Uma Abordagem Baseada em LLMs para Auditoria Contínua de Conformidade com a LGPD em Microsserviços Integrada a Pipelines CI/CD

Lucas Matheus Silva[1], **Danyllo Albuquerque**[1]
Emanuel Dantas Filho[1,2], **Bruno Neiva Moreno**[1]

[1] Instituto Federal da Paraíba (IFPB) – Laboratório de Inovação em Software (LIS)

[2]Instituto Federal de Pernambuco (IFPE)

{lucas.silva.1,danyllo.albuquerque}@academico.ifpb.edu.br
emanuel.filho@jaboatao.ifpe.edu.br

Abstract. *Microservices-based systems increase the challenges of complying with Brazil's General Data Protection Law (LGPD), as the distribution of data processing across multiple services and the high frequency of changes make manual auditing poorly scalable. In addition, traditional static analysis tools have limited capability to semantically interpret legal requirements. This work proposes a continuous auditing architecture that combines static detection and LLM-based semantic analysis, supported by Retrieval-Augmented Generation (RAG) over LGPD provisions and integrated into CI/CD pipelines as a GitHub Action. The LGPD Guard prototype was evaluated in an exploratory study on three open-source repositories, including two Brazilian government systems in production. The results provide preliminary evidence that the LLM+RAG layer can complement the static detector by helping identify additional potential violations and by supporting the incorporation of privacy by design into DevSecOps pipelines.*

Resumo. *Sistemas baseados em microsserviços ampliam os desafios de conformidade com a LGPD, pois a distribuição do tratamento de dados e a alta frequência de mudanças tornam auditorias manuais pouco escaláveis. Além disso, ferramentas tradicionais de análise estática possuem limitações para interpretar requisitos legais de forma semântica. Este trabalho propõe uma arquitetura de auditoria contínua que combina detecção estática e análise semântica baseada em LLMs, apoiada em Retrieval-Augmented Generation (RAG) sobre dispositivos da LGPD e integrada a pipelines CI/CD como GitHub Action. O protótipo LGPD Guard foi avaliado em um estudo exploratório com três repositórios de código aberto, incluindo dois sistemas governamentais brasileiros em produção. Os resultados fornecem evidências preliminares de que a camada LLM+RAG pode complementar o detector estático, ajudando a identificar possíveis violações adicionais e a apoiar a incorporação de privacy by design em pipelines de DevSecOps.*

1. Introdução

Arquiteturas baseadas em microsserviços, frequentemente combinadas com práticas de DevOps e CI/CD, têm permitido ciclos de desenvolvimento e entrega cada vez mais curtos, mas também ampliam desafios de governança e conformidade em ambientes distribuídos [Dakić 2024]. No contexto brasileiro, a Lei Geral de Proteção de Dados (LGPD), em vigor desde 2020, estabelece obrigações para sistemas que coletam, armazenam ou processam dados pessoais. Garantir a conformidade nesses ambientes é particularmente complexo, pois o tratamento de dados tende a estar distribuído entre múltiplos serviços, linguagens e repositórios, frequentemente mantidos por equipes distintas. Além disso, ações recentes de fiscalização pela Autoridade Nacional de Proteção de Dados (ANPD) indicam uma intensificação da aplicação da lei, com sanções que podem alcançar até 2% do faturamento da organização, limitadas a R$ 50 milhões por infração [Chambers and Partners 2025].

Na prática, auditorias de conformidade ainda são majoritariamente conduzidas de forma manual ou apoiadas por ferramentas tradicionais de análise estática. Embora úteis para identificar vulnerabilidades técnicas, essas ferramentas possuem limitações para interpretar requisitos legais que dependem de contexto e significado semântico do código, o que pode levar à identificação tardia de problemas de conformidade [Amaral et al. 2022]. Paralelamente, avanços recentes em Modelos de Linguagem de Grande Escala (LLMs) têm demonstrado potencial para apoiar atividades de engenharia de software, como revisão de código, análise de requisitos e detecção de vulnerabilidades [Li et al. 2024]. Alguns estudos recentes também investigaram o uso desses modelos em tarefas relacionadas à conformidade regulatória, especialmente no contexto do GDPR [Rodriguez et al. 2024]. Apesar desses avanços, trabalhos existentes ainda tendem a tratar separadamente conformidade regulatória, análise estática e integração a pipelines CI/CD [El Hamdani et al. 2021, Rodriguez et al. 2024, Chen 2024, Johnson et al. 2013, Dakić 2024]. Ainda são escassos estudos que integrem simultaneamente três dimensões relevantes: (i) foco explícito na LGPD brasileira; (ii) suporte à auditoria contínua integrada a pipelines CI/CD; e (iii) aplicação em cenários de software distribuído, como arquiteturas baseadas em microsserviços. A articulação dessas três dimensões constitui a principal motivação deste trabalho.

Neste artigo, investigamos o uso de LLMs integrados a pipelines CI/CD como mecanismo de apoio à auditoria automatizada de código-fonte em arquiteturas de microsserviços, com foco na conformidade com a LGPD. A proposta é apresentada por meio de uma arquitetura conceitual, da implementação de um protótipo funcional e de uma avaliação exploratória conduzida em repositórios reais de código governamental aberto. Como estratégia central de interação com o modelo, adotamos a combinação de *Role Prompting* e *Retrieval-Augmented Generation* (RAG), abordagem adequada a tarefas que exigem alinhamento entre conhecimento externo e geração de respostas consistentes [Girardi et al. 2025]. As principais contribuições deste trabalho são:

- Uma abordagem de Engenharia de Software Inteligente voltada à auditoria de conformidade com a LGPD em arquiteturas baseadas em microsserviços, integrável a pipelines CI/CD;
- Uma arquitetura híbrida que combina detecção estática, recuperação de contexto regulatório via RAG e análise semântica baseada em LLMs;
- O protótipo operacional *LGPD Guard*, implementado como GitHub Action e avaliado em três repositórios de código aberto, incluindo dois sistemas governamentais brasileiros em produção; e
- Evidências preliminares de que a camada LLM+RAG pode complementar o detector estático, ajudando a identificar potenciais violações adicionais, especialmente em casos de natureza semântica ou de contexto insuficiente, associados ao resultado *incerto* da função $\mathcal{F}$.

2. Contextualização

Esta seção apresenta o contexto que fundamenta o problema investigado, explicita a questão de pesquisa que orienta o estudo, introduz um exemplo motivador e formaliza o problema de auditoria contínua de conformidade adotado neste trabalho.

2.1. Definição da Questão de Pesquisa

A conformidade com a LGPD exige que princípios legais abstratos, como minimização de dados e limitação de finalidade, sejam traduzidos em decisões técnicas concretas. Em arquiteturas baseadas em microsserviços e práticas de DevOps, nas quais o tratamento de dados é distribuído entre múltiplos serviços e repositórios, auditorias pontuais tornam-se rapidamente obsoletas [Donda 2021]. Além disso, a LGPD exige que controladores mantenham registros das operações de tratamento (Art. 37) e implementem medidas técnicas e administrativas para

proteger dados pessoais (Art. 46), requisitos difíceis de garantir de forma sistemática em ambientes distribuídos [Chambers and Partners 2025]. Ferramentas tradicionais de análise estática também apresentam limitações para capturar violações que dependem de interpretação contextual de requisitos legais [El Hamdani et al. 2021]. Nesse contexto, LLMs integrados a pipelines CI/CD surgem como uma alternativa promissora para apoiar auditorias de conformidade de forma contínua [Chen 2024].

Diante desse cenário, este trabalho investiga a seguinte questão de pesquisa: *Como LLMs podem ser integrados a pipelines CI/CD para apoiar a auditoria automatizada de código-fonte voltada à conformidade com a LGPD em arquiteturas de microsserviços?* Responder a essa questão é relevante para avançar a integração entre engenharia de software, conformidade regulatória e inteligência artificial, permitindo analisar em que medida mecanismos automatizados podem apoiar a identificação precoce de riscos regulatórios durante o ciclo de desenvolvimento.

2.2. Exemplo Motivador

Para ilustrar o problema, considere um microsserviço de rastreamento de checkout em uma aplicação de comércio eletrônico. Durante um ciclo normal de desenvolvimento, um desenvolvedor submete um *pull request* (PR) adicionando instrumentação de log para depuração. O trecho a seguir, parte do diff em `examples/violations/checkout_tracking.js`, é adicionado à branch:

```
// L6 -- log de depuração
console.log(`email=${usuario.email} cpf=${usuario.cpf}`);

// L17 -- chamada ao serviço de pagamentos
fetch(`https://payments.exemplo.com/process?cpf=${usuario.cpf}`);
```

O código é sintaticamente correto, passa em todos os testes automatizados e não apresenta vulnerabilidades clássicas de segurança. Ferramentas SAST tradicionais tendem a não sinalizar esse tipo de problema regulatório, pois seu foco recai principalmente sobre padrões sintáticos inseguros e vulnerabilidades técnicas previamente especificadas [Johnson et al. 2013]. Ainda assim, o trecho apresenta indícios relevantes de não conformidade com a LGPD: (i) exposição de CPF e e-mail em logs de aplicação sem mascaramento, potencialmente em desacordo com requisitos de proteção de dados pessoais (Art. 46); e (ii) transmissão de CPF como parâmetro em *query string*, o que amplia a superfície de exposição do dado em *access logs*, proxies reversos e cabeçalhos HTTP `Referer`, podendo violar os princípios de necessidade e segurança (Arts. 6º e 46). Observe-se que `encodeURIComponent()` apenas codifica caracteres especiais, não criptografando o dado.

Com o LGPD Guard integrado ao pipeline CI/CD, a abertura do PR dispara automaticamente o fluxo de auditoria via GitHub Actions: (i) extração do diff; (ii) detecção de padrões como `cpf` e `email`; (iii) recuperação de dispositivos legais relevantes na base RAG; e (iv) publicação de um comentário estruturado no PR contendo arquivo, linha, dispositivo legal associado e sugestão de correção, podendo bloquear o *merge*. Esse exemplo ilustra como indícios de não conformidade regulatória, pouco acessíveis a mecanismos técnicos tradicionais, podem ser sinalizados continuamente durante o ciclo de desenvolvimento.

Esse cenário motivador evidencia a necessidade de mecanismos automatizados capazes de identificar indícios de não conformidade regulatória durante o processo de integração contínua. A subseção seguinte apresenta uma formalização do problema de auditoria contínua considerado neste trabalho.

2.3. Definição Formal do Problema

Considere $\mathcal{S} = \{m_1, \ldots, m_n\}$ um sistema baseado em microsserviços, no qual cada serviço m_i possui um conjunto de artefatos $\mathcal{A}_i$. O conjunto total de artefatos do sistema é dado por

$\mathcal{A} = \bigcup_{i=1}^{n} \mathcal{A}_i$. Uma alteração $\Delta \subseteq \mathcal{A}$ representa o *diff* associado a um *pull request*. Seja $\mathcal{R}$ o conjunto de requisitos operacionais derivados da LGPD, incluindo princípios legais (Arts. 6 e 7) e obrigações técnicas relacionadas à proteção de dados pessoais (Art. 46).

O problema consiste em determinar, dado Δ, se a alteração introduz evidências de violação de algum requisito $r_j \in \mathcal{R}$. Formalmente, essa verificação pode ser representada pela função de auditoria:

$$\mathcal{F} : (\Delta, \mathcal{R}) \rightarrow \{\textit{conforme},\ \textit{não-conforme},\ \textit{incerto}\}$$

em que $\mathcal{F}$ classifica a alteração analisada de acordo com seu potencial impacto na conformidade com a LGPD. O resultado *incerto* é tratado de forma conservadora como *aprovar com alertas*: o *merge* não é bloqueado automaticamente, mas o desenvolvedor é notificado para revisão humana, preservando a agilidade do processo de integração contínua sem comprometer a rastreabilidade regulatória. Em arquiteturas distribuídas, essa avaliação deve considerar não apenas o trecho de código modificado, mas também o contexto técnico da alteração, incluindo artefatos relacionados, metadados e possíveis fluxos de dados entre serviços.

3. Trabalhos Relacionados

Esta seção revisa trabalhos relacionados ao uso de técnicas automatizadas e LLMs para apoio à conformidade regulatória em software. A literatura é discutida em três eixos: conformidade automatizada, uso de LLMs em repositórios e pipelines CI/CD, e aplicações de RAG em análise jurídica. Ao final, sintetizamos a lacuna que motiva a proposta deste trabalho.

Conformidade Automatizada e LLMs. Amaral et al. [Amaral et al. 2022] propuseram IA para verificação de completude de políticas de privacidade, alcançando resultados promissores na identificação de lacunas documentais. Hamdani et al. [El Hamdani et al. 2021] combinaram regras e aprendizado de máquina para conformidade com GDPR, demonstrando que métodos híbridos tendem a superar abordagens puramente baseadas em regras. No contexto brasileiro, estudos apontam que 81% das entidades auditadas apresentam conformidade LGPD insignificante [Chambers and Partners 2025]. Rodriguez et al. [Rodriguez et al. 2024] investigaram LLMs na análise de políticas de privacidade em larga escala, mas o foco permanece em análises pontuais sobre artefatos documentais, sem atuação sobre diffs de código nem integração contínua a pipelines CI/CD orientados a código-fonte.

LLMs em Repositórios e Pipelines CI/CD. Gloaguen et al. [Gloaguen et al. 2026] avaliaram o impacto de arquivos de contexto (`AGENTS.md`, `CLAUDE.md`) sobre agentes de codificação em repositórios reais, observando que contextos gerados por LLMs podem reduzir o desempenho e aumentar custos, enquanto contextos humanos mínimos e específicos produzem ganhos marginais. O estudo restringiu-se a repositórios Python de propósito geral, sem investigar conformidade jurídica, lacuna que o presente trabalho busca endereçar via RAG sobre dispositivos legais explícitos da LGPD. No front de segurança, ferramentas SAST/SCA como SonarQube, Semgrep e Trivy são amplamente usadas em DevSecOps, mas focam em segurança técnica (CVEs, injeções), sem considerar requisitos regulatórios [Johnson et al. 2013]: um sistema pode estar tecnicamente seguro e ainda apresentar indícios de violação da LGPD ao registrar dados sensíveis em logs sem mascaramento (Art. 46), distinção central desta proposta.

RAG para Conformidade Legal. Girardi et al. [Girardi et al. 2025] mostraram que a combinação de *Role Prompting* e RAG alcança 98% de similaridade semântica (BERTScore) em geração de código Python, superando combinações mais complexas, o que sugere que arquiteturas enxutas de prompt podem ser eficazes [Schulhoff et al. 2025]. Abualhaija et al. [Abualhaija et al. 2025] demonstraram RAG com LangChain para extração de requisitos do GDPR (chunking de 1.000 caracteres, overlap de 200), metodologia adotada neste trabalho. Chen et al. [Chen et al. 2025] propuseram um framework de três camadas (conhecimento estático, regulações dinâmicas e raciocínio contextual) que serve de referência arquitetural.

Além disso, estudos indicam que LLMs sem recuperação de contexto externo podem alucinar com frequência em tarefas jurídicas [Harvard Journal of Law & Technology 2025], o que reforça a adoção de RAG sobre dispositivos legais explícitos como escolha metodológica desta proposta.

Síntese e Lacuna de Pesquisa. A Tabela 1 sintetiza os trabalhos e evidencia três dimensões ainda pouco articuladas conjuntamente: foco na LGPD, integração CI/CD e suporte a microsserviços. Estudos de conformidade tendem a concentrar-se em políticas e documentos; estudos sobre repositórios raramente consideram requisitos legais; e ferramentas integradas a pipelines CI/CD permanecem centradas em segurança técnica. Esta proposta emerge da articulação simultânea desses três eixos em um cenário de auditoria contínua orientada a código-fonte.

Tabela 1. Comparação com trabalhos relacionados.

Trabalho	LGPD	CI/CD	Microserv.	Conf. Legal
Amaral et al. [Amaral et al. 2022]	✓	×	×	✓
Hamdani et al. [El Hamdani et al. 2021]	×	×	×	✓
Rodriguez et al. [Rodriguez et al. 2024]	×	×	×	✓
Chen [Chen 2024]	×	✓	×	×
Gloaguen et al. [Gloaguen et al. 2026]	×	✓	×	×
SAST/SCA (ex: Trivy)	×	✓	✓	×
Esta proposta	✓	✓	✓	✓

4. Abordagem Proposta

Esta seção apresenta a arquitetura do *LGPD Guard*, seus principais componentes e o fluxo de auditoria executado no pipeline CI/CD.

Visão Geral da Arquitetura. A Figura 1 ilustra o fluxo da abordagem em cinco etapas: (1) identificação de trechos que potencialmente processam dados pessoais por análise sintática determinística; (2) extração de contexto e metadados do diff; (3) recuperação de dispositivos legais relevantes via RAG; (4) análise semântica com apoio de LLM; e (5) geração de relatório e sinalização de bloqueio ou aprovação com alertas. Ao submeter um PR, essas etapas são executadas automaticamente no pipeline.

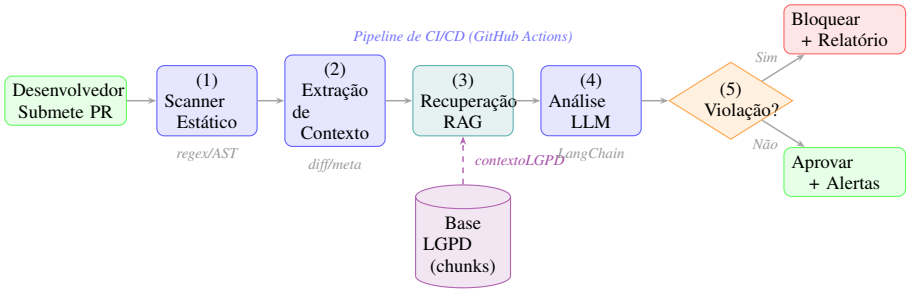

Figura 1. Arquitetura do *LGPD Guard*: fluxo de auditoria em 5 etapas no pipeline CI/CD. O desenvolvedor submete um PR (*pull request*); as etapas (1)–(5) são executadas automaticamente via GitHub Actions.

Componentes Principais. Os componentes do LGPD Guard implementam as etapas centrais do fluxo apresentado na Figura 1. Nesta descrição conceitual, os componentes são apresentados pelo papel arquitetural, sem depender dos nomes específicos dos arquivos da implementação.

O *Scanner Estático* analisa o diff do pull request por meio de expressões regulares e heurísticas léxicas, identificando padrões de dados pessoais (CPF, e-mail, IP, nome) e violações estruturais, como HTTP sem TLS, credenciais *hardcoded* e dados em URLs. Opera deterministicamente, sem invocação de LLM, favorecendo baixa latência e maior reprodutibilidade dos resultados.

A *Extração de Contexto* extrai metadados da alteração, como nome do arquivo, tipo de operação e linguagem, e monta a representação estruturada do diff que alimentará o módulo RAG.

A *Recuperação RAG* consulta a base vetorial com artigos da LGPD segmentados em chunks de 1.000 caracteres, com overlap de 200 [Abualhaija et al. 2025], ancorando a análise semântica em textos legais explícitos e buscando mitigar alucinações [Harvard Journal of Law & Technology 2025]. Na versão atual do protótipo, a base contempla os Arts. 6, 7, 15, 18, 20, 37, 46 e 48, organizada com base na referência arquitetural de Chen et al. [Chen et al. 2025].

A *Análise LLM e Geração de Relatório* produz uma classificação (*conforme/não-conforme/incerto*), um nível de severidade (Crítica/Alta/Média) e uma sugestão de correção; publica comentário estruturado no PR e sinaliza bloqueio quando a severidade máxima for Alta ou Crítica.

Em conjunto, esses componentes compõem o fluxo de auditoria contínua executado no pipeline de CI/CD. A subseção seguinte apresenta o algoritmo que formaliza esse processo.

Algoritmo de Auditoria Contínua. O Algoritmo 1 resume o pipeline executado a cada evento de integração contínua. A separação entre etapas determinísticas (detecção e recuperação) e probabilísticas (inferência via LLM) reforça a arquitetura híbrida proposta.

Algorithm 1 LGPD-Guard-Audit

Require: Δ (diff do PR), $\mathcal{R}$ (requisitos LGPD)
Ensure: Relatório de conformidade, `severidade_max`
1: $T \leftarrow$ DETECTARTRECHOS(Δ)
2: $E \leftarrow \emptyset$
3: `severidade_max` $\leftarrow$ "nenhuma"
4: **for all** $t \in T$ **do**
5: $D \leftarrow$ DETECTARDADOSPESSOAIS(t) ▷ regex/AST
6: $C \leftarrow$ RECUPERARCONTEXTOLGPD($D, \mathcal{R}$) ▷ RAG
7: $\langle J, sev \rangle \leftarrow$ AVALIARCOMLLM(t, C) ▷ LLM
8: $E \leftarrow E \cup \{J\}$
9: **if** $sev >$ `severidade_max` **then**
10: `severidade_max` $\leftarrow sev$ ▷ {média, alta, crítica}
11: **end if**
12: **end for**
13: Relatório $\leftarrow$ GERARRELATÓRIO(E, `severidade_max`)
14: **if** `severidade_max` $\in$ {"alta", "crítica"} **then**
15: BLOQUEARMERGE(Relatório)
16: **else**
17: APROVARCOMALERTAS(Relatório)
18: **end if**

O algoritmo inicia identificando trechos relevantes do diff por meio de análise determinística baseada em regex e AST. Para cada trecho identificado, são detectados possíveis dados pessoais e recuperados dispositivos legais relevantes da LGPD via RAG. A função AVALIARCOMLLM realiza a análise semântica, produzindo uma classificação J e um nível de severidade sev. A variável `severidade_max` mantém o maior nível de severidade observado entre os achados e orienta, nesta versão do protótipo, a decisão final de bloqueio ou aprovação do *merge* (linha 15).

Esse procedimento operacionaliza a função de auditoria definida na Seção 2, sendo instanciado e avaliado experimentalmente conforme descrito na Seção 5.

5. Avaliação da Abordagem

Para explorar a viabilidade técnica da proposta, foi desenvolvido o *LGPD Guard* como GitHub Action[1], disponível em repositório público (Seção 8), permitindo inspecionar a implementação e reproduzir os experimentos.

[1] `https://github.com/features/actions`

Configuração do protótipo. O protótipo é composto por quatro componentes funcionais: análise estática por expressões regulares sobre o diff do PR, análise semântica usando LangChain com RAG (*Retrieval-Augmented Generation*) [Lewis et al. 2020] sobre o texto da LGPD, formatação dos resultados como comentário estruturado no PR e orquestração do pipeline completo. A camada LLM utiliza `claude-sonnet-4-6` (`temperature=0`) como provedor padrão, com `gpt-4o` como alternativa.

A Tabela 2 apresenta os sete padrões de violação considerados, definidos com base em recorrência técnica [El Hamdani et al. 2021, Johnson et al. 2013] e em sua relevância regulatória no contexto da LGPD [Chambers and Partners 2025]. Esses padrões constituem as entradas detectáveis pelo protótipo na versão atual.

Tabela 2. Padrões de violação detectáveis pelo protótipo atual do LGPD Guard.

Violação	Artigo	Sev.
Dado pessoal em log sem mascaramento	Art. 46	Alta
Credencial *hardcoded*	Art. 46	Crítica
Senha sem hash criptográfico	Art. 46	Crítica
SQL injection com dado pessoal	Art. 46	Crítica
HTTP sem criptografia	Art. 46	Alta
CPF/e-mail enviado para analytics	Arts. 6+7	Alta
Dado pessoal exposto em URL	Arts. 6+46	Média

Configuração da avaliação. A avaliação exploratória foi conduzida em três repositórios públicos no GitHub (Tabela 3), cobrindo tanto um cenário controlado quanto sistemas reais em produção. O repositório *lgpd-guard* foi usado como caso controlado por conter violações intencionalmente construídas, permitindo verificar o comportamento esperado do protótipo. Os repositórios SIGPAE-API e Susep foram escolhidos por serem sistemas governamentais brasileiros de código aberto, com domínios nos quais o tratamento de dados pessoais pode ocorrer em fluxos reais de negócio. O objetivo foi observar se a abordagem produz alertas tecnicamente plausíveis em diferentes contextos, sem a pretensão de estabelecer métricas estatisticamente robustas ou cobertura representativa. Os commits analisados foram selecionados via *pickaxe search* (`git log -S "cpf"`), incluindo exemplos com indícios de dados pessoais e commits de referência para contraste qualitativo.

Tabela 3. Repositórios avaliados.

Repositório	Contexto	Linguagem	Domínio
lgpd-guard[1]	Controlado	Python/JS	CPF, e-mail
SIGPAE-API[2]	Produção (SP)	Python/Django	Alimentação escolar
Susep[3]	Produção (Fed.)	C#/.NET	Gestão financeira

Resultados. A Tabela 4 consolida os resultados de cinco execuções, cada uma correspondente à aplicação do protótipo sobre um PR ou commit específico. O número de execuções foi definido para cobrir, de forma exploratória, um caso controlado, três alterações com indícios de dados pessoais em sistemas reais e um commit de contraste sem achados relevantes. Portanto, as cinco execuções não devem ser interpretadas como repetições estatísticas independentes, mas como cenários qualitativos de avaliação. Nesta análise, o detector estático foi usado como *baseline* operacional, por representar a camada determinística baseada em padrões sintáticos. Consideramos *achados confirmados* aqueles identificados pelo detector estático e inspecionados manualmente pelos autores como tecnicamente plausíveis no diff analisado. Consideramos *achados adicionais* aqueles sinalizados apenas pela camada LLM+RAG, geralmente associados a interpretação semântica ou contexto regulatório não capturado por regras sintáticas. O

[1] `https://github.com/L42Matheus/lgpd-guard`
[2] `https://github.com/prefeiturasp/SME-SIGPAE-API`
[3] `https://github.com/spbgovbr/Sistema_Programa_de_Gestao_Susep`

módulo estático identificou **5 achados confirmados**, enquanto a camada LLM+RAG adicionou **6 achados adicionais**. Em dois dos cinco casos, a severidade máxima resultou em bloqueio do merge. Todos os achados foram avaliados manualmente pelos autores, mas não houve, nesta etapa, validação independente por especialistas jurídicos ou construção de um *ground truth* normativo.

Tabela 4. Resultados da avaliação exploratória do LGPD Guard.

Repositório	Commit/PR	Dados	Viol. Est.	Viol. LLM	Sev. máx.	Pipeline
lgpd-guard	PR#6	5	2	3	Alta	**Bloqueado**
SIGPAE	3fb8d6ba1	2	1	3	Média	Passou
SIGPAE	1e42fbb5a	9	2	0*	Alta	**Bloqueado**
Susep	48f027f†	50	0	2‡	Potencial	Passou
Susep	6bce9af†	0	0	0	Nenhuma	Passou

† Diff filtrado para `src/` (bundle minificado gerava falsos positivos).
* LLM sinalizou no `llm_analysis_raw`; `llm_violations` vazio por JSON truncado.
‡ Classificadas como `POTENCIAL` com `confirmada: false`.

No total, os achados do detector estático concentraram-se em dois tipos principais: `log_com_dado_pessoal` (Alta) e `dado_pessoal_em_url` (Média). Já a camada LLM+RAG evidenciou capacidade de identificar violações dependentes de contexto, ampliando a cobertura da análise. Esses resultados indicam um papel complementar entre as abordagens: enquanto o detector estático captura padrões estruturais explícitos, a camada semântica permite identificar indícios de não conformidade mais sutis.

Análise Qualitativa. A seguir, discutimos casos representativos que evidenciam as diferenças entre as duas camadas de análise.

1. **PR#6 (lgpd-guard), cenário controlado.** O PR introduz `examples/violations/checkout_tracking.js` com violações intencionais. O detector estático identificou 2 achados diretos. A camada LLM classificou risco global como **ALTO** e estruturou três violações: (V001) CPF e e-mail em log sem mascaramento (Art. 46); (V002) CPF em *query string* para serviço externo (Art. 46); (V003) CPF em objeto de *analytics* sem evidência de base legal (Art. 6º, I). O merge foi sinalizado para bloqueio. As Figuras 2 e 3 ilustram, respectivamente, o relatório publicado e a análise semântica produzida.

2. **Commit 3fb8d6ba1 (SME-SIGPAE-API): achados semânticos.** Este commit ilustra um caso em que a camada LLM sinalizou indícios adicionais não capturados pelo detector estático. O commit utiliza CPF como *username* de login. O detector identificou uma violação Média (`dado_pessoal_em_url`). O LLM sinalizou três achados extras: (V001) CPF como *username* em *query string* (Arts. 6º, III e 46); (V002) CPF *hardcoded* `"11111111129"` em script de carga, passível de ir inadvertidamente a produção (Art. 46); e (V003) ausência de *audit log* (Art. 37). Os achados V002 e V003 fornecem evidências preliminares de que a camada LLM+RAG pode complementar ferramentas SAST baseadas em regex em situações dependentes de contexto [Johnson et al. 2013]. A saída completa deste caso foi disponibilizada no repositório de artefatos.

3. **Commits Susep: falsos positivos e resultado *incerto*.** Na execução inicial, o detector estático sinalizou dados pessoais em arquivos JavaScript minificados (*bundles* de terceiros), gerando falsos positivos estruturais. Ao restringir a análise a `src/`, nenhum achado adicional relevante foi sinalizado. Esse comportamento, também reportado em ferramentas SAST tradicionais [Johnson et al. 2013], reforça a importância de configurar adequadamente o escopo de análise, como no uso de `--src-path`. A camada LLM retornou sinalizações `POTENCIAL` com `confirmada: false` para o commit `48f027f` – resultado *incerto* da Seção 2 – ao identificar campos que *podem* contrariar o Art. 6º, III (Necessidade), mas cuja confirmação dependeria de inspeção adicional do contexto da *query*. A saída completa desse caso foi disponibilizada no repositório de artefatos.

Discussão. Em conjunto, os resultados fornecem evidências iniciais de viabilidade técnica do LGPD Guard em cenários controlados e reais. A combinação entre análise estática

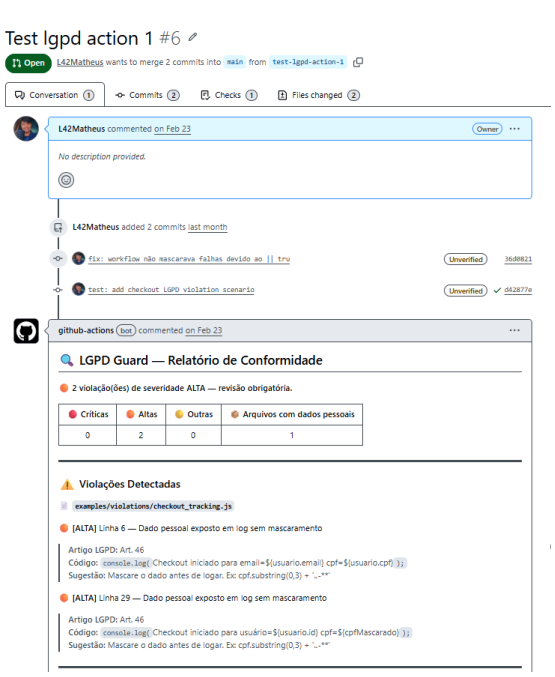

Figura 2. Relatório do LGPD Guard no PR#6: dois achados Alta em `checkout_tracking.js` (Art. 46).

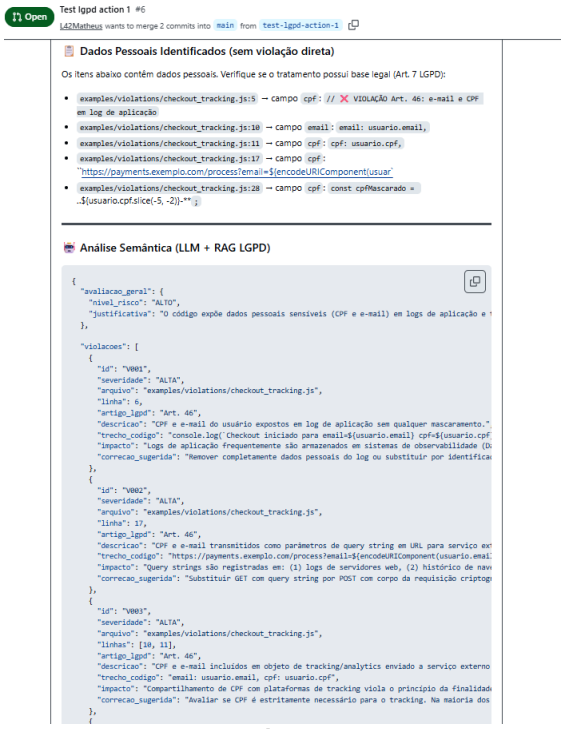

Figura 3. Análise LLM+RAG no PR#6: três violações (V001–V003), risco ALTO.

e análise semântica mostrou-se complementar: enquanto o detector estático captura padrões estruturais explícitos, a camada LLM+RAG pode sinalizar indícios dependentes de contexto. Por outro lado, a avaliação também evidencia limites importantes: arquivos minificados podem induzir falsos positivos, respostas de LLM podem exigir tratamento robusto de formato, e achados classificados como potenciais dependem de inspeção humana para confirmação. Assim, os resultados refletem a plausibilidade técnica da abordagem como apoio à auditoria contínua, mas não comprovam, isoladamente, efetividade jurídica. A seleção dirigida de commits, o número reduzido de repositórios e a ausência de validação independente por especialistas jurídicos limitam inferências sobre precisão, revocação, cobertura ou efetividade regulatória.

6. Implicações

Esta proposta apresenta implicações para a prática e para a pesquisa em Engenharia de Software. Na prática, a integração de LLMs com RAG em pipelines CI/CD pode tornar auditorias de conformidade menos reativas e mais contínuas, incorporando a análise de riscos regulatórios ao próprio fluxo de desenvolvimento. Isso favorece a identificação precoce de problemas, reforça o princípio de *privacy by design* e amplia o escopo tradicional de DevSecOps ao incluir conformidade jurídica como dimensão complementar à segurança técnica.

Do ponto de vista organizacional, a abordagem pode apoiar maior rastreabilidade e *accountability*, uma vez que desenvolvedores passam a receber *feedback* sobre possíveis impactos regulatórios de suas alterações ainda no Pull Request. Para a pesquisa, o trabalho abre caminhos relacionados à formalização computacional de requisitos legais, à confiabilidade de LLMs em tarefas normativas e à definição de métricas específicas para avaliação de conformidade automatizada.

7. Ameaças à Validade

A interpretação dos resultados deve considerar algumas ameaças à validade. Quanto à validade interna, as análises podem ser influenciadas por decisões de implementação, como engenharia de *prompt*, estratégia de *chunking*, cobertura da base normativa e política de bloqueio do pipeline. Além disso, a detecção baseada em regex e heurísticas pode gerar falsos positivos ou

falsos negativos, especialmente em código minificado, nomes ambíguos ou estruturas pouco convencionais. A análise semântica via LLM também pode ser afetada pela ausência de contexto completo do sistema, já que partes relevantes do comportamento podem estar distribuídas em outros arquivos, serviços ou repositórios não inspecionados naquela execução.

Quanto à validade externa, os resultados são limitados pelo foco na LGPD, pelo uso de GitHub Actions e pelo número reduzido de repositórios avaliados. A seleção dos commits foi dirigida por termos específicos, como `cpf`, com objetivo exploratório, o que reduz a representatividade da amostra em relação à diversidade de cenários industriais reais.

Quanto à validade de construto, a tradução de conceitos jurídicos abstratos em verificações técnicas pode não capturar integralmente sua interpretação normativa, sobretudo quando a conformidade depende da finalidade do tratamento, da base legal aplicável ou de políticas internas não explícitas no código-fonte. Por fim, quanto à validade de conclusão, os resultados são exploratórios e não permitem inferências estatísticas robustas sobre precisão, revocação ou cobertura, pois não houve *dataset* anotado por múltiplos especialistas jurídicos nem comparação sistemática com um *ground truth* independente.

8. Considerações Finais

Este artigo apresentou uma abordagem para auditoria automatizada de conformidade com a LGPD em pipelines CI/CD, baseada na integração de detecção estática, LLMs e RAG sobre dispositivos legais explícitos. O protótipo *LGPD Guard*, avaliado de forma exploratória em três repositórios de código aberto, forneceu evidências preliminares de que a camada semântica pode complementar a análise estática ao sinalizar potenciais violações dependentes de contexto e lidar com incertezas por meio da classificação *incerto*. A principal contribuição do trabalho está na articulação de três dimensões ainda pouco exploradas em conjunto: LGPD brasileira, integração contínua em pipelines CI/CD e suporte a arquiteturas baseadas em microsserviços. Entretanto, permanecem desafios relacionados à latência, à confiabilidade das respostas dos LLMs, à rastreabilidade de dados em sistemas distribuídos, à validação empírica e à responsabilidade sobre decisões automatizadas.

Como trabalhos futuros, pretende-se: (1) avaliar o LGPD Guard com um *dataset* anotado por especialistas, permitindo mensurar precisão, revocação e F1; (2) conduzir estudos de caso em ambientes corporativos reais; (3) integrar mecanismos de *Self-Verification* [Girardi et al. 2025] para reduzir falsos positivos; e (4) explorar arquiteturas *LLM-as-a-Judge* [Zheng et al. 2023, Chan et al. 2024] com modelos especializados por dimensão regulatória.

Disponibilidade de Artefatos

O código-fonte do **LGPD Guard**, incluindo módulos *Python*, *workflow do GitHub Actions*, base de conhecimento da LGPD, exemplos anotados de violações, saídas completas das execuções e versões em alta resolução das figuras, está disponível em: `https://github.com/L42Matheus/lgpd-guard`. O repositório contém instruções de instalação, configuração de *API secrets* e guia para reprodução dos experimentos.

Uso de Inteligência Artificial

Em conformidade com o Código de Conduta para Autores da SBC, declaramos que as ferramentas de IA foram utilizadas como apoio à revisão textual deste manuscrito. Especificamente, o *Grammarly* foi empregado para revisão gramatical, sugestões de melhoria de clareza e organização do texto. Todo o conteúdo científico é de responsabilidade exclusiva dos autores.

Referências

Abualhaija, S., Ceci, M., Sannier, N., Bianculli, D., Lannier, S., Siclari, M., Voordeckers, O., and Tosza, S. (2025). LLM-assisted extraction of regulatory requirements: A case study on the GDPR. In *Proceedings of the IEEE International Requirements Engineering Conference (RE)*. IEEE.

Amaral, O., Abualhaija, S., Torre, D., Sabetzadeh, M., and Briand, L. C. (2022). AI-enabled automation for completeness checking of privacy policies. *IEEE Transactions on Software Engineering*, 48(11):4647–4674.

Chambers and Partners (2025). Brazil: Data protection & privacy 2025. `https://practiceguides.chambers.com/practice-guides/data-protection-privacy-2025/brazil`. Acesso em: fev. 2026.

Chan, C.-M., Chen, W., Su, Y., Yu, J., Xue, W., Zhang, S., Fu, J., and Liu, Z. (2024). ChatEval: Towards better LLM-based evaluators through multi-agent debate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chen, D., Tian, D., Li, H., and Liu, L. (2025). A compliance checking framework based on retrieval-augmented generation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*. ACL Anthology.

Chen, T. (2024). Challenges and opportunities in integrating LLMs into continuous integration/continuous deployment (CI/CD) pipelines. In *Proceedings of the 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 364–367. IEEE.

Dakić, P. (2024). Software compliance in various industries using ci/cd, dynamic microservices, and containers. *Open Computer Science*, 14.

Donda, D. (2021). *Guia Prático de Implementação da LGPD*. Editora Labrador, São Paulo.

El Hamdani, R., Mustapha, M., Amariles, D. R., Troussel, A., Meeùs, S., and Krasnashchok, K. (2021). A combined rule-based and machine learning approach for automated GDPR compliance checking. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law (ICAIL)*, pages 40–49. ACM.

Girardi, C., Fernandes, D. Y. d. S., and Rêgo, A. S. d. C. (2025). Unveiling power on combining prompt engineering techniques: An experimental evaluation on code generation. In *Proceedings of the 40th Brazilian Symposium on Data Bases (SBBD)*, pages 357–370, Fortaleza, CE, Brasil. SBC.

Gloaguen, T., Mündler, N., Müller, M., Raychev, V., and Vechev, M. (2026). Evaluating AGENTS.md: Are repository-level context files helpful for coding agents? *arXiv preprint arXiv:2602.11988*.

Harvard Journal of Law & Technology (2025). Retrieval-augmented generation (RAG): Towards a promising LLM architecture for legal work? *Harvard Journal of Law & Technology Digest*. Acesso em: fev. 2026.

Johnson, B., Song, Y., Murphy-Hill, E., and Bowdidge, R. (2013). Why don't software developers use static analysis tools to find bugs? In *Proceedings of the 35th IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 672–681. IEEE.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474. Curran Associates.

Li, D. et al. (2024). Enhancing legal compliance and regulation analysis with large language models. *arXiv preprint arXiv:2404.17522*.

Rodriguez, D., Yang, I., Del Alamo, J. M., and Sadeh, N. (2024). Large language models: A new approach for privacy policy analysis at scale. *Computing*, 106(12):3879–3903.

Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., Li, Y., et al. (2025). The prompt report: A systematic survey of prompting techniques. *arXiv preprint arXiv:2406.06608*.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. (2023). Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36. Curran Associates.