Relatório Etapa 3 – Kanban-Lite

Anne Fernandes da Costa Oliveira (20240010789)

08/10/2025

Contents

Relatório Técnico – Etapa 3

Projeto: Kanban-Lite (Trabalho Final POO – C++)
Aluna: Anne Fernandes da Costa Oliveira
Matrícula: 20240010789
Data: 08/10/2025

---

1. Objetivo da Etapa

A Etapa 3 teve como objetivo estender o sistema Kanban-Lite com **persistência de dados em JSON** e **interface gráfica utilizando o framework Qt6**, consolidando a aplicação prática de conceitos avançados de **Programação Orientada a Objetos em C++**.

As principais metas foram: - Implementar **serialização e desserialização completas** de todos os objetos de domínio.
- Integrar **persistência unificada** entre CLI e GUI.
- Desenvolver uma **interface gráfica funcional** com suporte a drag & drop.
- Garantir **testes automatizados e CI/CD** para validação contínua.

---

2. Escopo Implementado

- **Persistência JSON:**
  - Biblioteca *nlohmann/json* incorporada em `include/external/json.hpp`.
  - Métodos `toJson()` e `fromJson()` implementados para *User*, *Card*, *Column*, *Board* e *ActivityLog*.
  - Estrutura hierárquica serializada com timestamps em milissegundos.
  - Persistência centralizada no arquivo `data/kanban_data.json`.
- **Interface Gráfica (Qt6):**
  - GUI construída com *Qt Widgets* (versão 6.2.4).
  - Arquitetura em camadas: *MainWindow* → *BoardView* → *ColumnView* → *CardView*.
  - Suporte completo a **drag & drop** entre colunas.
  - Layout responsivo e indicadores de limite WIP por cores.
  - Persistência integrada e compatível com CLI.
- **Testes e CI/CD:**
  - 24 testes automatizados cobrindo serialização, desserialização e integração.
  - Adição de *persistence_tests* ao *CTest* e *GitHub Actions*.
  - 100% dos testes executados com sucesso.

---

3. Conceitos de POO Aplicados

- **Encapsulamento:**
  Atributos privados com acesso controlado via *getters* e *setters*.
- **Composição:**
  Board contém *Columns*; *Columns* contêm *Cards*.
- **Polimorfismo Estático:**
  Métodos estáticos `fromJson()` usados para reconstrução de objetos, garantindo *type safety*.
- **RAII e Smart Pointers:**
  Uso de `std::unique_ptr` para gerenciamento automático de memória e ownership explícito.
- **Tratamento de Exceções:**

Validação robusta em camadas, garantindo segurança e recuperação em casos de erro.

- **Padrões de Projeto:**
  Aplicação de *Factory Method* para desserialização e *Observer pattern* na GUI.

---

4. Testes e Qualidade

- **Ferramentas:** *CMake*, *CTest* e *GitHub Actions*.
- **Suite de Testes:**
  - *User*, *Card*, *Column*, *Board* e *ActivityLog* testados individualmente.
  - Testes de integração CLI–GUI com persistência compartilhada.
- **Resultados:**
  - 24 testes executados, 0 falhas.
  - Compilação e execução automatizadas no pipeline CI/CD.

---

5. Interface Gráfica (Qt6)

A GUI foi implementada em **C++ com Qt6**, oferecendo visualização interativa dos quadros Kanban.

Principais características: - *MainWindow*: menus de navegação, carregamento e salvamento.
- *BoardView*: exibe múltiplos boards com scroll horizontal.
- *ColumnView*: suporte a drop zones e limites de WIP.
- *CardView*: cartões arrastáveis com feedback visual.
- Tela inicial com botão "Criar Primeiro Board".

Integração com CLI: - Persistência compatível via `kanban_data.json`.
- Arquivos criados no modo CLI podem ser abertos e modificados na GUI e vice-versa.

---

6. Conclusão

A Etapa 3 consolidou o projeto Kanban-Lite como uma aplicação completa, unindo persistência e interface gráfica funcional.
O sistema agora conta com: - **Persistência JSON robusta e validada**.

- **Interface Qt6 com drag & drop e layout responsivo**.
- **Cobertura total de testes e integração contínua.**

A arquitetura permanece extensível, permitindo evoluções futuras — como autenticação e colaboração em rede — sem comprometer o design modular estabelecido.

Arquivos principais modificados: - `src/main.cpp`, `tests/persistence_tests.cpp`
- `include/*.h`, `src/*.cpp` (serialização JSON)
- `ui/*.h`, `ui/*.cpp` (componentes Qt6)
- `CMakeLists.txt` e `.github/workflows/ci.yml`

Compilação:
“‘bash cmake -S . -B build && cd build && make -j4

Execução: ./build/kanban_cli # Modo CLI
./build/ui/kanban_gui # Modo GUI

Testes: ctest –output-on-failure