
Gabriel Dolabela e Henrique Almeida

gabriel.marques.1203633@sga.pucminas.br
me@h3nc4.com

Documento de Visão para o Sistema HookCI

16 de Março de 2025

Proposta do(s) aluno(s) Gabriel Dolabela Marques e Henrique Carvalho Almeida ao curso de Engenharia de Software como projeto de Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo do professor PPP e orientação acadêmica do professor YYY.

OBJETIVOS

Criar uma ferramenta que integre a prática de Integração Contínua (CI, do inglês *Continuous Integration*) ao ambiente local de desenvolvimento. A solução permitirá que os testes de um sistema sejam executados antes dos *commits* ou *pushes*, evitando que falhas e inconsistências sejam propagadas para o repositório remoto. Desta forma, o sistema visa aumentar a confiabilidade do código, reduzir o retrabalho e otimizar o uso dos recursos, contribuindo para um ciclo de desenvolvimento mais ágil e eficaz.

ESCOPO

HookCI consiste em uma ferramenta para a execução da CI de forma local, integrando Docker e Git Hooks para validar *commits* e *pushes*. Suas principais características são:

- Integração com Git: A ferramenta será instalada como parte do fluxo de trabalho do Git, utilizando hooks como *pre-commit* e *pre-push* para disparar a execução da CI.
- Execução da CI localmente: Em vez de esperar que os testes do projeto sejam realizados em servidores remotos, o sistema permite que as validações ocorram na máquina do desenvolvedor, promovendo feedback imediato.
- Isolamento via Docker: Para garantir a consistência do ambiente de CI e evitar a interferência do ambiente local, a execução ocorrerá em containers Docker configurados conforme as necessidades do projeto.

-
- Configuração flexível: Através de um arquivo do tipo YAML, os usuários poderão definir variáveis de ambiente, comandos de teste e parâmetros de execução, adaptando a ferramenta a diferentes contextos de desenvolvimento.
 - Interface de linha de comando (CLI, do inglês *Command Line Interface*): A interação com o sistema será realizada via CLI, possibilitando uma integração simples e eficiente ao fluxo de trabalho dos desenvolvedores.

A proposta se diferencia dos serviços de CI tradicionais, como GitHub Actions ou GitLab CI, pois traz o conceito de CI para o ambiente local, reduzindo o tempo de feedback e evitando a sobrecarga em servidores remotos. Assim, a ferramenta atua como uma camada preventiva que assegura que somente códigos que passam nos testes sejam integrados ao repositório central.

Outra ferramenta semelhante é o Husky, que facilita a execução de scripts via Git Hooks, mas roda os comandos diretamente no ambiente local. Nossa ferramenta pretende isolar os testes em Docker para garantir a consistência, oferecendo configuração via YAML e uma CLI dedicada, tornando o processo de CI local mais estruturado.

FORA DO ESCOPO

Para manter o foco na solução do problema principal e evitar complexidades desnecessárias, foram definidos os seguintes itens como fora do escopo:

- Execução remota de testes: O sistema não prevê a criação ou o gerenciamento de servidores remotos para CI; a execução dos testes se restringe ao ambiente local.
- Interface gráfica: O foco será em uma interface YAML e CLI, não havendo, nesta versão, uma interface gráfica interativa para configuração ou monitoramento.
- Integração com múltiplos sistemas de versionamento: O sistema foi projetado para funcionar com Git, não sendo contemplada a integração com outros sistemas de versionamento.
- Geração de relatórios entre operações: Embora feedbacks de erro e estado sejam apresentados, a geração de relatórios e dashboards interativos está fora do escopo inicial.

GESTORES, USUÁRIOS E OUTROS INTERESSADOS

Nome	Gabriel Dolabela Marques
Qualificação	Estudante
Responsabilidades	Desenvolvedor

Nome	Henrique Carvalho Almeida
Qualificação	Estudante
Responsabilidades	Desenvolvedor

LEVANTAMENTO DE NECESSIDADES

Enumere as necessidades identificadas no ambiente de negócio e que justifiquem a criação do sistema. As necessidades devem ser numeradas e justificadas.

1. Validação Preventiva de Código. Evitar que commits e pushes com falhas ou erros de testes sejam enviados para o repositório remoto, garantindo a integridade do código e diminuindo retrabalhos.
2. Feedback Imediato aos Desenvolvedores. Proporcionar uma resposta rápida quanto à qualidade do código, possibilitando correções imediatas e maior agilidade no processo de desenvolvimento.
3. Isolamento do Ambiente de CI. Utilizar Docker para garantir que a CI seja executada em um ambiente controlado e padronizado, independente das configurações locais dos desenvolvedores.
4. Flexibilidade na Configuração e Execução. Permitir que diferentes projetos e equipes customizem os parâmetros de execução dos testes via arquivo YAML ou CLI, adaptando a ferramenta às necessidades específicas de cada contexto.
5. Especificar imagens docker para runtime de testes. Permitir que a ferramenta utilize uma imagem Docker pré-definida ou um Dockerfile para criar o ambiente de testes, garantindo suporte a diferentes runtimes.
6. Pré configuração na inicialização da ferramenta no repo. Inicializar os arquivos e configurações Git necessárias para a execução da ferramenta de forma automática.
7. Depuração de testes. Criar ferramentas de verbosidade e debug em casos de erro.

FUNCIONALIDADES DO PRODUTO

Necessidade: Validação Preventiva de Código	
Funcionalidade	Categoria
1. Integração com Git Hooks (<i>pre-commit</i> e <i>pre-push</i>) para	Crítico

disparo automático da CI.	
2. Bloqueio de commits ou pushes em caso de falha nos testes.	Crítico
3. Mecanismo de bloqueio condicional para testes não essenciais.	Útil

Necessidade: Flexibilidade na Configuração e Execução	
Funcionalidade	Categoria
1. Utilização de arquivo YAML para definição de testes e ambientes.	Crítico
2. Permitir evocação da CLI pelo usuário para execução de testes.	Importante
3. Interface de configuração para permitir ou não que commits e pushes ocorram em caso de erro.	Útil

Necessidade: Feedback Imediato aos Desenvolvedores	
Funcionalidade	Categoria
1. Log dos resultados da CI apresentados na interface CLI em caso de Falha	Crítico
2. Adicionar cores e símbolos para destacar erros e avisos.	Útil

Necessidade: Isolamento do Ambiente de CI	
Funcionalidade	Categoria
1. Execução da CI em containers Docker, garantindo ambiente padronizado.	Crítico
2. Montagem do diretório atual do repositório no container para execução da CI.	Crítico

Necessidade: Especificar imagens docker para runtime de testes	
Funcionalidade	Categoria

1. Usuário especifica uma imagem docker através de nome e versão	Crítico
2. Usuário especifica uma imagem docker através de Dockerfile	Importante

Necessidade: Pré configuração na inicialização da ferramenta no repo	
Funcionalidade	Categoria
1. Chamada por comando para se inicializar no repositório.	Crítico
2. Instalação automatizada de Git Hooks durante a configuração inicial do projeto.	Crítico
3. Gerar automaticamente um arquivo de configuração YAML padrão.	Crítico
4. Verificar dependências essenciais e alertar sobre falhas e faltas no ambiente.	Importante
5. Criar opções CLI para customizar configurações iniciais.	Importante
6. Migração das configurações caso uma nova versão (compatível com a anterior) da ferramenta seja detectada.	Útil

Necessidade: Depuração de testes	
Funcionalidade	Categoria
1. Permitir parâmetros de verbosidade na CLI.	Útil
2. Console de depuração aberto para o usuário em caso de erro ou fim de execução quando a depuração está ativa.	Útil

INTERLIGAÇÃO COM OUTROS SISTEMAS

O sistema será interligado com Git por meio de sua ferramenta de Git Hooks e com Docker para a execução dos testes.

RESTRICOES

- Restrições do Produto
 - O sistema deverá ser implementado utilizando Python e Shell.
 - O sistema deverá ser entregue via um único binário executável.

-
- O retorno dos testes deverá ser otimizado, com um limite de 2 segundos para feedback após sua execução.
 - Requisitos de Portabilidade
 - O sistema deverá ser executado em uma distribuição GNU/Linux com Docker e Git.
 - O sistema deverá ter, ao menos, a versão 27 do Docker Engine e 2 do Git.
 - Requisitos de Segurança
 - Os binários distribuídos pelo projeto devem ser assinados com chaves criptográficas OpenPGP.
 - Requisitos de Licenciamento
 - O usuário estará protegido pela *GNU General Public License v3* ou versão posterior.
 - O código do sistema será disponibilizado de forma aberta.

DOCUMENTAÇÃO

Toda a documentação detalhada, incluindo instruções avançadas, exemplos de uso, opções de configuração e manutenção, estará disponível por meio de *manpages* escritas em GNU roff. Estas servirão como referência robusta para usuários que desejem explorar todas as funcionalidades do sistema.

No repositório, um arquivo README.md conterá uma documentação simplificada e objetiva. Este guia oferecerá as informações essenciais sobre instalação, configuração básica e utilização imediata da ferramenta, facilitando o acesso rápido e intuitivo aos principais recursos do sistema.