UNIVERSIDADE
FEDERAL DO CARIRI

Arthur Ricardo Macêdo Pereira e Everton Lohan Pereira Ferreira

Programação Orientada à Objetos

Documentação do Projeto - To-Do List

Juazeiro do Norte - CE
2024

SUMÁRIO

Link do repositório : https://github.com/ev3rton0/ToDoList2.0

1. Descrição do Sistema

Com o objetivo de usar os conceitos de POO aprendidos em sala de aula, desenvolvemos um programa de lista de tarefas. O ToDoList é um sistema criado para ajudar na organização de tarefas do dia a dia, facilitando o gerenciamento das atividades e responsabilidades. Com ele, o usuário pode adicionar novas tarefas, editar detalhes, marcar como concluídas, e remover aquelas que já não são mais necessárias. O sistema também permite organizar e visualizar as tarefas por prioridade e data de entrega, o que ajuda o usuário a planejar seu tempo conforme a urgência das atividades.

Com isso, o programa tem dois módulos. O primeiro módulo será acessível somente pelo usuário administrador, no qual poderá criar, editar, remover tarefas, bem como listar todos os tipos de tarefas, os tipos de tarefas contidos no programa são as concluídas e as pendentes e a opção de listar todas as tarefas juntas. Já o segundo módulo será acessível pelo leitor, o qual só poderá listar todos os tipos de tarefas, bem como marcá-las como concluídas. Assim criando um ambiente que será possível compartilhar as tarefas com outras pessoas.

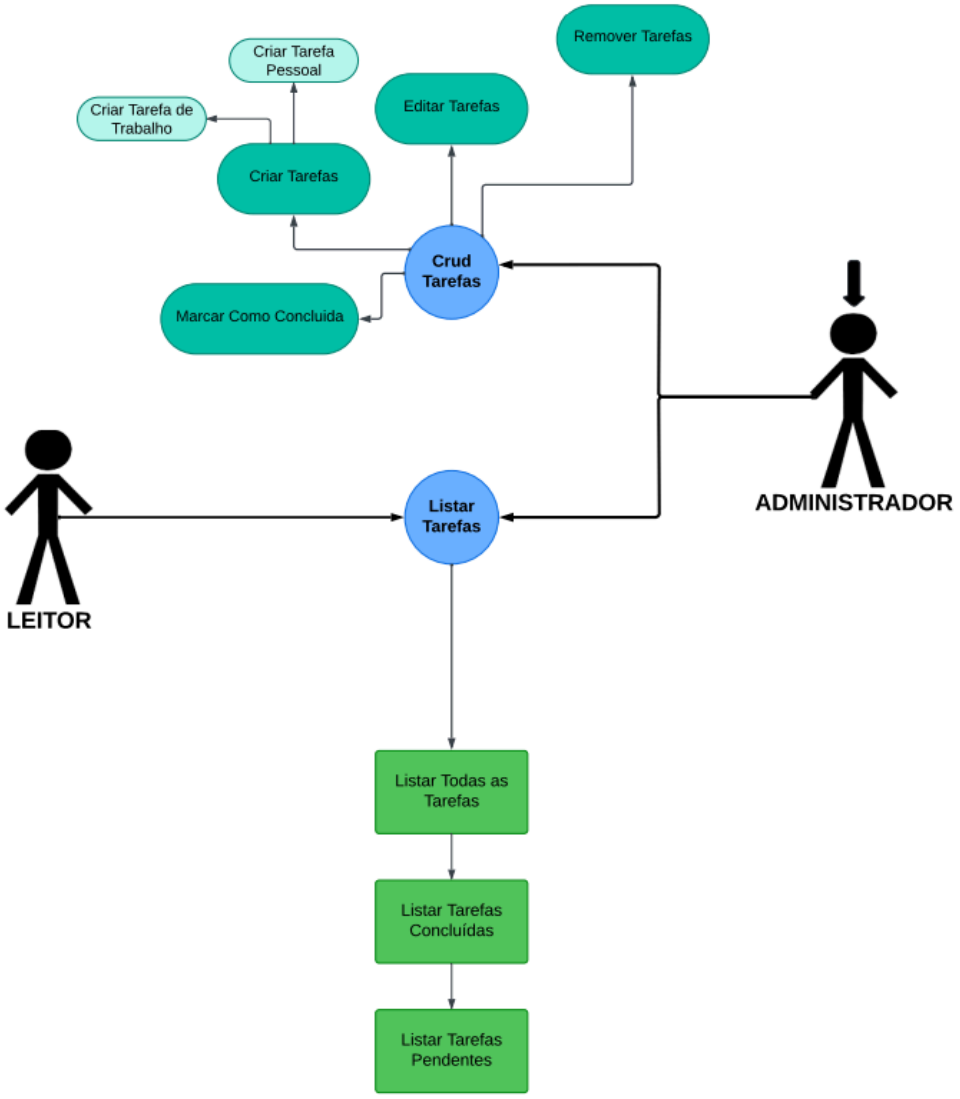
O público-alvo do ToDoList é bem variado, indo desde estudantes que precisam de ajuda para se organizar até profissionais de várias áreas que querem manter suas atividades em dia. Esse sistema é ideal para quem quer ganhar mais controle sobre seu tempo e não esquecer de nada importante. Pensamos numa interface simples e fácil de usar, para que o ToDoList seja acessível a qualquer pessoa, independentemente do nível de experiência com tecnologia. Nosso objetivo é tornar o dia a dia mais produtivo e tranquilo, ajudando o usuário a gerenciar melhor suas tarefas e, com isso, aproveitando melhor o seu tempo.

2. Backlog

FUNCIONALIDADE	RESPONSÁVEL
Tarefa	Arthur
TarefaDeTrabalho	Everton
TarefaPessoal	Arthur
CrudTarefa	Arthur
TipoUsuario	Everton
AsciiArt	Everton
ListarTarefa	Arthur
Main	Everton
PadrãoFachada	Everton,Arthur

3. Arquitetura de Sistema

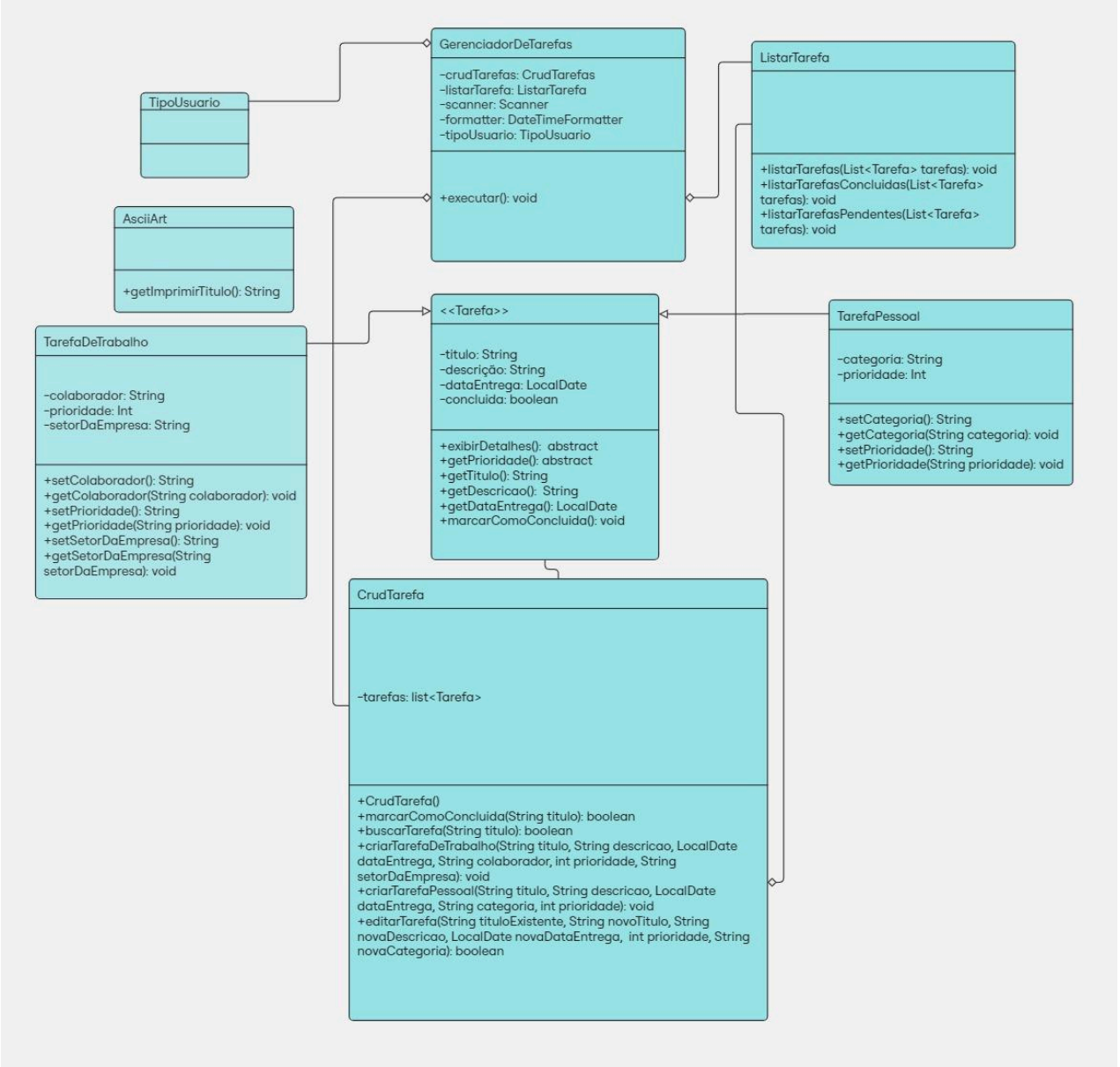
Diagrama de casos de uso:



Funcionalidades:

- Criar Tarefas:
  - Criar Tarefas de Trabalho;
  - Criar Tarefa pessoal;
- Editar Tarefas;
- Remover Tarefas;
- Marcar Tarefas como Concluída;
- Ver Tarefas:
  - Todas as Tarefas;
  - Tarefas Pendentes;
  - Tarefas Concluídas;

Diagrama UML :



1. Classe Abstrata Tarefa :

- **Descrição**: A classe **Tarefa** representa uma tarefa genérica com atributos essenciais como título, descrição, data de entrega e status de conclusão. Ela possui métodos abstratos para exibir detalhes e obter a prioridade, sendo esses específicos para cada tipo de tarefa. Métodos adicionais incluem get e set para os atributos comuns.A classe Tarefa foi definida como abstrata para servir como uma base comum para diferentes tipos de tarefas. Com isso, a especialização e personalização dos métodos **exibirDetalhes** e **getPrioridade** nas subclasses permite que cada tipo de tarefa se adapte a necessidades específicas (como tarefas pessoais e de trabalho), promovendo flexibilidade e reutilização de código.

2. Subclasse **TarefaDeTrabalho :**

- **Descrição**: Esta subclasse representa tarefas relacionadas ao trabalho, incluindo atributos como colaborador, prioridade e setor da empresa, que são específicos para tarefas do ambiente corporativo. Ela herda de **Tarefa** e implementa os métodos abstratos com uma lógica própria para o contexto de trabalho.

3. Subclasse **TarefaPessoal :**

- **Descrição:** A classe **TarefaPessoal** modela tarefas pessoais, contendo atributos como categoria e prioridade, específicos para o contexto pessoal. Ela herda de **Tarefa**, implementando métodos abstratos com a lógica de classificação e prioridade de tarefas pessoais.

4. Classe **CrudTarefa :**

- **Descrição:** A classe **CrudTarefa** representa a camada de lógica de negócios para gerenciamento de tarefas. Ela mantém uma lista de Tarefa (implementada como agregação) e fornece métodos para criar, buscar, editar e marcar tarefas como concluídas.

5. Classe **ListarTarefa :**

- **Descrição:** Esta classe é responsável pela visualização de tarefas, oferecendo métodos para listar todas as tarefas, as concluídas e as pendentes. A criação da classe **ListarTarefa** como uma classe dedicada à exibição permite separar a lógica de apresentação da lógica de dados (presente em **CrudTarefa**).

6. Classe **GerenciadorDeTarefas (Fachada) :**

- Descrição: A classe GerenciadorDeTarefas atua como a fachada do sistema, e fica responsável pela interação entre o usuário e as operações de gerenciamento de tarefas. Ela está associada a **CrudTarefa**, **ListarTarefa**, e **TipoUsuario**, além de gerenciar a entrada do usuário e o formato das datas.

7. Classe **TipoUsuario** (Enumeração) :

- **Descrição:** **TipoUsuario** é uma enumeração que diferencia os tipos de usuários (ADM e Leitor), permitindo controle sobre as permissões de acesso. A enumeração **TipoUsuario** permite um controle simples e eficiente dos tipos de usuários, facilitando a manutenção e compreensão do código.

8. Classe **AsciiArt**

- **Descrição:** A classe AsciiArt é responsável por exibir uma arte ASCII ao iniciar o programa, criando uma interface visual mais interessante.

9. Classe **Main**

- **Descrição:** A classe Main é a entrada do sistema, responsável apenas por instanciar GerenciadorDeTarefas e iniciar o programa.

Este design segue princípios de POO como abstração, encapsulamento, herança e polimorfismo. A abstração em **Tarefa** permite que novos tipos de tarefas sejam facilmente adicionados, enquanto a separação entre **CrudTarefa** e **ListarTarefa** organiza o sistema em camadas de responsabilidade distintas. O padrão de fachada implementado em **GerenciadorDeTarefas** centraliza a execução, simplificando o uso e promovendo escalabilidade. Esse design modular facilita futuras expansões e manutenção, mantendo o código limpo e eficiente.