Geração automática de SDK em ES6 para APIs do Ginga CCWS

Raphael Abreu
raphael.abreu@midiacom.uff.br
Laboratorio MídiaCom
Universidade Federal Fluminense
Niterói, Brasil

Joel A. Ferreira dos Santos
jsantos@eic.cefet-rj.br
CEFET/RJ
Rio de Janeiro, Brasil

Débora C. Muchaluat-Saade
debora@midiacom.uff.br
Laboratorio MídiaCom
Universidade Federal Fluminense
Niterói, Brasil

ABSTRACT

The evolution of the Ginga standard for Digital TV requires tools that facilitate the adaptation and development of interactive applications. This article presents an SDK generator in JavaScript (ES6) based on the OpenAPI (Swagger) specification of the Ginga CCWS standard for TV 2.5 (ABNT NBR 15606-11). The SDK has the potential to simplify the interaction with the CCWS API services of the TV receiver, promoting interoperability and speeding up the development of applications for Brazilian Digital TV. In addition, the use of OpenAPI guarantees adaptability to future versions of the standard, facilitating the updating and maintenance of applications.

KEYWORDS

Ginga, TV Digital, OpenAPI, SDK, JavaScript, ES6, CCWS

1 INTRODUÇÃO

A norma Ginga, estabelecida pela ABNT NBR 15606 [1], é o middleware padrão para a TV Digital no Brasil, proporcionando um ambiente para a criação de aplicações interativas e personalizadas. A evolução da norma, culminando na subseção ABNT NBR 15606-11, introduziu o conceito de Webservices, abrindo um leque de possibilidades para a integração com serviços externos que rodam em dispositivos de segunda tela e a criação de experiências mais ricas para os usuários.

O Ginga Common-Core Webservice (CCWS) [2] oferece um conjunto de interfaces de programação de aplicações (APIs) baseadas em HTTP, que permitem a comunicação entre o Ginga e aplicações HTML5 locais ou aplicações remotas. Através das funcionalidades oferecidas pelo CCWS, como autorização e autenticação, gerenciamento de dispositivos, notificações e conteúdo personalizado, o Ginga pode se integrar de forma mais eficiente com outros dispositivos e serviços na rede. Isso permite, por exemplo, que um aplicativo Ginga em um televisor se comunique com um smartphone para compartilhar informações ou controlar a reprodução de conteúdo, ou que um serviço externo envie notificações personalizadas para o usuário com base em suas preferências e histórico de uso. A utilização dos CCWS impulsiona a criação de um ecossistema de aplicações e serviços interconectados, tornando a experiência da TV Digital mais interativa, personalizada e relevante para os usuários.

No desenvolvimento de software, a utilização de SDKs (Software Development Kits) tem se mostrado uma prática eficaz para acelerar o processo de desenvolvimento e reduzir a curva de aprendizado em novas tecnologias [4]. SDKs fornecem um conjunto de ferramentas, bibliotecas e exemplos que abstraem a complexidade da comunicação com APIs e serviços, permitindo que os desenvolvedores se concentrem na lógica de suas aplicações.

No contexto da norma Ginga, um SDK para as APIs do CCWS desempenha um papel crucial na promoção da interoperabilidade e na facilitação da criação de aplicações. Ao abstrair a complexidade da comunicação HTTP e da serialização de dados, o SDK permite que os desenvolvedores utilizem os serviços da norma de forma mais intuitiva e eficiente.

Este trabalho propõe um gerador de SDK em JavaScript (ES6) a partir da especificação OpenAPI (Swagger) da norma Ginga CCWS. A escolha do JavaScript como linguagem para o SDK visa atender à crescente demanda por aplicações web e híbridas para a TV Digital, além de aproveitar a vasta comunidade de desenvolvedores e a ampla disponibilidade de ferramentas e bibliotecas.

O uso do OpenAPI como base para a geração do SDK garante a aderência aos padrões da indústria e a interoperabilidade com outras ferramentas e plataformas. Além disso, a especificação OpenAPI facilita a atualização e manutenção do SDK, pois qualquer alteração na API pode ser refletida automaticamente no código gerado.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta uma revisão dos trabalhos relacionados, explorando outras abordagens para geração de SDKs e uso de OpenAPI em diferentes contextos. A Seção 3 detalha a especificação OpenAPI do Ginga CCWS, explicando sua estrutura e importância. Na Seção 4, descrevemos o gerador de SDK CCWS proposto, incluindo sua arquitetura, funcionamento e processo de geração de código. Finalmente, a Seção 5 conclui o artigo, discutindo as implicações do trabalho, suas limitações e possíveis direções para pesquisas futuras.

2 TRABALHOS RELACIONADOS

Em [5], os autores exploram a aplicação do framework OpenAPI para descrever e interagir com dispositivos na Web das Coisas (WoT). O trabalho propõe uma estrutura que utiliza o OpenAPI para gerar descrições de dispositivos WoT, permitindo a descoberta e interação padronizada com dispositivos heterogêneos através de interfaces RESTful. Essa abordagem visa simplificar a integração de dispositivos IoT em aplicações web e promover a interoperabilidade entre diferentes plataformas e tecnologias.

Em [3], os autores apresentam o Swagger Codegen, uma ferramenta de código aberto para a geração de SDKs a partir de especificações OpenAPI. O Swagger Codegen suporta diversas linguagens de programação e frameworks, permitindo a criação de SDKs personalizados para diferentes plataformas e tecnologias.

In: VII Workshop Futuro da TV Digital Interativa (WTVDI 2024). Anais Estendidos do XXX Simpósio Brasileiro de Sistemas Multimídia e Web (WTVDI'2024). Juiz de Fora/MG, Brazil. Porto Alegre: Brazilian Computer Society, 2024.

ISSN 2596-1683

Além dos trabalhos acadêmicos, diversas iniciativas governamentais têm adotado o OpenAPI como padrão para a descrição de suas APIs. No Brasil, o Portal Brasileiro de Transparência[1] utiliza o OpenAPI para documentar e disponibilizar seus serviços, facilitando o acesso e a integração com dados públicos.

A disponibilização de especificações OpenAPI para serviços governamentais e padrões tecnológicos, como a norma Ginga CCWS, representa um passo importante para a comunidade de desenvolvedores. Essa prática promove a transparência, a interoperabilidade e a inovação, facilitando a criação de aplicações e serviços que utilizam esses recursos. A geração de SDKs a partir dessas especificações, como o SDK JavaScript apresentado neste trabalho, simplifica ainda mais o processo de desenvolvimento, permitindo que os desenvolvedores se concentrem na lógica de suas aplicações e contribuam para um ecossistema mais rico e diversificado.

3 ESPECIFICAÇÃO OPENAPI DO GINGA CCWS DA TV 2.5

Como primeira contribuição deste trabalho foi feita uma especificação OpenAPI para a norma Ginga CCWS [2] da TV 2.5. Ela descreve os endpoints, métodos, parâmetros, tipos de dados e respostas esperadas da API. Ela serve como base para a geração automática do SDK e garante a interoperabilidade entre diferentes implementações. A especificação está disponível em formato YAML e pode ser visualizada em https://drive.google.com/file/d/1Bb0UV41ysQXhyEurePsOEaDl3Uc/view?usp=sharing testada utilizando ferramentas como o Swagger Editor[2].

O trecho de código na Figura 1 descreve um endpoint na API Ginga CCWS: o $/dtv/authorize$, acessado via método GET. Ele serve para solicitar o estado de autorização de um cliente, identificado pelo parâmetro obrigatório $clientid$. Adicionalmente, o $display-name$ é usado para apresentar o nome do cliente ao usuário durante o processo de autorização. Parâmetros opcionais como pm, kxp e key permitem lidar com diferentes métodos de pareamento e segurança para clientes não locais. A especificação detalha as possíveis respostas, incluindo sucesso (código 200) com exemplos de diferentes cenários de acesso, e erro "não encontrado" (código 404) com referências a exemplos de erros específicos.

De posse de uma da especificação OpenAPI, é possível gerar documentação interativa e de fácil navegação, como a ilustrada na Figura 2. Essa documentação permite que desenvolvedores explorem os endpoints da API, seus parâmetros e as respostas esperadas, facilitando a compreensão e o uso da API. Adicionalmente, a especificação OpenAPI pode ser utilizada para gerar automaticamente testes que validam o comportamento da API, garantindo sua qualidade e confiabilidade.

Ao clicar em um endpoint específico, é possível visualizar uma descrição detalhada da rota, incluindo os parâmetros necessários, exemplos de requisições e respostas, e até mesmo a possibilidade de testar a rota diretamente na documentação. O endpoint $/dtv/authorize$ pode ser visualizado na Figura 3 agilizando o processo de desenvolvimento e integração. Adicionalmente, a especificação OpenAPI pode ser utilizada para gerar automaticamente testes que validam o comportamento da API, garantindo sua qualidade e confiabilidade.

[1]https://api.portaldatransparencia.gov.br
[2]https://editor.swagger.io/

```yaml
paths:
  /dtv/authorize:
    get:
      operationId: dtvAuthorizeGet
      tags:
        - Autorizacao de clientes
      summary: Requisita o estado de autorizacao do cliente.
      description: >...
      parameters:
        - name: clientid...
        - name: display-name...
        - name: pm...
        - name: kxp...
        - name: key...
      responses:
        '200':
          description: Sucesso
          content:...
        '404':
          description: Não encontrado
          content:
            application/json:
              examples:
                '101':
                  $ref: '#/components/examples/101'
                '102':
                  $ref: '#/components/examples/102'
                '105':
                  $ref: '#/components/examples/105'
                '106':
                  $ref: '#/components/examples/106'
```

Figure 1: Exemplo especificação OpenAPI para a rota de autorização de clientes.

Figure 2: Exemplo de visualização de lista de APIs OpenAPI.

É importante ressaltar que a especificação OpenAPI permite a flexibilidade na definição do host da API. Ao utilizar o SDK, o desenvolvedor poderá configurar o host da API de acordo com suas

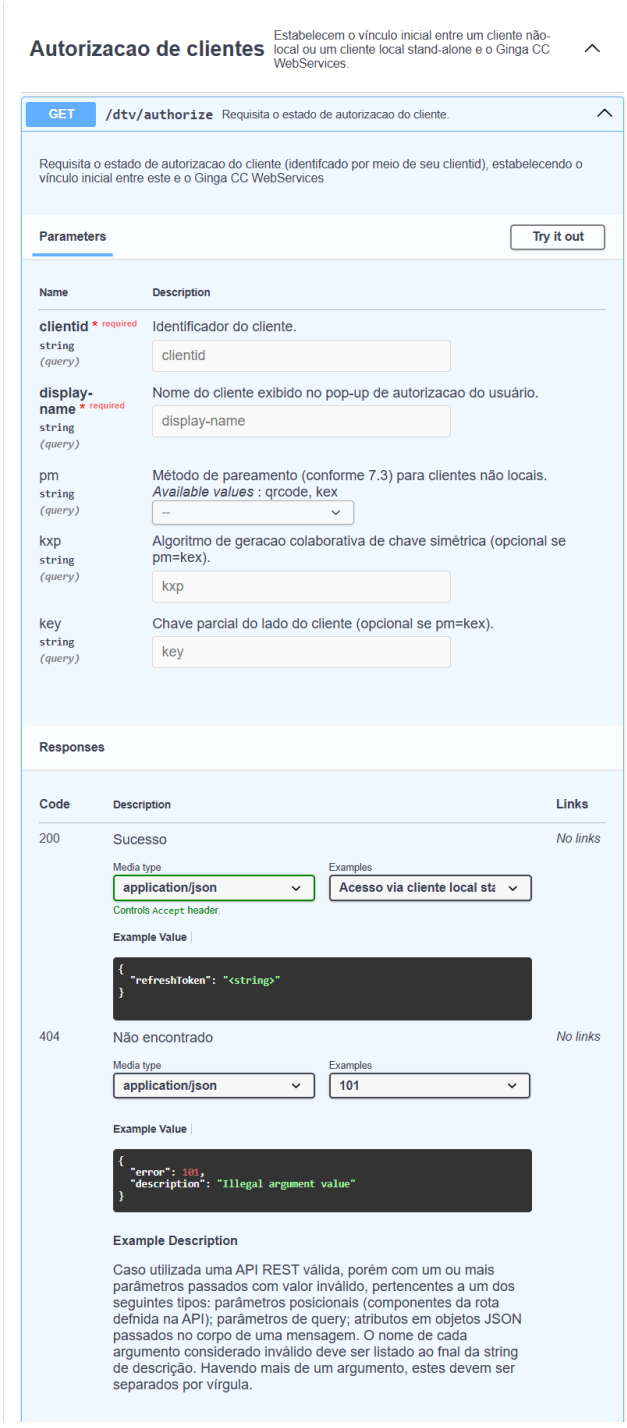

Figure 3: Exemplo de visualização do endpoint *dtv/authorize* na interface gráfica.

necessidades, seja em um ambiente de desenvolvimento, homologação ou produção. Essa flexibilidade é fundamental para garantir a adaptabilidade do SDK a diferentes cenários e infraestruturas.

4 GERADOR DE SDK CCWS

O gerador de SDK é um script Node.js que utiliza a especificação OpenAPI do Ginga CCWS como entrada e gera automaticamente o código JavaScript do SDK. Ele automatiza a criação de métodos para cada endpoint da API, simplificando o uso dos serviços Ginga CCWS. O código do gerador realiza as seguintes etapas: lê o arquivo JSON da especificação OpenAPI da Ginga CCWS 2.5, analisa-o para identificar os endpoints, métodos, parâmetros e respostas da API, gera o código JavaScript do SDK com um objeto API contendo métodos para cada endpoint, e salva o código em um arquivo sdk.js. A classe FetchWrapper encapsula a lógica de comunicação HTTP usando a função fetch do JavaScript. O gerador de SDK encontra-se em https://drive.google.com/file/d/1XpBi6iEQU3cQAWF5ZCnvIwX0jObSWojR/view?usp=sharing

4.1 Usando o SDK gerado

O SDK gerado oferece uma API onde cada endpoint é representado por um método com parâmetros correspondentes à especificação OpenAPI. A Listagem 1 demonstra um exemplo de uso, importando o SDK e realizando uma chamada à função dtvAuthorizeGet. Note que é possível customizar o host da API, caso necessário.

Listing 1: Exemplo de uso do SDK gerado

```
import API from './sdk.js';
API.host = '<ip>:<porta>'; // opcional

await API.dtvAuthorizeGet({
    clientid: 'xpto',
    displayName: 'xpto',
    pm: 'qrcode'
})
.then((value) => {
    console.log(value);
})
.catch(error => {
    console.log(error);
});
```

Além disso, o gerador permite customizar a especificação OpenAPI antes da geração do SDK, possibilitando a adaptação da API às necessidades específicas do projeto, adicionando ou removendo funcionalidades conforme necessário.

5 CONCLUSÃO

Este artigo apresentou um gerador de SDK em JavaScript para a norma Ginga CCWS. Foi criada a especificação OpenAPI da norma referente ao Ginga CCWS da TV 2.5 para usar como base do gerador. O SDK gerado tem potencial de simplificar o desenvolvimento de aplicações interativas para a TV Digital, promovendo a interoperabilidade e a adaptabilidade a futuras versões da norma.

A utilização do OpenAPI e do JavaScript permite que os desenvolvedores aproveitem as vantagens de uma linguagem moderna e amplamente utilizada, além de garantir a aderência aos padrões da indústria e a facilidade de integração com outras ferramentas e plataformas.

Como trabalho futuro pretende-se expandir o gerador de SDK para outras linguagens de programação, como Python. Além disso, com a evolução da norma Ginga para a TV 3.0, planeja-se atualizar a especificação OpenAPI e o gerador de SDK para suportar as novas funcionalidades e serviços que serão introduzidos.

Por fim, um passo crucial será a realização de testes práticos do SDK com desenvolvedores de aplicações para a TV Digital, a fim de coletar feedback sobre a usabilidade, a eficiência e a capacidade do SDK de atender às necessidades reais do desenvolvimento de aplicações Ginga.

REFERENCES

[1] ABNT. 2012. Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 1: Codificação de dados.
[2] ABNT. 2017. Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 11: Ginga CC WebServices – Especificação de WebServices do Ginga Common Core.
[3] Swagger Codegen. 2024. swagger-codegen contains a template-driven engine to generate documentation, API clients and server stubs in different languages by parsing your OpenAPI / Swagger definition. https://github.com/swagger-api/swagger-codegen
[4] Mary Shaw and David Garlan. 1996. *Software architecture: perspectives on an emerging discipline.* Prentice-Hall, Inc.
[5] Aimilios Tzavaras, Nikolaos Mainas, and Euripides G.M. Petrakis. 2023. OpenAPI framework for the Web of Things. *Internet of Things* 21 (2023), 100675. https://doi.org/10.1016/j.iot.2022.100675