBefore = false;
122         time = {};
123
124         if isDate == false then
125                 time.day = 1;
126                 time.month = 1;
127                 time.year = 2000;
128                 time.hour = timeTable1.hour;
129                 time.min = timeTable1.min;
130         else
131                 time.day = timeTable1.day;
132                 time.month = timeTable1.month;
133                 time.year = timeTable1.year;
134                 time.hour = 0;
135                 time.min = 0;
136         end
137
138         time.sec = 0;
139         timeTable2.sec = 0;
140
141         diff = os.difftime (os.time(time), os.time(timeTable2));
142
143         if(diff < 0 or diff == 0) then
144                 isBefore = true;
145         end
146
147         return isBefore;
148 end
```

No Código C.20 pode ser visto o código que trata do registro das ações tomadas pelo usuário da aplicação. Durante o tratamento de um evento, são chamadas as funções *log_registerTransition1* e *log_registerTransition2* para armazenar em arquivo a transição realizada.

Código C.20: Arquivo Log.lua.

```
1  --=============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  --=============================================================
5
6  ------------------------------------------------------------
7  --[=[ This function writes in argument file: current date,
8  transitionId, persona, fromScreen and fromComponent. ]=]
9
10 function log_registerTransition1(file, transitionId, persona,
11         fromScreen, fromComponent)
12
13         file:write(date_getStringDate(os.date('*t')),
14                 ' : (\"', transitionId, '\") persona = \"');
15         file:write(persona, '\" fromScreen = \"', fromScreen,
16                 '\" fromComponent = \"', fromComponent);
17 end
18
19 ------------------------------------------------------------
20 --[=[ This function writes in argument file: toScreen,
21 toComponent and action. ]=]
22
23 function log_registerTransition2(file, toScreen, toComponent, action)
24
25         file:write('\" toScreen = \"', toScreen,
26                 '\" toComponent = \"', toComponent,
27                 '\" action = \"', action, '\"\n');
28 end
```

APÊNDICE D – Manual do Usuário

Este apêndice tem como objetivo orientar o usuário na execução das funcionalidades oferecidas pela ferramenta *Contextual Ginga*. Neste capítulo, são listados os requisitos necessários para a instalação, como esta deve ser realizada e como desinstalá-la do computador. Na Seção D.3, são descritos os procedimentos para a execução de cada funcionalidade. Por fim, Na Seção D.4, são descritas instruções para a execução da aplicação de TV digital gerada pela ferramenta.

D.1 Requisitos

Os requisitos mínimos de hardware, para executar a ferramenta *Contextual Ginga* são:

1.Memória RAM: 256 MB

2.Processador: AMD Turion 1.8 GHz ou equivalente

Como requisito de software, a ferramenta precisa que esteja instalada na máquina a *Java Runtime Environment* (JRE) da Sun Microsystems versão 1.6 ou superior (MICROSYSTEMS, 2006). Dessa forma, a ferramenta pode ser executada em qualquer sistema operacional para o qual exista uma implementação da JRE, em função da característica de portabilidade desta. Mas, é válido destacar que a ferramenta foi executada apenas no sistema operacional Windows, versões XP e Vista. A ferramenta é melhor visualizada com resolução do monitor igual ou superior a 1280 por 720 pixels.

D.2 Instalação e Desinstalação

Como a ferramenta foi desenvolvida na linguagem Java, a ferramenta não precisa ser instalada. É suficiente executar o arquivo *ContextualGinga.jar*. Para remover a ferramenta, é suficiente excluir o arquivo *ContextualGinga.jar* do disco.

D.3 Funcionalidades

Nesta seção serão descritos os passos necessários para a execução das funcionalidades proporcionadas pela ferramenta *Contextual Ginga*. Nesta, o conjunto de elementos gráficos que devem formar uma aplicação devem estar representados na ferramenta dentro de um mesmo projeto. Estes elementos são as telas que reúnem um conjunto de componentes que são exibidos por vez. As telas são interligadas através de transições que determinam os possíveis fluxos entre as telas que podem ser visualizadas. Os possíveis fluxos dependem do contexto do usuário da aplicação.

A Figura D.1 exibe uma imagem da ferramenta com suas principais áreas destacadas. Na área 1, são exibidos o nome do projeto e as telas pertencentes a ele. Na área 2, é exibida a tela selecionada ou a navegação entre telas para o *persona* selecionado. Na área 3, são exibidas as propriedades e seus valores para um componente selecionado. Na área 4, são exibidos os tipos de componentes que podem ser adicionados às telas. Na área 5, é exibida a lista dos *personas* do projeto. Por fim, na área 6, são exibidas as três categorias (*who*, *when* e *where*) das características de contexto que podem ser alteradas para o *persona* selecionado.

As funcionalidades estão agrupadas pelos conceitos de projeto, de tela, de componente, de *persona*, de transição e de geração de código NCLua. As funcionalidades diretamente relacionadas a contexto são as dos conceitos de *persona* e de transição.

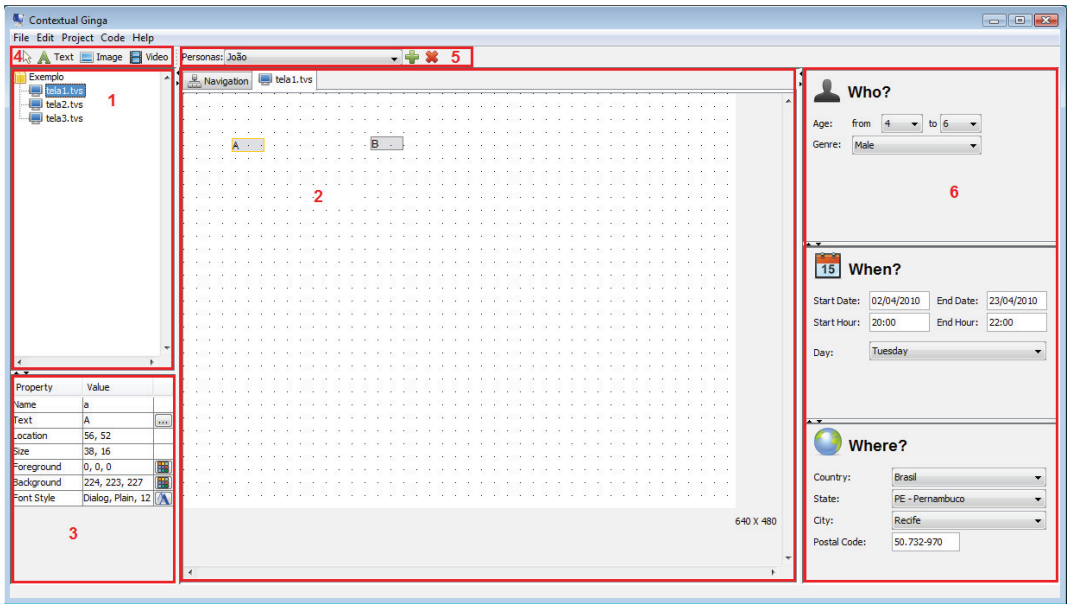


Figura D.1: *Contextual Ginga.*

D.3.1 Projeto

Esta subseção descreve as funcionalidades referentes ao conceito de projeto.

Criar Projeto

Esta funcionalidade permite agrupar um conjunto de telas, numa determinada resolução, para formar uma aplicação de TV digital. Para executá-la, o usuário deve, no menu *File*, submenu *New*, clicar no item *Project* (Figura D.2) ou digitar o atalho *Ctrl+P*. Na janela aberta, informar o diretório onde os arquivos do projeto devem ser armazenados e clicar no botão *Save* (Figura D.3). Na outra janela aberta, selecionar a resolução desejada para as telas do projeto e clicar no botão *Finish* (Figura D.4).

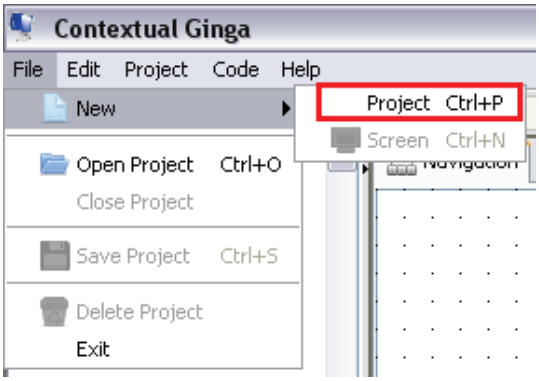


Figura D.2: Criação de um novo projeto.

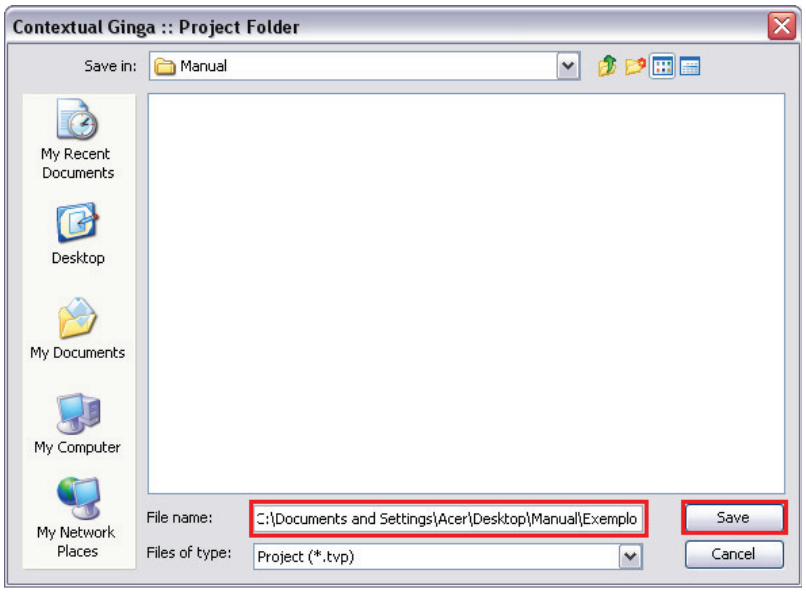


Figura D.3: Diretório e nome do projeto.

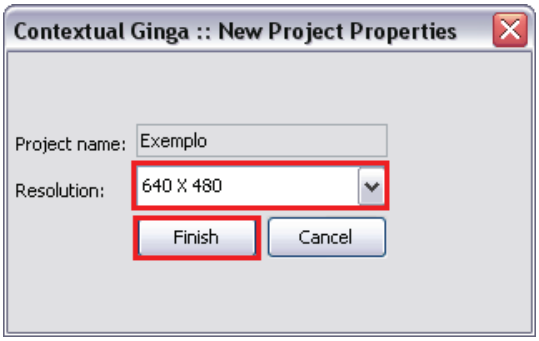


Figura D.4: Resolução das telas de um projeto.

Abrir Projeto

Esta funcionalidade permite visualizar todos os dados de um projeto e trabalhar nele. Para executá-la, o usuário deve, no menu *File*, clicar no item *Open Project* (Figura D.5) ou digitar o atalho *Ctrl+O*. Na janela aberta, selecionar o arquivo *.tvp* que representa o projeto que deve ser aberto e clicar no botão *Open* (Figura D.6).

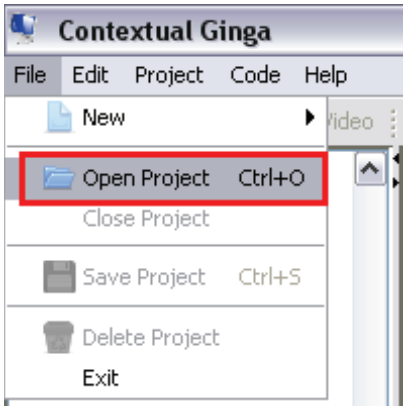


Figura D.5: Abrir projeto.

Salvar Projeto

Esta funcionalidade permite que sejam salvos todos os dados do projeto, ou seja, todas as telas com seus componentes e transições, bem como os *personas*. Para executá-la, o usuário deve, no menu *File*, clicar no item *Save Project* (Figura D.7) ou digitar o atalho *Ctrl+S* (Figura D.7).

Fechar Projeto

Esta funcionalidade permite encerrar a visualização dos dados de um projeto. Para executá-la, o usuário deve, no menu *File*, clicar no item *Close Project* (Figura D.8).

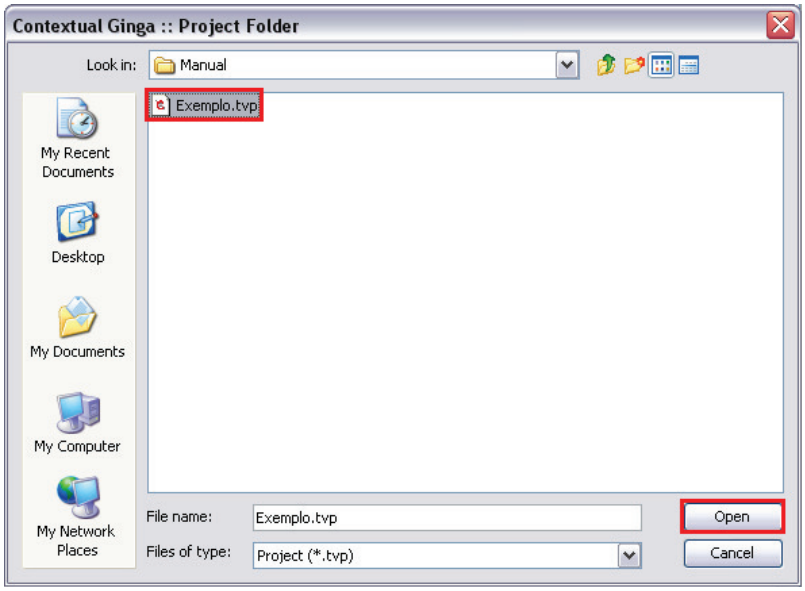


Figura D.6: Seleção de projeto.

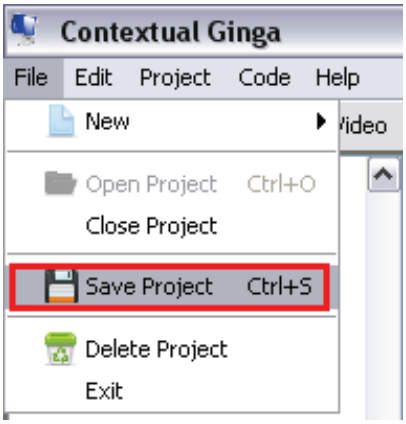


Figura D.7: Salvar projeto.

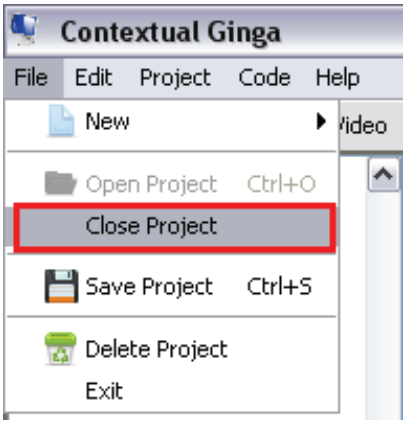


Figura D.8: Fechar projeto.

Excluir Projeto

Esta funcionalidade permite excluir o projeto que está aberto. Para executá-la, o usuário deve, no menu *File*, clicar no item *Delete Project* (Figura D.9). Na janela de confirmação de exclusão, o usuário deve clicar no botão *Yes* (Figura D.10).

D.3.2 Tela

Esta subseção descreve as funcionalidades referentes ao conceito de tela.

Criar Tela

Esta funcionalidade permite adicionar uma tela ao projeto. Para executá-la, o usuário

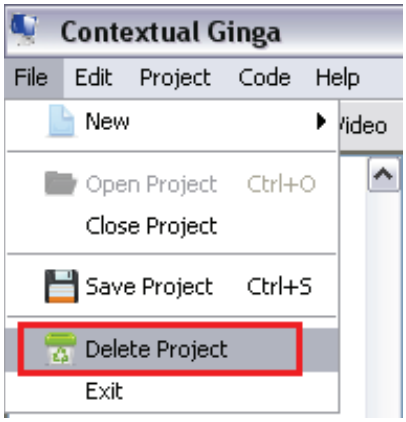


Figura D.9: Excluir projeto.

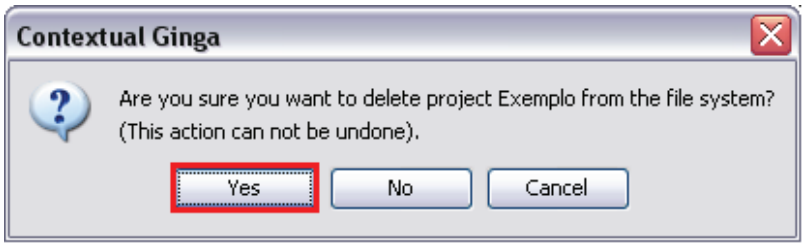


Figura D.10: Confirmar exclusão do projeto.

deve, no menu *File*, submenu *New*, clicar no item *Screen* (Figura D.11) ou digitar o atalho *Ctrl+N*. Em seguida, deve informar o nome para a nova tela e clicar no botão *Finish* (Figura D.12).

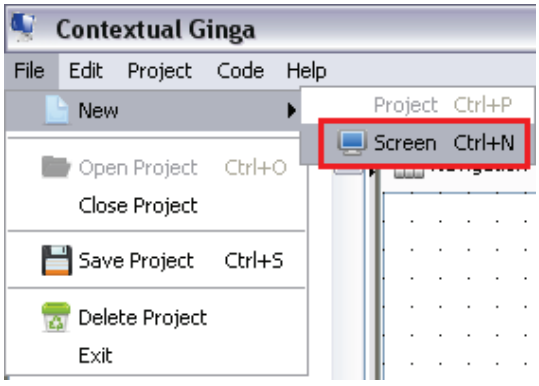


Figura D.11: Criar tela.

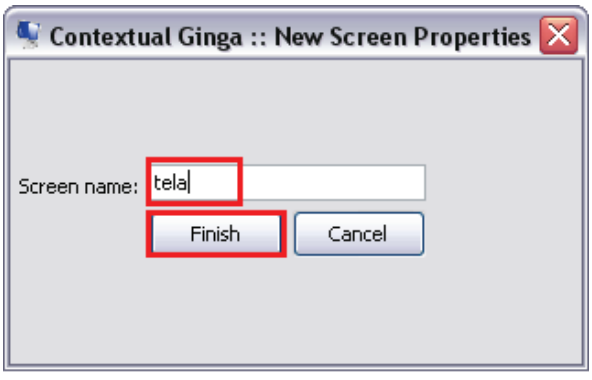


Figura D.12: Nome da tela.

Listar Telas

Esta funcionalidade permite que sejam exibidos os nomes de todas as telas pertencentes ao projeto. Sempre que o usuário está com um projeto aberto, são listados os nomes das telas deste projeto (Figura D.13).

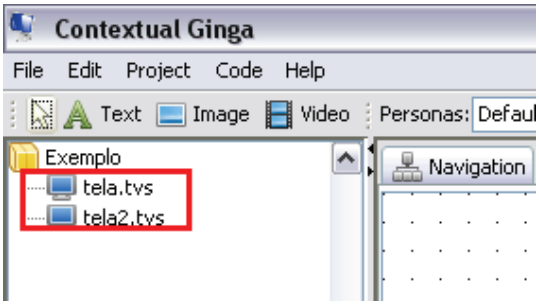


Figura D.13: Listar telas.

Exibir Tela em Editor Gráfico

Esta funcionalidade permite que as telas e seus componentes sejam exibidos em formato gráfico. Apenas uma tela pode ser visualizada por vez. Para executar esta funcionalidade, o usuário deve clicar com o botão esquerdo no nome de uma tela listada (Figura D.13). Em seguida, a ferramenta exibe a tela e seus componentes (Figura D.14).

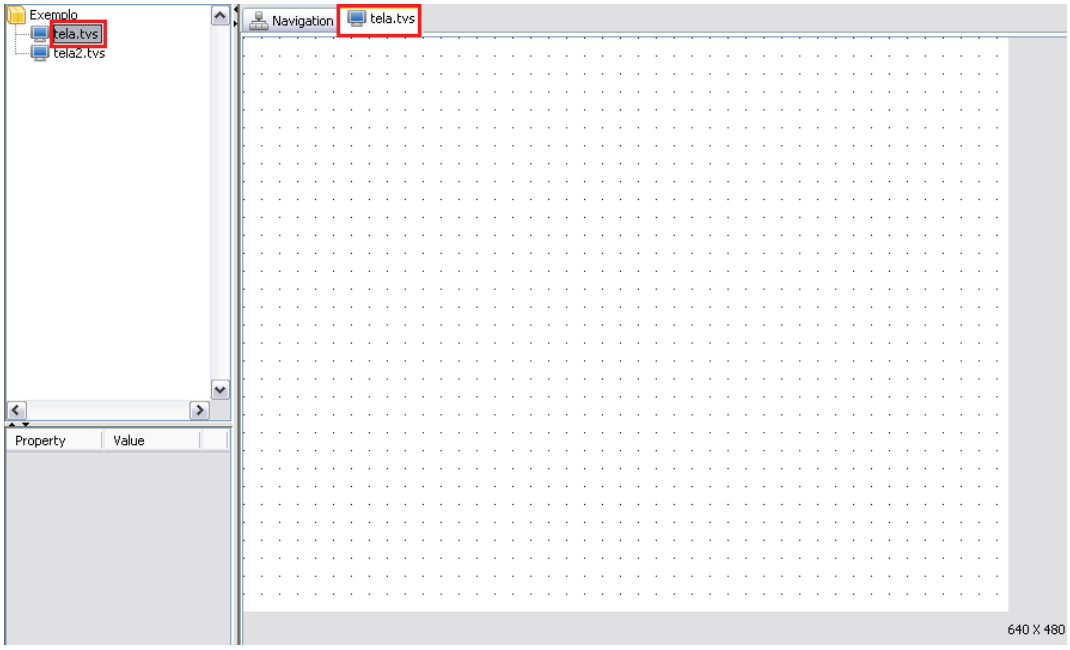


Figura D.14: Exibir tela em editor gráfico.

Renomear Tela

Esta funcionalidade permite alterar o nome de uma tela. Para executar esta funcionalidade, o usuário deve clicar com o botão esquerdo no nome da tela que deseja renomear para selecioná-la e em seguida com o direito para abrir a lista de itens. Depois, o usuário deve clicar no item *Rename Screen* (Figura D.15). Na janela aberta, o usuário deve informar o novo nome para a tela e clicar no botão *OK* (Figura D.16).

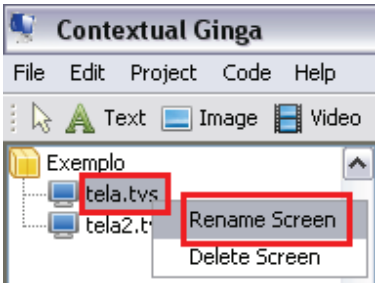


Figura D.15: Renomear tela.

Figura D.16: Novo nome da tela.

Excluir Tela

Esta funcionalidade permite excluir uma tela de um projeto. Com esta exclusão, também são excluídas as transições que se originam desta tela e que se destinam a mesma, como também os pontos iniciais (combinação de tela e componente iniciais) formados por esta tela. Para executar esta funcionalidade, o usuário deve clicar com o botão esquerdo no nome da tela que deseja excluir para selecioná-la e em seguida com o direito para abrir a lista de itens. Depois, o usuário deve clicar no item *Delete Screen* (Figura D.17). Na janela aberta, o usuário deve confirmar a exclusão, clicando no botão *Yes* (Figura D.18).

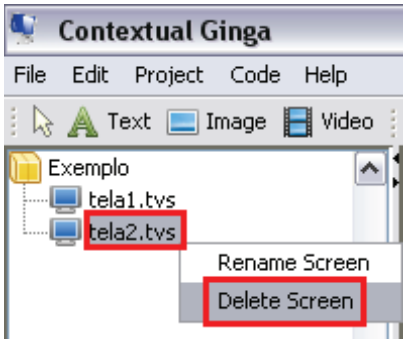


Figura D.17: Excluir tela.

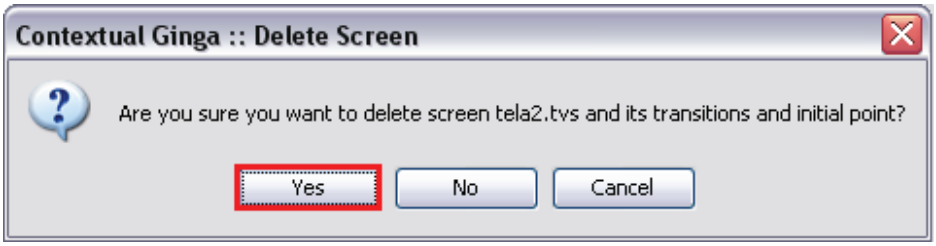


Figura D.18: Confirmar exclusão da tela.

D.3.3 Componente

Esta subseção descreve as funcionalidades referentes ao conceito de componente.

Adicionar Componente

Esta funcionalidade permite adicionar um novo componente à tela que está sendo exibida na interface da ferramenta. Para executar esta funcionalidade, o usuário deve clicar em um dos três tipos de componente disponíveis: *Text*, *Image* ou *Video* (Figura D.19).

Para os dois primeiros tipos de componentes, o usuário deve clicar em seguida, dentro da representação da tela, onde deseja que o componente seja adicionado. Na janela aberta, o usuário deve informar o nome do componente e clicar no botão *Finish* (Figura D.20). Além disso, se o componente for do tipo *Image*, na janela aberta, o usuário deve selecionar o arquivo da imagem que se deseja adicionar. O nome de um componente deve ser formado por uma cadeia de letras, dígitos e o caracter *underscore* "_"; contudo, a cadeia não deve começar com dígito.

Para definir o vídeo principal da aplicação gerada, que é o terceiro tipo de componente, o usuário deve clicar no botão *Video*. Na janela aberta, o usuário deve clicar no botão *Browse* para escolher qual o arquivo que representa o vídeo. Em seguida, deve clicar no botão *Close* (Figura D.21).

Figura D.19: Tipos de componente.

Figura D.20: Nome do componente.

Excluir Componente

Esta funcionalidade permite excluir um componente de uma tela. Quando o componente é excluído, ele deixa de existir na visão de todos os *personas* adicionados ao projeto. Para executar esta funcionalidade, o usuário deve clicar com o botão direito sobre o componente a ser excluído e clicar no item *Delete Component* (Figura D.22). Na janela aberta, clicar no botão *Yes* para confirmar a exclusão do componente (Figura D.23).

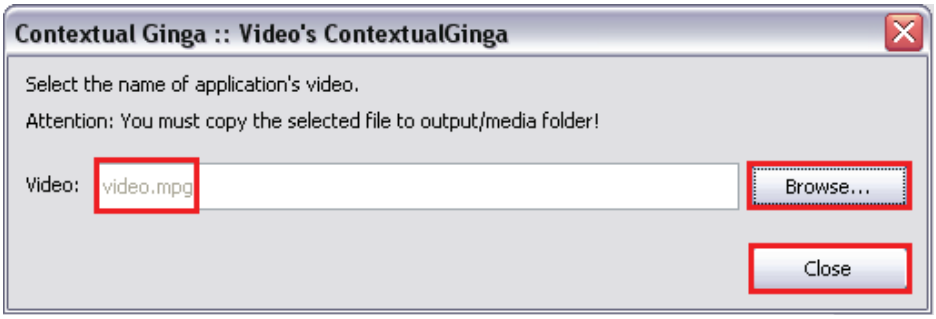


Figura D.21: Nome do vídeo principal.

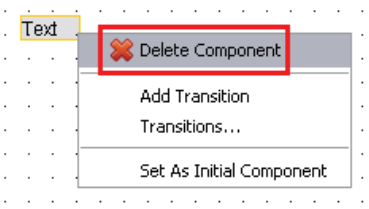


Figura D.22: Excluir componente.

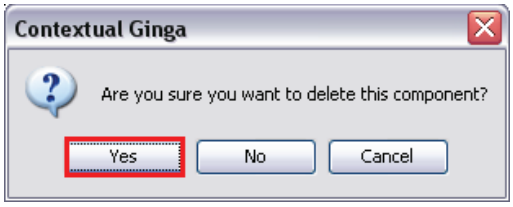


Figura D.23: Confirmar exclusão do componente.

Mover Componente

Esta funcionalidade permite mover graficamente um componente através de *drag-and-drop* dentro da área da tela. Para executar esta funcionalidade, o usuário deve clicar com o botão esquerdo no componente e arrastar o componente até a nova posição desejada. Quando chegar nesta posição, o usuário deve soltar o botão pressionado.

Definir Componente e Tela Iniciais

Esta funcionalidade permite determinar para cada *persona* do projeto, qual a primeira tela a ser exibida e qual o componente desta tela que receberá foco inicialmente, ou seja, dependendo das características do usuário que interage com a aplicação, esta começa a ser executada de maneira diferente.

Para executar esta funcionalidade, o usuário deve clicar no componente com o botão direito e clicar no item *Set As Initial Component* (Figura D.24). Na janela aberta para exibir a mensagem que o componente inicial foi definido, o usuário deve clicar no botão *OK* (Figura D.25).

Outra forma de executar esta funcionalidade é, selecionar o *persona* desejado, ir na aba *Navigation* e, então, clicar com o botão direito em cima do nome da tela. Em seguida, clicar no item *Set As Initial Screen* (Figura D.26). Na janela aberta, o usuário deve selecionar um dos componentes da tela para ser o primeiro a receber foco (Figura D.27).

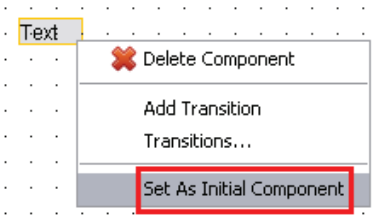


Figura D.24: Definir componente inicial.

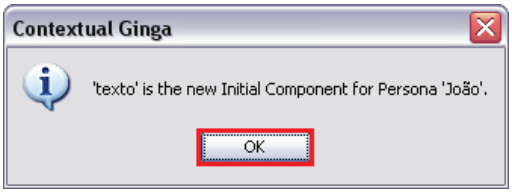


Figura D.25: Componente inicial definido.

Na janela aberta que mostra que a tela e o componente iniciais foram definidos, o usuário deve clicar no botão *OK* (Figura D.28).

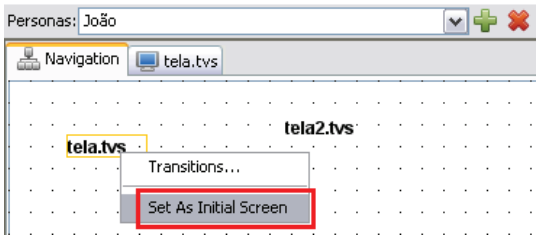


Figura D.26: Definir tela inicial.

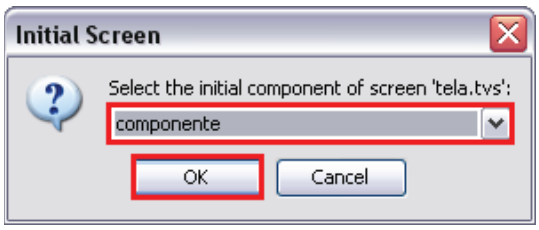


Figura D.27: Definir componente inicial.

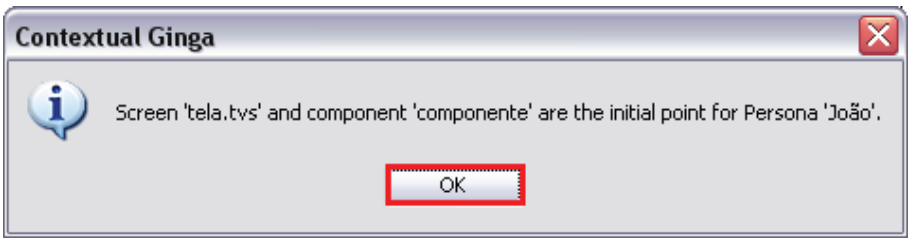


Figura D.28: Tela e componente inicial definidos.

Alterar Propriedade de Componente

Esta funcionalidade permite alterar o valor de uma propriedade de um componente a partir de uma lista de propriedades que variam de acordo com o tipo do componente. Para executar esta funcionalidade, o usuário deve clicar no componente e a ferramenta exibirá as respectivas propriedades e seus valores atuais.

As propriedades de um componente do tipo texto são: nome, texto, posição, dimensões, cor da fonte, cor do fundo e estilo da fonte (Figura D.29). Um componente deste tipo pode conter texto em uma ou mais linhas.

As propriedades de um componente do tipo imagem são: nome, posição, dimensões, imagem e imagem com foco (Figura D.30). Com esta possibilidade de permitir que um componente possua duas imagens juntamente com transições que permitem mudanças de

foco para uma dada ação, pode-se fazer que ele apresente o comportamento de um botão. Se o componente representar sempre a mesma imagem, é suficiente atribuir valor somente para imagem sem foco.

Property	Value
Name	texto
Text	Text
Location	53, 80
Size	38, 16
Foreground	0, 0, 0
Background	224, 223, 227
Font Style	Dialog, Plain, 12

Figura D.29: Propriedades de componente tipo texto.

Property	Value
Name	imagem
Location	115, 80
Size	38, 16
Image	C:\Document...
Image on Focus	

Figura D.30: Propriedades de componente tipo imagem.

Para alterar a propriedade *Name*, o usuário deve clicar duas vezes na célula da coluna *Value*, digitar o novo nome e apertar a tecla *Enter*. O mesmo pode ser feito para as propriedades *Location* e *Size*. Alterar o valor de *Location* é outra forma de mudar um componente de posição além do *drag-and-drop*.

Para alterar a propriedade *Text*, o usuário deve clicar no botão "..." ou na tecla F2. Na janela aberta, digitar o novo texto e clicar no botão *OK* (Figura D.31). Para alterar as propriedades *Foreground* e *Background*, o usuário deve clicar no botão que representa um diagrama de cores. Na janela aberta, selecionar a nova cor e clicar no botão *OK* (Figura D.32).

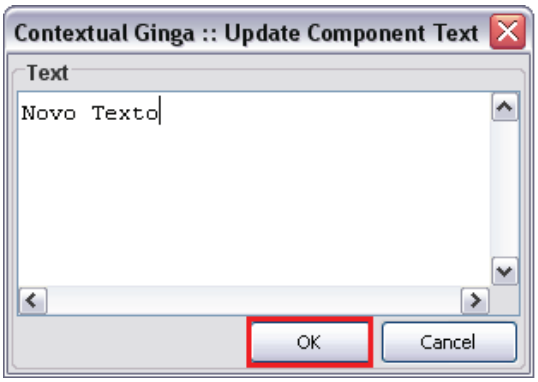


Figura D.31: Alterar propriedade de texto.

Para alterar a propriedade *Font Style*, o usuário deve clicar no botão que representa letras. Na janela aberta, selecionar o novo estilo de fonte e clicar no botão *OK* (Figura D.33). Por fim, para alterar as propriedades *Image* e *Image on Focus*, o usuário deve

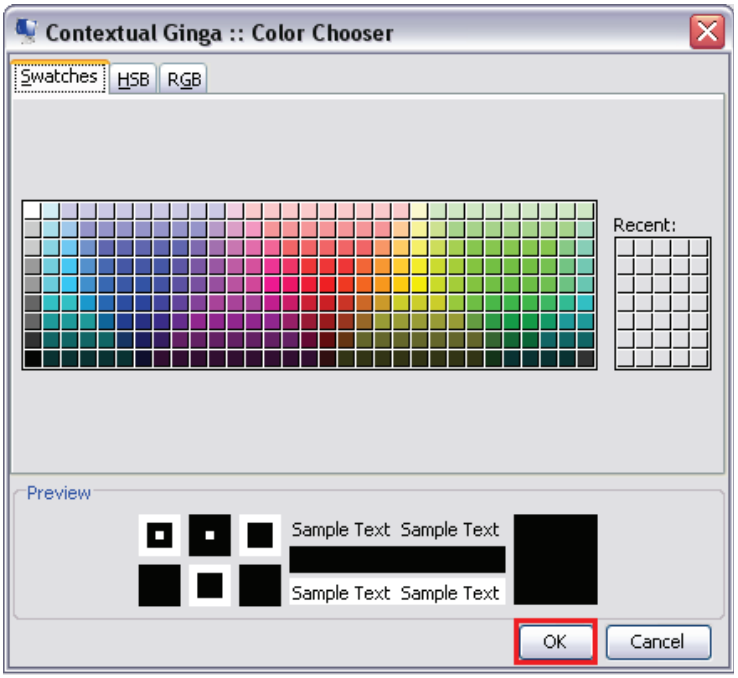


Figura D.32: Alterar propriedade de cor.

clicar no botão que representa a abertura de uma pasta. Na janela aberta, selecionar o novo arquivo da imagem e clicar no botão *OK*.

D.3.4 Persona

Esta subseção descreve as funcionalidades referentes ao conceito de *persona* e suas propriedades contextuais.

Adicionar Persona

Esta funcionalidade permite adicionar um *persona* ao projeto. Para executar esta funcionalidade, o usuário deve clicar no botão "+" (Figura D.34). Na janela aberta, deve digitar o nome para o *persona* e clicar no botão *Finish* (Figura D.35).

Excluir Persona

Esta funcionalidade permite excluir um *persona* do projeto. Quando um *persona* é excluído, as transições associadas a ele são excluídas também. Para executar esta funcionalidade, o usuário deve selecionar o *persona* que deseja excluir e clicar no botão "x" (Figura D.36). Na janela aberta, clicar no botão *Yes* para confirmar a exclusão do *persona* e suas transições associadas (Figura D.37).

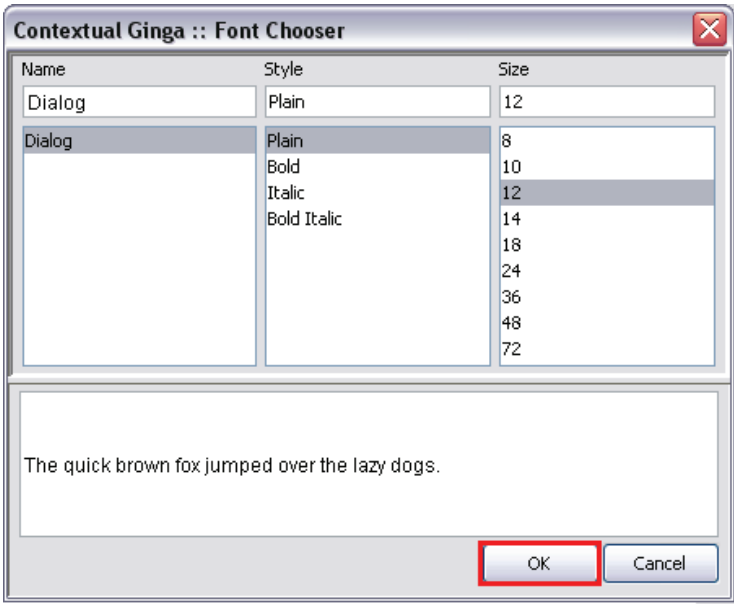


Figura D.33: Alterar propriedade de estilo de fonte.

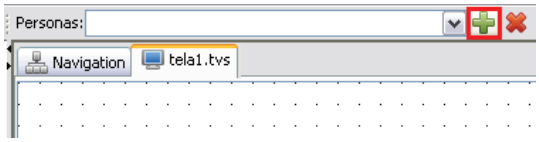


Figura D.34: Adicionar *persona*.

Figura D.35: Nome do *persona*.

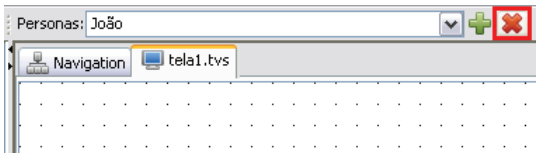


Figura D.36: Excluir *persona*.

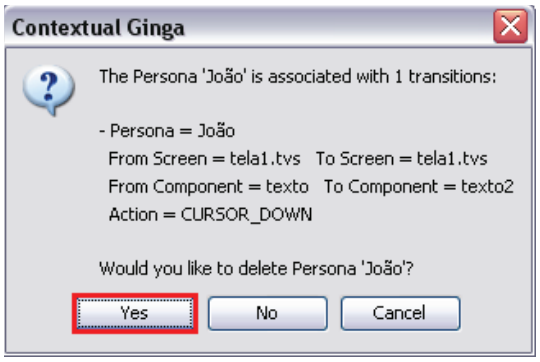


Figura D.37: Confirmar exclusão do *persona*.

Alterar Contexto do Persona

Esta funcionalidade permite alterar as características contextuais de um *persona*: *Who*, *When* e *Where*. A categoria *Who* possui propriedades que definem a faixa etária e o sexo do usuário (Figura D.38). A categoria *When* possui propriedades que definem a data e a hora inicial e final, e o dia da semana (Figura D.39). A categoria *Where* possui como propriedades o país, o estado, a cidade e o código postal (Figura D.40). Para alterar os valores, o usuário deve alterar o valor do campo desejado e tirar o foco do campo.

Visando estimular a inserção de sensibilidade a contexto nas aplicações geradas, os campos que os usuários utilizam para construir as características de um *persona* e o cenário do momento da interação estão sempre visíveis enquanto a ferramenta está sendo executada. Mesmo assim, o usuário pode criar através da ferramenta uma aplicação que não seja sensível a contexto, para tanto, deve-se utilizar um *persona Default*, ou seja, sem valores para as características contextuais.

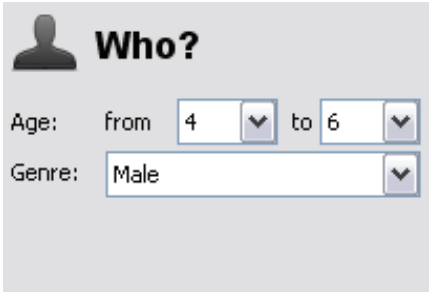


Figura D.38: Característica *who*.

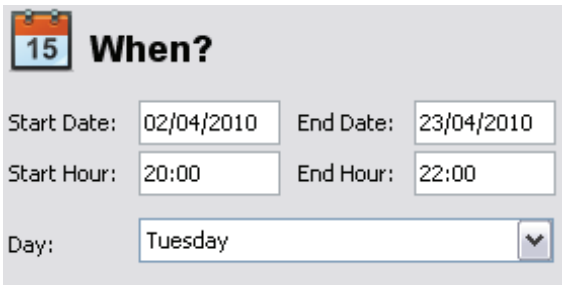


Figura D.39: Característica *when*.

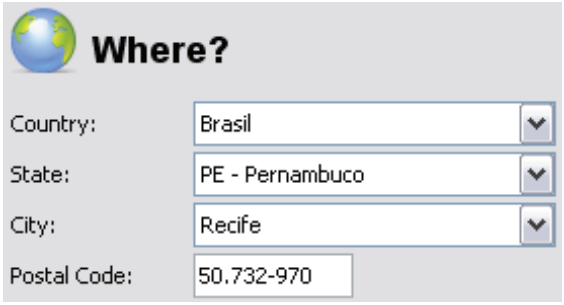


Figura D.40: Característica *where*.

D.3.5 Transição

Esta subseção descreve as funcionalidades referentes ao conceito de transição.

Adicionar Transição

Esta funcionalidade permite criar uma transição entre dois componentes, estando eles na mesma tela ou em telas distintas. Para executar esta funcionalidade, com o *persona* desejado selecionado, o usuário deve clicar com o botão direito sobre o componente de origem da transição e clicar no item *Add Transition* (Figura D.41). Na janela aberta, selecionar a tela e o componente de destino da transição e qual a ação que vai disparar a transição. Em seguida, o usuário deve clicar no botão *OK* (Figura D.42).

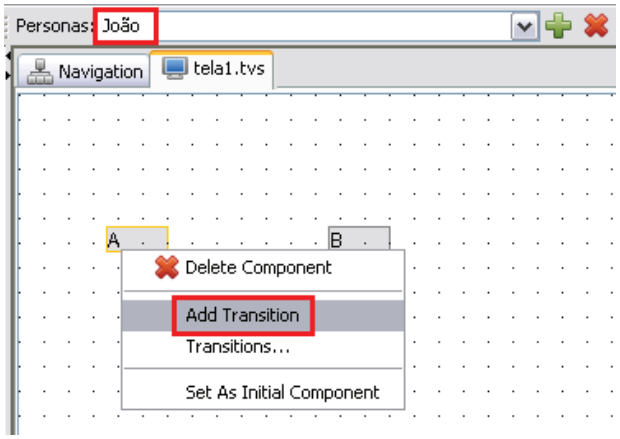


Figura D.41: Adicionar transição.

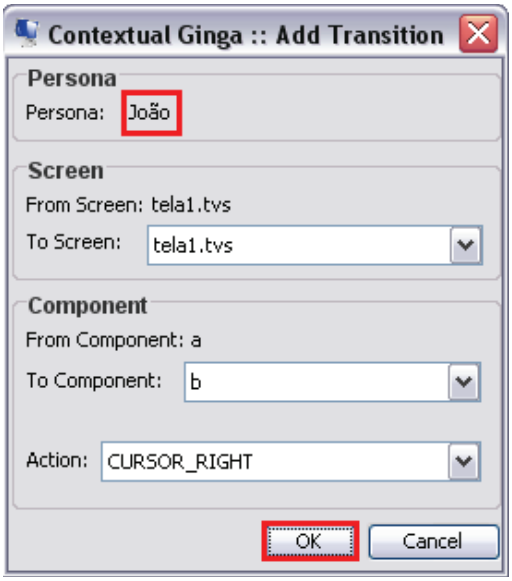


Figura D.42: Valores da transição.

Visualizar Transições

Esta funcionalidade permite visualizar as transições que são originadas em um componente ou tela por *persona*. Para executar esta funcionalidade, o usuário deve clicar com o botão direito sobre o componente de origem da transição e clicar no item *Transitions...* (Figura D.43). Na janela aberta, é exibida a lista de transições que são originadas neste componente (Figura D.44).

Excluir Transição

Esta funcionalidade permite excluir uma transição entre dois componentes. Para executar esta funcionalidade, na tela que lista as transições de um componente, o usuário deve clicar em uma das linhas da tabela que representa uma transição e clicar no botão "x" para a exclusão (Figura D.44).

Alterar Transição

Esta funcionalidade permite alterar os valores das propriedades de uma transição.

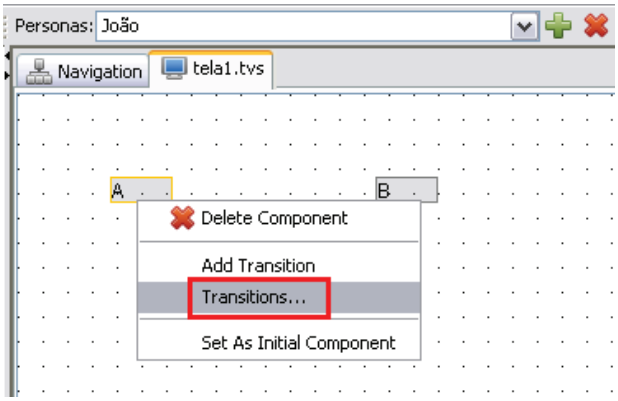


Figura D.43: Visualizar transições.

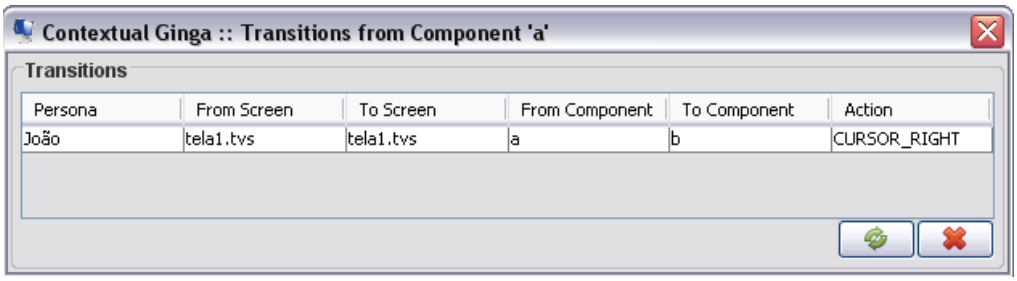


Figura D.44: Lista de transições.

Para executar esta funcionalidade, na tela que lista as transições de um componente, o usuário deve clicar em uma das linhas da tabela que representa uma transição e clicar no botão de atualizar, representado por duas setas (Figura D.44). Na janela aberta, semelhante à Figura D.42, o usuário deve alterar os valores desejados e clicar no botão *OK*.

Visualizar Transições Graficamente

Esta funcionalidade permite que sejam visualizadas, para um dado *persona*, as transições existentes entre as telas de maneira similar a um diagrama de estados. Mesmo que não existam transições entre duas telas, não existirão ligações entre elas, mas seus nomes serão exibidos. Esta funcionalidade também permite que o usuário da aplicação altere a exibição da navegação entre as telas, ou seja, reorganize os nomes das telas para adequar a visualização das transições as suas preferências. Para tanto, deve-se arrastar o nome das telas através de *drag-and-drop*.

As transições são exibidas na aba *Navigation* (Figura D.45). Uma linha entre os nomes de duas telas indica que existe uma transição entre as mesmas e um arco indica o seu sentido da origem para o destino (transição da *tela1.tvs* para a *tela2.tvs*). Se

entre duas telas existem transições nos dois sentidos, os dois arcos, formam um círculo (transição entre as telas *tela2.tvs* e *tela3.tvs*). Um círculo, no canto superior esquerdo do nome da tela, também representa uma transição, mas entre componentes da mesma tela (componentes da *tela1.tvs*). Se o usuário selecionar outro *persona*, a visualização é automaticamente alterada para a configuração definida para o novo *persona* selecionado.

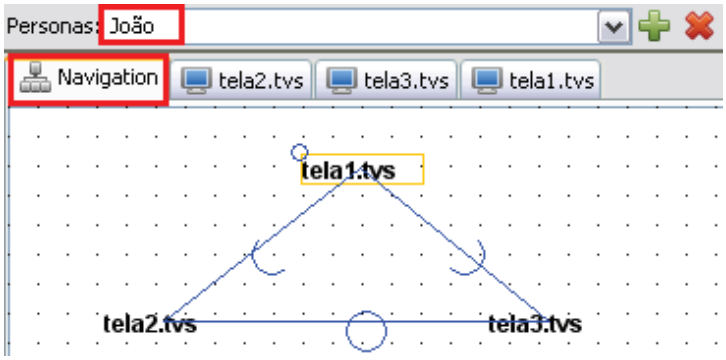


Figura D.45: Navegação entre as telas por *persona*.

D.3.6 Geração de Código

Esta subseção descreve as funcionalidades referentes à geração de código NCLua para a aplicação de TV digital.

Gerar Código NCLua

Esta funcionalidade permite produzir a aplicação para a plataforma Ginga-NCL a partir dos dados presentes em um projeto construído na ferramenta *Contextual Ginga*. Para executar esta funcionalidade, o usuário deve clicar no menu *Code*, no item *Generate NCLua Code* (Figura D.46) ou digitar o atalho *Ctrl+G*. Antes da geração, será aberta uma janela para o usuário confirmar que deseja prosseguir, pois para a geração, é necessário salvar o projeto. Nesta janela, o usuário deve clicar no botão *Yes* (Figura D.47). Depois do código gerado, é aberta uma janela com este aviso, devendo o usuário clicar no botão *OK* (Figura D.48).

O código gerado é colocado na pasta *output*, criada automaticamente dentro da pasta que contém os arquivos do projeto. Dentro da pasta *output*, é criada também a pasta *media* que armazena as imagens utilizadas pela aplicação de TV digital.

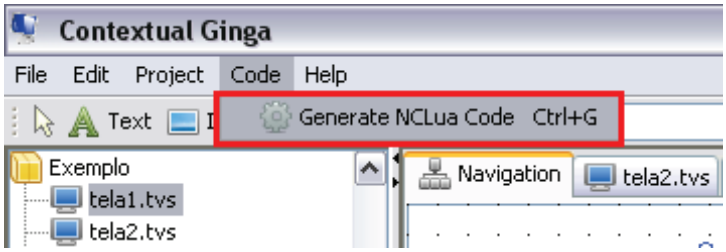


Figura D.46: Gerar código NCLua.

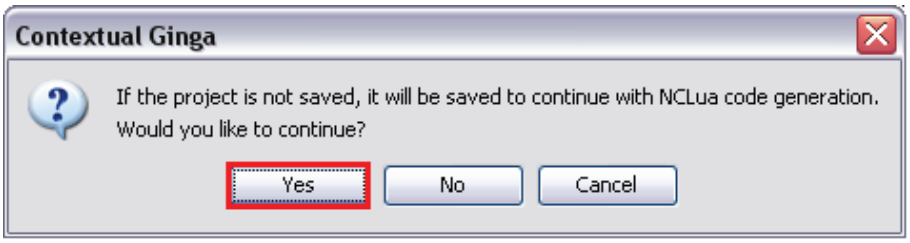


Figura D.47: Confirmação para salvar projeto.

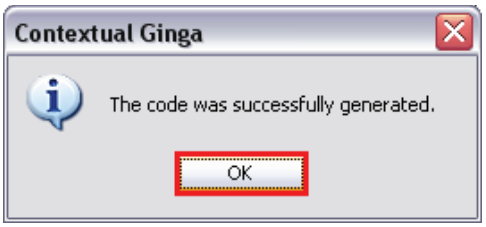


Figura D.48: Código da aplicação NCLua gerado.

D.4 Execução de Aplicações

As aplicações desenvolvidas como testes para a ferramenta foram executadas na máquina virtual Linux *Ginga-NCL Virtual Set-Top-Box v.0.11.2* (PORTAL, 2009). O código da aplicação gerada deve ser organizado na máquina virtual da forma exposta na Figura D.49.

Na máquina virtual, dentro do diretório */misc* deve ser criado um diretório para que os arquivos da aplicação sejam organizados. Neste diretório, devem ser colocados os arquivos gerados pela ferramenta e o diretório *media*. Dentro do diretório *media* deve ser colocado ainda o arquivo do vídeo principal com o mesmo nome fornecido para o projeto.

O diretório */misc* também deve conter: a Biblioteca Contextual Ginga Lua criada neste trabalho; e o diretório */userChoices* que deve ser criado pelo usuário. Este diretório armazenará os registros das interações do usuário com a aplicação de TV digita, que são

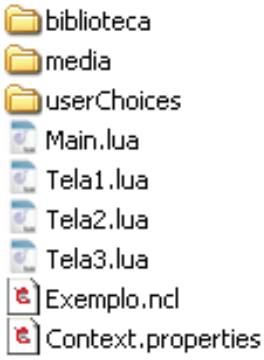


Figura D.49: Estrutura de diretórios na máquina virtual.

criados pela biblioteca para cada vez que a aplicação é executada.

Por fim, o diretório */misc* deve conter um arquivo denominado *Context.properties*, no qual a aplicação irá capturar as informações das categorias contextuais *Who* e *Where* do usuário. No Código D.1, é exibido um exemplo do conteúdo deste arquivo, no qual cada linha possui uma propriedade e um valor para informação contextual.

Código D.1: Arquivo *Context.properties*.

```
1 age=5;
2 genre=m;
3 country=Brasil;
4 state=PE - Pernambuco;
5 city=Recife;
6 postalCode=50.732-970;
```

Antes de executar a aplicação de TV digital sensível a contexto, é fundamental verificar se a data e a hora do sistema estão corretas ou de acordo com o teste que se deseja efetuar. Esta operação deve ser realizada através de uma aplicação de console remoto. Com a execução do comando *date*, são retornados a data e a hora correntes do sistema. Com o comando *date MMDDhhmmYYYY*, pode-se alterar estes valores, onde MM corresponde ao mês, DD ao dia, hh à hora com 24 horas, mm ao minuto e YYYY ao ano. Pode ser visto um exemplo na Figura D.50.

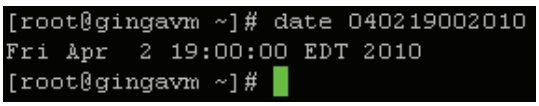


Figura D.50: Alterar valor da data e da hora do sistema.

APÊNDICE E – Código NCLua Gerado

Neste apêndice são transcritos o código do projeto do cenário da fase 2 do experimento e o código gerado pela ferramenta *Contextual Ginga* para a aplicação de TV digital.

E.1 Código do Projeto

Código E.1: Arquivo Canal.tvp.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<projectDescription>
    <projectName>Canal</projectName>
    <resolution>
        <width>640</width>
        <height>480</height>
    </resolution>
    <video>corrego1.mp4</video>
    <screens>
        <screen name="Noticias.tvs">
            <x>78</x>
            <y>265</y>
        </screen>
        <screen name="Esportes.tvs">
            <x>362</x>
            <y>265</y>
        </screen>
        <screen name="Menu.tvs">
            <x>223</x>
            <y>168</y>
        </screen>
    </screens>
</projectDescription>
```

Código E.2: Arquivo Menu.tvs (Parte 1).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<screen name="Menu.tvs">
    <component name="noticias">
        <type>javax.swing.JLabel</type>
        <x>238</x>
```

Código E.3: Arquivo Menu.tvs (Parte 2).

```
6           <y>113</y>
7           <width>99</width>
8           <height>32</height>
9           <image>C:\img\Inicio_Noticias.png</image>
10          <imageOnFocus>C:\img\Inicio_NoticiasSelecionado.png</imageOnFocus>
11      </component>
12      <component name="esportes">
13          <type>javax.swing.JLabel</type>
14          <x>237</x>
15          <y>204</y>
16          <width>99</width>
17          <height>32</height>
18          <image>C:\img\Inicio_Esportes.png</image>
19          <imageOnFocus>C:\img\Inicio_EsportesSelecionado.png</imageOnFocus>
20      </component>
21  </screen>
```

Código E.4: Arquivo Noticias.tvs.

```
1   <?xml version="1.0" encoding="iso-8859-1"?>
2   <screen name="Noticias.tvs">
3       <component name="n_noticias">
4           <type>javax.swing.JLabel</type>
5           <x>30</x>
6           <y>45</y>
7           <width>32</width>
8           <height>99</height>
9           <image>C:\img\Noticias.png</image>
10          <imageOnFocus>C:\img\NoticiasSelecionado.png</imageOnFocus>
11      </component>
12      <component name="texto">
13          <type>javax.swing.JLabel</type>
14          <x>105</x>
15          <y>44</y>
16          <width>500</width>
17          <height>300</height>
18          <text>Text Noticias</text>
19          <font>Dialog, Plain, 48</font>
20          <foreground>255, 255, 0</foreground>
21          <background>0, 204, 204</background>
22      </component>
23      <component name="n_esportes">
24          <type>javax.swing.JLabel</type>
25          <x>30</x>
26          <y>165</y>
27          <width>32</width>
28          <height>99</height>
29          <image>C:\imagens\Esportes.png</image>
30          <imageOnFocus>C:\imagens\EsportesSelecionado.png</imageOnFocus>
31      </component>
32  </screen>
```

Código E.5: Arquivo Esportes.tvs.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<screen name="Esportes.tvs">
    <component name="e_esportes">
        <type>javax.swing.JLabel</type>
        <x>30</x>
        <y>165</y>
        <width>32</width>
        <height>99</height>
        <image>C:\imagens\Esportes.png</image>
        <imageOnFocus>C:\imagens\EsportesSelecionado.png</imageOnFocus>
    </component>
    <component name="e_texto">
        <type>javax.swing.JLabel</type>
        <x>105</x>
        <y>44</y>
        <width>500</width>
        <height>300</height>
        <text>Texto sobre Esportes</text>
        <font>Dialog, Plain, 48</font>
        <foreground>255, 255, 0</foreground>
        <background>0, 204, 204</background>
    </component>
    <component name="e_noticias">
        <type>javax.swing.JLabel</type>
        <x>30</x>
        <y>45</y>
        <width>32</width>
        <height>99</height>
        <image>C:\imagens\Noticias.png</image>
        <imageOnFocus>C:\imagens\NoticiasSelecionado.png</imageOnFocus>
    </component>
</screen>
```

Código E.6: Arquivo Personas.ctx (Parte 1).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<personas>
    <persona initialComponent="e_esportes" initialScreen="Esportes.tvs"
        name="João">
        <who>
            <ageFrom>15</ageFrom>
            <ageTo>-1</ageTo>
            <genre>0</genre>
        </who>
        <when>
            <start>01/06/2010 08:00</start>
            <end>30/06/2010 18:00</end>
            <dayOfWeek/>
        </when>
        <where>
            <country>Brasil</country>
            <state>PE - Pernambuco</state>
```

Código E.7: Arquivo Personas.ctx (Parte 2).

```
18              <city>Recife</city>
19              <postalCode>50.610-220</postalCode>
20          </where>
21      </persona>
22      <persona initialComponent="n_noticias" initialScreen="Noticias.tvs"
23          name="Maria">
24          <who>
25              <ageFrom>20</ageFrom>
26              <ageTo>-1</ageTo>
27              <genre>1</genre>
28          </who>
29          <when>
30              <start>  /  /       :  </start>
31              <end>  /  /       :  </end>
32              <dayOfWeek/>
33          </when>
34          <where>
35              <country/>
36              <state/>
37              <city/>
38              <postalCode/>
39          </where>
40      </persona>
41      <persona initialComponent="noticias" initialScreen="Menu.tvs"
42          name="Default">
43          <who>
44              <ageFrom>-1</ageFrom>
45              <ageTo>-1</ageTo>
46              <genre>-1</genre>
47          </who>
48          <when>
49              <start>  /  /       :  </start>
50              <end>  /  /       :  </end>
51              <dayOfWeek/>
52          </when>
53          <where>
54              <country/>
55              <state/>
56              <city/>
57              <postalCode/>
58          </where>
59      </persona>
60  </personas>
```

Código E.8: Arquivo Transitions.ctx (Parte 1).

```
1   <?xml version="1.0" encoding="iso-8859-1"?>
2   <transitions>
3       <transition id="1">
4           <personaName>Default</personaName>
5           <fromScreen>Menu.tvs</fromScreen>
6           <toScreen>Menu.tvs</toScreen>
```

Código E.9: Arquivo Transitions.ctx (Parte 2).

```
7         <fromComponent>noticias</fromComponent>
8         <toComponent>esportes</toComponent>
9         <action>CURSOR_DOWN</action>
10    </transition>
11    <transition id="2">
12        <personaName>Default</personaName>
13        <fromScreen>Menu.tvs</fromScreen>
14        <toScreen>Menu.tvs</toScreen>
15        <fromComponent>esportes</fromComponent>
16        <toComponent>noticias</toComponent>
17        <action>CURSOR_UP</action>
18    </transition>
19    <transition id="3">
20        <personaName>Default</personaName>
21        <fromScreen>Menu.tvs</fromScreen>
22        <toScreen>Noticias.tvs</toScreen>
23        <fromComponent>noticias</fromComponent>
24        <toComponent>n_noticias</toComponent>
25        <action>ENTER</action>
26    </transition>
27    <transition id="4">
28        <personaName>Default</personaName>
29        <fromScreen>Menu.tvs</fromScreen>
30        <toScreen>Esportes.tvs</toScreen>
31        <fromComponent>esportes</fromComponent>
32        <toComponent>e_esportes</toComponent>
33        <action>ENTER</action>
34    </transition>
35    <transition id="8">
36        <personaName>Default</personaName>
37        <fromScreen>Menu.tvs</fromScreen>
38        <toScreen>Menu.tvs</toScreen>
39        <fromComponent>noticias</fromComponent>
40        <toComponent>esportes</toComponent>
41        <action>0</action>
42    </transition>
43    <transition id="9">
44        <personaName>João</personaName>
45        <fromScreen>Menu.tvs</fromScreen>
46        <toScreen>Menu.tvs</toScreen>
47        <fromComponent>noticias</fromComponent>
48        <toComponent>esportes</toComponent>
49        <action>CURSOR_DOWN</action>
50    </transition>
51    <transition id="10">
52        <personaName>João</personaName>
53        <fromScreen>Menu.tvs</fromScreen>
54        <toScreen>Menu.tvs</toScreen>
55        <fromComponent>esportes</fromComponent>
56        <toComponent>noticias</toComponent>
57        <action>CURSOR_UP</action>
58    </transition>
```

Código E.10: Arquivo Transitions.ctx (Parte 3).

```
59
60      <transition id="11">
61          <personaName>João</personaName>
62          <fromScreen>Menu.tvs</fromScreen>
63          <toScreen>Esportes.tvs</toScreen>
64          <fromComponent>esportes</fromComponent>
65          <toComponent>e_esportes</toComponent>
66          <action>ENTER</action>
67      </transition>
68      <transition id="12">
69          <personaName>João</personaName>
70          <fromScreen>Menu.tvs</fromScreen>
71          <toScreen>Noticias.tvs</toScreen>
72          <fromComponent>noticias</fromComponent>
73          <toComponent>n_noticias</toComponent>
74          <action>ENTER</action>
75      </transition>
76      <transition id="7">
77          <personaName>Default</personaName>
78          <fromScreen>Esportes.tvs</fromScreen>
79          <toScreen>Noticias.tvs</toScreen>
80          <fromComponent>e_esportes</fromComponent>
81          <toComponent>n_noticias</toComponent>
82          <action>CURSOR_UP</action>
83      </transition>
84      <transition id="15">
85          <personaName>João</personaName>
86          <fromScreen>Esportes.tvs</fromScreen>
87          <toScreen>Noticias.tvs</toScreen>
88          <fromComponent>e_esportes</fromComponent>
89          <toComponent>n_noticias</toComponent>
90          <action>CURSOR_UP</action>
91      </transition>
92      <transition id="5">
93          <personaName>Default</personaName>
94          <fromScreen>Noticias.tvs</fromScreen>
95          <toScreen>Esportes.tvs</toScreen>
96          <fromComponent>n_noticias</fromComponent>
97          <toComponent>e_esportes</toComponent>
98          <action>CURSOR_DOWN</action>
99      </transition>
100     <transition id="6">
101         <personaName>Default</personaName>
102         <fromScreen>Noticias.tvs</fromScreen>
103         <toScreen>Menu.tvs</toScreen>
104         <fromComponent>n_noticias</fromComponent>
105         <toComponent>noticias</toComponent>
106         <action>CURSOR_UP</action>
107     </transition>
108     <transition id="13">
109         <personaName>João</personaName>
110         <fromScreen>Noticias.tvs</fromScreen>
```

Código E.11: Arquivo Transitions.ctx (Parte 4).

```
111         <toScreen>Menu.tvs</toScreen>
112         <fromComponent>n_noticias</fromComponent>
113         <toComponent>noticias</toComponent>
114         <action>CURSOR_UP</action>
115     </transition>
116     <transition id="14">
117         <personaName>João</personaName>
118         <fromScreen>Noticias.tvs</fromScreen>
119         <toScreen>Esportes.tvs</toScreen>
120         <fromComponent>n_noticias</fromComponent>
121         <toComponent>e_esportes</toComponent>
122         <action>CURSOR_DOWN</action>
123     </transition>
124 </transitions>
```

E.2 Código Gerado pelo *Contextual Ginga*

Código E.12: Arquivo Canal.ncl (Parte 1).

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <ncl id="Canal" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
3  <head>
4          <!-- Regions. -->
5          <regionBase>
6                  <region width="640" height="480" left="0" top="0"
7                          id="rgVideo"/>
8                  <region width="640" height="480" left="0" top="0"
9                          id="rgLua"/>
10         </regionBase>
11
12         <!-- Descriptors. -->
13         <descriptorBase>
14                 <descriptor id="dsVideo" region="rgVideo"/>
15                 <descriptor id="dsLua" region="rgLua" focusIndex="1">
16                         <descriptorParam name="transparency" value="0.7"/>
17                 </descriptor>
18         </descriptorBase>
19
20         <!-- Connectors. -->
21         <connectorBase>
22                 <causalConnector id="onBeginStart">
23                         <simpleCondition role="onBegin"/>
24                         <simpleAction role="start" max="unbounded"
25                                 qualifier="par"/>
26                 </causalConnector>
27                 <causalConnector id="onEndStart">
28                         <simpleCondition role="onEnd"/>
29                         <simpleAction role="start" max="unbounded"
30                                 qualifier="par"/>
```

Código E.13: Arquivo Canal.ncl (Parte 2).

```
31                  </causalConnector>
32          </connectorBase>
33  </head>
34
35  <body>
36          <port id="entryPoint" component="video"/>
37
38          <!-- Medias -->
39          <media id="video" src="media/corrego1.mp4" descriptor="dsVideo"/>
40          <media id="lua" src="Main.lua" descriptor="dsLua"/>
41          <!-- Links -->
42          <link xconnector="onEndStart">
43                  <bind role="onEnd" component="video"/>
44                  <bind role="start" component="video"/>
45          </link>
46          <link xconnector="onBeginStart">
47                  <bind role="onBegin" component="video"/>
48                  <bind role="start" component="lua"/>
49          </link>
50  </body>
51  </ncl>
```

Código E.14: Arquivo Main.lua (Parte 1).

```
1  --============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  -- Contextual Ginga
5  --============================================================
6
7  ------------------------------------------------------------
8  package.path =
9          package.path .. ';./biblioteca/util/?.lua;./biblioteca/gui/?.lua';
10 require('biblioteca.util.ScriptManager');
11
12 ScriptManager:execute('biblioteca.gui.Screen');
13 ScriptManager:execute('biblioteca.gui.Text');
14 ScriptManager:execute('biblioteca.gui.Image');
15 ScriptManager:execute('biblioteca.gui.Font');
16 ScriptManager:execute('biblioteca.gui.Color');
17
18 ScriptManager:execute('biblioteca.util.Time');
19 ScriptManager:execute('biblioteca.util.Log');
20 ScriptManager:execute('biblioteca.util.Persona');
21
22 ScriptManager:execute('Esportes');
23 ScriptManager:execute('Menu');
24 ScriptManager:execute('Noticias');
25
26 ------------------------------------------------------------
27 -- User choices.
```

Código E.15: Arquivo Main.lua (Parte 2).

```
28 currentTime = os.date('*t');
29
30 fileName = 'userChoices/' .. time_getTimeString(currentTime) .. '.output';
31 file = io.open(fileName, 'w');
32
33 ------------------------------------------------------
34 -- Contextual Variables.
35
36 userContext = {age = '',
37                 genre='',
38                 country = '',
39                 city = '',
40                 state = '',
41                 postalCode = ''};
42
43 function getUserContext ()
44         for line in io.lines("Context.properties") do
45                 property = string.sub(line, 1, string.find(line, '=')-1);
46                 value = string.sub(line, string.find(line, '=')+1,
47                 string.find(line, ';')-1);
48
49                 if property == 'age' then
50                         userContext.age = tonumber(value);
51                 elseif property == 'genre' then
52                         userContext.genre = value;
53                 elseif property == 'country' then
54                         userContext.country = value;
55                 elseif property == 'state' then
56                         userContext.state = value;
57                 elseif property == 'city' then
58                         userContext.city = value;
59                 elseif property == 'postalCode' then
60                         userContext.postalCode = value;
61                 end
62         end
63 end
64 getUserContext();
65
66 ------------------------------------------------------
67 -- Persona definition.
68
69 persona = 'Default';
70 screen = 'Menu.tvs';
71 componentWithFocus = 'noticias';
72
73 personaSetted = false;
74
75 function getPersona ()
76         if persona_evaluateWho(15, -1, 'm')
77         and persona_evaluateWhen('01/06/2010', '08:00', '30/06/2010',
78                 '18:00', nil, currentTime)
```

Código E.16: Arquivo Main.lua (Parte 3).

```
79          and persona_evaluateWhere('Brasil', 'PE - Pernambuco', 'Recife',
80                  '50.610-220') then
81                  persona = 'João';
82                  screen = 'Esportes.tvs';
83                  componentWithFocus = 'e_esportes';
84                  drawScreen_Esportes();
85                  personaSetted = true;
86
87          elseif persona_evaluateWho(20, -1, 'f')
88          and persona_evaluateWhen(nil, nil, nil, nil, nil, currentTime)
89          and persona_evaluateWhere(nil, nil, nil, nil) then
90
91                  persona = 'Maria';
92                  screen = 'Noticias.tvs';
93                  componentWithFocus = 'n_noticias';
94                  drawScreen_Noticias();
95                  personaSetted = true;
96          end
97  end
98  getPersona();
99
100 if personaSetted == false and persona == 'Default' and screen ~= '' then
101         drawScreen_Menu();
102 end
```

Código E.17: Arquivo Menu.lua (Parte 1).

```
1   --============================================================
2   -- Universidade Federal de Pernambuco
3   -- Centro de Informática
4   -- Contextual Ginga
5   --============================================================
6
7   ------------------------------------------------------------
8   -- Components.
9   noticias = Image:new(
10          {name = 'noticias', x = 238, y = 113,
11          location = 'media/Inicio_Noticias.png',
12          locationOnFocus = 'media/Inicio_NoticiasSelecionado.png'});
13
14  esportes = Image:new(
15          {name = 'esportes', x = 237, y = 204,
16          location = 'media/Inicio_Esportes.png',
17          locationOnFocus = 'media/Inicio_EsportesSelecionado.png'});
18
19  ------------------------------------------------------------
20  -- Draw Screen.
21
22  function drawScreen_Menu()
23          if componentWithFocus ~= nil then
24
25                  if noticias.name == componentWithFocus then
```

Código E.18: Arquivo Menu.lua (Parte 2).

```
26                          noticias:draw(true);
27                  else
28                          noticias:draw(false);
29                  end
30
31                  if esportes.name == componentWithFocus then
32                          esportes:draw(true);
33                  else
34                          esportes:draw(false);
35                  end
36          end
37          canvas:flush();
38  end
39
40  ------------------------------------------------------
41  -- Events.
42
43  function handler_1(evt)
44          if evt.class == 'key' and evt.type == 'press' then
45
46                  if persona == 'Default' and
47                  screen == 'Menu.tvs' and
48                  componentWithFocus == 'noticias' and
49                  evt.key == 'CURSOR_DOWN' then
50
51                          transitionId = 1;
52                          log_registerTransition1(file, transitionId, persona,
53                                  screen, componentWithFocus);
54                          screen = 'Menu.tvs';
55                          componentWithFocus = 'esportes';
56
57                          log_registerTransition2(file, screen,
58                                  componentWithFocus, evt.key);
59                          cleanScreen(0, 0, 640, 480);
60                          drawScreen_Menu();
61                  end
62          end
63  end
64  event.register(handler_1);
65
66  function handler_2(evt)
67          if evt.class == 'key' and evt.type == 'press' then
68
69                  if persona == 'Default' and
70                  screen == 'Menu.tvs' and
71                  componentWithFocus == 'esportes' and
72                  evt.key == 'CURSOR_UP' then
73
74                          transitionId = 2;
75                          log_registerTransition1(file, transitionId, persona,
76                                  screen, componentWithFocus);
```

Código E.19: Arquivo Menu.lua (Parte 3).

```
77                          screen = 'Menu.tvs';
78                          componentWithFocus = 'noticias';
79
80                          log_registerTransition2(file, screen,
81                                  componentWithFocus, evt.key);
82                          cleanScreen(0, 0, 640, 480);
83                          drawScreen_Menu();
84                  end
85          end
86  end
87  event.register(handler_2);
88
89  function handler_3(evt)
90          if evt.class == 'key' and evt.type == 'press' then
91                  if persona == 'Default' and
92                  screen == 'Menu.tvs' and
93                  componentWithFocus == 'noticias' and
94                  evt.key == 'ENTER' then
95
96                          transitionId = 3;
97                          log_registerTransition1(file, transitionId, persona,
98                                  screen, componentWithFocus);
99                          screen = 'Noticias.tvs';
100                         componentWithFocus = 'n_noticias';
101                         log_registerTransition2(file, screen,
102                                 componentWithFocus, evt.key);
103                         cleanScreen(0, 0, 640, 480);
104                         drawScreen_Noticias();
105                 end
106         end
107 end
108 event.register(handler_3);
109
110 function handler_4(evt)
111         if evt.class == 'key' and evt.type == 'press' then
112
113                 if persona == 'Default' and
114                 screen == 'Menu.tvs' and
115                 componentWithFocus == 'esportes' and
116                 evt.key == 'ENTER' then
117
118                         transitionId = 4;
119                         log_registerTransition1(file, transitionId, persona,
120                                 screen, componentWithFocus);
121                         screen = 'Esportes.tvs';
122                         componentWithFocus = 'e_esportes';
123
124                         log_registerTransition2(file, screen,
125                                 componentWithFocus, evt.key);
126                         cleanScreen(0, 0, 640, 480);
127                         drawScreen_Esportes();
128                 end
```

Código E.20: Arquivo Menu.lua (Parte 4).

```
129         end
130 end
131 event.register(handler_4);
132
133 function handler_8(evt)
134         if evt.class == 'key' and evt.type == 'press' then
135
136                 if persona == 'Default' and
137                 screen == 'Menu.tvs' and
138                 componentWithFocus == 'noticias' and
139                 evt.key == '0' then
140
141                         transitionId = 8;
142                         log_registerTransition1(file, transitionId, persona,
143                                 screen, componentWithFocus);
144                         screen = 'Menu.tvs';
145                         componentWithFocus = 'esportes';
146
147                         log_registerTransition2(file, screen,
148                                 componentWithFocus, evt.key);
149                         cleanScreen(0, 0, 640, 480);
150                         drawScreen_Menu();
151                 end
152         end
153 end
154 event.register(handler_8);
155
156 function handler_9(evt)
157         if evt.class == 'key' and evt.type == 'press' then
158
159                 if persona == 'João' and
160                 screen == 'Menu.tvs' and
161                 componentWithFocus == 'noticias' and
162                 evt.key == 'CURSOR_DOWN' then
163
164                         transitionId = 9;
165                         log_registerTransition1(file, transitionId, persona,
166                                 screen, componentWithFocus);
167                         screen = 'Menu.tvs';
168                         componentWithFocus = 'esportes';
169
170                         log_registerTransition2(file, screen,
171                                 componentWithFocus, evt.key);
172                         cleanScreen(0, 0, 640, 480);
173                         drawScreen_Menu();
174                 end
175         end
176 end
177 event.register(handler_9);
178
179 function handler_10(evt)
180         if evt.class == 'key' and evt.type == 'press' then
```

Código E.21: Arquivo Menu.lua (Parte 5).

```
181                 if persona == 'João' and
182                 screen == 'Menu.tvs' and
183                 componentWithFocus == 'esportes' and
184                 evt.key == 'CURSOR_UP' then
185
186                         transitionId = 10;
187                         log_registerTransition1(file, transitionId, persona,
188                                 screen, componentWithFocus);
189                         screen = 'Menu.tvs';
190                         componentWithFocus = 'noticias';
191
192                         log_registerTransition2(file, screen,
193                                 componentWithFocus, evt.key);
194                         cleanScreen(0, 0, 640, 480);
195                         drawScreen_Menu();
196                 end
197         end
198 end
199 event.register(handler_10);
200
201 function handler_11(evt)
202         if evt.class == 'key' and evt.type == 'press' then
203                 if persona == 'João' and
204                 screen == 'Menu.tvs' and
205                 componentWithFocus == 'esportes' and
206                 evt.key == 'ENTER' then
207
208                         transitionId = 11;
209                         log_registerTransition1(file, transitionId, persona,
210                                 screen, componentWithFocus);
211                         screen = 'Esportes.tvs';
212                         componentWithFocus = 'e_esportes';
213
214                         log_registerTransition2(file, screen,
215                                 componentWithFocus, evt.key);
216                         cleanScreen(0, 0, 640, 480);
217                         drawScreen_Esportes();
218                 end
219         end
220 end
221 event.register(handler_11);
222
223 function handler_12(evt)
224         if evt.class == 'key' and evt.type == 'press' then
225                 if persona == 'João' and
226                 screen == 'Menu.tvs' and
227                 componentWithFocus == 'noticias' and
228                 evt.key == 'ENTER' then
229
230                         transitionId = 12;
231                         log_registerTransition1(file, transitionId, persona,
232                                 screen, componentWithFocus);
```

Código E.22: Arquivo Menu.lua (Parte 6).

```
233                         screen = 'Noticias.tvs';
234                         componentWithFocus = 'n_noticias';
235
236                         log_registerTransition2(file, screen,
237                                 componentWithFocus, evt.key);
238                         cleanScreen(0, 0, 640, 480);
239                         drawScreen_Noticias();
240                 end
241         end
242 end
243 event.register(handler_12);
```

Código E.23: Arquivo Noticias.lua (Parte 1).

```
1  --============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  -- Contextual Ginga
5  --============================================================
6
7  ------------------------------------------------------------
8  -- Components.
9
10 n_noticias = Image:new(
11         {name = 'n_noticias', x = 30, y = 45,
12         location = 'media/Noticias.png',
13         locationOnFocus = 'media/NoticiasSelecionado.png'});
14
15 texto = Text:new(
16         {text = 'Text Noticias',
17         font = Font:new({name = 'vera',
18                         style = 'nil',
19                         size = 48,
20                 color = Color:new({r = 255, g = 255, b = 0, alpha = 255})}),
21         x = 105, y = 44,
22         width = 500, height = 300,
23         background = Color:new({r = 0, g = 204, b = 204, alpha = 255})});
24
25 n_esportes = Image:new(
26         {name = 'n_esportes', x = 30, y = 165,
27         location = 'media/Esportes.png',
28         locationOnFocus = 'media/EsportesSelecionado.png'});
29
30 ------------------------------------------------------------
31 -- Draw Screen.
32
33 function drawScreen_Noticias()
34
35         if componentWithFocus ~= nil then
36
37                 if n_noticias.name == componentWithFocus then
38                         n_noticias:draw(true);
```

Código E.24: Arquivo Noticias.lua (Parte 2).

```
39                  else
40                          n_noticias:draw(false);
41                  end
42
43                  texto:draw();
44
45                  if n_esportes.name == componentWithFocus then
46                          n_esportes:draw(true);
47                  else
48                          n_esportes:draw(false);
49                  end
50
51          end
52          canvas:flush();
53  end
54
55  ------------------------------------------------
56  -- Events.
57
58  function handler_5(evt)
59          if evt.class == 'key' and evt.type == 'press' then
60
61                  if persona == 'Default' and
62                  screen == 'Noticias.tvs' and
63                  componentWithFocus == 'n_noticias' and
64                  evt.key == 'CURSOR_DOWN' then
65
66                          transitionId = 5;
67                          log_registerTransition1(file, transitionId, persona,
68                                  screen, componentWithFocus);
69                          screen = 'Esportes.tvs';
70                          componentWithFocus = 'e_esportes';
71
72                          log_registerTransition2(file, screen,
73                                  componentWithFocus, evt.key);
74                          cleanScreen(0, 0, 640, 480);
75                          drawScreen_Esportes();
76                  end
77          end
78  end
79  event.register(handler_5);
80
81  function handler_6(evt)
82          if evt.class == 'key' and evt.type == 'press' then
83
84                  if persona == 'Default' and
85                  screen == 'Noticias.tvs' and
86                  componentWithFocus == 'n_noticias' and
87                  evt.key == 'CURSOR_UP' then
88
89                          transitionId = 6;
90                          log_registerTransition1(file, transitionId, persona,
```

Código E.25: Arquivo Noticias.lua (Parte 3).

```
91                                  screen, componentWithFocus);
92                          screen = 'Menu.tvs';
93                          componentWithFocus = 'noticias';
94
95                          log_registerTransition2(file, screen,
96                                  componentWithFocus, evt.key);
97                          cleanScreen(0, 0, 640, 480);
98                          drawScreen_Menu();
99                  end
100         end
101 end
102 event.register(handler_6);
103
104 function handler_13(evt)
105         if evt.class == 'key' and evt.type == 'press' then
106
107                 if persona == 'João' and
108                 screen == 'Noticias.tvs' and
109                 componentWithFocus == 'n_noticias' and
110                 evt.key == 'CURSOR_UP' then
111
112                         transitionId = 13;
113                         log_registerTransition1(file, transitionId, persona,
114                                 screen, componentWithFocus);
115                         screen = 'Menu.tvs';
116                         componentWithFocus = 'noticias';
117
118                         log_registerTransition2(file, screen,
119                                 componentWithFocus, evt.key);
120                         cleanScreen(0, 0, 640, 480);
121                         drawScreen_Menu();
122                 end
123         end
124 end
125 event.register(handler_13);
126
127 function handler_14(evt)
128         if evt.class == 'key' and evt.type == 'press' then
129
130                 if persona == 'João' and
131                 screen == 'Noticias.tvs' and
132                 componentWithFocus == 'n_noticias' and
133                 evt.key == 'CURSOR_DOWN' then
134
135                         transitionId = 14;
136                         log_registerTransition1(file, transitionId, persona,
137                                 screen, componentWithFocus);
138                         screen = 'Esportes.tvs';
139                         componentWithFocus = 'e_esportes';
140
141                         log_registerTransition2(file, screen,
142                                 componentWithFocus, evt.key);
```

Código E.26: Arquivo Noticias.lua (Parte 4).

```
143                         cleanScreen(0, 0, 640, 480);
144                         drawScreen_Esportes();
145                 end
146         end
147 end
148 event.register(handler_14);
```

Código E.27: Arquivo Esportes.lua (Parte 1).

```
1  --=============================================================
2  -- Universidade Federal de Pernambuco
3  -- Centro de Informática
4  -- Contextual Ginga
5  --=============================================================
6
7  ------------------------------------------------------------
8  -- Components.
9
10 e_esportes = Image:new(
11         {name = 'e_esportes', x = 30, y = 165,
12         location = 'media/Esportes.png',
13         locationOnFocus = 'media/EsportesSelecionado.png'});
14
15 e_texto = Text:new(
16         {text = 'Texto sobre Esportes',
17         font = Font:new({name = 'vera',
18                         style = 'nil',
19                         size = 48,
20                 color = Color:new({r = 255, g = 255, b = 0, alpha = 255})}),
21         x = 105, y = 44,
22         width = 500, height = 300,
23         background = Color:new({r = 0, g = 204, b = 204, alpha = 255})});
24
25 e_noticias = Image:new(
26         {name = 'e_noticias', x = 30, y = 45,
27         location = 'media/Noticias.png',
28         locationOnFocus = 'media/NoticiasSelecionado.png'});
29
30 ------------------------------------------------------------
31 -- Draw Screen.
32
33 function drawScreen_Esportes()
34
35         if componentWithFocus ~= nil then
36
37                 if e_esportes.name == componentWithFocus then
38                         e_esportes:draw(true);
39                 else
40                         e_esportes:draw(false);
41                 end
42
43                 e_texto:draw();
```

Código E.28: Arquivo Esportes.lua (Parte 2).

```
44                  if e_noticias.name == componentWithFocus then
45                          e_noticias:draw(true);
46                  else
47                          e_noticias:draw(false);
48                  end
49
50          end
51          canvas:flush();
52  end
53
54  ------------------------------------------------------
55  -- Events.
56
57  function handler_7(evt)
58          if evt.class == 'key' and evt.type == 'press' then
59
60                  if persona == 'Default' and
61                  screen == 'Esportes.tvs' and
62                  componentWithFocus == 'e_esportes' and
63                  evt.key == 'CURSOR_UP' then
64
65                          transitionId = 7;
66                          log_registerTransition1(file, transitionId, persona,
67                                  screen, componentWithFocus);
68                          screen = 'Noticias.tvs';
69                          componentWithFocus = 'n_noticias';
70
71                          log_registerTransition2(file, screen,
72                                  componentWithFocus, evt.key);
73                          cleanScreen(0, 0, 640, 480);
74                          drawScreen_Noticias();
75                  end
76          end
77  end
78  event.register(handler_7);
79
80  function handler_15(evt)
81          if evt.class == 'key' and evt.type == 'press' then
82
83                  if persona == 'João' and
84                  screen == 'Esportes.tvs' and
85                  componentWithFocus == 'e_esportes' and
86                  evt.key == 'CURSOR_UP' then
87
88                          transitionId = 15;
89                          log_registerTransition1(file, transitionId, persona,
90                                  screen, componentWithFocus);
91                          screen = 'Noticias.tvs';
92                          componentWithFocus = 'n_noticias';
93
94                          log_registerTransition2(file, screen,
95                                  componentWithFocus, evt.key);
```

Código E.29: Arquivo Esportes.lua (Parte 3).

```
96                          cleanScreen(0, 0, 640, 480);
97                          drawScreen_Noticias();
98                  end
99          end
100 end
101 event.register(handler_15);
```