Parte Teórica - Construindo um Projeto Ágil no GitHub: Da Gestão ao Controle de Qualidade

Descrição do Projeto e seu Escopo Inicial

Este projeto foi desenvolvido para a TechFlow Solutions, uma empresa fictícia especializada em soluções de software, contratada por uma startup de logística. O escopo inicial consiste na criação de um sistema de gerenciamento de tarefas baseado em metodologias ágeis, com o objetivo de permitir o acompanhamento do fluxo de trabalho em tempo real, a priorização de tarefas críticas e o monitoramento do desempenho da equipe. A aplicação web, implementada em Flask, oferece funcionalidades básicas de CRUD (Create, Read, Update, Delete) para gerenciar tarefas, utilizando o GitHub como plataforma central para versionamento, organização e controle de qualidade.

Metodologia Ágil Utilizada

A metodologia escolhida foi o Kanban, implementada por meio da aba Projects do GitHub, com as colunas "A Fazer", "Em Progresso" e "Concluído". Essa abordagem foi adotada por promover uma gestão visual e flexível, permitindo a priorização dinâmica de tarefas e a adaptação contínua às mudanças de escopo, características essenciais para projetos ágeis em equipes de logística que demandam agilidade e visibilidade.

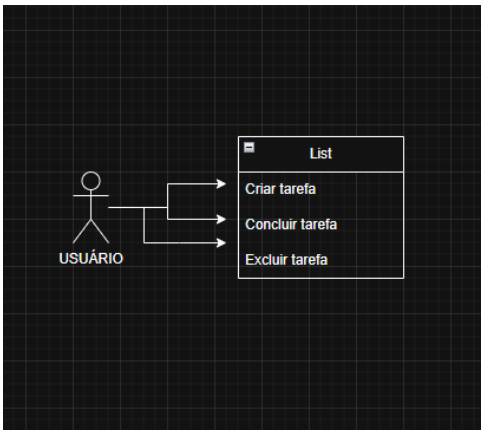
Importância da Modelagem na Engenharia de Software

A modelagem desempenha um papel crucial na Engenharia de Software, pois estrutura o projeto, clarifica os requisitos e facilita a comunicação entre a equipe e os stakeholders. Neste trabalho, ela serviu como base para definir as interações do sistema (via diagrama de casos de uso) e as classes necessárias (via diagrama de classes), reduzindo ambiguidades e erros durante a implementação do CRUD e da funcionalidade de notificação.

Diagramas UML Obrigatórios

Diagrama de Casos de Uso

- **Atores**: Usuário (dividido em Gerente e Membro da Equipe).
- **Casos de Uso**:
    - Adicionar Tarefa (iniciado pelo Gerente).
    - Marcar Tarefa como Concluída (executado pelo Membro).
    - Excluir Tarefa (realizado pelo Gerente).
    - Visualizar Lista de Tarefas (acessível a ambos).

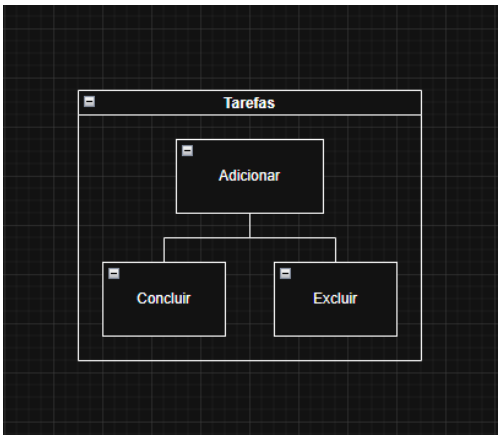


- **Descrição**: O sistema permite que o Gerente adicione e remova tarefas, enquanto os Membros atualizam o status e consultam a lista, garantindo um fluxo colaborativo.

Diagrama de Classes

- **Classes**:
    - Tarefa (atributos: título [string], descrição [string], completada [boolean]).
    - Usuário (atributos: nome [string], cargo [string]).

- **Relações**: Um Usuário gerencia várias Tarefas (associação 1:N), refletindo a estrutura hierárquica do sistema.



Breve Justificativa sobre a Mudança de Escopo

Durante o desenvolvimento, optou-se por adicionar uma funcionalidade de notificação de tarefas pendentes por mais de 7 dias. Essa alteração foi motivada pela necessidade de aumentar a proatividade da equipe, evitando atrasos no fluxo de trabalho. A mudança foi documentada no README.md e registrada no quadro Kanban com um novo card, demonstrando a adaptabilidade do projeto ágil.

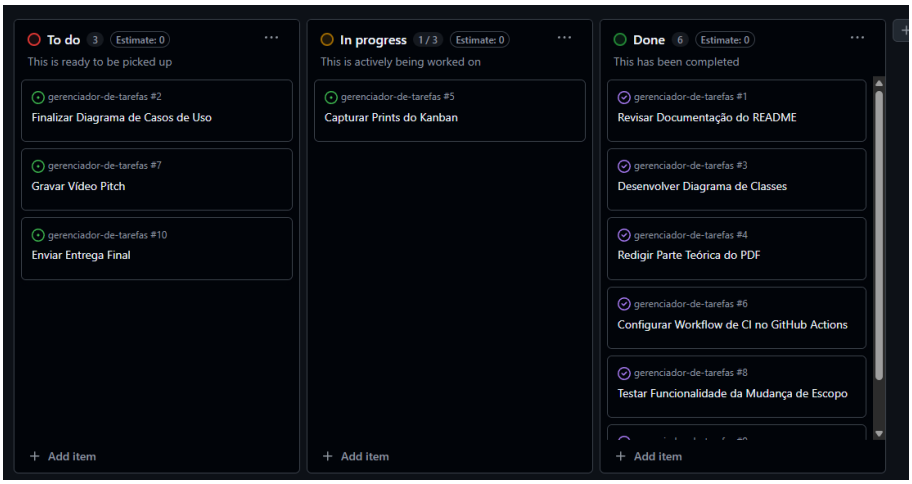
Explicação sobre os Testes Automatizados Utilizados

Os testes automatizados foram implementados utilizando PyTest, com foco na validação de entradas (ex.: título de tarefa não vazio) e na funcionalidade do CRUD (criação, leitura,

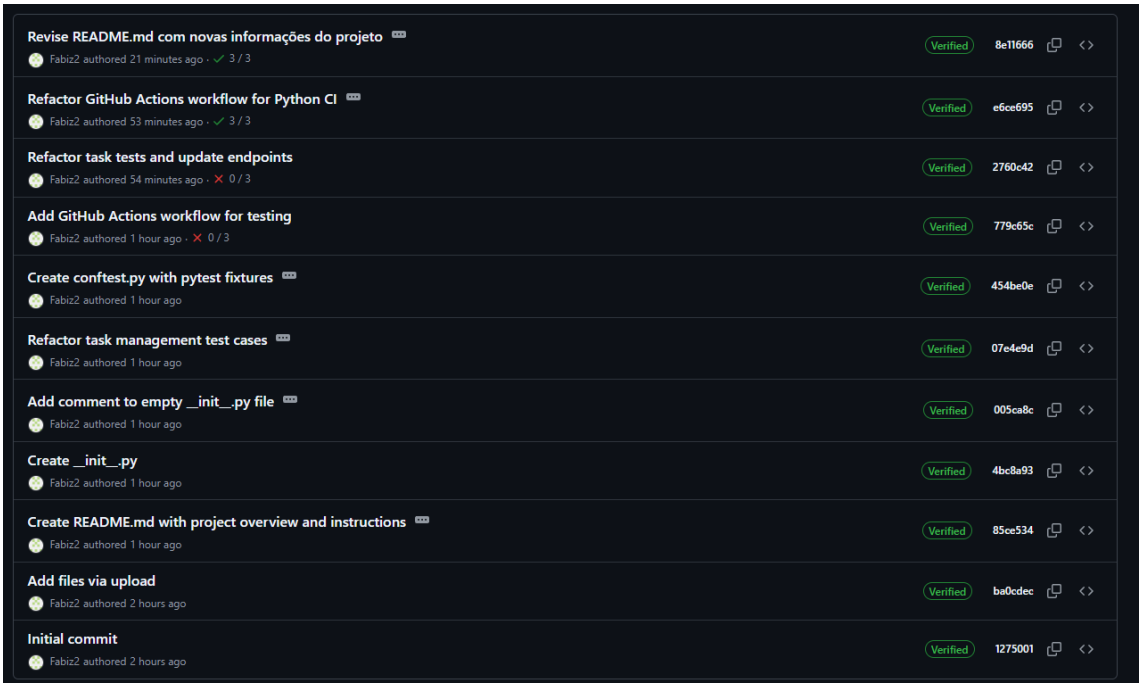
atualização e exclusão). Esses testes foram integrados a um pipeline de GitHub Actions, que executa verificações automáticas a cada commit, assegurando a qualidade do código, detectando erros precocemente e promovendo entregas confiáveis.

Prints Comentados do GitHub

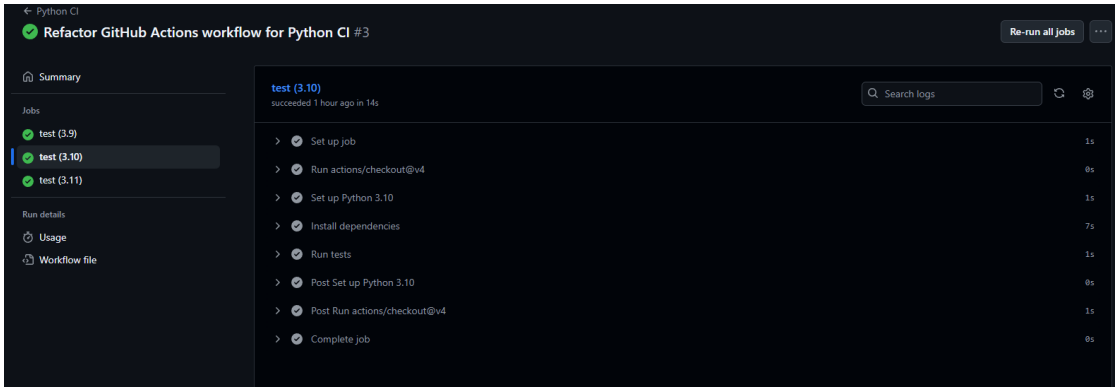
- **Kanban com Tarefas**: Imagem mostrando as colunas "A Fazer", "Em Progresso" e "Concluído" com pelo menos 10 cards distribuídos, como "Configurar repositório" (Concluído), "Gravar vídeo pitch" (A Fazer) e "Testar CRUD" (Concluído), evidenciando o progresso visual.



- **Commits Relevantes**: Captura exibindo commits como "Inicializar projeto com Flask" (data: 23/10/2025), "Adicionar funcionalidade de notificação" (data: 24/10/2025) e "Configurar pipeline de CI" (data: 24/10/2025), destacando a evolução do trabalho.



- **Workflow CI**: Imagem do GitHub Actions mostrando o pipeline rodando testes PyTest com status "sucesso" (data: 24/10/2025), confirmando o controle de qualidade.



Reflexões sobre as Questões Norteadoras

- **Quais são as principais causas de falhas em projetos ágeis e como o GitHub pode ajudar a mitigá-las?**
  Falhas como má gestão de tarefas e falta de comunicação são comuns. O GitHub mitiga isso com o Kanban visual, que centraliza o acompanhamento, e os commits descritivos, que registram o progresso, reduzindo ambiguidades.

- **Quem são os principais beneficiados por um sistema de gerenciamento ágil e como eles utilizam as funcionalidades desenvolvidas?**
  Gerentes e equipes de logística são os principais beneficiados. Os Gerentes utilizam o CRUD para priorizar tarefas, enquanto os Membros atualizam status e consultam a lista, otimizando o fluxo de trabalho em tempo real.

- **Como o uso de ferramentas de controle de qualidade, como GitHub Actions, pode garantir a entrega de um software confiável?**
  GitHub Actions executa testes automatizados (PyTest) a cada alteração, validando funcionalidades como o CRUD e detectando erros antes da integração, assegurando um software estável e confiável.

- **Quais são os principais desafios ao implementar mudanças em um projeto ágil e como lidar com eles?**
  Desafios incluem resistência à mudança e scope creep. A solução é manter comunicação frequente via Kanban e documentar ajustes (ex.: notificação de tarefas) no README, garantindo alinhamento da equipe.

- **Como as metodologias ágeis estudadas na disciplina podem ser aplicadas diretamente neste projeto?**
  O Kanban aplicado reflete a flexibilidade e a iteração contínua ensinadas, conectando teoria à prática ao permitir ajustes dinâmicos, como a adição da funcionalidade de notificação, em resposta às necessidades do cliente.