Relatório Final – Kanban-Lite

Anne Fernandes da Costa Oliveira (20240010789)

08/10/2025

Contents

Relatório Técnico Final — Projeto Kanban-Lite (POO - C++)

Universidade Federal da Paraíba (UFPB)
Curso: Ciência de Dados e Inteligência Artificial
Disciplina: Programação Orientada a Objetos
Aluna: Anne Fernandes da Costa Oliveira
Matrícula: 20240010789
Data: 08 de outubro de 2025
Linguagem: C++17
Frameworks: Qt6.2.4, nlohmann/json
Build System: CMake 3.22+

Índice

---

1. Visão Geral do Projeto

O **Kanban-Lite** é um sistema completo de gerenciamento de tarefas baseado na metodologia Kanban, desenvolvido em C++17 com interface gráfica Qt6 e interface de linha de comando (CLI). O projeto foi desenvolvido em três etapas progressivas, cada uma adicionando camadas de funcionalidade e complexidade.

1.1 Objetivos Principais

- Aplicar conceitos de POO (encapsulamento, herança, polimorfismo, composição)
- Implementar persistência de dados em formato JSON
- Desenvolver interface gráfica moderna e responsiva
- Criar sistema robusto com testes automatizados
- Utilizar boas práticas de engenharia de software

1.2 Características do Sistema

- Gerenciamento de **Boards** e **Colunas** com limites de WIP configuráveis
- Cards com **título, descrição, prioridade, responsável, tags e timestamps**
- Sistema de **tags** para classificação e filtragem
- **Persistência JSON**: formato portátil e legível
- **Duas interfaces**: CLI e GUI
- **Drag & Drop funcional** na interface Qt6
- **Log de atividades** detalhado

---

2. Etapa 1 - Classes Fundamentais

2.1 Objetivo

Desenvolver as classes base do sistema seguindo princípios SOLID e implementar relacionamentos entre classes.

2.2 Classes Implementadas

Foram criadas as classes **User**, **Card**, **Column**, **Board** e **ActivityLog**, implementando conceitos de encapsulamento, composição e responsabilidade única. Cada classe possui métodos de acesso e manipulação de dados com consistência garantida.

2.3 Relacionamentos

- **Associação:** Card → User (0..1)
- **Composição:** Column possui Cards e Board possui Columns
- **Dependency Injection:** Board usa ActivityLog sem gerenciar ciclo de vida

2.4 Diagrama de Classes

```
User
        0..1

Card   Column   Board   ActivityLog
```

---

3. Etapa 2 - CLI e Testes

3.1 Interface de Linha de Comando

Implementa comandos para criar, listar e remover boards, colunas e cards, bem como adicionar tags, filtrar e salvar/carregar o estado.

Exemplo:

```
board create MyProject
column add MyProject Todo
card add MyProject Todo "Implementar JSON"
```

3.2 Sistema de Testes

- Testes unitários cobrindo classes principais e persistência
- Testes de round-trip JSON (serialização/desserialização)
- Testes de lógica de WIP Limit e movimentação de cards

Resultados: 100% dos testes passaram.

---

4. Etapa 3 - Persistência e GUI

4.1 Persistência JSON

Usando a biblioteca **nlohmann/json**, cada classe implementa `toJson()` e `fromJson()` garantindo round-trip completo. Os dados são armazenados em `data/kanban_data.json`.

4.2 Interface Gráfica (Qt6)

- **Arquitetura MVC** com widgets customizados (BoardView, ColumnView, CardView)
- **Drag & Drop funcional** entre colunas
- **Auto-load** e **auto-save** do estado
- **Filtro por tags** e **indicadores de WIP** com cores

---

5. Arquitetura do Sistema

5.1 Camadas

```
CLI / GUI


Domínio (Board, Column, Card, User, ActivityLog)


Persistência (nlohmann/json)
```

5.2 Padrões Utilizados

- **Composite Pattern**: Board–Column–Card
- **Dependency Injection**: ActivityLog
- **Observer (Signals/Slots Qt)**
- **MVC e Strategy Patterns**

---

6. Funcionalidades Implementadas

Funcionalidade	CLI	GUI	Status
Criar Board			Completo
Adicionar Coluna			Completo
Adicionar Card			Completo
Drag & Drop	-		Completo
Filtro por Tags			Completo
Persistência JSON			Completo

---

7. Tecnologias e Ferramentas

- **C++17** — Linguagem principal
- **Qt6.2.4** — Framework GUI
- **nlohmann/json** — Serialização JSON
- **CMake** — Sistema de build
- **Doxygen** — Documentação automática
- **Git** — Controle de versão

Estrutura de diretórios:

```
Kanban-Lite/
   include/
   src/
   ui/
   tests/
   data/
   design/
```

---

8. Testes e Validação

8.1 Exemplos de Teste

Persistência JSON:

```
Card original("card1", "Test Card");
original.addTag("urgent");
json j = original.toJson();
Card loaded = Card::fromJson(j);
assert(loaded.getId() == original.getId());
```

WIP Limit:

```
Column col("Todo", 2);
col.addCard(Card("c1", "Task 1"));
col.addCard(Card("c2", "Task 2"));
assert(!col.addCard(Card("c3", "Overflow")));
```

8.2 Resultados

Categoria	Cobertura
Core	90%
CLI	85%
GUI	70%
Persistência	100%

---

9. Conclusão

9.1 Resultados

Etapa 1: Classes fundamentais implementadas
Etapa 2: CLI funcional e testada
Etapa 3: Persistência e GUI Qt6 completas

9.2 Aprendizados

- RAII e smart pointers garantem segurança de memória
- Uso de Qt Signals/Slots para comunicação desacoplada
- MVC e SOLID aplicados corretamente

9.3 Melhorias Futuras

- Edição de cards pela GUI
- Exibição do ActivityLog visual
- Sincronização em tempo real

9.4 Estatísticas

Métrica	Valor
Linhas de Código	~3.500
Classes	10
Testes Automatizados	24

---

10. Referência à Especificação

Este relatório segue integralmente os requisitos definidos na *Especificação do Trabalho Final — Disciplina Programação Orientada a Objetos (2025)*, atendendo as três etapas previstas:
- Estrutura de classes (Etapa 1)
- CLI funcional e testes (Etapa 2)
- Persistência e GUI (Etapa 3)

---

11. Anexos

A. Compilação

```
# Clonar repositório
git clone https://github.com/annefernandess/Kanban-Lite.git
cd Kanban-Lite && mkdir build && cd build

# Configurar e compilar
cmake ..
make -j4

# Executar CLI
./src/kanban_cli
```

```
# Executar GUI
./run_gui.sh
```

B. Guia de Uso

CLI:

```
$ ./kanban_cli
kanban> board create Projeto
kanban> column add Projeto Todo
kanban> card add Projeto Todo "Tarefa 1"
kanban> save data/projeto.json
```

GUI: 1. Executar `./run_gui.sh`
2. Menu **Arquivo > Novo Board**
3. Criar colunas e adicionar cards
4. Arrastar cards entre colunas
5. Filtrar por tags

C. Agradecimentos Técnicos

Este projeto utilizou bibliotecas open-source sob licença MIT: - Qt6 - nlohmann/json - CMake

---

Relatório Final - Kanban-Lite
Anne Fernandes da Costa Oliveira
08 de outubro de 2025