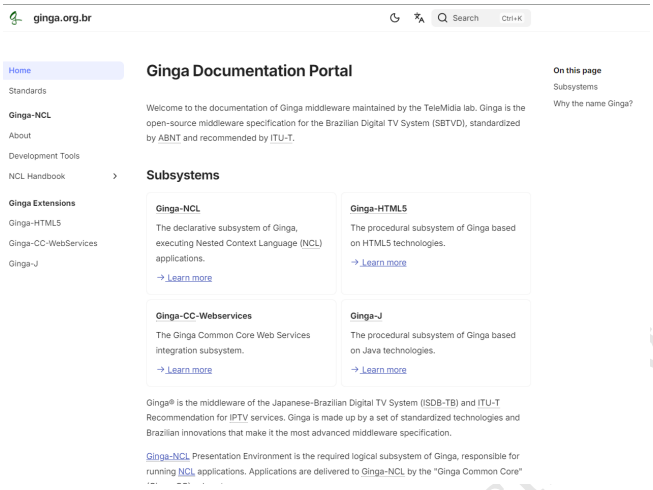
A new Ginga Developer Ecosystem with ginga.org.br portal, ginga-playground and ginga-code tools

Alan L. V. Guedes
a.guedes@reading.ac.uk
University of Reading
UK

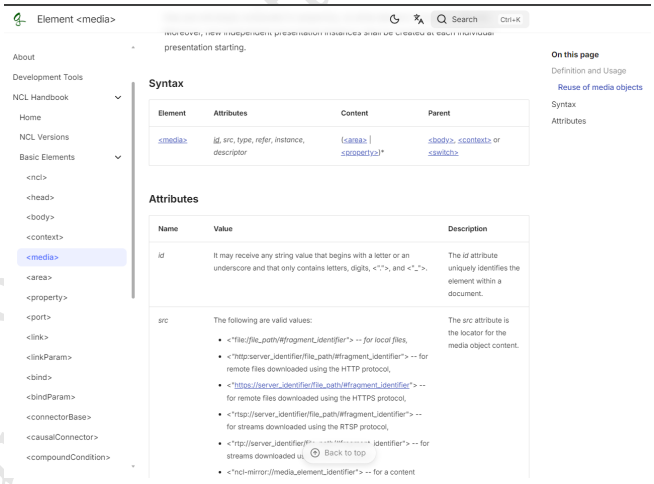
Paulo Victor Borges
paulo.victor.lima15@gmail.com
PUC-Rio
Brazil

Daniel Moraes
danielmoraes@telemidia.puc-rio.br
PUC-Rio
Brazil

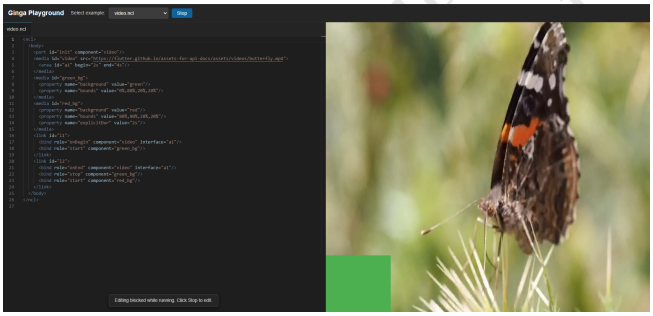
Sergio Colcher
colcher@puc-rio.br
PUC-Rio
Brazil



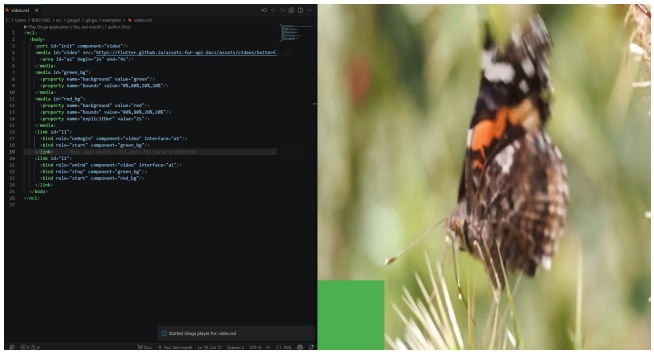
(a) Portal Landing Page



(b) Portal Documentation



(c) Ginga Playground on the Web



(d) Ginga VS Code Extension on Desktop

Figure 1: Ginga Developer Ecosystem interfaces: portal landing page (a), documentation (b), playground workbench (c), and VS Code extension (d).

Unpublished working draft. Not for distribution.

© 2026 SBC – Sociedade Brasileira de Computação.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Anais Estendidos do XXXII Simpósio Brasileiro de Sistemas Multimídia e Web (WTVDI 2026)*, https://doi.org/10.5753/webmedia_estendido.2026.34168.

ABSTRACT

The authoring and execution of Ginga-NCL and HTML5 applications for digital television (DTV) have long been constrained by the lack of developer access to lightweight prototyping and testing tools. In this paper, we present a new **Ginga Developer Ecosystem** designed to lower the barrier to entry for Ginga development.

The ecosystem integrates three pillars: (1) `http://ginga.org.br`, a web portal using a modern static website generator to host a introduction Ginga documentation, NCL Handbook and links to standards; (2) `ginga-playground`, a browser-native workbench for editing and executing applications; and (3) `ginga-code`, a Visual Studio Code extension acting as a launcher for applications directly from an IDE.

KEYWORDS

Ginga-NCL, Developer Ecosystem, Web Workbench, VS Code Extension, Debug Bridge, CanvasKit, Headless Execution

1 INTRODUCTION

The Ginga middleware is the international standard for interactive digital television by both the ITU-T [5] and the Brazilian Digital Terrestrial Television System (SBTVD) [1]. Ginga utilizes the Nested Context Language (NCL)—an XML-based declarative multimedia language. As the official middleware for interactive television applications, the Ginga standard is legally mandated and installed on 100% of certified Digital TV sets and set-top boxes sold in Brazil and several other Latin American countries adopting the ISDB-T framework.

Despite its standardization and wide adoption across SBTVD and international broadcast networks, the lack of developer access to testing and prototyping tools has historically been the primary barrier for Ginga developers. Traditional developer workflows require configuring native OS libraries, installing platform-specific dependencies, or relying on heavy set-top box (STB) hardware emulators [6, 9]. To overcome this lack of access, we introduce a new **Ginga Developer Ecosystem**, an open-source development suite unifying web authoring, portal demonstration, and IDE launching. Built around the modular `gingaf` presentation engine [4], the ecosystem integrates three core pillars:

- The Ginga portal: **http://ginga.org.br**;
- The `ginga-playground` web editor: **https://ginga.org.br/gingaf/playground/**; and
- The `ginga-code` extension: **https://marketplace.visualstudio.com/items?itemName=ginga-org-br.ginga-code**.

To support reproducibility, the source code, unit tests, and demonstration videos are available at **https://github.com/orgs/ginga-org-br/repositories**.

The remainder of this paper is structured as follows. Section 2 reviews related work in Ginga middleware tooling. Section 3 details the architecture of the Ginga Developer Ecosystem and its core pillars. Finally, Section 4 outlines final remarks and future directions.

2 RELATED WORK

The development of Ginga players has historically been led by the TeleMídia Lab at PUC-Rio, producing several versions. PUC-Rio Ginga v0.13.6 [9, 11] was designed for resource-constrained early-generation digital TV receivers. PUC-Rio Ginga v1.0 [3, 12] modernized the renderer and moved to a mono-threaded model inspired by synchronous execution. However, such implementations share a fundamental limitation: they are deeply tethered to native desktop libraries, making web execution impossible. An earlier work was WebNCL [7], which provided a web-based NCL presentation engine but did not support Lua scripting.

The closest related works to our initiative are NuGinga [2] and NuGinga Playcode [10]. Developed by Sidia, NuGinga [2] introduced a Node.js-based Ginga middleware engine. Subsequently, NuGinga Playcode [10] introduced a Node.js-based authoring tool for Ginga-NCL and NCLua using cross-compilation techniques; however, given its Node.js-based architecture, it requires server-side execution infrastructure. Furthermore, it lacks support for Ginga-HTML5 application coding. In contrast, the proposed Ginga Developer Ecosystem and its `ginga-playground` workbench provide a fully client-side, zero-installation environment that natively supports both Ginga-NCL and Ginga-HTML5 (via CCWS integration).

3 GINGA DEVELOPER ECOSYSTEM

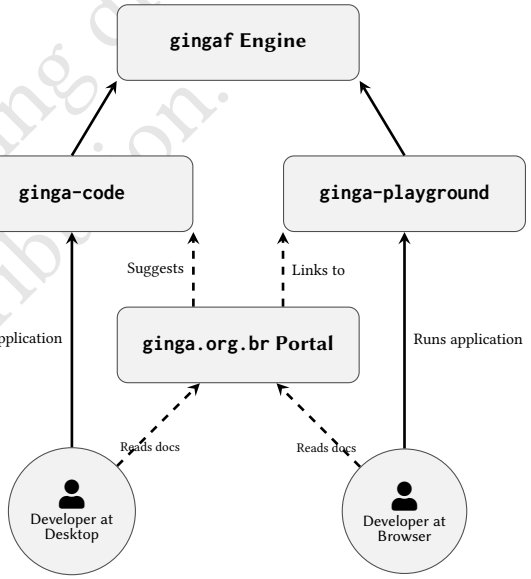


Figure 2: Components of the new Ginga Developer Ecosystem.

The proposed new Ginga Developer Ecosystem consists of components built on top of the `gingaf` presentation engine [4]. The ecosystem isolates the core Ginga execution logic, allowing it to run cross-platform and target web sandboxes and desktop IDEs. As illustrated in Figure 2, the ecosystem accommodates different developer profiles:

- The `Developer at Desktop` profile represents advanced users or academics developing new Ginga extensions.
- The `Developer at Browser` profile represents new users learning Ginga-NCL development in a lightweight sandbox.

These components are organized into the following sections.

3.1 `gingaf`

The `gingaf` presentation engine is an open-source, cross-platform Ginga digital television middleware player built on Dart and the

Unpublished working draft. Not for distribution.

Flutter framework. Unlike traditional native middleware architectures that tightly bind execution state machines directly to OS graphics contexts (such as SDL or GTK) and native AV pipelines, gingaf introduces a decoupled layered architecture that strictly separates headless document logic components from visual UI components.

The core execution engine (ncldoc / NCLDocument) manages the declarative NCL state machine, temporal synchronization loops, and reactive DOM events entirely in pure and cross-platform Dart code. The Ginga presentation layer sits above this logic core, leveraging Flutter to paint visual elements natively on desktop and mobile targets or onto a hardware-accelerated CanvasKit surface (a WebAssembly build of Skia) for in-browser web execution. The engine natively supports Ginga-NCL (including embedded Lua scripting via a pure Dart interpreter), Ginga-HTML5, and Ginga-CC-Webservices (CCWS REST APIs exposed via an internal HTTP server). This modular design powers the rendering surfaces of the web workbench (ginga-playground) and desktop execution targets (ginga-code).

3.2 New ginga.org.br Portal

The official portal ginga.org.br serves as the central hub of the Ginga Developer Ecosystem. This new version revamps the old portal architecture, featuring fully reviewed text and documentation with multilingual support across three languages: Portuguese, Spanish, and English. It is built using Zensical[1], a modern static website generator. The portal integrates specification documentation and already includes the complete NCL Handbook (Figure 1b) alongside live code examples.

3.3 ginga-playground

The ginga-playground is a browser-native code workbench that provides an interactive execution environment running entirely on the client side without native plugins. It combines a code editor built on Monaco Editor [8] and a web deployment of the gingaf presentation engine. To ensure document persistence across sessions, the workbench leverages HTML LocalStorage[2] to store user files and enable seamless editing. Furthermore, the playground comes pre-loaded with a catalog of example NCL applications, enabling developers to instantly explore, edit, and preview interactive multimedia scenarios.

3.4 ginga-code

For desktop development, the ginga-code extension enables the execution of Ginga-NCL and HTML5 applications directly within Visual Studio Code. The extension automatically downloads the native gingaf desktop binary matching the host operating system to serve as an IDE launcher. It then provides an interactive button within the IDE interface that allows developers to launch and execute the Ginga application currently open in the active editor.

4 FINAL REMARKS

The Ginga Developer Ecosystem bridges web distribution and tool launcher interfaces for NCL digital television applications. By integrating ginga.org.br, ginga-playground, and ginga-code on top of the gingaf engine, the suite eliminates environment setup friction and provides instant preview capabilities.

Future work will focus on expanding comprehensive ginga.org.br documentation for Ginga-HTML5 and Ginga-CCWS, as well as introducing an NCL Language Server Protocol (LSP) provider for static semantic verification.

[1] https://zensical.org/
[2] https://html.spec.whatwg.org/multipage/webstorage.html

REFERENCES

[1] ABNT. 2023. *ABNT NBR 15606-2: Digital Terrestrial Television – Ginga-NCL for Fixed and Mobile Receivers – XML Application Language for Application Coding.* Technical Report. Associação Brasileira de Normas Técnicas.

[2] Rodrigo Braga, Volnei Klehm, Leonardo Santana, Kevin Takano, Lauro Gama, Addison Lima, Vinícius Melo, Taynara Paiva, Diego Gonzalez, Michael Mello, Chak Lee, and Lois Nascimento. 2018. NuGinga: a JavaScript Driven Implementation of Brazilian Digital TV Middleware for Android. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web* (Salvador, BA, Brazil) *(WebMedia '18)*. Association for Computing Machinery, New York, NY, USA, 423–426. doi:10.1145/3243082.3267454

[3] J. P. Dodsworth, L. d. M. Terças, A. L. V. Guedes, G. F. Lima, C. d. S. S. Neto, and S. Colcher. 2018. Dynamic Integration of Foreign-Language Parsers into an NCL Player. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*. doi:10.1145/3243082.3243095

[4] Alan Guedes. 2026. gingaf: Headless and Multiplatform Execution of ITU-SBTVD NCL Applications. In *Proceedings of the 34th ACM International Conference on Multimedia* (Rio de Janeiro, Brazil) *(MM '26)*. doi:10.1145/3767308.3834744

[5] ITU-T. 2009. *H.761: Nested Context Language (NCL) and Ginga-NCL for IPTV Services.* Technical Report. International Telecommunication Union, Geneva, Switzerland.

[6] Guilherme Ferreira Lima. 2018. *Ginga-NCL 1.0: A Modular and Extensible Implementation of the Declarative Middleware for Digital TV.* Master's thesis. Pontifícia Universidade Católica do Rio de Janeiro.

[7] Erick Lazaro Melo, Caio César Viel, Cesar A. C. Teixeira, Alexandre Coelho Rondon, and Daniel de Paula Silva. 2013. WebNCL: a web-based presentation machine for multimedia documents. In *Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia)*. doi:10.1145/2382636.2382719

[8] Microsoft. 2024. Monaco Editor: The Browser-Based Code Editor. https://microsoft.github.io/monaco-editor/.

[9] Marcio Ferreira Moreno. 2010. *Conciliando Flexibilidade e Eficiência no Desenvolvimento do Ambiente Declarativo Ginga-NCL.* Ph. D. Dissertation. Pontifícia Universidade Católica do Rio de Janeiro. doi:10.17771/PUCRio.acad.35200

[10] Dina Nogueira, Lois Nascimento, Michael Mello, and Rodrigo Braga. 2020. NuGinga Playcode: A web NCL/NCLua authoring tool for Ginga-NCL digital TV applications. In *Anais Estendidos do XXVI Simpósio Brasileiro de Sistemas Multimídia e Web* (São Luís). SBC, Porto Alegre, RS, Brasil, 75–78. doi:10.5753/nuginga_playcode20.2020.13066

[11] TeleMídia Lab. 2013. PUC-Rio Ginga v0.13.6. https://github.com/TeleMidia/ginga/releases/tag/v0.13.6.

[12] TeleMídia Lab. 2018. PUC-Rio Ginga v1.0. https://github.com/TeleMidia/ginga/releases/tag/v1.0.

Unpublished working draft. Not for distribution.