
**UNIVERSIDADE FEDERAL DE SERGIPE**

DEPARTAMENTO DE COMPUTAÇÃO

PROF. GLAUCO DE FIGUEIREDO CARNEIRO

## ENGENHARIA DE SOFTWARE II

# ATIVIDADE III - Relatório de Auditoria de DevOps

**DISCENTES**

**ARTHUR COSTA OLIVEIRA**

**DAVI LIRA SANTANA**

**GABRIEL BATISTA BARBOSA**

**JOÃO HENRIQUE BRITTO BOMFIM**

**LUAN ALMEIDA VALENÇA**

**MATHEUS NASCIMENTO DOS SANTOS**

**PAULO HENRIQUE MELO RUGANI DE SOUSA**

**TASSIO MATEUS DE CARVALHO**

**SÃO CRISTÓVÃO – SE**

**28/01/2025**

# 1. Integrantes e organização

O grupo é composto pelos seguintes integrantes: Arthur Costa Oliveira (202300027104), Davi Lira Santana (202300083319), Gabriel Batista Barbosa (202300027249), João Henrique Britto Bomfim (202300027409), Luan Almeida Valença (202300027866), Matheus Nascimento dos Santos (202300083810), Paulo Henrique Melo Rugani de Sousa (202300027919) e Tassio Mateus de Carvalho (202300083963).

**Links:**
Repositório
Vídeo
Apresentação

Tássio mapeou o cenário e o fluxo atual do projeto, Luan diagnosticou os riscos críticos de manutenibilidade. A coleta técnica de evidências sobre a falta de automação foi realizada por Paulo e João. Para a proposta de melhoria, Davi desenhou o fluxo ideal (TO-BE) , Arthur e Matheus detalharam os impactos e a profissionalização do pipeline , e Gabriel consolidou o veredito final da auditoria.

# 2. Projeto escolhido (JARVIS)

O JARVIS (também conhecido como HuggingGPT) é um sistema colaborativo desenvolvido pela Microsoft Research que utiliza Modelos de Linguagem de Grande Escala (LLMs), como o GPT-4 ou ChatGPT, como um "controlador central" para conectar diversos modelos de Inteligência Artificial especializados disponíveis na comunidade Hugging Face.

O problema que ele resolve: Modelos de linguagem (como o ChatGPT) são ótimos em texto, mas não conseguem gerar imagens, editar vídeos ou falar. O JARVIS resolve isso agindo como um "maestro": ele entende o pedido do usuário (ex: "Gere uma imagem de um gato e descreva-a em áudio"), planeja quais tarefas são necessárias, escolhe o modelo de IA correto para cada tarefa no Hugging Face, executa o trabalho e entrega o resultado final integrado.

# 3. Diagnóstico do Processo Atual (AS-IS)

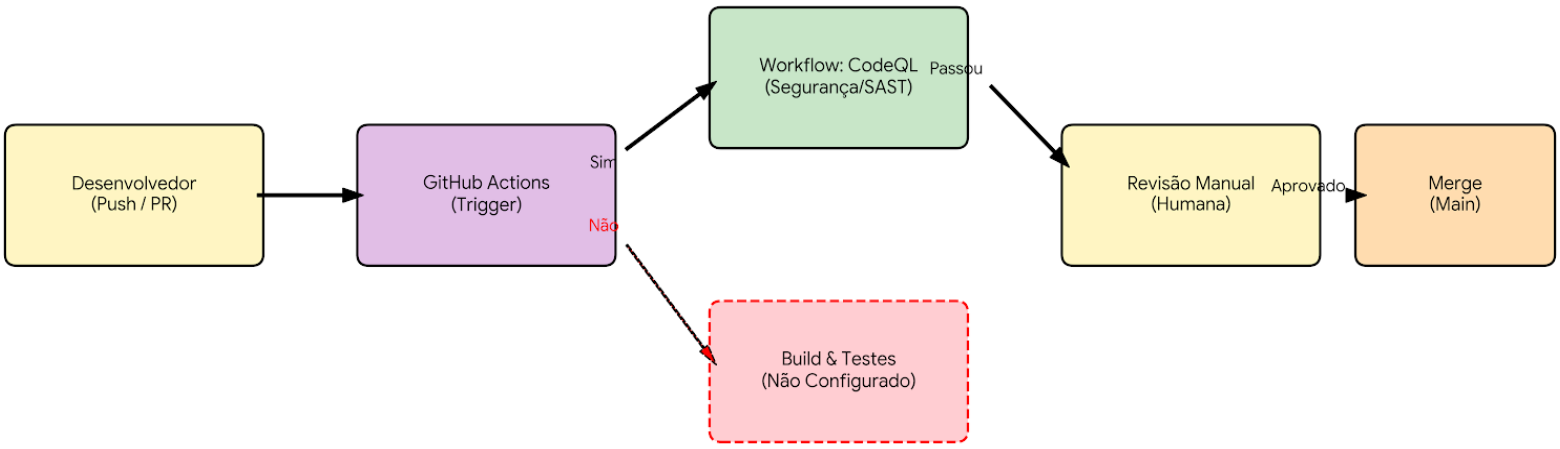



*Figura 1: Fluxo atual de contribuição no repositório Microsoft/JARVIS, evidenciando a execução exclusiva de verificação de segurança (CodeQL) e a ausência completa de etapas de Build e Testes funcionais automatizados.*

## 3.1. Análise dos Mecanismos de Automação (CI/CD)

A auditoria técnica do repositório revelou que o projeto utiliza a infraestrutura do GitHub Actions. No entanto, a implementação atual é voltada exclusivamente para DevSecOps (Segurança), negligenciando a Integração Contínua funcional.

Workflows Identificados: A inspeção da aba "Actions" confirmou a existência de um único workflow ativo denominado CodeQL. Esta ferramenta realiza Análise Estática de Segurança (SAST) para identificar vulnerabilidades conhecidas no código.

O "Gap" de Automação: Não foram encontrados workflows essenciais para a garantia de qualidade de software, tais como:

- Instalação automática de dependências (Build);
- Execução de suítes de testes unitários ou de integração;
- Verificação de estilo de código (Linting/Formatação).

## 3.2. Análise de Frequência e Manutenibilidade (Software Decay)

Identificamos um cenário de risco elevado relacionado à degradação do software ("Software Decay"). O histórico de *commits* mostra que a última contribuição significativa ocorreu há aproximadamente 6 meses.

Embora existam execuções recentes na aba "Actions" (ex: Jan 18), observou-se que estas foram disparadas pelo gatilho Scheduled (agendamento automático) e executadas por bots de segurança (github-advanced-security bot). Isso indica um monitoramento passivo de segurança por parte da infraestrutura da Microsoft, mas confirma a ausência de desenvolvimento ativo ou validação de compatibilidade com novas versões de bibliotecas externas nas últimas semanas.

## 3.3. Análise da Documentação de Contribuição

A revisão do arquivo CONTRIBUTING.md reforçou a hipótese de inexistência de uma cultura de testes automatizados. O documento limita-se a instruir o fluxo básico de versionamento (comandos git clone, fork, push), omitindo completamente instruções sobre:

- Como configurar o ambiente de desenvolvimento local;
- Como rodar testes antes de submeter um Pull Request;
- Critérios de aceitação de código.

Essa lacuna transfere a responsabilidade da verificação de qualidade inteiramente para a revisão manual dos mantenedores, criando um gargalo operacional e aumentando o risco de regressão.

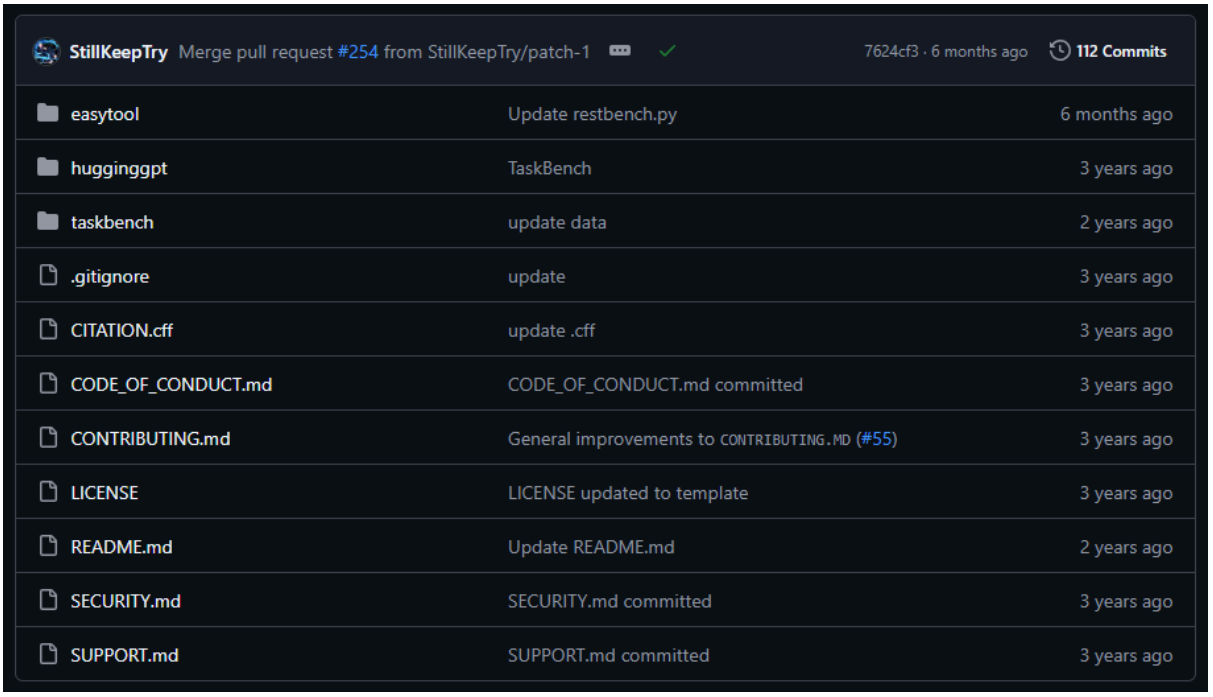
## 3.4. Mapeamento de Riscos (Conclusão do Diagnóstico)

A combinação da falta de testes funcionais, dependência de bibliotecas de IA (que atualizam frequentemente) e baixa atividade de manutenção cria os seguintes riscos críticos:

- Risco de "Dependency Hell": Alta probabilidade de o projeto não compilar atualmente devido a conflitos de versões não detectados.
- Regressão Funcional: Novos contribuidores não possuem feedback imediato (Feedback Loop) sobre seus códigos, podendo introduzir bugs que só serão descobertos em produção.
- Dependência Humana: O processo de aprovação de PRs é lento e subjetivo, dependendo de testes manuais que nem sempre são executados.

# 4. Evidências Coletadas (A, B e C)

## 4.1. Evidência A – Ausência de Workflows de Integração Contínua




Durante a inspeção do repositório oficial do projeto **Microsoft JARVIS**, foi realizada uma análise da estrutura de diretórios e das configurações de automação disponíveis no GitHub. Especificamente, buscou-se a presença da pasta padrão `.github/workflows`, local onde normalmente são definidos os pipelines de Integração Contínua (CI) e Entrega Contínua (CD) por meio do GitHub Actions.

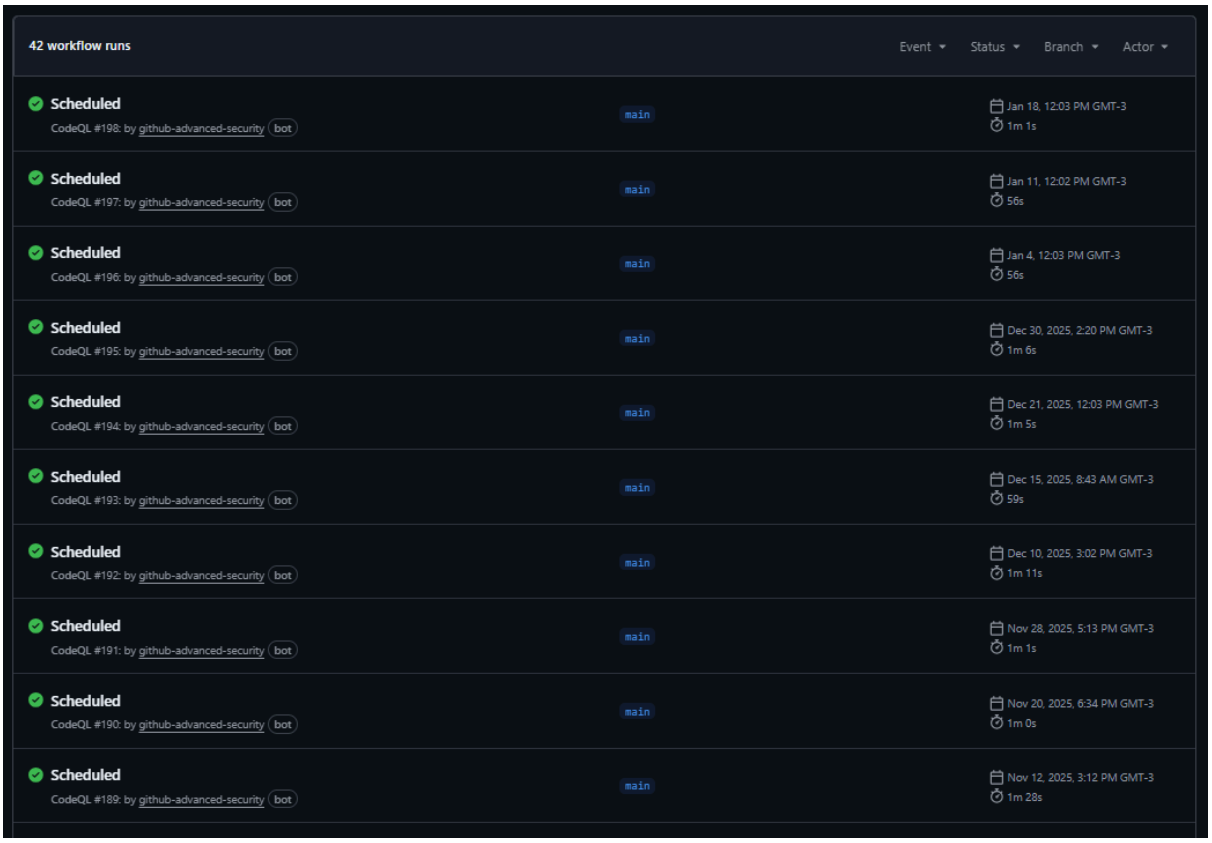
No entanto, como já foi citado na seção 3.1 não foram encontrados workflows de CI/CD voltados à construção do projeto, execução de testes automatizados ou validações de estilo de código. A análise da aba **"Actions"** do repositório confirmou que o único processo automatizado existente corresponde à ferramenta **CodeQL**, utilizada exclusivamente para análise estática de segurança do código-fonte.

A ausência de workflows de integração contínua pode ser justificada pelo contexto do projeto, que se caracteriza como um protótipo de pesquisa desenvolvido pela Microsoft Research, com foco principal na experimentação e validação conceitual da arquitetura proposta (HuggingGPT), e não na manutenção contínua ou evolução incremental do software. Nesse tipo de projeto, é comum que o esforço de automação seja direcionado prioritariamente à detecção de vulnerabilidades de segurança, em detrimento de pipelines completos de build e testes.

Entretanto, do ponto de vista de Engenharia de Software, essa ausência representa uma limitação relevante no processo de desenvolvimento, pois impede a validação automática de alterações no código, aumenta a dependência de revisões manuais e eleva o risco de regressões funcionais. Dessa forma, a Evidência A sustenta o diagnóstico de que o

projeto possui automação parcial, restrita a aspectos de segurança, carecendo de práticas consolidadas de Integração Contínua.

## 4.2. Evidência B – Histórico de Execuções do GitHub Actions



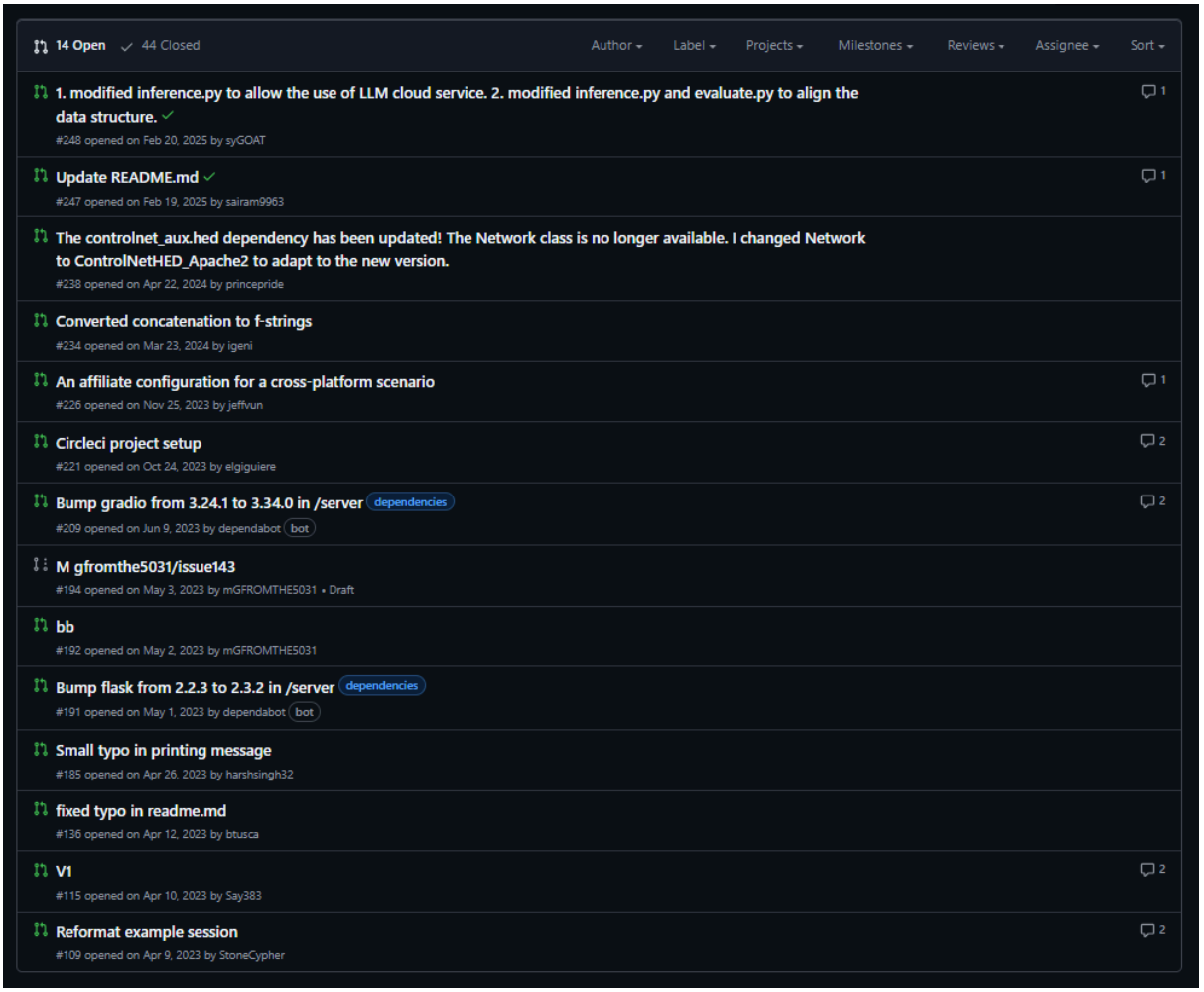

De acordo com a figura acima retirada do GitHub Actions no repositório Microsoft/JARVIS, fica evidenciado a ocorrência de 42 execuções associadas exclusivamente ao workflow de análise estática de segurança (CodeQL). As execuções registradas correspondem a verificações automáticas de vulnerabilidades no código-fonte, normalmente acionadas por eventos agendados (*scheduled*) ou por mecanismos internos de monitoramento de segurança da plataforma GitHub.

A análise detalhada dessas execuções indica que os workflows ativos realizam apenas etapas de varredura de código, sem contemplar fases típicas de Integração Contínua, tais como instalação de dependências, construção do projeto (build), execução de testes unitários ou de integração, e validações de estilo ou qualidade do código. Não há registros de jobs que envolvam a execução do código Python do projeto ou a verificação de sua funcionalidade em ambiente controlado.

Esse histórico reforça que a automação existente no projeto está restrita a práticas de **DevSecOps**, voltadas exclusivamente à detecção de falhas de segurança, não abrangendo mecanismos de garantia de qualidade funcional. Dessa forma, mesmo com um número

significativo de execuções registradas, o processo automatizado não fornece feedback contínuo sobre a estabilidade ou corretude do software ao longo do tempo.

## 4.3. Evidência C – Comportamento das Pull Requests no Repositório




De acordo com a imagem acima da listagem de Pull Requests no repositório *microsoft/JARVIS* mostrando PRs abertos e fechados, sem indicadores de status checks automatizados de build/test. A partir da visualização da interface de Pull Requests, observa-se que os itens listados apresentam seus títulos e estado (por exemplo, “aberto” ou “fechado”), porém **não há exibição de verificações obrigatórias associadas a pipelines de CI/CD**, como normalmente ocorre quando existem gates automáticos configurados.

Essa ausência de indicadores de CI em Pull Requests reforça que **não existem workflows de integração contínua configurados para rodar automaticamente quando uma contribuição externa é submetida ao repositório**. Apesar de haver um workflow configurado para análise de segurança (CodeQL), ele **não aparece como um status check obrigatório diretamente associado ao processo de revisão dos Pull Requests**. Isso

significa que os colaboradores que enviam alterações não recebem automaticamente feedback sobre a construção do código, execução de testes funcionais ou outras medidas de qualidade de software antes da revisão manual.

- **Observações que fortalecem essa evidência:**
    - A página de Pull Requests lista as contribuições (abertas/fechadas) sem exibir checks de CI ativos.
    - Não existem status automáticos adicionais obrigatórios indicados no nível de PR.
    - Mesmo quando um PR é aberto, a interface não apresenta jobs de build ou teste como parte dos *checks* obrigatórios.

Do ponto de vista de processos de engenharia de software, tal cenário evidencia uma dependência **quase que exclusiva da revisão manual de código para aprovação de PRs**, sem assistências automáticas que ajudem a evitar regressões funcionais ou falhas introduzidas por alterações. Essa limitação pode impactar a velocidade de desenvolvimento, a confiança nas contribuições e a qualidade do código incorporado ao projeto.

# 5. Conclusões Gerais da Análise

Este relatório apresentou uma auditoria do processo de desenvolvimento e automação DevOps do projeto Microsoft JARVIS (HuggingGPT), com foco na identificação de práticas de Integração Contínua, Entrega Contínua e garantia de qualidade de software. A análise evidenciou que, embora o projeto utilize a infraestrutura do GitHub Actions, sua automação encontra-se restrita a verificações de segurança por meio da ferramenta CodeQL, caracterizando uma abordagem de DevSecOps parcial.

O diagnóstico do processo atual (AS-IS) demonstrou a ausência de pipelines essenciais de Integração Contínua, tais como build automatizado, execução de testes unitários ou de integração e validações de estilo de código. Essa limitação é reforçada pelas evidências coletadas, que mostram um histórico de execuções exclusivamente voltado à análise estática de segurança, bem como a inexistência de verificações automáticas associadas ao fluxo de Pull Requests.

Do ponto de vista de Engenharia de Software, esse cenário implica riscos relevantes, como maior probabilidade de regressões funcionais, dependência excessiva de revisões manuais e dificuldade de contribuição por novos desenvolvedores. Apesar de que tais limitações podem ser parcialmente justificadas pelo caráter experimental e acadêmico do projeto, elas representam fragilidades quando analisadas sob a ótica de manutenção, evolução e escalabilidade do software.

Conclui-se, portanto, que o projeto Microsoft JARVIS apresenta um nível básico de automação, adequado à detecção de vulnerabilidades de segurança, mas insuficiente para garantir a qualidade funcional do software de forma contínua. A adoção de pipelines de Integração Contínua completos, incluindo testes automatizados e validações de build, seria

uma melhoria significativa para reduzir riscos e alinhar o projeto às boas práticas modernas de DevOps.