Documentação de Projeto

para o sistema

HookCI

Versão 1.0

Projeto de sistema elaborado pelo(s) aluno(s) Gabriel Dolabela Marques e Henrique Carvalho Almeida e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo da professora Joana Gabriela Ribeiro de Souza, orientação acadêmica do professor Cleiton Silva Tavares e orientação de TCC II do professor (a ser definido no próximo semestre).

06/04/2025

Tabela de Conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Gabriel Dolabela Henrique Almeida	06/04	Criação do documento	0.1.0

1. Introdução

Este documento agrega: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o sistema HookCI. A referência principal para a descrição geral do problema, domínio e requisitos do sistema é este documento de especificação que descreve a visão de domínio do sistema. Tal especificação acompanha este documento. Anexo a este documento também se encontra o Glossário.
HookCI consiste em uma ferramenta para a execução da Integração Contínua (CI, do inglês *Continuous Integration*) de forma local, integrando Docker e Git Hooks para validar *commits* e *pushes*.
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 retorno 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.
Outras ferramentas como Husky executam os comandos diretamente no ambiente local. Pretende-se 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.

2. Modelos de Usuário e Requisitos

2.1 Descrição de Atores

Desenvolvedor: É o ator principal, responsável por implementar funcionalidades, escrever código e executar testes. Utiliza o HookCI para validar *commits* e *pushes*, garantindo que somente alterações que passam na suíte de testes sejam integradas ao repositório. O ator se beneficia do retorno imediato proporcionado pela execução dos testes em ambiente local, que utiliza Docker para isolamento e consistência.

Desenvolvedor / Líder Técnico: Atua na configuração e na manutenção do ambiente de desenvolvimento e dos parâmetros de execução da ferramenta. É responsável por definir, por meio de arquivos YAML, as regras e os comandos de teste a serem utilizados pelo HookCI.

2.2 Modelos de Usuários

Para melhor compreender as necessidades e expectativas dos usuários do HookCI, foram criadas personas que representam os perfis de uso da ferramenta. A seguir, são apresentadas duas personas que englobam os principais perfis de desenvolvedores e gestores envolvidos.
A Tabela 1 descreve a persona do usuário Márcio Nunes, um desenvolvedor iniciante que deseja ter um retorno rápido e assertivo relativo à qualidade de seu código.

	Márcio Nunes, Desenvolvedor Júnior
Descrição	Márcio tem 23 anos e está iniciando sua carreira como desenvolvedor. Trabalha em uma pequena equipe de desenvolvimento e precisa de ferramentas que facilitem o aprendizado e ofereçam retorno rápido sobre a qualidade do código.
Objetivos	• Obter respostas imediatas sobre possíveis falhas em seus commits, permitindo aprendizado contínuo. • Adquirir confiança na integração de novas funcionalidades sem comprometer a integridade do repositório.
Dores	• Dificuldade em identificar rapidamente os erros decorrentes de testes mal sucedidos. • Processos de CI que, por serem remotos, atrasam seu fluxo de trabalho.

O sistema HookCI fornece retorno imediato através da execução dos testes localmente e isolando o ambiente de testes com Docker, garantindo consistência e evitando conflitos com as configurações locais.
A Tabela 2 descreve a persona do usuário Paulo Lopes, um desenvolvedor líder de um time que deseja garantir a qualidade de seu projeto a todo momento de forma automatizada e padronizada.

	Paulo Lopes, Desenvolvedor Sênior
Descrição	Paulo tem 29 anos e atua como desenvolvedor sênior em uma equipe de médio porte. Além de codificar, ele também assume funções de liderança técnica, configurando o ambiente de desenvolvimento e definindo as diretrizes para a integração contínua.
Objetivos	• Estabelecer um fluxo de trabalho que minimize falhas e garanta a qualidade do código entregue. • Otimizar o tempo de execução dos testes, mantendo um ciclo de retorno rápido e eficiente.
Dores	• Dificuldade em padronizar os ambientes de desenvolvimento entre os membros da equipe.

	• Gerenciar configurações de testes que, quando executados em ambientes distintos, podem apresentar resultados inconsistentes.

O sistema HookCI possibilita configuração centralizada via arquivos YAML, facilitando a instalação automatizada dos Git Hooks e isolando os testes em containers Docker, assegurando que todos os membros da equipe trabalhem com o mesmo ambiente de CI, independentemente das particularidades de suas máquinas.

2.3 Modelo de Casos de Uso e Histórias de Usuários

Esta subseção apresenta o diagrama de casos de uso e as histórias de usuários que guiam o desenvolvimento da aplicação HookCI.

2.3.1 Diagrama de Casos de Uso

A Figura 1 ilustra o diagrama de casos de uso do sistema HookCI, no qual são representados quatro atores: Usuário, Desenvolvedor, Administrador e o sistema Git. O Usuário engloba os agentes humanos que interagem com a documentação do sistema; o Desenvolvedor inicia a execução de testes por meio da CLI; o Administrador realiza a inicialização do projeto na ferramenta; e o sistema Git, por sua vez, dispara a execução dos testes automaticamente via Git hooks.

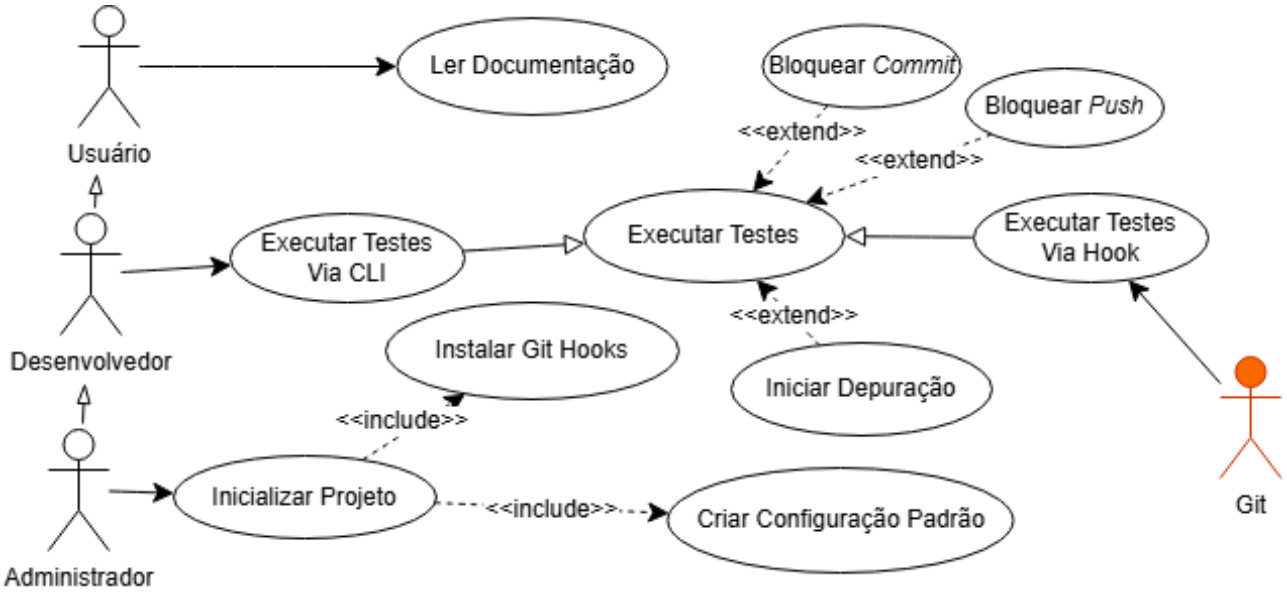

Figura 1. Diagrama de caso de uso

2.3.2 História de Usuários

US01:

Como desenvolvedor, desejo inicializar um projeto com o HookCI, para configurar o ambiente local e preparar a estrutura básica de integração contínua.

US02:

Como desenvolvedor, desejo que seja criado um arquivo YAML de configuração de forma automática, para definir os parâmetros de execução de testes no projeto.

US03:

Como desenvolvedor, desejo que os hooks do Git sejam instalados automaticamente no projeto, para evitar configurações manuais propensas a erro.

US04:

Como desenvolvedor, desejo editar o arquivo de configuração YAML, para personalizar comandos, variáveis de ambiente e políticas de execução.

US05:

Como desenvolvedor, desejo que os testes sejam executados automaticamente ao realizar um commit ou push, para evitar que código defeituoso seja integrado ao repositório.

US06:

Como desenvolvedor, desejo executar testes manualmente via CLI, para verificar o comportamento do sistema sob demanda.

US07:

Como desenvolvedor, desejo que os testes sejam executados em um ambiente Docker isolado, para garantir a consistência entre ambientes de desenvolvimento.

US08:

Como desenvolvedor, desejo visualizar os logs da execução de testes na linha de comando em caso de erro, a fim de entender o que foi executado e identificar falhas rapidamente.

US09:

Como desenvolvedor, desejo ajustar o nível de verbosidade dos logs da CLI, para escolher entre mensagens mais detalhadas ou mais concisas conforme a situação.

US10:

Como desenvolvedor, desejo ter acesso a um modo de depuração em caso de falha nos testes, para investigar interativamente o ambiente de execução e entender a origem dos problemas.

US11:

Como desenvolvedor, desejo validar a estrutura e os valores do arquivo YAML de configuração, para garantir que ele esteja correto antes da execução.

US12:

Como desenvolvedor, desejo acessar a documentação dos comandos disponíveis via CLI, para entender como utilizar a ferramenta corretamente

US13:

Como desenvolvedor, desejo configurar testes não essenciais que apenas avisem o desenvolvedor caso falhem.

US14:

Como desenvolvedor, desejo configurar uma imagem Docker personalizada através de um nome e versão.

US15:

Como desenvolvedor, desejo utilizar um *Dockerfile* para definir o ambiente de testes, para maior controle sobre dependências e ferramentas instaladas.

US16:

Como desenvolvedor, desejo montar automaticamente o diretório atual do repositório no contêiner durante a execução dos testes, para garantir que o código-fonte correto seja testado.

2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Nesta subseção é apresentado o diagrama de sequência do sistema e os Contratos de Operações.

Formato para cada contrato de operação

Contrato	
Operação	
Referências cruzadas	
Pré-condições	
Pós-condições	

3. Modelos de Projeto

3.1 Diagrama de Classes

Diagrama de classes do sistema

3.2 Diagramas de Sequência

Diagramas de sequência para realização de casos de uso.

3.3 Diagramas de Comunicação

Diagramas de comunicação para realização de casos de uso.

3.4 Arquitetura

Pode ser descrita com um diagrama apropriado da UML ou C4 Model

3.5 Diagramas de Estados

Diagramas de estados do sistema.

3.6 Diagrama de Componentes e Implantação.

Diagramas de componentes do sistema. Diagrama de implantação mostrando onde os componentes estarão alocados para a execução.

4. Projeto de Interface com Usuário

Esta seção apresenta as principais interações do usuário com o sistema HookCI por meio de sua CLI. Como o sistema opera exclusivamente via terminal, não são apresentados *mockups* ou *wireframes* gráficos. Em vez disso, são exibidas representações textuais das saídas geradas pelos comandos principais, ilustrando o fluxo de interação e o retorno fornecido ao usuário. Essas interações foram projetadas para cobrir os casos de uso definidos na Seção 2.3.

4.1 Esboço das Interfaces Comuns a Todos os Atores

A Figura 2 exibe a saída do comando de ajuda, invocado quando o usuário executa a ferramenta sem argumentos ou com o comando help. A interface lista os comandos disponíveis e fornece uma breve descrição de cada um, auxiliando o usuário a entender as funcionalidades da ferramenta. Esta tela contempla diretamente o caso de uso US12 (Acessar documentação dos comandos via CLI).

Figura 2. Descrição de comandos disponíveis

A Figura 3 mostra a resposta do sistema quando um comando inválido é fornecido pelo usuário. A interface apresenta uma mensagem de erro clara, indicando qual comando foi considerado desconhecido e sugerindo o uso do comando help para visualizar as opções válidas. Embora não corresponda diretamente a um caso de uso funcional principal, esta interface é crucial para a usabilidade e orientação do usuário, relacionando-se indiretamente com US12 ao direcionar para a ajuda.

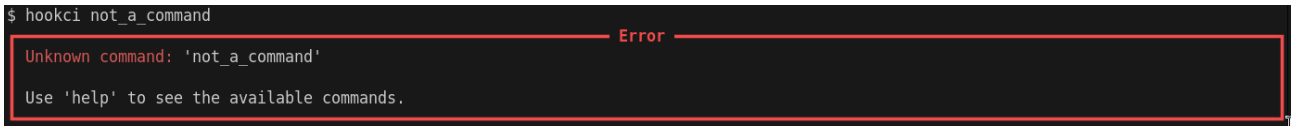

Figura 3. Comando inválido

4.2 Esboço das Interfaces Usadas pelo Desenvolvedor

A Figura 4 demonstra a saída da CLI durante a execução dos testes, acionada manualmente pelo comando test ou automaticamente por um Git hook (commit ou push). A interface exibe o progresso das etapas principais, como a inicialização, criação do contêiner Docker e execução da suíte de testes, utilizando indicadores visuais (⏳ para pendente/executando, ✔ para concluído) para fornecer retorno claro sobre o andamento. Esta interação está relacionada aos casos de uso US05 (Executar testes automaticamente), US06 (Executar testes manualmente via CLI) e US07 (Executar testes em ambiente Docker).

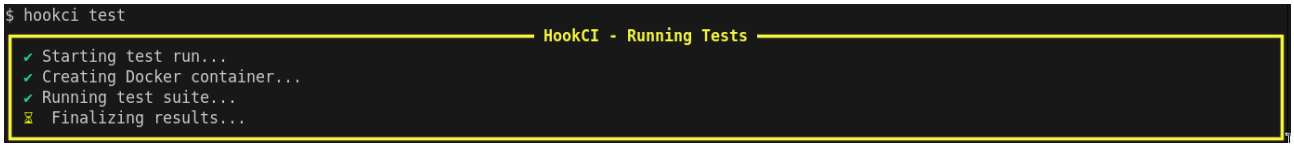

Figura 4. Execução de testes em andamento

A Figura 5 ilustra a conclusão bem-sucedida da execução dos testes. Todas as etapas são marcadas como concluídas (✔), e uma mensagem final confirma o sucesso da operação. Assim como a Figura 4, esta saída está associada aos casos de uso US05, US06 e US07, representando o estado final esperado quando os testes passam sem erros. Adicionalmente, relaciona-se com US08 (Visualizar logs), pois apresenta o resultado sumarizado da execução.

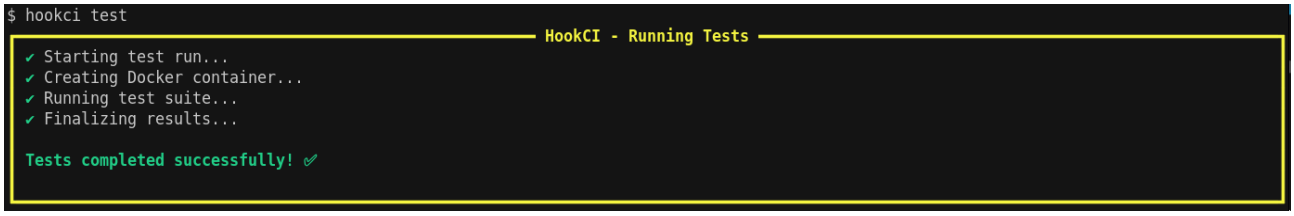

Figura 5. Execução de testes bem sucedida

4.3 Esboço das Interfaces Usadas pelo Administrador

A Figura 6 representa a saída do comando init, responsável por inicializar o HookCI em um repositório. A interface informa o início do processo e detalha as ações realizadas, como a criação do arquivo de configuração YAML, a instalação dos Git hooks necessários e a verificação de dependências. Esta interação abrange os casos de uso US01 (Inicializar projeto com HookCI), US02 (Criar arquivo YAML automaticamente) e US03 (Instalar hooks do Git automaticamente).

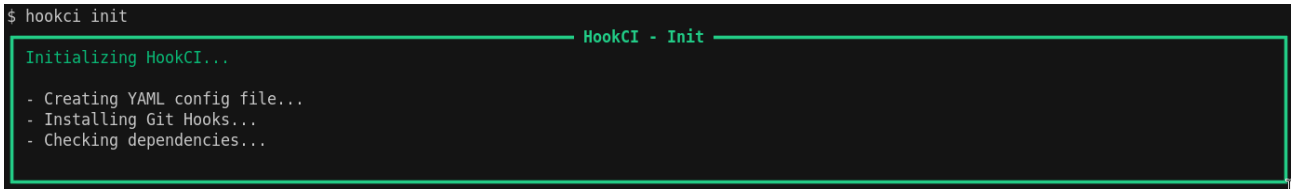

Figura 6. Inicialização de um repositório

5. Glossário e Modelos de Dados

Deve-se apresentar o glossário para o sistema. Também apresente esquemas de banco de dados e as estratégias de mapeamento entre as representações de objetos e não-objetos.

6. Casos de Teste

Uma descrição de casos de teste para validação do sistema.

7. Cronograma e Processo de Implementação

Uma descrição do cronograma para implementação do sistema e do processo que será seguido durante a implementação.