Kanban-Lite

Generated by Doxygen 1.9.1

Chapter 1

Todo List

Member Board::addCard (const std::string &columnName, const Card &card)

Implementar delegação para Column::addCard()

Member Board::addColumn (const Column &column)

Implementar verificação de unicidade de nome

Member Board::moveCard (const std::string &cardId, const std::string &fromCol, const std::string &toCol)

Implementar verificação de regras de WIP

Member Board::removeColumn (const std::string &name)

Implementar validação de existência antes da remoção

Member Card::setPriority (int p)

Implementar validação de faixa (ex: 0..5)

Member Card::setTitle (const std::string &t)

Implementar validação para título não vazio

Member Column::addCard (const Card &card)

Implementar verificação de limite WIP

Member Column::isFull () const

Implementar lógica quando WIP limits estiverem ativos

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 ActivityEntry Struct Reference

Representa uma entrada individual no log de atividades.

`#include <ActivityLog.h>`

Public Attributes

- std::chrono::system_clock::time_point timestamp

  Momento exato do evento.
- std::string message

  Descrição do evento ocorrido.

4.1.1 Detailed Description

Representa uma entrada individual no log de atividades.

Estrutura simples que armazena um evento com timestamp preciso. Utiliza std::chrono para garantir precisão temporal e permitir operações matemáticas com datas.

The documentation for this struct was generated from the following file:

- include/ActivityLog.h

4.2 ActivityLog Class Reference

Sistema centralizado de auditoria para eventos do sistema Kanban.

`#include <ActivityLog.h>`

Public Member Functions

- void record (const std::string &event)

  Registra um novo evento no log com timestamp automático.
- std::vector< ActivityEntry > all () const

  Retorna cópia de todas as entradas do log.

4.2.1 Detailed Description

Sistema centralizado de auditoria para eventos do sistema Kanban.

Responsável por registrar e armazenar eventos do sistema para fins de auditoria, debug e rastreabilidade. Implementa padrão de logging simples com timestamps automáticos e armazenamento em memória.

- Responsabilidade única: logging e auditoria de eventos
- Thread-safety: não implementada (monothreaded na Etapa 1)
- Persistência: planejada para etapas futuras
- Armazenamento: em memória (std::vector)

Conceitos POO aplicados:

- Encapsulamento: dados privados com interface controlada
- Abstração: interface simples record() e all()
- Coesão: foca apenas em logging

4.2.2 Member Function Documentation

4.2.2.1 all()

```
std::vector<ActivityEntry> ActivityLog::all ( ) const
```

Retorna cópia de todas as entradas do log.

Fornece acesso completo ao histórico de eventos registrados. Retorna cópia para evitar modificações acidentais e garantir thread-safety básica.

Returns

std::vector<ActivityEntry> Cópia completa do histórico

Postcondition

Estado do objeto não é alterado (método const)

Note

Retorna cópia (não referência) para simplicidade na Etapa 1

Para listas grandes, considerar iteradores ou paginação em versões futuras

4.2.2.2 record()

```
void ActivityLog::record (
            const std::string & event )
```

Registra um novo evento no log com timestamp automático.

Cria uma entrada de log com o evento especificado e timestamp do momento atual. O timestamp é gerado automaticamente usando std::chrono::system_clock::now().

Parameters

event	Descrição do evento a ser registrado

Precondition

event não deve ser vazio (recomendação)

Postcondition

Nova entrada adicionada ao final do vetor de entries

Timestamp definido como momento atual da chamada

Note

Thread-safety não garantida na versão atual

The documentation for this class was generated from the following file:

- include/ActivityLog.h

4.3 Board Class Reference

Representa um quadro Kanban que agrega e gerencia colunas.

```
#include <Board.h>
```

Public Member Functions

- Board (std::string id, std::string name)
  Construtor da classe Board.
- bool addColumn (const Column &column)
  Adiciona uma nova coluna ao board.
- bool removeColumn (const std::string &name)
  Remove coluna do board pelo nome.
- Column * findColumn (const std::string &name)
  Busca coluna pelo nome (versão não-const).
- const Column * findColumn (const std::string &name) const

Busca coluna pelo nome (versão const).

- bool addCard (const std::string &columnName, const Card &card)

  Adiciona card a uma coluna específica.
- bool moveCard (const std::string &cardId, const std::string &fromCol, const std::string &toCol)

  Move card entre colunas.
- void attachActivityLog (ActivityLog *log)

  Injeta dependência do sistema de log.
- const std::string & **getId** () const
- const std::string & **getName** () const
- const std::vector< Column > & **getColumns** () const

4.3.1 Detailed Description

Representa um quadro Kanban que agrega e gerencia colunas.

A classe Board é o agregador principal do sistema, responsável por coordenar operações de alto nível sobre colunas e cards. Implementa padrão de composição para gerenciar colunas e delegação para operações específicas de cards.

- Responsabilidade única: agregar colunas e coordenar operações de alto nível
- Encapsulamento: atributos privados com acesso controlado via métodos
- Coesão: foca apenas em gerenciar colunas e delegar operações de cartão
- Baixo acoplamento: interage com Column através de interface pública
- Composição: Board possui e gerencia o ciclo de vida das Columns
- Dependency Injection: ActivityLog injetado externamente

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Board()

```
Board::Board (
            std::string id,
            std::string name )
```

Construtor da classe Board.

Cria uma nova instância de Board com identificador e nome obrigatórios. O board é inicializado sem colunas e sem log de atividades.

Parameters

id	Identificador único do board
name	Nome descritivo do board

Precondition

id não deve ser string vazia

name não deve ser string vazia

Postcondition

Board criado sem colunas e sem ActivityLog

4.3.3 Member Function Documentation

4.3.3.1 addCard()

```
bool Board::addCard (
            const std::string & columnName,
            const Card & card )
```

Adiciona card a uma coluna específica.

Operação de conveniência que localiza a coluna e delega a operação de adição do card.

Parameters

columnName	Nome da coluna de destino
card	Card a ser adicionado

Returns

true se adicionado com sucesso, false se coluna não existe

Todo Implementar delegação para Column::addCard()

4.3.3.2 addColumn()

```
bool Board::addColumn (
            const Column & column )
```

Adiciona uma nova coluna ao board.

Insere uma coluna no board garantindo unicidade de nome. A coluna é copiada para o container interno (composição).

Parameters

column	Coluna a ser adicionada

Returns

true se adicionada com sucesso, false se nome já existe

Postcondition

Se sucesso, coluna adicionada ao final do vetor de colunas

Todo Implementar verificação de unicidade de nome

4.3.3.3 attachActivityLog()

```
void Board::attachActivityLog (
            ActivityLog * log )
```

Injeta dependência do sistema de log.

Implementa padrão Dependency Injection para o sistema de auditoria. Board não gerencia o ciclo de vida do ActivityLog.

Parameters

log	Ponteiro para o sistema de log (pode ser nullptr)

Postcondition

ActivityLog configurado para uso (se não nullptr)

Note

Board não possui o ActivityLog - gerenciamento externo

4.3.3.4 findColumn() [1/2]

```
Column* Board::findColumn (
            const std::string & name )
```

Busca coluna pelo nome (versão não-const).

Localiza coluna por nome permitindo modificação.

Parameters

name	Nome da coluna procurada

Returns

Ponteiro para a coluna encontrada ou nullptr se não existir

Note

Permite modificação da coluna retornada

4.3.3.5 findColumn() [2/2]

```
const Column* Board::findColumn (
            const std::string & name ) const
```

Busca coluna pelo nome (versão const).

Localiza coluna por nome para acesso somente leitura.

Parameters

name	Nome da coluna procurada

Returns

Ponteiro const para a coluna encontrada ou nullptr se não existir

Note

Acesso somente leitura à coluna retornada

4.3.3.6 moveCard()

```
bool Board::moveCard (
            const std::string & cardId,
            const std::string & fromCol,
            const std::string & toCol )
```

Move card entre colunas.

Remove card da coluna origem e adiciona na coluna destino. Verifica regras de WIP (Work In Progress) antes da movimentação.

Parameters

cardId	Identificador do card a ser movido
fromCol	Nome da coluna origem
toCol	Nome da coluna destino

Returns

true se movido com sucesso, false caso contrário

Todo Implementar verificação de regras de WIP

4.3.3.7 removeColumn()

```
bool Board::removeColumn (
            const std::string & name )
```

Remove coluna do board pelo nome.

Localiza e remove a coluna especificada. Todos os cards contidos na coluna são perdidos (comportamento de composição).

Parameters

name	Nome da coluna a ser removida

Returns

true se removida com sucesso, false se não encontrada

Todo Implementar validação de existência antes da remoção

The documentation for this class was generated from the following file:

- include/Board.h

4.4 Card Class Reference

Representa uma tarefa/unidade de trabalho no sistema Kanban.

```
#include <Card.h>
```

Public Member Functions

- Card (std::string id, std::string title)

  Construtor da classe Card.
- void setTitle (const std::string &t)

  Define o título do card.
- void setDescription (const std::string &d)

  Define a descrição detalhada do card.
- void setAssignee (User ∗u)

Atribui um responsável ao card.

- void setPriority (int p)

  Define a prioridade numérica do card.
- bool operator== (const Card &other) const

  Operador de igualdade para comparação de cards.
- const std::string & **getId** () const
- const std::string & **getTitle** () const
- const std::string & **getDescription** () const
- User * **getAssignee** () const
- int **getPriority** () const
- const std::chrono::system_clock::time_point & **getCreatedAt** () const
- const std::chrono::system_clock::time_point & **getUpdatedAt** () const

4.4.1 Detailed Description

Representa uma tarefa/unidade de trabalho no sistema Kanban.

A classe Card encapsula todas as informações relacionadas a uma tarefa, incluindo identificação, conteúdo, responsável, prioridade e timestamps de auditoria. Implementa conceitos de POO como encapsulamento através de métodos setters com validação futura.

- Responsabilidade única: gerenciar dados e comportamento de uma tarefa
- Encapsulamento: atributos privados com acesso controlado via métodos
- Associação: referencia User sem gerenciar seu ciclo de vida
- Timestamps automáticos para auditoria (criação/modificação)

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Card()

```
Card::Card (
            std::string id,
            std::string title )
```

Construtor da classe Card.

Cria uma nova instância de Card com ID e título obrigatórios. Os timestamps de criação e atualização são definidos automaticamente para o momento atual.

Parameters

id	Identificador único do card (não pode ser vazio)
title	Título da tarefa (não pode ser vazio)

Precondition

id não deve ser string vazia

title não deve ser string vazia

Postcondition

Card criado com priority = 0, assignee = nullptr

createdAt e updatedAt definidos para momento atual

4.4.3 Member Function Documentation

4.4.3.1 operator==()

```
bool Card::operator== (
            const Card & other ) const
```

Operador de igualdade para comparação de cards.

Compara dois cards baseado em seus IDs únicos. Permite uso em containers STL e algoritmos.

Parameters

other	Card a ser comparado

Returns

true se os IDs são iguais, false caso contrário

Note

Comparação baseada apenas no ID (chave primária)

Método const - não modifica o objeto

4.4.3.2 setAssignee()

```
void Card::setAssignee (
            User * u )
```

Atribui um responsável ao card.

Define o usuário responsável pela execução da tarefa. O Card não gerencia o ciclo de vida do User (associação fraca).

Parameters

u	Ponteiro para o User responsável (pode ser nullptr)

Postcondition

assignee definido como u

updatedAt atualizado para momento atual

Note

Card não possui o User - gerenciamento de memória externo

4.4.3.3 setDescription()

```
void Card::setDescription (
            const std::string & d )
```

Define a descrição detalhada do card.

Atualiza a descrição da tarefa. Descrição pode ser vazia.

Parameters

d	Nova descrição (pode ser vazia)

Postcondition

updatedAt atualizado para momento atual

4.4.3.4 setPriority()

```
void Card::setPriority (
            int p )
```

Define a prioridade numérica do card.

Estabelece a prioridade da tarefa usando escala numérica. Validação de faixa será implementada (ex: 0-5).

Parameters

p	Valor da prioridade

Precondition

p deve estar em faixa válida (validação futura: 0-5)

Postcondition

priority definido como p

updatedAt atualizado para momento atual

Todo Implementar validação de faixa (ex: 0..5)

4.4.3.5 setTitle()

```
void Card::setTitle (
            const std::string & t )
```

Define o título do card.

Atualiza o título da tarefa e o timestamp de modificação. Validação será implementada para garantir título não vazio.

Parameters

t	Novo título do card

Precondition

t não deve ser string vazia (validação futura)

Postcondition

updatedAt atualizado para momento atual

Todo Implementar validação para título não vazio

The documentation for this class was generated from the following file:

- include/Card.h

4.5 Column Class Reference

Representa uma coluna do quadro Kanban que organiza cards.

```
#include <Column.h>
```

Public Member Functions

- Column (std::string name, int wipLimit=-1)
  Construtor da classe Column.
- bool addCard (const Card &card)
  Adiciona card à coluna.
- bool removeCard (const std::string &cardId)
  Remove card da coluna pelo ID.
- Card * findCard (const std::string &cardId)
  Busca card pelo ID (versão não-const).
- const Card * findCard (const std::string &cardId) const
  Busca card pelo ID (versão const).
- bool isFull () const
  Verifica se coluna atingiu limite WIP.
- const std::vector< Card > & getCards () const
  Obtém referência const para vetor de cards.
- const std::string & getName () const
  Obtém nome da coluna.
- int getWipLimit () const
  Obtém limite WIP configurado.
- size_t getCardCount () const
  Obtém quantidade atual de cards.
- bool operator== (const Column &other) const
  Operador de igualdade para comparação de colunas.

4.5.1 Detailed Description

Representa uma coluna do quadro Kanban que organiza cards.

A classe Column mantém e organiza um conjunto de cards, implementando conceitos de Work In Progress (WIP) limits para controle de fluxo. Utiliza composição para gerenciar o ciclo de vida dos cards contidos.

- Responsabilidade única: manter e organizar conjunto de cards
- Composição: Column possui e gerencia cards
- WIP Limits: controle de limite de trabalho em progresso
- Encapsulamento: acesso controlado aos cards via métodos públicos

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Column()

```
Column::Column (
            std::string name,
            int wipLimit = -1 )
```

Construtor da classe Column.

Cria uma nova coluna com nome obrigatório e limite WIP opcional. Valor -1 para wipLimit indica ausência de limite.

Parameters

name	Nome identificador da coluna
wipLimit	Limite máximo de cards (-1 = sem limite)

Precondition

name não deve ser string vazia

Postcondition

Column criada vazia (sem cards)

4.5.3 Member Function Documentation

4.5.3.1 addCard()

```
bool Column::addCard (
            const Card & card )
```

Adiciona card à coluna.

Insere novo card na coluna respeitando limite WIP quando ativo. O card é copiado para o container interno (composição).

Parameters

card	Card a ser adicionado

Returns

true se adicionado com sucesso, false se limite WIP atingido

Postcondition

Se sucesso, card adicionado ao final do vetor

Todo Implementar verificação de limite WIP

4.5.3.2 findCard() [1/2]

```
Card* Column::findCard (
            const std::string & cardId )
```

Busca card pelo ID (versão não-const).

Localiza card por identificador permitindo modificação.

Parameters

cardId	Identificador do card procurado

Returns

Ponteiro para o card encontrado ou nullptr se não existir

Note

Permite modificação do card retornado

4.5.3.3 findCard() [2/2]

```
const Card* Column::findCard (
            const std::string & cardId ) const
```

Busca card pelo ID (versão const).

Localiza card por identificador para acesso somente leitura.

Parameters

cardId	Identificador do card procurado

Returns

Ponteiro const para o card encontrado ou nullptr se não existir

Note

Acesso somente leitura ao card retornado

4.5.3.4 getCardCount()

```
size_t Column::getCardCount ( ) const
```

Obtém quantidade atual de cards.

Returns

Número de cards presentes na coluna

4.5.3.5 getCards()

```
const std::vector<Card>& Column::getCards ( ) const
```

Obtém referência const para vetor de cards.

Fornece acesso somente leitura à coleção interna de cards para iteração e consulta sem permitir modificação.

Returns

Referência const para std::vector<Card> interno

Note

Permite iteração segura sem modificação dos dados

4.5.3.6 getName()

```
const std::string& Column::getName ( ) const
```

Obtém nome da coluna.

Returns

Nome identificador da coluna

4.5.3.7 getWipLimit()

```
int Column::getWipLimit ( ) const
```

Obtém limite WIP configurado.

Returns

Valor do limite WIP (-1 indica sem limite)

4.5.3.8 isFull()

```
bool Column::isFull ( ) const
```

Verifica se coluna atingiu limite WIP.

Determina se a coluna está cheia baseado no limite WIP configurado. Sempre retorna false se wipLimit = -1 (sem limite).

Returns

true se limite atingido, false caso contrário

Note

Sempre false quando wipLimit = -1

Todo Implementar lógica quando WIP limits estiverem ativos

4.5.3.9 operator==()

```
bool Column::operator== (
            const Column & other ) const
```

Operador de igualdade para comparação de colunas.

Compara duas colunas baseado no nome identificador.

Parameters

other	Coluna a ser comparada

Returns

true se nomes forem iguais, false caso contrário

4.5.3.10 removeCard()

```
bool Column::removeCard (
            const std::string & cardId )
```

Remove card da coluna pelo ID.

Localiza e remove o card especificado da coluna.

Parameters

cardId	Identificador único do card a ser removido

Returns

true se removido com sucesso, false se não encontrado

Postcondition

Card removido do container se encontrado

The documentation for this class was generated from the following file:

- include/Column.h

4.6 User Class Reference

Representa um usuário/participante do sistema Kanban.

```
#include <User.h>
```

Public Member Functions

- User (std::string id, std::string name, std::string email)
  Construtor da classe User.
- const std::string & getName () const
  Obtém nome do usuário.
- const std::string & getEmail () const
  Obtém email do usuário.
- const std::string & getId () const
  Obtém ID do usuário.
- bool operator== (const User &other) const
  Operador de igualdade para comparação de usuários.

4.6.1 Detailed Description

Representa um usuário/participante do sistema Kanban.

A classe User encapsula informações de identificação de usuários que podem ser criadores ou responsáveis (assignees) por cards. Implementa conceitos de encapsulamento através de getters const.

- Responsabilidade única: manter dados de identificação do usuário
- Encapsulamento: atributos privados com acesso somente leitura
- Imutabilidade: após criação, dados não podem ser alterados
- Associação: é referenciado por Card sem gerenciar ciclo de vida

4.6.2 Constructor & Destructor Documentation

4.6.2.1 User()

```
User::User (
            std::string id,
            std::string name,
            std::string email )
```

Construtor da classe User.

Cria novo usuário com identificador, nome e email obrigatórios. Todos os parâmetros são copiados e armazenados internamente.

Parameters

id	Identificador único do usuário
name	Nome completo do usuário
email	Endereço de email do usuário

Precondition

id não deve ser string vazia

name não deve ser string vazia

email não deve ser string vazia

Postcondition

User criado com dados imutáveis

4.6.3 Member Function Documentation

4.6.3.1 getEmail()

```
const std::string& User::getEmail ( ) const
```

Obtém email do usuário.

Fornece acesso somente leitura ao email do usuário.

Returns

Referência const para o email

Note

Método const - não modifica o objeto

4.6.3.2 getId()

```
const std::string& User::getId ( ) const
```

Obtém ID do usuário.

Fornece acesso somente leitura ao identificador único.

Returns

Referência const para o ID

Note

Método const - não modifica o objeto

4.6.3.3 getName()

```
const std::string& User::getName ( ) const
```

Obtém nome do usuário.

Fornece acesso somente leitura ao nome do usuário.

Returns

Referência const para o nome

Note

Método const - não modifica o objeto

4.6.3.4 operator==()

```
bool User::operator== (
            const User & other ) const
```

Operador de igualdade para comparação de usuários.

Compara dois usuários baseado em seus IDs únicos.

Parameters

other	Usuário a ser comparado

Returns

true se os IDs são iguais, false caso contrário

Note

Comparação baseada apenas no ID (chave primária)

The documentation for this class was generated from the following file:

- include/User.h

Chapter 5

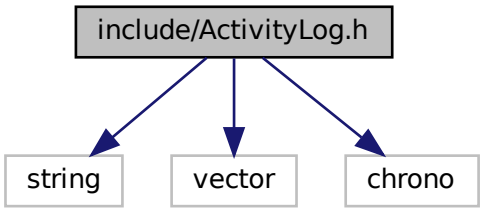
File Documentation

5.1 include/ActivityLog.h File Reference

Sistema de auditoria e logging para operações do Kanban.

```
#include <string>
#include <vector>
#include <chrono>
```

Include dependency graph for ActivityLog.h:



Classes

- struct ActivityEntry

  Representa uma entrada individual no log de atividades.
- class ActivityLog

  Sistema centralizado de auditoria para eventos do sistema Kanban.

5.1.1 Detailed Description

Sistema de auditoria e logging para operações do Kanban.

Author

Anne Fernandes da Costa Oliveira

Date

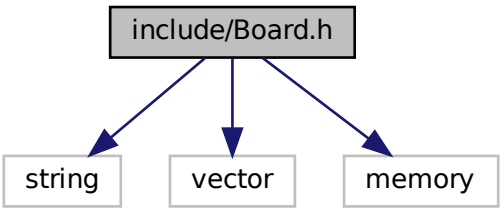
25/09/2025

5.2 include/Board.h File Reference

Definição da classe Board para gerenciamento de quadros Kanban.

```
#include <string>
#include <vector>
#include <memory>
```

Include dependency graph for Board.h:



Classes

- class Board

  Representa um quadro Kanban que agrega e gerencia colunas.

5.2.1 Detailed Description

Definição da classe Board para gerenciamento de quadros Kanban.

Author

Anne Fernandes da Costa Oliveira

Date

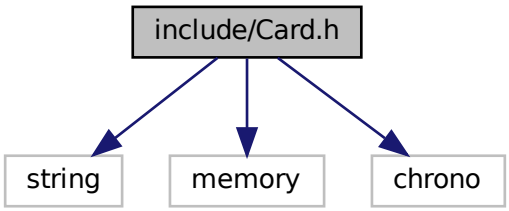
25/09/2025

5.3 include/Card.h File Reference

Definição da classe Card para representar tarefas no sistema Kanban.

```
#include <string>
#include <memory>
```

```
#include <chrono>
```
Include dependency graph for Card.h:



Classes

- class Card

  Representa uma tarefa/unidade de trabalho no sistema Kanban.

5.3.1 Detailed Description

Definição da classe Card para representar tarefas no sistema Kanban.

Author

Anne Fernandes da Costa Oliveira

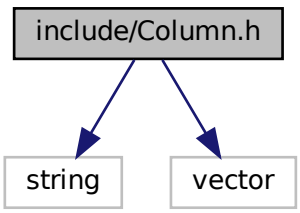
Date

25/09/2025

5.4 include/Column.h File Reference

Definição da classe Column para organização de cards no Kanban.

```
#include <string>
#include <vector>
```
Include dependency graph for Column.h:



Classes

- class Column

  Representa uma coluna do quadro Kanban que organiza cards.

5.4.1 Detailed Description

Definição da classe Column para organização de cards no Kanban.

Author

Anne Fernandes da Costa Oliveira

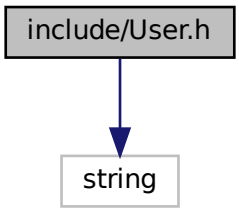
Date

25/09/2025

5.5 include/User.h File Reference

Definição da classe User para representar participantes do sistema.

`#include <string>`
Include dependency graph for User.h:



Classes

- class User

  Representa um usuário/participante do sistema Kanban.

5.5.1 Detailed Description

Definição da classe User para representar participantes do sistema.

Author

Anne Fernandes da Costa Oliveira

Date

25/09/2025

Index