UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RAFAEL KERNI

DESENVOLVIMENTO DE UM CLIENTE ANDROID PARA LEITURA DE NOTÍCIAS E EXECUÇÃO DE UMA RÁDIO ONLINE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2016

RAFAEL KERNI

DESENVOLVIMENTO DE UM CLIENTE ANDROID PARA LEITURA DE NOTÍCIAS E EXECUÇÃO DE UMA RÁDIO ONLINE

Monografia apresentada como requisito parcial para obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná, Campus Pato Branco.

Orientador: Prof. Robison Cris Brito

PATO BRANCO
2016

ATA Nº: 278

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **RAFAEL KERNI**.

Às 10:30 hrs do dia 22 de junho de 2016, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Robison Cris Brito (Orientador), Éden Ricardo Dosciatti (Convidado) e Andréia Scariot Beulke (Convidada), para avaliar o Trabalho de Diplomação do aluno Rafael Kerni, matrícula 1295098, sob o título **Desenvolvimento de um Cliente Android para Leitura de Notícias e Execução de uma Rádio Online**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 10:55 hrs foi encerrada a sessão.

_____
Prof. Robison Cris Brito, M.Sc.
Orientador

_____
Prof. Éden Ricardo Dosciatti, Esp.
Convidado

_____
Profa. Andréia Scariot Beulke, Esp.
Convidada

_____
Profa. Eliane Maria de Bortoli Fávero, M.Sc
Coordenadora do Trabalho de Diplomação

_____
Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

Kerni, Rafael. Desenvolvimento de um cliente Android para leitura de notícias e execução de uma rádio *online*. 2016. 43 f. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2016.

O presente trabalho tem como objetivo desenvolver um aplicativo Android que permita capturar dados de uma rádio *online*, assim como recepção de sua programação, tendo como norte os objetivos específicos, que giram em torno do recebimento de dados via *streaming* de uma rádio *online*; a obtenção de dados do portal da rádio (programação e notícias) e apresentação destes ao usuário; a possibilidade do usuário enviar uma mensagem à rádio. A metodologia utilizada para a elaboração deste trabalho foi a revisão bibliográfica, sendo realizada a sistematização dos materiais selecionados como referência. A relevância da elaboração deste trabalho vem ao encontro da necessidade criada, com o aumento do uso de smartphones, de reproduzir rádios por meio de aplicativos, possibilitando maior alcance destas.

Palavras-chave: Radio *Online*. RSS. Notícias. Smartphone. Android.

ABSTRACT

Kerni, Rafael. Development of an Android client for reading news and executing an online radio. 2016. 43 f. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2016.

This study aims to develop an Android application that allows capture data from an online radio as well as reception of its programming, with the north the specific objectives, which revolve around the receiving data by streaming an online radio; obtaining radio portal data (programming and news) and presenting these to the user; the user's ability to send a message to the radio. The methodology used for the preparation of this work was the literature review, being carried out the systematization of the materials selected as reference. The relevance of the preparation of this work is to meet the need created by the increased use of smartphones, playing radios through applications, allowing greater scope of these.

Keywords: Radio *Online*. RSS. News. Smartphone. Android.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

RSS	Rich Site Summary ou Really Simple Syndication
XML	eXtensible Markup Language
PET	Programa de Educação Tutorial
UCS	Universidade de Caxias do Sul
UML	Unified Modeling Language
ADT	Android Developer Tools
IDE	Integrated Development Environment

LISTA DE LISTAGENS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais do trabalho, os objetivos, a justificativa assim como a estrutura em que o trabalho se apresenta.

1.1 CONSIDERAÇÕES INICIAIS

Por muitos anos o rádio foi o meio de comunicação mais comum, dado o baixo custo para o desenvolvimento de um transmissor, assim como a simplicidade dos receptores. Atualmente, com tecnologias, como, por exemplo, TV e Internet, o rádio, deixou de ser o meio de comunicação mais comum. Uma pesquisa feita pelo Ministério das Comunicações mostra que em 2005 existiam 1.915 emissoras outorgadas, sendo que em 2010 esse número chegou a 2.602. Mesmo assim, outra pesquisa realizada em 2009 mostra que 87,9% dos domicílios brasileiros têm rádio. Esse índice pode ser ainda maior por conta das novas mídias capazes de receber sinal de rádio, tais como *MP3 Players*, celulares, sons automotivos, entre outros. (TUDO RÁDIO, 2015)

Com o surgimento da Internet, mudou também a forma de se ouvir rádio, podendo este ser feito pela recepção de *Streaming* que são dados digitais transmitidos pela Internet, podendo ser incluído neste termo o áudio e vídeo digitalizados.

Anteriormente à tecnologia *streaming*, o usuário precisava fazer o *download* do arquivo completo para poder executá-lo. Com a tecnologia *streaming*, é possível que a mídia seja executada conforme é recebida pelo usuário, em tempo real, ou muito próximo disso, permitindo uma execução sob demanda. Para garantir o desempenho, é feito inicialmente *buffer* na memória, caso a conexão com o servidor que provém o *streaming* fique temporariamente lento. Existem muitos dispositivos aptos a receberem *streaming* bem como executá-los, desde os computadores até os smartphones.

Além dos dados multimídia que podem ser recebidos via *Streaming*, existe um padrão para disseminação de notícias pela Internet, o RSS (*Rich Site Summary* ou *Really Simple Syndication*). Muitos sites, em especial, os que provêm informações como rádios *online*, fornecem também notícias em formato de texto usando este padrão.

RSS é uma forma simplificada de apresentar o conteúdo de um site de notícias. Um documento RSS é feito na linguagem XML e geralmente exibe o grande volume de informações existente em uma página na Internet de forma resumida.

Com a popularização dos dispositivos móveis com os smartphones, este passa a ser um cliente potencial para reprodução de dados digitais de áudio recebidos via *streaming*, assim como a recepção de notícias via RSS.

1.2 OBJETIVOS

A seguir serão apresentados os objetivos gerais e específicos do presente trabalho.

1.2.1 Objetivo geral

Desenvolver um aplicativo Android que permite reproduzir dados enviados por uma rádio *online*, assim como recepção de notícias no formato RSS.

1.2.2 Objetivos específicos

- Receber dados via *streaming* de uma rádio *online*;
- Obter dados do portal da rádio (programação e notícias) no formato RSS e apresentar ao usuário;
- Possibilitar o usuário enviar mensagens para à rádio.

1.3 JUSTIFICATIVA

O rádio é um dos meios de comunicação mais simples que existe atualmente , chegando seu sinal na maioria das residências (inclusive no meio rural). Ao mesmo tempo, os dispositivos móveis, como smartphones, possuem capacidade de receber este tipo de informação, em algumas situações com um receptor de rádio tradicional, em outras, recebendo os dados trafegados pela Internet.

Com a facilidade da Internet, qualquer pessoa pode criar uma rádio *online* e montar sua própria programação, ampliando horizontes que ultrapassam a

localização física, que nas rádios tradicionais se limitam a antenas e repetidores das emissoras.

Atualmente o Android é a tecnologia para smartphones mais utilizada no mundo. Um estudo realizado pela Universidade Fundação Getúlio Vargas no ano de 2015 verificou que, dos 300 milhões de dispositivos conectados à Internet no Brasil, 154 milhões são smartphones (EFE, 2015). Destes 154 milhões, cerca de 86,9% utilizam a tecnologia Android, segundo dados da empresa Kantar Wordpanel.(Landim, 2014).

Em outro estudo, da Qualcomm da Sociedade da Inovação (QuISI), em parceria com a Convergencia Research, 73% dos usuários de Internet consomem conteúdo multimídia de seus dispositivos móveis. Os dados apontaram também que o consumo multimídia é maior que outros serviços tradicionais da plataforma, como acessar bancos *online*, *e-commerce* e *games*. (iMASTERS, 2014)

Dadas as facilidades para a criação de rádios *online*, muitas empresas e instituições de ensino estão investindo neste tipo de mídia para a difusão de informação, sendo este o caso do grupo PET da UTFPR - Campus Pato Branco, que possui uma rádio *online* e um portal de notícias. Há muitos anos, os coordenadores deste projeto demandam a criação de um software para execução da rádio em um smartphone, sendo que este software também deveria permitir a leitura das notícias presentes na página do grupo. Assim, esse trabalho visa atender essa demanda por meio do desenvolvimento deste aplicativo.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho está dividido em capítulos, sendo este o primeiro, no qual se encontra as considerações iniciais do trabalho, os objetivos e a justificativa. No Capitulo 2 é apresentado o referencial teórico e os principais conceitos essenciais para a compreensão do tema proposto nesta monografia. No Capítulo 3 são apresentados os materiais e métodos utilizados para o desenvolvimento do software proposto. Já o capítulo 4 apresenta os resultados, estes formados pela análise do aplicativo, as telas e as principais partes de código desenvolvido. Por fim, o Capítulo 5 apesenta as conclusões do trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta conceitos sobre a reprodução de rádios *online* em dispositivos móveis, assim como conceitos de RSS e execução de mídias na plataforma Android.

2.1 HISTÓRIA DO RÁDIO

Desde seu surgimento, o ser humano sempre buscou meios de se expressar, para deixar sua marca na história e para demonstrar seus sentimentos, afetos religiosos, em síntese, poder registrar de alguma maneira sua história e forma de pensar e agir.

As primeiras tentativas de criar um sistema de escrita aconteceram por volta de 4000 a.C., sendo utilizadas inicialmente marcas gráficas nas paredes. Com estas marcas, o homem descobriu que era possível emitir pequenos sons para cada um daqueles objetos, formando a fonética e posteriormente a linguagem que conhecemos hoje. (PERLES, 2015)

O desenvolvimento tecnológico é fruto, normalmente, do trabalho de uma série de pesquisadores e cada descoberta abre novas possibilidades. Com a evolução da ciência, e tendo em vista as necessidades de uma comunicação que fosse interpessoal, podendo, ao mesmo tempo, atingir uma grande quantidade de pessoas, foi desenvolvido o rádio no final do século XIX.

Em 1887, o estudante Heinrch Rudolf Hertz descobre as Ondas Hertzianas, por meio de um aparelho que produzia correntes alternadas, dando espaço para que em 1896 Guglielmo Marconi, com os conhecimentos de Hertz, pudesse enviar a primeira mensagem em Código Morse de Dover (Inglaterra) à Viemeux (França). Ainda, com a evolução dessas pesquisas, foi enviado um sinal radiotelegráfico transoceânico, da Inglaterra para o Canadá em 1901. (GOMES JÚNIOR, 2015)

Porém, praticamente ao mesmo tempo, no Brasil, o Padre Brasileiro Roberto Landell de Moura, transmitiu e recebeu a primeira transmissão com voz humana em 1892, na cidade de Campinas, sendo incompreendido e até acusado de atos de bruxaria pela sociedade. Assim, entre 1892 e 1894 foram realizadas as primeiras experiências de radiofusão no Brasil, transmitindo pela primeira vez na história a palavra humana pelo espaço.

Mas somente quatro anos após Guglielmo Marconi receber sua patente, o Governo Brasileiro - em 1901 - lhe concedeu a patente 3.279, e nos anos de 1903 e 1904 nos Estados Unidos conseguiu também receber patente nos Estados Unidos de seus três inventos: o transmissor de ondas (hertzianas ou landellianas), o telefone sem fio e o telégrafo sem fio. (GOMES JÚNIOR, 2015)

Existem diversas questões de quem seria realmente o inventor do rádio, mas verifica-se que os dois tiveram invenções diferentes, Gublielmo Marconi é o iniciador da emissão-recepção eletrônica telegráfica, enquanto o padre Roberto Landell de Moura é o pioneiro da emissão-recepção fotônica-eletrônica em fonia, sendo o precursor da radiodifusão. (CUNHA; HAUSSEN, 2003)

O Rádio, como veículo de comunicação que se conhece hoje, nasceu nos EUA na década de 20 com Frank Conrad que, por passatempo, montou um transmissor e começou a transmitir notícias lidas de jornais e músicas de discos em uma garagem de Pittsburgh, Pensilvânia, EUA. Os radioamadores que surgiram na época foram se acostumando e gostando e começaram a escrever pedindo músicas e em pouco tempo se começou a vender receptores especialmente para ouvir a rádio (GOMES JÚNIOR, 2015).

2.2 RÁDIO *ONLINE* PARA DISPOSITIVOS MÓVEIS

Atualmente, há diversos aplicativos de rádios para dispositivos móveis genéricos, que podem ser utilizados para recepção de uma grande quantidade de rádios existentes na Internet, tanto AM como FM. Estes aplicativos são comumente chamados de *player*. Dentre eles, percebe-se um padrão para a apresentação de informação e a interatividade com o usuário. Em todos os softwares estudados, o *player* da rádio se localiza na parte inferior do aplicativo com um botão de *play/pause* e a barra de volume, conforme apresentado na Figura 1.

Figura 1 **- Aplicativo da rádio UCS FM**

Neste *player*, é possível também visualizar outros conteúdos de texto, como as notícias, enviar mensagens à radio, redes sociais da rádio, como também obter informações do aplicativo, como apresentado na Figura 2.

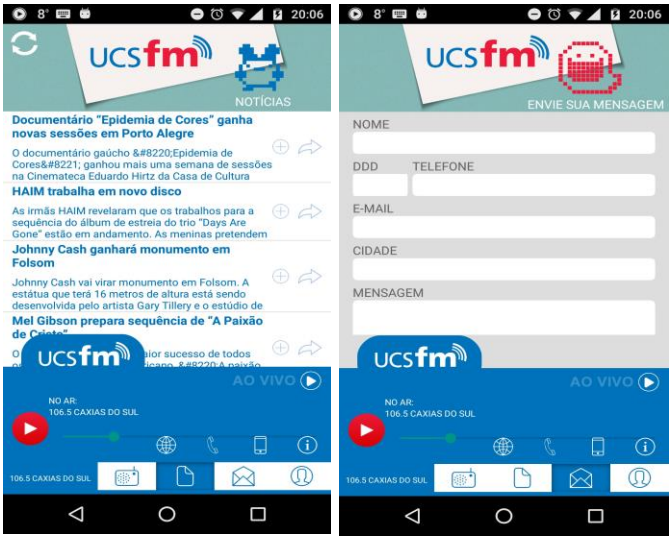

Figura 2 - Interações UCS FM

O *player* apresentado na Figura 2 é o UCS FM e pode ser obtido em https://play.google.com/store/apps/details?id=com.mobsolution.ucsfm. O *player* apresentado é específico para a Rádio UCS.

Na Figura 3, pode-se visualizar a página *web* da rádio UCS FM, podendo perceber que os mesmos dados na página estão no aplicativo.

Figura 3 - Página Web da Rádio UCS FM

Na Figura 4, pode-se observar o aplicativo da Rádio Gaúcha (disponibilizado em https://play.google.com/store/apps/details?id=com.clicrbs.gaucha), no qual é possível visualizar os Twittes realizados pela rádio, áudios gravados e vídeos, além de poder ouvir a rádio pelo *player* localizado na parte inferior do aplicativo.

Figura 4 - Aplicativo da rádio Gaúcha

2.3 RSS

A grande maioria das pessoas se interessa em sites que possui alterações periódicas de conteúdo, como sites de notícia. Para evitar que as pessoas precisem acessar periodicamente estes sites, foi criado um formato para divulgação de notícias, chamado de RSS (*Rich Site Summary* ou *Really Simple Syndication*).

O RSS é uma forma de entregar dados de conteúdo *web* que são atualizados com frequência, possibilitando que os leitores não precisem acessar diversas portais diferentes para manterem-se atualizados. Basta fazer a leitura das notícias por meio de um aplicativo que receba os RSS, que podem ser recuperados de fontes diferentes. Além disso, outra vantagem está na privacidade dos leitores, não sendo necessário se cadastrar em um boletim de *e-mail* para cada site. (GARDEN, 2004)

O RSS utilizado como padrão o envio de texto no formato XML que disponibiliza as informações básicas dos conteúdos, ordenados do mais recente ao mais antigo. Geralmente, cada item consiste de apenas um título, data de postagem, uma breve descrição do conteúdo, *link* para o conteúdo original, podendo ter também um *link* para uma imagem, que pode ser baixada e apresentada ao usuário.

A Listagem 1 apresenta um exemplo de texto RSS recuperado da rádio *online* do grupo PET, da UTFPR-Campus Pato Branco.

```
1. <? xml version = "1.0" encoding = "UTF-8"?>
2. <rss  version = "2.0" >
3.    <channel>
4.        <title>Web Rádio UTFPR</title>
5.       <link>http://www.pb.utfpr.edu.br/radio/site</link>
6.       <description>Campus Pato Branco</description>
7.       <lastBuildDate>Mon, 30 Mar 2015 22:10:41 +0000</lastBuildDate>
8.       <language>pt-BR</language>
9.        <item>
10.            <title>Materiais que disconcordam com as leis da física</title>
11.            <link>http://www.pb.utfpr.edu.br/radio/site/?p=45</link>
12.             <pubDate>Tue, 24 Mar 2015 01:39:40 +0000</pubDate>
13.            <description>Auxéticos Puxe um elástico e ele se tornará mais longo e mais fino. Faça
14.            isso  com virtualmente qualquer outro material, e o fenômeno se repetirá. Por isso, até
15.            há pouco tempo os cientistas acreditavam que...
16.            </description>
17.       </item>
18.       <item>...</item>
19.       <item>...</item>
20.    </ channel>
21. </ rss>
```

Listagem 1 – Exemplo de XML utilizado pra RSS

2.4 MEDIAPLAYER

A plataforma Android dispõe de classes para a reprodução de dados multimídia, sendo que destas, a classe MediaPlayer é a mais utilizada, seja para a execução de arquivos existentes no dispositivo móvel (música ou vídeo), como também de informações vindas da Internet por meio de *streaming*.

O controle da classe MediaPlayer é feito por meio de estados. O diagrama apresentado na Figura 5 mostra o ciclo de vida e os estados do objeto MediaPlayer por meio dos controles de reprodução suportados.

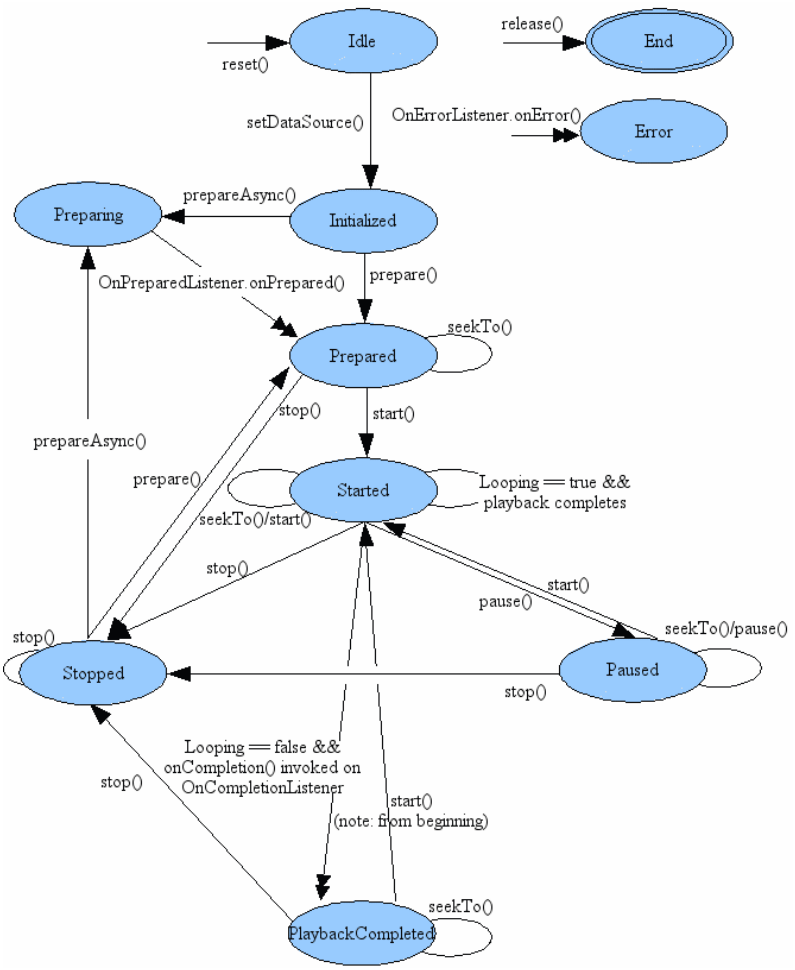

Figura 5 - Ciclo de vida e estados do objeto MediaPlayer

A partir do diagrama apresentado na Figura 5, observa-se que um objeto *MediaPlayer* tem os seguintes estados:

Quando um objeto *MediaPlayer* é criado ou após o estado *reset()* ser chamado, ele está ocioso, no estado *Idle*; e depois do método *release()* ser chamado, o estado *End* é chamado.

Recomenda-se que quando o objeto MediaPlayer não está mais sendo utilizado, deve-se chamar o método *release()* para que os recursos usados possam ser liberados imediatamente.

Em geral, uma operação do controle de reprodução pode falhar devido a várias razões, tais como áudio sem suporte/formato de vídeo, áudio mal intercalado/vídeo, resolução muito alta, *streaming* de *timeout*, e similares. Assim, um relatório de erros e recuperação é uma preocupação importante a se ter. Às vezes,

devido a erros de programação, chamando uma operação de controle de reprodução em um estado inválido também podem acontecer. Sob todas estas condições de erro, o objeto *MediaPlayer* invoca o método *OnErrorListener.onError()* caso tenha sido criado através do *setOnErrorListener (OnErrorListener).*

É importante se destacar que quando ocorre um erro, o objeto *MediaPlayer* entra no estado *Error* (exceto como mencionado acima), mesmo se um *Listener* de erro não foi registrado pelo aplicativo.

A fim de reutilizar um objeto *MediaPlayer* que está no estado *Error* e recuperar do erro, o *reset()* pode ser chamado para restaurar o objeto ao seu estado *Idle*.

É uma boa prática de programação ter em seu aplicativo um *OnErrorListener* para receber notificações de erro do motor do *player*.

Chamando *setDataSource(),* transfere um objeto *MediaPlayer* no estado *Idle* estado para o estado *Initialized*. Um *IllegalStateException* é lançado se *setDataSource()* é chamado em qualquer outro estado.

É uma boa prática de programação sempre olhar para *IllegalArgumentException* e *IOException* que pode ser invocado a partir das método setDataSource.

Um objeto *MediaPlayer* deve primeiro entrar no estado *Prepared* antes da reprodução poder ser iniciada.

Há duas maneiras (síncrono ou assíncrono) que o estado *Prepared* pode ser atingido: com uma chamada para *prepare()* (síncrono) que transfere o objeto para o estado *Prepared* uma vez que a chamada do método retorna, ou uma chamada para o *prepareAsync()* (assíncrono), que primeiro transfere o objeto para o estado *Prepared* após o retorno da chamada, enquanto objeto MediaPlayer continua trabalhando sobre o resto da preparação até que seja completada. Quando a preparação é concluída ou quando a chamada do *prepare()* retorna, objeto MediaPlayer em seguida, chama um método de retorno, *onPrepared()* da interface *OnPreparedListener*, se um *OnPreparedListener* foi implementado com antecedência através do *setOnPreparedListener(OnPreparedListener).*

Um *IllegalStateException* é lançado se um método *prepare()* ou *prepareAsync()* é chamado em qualquer outro estado.

Enquanto no estado *Prepared* propriedades como volume, *screenOnWhilePlaying, looping* podem ser modificadas invocando os métodos set correspondentes.

Para iniciar a reprodução o *start()* deve ser chamado. Depois do método *start()* ser retornado com êxito, o objeto MediaPlayer está no estado *Initialized*. *IsPlaying()* pode ser chamado para testar se o objeto MediaPlayer está no estado *Initialized*.

Enquanto no estado *Initialized*, o motor do player chama o *OnBufferingUpdateListener.onBufferingUpdate()*, um método de *callback* para caso o *OnBufferingUpdateListener* tenha sido implementado anteriormente através do *setOnBufferingUpdateListener(OnBufferingUpdateListener)*. Este *callback* permite que os aplicativos possam acompanhar o estado de buffer durante a transmissão de áudio/vídeo.

Chamar o método *start()* não tem efeito sobre um objeto MediaPlayer que já está no estado *Initialized*.

A reprodução pode ser pausada, parada, e a posição de reprodução atual pode ser ajustada. A reprodução pode ser interrompida através do método *pause()*. Quando o método *pause()* retorna, o objeto MediaPlayer entra no estado *Paused*. A transição do estado *Initialized* para o estado *Paused* e vice-versa acontece de forma assíncrona no objeto. Pode levar algum tempo antes que o estado ser atualizado em chamadas para *isPlaying(),* e pode levar alguns segundos no caso de *streaming*.

Chamando o método *start()* para retomar a reprodução de um objeto MediaPlayer pausado, retoma a posição de reprodução a mesma onde foi pausada. Quando a chamada para o método *start()* retorna, o objeto *MediaPlayer* parado vai voltar para estado *Initialized*.

Chamar o método *pause()* não tem efeito sobre um objeto *MediaPlayer* que já está em estado *Paused*.

Chamando o método *stop()* interrompe a reprodução e faz com que um *MediaPlayer* no estado *Initialized*, *Paused*, *Prepared* ou *PlaybackCompleted* entrar no estado *Stopped*.

Uma vez no estado *Stopped*, a reprodução não pode ser iniciada até que o método *prepare()* ou *prepareAsync()* é chamada para definir o objeto *MediaPlayer* para o estado *Prepared* novamente.

Chamando o método *stop()* não tem efeito sobre um objeto *MediaPlayer* que já está no estado *Stopped*.

2.5 XMLPullParser

Para a leitura de mensagens no formato RSS, a plataforma Android dispõe de classes para a análise de arquivos XML, sendo uma destas, a classe XMLPullParser. Essa classe utiliza dois métodos principais: *next()* e *nextText()*. Enquanto o método *next()* fornece acesso a eventos de análise de alto nível, *nextText()* permite o acesso a eventos de nível inferior.

O evento atual do analisador pode ser determinado chamando o método *getEventType().* Inicialmente, o analisador está no estado START_DOCUMENT.

O método *next()* avança o analisador para o próximo evento, o valor int retornado determina o estado do analisador.

Os seguintes tipos de evento são vistos por *next()*:

- START_DOCUMENT: O evento inicial, quando o analisador ainda não começou a ler o XML.
- START_TAG: Quando o XML começa a ser lido;
- TEXT: O conteúdo a ser lido;
- END_TAG: Quando o analisador está na tag final;
- END_DOCUMENT: Quando o documento terminou e não há mais eventos disponíveis.

3 MATERIAIS E MÉTODOS

Este capítulo apresenta os materiais utilizados para o desenvolvimento do aplicativo desenvolvido como resultado deste trabalho, que se resume aos softwares utilizados para a análise e codificação, assim como o modelo do smartphone utilizado para os testes. Nos métodos é apresentada a sequência de passos utilizados para atingir o resultado final.

3.1 MATERIAIS

Os materiais e ferramentas utilizadas para desenvolver o aplicativo, resultado deste trabalho, são apresentados no Quadro 1.

Nome	**URL**	**Versão**	**Utilização**
Astah	http://astah.net/download	Professional 7.0.0/846701	Ferramenta de modelagem UML
ADT Bundle	http://moodleesvnemesio.info/version7D/node/20	20130522	Pacote de Ferramentas de Desenvolvimento Android
Android SDK	https://developer.android.com/sdk/	24.3.4	Pacote necessário para desenvolver aplicativos para plataforma Android
Eclipse	https://eclipse.org.	Luna Release (4.4.0)	Ambiente de programação
Motorola Razr D3	http://www.tudocelular.com/Motorola/fichas-tecnicas/n2417/Motorola-RAZR-D3.html	Android 4.2.2 Jelly Bean	Realização de Testes
Motorola Moto X Play	http://www.motorola.com.br/products/moto-x-play	Android 6.0.1 Marshmallow	Realização de Testes

Quadro 1 - Ferramentas Utilizadas

Fonte: Autoria própria

A seguir, algumas informações sobre cada uma das ferramentas utilizadas:

- **Astah**: Software com interface amigável para criação de UML (*Unified Modeling Language*), dando suporte a todo o desenvolvimento, deixando-o mais fácil de compreender;
- **ADT Bundle -** O ADT é uma IDE que possui um ambiente completo para desenvolvimento, sendo nele incluso o SDK do Android, o Eclipse como IDE de ambiente de codificação e o ADT como *plugin*;
- **Android SDK**: Uma ferramenta disponibilizada pela Google para desenvolver, debugar e testar aplicativos da plataforma Android. Essa ferramenta não possui um ambiente de trabalho, sendo necessário a integração com ambientes como o Eclipse ou qualquer outro disponível na comunidade;
- **Eclipse**: Um Ambiente de Desenvolvimento Integrado (IDE) de código aberto que contempla várias linguagens de programação e oferece um completo ambiente para a programação na tecnologia selecionada;
- **Motorola Razr D3:** Smartphone com a versão 4.2.2 do Android que possui uma tela de 4 polegadas com resolução de 800x480 pixels, processador Dual-Core de 1.2GHz e 1GB de memória RAM**;**
- **Motorola Moto X Play:** Smartphone Android com a versão 6.0.1 do Android que possui uma tela de 5.5 polegadas com resolução de 1080 x 1920 pixels, processador Quad-core 1.7 GHz Cortex-A53 + Quad-core 1.0 GHz Cortex-A53 e 2GB de memória RAM.

3.2 MÉTODOS

O desenvolvimento do aplicativo de rádio web foi dividido em etapas. Essas etapas são:

a) **Requisitos:** A definição dos requisitos para verificar as características necessárias para o *player*, fruto deste trabalho.
b) **Análise:** Por meio da criação de diagramas de caso de uso e de atividade foram definidas as funcionalidades do aplicativo. Dado que se trata de um

aplicativo com baixa interatividade com o usuário, com apenas estes diagramas foi possível definir suas funcionalidades.

c) **Desenvolvimento:** O Desenvolvimento foi realizado com base nos materiais apresentados na seção 3.1.

d) **Testes:** Os testes foram realizados para o funcionamento correto do sistema. Foram utilizados os seguintes dispositivos: Motorola Razr D3 e Motorola Moto X Play.

4 RESULTADOS

Este capítulo apresenta o aplicativo desenvolvido, assim como o processo de análise, telas e codificação.

4.1 REQUISITOS

Os requisitos foram levantados, visando possibilitar ao usuário receber informações da rádio do grupo PET por meio de um dispositivo móvel e são apresentados nos Quadros 2 e 3.

Identificação	**Nome**	**Descrição**
RF001	escutar a rádio	retornar por meio de *streaming* o áudio da rádio.
RF002	enviar mensagem à radio	permitir ao usuário a possibilidade de enviar mensagens à rádio.
RF003	visualizar notícias do portal da rádio	permitir a visualização de notícias postadas no portal da rádio.

Quadro 2 - Requisitos funcionais definidos para o sistema

Identificação	**Nome**	**Descrição**
RNF001	versões SO Android	requisitos de versão do sistema operacional Android: mínima: 4.2.2.
RNF002	Disponibilidade	a disponibilidade de ouvir a rádio, ver as notícias e enviar mensagem à radio só será possível caso o dispositivo tenha acesso a Internet.

Quadro 3 - Requisitos não funcionais definidos para o sistema

4.2 MODELAGEM DO SOFTWARE

Esta seção apresenta a análise do aplicativo, abordando os diagramas utilizados, assim como suas principais funcionalidades. Dentro destes diagramas encontra-se o diagrama de caso de uso, apresentado na Figura 6, que mostra as funcionalidades que o usuário terá com o aplicativo.

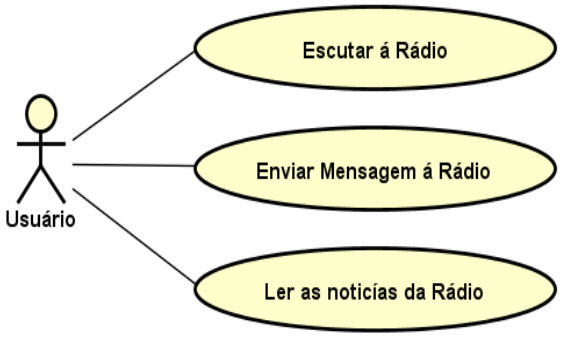

Figura 6 - Diagrama de Caso de Uso
Fonte: Autoria Própria

O sistema só terá um perfil de usuário, por isso não será preciso *login* ou autenticação. Logo ao iniciar o sistema será possível iniciar, pausar ou parar a programação, assim como alterar o volume, ler as notícias e enviar mensagem para a rádio.

Após a definição dos requisitos e o desenvolvimento do caso de uso, o próximo passo foi desenvolver o diagrama de atividade (Figura 7), que visa auxiliar o desenvolvimento de fluxo entre telas.

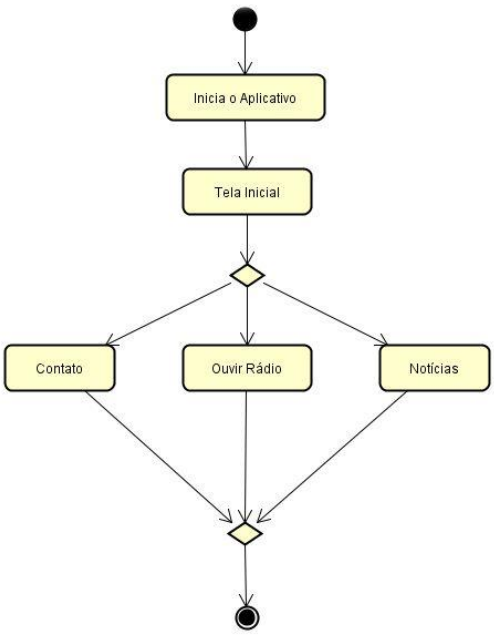

Figura 7 - Diagrama de Atividade
Fonte: Autoria Própria

Não houve a necessidade de persistência de dados no aplicativo cliente, assim, não foi necessário modelar um Diagrama de Entidade e Relacionamento (DER).

4.3 MODELAGEM DO SISTEMA

O passo seguinte foi a estruturação do projeto, identificando o número de classes. Para o desenvolvimento, foi utilizado o projeto apresentado na Figura 8, o qual possui os códigos fontes do aplicativo.

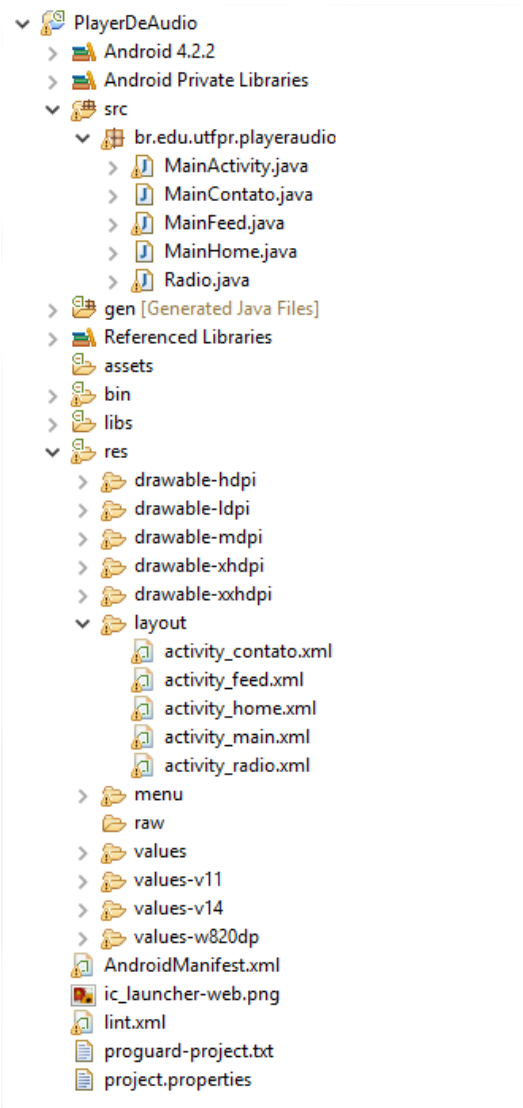

Figura 8 - Estrutura do Projeto

4.4 DESENVOLVIMENTO DO SISTEMA

Para o desenvolvimento do *player* foi utilizada a versão 6.0 do Android, chamado de “Marshmallow”, e como requisito mínimo a versão 4.2.2 , chamada de “Jelly Bean”, para ser possível utilizar o aplicativo também em versões mais antigas do sistema operacional. Assim, garante-se que o aplicativo rodará em aproximadamente 89% dos *devices* Android existentes no mercado. (ANDROID, 2016).

Na Figura 9 é possível visualizar a tela que o usuário visualizará ao iniciar o aplicativo:

Figura 9 - Tela Inicial do Aplicativo

De acordo com a Figura 9, é possível verificar o *player* da rádio na parte inferior do aplicativo, tendo os controles *Play*, *Pause* e *Stop*, uma barra para o ajuste do volume, e um campo de texto informando os estados do *player* que pode ser Reproduzindo, Pausado e Parado.

Na primeira execução, é necessário fazer um carregamento prévio do áudio (*Buffer*), enquanto neste estado é substituído o texto informativo por uma barra de carregamento, conforme apresentada na Figura 10.

Figura 10 - Carregando áudio

Na parte superior do aplicativo é possível visualizar 3 botões (Início, Contato e Notícias). O botão “Início” apresenta a tela inicial, conforme apresentada na Figura 10. Comumente, esse botão é utilizado quando o usuário se encontra em outra tela, como a de notícias, por exemplo, e deseja voltar para a tela da rádio. Ao clicar no botão “Contato” é exibida a tela que permite, por meio do aplicativo de *e-mail* padrão instalado no dispositivo, enviar um *e-mail* a rádio (Figura 11).

Figura 11 - Tela de Contato do aplicativo

Esta tela permite enviar um *e-mail* para o administrador da rádio, solicitando alguma música ou interagir com a rádio.

Ao clicar no botão “Notícias”, o usuário tem acesso as notícias postadas no site (Figura 12). Esta informação é recebida no formato RSS, extraídas as informações mais relevantes para, posteriormente, serem apresentadas ao usuário.

Figura 12 - Tela de Notícias do aplicativo

Ao clicar no *link* de uma notícia, o sistema permite ao navegador padrão instalado no dispositivo exibir o texto completo da notícia., conforme ilustra a Figura 13.

Figura 13 - Acessando notícias da rádio

4.5 CODIFICAÇÃO DAS FUNCIONALIDADES

A seguir os principais trechos de códigos desenvolvidos em linguagem Java para o desenvolvimento do presente trabalho.

4.5.1 Execução da rádio *online*

Para que a aplicação utilize os recursos de Internet é necessário disponibilizar permissões no arquivo AndroidManifest.xml, conforme a Listagem 2.

```
1. <uses-permission android:name="android.permission.INTERNET" />
```

Listagem 2- AndroidManifest.xml – Permissão para o uso de Internet.

Para a reprodução do áudio da rádio foi desenvolvida a classe Radio.java, sendo detalhada na sequência. A Listagem 3 apresenta a declaração dos objetos utilizadas por esta classe.

```
1. public class Radio extends Fragment implements
OnPreparedListener,OnBufferingUpdateListener,OnErrorListener,
OnSeekBarChangeListener {
2.
3.    private MediaPlayer player;
4.    private SeekBar sbVolume;
5.    private TextView tvInfo;
6.    private ProgressBar pbBuffer;
7.    private AudioManager audioManager;
8.    private boolean isPlaying;
9.    private int maxVolume;
10.   private int curVolume;
11.   private ImageButton btPlay;
12.   private ImageButton btPause;
13.   private ImageButton btStop;
```

Listagem 3 – Radio.java – Activity para a reprodução da rádio.

A Listagem 3 apresenta a declaração da classe “Radio” que é uma extensão da classe “*Fragment*” implementada pelos *Listeners* utilizados pelo objeto MediaPlayer.

Dos objetos declarados, destaca-se o MediaPlayer que fará todo o controle de reprodução da rádio, o *Seekbar* que controlará o volume, um *TextView* que informará o estado de reprodução, o *ProgressBar* que será exibido quando o áudio da rádio estiver carregando, o *AudioManager* que será o intermediário pelo controle do volume entre o aplicativo e o sistema Android e três *ImageButtons* que servirão como gatilho para ações do objeto MediaPlayer. Quando a classe Radio é criada, são executadas algumas funções para obter o controle do áudio do Android e atribui-lo a barra de volume listadas na Listagem 4.

```
1.    this.audioManager =
((AudioManager)getActivity().getSystemService(Context.AUDIO_SERVICE));
2.        int maxVolume =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
3.        int curVolume =
audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
4.        sbVolume.setMax(maxVolume);
5.        sbVolume.setProgress(curVolume);
6.        sbVolume.setOnSeekBarChangeListener(this);
7.
8.        if(savedInstanceState != null){
9.          isPlaying = savedInstanceState.getBoolean("isPlaying");
10.
11.         if(isPlaying){
12.               try {
13.                           playMusic(null);
14.                     } catch (IOException e) {
15.                     } catch (ParserConfigurationException e)
{e.printStackTrace();
16.                     } catch (SAXException e) {e.printStackTrace();}
17.         }
18.
19.       }
20.
21.         return(view);
22.   }
23.
24.   @Override
25.   public void onSaveInstanceState (Bundle output){
26.         super.onSaveInstanceState(output);
27.         output.putBoolean("isPlaying", isPlaying);
28.   }
```

Listagem 4 – Iniciando a classe Radio

O objeto *AudioManager* é setado para utilizar o serviço de áudio do Android (linha 1), nas linhas 2 e 3 são atribuídos os valores de volume máximo e o volume atual do sistema, posteriormente esses valores são setados para a barra na qual o usuário poderá interagir com o volume.

Na linha 8 é verificado se uma sessão já existe, caso for verdadeiro é atribuída a variável *isPlaying* salva no método *onSaveInstanceState(),* e caso for verdadeiro é executado o método *playMusic()*.

Na linha 25 o método *onSaveInstanceState(),* para salvar a variável *isPlaying* caso o aplicativo seja fechado, para retornar a reprodução quando reaberto.

```
1.    @Override
2.    public void onDestroy(){
3.          super.onDestroy();
4.          if(player != null){
5.                player.stop();
6.                player.release();
7.                player = null;
8.          }
9.    }
10.
11.   public void playMusic(View view) throws IOException,
ParserConfigurationException, SAXException{
12.
13.         if(player == null){
14.               try{
15.                     player = new MediaPlayer();
16.               player.setAudioStreamType(AudioManager.STREAM_MUSIC);
17.               player.setDataSource("http://167.114.209.213:8405");
18.                     player.prepareAsync();
19.                     } catch(Exception e ) { e.printStackTrace();}
20.
21.                     tvInfo.setVisibility(view.GONE);
22.               pbBuffer.setVisibility(view.VISIBLE);
23.
24.                     player.setOnBufferingUpdateListener(this);
25.                     player.setOnErrorListener(this);
26.                     player.setOnPreparedListener((OnPreparedListener)
this);
27.
28.         }else{
29.               player.start();
30.               tvInfo.setVisibility(view.VISIBLE);
31.               tvInfo.setText("Reproduzindo");
32.               isPlaying = true;
33.         }
34.   }
```

Listagem 5 – onDestroy e playMusic

Na Listagem 5, na linha 2 o método *onDestroy()* é sobreposto para caso o objeto MediaPlayer já existir ser parado e liberado da memória.

No método *playMusic()* (linha 11) é verificado se o objeto MediaPlayer já existe. Caso não existir é instanciado um novo MediaPlayer, informado o tipo de áudio, que neste caso será *Streaming*, a fonte de áudio para o objeto, e executada uma reprodução Assíncrona. O *TextView* tvInfo é ocultado (linha 21) e em seu lugar é mostrada a barra de progresso indicando o carregamento do áudio.

Caso o objeto MediaPlayer já exista (linha 28) é iniciada a reprodução, setado o *TexView* tvInfo para que seja visível e exiba a informação ao usuário que está reproduzindo, sendo também setada a variável *isPlaying* para *true*.

```
1.    public void pauseMusic(){
2.          isPlaying = false;
3.          if(player != null){
4.                      player.pause();
5.                      tvInfo.setText("Pausado");
6.                      pbBuffer.setVisibility(View.GONE);
7.                }
8.    }
9.
10.   public void stopMusic(){
11.         isPlaying = false;
12.         if(player != null){
13.                     player.stop();
14.                     player.release();
15.                     player = null;
16.                     tvInfo.setText("Parado");
17.                     pbBuffer.setVisibility(View.GONE);
18.               }
19.   }
```

Listagem 6 – pauseMusic e stopMusic

Na listagem 6, no método *pauseMusic()* (linha 1) é setada a variável *isPlaying* para *false*. Caso o objeto MediaPlayer exista é pausa da reprodução, substituído o texto para Pausado e ocultado a barra de progresso.

No método *stopMusic()* (linha 10) é setada a variável *isPlaying* para *false*. Caso o objeto MediaPlayer exista é parada da reprodução, liberado o objeto MediaPlayer, substituído o texto para Parado e ocultado a barra de progresso.

No método *playerMusic()* (Listagem 5), foi realizada uma reprodução Assíncrona, que carrega os recursos fora da *Thread* Principal, chamando logo após completada o método *onPrepared()* (Listagem 7), onde é atribuído verdadeiro a variável *isPlaying*, iniciada a reprodução no objeto MediaPlayer. É desocultado o *TextView* tvInfo e substituído o texto para Reproduzindo, sendo após ocultada a barra de carregamento.

```
1.    @Override
2.    public void onPrepared(MediaPlayer mp) {
3.          Log.i("Script", "onPrepared()");
4.          isPlaying = true;
5.
6.          mp.start();
7.          tvInfo.setVisibility(View.VISIBLE);
8.          tvInfo.setText("Reproduzindo");
9.          pbBuffer.setVisibility(View.GONE);
10.   }
```

Listagem 7 – onPrepared

4.5.2 Leitor RSS

Para a obtenção dos dados RSS foi criada a classe MainFeed.java demonstrada na Listagem 8, e especificada em seguida.

```
1. try{
2.                 XmlPullParserFactory factory =
XmlPullParserFactory.newInstance();
3.                 factory.setNamespaceAware(false);
4.                 XmlPullParser xpp = factory.newPullParser();
5.
6.                 GetData data = new GetData();
7.                 xml = data.execute().get();
8.                 xpp.setInput(new StringReader(xml));
9.
10.                boolean insideItem = false;
11.
12.                int eventType = xpp.getEventType();
13.                while(eventType != XmlPullParser.END_DOCUMENT){
14.                      if(eventType == XmlPullParser.START_TAG){
15.                      if(xpp.getName().equalsIgnoreCase("item")){
16.                                  insideItem = true;
17.                      } else if
(xpp.getName().equalsIgnoreCase("title")){
18.                                  if (insideItem){
19.                                        titulo.add(xpp.nextText());
20.                                  }
21.                            }else
if(xpp.getName().equalsIgnoreCase("link")){
22.                                  if (insideItem){
23.                                        link.add(xpp.nextText());
24.                                  }
25.                            }
26.
27.                      }else if(eventType == XmlPullParser.END_TAG &&
xpp.getName().equalsIgnoreCase("item")){
28.                            insideItem = false;
29.                      }
30.
31.                      eventType = xpp.next();
32.                }
```

Listagem 8 – MainFeed. java – Activity para tratamento das notícias

Nas linhas 2 a 4 é criado o XmlPullParser, o analisador para extrair as informações necessárias do XML.

Na linha 6 é instanciada a classe GetData que retornará o arquivo XML com os dados RSS do site, sendo atribuído seu resultado a variável xml e após sendo atribuída ao analisador (linha 8).

Na linha 10 é criada uma variável *boolean* chamara *insideItem*, utilizada para identificar se está dentro de um item referente a uma notícia.

Na linha 12 é criada outra variável do tipo inteiro para verificar o tipo do evento do analisador no XML.

Na Linha 13 é criado um laço de repetição while para que não termine até encontrar o evento END_DOCUMENT

Na linha 14 é criado um if para caso o evento seja START_TAG é lido seu conteúdo. Caso o nome da tag seja item é atribuído verdade a variável insideItem, caso a tag tenha o nome *title* e a variável insideItem seja verdadeira é atribuído o seu texto ao ArrayList título, para o link da notícia é repetido o mesmo processo.

Na linha 27 é verificado se o evento é END_TAG e se a tag possui o nome item, informando que a tag acabou. Caso seja verdadeiro é atribuído falso para a variável insideItem. Sendo logo após chamado o próximo evento do analisador.

```
1. public String downloadData(){
2.          InputStream in = null;
3.          String rssFeed = "";
4.         try {
5.          URL url = new
URL("http://www.pb.utfpr.edu.br/radio/site/?feed=rss2");
6.              HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
7.              in = conn.getInputStream();
8.              ByteArrayOutputStream out = new ByteArrayOutputStream();
9.              byte[] buffer = new byte[1024];
10.              for (int count; (count = in.read(buffer)) != -1; ) {
11.                  out.write(buffer, 0, count);
12.              }
13.              byte[] response = out.toByteArray();
14.              rssFeed = new String(response, "UTF-8");
15.              return rssFeed;
16.          } catch (IOException e) {
17.              e.printStackTrace();
18.          } finally {
19.              if (in != null) {
20.                  try {
21.                      in.close();
22.                  } catch (IOException e) {
23.                      e.printStackTrace();
24.                  }
25.              }
26.          }
27.          return rssFeed;
28.    }
```

Listagem 9 - downloadData

Na listagem 9 é possível visualizar a classe *downloadData()*, que será responsável por retornar a página XML com as notícias.

Dentro de um *try catch* é informada a url do XML que será lido, é aberta uma conexão HTTP, feita a leitura dos bytes e sendo atribuída a variável rssFeed utilizando a codificação UTF-8.

```
1.    @Override
2.    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
3.          Uri uri = Uri.parse((String) link.get(position));
4.       Intent intent = new Intent(Intent.ACTION_VIEW, uri);
5.       startActivity(intent);
6.
7.    }
```

Listagem 10 – onItemClick

O método *onItemClick()* demonstrado na Listagem 10 executará uma ação ao ser clicada em uma notícia no *ListView*. É criada a variável Uri que recebe a uri da notícia percorrendo o *ArrayList* link. Será instanciada um intenção para visualizar a notícia através do navegador do dispositivo (linha 4), e logo após chamada essa intenção (linha 5).

4.5.3 Contato

Para o envio de *e-mail* de contato, foi desenvolvida a classe MainContato.java, detalhada na sequência. A Listagem 11 apresenta a declaração dos objetos utilizados pela classe.

```
1. public class MainContato extends Fragment {
2.
3.    private EditText etNome;
4.    private EditText etMensagem;
5.    private Button btEnviar;
6.    private Button btLimpar;
```

Listagem 11 – MainContato. java – Activity para contato da rádio

É declarada a classe "MainContato" que é uma extensão da classe "*Fragment*". São declarados dois *EditText* que recuperarão o Nome e a Mensagem informados pelo usuário e dois botões, responsáveis por enviar a mensagem e limpar os campos de nome e mensagem.

Caso o usuário clique no botão de enviar, será aberta uma intenção de envio, informando que o tipo de intenção é um e-mail. O assunto do e-mail será o nome informado e o texto será o respectivo campo. Logo após será solicitado ao usuário escolher o cliente de e-mail previamente instalado no dispositivo e é feita a limpeza dos campos pelo método *clearFields()*, como pode ser visto na Listagem 12 abaixo:

```
1. private void sendEmail(){
2.          Intent i = new Intent(Intent.ACTION_SEND);
3.          i.setType("message/rfc822");
4.          i.putExtra(Intent.EXTRA_EMAIL, new
String[]{"petagroutfpr@gmail.com"} );
5.          i.putExtra(Intent.EXTRA_SUBJECT,
etNome.getText().toString());
6.          i.putExtra(Intent.EXTRA_TEXT, etMensagem.getText());
7.          startActivity(Intent.createChooser(i, "Escolha um cliente de
E-mail: "));
8.          clearFields();
9.    }
10.
11.   private void clearFields(){
12.         etNome.setText("");
13.         etMensagem.setText("");
14.   }
```

Listagem 12 – sendEmail

4.6 TESTES

Os testes foram realizados em dispositivos reais, com Android 4.2.2 e 6.0.1, a execução funcionou sem travamentos. O buffer tanto com a rede Wifi como a rede da operadora não demonstrou atrasos, sendo por volta de 2 segundos para iniciar a transmissão do áudio.

Foi disponibilizado o aplicativo para alguns voluntários testarem o aplicativo sem manuais ou orientações, as telas ficaram intuitivas possibilitando o acesso sem dificuldades.

5 CONCLUSÂO

O presente trabalho teve como objetivo realizar um estudo sobre a reprodução de *streaming* na plataforma Android, como também aplica-la em um *player* para rádio *online*.

Durante o desenvolvimento, foi realizado um estudo aplicado ao sistema operacional Android.

O desenvolvimento ocorreu com a utilização da combinação de vários recursos, sendo o Sistema Operacional Android, ADT Bundle, IDE Eclipse e Android SDK. O sistema foi desenvolvido conforme os requisitos levantados.

Foi possível cumprir com o objetivo proposto, pois é possível ter acesso à Radio UTFPR, como também verificar suas notícias.

Durante o desenvolvimento foi encontrada uma dificuldade, o formato pela qual a rádio foi disponibilizada inicialmente ainda não é suportado para a plataforma Android (AAC+ HE v2), sendo apenas possível utilizá-lo por meio de um *codec* disponibilizado pela comunidade de programadores Android, mas não estável suficiente para ser aplicado ao trabalho, por este motivo então foi solicitado outro formato aos responsáveis da rádio.

Para trabalhos futuros, o aplicativo poderá ser melhorado implementando a possibilidade de o usuário compartilhar o link das notícias com outros aplicativos e com a rede social, e permitir que o aplicativo execute em background, o que permitiria que o usuário continuasse trabalhando em outros aplicativos enquanto escutasse a rádio *online*.

REFERÊNCIAS

ANDROID, Developer DASHBOARD. Disponível em: <https://developer.android.com/about/dashboards/index.html>. Acesso em 10 de junho de 2016.

CUNHA, Mágda Rodrigues da; HAUSSEN, Doris Fagundes. **Rádio brasileiro: episódios e personagens**. 29. ed. Porto Alegre: Edipucrs, 2003. 291 p. Disponível em: <https://books.google.com.br/books?id=EMxReJ BVr4C&lpg=PA182&dq=ALBUQUERQUE%2C%20Otto.%20En%20el%20aire%3A%20la%20luz%20que%20habla.%20Porto%20Alegre%3A%20FEPLAM%2C%201993&hl=pt-BR>. Acesso em 10 junho de 2016.

EFE, Agência. **Número de smartphones supera o de computadores no Brasil**; Disponível em: <http://info.abril.com.br/noticias/mercado/2015/04/numero-de-smartphones-supera-o-de-computadores-no-brasil.shtml>. Acesso em 02 setembro de 2015.

GARDEN, Software. **What is RSS?** 2004. Disponível em: <http://rss.softwaregarden.com/aboutrss.html>. Acesso em 08 junho de 2016.

GOMES JÚNIOR, José. **A publicidade no Rádio: origem e evolução**. Disponível em: <http://www.portcom.intercom.org.br/pdfs/40c31f36d4d023b0726c48094dd32b21.pdf>. Acesso em 02 setembro de 2015.

iMASTERS, Redação. **Pesquisa revela que 73% dos usuários de Internet consomem conteúdo multimídia via mobile**. Disponível em: <http://imasters.com.br/noticia/pesquisa-revela-que-73-dos-usuarios-de-internet-consomem-conteudo-multimidia-via-mobile/>. Acesso em 12 outubro de 2015.

LANDIM, Wikerson. **Smartphones com Android dominam 86,9% do mercado brasileiro**. Disponível em: <http://www.tecmundo.com.br/celular/59589-smartphones-android-dominam-86-9-mercado-brasileiro.htm>. Acesso em 02 setembro de 2015.

PERLES, João Batista. **Comunicação: conceitos, fundamentos e história**. Disponível em: <http://www.bocc.ubi.pt/pag/perles-joao-comunicacao-conceitos-fundamentos-historia.pdf>. Acesso em 02 setembro de 2015.

TUDO RÁDIO: Em Números. Disponível em: <http://tudoradio.com/conteudo/ver/27-O-Radio>. Acesso em 02 setembro de 2015