**PAULO CÉSAR BÜTTENBENDER**

# PROVA DE CONCEITO DE INTERATIVIDADE PARA TV DIGITAL VIA WEB EM DISPOSITIVOS MÓVEIS

**Monografia apresentada ao Curso de Análise e desenvolvimento de software, Centro de ciências exatas e tecnológicas, Universidade do vale do rio dos sinos, como requisito parcial para a obtenção do título de Tecnólogo em Análise e desenvolvimento de software.**

**Orientador: Prof. Dr. Sérgio Crespo Coelho da Silva Pinto**

**SÃO LEOPOLDO**

**2011**

## FICHA CATALOGRÁFICA

Büttenbender, Paulo César

Prova de conceito de interatividade para TV digital via web para dispositivos móveis a partir da especificação e desenvolvimento de aplicação de servidor e cliente móvel – São Leopoldo, 2011.

Nº de páginas: 77

Área de concentração: Análise e desenvolvimento de software.

Orientador: Prof. Dr. Sérgio Crespo Coelho da Silva Pinto.

Aos meus pais, Anaide e Antonio, a eles todos os créditos...

Dedico

## AGRADECIMENTOS

Ao Prof. Dr. Sérgio Crespo Coelho da Silva Pinto pela dedicação nas orientações neste período de aprendizado.

Aos meus amigos pela compreensão e apoio neste período de longa dedicação.

A todos aqueles que contribuíram, direta ou indiretamente, na concepção e na transformação desta simples ideia em realidade.

“Great things are done by a series of small things brought together.”

Vincent Van Gogh.

**SUMÁRIO**

## LISTA DE ILUSTRAÇÕES

## RESUMO

PROVA DE CONCEITO DE INTERATIVIDADE PARA TV DIGITAL VIA WEB EM DISPOSITIVOS MÓVEIS A PARTIR DA ESPECIFICAÇÃO E DESENVOLVIMENTO DE APLICAÇÃO DE SERVIDOR E CLIENTE MÓVEL

**Objetivo:** Criar uma prova de conceito de uma arquitetura flexível que permita a exibição de TV Digital em dispositivos móveis e o desenvolvimento de aplicativos interativos que possam ser disponibilizados para diversas plataformas. **Material e Método:** Criação da especificação, arquitetura, projeto e desenvolvimento das interfaces de comunicação entre aplicativo cliente e servidor, e dos softwares necessários para provar a viabilidade em dispositivos Android. **Resultados:** A arquitetura se mostrou flexível e viável para implementação em diversos dispositivos, foi identificado apenas um problema de performance no dispositivo usado para teste, mas o mesmo foi considerado irrelevante uma vez que a versão de dispositivo utilizada já é considerada ultrapassada. **Conclusão**: Com a rápida evolução dos smartphones é possível transformar a prova de conceito apresentada neste trabalho em uma realidade, presente nos principais sistemas operacionais para dispositivos móveis.



**Descritores:** TV Digital, Sistema operacional Android, Interatividade, Mobilidade

## ABSTRACT

DIGITAL TV THROUGH WEB PROOF OF CONCEPT OF STREAMING AND INTERACTIVITY FOR MOBILE DEVICES BY SPECIFICATING AND DEVELOPING MOBILE CLIENT AND SERVER APPLICATIION

**Purpose:** Create a proof of concept of an flexible architecture that allows a Digital TV stream to mobile devices and the development of interactivity applications supporting a wide range of platforms **Material and Method:** Specification, architecture, design and development of communication interfaces between client application and server, and of needed software to determine technical viability at Android devices. **Results:** The architecture shown to be flexible enough to be implemented in many devices, a performance issue was identified in testing devce, altough irrelevante due to the fact that it was a device from past generation. **Conclusion**: The current evolution growth of mobile devices is in fast pace, and due to that it is possible to transfor this proof of concept shown in this paper in reality.



**Key words:** Digital TV, Android operating system, Interactivity, Mobility

# 1 INTRODUÇÃO

# 1 INTRODUÇÃO

Segundo o instituto Gartner, no primeiro trimestre de 2010 a venda de smartphones (dispositivos móveis multifuncionais e inteligentes) para usuários finais (desconsiderando a venda para empresa) alcançou a marca de 1,2 milhão de unidades, um aumento de 170% contra o mesmo período do ano anterior. Em nível mundo, foi identificado um aumento de 50,5% nas vendas sobre o mesmo período do ano anterior, contabilizando mais de 60 milhões de aparelhos no segundo trimestre de 2010. (GOASDUFF e PETTEY, 2010)

Esse considerável aumento nas vendas de smartphones ilustra o fato de que cada vez mais os dispositivos móveis tendem a abandonar a ideia de apenas conectar as pessoas por voz, e se tornarem multifuncionais e conectados, como fonte de conteúdo e abrindo novas possibilidades para geração de conteúdo.

## 1.1 MOTIVAÇÃO

De acordo com (CASSOL, SANTOS, *et al.*, 2008) o cenário de telefonia celular no Brasil apresenta algumas tendências consolidadas especialmente quanto à proliferação da tecnologia 3G, a forte ampliação dos serviços disponíveis via celular (inclusive a de TV móvel) e a maneira de como o a telefonia celular é vista como recurso de inclusão digital.

O estudo indica com clareza que a tendência é que ao chegar no ano de 2016 o Brasil terá uma maior cobertura de telefonia celular aliada a uma grande disponibilidade de conectividade nos dispositivos móveis em grande parte do território. (CASSOL, SANTOS, *et al.*, 2008)

A união de TV e internet se mostra iminente especialmente com a iniciativa Google TV, mas as possibilidades da combinação de TV, internet e mobilidade ainda não se mostraram exploradas. Desde 2007 a televisão digital aberta opera no Brasil, e vem substituindo gradativamente a transmissão analógica, que está ativa desde 1950. De acordo com a portaria 652 do ministério das comunicações todas as cidades brasileiras devem estar cobertas com o sinal digital até abril de 2011 (MINISTÉRIO DAS COMUNICAÇÕES, 2006), e o fim do ciclo será marcado pelo

desligamento do sinal analógico em 2016. (MINISTÉRIO DAS COMUNICAÇÕES, 2009)

Essa substituição deve ser amplamente impulsionada pela copa do mundo de 2014 e está prevista para chegar ao fim em 2016, quando a transmissão do sinal analógico será desligado, completando o ciclo de substituição do sinal.

## 1.2 OBJETIVO GERAL

Este trabalho objetiva definir e projetar uma arquitetura flexível e construir um aplicativo como prova de conceito para prover aplicações aliadas a streaming de vídeo para dispositivos móveis.

## 1.3 OBJETIVOS ESPECÍFICOS

Dentre os objetivos específicos deste trabalho, podemos destacar:

1. a capacidade de dispositivos móveis se comunicarem tanto com servidores de streaming e quanto servidores de interatividade simultâneamente;
2. analisar e sugerir uma arquitetura que atenda aos requisitos de executar aplicações sobre streaming de vídeos;
3. desenvolver o servidor para conexão entre streaming de vídeo e aplicativos, que implemente a arquitetura proposta;
4. desenvolver uma aplicação cliente para a arquitetura proposta;
5. desenvolver um aplicativo de chat como prova de conceito desta arquitetura;

## 1.4 ESTRUTURA DO TRABALHO

Estre trabalho está estruturado na seguinte forma:

No capítulo 2 será realizada uma revisão de literatura, com objetivo de introduzir assuntos pertinentes ao escopo deste trabalho, como a linguagem de modelagem, as plataformas líderes de mercado entre dispositivos móveis, e os

diferentes protocolos utilizados para a disponibilização de vídeo via rede, bem como vantagens e desvantagens de cada.

No capítulo 3 será discutida a arquitetura da prova de conceito como um todo, quais as funcionalidades esperadas, as decisões de projeto, os diagramas de bloco, a escolha de protocolos e a composição das aplicações para os dispositivos móveis.

No capítulo 4 será abordado em mais detalhe a implementação do servidor para catálogo de serviços e vídeos, suas funcionalidades, projetos e funcionamento, enquanto no capítulo 5 estão detalhados os mesmos itens para a aplicação cliente, e no capítulo 6 para o aplicativo de exemplo da prova de conceito.

O resultado da prova de conceito e dos demais desenvolvimentos será apresentado no capítulo 7, enquanto as conclusões e a discussão sobre as possibilidades de trabalhos futuros será discutida no capítulo 8.

# 2 REVISÃO DE LITERATURA

# 2 REVISÃO DE LITERATURA

## 2.1 LINGUAGEM DE MODELAGEM.

Os modelos e diagramas apresentados neste trabalho seguirão o padrão de modelagem *Technical Architecture Modeling* (TAM), um padrão de modelagem baseado nas especificações *Meta-Object Facility* (MOF) 2.0 e *Unified Modeling Language* (UML) 2.0 que visa simplificar e reduzir a quantidade de diagramas estruturais e comportamentais necessários para modelagem da arquitetura técnica de um sistema. (SAP, 2007)

Dentre os diagramas e técnicas de modelagem disponíveis no TAM, foram escolhidos três principais diagramas para a análise e projeto das aplicações propostas. Para o melhor entendimento dos requisitos funcionais foi escolhido a criação e o detalhamento do diagrama de casos de uso, para definição da arquitetura foi escolhido o diagrama de blocos e para o melhor detalhamento da arquitetura foi escolhido o diagrama de classes.

### 2.1.1 CASOS DE USO.

De acordo com (WEILKIENS e OESTEREICH, 2007) um caso de uso é uma definição de uma séria de ações que são executadas por um sistema e que geram um resultado com alguma importância para um Ator, sendo o Ator um papel fora do sistema modelado pelo caso de uso e que tem como objetivo interagir com o sistema.

Um diagrama de caso de uso é uma descrição visual das relações entre atores e casos de uso, ou seja, o diagrama de caso de uso não descreve fluxo ou comportamento, mas sim como os elementos dos casos de uso se relacionam, facilitando a análise de requisitos. (WEILKIENS e OESTEREICH, 2007)

A Figura 1 ilustra a principal notação para representação de diagrama de caso de uso, sendo o ator representado pelo boneco, o relacionamento representado por um traço conectando o ator ao caso de uso, e o caso de uso representado pela forma eliptica. (WEILKIENS e OESTEREICH, 2007)

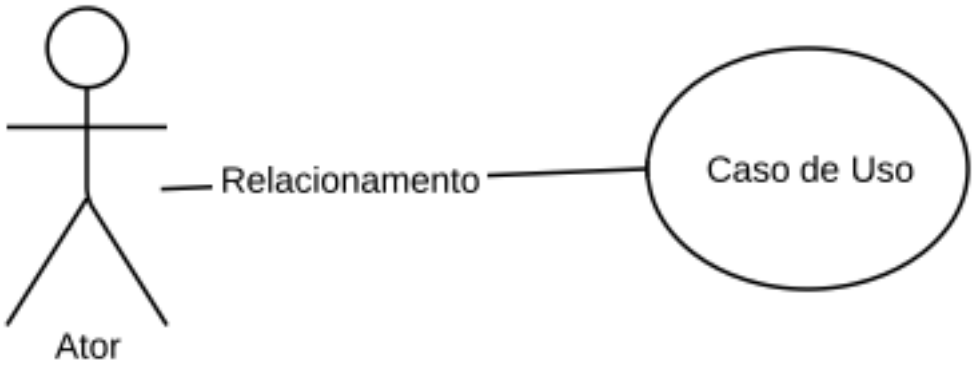


FIGURA 1 - EXEMPLO DE DIAGRAMA DE CASO DE USO

Além do relacionamento entre ator e caso de uso representando a interação entre ambos, na UML 2.0 também está previsto o relacionamento entre casos de uso, podendo ser um relacionamento de *include* ou de *extend*. A relação de *include* visa integrar um caso de uso em outro, tornando um caso de uso parte de outro, enquanto a relação de *extend* representa que um determinado caso de uso será ativado por outro em um determinado momento, quando as condições do ponto de extensão forem válidas, a figura abaixo ilustra a notação para os relacionamentos.

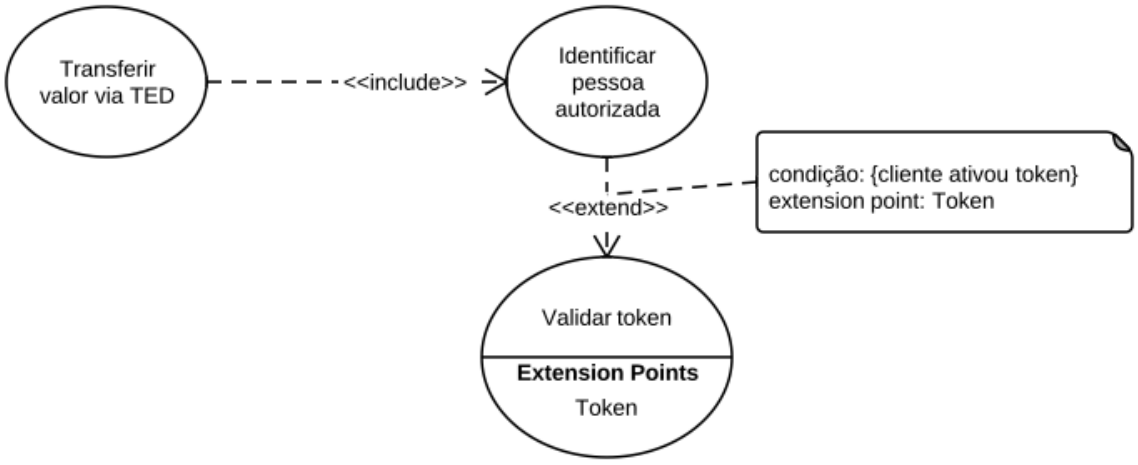


FIGURA 2 - EXEMPLO DE RELACIONAMENTO ENTRE CASOS DE USO

Para realizar o detalhamento dos casos de uso este trabalho fará uso de uma tabela padronizada conforme ilustrada no exemplo abaixo seguindo alguns itens de boas práticas determinadas por (ROSENBERG e STEPHENS, 2007), onde a descrição do caso de uso deve seguir um fluxo de eventos e respostas do sistema, seguido de um protótipo de tela para facilidade de entendimento do contexto.

| Caso de Uso | Nome descritivo do caso de uso<br>Exemplo: Selecionar produto |
|---|---|
| Atores | Identificação dos atores envolvidos<br>Exemplo: Cliente da loja |
| Objetivo | Objetivo principal do caso de uso<br>Exemplo: Permitir ao cliente da loja seleção de um artigo para compra |
| Pré-condições | Identificação de alguma condição que deve ser satisfeita antes da execução do caso de uso<br>Exemplo: O cliente da loja já está identificado no sistema |
| Pós-condições | Identificação da ação que deve ter sido realizada para considerar o caso de uso como executado corretamente.<br>Exemplo: O item escolhido foi adicionado ao carrinho de compras do cliente |
| Fluxo Básico | Descrição do fluxo principal do caso de uso, utilizando a seguinte notação:<br>[Número do item do fluxo] . [Ator] + [Ação]<br>Exemplo:<br>1. Usuário solicita listagem de itens;<br>2. Sistema lista os produtos disponíveis;<br>3. Usuário seleciona o item desejado;<br>4. Sistema adiciona item ao carrinho de compras; |

| | |
|---|---|
| Fluxo Alternativo | Descrição de fluxos alternativos, seguindo a notação abaixo:<br>[Item do fluxo básico] + [letra identificando fluxo alternativo]. [Ator] + [Ação]<br>Exemplo:<br>3a1. Usuário clica em alterar categoria;<br>3a2. Sistema encaminha usuário para Caso de uso X; |

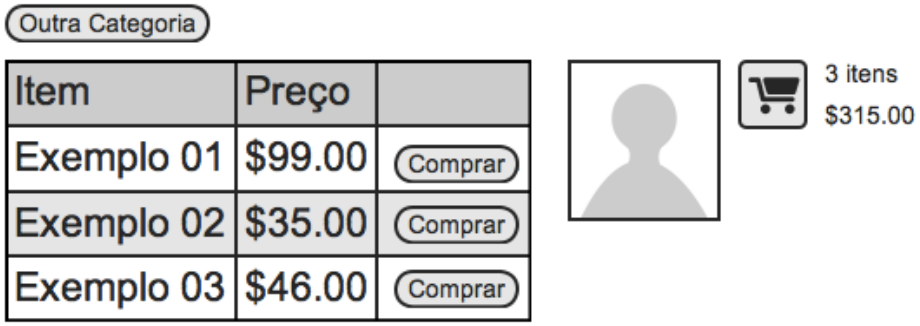


FIGURA 3 – EXEMPLO DE PROTÓTIPO DE TELA

### 2.1.2 DIAGRAMA DE BLOCO.

O objetivo do diagrama de bloco é descrever conceitualmente um sistema de informação, permitindo que o entendimento básico de um sistema seja disseminado de forma visual e facilitada. Este diagrama é composto basicamente por componentes ativos (Agentes), componentes passivos (Armazenamentos) e formas de comunicação (Acesso, Canal e Protocolo).

O componente de Agente representado pela Figura 4 corresponde a um elemento ativo e capaz de realizar uma determinada ação, de forma autônoma ou a partir de um estímulo. (SAP, 2007)

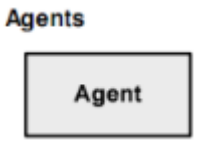


FIGURA 4 - REPRESENTAÇÃO DO AGENTE NO TAM

O componente de Canal, conforme representado na Figura 5, é um elemento passivo que deve ser utilizado para comunicação entre agentes, podendo apresentar o sentido do fluxo dos dados, e por consequência, a origem da requisição. (SAP, 2007)

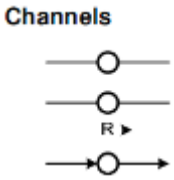


FIGURA 5 - REPRESENTAÇÃO DE UM CANAL NO TAM

O componente de Armazenamento, conforme representação da Figura 6, é um elemento passivo onde um Agente pode realizar uma ação de leitura ou escrita, e é responsável pela retenção de dados de qualquer natureza. Um armazenamento pode estar contido em um Agente, em outro Armazenamento ou em componentes e subsistemas. (SAP, 2007)

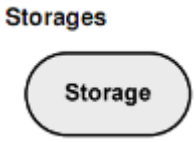


FIGURA 6 - REPRESENTAÇÃO DO ARMAZENAMENTO NO TAM

Quando um elemento ativo (Agente) realiza uma ação de leitura ou escrita sobre um elemento passivo (Armazenamento), esta ação deve estar representada com o componente de Acesso, conforme representado pela Figura 7. (SAP, 2007)

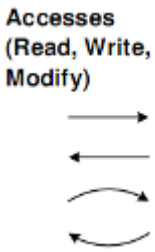


FIGURA 7 - REPRESENTAÇÃO DE ACESSOS NO TAM

### 2.1.3 DIAGRAMA DE CLASSES.

O diagrama de classes no TAM segue os mesmos padrões definidas pela UML 2.0, e portanto possui as mesmas notações para as relações de generalização, agregação e composição. Neste trabalho os diagramas de classes serão utilizados para o detalhamento dos diagramas de blocos, facilitando assim a compreensão do sistema proposto.

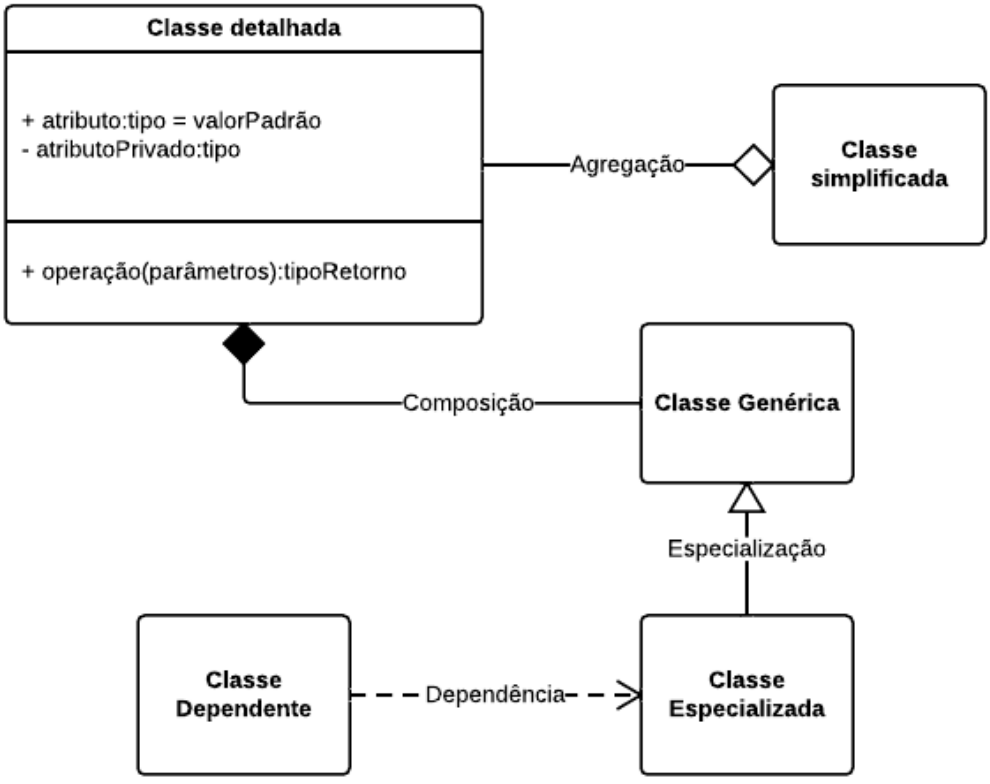


FIGURA 8 – EXEMPLO DE DIAGRAMA DE CLASSES

## 2.2 MERCADO DE SISTEMAS OPERACIONAIS PARA SMARTPHONES.

O mercado atual de smartphones está dividido entre cinco principais sistemas operacionais que de acordo com as previsões realizadas pela Gartner, tendem a mudar muito nos próximos anos, dentre os principais sistemas operacionais é possível destacar Symbian, Google Android, Blackberry OS, Apple iOS, Microsoft Windows Mobile. (PETTEY e STEVENS , 2011)

O Gráfico 1 resume o cenário atual e apresenta as previsões para as vendas de dispositivos móveis nos próximos anos e mostram uma tendência para forte ascensão do Android sobre os demais sistemas operacionais, conforme estudo realizado pela Gartner em abril de 2011. (PETTEY e STEVENS , 2011)

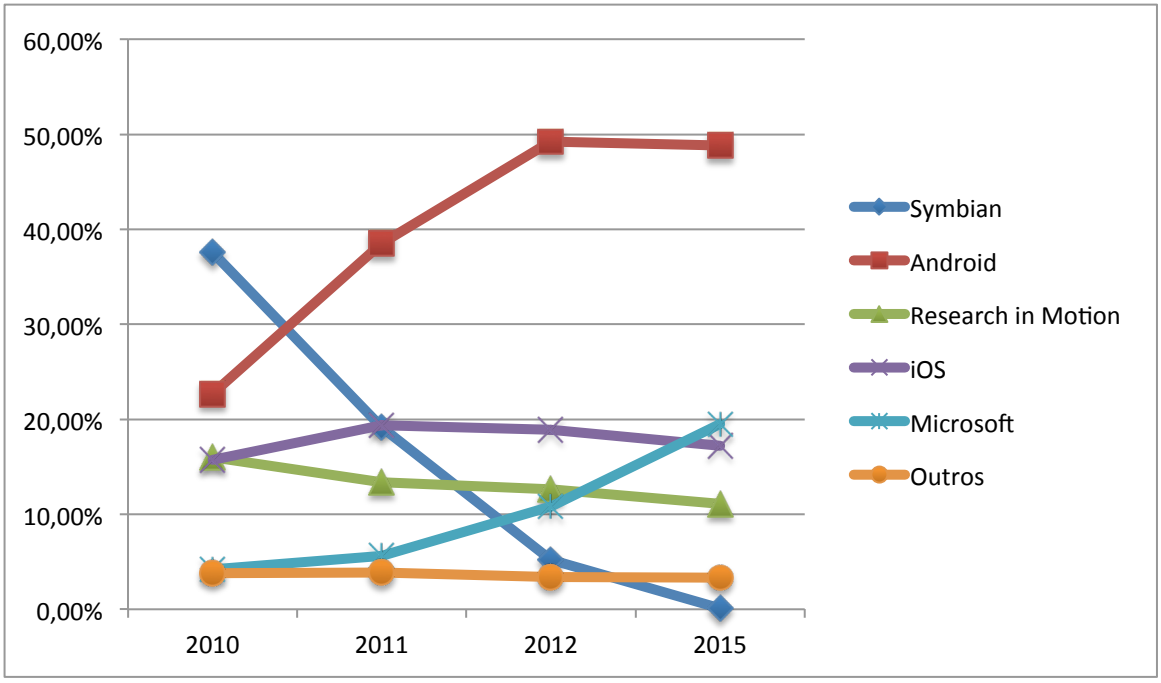


GRÁFICO 1 - DISTRIBUIÇÃO DE SISTEMAS OPERACIONAIS NA VENDA DE DISPOSITIVOS MÓVEIS PARA USUÁRIOS FINAIS

### 2.2.1 SYMBIAN.

O sistema operacional Symbian é uma plataforma desenvolvida e mantida pela fundação Symbian (formada por Nokia, Sony Ericsson, Motorola e outras

empresas). Em 2009 esta plataforma se tornou *open source* e ganhou uma loja de aplicativos, chamada Ovi Store. (DOROKHOVA e AMELICHEV, 2010)

O Symbian é baseado em *micro-kernel* e reconhecido pela alta performance, possui suporte completo a *multi-tasking*, possui um instalador nativo para verificar requisitos de segurança, e em alguns modelos traz um navegador com suporte a HTML4 e CSS2. (DOROKHOVA e AMELICHEV, 2010)

Em 2010 o Symbian representou 37,6% das vendas globais de dispositivos móveis e, de acordo com a previsão do Gartner, esse número tende a reduzir para 5,2% do total de vendas até 2012. (PETTEY e STEVENS , 2011)

### 2.2.2 IOS.

O sistema operacional iOS foi desenvolvido pela Apple para sua família de dispositivos móveis (iPod touch, iPhone e iPad) baseado no Mac OS. Para software desenvolver para iOS é necessário fazer uso das API's disponibilizadas pela Apple, utilizando o ambiente de desenvolvimento proprietário, conhecido como xcode, e usando a linguagem Objective-C, o browser é o Safari adaptado para dispositivos móveis, e portanto tem um ótimo suporte a HTML5 e CSS2. Outra característica importante do iOS é seu *multi-tasking* limitado, permitindo que aplicações que não estão ativas apenas possam tocar música ou receber pacotes via *push*. (DOROKHOVA e AMELICHEV, 2010)

Em 2010 o iOS representou 15,7% do total de vendas globais de dispositivos móveis, e está estimado em uma participação de 18,9% em 2012. (PETTEY e STEVENS , 2011)

### 2.2.3 ANDROID.

O sistema operacional Android é uma agregação de três sistemas principais, sendo a base composta pelo sistema operacional baseado em Linux, o middleware que permite a construção de aplicativos utilizando Java e uma série de aplicativos-chave construídos sobre o middleware que permitem o funcionamento básico do dispositivo. (SAHA, 2008)

Este middleware é conhecido como Dalvik, uma máquina virtual semelhante a *Java virtual machine*, mas diferenciada pelo formato dos arquivos compilados e pela forma de inicialização, uma vez que a Dalvik tem toda memória pré-alocada, que é simplesmente copiada para RAM toda vez que a Dalvik necessita iniciar, sendo assim inicializada com muito mais agilidade. (DOROKHOVA e AMELICHEV, 2010)

A abrangência de marcas e modelos do Android se deve à Open Handset Alliance, um consórcio de 78 empresas de hardware, software e telecomunicações, que trabalham juntas para definir os padrões deste sistema. (OPEN HANDSET ALLIANCE, 2007)

Conforme (PETTEY e STEVENS , 2011) o Android representou 22,7% do mercado total de vendas de dispositivos móveis, e tem estimado aumentar esta participação para 49,2% em 2012.

### 2.2.4 BLACKBERRY OS.

O sistema operacional Blackberry OS é mantido pela Research in motion, sendo uma plataforma fechada mas com um ambiente Java para rodar aplicativos desenvolvidos especificamente para este sistema operacional. Na versões mais recentes possui um browser baseada em WebKit, o que permite a exibição da maior parte da especificação do HTML5. (DOROKHOVA e AMELICHEV, 2010)

Em 2010 as vendas de sistemas com Blackberry OS representou 16,0% do total, sendo previsto uma retração nas vendas nos próximos anos, totalizando 12,6% em 2012. (PETTEY e STEVENS , 2011)

### 2.2.5 WINDOWS PHONE 7.

O sistema operacional Windows Phone 7, criado pela Microsoft, foi totalmente redesenhado quando comparado a versões anteriores. A versão 7 é baseado no Windows CE mas modificado para que todo o sistema rode sobre um único aplicativo (no caso, o único aplicativo nativo). (DOROKHOVA e AMELICHEV, 2010)

O suporte a aplicativos .NET que o são executados por este aplicativo nativo que controla o dispositivo, e portanto não tem acesso direto ao *hardware*, e sim as API's disponibilizadas por este aplicativo base. O browser é uma versão reduzida do Internet Explorer 7, com o suporte limitado aos padrões de HTML5 e CSS3. (DOROKHOVA e AMELICHEV, 2010)

A participação de sistemas operacionais Microsoft na venda de dispositivos móveis em 2010 foi de apenas 4,2%, sendo que é previsto uma rápida ascensão para 10,8% até 2012. (PETTEY e STEVENS , 2011)

## 2.3 PROTOCOLOS DE STREAMING.

Para conseguir atingir os objetivos gerais deste trabalho foi necessário realizar um estudo sobre as vantagens e desvantagens dos principais protocolos de *stream* de vídeo, e os principais softwares que implementam estes protocolos. Como resultado foi identificado que o RTSP e o *HTTP Live streaming* são os principais protocolos, sendo que para o escopo deste trabalho o *HTTP Live streaming* se mostrou mais vantajoso.

### 2.3.1 REAL TIME STREAMING PROTOCOL (RTSP).

O *Real Time Streaming Protocol* (RTSP) é um protocolo para transmissão de conteúdo multimídia em tempo real sugerido como padrão na RFC 2326 do *Internet Engineering Task Force* (IETF), no âmbito deste trabalho tem como vantagem fazer o *stream* em tempo real e já ser implementado junto a alguns componentes do Android SDK 1.6, mas como desvantagem apresenta o pouco controle sobre as propriedades do protocolo, e o fato de que redes com firewalls restritivos podem bloquear o acesso do dispositivo ao stream. (SCHULZRINNE, RAO e LANPHIER, 1998)

### 2.3.2 HTTP LIVE STREAMING.

O *HTTP adaptive bitrate streaming* não é um protocolo definido mas um conceito do funcionamento de um protocolo de *stream* sobre o protocolo HTTP. Este

conceito já é utilizado no Apple *HTTP Live Stream* (amplamente utilizado para *stream* de conteúdo no iOS, implementado desde a versão 3.0 deste sistema operacional) e no Adobe *Dynamic Streaming for Flash*, e consiste em detectar em tempo real o poder de processamento e a largura de banda do cliente permitindo ajustar o *stream* para cada tipo de cliente. Isso faz com que um mesmo vídeo possa ser codificado em diferentes *bitrates*, e a escolha do *bitrate* é realizada durante o *stream*, de acordo com a capacidade do cliente. (GANNES, 2009)

Dentre as vantagens estão o menor tempo de *buffering*, a maior qualidade possível entregue de acordo com as capacidades do dispositivo cliente, e uma vez que todo conteúdo é codificado como pequenos arquivos enviados via HTTP, o *streaming* pode funcionar através de *firewalls*, e ser mantido em cache por *proxies* ou *Content Delivery Networks (CDN)*.

A principal desvantagem deste protocolo é que a implementação nativa na plataforma Android está disponível somente a partir do Android SDK 2.3.3, e portanto para todos os tablets modernos, mas apenas poucos telefones da atualidade.

# 3 PROVA DE CONCEITO

# 3 PROVA DE CONCEITO

O escopo deste trabalho está limitado a uma prova de conceito sobre a interatividade sobre TV Digital (ou outras fontes de vídeo) aplicada a dispositivos móveis verificando se o estágio atual de tecnologia móvel suporta o desenvolvimento de aplicativos interativos padronizados em uma arquitetura que suporte futuramente outros dispositivos (televisores, computadores ou outros dispositivos eletrônicos), e portanto, será composto por dois componentes principais: o servidor de *stream* e interatividade e o cliente para o dispositivo móvel.

O servidor de *stream* e interatividade será responsável pelas informações sobre *stream* de vídeo e receberá requisições de dispositivos com a aplicação cliente e retornar os informações necessárias para montagem da interface e lógica de interatividade.

A aplicação cliente para dispositivo móvel será responsável por conter a lógica de montagem de interface com o usuário e comportamento do dispositivo baseado no retorno do servidor, e será capaz de exibir o *stream* de vídeo e fazer requisições para os aplicativos interativos fornecidos pelo servidor de interatividade.

A figura abaixo ilustra a concepção do sistema em termos gerais, se limitando aos principais componentes e funcionalidades necessárias para atender às expectativas desta prova de conceito.

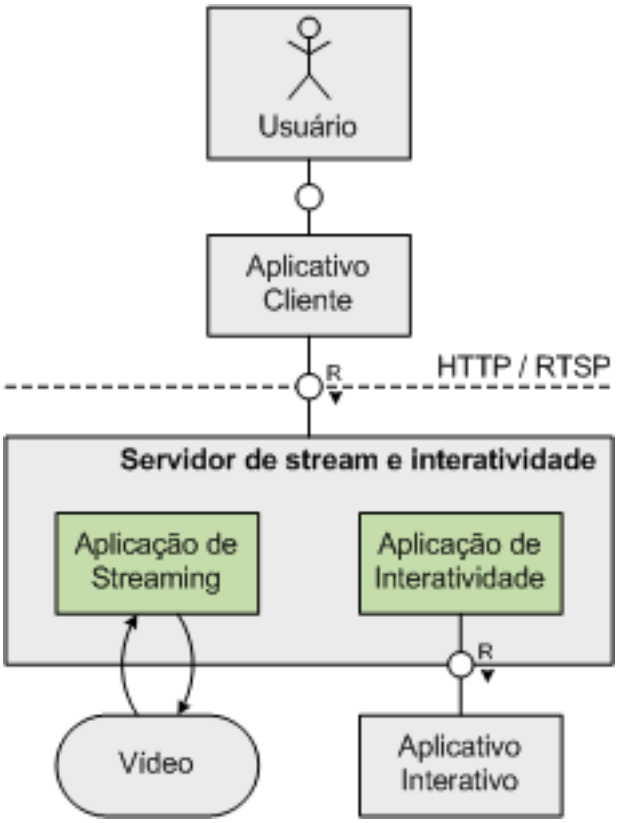


FIGURA 9 - CONCEPÇÃO DO SISTEMA PROPOSTO

## 3.1 FUNCIONALIDADES ESPERADAS

Baseado nos objetivos gerais deste trabalho, é possível determinar o mínimo de funcionalidades esperadas para todo o sistema a ser desenvolvido para a prova de conceito, que por sua vez representam os requisitos funcionais (em alto nível) deste sistema:

1. deve existir um servidor para catálogo que unifique tanto *streams* de videos quanto aplicativos interativos;
2. o servidor de catálogo deve permitir a conexão de qualquer servidor de *stream* de vídeo, desde que o protocolo de *stream* seja suportado pelo sistema;
3. o aplicativo cliente deve ter a capacidade de executar o *stream* de vídeo e um aplicativo a partir das definições que serão retornadas pelo servidor de catálogo;

4. o aplicativo cliente deve ser extensível, permitindo implementações em diversos sistemas operacionais;
5. o aplicativo cliente deve suportar diversas resoluções de telas (tanto *tablets* quanto *smartphones*) e orientação vertical e horizontal, redimensionando o vídeo e o aplicativo de acordo com o dispositivo;
6. o aplicativo cliente deve ser implementada de maneira que permita a evolução paralela entre as interfaces de execução de aplicativos e a interface para execução do *streaming* de vídeo;
7. o protocolo de comunicação entre o catálogo e o dispositivo móvel deve ser de fácil adaptação para os diversos sistemas operacionais;
8. a definição e codificação de novos aplicativos para os dispositivos móveis deve permitir a evolução constante em termos de tecnologia de hardware e software, e a curva de aprendizagem para desenvolvedores deve ser baixa para facilitar a adoção do sistema como um todo;

## 3.2 DECISÕES DE PROJETO

Considerando os requisitos para a prova de conceito, foi identificado que a independência de protocolo de streaming e a utilização de um padrão estabelecido para criação de aplicativos é a melhor saída para um sistema de streaming multi-plataforma e multi-dispositivos.

Para ter uma aplicação cliente independente de protocolo e utilizar um padrão bem estabelecido para criação dos aplicativos interativos, a melhor opção identificada foi adaptar um *web browser* transparente sobre a execução do vídeos, utilizando assim os recursos nativos dos sistemas operacionais tanto de renderização do conteúdo HTML, quanto de renderização do *stream* de vídeo, e fazer uso de *JavaScript Interface* (JNI) para realizar a comunicação entre o Javascript do aplicativo interativo e a aplicação cliente no dispositivo.

Para esta prova de conceito foi optado por desenvolver uma aplicação cliente para o sistema operacional Android (por ser o líder de mercado e pela variedade de dispositivos), um servidor para manter o catálogo de canais e

aplicativos interativos em Java (por motivos de compatibilidade com diversas plataformas), e aplicativos interativos hospedados na nuvem do Google (conhecido como Google App Engine).

## 3.3 PROJETO EM ALTO NÍVEL

A Figura 10 ilustra a arquitetura em alto nível para o servidor de catálogo e para o dispositivo móvel, explorando os canais de comunicação entre a interface do usuário final e do administrador de sistema com a aplicação cliente e servidor, respectivamente.

O servidor de *stream* é propositalmente colocado com multiplicidade N uma vez que o mesmo pode ser diversas instâncias capturando e convertendo vídeo simultaneamente, todos disponibilizados como canais para o aplicativo cliente.

Já a multiplicidade N do agente de Aplicativo Cliente se deve ao fato de que implementações para outros sistemas operacionais serão possíveis quando implementada corretamente a interface com o servidor de catálogo.

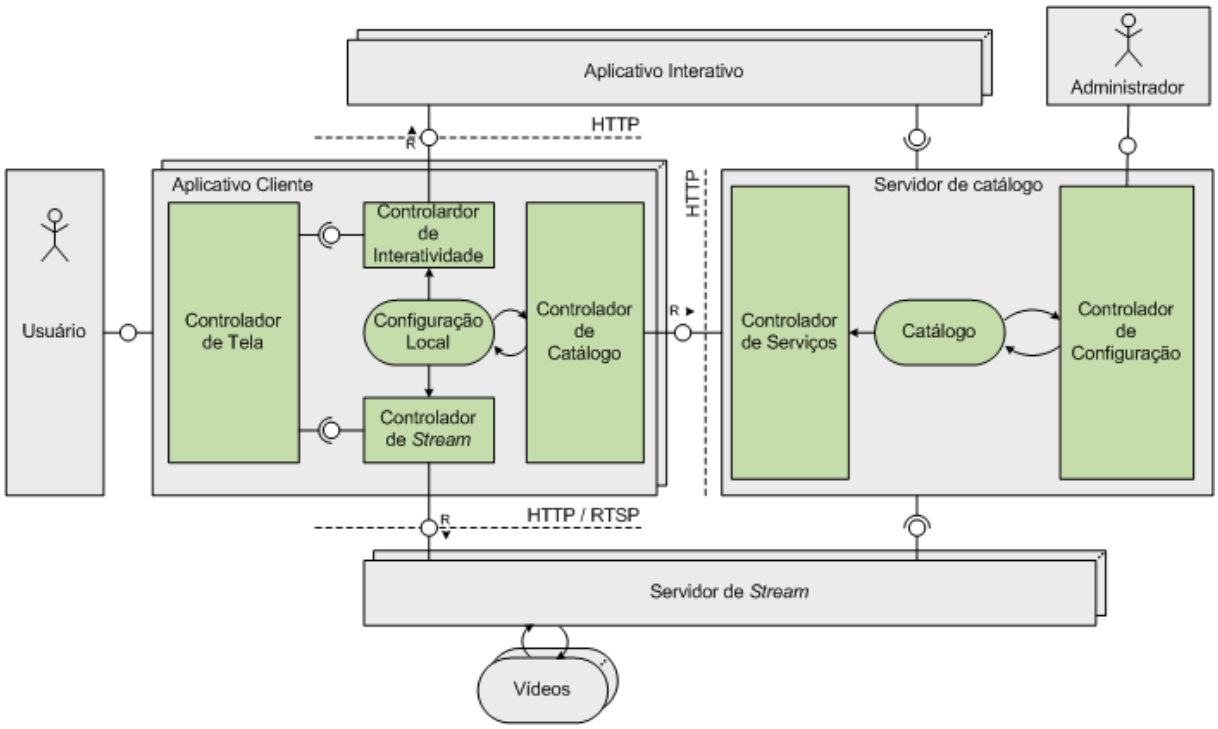


FIGURA 10 - ARQUITETURA EM ALTO NÍVEL PARA O SISTEMA PROPOSTO

## 3.4 PROTOCOLO DE STREAMING

Considerando a arquitetura proposta neste trabalho, o protocolo de streaming é dependente apenas do servidor de streaming configurado no servidor de catálogo de aplicativos e canais, mas considerando as vantagens e desvantagens dos principais protocolos de *stream*, o *HTTP Live Streaming* é o mais indicado para a transmissão de vídeos para dispositivos móveis com diferentes velocidades de conexão e diferentes formatos de tela.

Conforme descrito por (APPLE, 2011), o funcionamento do HTTP Live Streaming é uma extensão das listas de reprodução do formato *Extended M3U Playlist* (M3U). Para a transmissão de uma fonte de vídeo via *HTTP Live Streaming* o vídeo a ser transmitido deve ser dividida em partes menores, estas partes alimentarão um arquivo M3U, que será constantemente atualizado pelo servidor de *stream* com o caminho das partes mais atualizadas.

Fica a cargo do aplicativo que quer exibir o vídeo recuperar cada um dos pedaços indicados na lista de reprodução, e atualizar constantemente a definição da lista para recuperar os pedaços de vídeos mais recentes.

Os novos rótulos definidos na extensão do arquivo M3U descrita no *HTTP Live Streaming* buscam informar o aplicativo que está exibindo o vídeo sobre os atributos relevantes para a correta exibição da fonte de vídeo, como por exemplo identificar a duração de cada trecho de vídeo e a sequência de reprodução dos mesmos. (APPLE, 2011)

Outros rótulos adicionados pela extensão buscam informar qual a largura de banda recomendada para a exibição, e portanto uma lista de reprodução pode apontar para outras listas, cada qual possuindo uma fonte de qualidade diferente para uma mesma transmissão de vídeo. (APPLE, 2011)

No exemplo de lista de reprodução abaixo, baseado nos exemplos de (APPLE, 2011), está ilustrada a transmissão de um vídeo com trechos de 8 segundos cada.

```
#EXTM3U
#EXT-X-TARGETDURATION:8
#EXT-X-MEDIA-SEQUENCE:2680
```

```
#EXTINF:8,
https://priv.example.com/fileSequence2680.ts
#EXTINF:8,
https://priv.example.com/fileSequence2681.ts
#EXTINF:8,
https://priv.example.com/fileSequence2682.ts
```

Neste segundo exemplo, também baseado nos exemplos de (APPLE, 2011), está ilustrado uma lista de reprodução apontando para outras listas de reprodução, levando em consideração a largura de banda do aplicativo cliente para determinar a melhor qualidade possível para cada conexão.

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1280000
http://example.com/low.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2560000
http://example.com/mid.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=7680000
http://example.com/hi.m3u8
```

## 3.5 PROTOCOLO PARA TROCA DE MENSAGENS CLIENTE-SERVIDOR

Levando em consideração a variedade de dispositivos que podem se comunicar com o servidor de catálogo, o protocolo proposto para a comunicação entre cliente e servidor deve ser simplificado e baseado em uma tecnologia presente em diversos dispositivos e de fácil implementação. Para atender estes requisitos funcionais a comunicação foi baseada no protocolo HTTP e desenhada utilizando o estilo arquitetural conhecido como REST.

De acordo com (FIELDING, 2000), o *Representational State Transfer* (REST) é um estilo arquitetural para sistemas hipermídia distribuídos, utilizando uma série de métodos sobre o protocolo HTTP para realizar ações em determinados objetos.

No âmbito deste trabalho, foi utilizado o método GET, seguido da URL que representa qual lista de objetos que o aplicativo cliente deseja receber como o padrão para obtenção dos canais ou aplicativos disponíveis, sendo o retorno uma

serialização em XML dos objetos existentes do servidor e visíveis para o dispositivo requisitante.

A figura abaixo ilustra um dispositivo móvel requisitando a lista de canais disponíveis para o servidor, e o servidor respondendo a requisição com os objetos de canal presentes no dispositivo de forma serializada.

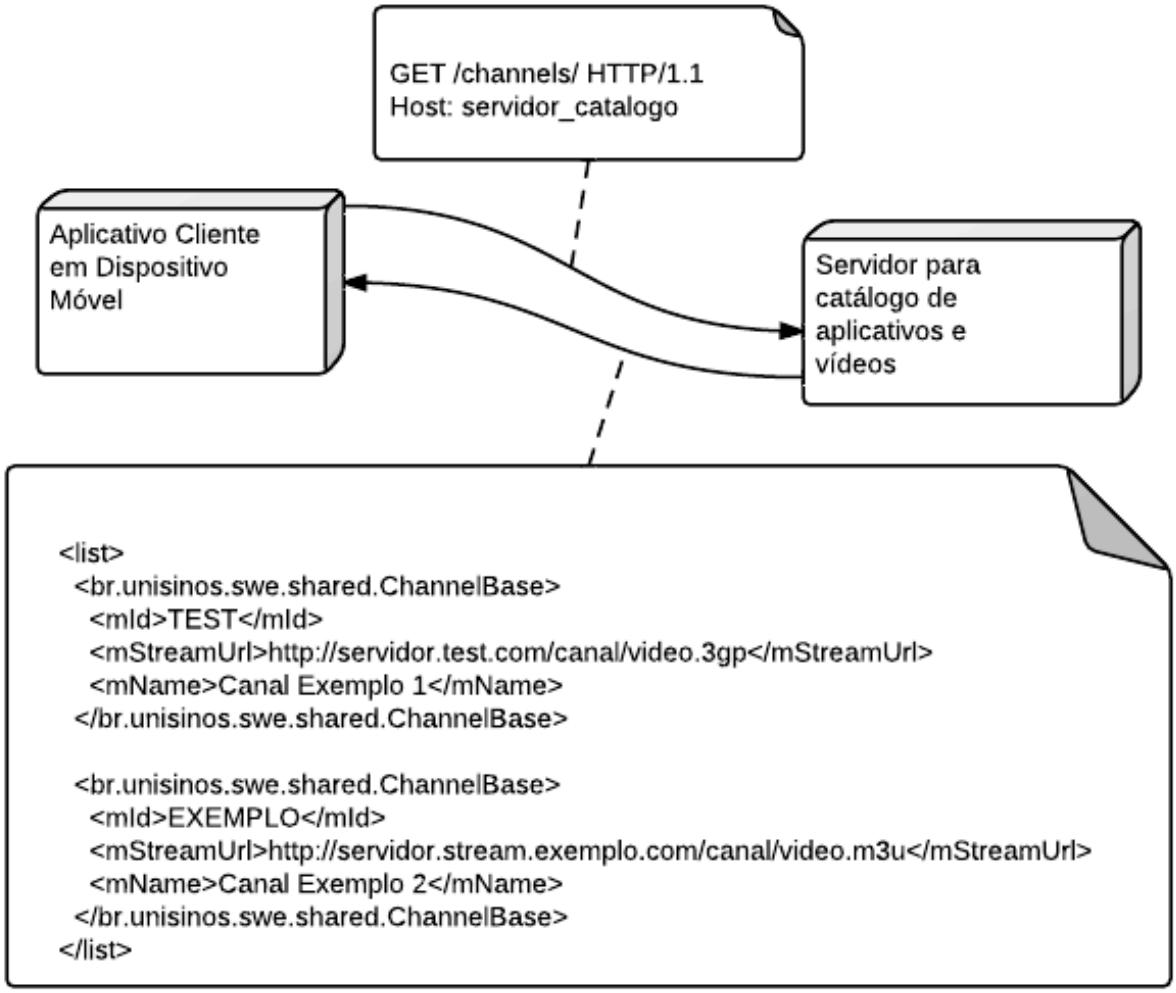


FIGURA 11 - EXEMPLO DE REQUISIÇÃO REST PARA LISTAGEM DE CANAIS

Conforme ilustrado na figura acima, quando um dispositivo móvel realiza uma requisição HTTP GET no endereço do servidor seguido pelo diretório /channels/ (no formato http://servidor/channels) o servidor para catálogo de aplicativos e canais fica responsável por responder com um XML representando a serialização de todos os canais disponíveis.

Neste trabalho esta abordagem será utilizado por toda comunicação entre aplicativo cliente e servidor de catálogo, com o detalhamento sobre os métodos e funcionalidades disponíveis no servidor para catálogo de aplicativos e canais descritos no capítulo 4.

## 3.6 COMPOSIÇÃO DE APLICATIVOS DE INTERATIVIDADE

Conforme já descrito no subcapítulo de decisões de projeto, foi escolhida a padronização da *World Wide Web* para alcançar o objetivo de que o aplicativo de interatividade seja de fácil desenvolvimento e que possa ser interpretado por qualquer plataforma.

Desta forma, para realizar o desenvolver aplicativos interativos para prova de conceito é apenas necessário criar uma página utilizando HTML (tanto a versão 4 quanto a versão 5 já são suportadas pelos principais sistemas operacionais) e JavaScript.

A tecnologia proposta para comunicação entre aplicativo interativo e sistema operacional é o uso de JavaScript Interface, sendo assim possível para o aplicativo interativo receber informações do dispositivo móvel através de métodos pré-definidos, detalhados no capítulo 5.

A transmissão de dados nesta interface proposta será através da serialização de objetos utilizando *JavaScript Object Notation* (JSON), visto a facilidade de serialização e a interface nativa para de-serialização no JavaScript.

O JSON é um formato de serialização de dados estruturados em forma de texto, sendo capaz de representar quatro tipos primitivos (*strings*, números, booleanos e nulos) e dois tipos estruturados (objetos e *arrays*), e é nativamente suportado pelo linguagem JavaScript. (CROCKFORD, 2006)

O exemplo abaixo ilustra a serialização de um objeto, com os atributos de nome, sobrenome, idade, endereço (representando um objeto) e números para contato (representando um *array* de objetos).

```
{
     "nome": "Roberto",
     "sobrenome": "Schmit",
```

```
    "idade": 35,
    "endereco":
    {
        "rua": "Av. Unisinos, 500",
        "cidade": "São Leopoldo",
        "estado": "RS"
    },
    "contato":
    [
        {
          "tipo": "residencial",
          "numero": "(51) 3535-3535"
        },
        {
          "tipe": "celular",
          "numero": "(51) 9999-0000"
        }
    ]
}
```

# 4 SERVIDOR PARA CATÁLOGO DE APLICATIVOS E CANAIS

# 4 SERVIDOR PARA CATÁLOGO DE APLICATIVOS E CANAIS

Conforme descrito nas decisões de projeto, o servidor de catálogo de serviços foi criado com o objetivo de permitir uma ampla variedade de servidores e tecnologias de *stream*, e o reaproveitamento de aplicações de interatividade sobre o *stream*, sendo o ponto de referencia para o aplicativo cliente localizar os canais disponíveis para visualização e os aplicativos disponíveis para o canal selecionado.

A tecnologia escolhida para o servidor de catálogos foi Java, utilizando o *Restlet framework* que permite um controle sobre as requisições feitas ao servidor de catálogo, servidor tanto como *web server* Java para a construção da área administrativa, e fornece facilidades para criação de serviços *RESTful*, técnica escolhida para troca de mensagens entre cliente e servidor de catálogo.

## 4.1 FUNCIONALIDADES E REQUISITOS

Para a construção do servidor de catálogo de serviços e vídeos, foram identificados os seguintes requisitos funcionais:

1. O administrador deve poder incluir, alterar e excluir canais do catálogo;
2. O administrador deve poder incluir, alterar e excluir aplicativos do catálogo;
3. O administrador deve poder incluir ou excluir conexões entre aplicativos e canais;
4. O servidor de catálogo deve responder a uma requisição de listagem de canais feita pelo aplicativo cliente instalado no dispositivo móvel com a lista de canais disponíveis;
5. O servidor de catálogo deve responder a uma requisição de listagem de aplicativos para um determinado canal feita a partir do aplicativo cliente instalado no dispositivo móvel com a lista de aplicativos disponíveis para o canal selecionado.

Os requisitos funcionais podem ser traduzidos em casos de uso, e por sua vez descritos de forma sistemática, fornecendo assim uma base para a construção. A figura abaixo ilustra os atores e casos de uso identificados para este sistema.

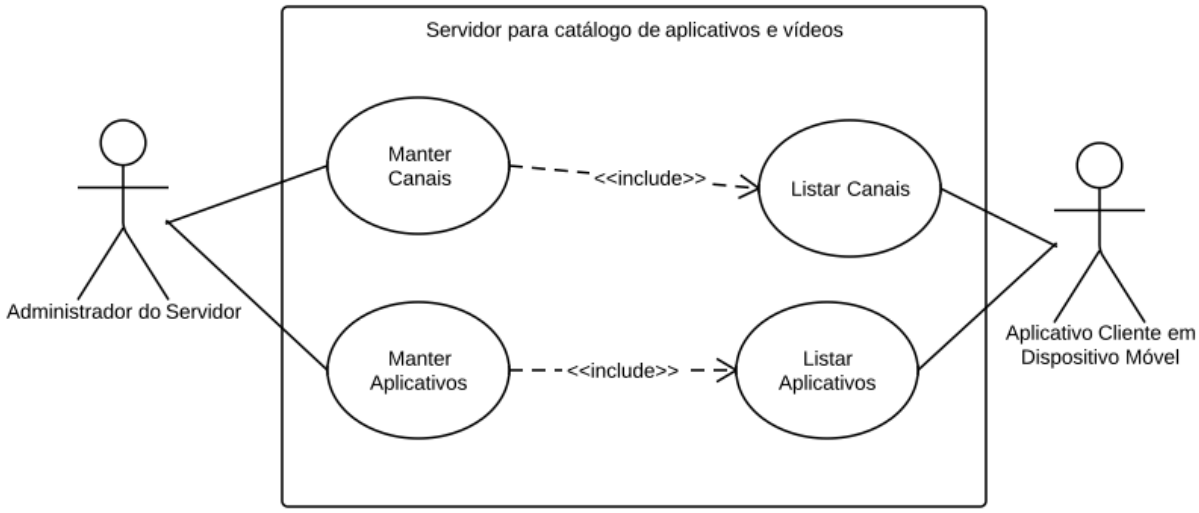


FIGURA 12 - CASOS DE USO DO SERVIDOR DE CATÁLOGO

| Caso de Uso | Manter Canais |
|---|---|
| Atores | Administrador do sistema |
| Objetivo | Permitir a inclusão, alteração e exclusão de Canais.<br>Permitir a conexão entre Aplicativo e Canal. |
| Pré-condições | Nenhuma |
| Pós-condições | O canal foi adicionado com sucesso |
| Fluxo Básico | 1. Usuário acessa a tela de cadastro de canal;<br>2. Sistema lista os canais cadastrados;<br>3. Usuário preenche o cadastro do canal;<br>4. Usuário clica em salvar;<br>5. Sistema armazena o novo canal; |
| Fluxo Alternativo | Fluxo alternativo de exclusão de canal<br>3a. Usuário escolhe um canal da lista de canais;<br>3a1. Usuário seleciona excluir canal;<br>3a2. Sistema remove o canal selecionado; |

| | |
|---|---|
| | Fluxo alternativo de alteração de canal<br>3b. Usuário escolhe um canal da lista de canais;<br>3b1. Usuário altera alguma informação do canal;<br>3b2. Usuário clica em salvar;<br>3b3. Sistema atualiza as informações do canal selecionado;<br><br>Fluxo alternativo de conexão de aplicativo:<br>3c. Usuário escolhe um canal da lista de canais;<br>3c1. Usuário escolhe opção de conectar aplicativo;<br>3c2. Sistema exibe a lista de aplicativos disponíveis;<br>3c3. Usuário escolhe um aplicativo;<br>3c4. Usuário clica em salvar;<br>3c5. Sistema atualiza a base de dados conectando aplicativo e canal; |

A figura abaixo ilustra a concepção do caso de uso de Manter canais, junto ao *pop-up* de fluxo alternativo para adicionar aplicativos.

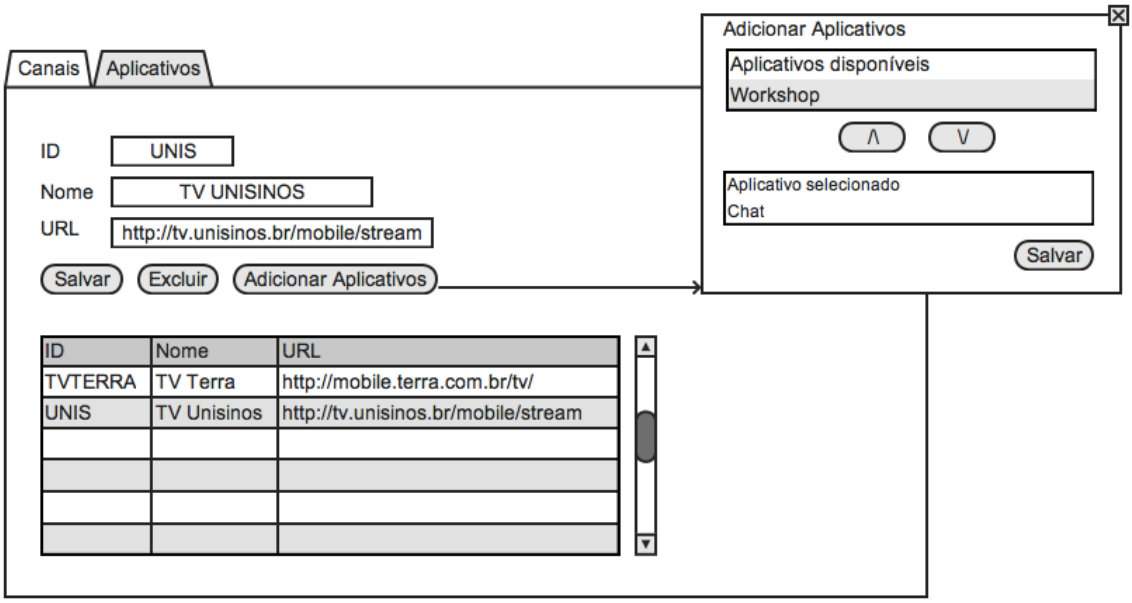


FIGURA 13 - PROTÓTIPO DO CASO DE USO MANTER CANAIS

| Caso de Uso | Manter Aplicativos |
|---|---|
| Atores | Administrador do sistema |
| Objetivo | Permitir a inclusão, alteração e exclusão de Aplicativos. |
| Pré-condições | Nenhuma |
| Pós-condições | O aplicativo foi adicionado com sucesso |
| Fluxo Básico | 1. Usuário acessa a tela de cadastro de aplicativos;<br>2. Sistema lista os aplicativos cadastrados;<br>3. Usuário preenche o cadastro do aplicativo;<br>4. Usuário clica em salvar;<br>5. Sistema armazena o novo aplicativo; |
| Fluxo Alternativo | Fluxo alternativo de exclusão de aplicativo<br>3a. Usuário escolhe um aplicativo da lista de aplicativos;<br>3a1. Usuário seleciona excluir aplicativo;<br>3a2. Sistema remove o aplicativo selecionado; |

| | |
|---|---|
| | Fluxo alternativo de alteração de aplicativos<br>3b. Usuário escolhe um aplicativo da lista de canais;<br>3b1. Usuário altera alguma informação do aplicativo selecionado;<br>3b2. Usuário clica em salvar;<br>3b3. Sistema atualiza as informações do aplicativo selecionado; |

A figura abaixo ilustra a interface administrativa para o caso de uso de Manter Aplicativos.

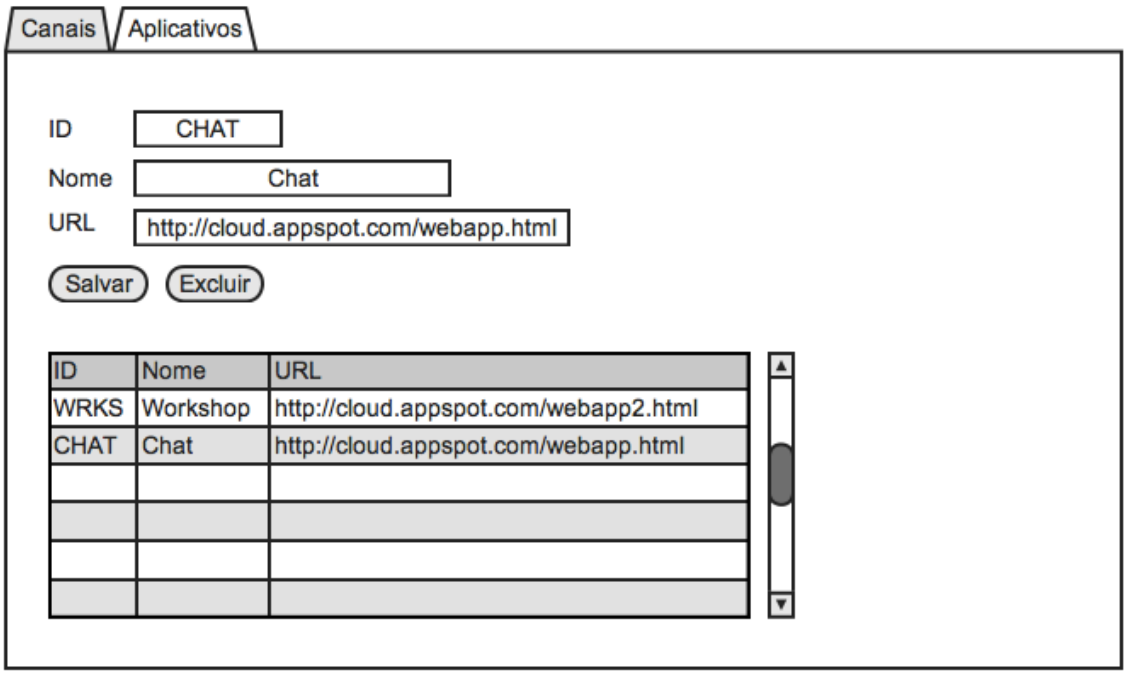


FIGURA 14 - PROTÓTIPO DO CASO DE USO MANTER APLICATIVOS

| Caso de Uso | Listar canais |
|---|---|
| Atores | Aplicativo cliente em dispositivo móvel |
| Objetivo | Permitir o retorno da lista de canais disponíveis |
| Pré-condições | Nenhuma |
| Pós-condições | O aplicativo cliente recebe uma lista com os canais cadastrados |

| | |
|---|---|
| Fluxo Básico | 1. Aplicativo cliente realiza uma requisição solicitando a listagem de canais;<br>2. Sistema retorna lista de canais disponíveis conforme formato definido na arquitetura; |

A figura abaixo ilustra o processo de comunicação entre dispositivo móvel e servidor de catálogo, exemplificando a serialização do objeto de retorno.

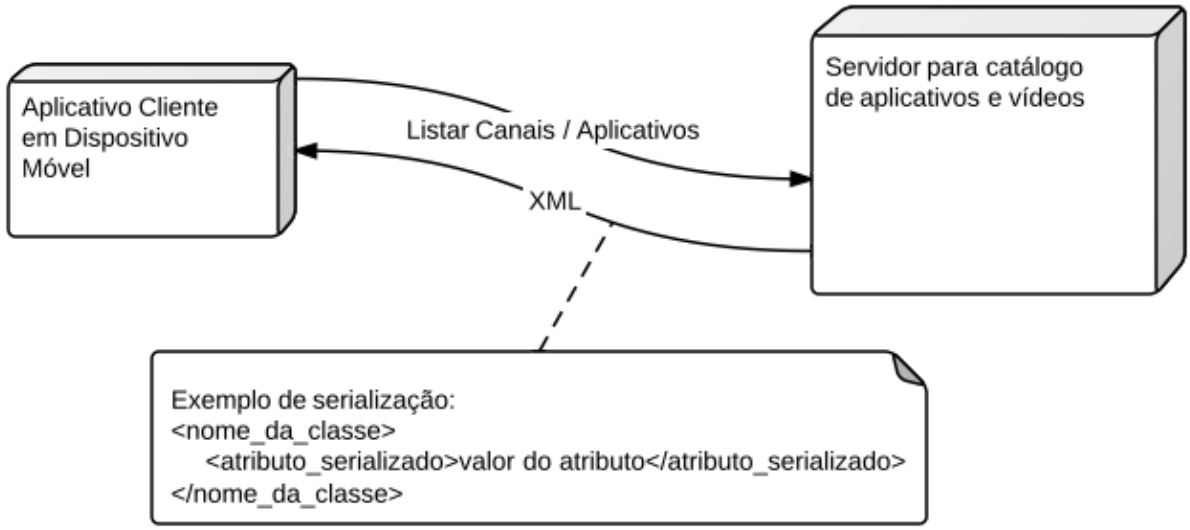


FIGURA 15 - EXEMPLO DO FLUXO DE COMUNICAÇÃO DOS CASOS DE USO DE LISTAGEM

| Caso de Uso | Listar aplicativos de um canal |
|---|---|
| Atores | Aplicativo cliente em dispositivo móvel |
| Objetivo | Permitir o retorno da lista de aplicativos disponíveis para um determinado canal |
| Pré-condições | Nenhuma |
| Pós-condições | O aplicativo cliente recebe uma lista com os aplicativos disponíveis |
| Fluxo Básico | 1. Aplicativo cliente realiza uma requisição solicitando a listagem de aplicativos para um determinado canal, conforme estrutura de requisição definida na arquitetura; |

| | |
|---|---|
| | 2. Sistema retorna lista de aplicativos disponíveis para o canal selecionado conforme formato definido na arquitetura; |

## 4.2 ARQUITETURA E PROJETO DE SOFTWARE

No diagrama de blocos da Figura 16 está ilustrada a arquitetura do sistema em nível de funcionalidade, onde foram identificados os seguintes agentes:

1. **Façade Servlets:** É o agente de fronteira do servidor de catálogo, composto por dois outros agentes que funcionam como pontos de comunicação com os agentes externos, detalhados nos itens a seguir;
2. **Páginas administrativas:** É o agente responsável pelo controle de canais e aplicativos, permitindo o cadastro, exclusão ou alteração de qualquer objeto disponível no catálogo;
3. **Serviços RESTful:** É o agente responsável por fornecer as informações conforme requisições HTTP realizadas pelos dispositivos móveis utilizando o estilo arquitetural REST;
4. **Controlador de Aplicativos:** É o agente que tem por objetivo persistir e buscar os aplicativos conforme requisitado pelos agentes Façade e Controlador de Canais;
5. **Controlador de Canais:** Este agente é semelhante ao agente Controlador de Aplicativos, com a diferença que além de persistir e buscar canais conforme requisição do agente Façade, este agente gera requisições ao Controlador de Aplicativos e é responsável por persistir o relacionamento entre canais e aplicativos.

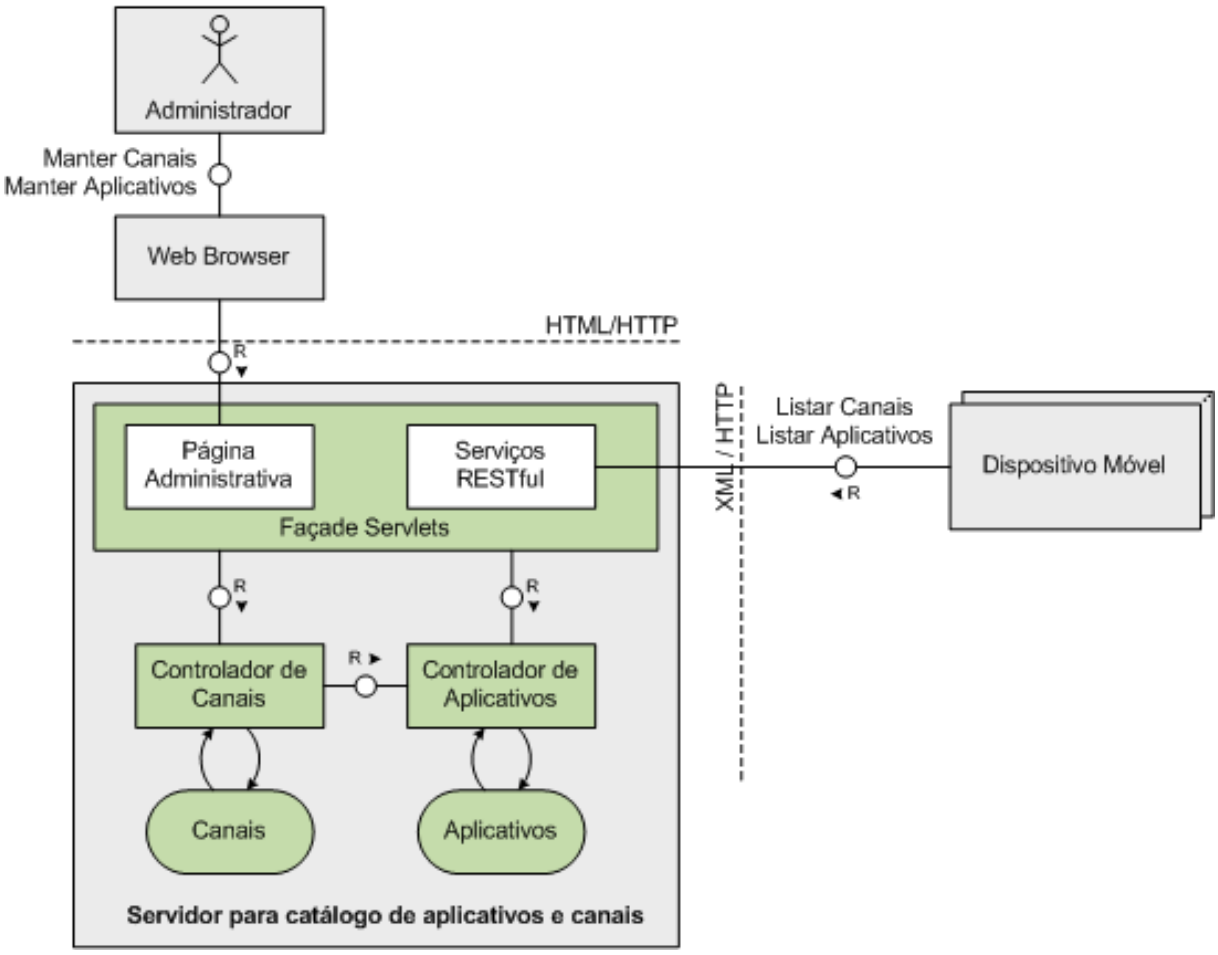


FIGURA 16 - ARQUITETURA PROPOSTA PARA O SERVIDOR PARA CATÁLOGO DE APLICATIVOS E CANAIS

Conforme já definido no capítulo 3, o servidor para catálogo de aplicativos e canais deve disponibilizar serviços REST para os aplicativos clientes, para atender tal requisito o servidor de catálogo responderá às requisições HTTP solicitando a listagem de canais e de aplicativos.

A Figura 17 ilustra o diagrama de classes dos componentes que serão compartilhados entre os projetos de servidor e aplicativo cliente, sob o pacote nomeado “br.unisinos.swe.shared“.

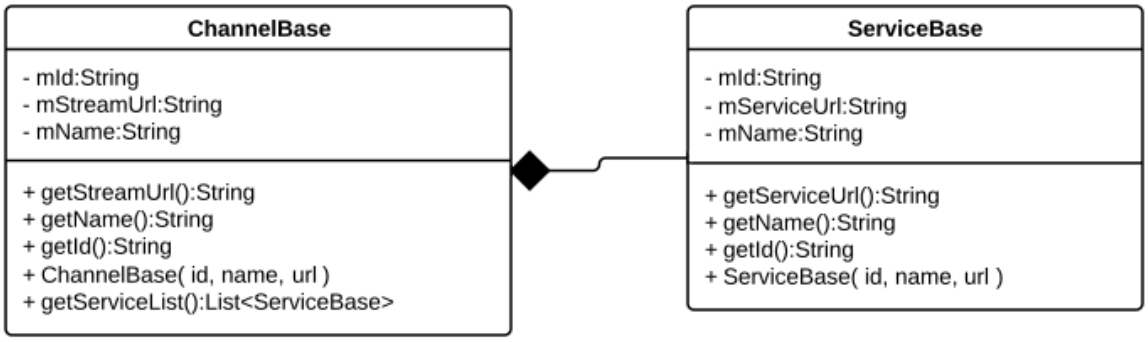


FIGURA 17 - DIAGRAMA DE CLASSES DOS COMPONENTES COMPARTILHADOS

Os métodos disponíveis nesta prova de conceito para os dispositivos móveis e as possíveis respostas já serializadas estão exemplificados abaixo.

1. GET /channels/

   Esta requisição será respondida com a lista de canais serializados em XML, conforme estrutura abaixo:

```
<list>
  <br.unisinos.swe.shared.ChannelBase>
    <mId>0</mId>
    <mStreamUrl>http://tv.terra.com.br/video/list.m3u</mStreamUrl>
    <mName>Terra TV</mName>
  </br.unisinos.swe.shared.ChannelBase>
</list>
```

2. GET /channels/{ID}/services

   Esta requisição deverá ter o campo {ID} preenchido com o atributo ID do canal que se deseja selecionar os serviços, conforme estrutura descrita abaixo:

```
<list>
  <br.unisinos.swe.shared.ServiceBase>
    <mId>0</mId>
    <mServiceUrl>http://interativo.appspot.com/chat/chat.html</mServiceUrl>
    <mName>Sala de bate-papo</mName>
  </br.unisinos.swe.shared.ServiceBase>
</list>
```

# 5 APLICAÇÃO CLIENTE

# 5 APLICAÇÃO CLIENTE

Conforme descrito no capítulo de decisão de projeto, a plataforma de dispositivo móvel escolhida para o desenvolvimento do aplicativo cliente foi a Android, e portanto o desenvolvimento deste aplicativo deve ser realizada em Java, utilizando o SDK liberado pelo Google para desenvolvimentos de aplicativos para Android.

A premissa básica para poder desenvolver o aplicativo cliente é de que o sistema operacional do dispositivo móvel suporte processamento paralelo, permitindo assim sobrepor um *web browser* (seja nativo, ou baseado nos motores *open source* como Gecko ou WebKit) em um *media player* (preferencialmente nativo, devido ao alto uso de processamento).

No caso do Android, é possível utilizar e sobrepor componentes nativos (tanto de *web browser* quanto de *media player*), com apenas uma limitação no *media player* de protocolos de *streaming*, visto que o *HTTP Live Streaming* foi implementado apenas na versão 2.3.3.

Neste trabalho o aplicativo será desenvolvido para ser compatível com a versão 2.1 ou superior do Android, devido ao dispositivo utilizado para teste da prova de conceito, que é um Motorola Dext com upgrade para Android 2.3.3.

## 5.1 FUNCIONALIDADES E REQUISITOS

Os requisitos funcionais identificados para a construção do aplicativo cliente para dispositivos móveis foram os seguintes:

1. O usuário deve poder configurar o endereço servidor de catálogo para recuperar informações de canais e serviços;
2. O usuário deve poder escolher um canal para exibição a partir da lista do catálogo;
3. O usuário deve poder escolher um aplicativo interativo da lista de aplicativos interativos configuradas no servidor de catálogo para um determinado canal;
4. O usuário deve poder parar a exibição do vídeo e reiniciar a mesma;

5. O usuário deve poder esconder e mostrar um aplicativo interativo, para facilitar a visualização do canal sem encerrar um aplicativo interativo;
6. O usuário deve poder terminar a execução de um aplicativo interativo independente a exibição do vídeo do canal.

A figura abaixo ilustra os casos de uso e atores identificados para construção da aplicação cliente a partir dos requisitos funcionais, e é seguida pelo protótipo de tela (ilustrado pela Figura 19), que tem por objetivo facilitar o entendimento dos casos de uso.

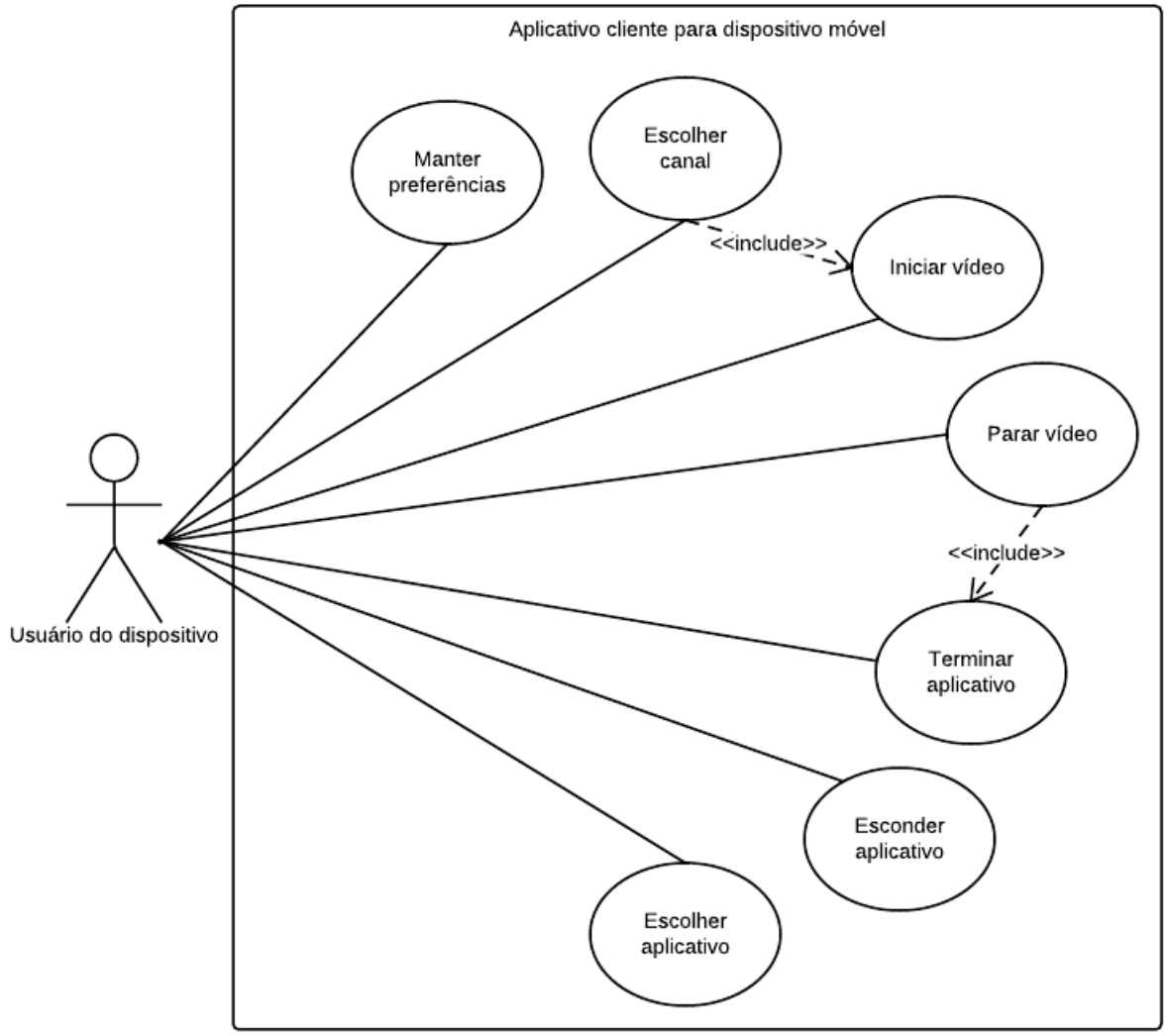


FIGURA 18 - CASOS DE USO DO APLICATIVO CLIENTE

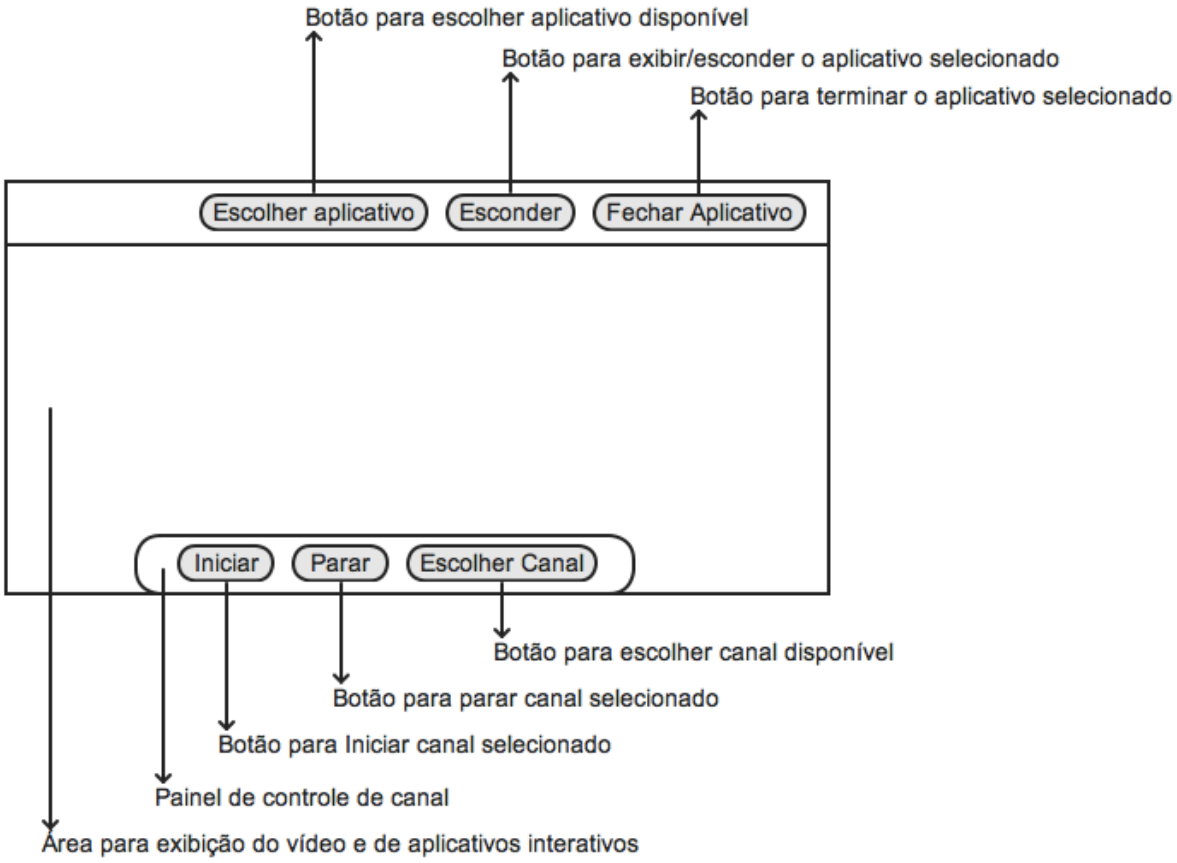


FIGURA 19 - PROTÓTIPO DE TELA DO APLICATIVO CLIENTE

| Caso de Uso | Manter preferências |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário configure as preferencias sobre o aplicativo |
| Pré-condições | Nenhuma |
| Pós-condições | O sistema registra as informações das preferencias do usuário |
| Fluxo Básico | 1. Usuário abre o menu;<br>2. Usuário seleciona o item de preferencias;<br>3. Sistema retorna lista de preferências;<br>4. Usuário altera as preferências;<br>5. Usuário clica no botão voltar; |

| | |
|---|---|
| | 6. Sistema registra as preferências;<br>7. Sistema carrega a lista de canais disponíveis;<br>8. Sistema volta para tela anterior; |

| Caso de Uso | Escolher canal |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário escolha o canal para ser exibido |
| Pré-condições | O usuário já definiu as preferências de servidor de catálogo |
| Pós-condições | O sistema define um canal como selecionado |
| Fluxo Básico | 1. Usuário clica no botão de escolher canal;<br>2. Sistema exibe a lista de canais;<br>3. Usuário seleciona o canal e clica em OK;<br>4. Sistema inicia o vídeo do canal selecionado; |

| Caso de Uso | Iniciar vídeo |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário veja o vídeo do canal selecionado |
| Pré-condições | O usuário já escolheu o canal que será exibido |
| Pós-condições | O sistema exibe o vídeo do canal selecionado |
| Fluxo Básico | 1. Usuário clica no botão de iniciar vídeo;<br>2. Sistema carrega o vídeo do canal selecionado;<br>3. Sistema exibe o vídeo do canal selecionado; |
| Fluxo de Exceção | 2a1. Sistema não consegue carregar o vídeo do canal selecionado; |

| | |
|---|---|
| | 2a2. Sistema exibe mensagem de erro; |

| Caso de Uso | Parar vídeo |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário pare a exibição do canal selecionado |
| Pré-condições | O usuário já está visualizando um canal selecionado |
| Pós-condições | O sistema termina a exibição do canal selecionado |
| Fluxo Básico | 1. Usuário clica no botão de parar vídeo;<br>2. Sistema termina o aplicativo em execução;<br>3. Sistema termina a exibição de vídeo do canal selecionado; |

| Caso de Uso | Terminar aplicativo |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário termine a execução de um aplicativo interativo |
| Pré-condições | O usuário já está executando um aplicativo interativo |
| Pós-condições | O sistema termina o aplicativo interativo em execução |
| Fluxo Básico | 1. Usuário clica no botão de terminar aplicativo;<br>2. Sistema esconde o aplicativo em execução;<br>3. Sistema para a execução do aplicativo; |

| Caso de Uso | Esconder aplicativo |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário esconda ou mostre um aplicativo interativo |
| Pré-condições | O usuário já está executando um aplicativo interativo |

| | |
|---|---|
| Pós-condições | Se o aplicativo interativo estiver visível o sistema esconde o mesmo, senão, exibe |
| Fluxo Básico | 1. Usuário clica no botão de esconder aplicativo;<br>2. Sistema inverte a visibilidade do aplicativo em execução; |

| Caso de Uso | Escolher aplicativo |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário inicie a execução de um aplicativo interativo |
| Pré-condições | O usuário já está visualizando um canal selecionado |
| Pós-condições | O sistema inicia a execução do aplicativo interativo |
| Fluxo Básico | 1. Usuário clica no botão de escolher aplicativo;<br>2. Sistema carrega a lista de aplicativos interativos disponíveis para o canal selecionado;<br>3. Sistema exibe a lista de aplicativos interativos;<br>4. Usuário escolhe um aplicativo interativo;<br>5. Sistema inicia a execução do aplicativo interativo; |

## 5.2 ARQUITETURA E PROJETO DE SOFTWARE

No diagrama de blocos ilustrado pela Figura 20 está concepção da arquitetura proposta para o aplicativo cliente, onde foram identificados os seguintes agentes:

1. **Aplicativo:** É o ponto de entrada do aplicativo cliente, que é executado pelo Android com objetivo de iniciar a aplicação;
2. **Preferências:** O Android possui nativamente componentes para controle de preferências de aplicativos, a tela de preferências é gerada

automaticamente pelo sistema a partir de um arquivo XML de configuração, e o armazenamento destas preferências fica a cargo do próprio sistema operacional;

3. **Controlador:** é o agente principal de todo o aplicativo cliente, é responsável por controlar a reação do sistema quando o usuário interage com a tela do aplicativo, e portanto tem acesso a todos os agentes do aplicativo;
4. **Gerenciador de sessão:** é responsável por controlar o acesso e manter centralizada toda informação em uso pelo aplicativo, mantendo em memória a lista de canais, de aplicativos e uma copia das preferências persistidas pelo sistema operacional;
5. **Wrappers:** este agente é um encapsulamento das classes compartilhadas (seguindo o padrão de *Data Transfer Object*) entre servidor de catálogo e aplicativo cliente, este agente também é capaz de se conectar ao servidor de catálogo a partir das informações fornecidas pelo agente Gerenciador de sessão e recuperar listas de canais e de aplicativos (portanto realizando também a função de *Data Access Object*) utilizando as definições de formato e protocolo detalhadas no capítulo 3;
6. **Controlador de interatividade:** este agente controlador tem por responsabilidade intermediar a comunicação do *web browser* na superfície interativa na tela do aplicativo cliente com o aplicativo interativo em um servidor externo;
7. **Views:** este agente é subdivido em três outros agentes que são diretamente responsáveis pelo conteúdo exibido ao usuário da aplicação, o agente Menu é responsável pela criação dos botões em tela para escolha de canal, aplicativo, e demais comandos disponíveis ao usuário, o agente de Superfície de *stream* é responsável por determinar a área de execução do vídeo e pela comunicação com o servidor de *stream* utilizando os componentes nativos do sistema operacional, por fim o agente de Superfície de interatividade é responsável pela área que será disponibilizado para interação do usuário com o aplicativo de interatividade, e de repassar os comandos enviados via *JavaScript Interface* (JSI) para o agente Controlador de interatividade.

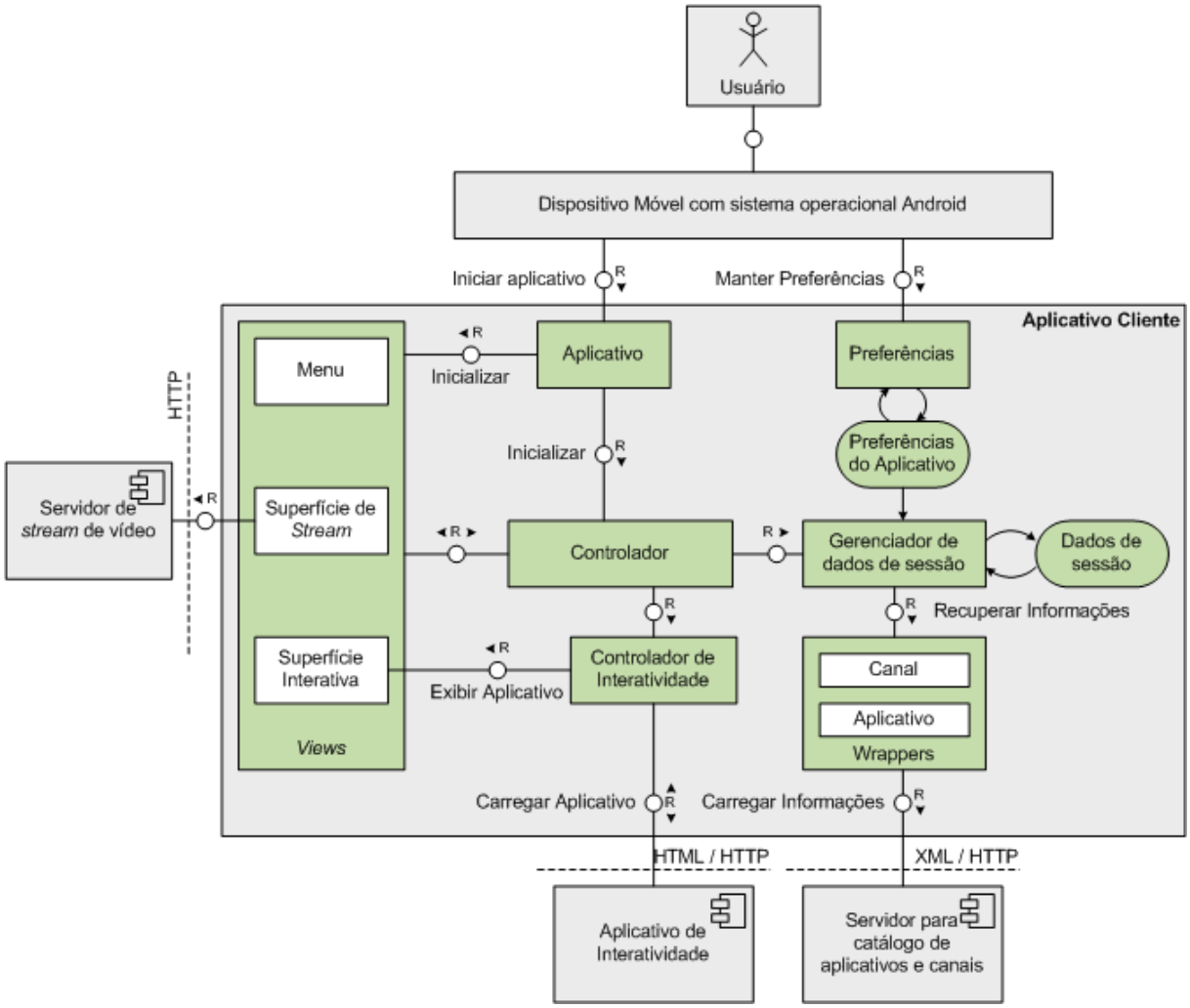


FIGURA 20 - ARQUITETURA PROPOSTA PARA O APLICATIVO CLIENTE

Conforme descrito no capítulo 3.6 a troca de mensagens entre aplicativo cliente e aplicativo interativo se dará através da chamada de métodos *JavaScript* que devem ser implementados pelo aplicativo interativo para receber informações sobre o dispositivo e vídeo em execução.

O aplicativo cliente irá invocar um método chamado *MobiApp_Init* no inicio da execução do aplicativo interativo, enquanto o método *MobiApp_Finish* será chamado quando o mesmo for fechado pelo aplicativo cliente, em ambos os casos o aplicativo cliente deverá enviar como parâmetro um objeto serializado em JSON contendo o atual status do dispositivo em questão, e o vídeo em exibição. Para a

prova de conceito foi utilizado um objeto contendo as informações conforme ilustrado na figura a seguir.

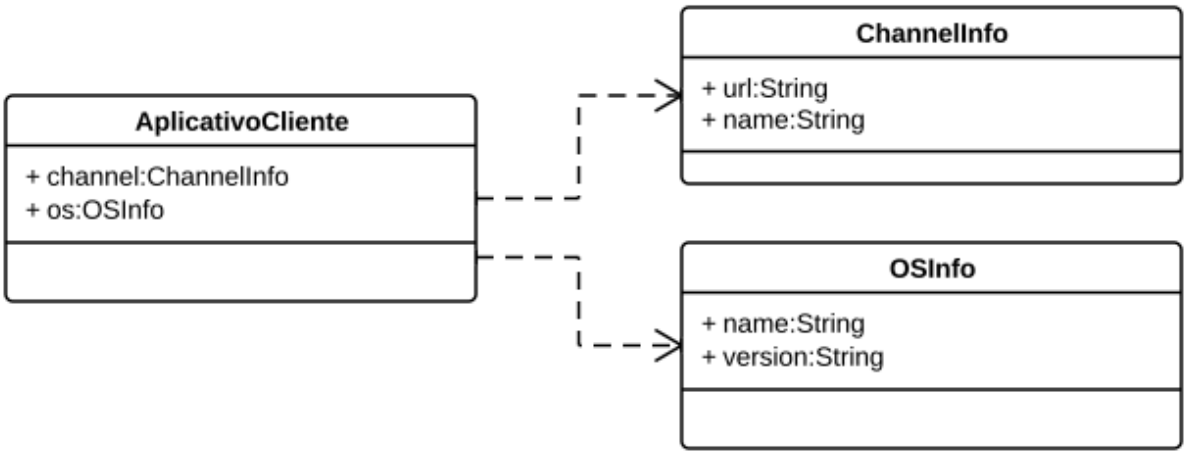


FIGURA 21 - DIAGRAMA DE CLASSES DA MENSAGEM PARA APLICATIVO INTERATIVO

Abaixo está ilustrado o objeto já serializado com um exemplo de mensagem enviada pelo aplicativo cliente para o aplicativo de interatividade, e devido a flexibilidade do JSON, este objeto de mensagem pode ser ampliado sem muito esforço de desenvolvimento a partir do aplicativo cliente e sem gerar erros ou inconsistências no aplicativo interativo, uma vez que o objeto serializado em JSON é interpretado em tempo de execução pelo *JavaScript*.

```
{
     "channel":
     {
          "url": "http://devimages.apple.com/…/bipbopall.m3u8",
          "name": "Apple Test"
     },
     "OS":
     {
          "name": "Android",
          "version": "10"
     }
 }
```

# 6 APLICATIVOS DE INTERATIVIDADE

# 6 APLICATIVOS DE INTERATIVIDADE

Com o objetivo de testar a viabilidade da criação de aplicativos interativos para a execução sobre o vídeo foram definidos dois aplicativos semelhantes mas com objetivos distintos.

O primeiro aplicativo interativo tem como objetivo disponibilizar uma sala de bate papo sobre o aplicativo de vídeo para, por exemplo, discutir com outros telespectadores o conteúdo do vídeo, enquanto o segundo aplicativo busca disponibilizar uma maneira de telespectadores enviarem questões a um moderador, permitindo uma interação a distância do telespectador com uma palestra ou workshop.

Recebendo informações do aplicativo cliente do dispositivo móvel via JavaScript (cuja interface proposta está detalhada no capítulo de arquitetura e projeto do aplicativo cliente) cada aplicativo interativo pode tomar decisões de como tratar uma conexão específica.

Em termos de tecnologia, qualquer aplicativo *web* que rode em um *web browser* pode ser executado sobre o vídeo, desde que não tenha nenhuma cor de fundo de página definida.

Nos aplicativos criados para a prova de conceito foi utilizada a biblioteca JQuery para controle do visual das páginas e Java com o Google App Engine SDK como tecnologia de servidor para permitir a hospedagem gratuita na nuvem.

O tamanho e layout de ambos aplicativos foi minimalista, utilizando um pequeno espaço em tela devido ao tamanho reduzido dos dispositivos móveis para o qual foram projetados.

## 6.1 CHAT

Como funcionalidade básica para criação de uma sala de bate papo, é possível identificar os seguintes requisitos funcionais:

1. O usuário deve poder escolher seu apelido e a sala em que deseja conectar;

2. O usuário deve poder enviar e receber mensagens para os outros usuários conectados na mesma sala;

A Figura 22 traduz os requisitos funcionais apresentados em um casos de uso, seguido pelo detalhamento de cada caso de uso.

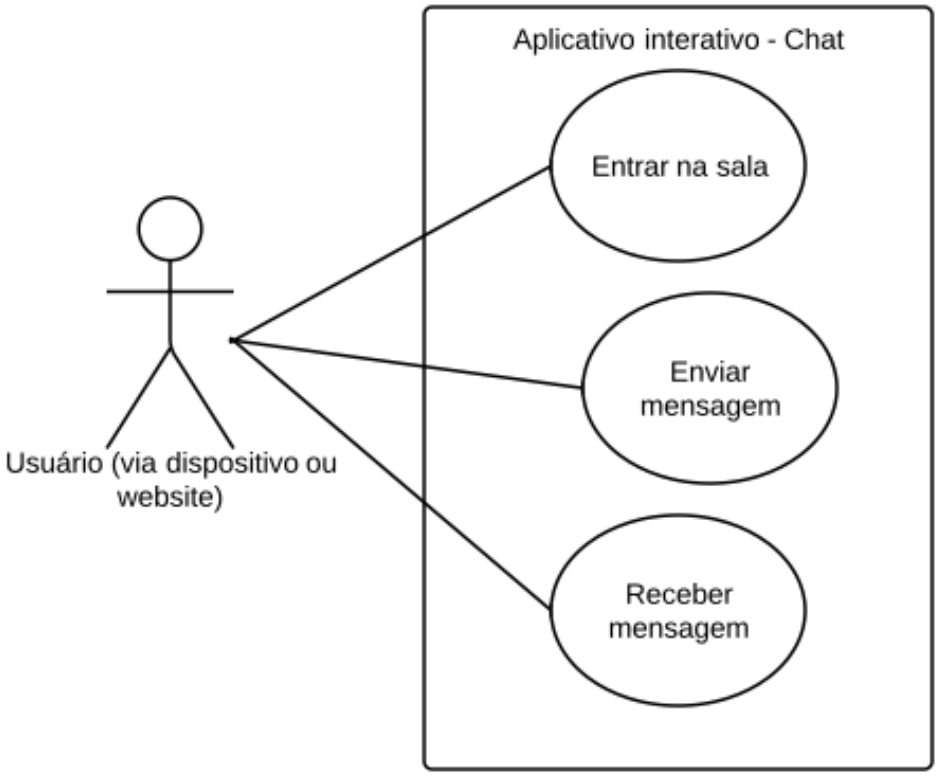


FIGURA 22 - CASOS DE USO DO APLICATIVO INTERATIVO DE CHAT

| Caso de Uso | Entrar na sala |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário acesse uma sala de bate papo |
| Pré-condições | Nenhuma |
| Pós-condições | O sistema registra a entrada do usuário na sala e exibe a tela com as conversas da sala; |
| Fluxo Básico | 1. Usuário entra com o nome da sala e o apelido desejado;<br>2. Sistema registra a entrada do usuário na sala; |

| | |
|---|---|
| | 3. Sistema exibe a tela com a sala de bate papo; |

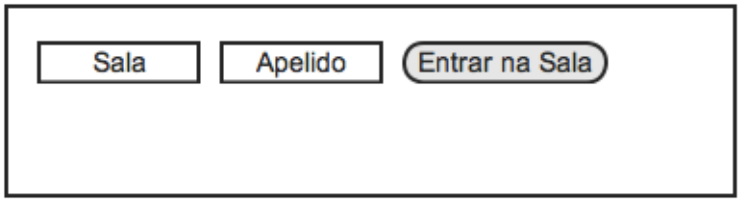


FIGURA 23 - PROTÓTIPO DE TELA PARA O CASO DE USO DE ENTRAR NA SALA DO APLICATIVO DE CHAT

| Caso de Uso | Enviar mensagem |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário envie mensagens para sala de bate papo |
| Pré-condições | O usuário já está registrado em uma sala de bate papo |
| Pós-condições | O sistema registra a mensagem enviada pelo usuário |
| Fluxo Básico | 1. Usuário digita a mensagem;<br>2. Usuário clica em enviar mensagem;<br>3. Sistema registra a mensagem e a hora em que a mensagem foi recebida;<br>4. Sistema retorna para o usuário a mensagem formatada; |

| Caso de Uso | Receber mensagem |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário receba novas mensagens enviadas por outros usuários |
| Pré-condições | O usuário já está registrado em uma sala de bate papo |
| Pós-condições | O sistema exibe todas as novas mensagens da tela |

| | |
|---|---|
| Fluxo Básico | 1. Usuário está visualizando a sala;<br>2. Sistema verifica se existe novas mensagens;<br>3. Sistema exibe as novas mensagens; |

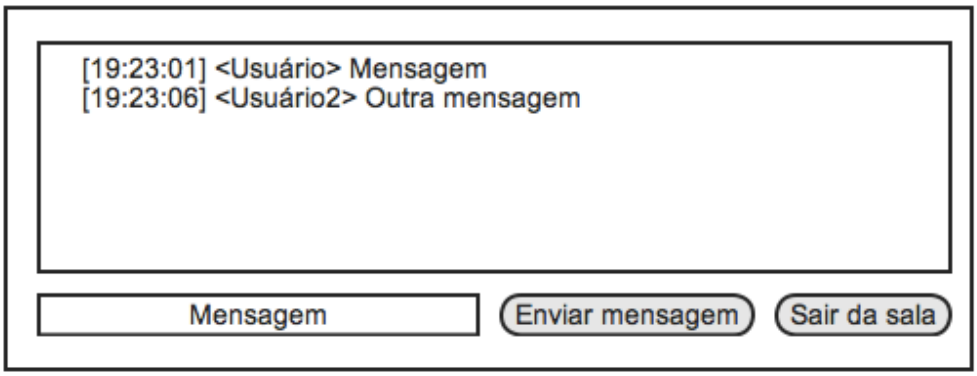


FIGURA 24 - PROTÓTIPO DA TELA PRINCIPAL DO APLICATIVO DE CHAT

## 6.2 WORKSHOP

O principal objetivo do aplicativo de workshop é permitir que usuários visualizando um evento remotamente em seus respectivos dispositivos móveis possam interagir com o evento a realizando perguntas no aplicativo, que será responsável por repassar as perguntas ao moderador do evento.

O aplicativo é composto por duas simples funcionalidades:

1. O usuário em um dispositivo móvel deve poder enviar questões para um determinado evento;
2. O moderador utilizando um web browser deve poder visualizar a lista de questões enviadas para o evento.

Os protótipos de tela e o detalhamento de casos de uso estão definidos nas figuras abaixo.

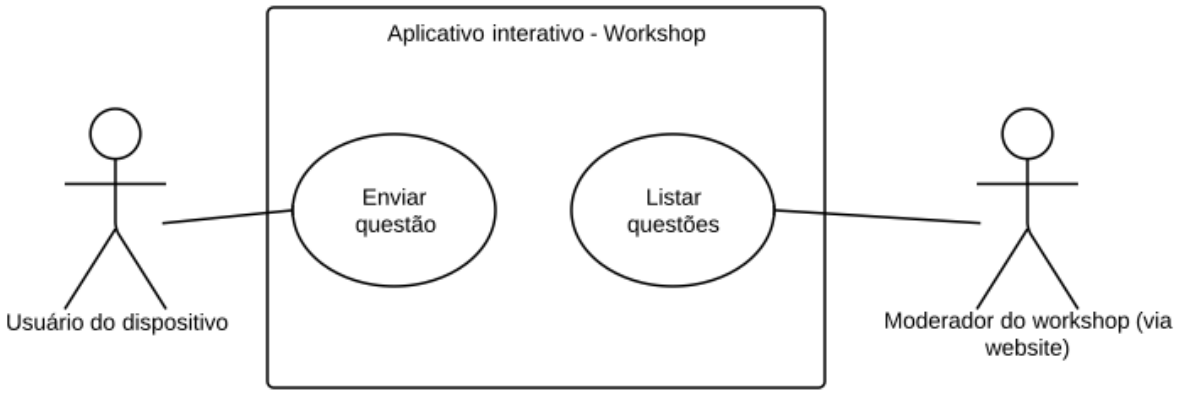


FIGURA 25 - CASOS DE USO DO APLICATIVO INTERATIVO DE WORKSHOP

| Caso de Uso | Enviar questão |
|---|---|
| Atores | Usuário do dispositivo |
| Objetivo | Permitir que o usuário do dispositivo interaja com um evento a partir de questões |
| Pré-condições | Nenhuma |
| Pós-condições | O sistema registra a questão enviada |
| Fluxo Básico | 1. Usuário acessa o aplicativo de workshop;<br>2. Usuário escreve a pergunta e clica no botão de envio;<br>3. Sistema registra a pergunta e o horário;<br>4. Sistema exibe mensagem de pergunta enviada com sucesso ao usuário; |

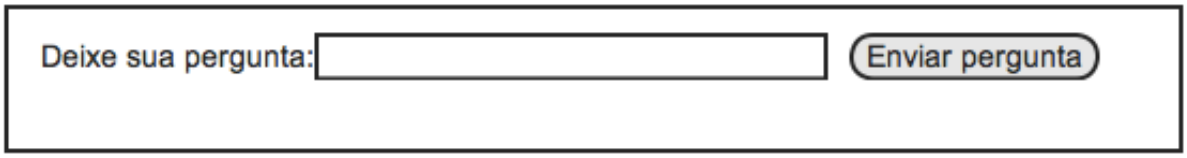


FIGURA 26 - PROTÓTIPO PARA CASO DE USO DE ENVIAR QUESTÃO DO APLICATIVO INTERATIVO DE WORKSHOP

| Caso de Uso | Listar questões |
|---|---|
| Atores | Moderador do workshop |
| Objetivo | Permitir que o moderador acesse a lista de questões realizadas |
| Pré-condições | Nenhuma |
| Pós-condições | O sistema exibe a lista de questões feitas por usuários dos dispositivos móveis |
| Fluxo Básico | 1. Moderador acessa endereço de listagem de questões do workshop;<br>2. Sistema exibe a lista de questões realizadas para o workshop; |

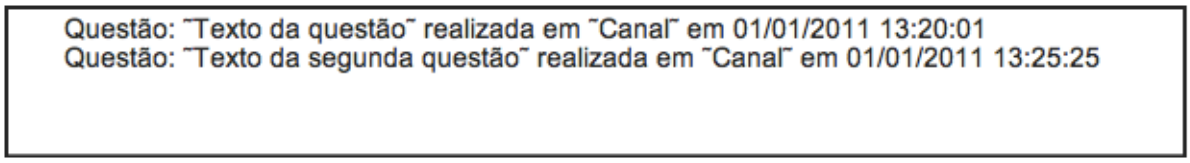


FIGURA 27 - PROTÓTIPO PARA CASO DE USO DE LISTAR QUESTÕES DO APLICATIVO INTERATIVO DE WORKSHOP

# 7 RESULTADOS

# 7 RESULTADOS

Nas figuras abaixo estão ilustrados os resultados obtidos no desenvolvimento baseado nos requisitos, arquitetura e projeto detalhados nos capítulos anteriores.

O servidor de catálogo para aplicativos e serviços foi desenvolvido para suportar tanto a execução na nuvem (hospedado no Google App Engine, conforme ilustrado nas figuras abaixo) quanto execução "*stand alone*", direto pelo arquivo JAR.

Para permitir essa abordagem, foi utilizado o padrão de projeto de Abstract Factory para determinar que tipo de persistência deve ser utilizada (na nuvem ou em arquivos), dependendo do modo de execução.

Como interface com o usuário administrador foi escolhido utilizar a biblioteca de JavaScript JQuery para controlar tanto os componentes da tela quanto as requisição o AJAX para o servidor.

A figura abaixo ilustra o cadastro de dois canais no servidor de catálogo, um apontando para um vídeo estático e outro para um *stream* de vídeo para teste de protocolo.

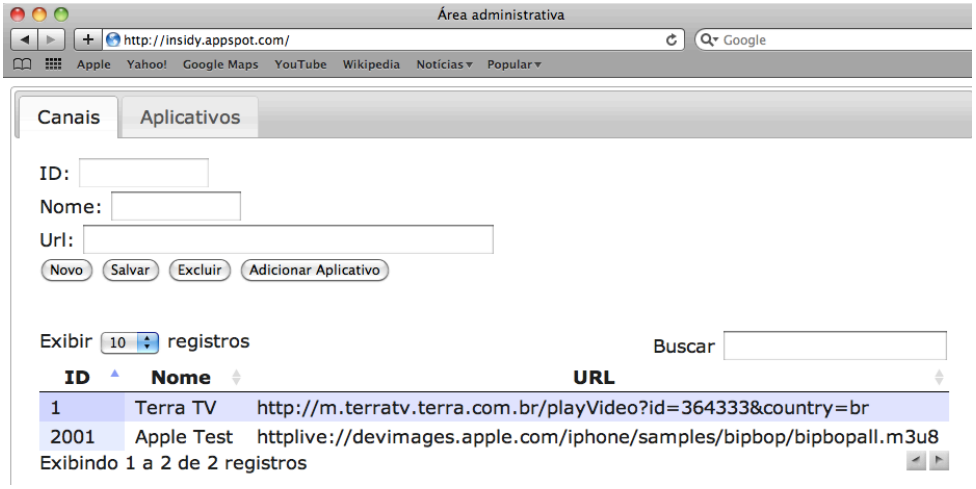


FIGURA 28 - IMPLEMENTAÇÃO DO INTERFACE ADMINISTRATIVA DE CANAIS

A figura abaixo ilustra o cadastro de dois aplicativos interativos disponíveis na nuvem, ambos definidos no capítulo 6.

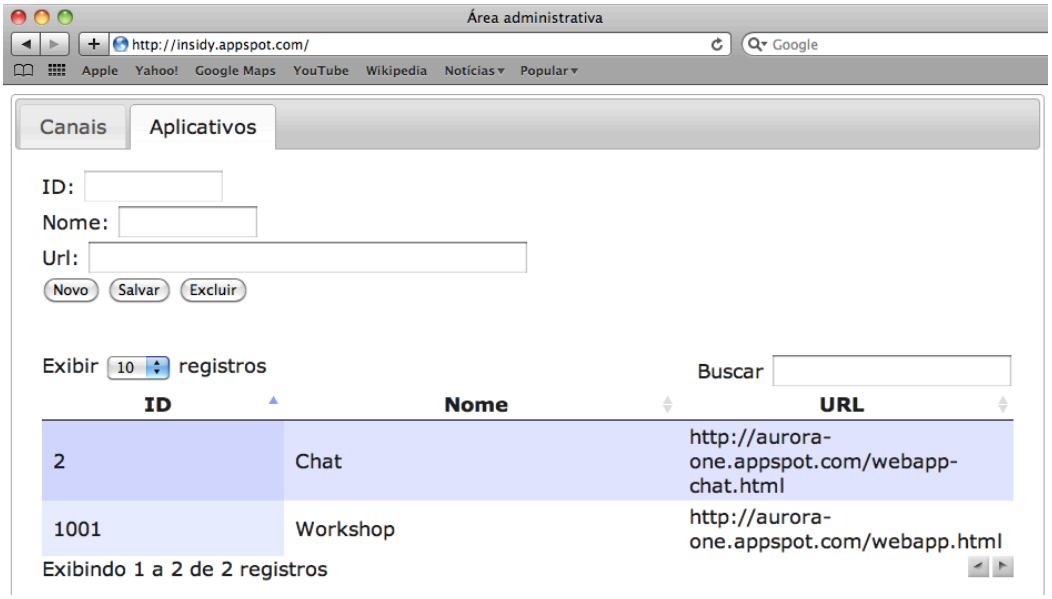


FIGURA 29 - IMPLEMENTAÇÃO DA INTERFACE ADMINISTRATIVA DE APLICATIVOS

A Figura 30 ilustra a seleção de quais aplicativos estarão disponíveis para o canal selecionado, na situação abaixo foi escolhido o aplicativo de Workshop.

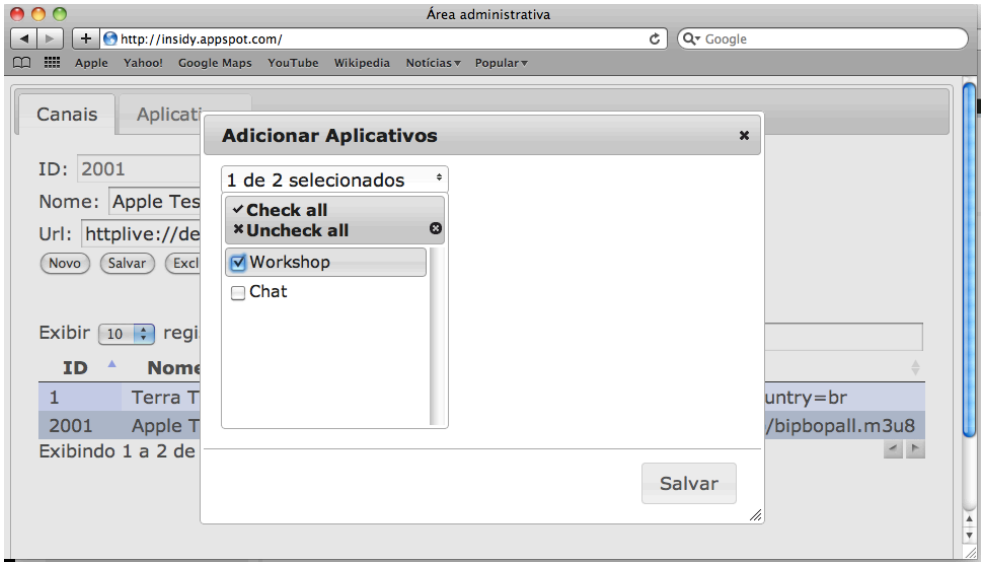


FIGURA 30 - IMPLENTAÇÃO DA INTERFACE DE CONEXÃO DE APLICATIVOS E CANAIS

A figura abaixo ilustra o retorno serializado em XML da listra de canais disponíveis no servidor de catálogo.

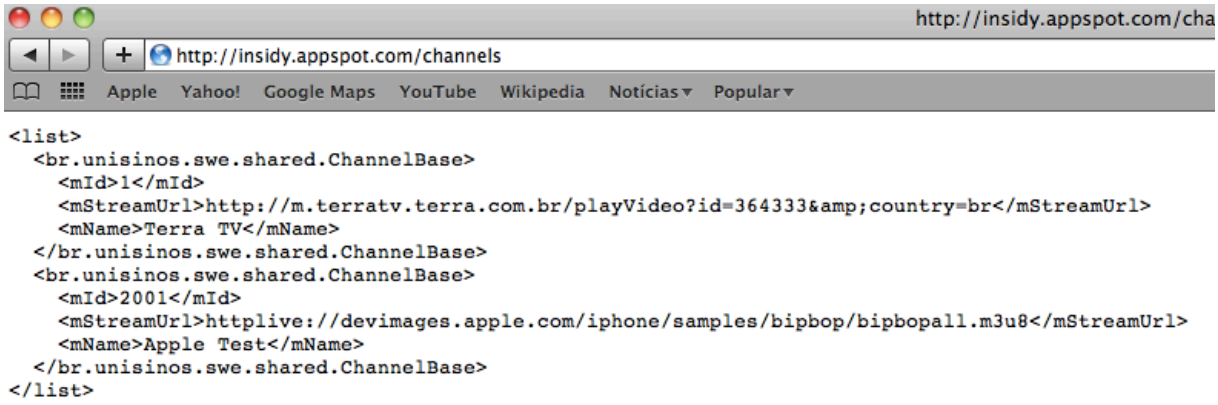


FIGURA 31 - RESULTADO DA REQUISIÇÃO DE LISTA DE CANAIS

A figura abaixo ilustra o retorno serializado em XML da listra de aplicativos interativos disponíveis para o canal de código “2001”, e conforme ilustrado pelo retorno acima, representa o canal “Apple Test”.

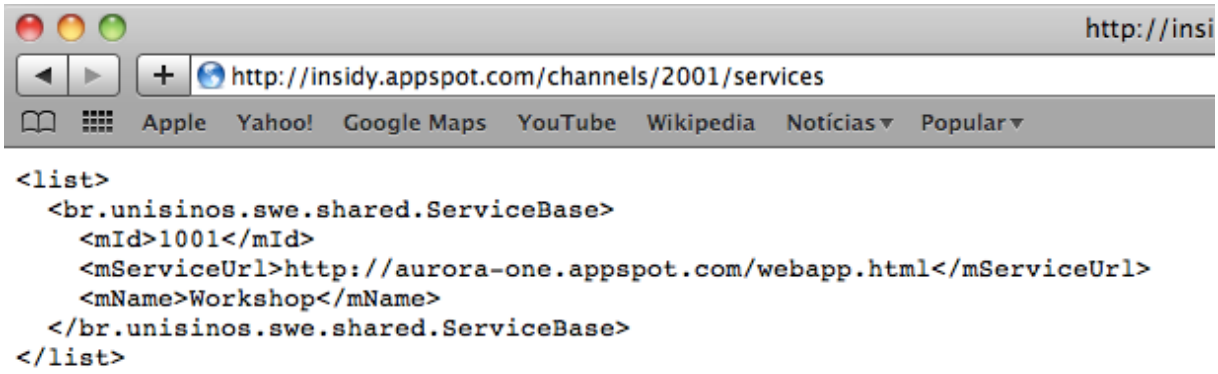


FIGURA 32 - RESULTADO DA REQUISIÇÃO DE LISTA DE APLICATIVOS

Para o teste da aplicação móvel foi utilizado como dispositivo físico o Motorola Dext, atualizado para versão 2.3.3 do Android. O Motorola Dext é um dispositivo com clock de 528MHz e 256 MiB de memória RAM, que vem de fábrica com a versão 1.5 do Android. Durante os testes foi percebido um tempo de resposta muito baixo quando executando a versão 2.3.3 do Android rodando o aplicativo cliente. A performance foi desconsiderada para a avaliação de usabilidade, uma vez que os dispositivos mais recentes (como o Motorola Atrix, ou o LG Optimus 2X) dotam de processadores com dois núcleos e GPU (*graphical processing unit*) dedicada para jogos e vídeos em alta definição.

As figuras abaixo ilustram o resultado final da implementação da interface definida no capítulo 5 no dispositivo Android.

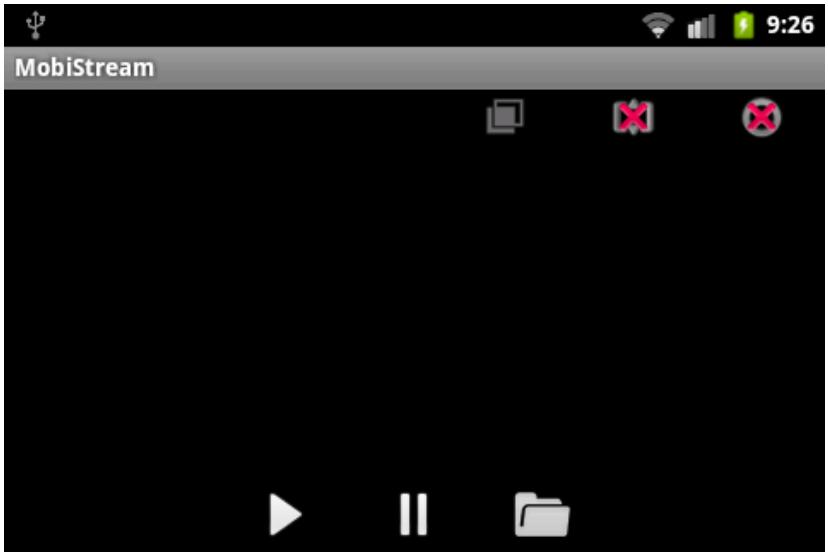


FIGURA 33 - IMPLEMENTAÇÃO DA TELA INICIAL DO DISPOSITIVO MÓVEL

A figura abaixo ilustra a alteração das preferências para apontar ao servidor disponibilizado na nuvem do Google App Engine.

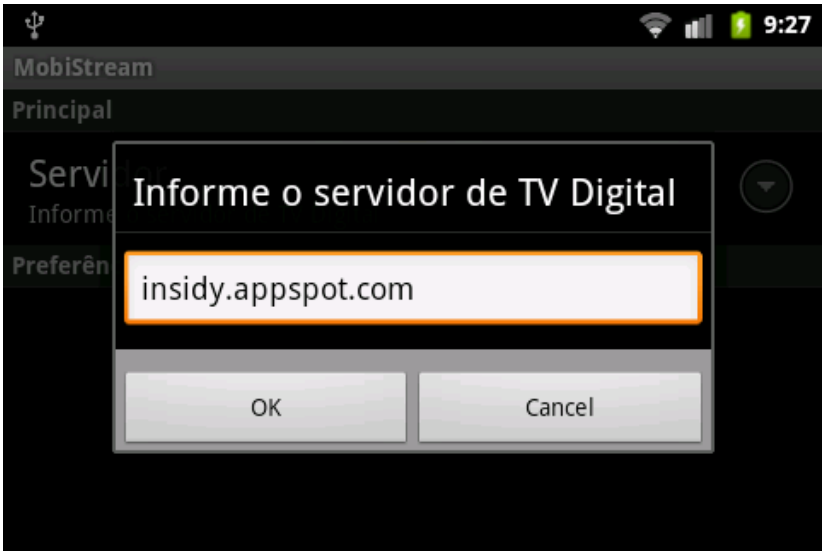


FIGURA 34 - IMPLEMENTAÇÃO DAS CONFIGURAÇÕES DO APLICATIVO

A Figura 35 ilustra o retorno com a lista de canais disponíveis no servidor de catálogo conectando diretamente na nuvem.

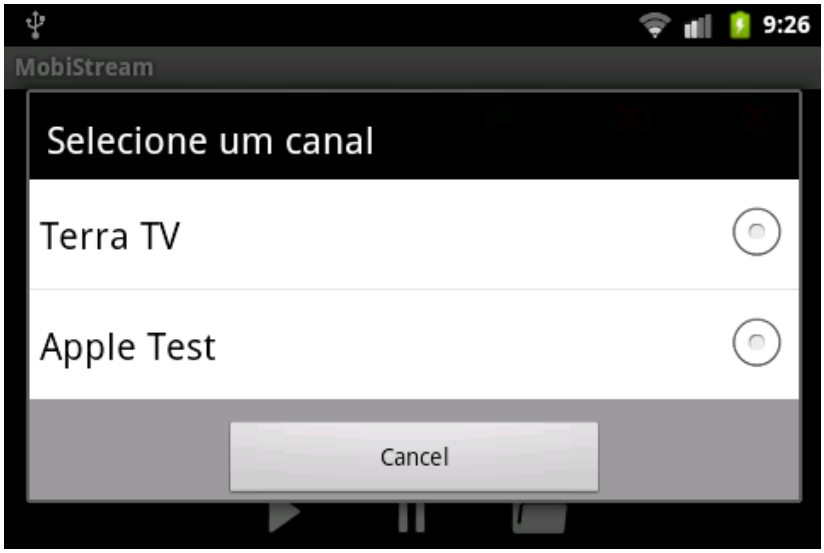


FIGURA 35 - IMPLEMENTAÇÃO DA SELEÇÃO DE CANAIS

A figura abaixo ilustra o resultado da seleção do canal “Apple Test“, cujo retorno é um vídeo para teste do protocolo *HTTP Live Stream*.

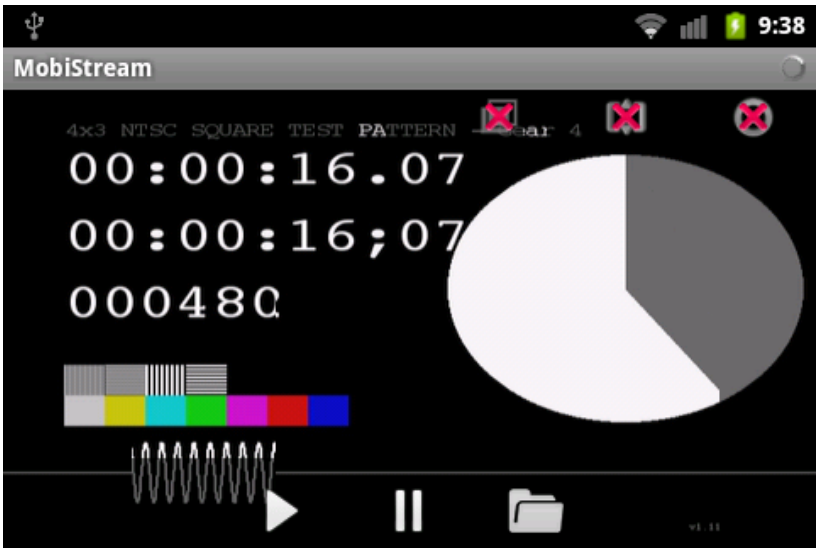


FIGURA 36 - EXIBIÇÃO DE VIDEO HTTP LIVE STREAMING

A figura abaixo ilustra o resultado a lista de seleção de aplicativos durante a seleção do aplicativo interativo de Chat.

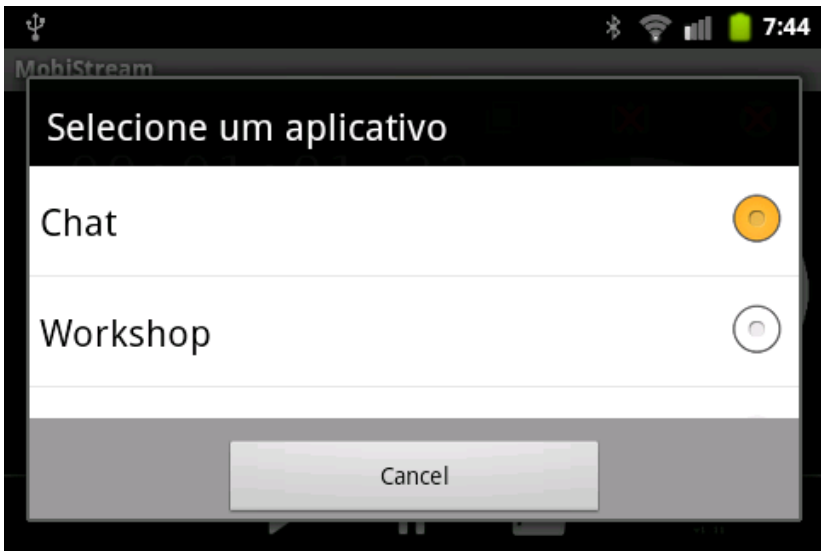


FIGURA 37 - IMPLEMENTAÇÃO DA SELEÇÃO DE APLICATIVO

A figura abaixo ilustra o aplicativo interativo de chat rodando sobre o vídeo selecionado, já em uma sala para o vídeo em questão.


FIGURA 38 - APLICATIVO DE CHAT RODANDO SOBRE O VÍDEO

A figura abaixo ilustra o aplicativo interativo de workshop rodando sobre o vídeo selecionado, utilizando a interface disponibilizada pelo aplicativo cliente para enviar as perguntas de cada workshop para o moderador do evento.


FIGURA 39 – APLICATIVO DE WORKSHOP RODANDO SOBRE O VÍDEO

# 8 CONCLUSÕES

# 8 CONCLUSÕES

Conforme objetivos identificado no capítulo de introdução, este trabalho buscou determinar a viabilidade de construção de aplicativos interativos multi-plataforma para streaming de vídeo em dispositivos móveis.

Baseado nos resultados obtidos pode-se concluir que:

1. a implementação de uma arquitetura para unificar vídeos e aplicações interativas em dispositivos móveis é viável;
2. dispositivos de nova geração tem capacidade de processamento suficiente para executar as atividades de interatividade e de vídeo em paralelo;
3. é possível fazer uso de diferentes padrões para permitir uma pequena curva de aprendizado e uma liberdade no desenvolvimento dos aplicativos interativos;
4. é possível aproveitar a própria evolução dos diferentes sistemas operacionais e dos diferentes padrões utilizados para ampliar as capacidades da arquitetura de unificação com pouco esforço de manutenção.

## 8.1 TRABALHOS FUTUROS

Devido ao tempo restrito para a execução e término deste trabalho, muitas ideias para ampliação do servidor de catálogo e para interface entre aplicativo interativo e aplicativo cliente acabaram adiadas. Para a continuidade para este trabalho e aumentar a aplicabilidade do mesmo, é possível enumerar as seguintes extensões:

1. a implementação nativa de um servidor de streaming baseado em HTTP Live Streaming, garantindo a homogeneidade do protocolo de streaming, e a capacidade nativa do catálogo de transmitir vídeos a partir da nuvem, sem a necessidade de um outro servidor;

2. a implementação de interfaces para auto-registro para servidores de streaming e aplicativos interativos, permitindo ao administrador do servidor de catálogo aceitar ou não a inscrição;
3. portar o aplicativo cliente para outros sistemas operacionais (iOS e Windows Phone 7 se considerar as prospecções da Gartner e o público-alvo);

## REFERÊNCIAS


WEILKIENS, T.; OESTEREICH, B. **UML 2 Certification Guide:** Fundamental and Intermediate Exams. San Francisco: Morgan Kaufmann Publishers, 2007. 317 p.

APPLE. HTTP Live Streaming. **Internet Engineering Task Force**, 2011. Disponivel em: <http://tools.ietf.org/html/draft-pantos-http-live-streaming-06>. Acesso em: 5 maio 2011.

CASSOL, L. P. et al. Cenários prospectivos para telefonia celular no Brasil: 2008-2016. **Revista Gestão & Regionalidade**, v. 24, p. 25, 2008.

CROCKFORD, D. The application/json Media Type for JavaScript Object Notation (JSON). **Internet Engineering Task Force**, 2006. Disponivel em: <http://tools.ietf.org/html/rfc4627>. Acesso em: 05 maio 2011.

DOROKHOVA, R.; AMELICHEV, N. **Comparison of Modern Mobile Platform from the Developer Standpoint**. St. Petersburg Electrotechnical University. St. Petersburg, p. 10. 2010.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. University of California. [S.l.]. 2000.

GANNES, L. The Lowdown on Apple's HTTP Adaptive Bitrate Streaming. **GIGAom**, 2009. Disponivel em: <http://gigaom.com/video/the-lowdown-on-apples-http-adaptive-bitrate-streaming/>. Acesso em: 03 maio 2011.

GOASDUFF, L.; PETTEY, C. Gartner press release on worldwide mobile devices sales. **Gartner**, 2010. Disponivel em: <http://www.gartner.com/it/page.jsp?id=1421013>. Acesso em: 03 maio 2011.

MINISTÉRIO DAS COMUNICAÇÕES. **Portaria 652 de 10 de outubro de 2006**. Ministério das Comunicações. [S.l.]. 2006.

MINISTÉRIO DAS COMUNICAÇÕES. Sistema Latino-americano de TVDigital vira realidade. **Ministério das Comunicações**, 2009. Disponivel em: <http://goo.gl/mgTk>. Acesso em: 03 maio 2011.

OPEN HANDSET ALLIANCE. Alliance members. **Open Handset Alliance**, 2007. Disponivel em: <http://www.openhandsetalliance.com/oha_members.html>. Acesso em: 03 maio 2011.

PETTEY , C.; STEVENS , H. Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012. **Gartner**, 2011. Disponivel em: <http://www.gartner.com/it/page.jsp?id=1622614>. Acesso em: 03 maio 2011.

PETTEY, C.; TUDOR, B. Gartner Says Android to Become No. 2 Worldwide Mobile Operating System in 2010 and Challenge Symbian for No. 1 Position by 2014. **Gartner**, 2010. Disponivel em: <http://www.gartner.com/it/page.jsp?id=1434613>. Acesso em: 03 maio 2011.

SAHA, A. K. A Developer's First Look At Android. **Linux for you**, p. 48-50, 2008.

SAP. **Standardized Technical Architecture Modeling: Conceptual and Design Level**. [S.l.]. 2007.

SCHULZRINNE, H.; RAO, A.; LANPHIER, R. **RFC 2326: Real Time Streaming Protocol (RTSP)**. Internet Engineering Task Force. [S.l.]. 1998.

ROSENBERG, D.; STEPHENS, M. **Use case driven object modeling with uml:** theory and practice. [S.l.]: Apress, 2007. 473 p.