Relatório — Etapa 1 (Design)

Projeto: Kanban Lite
Autor: — Vinícius Ferraz do Nascimento
Matrícula - 20240010760

Objetivo da etapa

Definir o modelo de domínio e as interfaces principais em C++17, com cabeçalhos compiláveis, diagrama UML exportado (PNG/SVG) e justificativa das decisões de design que serão implementadas nas próximas etapas.

Visão geral do modelo

O sistema modela um quadro Kanban com entidades: Board, Column, Card, User e o componente de log ActivityLog. Board agrega e orquestra Columns e Cards; Columns mantêm a ordenação dos cartões por seus ids; Cards encapsulam metadados (título, descrição, vencimento, tags, prioridade e assignee opcional).

Interfaces especializadas promovem baixo acoplamento e extensibilidade: ICardFilter (com TagFilter e AssigneeFilter), ISerializer (com JsonSerializer) e IView (base para CLI/GUI). O diagrama de classes acompanha este relatório em design/docs/uml-classes.png|svg.

Decisões de arquitetura

- Composição sobre herança: Board contém vetores de Column, Card e User, refletindo posse e ciclo de vida controlado; Column mantém apenas Card::Id para evitar dependência forte de Card dentro de Column.
- Polimorfismo por interface: filtros de cartão implementam ICardFilter, permitindo novas estratégias sem alterar Board::filterCards; persistência define contrato via ISerializer com implementação JsonSerializer planejada; apresentação define IView para futuras camadas CLI/GUI.
- Baixo acoplamento e coesão: assignee é modelado como userId opcional no Card; ActivityLog é usado pelo Board como componente de auditoria, evitando acoplamento a frameworks externos nesta etapa.

- Preparação para qualidade: assinaturas usam std::optional, std::chrono::time_point, containers STL e const-correctness para interfaces de leitura, facilitando testes e imutabilidade de consultas.

Mapeamento de requisitos POO → solução

- Encapsulamento: getters/setters em Card, Column, User e Board; validações (título não vazio, nome de coluna, ids existentes) serão aplicadas na Etapa 2, mantendo a API estável já definida nos headers.
- Composição: Board→Column/Card/User e ActivityLog→ActivityLogEntry; Column referencia Card por id, mantendo a ordenação interna e operações de inserção/remoção/movimento.
- Polimorfismo: ICardFilter com TagFilter e AssigneeFilter; Board::filterCards recebe const ICardFilter& e retorna vector<const Card*> para consultas imutáveis.
- Sobrecarga de operadores: Card::operator< define ordenação canônica; igualdades por id em User e Column; estes operadores estão declarados para possibilitar ordenações e buscas na próxima etapa.
- STL/RAII/Exceções: uso de containers STL e std::optional; ownership planejado para ser explícito no Board; exceções serão lançadas na implementação (Etapa 2) para violações de regra de negócio.

Estrutura do repositório

- design/include/: headers Card.h, Column.h, Board.h, User.h, ActivityLog.h, ICardFilter.h, TagFilter.h, AssigneeFilter.h, ISerializer.h, JsonSerializer.h, IView.h.
- design/docs/: uml-classes.png (exportado via Mermaid Chart) e este relatório relatorio-etapa1.pdf.
- CMakeLists.txt na raiz: alvo “kanban” (INTERFACE) e “headers_check” para validar compilação dos cabeçalhos.

Critérios de aceitação e verificação

- Headers compilam com CMake (alvo headers_check), sem dependências externas de implementação nesta etapa.
- UML condiz com os headers e inclui relações e multiplicidades relevantes; exportado em PNG/SVG conforme exigido.
- Este relatório documenta decisões e mapeia requisitos de POO para o design proposto, servindo de base para a Etapa 2.