TechFlow Solutions

1. Descrição do Projeto e Escopo Inicial

Este projeto simula o desenvolvimento de um sistema de gerenciamento de tarefas para a "TechFlow Solutions", uma empresa fictícia contratada por uma startup de logística. O objetivo é permitir que a equipe de logística acompanhe seu fluxo de trabalho, priorize tarefas e monitore o desempenho.

O escopo inicial (MVP - Minimum Viable Product) definido para este projeto concentrou-se na criação da funcionalidade central de gerenciamento de tarefas, dividida em duas frentes principais:

API Backend (Node.js & Express):

Um servidor RESTful foi implementado para gerenciar o ciclo de vida das tarefas, oferecendo um conjunto completo de operações CRUD (Create, Read, Update, Delete) para as tarefas. Os endpoints definidos são:

- POST /tasks: Cria uma nova tarefa.
- GET /tasks: Lista todas as tarefas existentes.
- PUT /tasks/:id: Atualiza uma tarefa específica (ex: título, status).
- DELETE /tasks/:id: Exclui uma tarefa.

Interface Frontend (HTML, CSS & Vanilla JS):

Uma interface de usuário (UI) web foi desenvolvida para consumir a API do backend. Esta interface inclui:

- Um formulário para a criação de novas tarefas.
- Um quadro visual estilo Kanban com colunas "A Fazer", "Em Progresso" e "Concluído", onde as tarefas são renderizadas.
- Funcionalidade de "Arrastar e Soltar" (Drag-and-Drop) para mover tarefas entre as colunas, atualizando seu status em tempo real via API.
- Funcionalidade para deletar tarefas diretamente da interface.

2. Metodologia Ágil Utilizada

Para este projeto, foi adotada uma **Metodologia Híbrida**, combinando os pontos fortes do Kanban e do Scrum .

- **Kanban:** Utilizado para a gestão visual e o fluxo de trabalho contínuo. Um quadro no GitHub Projects foi configurado com as colunas "A Fazer"

(Backlog), "Em Progresso" e "Concluído", permitindo uma visualização clara do progresso de cada funcionalidade (tanto de backend quanto de frontend).

- **Scrum:** Os conceitos do Scrum foram aplicados no planejamento. O trabalho foi dividido em Features (ou Epics) claras, como "Implementar API CRUD" e "Desenvolver Interface Kanban". Cada funcionalidade foi tratada como um pequeno incremento de valor entregue, refletindo a natureza iterativa do Scrum. A "Simulação de Mudança" também seguiu o princípio de adaptabilidade e repriorização do backlog, central para o Scrum.

3. Explicação sobre a Importância da Modelagem

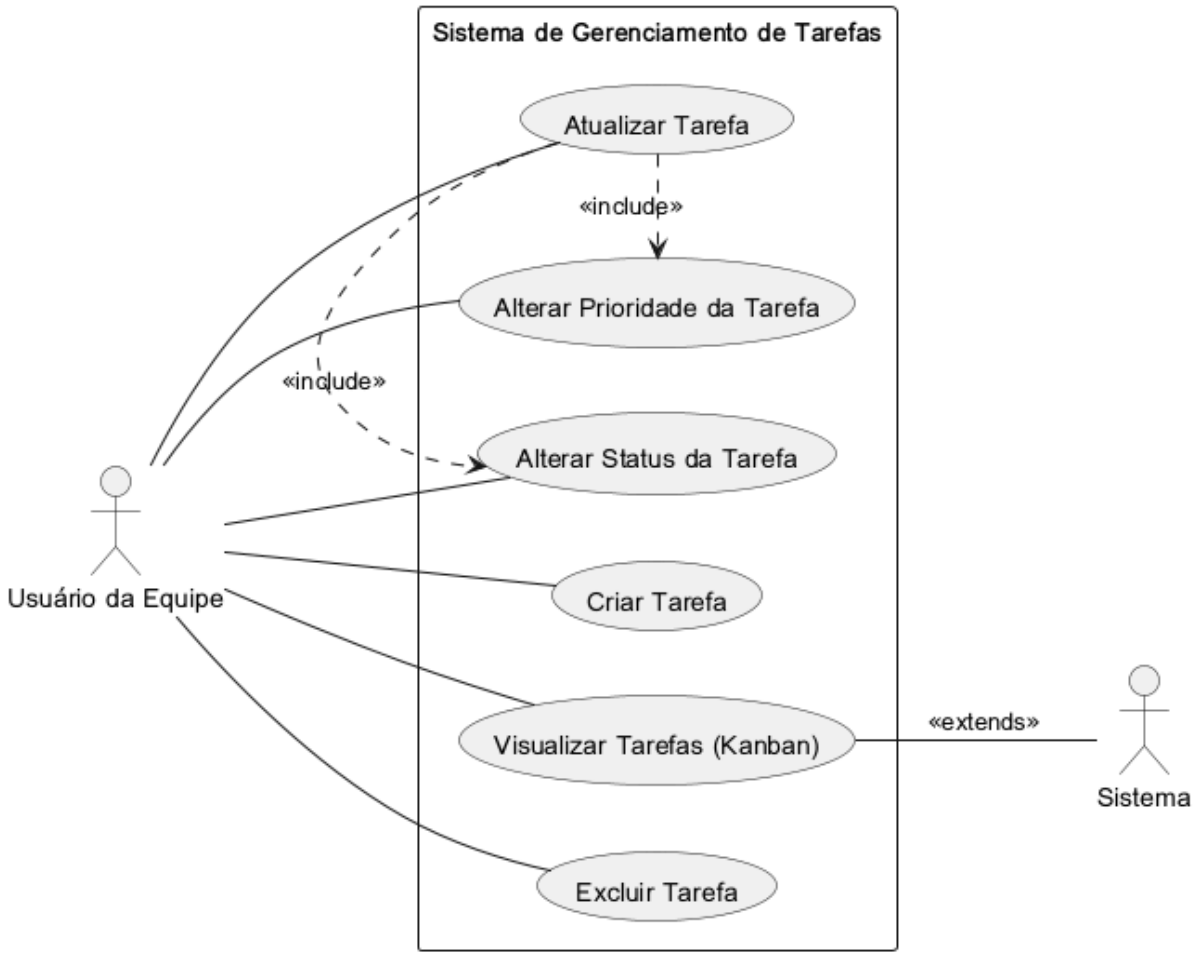
A modelagem na engenharia de software é o processo de criar abstrações (modelos) para descrever um sistema de software em diferentes níveis de detalhe. Sua importância é fundamental por diversas razões :

- **Comunicação (A "Língua Franca"):** Modelos, como os diagramas UML, servem como uma linguagem visual universal. Eles permitem que stakeholders (clientes), gerentes de projeto, designers e desenvolvedores tenham uma compreensão compartilhada do que o sistema deve fazer (Casos de Uso) e como ele é estruturado (Diagrama de Classes), reduzindo ambiguidades.
- **Detecção Antecipada de Falhas:** É exponencialmente mais barato corrigir um erro de lógica no papel (ou em um diagrama) do que em um sistema já codificado e em produção. A modelagem ajuda a identificar requisitos conflitantes, funcionalidades ausentes e estruturas de dados ineficientes antes que a primeira linha de código seja escrita.
- **Abstração e Complexidade:** Softwares modernos são complexos. A modelagem permite "dividir para conquistar", decompondo um sistema grande em componentes menores e mais fáceis de entender (classes, módulos, serviços). Isso torna o design e a implementação muito mais gerenciáveis.
- **Documentação (A "Planta" do Sistema):** Os modelos servem como a "planta" do software. Para manutenção futura ou para a integração de novos membros na equipe, essa documentação é vital para entender a arquitetura do sistema, as responsabilidades de cada componente e as regras de negócio implementadas.

4. Diagramas UML

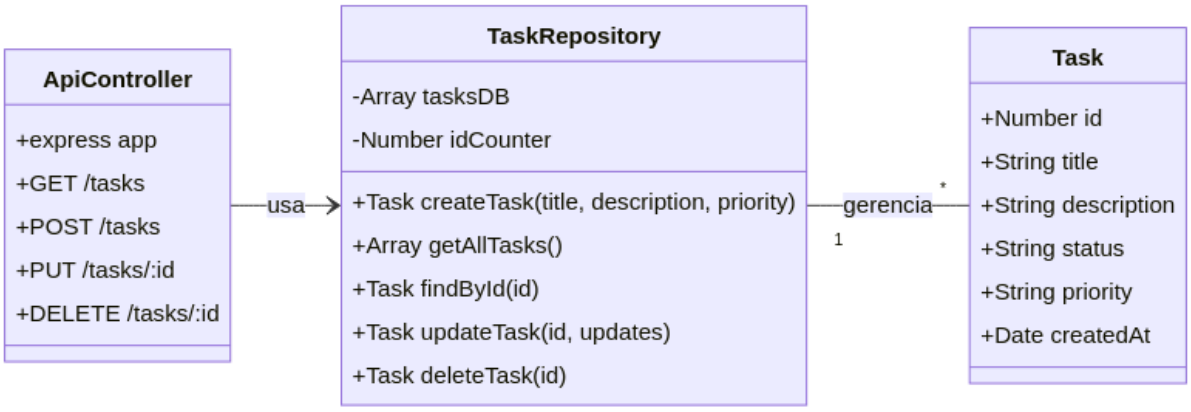
4.1. Diagrama de Casos de Uso

Este diagrama mostra as interações entre o ator (usuário) e as principais funcionalidades do sistema



4.2. Diagrama de Classes

Este diagrama descreve a estrutura do nosso sistema backend, focando na entidade Task e no TaskRepository que a gerência.



5. Breve Justificativa sobre a Mudança de Escopo

Durante o desenvolvimento, foi simulada uma solicitação de mudança vinda do cliente (a startup de logística), conforme a Task 6 do projeto .

A Mudança: Adição de um campo priority (prioridade) ao modelo de Tarefas, permitindo a classificação em 'Baixa', 'Média' ou 'Alta'.

Justificativa: O cliente percebeu que, para seu fluxo de trabalho de logística, apenas o status ('A Fazer', 'Em Progresso') não era suficiente. Eles precisavam de uma forma de sinalizar quais tarefas eram críticas e deveriam ser feitas primeiro.

Impacto e Ação (Ágil): A mudança foi gerenciada de forma ágil:

- **Documentação:** A mudança e sua justificativa foram registradas no README.md .
- **Planejamento:** Novas tarefas foram criadas no Kanban ("A Fazer") para refletir o trabalho necessário.
- **Implementação:** A mudança foi aplicada em todo o stack:
    - **Backend:** taskRepository.js foi modificado para aceitar e definir um valor padrão para priority.
    - **API:** A rota POST /tasks foi atualizada para receber o novo campo. A rota PUT /tasks já estava pronta para atualizá-lo.
    - **Testes:** Os testes do Jest foram atualizados para validar o novo campo.
    - **Frontend:** O index.html (formulário) e o app.js (envio do POST e renderização) foram atualizados para incluir a seleção e exibição da prioridade.

6. Explicação sobre os Testes Automatizados

O controle de qualidade foi implementado no backend usando um pipeline de Integração Contínua (CI) .

Ferramentas: Foram utilizados o Jest (um framework de testes do JavaScript) em conjunto com o Supertest (uma biblioteca para testar APIs HTTP simulando requisições).

Testes Criados: Testes essenciais foram escritos para validar o comportamento da API, incluindo:

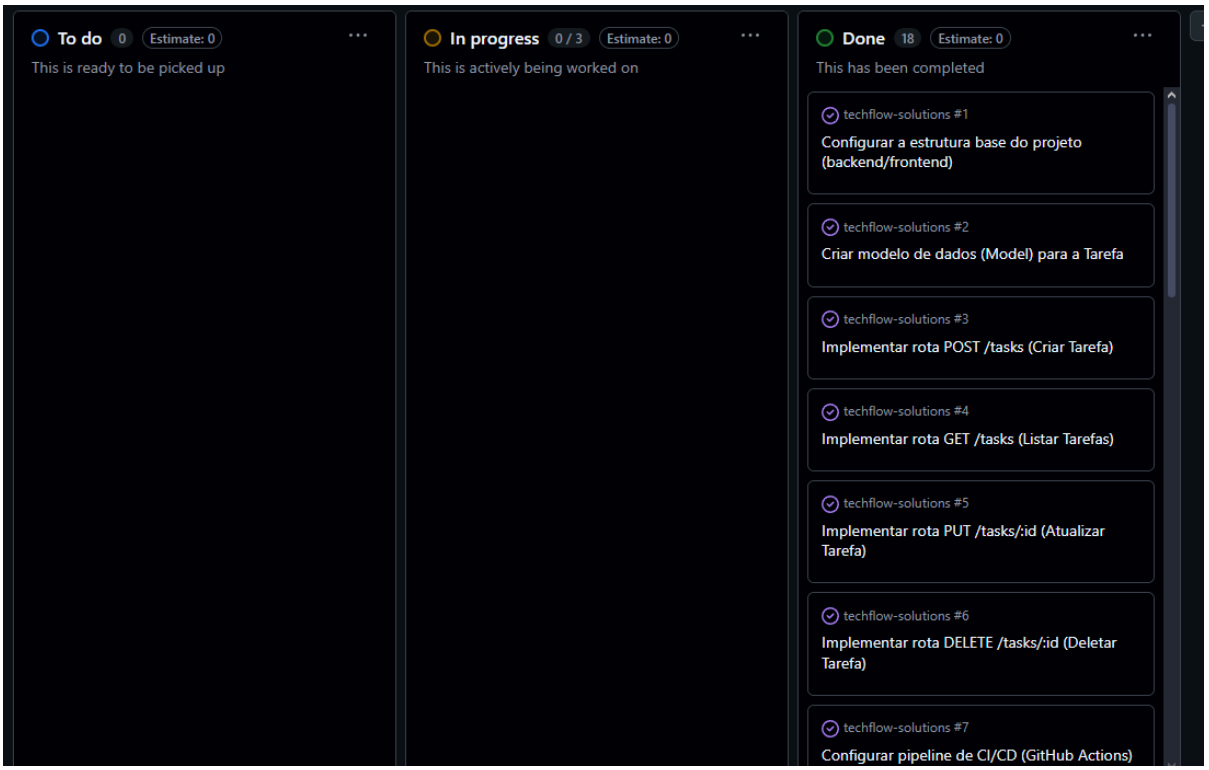
- **Teste de Sucesso (POST):** Garante que POST /tasks com dados válidos retorna Status 201 e o objeto da tarefa criada (incluindo o campo priority).
- **Teste de Validação de Entrada (POST):** Garante que POST /tasks sem um campo title falha e retorna Status 400 (Bad Request), protegendo o sistema contra dados inválidos.
- **Teste de Listagem (GET):** Garante que GET /tasks retorna Status 200 e um array como resposta.

Automação (GitHub Actions): Foi configurado um workflow em .github/workflows/ci.yml. Este pipeline é disparado automaticamente a cada push para a branch main. Ele cria um ambiente virtual, instala as dependências (npm ci) e executa o comando npm test. Se qualquer teste falhar, o GitHub sinaliza a falha, impedindo (em um cenário de produção) que código quebrado seja mesclado.

7. Prints Comentados do GitHub

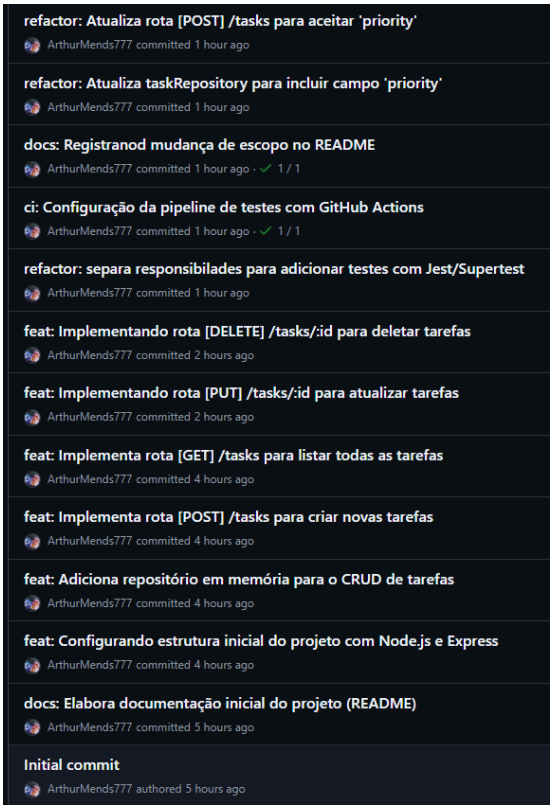
(Esta seção requer a inserção manual de imagens por parte do usuário, conforme as instruções originais. Os prints deverão ser do quadro Kanban, histórico de commits e workflow de CI do repositório GitHub.)

Print 1: Quadro Kanban (GitHub Projects)



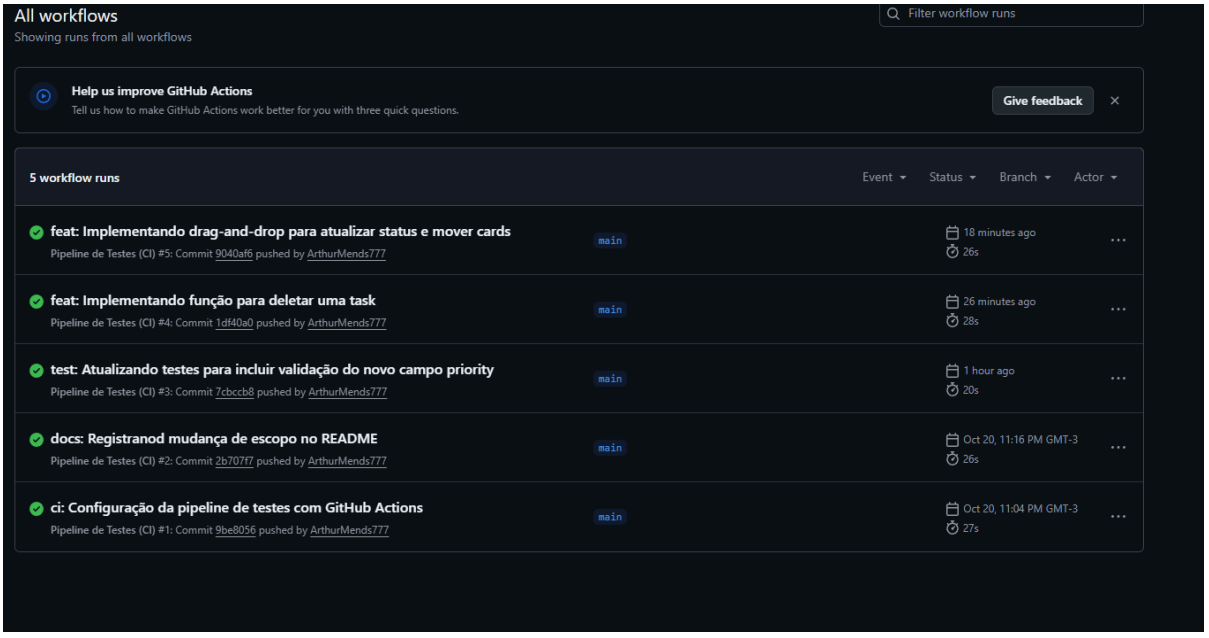
Este print mostra o quadro Kanban do projeto, dividido em "A Fazer", "Em Progresso" e "Concluído". As tarefas (cards) estão claramente prefixadas (ex: [FRONT], [MUDANÇA]), mostrando o planejamento visual e o progresso das features de backend e frontend.

Print 2: Histórico de Commits Relevantes



Comentário: O histórico de commits demonstra a aplicação de commits semânticos. Os prefixos (como feat:, refactor:, test:, ci:, docs:) tornam claro o que foi feito e por que. Por exemplo, feat(front): ... indica uma nova funcionalidade no frontend, enquanto ci: Configura pipeline... indica uma mudança na Integração Contínua.

Print 3: Workflow de CI (GitHub Actions) Funcionando



Comentário: Este print mostra o workflow do GitHub Actions ("Pipeline de Testes (CI)") sendo executado com sucesso (indicado pelo ✔ verde). A imagem detalha os passos executados (Configurar Node.js, Instalar dependências, Rodar os testes), provando que os testes automatizados passaram no servidor antes do código ser considerado "pronto".

Link para o projeto: https://github.com/ArthurMends777/techflow-solutions