Documentação de Projeto

para o sistema

HookCI

Versão 1.0

Projeto de sistema elaborado pelo(s) aluno(s) AAAA e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo do professor PPP, orientação acadêmica do professor YYY e orientação de TCC II do professor (a ser definido no próximo semestre).

02/04/2025

Tabela de Conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Henrique Almeida	02/04	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 o 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 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.

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. Este ator se beneficia do feedback imediato proporcionado pela execução dos testes em ambiente local, que utiliza Docker para isolamento e consistência.

Administrador de Projeto / 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.

Engenheiro *DevOps*: Atua na configuração dos containers Docker e na otimização dos ambientes de execução dos testes. Este ator contribui para que o ambiente local mantenha as características de isolamento e padronização necessárias, além de possibilitar a escalabilidade do processo de CI conforme o crescimento dos projetos.

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 *feedback* 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 feedback 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 ajuda a persona acima fornecendo feedback 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 uma forma simples e eficiente.

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 feedback rápido e eficiente.

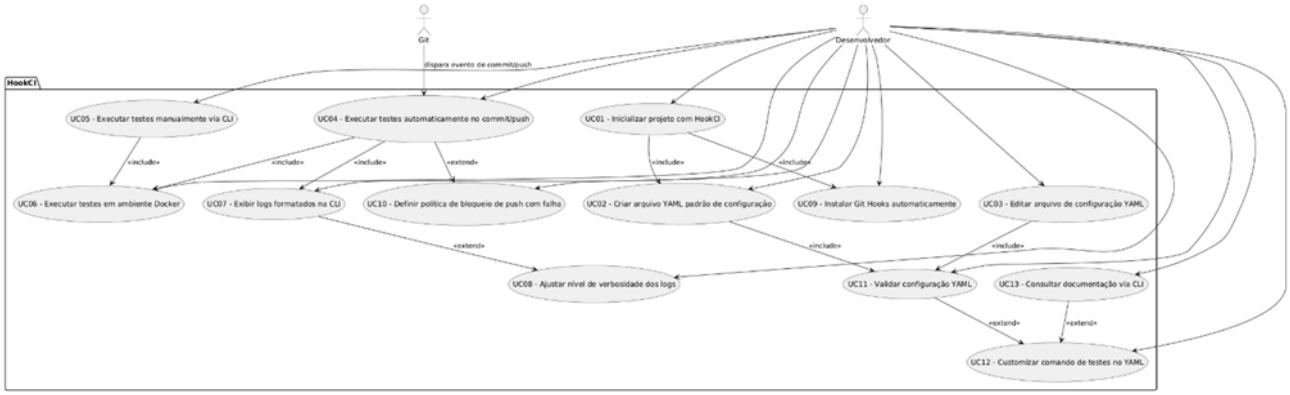
Dores	• Dificuldade em padronizar os ambientes de desenvolvimento entre os membros da equipe. • Gerenciar configurações de testes que, quando executados em ambientes distintos, podem apresentar resultados inconsistentes.

O sistema HookCI ajuda a persona acima permitindo a 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



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 criar automaticamente um arquivo YAML de configuração, para definir os parâmetros de execução de testes no projeto.

US03:

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

US04:

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

US05:

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

US06:

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

US07:

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

US08:

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

US09:

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

US10:

Como desenvolvedor, desejo configurar se o push deve ser bloqueado quando os testes falharem, para garantir que apenas código validado seja enviado ao repositório remoto.

US11:

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

US12:

Como desenvolvedor, desejo customizar o comando de testes que será utilizado, para adaptar a execução ao framework ou ferramenta de testes do projeto.

US13:

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

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.

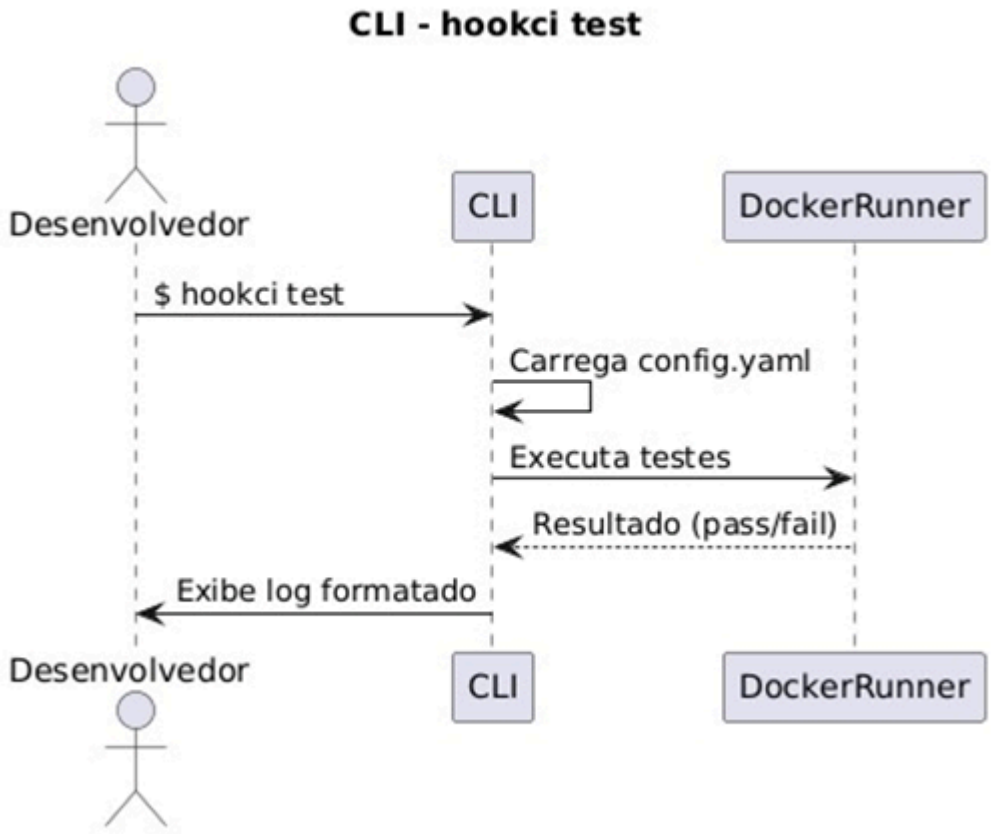
Diagramas de componentes do sistema. Diagrama de implantação mostrando onde os componentes estarão alocados para a execução.

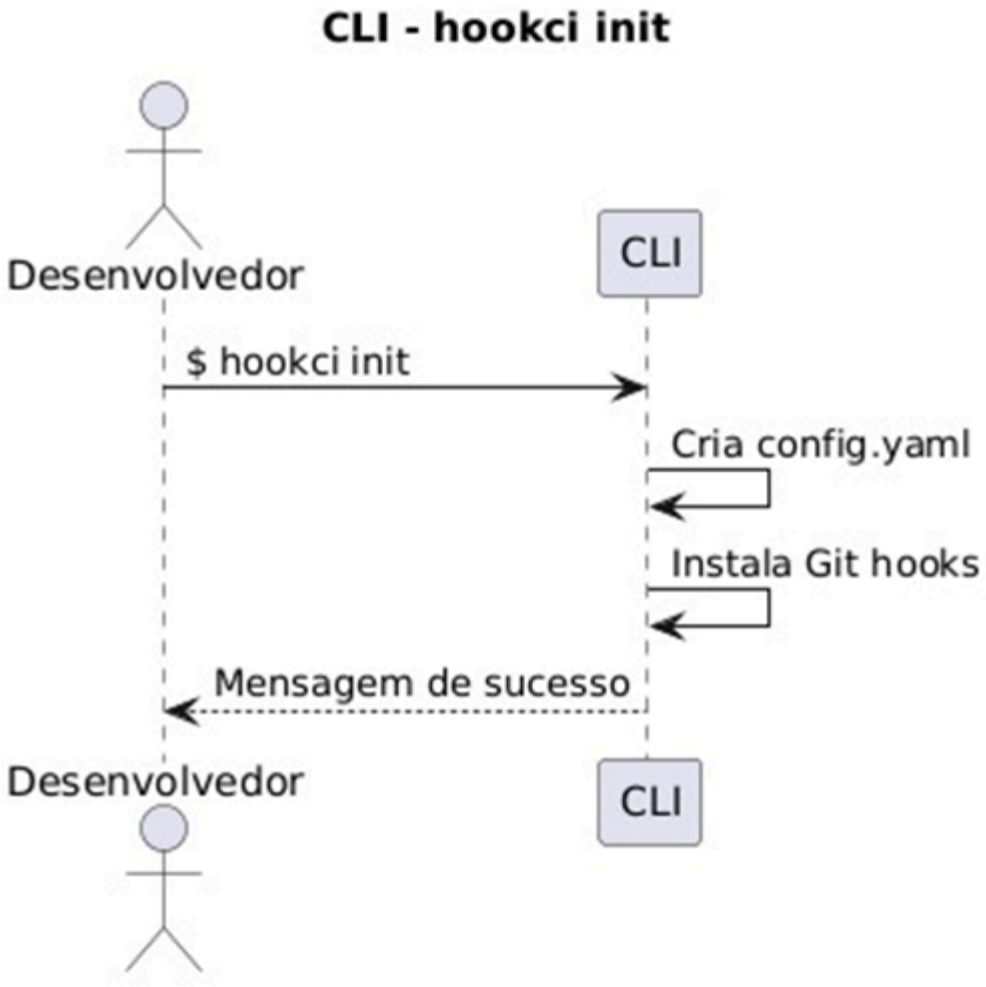
4. Projeto de Interface com Usuário

4.1 Esboço das Interfaces Comuns a Todos os Atores

Wireframe/mockup/storyboard das interfaces que são comuns a todos os atores do sistema.

O sistema HookCI é operado exclusivamente via interface de linha de comando (CLI). Portanto, os esboços se concentram em fluxos e estados exibidos ao usuário ao interagir com os comandos do sistema.





4.2 Esboço das Interfaces Usadas pelo Ator <*A*>

Wireframe/mockup/storyboard das interfaces exclusivas do ator <*A*>

4.3 Esboço das Interfaces Usadas pelo Ator <*B*>

Wireframe/mockup/storyboard das interfaces exclusivas do ator <B>

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.