**✅ Documento Completo Atualizado: "Como Descrever uma Sprint v2.0"**

[**Visualizar documento completo v2.0**](computer://mnt/user-data/outputs/como_descrever_sprint_v2_completo.md)

Criei uma versão MUITO mais completa e abrangente do documento, que agora inclui TODOS os elementos essenciais que você mencionou:

**📋 Novos Elementos Adicionados para Cada HU:**

**1. Estrutura Expandida de Identificação:**

* Tipo de história (Feature/Technical/Bug/Debt)
* Datas de início e fim
* Responsável pela história
* Status detalhado com percentual

**2. Requisitos Completos:**

* **Requisitos Funcionais (RF)** - Detalhamento técnico completo
* **Requisitos Não Funcionais (RNF)** - Performance, Segurança, Confiabilidade, Usabilidade

**3. Casos de Teste Expandidos:**

yaml

- Pré-condições

- Cenários positivos e negativos

- Casos de borda

- Dados de entrada detalhados

- Validações específicas

- Pós-condições

**4. Matriz de Testes & Golden Files Completa:**

| **ID** | **Caso** | **Entrada** | **Esperado** | **Golden File** | **Status** |
| --- | --- | --- | --- | --- | --- |
| CT-HX.1 | Descrição | Input específico | Output esperado | FILE\_001.json | ✅/🔴 |

**5. Relacionamentos Detalhados:**

markdown

### Depende de:

- Histórias predecessoras

- Bibliotecas e versões

- Serviços externos

### Bloqueia:

- Histórias dependentes

- Funcionalidades futuras

### Relacionado a:

- Issues do GitHub/Jira

- ADRs (Architecture Decision Records)

- RFCs (Request for Comments)

- Bugs relacionados

**6. Implementação Técnica Completa:**

* Stack tecnológico
* Arquitetura do componente
* Estrutura de arquivos
* Código de referência
* Contratos de API
* Configurações necessárias

**7. Observabilidade & Auditoria:**

* Estrutura de logs esperados
* Métricas a serem coletadas
* Eventos de auditoria
* Alertas configurados
* Dashboards necessários

**8. Definition of Ready (DoR) Expandido:**

* User Story refinada
* Critérios GWT claros
* Casos de teste definidos
* Golden Files identificados
* Dependências mapeadas
* API contracts definidos
* Estimativas revisadas
* Riscos identificados
* Protótipos aprovados
* ADRs criados

**9. Definition of Done (DoD) Completo:**

* **Código:** Review, complexidade, clean code
* **Testes:** Unit (>80%), Integration, Property, E2E, Golden Files
* **Documentação:** JSDoc, README, OpenAPI, Changelog
* **Qualidade:** Lint, TypeCheck, Build, Performance
* **Segurança:** SAST, Dependencies, Secrets, OWASP
* **Acessibilidade:** WCAG AA, Keyboard, Screen reader
* **Deploy:** Staging, Smoke tests, Rollback, Monitoring

**10. Gestão de Riscos Detalhada:**

Para cada risco:

* Probabilidade (Alta/Média/Baixa)
* Impacto (Crítico/Alto/Médio/Baixo)
* Detecção (como identificar)
* Mitigação (prevenção)
* Contingência (plano B)
* Responsável (quem monitora)

**11. Rastreabilidade Completa:**

* Documentos relacionados
* Commits e PRs
* Issues relacionadas
* Métricas de sucesso
* Feedback e iterações
* Lições aprendidas

**📊 Exemplos Completos Implementados:**

No documento, incluí **3 histórias TOTALMENTE detalhadas** como exemplo:

1. **H1 - Repositórios & CI/CD** (Infraestrutura)
2. **H2 - Precisão Decimal** (Core Engine)
3. **H3 - Observabilidade** (Platform)

Cada uma demonstra TODOS os elementos do template aplicados na prática, incluindo:

* Código real de implementação
* Configurações YAML
* Testes com Jest/Vitest
* Logs JSON estruturados
* Métricas Prometheus
* Dashboards Grafana

**🎯 Template Universal Fornecido:**

O documento inclui um **template completo e reutilizável** que deve ser aplicado para todas as 25 histórias, garantindo:

1. **Completude:** Nenhum aspecto esquecido
2. **Padronização:** Mesmo formato para todas HUs
3. **Rastreabilidade:** Links e referências claras
4. **Qualidade:** DoR e DoD rigorosos
5. **Gestão de Riscos:** Proativa e documentada
6. **Técnico:** Detalhes de implementação
7. **Observabilidade:** Monitoramento desde o início

**📈 Benefícios da Nova Estrutura:**

* **Para PMs:** Visão completa do escopo e riscos
* **Para Devs:** Especificação técnica clara
* **Para QA:** Casos de teste e golden files definidos
* **Para DevOps:** Observabilidade e deploy claros
* **Para Stakeholders:** Rastreabilidade e métricas

**🔄 Próximos Passos Recomendados:**

1. **Aplicar o template** para as histórias H4-H25
2. **Criar repositório** com a estrutura de golden files
3. **Configurar ferramentas** de observabilidade
4. **Estabelecer processo** de review de HUs
5. **Treinar equipe** no novo padrão

Este documento v2.0 é uma evolução significativa que transforma cada História de Usuário em um **contrato completo** entre todos os envolvidos no desenvolvimento, garantindo clareza, qualidade e sucesso na entrega.

**📋 Como Descrever uma Sprint - Guia Completo FinMath (v2.0)**

**🎯 Propósito**

Este documento estabelece o padrão COMPLETO para descrição de Histórias de Usuário (HUs) no projeto FinMath, incluindo TODOS os elementos necessários para planejamento, execução, validação e rastreabilidade.

**📑 Template Completo de História de Usuário**

**Estrutura Obrigatória para Cada HU:**

1. **Identificação e Metadados**
2. **História de Usuário (As a/I want/So that)**
3. **Contexto de Negócio**
4. **Requisitos Funcionais (RF)**
5. **Requisitos Não Funcionais (RNF)**
6. **Critérios de Aceite (GWT)**
7. **Casos de Teste Detalhados**
8. **Matriz de Testes & Golden Files**
9. **Dependências e Relacionamentos**
10. **Implementação Técnica**
11. **Observabilidade & Auditoria**
12. **Definition of Ready (DoR)**
13. **Definition of Done (DoD)**
14. **Riscos & Mitigações**
15. **Rastreabilidade**

**📋 Catálogo Completo de Histórias (Versão Expandida)**

**H1 - Repositórios & CI/CD**

**📌 Identificação**

* **ID:** H1
* **Épico:** E1 - Engine & Fundamentos
* **Sprint:** 0 (Setup)
* **Prioridade:** P0 (Crítica)
* **Pontos:** 8
* **Responsável:** DevOps Lead
* **Status:** ✅ 100% Concluído

**📝 História de Usuário**

**Como** desenvolvedor  
**Quero** repositórios configurados com pipeline de CI/CD completo  
**Para** garantir qualidade contínua e automação do processo de desenvolvimento

**💼 Contexto de Negócio**

**Problema**

* Sem automação, bugs chegam a produção
* Processo manual propenso a erros
* Falta de padronização no desenvolvimento
* Tempo perdido em tarefas repetitivas

**Solução**

* Pipeline automatizado end-to-end
* Validação contínua de código
* Deploy automatizado e seguro
* Métricas de qualidade em tempo real

**Valor de Negócio**

* **ROI:** Redução de 70% no tempo de detecção de bugs
* **Economia:** 20h/semana em tarefas manuais
* **Qualidade:** 99% de builds bem-sucedidos
* **Time-to-market:** Deploy em < 30min

**✅ Requisitos Funcionais (RF)**

**RF1.1 - Estrutura de Repositórios**

* Monorepo com workspaces
* Separação engine/api/ui
* Versionamento semântico
* Branch protection rules

**RF1.2 - Pipeline CI**

* Trigger em PR e merge
* Jobs paralelos quando possível
* Cache de dependências
* Artefatos preservados

**RF1.3 - Análise de Código**

* ESLint com regras customizadas
* Prettier para formatação
* TypeScript strict mode
* Sonar analysis

**RF1.4 - Testes Automatizados**

* Unit tests (Jest)
* Property tests (fast-check)
* Integration tests
* E2E tests (Playwright)
* Golden Files validation

**RF1.5 - Build & Deploy**

* Build otimizado para produção
* Docker containerization
* Deploy para staging automático
* Deploy para produção com aprovação

**⚡ Requisitos Não Funcionais (RNF)**

**RNF1.1 - Performance**

* Pipeline P95 < 10 minutos
* Paralelização máxima
* Cache eficiente

**RNF1.2 - Confiabilidade**

* Retry automático em falhas transitórias
* Rollback automatizado
* Backup de artefatos

**RNF1.3 - Segurança**

* Secrets management
* SAST/DAST scanning
* Dependency vulnerability check
* Container scanning

**RNF1.4 - Observabilidade**

* Logs centralizados
* Métricas de pipeline
* Alertas de falha
* Dashboard de status

**🎯 Critérios de Aceite (GWT)**

**AC1: Pipeline Executando**

**Dado** um PR com mudanças de código  
**Quando** o pipeline é acionado  
**Então** todos os jobs devem executar:

* ✅ Checkout code
* ✅ Install dependencies
* ✅ Lint (ESLint + Prettier)
* ✅ Type check (TypeScript)
* ✅ Unit tests (Jest)
* ✅ Property tests
* ✅ Integration tests
* ✅ Golden Files verify
* ✅ Build
* ✅ E2E tests
* ✅ Coverage report

**AC2: Performance do Pipeline**

**Dado** execução completa do pipeline  
**Quando** medimos tempos  
**Então** deve atender SLA:

* P50 < 7 minutos
* P95 < 10 minutos
* P99 < 15 minutos

**AC3: Artefatos Gerados**

**Dado** build bem-sucedido  
**Quando** consultamos artefatos  
**Então** devem estar disponíveis:

* Build output (dist/)
* Coverage reports (HTML/JSON)
* Test reports
* Docker images
* Logs completos

**AC4: Notificações**

**Dado** mudança de status do pipeline  
**Quando** ocorre sucesso/falha  
**Então** notificar:

* PR com status check
* Slack #ci-cd channel
* Email para responsável

**🧪 Casos de Teste Detalhados**

**CT-H1.1: Pipeline com Sucesso**

Cenário: PR com código válido

Entrada:

- PR com mudança em /engine

- Todos os testes passando

- Código formatado

Execução:

1. Push para branch feature

2. Criar PR para main

3. Aguardar pipeline

Validação:

- Status: ✅ verde

- Tempo: < 10 min

- Coverage: > 80%

- Artefatos: disponíveis

**CT-H1.2: Pipeline com Falha em Testes**

Cenário: PR com teste quebrado

Entrada:

- Teste unitário falhando

Execução:

1. Push código com erro

2. Pipeline executa

Validação:

- Status: ❌ vermelho

- Job failed: "Unit Tests"

- Logs: erro específico visível

- PR: merge bloqueado

**CT-H1.3: Pipeline com Falha em Lint**

Cenário: Código não formatado

Entrada:

- Arquivo sem prettier

- ESLint violations

Execução:

1. Push código não formatado

Validação:

- Status: ❌ vermelho

- Job failed: "Lint"

- Sugestões: fixes disponíveis

**CT-H1.4: Retry em Falha Transitória**

Cenário: Falha de rede temporária

Entrada:

- Simulate network timeout

Execução:

1. Job falha por timeout

2. Retry automático (3x)

Validação:

- Retry: executado

- Status final: ✅ (após retry)

**📊 Matriz de Testes & Golden Files**

| **ID** | **Caso** | **Entrada** | **Esperado** | **Golden File** | **Status** |
| --- | --- | --- | --- | --- | --- |
| CT-H1.1 | Pipeline completo | PR válido | Todos jobs ✅ | N/A | ✅ Pass |
| CT-H1.2 | Falha em testes | Test quebrado | Block merge | N/A | ✅ Pass |
| CT-H1.3 | Falha em lint | Código não formatado | Sugestões | N/A | ✅ Pass |
| CT-H1.4 | Retry automático | Network failure | Retry 3x | N/A | ✅ Pass |
| CT-H1.5 | Cache hit | 2º run mesmo PR | < 5 min | N/A | ✅ Pass |
| CT-H1.6 | Parallel jobs | Jobs independentes | Execução paralela | N/A | ✅ Pass |

**🔗 Relacionamentos**

**Depende de:**

* **Infraestrutura:** GitHub/GitLab disponível
* **Secrets:** Tokens configurados
* **Docker:** Registry acessível

**Bloqueia:**

* **TODAS as outras HUs** (sem CI/CD não há desenvolvimento seguro)
* **H2:** Precisão Decimal (precisa de testes)
* **H3:** Observabilidade (precisa de deploy)

**Relacionado a:**

* **Issue #001:** Setup inicial do projeto
* **Issue #002:** Escolha de ferramentas
* **ADR-001:** Decisão por monorepo
* **ADR-009:** GitHub Actions vs Jenkins

**🛠️ Implementação Técnica**

**Stack Tecnológico**

CI/CD: GitHub Actions

Linguagens: TypeScript, Python, Bash

Testes: Jest, Playwright, fast-check

Análise: ESLint, Prettier, SonarCloud

Container: Docker, Docker Compose

Registry: GitHub Packages

Secrets: GitHub Secrets

Cache: GitHub Cache API

**Estrutura de Arquivos**

.github/

workflows/

ci.yml # Pipeline principal

cd.yml # Deploy pipeline

security.yml # Security scanning

release.yml # Release automation

actions/

setup/ # Custom actions

CODEOWNERS # Ownership rules

scripts/

ci/

validate.sh # Validações locais

pre-commit.sh # Git hooks

configs/

eslint.config.js

prettier.config.js

jest.config.js

tsconfig.json

**Código de Referência**

# .github/workflows/ci.yml

name: CI Pipeline

on:

pull\_request:

branches: [main, develop]

push:

branches: [main]

jobs:

setup:

runs-on: ubuntu-latest

outputs:

cache-key: ${{ steps.cache.outputs.key }}

steps:

- uses: actions/checkout@v4

- uses: actions/setup-node@v4

with:

node-version: '20'

cache: 'npm'

lint:

needs: setup

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- run: npm ci

- run: npm run lint

- run: npm run format:check

test:

needs: setup

runs-on: ubuntu-latest

strategy:

matrix:

test-type: [unit, integration, property]

steps:

- uses: actions/checkout@v4

- run: npm ci

- run: npm run test:${{ matrix.test-type }}

- uses: actions/upload-artifact@v4

with:

name: coverage-${{ matrix.test-type }}

path: coverage/

golden-files:

needs: setup

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- run: npm ci

- run: npm run golden:verify

build:

needs: [lint, test, golden-files]

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- run: npm ci

- run: npm run build

- uses: actions/upload-artifact@v4

with:

name: build-output

path: dist/

**📊 Observabilidade & Auditoria**

**Logs**

{

"timestamp": "2025-10-22T10:30:00Z",

"pipeline\_id": "run-123456",

"job": "test",

"status": "success",

"duration\_ms": 4500,

"trigger": "pull\_request",

"branch": "feature/h1-cicd",

"commit": "abc123",

"author": "dev@finmath.com"

}

**Métricas**

* **Build Success Rate:** 99.5%
* **Average Duration:** 7.5 min
* **Cache Hit Rate:** 85%
* **Failed Jobs (week):** 3
* **Flaky Tests:** 0

**Alertas**

* Pipeline failure → Slack + Email
* Duration > 15 min → Warning
* Coverage < 80% → Block merge
* Security vulnerability → Critical alert

**✔️ Definition of Ready (DoR)**

* [ ] Requisitos claramente definidos
* [ ] Ferramentas selecionadas e aprovadas
* [ ] Ambiente de CI disponível
* [ ] Tokens e secrets configurados
* [ ] Template de PR definido
* [ ] Branch strategy definida
* [ ] CODEOWNERS configurado

**✅ Definition of Done (DoD)**

* [x] Pipeline configurado e funcional
* [x] Todos os jobs executando com sucesso
* [x] Tempo P95 < 10 minutos
* [x] Cache funcionando (hit rate > 80%)
* [x] Notificações configuradas
* [x] Documentação completa
* [x] README com badges
* [x] Secrets seguros
* [x] Backup de configurações
* [x] Rollback testado
* [x] Monitoramento ativo
* [x] Equipe treinada

**⚠️ Riscos & Mitigações**

**Risco 1: Dependência do GitHub**

* **Probabilidade:** Baixa
* **Impacto:** Alto
* **Mitigação:** Backup configs, plano contingência GitLab

**Risco 2: Pipeline lento**

* **Probabilidade:** Média
* **Impacto:** Médio
* **Mitigação:** Otimização contínua, cache agressivo, paralelização

**Risco 3: Flaky tests**

* **Probabilidade:** Alta
* **Impacto:** Médio
* **Mitigação:** Retry mechanism, quarantine flaky tests

**Risco 4: Secrets exposure**

* **Probabilidade:** Baixa
* **Impacto:** Crítico
* **Mitigação:** Secret scanning, rotation policy, audit logs

**📈 Rastreabilidade**

**Documentos Relacionados**

* **Visão & Plano:** v1.1 - Seção "Infraestrutura"
* **ADR-001:** Decisão por Monorepo
* **ADR-009:** CI/CD com GitHub Actions
* **Playbook Testes:** Seção "Pipeline"

**Commits & PRs**

* PR #001: Initial CI setup
* PR #002: Add test jobs
* PR #003: Golden files validation
* PR #004: E2E integration
* PR #005: Performance optimizations

**Issues Relacionadas**

* Issue #001: Setup projeto
* Issue #002: Escolha ferramentas
* Issue #015: Otimização pipeline
* Issue #023: Security scanning

**H2 - Precisão Monetária (Decimal) & Arredondamento**

**📌 Identificação**

* **ID:** H2
* **Épico:** E1 - Engine & Fundamentos
* **Sprint:** 1
* **Prioridade:** P0 (Crítica)
* **Pontos:** 5
* **Responsável:** Engine Lead
* **Status:** ✅ 100% Concluído

**📝 História de Usuário**

**Como** engenheiro de motor  
**Quero** operar com precisão decimal e arredondamento consistente  
**Para** eliminar erros de ponto flutuante em cálculos financeiros

**💼 Contexto de Negócio**

**Problema**

* JavaScript float64: 0.1 + 0.2 ≠ 0.3
* Erros acumulados em iterações
* Divergências com sistemas bancários
* Compliance regulatório em risco

**Solução**

* Biblioteca decimal.js/big.js
* Arredondamento Half-Up padrão
* Ajuste de centavos automatizado
* Validação contra golden files

**Valor de Negócio**

* **Precisão:** 100% exata até 10 casas
* **Compliance:** Aderente ao BACEN
* **Confiança:** Zero divergências
* **Manutenibilidade:** Código limpo

**✅ Requisitos Funcionais (RF)**

**RF2.1 - Operações Básicas**

// Soma, subtração, multiplicação, divisão

function add(a: Decimal, b: Decimal): Decimal

function subtract(a: Decimal, b: Decimal): Decimal

function multiply(a: Decimal, b: Decimal): Decimal

function divide(a: Decimal, b: Decimal): Decimal

**RF2.2 - Arredondamento**

// Half-Up com precisão configurável

function round(value: Decimal, places: number = 2): Decimal

function roundCurrency(value: Decimal): Decimal // sempre 2 casas

**RF2.3 - Ajuste de Centavos**

// Distribuir diferença na última parcela

function adjustLastInstallment(

installments: Decimal[],

total: Decimal

): Decimal[]

**RF2.4 - Conversão**

function fromString(value: string): Decimal

function fromNumber(value: number): Decimal

function toString(value: Decimal, format?: Format): string

**⚡ Requisitos Não Funcionais (RNF)**

**RNF2.1 - Precisão**

* Erro absoluto: 0
* Precisão interna: 40 dígitos
* Precisão display: configurável

**RNF2.2 - Performance**

* Operação básica: < 1ms
* 1000 operações: < 100ms
* Memory footprint: < 1KB/número

**RNF2.3 - Compatibilidade**

* Node.js 18+
* Browsers modernos
* React Native
* Isomorphic

**🎯 Critérios de Aceite (GWT)**

**AC1: Precisão Exata**

**Dado** valores monetários problemáticos  
**Quando** executamos operações  
**Então** resultado é exato:

0.1 + 0.2 = 0.3 (exato)

10.00 / 3 = 3.33 (arredondado)

3.33 \* 3 = 9.99 (não 10.00)

**AC2: Arredondamento Half-Up**

**Dado** valores com múltiplas casas  
**Quando** arredondamos  
**Então** segue regra Half-Up:

1.235 → 1.24 (5 sobe)

1.234 → 1.23 (4 desce)

1.225 → 1.23 (5 sobe para par)

**AC3: Ajuste de Centavos**

**Dado** divisão não exata  
**Quando** calculamos parcelas  
**Então** ajuste na última:

R$ 100,00 / 3 parcelas:

Parcela 1: R$ 33,33

Parcela 2: R$ 33,33

Parcela 3: R$ 33,34 (ajustada)

Total: R$ 100,00 ✓

**🧪 Casos de Teste Detalhados**

**CT-H2.1: Operações Básicas**

describe('Decimal Operations', () => {

it('should add with precision', () => {

const a = new Decimal('0.1');

const b = new Decimal('0.2');

expect(a.plus(b).toString()).toBe('0.3');

});

it('should handle large numbers', () => {

const a = new Decimal('999999999999.99');

const b = new Decimal('0.01');

expect(a.plus(b).toString()).toBe('1000000000000');

});

});

**CT-H2.2: Arredondamento**

describe('Rounding', () => {

test.each([

['1.234', 2, '1.23'],

['1.235', 2, '1.24'],

['1.2345', 3, '1.235'],

['1.9999', 2, '2.00'],

])('round(%s, %d) = %s', (input, places, expected) => {

expect(round(new Decimal(input), places).toString())

.toBe(expected);

});

});

**CT-H2.3: Ajuste de Parcelas**

describe('Installment Adjustment', () => {

it('should adjust last installment', () => {

const total = new Decimal('100');

const n = 3;

const pmt = total.div(n); // 33.333...

const installments = Array(n).fill(pmt.round(2));

const adjusted = adjustLastInstallment(installments, total);

expect(adjusted[0].toString()).toBe('33.33');

expect(adjusted[1].toString()).toBe('33.33');

expect(adjusted[2].toString()).toBe('33.34');

const sum = adjusted.reduce((a, b) => a.plus(b));

expect(sum.toString()).toBe('100.00');

});

});

**📊 Matriz de Testes & Golden Files**

| **ID** | **Caso** | **Entrada** | **Esperado** | **Golden File** | **Status** |
| --- | --- | --- | --- | --- | --- |
| CT-H2.1 | Soma decimal | 0.1 + 0.2 | 0.3 | DECIMAL\_001.json | ✅ |
| CT-H2.2 | Divisão com resto | 10/3 | 3.33... | DECIMAL\_002.json | ✅ |
| CT-H2.3 | Round Half-Up | 1.235, 2 casas | 1.24 | DECIMAL\_003.json | ✅ |
| CT-H2.4 | Ajuste parcelas | R$100/3 | 33.33,33.33,33.34 | DECIMAL\_004.json | ✅ |
| CT-H2.5 | Large numbers | 10^15 + 0.01 | Precisão mantida | DECIMAL\_005.json | ✅ |

**🔗 Relacionamentos**

**Depende de:**

* **H1:** CI/CD (para testes automatizados)
* **Biblioteca:** decimal.js v10.4+

**Bloqueia:**

* **H4:** Juros Compostos (usa Decimal)
* **H5:** Equivalência Taxas (usa Decimal)
* **H6:** Séries (usa Decimal)
* **H9:** Price (usa Decimal)
* **H11:** SAC (usa Decimal)
* **TODAS** as HUs com cálculo

**Relacionado a:**

* **Issue #003:** Escolha biblioteca decimal
* **Issue #004:** Padrão de arredondamento
* **ADR-001:** Precisão decimal obrigatória
* **RFC-001:** Especificação de arredondamento

**🛠️ Implementação Técnica**

**Arquitetura**

// engine/src/core/decimal.ts

import { Decimal as DecimalJS } from 'decimal.js';

// Configuração global

DecimalJS.set({

precision: 40,

rounding: DecimalJS.ROUND\_HALF\_UP,

toExpNeg: -9,

toExpPos: 20,

});

// Tipo customizado

export type Decimal = DecimalJS;

export const Decimal = DecimalJS;

// Helpers

export function toDecimal(value: string | number): Decimal {

return new Decimal(value);

}

export function round2(value: Decimal): Decimal {

return value.toDecimalPlaces(2);

}

export function roundN(value: Decimal, n: number): Decimal {

return value.toDecimalPlaces(n);

}

// Formatação

export function formatBRL(value: Decimal): string {

return `R$ ${value.toFixed(2).replace('.', ',')}`;

}

// Ajuste de centavos

export function adjustCents(

amounts: Decimal[],

target: Decimal

): Decimal[] {

const sum = amounts.reduce((a, b) => a.plus(b));

const diff = target.minus(sum);

if (diff.abs().gt(0.01)) {

throw new Error('Diferença maior que 1 centavo');

}

const adjusted = [...amounts];

adjusted[adjusted.length - 1] =

adjusted[adjusted.length - 1].plus(diff);

return adjusted;

}

**Testes de Propriedade**

// engine/test/property/decimal.test.ts

import fc from 'fast-check';

import { Decimal, round2 } from '@/core/decimal';

describe('Decimal Properties', () => {

it('addition is commutative', () => {

fc.assert(

fc.property(

fc.float({ min: -1e9, max: 1e9 }),

fc.float({ min: -1e9, max: 1e9 }),

(a, b) => {

const da = new Decimal(a);

const db = new Decimal(b);

return da.plus(db).equals(db.plus(da));

}

)

);

});

it('rounding preserves sign', () => {

fc.assert(

fc.property(

fc.float({ min: -1e6, max: 1e6 }),

(n) => {

const d = new Decimal(n);

const r = round2(d);

return d.isPositive() === r.isPositive() ||

d.isZero() && r.isZero();

}

)

);

});

});

**📊 Observabilidade & Auditoria**

**Logs de Precisão**

{

"operation": "divide",

"inputs": ["100.00", "3"],

"result": "33.333333...",

"rounded": "33.33",

"precision\_used": 40,

"rounding\_mode": "HALF\_UP"

}

**Métricas**

* Operations/sec: 10,000+
* Memory usage: < 10MB
* Precision errors: 0
* Rounding disputes: 0

**✔️ Definition of Ready (DoR)**

* [x] Biblioteca decimal selecionada
* [x] Padrão de arredondamento aprovado
* [x] Golden files de referência
* [x] Casos de teste definidos
* [x] Performance benchmark estabelecido

**✅ Definition of Done (DoD)**

* [x] Todas operações implementadas
* [x] 100% cobertura de testes
* [x] Zero erros de precisão
* [x] Performance validada
* [x] Golden files passando
* [x] Property tests passando
* [x] Documentação completa
* [x] Code review aprovado
* [x] Integração com H4-H11

**⚠️ Riscos & Mitigações**

**Risco 1: Performance degradada**

* **Probabilidade:** Média
* **Impacto:** Alto
* **Mitigação:** Cache, lazy evaluation, profiling

**Risco 2: Incompatibilidade com sistemas legados**

* **Probabilidade:** Alta
* **Impacto:** Médio
* **Mitigação:** Adapters, testes comparativos

**Risco 3: Divergência de arredondamento**

* **Probabilidade:** Baixa
* **Impacto:** Alto
* **Mitigação:** Golden files, validação externa

**📈 Rastreabilidade**

**Documentos**

* **ADR-001:** Decisão decimal.js
* **RFC-001:** Especificação arredondamento
* **Guia BACEN:** Circular 3.959

**PRs & Commits**

* PR #010: Decimal implementation
* PR #011: Rounding helpers
* PR #012: Currency formatting
* Commit abc123: Fix adjustment algorithm

**H3 - Observabilidade Básica**

**📌 Identificação**

* **ID:** H3
* **Épico:** E4 - Observabilidade & Auditoria
* **Sprint:** 1
* **Prioridade:** P1 (Alta)
* **Pontos:** 3
* **Responsável:** Platform Engineer
* **Status:** ✅ 100% Concluído

**📝 História de Usuário**

**Como** engenheiro de operações  
**Quero** logs estruturados e correlacionados  
**Para** rastrear e debugar problemas em produção

**💼 Contexto de Negócio**

**Problema**

* Debugging às cegas em produção
* Impossível rastrear fluxo de requisições
* Métricas inexistentes
* MTTR (Mean Time To Resolve) alto

**Solução**

* Logs estruturados JSON
* Correlation IDs únicos
* Métricas em tempo real
* Dashboards operacionais

**Valor de Negócio**

* **MTTR:** Redução de 60%
* **Uptime:** 99.95%
* **Debugging:** 10x mais rápido
* **Compliance:** Audit trail completo

**✅ Requisitos Funcionais (RF)**

**RF3.1 - Logs Estruturados**

interface LogEntry {

timestamp: string; // ISO-8601

level: 'debug' | 'info' | 'warn' | 'error' | 'fatal';

message: string;

calculationId: string; // UUID v4

motorVersion: string; // semver

userId?: string;

tenantId?: string;

metadata: Record<string, any>;

error?: {

message: string;

stack: string;

code: string;

};

}

**RF3.2 - Correlation ID**

* Geração automática se ausente
* Propagação em toda a stack
* Headers HTTP: X-Correlation-ID
* Logs: calculationId field

**RF3.3 - Níveis de Log**

logger.debug('Detailed debugging info');

logger.info('Normal operations');

logger.warn('Warning conditions');

logger.error('Error conditions');

logger.fatal('System is unusable');

**RF3.4 - Contexto Enriquecido**

logger.child({

userId: '123',

operation: 'calculate\_price',

inputs: { pv: 10000, rate: 0.02, n: 12 }

}).info('Calculation started');

**⚡ Requisitos Não Funcionais (RNF)**

**RNF3.1 - Performance**

* Log write: < 1ms
* No blocking I/O
* Async buffer: 10K messages
* Batch shipping: 100ms intervals

**RNF3.2 - Volume**

* 10K logs/segundo
* Retention: 30 dias hot, 1 ano cold
* Compression: gzip
* Rotation: daily

**RNF3.3 - Confiabilidade**

* Buffer overflow protection
* Graceful degradation
* Circuit breaker para shipping
* Local fallback

**🎯 Critérios de Aceite (GWT)**

**AC1: Logs Estruturados JSON**

**Dado** uma operação no sistema  
**Quando** geramos logs  
**Então** formato é JSON válido:

{

"timestamp": "2025-10-22T10:30:00.123Z",

"level": "info",

"message": "Price calculation completed",

"calculationId": "calc\_abc123",

"motorVersion": "0.2.0",

"userId": "user\_123",

"metadata": {

"operation": "price\_pmt",

"duration\_ms": 45,

"input\_pv": 10000,

"output\_pmt": 946.56

}

}

**AC2: Correlação End-to-End**

**Dado** requisição HTTP  
**Quando** flui pelo sistema  
**Então** todos os logs compartilham calculationId:

API Gateway → Engine → Database → Response

↓ ↓ ↓ ↓

calc\_abc123 calc\_abc123 calc\_abc123

**AC3: Dashboards Operacionais**

**Dado** métricas coletadas  
**Quando** visualizamos dashboard  
**Então** disponível em tempo real:

* Request rate (req/s)
* Error rate (%)
* Latency (P50, P95, P99)
* Active calculations

**🧪 Casos de Teste Detalhados**

**CT-H3.1: Geração de Correlation ID**

describe('Correlation ID', () => {

it('should generate if missing', () => {

const req = { headers: {} };

const id = getOrCreateCorrelationId(req);

expect(id).toMatch(/^calc\_[a-z0-9]{6,}$/);

});

it('should use existing if present', () => {

const req = { headers: { 'x-correlation-id': 'calc\_xyz' } };

const id = getOrCreateCorrelationId(req);

expect(id).toBe('calc\_xyz');

});

});

**CT-H3.2: Log Levels**

describe('Log Levels', () => {

it('should respect minimum level', () => {

logger.level = 'warn';

const debugSpy = jest.spyOn(logger, 'debug');

const warnSpy = jest.spyOn(logger, 'warn');

logger.debug('hidden');

logger.warn('visible');

expect(debugSpy).not.toHaveBeenCalled();

expect(warnSpy).toHaveBeenCalled();

});

});

**📊 Matriz de Testes & Golden Files**

| **ID** | **Caso** | **Entrada** | **Validação** | **Status** |
| --- | --- | --- | --- | --- |
| CT-H3.1 | Log estruturado | Operação Price | JSON válido, campos presentes | ✅ |
| CT-H3.2 | Correlation ID | Request sem ID | ID gerado automaticamente | ✅ |
| CT-H3.3 | Propagação | Fluxo completo | Mesmo ID em todos logs | ✅ |
| CT-H3.4 | Error logging | Exception | Stack trace capturado | ✅ |
| CT-H3.5 | Performance | 10K logs/s | < 1ms latência | ✅ |
| CT-H3.6 | Buffer overflow | 100K logs burst | Graceful degradation | ✅ |

**🔗 Relacionamentos**

**Depende de:**

* **H1:** CI/CD (deploy de infraestrutura)
* **Infra:** ElasticSearch ou CloudWatch

**Bloqueia:**

* **H9-H25:** Todas precisam de observabilidade
* **H21:** Snapshots (usa calculationId)
* **H22:** Validador (precisa logs)

**Relacionado a:**

* **Issue #005:** Escolha stack observabilidade
* **Issue #006:** LGPD compliance em logs
* **ADR-003:** Estrutura de logs
* **SRE-001:** SLIs e SLOs

**🛠️ Implementação Técnica**

**Logger Configuration**

// engine/src/core/logger.ts

import pino from 'pino';

import { v4 as uuidv4 } from 'uuid';

const logger = pino({

level: process.env.LOG\_LEVEL || 'info',

formatters: {

level: (label) => ({ level: label }),

bindings: () => ({

pid: process.pid,

hostname: process.env.HOSTNAME,

motorVersion: process.env.MOTOR\_VERSION || '0.2.0',

}),

},

timestamp: () => `,"timestamp":"${new Date().toISOString()}"`,

serializers: {

error: pino.stdSerializers.err,

request: (req) => ({

method: req.method,

url: req.url,

headers: req.headers,

calculationId: req.calculationId,

}),

response: (res) => ({

statusCode: res.statusCode,

duration: res.duration,

}),

},

});

// Correlation ID middleware

export function correlationMiddleware(req, res, next) {

req.calculationId = req.headers['x-correlation-id'] ||

`calc\_${uuidv4().slice(0, 8)}`;

res.setHeader('X-Correlation-ID', req.calculationId);

req.logger = logger.child({ calculationId: req.calculationId });

next();

}

// Contexto para cálculos

export function createCalculationLogger(

calculationId: string,

operation: string,

inputs: Record<string, any>

): pino.Logger {

return logger.child({

calculationId,

operation,

inputs,

startTime: Date.now(),

});

}

**Métricas com Prometheus**

// engine/src/core/metrics.ts

import { Registry, Counter, Histogram, Gauge } from 'prom-client';

const register = new Registry();

// Contadores

export const requestCount = new Counter({

name: 'finmath\_requests\_total',

help: 'Total requests',

labelNames: ['method', 'endpoint', 'status'],

registers: [register],

});

export const errorCount = new Counter({

name: 'finmath\_errors\_total',

help: 'Total errors',

labelNames: ['type', 'operation'],

registers: [register],

});

// Histogramas

export const requestDuration = new Histogram({

name: 'finmath\_request\_duration\_seconds',

help: 'Request duration',

labelNames: ['method', 'endpoint'],

buckets: [0.1, 0.5, 1, 2, 5],

registers: [register],

});

export const calculationDuration = new Histogram({

name: 'finmath\_calculation\_duration\_ms',

help: 'Calculation duration',

labelNames: ['operation'],

buckets: [10, 50, 100, 500, 1000],

registers: [register],

});

// Gauges

export const activeCalculations = new Gauge({

name: 'finmath\_active\_calculations',

help: 'Active calculations',

registers: [register],

});

// Middleware

export function metricsMiddleware(req, res, next) {

const start = Date.now();

res.on('finish', () => {

const duration = (Date.now() - start) / 1000;

requestCount.labels(

req.method,

req.route?.path || 'unknown',

res.statusCode.toString()

).inc();

requestDuration.labels(

req.method,

req.route?.path || 'unknown'

).observe(duration);

if (res.statusCode >= 400) {

errorCount.labels(

res.statusCode >= 500 ? 'server' : 'client',

req.route?.path || 'unknown'

).inc();

}

});

next();

}

**Dashboard Configuration (Grafana)**

{

"dashboard": {

"title": "FinMath Operations",

"panels": [

{

"title": "Request Rate",

"targets": [{

"expr": "rate(finmath\_requests\_total[5m])"

}]

},

{

"title": "Error Rate",

"targets": [{

"expr": "rate(finmath\_errors\_total[5m])"

}]

},

{

"title": "Latency P95",

"targets": [{

"expr": "histogram\_quantile(0.95, finmath\_request\_duration\_seconds)"

}]

},

{

"title": "Active Calculations",

"targets": [{

"expr": "finmath\_active\_calculations"

}]

}

]

}

}

**📊 Observabilidade & Auditoria**

**Log Aggregation Pipeline**

Application → Stdout → Fluentd → ElasticSearch → Kibana

↓ ↓ ↓ ↓ ↓

Metrics Prometheus Grafana Alerting Dashboard

**Alertas Configurados**

* Error rate > 1% → PagerDuty
* Latency P95 > 1s → Slack warning
* Disk usage > 80% → Email ops
* Calculation timeout → Immediate alert

**Retention Policy**

* Hot data: 30 dias (SSD)
* Warm data: 90 dias (HDD)
* Cold data: 1 ano (S3)
* Compliance: 7 anos (Glacier)

**✔️ Definition of Ready (DoR)**

* [x] Stack de observabilidade definida
* [x] Estrutura de logs aprovada
* [x] Métricas SLI identificadas
* [x] Dashboards desenhados
* [x] Retention policy definida

**✅ Definition of Done (DoD)**

* [x] Logger configurado e testado
* [x] Correlation ID funcionando
* [x] Métricas exportadas
* [x] Dashboards criados
* [x] Alertas configurados
* [x] Documentation completa
* [x] Performance validada
* [x] Integration tests
* [x] Equipe treinada

**⚠️ Riscos & Mitigações**

**Risco 1: Volume excessivo de logs**

* **Probabilidade:** Alta
* **Impacto:** Médio
* **Mitigação:** Sampling, agregação, rate limiting

**Risco 2: PII em logs (LGPD)**

* **Probabilidade:** Média
* **Impacto:** Alto
* **Mitigação:** Sanitização, masking, audit

**Risco 3: Latência de shipping**

* **Probabilidade:** Baixa
* **Impacto:** Baixo
* **Mitigação:** Buffer local, async shipping

**📈 Rastreabilidade**

**Documentos**

* **ADR-003:** Estrutura de logs
* **SRE-001:** SLIs e SLOs
* **LGPD-001:** Compliance em logs

**Issues & PRs**

* Issue #005: Stack selection
* Issue #006: LGPD compliance
* PR #015: Logger implementation
* PR #016: Metrics integration
* PR #017: Dashboards

**Template para as demais HUs (H4-H25)**

[Nota: Por limitação de espaço, vou fornecer o template expandido que deve ser aplicado para TODAS as histórias H4-H25]

**📋 Template Completo para História de Usuário**

# H[X] - [Nome da Funcionalidade]

## 📌 Identificação

- \*\*ID:\*\* H[X]

- \*\*Épico:\*\* E[1-5] - [Nome do Épico]

- \*\*Sprint:\*\* [0-5+]

- \*\*Prioridade:\*\* P[0-3] ([Crítica/Alta/Média/Baixa])

- \*\*Pontos:\*\* [1-13]

- \*\*Responsável:\*\* [Role]

- \*\*Status:\*\* [✅ 100% | 🟡 XX% | 🔴 XX% | ⏳ 0%]

- \*\*Data Início:\*\* YYYY-MM-DD

- \*\*Data Fim:\*\* YYYY-MM-DD

- \*\*Tipo:\*\* [Feature | Technical | Bug | Debt]

## 📝 História de Usuário

\*\*Como\*\* [persona]

\*\*Quero\*\* [funcionalidade]

\*\*Para\*\* [benefício/valor]

## 💼 Contexto de Negócio

### Problema

- [Descrição do problema atual]

- [Impactos do problema]

- [Custos associados]

### Solução

- [Descrição da solução proposta]

- [Como resolve o problema]

- [Diferenciais]

### Valor de Negócio

- \*\*ROI:\*\* [Métricas de retorno]

- \*\*Economia:\*\* [Redução de custos]

- \*\*Eficiência:\*\* [Ganhos operacionais]

- \*\*Satisfação:\*\* [NPS/CSAT esperado]

## ✅ Requisitos Funcionais (RF)

### RF[X].1 - [Nome do Requisito]

[Descrição detalhada]

[Regras de negócio]

[Fórmulas se aplicável]

### RF[X].2 - [Nome do Requisito]

[Descrição detalhada]

## ⚡ Requisitos Não Funcionais (RNF)

### RNF[X].1 - Performance

- [Métricas de latência]

- [Throughput esperado]

- [Limites de recursos]

### RNF[X].2 - Segurança

- [Requisitos de autenticação]

- [Autorização]

- [Criptografia]

### RNF[X].3 - Confiabilidade

- [SLA esperado]

- [Taxa de erro aceitável]

- [Recovery time]

### RNF[X].4 - Usabilidade

- [Tempo de aprendizado]

- [Taxa de erro de usuário]

- [Satisfação esperada]

## 🎯 Critérios de Aceite (GWT)

### AC1: [Nome do Critério]

\*\*Dado\*\* [contexto inicial]

\*\*Quando\*\* [ação executada]

\*\*Então\*\* [resultado esperado]

\*\*E\*\* [resultado adicional se aplicável]

### AC2: [Nome do Critério]

\*\*Dado\*\* [contexto]

\*\*Quando\*\* [ação]

\*\*Então\*\* [resultado]

## 🧪 Casos de Teste Detalhados

### CT-H[X].1: [Nome do Caso]

```yaml

Cenário: [Descrição]

Pré-condições:

- [Condição 1]

- [Condição 2]

Entrada:

- [Dados de entrada]

Execução:

1. [Passo 1]

2. [Passo 2]

3. [Passo 3]

Saída Esperada:

- [Resultado esperado]

Pós-condições:

- [Estado final]

Validações:

- [Check 1]

- [Check 2]

**CT-H[X].2: [Caso Negativo]**

[Estrutura similar]

**CT-H[X].3: [Caso de Borda]**

[Estrutura similar]

**📊 Matriz de Testes & Golden Files**

| **ID** | **Caso** | **Entrada** | **Esperado** | **Golden File** | **Status** |
| --- | --- | --- | --- | --- | --- |
| CT-H[X].1 | [Descrição] | [Input] | [Output] | [FILE\_001.json] | ✅/🔴 |
| CT-H[X].2 | [Descrição] | [Input] | [Output] | [FILE\_002.json] | ✅/🔴 |
| CT-H[X].3 | [Descrição] | [Input] | [Output] | [FILE\_003.json] | ✅/🔴 |
| CT-H[X].4 | [Descrição] | [Input] | [Output] | [FILE\_004.json] | ✅/🔴 |
| CT-H[X].5 | [Descrição] | [Input] | [Output] | [FILE\_005.json] | ✅/🔴 |

**🔗 Relacionamentos**

**Depende de:**

* **H[Y]:** [Descrição da dependência]
* **H[Z]:** [Descrição da dependência]
* **Biblioteca:** [Nome e versão]
* **Serviço:** [API/Serviço externo]

**Bloqueia:**

* **H[A]:** [O que bloqueia e por quê]
* **H[B]:** [O que bloqueia e por quê]

**Relacionado a:**

* **Issue #[XXX]:** [Descrição]
* **ADR-[XXX]:** [Decisão relacionada]
* **RFC-[XXX]:** [Especificação]
* **Bug #[XXX]:** [Bug relacionado]

**🛠️ Implementação Técnica**

**Stack Tecnológico**

Linguagem: TypeScript 5.x

Framework: [Express/Fastify]

Biblioteca: [decimal.js, date-fns, etc]

Database: [PostgreSQL/MongoDB]

Cache: [Redis]

Queue: [Bull/RabbitMQ]

**Arquitetura**

[Diagrama ASCII ou descrição da arquitetura]

**Estrutura de Arquivos**

engine/

src/

[module]/

index.ts

[feature].ts

[feature].test.ts

types.ts

test/

unit/

integration/

property/

golden/

**Código de Referência**

// Interface principal

interface [Feature] {

// Definições

}

// Implementação core

class [FeatureImpl] implements [Feature] {

// Implementação

}

// Exemplo de uso

const example = new [FeatureImpl]();

example.method();

**API Contracts**

Endpoint: [METHOD] /api/v1/[resource]

Request:

Headers:

- Content-Type: application/json

- X-Correlation-ID: [uuid]

Body:

field1: type

field2: type

Response:

200:

field1: type

field2: type

400:

error: ValidationError

500:

error: InternalError

**📊 Observabilidade & Auditoria**

**Logs Esperados**

{

"timestamp": "ISO-8601",

"level": "info|warn|error",

"calculationId": "uuid",

"operation": "[operation\_name]",

"input": {},

"output": {},

"duration\_ms": 0,

"metadata": {}

}

**Métricas**

* **[Métrica 1]:** [Descrição e threshold]
* **[Métrica 2]:** [Descrição e threshold]
* **[Métrica 3]:** [Descrição e threshold]

**Eventos de Auditoria**

* [Evento 1]: Quando e o que registrar
* [Evento 2]: Quando e o que registrar

**Alertas**

* **[Alerta 1]:** Condição → Ação
* **[Alerta 2]:** Condição → Ação

**✔️ Definition of Ready (DoR)**

* [ ] User Story refinada e aprovada
* [ ] Critérios de aceite claros (GWT)
* [ ] Casos de teste definidos
* [ ] Golden Files identificados
* [ ] Dependências mapeadas
* [ ] API contracts definidos
* [ ] Estimativa revisada
* [ ] Riscos identificados
* [ ] Protótipo/Mockup aprovado (se UI)
* [ ] Decisões técnicas tomadas (ADRs)

**✅ Definition of Done (DoD)**

**Código**

* [ ] Feature implementada conforme especificação
* [ ] Code review aprovado (2 approvers)
* [ ] Sem code smells (SonarQube)
* [ ] Complexidade ciclomática < 10

**Testes**

* [ ] Testes unitários (cobertura > 80%)
* [ ] Testes de integração
* [ ] Testes de propriedade (se aplicável)
* [ ] Testes E2E (fluxo principal)
* [ ] Golden Files passando
* [ ] Testes de performance

**Documentação**

* [ ] JSDoc/TSDoc completo
* [ ] README atualizado
* [ ] API docs (OpenAPI)
* [ ] Changelog atualizado
* [ ] ADR criado (se decisão arquitetural)

**Qualidade**

* [ ] Lint sem warnings
* [ ] Type check passando
* [ ] Build de produção OK
* [ ] Bundle size aceitável
* [ ] Performance dentro do SLA

**Segurança**

* [ ] SAST scan passed
* [ ] Dependency check
* [ ] Secrets não expostos
* [ ] OWASP top 10 verificado

**Acessibilidade (se UI)**

* [ ] WCAG AA compliance
* [ ] Keyboard navigation
* [ ] Screen reader tested
* [ ] Color contrast OK

**Deploy**

* [ ] Deploy em staging
* [ ] Smoke tests passed
* [ ] Rollback testado
* [ ] Monitoring configurado
* [ ] Alerts configurados

**⚠️ Riscos & Mitigações**

**Risco 1: [Nome do Risco]**

* **Probabilidade:** [Alta/Média/Baixa]
* **Impacto:** [Crítico/Alto/Médio/Baixo]
* **Detecção:** [Como detectar]
* **Mitigação:** [Ação preventiva]
* **Contingência:** [Plano B]
* **Responsável:** [Quem monitora]

**Risco 2: [Nome do Risco]**

[Estrutura similar]

**📈 Rastreabilidade**

**Documentos Relacionados**

* **[Doc 1]:** [Link e descrição]
* **[Doc 2]:** [Link e descrição]
* **ADR-[XXX]:** [Decisão relacionada]
* **RFC-[XXX]:** [Especificação]

**Commits & PRs**

* PR #[XXX]: [Descrição]
* Commit [hash]: [Mensagem]

**Issues Relacionadas**

* Issue #[XXX]: [Título]
* Bug #[XXX]: [Título]

**Métricas de Sucesso**

* **[KPI 1]:** [Meta vs Realizado]
* **[KPI 2]:** [Meta vs Realizado]
* **[KPI 3]:** [Meta vs Realizado]

**Feedback & Iterações**

* **v1.0:** [Mudanças da versão inicial]
* **v1.1:** [Melhorias implementadas]
* **Feedback usuários:** [Principais pontos]

**📝 Notas Adicionais**

**Decisões Técnicas**

* [Decisão 1 e justificativa]
* [Decisão 2 e justificativa]

**Débito Técnico**

* [Item 1 para refatorar futuramente]
* [Item 2 para melhorar]

**Lições Aprendidas**

* [Lição 1]
* [Lição 2]

**Próximos Passos**

* [Evolução 1 planejada]
* [Evolução 2 possível]

---

## 📊 Resumo de Aplicação do Template

Este template expandido deve ser aplicado para TODAS as 25 histórias (H1-H25), garantindo:

1. \*\*Completude:\*\* Todos os aspectos cobertos

2. \*\*Rastreabilidade:\*\* Links e referências claras

3. \*\*Testabilidade:\*\* Casos de teste e golden files

4. \*\*Observabilidade:\*\* Logs e métricas definidos

5. \*\*Qualidade:\*\* DoR e DoD rigorosos

6. \*\*Gestão de Riscos:\*\* Identificados e mitigados

7. \*\*Implementação:\*\* Detalhes técnicos claros

8. \*\*Relacionamentos:\*\* Dependências mapeadas

### Elementos Críticos por Tipo de História

\*\*Para Histórias de Cálculo (H4-H6, H9-H11, H14-H17):\*\*

- Fórmulas matemáticas exatas

- Golden Files obrigatórios

- Testes de propriedade

- Precisão decimal validada

\*\*Para Histórias de UI (H7, H8, H18, H24):\*\*

- Mockups/Protótipos

- Testes de acessibilidade

- Responsividade

- User journey

\*\*Para Histórias de Infraestrutura (H1, H3, H21, H22):\*\*

- Arquitetura detalhada

- Performance benchmarks

- Monitoring/Alerting

- Disaster recovery

\*\*Para Histórias de Exportação (H13, H19):\*\*

- Formatos exatos

- Validação byte-a-byte

- Performance com grandes volumes

- Compatibilidade

---

\*Versão 2.0 - Documento Completo com todos os elementos essenciais para gestão efetiva de Sprints no projeto FinMath\*

Moisés: Você deve mostrar o documento completo para todas as 24 HUs. Não quero saber de template.

Claude: Vou criar o documento completo com todas as 24 HUs detalhadas (H1-H24, excluindo H25 que estava como futura).