Relatório Detalhado - Projeto A3: UniversalConverter

1. Introdução

Este relatório documenta o projeto A3 da disciplina Gestão e Qualidade de Software (GQS). O objetivo do trabalho é aplicar princípios de Clean Code, SOLID e padrões de projeto para refatorar um código legado e torná-lo mais legível, testável e mantível.

2. Descrição do Problema

2. Descrição do Problema O código original ("codigo-original/legacy_universal.py") é um monólito procedural que mistura múltiplas responsabilidades: validação, lógica de conversão, interface com o usuário e impressão. Os principais problemas encontrados: - Funções longas e duplicadas. - Ifs aninhados e lógica espalhada. - Falta de abstração e encapsulamento. - Ausência de testes automatizados na versão original. - Alto acoplamento entre IO e lógica.

3. Requisitos

3. Requisitos (funcionais e não funcionais) Funcionais: - Converter entre unidades de temperatura (Celsius, Fahrenheit, Kelvin, Rankine). - Converter entre unidades de comprimento (m, cm, mm, km, in, ft). - Converter entre unidades de massa (kg, g, mg, lb). - Fornecer CLI simples e endpoint HTTP opcional. Não funcionais: - Código legível e modular. - Cobertura de testes automatizados com pytest. - Estrutura de projeto que permita expansão (ex: novos conversores). - Documentação e relatório completos.

4. Decisões de Design

4. Decisões de Design - Padrão Strategy: cada conjunto de unidades tem conversores que implementam métodos to_base/from_base. - Factory: mapeia strings de unidades para classes conversoras. - ConversionService: orquestra a conversão usando conversores e unidades base (Kelvin, Meter, Kilogram). - Separação de camadas: 'codigo-original' e 'codigo-refatorado' para demonstrar antes e depois. - Testes automatizados e CI via GitHub Actions.

5. Arquitetura e Estrutura de Código

5. Arquitetura e Estrutura de Código Estrutura principal do projeto (diretórios): - codigo-original/: código legado (monolítico). - codigo-refatorado/ - universal_converter/ - __init__.py - converters.py - factory.py - service.py - cli.py - webapp.py (opcional) - tests/ - Relatorio_A3_detalhado.pdf - README.md - .github/workflows/python-package.yml

6. Diagrama de Classes (texto)

6. Diagrama de Classes (simplificado) Converter (ABC) ■■ TempConverterBase ■ ■■ CelsiusConverter ■ ■■ FahrenheitConverter ■ ■■ KelvinConverter ■ ■■ RankineConverter ■■ LengthConverterBase ■ ■■ MeterConverter ■ ■■ CentimeterConverter ■ ■■ ... ■■ WeightConverterBase ■■ KilogramConverter ■■ ... ConverterFactory -> retorna instância de Converter ConversionService -> usa duas instâncias Converter (from/to) -> realiza conversão via base

7. Testes e Integração Contínua

7. Testes e Integração Contínua - Testes implementados com pytest (arquivo: codigo-refatorado/tests/test_full_conversions.py) - Casos de teste cobrem: conversões de temperatura, comprimento e massa, inputs inválidos e não numéricos. - Workflow do GitHub Actions roda pytest em push/PR para branch main.

8. Execução, Verificação e Critérios de Aceitação

8. Execução, Verificação e Critérios de Aceitação Passos para reproduzir (local): 1) Instalar dependências: pip install -r codigo-refatorado/requirements.txt 2) Rodar testes: pytest -q codigo-refatorado/tests 3) Executar CLI: python codigo-refatorado/run_converter.py 100 c f Critérios de aceitação: - Todos os testes passam. - Conversões conhecidas (ex: 0°C -> 32°F) resultam nos valores corretos. - Código organizado e documentado.

9. Divisão de Trabalho e Cronograma

9. Divisão de Trabalho e Cronograma de Commits - Até 30/09: subir codigo-original (commit inicial) - 01/10 a 27/11: desenvolver refatoração com commits significativos Sugestão de commits (cada aluno ao menos 1 commit): - "chore: add legacy code" - "feat: add converters and conversion service" - "refactor: apply factory and strategy" - "test: add pytest cases" - "ci: add github actions" - "docs: add detailed report"

10. Apêndice - Trechos de Código

10. Apêndice - Trechos de Código Relevantes - Conversão padrão no service: base = from_conv.to_base(value) result = to_conv.from_base(base) - Exemplo de uso CLI: python run_converter.py 100 c f

11. Notas para Apresentação e Atribuição de Papéis

Sugestão de roteiro (cada membro: 1 min): - Membro 1 (Introdução): Contexto do problema e motivação. - Membro 2 (Código original): Demonstração do código legado e suas falhas. - Membro 3 (Refatoração): Explicar arquitetura, padrões utilizados e mostrar código refatorado. - Membro 4 (Testes & Conclusão): Exibir testes, evidências de execução e lições aprendidas.