Capítulo 4

TV 3.0: Especificações da Camada de Codificação de Aplicações

Marcelo F. Moreno[1], Débora Muchaluat-Saade[2], Guido Lemos[3], Sérgio Colcher[4], Carlos Soares Neto e Li-Chang Shuen C. S. Sousa[5], Joel dos Santos[6]

[1]Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF)
Juiz de Fora – MG – Brasil

[2]Instituto de Computacão
Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

[3]Centro de Informática
Universidade Federal da Paraíba (UFPB)
João Pessoa – PB – Brasil

[4]Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro – RJ – Brasil

[5]Departamento de Informática
Universidade Federal do Maranhão (UFMA)
São Luís – MA – Brasil

[6]Escola de Informática e Computação
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ)
Rio de Janeiro – RJ – Brasil

marcelo.moreno@ufjf.br, debora@midiacom.uff.br, guido@lavid.ufpb.br,
colcher@inf.puc-rio.br, carlos.salles@ufma.br, li.chang@ufma.br,
jsantos@eic.cefet-rj.br

Abstract

This chapter presents the technical specifications resulting from R&D and standardization efforts in the application coding layer of the next generation of the Brazilian Digital Terrestrial Television System (SBTVD), known as TV 3.0 or DTV+. The architecture evolves

the Ginga middleware into an application-oriented platform, specifying APIs and definitions that enable personalized experiences, integration with OTT services, new forms of interaction and immersion, enhanced accessibility, and audience data collection with privacy safeguards. The initiative is the result of collaboration among the SBTVD Forum, its associated companies, the Ministry of Communications, the National Education and Research Network (RNP), and Brazilian universities. The research team includes several members of the WebMedia Community, who contributed to the design, validation, and standardization of this core layer of TV 3.0.

Resumo

Este capítulo apresenta as especificações técnicas resultantes de esforços de P&D e padronização na codificação de aplicações para a próxima geração do Sistema Brasileiro de Televisão Digital Terrestre (SBTVD), denominada TV 3.0 ou DTV+. A arquitetura evolui o middleware Ginga para uma plataforma orientada a aplicativos, especificando APIs e definições que viabilizam experiências personalizadas, integração com serviços OTT, novas formas de interação e imersão, acessibilidade avançada e coleta de dados de audiência com garantia de privacidade. A proposta é fruto da colaboração entre o Fórum SBTVD, empresas associadas, o Ministério das Comunicações, a Rede Nacional de Ensino e Pesquisa e universidades brasileiras. A equipe de pesquisa inclui diversos membros da Comunidade WebMedia, que contribuíram para a concepção, validação e normatização dessa camada central da TV 3.0.

4.1. Introdução

O Brasil assiste à televisão desde 1950, com a inauguração da TV Tupi em São Paulo. Em sete décadas, a televisão brasileira passou por desenvolvimentos tecnológicos, sociais e de conteúdo que tornam a experiência nacional uma das mais ricas do mundo. Notavelmente, o Sistema Brasileiro de TV Digital Terrestre (SBTVD) atual especifica o *middleware* Ginga, conforme ABNT NBR 15606-2 (2023), uma tecnologia nacional, como o padrão para interatividade multimídia desde 2007. O Ginga provou suportar uma evolução consistente, tornando-se a primeira tecnologia integralmente brasileira adotada como padrão internacional ITU-T H.761 (2009) e reconhecida pela ITU-R como um sistema integrado *broadcast-broadband* ITU-R BT2075-1 (2017).

Desde 2020, está em curso a definição de uma próxima geração para o SBTVD, no âmbito do Projeto TV 3.0 do Fórum SBTVD (2020a), motivada pela mudança rápida nos hábitos de consumo de mídia das pessoas e nos possíveis avanços em novas experiências pela TV. Em sua Fase 3, encerrada em setembro de 2024, o projeto realizou testes e avaliações sobre as camadas físicas e de codificação de vídeo, bem como o desenvolvimento de um mux/demux de referência. No que diz respeito à camada de codificação de aplicações, a maioria dos requisitos inovadores estabelecidos pela Chamada de Propostas do Fórum SBTVD (2020b) foi submetida ao estudo por grupos de pesquisa selecionados, uma vez que esses requisitos não foram devidamente endereçados nas fases anteriores do Projeto TV 3.0, conforme reportado em Fórum SBTVD (2021). Tais requisitos avançados incluem suporte à experiência de TV baseada em aplicativos, conteúdo audiovisual imersivo, interação multimodal, efeitos sensoriais, identificação de múltiplos telespectadores, personalização da experiência e do conteúdo, coleta de dados de audiência, proteção a privacidade, convergência IP e extensibilidade, para citar apenas alguns deles.

Concomitante ao início da Fase 3 do Projeto TV 3.0, em abril de 2023, foi publicado o Decreto Presidencial nº 11.484, Brasil (2023), que estabelece as diretrizes para a evolução do Sistema Brasileiro de Televisão Digital Terrestre e para garantir a disponibilidade do espectro de radiofrequência para sua implementação. O decreto estabelece que o sistema de TV de próxima geração no Brasil deve ter as seguintes características:

1. qualidade audiovisual superior à do SBTVD de primeira geração;
2. recepção fixa, com antena externa e interna, e recepção móvel;
3. integração entre conteúdos transmitidos pelo serviço de radiodifusão e pela Internet;
4. interface de usuário baseada em aplicativos;
5. segmentação de conteúdo de acordo com a localização geográfica dos espectadores;
6. personalização de conteúdo de acordo com as preferências dos espectadores;
7. uso otimizado do espectro destinado à radiodifusão de TV terrestre; e
8. novas formas de acesso a conteúdos culturais, educacionais e informativos.

O decreto também estabelece que o Ministério das Comunicações (MCom) apoie o Fórum SBTVD nos estudos relacionados às inovações tecnológicas a compor a TV 3.0. A expectativa inicial era que os estudos fossem concluídos até 31 de dezembro de 2024.

O Projeto TV 3.0, impulsionado pelo decreto presidencial, desenvolveu novas especificações para o SBTVD. Essas especificações foram prototipadas, avaliadas e resultaram, em novembro de 2024, na aprovação preliminar do novo conjunto de normas da série ABNT NBR 25600 (a ser publicada), pela Comissão de Estudo Especial em TV Digital da ABNT (ABNT/CEE-85). De acordo com o planejamento do Fórum SBTVD, essas normas passarão por consulta pública em meados de 2025. Por isso, até o momento, tais especificações ainda são pouco conhecidas por pesquisadores e profissionais do setor que não participam diretamente das discussões conduzidas pelo Fórum SBTVD.

Algumas das tecnologias discutidas foram selecionadas a partir de propostas submetidas para avaliação durante as Fases 1 e 2 do Projeto TV 3.0 do Fórum SBTVD (2021). Já a Fase 3 do projeto estabeleceu um processo de pesquisa e desenvolvimento próprio, visando atender a requisitos que não foram plenamente satisfeitos nas fases anteriores (Moreno et al., 2023). Esse esforço contínuo (que, espera-se, se perpetuará) teve como objetivo assegurar que o SBTVD permaneça na vanguarda da inovação tecnológica, respondendo às crescentes demandas por qualidade, personalização, imersão, interatividade e acessibilidade no consumo de conteúdo televisivo no Brasil.

O desenvolvimento dessas novas tecnologias é resultado de uma colaboração entre o Fórum SBTVD, suas empresas associadas, o MCom, a Rede Nacional de Ensino e Pesquisa (RNP) e uma equipe multidisciplinar de pesquisadores de universidades brasileiras. Muitos deles são membros ativos da Comunidade WebMedia e, ao longo dos anos, têm divulgado avanços do Projeto TV 3.0 em edições do evento, tanto na Trilha Principal quanto no Workshop de Futuro da TV Digital Interativa (WTVDI).

Nesse contexto, este capítulo tem como principal objetivo apresentar e detalhar os resultados obtidos por meio do esforço de pesquisa, desenvolvimento e normatização especificamente da Camada de Codificação de Aplicações para a TV 3.0.

Ao final deste capítulo, espera-se que o leitor tenha adquirido uma compreensão

das principais especificações técnicas, fundamentos conceituais e interfaces de programação de aplicações (APIs) relacionados à Camada de Codificação de Aplicações da TV 3.0. Tal conhecimento constitui um pilar fundamental tanto para o aprofundamento acadêmico quanto para a atuação profissional no desenvolvimento de aplicações interativas.

O capítulo está organizado da seguinte maneira. A Seção 4.2 descreve uma visão geral da nova Camada de Codificação de Aplicações da TV 3.0 e apresenta o conceito da plataforma de TV orientada a aplicativos. A Seção 4.3 apresenta as novas APIs do Ginga-NCL. A Seção 4.4 apresenta as novas APIs do Ginga-HTML5. A Seção 4.5 apresenta as novas APIs do TV 3.0 WebServices. Finalmente, a Seção 4.6 conclui o capítulo apresentando perspectivas futuras.

4.2. TV 3.0 - Arquitetura da Plataforma de TV Orientada a Aplicativos

A Codificação de Aplicações da TV 3.0 evolui de uma especificação de *middleware* definida na ABNT NBR 15606 (todas as partes) para uma especificação de plataforma abrangente de software, a ser publicada como ABNT NBR 25608. Uma especificação de plataforma amplia o foco para incluir a infraestrutura de *back-end* e os componentes de *front-end*, criando um sistema mais coeso e integrado.

Essa abordagem de plataforma abrange funções como navegação e seleção de serviços de radiodifusão, gerenciamento de perfis de telespectador e agregação, busca e recuperação de conteúdo. Esses elementos são essenciais para fornecer uma experiência contínua e intuitiva ao telespectador. Ao incorporar essas funções em uma plataforma unificada, a TV 3.0 posiciona os aplicativos como elementos centrais na jornada dos telespectadores, capaz de habilitar a construção e uso de perfis desde o início, visando a melhor experiência personalizada.

Assim, com o avanço para uma plataforma orientada a aplicativos, a TV 3.0 permite que as emissoras ofereçam aos telespectadores uma experiência interativa, personalizada e dinâmica, na qual cada aplicativo pode oferecer conteúdos e funcionalidades específicas que vão além da transmissão audiovisual convencional.

A arquitetura da Camada de Codificação de Aplicações da TV 3.0 organiza-se em um conjunto de componentes interdependentes que suportam a execução, gestão e personalização de experiências multimídia interativas em dispositivos receptores. Conforme ilustrado na Figura 4.1, no centro dessa arquitetura encontra-se o Catálogo de Aplicativos (*Application Catalog*), responsável por apresentar ao telespectador a lista de *Bootstrap Applications* disponíveis, conforme anunciado pelos serviços de TV 3.0. Esse catálogo centraliza o acesso aos aplicativos relacionados à radiodifusão, permitindo identificar emissoras, explorar conteúdos, acessar guias de programação (EPG) e de conteúdo (ECG) e gerenciar perfis e preferências do usuário.

O Painel de Acesso (*Access Panel*), componente associado ao Catálogo de Aplicativos, fornece um mecanismo de acesso rápido a aplicativos sinalizados pelo atual serviço TV 3.0 e às funcionalidades essenciais da plataforma, como busca, guias e configurações de acessibilidade. Pode ser evocado pelo telespectador por meio do controle remoto, mesmo durante a execução de um aplicativo, sobrepondo-se à interface em exibição e adaptando-se dinamicamente ao contexto do conteúdo.

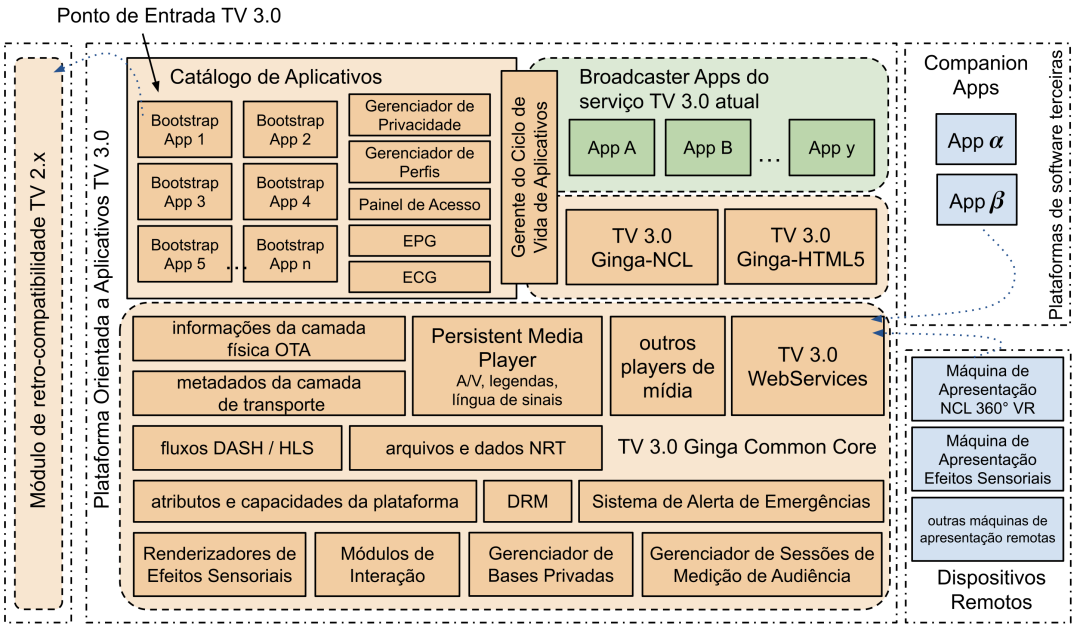

Figura 4.1: Arquitetura da Plataforma Orientada a Aplicativos da TV 3.0.

Os *Bootstrap Applications* são aplicativos iniciais que constituem os pontos de entrada para os serviços de radiodifusão, sendo instanciados automaticamente após a varredura de sinal e, quando acionados por meio do Catálogo de Aplicativos, são vinculados diretamente ao serviço linear correspondente, entregue por OTA (*over the air*) e/ou OTT (*over the top*). Elas são projetadas para desempenho rápido e simplicidade, para permitir que o mecanismo de troca rápida entre emissoras (zapeamento) continue sendo possível como parte da experiência de TV 3.0. Assim, cada *Bootstrap Application* oferece uma interface simples e estável para inicialização de um serviço TV 3.0, controlando o *Persistent Media Player* para apresentar o devido conteúdo OTA ou OTT. Sua robustez é fundamental para que, em caso de falhas em outros aplicativos, o *Bootstrap Application* do serviço atual possa ser reiniciado automaticamente para garantir a continuidade da experiência.

Os *Broadcaster Applications* são aplicativos sinalizados pelos radiodifusores que ampliam a funcionalidade básica oferecida pelos *Bootstrap Applications*, permitindo a entrega de experiências interativas avançadas. Esses aplicativos são sinalizados pelo sinal de radiodifusão e podem ser entregues tanto por OTA como OTT. São os *Broadcaster Applications* que viabilizam personalização, recomendações, publicidade segmentada e outros recursos que avançam na experiência do telespectador junto ao radiodifusor. A seleção ou alternância entre esses aplicativos pode ser feita pelo próprio telespectador por meio do Painel de Acesso ou definida automaticamente pela emissora com base em critérios contextuais de seus aplicativos e por códigos de controle na sinalização. Mantendo a compatibilidade com a TV 2.5, os aplicativos DTV+ podem ser desenvolvidos nas linguagens NCL, conforme a Norma ABNT NBR 15606-2 (2023), ou HTML5, conforme a Norma ABNT NBR 15606-10 (2023), considerando as novas APIs da TV 3.0, como será detalhado nas seções seguintes deste capítulo.

O Gerenciador de Ciclo de Vida de Aplicativos (*Application Lifecycle Manager*), implementa processos como recuperação, instalação, ativação, atualização, suspensão, encerramento e reinício de aplicativos. Ele também assegura transições suaves entre di-

ferentes aplicativos e implementa mecanismos de tolerância a falhas, como o reinício automático do *Bootstrap Application* correspondente ao serviço TV 3.0 ativo, em caso de mau funcionamento de algum *Broadcaster Application*.

Os *Broadcaster Applications* são desenvolvidos com base em linguagens e ambientes padronizados, suportados pelas máquinas de apresentação TV 3.0 Ginga-NCL e TV 3.0 Ginga-HTML5. Essas máquinas devem assegurar a conformidade com as APIs e comportamentos esperados, especificados na ABNT NBR 25608 (a ser publicada), por meio de testes de conformidade a serem definidos, de forma a promover portabilidade e interoperabilidade das aplicações entre diferentes dispositivos receptores.

O *TV 3.0 Ginga Common Core* agrega diversos componentes que proveem funcionalidades compartilhadas para os aplicativos, via APIs das máquinas de apresentação, e para o Catálogo de Aplicativos. Um desses componentes de destaque é o *Persistent Media Player*, que permite a reprodução contínua de conteúdo, independentemente do aplicativo atualmente ativo, viabilizando transições rápidas entre aplicativos e mantendo o desempenho e a consistência na apresentação de mídia entregue por OTA e OTT. Para isso, o ciclo de vida do *Persistent Media Player* não é vinculado a nenhum aplicativo específico, tal que o estado do *player* e sua fonte de mídia sejam mantidos ao alternar de um aplicativo para outro, preservando, assim, a continuidade da experiência. O aplicativo ativo pode modificar a fonte de mídia ou alterar o estado do *player*, em resposta à interação do telespectador ou automaticamente, dependendo da funcionalidade que implementa.

O componente TV 3.0 WebServices expõe APIs que permitem o acesso a dados e funcionalidades da plataforma por parte de *Broadcaster Applications* e por *Companion Applications*, ampliando a interoperabilidade com dispositivos do ambiente doméstico. Os *Companion Applications* são aplicativos executados em ambiente externo à TV 3.0, nativamente no próprio receptor ou em dispositivos remotos — como *smartphones* — que se integram à plataforma por meio do TV 3.0 WebServices, mediante autorização do telespectador e da emissora, promovendo experiências expandidas e interativas.

Por fim, o Módulo de retro-compatibilidade TV 2.X assegura a disponibilização de serviços, conteúdos e aplicativos legados, conforme o conjunto normativo do SBTVD da geração atual, promovendo continuidade tecnológica e transição suave dos serviços.

4.2.1. Catálogo de Aplicativos

O Catálogo de Aplicativos constitui a interface primária de acesso aos serviços de radiodifusão na TV 3.0, funcionando como ponto central para descoberta, organização e inicialização dos *Bootstrap Applications* associados aos serviços de TV 3.0. Ele é acessado por meio de um ícone padronizado na tela inicial dos receptores, identificado pela marca DTV+, criada pelo Fórum SBTVD e ilustrada na Figura 4.2.

Figura 4.2: Marca DTV+ criada pelo Fórum SBTVD. Processo INPI 935790616.

Esse ícone restabelece a proeminência da TV aberta nos dispositivos conectados, superando uma limitação recorrente observada nos sistemas operacionais de *smart TVs*, nos quais os conteúdos de radiodifusão são frequentemente relegados a áreas de difícil acesso, baixa visibilidade e/ou misturados com conteúdos de *streaming*, com aspectos regulatórios bastante distintos. A Figura 4.3 demonstra tal característica em uma tela *Home* de uma *Smart TV* hipotética.

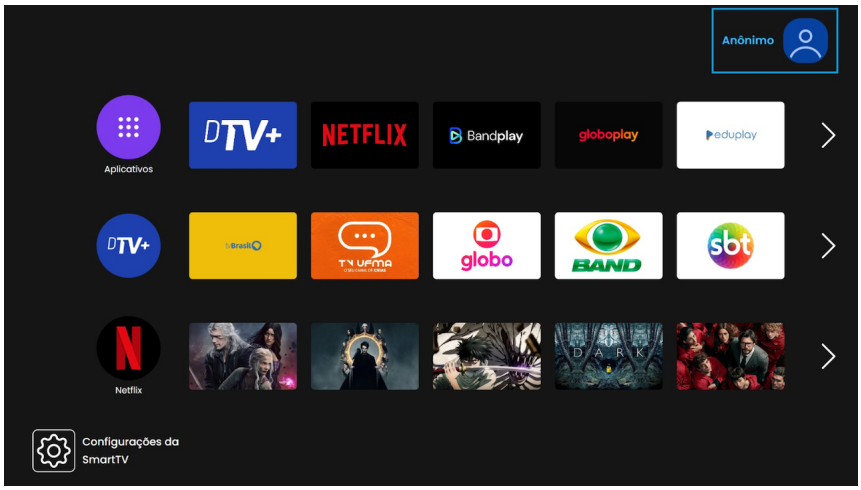

Figura 4.3: Prototipagem de Tela *Home* hipotética de uma *smart TV.*

Centralizando o acesso aos serviços TV 3.0 de maneira uniforme e integrada entre diferentes fabricantes e modelos de receptores, o Catálogo de Aplicativos se estabelece como a interface principal para listagem e acesso aos aplicativos de TV 3.0, compatível com os princípios de interoperabilidade e neutralidade da plataforma. Projetado com ênfase na usabilidade, o catálogo consolida os aplicativos em uma única interface acessível e intuitiva, favorecendo a navegação fluida entre diferentes emissoras e serviços interativos. A Figura 4.4 ilustra o Catálogo de Aplicativos, como prototipado para validação funcional, discussão em grupos focais, em pesquisa de opinião e demonstrações.

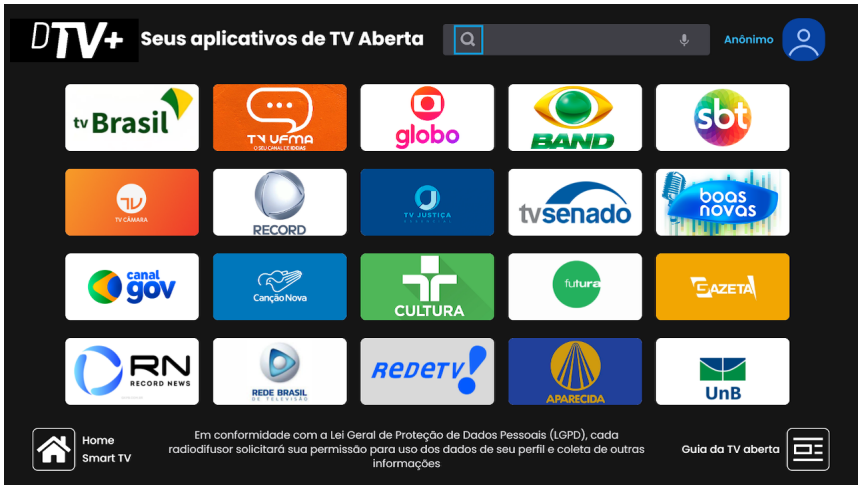

Figura 4.4: Prototipagem do Catálogo de Aplicativos DTV+.

Após um processo de varredura de sinal, o catálogo é automaticamente populado com todos os *Bootstrap Applications* anunciados, refletindo dinamicamente as atualizações de sinalizações, tanto de transmissões da TV 3.0 quanto da TV 2.X. A arquitetura prevê mecanismos de atualização contínua, com persistência dos dados e controle do número máximo de *Bootstrap Applications* instalados (60 no total). Restrições adicionais garantem o equilíbrio entre a disponibilidade de serviços e a eficiência de navegação, como o limite de até 4 *Bootstrap Applications* por canal de radiofrequência com largura de 6 MHz para programações lineares transmitidas por radiodifusão e mais 4 *Bootstrap Applications* para programações lineares exclusivamente entregues por Internet.

Cada *Bootstrap Application* no catálogo é representado com nome e ícone, permitindo identificação rápida pelo telespectador. A personalização da ordenação dos aplicativos e a presença de filtros de busca baseados em metadados enriquecem a experiência de uso, contribuindo para o acesso eficiente a conteúdos relevantes. O catálogo também integra recursos complementares da plataforma, como os guias EPG e ECG, além de oferecer acesso ao gerenciador de perfis do usuário, que permite configurar preferências individuais de idioma, acessibilidade e controle parental.

Durante o consumo de um serviço TV 3.0, o Catálogo de Aplicativos pode ser acionado pelo telespectador para expor uma interface de segundo nível, o Painel de Acesso. Esse painel permite o acesso direto a funcionalidades contextuais e à lista de Aplicativos Secundários disponíveis, obtida por meio do Gerenciador de Ciclo de Vida de Aplicativos, ampliando as possibilidades de interação e customização da experiência audiovisual.

4.2.2. Guia Eletrônico de Programação

O Guia Eletrônico de Programação (EPG - *Electronic Programming Guide*) oferece uma visão da programação linear das emissoras, permitindo ao telespectador escolher e gerenciar conteúdos em grade. O EPG permite que os telespectadores planejem o que assistir. Em receptores conectados à Internet, é possível ao telespectador quebrar a linearidade da programação, com a opção de assistir programas que já foram ou que ainda serão exibidos, para aqueles que as emissoras sinalizem estar disponíveis via *streaming* OTT. Esta é uma das principais mudanças na forma de consumo de televisão aberta introduzida pela TV 3.0: a centralidade da programação deixa de ser atributo exclusivo das emissoras e passa a ser gerenciada de acordo com a temporalidade de preferência do telespectador.

Diferentemente dos guias convencionais, o EPG da TV 3.0 organiza e exibe informações sobre programas passados, presentes e futuros de múltiplos radiodifusores. Além disso, ele integra funcionalidades adicionais, como descrições de programas, informações de agendamento, metadados detalhados e funcionalidades de busca e filtragem por categorias, facilitando o acesso a conteúdos ao vivo e sob demanda.

O EPG oferece duas interfaces distintas para a navegação. A Visualização Simples exibe informações básicas sobre os programas anterior, atual e seguinte de cada emissora, permitindo acesso rápido a dados imediatos. Para uma exploração mais detalhada, a Visualização Estendida organiza os programas em uma linha do tempo, alinhando-os com base nos horários de início e fim, com dimensões proporcionais à duração de cada programa. A Figura 4.5 ilustra tal parte da prototipagem do EPG.

Ao ser selecionado um programa ao vivo, o EPG promove o acesso direto ao

serviço linear correspondente via *Bootstrap Application* da emissora. No caso de ser selecionado um conteúdo fora do horário, o EPG exibe uma tela intermediária de descrição do conteúdo. Caso esteja disponível por *streaming*, essa tela traz habilitada a função "Assista agora", que, quando acionada, dispara o *Bootstrap Application* da emissora, desta vez parametrizado com a fonte de mídia do conteúdo escolhido.

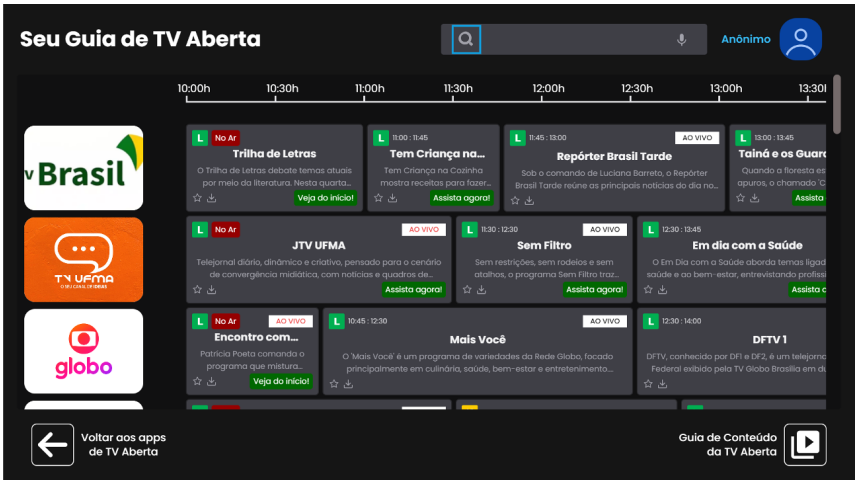

Figura 4.5: Prototipagem do EPG.

4.2.3. Guia Eletrônico de Conteúdo

O Guia Eletrônico de Conteúdo (ECG - *Electronic Content Guide*) da TV 3.0 é uma interface dedicada à agregação de conteúdos sob demanda anunciados pelos serviços de TV 3.0. Diferentemente do Guia Eletrônico de Programação (EPG), que combina programações lineares e sob demanda, o ECG foca exclusivamente em conteúdos sob demanda, oferecendo uma visão abrangente e acessível para explorar programas além das limitações de horários tradicionais de transmissão. Essa interface permite aos telespectadores navegar por um catálogo de conteúdos disponíveis, organizados e apresentados de forma intuitiva e agregados seja por emissoras, seja por gênero televisivo. A Figura 4.6 lustra uma prototipagem do ECG.

O ECG mostra ao telespectador programas disponíveis nos últimos sete dias e nos sete dias seguintes à programação linear. Ele também oferece ferramentas para busca de conteúdo por critérios como título, gênero, serviço e outros atributos, além de permitir a filtragem com base em características específicas, como tipo de conteúdo e emissora.

4.2.4. Descoberta de *Bootstrap Applications*

A descoberta de *Bootstrap Applications* ocorre durante o processo de varredura do espectro de frequências, etapa fundamental na inicialização do receptor de TV 3.0. Nessa fase, o dispositivo explora as faixas de transmissão terrestre disponíveis em sua região geográfica, identificando automaticamente os serviços de televisão ativa presentes no ambiente. Para cada serviço detectado, o receptor instancia localmente um *Bootstrap Application*, sem a necessidade de transferência real de aplicativos via OTA ou OTT. A criação desses aplicativos baseia-se em um *aplicativo-modelo comum*, padronizado pelo fabricante para todos os serviços de TV 3.0, e que é dinamicamente personalizado com metadados trans-

Figura 4.6: Prototipagem do ECG.

mitidos no próprio sinal. Esses metadados compõem o *Bootstrap Application Manifest* (BAM), estruturado em formato XML e transportado pela *Bootstrap Application Manifest Table* (BAMT), componente da *Low-Level Signaling* (LLS), conforme discutido no capítulo anterior. Essa abordagem evita estender a duração da varredura, assegura isonomia entre os serviços e viabiliza a inicialização imediata dos conteúdos radiodifundidos por meio de seus respectivos aplicativos. Como a BAMT pode transportar múltiplos BAMs, emissoras que operam em regime de multiprogramação ou compartilham o mesmo canal de RF passam a ter igual visibilidade no catálogo de aplicativos, promovendo tratamento equitativo e pluralidade de oferta.

O elemento BAM, codificado em XML, encapsula os metadados essenciais que personalizam a instância do *Bootstrap Application* gerado para cada serviço de TV 3.0. Seus atributos principais incluem o identificador global do serviço (`@globalServiceId`), a versão do aplicativo (`@appVersion`), o nome do serviço (`@appName`), e descritores visuais como o logotipo quadrado (`@appIcon`) e o logotipo em formato 16:9 (`@bannerIcon`), ambos no formato SVG e limitados a 16KB. Informações adicionais, como o slogan do serviço (`@appDescription`) e os códigos de cor de fundo e de primeiro plano da interface (`@backgroundColor` e `@foregroundColor`), completam a caracterização visual do aplicativo. Três elementos filhos complementam o BAM: `initialMediaURLs`, que define uma lista de URLs padrão para acesso ao conteúdo linear; `ecgAppDeepLink`, que aponta para um *Broadcaster Application* com catálogo de conteúdo sob demanda daquela emissora; e `audienceMeasurementSession`, que permite a configuração para início de uma sessão de coleta de dados de audiência, respeitando as preferências de privacidade do telespectador. Sessões de coleta de dados de audiência podem também ser iniciadas por meio de API do TV 3.0 WebServices.

4.2.5. Coleta de Dados de Audiência

O componente *Audience Measurement Manager* (AMM) da TV 3.0 é responsável por gerenciar sessões de medição de audiência de forma autônoma, realizando no receptor a coleta, armazenamento e entrega segura de dados, mediante solicitação das emissoras e com consentimento do telespectador. Essa abordagem elimina a necessidade de in-

serção de *scripts*, *cookies* ou gatilhos específicos nos aplicativos, centralizando a coleta no *middleware*. A coleta respeita o contexto de execução de cada sessão, definido pelo par <service-globalServiceID, viewer-id>, garantindo que os dados estejam vinculados à emissora e ao perfil de telespectador no momento da ativação.

As sessões de coleta podem ser iniciadas por duas vias: a partir de metadados BAM de um *Bootstrap Application*, ou sob demanda por um *Broadcaster Application* via TV 3.0 WebServices. Uma vez ativada, a sessão é gerenciada integralmente pelo AMM. A coleta continua mesmo com a troca de aplicativo no mesmo serviço TV 3.0, e é automaticamente pausada se o contexto mudar (como no zapeamento, troca de perfil ou perda de energia). Ao ter seu contexto recuperado, a sessão é retomada do ponto em que foi pausada. Cada sessão possui um tempo de validade e pode incluir pontos de verificação para envio progressivo dos dados, antes da entrega final à emissora.

O ciclo de vida de uma sessão é modelado por uma máquina de estados, que permite transições entre os estados de inicialização, coleta, pausa, interrupção, conclusão, entrega e falha. Essas transições dependem de eventos como expiração de tempo, comandos da emissora ou mudanças contextuais. A autenticidade dos dados é assegurada por assinaturas digitais e qualquer falha no processo de autenticação ou entrega leva a estados finais específicos com descarte dos dados, mantendo apenas metadados básicos da sessão.

Os dados de medição coletados abrangem informações do serviço, receptor, status físico do sinal, configurações do telespectador e eventos de mídia, de interação e de aplicativos. Cada sessão gera identificadores únicos (`session-handle`, `session-reportId`) e possui URLs específicas definidas pela emissora para a entrega dos relatórios. O conjunto de dados permite, assim, rastrear todo o ambiente de execução de aplicativos, de consumo de mídia e o comportamento do telespectador, compondo um panorama detalhado da experiência de uso da TV 3.0.

A entrega dos dados ocorre no formato JSON, com três objetos principais: *am-session-header* (dados básicos da sessão), *am-session-initstatus* (estado inicial do dispositivo, serviço e plataforma) e *am-session-event* (eventos com marcação temporal). A estrutura suporta entregas parciais ao longo da sessão, favorecendo o monitoramento contínuo e agregações em tempo quase real.

No entanto, o compartilhamento de dados de consumo de mídia pode levantar preocupações legítimas quanto à privacidade dos telespectadores, especialmente quando há o envolvimento de dados pessoais definidos nos perfis, mantidos pelas emissoras. Por esse motivo, a TV 3.0 define um *framework* de privacidade, detalhado na seção seguinte.

4.2.6. Framework de Privacidade

O *Privacy Manager* da TV 3.0 é o componente responsável por implementar um *framework* de privacidade compatível com a ISO/IEC 27560 (2023). O *framework* combina registros formais de consentimento e recibos verificáveis, permitindo que telespectadores controlem suas preferências de privacidade de forma clara e auditável.

Caso a emissora se engaje no tratamento de dados pessoais, deve incluir na tranismissão uma descrição de registro de privacidade (PRRD - *Privacy Record Request Description*), como fragmento do Service Layer Signaling (SLS), em formato XML. A PRRD

descreve detalhadamente os tipos de dados que serão coletados, as finalidades do processamento, as bases legais utilizadas, e os agentes envolvidos. O esquema da PRRD adota vocabulários padronizados, como o *Data Privacy Vocabulary* (DPV) do W3C (2024a) e *Personal Data Categories* (PD), também do W3C (2024b), para identificar de forma semântica as finalidades de uso, os tipos de dados e os direitos aplicáveis. Com base nesses dados, o *Privacy Manager* gera uma interface que permite ao telespectador exercer escolhas informadas, em conformidade com o princípio da autodeterminação informativa.

Após a interação do telespectador com a interface de privacidade, o *Privacy Manager* gera um registro de privacidade formal, no qual as decisões do usuário são documentadas. Esses registros, armazenados em formato XML, servem como evidência do consentimento ou recusa, contendo eventos relacionados às escolhas feitas, com marcação temporal e validade definida. Cada evento é identificado com base nas finalidades selecionadas e pode ser utilizado para controlar o acesso a APIs sensíveis.

A entrega bem-sucedida do registro de privacidade à emissora resulta na geração de um recibo de privacidade, que espelha o conteúdo do registro e é armazenado no dispositivo do telespectador. Esse recibo permite consulta posterior e revisão das escolhas. Quando uma nova PRRD é detectada, ou quando o telespectador acessa manualmente o *Privacy Manager*, a interface exibe as opções anteriores para edição ou reafirmação.

As visualizações gráficas se adaptam ao conteúdo da PRRD. Se não houver finalidades baseadas em consentimento, uma visualização simplificada é apresentada. Caso contrário, o usuário pode aceitar todas as finalidades ou optar por gerenciar individualmente suas escolhas, por meio de botões de alternância. As opções baseadas em consentimento têm o estado padrão “negar”, enquanto as baseadas em outras bases legais, como interesse legítimo, assumem “permitir” como padrão.

O *Privacy Manager* também oferece acesso manual às configurações de privacidade a qualquer momento, inclusive quando nenhum serviço de TV 3.0 está ativo. Nesse caso, uma lista de emissoras que já enviaram PRRDs é apresentada, permitindo ao telespectador revisar ou modificar suas preferências. Todas as ações geram novos eventos no registro e no recibo de privacidade, garantindo rastreabilidade e transparência contínuas.

4.2.7. Extensibilidade

A camada de Codificação de Aplicações da TV 3.0 foi concebida com foco em extensibilidade, permitindo a evolução contínua de funcionalidades e a adaptação dos aplicativos às capacidades específicas das implementações do DTV+ nos receptores. Para isso, foram definidos mecanismos que permitem aos aplicativos identificar dinamicamente os recursos disponíveis no ambiente de execução, tornando possível a adequação do seu comportamento a diferentes perfis de dispositivos. Esse suporte é viabilizado por meio de APIs de consulta de capacidades disponibilizadas tanto pelo TV 3.0 Ginga-NCL quanto pelo TV 3.0 WebServices. Cada API desses componentes possui um identificador exclusivo associado. Os *Broadcaster Applications* podem utilizar esses identificadores para consultar a versão de cada API suportada pela implementação, permitindo verificar se determinada funcionalidade está presente e qual sua versão de suporte.

Essa abordagem permite que aplicativos de TV 3.0 desenvolvidos com base em recursos mais recentes possam identificar se estão sendo executados em receptores com-

patíveis, e adaptar seu comportamento caso contrário. Dessa forma, a compatibilidade entre diferentes gerações de dispositivos é preservada sem sacrificar a inovação. Aplicativos mais sofisticados podem oferecer experiências aprimoradas em receptores com maior capacidade, ao mesmo tempo em que permanecem funcionais, com funcionalidades reduzidas, em receptores mais simples.

4.3. TV 3.0 - Ginga-NCL e suas novas APIs (NCL 4.0)

A evolução do Ginga-NCL na TV 3.0, consolidada pela versão 4.0 da linguagem NCL, introduz um conjunto de novas APIs projetadas para ampliar as possibilidades de interação, personalização e imersão nos aplicativos. Essa nova geração de funcionalidades inclui suporte à identificação e ao gerenciamento de perfis de múltiplos telespectadores, interação multimodal (com comandos de voz e gestos), experiências colaborativas com múltiplos telespectadores, acionamento de efeitos sensoriais (como luzes e vibração), integração com múltiplos dispositivos e suporte a experiências imersivas com realidade virtual. Adicionalmente, NCLua foi estendido para permitir o acesso estruturado ao TV 3.0 WebServices, viabilizando uma maior sinergia entre os dois subsistemas.

4.3.1. Identificação e perfis de múltiplos telespectadores

Um perfil de telespectador (Josué et al., 2023) é caracterizado por um conjunto de atributos que permitem que fabricantes de dispositivos (ou desenvolvedores de middleware) e emissoras usem esses atributos e adicionem outros que julgarem necessários para fornecer serviços personalizados ao telespectador. Neste contexto, a TV 3.0 define um conjunto de atributos, organizados em uma hierarquia de três níveis, conforme mostra a Figura 4.7.

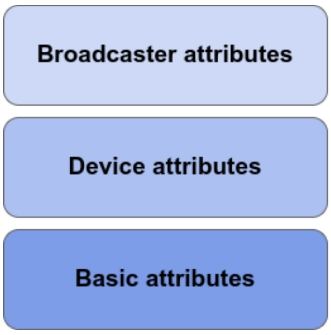

Figura 4.7: Organização hierárquica dos atributos de perfil de telespectador.

Os atributos básicos de um perfil de telespectador são os atributos padronizados, que podem ser acessados pelo fabricante do receptor e pelas emissoras às quais o telespectador tenha dado permissão. Os telespectadores podem criar esses atributos ao cadastrar seu perfil, e atualizá-los sempre que desejarem através da plataforma de software da TV 3.0, ou seja, por um aplicativo nativo fornecido pelo receptor de TV 3.0. O conjunto de atributos básicos de um perfil de telespectador está descrito na Tabela 4.1.

O fabricante do dispositivo (ou desenvolvedor de *middleware*), por sua vez, pode especificar um conjunto de atributos adicionais e também gerenciar seu acesso. Esse gerenciamento deve incluir a solicitação de consentimento do telespectador para acessar tais atributos. Os atributos do dispositivo são aqueles que podem ser customizados pelo teles-

Tabela 4.1: Atributos básicos de perfil de telespectador

Atributo	Tipo	Obrigatoriedade
id	string UUID	sim
nickname	string	sim
parentalControl	booleano	sim
maxContentRating	valores predefinidos conforme o país	sim
avatar	caminho para arquivo	não
audioLanguage	string	sim
closedCaptioningLanguage	string	sim
userInterfaceLanguage	string	sim
closedCaptioning	booleano	sim
closedSigning	booleano	sim
closedSigningSide	left (esquerda) ou right (direita)	sim
closedSigningWidth	inteiro (14 a 28)	sim
audioDescription	booleano	sim
dialogueEnhancement	booleano	sim
voiceGuidance	booleano	sim

pectador em seu perfil e que podem ser úteis para personalizar o conteúdo das emissoras. Finalmente, cada emissora também pode especificar seus atributos adicionais. Os atributos da emissora são armazenados dentro de seu contexto localmente no receptor de TV 3.0 e só podem ser utilizados pelos aplicativos e serviços transmitidos por aquela emissora. Os atributos básicos ou atributos do dispositivo podem ter seu conteúdo redefinido no contexto de uma emissora. Assim, os telespectadores podem personalizar seus valores de atributos para cada emissora de uma maneira diferente, se desejarem.

A linguagem NCL 4.0 suporta identificação multiusuário por meio de um módulo de linguagem chamado Users, que inclui dois elementos NCL – `<userBase>` e `<userProfile>` conforme descrito na Tabela 4.2. Um elemento `<userBase>` pode ser declarado dentro do elemento `<head>` do documento NCL e define um conjunto de elementos `<userProfile>`. O elemento `<userProfile>` descreve um conjunto de características para selecionar o usuário alvo de uma aplicação.

Tabela 4.2: Elementos e atributos do módulo Users

Elemento	Atributo	Conteúdo
userBase	id	(importBase|userProfile)+
userProfile	id, src, min, max	N/A

O atributo id dos elementos `<userBase>` e `<userProfile>` identifica cada elemento dentro da aplicação NCL 4.0. O atributo src especifica o caminho para um arquivo JSON contendo expressões lógicas que combinam os valores de atributos de um telespectador com os operadores lógicos AND e OR. Este arquivo permite especificar um telespectador ou grupo de perfis de telespectadores através de uma consulta formalizada na linguagem BNF (Backus-Naur Form) conforme segue[1].

[1]O JSON Schema que especifica o formato do arquivo a ser indicado no atributo src do elemento `<userProfile>` está disponível em http://ncl.org.br/NCL4.0/user.schema.json.

```
<expression> := <simpleExpression> | <compositeExpression>
<expressionList> := <expression> | <expression>, <expressionList>
<compositeExpression> :=
  {"and":[<expressionList>]} | {"or":[<expressionList>]}
<simpleExpression> :=
  {"attribute":"<attName>", "comparator":"<comp>", "value":"<string>"}
<attName> :=
  language | age | gender | isGroup | captions | signLanguageVideo |
  audioDescription | dialogueEnhancement | ...
<comp> := eq | neq | lt | lte | gt | gte
```

O atributo “min” define o número mínimo de telespectadores que devem seguir este perfil na aplicação NCL. Por padrão, o valor do atributo min é zero e o autor pode especificar valores inteiros maiores ou iguais a zero. Já o atributo “max” define o número máximo de telespectadores que se enquadram em um perfil e que podem ser instanciados pela aplicação NCL. O atributo “max” aceita valores maiores ou iguais ao valor de “min” ou “unbounded” (valor padrão). Para acessar as propriedades do usuário em uma aplicação NCL, é necessário utilizar um tipo de nó de mídia chamado UserSettingsNode. Um documento NCL pode ter uma ou mais mídias do tipo `x-ncl-user-settings` e permite o carregamento de informações do telespectador nesses nós de conteúdo. Além disso, o elemento `<media>` do tipo `x-ncl-user-settings` pode conter um atributo chamado “user”. Este atributo pode fazer referência a um perfil especificado no elemento `<userProfile>` presente no cabeçalho do documento NCL ou fazer referência ao usuário atualmente ativo. Durante a análise do documento NCL, se houver uma mídia `x-ncl-user-settings` cujo atributo de usuário aponta para um `<userProfile>`, o *middleware* consultará a base de usuários considerando a consulta especificada. Se houver telespectadores registrados no receptor que correspondam à consulta de pesquisa, o *middleware* instancia novo nó de mídia do tipo `x-ncl-user-settings` para cada um desses telespectadores e suas propriedades são nele carregadas.

Em receptores de TV 3.0, o telespectador pode selecionar um perfil, indicando quem está assistindo a TV no momento. Essa seleção é feita quando o telespectador entra no DTV+, ou caso queira alterar o perfil selecionado. A variável global currentUser permite que uma aplicação NCL identifique qual perfil está ativo enquanto a aplicação é executada. O autor pode relacionar um nó do tipo `x-ncl-user-settings` ao perfil ativo, declarando o atributo user no elemento de mídia `x-ncl-user-settings`, com o valor currentUser, conforme mostrado a seguir. Quando o *middleware* identifica um nó do tipo `x-ncl-user-settings` com a propriedade do usuário igual a currentUser, ele carrega automaticamente as propriedades do perfil do telespectador ativo para este nó.

```
<media id="user" type="application/x-ncl-user-settings"
       user="currentUser">
   <property name="userInterfaceLanguage"/>
</media>
```

4.3.2. Interação multimodal

O suporte à interação multimodal em NCL 4.0 (Barreto et al., 2019b, 2020, 2024) ocorre através da especificação de tipos de eventos NCL, como: selection, VoiceRecognition, HandPoseRecognition, FaceExpressionRecognition, Touch e EyeGaze. Semelhante ao

evento de seleção, onde o atributo key indica a tecla que foi pressionada pelo usuário, os demais tipos de eventos de interação também utilizam o atributo key para especificar o que foi reconhecido na interação multimodal. Por exemplo, para um evento VoiceRecognition, o valor do atributo key indica o comando de voz (palavra ou frase) a ser reconhecido. A definição da semântica dos valores dos atributos key, ou seja, os tipos de gestos, comandos de voz, expressões faciais, etc., depende do tipo de evento correspondente. A Tabela 4.3 apresenta os nomes de papéis predefinidos de condição de conector relacionados aos eventos de interação multimodal e a semântica utilizada para o valor do atributo key.

Tabela 4.3: Papéis predefinidos relacionados a eventos de interação multimodal

Papel	**key**
onSelection	tecla do controle remoto
onVoiceRecognition	palavra ou frase a ser reconhecida
onFaceExpressionRecognition	tipo da expressão facial: HAPPINESS, SURPRISE, NEUTRAL, SADNESS, FEAR, DISGUST, e ANGER
onHandPoseRecognition	tipo do gesto com a mão: CLOSED_FIST, OPEN_PALM, POINTING_UP, THUMBS_DOWN, THUMBS_UP, VICTORY, LOVE, HANG_LOOSE, THREE, FOUR, HORNS.
onBeginEyeGaze	N/A
onEndEyeGaze	N/A
onAbortEyeGaze	N/A
onBeginTouch	N/A
onEndTouch	N/A

4.3.3. Interação de múltiplos telespectadores

A identificação multiusuário (Barreto et al., 2023) também pode ser aplicada a links para especificar comportamentos, dependendo de quem interage com a aplicação. Para isso, o autor do aplicativo deve adicionar o parâmetro do conector "user" ao elemento `<link>`. Este atributo "user" deve fazer referência ao ID de um elemento `<userProfile>`. Assim, quando a aplicação é executada, o *middleware* Ginga cria links dinamicamente para todos os telespectadores deste receptor que correspondam ao perfil. A Figura 4.8 ilustra o processo de criação de links dinâmicos pelo Ginga, onde são criados links para diferentes telespectadores (user01, ..., userN) a partir do link original que referencia um perfil.

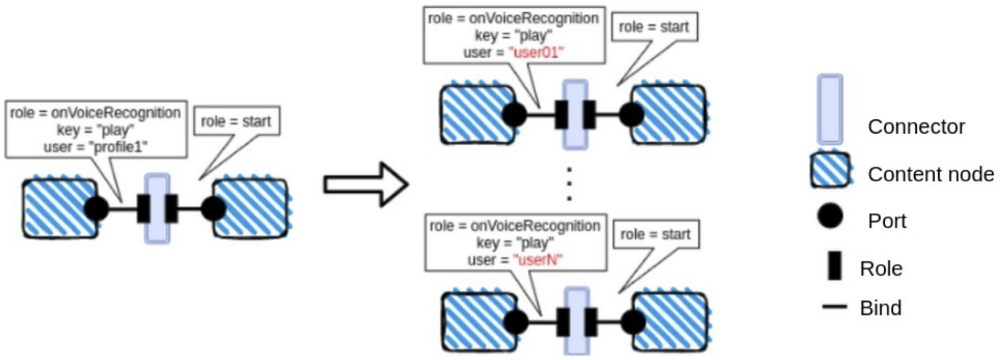

Figura 4.8: Representação da criação de link dinâmico para cada telespectador que corresponde ao profile1.

Por exemplo, considere uma aplicação NCL 4.0 que define um elemento userProfile e um link que faz referência a esse perfil da seguinte forma:

```
<ncl>
  <head>
    ...
    <userBase>
      <userProfile id="profile1" max="2" src="profile1.json" />
    </userBase>
    ...
  </head>
  <body>
    ...
    <link id="l1" xconnector="onVoiceRecognitionStop">
      <bind role="onVoiceRecognition" component="video">
        <bindParam name="key" value="parar"/>
        <bindParam name="user" value="profile1"/>
      </bind>
      <bind role="stop" component="video"/>
    </link>
    ...
  </body>
</ncl>
```

Suponha que dois telespectadores correspondam ao perfil de usuário "perfil1" (por exemplo, U01 e U02) no receptor que executará esta aplicação. Neste caso, o link "l1" será replicado em dois links, conforme segue:

```
<link xconnector="onVoiceRecognitionStop">
  <bind role="onVoiceRecognition" component="video">
     <bindParam name="key" value="parar"/>
     <bindParam name="user" value="U01"/>
  </bind>
  <bind role="stop" component="video"/>
</link>

<link xconnector="onVoiceRecognitionStop">
  <bind role="onVoiceRecognition" component="video">
    <bindParam name="key" value="parar"/>
    <bindParam name="user" value="U02"/>
  </bind>
  <bind role="stop" component="video"/>
</link>
```

4.3.4. Efeitos sensoriais

Visando aumentar a qualidade da experiência dos telespectadores, a linguagem NCL 4.0 permite especificar efeitos sensoriais (Barreto et al., 2019a; Ivanov et al., 2024), que podem ser sincronizados com o conteúdo audiovisual exibido. Por exemplo, em uma transmissão de vídeo que apresenta uma praia, é possível renderizar um efeito de cheiro de mar e um efeito de vento, emulando a brisa da praia.

Os efeitos sensoriais são representados como um nó do documento NCL, da mesma forma que acontece com uma mídia audiovisual. Assim, o autor do documento NCL 4.0 pode utilizar os mesmos conceitos já utilizados para manipular nós de mídia para nós de

efeito sensorial. O elemento `<effect>` possui um conjunto de atributos para identificar e caracterizar a ocorrência do efeito sensorial. Um elemento `<effect>` é especializado pelo atributo "type", que pode receber um dos seguintes valores: LightEffect, TemperatureEffect, WindEffect, VibrationEffect, SprayingEffect, ScentEffect, FogEffect.

Um efeito sensorial pode ter um conjunto de propriedades que definem seu comportamento ao ser renderizado. Essas propriedades podem ser comuns a todos os tipos de efeitos ou específicas a um tipo de efeito sensorial. A Tabela 4.4 apresenta o conjunto de propriedades que podem ser definidas para efeitos sensoriais.

Tabela 4.4: Propriedades de efeitos sensoriais

Nome da Propriedade	**Descrição**
intensityValue	intensidade do efeito de forma relativa, que pode variar de 0 a 10
color	cor do efeito de luz como um nome ou valor RGB
scent	tipo de aroma a ser usado
frequency	número de oscilações por segundo (Hz) para o tipo LightType

As propriedades de um efeito sensorial podem ser definidas em um documento NCL através do elemento `<property>`, filho de `<effect>`. Além disso, as propriedades de um efeito sensorial também podem ser especificadas como parâmetros de um elemento `<descriptor>` referenciado no atributo descriptor do elemento `<effect>`. A posição do efeito sensorial é definida através de novos atributos adicionados ao elemento `<region>`. Desta forma, o elemento `<region>` pode ser utilizado para especificar a região de exibição para mídias tradicionais ou efeitos sensoriais. Para efeitos sensoriais, é possível especificar a localização de duas maneiras. Primeiro, o autor do NCL 4.0 pode utilizar um sistema de coordenadas esféricas, no qual a região começa no ponto indicado pelos atributos azimuthal e polar, e os atributos width e height indicam o tamanho da área a ser utilizada para renderizar o efeito. Com este sistema de coordenadas, o autor da aplicação NCL 4.0 não precisa se preocupar com o tamanho do ammbiente física. Considere o exemplo a seguir, que representa a posição P na Figura 4.9. Esta posição pode ser definida atribuindo os valores -45 e 45 aos atributos de posicionamento polar e azimuthal, respectivamente, conforme mostrado na Figura 4.9.

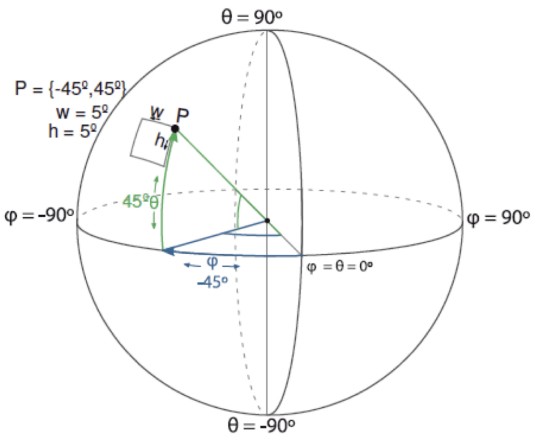

Figura 4.9: Definição de região de renderização de efeito usando coordenadas esféricas.

O autor da aplicação também pode definir a direção do efeito usando coordenadas (inter)cardinais (atributo direction) e também o atributo location, que define a localização do efeito em um espaço 3D. O valor da propriedade location é uma concatenação das posições x, y e z definidas de acordo com o padrão MPEG-V ISO/IEC 23005-3:2019 (2019). A Figura 4.10 ilustra todas as posições possíveis de acordo com o MPEG-V.

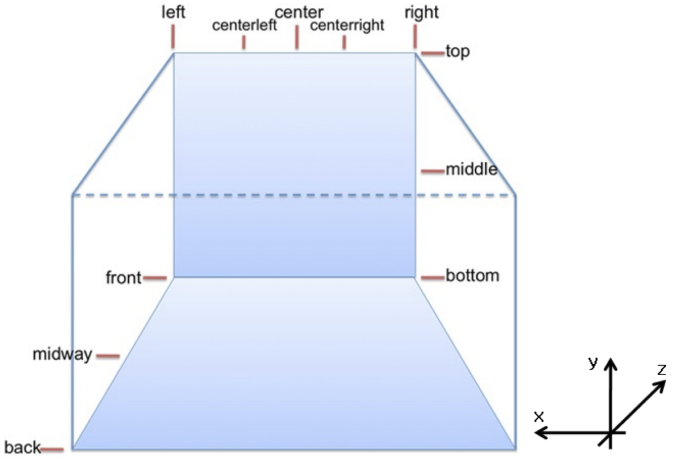

Figura 4.10: Modelo espacial para posicionar efeitos sensoriais no MPEG-V.

4.3.5. Múltiplos dispositivos

O suporte a múltiplos dispositivos (dos Santos et al., 2024) é fornecido por uma nova API da TV 3.0 *WebServices*, para registro de dispositivos no ambiente doméstico como dispositivos remotos, juntamente com um conjunto de mensagens para troca de metadados entre o *middleware* Ginga e o mecanismo de apresentação do dispositivo remoto. Essas mensagens permitem que o Ginga gerencie a apresentação de um nó no dispositivo remoto e atualize o estado da apresentação no Ginga. A Figura 4.11 apresenta parte do receptor TV 3.0 considerando os componentes necessários para suporte a múltiplos dispositivos.

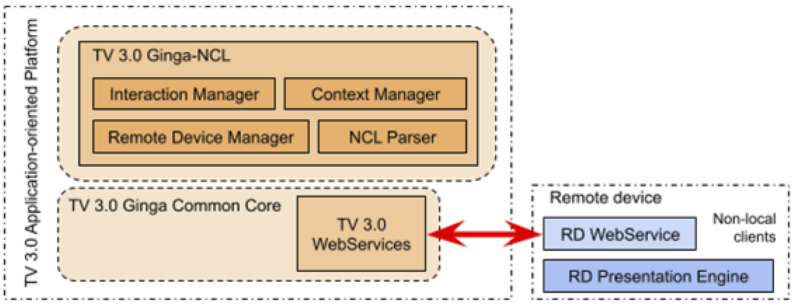

Figura 4.11: Arquitetura resumida do receptor de TV 3.0.

O *Interaction Manager* é o componente responsável por gerenciar as interações que ocorrem durante a execução de um aplicativo NCL. As interações por ele gerenciadas podem ocorrer na própria TV ou por meio de dispositivos de reconhecimento externos conectados à TV. O *Context Manager* é responsável por gerenciar o contexto de execução do aplicativo, como o suporte multiusuário para aplicativos NCL. Finalmente, o componente *XML Parser* carrega um documento NCL para execução.

4.3.5.1. Mensagens de metadados

Um dispositivo remoto registrado no TV 3.0 *WebServices*, pode trocar mensagens com o Ginga por meio do *WebSocket* criado durante o registro. Essas mensagens contêm metadados para permitir que o Ginga gerencie a apresentação de um nó no dispositivo remoto e atualize o estado da apresentação no Ginga. O formato das mensagens de metadados trocadas entre um Dispositivo Remoto e o Ginga é definido nas Tabelas de 4.5 a 4.7.

Tabela 4.5: Mensagem com metadados sobre o nó a ser reproduzido no dispositivo remoto

Sentido de Mensagem:	Do Ginga para o dispositivo remoto

Corpo da Mensagem:	`{`
	`  "nodeId" : <nodeId>,`
	`  "nodeSrc" : <URL>,`
	`  "appId" : <appId>,`
	`  "type" : <mimeType>,`
	`  "properties" : [`
	`  {"name" : <pName>, "value" : <pValue>},`
	`  ]`
	`}`

Descrição:	Permite que o Ginga envie informações sobre um nó para a máquina de apresentação do dispositivo remoto. "nodeId" é o identificador do nó a ser executado pela máquina de apresentação do dispositivo remoto. O atributo "nodeSrc" indica onde obter o conteúdo deste nó, se aplicável. O atributo "appId" apresenta o identificador do aplicativo a ser usado em rotas do TV 3.0 *webservices* para obter arquivos enviados junto com a aplicação NCL. O atributo "type" indica o tipo do nó a ser executado. O vetor "properties" carrega as propriedades do nó NCL, conforme definido na aplicação NCL.
Nota:	A mensagem é enviada pelo *middleware* Ginga assim que a aplicação NCL é iniciada e a existência de um nó a ser executado em um dispositivo remoto é identificada. Vale ressaltar que diferentes tipos de URLs podem ser indicados no atributo "nodeSrc". No caso de URLs que não definem um protocolo e apresentam uma URL relativa, o dispositivo remoto deve utilizar a API TV 3.0 *webservices* para obter o conteúdo enviado junto com a aplicação NCL. Caso contrário, o conteúdo será acessado de acordo com o protocolo especificado, se suportado pelo dispositivo remoto. Ao receber informações do nó NCL, o dispositivo remoto agora é capaz de carregar as informações necessárias para sua apresentação.

Conforme a execução da aplicação NCL se desenrola, Ginga envia ações para alterar o estado do evento do nó na máquina de apresentação do dispositivo remoto. Este último deve responder prontamente com uma notificação de transição de evento se a ação for bem-sucedida. Além disso, a qualquer momento em que o usuário interage com o

nó, ou ocorre um fim natural de seu conteúdo, a máquina de apresentação do dispositivo remoto envia uma mensagem para notificá-lo. O Ginga só deve alterar o estado do evento do nó se um dispositivo remoto notificar essa transição para o Ginga.

Tabela 4.6: Mensagem com metadados de ação para a máquina de apresentação do dispositivo remoto

Sentido de Mensagem:	Do Ginga para o dispositivo remoto
Corpo da Mensagem:	{ "nodeId" : <nodeId>, "label" : <label>, "appId" : <appId>, "eventType" : <presentation|preparation|...>, "action" : <start|stop|abort|pause|resume>, "value" : <value>, "delay" : <delay> }
Descrição:	Permite que o Ginga execute ações em um nó apresentado no dispositivo remoto. "nodeId" é o identificador do nó alvo da ação. O atributo "appId" apresenta o identificador da aplicação. Os atributos "eventType" e "action", juntos, identificam a ação a ser executada. O atributo "value" é um atributo opcional para eventos do tipo atribuição. O "delay" é um atributo opcional que indica se a ação deve ser executada com algum atraso após ser recebida. O valor padrão do atributo "delay" é zero, o que indica que a ação é executada imediatamente.
Nota:	-

Tabela 4.7: Mensagem de notificação de transição de evento para o *middleware* Ginga

Sentido de Mensagem:	Do dispositivo remoto para o Ginga
Corpo da Mensagem:	{ "nodeId" : <nodeId>, "label" : <label>, "appId" : <appId>, "eventType" : <presentation|selection|view|...>, "transition" : <starts|stops|aborts|pauses|resumes>, "value" : <value>, "user" : <userId> }

Descrição:	Permite que o dispositivo remoto relate a ocorrência de uma transição de estado de evento, sempre que necessário. "nodeId" é o identificador do nó no qual a transição de evento ocorreu. O atributo "appId" apresenta o identificador do aplicativo. "eventType" identifica o tipo de evento e "transition" a transição de evento. O atributo "value" é um atributo opcional para eventos do tipo atribuição. O atributo "user" é usado quando é necessário identificar o usuário que realiza a interação notificada.
Nota:	Um dispositivo remoto pode identificar seu usuário a partir da lista de usuários registrados na TV. Para acessar a lista de usuários da TV, ele usa a API específica fornecida para esse fim. Deve-se notar que a identificação do usuário pode ser alterada a qualquer momento. Sempre que ela é refeita, o atributo do usuário é atualizado para as mensagens subsequentes.

4.3.5.2. Gerenciamento de dispositivos remotos

Dispositivos remotos conectados ao Ginga são gerenciados pelo componente *Remote Device Manager* (RDM) (ver Figura 4.11). O RDM mantém a lista de dispositivos conectados, bem como as informações recebidas durante seu registro.

Cada mensagem trocada na comunicação entre Ginga e dispositivos remotos é encaminhada de/por TV 3.0 *webservices* para RDM. As mensagens são acompanhadas pelo identificador do dispositivo remoto. Com base nessas informações, é possível ao *webservices* identificar o *WebSocket* correto para encaminhar uma mensagem ou o RDM identifica o nó NCL correspondente para atualizar seu estado na aplicação NCL.

Quando uma ação é executada em um nó, ela é passada ao RDM e, então, encaminhada ao *webservices* junto com o identificador do dispositivo remoto para que a mensagem correspondente seja enviada ao dispositivo remoto. A Tabela 4.8 resume o comportamento de eventos associados a um nó em execução em um dispositivo remoto.

Tabela 4.8: Comportamento de eventos de mídia

Transição	Tipo evento	Descrição
sleeping → occurring (starts)	Preparação	Início da preparação, o RDM envia todos os metadados disponíveis do nó para a máquina de apresentação do dispositivo remoto.
occurring → sleeping (stops)	Preparação	Fim da preparação, a máquina de apresentação do dispositivo remoto terminou de carregar o conteúdo do nó, no caso de um nó de mídia, ou a configuração do atuador, no caso de um nó de efeito sensorial.

occurring → sleeping (aborts)	Preparação	A preparação terminou com um erro. Entre outras causas, um abort de preparação pode ocorrer quando o dispositivo remoto não tem o *bind-token* apropriado para acessar o conteúdo da aplicação NCL.
sleeping → occurring (starts)	Apresentação	Início da execução.
occurring → sleeping (stops)	Apresentação	Fim da execução.
occurring → sleeping (aborts)	Apresentação	Perda de conexão com o dispositivo. Desse ponto em diante, quaisquer ações no nó são ignoradas.
occurring → paused (pauses)	Apresentação	O dispositivo remoto entrou no modo de espera. Desse ponto em diante, quaisquer ações no nó são armazenadas pelo RDM.
paused → occurring (resumes)	Apresentação	O dispositivo remoto saiu do modo de espera. As ações armazenadas pelo RDM são enviadas ao dispositivo.

Mensagens de transição de evento são recebidas a qualquer momento em que o usuário interage com o nó ou ocorre um fim natural de seu conteúdo. Também é esperado que dispositivos remotos respondam prontamente com uma notificação de transição de evento se a ação recebida for bem-sucedida. Quando uma mensagem de transição de evento é recebida pelo *webservice* e encaminhada ao RDM, este atualiza o estado do evento do nó NCL correspondente à mensagem recebida. O RDM só deve alterar o estado do evento de um nó se um dispositivo remoto notificar essa transição ao RDM. Além disso, em caso de desconexão do dispositivo remoto, o RDM aciona uma transição de *abort* para o nó correspondente ao dispositivo desconectado. Desse ponto em diante, quaisquer ações no nó são ignoradas.

4.3.5.3. Suporte do Parser a múltiplos dispositivos

Executar um nó NCL em uma máquina de apresentação de dispositivo remoto requer identificação explícita da classe do dispositivo remoto. A proposta apresentada aqui não restringe nomes de classe, permitindo que cada tipo de dispositivo defina seu nome de classe. O código NCL abaixo exemplifica a especificação de um nó que deve ser executado em um dispositivo remoto de uma classe identificada por “class_id”.

```
<media id="M1" src="video.mp4">
   <property name="deviceClass" value="class_id"/>
</media>
```

Ao realizar o *parsing* do documento NCL, sempre que um elemento `<media>` ou `<effect>` é definido para ser executado em um dispositivo remoto – ou seja, ele define uma propriedade “deviceClass” ou é apresentado em uma base de regiões associada a um dispositivo remoto –, o analisador se comunica com o RDM para obter a lista de

dispositivos dessa classe e capazes de reproduzir o tipo de elemento. No exemplo acima, a mídia M1 deve ser apresentada em um dispositivo remoto capaz de executar um nó de vídeo do tipo vídeo/mp4. A associação de um elemento `<media>` ou `<effect>` com dispositivos remotos de uma determinada classe ocorre de duas maneiras. Em resumo, (i) nenhuma identificação explícita de dispositivo e (ii) identificação explícita de dispositivo.

(i) Nenhuma identificação explícita de dispositivo: Nesta forma de associação, o autor da aplicação identifica apenas a classe de dispositivo como um todo, sem informação específica de dispositivo, conforme ilustrado no código abaixo para um nó de mídia.

```
<media id="intro" src="intro.mp4"/>
<media id="M1" src="video.mp4">
  <property name="deviceClass" value="class_id"/>
  <area id="A1" ... />
</media>
<link>
  <bind role="onEnd" component="intro"/>
  <bind role="start" component="M1"/>
</link>
```

Neste caso, o nó M1 e os links dos quais ele participa serão replicados para cada dispositivo remoto conectado que pertença à classe “class_id”, com base na lista mantida pelo RDM. As informações sobre em qual dispositivo remoto cada nó é executado serão preenchidas automaticamente pelo parser, de acordo com a lista de dispositivos remotos fornecida pelo RDM. O código abaixo apresenta o resultado da replicação do nó M1 para dois dispositivos.

```
<media id="intro" src="intro.mp4"/>
<media id="M1RD00" src="video.mp4">
   <property name="deviceClass" value="class_id(0)"/>
   <area id="A1RD00" ... />
</media>
<media id="M1RD01" src="video.mp4">
   <property name="deviceClass" value="class_id(1)"/>
   <area id="A1RD01" ... />
</media>
<link>
   <bind role="onEnd" component="intro"/>
   <bind role="start" component="M1RD00"/>
</link>
<link>
   <bind role="onEnd" component="intro"/>
   <bind role="start" component="M1RD01"/>
</link>
```

Observe que os ids dos elementos `<media>`, `<effect>` e `<area>` são alterados durante o parsing para manter a propriedade de exclusividade desses ids de elementos no documento NCL.

(ii) Identificação explícita do dispositivo: Nesta forma de associação, o autor do aplicativo especifica explicitamente o dispositivo remoto no qual o nó deve ser reproduzido, conforme ilustrado no código abaixo.

```
<media id="intro" src="intro.mp4"/>
<media id="M1" src="video.mp4">
  <property name="deviceClass" value="class_id(0)"/>
  <area id="A1" ... />
</media>
<media id="M2" src="video.mp4">
  <property name="deviceClass" value="class_id(1)"/>
  <area id="A2" ... />
</media>
<link>
   <bind role="onEnd" component="intro"/>
   <bind role="start" component="M1"/>
   <bind role="start" component="M2"/>
</link>
```

Neste caso, não há necessidade de o *parser* replicar os nós M1 e M2 para cada dispositivo conectado, pois eles já especificam o dispositivo remoto no qual serão executados. Neste caso, em dispositivos da classe "class_id" e de acordo com a ordem de conexão desses dispositivos, conforme fornecido pelo sufixo "(i)" de cada nó, onde i começa em zero e significa o primeiro dispositivo daquela classe a se conectar.

Observe que, neste exemplo, o aplicativo prevê a possibilidade de apenas dois dispositivos. Se o RDM tiver um número menor de dispositivos conectados, o nó excedente estará no estado de *sleeping* e o formatador NCL ignorará qualquer ação realizada naquela mídia. Se o RDM tiver um número maior de dispositivos conectados, os dispositivos extras não apresentarão nada.

Adaptação ao perfil do telespectador em dispositivos remotos. A funcionalidade de múltiplos dispositivos pode ser usada em conjunto com a funcionalidade multiusuário para adaptar a execução do nó na máquina de apresentação do dispositivo remoto de acordo com o usuário do dispositivo remoto. Para exemplificar o uso combinado de ambas as funcionalidades, serão considerados dois perfis de usuário: adulto e criança. Os usuários são associados a esses perfis de acordo com sua idade. Neste caso, maiores ou menores de 18 anos, respectivamente.

O trecho de código abaixo estabelece um comportamento alternativo para o conteúdo a ser apresentado, de acordo com o perfil do usuário. Note que, neste caso, a identificação do usuário está sendo feita com base na identificação do usuário que interage com o vídeo em seu trecho de introdução. Para usuários adultos, o vídeo será reproduzido a partir da parte 1, enquanto usuários crianças pularão para a parte 2.

```
<media id="M1" src="video.mp4">
   <property name="deviceClass" value="class_id"/>
   <area id="intro" .../>
   <area id="part1" .../>
   <area id="part2" .../>
</media>
<link>
    <bind role="onSelection" component="M1" interface="intro">
       <bindParam name="user" value="adult"/>
    </bind>
    <bind role="start" component="M1" interface="part1"/>
</link>
```

```
<link>
    <bind role="onSelection" component="M1" interface="intro">
       <bindParam name="user" value="child"/>
    </bind>
    <bind role="start" component="M1" interface="part2"/>
</link>
```

Neste caso, o nó M1 e os links dos quais ele participa serão replicados para cada dispositivo conectado. Após esta primeira etapa de replicação, o código da aplicação ainda identificará os usuários por seus perfis. Portanto, a próxima etapa é a replicação de links com base nos usuários registrados na TV, de acordo com a lista fornecida pelo *Context Manager*. Como apenas um usuário pode usar um dispositivo remoto por vez, identificar o usuário que interage garante que o conteúdo que está sendo entregue corresponda ao seu perfil.

```
<media id="M1RD00" src="video.mp4">
  <property name="deviceClass" value="class_id(0)"/>
  <area id="introRD00" .../>
  <area id="part1RD00" .../>
  <area id="part2RD00" .../>
</media>
<media id="M1RD01" src="video.mp4">
  <property name="deviceClass" value="class_id(1)"/>
  <area id="introRD01" .../>
  <area id="part1RD01" .../>
  <area id="part2RD01" .../>
</media>
<link>
  <bind role="onSelection" component="M1RD00" interface="introRD00">
    <bindParam name="user" value="adultUser00"/>
  </bind>
  <bind role="start" component="M1RD00" interface="part1RD00"/>
</link>
<link>
  <bind role="onSelection" component="M1RD01" interface="introRD01">
    <bindParam name="user" value="adultUser00"/>
  </bind>
  <bind role="start" component="M1RD01" interface="part1RD01"/>
</link>
<link>
  <bind role="onSelection" component="M1RD00" interface="introRD00">
    <bindParam name="user" value="adultUser01"/>
  </bind>
  <bind role="start" component="M1RD00" interface="part1RD00"/>
</link>
<link>
  <bind role="onSelection" component="M1RD01" interface="introRD01">
    <bindParam name="user" value="adultUser01"/>
  </bind>
  <bind role="start" component="M1RD01" interface="part1RD01"/>
</link>
<link>
  <bind role="onSelection" component="M1RD00" interface="introRD00">
    <bindParam name="user" value="childUser00"/>
  </bind>
```

```
  <bind role="start" component="M1RD00" interface="part2RD00"/>
</link>
<link>
  <bind role="onSelection" component="M1RD01" interface="introRD01">
    <bindParam name="user" value="childUser00"/>
  </bind>
  <bind role="start" component="M1RD01" interface="part2RD01"/>
</link>
<link>
  <bind role="onSelection" component="M1RD00" interface="introRD00">
    <bindParam name="user" value="childUser01"/>
  </bind>
  <bind role="start" component="M1RD00" interface="part2RD00"/>
</link>
<link>
  <bind role="onSelection" component="M1RD01" interface="introRD01">
    <bindParam name="user" value="childUser01"/>
  </bind>
  <bind role="start" component="M1RD01" interface="part2RD01"/>
</link>
```

No código acima, considera-se que dois usuários foram identificados como perfil adulto (adultUser00 e adultUser01) e dois usuários foram identificados como perfil criança (childUser00 e childUser01).

4.3.6. Realidade virtual

Uma máquina de apresentação do dispositivo remoto registrada no TV 3.0 *webservices* usando a classe `ncl-360-vr` deve fornecer suporte de realidade virtual para aplicações NCL na forma de cenas 360. Portanto, na fase de registro, o dispositivo deverá utilizar o seguinte corpo de mensagem para a API de registro.

```
{
   "deviceClass" : "ncl-360-vr",
   "supportedTypes" : ["application/x-ncl360"]
}
```

Quando no estado *Running*, a máquina de apresentação do dispositivo remoto troca mensagens com o Ginga por meio do *WebSocket* criado durante o registro. Essas mensagens contêm metadados para permitir que o Ginga controle a apresentação de cena 360 no dispositivo remoto e a máquina de apresentação do dispositivo remoto para notificar o Ginga sobre transições de eventos que ocorreram durante a apresentação da cena. Essas mensagens de metadados seguem o formato apresentado na Seção 4.3.5.1.

4.3.6.1. Extensão Ginga para cenas 360

Para habilitar a integração entre uma aplicação NCL e uma cena 360, a cena 360 é representada no documento NCL como um novo tipo de mídia. Essa mídia usa o tipo `application/x-ncl360` ou tem a extensão ".ncl360" em seu atributo *src*. Um exemplo de uma especificação desse tipo de mídia é apresentado a seguir.

```
<media id="scene" type="application/x-ncl360" src="*.ncl360">
   <area id="anchorId" label="sceneElmId"/>
</media>
```

Esse novo tipo de mídia é implicitamente associado à classe de dispositivo remoto "ncl-360-vr". Portanto, o código anterior é equivalente ao código a seguir.

```
<media id="scene" type="application/x-ncl360" src="*.ncl360">
   <property name="deviceClass" value="ncl-360-vr"/>
   <area id="anchorId" label="sceneElmId"/>
</media>
```

O comportamento do *parser* de duplicação de mídia e links funciona conforme o esperado para conteúdo executado em um dispositivo remoto e definido na Seção 4.3.5.3.

A mídia NCL identificada por *scene* no exemplo acima corresponde a toda a cena 360 a ser executada em uma máquina de apresentação de dispositivo remoto. Ela segue o comportamento especificado na Seção 4.3.5.2. Além disso, uma mídia do tipo ncl360 deve definir uma âncora para cada elemento da cena 360 a ser controlado pela aplicação NCL. A associação de cada âncora com um elemento de cena é feita pelo atributo *label*, que indica o identificador do elemento de cena 360 correspondente. As ações executadas nas âncoras são transmitidas para o elemento de cena 360 correspondente na máquina de apresentação de dispositivo remoto. As transições de eventos que ocorrem com o elemento de cena 360 também são notificadas pela máquina de apresentação de dispositivo remoto e tratadas em NCL atualizando os estados das âncoras correspondentes.

A máquina de apresentação de dispositivo remoto, em execução em um HMD, é capaz de gerar transições de eventos *View* para cada elemento de cena. O evento *View* segue a máquina de estados de eventos NCL, conforme apresentado na Tabela 4.9.

Tabela 4.9: Comportamento do evento *View*

Transição	**Nome do papel**	**Motivo**
sleeping $\rightarrow$ *occurring (start)*	*onBeginView*	Elemento da cena entra no campo de visão do usuário
occurring $\rightarrow$ *sleeping (stop)*	*onEndView*	Elemento da cena sai do campo de visão do usuário

4.3.6.2. Linguagem NCL 360

Com o objetivo de apresentar uma notação simplificada para a construção de uma cena 360, esta seção apresenta o formato chamado NCL360 (Souza et al., 2020). Este formato foi projetado para simplificar a troca de informações entre uma aplicação NCL tradicional e uma máquina de apresentação de dispositivo remoto da classe *ncl-360-vr*, bem como manter a criação de ambas as aplicações usando conceitos semelhantes. O formato NCL360 herda a definição dos elementos NCL ao mesmo tempo em que define uma estrutura de documento simplificada. Os elementos NCL360 são apresentados na Tabela 4.10.

Tabela 4.10: Elementos da linguagem NCL 360

Módulo NCL360Profile		**API id:** ncl360-profile **Versão atual:** 1.0
Namespace: http://www.ncl.org.br/NCL360/NCL360Profile		
Elemento	**Atributos**	**Conteúdo**
ncl360	id	(head, body)
head		((region | ncl-connectors-causalConnectorFunctionality:causalConnector)*, descriptor+)
body		((ncl-interfaces-compositeNodeInterface:port | link)*, media+)
Módulo NCL360Layout		**API id:** ncl360-layout **Versão atual:** 1.0
Namespace: http://www.ncl.org.br/NCL360/NCL360Layout		
Elemento	**Atributos**	**Conteúdo**
region	id, polar, azimuthal, width, height, radius, zIndex, pin	-
descriptor	id, region, dur, soundType, volume, project	(ncl-presentationSpecification-descriptor:descriptorParam)*
Módulo NCL360Media		**API id:** ncl360-media **Versão atual:** 1.0
Namespace: http://www.ncl.org.br/NCL360/NCL360Media		
Elemento	**Atributos**	**Conteúdo**
media	id, src, descriptor	(area | property)*
area	id, begin, end	-
property	name, value	-
Módulo NCL360Link		**API id:** ncl360-link **Versão atual:** 1.0
Namespace: http://www.ncl.org.br/NCL360/NCL360Link		
Elemento	**Atributos**	**Conteúdo**
link	id, xconnector	(ncl-linking:linkParam*, ncl-linking:bind+)

O elemento principal `<ncl360>` contém toda a definição de uma cena. Assim como no NCL, o elemento `<head>` reúne as características de exibição da cena e as relações genéricas, enquanto o elemento `<body>` reúne os objetos que compõem a cena e as relações entre eles. No cabeçalho de um documento NCL360, são definidos os elementos `<region>`, `<descriptor>` e `<causalConnector>`.

O elemento `<region>` define uma posição no ambiente virtual onde um objeto de mídia será apresentado. O posicionamento é definido usando coordenadas polares em relação à posição inicial do usuário (polar, azimutal). Uma região também permite definir a distância na qual um determinado objeto de mídia será apresentado (raio) e um índice

de sobreposição (zIndex). Finalmente, é possível especificar se o objeto permanecerá fixo em relação ao ambiente ou se moverá junto com a câmera (pino).

O elemento `<descriptor>` define as características de apresentação de uma mídia, incluindo sua posição (região), duração (dur) e características de som (volume), bem como seu formato de som 2D ou 3D (soundType) e o tipo de projeção (projeção) para mídia 360. Alguns valores possíveis para esse atributo são *equirectangular*, *cylindrical-equal-area*, *icosahedron*, *cubemap*, *adjusted-cubemap*, *equiangular-cubemap* e *rotated-sphere*. Um descritor também pode definir que uma mídia deve ser exibida em um canal predefinido usando identificadores predefinidos no atributo *region*. Para o caso de exibição no ambiente como um todo (vídeo 360^o) usando o identificador predefinido *default.sky*.

O elemento `<causalConnector>` segue sua definição NCL 4.0.

Finalmente, no corpo do documento NCL360, os elementos `<media>` definem o conteúdo de uma cena e suas características de apresentação. Partes de um dado objeto de mídia podem ser usadas para propósitos de sincronização definindo um elemento `<area>`. Esse elemento define uma âncora temporal cujos tempos de início e término são definidos em relação à duração da apresentação da mídia. Além disso, os elementos `<property>` devem ser usados para armazenar ou acessar dados relacionados à apresentação da mídia.

Além do conteúdo da cena, o corpo do documento NCL360 pode definir a sincronização entre os componentes da cena. O elemento `<port>` define uma mídia a ser executada quando a cena começa. Mais de uma porta pode ser definida. O elemento `<link>` define uma relação causal entre a apresentação dos componentes da cena e suas âncoras. Para definir um relacionamento causal, um elemento `<link>` herda uma relação genérica definida por um `<causalConnector>` por meio de seu atributo *xconnector*. Diferente do NCL, esse atributo é opcional e pode ser omitido quando o elo usa apenas funções predefinidas. As Tabelas 4.11 e 4.12 resumem a lista de funções predefinidas usadas em condições e ações de elos, respectivamente.

Tabela 4.11: Lista de papeis de condição predefinidas

Papel	**Descrição**
onBegin	A mídia inicia sua apresentação
onEnd	A mídia encerra sua apresentação
onAbort	A mídia aborta sua apresentação
onPause	A mídia pausa sua apresentação
onResume	A mídia retoma sua apresentação
onEndPreparation	A mídia conclui sua preparação
onBeginView	A mídia entra no campo de visão do usuário
onEndView	A mídia sai do campo de visão do usuário

4.3.7. Acesso via NCLua ao TV 3.0 WebServices

A integração entre aplicações NCLua e o TV 3.0 WebServices é viabilizada por meio da classe de eventos *dtvwebservices*, que amplia as capacidades de comunicação dos apli-

Tabela 4.12: Lista de papeis de ação predefinidas

Papel	**Descrição**
start	Inicia a apresentação de uma mídia
stop	Finaliza a apresentação de uma mídia
abort	Aborta a apresentação de uma mídia
pause	Pausa a apresentação de uma mídia
resume	Reinicia a apresentação de uma mídia
prepare	Inicia a preparação de uma mídia

cativos desenvolvidas em NCL 4.0. Essa extensão da API de eventos do Ginga-NCL permite que aplicações NCLua atuem como clientes locais dos serviços oferecidos pelo TV 3.0 WebServices, utilizando uma interface assíncrona padronizada para requisições e respostas. Com isso, os aplicativos podem consumir dados dinâmicos, interagir com recursos da TV 3.0 e reagir a notificações, de forma compatível com o modelo orientado a eventos do ambiente NCLua.

O uso da classe *dtvwebservices* inicia-se com o envio de um evento do tipo *request* pelo aplicativo, definindo a operação desejada (e.g. *get*, *post* ou emphdelete), a API identificada por seu API id, e opcionalmente o endpoint, cabeçalhos, corpo da mensagem e um identificador de sessão. Essa sessão pode ser usada pelo aplicativo para correlacionar respostas ou cancelar solicitações em andamento. A construção do corpo da requisição pode ser feita de maneira flexível em Lua, usando tabelas, strings ou buffers, com conversão automática para JSON sempre que apropriado.

```
evt = { class='dtvwebservices', type='request', method: string, api:
    string [, endpoint: string] [, session: object] [, headers: table]
    [, body: (table | string | buffer)] [, timeout: number] }
```

A resposta à solicitação é entregue à aplicação como um evento do tipo *response*, contendo o código HTTP da resposta, os cabeçalhos, o corpo da resposta (convertido para uma estrutura Lua conforme seu tipo declarado) e um campo *finished* que indica o fim da transmissão. Em caso de erro de comunicação, o campo *error* é preenchido, permitindo o tratamento apropriado pela aplicação. A correspondência entre solicitações e respostas é feita por meio do identificador de sessão, se fornecido.

```
evt = { class='dtvwebservices', type='response', response_data_type:
    string, method: string, api: string [, endpoint: string], code:
    number [, session: object], headers: table [, body: (table | string
     | buffer)] [, finished: boolean] [, error: string] }
```

Além das respostas imediatas, certas APIs do TV 3.0 WebServices retornam um *handle* para comunicação assíncrona futura. Nesse caso, a aplicação NCLua recebe novos eventos do tipo *webservicesevent*, contendo dados adicionais relacionados à operação iniciada anteriormente. Essa funcionalidade é particularmente útil para serviços que envolvem longas execuções ou fluxos de dados contínuos, permitindo que o aplicativo reaja progressivamente a novos dados recebidos sem bloquear sua execução principal.

```
evt = { class='dtvwebservices', type='webservicesevent', handle: handle
    , data: buffer [, error: string] }
```

Um aplicativo pode cancelar uma requisição ativa usando um evento do tipo *cancel*, que referencia a sessão da solicitação original. Esse mecanismo oferece à aplicação controle sobre a duração e relevância de cada interação com o WebServices.

Por fim, vale ressaltar que aplicativos NCLua são considerados clientes locais pelo TV 3.0 WebServices e, portanto, não devem utilizar mecanismos de autenticação como *tokens* de acesso ou chamadas às APIs de pareamento. Além disso, a utilização da classe *dtvwebservices* deve ser cuidadosamente avaliada conforme a API desejada, pois, em alguns casos, podem existir alternativas equivalentes mais simples e eficientes implementadas diretamente na API NCL/NCLua.

4.4. TV 3.0 - Ginga-HTML5 e suas novas APIs

Com o avanço das capacidades de execução de aplicativos na TV 3.0, a máquina de apresentação Ginga-HTML5 foi concebida para oferecer um ambiente compatível e interoperável para aplicativos baseados em tecnologias web. A especificação adota como referência principal a norma CTA-5000-F (2023), que define um conjunto mínimo e interoperável de recursos HTML5, CSS e JavaScript para televisores conectados. O objetivo da adoção desse perfil é garantir que os *Broadcaster Applications* possam ser executados com previsibilidade e desempenho consistente em diferentes modelos de receptores.

A especificação CTA-5000-F é voltada para garantir a interoperabilidade entre navegadores embutidos em dispositivos de consumo — como *Smart TVs* e *set-top boxes* — e define uma base comum alinhada com as principais demandas de aplicativos em torno do audiovisual. O alinhamento da TV 3.0 com essa especificação internacional contribui para reduzir a fragmentação entre plataformas e facilita a adaptação de conteúdos desenvolvidos originalmente para o ambiente da web.

O TV 3.0 Ginga-HTML5 deve se comportar como um User Agent HTML5 conforme os requisitos normativos da CTA-5000-F. Isso inclui o suporte a recursos multimídia, modelagem gráfica avançada e comunicação com servidores remotos, bem como integração com as demais camadas da arquitetura de TV 3.0. Entre as principais APIs JavaScript suportadas no ambiente Ginga-HTML5 da TV 3.0, destacam-se:

- MediaSource e MediaCapabilities, para gerência e adaptação de fluxos de mídia;
- Web Audio API, para manipulação avançada de áudio;
- Canvas e WebGL, para renderização gráfica bidimensional e tridimensional;
- Storage e IndexedDB, para armazenamento local persistente;
- Fetch, XMLHttpRequest e WebSocket, para comunicação assíncrona;
- Fullscreen API e Picture-in-Picture, para controle de exibição de vídeo;
- Device e Screen Orientation, para detecção de movimento e orientação da tela;
- Encrypted Media Extensions (EME) e Media Session API, para controle de reprodução e interoperabilidade com DRM.

4.5. TV 3.0 - WebServices e suas novas APIs

A seguir, são apresentadas as principais APIs do TV 3.0 WebServices que estendem a capacidade de interação de aplicações com os serviços de TV 3.0 e o receptor. As APIs

seguem uma abordagem padronizada e segura, permitindo que aplicações locais e, quando autorizado, clientes na mesma rede local, interajam com o sistema. As subseções seguintes descrevem, com base na norma, as funcionalidades de notificação de eventos, acesso a fluxos de acessibilidade, manipulação da janela de língua de sinais, geolocalização, entrega de alertas de emergência e outros mecanismos relevantes para a personalização e ampliação da experiência de uso na TV 3.0.

4.5.1. API atualizada de players de mídia

As APIs de players de mídia do TV 3.0 WebServices evoluem o modelo introduzido no Ginga da TV 2.5, mantendo compatibilidade com operações fundamentais — como inicialização, controle e consulta de estado — e incorporando importantes extensões funcionais. Entre os principais avanços, destaca-se a identificação do *Persistent Media Player* (com id = 1), que viabiliza continuidade na reprodução de conteúdo ao alternar entre diferentes contextos de aplicação (ver Seção 4.2).

A especificação foi expandida para suportar áudio imersivo MPEG-H, por meio do objeto *ngaInteractivity*, que descreve cenas auditivas e permite à aplicação configurar *presets*, objetos sonoros e grupos de alternância, controlando atributos como azimute, elevação, proeminência e silenciamento. Essa estrutura permite experiências auditivas personalizadas, inclusive com foco em acessibilidade e preferência do telespectador.

Além disso, a API de players de mídia está integrada à API unificada de notificações de eventos, permitindo à aplicação receber eventos como início, pausa, conclusão, mudança de taxa de reprodução e alterações na cena ou configuração do áudio imersivo. Isso possibilita um controle refinado da experiência do telespectador em tempo real, alinhando a reprodução de mídia a sincronizações e opções interativas.

4.5.2. API de geolocalização

A API de geolocalização do TV 3.0 WebServices oferece uma interface padronizada para que aplicações possam obter a posição geográfica do receptor de TV, com suporte a múltiplas fontes de localização. Essa flexibilidade permite adaptar o nível de precisão e confiabilidade dos dados de acordo com o contexto de uso, seja para serviços personalizados, recomendações regionais ou entrega de conteúdos localizados.

A interface é acessada por meio de uma requisição HTTP GET ao *endpoint* /tv3/geolocation, com um parâmetro opcional *source*. Esse parâmetro define a fonte de geolocalização a ser utilizada, podendo assumir os seguintes valores: gnss (sistemas de satélite), wifi (posicionamento por redes Wi-Fi), mobileNetwork (redes móveis), zip (código postal registrado), tbsat (área de cobertura segundo a Tabela do Melhor Servidor da Transmissão Terrestre) e ip (localização aproximada por IP). Caso o parâmetro *source* não seja especificado, o sistema seleciona a fonte mais precisa e confiável disponível.

A resposta da API é formatada em JSON e inclui campos como latitude, longitude, precisão horizontal e vertical, altitude e uma marca de tempo indicando o momento da aquisição da posição. A inclusão do campo *source* na resposta permite que a aplicação saiba qual foi a origem real dos dados recebidos.

Entre os cuidados da especificação estão as mensagens de erro apropriadas para

fontes inválidas, chamadas não autorizadas ou falhas de comunicação com serviços externos. A API está limitada ao serviço de TV 3.0 atualmente selecionado, o que garante o vínculo entre os dados de geolocalização e o perfil de uso ativo do receptor.

Essa funcionalidade é essencial para aplicações que dependem do contexto geográfico do usuário e está preparada para ser utilizada em conformidade com as diretrizes de privacidade e consentimento estabelecidas pela TV 3.0.

4.5.3. API unificada de notificações de eventos

O TV 3.0 WebServices introduz uma API unificada para o gerenciamento de notificações assíncronas, projetada para permitir que clientes se registrem para receber eventos gerados dinamicamente pelo ambiente de execução da TV 3.0. Essa abordagem foi inspirada diretamente no modelo de eventos da API NCLua, oferecendo uma estrutura coerente e extensível para lidar com diferentes tipos de notificações de forma padronizada.

A API de registro é acessada via requisição GET ao *endpoint* /tv3/notifications/<class-name>, onde <class-name> identifica a classe de evento desejada. Uma chamada bem-sucedida retorna um identificador único (*handle*) e uma URL de *WebSocket* pela qual as notificações serão transmitidas. A interface permite que múltiplos eventos sejam emitidos ao longo do tempo para um único registro, com suporte a parâmetros de consulta adicionais definidos por cada classe de evento.

A comunicação via *WebSocket* permite menor latência e entrega contínua de notificações, essenciais para funcionalidades como acompanhamento de estado de players de mídia, alertas de emergência, substituição de conteúdo, resolução de Xlinks ou acompanhamento de fluxos DASH. As classes de eventos incluem, por exemplo: media-player, emergency-warning, dash-stream, xlink-resolution e content-replaced.

Para encerrar o interesse em uma classe de eventos e lberar recursos, o cliente utiliza o *endpoint* DELETE /tv3/notifications/<handle>, onde o *handle* é o identificador retornado durante o registro. A operação remove o vínculo entre o cliente e a classe de eventos correspondente, interrompendo a transmissão de notificações subsequentes.

4.5.4. API de controle de sessões de coleta de dados de audiência

O TV 3.0 WebServices introduz um conjunto de APIs dedicadas à criação, controle e monitoramento de sessões de coleta de dados de audiência, conforme introduzido na Seção 4.2.5. Essas sessões de coleta são sempre associadas ao serviço de TV 3.0 atualmente em exibição e podem ser configuradas com parâmetros específicos que definem o escopo, a duração e os critérios de entrega dos relatórios gerados.

A criação de uma nova sessão de coleta é feita por meio de uma requisição POST à API /tv3/current-service/audience-measurement, contendo campos como *campaignId*, emphdeliveryURLs, e *filterConfig*, que determinam o identificador da campanha, os pontos de entrega dos dados e os tipos de informações a serem incluídas no relatório, respectivamente. A sessão pode ainda ser vinculada a um conteúdo específico, ter horário de início agendado ou permitir entregas parciais e progressivas ao longo da execução.

Durante a sessão, a emissora pode realizar ações como pausar, retomar, abortar ou finalizar a coleta, utilizando o mesmo *endpoint* com diferentes valores no campo *action*

e informando o identificador *handle*, fornecido pela plataforma no momento da criação da sessão. Essas ações permitem flexibilidade no ciclo de vida da coleta, como reagir dinamicamente a mudanças em campanhas ou mesmo na programação.

Além da criação e controle, a especificação oferece uma API complementar baseada em requisições GET ao *endpoint* /tv3/current-service/audience-measurement/<handle>, permitindo consultar o status de uma sessão ativa ou encerrada. A resposta dessa API inclui informações detalhadas como o estado atual da sessão (ex.: *collecting*, *paused*, *finished*), horários de início e término, política de entrega de relatórios e identificadores únicos tanto da sessão quanto do relatório.

4.5.5. API de suporte a múltiplos dispositivos

A comunicação de um dispositivo remoto com o *middleware* Ginga é estabelecida via TV 3.0 *WebServices*. Para esse fim, APIs de comunicação de dispositivo remoto são fornecidas para permitir que um dispositivo conectado à mesma rede local que o receptor se registre como um dispositivo remoto e se comunique com clientes locais. A Figura 4.12 descreve as etapas para a comunicação entre TV 3.0 *WebServices* e um cliente não local. Após sua inicialização, na fase de descoberta, o *webservice* do dispositivo remoto (RDWS) obtém a localização do receptor na rede local usando o padrão SSDP.

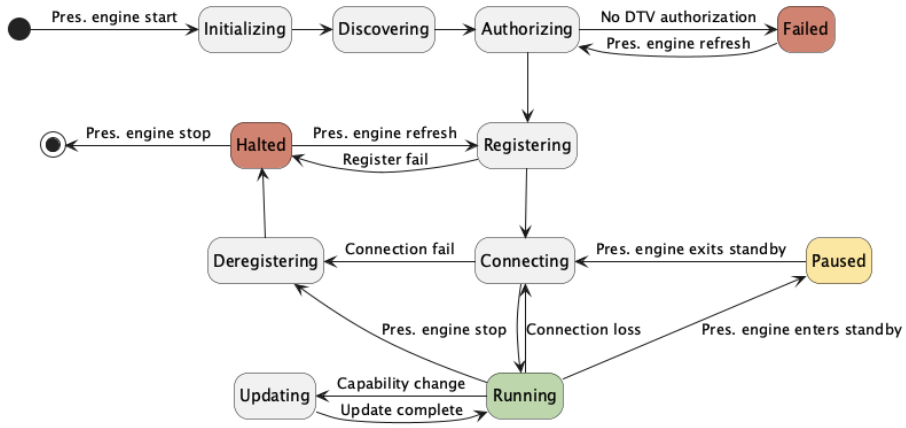

Figura 4.12: Etapas de comunicação do *webservice* do dispositivo remoto

Na etapa *Authorizing*, o RDWS obtém o *AccessToken* do receptor e estabelece um vínculo com o receptor. Dado que a autorização do usuário é necessária no receptor, se nenhuma autorização for concedida, o RDWS vai para a etapa *Failed*. Ele tentará novamente obter o *AccessToken* quando acionado pelo usuário do dispositivo remoto.

Uma vez autorizado, na etapa *Registering*, o RDWS faz uma requisição `POST` para a API de registro. A Tabela 4.13 apresenta detalhes da rota para registrar dispositivos remotos. Uma vez registrado, no estágio *Connecting*, o RDWS se conecta ao ponto de entrada do WebSocket. Se a conexão falhar, ele vai para a etapa *Deregistering*. Caso contrário, ele vai para a etapa *Running*.

Na etapa *Deregistering*, o RDWS cancela seu registro junto ao *webservice* da TV 3.0, liberando o *WebSocket* criado anteriormente. A Tabela 4.14 apresenta detalhes da rota para cancelar o registro de dispositivos remotos. Um cancelamento de registro

também pode ser acionado quando o mecanismo de apresentação do dispositivo remoto para. Da etapa *Deregistering*, ele vai para o estágio *Halted*, onde pode reiniciar a etapa de registro quando acionado pelo usuário do dispositivo remoto ou encerrar a execução caso a máquina de apresentação do dispositivo remoto pare.

Na etapa *Running*, a máquina de apresentação do dispositivo remoto executa de acordo com as mensagens de metadados recebidas do Ginga. O formato dessas mensagens é definido por cada classe de dispositivo remoto apresentado na Seção 4.3. Da etapa *Running*, ele pode voltar para a etapa *Connecting* em caso de perda de conexão. Ele pode ir para a etapa *Paused* caso o dispositivo entre em *stand-by*. Se uma alteração nas capacidades do dispositivo remoto acontecer durante a execução da apresentação do nó na máquina de apresentação do dispositivo remoto, o RDWS deve ir para a etapa *Updating*, onde atualiza seu registro de acordo com a Tabela 4.13.

Tabela 4.13: Especificação da API de Registro de Dispositivo Remoto

Formato da Requisição:	`http(s)://<host>/tv3/remote-device/[<handle>]`
Tipo de Operação:	POST
Descrição:	Permite que um dispositivo no ambiente doméstico e conectado à mesma rede local que o receptor se registre ou atualize seu registro como um dispositivo remoto. Caso um <handle> existente seja fornecido, a rota atualiza um registro existente. Caso contrário, ela abre um servidor *websocket* em uma porta atribuída dinamicamente e retorna as informações necessárias para que o dispositivo remoto possa trocar mensagens com uma aplicação Ginga.
Parâmetros da Consulta:	-

Corpo da Mensagem:	O corpo da requisição é um objeto JSON, de acordo com a estrutura abaixo:
	`{`
	`"deviceClass" : "<class_id>",`
	`"supportedTypes" : ["mimeTypes"],`
	`}`
	Onde "deviceClass" é um identificador para a classe do cliente não local que está se registrando como um dispositivo remoto. As classes de dispositivos remotos são definidas na Seção 4.3. O vetor "supportedTypes" indica os tipos MIME dos nós que o dispositivo remoto é capaz de executar. Os tipos MIME a serem considerados são aqueles suportados pelo Ginga.

Exemplo/Formato de Resposta:	O retorno é um objeto JSON, conforme a estrutura abaixo:
	`{`
	`"handle": "<handle>",`
	`"url": "wss://<entry-point>"`
	`}`
	Onde "handle" é um identificador gerado pela implementação do TV 3.0 *WebServices*, que deve ser usado para posteriormente liberar este recurso através da API de desregistro ou atualizar um registro.
	Ao se conectar ao *entry-point* do *WebSocket* indicado em "url" com um cliente *websocket*, o dispositivo remoto poderá trocar mensagens de metadados com uma aplicação Ginga.
	No caso de uma atualização, o retorno é um objeto JSON com o identificador, indicando a atualização do registro:
	`{`
	`"handle": "<handle>"`
	`}`

Possíveis códigos de erro:	• 107: Se invocado de um cliente não local ou cliente *stand-alone* local, e o *accessToken* for omitido, inválido ou tiver expirado. • 200: Se a solicitação exceder o número de dispositivos que podem ser registrados na plataforma. • 300: Se a função DTV não estiver em uso no receptor. • 302: Se não houver recepção de sinal DTV no momento. • 305: Se <*handle*> não corresponder a um dispositivo registrado existente.
Restrições:	-
Requisitos de segurança:	-
Observação:	O número de clientes não locais que podem se conectar à plataforma para esta funcionalidade fica a critério da implementação do receptor TV 3.0.

Tabela 4.14: Especificação da API de cancelamento de registro de dispositivo remoto

Formato da Requisição:	`http(s)://<host>/tv3/remote-device/[<handle>]`
Tipo de Operação:	DELETE
Descrição:	Cancela o registro de um dispositivo registrado anteriormente.

Parâmetros da Consulta:	-
Corpo da Mensagem:	-

Exemplo/Formato de Resposta:	Se bem-sucedido, o retorno será um objeto JSON vazio, indicando o cancelamento do registro:
	{
	}

Possíveis códigos de erro:	• 105: Se <*handle*> não for especificado. • 107: Se *accessToken* for omitido, inválido ou tiver expirado. • 300: Se a função DTV não estiver em uso no receptor. • 302: Se não houver recepção de sinal DTV no momento. • 305: Se <*handle*> não corresponder a um dispositivo registrado existente.
Restrições:	-
Requisitos de segurança:	-
Observação:	-

4.5.5.1. Mensagens de Metadados

O RDWS deve continuar na etapa *Running* durante toda a execução da máquina de apresentação do dispositivo remoto. Nesse ponto, ele segue o comportamento esperado da máquina de apresentação do dispositivo remoto, conforme definido para cada classe de dispositivo remoto.

Enquanto o RDWS está na etapa *Running*, ele troca mensagens com o Ginga por meio do *WebSocket* criado durante o registro. Essas mensagens contêm metadados para permitir que o Ginga controle e receba notificações sobre a máquina de apresentação do dispositivo remoto. As Tabelas 4.15 e 4.16 detalham o formato de duas dessas mensagens.

O comportamento esperado da máquina de apresentação do dispositivo remoto ao receber a mensagens de metadados deve ser definido para cada classe de dispositivo remoto. No entanto, o comportamento básico de uma máquina de apresentação de dispositivo remoto considerando as mensagens nas Tabelas 4.15 e 4.16 é como segue.

A qualquer momento após o RDWS atingir a etapa *Running*, o Ginga pode enviar uma mensagem para solicitar capacidades da máquina de apresentação de dispositivo remoto. Essa mensagem deve ser imediatamente respondida com os valores para essas capacidades. Além disso, quando julgar necessário, a máquina de apresentação de dispositivo remoto pode enviar uma mensagem para informar os valores das capacidades.

Tabela 4.15: Mensagem para consultar capacidades do dispositivo

Sentido de Mensagem:	Do Ginga para o dispositivo remoto

Corpo da Mensagem:	`{`
	`  "type" : <mimeType>,`
	`  "capabilities" : [`
	`  {"name" : <cName>},`
	`  ]`
	`}`

Descrição:	Permite que Ginga consulte o valor de uma ou mais capacidades do dispositivo remoto relacionados a um tipo de nó que é controlado por um dispositivo remoto, como sua duração de preparação. "cName" é o nome da capacidade a ser consultada. A lista "capabilities" carrega os nomes das capacidades a serem consultadas. Se a capacidade for independente de um tipo de nó, o atributo "type" pode ser omitido.
Nota:	-

Tabela 4.16: Mensagem para informar capacidades do dispositivo

Sentido de Mensagem:	Do dispositivo remoto para o Ginga

Corpo da Mensagem:	`{`
	`  "type" : <mimeType>,`
	`  "capabilities" : [`
	`  {"name" : <cName>, "value" : <cValue>},`
	`  ]`
	`}`

Descrição:	Permite que o dispositivo remoto informe ao Ginga o valor de uma ou mais capacidades do dispositivo remoto relacionadas a um tipo de nó que é controlado por um dispositivo remoto. No caso de uma mensagem anterior do Ginga solicitando uma lista de valores de capacidades, o dispositivo deve responder com os valores de todas as capacidades. O vetor "capabilities" carrega o par nome-valor das capacidades a serem informadas. Se a capacidade for independente de um tipo de nó, o atributo "type" pode ser omitido.
Nota:	-

Vale notar que durante sua execução, a máquina de apresentação de dispositivo remoto pode requerer acesso a arquivos enviados juntamente com uma aplicação Ginga ou acessar informações sobre os usuários registrados no receptor de TV. Essas informações são acessíveis por meio de APIs específicas que requerem *bind-tokens* para o provedor de serviço atual. A obtenção de um *bind-token* deve ser feita diretamente com os provedores de serviço.

4.5.6. Acessibilidade avançada

A especificação do TV 3.0 WebServices define APIs para controle, entrega e personalização de recursos de acessibilidade, com destaque para conteúdos em língua de sinais, audiodescrição e legendas. Essas interfaces permitem manipulação explícita da janela de exibição de sinais, acesso individualizado a diferentes fluxos de mídia e controle sobre os elementos visuais envolvidos na apresentação.

A personalização da janela de língua de sinais é granular e controlada por meio do *endpoint* /tv3/current-service/signlanguage, com operações realizadas via requisições GET e POST. A interface permite o ajuste da posição e dimensões relativas da janela, configuração da cor de fundo e definição de sobreposição com o conteúdo principal. Quando o modo de exibição se dá por avatar digital, a API permite parâmetros adicionais de customização, como atributos de aparência (cabelo, pele, olhos, vestuário) e inclusão de logotipos em formato SVG. Tais atributos podem ser ajustados dinamicamente, com base nas capacidades declaradas pela plataforma e nas preferências do perfil do telespectador.

A operação da janela de sinais suporta ações como ativação, pausa e interrupção, que podem ser combinadas com comandos de reposicionamento ou personalização visual. O estado atual da janela pode ser consultado a qualquer momento, incluindo os parâmetros de reprodução, sobreposição, taxas de quadros e avatar em uso.

Já o acesso a fluxos específicos de mídia sinalizados por radiodifusão é realizado por meio do *endpoint* /tv3/current-service/stream/<alias>, onde *alias* pode assumir valores como *videosignlanguage*, *glosssignlanguage*, *motionsignlanguage*, *captions*, *audiodescription*, entre outros definidos pela norma. Para fluxos baseados em texto, como glosas e legendas, os dados são transmitidos via *WebSocket*. Já os fluxos de vídeo e áudio acessíveis são entregues por meio de URLs dinâmicas no formato HTTP DASH. O parâmetro opcional *stream-id* pode ser utilizado para selecionar uma instância específica do fluxo, quando múltiplos estiverem disponíveis para um mesmo tipo de mídia.

Esse modelo de acesso individual a fluxos permite que dispositivos externos, como *smartphones*, recebam e processem conteúdos acessíveis de forma independente. Isso viabiliza, por exemplo, que uma pessoa surda visualize a janela de língua de sinais em seu próprio dispositivo, ajustada ao seu campo de visão e preferências de avatar, ou que uma pessoa cega escute a audiodescrição diretamente em seus fones de ouvido. A entrega personalizada dos recursos de acessibilidade em dispositivos pessoais contribui significativamente para a autonomia, conforto e inclusão de telespectadores com deficiência, respeitando suas demandas individuais de usabilidade.

4.5.7. Alertas de emergência

A TV 3.0 incorpora suporte estruturado a alertas de emergência, conforme definido na norma ABNT NBR 25607 (a ser publicada). A especificação permite a emissão de mensagens por radiodifusão, além de oferecer monitoramento alternativo via redes IP, conforme parametrizado no Broadcast Application Metadata (BAM). Por meio do elemento XML emergencyWarningTransmission, a emissora pode sinalizar a capacidade de emitir alertas. Opcionalmente, pode fornecer uma URL OTT (@url) para fallback em situações quando a recepção do sinal de radiodifusão não seja possível.

As mensagens de alerta são estruturadas no formato AEA (*Advanced Emergency Alert*) e encapsuladas em fragmentos AEAT (*AEA Table*), em XML. Tais mensagens podem ser exibidas por uma aplicação nativa obrigatória do receptor de TV 3.0, que opera em primeiro plano, sobrepondo qualquer conteúdo audiovisual ativo, ou, quando explicitamente sinalizado, por um *Broadcaster Application* que assuma esse papel, integrando a apresentação do alerta à sua interface.

Para acesso programático, a API unificada de eventos define a classe *emergency-warning*, por meio da qual *Broadcaster Applications* registrados como ouvintes recebem notificações assíncronas. As mensagens incluem os campos *eventType* (tipos *alert*, *update* ou *cancel*), a carga útil nos formatos AEAT ou CAP, e o campo opcional *managerTargetApp*, que fornece um *deep link* para a aplicação responsável pela renderização do alerta. A aplicação padrão é restaurada sempre que o serviço de TV 3.0 for encerrado.

Adicionalmente, a API de acesso a streams individuais permite recuperar o conteúdo de alertas por meio do *endpoint* /tv3/current-service/stream/emergency. Este *endpoint* retorna uma URL *WebSocket* por onde são transmitidos, periodicamente, os documentos AEAT ou CAP. Esse mecanismo habilita a apresentação de alertas também em dispositivos que atuem como clientes não-locais junto ao TV 3.0 WebServices, reforçando a acessibilidade e a abrangência do sistema de notificação.

4.6. Conclusão

A evolução da Camada de Codificação de Aplicações da TV 3.0 representa uma mudança de paradigma na arquitetura de sistemas televisivos, substituindo o modelo centrado em transmissões lineares e canais por uma plataforma orientada a aplicativos, extensível e interoperável. As especificações discutidas neste capítulo — abrangendo o Ginga-NCL, o novo perfil Ginga-HTML5 e o conjunto de APIs do TV 3.0 WebServices — demonstram o amadurecimento do ecossistema normativo nacional em direção a uma infraestrutura capaz de acomodar inovação contínua, personalização da experiência do usuário, integração multimodal e entrega multiplataforma de conteúdo.

As extensões do Ginga-NCL ampliam significativamente as capacidades declarativas do *middleware*, incorporando suporte a múltiplos usuários, efeitos sensoriais, perfis personalizados e novos dispositivos de interação. Em paralelo, a introdução do Ginga-HTML5 aproxima o desenvolvimento de aplicações televisivas das práticas e ferramentas adotadas na Web moderna, facilitando a entrada de novos atores no ecossistema. Complementarmente, o TV 3.0 WebServices provê uma base uniforme e agnóstica de protocolos para comunicação entre aplicações e a infraestrutura do receptor, contemplando desde personalização de perfis e controle de dispositivos até funcionalidades críticas como geolocalização, acessibilidade e recepção de alertas de emergência.

A diversidade de APIs descritas reflete o compromisso com requisitos como acessibilidade avançada, imersão e segmentação de conteúdo, fundamentais para atender aos anseios da radiodifusão aberta. Ao mesmo tempo, o desenho modular e documentado dessas interfaces abre espaço para validação científica, implementação incremental e eventual adoção por outros países. A série de normas ABNT NBR 25600 (a ser publicada) consolida esse avanço, posicionando o Brasil entre os poucos países com um sistema nacional de TV Digital em constante evolução e com respaldo técnico-acadêmico robusto.

As perspectivas futuras incluem a consolidação de ambientes de testes e certificação, o desenvolvimento de ferramentas de apoio ao ciclo de desenvolvimento dos aplicativos e a articulação com modelos de negócio viáveis para emissoras e desenvolvedores. A comunidade científica tem um papel essencial nesse cenário, tanto na produção de conhecimento que alimente o aprimoramento das especificações, quanto na formação de profissionais capazes de construir as experiências da próxima geração televisiva.

Referências

ABNT NBR 15606-10 (2023). Televisão digital terrestre - codificação de dados e especificações de transmissão para radiofusão digital parte 10: Ginga-html5 – especificação do perfil html5 no ginga. Norma técnica, ABNT, São Paulo, BR.

ABNT NBR 15606-2 (2023). Televisão digital terrestre - codificação de dados e especificações de transmissão para radiofusão digital parte 2: Ginga-ncl para receptores fixos e móveis - linguagem de aplicação xml para codificação de aplicações. Norma técnica, ABNT, São Paulo, BR.

Barreto, F., Abreu, R. S., Josué, M. I. P., Montevecchi, E. B. B., Valentim, P., and Muchaluat-Saade, D. C. (2024). Providing multimodal and multi-user interactions for digital tv applications. *Multimedia Tools and Applications*, pages 1–26.

Barreto, F., Abreu, R. S., Montevecchi, E. B. B., Josué, M. I. P., Valentim, P., and Muchaluat-Saade, D. C. (2020). Extending ginga-ncl to specify multimodal interactions with multiple users. In *WebMedia '20: Brazillian Symposium on Multimedia and the Web*. SBC.

Barreto, F., Abreu, R. S., and Muchaluat-Saade, D. C. (2023). Tv 3.0: Interação multiusuário para tv digital aberta com ncl 4.0. In *Workshop Futuro da TV Digital Interativa - Brazilian Symposium on Multimedia Systems and The Web - WebMedia 2023*. SBC.

Barreto, F., Abreu, R. S., Santos, J. A. F. d., and Muchaluat-Saade, D. C. (2019a). Authoring sensory effects in ncl. In *Workshop Futuro da TV Digital Interativa - Brazilian Symposium on Multimedia Systems and The Web - WebMedia 2019*. SBC.

Barreto, F., Montevecchi, E. B. B., Abreu, R. S., Santos, J. A. F. d., and Muchaluat-Saade, D. C. (2019b). Providing multimodal user interaction in ncl. In *Workshop Futuro da TV Digital Interativa - Brazilian Symposium on Multimedia Systems and The Web - WebMedia 2019*. SBC.

Brasil (2023). Decreto nº 11.484, de 6 de abril de 2023, Presidência da República. Diário Oficial da União, p. 13, 06/04/2023. Disponível em `https://www.planalto.gov.br/ccivil_03/_ato2023-2026/2023/decreto/d11484.htm`.

CTA-5000-F (2023). Web media api snapshot 2023. Especificação cta, CTA.

dos Santos, J., Vieira, R., Josué, M. I., Oliveira, K. S., and Muchaluat-Saade, D. C. (2024). Multidevice support in the next generation of the brazilian terrestrial tv system. In *IMXw '24: Proceedings of the 2024 ACM International Conference on Interactive Media Experiences Workshop*. ACM.

Fórum SBTVD (2020a). Tv 3.0 project. Website. Disponível em `https://forumsbtvd.org.br/tv3_0`.

Fórum SBTVD (2020b). Tv 3.0 project - call for proposals. Chamada pública, Fórum SBTVD, São Paulo, BR. Disponível em `https://forumsbtvd.org.br/wp-content/uploads/2020/07/SBTVDTV-3-0-CfP.pdf`.

Fórum SBTVD (2021). Tv 3.0 project phase 2 - results. Relatório de avaliação de resultados, Fórum SBTVD, São Paulo, BR. Disponível em `https://forumsbtvd.org.br/tv3_0/#panel-phase2`.

ISO/IEC 23005-3:2019 (2019). Information technology — media context and control - part 3: Sensory information. Norma técnica, ISO/IEC, Geneva, SW.

ISO/IEC 27560 (2023). Privacy technologies — consent record information structure. Norma técnica, ISO/IEC, Geneva, SW.

ITU-R BT2075-1 (2017). Integrated broadcast-broadband system. Recomendação uit-r, International Telecommunication Union, Geneva, CH.

ITU-T H.761 (2009). Nested context language (ncl) and ginga-ncl. Standard, International Telecommunication Union, Geneva, CH.

Ivanov, M., Moreno, M. F., and Muchaluat-Saade, D. C. (2024). Automatic preparation of sensory effects. In *Proceedings of MMSys '24: ACM Multimedia Systems Conference 2024*. ACM.

Josué, M. I. P., Valentim, P. A., and Muchaluat-Saade, D. C. (2023). Tv 3.0: Definição e uso de perfil de telespectador no ambiente de tv digital aberta. In *Workshop Futuro da TV Digital Interativa - Brazilian Symposium on Multimedia Systems and The Web - WebMedia 2023*. SBC.

Moreno, M., Pernisa Júnior, C., Barrere, E., Teixeira, S., Turnes Montezano, C., Shuen Sousa, L.-C., Soares Neto, C., Muchaluat-Saade, D. C., Josué, I., M., dos Santos, J., Colcher, S., Moraes, D., Omaia, D., Araújo, T., and Lemos, G. (2023). R&d progress on tv 3.0 application coding layer. *SET INTERNATIONAL JOURNAL OF BROADCAST ENGINEERING*, pages 9–21.

Souza, G., Silva, D., Delgado, M., Rodrigues, R., Mendes, P. R. C., Amorim, G. F., Guedes, L. V., and dos Santos, J. A. F. d. (2020). Interactive 360-degree videos in ginga-ncl using head-mounted-displays as second screen devices. In *WebMedia '20: Brazillian Symposium on Multimedia and the Web*. SBC.

W3C (2024a). Data privacy vocabulary (dpv). Final w3c community report, W3C.

W3C (2024b). Personal data categories (pd). Final w3c community report, W3C.