Integração de Dados JSON em Aplicações Ginga-NCL para Alerta de Pessoas Desaparecidas

Alice Lima Soares
alsoares086@gmail.com
Instituto Federal do Acre
Rio Branco, Acre

Marcos Vinícios Mesquita de Moraes
marcos646362@gmail.com
Instituto Federal do Acre
Rio Branco, Acre

Flávio Miranda de Farias
fmflavio@gmail.com
Instituto Federal do Acre
Rio Branco, Acre

ABSTRACT

Introduction: This paper explores the use of JSON for dynamic content management in interactive Digital TV applications developed with the Ginga-NCL middleware and NCLua scripting language ***Objective:*** The proposed application aims to simulate a missing persons alert system, inspired by the Amber Alert Brazil initiative. ***Methodology:*** The system's architecture leverages Ginga-NCL's modularity to facilitate maintenance and future scalability. Using JSON as the data format allows efficient, real-time updates of content, improving the responsiveness of the application. under the Brazilian standard. ***Results:*** The successful development of a functional prototype demonstrates the practicality of integrating JSON-based dynamic content in Digital TV applications, reinforcing its potential for public utility services

KEYWORDS

JSON, NCL, ginga, NCLua, TV Digital, pessoas desaparecidas

1 INTRODUÇÃO

A televisão (TV) continua sendo um elemento central na vida dos brasileiros, presente em 88% dos domicílios com acesso a sinais abertos ou por assinatura, conforme dados recentes do IBGE [7]. Sua amplitude em todas as regiões do país mantém a TV como um meio estratégico para a disseminação de informações, inclusive em locais com conectividade limitada à internet.

Nesse cenário, o Sistema Brasileiro de Televisão Digital (SBTVD) adotou como padrão de interatividade o *middleware* Ginga[1]. Ele utiliza a linguagem declarativa NCL (*Nested Context Language*) para a criação de aplicações interativas. Apesar da padronização e flexibilidade, o desenvolvimento de aplicações em NCL ainda apresenta barreiras significativas, como a escassez de materiais didáticos acessíveis, a exigência de ambientes específicos de simulação e a pouca familiaridade dos desenvolvedores com o paradigma declarativo [13].

Para atenuar essas dificuldades, é proposta a integração do NCL com formatos de dados amplamente conhecidos, como o JSON (*JavaScript Object Notation*)[2], facilitando a manipulação de informações estruturadas. Essa integração é viabilizada por meio do uso do NCLua[3], uma extensão da linguagem Lua incorporada ao Ginga, que permite o acesso dinâmico a dados externos e a atualização do conteúdo da aplicação em tempo de execução. Outro ponto a ser observado é que essa abordagem permite que desenvolvedores trabalhem com dados em um formato familiar e utilizem a lógica imperativa do NCLua, reduzindo a curva de aprendizado associada ao paradigma puramente declarativo do NCL.

Este trabalho demonstra a viabilidade de integrar dados estruturados em formato JSON a aplicações interativas desenvolvidas em Ginga-NCL, utilizando NCLua para leitura e atualização dinâmica das informações. Como estudo de caso,foi desenvolvida uma interface de TV Digital voltada à divulgação de alertas sobre pessoas desaparecidas, inspirada no sistema Amber Alert Brasil[4]. A proposta reforça o papel da televisão aberta como ferramenta de utilidade pública, por não exigir novos dispositivos, aplicações adicionais ou competências técnicas do espectador.

2 REFERENCIAL TEÓRICO

Esta seção apresenta os principais conceitos que fundamentam o desenvolvimento da aplicação proposta, incluindo o sistema Amber Alert Brasil, o *middleware* Ginga, a linguagem NCL, a extensão NCLua e a abordagem Lua2NCL.

2.1 Amber Alert Brasil

O Amber Alert é um sistema de notificação criado nos Estados Unidos na década de 1990, após o sequestro e assassinato da menina Amber Hagerman. O caso teve grande repercussão, em parte pela ausência de mecanismos imediatos de alerta à população. Em resposta, autoridades e a mídia criaram um protocolo emergencial para divulgar informações sobre desaparecimentos infantis, incentivando a colaboração da sociedade [14].

Desde então, o sistema foi adotado por diversos países. No Brasil, o Amber Alert foi implementado em 2023 por meio de parceria entre o Ministério da Justiça[5] e a empresa Meta[6]. A iniciativa utiliza plataformas como Facebook e Instagram para divulgar alertas de desaparecimento de crianças e adolescentes, segmentando as mensagens geograficamente para atingir usuários com maior potencial de colaboração [10].

Para ativar um alerta no Brasil, é necessário atender aos seguintes critérios:

- Criança ou adolescente desaparecido em situação de risco iminente;

[1]Fórum do Sistema Brasileiro de Televisão: https://forumsbtvd.org.br/
[2]JSON: https://www.json.org/json-en.html
[3]NCLua: http://www.telemidia.puc-rio.br/~francisco/nclua/
[4]Amber Alert Brasil: https://amberalertbrasil.mj.gov.br/
[5]Ministério da Justiça: https://www.gov.br/mj/pt-br
[6]Meta: https://www.meta.com/

In: VIII Workshop Futuro da TV Digital Interativa (WTVDI 2025). Anais Estendidos do XXXI Simpósio Brasileiro de Sistemas Multimídia e Web (WTVDI'2025). Rio de Janeiro/RJ, Brazil. Porto Alegre: Brazilian Computer Society, 2025.

ISSN 2596-1683

- Desaparecimento recente e não voluntário;
- Autorização dos responsáveis;
- Fotografia recente e em boa qualidade;
- Solicitação formal de autoridade policial estadual.

O alerta é ativado num raio de até 160 km a partir do último local em que a vítima foi vista e permanece por até 24 horas. A ausência de imagem não impede a divulgação, mas reduz a eficácia da comunicação (exemplo de alerta, Figura 1).

Figure 1: Captura de tela do site Amber Alert Brasil (https://amberalertbrasil.mj.gov.br/).

2.2 *Middleware* Ginga e Linguagem NCL

O Ginga, estabelecido pela norma ABNT NBR 15606 [1], é o *middleware* oficial do Sistema Brasileiro de Televisão Digital (SBTVD). Sua adoção garante interoperabilidade entre dispositivos e permite a execução de aplicações interativas em televisores compatíveis [3].

O módulo Ginga-NCL interpreta aplicações desenvolvidas com a linguagem declarativa NCL , baseada em XML e estruturada no modelo NCM (*Nested Context Model*). A NCL permite definir a organização temporal e espacial de objetos de mídia, com flexibilidade quanto aos tipos de conteúdo, facilitando a criação de aplicações interativas [5].

2.3 Linguagem NCLua

O NCLua é uma extensão da linguagem Lua integrada ao Ginga-NCL. Sua principal função é oferecer suporte a lógica procedural e controle de interatividade, compensando o caráter estático da NCL [16]. Com ele, é possível manipular eventos, sincronizar mídias e acessar dados externos dinamicamente.

Neste trabalho, o NCLua é utilizado em conjunto com a biblioteca *dkjson* [9], que converte dados no formato JSON em tabelas Lua. Essa abordagem possibilita o carregamento dinâmico de dados sobre pessoas desaparecidas e sua conversão em elementos NCL interpretáveis pelo middleware.

2.4 Lua2NCL

Um desafio inicial do projeto foi a escassez de documentação sobre o uso integrado de Lua e NCL. O trabalho Lua2NCL, de Moraes [4], serviu como base por apresentar um modelo de desenvolvimento voltado a desenvolvedores não familiarizados com NCL. O framework propõe representar elementos NCL como tabelas Lua, convertendo-os por meio de concatenação de strings, visível na Listagem 1, conceito central na solução implementada neste estudo

```
function writeDescriptor()
  -- body
  local aux = false
  for i,t in pairs(headElem) do
    if t:getType() == "descriptor" then
      aux = true
    end
  end

  if (aux) then
    local text = ""
    text = text.."\t\t<descriptorBase>\n"
    for i,t in pairs(headElem) do
      if t:getType() == "descriptor" then
        string = t:print()
        text = text.."\t\t\t"..string
      end
    end
    text = text.."\t\t</descriptorBase>\n\n"
    return text
  else
    return ""
  end
end
```

Listing 1: Exemplo de concatenação utilizando Lua2NCL.

3 PROPOSTA

O desaparecimento de pessoas no Brasil é um problema grave, com impactos significativos para famílias, comunidades e para a formulação de políticas públicas. Dados consolidados da Secretaria Nacional de Segurança Pública (SENASP)[7] apontam que, em 2022, foram registrados 82.216 casos de desaparecimentos no país, enquanto em 2023 esse número caiu para 77.060. Isso equivale a uma média de aproximadamente 225 desaparecimentos por dia em 2022 e 211 em 2023, totalizando 159.276 casos no biênio [12]. A maioria dos desaparecimentos envolveu pessoas adultas (de 18 a 59 anos), que corresponderam a 65,5% dos casos em 2023. Também se observou uma predominância de vítimas do sexo masculino, representando 64,9% dos desaparecimentos no mesmo ano.

No recorte mais recente e específico sobre crianças, o Relatório Estatístico Anual de Crianças Desaparecidas (Ano-base 2022) identificou 2.169 casos de desaparecimentos de crianças de até 12 anos, o que representa cerca de seis desaparecimentos infantis por dia no Brasil [11]. Embora esse número represente uma fração do total de desaparecimentos no país, trata-se de uma faixa etária especialmente vulnerável, que exige respostas rápidas e eficazes.

Apesar da recente implementação do sistema Amber Alert Brasil, voltado à divulgação de alertas sobre o desaparecimento de crianças e adolescentes via redes sociais, persistem desafios quanto à conectividade desigual, uso limitado de smartfones em determinadas regiões e o analfabetismo digital[6], fatores que limitam o alcance e a eficácia desses alertas.

Diante desse cenário, a TV Digital surge como um meio acessível e consolidado, presente em cerca de 88% dos lares brasileiros, capaz

[7]SENASP: https://www.gov.br/mj/pt-br/assuntos/sua-seguranca/seguranca-publica

de ampliar a visibilidade das mensagens de alerta, especialmente entre populações com acesso restrito à internet. Seu alcance massivo e imediato reforça seu potencial como ferramenta de utilidade pública.

Diante disso, este trabalho propõe o desenvolvimento de uma interface interativa de TV Digital, baseada em Ginga-NCL, que consome dados públicos no formato JSON por meio de NCLua para atualização dinâmica das informações. Com base nesse potencial, este trabalho detalha o desenvolvimento de uma interface interativa para TV Digital que, inspirada no modelo Amber Alert Brasil, consome dados públicos em JSON para apresentar, de forma dinâmica e acessível, alertas de pessoas desaparecidas.

4 APLICAÇÃO

Esta seção apresenta a aplicação criada com base na proposta anterior: um protótipo interativo para TV Digital voltado à divulgação de informações sobre pessoas desaparecidas. A solução foi desenvolvida na plataforma Ginga-NCL, utilizando arquivos NCL, scripts em Lua e dados simulados em JSON, baseados em alertas reais.

A aplicação foi desenvolvida em um ambiente de emulação configurado em uma máquina virtual com Ubuntu 20.04.6 LTS (Focal Fossa)[8], utilizando o *VMware Workstation Player 16*[9]. O *middleware* Ginga empregado foi uma versão modificada, disponível no GitHub da pesquisadora Marina Ivanov[10], autora de projetos relevantes na área, como Ambientes multissensoriais aplicados à saúde: desenvolvimento de aplicações e tendências futuras [8]. Essa versão é baseada no projeto original da equipe TeleMidia da PUC-Rio[11].

A aplicação recebe dados simulados por arquivos de imagem e JSON, integrados dinamicamente à interface. Esses dados são fornecidos pela fonte de transmissão digital, representada pelo Broadcast, que disponibiliza os artefatos necessários para a renderização da interface. Essa estrutura é ilustrada no diagrama UML de implantação (Figura 2) baseada nas técnicas de [2, 15], que mostra a arquitetura geral da solução em uma transmissão digital.

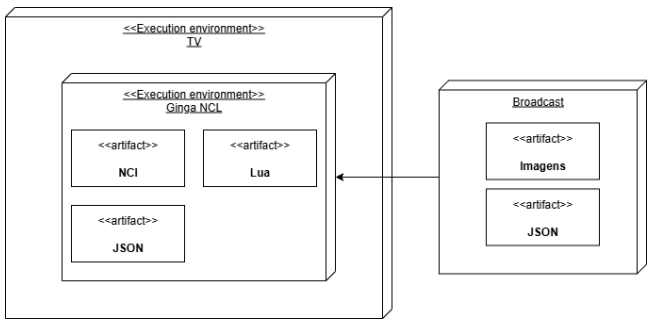

Figure 2: Diagrama UML de implantação

O diagrama UML de sequência (Figura 3) descreve a interação proposta entre o telespectador e a aplicação por meio do controle remoto. Após ligar a TV, a aplicação seria carregada e receberia os

[8]Linux Ubuntu: https://releases.ubuntu.com/focal/
[9]*VMware Workstation Player 16*: https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion
[10]Ginga-Mulsemedia: https://github.com/marinaivanov/ginga-mulsemedia/tree/ncl4
[11]TeleMidia - Ginga: https://github.com/TeleMidia/ginga

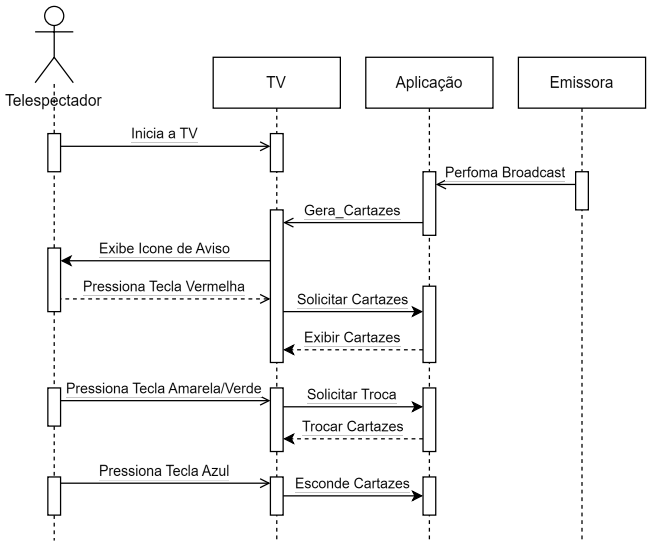

Figure 3: Diagrama UML de Sequência

dados transmitidos pela emissora. Quando o usuário pressionasse a tecla vermelha, os cartazes seriam exibidos na tela. As teclas amarela e verde permitiriam alternar entre diferentes cartazes, enquanto a tecla azul ocultaria a visualização. Essas ações foram modeladas para simular uma experiência interativa e responsiva no contexto da TV Digital.

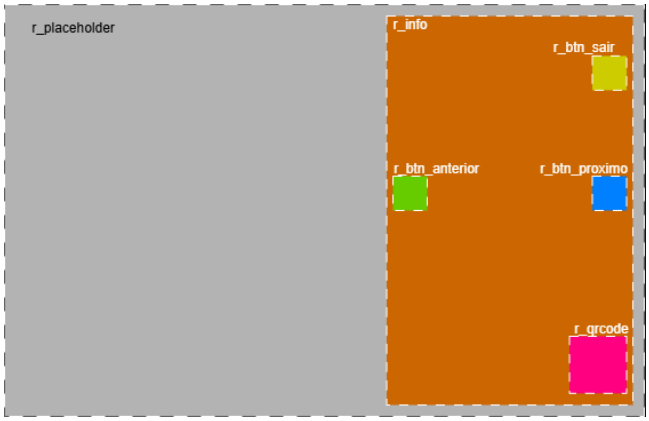

Figure 4: Disposição das regiões na tela

A interface foi projetada com base em uma divisão por regiões, visando otimizar a navegação via controle remoto e garantir clareza na exibição das informações. As principais regiões são descritas a seguir e estão visualmente representadas na Figura 4:

- **r_info**: região principal de exibição dos dados da pessoa desaparecida, localizada à direita da tela;
- **r_btn_anterior** e **r_btn_proximo**: posicionadas ao lado da *r_info*, permitem navegar entre registros;
- **r_placeholder**: simula o conteúdo assistido pelo telespectador, não influencia na aplicação final;

- **r_btn_alerta**: no canto superior direito, usada para ativação ou destaque de alertas;
- **r_btn_sair**: botão de saída, também localizado no canto superior direito;
- **r_qrcode**: exibe um código QR no canto inferior direito, facilitando o acesso a informações complementares. O QR Code é um elemento interativo importante da interface e é posicionado pelo NCL na região *r_qrcode*, conforme visível no listing 2.

A Listagem 2 apresenta o trecho do script NCL responsável pela definição dessas regiões:

```
<regionBase>
<region id="r_info" left="60%" top="0" width="40%" height="100%"
      zIndex="3"/>
<region id="r_btn_anterior" left="60%" top="40%" width="5%"
      height="6%" zIndex="3"/>
<region id="r_btn_proximo" left="95%" top="40%" width="5%" height=
      "6%" zIndex="3"/>
<region id="r_placeholder" width="100%" height="100%" zIndex="1"/>
<region id="r_btn_alerta" left="85%" top="0%" width="15%" height="
      15%" zIndex="2"/>
<region id="r_btn_sair" left="94%" top="9.5%" width="6%" height="
      7%" zIndex="3" />
<region id="r_qrcode" left="88.8%" top="88.5%" width="11%" height=
      "11%" zIndex="3"/>
</regionBase>
```

Listing 2: Trecho do código NCL declarando as regiões

A principal região funcional, *r_info*, é responsável por exibir as informações da pessoa desaparecida, incluindo nome, idade, descrição física, data e local do desaparecimento, além do nome do arquivo da imagem correspondente. A exibição é atualizada dinamicamente por um script NCLua, que lê os dados a partir de arquivos JSON e monta o par entre imagem e informações diretamente na região *r_info*. A Figura 5 ilustra apenas os elementos gerados por esse script; o QR Code, por exemplo, é posicionado separadamente pelo NCL na região *r_qrcode*, e por isso não aparece na figura.

Além disso, a interface inclui uma faixa fixa na parte inferior da tela, que exibe continuamente os contatos emergenciais. O texto "Denuncie: ligue 190 ou" é inserido via NCLua, enquanto o QR Code aparece após o "ou" e facilita o acesso a informações complementares (Figura 4).

Abaixo está um exemplo da estrutura JSON utilizada na aplicação (Listagem 3). O arquivo contém campos como nome, idade, descrição física e último local de avistamento. Esse formato facilita o processamento automatizado dos dados pelo script, mantendo clareza e consistência na apresentação das informações ao usuário.

```
{
  "nome": "Carlos Eduardo Martins",
  "idade": 42,
  "descricao": {
    "cor": "clara",
    "olhos": "castanhos",
    "cabelo": "castanho escuro curto",
    "altura": "1,75 m"
  },
  "ultima_vez_visto": {
    "data": "2025-05-28",
    "hora": "18:30",
    "roupa": "camisa polo cinza, calca jeans azul escuro"
  },
  "ultimo_local_avistamento": "Praca da Matriz, centro de Campinas
      /SP",
  "foto": "person1.jpeg",
}
```

Listing 3: Exemplo de JSON

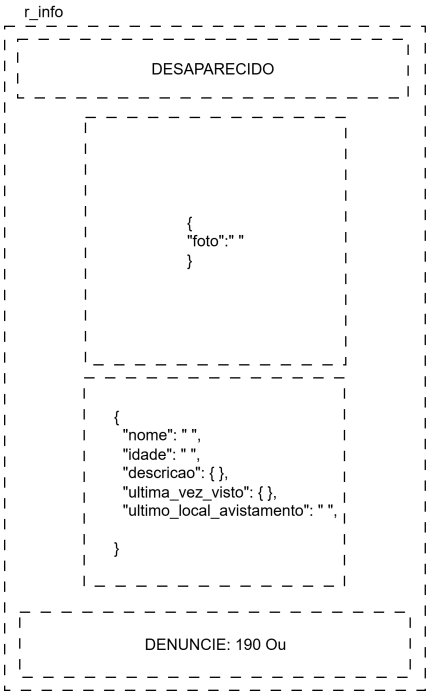

Figure 5: Diagramação da região *r_info*

A biblioteca **dkjson** foi obtida do site oficial e colocada no diretório do script principal para facilitar sua importação via *dofile*. Neste trabalho, foi utilizada a função *decode* para ler o conteúdo do arquivo JSON apresentado na Listagem 4.

```
-- Carrega e executa o módulo dkjson.lua.
    local json = dofile("dkjson.lua")
-- lê o JSON, índice (se for lista) e possíveis erros
    local dados, _, err = json.decode(conteudo)
```

Listing 4: Utilizando biblioteca dkjson.lua no script

Após a leitura dos dados, estes são processados para adaptação ao formato necessário, como demonstra a função na Listagem 5.

```
--- Formata os dados do JSON em uma tabela legível para exibicao
local function formatarTextoJson(t)
    local descricao = t.descricao or {}
    local ultima_vez = t.ultima_vez_visto or {}
    return {
        {"NOME:", t.nome or "N/A"},
        {"IDADE:", tostring(t.idade or "N/A")},
        {"DESCRICAO:", "pele " .. (descricao.cor or "N/A")
            .. ", olhos " .. (descricao.olhos or "N/A")
            .. ", cabelo " .. (descricao.cabelo or "N/A")
            .. ", com " .. (descricao.altura or "N/A") .. " altura
      ."},
        {"ULTIMA VEZ VISTO(A):", (ultima_vez.data or "N/A") .. "
      as " .. (ultima_vez.hora or "N/A")
            .. ", trajava " .. (ultima_vez.roupa or "N/A")},
        {"ULTIMO LOCAL AVISTADO(A):", t.ultimo_local_avistamento
      or "N/A"},
    }
end
```

Listing 5: Função utilização para construir parte de r_info

A função *desenharTextoFormatado* desenha no canvas uma tabela de texto formatado, quebrando linhas para caber na largura máxima e centralizando-as. Para cada par *label*/valor, exibe o *label* em

negrito e vermelho, e o valor em negrito preto na primeira linha; linhas seguintes são em negrito preto. A posição vertical é ajustada conforme o espaçamento definido (ver Listagem 6).

```
local function desenharTextoFormatado(textoFormatado, maxWidth, startY)
    local canvas_dx = canvas:attrSize()
    local y = startY

    for _, par in ipairs(textoFormatado) do
        local label, valor = par[1], par[2]
        local textoCompleto = label .. " " .. valor
        local linhas = quebrarTexto(textoCompleto, maxWidth)

        for i, linha in ipairs(linhas) do
            local linhaLargura = canvas:measureText(linha)
            local x = (canvas_dx - linhaLargura) / 2

            if i == 1 then
                local labelFim = linha:find(":")
                if labelFim then
                    local labelTexto = linha:sub(1, labelFim)
                    local valorTexto = linha:sub(labelFim + 1)

                    -- label em itálico, negrito e vermelho
                    canvas:attrFont(FONT_NAME, FONT_SIZE, "bold italic")
                    canvas:attrColor("red")
                    canvas:drawText(x, y, string.upper(labelTexto))

                    local labelLargura = canvas:measureText(labelTexto)
                    canvas:attrFont(FONT_NAME, FONT_SIZE, "bold")
                    canvas:attrColor("black")
                    canvas:drawText(x + labelLargura, y, valorTexto)

                else
                    canvas:attrFont(FONT_NAME, FONT_SIZE, "bold italic")
                    canvas:attrColor("red")
                    canvas:drawText(x, y, linha)
                end
            else
                canvas:attrFont(FONT_NAME, FONT_SIZE, "bold")
                canvas:attrColor("black")
                canvas:drawText(x, y, linha)
            end

            y = y + TEXTO_Y_SPACING
        end
    end
end
```

Listing 6: Função para formatar e exibir texto

A seguir, na Listagem 7, é apresentado um trecho da função *redraw*, que desenha o título principal com fundo vermelho centralizado, exibe a imagem e as informações formatadas abaixo e cria uma faixa inferior vermelha com contatos de emergência. O QR code é adicionado separadamente pelo NCL.

```
-- Título com retângulo vermelho
canvas:attrFont("vera", 21, "bold")
local titulo = "DESAPARECIDO(A)"
local larguraTitulo = canvas:measureText(titulo)
local alturaTexto = 30
local margem = 30

local xTexto = (canvas_dx - larguraTitulo) / 2
local yTexto = 10  -- posiciona título mais para cima

-- Retângulo vermelho atrás do título
canvas:attrColor("red")
canvas:drawRect("fill",
    xTexto - margem,
    yTexto - margem / 2,
    larguraTitulo + 2 * margem,
    alturaTexto + margem
)

-- Texto branco do titulo
canvas:attrColor("white")
canvas:drawText(xTexto, yTexto, titulo)

-- Imagem e informacoes formatadas
canvas:compose(img_x, img_y, imagemCanvas)
desenharTextoFormatado(item.texto, canvas_dx - 2 * TEXTO_X_MARGIN,
      img_y + IMG_HEIGHT + 20)

-- Faixa inferior vermelha com informacoes de emergêecia
local faixaAltura = 70
local texto1 = "DENUNCIE:"
local texto2 = "☎ 190 ou "

canvas:attrColor("red")
canvas:drawRect("fill", 0, canvas_dy - faixaAltura, canvas_dx, faixaAltura)

canvas:attrColor("white")
canvas:attrFont(FONT_NAME, 12, "bold")
```

Listing 7: Trecho da função *redraw* para desenhar título, faixa inferior e exibir a imagem

A seguir, imagens do protótipo que simula a programação aberta com a qual o usuário está habituado, utilizando como base uma reportagem do vídeo "Achamos no Brasil: Conheça a vila isolada, aonde só se chega a pé ou a cavalo", disponível no YouTube[12]. As imagens apresentadas no protótipo não são reais, tendo sido geradas pelo site *This Person Does Not Exist*[13], e servem exclusivamente para demonstrar o funcionamento e a interface do sistema, sem reproduzir conteúdos ou informações reais.

A Figura 6: Botão informativo no canto superior da tela, indicando que o **botão vermelho** do controle remoto inicia a aplicação. Nas Figuras 7 e 8 são exibidos os cartazes 1 e 2, onde os **botões amarelo** e **verde** navegam entre os alertas, enquanto o **azul** fecha a aplicação, retornando à programação normal.

Figure 6: Botão no canto superior da tela que inicia a aplicação

A aplicação está licenciada sob a Licença Pública Geral GNU (GPL), conforme os modelos de licenciamento de software *open source* discutidos por Sommerville [17]. Essa escolha está alinhada à proposta do sistema, que visa ampliar a divulgação de informações sobre pessoas desaparecidas por meio da TV Digital, incentivando o uso, a modificação e a redistribuição da solução com foco no benefício social.

[12] Reportagem: https://www.youtube.com/watch?v=13Hkq_jm0Tc
[13] *ThisPersonDoesNotExist*: https://thispersonnotexist.org/

Figure 7: Aplicação exibindo o cartaz 1

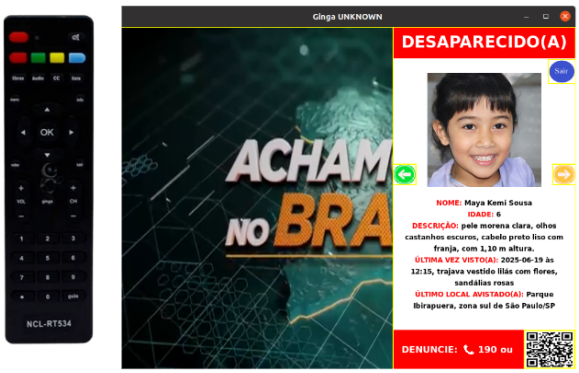

Figure 8: Aplicação exibindo o cartaz 2

5 CONSIDERAÇÕES FINAIS

Este trabalho explorou o suporte ao processamento dinâmico de dados via JSON na linguagem NCL, integrado ao NCLua, demonstrando o potencial dessa abordagem para aplicações na TV digital aberta no Brasil. O uso da biblioteca *dkjson* demonstrou eficiência para o *parsing* de dados em tempo real, embora a depuração de scripts em ambiente emulado permaneça um desafio técnico.

Foi apresentado um caso de uso inspirado no Amber Alert, voltado à divulgação de informações sobre pessoas desaparecidas, aproveitando o alcance da televisão aberta e sua capacidade de transmitir mensagens sem exigir novos dispositivos, aplicativos ou conhecimentos técnicos dos espectadores.

Como trabalhos futuros, além da busca por parcerias para um projeto piloto ao vivo no estado do Acre, é almejada a evolução do protótipo para consumir dados diretamente de uma API governamental, em substituição aos arquivos JSON estáticos, bem como a condução de estudos de usabilidade com telespectadores para avaliar a eficácia da interface e da interação proposta, contribuindo para o fortalecimento de políticas públicas voltadas à localização de pessoas desaparecidas.

REFERENCES

[1] Associação Brasileira de Normas Técnicas. 2007. *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 1: Codificação de dados.* Norma Técnica NBR 15601-1. Associação Brasileira de Normas Técnicas, Rio de Janeiro.

[2] G. Booch and J. Rumbaugh. 2006. *UML: guia do usuário.* Elsevier. https://books.google.com.br/books?id=ddWqxcDKGF8C

[3] Antonio Celestino and Carlos Pernisa Júnior. 2024. Sincronismo e Interatividade com a Segunda Tela frente à Implementação da TV 3.0 no Brasil. In *Anais Estendidos do XXX Simpósio Brasileiro de Sistemas Multimídia e Web* (Juiz de Fora/MG). SBC, Porto Alegre, RS, Brasil, 309–314. https://doi.org/10.5753/webmedia_estendido.2024.243863

[4] Daniel de Sousa Moraes. 2016. Lua2NCL: Framework para autoria textual de aplicações NCL usando Lua. Monografia de Graduação, Universidade Federal do Maranhão. Orientador: Prof. Dr. Carlos de Salles Soares Neto.

[5] Luiz Fernando Gomes Soares e Simone Diniz Junqueira Barbosa. 2009. *Programando em NCL 3.0* (2º ed.). Campus, Rio de Janeiro, RJ.

[6] Adriano França, Aline Silva, Jair Assis, Valdir Silva, Rayanne Santos, Ricson Santana, and Thiago Santos. 2022. Ação para redução do analfabetismo digital em instituições sociais da mata sul de Pernambuco. In *Anais do XIX Congresso Latino-Americano de Software Livre e Tecnologias Abertas* (Evento Híbrido). SBC, Porto Alegre, RS, Brasil, 62–68. https://doi.org/10.5753/latinoware.2022.228049

[7] Instituto Brasileiro de Geografia e Estatística(IBGE). 2024. Pesquisa Nacional por Amostra de Domicílios Contínua: Acesso à Internet e à televisão e posse de telefone móvel celular para uso pessoal 2023. https://agenciadenoticias.ibge.gov.br/media/com_mediaibge/arquivos/f070dbf1d5a8e94ff1d37b7b516e0eb5.pdf Coordenação de Pesquisas por Amostra de Domicílios. Diretoria de Pesquisas.

[8] Marina Ivanov, Eyre Montevecchi, Raphael Abreu, Fabio Barreto, Joel dos Santos, and Débora Muchaluat-Saade. 2020. Ambientes multissensoriais aplicados à saúde: desenvolvimento de aplicações e tendências futuras. *Minicursos SBCAS* 1 (2020), 48–89.

[9] David Kolf. 2023. *dkjson: a JSON module for Lua.* David Kolf. https://dkolf.de/dkjson-lua/ Acesso em: 20 jun. 2025.

[10] Ministério da Justiça e Segurança Pública. 2025. *Amber Alert Brasil.* Ministério da Justiça e Segurança Pública. https://amberalertbrasil.mj.gov.br/ Acesso em: 27 jun. 2025.

[11] Ministério da Justiça e Segurança Pública (MJSP). 2024. RELATÓRIO ESTATÍSTICO ANUAL DE CRIANÇAS DESAPARECIDAS E LOCALIZADAS ANO-BASE - 2022. https://www.gov.br/mj/pt-br/assuntos/sua-seguranca/seguranca-publica/estatistica/download/dados-nacionais-de-seguranca-publica-mapa/relatorio-estatitisco-anual-de-criancas-desaparecidas-e-localizadas-ano-base-2022.pdf

[12] Ministério da Justiça e Segurança Pública (MJSP). 2024. RELATÓRIO ESTATÍSTICO ANUAL DE PESSOAS DESAPARECIDAS PERÍODO: ANOS-BASE 2022 E 2023. https://www.gov.br/mj/pt-br/assuntos/sua-seguranca/seguranca-publica/estatistica/download/desaparecidos/relatorio-estatistico-anual-pessoas-desaparecidas-2019_2021.pdf

[13] Daniel Moraes, Polyana da Costa, Antonio Busson, José Boaro, Carlos Neto, and Sergio Colcher. 2023. On the Challenges of Using Large Language Models for NCL Code Generation. In *Anais Estendidos do XXIX Simpósio Brasileiro de Sistemas Multimídia e Web* (Ribeirão Preto/SP). SBC, Porto Alegre, RS, Brasil, 151–156. https://doi.org/10.5753/webmedia_estendido.2023.236175

[14] Office of Juvenile Justice and Delinquency Prevention (OJJDP). 2010. AMBER Advocate, Volume 4, Issue 1. https://amberalert.ojp.gov/sites/g/files/xyckuh201/files/media/document/advocate_1004.pdf Prepared under Cooperative Agreement number 2008-MC-CS-K028 from the Office of Juvenile Justice and Delinquency Prevention (OJJDP), U.S. Department of Justice.

[15] R.S. Pressman, B.R. Maxim, J. Arakaki, R. Arakaki, R.M. de Andrade, and F.E. Costa. 2021. *Engenharia de software:.* AMGH. https://books.google.com.br/books?id=xIv5zwEACAAJ

[16] Francisco Sant'Anna, Carlos de Salles Soares Neto, Roberto Gerson de Albuquerque Azevedo, and Simone Diniz Junqueira Barbosa. 2009. Desenvolvimento de aplicações declarativas para TV Digital no middleware Ginga com objetos imperativos nclua. *PUC–Rio, Junho* 1, 0 (2009), 0–0.

[17] Ian Sommerville. 2011. *Software Engineering* (9 ed.). Addison-Wesley, Boston, MA.