Refatoração do Código

A refatoração realizada teve como objetivo aprimorar a organização, a legibilidade e a manutenibilidade do código-fonte do sistema. Para isso, seguimos os princípios fundamentais da programação orientada a objetos, buscando uma estrutura clara que facilite futuras modificações e extensões.

Durante esse processo, as responsabilidades foram cuidadosamente distribuídas entre as classes, garantindo que cada uma tivesse uma única responsabilidade, o que melhora a coesão interna e reduz o acoplamento entre os componentes do sistema.

Responsabilidades das Classes

GerenciadorProjetos
- Responsabilidade Principal: Coordenar todas as operações do sistema.
- Detalhes:
- Manter o registro central de projetos, membros e tarefas.
- Garantir a consistência das operações.
- Gerar relatórios consolidados.
- Validar regras de negócio antes de operações.

Projeto
- Responsabilidade Principal: Representar um projeto com seu contexto.
- Detalhes:
- Gerenciar membros participantes.
- Controlar tarefas associadas.
- Calcular prazos e atrasos.
- Manter informações descritivas.

Membro
- Responsabilidade Principal: Representar um membro da equipe.
- Detalhes:
- Armazenar informações pessoais (nome, função, email).
- Gerenciar tarefas atribuídas.
- Fornecer dados para relatórios individuais.

Tarefa
- Responsabilidade Principal: Representar uma unidade de trabalho.
- Detalhes:
- Manter estado (pendente, em andamento, concluída).
- Controlar prazos e prioridades.
- Calcular atrasos.
- Relacionar-se com o membro responsável.

Exceções (GerenciadorProjetosError)
- Responsabilidade Principal: Comunicar erros específicos do domínio.
- Detalhes:

- Prover mensagens claras para usuários.
- Transportar contexto do erro (nomes, valores problemáticos).
- Categorizar tipos de falhas para tratamento adequado.

Princípios Aplicados

1. **Responsabilidade Única**: Cada classe possui uma única razão para mudar, facilitando a manutenção.

2. **Encapsulamento**: Detalhes internos das classes são protegidos para evitar manipulações indevidas.

3. **Baixo Acoplamento**: As classes interagem por meio de interfaces claras, reduzindo dependências diretas.

4. **Alta Coesão**: Operações relacionadas são mantidas juntas dentro das mesmas classes.

5. **Inversão de Dependência**: O sistema depende de abstrações, garantindo maior flexibilidade.

Testes Automatizados

Para garantir que as alterações não comprometessem a funcionalidade do sistema, desenvolvemos uma suíte de testes automatizados utilizando a biblioteca **pytest**. Todos os testes passaram com sucesso, confirmando que o sistema se mantém estável e confiável após a refatoração.

Segue o resultado da execução dos testes:

(venv) juju@juju-Aspire-A515-45:~/EngenhariaSoftware$ python -m pytest testes/ -v ==========
test session starts ==========
platform linux – Python 3.10.12, pytest-8.4.0, pluggy-1.6.0 – /home/juju/EngenhariaSoftware/venv/bin/python
cachedir: .pytest_cache rootdir: /home/juju/EngenhariaSoftware collected 27 items

testes/integracao/test_integracao.py::TestIntegracaoGerenciador::test_adicionar_responsavel_nao_membro
PASSED [3%] testes/integracao/test_integracao.py::TestIntegracaoGerenciador::test_fluxo_completo_projet
PASSED [7%] testes/integracao/test_integracao.py::TestIntegracaoGerenciador::test_tarefa_atrasada
PASSED [11%] testes/test_excecoes.py::TestExcecoes::test_membro_nao_encontrado
PASSED [14%] testes/test_excecoes.py::TestExcecoes::test_projeto_nao_encontrado
PASSED [18%] testes/test_excecoes.py::TestExcecoes::test_responsavel_nao_membro
PASSED [22%] testes/test_excecoes.py::TestExcecoes::test_tarefa_nao_encontrada
PASSED [25%] testes/test_gerenciador.py::TestGerenciadorProjetos::test_adicionar_projeto
PASSED [29%] testes/test_gerenciador.py::TestGerenciadorProjetos::test_cadastrar_membro
PASSED [33%] testes/test_gerenciador.py::TestGerenciadorProjetos::test_concluir_tarefa

```
PASSED [ 37%] testes/test_gerenciador.py::TestGerenciadorProjetos::test_criar_tarefa_com_responsavel_na
PASSED [ 40%] testes/test_gerenciador.py::TestGerenciadorProjetos::test_criar_tarefa_valida
PASSED [ 44%] testes/test_gerenciador.py::TestGerenciadorProjetos::test_relatorios
PASSED [ 48%] testes/test_membro.py::TestMembro::test_adicionar_tarefa
PASSED [ 51%] testes/test_membro.py::TestMembro::test_criacao_membro
PASSED [ 55%] testes/test_membro.py::TestMembro::test_remover_tarefa
PASSED [ 59%] testes/test_membro.py::TestMembro::test_representacao_string
PASSED [ 62%] testes/test_projeto.py::TestProjeto::test_adicionar_membro
PASSED [ 66%] testes/test_projeto.py::TestProjeto::test_adicionar_tarefa
PASSED [ 70%] testes/test_projeto.py::TestProjeto::test_criacao_projeto
PASSED [ 74%] testes/test_projeto.py::TestProjeto::test_relatorio_projeto
PASSED [ 77%] testes/test_projeto.py::TestProjeto::test_representacao_string
PASSED [ 81%] testes/test_tarefa.py::TestTarefa::test_concluir_tarefa
PASSED [ 85%] testes/test_tarefa.py::TestTarefa::test_criacao_tarefa
PASSED [ 88%] testes/test_tarefa.py::TestTarefa::test_iniciar_tarefa PASSED
[ 92%] testes/test_tarefa.py::TestTarefa::test_representacao_string PASSED [
96%] testes/test_tarefa.py::TestTarefa::test_tarefa_atrasada PASSED [100%]

============================================================
27 passed in 0.03s ============================================
```

Diagrama de Classes

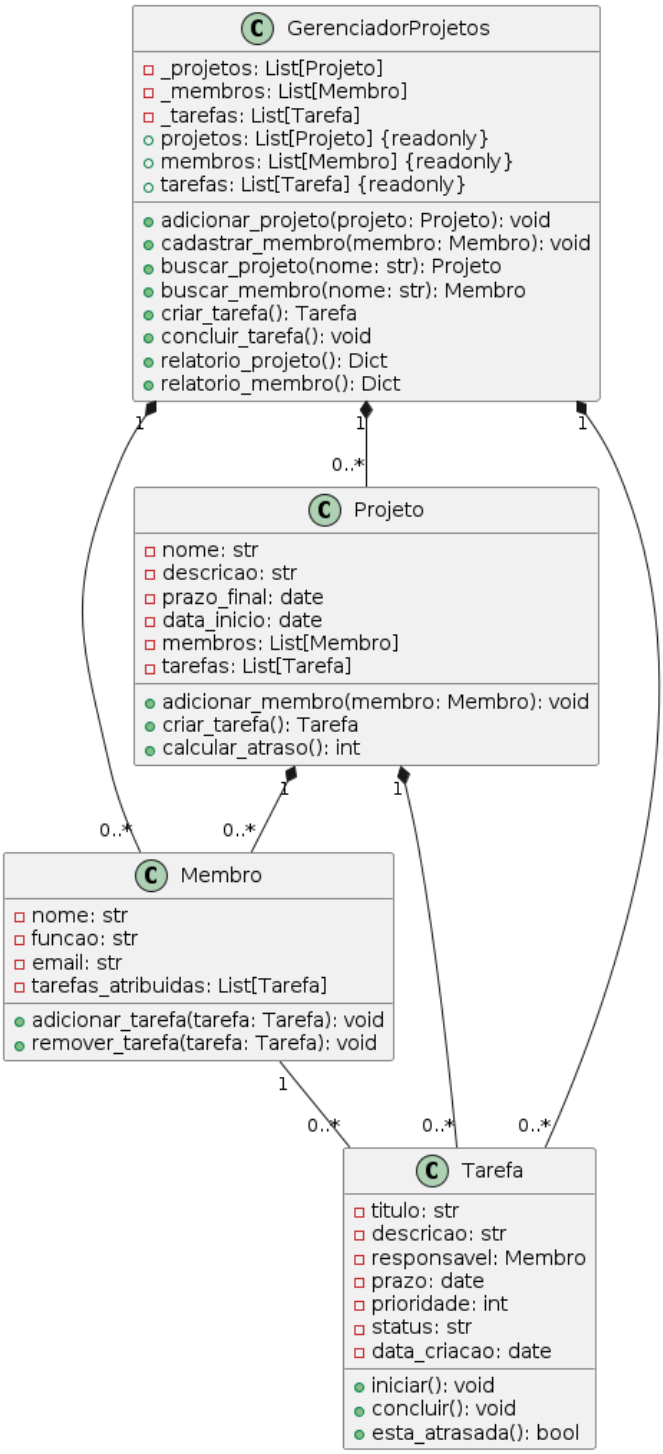

Figure 1: Diagrama de Classes