Relatório Etapa 1 – Kanban-Lite

Anne Fernandes da Costa Oliveira (20240010789)

25/09/2025

Contents

Etapa 1 – Relato Técnico de Design

Projeto: Kanban-Lite (Trabalho Final POO – C++)
Autora: Anne Fernandes da Costa Oliveira
Matrícula: 20240010789
Data: 25/09/2025

---

1. Objetivo desta etapa

Estabelecer a base arquitetural do sistema **Kanban-Lite**, demonstrando domínio dos conceitos fundamentais de Programação Orientada a Objetos antes da implementação completa.

---

2. Escopo contemplado

- Estrutura de diretórios (`design/`, `include/`, `src/`, `tests/`)

- Cabeçalhos principais de domínio
- Diagrama UML preliminar
- Configuração mínima de build com CMake
- Pipeline CI inicial

---

3. Visão Geral do Domínio

O sistema gerencia **quadros (Boards)** compostos por **colunas (Columns)** que organizam **cartões (Cards)** representando tarefas.
Usuários (Users) interagem criando e movendo cartões.
Um **registro de atividades (ActivityLog)** armazena eventos para auditoria.

---

4. Decisões de Design

- Separação headers/implementação para modularidade.
- Uso de IDs simples (`std::string` / `int`), podendo evoluir para UUID.
- ActivityLog centralizado para rastreabilidade.
- Sem herança prematura: polimorfismo será adicionado conforme necessidade.
- Board contém Column, Column contém Card → composição natural do domínio.

---

5. Aplicação dos Conceitos POO

- **Abstração e Encapsulamento:** atributos privados e acesso via getters/setters.
- **Classes e Objetos:** definidas `Board`, `Column`, `Card`, `User` e `ActivityLog`.
- **Herança e Polimorfismo:** não aplicados nesta fase; planejada interface `Persistable` no futuro.
- **Composição:** usada em `Board-Column` e `Column-Card` para refletir posse forte.

- **Polimorfismo dinâmico:** previsto via ponteiros inteligentes (`std::shared_ptr`) em etapas futuras.
- **Gerenciamento de recursos:** RAII com containers STL (`std::vector`, `std::string`).
- **STL/Templates:** uso de `std::vector`, `std::map`, `std::optional`.
- **Sobrecarga de operadores:** planejado `Card::operator==` para comparação.
- **Exceções:** uso de `std::runtime_error` para operações inválidas.
- **Documentação:** UML em `design/class_diagram.puml` + este relatório técnico.

---

6. Itens Fora de Escopo

- Persistência em disco
- Regras de WIP completas
- Filtros avançados e interface CLI
- Testes unitários elaborados

---

7. Próximos Passos

- Implementar regras de negócio (criação e movimentação de cards, limites de coluna).
- Adicionar testes automatizados.
- Definir formato de persistência (JSON ou SQLite).

---

8. Conclusão

A base do projeto foi definida com foco em **clareza e extensibilidade**.
O diagrama UML e os cabeçalhos estabelecem um contrato inicial que guiará a implementação incremental nas próximas etapas.
