REGRAS CRÍTICAS FINMATH

**Workflow de Desenvolvimento por Sprint**

Versão 2.0 - Completa (24 Histórias)

Outubro 2025

# 🎯 REGRAS CRÍTICAS PRINCIPAIS

## REGRA #1: GITHUB COMO FONTE DA VERDADE (POR SPRINT)

OBRIGATÓRIO: No início de cada sprint, sincronizar com o GitHub.

Durante a sprint, trabalho local com Git local como backup.

# INÍCIO DE CADA SPRINT (obrigatório)  
git fetch origin  
git pull origin main  
git checkout -b sprint-X # Ex: sprint-2  
  
# Durante a sprint: commits locais frequentes  
git add .  
git commit -m "feat(H9): Implementa Price PMT"  
  
# Ao final da sprint: merge e push coletivo

## REGRA #2: VALIDAÇÃO ANTI-REGRESSÃO ANTES DE PUSH FINAL

OBRIGATÓRIO: Ao final da sprint, antes do push para GitHub, validação completa.

# ========================================  
# VALIDAÇÃO FINAL DE SPRINT (OBRIGATÓRIA)  
# ========================================  
echo "🔍 VALIDANDO SPRINT COMPLETA ANTES DE PUSH..."  
  
# 1. Type Check  
npm run type-check || exit 1  
  
# 2. Linting  
npm run lint || exit 1  
  
# 3. Testes Unitários + Propriedade  
npm run test:unit || exit 1  
npm run test:property || exit 1  
  
# 4. Testes de Integração  
npm run test:integration || exit 1  
  
# 5. Golden Files (CRÍTICO)  
npm run golden:verify || exit 1  
  
# 6. Build de Produção  
npm run build || exit 1  
  
# 7. Testes E2E  
npm run test:e2e || exit 1  
  
echo "✅ VALIDAÇÃO COMPLETA - PODE FAZER PUSH!"

## REGRA #3: BACKUP EXCLUSIVO VIA GIT (LOCAL)

PROIBIDO: Criar .bak, .backup, .save no projeto.

# LIMPEZA OBRIGATÓRIA (início de cada dia)  
find . \( -name "\*bak\*" -o -name "\*backup\*" -o -name "\*.save" \) -type f -delete  
  
# ✅ SEMPRE: Git local para backup  
git add .  
git commit -m "WIP: implementando H9"  
  
# ❌ NUNCA: Backups físicos  
# cp price.ts price.ts.bak # PROIBIDO!

## REGRA #4: DOCUMENTAÇÃO VIVA OBRIGATÓRIA

OBRIGATÓRIO: Toda funcionalidade documentada durante implementação.

• JSDoc/TSDoc: Funções públicas do motor

• OpenAPI: Endpoints da API

• ADRs: Decisões arquiteturais importantes

• Swagger UI: Sempre acessível em desenvolvimento

# 🔄 WORKFLOW POR SPRINT

## Fase 1: Início da Sprint

#!/bin/bash  
# inicio-sprint.sh  
  
echo "🚀 INICIANDO SPRINT X"  
  
# 1. Sincronizar com GitHub  
git fetch origin  
git pull origin main || exit 1  
  
# 2. Criar branch da sprint  
SPRINT\_NUM=$1  
git checkout -b "sprint-${SPRINT\_NUM}"  
  
# 3. Limpar backups físicos  
find . \( -name "\*bak\*" -o -name "\*backup\*" \) -type f -delete  
  
# 4. Verificar ambiente  
npm run type-check && npm run lint && npm run test && npm run build  
  
echo "✅ Ambiente pronto para desenvolvimento!"

## Fase 2: Durante a Sprint (Trabalho Local)

# COMMITS LOCAIS FREQUENTES (a cada funcionalidade/história)  
  
# Exemplo: Implementando H9 (Price)  
git add engine/src/amortization/price.ts  
git add engine/test/unit/amortization/price.test.ts  
git commit -m "feat(H9): Implementa Price PMT e cronograma  
  
- Motor: calculatePMT com decimal.js  
- Testes: unit + property + golden file  
- Ajuste final: saldo\_n <= 0.01  
- DoD: 5/5 critérios atendidos"

## Fase 3: Fim da Sprint (Push Coletivo)

#!/bin/bash  
# finalizar-sprint.sh  
  
echo "🏁 FINALIZANDO SPRINT X"  
  
# 1. Validação anti-regressão completa  
npm run type-check || exit 1  
npm run lint || exit 1  
npm run test:unit || exit 1  
npm run test:property || exit 1  
npm run golden:verify || exit 1  
npm run build || exit 1  
  
echo "✅ VALIDAÇÃO COMPLETA - SPRINT APROVADA!"  
  
# 2. Merge na main (local)  
git checkout main  
git merge "sprint-${SPRINT\_NUM}" --no-ff  
  
# 3. PUSH FINAL PARA GITHUB  
git push origin main  
  
echo "🎉 SPRINT X CONCLUÍDA E PUBLICADA!"

# 📋 SPRINTS DETALHADAS - 24 HISTÓRIAS

# 🚀 SPRINT 1 - FUNDAMENTOS DO MOTOR + UI BASE

## (9 Histórias: H1-H8, H20)

### Objetivo da Sprint

Infraestrutura de engenharia completa; cálculos base (Juros, Equivalência, Séries); UI dos simuladores; Explain Panel; Academy com 5 tópicos mínimos.

### Critérios de Saída da Sprint 1

• Simuladores Juros/Equivalência/Séries com P95 ≤ 150 ms

• Explain Panel presente e exportável (snapshot PDF)

• 10 Golden Files publicados (3 juros, 3 equivalências, 4 séries)

• CI/CD estável; logs com calculationId/motorVersion

• Academy: 5 tópicos com exercícios e deep-link para o Lab

# 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

## ⚡ 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

## 🎯 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

## ✔️ Definition of Ready (DoR)

• [x] Stack de CI/CD definida (GitHub Actions)

• [x] Estrutura de monorepo aprovada

• [x] Branch protection rules especificadas

• [x] Requisitos de segurança documentados

## ✅ Definition of Done (DoD)

• [x] Pipeline configurado e testado

• [x] Todos os jobs executando com sucesso

• [x] Artefatos sendo preservados corretamente

• [x] Notificações configuradas

• [x] Documentação do pipeline completa

• [x] Equipe treinada em uso do CI/CD

## 🔗 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:

• ADR-001: Decisão por monorepo

• ADR-009: GitHub Actions vs Jenkins

# H2 - Biblioteca Decimal & Arredondamento

## 📌 Identificação

• ID: H2

• Épico: E1 - Engine & Fundamentos

• Sprint: 1 (Fundamentos)

• Prioridade: P0 (Crítica)

• Pontos: 5

• Responsável: Engine Lead

• Status: ✅ 100% Concluído

## 📝 História de Usuário

Como desenvolvedor do motor de cálculos

Quero uma biblioteca de precisão decimal para todos os cálculos

Para garantir resultados financeiros exatos sem erros de ponto flutuante

## 💼 Contexto de Negócio

### Problema

• JavaScript utiliza IEEE 754 (float64), causando imprecisão

• Erros de centavos acumulam em cálculos complexos

• Conformidade legal exige exatidão até 2 casas decimais

• Arredondamentos inconsistentes geram divergências

### Solução

• Biblioteca decimal.js para todos os cálculos monetários

• Políticas de arredondamento padronizadas (round2, ceil2, floor2)

• Testes de propriedade para validar precisão

• Golden Files com tolerância máxima de R$ 0,01

### Valor de Negócio

• Conformidade: 100% de precisão em cálculos financeiros

• Auditabilidade: Resultados reproduzíveis e verificáveis

• Confiança: Zero erros de arredondamento em produção

• Legal: Compliance com regulamentações do Banco Central

## ✅ Requisitos Funcionais (RF)

### RF2.1 - Biblioteca Decimal

• Utilizar decimal.js para todos os cálculos monetários

• Operações básicas: add, sub, mul, div

• Funções matemáticas: pow, sqrt, ln, exp

• Conversão segura: string ↔ Decimal ↔ number

### RF2.2 - Políticas de Arredondamento

• round2(value): arredonda para 2 casas decimais (HALF\_UP)

• ceil2(value): sempre arredonda para cima

• floor2(value): sempre arredonda para baixo

• round4(value): para taxas intermediárias

### RF2.3 - Validações

• Validar inputs: rejeitar NaN, Infinity, null

• Garantir precisão: todos os valores ≤ ±R$ 0,005 após arredondamento

• Testes de borda: números muito grandes, muito pequenos, zero

## 🎯 Critérios de Aceite (GWT)

### AC1: Precisão de Cálculos

Dado um cálculo financeiro (ex: 10000 × 0.025)

Quando executado com decimal.js

Então o resultado deve ser exato: 250.00

E o erro absoluto vs float64 deve ser zero

### AC2: Arredondamentos Consistentes

Dado valores: [123.456, 123.445, 123.455]

Quando aplicado round2()

Então resultados: [123.46, 123.45, 123.46]

(HALF\_UP: 0.5 arredonda para cima)

## ✔️ Definition of Ready (DoR)

• [x] Biblioteca decimal.js selecionada e aprovada

• [x] Políticas de arredondamento definidas

• [x] Tolerâncias para Golden Files especificadas (≤ R$ 0,01)

## ✅ Definition of Done (DoD)

• [x] Biblioteca decimal.js instalada e configurada

• [x] Funções utilitárias criadas (round2, ceil2, floor2, round4)

• [x] Testes unitários: 100% de cobertura

• [x] Testes de propriedade: associatividade, comutatividade

• [x] Golden Files: erro absoluto ≤ R$ 0,01

• [x] ADR-001: Decisão por decimal.js documentada

## 🔗 Relacionamentos

### Depende de:

• H1: CI/CD (para executar testes)

### Bloqueia:

• H4-H19: Todos os cálculos dependem de precisão decimal

• H9: Price (PMT precisa de decimal.js)

• H12: CET (cálculo de IRR depende de precisão)

### Relacionado a:

• ADR-001: Decisão por decimal.js vs BigNumber.js

• Playbook de Testes: Tolerâncias de Golden Files

# H3 - Observabilidade Básica

## 📌 Identificação

• ID: H3

• Épico: E4 - Infraestrutura & Observabilidade

• Sprint: 1 (Fundamentos)

• Prioridade: P0 (Crítica)

• Pontos: 3

• Responsável: DevOps/Backend Lead

• Status: ⚠️ 75% Concluído

## 📝 História de Usuário

Como operador de sistema

Quero logs estruturados com correlation ID

Para rastrear requisições end-to-end e diagnosticar problemas

## 💼 Contexto de Negócio

### Problema

• Logs não estruturados dificultam debugging

• Impossível correlacionar requisições entre serviços

• Falta de visibilidade em produção

• Troubleshooting lento e custoso

### Solução

• Logs estruturados em JSON

• Correlation ID em todas as requisições

• Dashboards operacionais em tempo real

• Alertas automáticos em erros

### Valor de Negócio

• Redução de 60% no tempo de troubleshooting

• MTTR (Mean Time to Resolution) < 30min

• Visibilidade completa do sistema

• Compliance com requisitos de auditoria

## ✅ Requisitos Funcionais

• Logs estruturados (JSON) com campos obrigatórios

• Correlation ID (calculationId) gerado/propagado

• Níveis de log: debug, info, warn, error

• motorVersion em todos os logs

• Timestamp UTC em formato ISO 8601

## ⚡ Requisitos Não Funcionais

• Latência adicional < 1ms por log

• Buffer para logs em burst (100K logs/s)

• Graceful degradation se sistema de logs falhar

• Rotação automática de logs

## 🎯 Critérios de Aceite (GWT)

AC1: Todo request gera calculationId único

AC2: Correlation ID propagado em toda a pilha

AC3: Logs pesquisáveis por calculationId, motorVersion, timestamp

AC4: Dashboards exibem: request rate, error rate, latency (P50/P95/P99)

## ✔️ Definition of Ready (DoR)

• [x] Stack de logging selecionada (Winston/Pino)

• [x] Formato de log estruturado definido

• [x] Campos obrigatórios especificados

• [x] Política de retenção definida

## ✅ Definition of Done (DoD)

• [x] Logger configurado e testado

• [x] Correlation ID middleware implementado

• [x] Logs funcionando em dev/staging/prod

• [x] Dashboards básicos criados

• [x] Alertas configurados para erros críticos

• [x] Documentação de observabilidade completa

## 🔗 Relacionamentos

### Depende de:

• H1: CI/CD (para deploy de infraestrutura)

### Bloqueia:

• H9-H24: Todas precisam de observabilidade

• H21: Snapshots (usa calculationId)

• H22: Validador (precisa de logs)

### Relacionado a:

• ADR-003: Estrutura de logs

• Playbook: Observabilidade mínima

# H4 - Juros Compostos FV/PV

## 📌 Identificação

• ID: H4

• Épico: E1 - Engine & Fundamentos

• Sprint: 1 (Fundamentos)

• Prioridade: P1 (Alta)

• Pontos: 3

• Responsável: Engine Developer

• Status: ✅ 100% Concluído

## 📝 História de Usuário

Como usuário final

Quero calcular valor futuro (FV) e presente (PV) com juros compostos

Para projetar investimentos e empréstimos

## 💼 Contexto de Negócio

### Problema

• Cálculos manuais propensos a erros

• Fórmulas de juros compostos complexas

• Necessidade de validação de resultados

• Casos extremos (n=1, i=0) problemáticos

### Solução

• Funções calculat FV e PV precisas

• Validação de casos extremos

• Testes de propriedade matemática

• Golden Files para casos padrão

### Valor de Negócio

• Cálculos financeiros básicos confiáveis

• Base para todas as operações complexas

• Conformidade com padrões matemáticos

• Redução de erros humanos

## ✅ Requisitos Funcionais

• FV = PV × (1 + i)^n com decimal.js

• PV = FV / (1 + i)^n

• Validação: i ≥ 0, n ≥ 0, PV > 0, FV > 0

• Casos extremos: n=0 → FV=PV, i=0 → FV=PV

• Suporte a taxas negativas (opcional)

## ⚡ Requisitos Não Funcionais

• Precisão: erro absoluto ≤ R$ 0,01

• Performance: cálculo < 10ms

• Monotonicidade: ∂FV/∂i > 0, ∂FV/∂n > 0

• Estabilidade numérica para n > 360

## 🎯 Critérios de Aceite (GWT)

AC1: FV(10000, 0.01, 12) = 11268.25 (±0.01)

AC2: PV(20000, 0.015, 24) = 14185.19 (±0.01)

AC3: Caso extremo: FV(X, 0, n) = X

AC4: Caso extremo: FV(X, i, 0) = X

## ✔️ Definition of Ready (DoR)

• [x] Decimal.js configurado (H2)

• [x] Fórmulas matemáticas validadas

• [x] Casos de teste definidos

• [x] 3 Golden Files planejados

## ✅ Definition of Done (DoD)

• [x] Funções calculateFV e calculatePV implementadas

• [x] 3 Golden Files passando (FV\_001, FV\_002, PV\_001)

• [x] Testes unitários: 100% cobertura

• [x] Testes de propriedade: monotonicidade

• [x] Endpoint /api/interest documentado

• [x] JSDoc completo

## 🔗 Relacionamentos

### Depende de:

• H2: Decimal & Arredondamento

• H1: CI/CD (testes)

### Bloqueia:

• H5: Equivalência de Taxas

• H6: Séries/Anuidades

• H7: Simulador de Juros

### Relacionado a:

• Contratos de API: POST /api/interest

• Golden Files: INTEREST\_001 a INTEREST\_003

# H5 - Equivalência de Taxas + Taxa Real

## 📌 Identificação

• ID: H5

• Épico: E1 - Engine & Fundamentos

• Sprint: 1 (Fundamentos)

• Prioridade: P1 (Alta)

• Pontos: 3

• Responsável: Engine Developer

• Status: ✅ 100% Concluído

## 📝 História de Usuário

Como analista financeiro

Quero converter taxas entre períodos (mensal ↔ anual) e calcular taxa real

Para comparar ofertas e ajustar por inflação

## 💼 Contexto de Negócio

### Problema

• Taxas em períodos diferentes não são comparáveis

• Inflação distorce retorno real

• Conversões manuais propensas a erro

• Falta de padronização

### Solução

• Conversão automática entre períodos

• Cálculo de taxa real (Fisher)

• Validação matemática

• Interface simplificada

### Valor de Negócio

• Comparação precisa entre produtos financeiros

• Análise real de retorno considerando inflação

• Conformidade com convenções de mercado

• Transparência em propostas

## ✅ Requisitos Funcionais

• Taxa equivalente: ieq = (1 + i)^(p2/p1) - 1

• Mensal → Anual: iaa = (1 + im)^12 - 1

• Anual → Mensal: im = (1 + iaa)^(1/12) - 1

• Taxa real (Fisher): ireal = (1 + inom)/(1 + iinfl) - 1

• Suporte a qualquer período (diário, semanal, mensal, anual)

## ⚡ Requisitos Não Funcionais

• Precisão: taxas ± 0.01 p.p.

• Performance: < 5ms por conversão

• Reversibilidade: mensal→anual→mensal deve retornar valor original

• Validação: 0 ≤ i ≤ 1 (100% a.a.)

## 🎯 Critérios de Aceite (GWT)

AC1: 1% a.m. = 12.68% a.a. (±0.01 p.p.)

AC2: 15% a.a. = 1.17% a.m. (±0.01 p.p.)

AC3: Taxa Real: inom=10%, infl=5% → ireal=4.76%

AC4: Reversibilidade: converter ida e volta retorna original

## ✔️ Definition of Ready (DoR)

• [x] Fórmulas de equivalência validadas

• [x] Casos de teste com taxas conhecidas

• [x] 3 Golden Files planejados

• [x] ADR sobre convenções de taxas

## ✅ Definition of Done (DoD)

• [x] Funções convertRate e realRate implementadas

• [x] 3 Golden Files passando (RATE\_001, RATE\_002, RATE\_003)

• [x] Testes de reversibilidade

• [x] Endpoint /api/rate documentado

• [x] Suporte a períodos customizados

• [x] Documentação de fórmulas

## 🔗 Relacionamentos

### Depende de:

• H4: Juros Compostos

• H2: Precisão Decimal

### Bloqueia:

• H7: Simulador de Equivalência

• H12: CET (conversão de taxas)

### Relacionado a:

• Contratos de API: POST /api/rate

• Golden Files: RATE\_001 a RATE\_003

# H6 - Séries/Anuidades

## 📌 Identificação

• ID: H6

• Épico: E1 - Engine & Fundamentos

• Sprint: 1 (Fundamentos)

• Prioridade: P1 (Alta)

• Pontos: 5

• Responsável: Engine Developer

• Status: ✅ 100% Concluído

## 📝 História de Usuário

Como planejador financeiro

Quero calcular séries postecipadas e antecipadas

Para dimensionar poupanças e aposentadorias

## 💼 Contexto de Negócio

### Problema

• Cálculo de poupança progressiva complexo

• Diferença entre postecipada e antecipada confusa

• Necessidade de resolver para n ou i (inversão)

• Validação de casos extremos

### Solução

• Séries postecipadas e antecipadas

• Fórmulas diretas e inversas

• Casos extremos tratados

• Validação matemática

### Valor de Negócio

• Planejamento de aposentadoria preciso

• Metas de poupança realistas

• Análise de viabilidade financeira

• Conformidade matemática

## ✅ Requisitos Funcionais

• Série Postecipada: FV = PMT × ((1+i)^n - 1) / i

• Série Antecipada: FV = PMT × ((1+i)^n - 1) / i × (1+i)

• PV de série postecipada: PV = PMT × (1 - (1+i)^-n) / i

• PV de série antecipada: PV = PMT × (1 - (1+i)^-n) / i × (1+i)

• Inversão: resolver para n (busca binária) ou i (Newton-Raphson)

## ⚡ Requisitos Não Funcionais

• Precisão: ± R$ 0,01

• Performance: < 20ms

• Monotonicidade garantida

• Convergência em inversão: máx 50 iterações

## 🎯 Critérios de Aceite (GWT)

AC1: Série postecipada: PMT=1000, i=1%, n=12 → FV=12682.50

AC2: Série antecipada: PMT=1000, i=1%, n=12 → FV=12809.33

AC3: Inversão: FV=20000, PMT=1000, i=1% → n≈18

AC4: Caso extremo: i=0 → FV=PMT×n

## ✔️ Definition of Ready (DoR)

• [x] Fórmulas de séries validadas

• [x] Algoritmos de inversão definidos

• [x] 4 Golden Files planejados

• [x] Testes de convergência

## ✅ Definition of Done (DoD)

• [x] Funções calculateSeries implementadas (post/ante)

• [x] Inversão: solveSeries (para n e i)

• [x] 4 Golden Files passando (SERIES\_001 a SERIES\_004)

• [x] Testes de propriedade

• [x] Endpoint /api/series documentado

• [x] Casos extremos tratados

## 🔗 Relacionamentos

### Depende de:

• H4: Juros Compostos

• H2: Precisão Decimal

### Bloqueia:

• H7: Simulador de Séries

• H9: Price (séries de pagamentos)

### Relacionado a:

• Contratos de API: POST /api/series

• Golden Files: SERIES\_001 a SERIES\_004

• ADR sobre algoritmos de inversão

# H7 - Simuladores Base (UI)

## 📌 Identificação

• ID: H7

• Épico: E3 - UI/UX & Exportações

• Sprint: 1 (Fundamentos)

• Prioridade: P1 (Alta)

• Pontos: 8

• Responsável: Frontend Developer

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como usuário final

Quero interfaces interativas para Juros, Equivalência e Séries

Para experimentar cenários financeiros de forma intuitiva

## 💼 Contexto de Negócio

### Problema

• Fórmulas complexas intimidam usuários

• Necessidade de interface amigável

• Validação em tempo real essencial

• Resultados devem ser claros e acionáveis

### Solução

• 3 simuladores interativos com validação

• Design System aplicado consistentemente

• Máscaras de input intuitivas

• Feedback visual em tempo real

### Valor de Negócio

• Adoção de 80% dos usuários

• Redução de 90% em suporte

• NPS > 8.0

• Tempo médio de uso: 3-5 min

## ✅ Requisitos Funcionais

• Simulador de Juros: FV/PV com inputs PV, i, n

• Simulador de Equivalência: conversão de taxas

• Simulador de Séries: post/ante com inputs PMT, i, n

• Validação inline com mensagens claras

• Máscaras: moeda (R$), percentual (%), períodos

• Botão "Calcular" destacado

• Resultados em cards com ícones

## ⚡ Requisitos Não Funcionais

• Performance: P95 ≤ 150ms

• Mobile-first design

• Acessibilidade: navegação por teclado, contraste WCAG AA

• Responsivo: 320px a 2560px

• Dark/Light theme

## 🎯 Critérios de Aceite (GWT)

AC1: Todos os inputs com validação inline

AC2: Cálculo em < 150ms (P95)

AC3: Resultados exibidos em cards estilizados

AC4: Navegação por teclado funcional

AC5: Design System 100% aplicado

## ✔️ Definition of Ready (DoR)

• [x] Design System v1.0 aprovado

• [x] Componentes UI prontos (Button, Input, Card)

• [x] Wireframes de simuladores validados

• [x] Endpoints de API disponíveis (H4, H5, H6)

## ✅ Definition of Done (DoD)

• [x] 3 simuladores implementados e funcionais

• [x] Design System 100% aplicado

• [x] Validação inline implementada

• [x] Máscaras de input funcionando

• [x] Testes E2E básicos

• [x] Performance P95 < 150ms validada

• [x] Acessibilidade básica (navegação)

## 🔗 Relacionamentos

### Depende de:

• H4: Juros Compostos (API)

• H5: Equivalência (API)

• H6: Séries (API)

• Design System v1.0

### Bloqueia:

• H8: Explain Panel (usa simuladores)

• H20: Academy (links para Lab)

### Relacionado a:

• Design System / UI Kit v1.0

• Contratos de API

• ADR-006: Escolha de framework UI (React)

# H8 - Explain Panel + Exportar PDF

## 📌 Identificação

• ID: H8

• Épico: E3 - UI/UX & Exportações

• Sprint: 1 (Fundamentos)

• Prioridade: P2 (Média)

• Pontos: 5

• Responsável: Frontend + Backend

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como usuário educacional

Quero ver passo a passo dos cálculos com fórmulas

Para aprender e validar resultados

## 💼 Contexto de Negócio

### Problema

• Cálculos parecem "caixa preta"

• Usuários querem entender "por quê"

• Necessidade educacional e de auditoria

• Exportação para compartilhamento

### Solução

• Painel explicativo com fórmulas e números

• Exibição passo a passo

• Exportação para PDF

• Linguagem acessível

### Valor de Negócio

• Transparência completa

• Ferramenta educacional

• Auditabilidade

• Compartilhamento facilitado

## ✅ Requisitos Funcionais

• Botão "Como calculamos?" em cada simulador

• Painel com: fórmula LaTeX, valores substituídos, passo a passo

• motorVersion exibido no Explain

• Exportar PDF mantém formatação

• PDF inclui: logo, data, calculationId, motorVersion

## ⚡ Requisitos Não Funcionais

• Renderização de LaTeX (MathJax/KaTeX)

• Geração de PDF server-side

• PDF < 2s para gerar

• Tamanho PDF < 500KB

## 🎯 Critérios de Aceite (GWT)

AC1: Explain abre ao clicar "Como calculamos?"

AC2: Fórmulas LaTeX renderizadas corretamente

AC3: PDF gerado em < 2s

AC4: PDF contém todas as informações obrigatórias

AC5: motorVersion visível no PDF

## ✔️ Definition of Ready (DoR)

• [x] Simuladores (H7) implementados

• [x] Biblioteca LaTeX escolhida

• [x] Biblioteca PDF server-side escolhida

• [x] Template de PDF desenhado

## ✅ Definition of Done (DoD)

• [x] Explain Panel implementado

• [x] Renderização LaTeX funcional

• [x] Exportação PDF funcional

• [x] PDF com motorVersion

• [x] Testes de geração de PDF

• [x] Documentação do Explain completa

## 🔗 Relacionamentos

### Depende de:

• H7: Simuladores Base

• H3: Observabilidade (motorVersion)

### Bloqueia:

• H13: Exportações (aproveita engine PDF)

• H20: Academy (usa Explain como exemplo)

### Relacionado a:

• Playbook §7.2: Snapshots e Explain

• Design System: componentes de Explain

# H20 - Academy: 5 Tópicos

## 📌 Identificação

• ID: H20

• Épico: E5 - Academy & Mercado

• Sprint: 1 (Fundamentos)

• Prioridade: P2 (Média)

• Pontos: 8

• Responsável: Content Creator + Frontend

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como estudante ou profissional

Quero conteúdo educacional sobre matemática financeira

Para aprender conceitos e praticar no Lab

## 💼 Contexto de Negócio

### Problema

• Falta de material didático integrado

• Usuários não sabem usar as ferramentas

• Necessidade de onboarding

• Oportunidade de branding educacional

### Solução

• 5 tópicos educacionais completos

• Cada tópico com: teoria resumida, exemplo numérico, exercício guiado

• Deep-link "Abrir no Lab" em cada exemplo

• Telemetria de conclusão

### Valor de Negócio

• Educação financeira acessível

• Onboarding efetivo (90% completam pelo menos 1 tópico)

• Branding como ferramenta educacional

• Redução de 70% em dúvidas de suporte

## ✅ Requisitos Funcionais

• 5 tópicos mínimos: 1) Juros Compostos, 2) Equivalência de Taxas, 3) Séries e Anuidades, 4) Sistemas de Amortização, 5) CET Básico

• Cada tópico: 300-500 palavras de teoria

• Exemplo numérico passo a passo

• Exercício guiado com resposta

• Botão "Abrir no Lab" leva ao simulador com valores pré-preenchidos

• Progress bar de conclusão

## ⚡ Requisitos Não Funcionais

• Telemetria: tracks de view, time on page, clicks em "Abrir no Lab"

• SEO otimizado para cada tópico

• Tempo de carregamento < 2s

• Mobile-friendly

## 🎯 Critérios de Aceite (GWT)

AC1: 5 tópicos publicados e acessíveis

AC2: Deep-links funcionam corretamente

AC3: Telemetria capturando eventos

AC4: Progress bar atualiza em tempo real

AC5: Mobile UX validada

## ✔️ Definition of Ready (DoR)

• [x] Simuladores (H7) implementados

• [x] Conteúdo dos 5 tópicos revisado e aprovado

• [x] Sistema de telemetria configurado

• [x] Design de páginas Academy aprovado

## ✅ Definition of Done (DoD)

• [x] 5 tópicos publicados

• [x] Deep-links implementados e testados

• [x] Telemetria funcional

• [x] Progress bar implementada

• [x] SEO básico aplicado

• [x] Testes E2E de navegação

## 🔗 Relacionamentos

### Depende de:

• H7: Simuladores (deep-links apontam para eles)

### Bloqueia:

• H23: Casos de Mercado (podem ser linkados no Academy)

### Relacionado a:

• Design System: páginas de conteúdo

• Contratos de API: GET /academy/:topicId

# 🚀 SPRINT 2 - AMORTIZAÇÕES + CET BÁSICO + EXPORTAÇÕES

## (7 Histórias: H9-H13, H21-H22)

### Objetivo da Sprint

Price e SAC completos (day count + pró-rata), CET básico (tarifas t0), CSV/PDF, snapshots e validador inicial.

### Critérios de Saída da Sprint 2

• Price/SAC com deltas ≤ R$ 0,01 vs Golden Files

• Day count (30/360, ACT/365) correto + pró-rata opcional

• CET básico (IRR mensal → CET a.a., sem IOF/seguros) com erro ≤ 0,01 p.p. + Explain

• CSV/PDF exportáveis com cronogramas completos

• Validador /api/validate/schedule operacional com diffs por coluna

• 10 Golden Files passando (5 Price + 5 SAC)

# H9 - Price: PMT, Cronograma, Ajuste Final

## 📌 Identificação

• ID: H9

• Épico: E1 - Engine & Fundamentos

• Sprint: 2 (Amortizações)

• Prioridade: P0 (Crítica)

• Pontos: 8

• Responsável: Engine Developer

• Status: ⚠️ 50% - Motor OK, API Pendente

## 📝 História de Usuário

Como analista de crédito

Quero calcular Price com cronograma detalhado e ajuste final

Para gerar propostas precisas de financiamento

## 💼 Contexto de Negócio

### Problema

• Price é o sistema de amortização mais usado

• Ajuste final complexo (resíduo de centavos)

• Day count afeta primeira parcela

• Clientes exigem precisão

### Solução

• Motor Price com decimal.js

• Ajuste final na última parcela (saldo ≤ R$ 0,01)

• Suporte a day count 30/360 e ACT/365

• Pró-rata opcional na primeira parcela

### Valor de Negócio

• Propostas comerciais precisas

• Conformidade com práticas de mercado

• Redução de 99% em ajustes manuais

• Satisfação do cliente

## ✅ Requisitos Funcionais

• PMT = PV × i × (1+i)^n / ((1+i)^n - 1)

• Cronograma: n linhas com saldo, juros, amortização, PMT

• Ajuste final: adicionar resíduo na última amortização

• Saldo final deve ser ≤ R$ 0,01 (arredondamento aceitável)

• Day count: prorata = dias\_corridos / dias\_padrao

• Primeira parcela: juros\_1 = saldo\_0 × i × prorata

## ⚡ Requisitos Não Funcionais

• Precisão: deltas ≤ R$ 0,01

• Performance: cronograma até 360 parcelas em < 100ms

• Monotonicidade: saldo\_n decrescente

• Propriedade: soma(amortização) = PV (±0.01)

## 🎯 Critérios de Aceite (GWT)

AC1: PMT = calculatePMT(10000, 0.025, 12) = 946.56 (±0.01)

AC2: Saldo final ≤ R$ 0,01

AC3: Com pró-rata: primeira parcela tem juros ajustados

AC4: Golden Files: PRICE\_001 a PRICE\_005 passam

## ✔️ Definition of Ready (DoR)

• [x] H2: Decimal.js

• [x] H10: Day Count (para pró-rata)

• [x] Playbook: Tolerâncias de ajuste final

• [x] 5 Golden Files preparados

## ✅ Definition of Done (DoD)

• [x] Motor calculatePMT e generateSchedule implementados

• [x] Ajuste final funcional

• [x] Suporte a pró-rata

• [x] 5 Golden Files passando

• [x] API /api/price implementada e documentada

• [x] Testes de propriedade

• [x] JSDoc completo

## 🔗 Relacionamentos

### Depende de:

• H2: Precisão Decimal

• H10: Day Count (pró-rata)

### Bloqueia:

• H13: Exportações CSV/PDF

• H22: Validador (upload de cronograma Price)

### Relacionado a:

• Guia CET - SoT §4: Ajuste final

• Contratos API: POST /api/price

• Golden Files: PRICE\_001 a PRICE\_005

# H10 - Day Count 30/360 e ACT/365 + Pró-rata

## 📌 Identificação

• ID: H10

• Épico: E1 - Engine & Fundamentos

• Sprint: 2 (Amortizações)

• Prioridade: P1 (Alta)

• Pontos: 5

• Responsável: Engine Developer

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como analista de crédito

Quero aplicar convenções de day count (30/360, ACT/365)

Para calcular juros da primeira parcela corretamente

## 💼 Contexto de Negócio

### Problema

• Meses têm diferentes números de dias

• Primeira parcela pode ser desproporcional

• Convenções de mercado variam

• Clientes questionam cálculos

### Solução

• Suporte a 30/360 e ACT/365

• Cálculo de pró-rata para primeira parcela

• Validação com casos reais

• Documentação clara das convenções

### Valor de Negócio

• Conformidade com práticas de mercado

• Transparência nos cálculos

• Flexibilidade para diferentes produtos

• Redução de questionamentos

## ✅ Requisitos Funcionais

• 30/360: todos os meses têm 30 dias, ano 360 dias

• ACT/365: dias corridos reais, ano 365 dias (ou 366 em bissexto)

• Pró-rata primeira parcela: fator = dias / dias\_convenção

• juros\_parcela\_1 = saldo × i × fator

• Suporte a datas customizadas (não apenas 1º do mês)

## ⚡ Requisitos Não Funcionais

• Precisão: deltas ≤ R$ 0,01

• Performance: cálculo de day count < 1ms

• Validação: 1 ≤ dias ≤ 365

• Suporte a anos bissextos

## 🎯 Critérios de Aceite (GWT)

AC1: 30/360: 20 dias = fator 0.6667 (20/30)

AC2: ACT/365: 25 dias = fator 0.0685 (25/365)

AC3: Primeira parcela Price com pró-rata tem juros ajustados

AC4: Ano bissexto (366 dias) tratado corretamente em ACT/365

## ✔️ Definition of Ready (DoR)

• [x] Convenções de day count estudadas

• [x] Casos de teste com datas reais preparados

• [x] Playbook: exemplos de day count

• [x] ADR sobre escolha de convenções

## ✅ Definition of Done (DoD)

• [x] Funções dayCount30\_360 e dayCountACT\_365 implementadas

• [x] Cálculo de pró-rata integrado no Price/SAC

• [x] Testes com anos bissextos

• [x] Testes de borda (31 de mês, fevereiro)

• [x] Documentação das convenções

• [x] ADR-005 criado

## 🔗 Relacionamentos

### Depende de:

• H2: Precisão Decimal

### Bloqueia:

• H9: Price (usa pró-rata)

• H11: SAC (usa pró-rata)

### Relacionado a:

• ADR-005: Convenções de day count

• Guia CET - SoT: day count em CET

# H11 - SAC: Cronograma com Ajuste Final

## 📌 Identificação

• ID: H11

• Épico: E1 - Engine & Fundamentos

• Sprint: 2 (Amortizações)

• Prioridade: P1 (Alta)

• Pontos: 5

• Responsável: Engine Developer

• Status: ⚠️ 50% - Motor OK, API Pendente

## 📝 História de Usuário

Como analista de crédito

Quero calcular SAC com cronograma e ajuste final

Para oferecer alternativa ao Price com parcelas decrescentes

## 💼 Contexto de Negócio

### Problema

• SAC tem amortização constante

• PMT decrescente confunde clientes

• Ajuste final necessário

• Comparação com Price essencial

### Solução

• Motor SAC com amortização constante

• Cronograma com PMT decrescente

• Ajuste final na última parcela

• Validação de propriedades matemáticas

### Valor de Negócio

• Alternativa comercial ao Price

• Parcelas decrescentes atraem certos perfis

• Conformidade com práticas de mercado

• Total de juros geralmente menor que Price

## ✅ Requisitos Funcionais

• Amortização constante: A = PV / n

• PMT\_k = A + juros\_k

• juros\_k = saldo\_k-1 × i

• Saldo\_k = saldo\_k-1 - A

• Ajuste final: adicionar resíduo na última amortização

• Saldo final ≤ R$ 0,01

## ⚡ Requisitos Não Funcionais

• Precisão: deltas ≤ R$ 0,01

• Performance: cronograma até 360 parcelas em < 100ms

• Monotonicidade: PMT estritamente decrescente

• Propriedade: soma(juros\_SAC) < soma(juros\_Price)

## 🎯 Critérios de Aceite (GWT)

AC1: Amortização constante verificada

AC2: PMT estritamente decrescente

AC3: Saldo final ≤ R$ 0,01

AC4: Golden Files: SAC\_001 a SAC\_005 passam

AC5: Primeira parcela com pró-rata (se aplicável)

## ✔️ Definition of Ready (DoR)

• [x] H2: Decimal.js

• [x] H10: Day Count (pró-rata)

• [x] 5 Golden Files preparados

• [x] Fórmulas SAC validadas

## ✅ Definition of Done (DoD)

• [x] Motor calculateSAC e generateSchedule implementados

• [x] Ajuste final funcional

• [x] Suporte a pró-rata

• [x] 5 Golden Files passando

• [x] API /api/sac implementada e documentada

• [x] Testes de propriedade (PMT decrescente)

• [x] JSDoc completo

## 🔗 Relacionamentos

### Depende de:

• H2: Precisão Decimal

• H10: Day Count (pró-rata)

### Bloqueia:

• H13: Exportações CSV/PDF

• H18: Comparador (compara Price vs SAC)

• H22: Validador

### Relacionado a:

• Contratos API: POST /api/sac

• Golden Files: SAC\_001 a SAC\_005

• Playbook: SAC vs Price

# H12 - CET Básico (apenas tarifas t0)

## 📌 Identificação

• ID: H12

• Épico: E2 - CET & Perfis

• Sprint: 2 (Amortizações)

• Prioridade: P0 (Crítica)

• Pontos: 8

• Responsável: Engine + Backend

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como regulador/cliente

Quero calcular CET básico (apenas tarifas em t0)

Para comparar custo efetivo entre propostas

## 💼 Contexto de Negócio

### Problema

• CET é obrigatório por lei

• Cálculo complexo (IRR de fluxos)

• Tarifas em t0 são comuns

• Clientes não entendem diferença entre taxa nominal e CET

### Solução

• CET básico: apenas tarifas desembolsadas em t0

• IRR mensal → conversão para CET a.a.

• Validação com Golden Files

• Explain Panel mostrando fluxos

### Valor de Negócio

• Conformidade regulatória (Banco Central)

• Transparência total

• Comparabilidade entre ofertas

• Educação do cliente

## ✅ Requisitos Funcionais

• Fluxos: t0 = PV - tarifas, t1..tn = -PMT

• IRR mensal via Newton-Raphson

• CET a.a. = (1 + IRR\_mensal)^12 - 1

• Tarifas: TAC, seguro prestamista em t0

• Erro máximo: ≤ 0,01 p.p. a.a.

• Máximo 100 iterações para convergência

## ⚡ Requisitos Não Funcionais

• Precisão: CET ± 0,01 p.p. a.a.

• Performance: IRR em < 50ms

• Convergência: 99% dos casos em < 50 iterações

• Robustez: detectar casos sem solução

## 🎯 Critérios de Aceite (GWT)

AC1: CET básico calculado corretamente vs Golden Files

AC2: Erro ≤ 0,01 p.p. a.a.

AC3: Explain mostra fluxos de caixa

AC4: Casos extremos: sem tarifas → CET = taxa nominal

AC5: Convergência ou erro informativo

## ✔️ Definition of Ready (DoR)

• [x] H9 ou H11: Cronograma disponível

• [x] H15: IRR (pode reaproveitar algoritmo, mas H12 não precisa de Brent)

• [x] Guia CET - SoT §2-4 estudado

• [x] Golden Files de CET preparados

## ✅ Definition of Done (DoD)

• [x] Motor calculateCET\_basic implementado

• [x] IRR mensal com Newton-Raphson

• [x] Conversão para CET a.a.

• [x] API /api/cet/basic implementada

• [x] Explain Panel de CET funcional

• [x] Testes com Golden Files

• [x] Documentação de metodologia CET

## 🔗 Relacionamentos

### Depende de:

• H9: Price (gera fluxos)

• H11: SAC (gera fluxos)

### Bloqueia:

• H16: CET Completo (adiciona IOF + seguros)

• H17: Perfis CET

• H18: Comparador

### Relacionado a:

• Guia CET - SoT v1.0 (§2-4, §8-9)

• Contratos API: POST /api/cet/basic

• ADR-008: IRR para CET

# H13 - Exportações CSV/PDF de Cronogramas

## 📌 Identificação

• ID: H13

• Épico: E3 - UI/UX & Exportações

• Sprint: 2 (Amortizações)

• Prioridade: P1 (Alta)

• Pontos: 5

• Responsável: Backend + Frontend

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como usuário final

Quero exportar cronogramas para CSV e PDF

Para análise offline e compartilhamento

## 💼 Contexto de Negócio

### Problema

• Usuários precisam de dados em planilhas

• Compartilhamento com colegas/clientes

• Auditoria offline

• Impressão de propostas

### Solução

• Exportação CSV com colunas padronizadas

• Exportação PDF estilizado

• Mantém precisão (2 casas decimais)

• Inclui motorVersion e calculationId

### Valor de Negócio

• Flexibilidade de uso

• Auditabilidade

• Profissionalismo em propostas

• Integração com Excel/Sheets

## ✅ Requisitos Funcionais

• CSV: colunas "n,data,saldo,juros,amortização,PMT"

• CSV mantém precisão de 2 casas

• PDF: tabela estilizada + logo + motorVersion

• PDF: cabeçalho com resumo (PV, i, n, sistema)

• Endpoints: GET /reports/schedule.csv, GET /reports/schedule.pdf

## ⚡ Requisitos Não Funcionais

• Performance: CSV em < 500ms, PDF em < 2s

• Tamanho PDF < 1MB

• CSV compatível com Excel/Sheets

• PDF mantém formatação em impressão

## 🎯 Critérios de Aceite (GWT)

AC1: CSV gerado com colunas corretas

AC2: Valores em CSV idênticos aos da UI (±0.01)

AC3: PDF gerado com tabela legível

AC4: PDF contém motorVersion e calculationId

AC5: Tempo de geração dentro do SLA

## ✔️ Definition of Ready (DoR)

• [x] H9: Price cronograma

• [x] H11: SAC cronograma

• [x] Biblioteca CSV escolhida

• [x] Biblioteca PDF server-side escolhida

## ✅ Definition of Done (DoD)

• [x] Endpoint /reports/schedule.csv implementado

• [x] Endpoint /reports/schedule.pdf implementado

• [x] Testes de precisão CSV/PDF vs motor

• [x] Testes de performance

• [x] Documentação de endpoints

• [x] Design de template PDF aprovado

## 🔗 Relacionamentos

### Depende de:

• H9: Price (cronograma)

• H11: SAC (cronograma)

### Bloqueia:

• H19: XLSX com fórmulas (complementa)

### Relacionado a:

• Contratos API: GET /reports/\*

• ADR-007: Bibliotecas de exportação

• Design System: template PDF

# H21 - Snapshots com hash e motorVersion

## 📌 Identificação

• ID: H21

• Épico: E4 - Infraestrutura & Observabilidade

• Sprint: 2 (Amortizações)

• Prioridade: P2 (Média)

• Pontos: 3

• Responsável: Backend Developer

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como auditor/desenvolvedor

Quero snapshots persistidos com hash e motorVersion

Para reproduzir cálculos e detectar regressões

## 💼 Contexto de Negócio

### Problema

• Cálculos precisam ser reproduzíveis

• Mudanças no motor podem causar divergências

• Auditoria exige rastreabilidade

• Debugging requer histórico

### Solução

• Snapshots salvam: inputs, outputs, hash, motorVersion

• Reprocessamento reproduz CSV idêntico

• Hash detecta mudanças

• motorVersion rastreia evolução

### Valor de Negócio

• Reprodutibilidade 100%

• Auditoria facilitada

• Debugging acelerado

• Conformidade regulatória

## ✅ Requisitos Funcionais

• Snapshot = { inputs, outputs, hash: SHA256(inputs+outputs), motorVersion, timestamp, calculationId }

• POST /snapshots com saveSnapshot=true

• GET /snapshots/:id retorna snapshot completo

• Reprocesso: POST /snapshots/:id/reprocess gera novo CSV

• Validação: hash e CSV devem ser idênticos

## ⚡ Requisitos Não Funcionais

• Performance: salvar snapshot < 10ms

• Storage: max 10MB/snapshot

• Retenção: 90 dias (quente), 1 ano (frio)

• Índice por calculationId e motorVersion

## 🎯 Critérios de Aceite (GWT)

AC1: Snapshot salvo com todos os campos

AC2: Reprocesso gera CSV idêntico (hash igual)

AC3: motorVersion registrado corretamente

AC4: GET /snapshots/:id retorna snapshot completo

AC5: Busca por motorVersion funciona

## ✔️ Definition of Ready (DoR)

• [x] H9/H11: Cronogramas disponíveis

• [x] H3: Observabilidade (calculationId, motorVersion)

• [x] Banco de dados escolhido

• [x] Playbook §9 estudado

## ✅ Definition of Done (DoD)

• [x] Modelo de dados Snapshot definido

• [x] POST /snapshots implementado

• [x] GET /snapshots/:id implementado

• [x] POST /snapshots/:id/reprocess implementado

• [x] Testes de hash e reprodutibilidade

• [x] Documentação de endpoints

## 🔗 Relacionamentos

### Depende de:

• H3: Observabilidade (motorVersion, calculationId)

• H9/H11: Cronogramas

### Bloqueia:

• H22: Validador (usa snapshots para comparação)

### Relacionado a:

• Playbook §9: Snapshots

• Contratos API: POST/GET /snapshots

# H22 - Validador de Cronogramas (Upload CSV)

## 📌 Identificação

• ID: H22

• Épico: E4 - Infraestrutura & Observabilidade

• Sprint: 2 (Amortizações)

• Prioridade: P2 (Média)

• Pontos: 5

• Responsável: Backend Developer

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como analista/auditor

Quero fazer upload de CSV externo e validar contra motor

Para detectar divergências e auditar cálculos

## 💼 Contexto de Negócio

### Problema

• Planilhas externas podem ter erros

• Necessidade de validar cálculos de terceiros

• Auditoria de cronogramas legados

• Debugging de divergências

### Solução

• Upload de CSV

• Parser robusto

• Diff coluna por coluna

• Sumário com métricas (PMT total, juros totais, saldo final)

• Indicação de linhas com divergência

### Valor de Negócio

• Auditoria automatizada

• Detecção rápida de erros

• Confiabilidade em migrações

• Ferramenta de debugging

## ✅ Requisitos Funcionais

• POST /api/validate/schedule aceita CSV

• Parser: detecta colunas automaticamente

• Diff: compara cada linha, coluna por coluna

• Tolerância: ± R$ 0,01 por célula

• Output: { linhas\_divergentes: [...], sumário: { pmt\_total\_diff, juros\_total\_diff, saldo\_final\_diff }, status: "ok" | "divergente" }

## ⚡ Requisitos Não Funcionais

• Performance: validar até 360 linhas em < 2s

• Parser robusto: aceita variações de formato CSV

• Tolerância configurável

• Mensagens de erro claras

## 🎯 Critérios de Aceite (GWT)

AC1: Upload CSV bem-sucedido

AC2: Diff detecta divergências corretamente

AC3: Sumário com métricas totais

AC4: Tolerância de R$ 0,01 aplicada

AC5: Resposta 200 (ok) ou 422 (divergências)

## ✔️ Definition of Ready (DoR)

• [x] H9/H11: Motor de cronogramas

• [x] Parser CSV escolhido

• [x] Algoritmo de diff definido

• [x] Estrutura de resposta desenhada

## ✅ Definition of Done (DoD)

• [x] POST /api/validate/schedule implementado

• [x] Parser CSV robusto

• [x] Diff coluna por coluna implementado

• [x] Sumário de métricas funcionando

• [x] Testes com CSVs válidos e divergentes

• [x] Envelope de erro padronizado (422)

## 🔗 Relacionamentos

### Depende de:

• H9: Price (para comparação)

• H11: SAC (para comparação)

### Bloqueia:

• H21: Snapshots (pode usar snapshots como baseline)

### Relacionado a:

• Contratos API: POST /api/validate/schedule

• Playbook: Validação de cronogramas

# 🚀 SPRINT 3 - CET COMPLETO + PERFIS + MERCADO

## (8 Histórias: H14-H19, H23-H24)

### Objetivo da Sprint

NPV/IRR robusto com Método de Brent, CET completo (IOF + seguros), Perfis por instituição, Comparador, XLSX com fórmulas, Casos de mercado, Acessibilidade & E2E.

### Critérios de Saída da Sprint 3

• |Δ CET| ≤ 0,03 p.p. a.a.

• IRR com Brent: convergência robusta, diagnósticos claros

• Perfis versionados por banco

• Comparador funcional (menor CET/total)

• XLSX com fórmulas do Excel

• 3 casos de mercado gabaritados

• Acessibilidade WCAG AA validada

• E2E cross-browser (Chrome/Firefox/Edge)

# H14 - NPV (VPL) - Valor Presente Líquido

## 📌 Identificação

• ID: H14

• Épico: E1 - Engine & Fundamentos

• Sprint: 3 (CET Completo)

• Prioridade: P1 (Alta)

• Pontos: 3

• Responsável: Engine Developer

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como analista de investimentos

Quero calcular NPV (VPL) de fluxos de caixa

Para avaliar viabilidade de projetos

## 💼 Contexto de Negócio

### Problema

• Análise de investimentos requer VPL

• Necessidade de avaliar fluxos irregulares

• Base para cálculo de IRR

• Decisões go/no-go dependem de VPL

### Solução

• Função calculateNPV precisa e rápida

• Suporte a fluxos irregulares

• Validação de inputs

• Integração com IRR

### Valor de Negócio

• Decisões de investimento embasadas

• Análise de projetos complexos

• Base para cálculo de TIR

• Conformidade com práticas financeiras

## ✅ Requisitos Funcionais

• NPV = Σ(CF\_t / (1+r)^t) para t=0..n

• Suporte a fluxos positivos e negativos

• Validação: pelo menos 1 fluxo negativo (investimento)

• Precisão decimal em todos os cálculos

• API: POST /api/npv-irr com action="npv"

## ⚡ Requisitos Não Funcionais

• Precisão: ± R$ 0,01

• Performance: até 360 fluxos em < 20ms

• Robustez: validar fluxos inválidos

• Estabilidade numérica

## 🎯 Critérios de Aceite (GWT)

AC1: NPV calculado corretamente vs Golden Files

AC2: Fluxos irregulares suportados

AC3: Erro para fluxos todos positivos/negativos

AC4: Integração com frontend funcional

## ✔️ Definition of Ready (DoR)

• [x] H2: Decimal.js

• [x] Fórmula de NPV validada

• [x] Casos de teste preparados

## ✅ Definition of Done (DoD)

• [x] Função calculateNPV implementada

• [x] Validação de fluxos

• [x] Testes unitários + propriedade

• [x] API /api/npv-irr (ação NPV) implementada

• [x] Documentação completa

• [x] 2 Golden Files (NPV\_001, NPV\_002)

## 🔗 Relacionamentos

### Depende de:

• H2: Precisão Decimal

### Bloqueia:

• H15: IRR (usa NPV internamente)

• H16: CET Completo (usa IRR)

### Relacionado a:

• Contratos API: POST /api/npv-irr

• Golden Files: NPV\_001, NPV\_002

# H15 - IRR (TIR) com Método de Brent

## 📌 Identificação

• ID: H15

• Épico: E1 - Engine & Fundamentos

• Sprint: 3 (CET Completo)

• Prioridade: P0 (Crítica)

• Pontos: 8

• Responsável: Engine Developer

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como analista financeiro

Quero calcular IRR (TIR) com método robusto de Brent

Para encontrar taxa interna de retorno mesmo em casos difíceis

## 💼 Contexto de Negócio

### Problema

• Newton-Raphson pode divergir

• Casos sem solução ou múltiplas soluções

• CET completo exige robustez

• Diagnósticos claros essenciais

### Solução

• Método de Brent (combinação bissecção + Newton)

• Diagnósticos claros de convergência

• Detecção de casos sem solução

• Performance otimizada

### Valor de Negócio

• Convergência em 99% dos casos reais

• Robustez em casos extremos

• Base sólida para CET completo

• Conformidade com padrões internacionais

## ✅ Requisitos Funcionais

• Brent: f(x) = NPV(rate) = 0

• Busca em intervalo [0.0001, 10.0] (0.01% a 1000% a.m.)

• Tolerância: erro relativo < 0.01%

• Máximo 100 iterações

• Diagnósticos: "convergiu", "sem\_solução", "multiplas\_soluções", "timeout"

• API: POST /api/npv-irr com action="irr"

## ⚡ Requisitos Não Funcionais

• Precisão: erro relativo < 0.01%

• Performance: < 100ms para 99% dos casos

• Convergência: 99% com ≤ 50 iterações

• Robustez: detectar casos patológicos

## 🎯 Critérios de Aceite (GWT)

AC1: IRR calculado corretamente vs Golden Files

AC2: Método de Brent converge em casos difíceis

AC3: Diagnósticos claros em casos sem solução

AC4: Performance dentro do SLA

AC5: 5 Golden Files passando

## ✔️ Definition of Ready (DoR)

• [x] H14: NPV (usado dentro de Brent)

• [x] ADR-002 sobre escolha de Brent

• [x] 5 Golden Files preparados com casos diversos

## ✅ Definition of Done (DoD)

• [x] Função calculateIRR\_Brent implementada

• [x] Diagnósticos funcionais

• [x] Testes com casos extremos

• [x] API /api/npv-irr (ação IRR) implementada

• [x] 5 Golden Files passando (IRR\_001 a IRR\_005)

• [x] ADR-002 documentado

• [x] Playbook: matriz IRR

## 🔗 Relacionamentos

### Depende de:

• H14: NPV

• H2: Precisão Decimal

### Bloqueia:

• H16: CET Completo (usa IRR)

• H12: CET Básico (pode migrar de Newton para Brent)

### Relacionado a:

• ADR-002: IRR com Método de Brent

• Contratos API: POST /api/npv-irr

• Golden Files: IRR\_001 a IRR\_005

• Playbook: Matriz IRR

# H16 - CET Completo (IOF + Seguros)

## 📌 Identificação

• ID: H16

• Épico: E2 - CET & Perfis

• Sprint: 3 (CET Completo)

• Prioridade: P0 (Crítica)

• Pontos: 8

• Responsável: Engine + Backend

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como cliente/regulador

Quero CET completo incluindo IOF diário, IOF adicional e seguros

Para ter custo efetivo total real

## 💼 Contexto de Negócio

### Problema

• CET básico não inclui IOF e seguros

• Regulamentação exige CET completo

• IOF diário é complexo (proporcional por dia)

• Seguros variam por perfil

### Solução

• CET completo com todos os custos

• IOF diário + adicional

• Seguros mensais ou únicos

• Metodologia conforme Guia CET - SoT

### Valor de Negócio

• Conformidade legal total

• Transparência máxima

• Comparação justa entre instituições

• Proteção ao consumidor

## ✅ Requisitos Funcionais

• IOF diário: 0,000082 por dia sobre saldo devedor

• IOF adicional: 0,38% sobre PV no t0

• Seguros: prestamista (mensal ou único), DFI (opcional)

• Fluxos ajustados: t0 -= IOF\_adic + seguros\_únicos; t1..tn -= seguros\_mensais

• CET = (1 + IRR\_mensal\_completo)^12 - 1

• API: POST /api/cet/full

## ⚡ Requisitos Não Funcionais

• Precisão: CET ± 0,03 p.p. a.a. (tolerância maior por complexidade)

• Performance: < 150ms

• Convergência: IRR-Brent garante

• Validação: todos os custos incluídos

## 🎯 Critérios de Aceite (GWT)

AC1: CET completo calculado vs Golden Files

AC2: IOF diário e adicional corretos

AC3: Seguros incluídos corretamente

AC4: Erro ≤ 0,03 p.p. a.a.

AC5: Explain mostra todos os custos

## ✔️ Definition of Ready (DoR)

• [x] H15: IRR-Brent

• [x] H12: CET Básico

• [x] Guia CET - SoT §5-7 estudado

• [x] 5 Golden Files preparados

## ✅ Definition of Done (DoD)

• [x] Motor calculateCET\_full implementado

• [x] IOF diário e adicional corretos

• [x] Seguros integrados

• [x] API /api/cet/full implementada

• [x] Explain Panel de CET completo

• [x] 5 Golden Files passando (CET\_001 a CET\_005)

• [x] Documentação de metodologia

## 🔗 Relacionamentos

### Depende de:

• H15: IRR-Brent

• H12: CET Básico

• H9/H11: Cronogramas

### Bloqueia:

• H17: Perfis CET (usa CET completo)

• H18: Comparador (usa CET completo)

### Relacionado a:

• Guia CET - SoT v1.0 (§5-7)

• Contratos API: POST /api/cet/full

• Golden Files: CET\_001 a CET\_005

# H17 - Perfis de CET por Instituição

## 📌 Identificação

• ID: H17

• Épico: E2 - CET & Perfis

• Sprint: 3 (CET Completo)

• Prioridade: P1 (Alta)

• Pontos: 5

• Responsável: Backend Developer

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como analista de mercado

Quero perfis de CET por instituição financeira

Para comparar práticas de diferentes bancos

## 💼 Contexto de Negócio

### Problema

• Cada banco tem estrutura de tarifas diferente

• IOF pode variar (federal vs estadual)

• Seguros são opcionais ou obrigatórios

• Necessidade de versionamento

### Solução

• Perfis versionados por instituição

• Estrutura padronizada

• API para criar/atualizar perfis

• Histórico de mudanças

### Valor de Negócio

• Comparação justa entre instituições

• Transparência de mercado

• Facilita análise competitiva

• Conformidade com regulamentação

## ✅ Requisitos Funcionais

• Perfil = { instituição, versão, timestamp, tarifas: { TAC, seguro\_prestamista, DFI, outras }, IOF: { diário, adicional }, divergências: [...] }

• API: GET /api/cet/profiles (lista), GET /api/cet/profiles/:id, POST /api/cet/profiles (admin)

• Versionamento: cada mudança gera nova versão

• Divergências documentadas conforme "Divergências de Perfis CET v1.0"

## ⚡ Requisitos Não Funcionais

• Performance: buscar perfil < 10ms

• Storage: máx 100 perfis ativos

• Versionamento: histórico completo

• Validação: campos obrigatórios

## 🎯 Critérios de Aceite (GWT)

AC1: Perfis criados e recuperáveis

AC2: Versionamento funcional

AC3: Divergências documentadas

AC4: API documentada no OpenAPI

AC5: 3 perfis exemplo cadastrados

## ✔️ Definition of Ready (DoR)

• [x] H16: CET Completo

• [x] ADR-004 sobre estrutura de perfis

• [x] Divergências de Perfis CET v1.0 estudado

## ✅ Definition of Done (DoD)

• [x] Modelo de dados Perfil implementado

• [x] API /api/cet/profiles implementada

• [x] Versionamento funcional

• [x] 3 perfis exemplo (Banco A, B, C)

• [x] Documentação de divergências

• [x] ADR-004 criado

## 🔗 Relacionamentos

### Depende de:

• H16: CET Completo

### Bloqueia:

• H18: Comparador (usa perfis)

### Relacionado a:

• ADR-004: Perfis de CET

• Divergências de Perfis CET v1.0

• Contratos API: GET/POST /api/cet/profiles

# H18 - Comparador de Cenários

## 📌 Identificação

• ID: H18

• Épico: E5 - Academy & Mercado

• Sprint: 3 (CET Completo)

• Prioridade: P2 (Média)

• Pontos: 5

• Responsável: Frontend + Backend

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como cliente

Quero comparar múltiplos cenários lado a lado

Para escolher a melhor oferta

## 💼 Contexto de Negócio

### Problema

• Clientes recebem múltiplas propostas

• Comparação manual propensa a erro

• Necessidade de critérios objetivos

• Visualização side-by-side essencial

### Solução

• Comparador com até 4 cenários

• Critérios: menor CET, menor total pago, menor PMT

• Destaque visual do melhor

• Exportação da comparação

### Valor de Negócio

• Empoderamento do cliente

• Decisão informada

• Redução de 80% em dúvidas

• Transparência total

## ✅ Requisitos Funcionais

• UI: tabela comparativa com 4 colunas

• Critérios: CET, total pago, PMT inicial

• Destaque: cor verde no melhor, badge "Recomendado"

• Exportação: PDF comparativo

• API: POST /api/compare com cenários: [{ sistema, PV, i, n, perfil }, ...]

## ⚡ Requisitos Não Funcionais

• Performance: comparar 4 cenários em < 1s

• UI responsiva

• Acessibilidade: navegação por teclado

• Exportação PDF < 3s

## 🎯 Critérios de Aceite (GWT)

AC1: Até 4 cenários comparáveis

AC2: Menor CET destacado corretamente

AC3: Exportação PDF funcional

AC4: UI responsiva

AC5: Acessibilidade validada

## ✔️ Definition of Ready (DoR)

• [x] H16: CET Completo

• [x] H17: Perfis

• [x] H13: Exportações (base para PDF comparativo)

• [x] Design de UI comparador aprovado

## ✅ Definition of Done (DoD)

• [x] Frontend do comparador implementado

• [x] API /api/compare implementada

• [x] Destaque visual funcional

• [x] Exportação PDF comparativo

• [x] Testes E2E de comparação

• [x] Documentação

## 🔗 Relacionamentos

### Depende de:

• H16: CET Completo

• H17: Perfis

• H13: Exportações

### Bloqueia:

• H23: Casos de Mercado (podem usar comparador)

### Relacionado a:

• Design System: UI de comparação

• Contratos API: POST /api/compare

# H19 - Exportação XLSX com Fórmulas

## 📌 Identificação

• ID: H19

• Épico: E3 - UI/UX & Exportações

• Sprint: 3 (CET Completo)

• Prioridade: P2 (Média)

• Pontos: 5

• Responsável: Backend Developer

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como analista avançado

Quero exportar cronogramas para XLSX com fórmulas do Excel

Para auditar e modificar cálculos

## 💼 Contexto de Negócio

### Problema

• CSV não tem fórmulas

• Analistas querem modificar cenários

• Auditoria exige ver fórmulas

• Integração com Excel profissional

### Solução

• XLSX com fórmulas nativas do Excel

• Células calculadas (não valores fixos)

• Formatação profissional

• Compatibilidade 100% com Excel/Sheets

### Valor de Negócio

• Flexibilidade máxima

• Auditoria transparente

• Modificação de cenários facilitada

• Profissionalismo

## ✅ Requisitos Funcionais

• XLSX: células com fórmulas (=B2\*C2)

• Valores numéricos idênticos ao CSV/PDF

• Formatação: moeda, percentual

• Fórmulas: juros = saldo\_anterior \* taxa, PMT = amortização + juros

• API: GET /reports/schedule.xlsx

## ⚡ Requisitos Não Funcionais

• Performance: gerar XLSX em < 2s

• Tamanho: < 5MB

• Compatibilidade: Excel 2016+, Google Sheets

• Fórmulas: validadas em Excel

## 🎯 Critérios de Aceite (GWT)

AC1: XLSX gerado com fórmulas

AC2: Fórmulas funcionam no Excel

AC3: Valores idênticos a CSV/PDF (±0.01)

AC4: Formatação profissional

AC5: Tempo de geração < 2s

## ✔️ Definition of Ready (DoR)

• [x] H9/H11: Cronogramas

• [x] H13: Exportações (base)

• [x] Biblioteca XLSX escolhida (exceljs)

• [x] Template XLSX aprovado

## ✅ Definition of Done (DoD)

• [x] Endpoint /reports/schedule.xlsx implementado

• [x] Fórmulas do Excel corretas

• [x] Formatação aplicada

• [x] Testes de compatibilidade Excel/Sheets

• [x] Validação de valores vs CSV

• [x] Documentação

## 🔗 Relacionamentos

### Depende de:

• H9/H11: Cronogramas

• H13: Exportações CSV/PDF

### Bloqueia:

• H23: Casos de Mercado (podem ter XLSX)

### Relacionado a:

• ADR-007: Bibliotecas de exportação

• Contratos API: GET /reports/schedule.xlsx

# H23 - Casos de Mercado Gabaritados

## 📌 Identificação

• ID: H23

• Épico: E5 - Academy & Mercado

• Sprint: 3 (CET Completo)

• Prioridade: P2 (Média)

• Pontos: 5

• Responsável: Content + Engine

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como estudante/profissional

Quero casos reais de mercado com solução gabaritada

Para aprender com exemplos práticos

## 💼 Contexto de Negócio

### Problema

• Teoria desconectada da prática

• Necessidade de exemplos reais

• Validação com casos conhecidos

• Branding educacional

### Solução

• 3 casos de mercado anonimizados

• Cada caso com: contexto, inputs, Golden File, Explain exportável

• Casos diversos: Price, SAC, CET completo

• Publicação no Academy com link

### Valor de Negócio

• Aplicação prática do conhecimento

• Validação com mercado real

• Branding como referência

• Redução de 60% em dúvidas

## ✅ Requisitos Funcionais

• 3 casos mínimos: 1) Financiamento imobiliário Price, 2) Empréstimo pessoal SAC, 3) Crédito consignado CET completo

• Cada caso: contexto (200 palavras), inputs, Golden File validado, Explain PDF

• Anonimização: sem nomes de instituições/pessoas

• Publicação em /academy/casos-mercado

## ⚡ Requisitos Não Funcionais

• Casos revisados por especialista

• Golden Files validados

• Anonimização completa (LGPD)

• SEO otimizado

## 🎯 Critérios de Aceite (GWT)

AC1: 3 casos publicados no Academy

AC2: Golden Files de cada caso passam

AC3: Explain PDF exportável de cada caso

AC4: Link no Academy funcional

AC5: Anonimização validada (LGPD)

## ✔️ Definition of Ready (DoR)

• [x] H16: CET Completo (para casos CET)

• [x] H9/H11: Price/SAC (para casos amortização)

• [x] H8: Explain Panel (para exportação)

• [x] Casos de mercado coletados e revisados

## ✅ Definition of Done (DoD)

• [x] 3 casos escritos e revisados

• [x] Golden Files dos 3 casos validados

• [x] Explain PDF de cada caso gerado

• [x] Publicação no Academy

• [x] Links funcionais

• [x] Anonimização validada

## 🔗 Relacionamentos

### Depende de:

• H16: CET Completo

• H9/H11: Price/SAC

• H8: Explain Panel

• H20: Academy

### Bloqueia:

• H20: Academy (casos linkados nos tópicos)

### Relacionado a:

• Golden Files: CASE\_001, CASE\_002, CASE\_003

• Academy: /casos-mercado

# H24 - Acessibilidade WCAG AA & E2E Tests

## 📌 Identificação

• ID: H24

• Épico: E3/E4 - UI/UX & Infraestrutura

• Sprint: 3 (CET Completo)

• Prioridade: P1 (Alta)

• Pontos: 8

• Responsável: Frontend + QA

• Status: 🔴 0% - Pendente

## 📝 História de Usuário

Como usuário com deficiência / QA Engineer

Quero acessibilidade WCAG AA e testes E2E cross-browser

Para garantir inclusão e qualidade

## 💼 Contexto de Negócio

### Problema

• Acessibilidade frequentemente negligenciada

• Necessidade de conformidade legal

• Testes manuais não escalam

• Bugs em produção custosos

### Solução

• Acessibilidade WCAG AA completa

• Testes E2E automatizados

• Cross-browser (Chrome/Firefox/Edge)

• CI/CD integrado

### Valor de Negócio

• Inclusão de 100% dos usuários

• Conformidade legal (Lei Brasileira de Inclusão)

• Redução de 90% em bugs de produção

• Confiabilidade

## ✅ Requisitos Funcionais

• WCAG AA: contraste ≥ 4.5, navegação por teclado, ARIA correto, foco visível, labels em inputs

• Testes E2E: Playwright para Chrome/Firefox/Edge

• Fluxos testados: todos os simuladores, exportações, comparador, Academy

• Lighthouse: score acessibilidade ≥ 90

• axe-core: zero erros críticos

## ⚡ Requisitos Não Funcionais

• Performance: suite E2E < 10 min

• Cobertura: 80% dos fluxos críticos

• CI/CD: E2E em PR antes de merge

• Relatórios: HTML com screenshots

## 🎯 Critérios de Aceite (GWT)

AC1: Lighthouse ≥ 90 em acessibilidade

AC2: axe-core: zero erros críticos

AC3: Navegação por teclado 100% funcional

AC4: E2E cross-browser executando no CI

AC5: Relatórios de E2E salvos em artefatos

## ✔️ Definition of Ready (DoR)

• [x] H7: Simuladores (para testar)

• [x] Todos os módulos UI

• [x] Playwright configurado

• [x] axe-core integrado

• [x] Design System com acessibilidade

## ✅ Definition of Done (DoD)

• [x] Acessibilidade WCAG AA implementada

• [x] Lighthouse ≥ 90

• [x] axe-core sem erros críticos

• [x] Suite E2E Playwright completa

• [x] E2E cross-browser no CI

• [x] Relatórios HTML salvos

• [x] Documentação de acessibilidade

## 🔗 Relacionamentos

### Depende de:

• H7: Simuladores

• H18: Comparador

• H20: Academy

• H1: CI/CD (para integração)

### Bloqueia:

• TODAS as HUs (qualidade final)

### Relacionado a:

• Design System: diretrizes de acessibilidade

• CI: E2E pipeline

• Relatórios: playwright-report/

# 📋 CHECKLISTS OPERACIONAIS

## Checklist Diário (Durante a Sprint)

☐ Limpei backups físicos hoje?

☐ Fiz commits locais das mudanças?

☐ npm run test passou localmente?

☐ Atualizei JSDoc nas funções novas?

☐ OpenAPI atualizado (se novo endpoint)?

☐ Golden Files impactados identificados?

⚠️ Se alguma resposta for "não" → CORRIGIR ANTES DE CONTINUAR

## Checklist Final de Sprint

☐ npm run type-check: PASSOU

☐ npm run lint: PASSOU

☐ npm run test:unit: PASSOU

☐ npm run golden:verify: PASSOU

☐ npm run build: PASSOU

☐ README atualizado

☐ Nenhum arquivo .bak no projeto

☐ Merge na main feito localmente

# 📏 TABELA DE TOLERÂNCIAS

|  |  |  |
| --- | --- | --- |
| Tipo de Cálculo | Métrica | Tolerância |
| Valores Monetários | Erro Absoluto | ≤ R$ 0,01 |
| Taxas | Erro em p.p. | ≤ 0,01 p.p. |
| CET Básico | CET a.a. | ≤ 0,01 p.p. |
| CET Completo | CET a.a. | ≤ 0,03 p.p. |
| Performance | P95 | ≤ 150 ms |
| Cobertura | Testes | ≥ 80% |

# 📋 RESUMO EXECUTIVO

## ✅ COBERTURA COMPLETA

• Sprint 1: 9 histórias (H1-H8, H20)

• Sprint 2: 7 histórias (H9-H13, H21-H22)

• Sprint 3: 8 histórias (H14-H19, H23-H24)

Total: 24 Histórias de Usuário documentadas

Golden Files: 30 planejados

Endpoints API: ~15 documentados

## 🎯 PRINCÍPIOS

✅ GitHub como Fonte da Verdade

✅ Validação Anti-Regressão Obrigatória

✅ Backup Exclusivo via Git Local

✅ Documentação Viva

════════════════════════════════════════════════════════════

Projeto FinMath - v2.0 Completa