Revisão de *Pull Requests* Assistida por Inteligência Artificial: um Estudo de Caso no CodeLab-IFPR

João Lucas Pinheiro de Oliveira, Ana Beatriz Tavares Malaquias, Gustavo Randi da Cruz, Eliandro Gomes da Silva, Marcelo Figueiredo Terenciani, Frank Willian Cardoso de Oliveira

Instituto Federal do Paraná (IFPR) – Campus Paranavaí
Paranavaí – PR – Brasil

{20241pvai10030039,20241pvai10030002}@estudantes.ifpr.edu.br
{20241pvai10030010,20241pvai10030027}@estudantes.ifpr.edu.br
{frank.willian,marcelo.terenciani}@ifpr.edu.br

O CodeLab-IFPR é um laboratório de extensão do curso de Engenharia de Software do IFPR, Campus Paranavaí, em que estudantes desenvolvem sistemas reais para parceiros da comunidade. Com várias equipes trabalhando em paralelo, a revisão de *Pull Requests* (PRs) tornou-se um gargalo: conferir cada PR contra a História de Usuário (HU) correspondente é demorado e sujeito a esquecimento. Este resumo relata a metodologia de revisão de PRs assistida por Inteligência Artificial (IA) adotada no Coopervai, sistema desenvolvido pelo CodeLab-IFPR para apoiar uma cooperativa de reciclagem de vidro da região de Paranavaí (PR) na gestão de bombonas deixadas em pontos de coleta[1]. O projeto é um estudo de caso para posterior ampliação da metodologia aos demais projetos do laboratório, e a regra central é que a IA faz o trabalho pesado de conferência enquanto a decisão final continua humana.

O Coopervai usa Laravel 12 (PHP 8.2), telas em Blade com Alpine.js e Tailwind, MySQL em produção e SQLite nos testes. O backlog fica no GitHub, organizado em épicos e HUs, cada uma com critérios de aceite de *front-end* e *back-end*, e o andamento é acompanhado em um quadro do GitHub Projects. Uma convenção de nomes liga a branch, a issue da HU e o PR pelo número da história, o que permite recuperar automaticamente o que foi pedido a partir do que foi entregue. O desenho da automação seguiu cinco princípios: o critério de aceite decide, e débito técnico vira ressalva, não veto; a IA prepara e o humano decide; evidência (arquivo e linha) em vez de opinião; separação entre o que é genérico (processo e especialistas) e o que é específico do projeto; e menor privilégio.

A automação usa o Claude Code [Anthropic 2026] e três de seus mecanismos: *skills*, pastas de instruções e arquivos de referência carregados sob demanda; subagentes, definições que rodam em contexto isolado e em paralelo, com ferramentas e modelo próprios; e *hooks*, scripts acionados pela ferramenta em eventos como o envio de uma mensagem ou a execução de um comando, capazes de negá-lo. A Tabela 1 lista os componentes construídos.

Os especialistas têm acesso somente de leitura ao código e usam um modelo mais barato que o do orquestrador. Devolvem achados com severidade, arquivo e linha, impacto, sugestão e confiança, mas não emitem veredito. O roteamento parte da lista de arquivos alterados pelo PR: código PHP aciona os agentes de PHP e de Laravel; telas

[1]Repositório do projeto: `https://github.com/CodeLab-IFPR/coop-recicla`.

Tabela 1. Componentes da automação (caminhos relativos a `~/.claude/`, exceto o da *skill* versionada no repositório)

Arquivo	Função
`skills/aprovar-pr/`	Orquestrador: localiza a HU, monta o *checklist*, aciona os especialistas, emite o veredito e executa o fechamento
`skills/validar-tela-pr/`	Validação visual das telas alteradas pelo PR, com as heurísticas de Nielsen [Nielsen 1994]
`.claude/skills/coopervai/` (versionada no repositório)	Contexto do projeto: glossário, papéis, *stack*, convenções e inconsistências conhecidas do backlog
`agents/especialista-*.md`	Seis agentes especialistas: PHP, Laravel, Blade/Alpine, React, MySQL e segurança/LGPD
`hooks/trava-aprovacao.mjs`	Exige a autorização explícita do revisor para ações públicas no GitHub
`hooks/protege-segredos.mjs`	Bloqueia a leitura do arquivo de segredos do projeto pelo terminal
`scripts/custo-revisao.mjs`	Apura *tokens* e custo por revisão, com registro em CSV
`settings.json`	Registra os *hooks* e as regras de permissão

e scripts de interface acionam o de Blade/Alpine; migrations e SQL cru acionam o de MySQL, que procura justamente o que passa no SQLite e quebra no MySQL. Rotas, *controllers*, autenticação, telas públicas e campos de dado pessoal acionam o agente de segurança, que recebe o PR inteiro e percorre o caminho do dado pessoal segundo a LGPD [Brasil 2018], exigindo um cenário de ataque concreto para cada achado. PRs pequenos não acionam especialistas.

Duas travas rodam fora do modelo. A primeira registra a autorização do revisor somente quando a mensagem começa com "aprovado" ou "reprovado" seguido do número do PR, liberação que vale apenas para aquele PR e é apagada em qualquer outra mensagem, e nega, fora desse turno, os comandos que tornam pública uma decisão no GitHub: aprovar, mergear, comentar, fechar a issue e mover o card; "reprovado" libera apenas o pedido de mudanças. A segunda impede a leitura do arquivo de segredos do projeto, tanto pela ferramenta de leitura quanto por comandos equivalentes no terminal. Toda decisão é gravada em log. O conteúdo do PR é tratado como dado, nunca como instrução, pois já houve agentes de revisão que vazaram segredos após ler instruções maliciosas em PRs [Daelman 2025].

No fluxo de revisão, o revisor pede "avalia o PR 23". O orquestrador carrega a *skill* de contexto do projeto, localiza a HU pela branch e monta o *checklist* de critérios de aceite. Os especialistas analisam o código em paralelo e cada critério é marcado como atendido, parcial, não atendido ou não verificado. Quando é preciso executar o sistema, isso é feito em uma cópia de trabalho isolada, com banco descartável, nunca no banco de desenvolvimento. O relatório traz o veredito em três níveis (reprovado, aprovado com ressalvas ou aprovado) e a evidência de cada critério. Com "aprovado 23", a automação aprova o PR, mergeia levando as ressalvas na mensagem de merge, move o card para *Done*, fecha a issue e confere no GitHub se cada ação de fato aconteceu; com "reprovado 23", publica o que o desenvolvedor precisa corrigir. Quando o PR volta, a re-revisão olha apenas o que mudou.

Os resultados são parciais e convém distinguir o que já foi exercitado do que ainda não foi. O fluxo básico de revisão, que consiste em localizar a HU, conferir os critérios,

aprovar, mergear, mover o card e fechar a issue, foi usado em PRs reais do Coopervai em agosto de 2026. As travas, por sua vez, foram verificadas em cenários simulados, construídos a partir das formas previsíveis de burlá-las: mensagem sem autorização, autorização dada para outro PR, autorização vinda de um turno anterior, tentativa partindo de um subagente e variações de escrita do mesmo comando. Foram 20 cenários para a trava de aprovação e 18 para a proteção de segredos; em todos a ação foi negada e registrada, como esperado. O script de custo, que lê os registros de sessão do Claude Code, mostrou que a sessão de construção do conjunto custaria cerca de US$ 18 pela API, 53% disso em releitura do contexto acumulado; por isso adotou-se uma revisão por sessão.

Ainda não foram avaliados em PRs reais os agentes especialistas, a validação visual de telas e a re-revisão, e também não há, até aqui, comparação sistemática entre as revisões assistidas por IA e as realizadas exclusivamente por humanos, comparação que constitui a próxima etapa do estudo. A validação visual será empregada como pré-análise, e não como substituta da inspeção humana, pois um estudo comparativo verificou que o GPT-4o identificou apenas 21,2% dos problemas de usabilidade apontados por especialistas [Guerino et al. 2025].

O estudo de caso indica que é possível usar IA na revisão de código sem transferir a ela a decisão, e que separar o processo genérico do contexto do projeto facilita a reutilização: para outro projeto, basta escrever a *skill* de contexto e escolher os especialistas da *stack*. Os próximos passos são medir o custo e a concordância entre o veredito da IA e a decisão humana e levar a metodologia aos demais projetos do CodeLab.

Referências

Anthropic (2026). Claude Code: documentação oficial. Disponível em: `https://docs.claude.com/en/docs/claude-code/overview`. Acesso em: 20 set. 2026.

Brasil (2018). Lei n. 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). Disponível em: `https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm`. Acesso em: 20 set. 2026.

Daelman, R. (2025). PromptPwnd: prompt injection vulnerabilities in GitHub Actions using AI agents. Aikido Security. Disponível em: `https://www.aikido.dev/blog/promptpwnd-github-actions-ai-agents`. Acesso em: 20 set. 2026.

Guerino, G., Rodrigues, L., Capeleti, B., Mello, R. F., Freire, A., e Zaina, L. (2025). Can GPT-4o evaluate usability like human experts? A comparative study on issue identification in heuristic evaluation. In *Human-Computer Interaction – INTERACT 2025*, Cham. Springer. Disponível em: `https://arxiv.org/abs/2506.16345`. Acesso em: 20 set. 2026.

Nielsen, J. (1994). Heuristic evaluation. In Nielsen, J. e Mack, R. L. (Eds.), *Usability Inspection Methods*, p. 25–62. John Wiley & Sons, New York.