Sincronização de Conteúdo Multimídia de Fontes Distintas

Thomás Marques Brandão Reis

JUIZ DE FORA

12, 2014

Sincronização de Conteúdo Multimídia de Fontes Distintas

Thomás Marques Brandão Reis

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computacão

Orientador: Marcelo Ferreira Moreno

JUIZ DE FORA

12, 2014

Sincronização de Conteúdo Multimídia de Fontes Distintas

Thomás Marques Brandão Reis

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTACÃO.

Aprovada por:

Marcelo Ferreira Moreno
Doutor em Informática pela PUC-Rio

Eduardo Barrére
Doutor em Engenharia de Sistemas e Computação COPPE/UFRJ

Carlos Pernisa Junior
Doutor em Comunicação e Cultura UFRJ

JUIZ DE FORA

09 DE 12, 2014

Aos meus amigos e irmãos.

Aos pais, pelo apoio e sustento.

Resumo

O consumo de conteúdo multimídia pela população vem crescendo consideravelmente. Com o surgimento da TV Digital e sua capacidade de transmitir objetos de mídia em alta resolução e interatividade com o usuário, alavancou ainda mais esse consumo. O processo de aquisição de infraestrutura para transmissão de TV Digital tem um custo elevado para os provedores de conteúdo e pode se tornar lento. O presente projeto tem como objetivo minimizar esse custo e estabelecer um modelo de sincronização de timelines de conteúdo multimídia entre diferentes fontes, solucionando o problema dos provedores em relação a limitação da quantidade de informação em um espaço de comunicação, visando também a portabilidade e aumentando a experiência do usuário que consome esse conteúdo.

Palavras-chave: TV Digital, Interatividade, Sincronização, Dispositivos secundários, Conteúdo multimídia.

Abstract

The consumption of multimedia content by the population has been growing considerably. With the advent of digital TV and its ability to stream media objects in high resolution and interactivity with the user, further leveraged this consumption. The infrastructure acquisition process for digital TV transmission has a high cost for content providers and can become slow. This project aims to minimize the cost and establish timelines synchronization model of multimedia content from different sources, solving the problem of providers with regard to limiting the amount of information in a communication space, also aimed at increasing the portability and user experience that consumes that content.

Keywords: Digital TV, Interactivity, Synchronization, Companion devices, multimedia content.

Agradecimentos

Aos meus pais, pela ajuda, encorajamento e apoio.

Ao professor Marcelo Ferreira Moreno pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria. Aos professores Eduardo Barrére e Carlos Pernisa Junior pela paciência e pelo tempo dedicado a avaliação deste trabalho.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

Aos meus amigos e colegas pela amizade, companheirismo e apoio durante esta trajetória.

"Julgue seu sucesso pelas coisas que você teve que renunciar para conseguir".

Dalai Lama

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
DTV	Digital Television
HTTP	Hyper Text Transfer Protocol
URL	Universal Resource Locator
NCL	Nexted Context Language
XML	eXtensible Markup Language
NCM	Nested Context Model
QoS	Quality of Service
QoE	Quality of Experience
RTP	Real-time Transfer Protocol
AAC	Advanced Audio Coding
JPEG	Join Photographic Expert Group
PNG	Portable Network Graphics
GIF	Graphics Interchange Format
SBTVD	Sistema Brasileiro de Televisão Digital
API	Application Programming Interface
IBGE	Instituto Brasileiro de Geografia e Estatística
SVG	Scalable Vector Graphics
XHTML	Extensible Hypertext Markup Language
W3C	World Wide Web Consortium
CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
UTC	Universal Time Coordinated
GMT	Greenwich Mean Time
TVDi	TV Digital Interativa

1 Introdução

A televisão é um dos mais importantes meios de comunicação no Brasil e sempre teve um forte papel integrador, sendo um agente de cultura e entretenimento. A televisão possui uma forte influência sobre a população, sendo que em algumas localidades esse é o único meio de comunicação (Brackmann, 2008).

Hoje é praticamente incomum um lar onde não possua pelo menos uma TV, sendo que em grande parte deles é possível encontrar até mais de uma. Segundo (IBGE, 2012), resultados apontam que 97,2% da população possuem aparelho de televisão em seus domicílios.

1.1 Contextualização

A TV Digital é uma evolução tecnológica da TV analógica e atualmente vem sofrendo uma expansão considerável, impulsionada pela possibilidade de transmissão de imagem em alta definição. Um serviço de TV Digital pode oferecer também, além do conteúdo audiovisual (imagem, vídeo, som, legendas), conteúdo interativo para os usuários (aplicações). Aplicações estas que podem ser desenvolvidas em NCL (Nexted Context Language) que é uma linguagem de aplicação XML que permitem aos autores criarem apresentações multimídia interativas.

A possibilidade de interação coloca como requisitos para o aparelho receptor de televisão capacidade para processar o código que define os elementos de interação, o "código do programa de tv interativo", e enviar o resultado da interação através de um "canal de retorno" para a estação emissora do programa interativo (Fernandes; Lemos e Silveira, 2004). Com a interatividade na TV Digital, há a possibilidade de que os telespectadores influenciem nos programas que estão assistindo e, portanto, segundo (Fernandes; Lemos e Silveira, 2004) espera-se uma penetração muito maior devido a capacidade de interação instantânea de milhões de telespectadores por meio da TVDi.

O caso mais tradicional de oferecer um serviço de TV Digital é através de um canal

de transmissão onde todos os serviços são transmitidos em conjunto. Devido à natureza unidirecional do canal de transmissão, exatamente o mesmo conteúdo é entregue a todos os usuários ao mesmo tempo.

A internet é um canal de transmissão bidirecional, onde o conteúdo pode ser transmitido tanto de forma linear quanto não linear, e com isso, graças a sua mão dupla, o conteúdo pode ser transmitido sob demanda. Com o aumento da disponibilidade de serviços de acesso à internet banda larga, fabricantes e prestadores de serviços de TV digital começaram a tomar vantagem da possibilidade de fornecer dispositivos e serviços de internet conectados (ITU-T J.205, 2012).

Dessa forma, há então a capacidade de estabelecer uma relação entre o conteúdo multimídia oferecido pela TV Digital, TV analógica, ou até mesmo oriundo da própria Internet, com os demais dispositivos que o telespectador possua, como computadores, notebooks, tablets, aparelhos celulares, etc.

1.2 Justificativa

Com a expansão da TV Digital e a possibilidade de consumo de conteúdo multimídia em alta definição e aplicações interativas é necessário algum componente capaz de reconhecer e tratar esses tipos de informações.

O SBTVD (Sistema Brasileiro de TV Digital), assim como outros sistemas de TV Digital no mundo, é organizado em camadas que funcionam de forma similar às camadas de rede de computadores, onde cada camada é responsável por tratar uma informação específica. As camadas são: modulação, transporte, compressão de áudio e vídeo, middleware e aplicação (Brackmann, 2008).

O Ginga, desenvolvido pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-RJ) e Universidade Federal da Paraíba (UFPB), é o middleware padronizado internacionalmente e utilizado atualmente nos receptores de TV Digital no Brasil. O Ginga pode ser dividido em três subsistemas: Ginga-NCL, Ginga-J e Ginga-CommonCore. O desenvolvimento das aplicações podem ser feitas de forma declarativa ou procedural, que são tratadas pelo Ginga-NCL e o Ginga-J respectivamente. As aplicações, em sua maioria, são desenvolvidas baseadas na linguagem NCL (Nested Context Language - declarativa -

linguagem baseada em XML) e Java (procedural - linguagem orientada a objetos).

No ambiente de TV Digital, além da capacidade de difusão de informação em várias mídias diferentes, como áudio, vídeo e dados (textos, legendas), é possível oferecer diversos tipos de serviços, desde a simples distribuição de conteúdo audiovisual da programação atual da TV analógica por difusão, até serviços hipermídia antes não disponíveis (Guimarães e Costa, 2006). Através das aplicações baseadas nas linguagens citadas acima, é possível utilizar diferentes mecanismos de sincronização espaço-temporal e interatividade, permitindo que os programas de TV sejam formados por aplicações hipermídia complexas, garantindo interação do usuário com o conteúdo disponibilizado pelas emissoras.

Portanto, para usufruir das vantagens TV Digital, além do usuário adquirir um receptor de sinal digital contendo todos componentes necessários (middleware e demais camadas) para a execução e/ou apresentação do conteúdo hipermídia/multimídia, é necessário que a emissora, que disponibiliza o conteúdo que o usuário deseja assistir, possua toda a infraestrutura padrão para a transmissão do conteúdo em qualidade digital.

Existem dois grandes fatores que impedem a adoção do ambiente de TV Digital Interativa no Brasil. Um deles é o fato das emissoras já possuírem seu modelo de negócio consolidado no ambiente de TV analógico, e consequentemente, atrasando o seu projeto de migração para o modelo de TV Digital Interativa. O segundo fator é a necessidade, por parte do telespectador, da aquisição de um aparelho receptor capaz de reproduzir este conteúdo digital, como um aparelho de TV mais atual ou um aparelho conversor de sinal de TV Digital.

De forma alternativa, pode-se contar com a gama enorme de dispositivos capazes de se conectar à Internet atualmente, e a possibilidade de praticamente todos serem capazes de reconhecer conteúdo multimídia, tanto áudio, vídeo e aplicativos. Assim, surge a possibilidade de utilizar este canal de transmissão de dados como aliado ao meio de radiodifusão comumente utilizado pelas emissoras de TV.

Portanto, é possível criar uma forma alternativa para o consumo de conteúdo multimídia em aplicações interativas, tomando vantagem do uso da Internet disponível nas residências, e a posse de algum outro dispositivo (celular, tablet, notebook, etc.)

conectado e que tenha capacidade de reproduzir esse conteúdo.

1.3 Objetivos

1.3.1 Objetivos Gerais

Este projeto tem como objetivo estabelecer uma forma de sincronização entre conteúdo multimídia provido de fontes de informação como a TV e outro dispositivo, baseando-se em fundamentos do modelo de sincronismo temporal na linguagem NCL, adotada pelo SBTVD (Associação Brasileira, 2008). Espera-se resolver o problema de interatividade na TV digital, bem como outros meios de comunicação. Em uma segunda abordagem, mas não menos importante, solucionar o problema que os provedores de conteúdo enfrentam na limitação da quantidade de informação que podem oferecer em um meio de comunicação.

Este novo modelo proporcionará uma forma alternativa de sincronização e uma melhor flexibilidade para o fornecimento adicional de conteúdo multimídia, já que esse conteúdo poderá ser visualizado em qualquer dispositivo conectado à Internet.

1.3.2 Objetivos Específicos

Propõe-se oferecer aos produtores de conteúdo maior facilidade para disponibilizar suas programações sincronizadas com a programação principal da TV, ou até mesmo de outras fontes de conteúdo, como a Internet, sem a necessidade de mudança imediata para o sistema de TV Digital e toda sua infraestrutura.

Tendo em vista que a experiência da TV, na maioria das vezes, é uma experiência coletiva e aplicações interativas executadas na mesma podem trazer insatisfação para os demais usuários. O presente projeto tem como finalidade minimizar o abuso da interatividade em primeira tela enviando essas aplicações para dispositivos de segunda tela.

Na perspectiva do usuário, é desejável que este novo modelo apresente uma forma de melhorar sua experiência de consumo de informação em segunda tela, permitindo que ele possa usufruir de conteúdo adicional de forma simples e descomplicada.

1.4 Estrutura do Projeto

Este projeto foi elaborado a partir de estudo bibliográfico e trabalhos recentes na área de sistemas multimídia, interatividade na TV Digital e fundamentos de sincronismo na linguagem NCL, seguido de uma implementação de uma aplicação neste mesmo tema. Assim, o projeto pode ser dividido nas seguintes etapas: estudo bibliográfico, levantamento de requisitos, implementação da aplicação do modelo de sincronismo, testes, análise de problemas e levantamento de possíveis soluções, revisão e conclusão.

No capítulo 2 o leitor será inserido no ambiente, onde haverá o embasamento teórico necessário para o projeto como um todo. Poderá visualizar os principais conceitos e o tema sobre o qual o projeto será elaborado. Será apresentado um modelo de sincronismo, no qual constarão todas as idéias e embasamento necessários para a implementação do projeto.

O capítulo 3 apresenta a solução proposta e modelos de sincronismo utilizado. Este capítulo abordará todas as etapas para a implementação do projeto, tanto na parte implementação do sistema cliente/servidor, como ferramentas que serão utilizadas.

Os testes e problemas serão abordados no capítulo 4. Serão analisados os diversos problemas que poderão ser encontrados a partir dos testes efetuados, bem como possíveis soluções para os mesmos. Por fim, o capítulo 6, fica reservado para conclusões e trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta conceitos e características importantes para o presente projeto, visando um embasamento teórico para se obter um conjunto de requisitos necessários para o alinhamento de timelines do conteúdo multimídia.

Esse alinhamento da linha do tempo do conteúdo multimídia entre TV e outros dispositivos atualmente é feita de diversas formas e utilizando muitas plataformas diferentes. Várias linguagens, como Java e aplicações de XML permitem flexibilidade na criação de aplicações que manipulam conteúdo multimídia.

Hoje, além do suporte de sincronismo oferecido pela linguagem NCL utilizado em aplicações para TV Digital no SBTVD, no mercado existem vários aplicativos de segunda tela para a exibição do conteúdo contendo alguma sincronização com conteúdo da TV. Para isto é preciso estabelecer uma base para criar a ligação e o sincronismo entre as duas fontes de conteúdo.

2.1 Extensible Markup Language e a Nested Context Language

O XML (W3C, 2013) é uma linguagem de marcação recomendada pelo W3C para produção de documentos contendo dados com o objetivo de organizá-los de forma hierárquica. Entre as linguagens baseadas em XML estão: XHTML (W3C, 2000), formato de páginas para web; MathML (W3C, 2014), formato para expressões matemáticas; SVG (W3C, 2011), formato gráfico vetorial.

As principais vantagens de se utilizar o XML, segundo (W3C, 2013), são a facilidade e clareza na leitura de documentos escritos em XML, fácil desenvolvimento de programas capazes de processar documentos XML, possibilidade de criação de tags sem limitação, entre outras. O (W3C, 2013) define que cada documento deve conter uma lógica e uma estrutura física e é composto de unidades chamadas entidades. Um docu-

mento começa com um root ou uma entidade documento. Este documento é composto de elementos, comentários e instruções de processamento que devem ser corretamente aninhados.

Uma aplicação hipermídia é formada por um conjunto de informações distribuídas no tempo e espaço, onde cada aplicação contém, além do seu conteúdo (áudio, vídeo, imagem) a especificação da sincronização espaço-temporal desses objetos de mídia (De Resende Costa, 2010). A Nested Context Language (NCL) é uma linguagem baseada no padrão XML, tendo como base o modelo conceitual hipermídia NCM, segundo (Antonacci, 2000). O NCM, segundo (Soares e Rodrigues, 2005), é um modelo que representa os conceitos estruturais dos dados, assim como os eventos e relacionamentos entre os dados.

Com a NCL é possível obter essa sincronização dos objetos de mídia no tempo e espaço. Além disso pode-se estruturar aplicações declarativas em NCL de forma a não só permitir o sincronismo entre intervalos absolutos em um eixo temporal, mas como permitir esse sincronismo determinado por eventos imprevisíveis definidos independente do ponto neste eixo temporal, como por exemplo, uma ação do usuário por meio de um botão em seu aplicativo ou controle da TV.

De fato, a NCL é uma opção e vem sendo utilizada para desenvolvimento de aplicações para TV Digital para relacionar e sincronizar o conteúdo multimídia vindo do canal de rádio difusão. Mas é possível explorar os conceitos de sincronismo pela utilização de linguagens de marcação declarativa, aliada à possibilidade de apresentação em segunda tela de forma desacoplada do conteúdo principal, de modo a possibilitar uma nova experiência no ambiente de aplicações para segunda tela.

2.2 JavaFX 2

O Java é uma linguagem orientada a objetos lançada na década de 90 (Oracle, 1993). O JavaFX 2 (Oracle, 2011), lançado em outubro de 2011 oferece APIs para desenvolvedores com um conjunto de pacotes gráficos facilitando a manipulação de mídias, como áudio e vídeo. Permite uma maior flexibilidade e facilidade para aplicar efeitos e transformações nos objetos presentes nas aplicações. O JavaFX permite que em poucas linhas de código seja possível exibir um vídeo armazenado localmente ou em um servidor web.

Um exemplo de software desenvolvido baseando-se fortemente no JavaFX é o LoadUI (SmartBear, 2010), que é uma ferramenta de teste e monitoramento open source voltada principalmente para serviços Web. O LoadUI fornece uma interface de usuário e utiliza intensamente os controles incluídos no JavaFX, como caixas de texto, botões, construtores de gráficos, A maioria dos componentes gráficos se utilizam de CSS para alcançar uma aparência desejável (SmartBear, 2010).

Com o surgimento do JavaFX 2.2 lançado em agosto de 2012, foi disponibilizada uma nova gama de recursos adicionais garantindo maior portabilidade, como suporte para Linux. A nova versão traz também suporte para transmissão ao vivo HTTP e eventos de Touch. Diante destas vantagens, aplicações em JavaFX oferecem uma grande capacidade no quesito de processamento de conteúdo multimídia e simplicidade no desenvolvimento, podendo se tornar uma ótima opção para desenvolvimento de aplicações que manipulam conteúdo multimídia.

2.3 Sincronismo na linguagem NCL

A linguagem NCL é baseada na meta-linguagem XML e no modelo hipermídia NCM. Através dela é possível apresentar conteúdo multimídia de diversas maneiras, desde o espaço e tempo em que serão exibidas, como os relacionamentos entre os objetos de mídia.

Na linguagem NCL, os objetos de mídia são definidos pelo elemento <media>. Neste objeto media estão presentes um conjunto de atributos responsáveis por identificar o objeto (atributo id) e o seu conteúdo (atributo src). Como elementos filhos do elemento <media> existem âncoras, que correspondem a uma porção das unidades de informação que compõem o conteúdo do objeto. Essas âncoras são representadas pelo elemento <area> (De Resende Costa, 2010).

Vale observar na Figura 2.1, os atributos “id”,“begin” e “end” que representam, respectivamente o identificador do elemento <area>; qual o exato momento, em segundos, que correspondem a esta âncora; e o exato momento em que se encerra a ligação com essa âncora. Esses atributos são responsáveis para determinar o limite de tempo(início e o fim) em que uma mídia será exibida.

```
43          <media id="background" src="../media/background.png" descriptor="backgroundDesc"/>
44          <media id="animation" src="../media/animGar.mp4" descriptor="screenDesc">
45              <area id="segDrible" begin="12s"/>
46              <area id="segPhoto" begin="41s"/>
47              <area id="segIcon" begin="45s" end="51s"/>
48          </media>
49          <media id="choro" src="../media/choro.mp4" descriptor="audioDesc"/>
50          <media id="drible" src="../media/drible.mp4" descriptor="dribleDesc"/>
51          <media id="photo" src="../media/photo.png" descriptor="photoDesc">
52              <property name="top"/>
53          </media>
```

Figura 2.1: elementos de media no documento NCL. (De Resende Costa, 2010)

Outro componente na linguagem NCL que é importante observar são os chamados descritores, representados pelo elemento <descriptor>. Esse elemento especifica a forma como o objeto de mídia definido pelo elemento <media> será apresentado.

```
23          <descriptorBase>
24              <descriptor id="backgroundDesc" region="backgroundReg"/>
25              <descriptor id="screenDesc" region="screenReg"/>
26              <descriptor id="photoDesc" region="frameReg" explicitDur="5s">
27                  <descriptorParam name="transparency" value="0.6"/>
28              </descriptor>
29              <descriptor id="audioDesc"/>
30              <descriptor id="dribleDesc" region="frameReg" transIn="trans1" transOut="trans2"/>
31              <descriptor id="iconDesc" region="iconReg" explicitDur="6s"/>
32              <descriptor id="shoesDesc" region="shoesReg"/>
33              <descriptor id="formDesc" region="formReg" focusIndex="1" explicitDur="15s"/>
34              <descriptor id="intDesc" region="intReg"/>
35          </descriptorBase>
```

Figura 2.2: elementos descritores no documento NCL. (De Resende Costa, 2010)

Na figura 2.2, o elemento <descriptor> possuí alguns atributos importantes, como “id”, “region” e “explicitDur”. Esses atributos representam, respectivamente, o identificador do elemento <descriptor>, o elemento <region> que é responsável por determinar a região em que será apresentado o objeto de mídia, e a duração explícita de quanto tempo o objeto de mídia relacionado será apresentado.

Com a utilização de mais dois elementos já é possível criar um relacionamento de sincronismo entre objetos de mídia. Esses dois elementos são o <causalConnector> e <Link> (Figura 2.3). O elemento <link> agrupa elementos que especificam os participantes da relação. Já o elemento <causalConnnector> é responsável por especificar a semântica da relação de um elo (link), ou seja, para que haja uma relação, certas condições devem ser satisfeitas para que ações possam ser disparadas (Romualdo, 2010).

Neste caso, os atributos “component” e “interface” devem ser observados. O atri-

```
5▼      <connectorBase>
6▼          <causalConnector id="onBeginStart_delay">
7               <connectorParam name="delay"/>
8               <simpleCondition role="onBegin"/>
9               <simpleAction role="start" delay="$delay" max="unbounded" qualifier="par"/>
10          </causalConnector>
11▼         <causalConnector id="onBeginStart">
12              <simpleCondition role="onBegin"/>
13              <simpleAction role="start" max="unbounded" qualifier="par"/>
14          </causalConnector>
15▼         <causalConnector id="onEndStop">
16              <simpleCondition role="onEnd"/>
17              <simpleAction role="stop" max="unbounded" qualifier="par"/>
18          </causalConnector>
19      </connectorBase>
20  </head>
21  <body>
22▼ <...>
23▼     <link  id="lMusic" xconnector="onBeginStart_delay">
24          <bind role="onBegin" component="animation"/>
25          <bind role="start" component="choro">
26              <bindParam name="delay" value="5s"/>
27          </bind>
28      </link>
29▼     <link  id="lDrible" xconnector="onBeginStart">
30          <bind role="onBegin" component="animation" interface="segDrible"/>
31          <bind role="start" component="drible"/>
32      </link>
33▼     <link  id="lPhoto" xconnector="onBeginStart">
34          <bind role="onBegin" component="animation" interface="segPhoto"/>
35          <bind role="start" component="photo"/>
36      </link>
37▼     <link  id="lEnd" xconnector="onEndStop">
38          <bind role="onEnd" component="animation"/>
39          <bind role="stop" component="choro"/>
40      </link>
```

Figura 2.3: elementos links e conectores no documento NCL. (De Resende Costa, 2010)

buto “component” determina qual elemento <media> será acionado através do atributo “id” atribuído ao elemento <media> enquanto que o atributo “interface” determina a partir de qual âncora especificada por um elemento de <area> também através do id atribuído.

Com a utilização desse conjunto de elementos e atributos é possível criar um certo grau de sincronismo e relacionamento entre os objetos de mídia.

3 Solução Proposta

Atualmente, existem diversas aplicações proprietárias que possibilitam o sincronismo de uma programação ou canal da TV em dispositivos móveis. Alguns exemplos podem ser citados, como o ESPNSync (ESPN, 2014); aplicativo da emissora Band (Band, 2014); aplicativo do Placar Uol (Uol, 2014); aplicativo da emissora Globo (Globo, 2014); entre outros. Porém, estes aplicativos são restritos aos propósitos das emissoras e aos seus próprio canais, tornando assim uma aplicação limitada. Basicamente essas aplicações funcionam como um portal de notícias, referentes a respectiva programação e não possuem um padrão de sincronismo com a mesma. Se um telespectador quiser sincronizar seu dispositivo com cada um os canais terá que adquirir todos os aplicativos e executá-los simultaneamente.

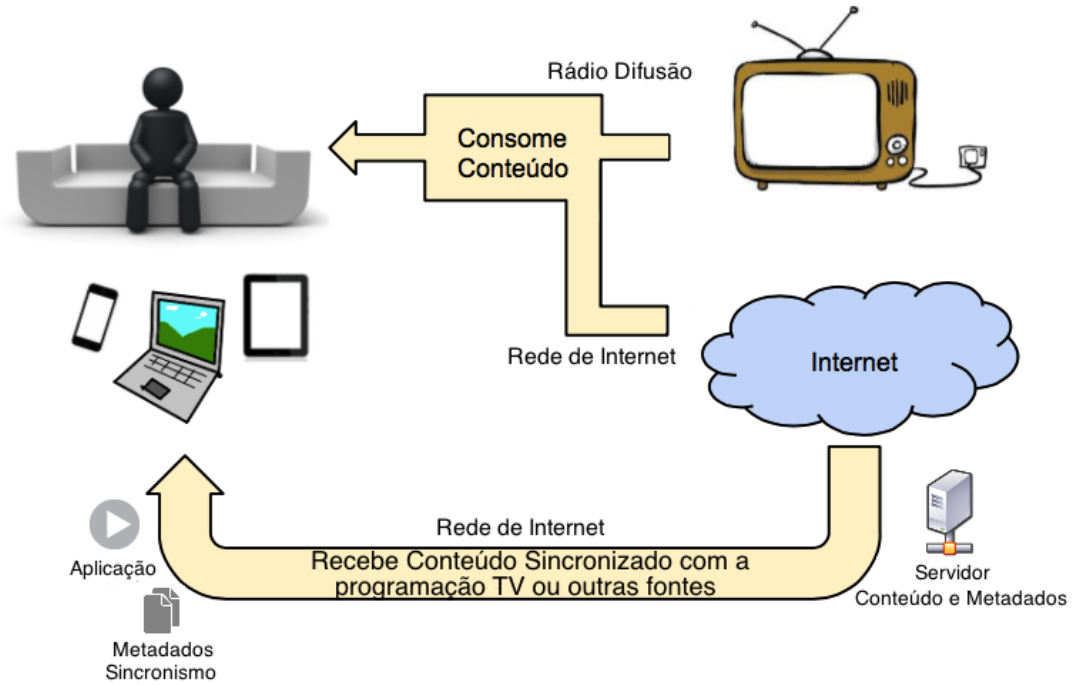

Figura 3.1: Ilustração de modelo de sincronismo

Este capítulo apresenta o modelo de sincronismo do conteúdo através da Internet. A proposta se resume em utilizar algumas funcionalidades baseadas na NCL, para que seja possível o sincronismo do conteúdo multimídia com a programação atual que está sendo consumida pelo telespectador.

Conforme a Figura 3.1, o telespectador que está usufruindo de algum conteúdo pela TV, ou até mesmo através de outra fonte de informação, como a Internet, poderá associar automaticamente ou manualmente, um segundo dispositivo com a programação atual. Com isso, a partir do momento em que esse dispositivo está associado, ele se torna uma segunda tela. Essa associação será feita por meio da utilização de um servidor de conteúdo armazenado, aplicação para segunda tela que deverá ser executada no dispositivo secundário e um conjunto de metadados que serão baixados no momento da sincronização.

Através da segunda tela, e enquanto o tempo decorre, será possível receber o conteúdo adicional sincronizado referente (ou não, dependerá do responsável pela programação que está sendo exibida) à programação assistida.

Baseando-se nos elementos da linguagem NCL citados no capítulo anterior, foi levantado a possibilidade de determinar os primeiros elementos de sincronismo de mídia no presente projeto.

3.1 Modelagem do sincronismo e Implementação

Esta seção descreve a estrutura e implementação do presente projeto, bem como as ferramentas e softwares utilizados. O projeto, deste ponto em diante, será nomeado como SyncTV.

O Projeto está baseado em uma comunicação entre o cliente desenvolvido na linguagem Java e um serviço web, desenvolvido em PHP (Lerdorf, 1995), responsável por facilitar a manipulação do conteúdo multimídia e seus metadados para sincronização.

3.1.1 Definição dos metadados de sincronismo

Inicialmente, através do estudo feito da linguagem NCL, foram definidas algumas características importantes deste modelo de sincronismo. Levando em conta a observação do

método de sincronismo utilizado em NCL, foram definidas duas estruturas de documentos XML para especificação dos metadados de sincronismo. Uma delas é mostrada na figura abaixo.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <synctv url="http://localhost/synctv/syncfiles/synctv.xml">
3    <channel name="channel1">
4      <programming name="prog1" date="15/12/2014" hour="11:31" available="02:00" url="http://
       localhost/synctv/syncfiles/uploads/prog1" type="synctv" service="linearTV"/>
5      <programming name="prog2" date="21/12/2014" hour="08:41" available="02:00" url="http://
       localhost/synctv/syncfiles/uploads/prog2" type="synctv" service="linearTV"/>
6    </channel>
7    <channel name="channel2">
8      <programming name="evt1" date="02/12/2014" hour="15:05" available="02:30" url="http://
       localhost/synctv/syncfiles/uploads/evt1" type="synctv"  service="linearTV"/>
9      <programming name="evt2" date="23/12/2014" hour="18:55" available="02:00" url="http://
       localhost/synctv/syncfiles/uploads/evt2" type="synctv" service="linearTV"/>
10     <programming name="evt3" date="22/10/2014" hour="22:00" available="04:20" url="http://
       localhost/synctv/syncfiles/uploads/evt3" type="synctv" service="linearTV"/>
11   </channel>
12   <channel name="channel3">
13     <programming name="program1" date="19/12/2014" hour="20:30" available="01:10" url="http://
       localhost/synctv/syncfiles/uploads/program1" type="synctv" service="linearTV"/>
14   </channel>
15 </synctv>
```

Figura 3.2: estruturas dos metadados SyncTV

Nesta primeira estrutura (Figura 3.2), nomeada como “syncfiles.xml”, constam os elementos <synctv>,<channel> e <programming>. O elemento <synctv> é o elemento root do documento. Nele está contido o atributo “url” que especifica a localização deste documento XML.

O elemento <channel> determina o canal, para o caso da TV, ou até mesmo um outro conteúdo oriundo de outros meios, como por exemplo pela web. Este conteúdo é identificado pelo atributo “name” que por sua vez sempre será distinto dos atributos “name” dos demais elementos <channel>.

Por fim, o elemento <programming> representa a programação que está sendo exibida no momento e contém os metadados referentes à mesma. Esses metadados são especificados pelos atributos “name”, “date”, “hour”, “avaiable”, “url”, ‘type” e “service”.

Inicialmente, para o atributo “name”, são seguidas as mesmas características do atributo “name” do elemento <channel> que deverá ser único para sua identificação, ou seja, não poderá haver atributos “name” de valor idêntico entre elementos <programming>, logo programações devem ter nomes distintos, devido a estrutura hierárquica definida mais a frente.

O atributo “date” e “hour” especificam, respectivamente o dia e o horário de início em que aquela programação definida pelo elemento <programming> será apresentada, mediante o prévio sincronismo com o conteúdo principal que está sendo consumido pela TV ou outra fonte.

No atributo “avaiable” é definida a duração ou disponibilidade daquela programação especificada pelo elemento <programming>, ou seja, a partir do momento em que o horário atual ultrapassa a disponibilidade (valor do atributo “hour” + “avaiable”) não será mais analisado o conteúdo referente a esta programação para sincronização.

Cada programação será especificada por um documento XML próprio. Esse documento será localizado através do atributo “url” no elemento <programming>. Tanto os conteúdos como os documentos foram estabelecidos de forma hierárquica, que será explicada mais a frente.

Por fim, restam apenas os atributos “type” e “service”. O atributo “type” identifica a qual tipo de conteúdo pertence este elemento <programming>, que no caso é somente “synctv”, o alvo deste projeto. O tipo de programação poderia ser, por exemplo, um documento NCL ou outro modelo de apresentação de conteúdo que seria integrado a esta estrutura de metadados. O atributo “service” define o tipo de serviço ao qual pertence o elemento <programming>. Neste caso, o valor de atributo “linearTV” especifica que o conteúdo será apresentado conforme o tempo decorre durante a programação exibida na tela principal (TV ou outros). Nada impede que exista um valor de atributo que determina que o conteúdo será exibido conforme a solicitação do usuário da aplicação, sob demanda.

O documento “syncfiles.xml” pode ser considerado o documento principal da estrutura de metadados já que ele será responsável por especificar os canais, ou fluxos principais de conteúdo, (<channel>) e o conjunto de programações (<programming>) e características pertencentes a estes canais, podendo também agregar programações que se comportam de forma distinta das que serão abordadas com mais ênfase neste projeto.

Em cada elemento <programming>, existe um atributo “url” obrigatoriamente, e é esse atributo que especifica a localização do próximo documento XML que contém todas as informações referentes à programação. Para cada programação haverá um documento

XML contendo os metadados de sincronismo de seus objetos de mídia.

```
<?xml version="1.0" encoding="utf-8"?>
<programming name="prog1" date="15/12/2014" hour="11:31" available="02:00" url="http://localhost/synctv/
syncfiles/uploads/prog1" type="synctv" service="linearTV">
  <media name="audio1.mp3" offset="2" dur="30" type="audio" url="http://localhost/synctv/syncfiles/uploads/
  prog1/audio1/audio1.mp3"/>
  <media name="audio2.mp3" offset="35" dur="10" type="audio" url="http://localhost/synctv/syncfiles/uploads/
  prog1/audio2/audio2.mp3"/>
  <media name="image1.jpg" offset="50" dur="25" type="imagem" url="http://localhost/synctv/syncfiles/uploads/
  prog1/image1/image1.jpg"/>
  <media name="image2.jpg" offset="10" dur="70" type="imagem" url="http://localhost/synctv/syncfiles/uploads/
  prog1/image2/image2.jpg"/>
  <media name="video1.mp4" offset="230" dur="120" type="video">
    <properties resolution="240p" url="http://localhost/synctv/syncfiles/uploads/prog1/video1/240p.mp4"/>
    <properties resolution="360p" url="http://localhost/synctv/syncfiles/uploads/prog1/video1/360p.mp4"/>
    <properties resolution="480p" url="http://localhost/synctv/syncfiles/uploads/prog1/video1/480p.mp4"/>
    <properties resolution="720p" url="http://localhost/synctv/syncfiles/uploads/prog1/video1/720p.mp4"/>
    <properties resolution="1080p" url="http://localhost/synctv/syncfiles/uploads/prog1/video1/1080p.mp4"/>
  </media>
  <media name="video2.mp4" offset="400" dur="35" type="video">
    <properties resolution="240p" url="http://localhost/synctv/syncfiles/uploads/prog1/video2/240p.mp4"/>
    <properties resolution="360p" url="http://localhost/synctv/syncfiles/uploads/prog1/video2/360p.mp4"/>
    <properties resolution="480p" url="http://localhost/synctv/syncfiles/uploads/prog1/video2/480p.mp4"/>
    <properties resolution="720p" url="http://localhost/synctv/syncfiles/uploads/prog1/video2/720p.mp4"/>
    <properties resolution="1080p" url="http://localhost/synctv/syncfiles/uploads/prog1/video2/1080p.mp4"/>
  </media>
  <media name="page1" offset="540" dur="160" type="webpage" url="http://www.ufjf.br/cursocomputacao/"/>
  <media name="page2" offset="825" dur="220" type="webpage" url="http://www.ufjf.br/portal/universidade/ufjf/"
  />
</programming>
```

Figura 3.3: estrutura dos metadados das programações

Como na estrutura “syncfiles.xml”, o documento da figura acima (Figura 3.3) também possui o elemento <programming> com os mesmos atributos “name”, “date”, “hour”, “avaiable”, “url”, “type” e “service”. O documento XML acima é identificado pelo nome da programação, que neste caso é “prog1.xml”. Também neste documento XML referente a programação, existe apenas mais dois elementos: <media> e <properties>. O elemento <media> especifica os objetos de mídia que farão parte da programação. Dentre seus atributos estão: ?name?, “offset”, “dur”, “type”, “url”. O atributo “name”, assim como nos outros elementos, identifica o objeto de media dentro do elemento <programming> obedecendo a unicidade. Os atributos “offset” e “dur” especificam, respectivamente, o momento exato (em segundos) em que o objeto de mídia deverá ser apresentado e por quanto tempo isso ocorrerá. O atributo “type” especifica o tipo da mídia. Os tipos de objeto de mídia são imagens, áudio, vídeo e páginas web. Devido as limitações do JavaFX 2.0, os formatos suportados são:

- **Vídeo:** MPEG4 H.264/AVC, com áudio AAC

- **Áudio:** MP3 (MPEG-1/2 Audio Layer 3) e AAC(Advanced Audio Coding)

- Imagem: JPEG/JPG (Joint Photographic Experts Group), PNG (Portable Network Graphics) e GIF (Graphics Interchange Format)

- Páginas web: HTML (HyperText Markupt Language)

O atributo “url” especifica a localização do objeto de mídia. Entretanto para o elemento <media> que possui atributo “type” igual a video, não possuirá este atributo. Por fim o elemento <properties> especifica algumas características para o objeto de mídia do tipo vídeo. Cada elemento de vídeo possui cinco elementos <properties> filhos. Cada elemento <properties> possui dois atributos: “resolution” e “url”. O atributo “url” especifica a localização do objeto de mídia e o atributo “resolution” representa a qualidade da resolução usada na transcodificação. Cada vídeo possui cinco opções de resolução que são: 1080p (1920x1080), 720p (1280x720), 480p (854x480), 360p (640 x 360) e 240p (426 x 240). Esta resolução será ajustada de acordo com o dispositivo em que a aplicação cliente está executando.

3.1.2 Servidor Web

Para implementação do servidor foram utilizadas algumas ferramentas, dentre elas, são: servidor Xampp 1.8.3-5 (Apache-Friends, 2002), sistema de gestão de conteúdo Joomla 3.3.6 (Joomla, 2005). O Xampp é um servidor independente de plataforma que oferece uma base de dados MySQL (MYSQL, 2009), servidor Web Apache (Foundation, 1995) e interpretador das linguagens PHP (Lerdorf, 1995) e Perl (Wall, 1987).

O Joomla oferece uma maior facilidade para o gerenciamento do portal web, é desenvolvido em PHP e pode facilmente ser executado no Apache Web e utiliza o banco de dados MySQL.

Primeiramente, o portal web foi desenvolvido de forma a oferecer um ambiente amigável para que o conteúdo multimídia a ser sincronizado possa ser cadastrado de forma simples e rápida.

Dentre as funcionalidades implementadas, existe um módulo de login, onde o usuário (gerente responsável pelo canal, grupo de canais ou conteúdo que será exibido na primeira tela) se cadastra, cria e configura seus canais. Depois de possuir pelo menos

um canal cadastrado, o usuário do sistema poderá cadastrar suas programações e enviar seus conteúdos multimídia, bem como definir suas datas e horários de apresentação na segunda tela.

À medida que o processo de cadastro de canal, programação e mídia vão sendo efetuados, os metadados vão sendo criados de forma transparente no documento "syncfiles.xml" e nos demais documentos XML referentes a cada programação, obedecendo uma estrutura hierárquica.

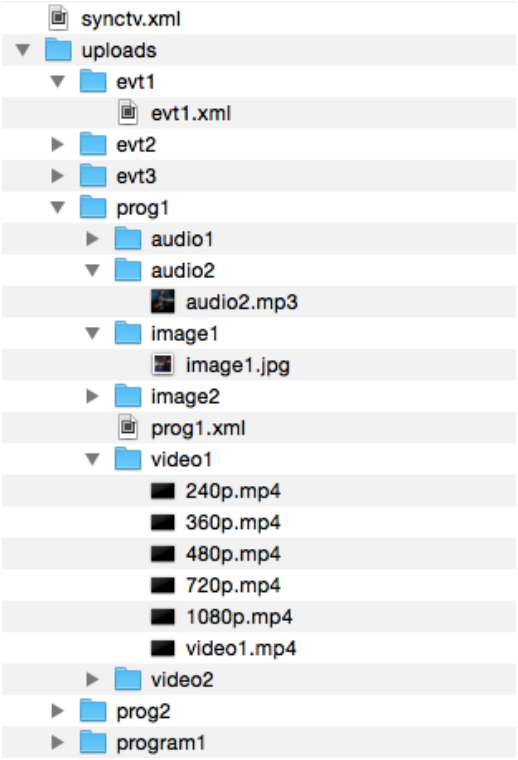

Figura 3.4: Hierarquia dos arquivos e diretórios no servidor

A estrutura hierárquica se resume em um documento XML principal "syncfiles.xml" contendo os metadados referentes a todos os canais e suas programações. No sistema de arquivos do servidor, dentro do diretório "uploads", cada programação será representada por um diretório com seu respectivo nome de identificação. No diretório de cada programação contém um diretório para cada objeto de mídia cadastrado. Para cada diretório de objeto de mídia, com exceção das mídias do tipo vídeo (mp4), é contido de sua mídia referente. No caso de mídias do tipo vídeo, quando cadastradas no portal e enviadas para o servidor, os objetos serão transcodificados para cinco formatos e resoluções com suas respectivas qualidades, do menor (240p) para o maior (360p).

Para auxiliar a apresentação dos metadados e seus valores no portal é utilizado o banco de dados MySQL, sempre de forma consistente com os dados dos documentos XML.

A transcodificação dos objetos de mídia é iniciada por um script quando o upload para o servidor é completado. Essa transcodificação é feita por meio do software FFmpeg (Bellard; Niedermayer, 2014), que possui um conjunto de softwares livres e bibliotecas de código aberto, bem como libavcodec, biblioteca de codec áudio e vídeo, e libavformat, um multiplexador/demultiplexador de conteúdo de áudio e vídeo.

A seguir, será apresentado o workflow do provedor de conteúdo ao cadastrar seu canal de transmissão no servidor web:

1- Cadastrando Canal: Inicialmente, será preciso cadastrar o canal, ou fonte de conteúdo que estará disponível para sincronização em segunda tela. Para isto, basta clicar no do menu esquerdo em "Cadastrar Canal".

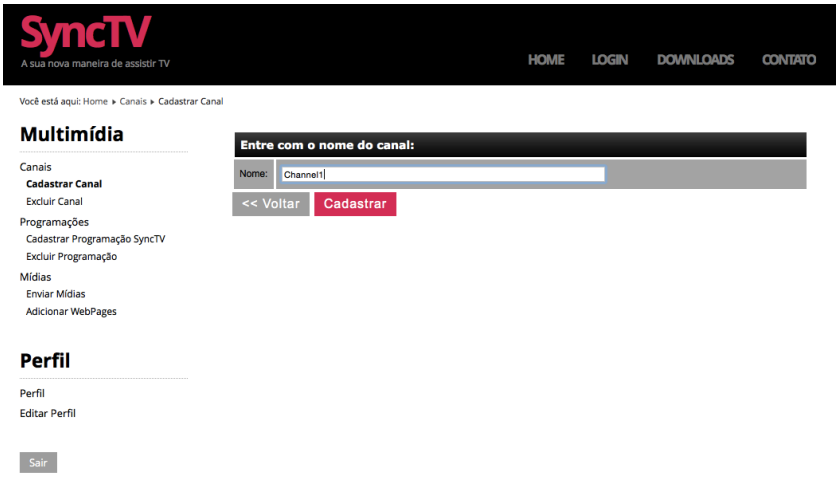

Figura 3.5: Cadastrando Canal. Primeira Etapa

Em seguida, deverá ser preenchido o nome do canal a ser cadastrado e clicar no botão "Avançar" (Figura 3.5).

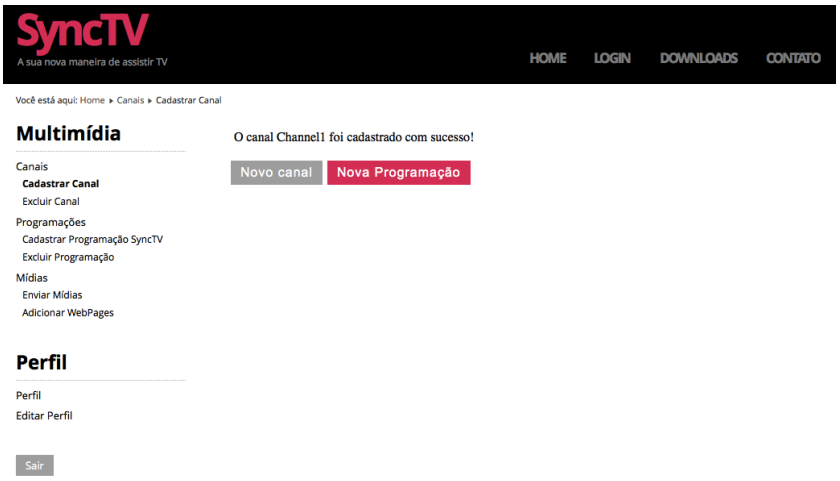

Figura 3.6: Cadastrando Canal. Segunda Etapa

Se o cadastro foi concluído com sucesso, será apresentado a tela acima (Figura 3.6).

Para visualizar os canais já cadastrados, basta clicar no menu esquerdo em “Canais”.

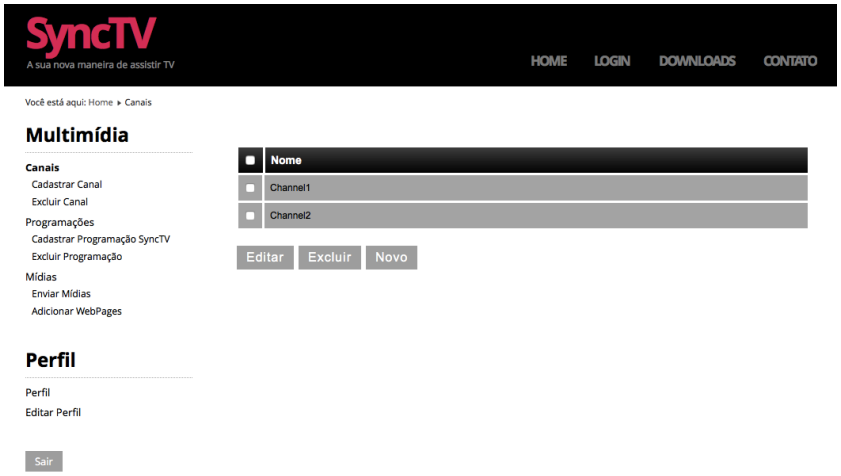

Figura 3.7: Listando Canais Cadastrados

2- Cadastrando Programação: O Próximo passo é o cadastro da programação. Para isto basta clicar no botão “Nova Programação” como mostra a Figura 3.6 ou no menu esquerdo em “Cadastrar Programação SyncTV”.

Figura 3.8: Cadastrando Programação. Primeira Etapa

Nessa etapa, é necessário preencher os campos mostrados na Figura 3.8. Selecionar a qual canal que a programação que está sendo cadastrada pertence, nome da programação, data e hora, qual o tipo de serviço, disponibilidade (ou tempo de duração) da programação, fuso horário. Por fim, depois de preenchido todos os campos, basta clicar no botão “Avançar”.

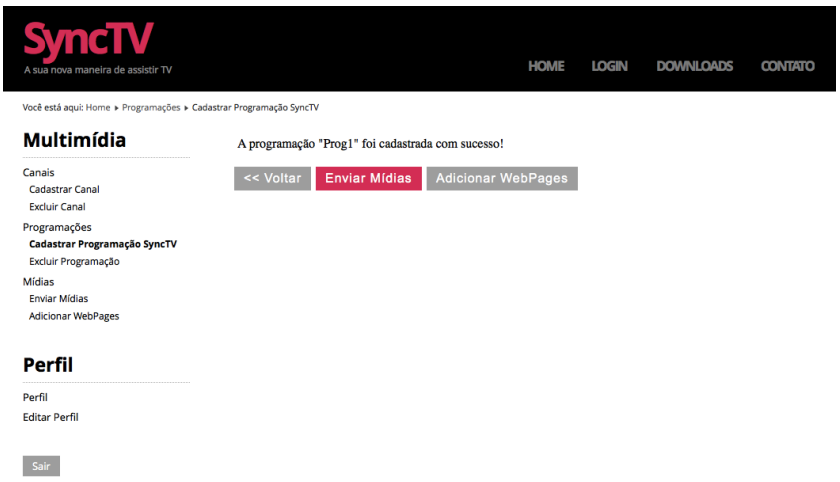

Figura 3.9: Cadastrando Programação. Segunda Etapa

Se a programação foi efetuada com sucesso, será apresentado a tela acima (Figura 3.9). Para listar as programações que já foram cadastradas, basta clicar no menu "Programações".

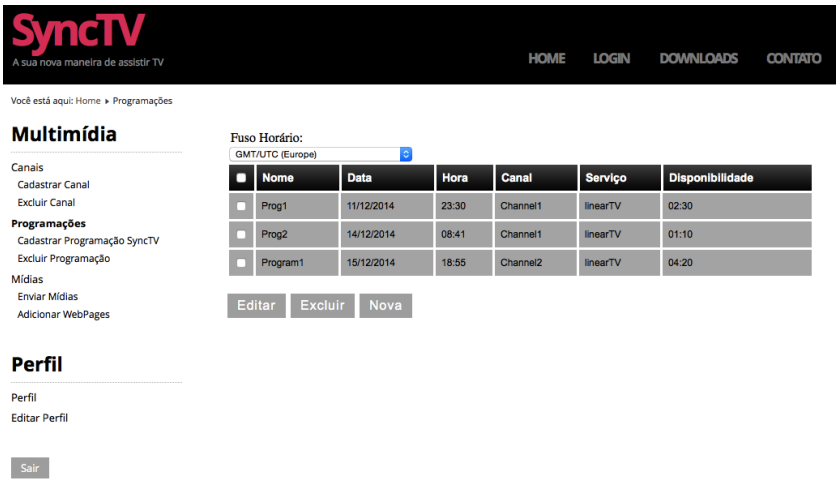

Figura 3.10: Listando Programações Cadastradas

Para editar alguma programação, e necessário marcar as programações correspondentes e clicar no botão "Editar" na tela acima (Figura 3.10).

3- Enviando Mídias: A última etapa é vincular as mídias a sua programação correspondente. Para isto basta clicar no botão "Enviar Mídias" mostrado na Figura 3.9 ou clicar no menu esquerdo "Enviar Mídias".

Figura 3.11: Enviando Mídias. Primeira Etapa

Como mostrado acima (Figura 3.11), é necessário escolher o canal e a programação antes de enviar as mídias. Após selecionado, clicar no botão "Escolher Arquivos".

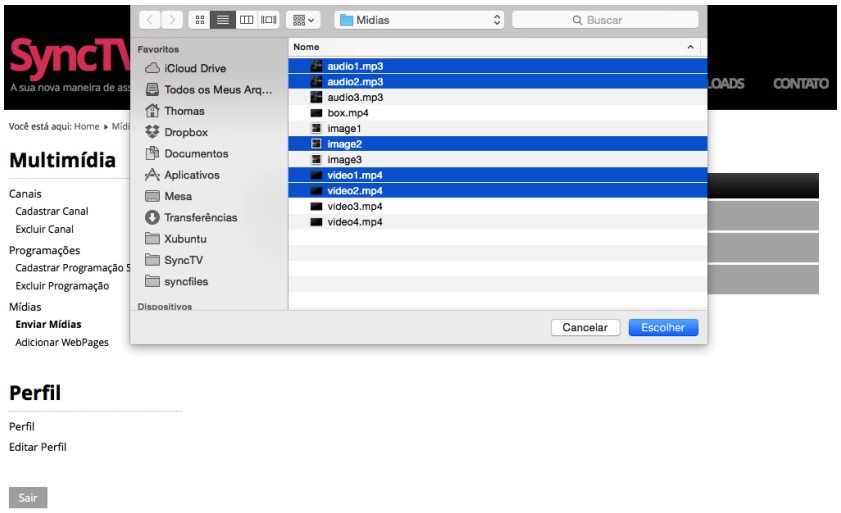

Figura 3.12: Enviando Mídias. Segunda Etapa

Os Arquivos poderão ser do tipo video(mp4), áudio(mp3, aiff ou aac) ou imagem(jpg/jpeg, png ou gif) e poderão ser selecionados várias mídias.

Figura 3.13: Enviando Mídias. Terceira Etapa

Em seguida, depois de selecionado as mídias, basta clicar no botão "Avançar" (Figura 3.13).

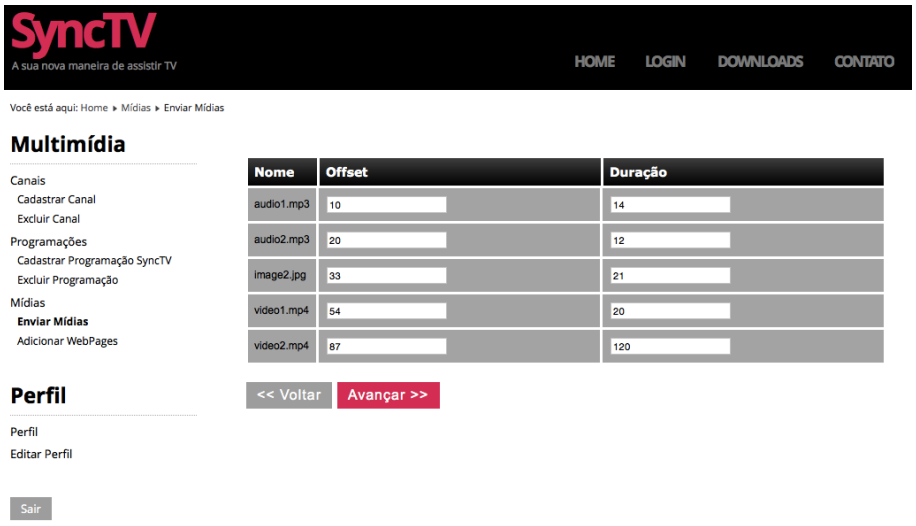

Figura 3.14: Enviando Mídias. Quarta Etapa

Agora é preciso definir o offset, que será o tempo (em segundos) em que cada mídia deverá ser exibida na segunda tela, e por quanto tempo ela permanecerá sendo exibida (também em segundos). Depois de preenchido os campos, basta clicar no botão "Avançar" (Figura 3.14).

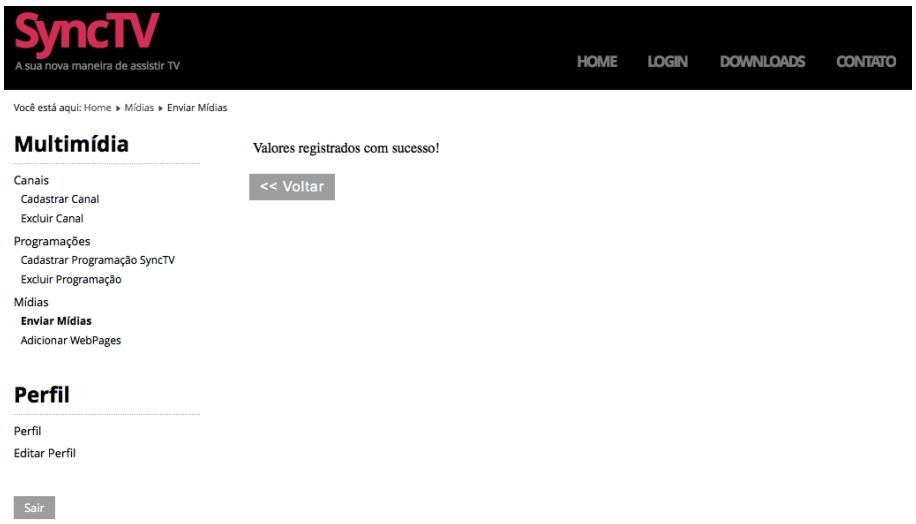

Figura 3.15: Enviando Mídias. Quinta Etapa

Por fim, se todas as mídias forem enviadas com sucesso, será apresentada a tela acima (Figura 3.15). Se for preciso adicionar páginas web como conteúdo multimídia a ser exibido em segunda tela, basta clicar no botão "Adicionar WebPages" como mostra a Figura 3.9 ou clicar em "Adicionar WebPages" no menu esquerdo.

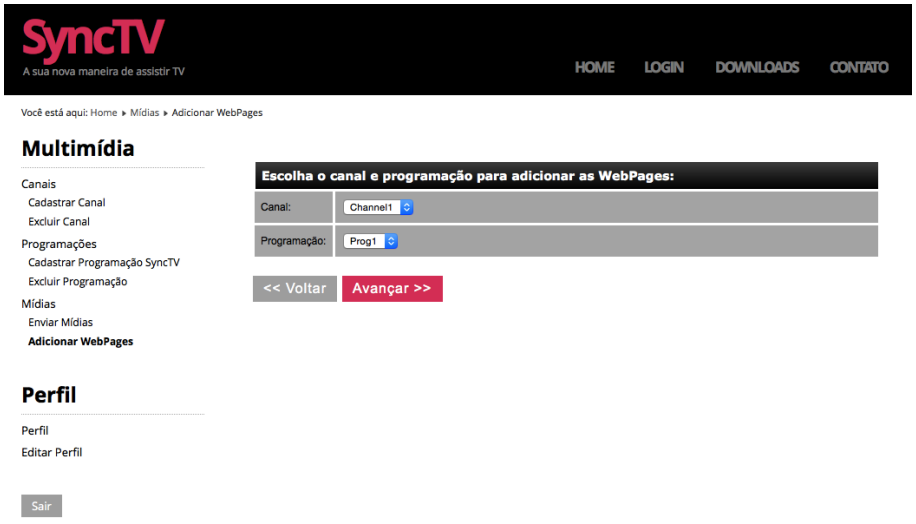

Figura 3.16: Adicionando WebPages. Primeira Etapa

Para adicionar as páginas web, deve-se também selecionar qual canal e programação e clicar no botão "Avançar" (Figura 3.16).

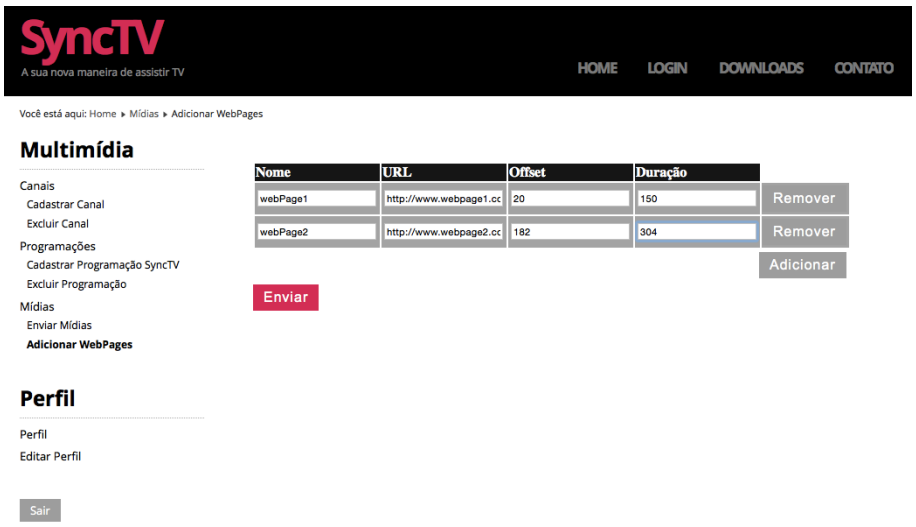

Figura 3.17: Adicionando WebPages. Segunda Etapa

Em seguida, preencher os valores de offset e duração referentes a cada webpage. Se necessário inserir mais campos para novas webpages, basta clicar no botão "Adicionar" ou se deseja remover campos, clicar no botão "Remover" correspondente. Após concluído o preenchimento, clicar no botão "Avançar" (Figura 3.17).

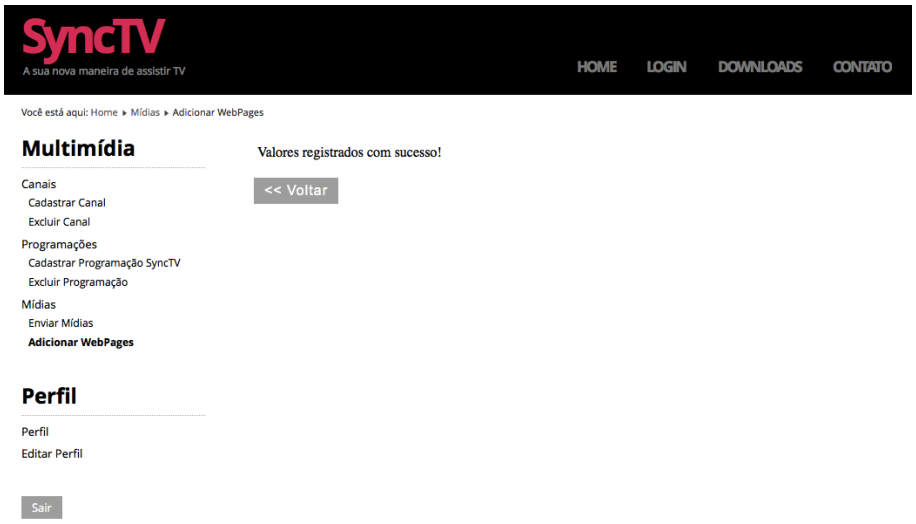

Figura 3.18: Adicionando WebPages. Terceira Etapa

Se tudo ocorreu de forma bem sucedida, será apresentado a tela acima (Figura 3.18).

Por fim, para listar todas as mídias e webpages enviadas, basta clicar no menu esquerdo “Mídias”. Após isso, será necessário selecionar de qual programação se deseja listar as mídias e então será exibida conforme a tela abaixo (Figura 3.19).

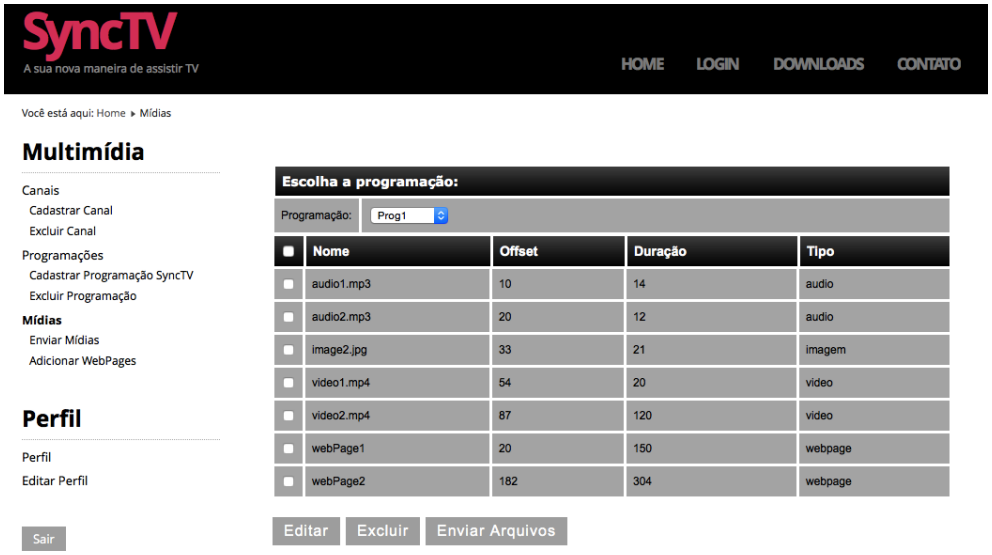

Figura 3.19: Listando Midias já enviadas

Finalmente o conteúdo já poderá ser enviado para a segunda tela e todos os metadados referentes aos canais, programações e mídias já foram gerados dinamicamente.

3.1.3 Aplicação Cliente

A aplicação cliente foi desenvolvida na linguagem Java, juntamente com a plataforma JavaFX, que possui um conjunto de API’s voltadas para o desenvolvimento de softwares com enfoque em manipulação de conteúdo multimídia. O JavaFX permite a criação de elementos gráficos de forma simples e objetiva.

Para o presente projeto, inicialmente foi definido um requisito: os metadados de sincronismo serão baixados no início da conexão. Foram definidos algumas classes para a construção do parsing dos metadados: “SyncTV.java”, “Channel.java”, “Programming.java” e “Midia.java”.

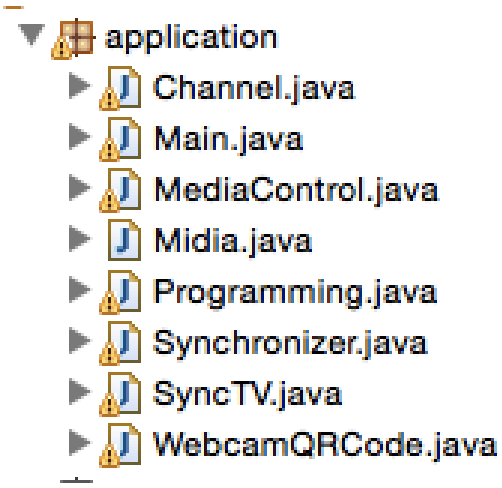

Figura 3.20: Classes da aplicação Cliente

Cada uma das classes acima representa as especificações dos metadados baixados no momento da conexão, cada uma delas representando seus respectivos elementos. “SyncTv.java” representa os metadados referentes ao elemento <synctv> bem como seus atributos “url” e a lista de filhos <Channel>. O mesmo vale para as demais classes “Channel.java”, “Programming.java” e “Midia.java” que também possuem os valores dos elementos <Channel>, <Programming> e <media> especificados nos documentos XML.

A classe “MediaControl.java” é responsável por controlar a exibição de objetos de mídia do tipo vídeo que estão sendo apresentados, como por exemplo, recursos de play, pause, seek. Os recursos de controle são previamente definidos e passados como objetos para a classe principal. Este tipo de opção somente será disponível em apresentações de objetos de mídia que se iniciam a partir da solicitação do usuário, ou seja, não serão exibidas mediante especificação dos atributos “offset” e “dur” no elemento <media>.

A classe “Synchronizer.java” é responsável por fazer o download dos metadados, executar o parsing e notificar para a aplicação o momento exato em que deverão ser apresentadas as mídias no decorrer do tempo da programação.

A classe principal “Main.java” é responsável pela interface gráfica, como a exibição dos recursos oferecidos pelas API’s do JavaFX, incluindo os objetos de mídia e apresentação de páginas web na aplicação, e a interação do usuário com botões, controles, etc.

A aplicação possui um recurso de identificação de QRCode através da câmera

do dispositivo. Este recurso é acionado e gerenciado pela classe "WebcamQRCode.java". Para execução desse recurso, foi utilizada uma API open-source de processamento de imagens de código de barras chamada ZXing (ZXing, 2014). A análise do QRCode é executada e seu resultado, endereço url juntamente com informação que referencia o conteúdo exibido na tela principal, é passado para a classe principal para validação.

Abaixo será apresentado o fluxo de utilização da aplicação cliente:

1- Iniciando Conexão: Ao executar a aplicação será apresentado a seguinte tela (Figura 3.21).

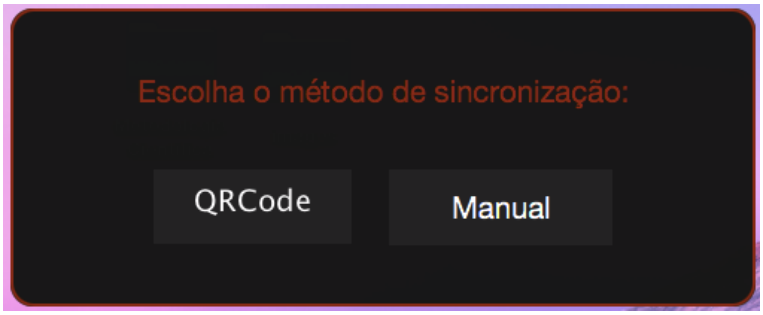

Figura 3.21: Método de conexão

2- Escolhendo método de sincronismo: Se o botão "QRCode" acima (Figura 3.21) for pressionado, será aberto uma janela com a imagem do driver da câmera do dispositivo e então o usuário deverá posiciona-lo para leia e identifique o código QRCode exibido na tela principal (TV ou outro conteúdo que está sendo exibido). Se o valor do QRCode for válido (cadastrado no servidor SyncTV) será iniciado o sincronismo. Outra forma de iniciar a conexão é clicando no botão "Manual".

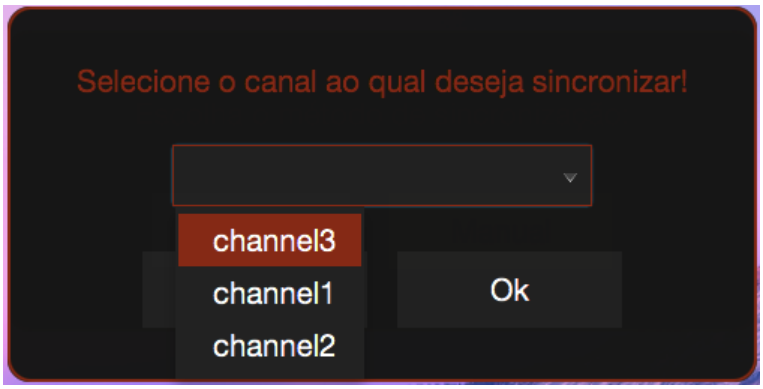

Figura 3.22: Iniciando conexão Manualmente

Nesse caso, como mostra a tela acima (Figura 3.22) o canal ao qual se deseja sincronizar poderá ser escolhido manualmente. Ao selecionar o canal desejado, basta clicar no botão "Ok".

3- Sincronização Iniciada Se a sincronização foi executada com sucesso, tanto de forma manual ou via detecção do QRCode, será exibida a tela abaixo (Figura **??**.

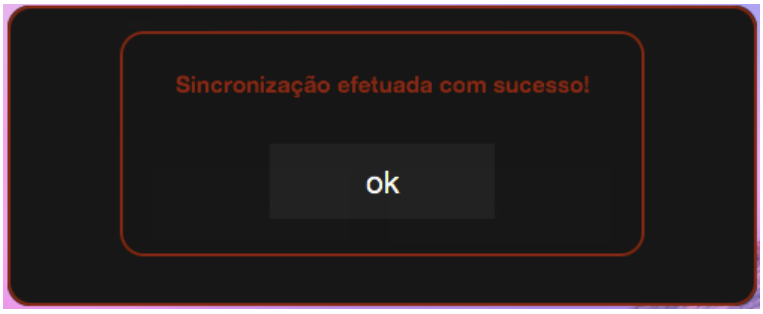

Figura 3.23: Conexao Efetuada com Sucesso

Ao clicar no botão "Ok" (Figura 3.23), será iniciado o player de mídia para receber o conteúdo sincronizado nos momentos pré-estabelecidos pelo provedor de conteúdo no servidor (Figura 3.24).

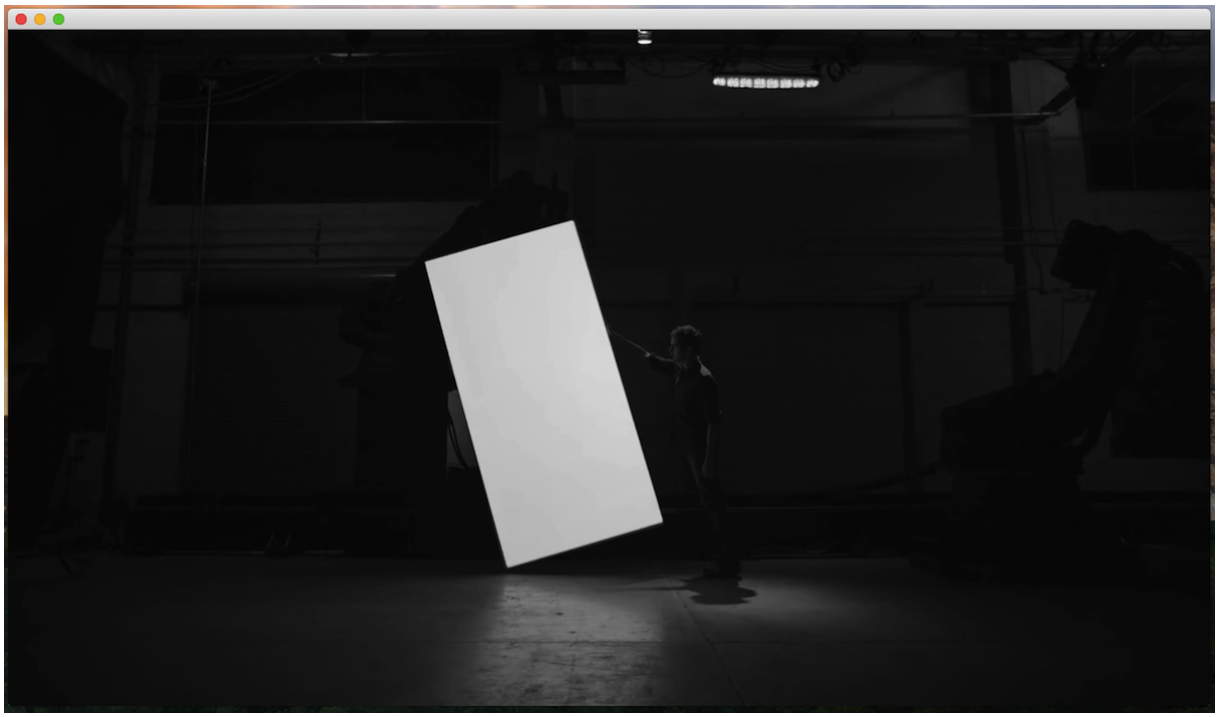

Figura 3.24: Download dos Metadados e sincronismo iniciado

4 Testes

Este capítulo descreve os testes e problemas encontrados, bem como algumas possíveis soluções. Inicialmente, a aplicação não apresentava um fluxo, ou seja, na execução da mesma, já era apresentado uma janela contendo as opções como a opção do início do sincronismo, segundo a vontade do usuário.

Foi decidido que para melhor experiência do usuário da aplicação deveria apresentar um fluxo: escolha no modo de conexão com o canal (manual ou pelo QRCode), conexão, download dos metadados de sincronismo, início da sincronização de forma automática.

Para a sincronização do conteúdo multimídia, é de extrema relevância o horário do sistema em que a aplicação cliente está sendo executada estar ajustado de acordo com o servidor, ou seja, estar correto. Para isso, inicialmente foi definido um "offset" de acordo com o horário UTC/GMT (Coordinated Universal Time). Mas com isso, era preciso o usuário informar os dados do seu fuso horário, ou da sua localização atual sempre que iniciar a sincronização.

Uma opção para solucionar o problema seria utilizar uma API de localização do Java, como por exemplo, a API Location (Nokia, 2014). Por fim, uma terceira opção viável, seria a vinculação do usuário por meio de sessão no servidor, mantendo assim a consistência em questão de horário.

Em uma rede local, foram feitos testes de performance e qualidade de experiência. Utilizando equipamentos de prateleira, um desktop básico como servidor oferecendo streaming RTP (Real-time Transport Protocol) dos objetos de mídia e um roteador TP-Link, obteve-se um resultado satisfatório, com apenas alguns engasgos na apresentação do conteúdo de vídeo.

5 Conclusão

O cenário de desenvolvimento de aplicações para TV Digital necessita de métodos eficientes e fáceis para sua execução, de modo a permitir que autores de diferentes níveis de conhecimento possam participar desta etapa (Josué, 2014). Diante disso, é necessário explorar diferentes meios para criação fornecimento de aplicações e conteúdo multimídia na TV Digital, ou outras fontes.

Através do modelo apresentado neste projeto é possível oferecer uma alternativa de criação e fornecimento de conteúdo independentemente do nível de conhecimento dos autores e administradores de programações de TV, além de oferecer uma nova experiência para o telespectador em sua segunda tela.

Espera-se com este trabalho, o surgimento de novas interatividades e ampliação da experiência do telespectador, não só para usuários de TV Digital e TV analógica, mas como usuários e entusiastas que buscam consumir conteúdo multimídia oriundos de outras fontes como a web.

Com isto, uma possibilidade de trabalho futuro,é a harmonização com um perfil de NCL, ou até mesmo a inclusão de todos os recursos oferecidos pela linguagem NCL, bem como integração de outros meios de fornecimento de aplicações e conteúdo interativo para este contexto, levando em conta portabilidade e praticidade, tanto por parte dos produtores de conteúdo como por parte dos usuários da aplicação. Aplicar também testes mais abrangentes e obter avaliações de QoE e QoS e parâmetros de usabilidade a partir de usuários reais.

Referências Bibliográficas

Antonacci, M. J. **Ncl: Uma linguagem declarativa para especificação de documentos hipermídia com sincronização temporal e espacial**. 2000. Dissertação de mestrado - PUC-Rio - Pontifícia Universidade Católica do Rio de Janeiro.

Apache-Friends. **Xampp**. `https://www.apachefriends.org/index.html`, 2002.

Associação Brasileira de Normas Técnicas, Rio de Janeiro. **NBR 15606-2**, abril 2008.

Band. **Aplicativo band**. `http://www.band.uol.com.br/segundatela/`, 2014.

Bellard, F.; Niedermayer, M. **Ffmpeg site**. `https://www.ffmpeg.org/`, 2014. Acesso em 20 de outubro de 2014.

Brackmann, C. P. **Sistema brasileiro de tv digital**. Pelotas, 2008. Monografia - Universidade Católica de Pelotas - Centro Politécnico.

de Resende Costa, R. M. **Controle do sincronismo temporal de aplicações hipermídia**. 2010. Tese de doutorado - PUC-Rio - Pontifícia Universidade Católica do Rio de Janeiro.

ESPN. **Espn sync**. `http://espn.uol.com.br/aplicativos`, 2014.

Fernandes, J.; Lemos, G. ; Silveira, G. Introdução à televisão digital interativa: Arquitetura, protocolos, padrões e práticas. 2004.

Apache. **Apache server**. `http://www.apache.org`, 1995.

Globo. **aplicativo globo**. `http://app.globoesporte.globo.com/aplicativoglobo/`, 2014.

Guimaraes, R. L.; de Resende Costa, R. M. **Interatividade e sincronismo em tv digital**. 2006. Monografia - PUC-Rio - Pontifícia Universidade Católica do Rio de Janeiro.

IBGE. **Pesquisa nacional por amostra de domicílios**. `http://biblioteca.ibge.gov.br/visualizacao/livros/liv65857.pdf`, 2012.

ITU-T. **Requirements for an application control framework using integrated broadcast and broadband digital television**. Recommendation J.205, International Telecommunication Union, Geneva, jan 2012.

Joomla. **Content management system joomla**. `http://www.joomla.org/about-joomla.html`, 2005.

Josué, M. I. P. **Uso de templates na automação do processo de criação de aplicações t-commerce**. 2014. Monografia - Universidade Federal de Juiz de Fora.

Lerdorf, R. **Php: Hypertext preprocessor**. `http://php.net/docs.php`, 1995.

Oracle. **Mysql**. `https://docs.oracle.com/cd/E17952_01/`, 2009.

Nokia. **Jsr-000179 location api for j2me**. https://jcp.org/aboutJava/communityprocess/final/jsr179/, 2014. Accesso em 01 de Agosto de 2014.

Oracle. **Java platform standart edition 7**. http://docs.oracle.com/javase/7/docs/api/, 1993. Accesso em 5 de Junho de 2014.

Oracle. **Javafx release 2.2.21**. http://docs.oracle.com/javafx/2/overview/jfxpub-overview.pdf, 2011. Accesso em 15 de Junho de 2013.

SmartBear. **Loadui**. http://www.loadui.org/Developers-Corner/technical-overview.html, 2010. Accesso em 15 de Junho de 2013.

Soares, L. F. G.; Rodrigues, R. F. **Nested context model 3.0**. 2005. Monografia - PUC-Rio - Pontifícia Universidade Católica do Rio de Janeiro.

Uol. **Placar uol**. http://esporte.uol.com.br/aplicativos/placar-uol/, 2014.

W3C. **Xhtml 1.0 the extensible hypertext markup language (second edition)**. http://www.w3.org/xhtml1/, 2000.

W3C. **Extensible markup language (xml) 1.0 (fifth edition)**. http://www.w3.org/TR/2008/REC-xml-20081126/, 2008. Accesso em 15 de Junho de 2013.

W3C. **Scalable vector graphics (svg) 1.1 (second edition)**. http://www.w3.org/TR/SVG/, 2011.

W3C. **Mathematical markup language (mathml) version 3.0 2nd edition**. http://www.w3.org/TR/MathML3/, 2014.

Wall, L. **Perl**. http://perldoc.perl.org, 1987.

ZXing. **Zxing open–source project**. https://github.com/zxing/zxing/, 2014. Acesso em 15 de agosto de 2014.