GUAPI: A Gateway-Agnostic, Unified, Data-Driven Platform for API Governance Based on Event Streaming

Estêvão Bissoli Saleme
Serviço Federal de Processamento de Dados (Serpro)
Belo Horizonte-MG, Brasil
estevao.saleme@serpro.gov.br

Hellyan Alves de Oliveira
Serviço Federal de Processamento de Dados (Serpro)
São Paulo-SP, Brasil
hellyan.oliveira@serpro.gov.br

João Vitor da Silva Gomes
Serviço Federal de Processamento de Dados (Serpro)
Recife-PE, Brasil
joao.gomes@serpro.gov.br

Flávio Gomes da Silva Lisboa
Serviço Federal de Processamento de Dados (Serpro)
São Paulo-SP, Brasil
flavio.lisboa@serpro.gov.br

Resumo

Sistemas intensivos em software, especialmente infraestruturas de grande escala do setor público, dependem cada vez mais da integração para fornecer serviços digitais em larga escala. Nesse contexto, garantir a qualidade das APIs é estratégico, mas continua sendo um desafio devido ao alto volume de requisições, rigorosos requisitos de disponibilidade e à complexidade operacional. Este artigo apresenta o GUAPI, uma plataforma unificada de suporte à governança de APIs, projetada e implantada no Serpro. Diferentemente das soluções existentes, que são fortemente acopladas a *gateways* proprietários, o GUAPI adota uma arquitetura orientada a dados e baseada em processamento de fluxos de eventos para avaliar continuamente a qualidade das APIs, processando milhões de requisições por dia em tempo quase real. Construída sobre tecnologias de código aberto, principalmente o Apache Kafka e seu ecossistema, a plataforma consolida métricas de desempenho, disponibilidade e confiabilidade em uma visão única e coerente de gestão. A experiência de uso em produção demonstra que ele amplia a visibilidade operacional e possibilita a tomada de decisões proativas, mantendo-se independente de tecnologias específicas de *gateway*.
Video de demonstração: https://doi.org/10.5281/zenodo.21752751

Palavras-chave

APIs, Governança, Kafka, *Event Streaming*, Serviços Digitais

1 Introdução

Interfaces de Programação de Aplicações (APIs) constituem o principal mecanismo de integração em sistemas modernos, viabilizando a interoperabilidade entre plataformas heterogêneas tanto na indústria quanto no governo [6, 12]. Estudos recentes indicam que 83% do tráfego global da Internet já é composta por chamadas de API [8], e organizações governamentais dependem cada vez mais dessas interfaces para compartilhamento de dados entre órgãos, empresas e serviços voltados ao cidadão [10, 14].

Apesar de sua importância estratégica, gerenciar APIs em escala permanece desafiador. Sua natureza distribuída, aliada a altos volumes de requisições e requisitos rigorosos de disponibilidade, torna a gestão intrinsecamente complexa [11]. Incidentes como picos de latência, indisponibilidades parciais e taxas elevadas de erro podem propagar-se por sistemas dependentes, sendo frequentemente agravados por observabilidade inadequada e pela ausência de indicadores unificados de qualidade [3, 16]. Na prática, organizações tendem a se apoiar em visões isoladas de desempenho ou disponibilidade, dificultando melhorias sistemáticas e orientadas por dados [10].

Nesse contexto, a governança de APIs ganha relevância estratégica. Definida como o conjunto de padrões, políticas e práticas que orientam o desenvolvimento, a implantação e o uso de APIs em uma organização [7], a governança vai além do monitoramento operacional: estabelece o arcabouço estratégico para produzir interfaces descobríveis, seguras, escaláveis e reutilizáveis. Contudo, como destacam Pillai et al., embora as APIs forneçam a base para a transformação digital e os ecossistemas de negócios, elas permanecem intrinsecamente difíceis de gerir e governar em escala [11]. Estudos recentes confirmam que, mesmo em organizações tecnologicamente maduras, a governança de APIs requer processos estruturados e ferramentas dedicadas para assegurar consistência e qualidade em escala [1].

As ferramentas existentes concentram-se, em geral, no monitoramento ou na gestão em nível de *gateway*, oferecendo métricas como disponibilidade, latência e volume de requisições, mas sem apoiar plenamente a governança ou a avaliação longitudinal da qualidade [10]. Estudos recentes também apontam a carência de *frameworks* unificados que integrem múltiplas dimensões de qualidade no gerenciamento de APIs [4]. Além disso, soluções comerciais frequentemente impõem acoplamento a *gateways* proprietários e custos de licenciamento, enquanto abordagens baseadas em lotes ou inspeção manual de logs tornam-se insuficientes em ambientes de grande escala. Nesse contexto, mecanismos contínuos de feedback e métricas de tempo de execução são essenciais para sistemas prontos para produção [5].

Para enfrentar essas limitações, este artigo apresenta o GUAPI (Gestão Unificada de APIs), uma plataforma unificada e orientada por dados para apoiar a governança de APIs, desenvolvida e implantada no Serviço Federal de Processamento de Dados (Serpro). O GUAPI adota uma arquitetura baseada em eventos e processamento em streaming para calcular continuamente métricas de qualidade a partir de logs de acesso em tempo quase real. Ao integrar indicadores de desempenho, confiabilidade, disponibilidade, experiência do usuário e segurança em uma única visão analítica, a plataforma apoia a tomada de decisão proativa e a melhoria contínua de serviços baseados em APIs.

2 Visão Geral do GUAPI

O GUAPI é uma ferramenta de software que fornece uma visão consolidada e longitudinal da qualidade das APIs em uma organização. Diferentemente de um *gateway* de API, o GUAPI não substitui a infraestrutura existente: opera como uma camada analítica independente que consome logs de acesso e dados de saúde já produzidos pelo ecossistema.

A plataforma foi concebida no programa interno de inovação do Serpro em 2024, onde obteve o primeiro lugar no Prêmio Serpro de Inovação na categoria Infraestrutura Tecnológica para Governo Digital, e consolidou-se em um sistema implantado em produção, em Agosto de 2025, que apoia serviços críticos que proveem acesso aos dados sobre diversos domínios de informações armazenados pela empresa, tais como CPF, CNPJ, veículos, biometria, comércio exterior, etc.

Seu projeto é guiado por quatro princípios:

(1) visibilidade holística das dimensões de qualidade;
(2) processamento em tempo quase real e em larga escala;
(3) independência de *gateways* proprietários; e
(4) suporte à tomada de decisão orientada por métricas.

A Figura 1 apresenta a página inicial do GUAPI, que reúne um resumo semanal e em tempo real das métricas consolidadas das APIs monitoradas. Os indicadores são exibidos em *cards*, incluindo APDEX (*Application Performance Index*) [2], disponibilidade, volume de acessos, acessos satisfeitos e toleráveis, além dos minutos de indisponibilidade. Cada *card* apresenta comparativos com a semana anterior e o último mês, acompanhados de tendências visuais que facilitam a identificação de melhorias ou degradações.

2.1 Funcionalidades, Interface e Usuários-Alvo

As funcionalidades do GUAPI estão organizadas em quatro eixos.

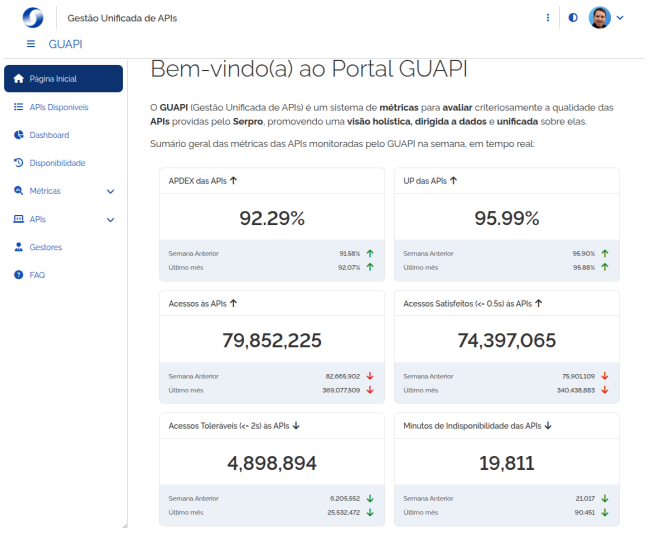

Figura 1: Página inicial do portal GUAPI, apresentando o sumário semanal das métricas consolidadas de todas as APIs monitoradas.

O primeiro é o **monitoramento e visualização em tempo real**. Painéis de controle (*dashboards*) oferecem uma visão holística de todas as APIs monitoradas, com sumários semanais e métricas atualizadas continuamente. A visualização pode ser detalhada por *endpoint*, e indicadores de tendência sinalizam se o desempenho ou a disponibilidade de cada API está melhorando ou piorando em relação a períodos anteriores.

O segundo eixo corresponde ao **sistema de métricas de qualidade**. O GUAPI processa dados de chamadas às APIs para extrair indicadores como APDEX (índice de satisfação baseado no tempo de resposta) [2], *uptime* (percentual de disponibilidade), taxa de erros, MTBF (tempo média entre falhas) e MTTR (confiabilidade e eficiência na recuperação) e vazão (TPR - *ThroughPut Rate*). Essas métricas abrangem requisitos não funcionais reconhecidos como críticos para a qualidade de APIs, incluindo eficiência de desempenho, confiabilidade e disponibilidade [13].

O terceiro eixo é a **gestão de disponibilidade**. *Healthchecks* contínuos verificam se as APIs estão acessíveis para consumo, alimentando um histórico diário de *uptime* que permite identificar padrões de quedas e instabilidades. O cadastro de indisponibilidades programadas evita alarmes falsos durante janelas de manutenção preventiva.

O quarto eixo é a **análise inteligente**. Com apoio do SerproLLM, serviço de Grandes Modelos de Linguagem (LLM) do Serpro, o sistema usa Inteligência Artificial (IA) para gerar automaticamente interpretações textuais sobre o comportamento das métricas e o histórico de disponibilidade, auxiliando gestores na tomada de decisão (Figura 2). A gestão de limiares permite configurar valores admissíveis e ideais para cada indicador, facilitando a identificação imediata de degradações. Um catálogo centralizado associa cada API aos seus respectivos gestores, reforçando responsabilização e rastreabilidade.

A Figura 3 apresenta o painel principal do GUAPI, que permite filtrar a visualização por API, código de serviço e período de análise. O painel exibe indicadores consolidados em *cards*, com codificação por cores e setas de tendência, abrangendo volume de chamadas, APDEX, vazão, taxa de erro, disponibilidade, MTBF e MTTR. A tabela inferior detalha as métricas por *endpoint*, incluindo método

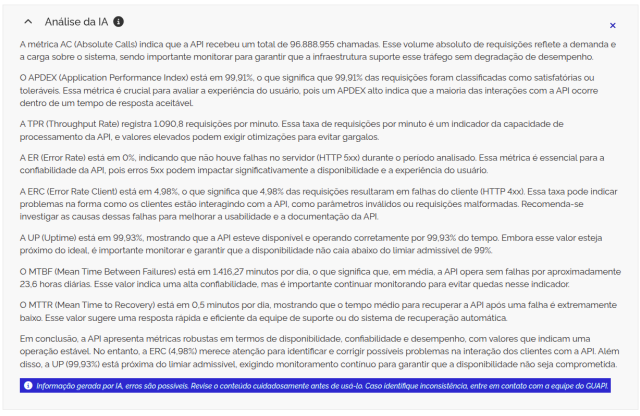

Figura 2: Análise descritiva das métricas de qualidade gerada por IA no GUAPI com o SerproLLM.

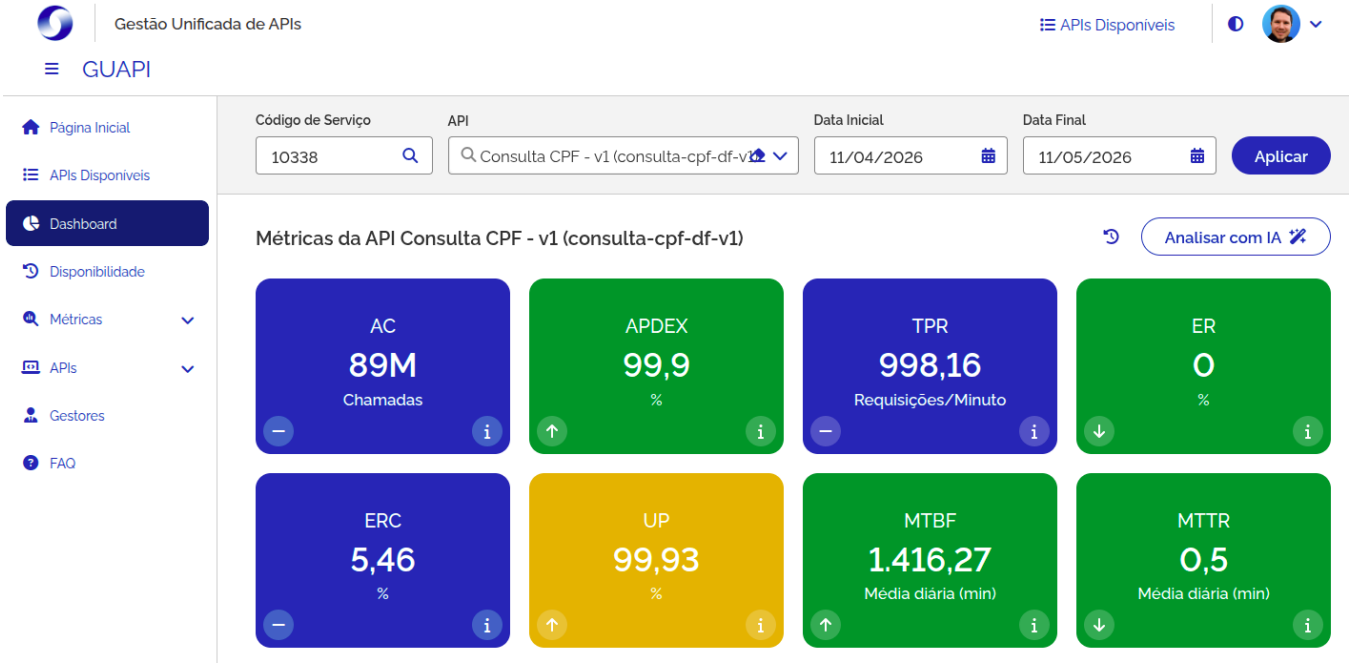

Figura 3: Painel principal (*dashboard*) do GUAPI exibindo os indicadores de qualidade consolidados de uma API. Os *cards* superiores apresentam métricas de volume (AC), desempenho e vazão (APDEX, TPR), erros (ER - 4XX, ERC - 5XX) e confiabilidade (UP, MTBF, MTTR), com codificação por cores indicando o estado de cada indicador, sendo azul - neutro, verde - excelente, amarelo - com oportunidades.

HTTP, domínio e caminho. O botão "Analisar com IA" permite acionar análises automatizadas sobre os dados exibidos.

Essas funcionalidades atendem a múltiplos perfis de usuários. Equipes de desenvolvimento utilizam as métricas para orientar refatorações, arquitetura e ajustes de desempenho. Equipes de confiabilidade e operações apoiam-se nos indicadores em tempo real para reduzir o tempo de resolução de incidentes. Gestores e tomadores de decisão que utilizam a visão consolidada para avaliar a saúde dos serviços, riscos e custos.

3 Arquitetura e Projeto Técnico

3.1 Fluxo Arquitetural

O GUAPI adota uma arquitetura orientada a eventos, composta por componentes fracamente acoplados e integrados por meio do Apache Kafka como plataforma de streaming. Essa abordagem favorece escalabilidade, resiliência, flexibilidade e baixo acoplamento entre as camadas de coleta, processamento, persistência e apresentação. De forma resumida, o fluxo arquitetural fim a fim pode ser compreendido a partir da seguinte sequência:

(1) Os consumidores externos acessam as APIs do Serpro por meio dos *gateways* corporativos, que encaminham as requisições aos serviços de *backend* e geram registros de tráfego em tempo quase real.
(2) Os registros dos *gateways* são capturados e padronizados por um sistema de logs, que os publica no tópico Kafka *api-event-api-acesso*. Paralelamente, um coletor de *healthcheck* realiza sondagens HTTP sobre as APIs cadastradas e publica os resultados em *metricas.up-raw*.
(3) O GUAPI gerencia o cadastro das APIs, metadados e configurações de monitoramento por meio do *Backend-for-Frontend*, persistindo essas informações no banco de dados PostgreSQL e publicando alterações no tópico *events.api-change*.
(4) O Apache Kafka atua como barramento de eventos, desacoplando os fluxos de configuração, acesso, disponibilidade e métricas consolidadas.
(5) O Processador de Métricas consome os eventos de acesso e disponibilidade, calcula indicadores como APDEX, vazão, taxa de erros, *uptime*, MTBF e MTTR, e publica os resultados em *metricas.api* e *metricas.disponibilidade*.
(6) O Kafka Connect persiste as métricas consolidadas no PostgreSQL, enquanto o *Backend-for-Frontend* disponibiliza esses dados ao *frontend*, composto por interfaces administrativas e painéis de acompanhamento.
(7) Recursos complementares, como o armazenamento de especificações OpenAPI em um "balde" de armazenamento de objetos e a integração com o Serpro LLM, apoiam a governança e a geração de análises textuais sobre métricas e disponibilidade.
(8) Os usuários internos acessam o painel do GUAPI para acompanhar os indicadores consolidados das APIs monitoradas e apoiar atividades de governança, operação e observabilidade.

A arquitetura do GUAPI é descrita a seguir a partir de quatro visões arquiteturais, desenhadas em nível de contêiner do modelo

C4[1]: coleta, processamento, persistência e apresentação. Essa decomposição permite evidenciar as principais responsabilidades dos componentes, seus fluxos de comunicação e os pontos de integração entre os componentes.

3.2 Visão de Coleta

A coleta de dados no GUAPI ocorre por dois fluxos complementares (Figura 4). O primeiro baseia-se na integração com o Odradek, sistema centralizado de logs da organização, responsável por capturar registros de tráfego real provenientes de diferentes *gateways* de API. Esses registros são padronizados e publicados no tópico Kafka *api-event-api-acesso*, assegurando um formato uniforme para os eventos de acesso, independentemente do *gateway* de origem.

O segundo fluxo é realizado pelo Coletor de *Healthcheck*, baseado no Vector, que executa sondagens HTTP ativas sobre as APIs cadastradas com o objetivo de aferir sua disponibilidade. Os resultados brutos dessas verificações são publicados no tópico Kafka

[1]Modelo C4 disponível em: https://c4model.com/

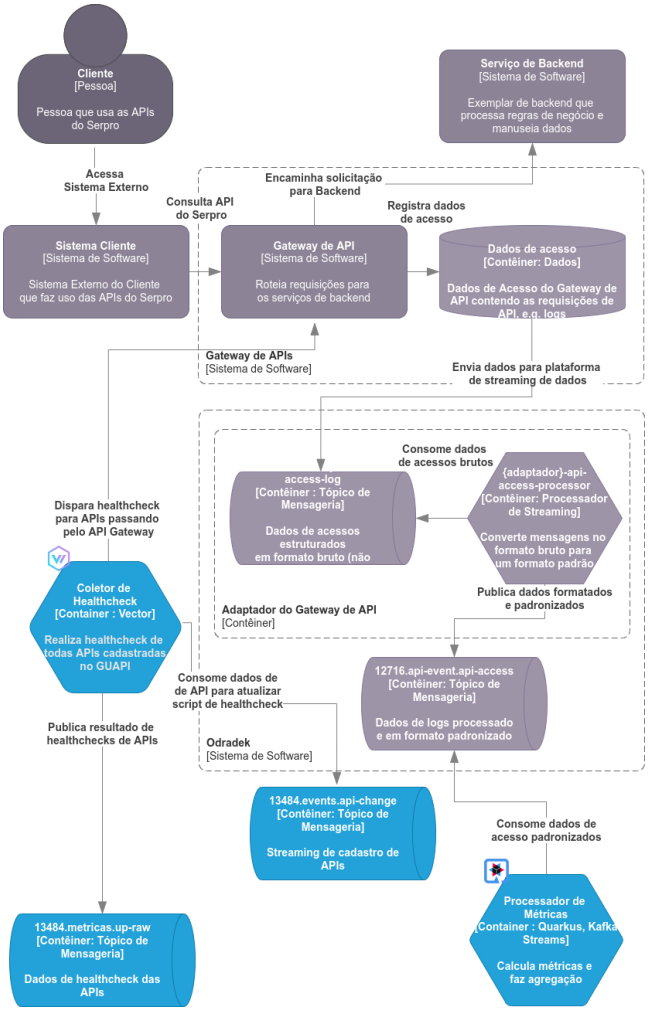

Figura 4: Visão arquitetural da camada de coleta.

metricas.up-raw. Esse componente também é capaz de se reconfigurar dinamicamente, em regime de *hot-load*, a partir de eventos de alteração de configuração recebidos pelo tópico *events.api-change*.

3.3 Visão de Processamento

O processamento dos dados, retratado na Figura 5, é realizado pelo Processador de Métricas, uma aplicação desenvolvida em Quarkus, na linguagem Java, com uso de Kafka Streams. Esse componente consome eventos de duas fontes principais: os dados de disponibilidade publicados em *metricas.up-raw* e os logs de acesso padronizados disponibilizados em *api-event-api-acesso*.

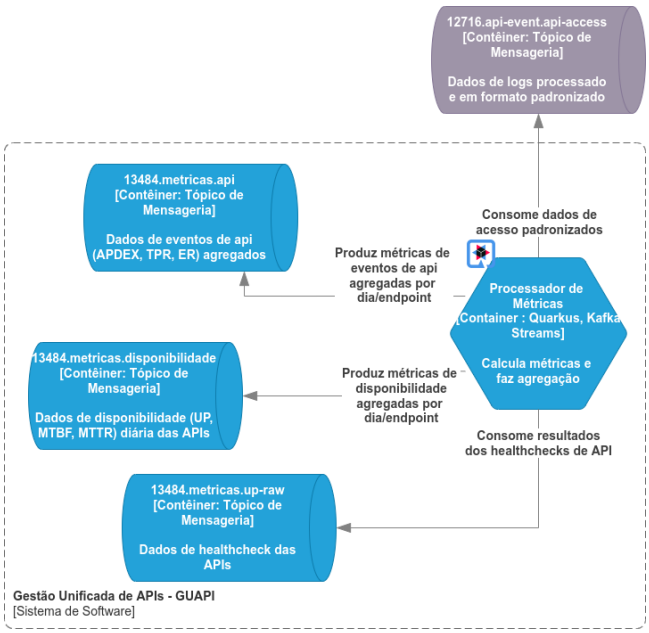

Figura 5: Visão arquitetural da camada de processamento.

A partir dos eventos de disponibilidade, são calculados indicadores como *uptime*, MTBF e MTTR. Já os eventos de acesso permitem a geração de métricas de desempenho, como APDEX, taxa de erros e vazão. As agregações são realizadas em janelas de tempo, com manutenção de estado local por meio de *stores* embutidos no Kafka Streams e tolerância a falhas provida por tópicos de *changelog*. Ao final do processamento, os resultados são publicados nos tópicos *metricas.api*, para indicadores de desempenho, e *metricas.disponibilidade*, para indicadores de disponibilidade.

3.4 Visão de Persistência

A persistência das métricas consolidadas é realizada pelo Kafka Connect, por meio de conectores JDBC Sink. Esses conectores consomem continuamente os tópicos *metricas.api* e *metricas.disponibilidade*, nos quais são publicados, respectivamente, os indicadores de desempenho e disponibilidade produzidos pela camada de processamento. Os dados processados são gravados automaticamente no banco PostgreSQL do GUAPI, estruturando uma base persistente para consulta histórica e visualização dos indicadores, conforme ilustrado na Figura 6.

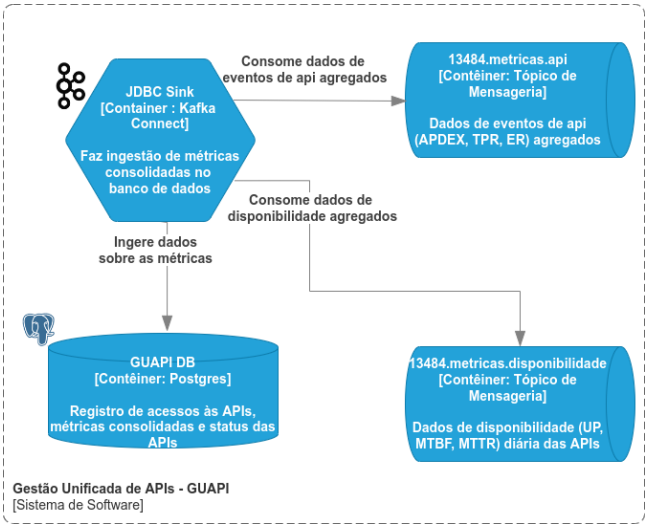

Figura 6: Visão arquitetural da camada de persistência.

Essa estratégia reduz o acoplamento entre o processamento em fluxo e a camada de consulta, permitindo que o *Backend-for-Frontend* acesse dados persistidos e atualizados para disponibilizá-los às interfaces do sistema.

3.5 Visão de Apresentação e Gestão

A Figura 7 mostra os componentes de apresentação e gestão do GUAPI.

A camada de apresentação compreende as interfaces de usuário e os componentes responsáveis pelo consumo dos dados processados. O *frontend*, desenvolvido em Angular, é composto por uma interface administrativa e por um painel interno de visualização. A interface administrativa permite cadastrar APIs, configurar métricas e registrar indisponibilidades programadas, enquanto o painel interno apresenta os *dashboards* com os indicadores consolidados de desempenho e disponibilidade.

O *Backend-for-Frontend*, desenvolvido em NestJS, atua como intermediário entre as interfaces e os serviços de dados. Além de consultar o PostgreSQL para fornecer métricas atualizadas ao *frontend*, esse componente gerencia os metadados das APIs e publica eventos de mudança de configuração no tópico Kafka *events.api-change* sempre que uma API é incluída ou alterada.

Como suporte às funcionalidades de governança, o GUAPI utiliza um *bucket* Ceph para armazenar especificações OpenAPI importadas do catálogo de APIs da organização. A integração com o Serpro LLM viabiliza a funcionalidade de análise com IA, responsável por gerar interpretações textuais automáticas sobre métricas e histórico de disponibilidade. Adicionalmente, a integração opcional com o PLIN, outra fonte corporativa de catálogo de APIs, permite a recuperação de metadados complementares.

Toda a infraestrutura é monitorada por Prometheus e Grafana, enquanto o rastreamento de erros dos componentes de *backend* é realizado com apoio do Sentry. Em conjunto, esses recursos ampliam a observabilidade da solução e favorecem a identificação de falhas operacionais.

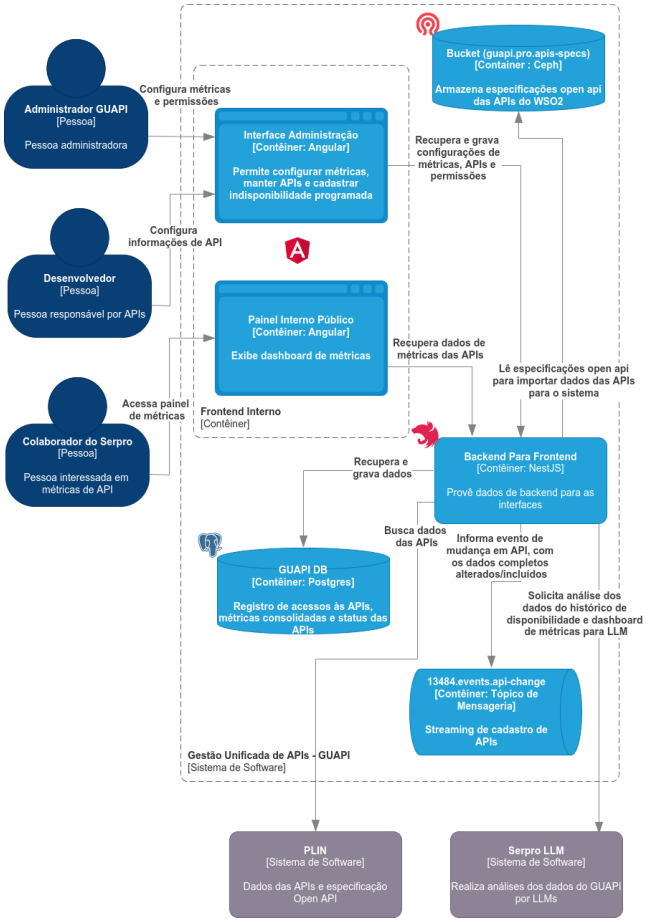

Figura 7: Visão arquitetural da camada de apresentação e gestão.

A arquitetura resultante é desacoplada e orientada a eventos, permitindo que os componentes de coleta, processamento, persistência e apresentação evoluam e sejam dimensionados de forma independente. Essa organização contribui para a escalabilidade, a resiliência e a confiabilidade do GUAPI no monitoramento das APIs corporativas.

4 Ferramentas Relacionadas

Diversas ferramentas e plataformas têm sido propostas para apoiar o monitoramento, a gestão e a governança de APIs. Mapeamentos sistemáticos recentes [4] e análises comparativas [15] confirmam que essas soluções tendem a focar em aspectos específicos do ciclo de vida das APIs e frequentemente abordam apenas um subconjunto dos desafios observados em ambientes críticos e de grande escala.

Ferramentas de monitoramento de disponibilidade, como o Instatus, oferecem páginas de status e verificação de *uptime*, medindo tempos de resposta e acessibilidade a partir de sondas externas a cada 30 segundos. Embora úteis para comunicar a disponibilidade aos consumidores, essas ferramentas não capturam características internas de execução nem oferecem indicadores mais profundos de

qualidade, como tendências de confiabilidade, métricas de *onboarding* ou governança [10].

Soluções voltadas à otimização de desempenho em nível de rede ou *gateway*, como o Akamai API Acceleration, buscam reduzir latência por meio de técnicas de *cache* e otimização em borda. Embora eficazes na melhoria dos tempos de resposta, oferecem visibilidade limitada sobre o comportamento do *backend* e não abordam a evolução da qualidade ao longo do tempo.

Plataformas abrangentes de gestão de APIs, como Google Apigee, MuleSoft Anypoint Platform, Kong Gateway, Amazon API Gateway e Azure API Management, oferecem funcionalidades completas de ciclo de vida, *design*, implantação, segurança, controle de tráfego e monetização. Elas, porém, estão tipicamente acopladas à sua própria infraestrutura de *gateway* [15], o que pode gerar aprisionamento tecnológico em ambientes heterogêneos, situação comum no setor público, onde a interoperabilidade entre sistemas heterogêneos é um requisito fundamental [9]. Existem altos custos de licenciamento atrelados, curvas de aprendizado acentuadas e significativo esforço de configuração. Mais recentemente, plataformas como Sensedia e Gravitee introduziram camadas de governança *multi-gateway*, consolidando catálogos e métricas entre provedores distintos.

O GUAPI diferencia-se dessas soluções em três aspectos-chave:

(1) É **agnóstico a *gateways*** por *design*, consumindo logs de infraestruturas heterogêneas já existentes, sem exigir substituição ou migração de *gateways*.
(2) Em vez de cobrir todo o ciclo de vida das APIs, foca especificamente na **governança da qualidade operacional**, consolidando indicadores de produção em uma visão analítica unificada.
(3) Utiliza uma **arquitetura orientada a eventos** baseada em Apache Kafka, Kafka Connect e Kafka Streams para processamento contínuo em tempo quase real, em vez de depender de análises em lote ou *polling* periódico.

Nesse sentido, o GUAPI não substitui plataformas de gestão de APIs, mas atua como uma camada complementar de governança que endereça uma lacuna frequentemente observada em ambientes de grande escala com múltiplos *gateways*: a ausência de indicadores de qualidade unificados, em tempo real e comparáveis entre todas as APIs em produção.

5 Conclusão

Este artigo apresentou o GUAPI, uma ferramenta unificada e agnóstica a *gateways* para governança de APIs em ambientes críticos e de grande escala. A plataforma integra indicadores de desempenho, disponibilidade, confiabilidade, experiência do usuário e segurança em um único modelo analítico, apoiando decisões orientadas por dados e ciclos de melhoria contínua.

Em produção no Serpro desde agosto de 2025, o GUAPI processa mais de 15 milhões de eventos diários, com pico mensal de 537 milhões de requisições em março de 2026. Entre agosto de 2025 e abril de 2026, os indicadores apresentaram melhora: o APDEX mediano aumentou de 93,74 para 95,80, a duração mediana das requisições caiu de 261,92 ms para 174,25 ms, e a taxa mediana de erros reduziu de 12,98% para 8,65%. No mesmo período, o tempo mensal de indisponibilidade caiu de 1.486 para 1.057 minutos, os eventos de indisponibilidade reduziram de 813 para 679, o MTTR diário mediano caiu de 0,57 para 0,33 minutos, e o MTBF diário mediano aumentou de 1.266,06 para 1.366,28 minutos. Esses resultados indicam melhora nos indicadores e apoiam o acompanhamento e a priorização de melhorias.

Embora concebido e validado no contexto do setor público brasileiro, o GUAPI endereça desafios comuns a organizações que operam ecossistemas de APIs em escala, como observabilidade fragmentada, ausência de indicadores unificados e dificuldade de governança em infraestruturas heterogêneas. Assim, sua arquitetura de referência pode ser aplicada em diferentes setores da indústria, incluindo financeiro, telecomunicação, *e-commerce*, etc., nos quais métricas unificadas, processamento baseado em *event streaming*, transparência operacional e ciclos contínuos de *feedback* também são fundamentais para a governança orientada por dados.

Como trabalhos futuros, a plataforma está sendo estendida com métricas voltadas à experiência do consumidor (*First-Success Bounce Rate*, *Time to First Call*), detecção de anomalias (*Anomaly Detection Rate*) e qualidade de especificações (*Static Analysis Score*), ampliando a cobertura para dimensões de adoção, comportamento e design.

Disponibilidade de Artefatos

O GUAPI é distribuído sob licença proprietária institucional do Serpro, como software de uso interno, em conformidade com as políticas de propriedade intelectual da empresa e com as diretrizes de segurança da informação aplicáveis ao setor público federal brasileiro.

Isso decorre de três fatores principais. Primeiro, a plataforma opera sobre infraestrutura crítica de governo, processando logs de acesso e métricas de disponibilidade de APIs associadas a dados sensíveis do cidadão brasileiro, como CPF, CNPJ, biometria e comércio exterior, o que impõe restrições regulatórias e de conformidade quanto à exposição de detalhes internos de arquitetura e configuração. Segundo, o GUAPI integra-se a sistemas internos do Serpro, como catálogo corporativo de APIs e SerproLLM, cujas interfaces e contratos não são públicos, dificultando a reprodução direta do código fora desse ecossistema. Terceiro, por ter sido desenvolvido no âmbito do Prêmio Serpro de Inovação, o GUAPI está sujeito às cláusulas de propriedade intelectual do programa, que atribuem os direitos patrimoniais à empresa pública.

Apesar do caráter proprietário do código-fonte, a plataforma é construída sobre tecnologias de código aberto, como Apache Kafka, PostgreSQL, Quarkus, NestJS, Angular, Vector, Prometheus e Grafana. A arquitetura de referência apresentada neste artigo pode ser reproduzida e adaptada por outras organizações. Assim, embora o software em si seja proprietário, o conhecimento arquitetural e as decisões de projeto permanecem abertos à comunidade.

Como trabalho futuro, os autores avaliam a possibilidade de disponibilizar módulos isolados da plataforma, como processadores Kafka Streams para cálculo de métricas, sob licença de código aberto, mediante aprovação institucional.

Agradecimentos

Os autores gostariam de agradecer às equipes do Serpro envolvidas no projeto, desenvolvimento e operação da plataforma GUAPI, bem como ao programa interno “Prêmio Serpro de Inovação” que permitiu que este trabalho fosse desenvolvido com sucesso.

Referências

[1] Mak Ahmad, J. J. Geewax, Andrew Macvean, David R. Karger, and Kwan-Liu Ma. 2024. API Governance at Scale. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '24)*. ACM, Lisbon, Portugal, 430–440. doi:10.1145/3639477.3639713

[2] Apdex Alliance. 2007. *Application Performance Index — Apdex Technical Specification, Version 1.1*. Technical Report. Apdex Alliance, Inc. https://www.apdex.org/wp-content/uploads/2020/09/ApdexTechnicalSpecificationV11_000.pdf Accessed: 2026-05-06.

[3] David Bermbach and Erik Wittern. 2020. Benchmarking Web API Quality — Revisited. *Journal of Web Engineering* 19, 5–6 (2020), 603–646.

[4] Eder dos Santos and Sandra Casas. 2024. Unveiling Quality in API Management: A Systematic Mapping Study. In *2024 L Latin American Computer Conference (CLEI)*. IEEE, 1–10. doi:10.1109/clei64178.2024.10700447

[5] Susan J. Fowler. 2017. *Production-Ready Microservices*. O'Reilly Media.

[6] Joshua J. Geewax. 2021. *API Design Patterns*. Manning Publications.

[7] IBM. 2026. What is API governance? https://www.ibm.com/think/topics/api-governance Accessed: 2026-05-06.

[8] Matthew Jones, Mahdi Bayesh, and Shahrzad Jahan. 2024. Unlocking Deeper Understanding: Leveraging Explainable AI for API Anomaly Detection. *Proceedings of the International Conference on Machine Learning and Computing* (2024).

[9] Keegan McBride, Sujani Kamalanathan, Sandhra-Mirella Valdma, Taavi Toomere, and Margus Freudenthal. 2022. Digital Government Interoperability and Data Exchange Platforms: Insights from a Twenty Country Comparative Study. In *Proceedings of the 15th International Conference on Theory and Practice of Electronic Governance* (Guimarães, Portugal) *(ICEGOV '22)*. Association for Computing Machinery, New York, NY, USA, 90–97. doi:10.1145/3560107.3560123

[10] Mehdi Medjaoui, Erik Wilde, Ronnie Mitra, and Mike Amundsen. 2021. *Continuous API Management*. O'Reilly Media.

[11] Sanjiv Pillai et al. 2024. *API Management Market Guide*. Technical Report. Gartner.

[12] Leonard Richardson, Mike Amundsen, and Sam Ruby. 2019. *RESTful Web APIs*. O'Reilly Media.

[13] Aumir Shabbir, Aziz Deraman, Mohamad Nor Bin Hassan, Kamal Uddin Sarker, and Shahid Kamal. 2026. A novel NFR-based conceptual quality framework for modern API industry. *Scientific Reports* 16 (2026). doi:10.1038/s41598-026-48021-4

[14] United Nations Department of Economic and Social Affairs. 2024. *United Nations E-Government Survey 2024: Accelerating Digital Transformation for Sustainable Development*. Technical Report. United Nations, New York. https://publicadministration.un.org/egovkb/en-us/Reports/UN-E-Government-Survey-2024 Accessed: 2026-05-06.

[15] Padmanabhan Venkiteela. 2025. Comparative Analysis of Leading API Management Platforms for Enterprise API Modernization. *International Journal of Computer Applications* 187, 54 (2025).

[16] Erik Wittern, Philipp Leitner, and Stefan Dustdar. 2017. An Empirical Study on the Evolution of Web APIs. *Software: Practice and Experience* (2017).