Projeto demonstrativo

Diagnóstico técnico do YOURLS: relatório-modelo

É o formato que a Deltatech entrega num diagnóstico, aplicado a um projeto real de código aberto: o YOURLS, encurtador de links em PHP, licença MIT. Estrutura, dependências, segurança, privacidade, desempenho e testes, com evidência no código, riscos priorizados e um plano em fases com horas.

`PHP 8` `MySQL` `Semgrep 1.185` `OSV.dev` `revisão manual`

⚠ **Projeto demonstrativo sobre código aberto; não é auditoria contratada.** Nem a Deltatech nem este relatório têm relação com os mantenedores do YOURLS. O objetivo é mostrar o método. O YOURLS é um projeto maduro e bem cuidado, e os pontos abaixo são oportunidades de melhoria comuns em sistemas com mais de quinze anos de história.

Versão on-line: https://deltatech.pages.dev/diagnostico/yourls/ · Deltatech Consultoria em TI · CNPJ 69.784.623/0001-07

Projeto YOURLS/YOURLS	Versão analisada `b5b94b9` `(master)` 1.10.6 + 26 commits, de 10/11/2026
Data da análise 10/11/2026	Responsável Deltatech Consultoria em TI

1 Sumário executivo

O YOURLS está em bom estado para um projeto do seu tamanho e idade: as consultas principais são parametrizadas, o painel usa login e nonce, a saída é escapada e há uma suíte de testes grande rodando em seis versões do PHP. **Não encontramos risco alto.**

Os quatro riscos médios são de configuração e de política, não de código quebrado: login sem limite de tentativas, credenciais da API que podem ir na URL, histórico de cliques com IP guardado sem prazo e um PHP mínimo que já saiu de suporte. Todos têm correção pequena.

0	4	4	32 h
riscos altos	riscos médios	riscos baixos	plano completo, em 4 fases

2 Escopo e método

O que foi feito

- Leitura da estrutura, da autenticação, da API, do acesso ao banco e do registro de cliques.
- Análise estática com Semgrep 1.185 (regras p/php, p/javascript e p/secrets), sem a pasta vendor e os testes.
- Revisão manual de cada um dos 14 alertas automáticos.
- Dependências do composer.lock e bibliotecas JavaScript conferidas no OSV.dev, e versões atuais no Packagist e no npm.
- Leitura da configuração de testes e de integração contínua.

O que não foi feito

- Não instalamos nem executamos o sistema, e não rodamos a suíte de testes nesta amostra.
- Não houve teste de invasão nem teste de carga.
- Plugins de terceiros ficaram fora.
- A parte de LGPD é análise técnica, não parecer jurídico.

3 Visão geral do código

50
arquivos PHP do sistema
15,9 mil linhas, sem vendor e testes

101
arquivos de teste
10,3 mil linhas; 426 métodos de teste

14
dependências PHP
versionadas em includes/vendor

6
versões do PHP no CI
8.1 a 8.6, com MySQL

A arquitetura mistura funções em `includes/functions-*.php` com classes PSR-4 mais novas (`Config`, `Database`, `Views`) e um sistema de ganchos (filtros e ações) que sustenta os plugins. As dependências ficam versionadas no próprio repositório, o que facilita instalar em hospedagem compartilhada sem Composer, mas exige atualizar à mão.

Pontos fortes

- Consultas ao banco parametrizadas nas rotinas principais (PDO via Aura.SQL)
- Comparações de segredo com hash_equals e senhas novas com password_hash
- Saída do painel escapada (yourls_esc_html, yourls_esc_url, yourls_esc_attr)
- Manutenção ativa: o commit analisado, do próprio dia, corrige avisos do PHP 8.5 e 8.6
- Ações do painel protegidas por login e nonce
- Limite por IP para criação de links por visitantes
- 426 métodos de teste em 101 arquivos e CI com PHP 8.1 a 8.6 e MySQL

4 Riscos priorizados

R1 Médio Segurança — Fase 1 · 4 h

Login do painel sem limite de tentativas

Uma falha de login só dispara o gancho login_failed; não há atraso nem bloqueio no núcleo. O limite por IP existente vale para a criação de links, não para o login. Senhas fracas ficam expostas a tentativa e erro.

Recomendação: Limitar tentativas por IP e por usuário com espera progressiva (no núcleo ou num plugin oficial), registrar as falhas e documentar a proteção no servidor web como alternativa.

Evidência: `includes/functions-auth.php:116`, `includes/functions.php:756-779`

R2 Médio Segurança — Fase 1 · 3 h

Credenciais da API podem trafegar na URL

A API aceita usuário e senha, ou uma assinatura sem prazo, por parâmetros que podem ir na URL ($_REQUEST). URLs ficam em logs de servidor, de proxy e no histórico do navegador. A variante com prazo (timestamp) já existe, mas é opcional.

Recomendação: Opção de configuração para aceitar só assinatura com prazo e só por POST; deixar o padrão mais restrito numa versão maior e destacar isso na documentação da API.

Evidência: `includes/functions-auth.php:64-71`, `includes/functions-auth.php:138`, `includes/functions-auth.php:446-461`

R3 Médio Privacidade (LGPD, análise técnica) — Fase 2 · 8 h

IP completo e navegador de cada clique guardados sem prazo

Cada redirecionamento grava IP, navegador, origem e país. O núcleo não tem rotina de limpeza nem de anonimização, então o histórico cresce para sempre. IP é dado pessoal; guardar sem prazo dificulta atender ao princípio de necessidade.

Recomendação: Prazo de retenção configurável com limpeza agendada, opção de anonimizar o IP (zerar o último bloco) e um texto-modelo para a política de privacidade de quem instala. A base legal é decisão de quem opera a instalação.

Evidência: `includes/functions-install.php:241-251`, `includes/functions.php:589-596`

R4 Médio Manutenção Fase 3 · 4 h

Versão mínima do PHP já sem correções de segurança

O projeto aceita PHP 8.1, que deixou de receber correções de segurança em 31/12/2025. Instalações antigas podem continuar nessa versão sem aviso. O CI já testa até o PHP 8.6, então a subida é viável.

Recomendação: Elevar o mínimo para 8.2 ou 8.3 na próxima versão maior e mostrar um aviso no painel quando o servidor estiver numa versão sem suporte.

Evidência: `composer.json (php ^8.1)`

R5 Baixo Segurança (código legado) Fase 4 · 3 h

Rotina de atualização antiga monta SQL e HTML sem escape

A migração da versão 1.3 para 1.4 interpola valores vindos do banco numa consulta UPDATE e numa mensagem HTML. Só roda para quem atualiza de uma versão de 2009, com login de administrador, então o risco prático é pequeno.

Recomendação: Remover as rotinas de atualização anteriores à 1.5 (ou exigir atualizar em etapas) e, se mantidas, usar parâmetros e escape.

Evidência: `includes/functions-upgrade.php:405-428`

R6 Baixo Segurança Fase 1 · 1 h

Senhas no formato md5 antigo ainda são aceitas

Senhas antigas gravadas como md5 com sal continuam válidas. O painel avisa, mas o aviso pode ser adiado por uma semana indefinidamente.

Recomendação: Converter para password_hash no primeiro login bem-sucedido e, numa versão maior, recusar o formato antigo.

Evidência: `includes/functions-auth.php:163-166`

R7 Baixo Desempenho Fase 4 · 5 h

Estatísticas de um link releem todo o histórico a cada visita

A página de estatísticas agrupa por dia todos os cliques do link, sem limite de período nem cache, e a tabela só tem índice por link. Links muito acessados ficam lentos de consultar, e a tabela cresce junto com R3.

Recomendação: Índice composto (shorturl, click_time), período padrão de 12 meses na consulta e cache curto do resultado.

Evidência: `yourls-infos.php:112-121`, `includes/functions-install.php:249-250`

R8 Baixo Manutenção Fase 3 · 4 h

Bibliotecas de front-end copiadas e desatualizadas

jQuery 3.5.1 (2020), clipboard.js 2.0.6 e tablesorter 2.31.1 estão copiados no repositório. Não há vulnerabilidade conhecida para essas versões no OSV.dev, mas também não há processo que avise quando houver.

Recomendação: Atualizar para as versões atuais (jQuery 3.7 ou 4.0, clipboard.js 2.0.11, tablesorter 2.32) e incluir essas bibliotecas na verificação automática de dependências.

Evidência: `js/jquery-3.5.1.min.js e outras em js/`

5 Dependências

Nenhuma das 14 dependências PHP nem das 3 bibliotecas JavaScript tem vulnerabilidade conhecida no OSV.dev (consulta de 10/11/2026). Oito estão com atualização disponível, uma delas de versão maior.

Pacote	No projeto	Atual
aura/sql	6.0.0	6.1.0
composer/ca-bundle	1.5.12	1.6.1
geoip2/geoip2	2.13.0	3.4.0 (versão maior)
maxmind-db/reader	1.13.1	1.14.0
maxmind/web-service-common	0.11.1	em dia
ozh/bookmarkletgen	1.3	em dia
ozh/http_build_url	1.0.3	em dia
pomo/pomo	1.5.0	em dia
psr/log	3.0.2	em dia
rmccue/requests	2.0.18	2.0.21
spatie/array-to-xml	3.4.4	em dia
symfony/polyfill-intl-idn	1.38.1	1.43.0
symfony/polyfill-intl-normalizer	1.38.0	1.43.0
symfony/polyfill-mbstring	1.38.1	1.43.3

6 Alertas automáticos

O Semgrep levantou 14 alertas (9 de saída de dados da requisição sem escape e 5 de SQL montado com entrada externa). Revisamos um a um:

11 falsos positivos: respostas JSON do painel protegidas por login e nonce, ordenação validada por lista fechada, valores convertidos para inteiro, consultas com parâmetros e saída já escapada.

3 confirmados, todos na mesma rotina legada de atualização (risco R5).

Por isso um relatório nunca é a saída crua de uma ferramenta: sem a revisão, o cliente receberia 14 “vulnerabilidades” onde há uma.

7 Plano em fases

Fase 1

Proteções rápidas

Limite de login, credenciais da API fora da URL e fim das senhas md5.

8 h

R1, R2, R6

Fase 2

Privacidade

Retenção configurável, anonimização de IP e texto-modelo para a política.

8 h

R3

Fase 3

Atualização

PHP mínimo, dependências PHP (incluindo geoip2 3.x) e bibliotecas de front-end.

8 h

R4, R8

Fase 4

Limpeza e desempenho

Remoção de rotinas legadas e consulta de estatísticas com índice e período.

8 h

R5, R7

Horas estimadas para um contrato hipotético, incluindo testes e revisão de cada mudança. As fases são independentes e podem ser contratadas uma a uma; a fase 1 resolve dois dos quatro riscos médios em um dia de trabalho.