**🚀 FINMATH - WORKFLOW COMPLETO E REGRAS CRÍTICAS v1.1**

**Documento de Referência Definitivo**  
**Data:** 23/Outubro/2025  
**Versão:** 1.1  
**Autor:** FinMath Core Team  
**Status:** 🟢 ATIVO

**📑 ÍNDICE**

1. [Regras Críticas Principais](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#regras-cr%C3%ADticas-principais)
2. [Workflow por Sprint](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#workflow-por-sprint)
3. [Checklists Operacionais](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#checklists-operacionais)
4. [Proibições Absolutas](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#proibi%C3%A7%C3%B5es-absolutas)
5. [Templates de Commit](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#templates-de-commit)
6. [Casos Especiais](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#casos-especiais)
7. [Métricas de Processo](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#m%C3%A9tricas-de-processo)
8. [🆕 Gestão de Dependências](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#gest%C3%A3o-de-depend%C3%AAncias)
9. [🆕 Ambiente de Desenvolvimento](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#ambiente-de-desenvolvimento)
10. [🆕 CI/CD Pipeline](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#cicd-pipeline)
11. [🆕 Troubleshooting](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#troubleshooting)
12. [🆕 Integração com Ferramentas](https://claude.ai/chat/0e88b07a-df22-4a9d-b77e-a5c4ae91885c#integra%C3%A7%C3%A3o-com-ferramentas)

**🎯 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

**📊 Fluxo Visual:**

GitHub (main) → Pull → Branch Local (sprint-X) → Commits Locais → Validação → Merge → Push → GitHub (main)

↑ ↓

└─────────────────────────────── CICLO DA SPRINT ──────────────────────────────────────┘

**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

pnpm -F @finmath/engine typecheck

if [ $? -ne 0 ]; then

echo "❌ Type check falhou - CORRIGIR ANTES DE PUSH"

exit 1

fi

# 2. Linting

pnpm -F @finmath/engine lint

if [ $? -ne 0 ]; then

echo "❌ Lint falhou - CORRIGIR ANTES DE PUSH"

exit 1

fi

# 3. Testes Unitários

pnpm -F @finmath/engine test:unit

if [ $? -ne 0 ]; then

echo "❌ Testes unitários falharam - CORRIGIR ANTES DE PUSH"

exit 1

fi

# 4. Testes de Propriedade

pnpm -F @finmath/engine test:property

if [ $? -ne 0 ]; then

echo "❌ Testes de propriedade falharam - CORRIGIR ANTES DE PUSH"

exit 1

fi

# 5. Testes de Integração

pnpm -F @finmath/engine test:integration

if [ $? -ne 0 ]; then

echo "❌ Testes de integração falharam - CORRIGIR ANTES DE PUSH"

exit 1

fi

# 6. Golden Files (CRÍTICO)

pnpm -F @finmath/engine test:golden

if [ $? -ne 0 ]; then

echo "❌ Golden Files falharam - CORRIGIR ANTES DE PUSH"

exit 1

fi

# 7. Build de Produção

pnpm -F @finmath/engine build

pnpm -F @finmath/api build

if [ $? -ne 0 ]; then

echo "❌ Build falhou - CORRIGIR ANTES DE PUSH"

exit 1

fi

# 8. Testes E2E (se aplicável)

pnpm -F @finmath/api test:e2e

if [ $? -ne 0 ]; then

echo "❌ E2E falharam - CORRIGIR ANTES DE PUSH"

exit 1

fi

# 9. Verificar Swagger UI

timeout 10s pnpm -F @finmath/api dev &

DEV\_PID=$!

sleep 5

curl -s http://localhost:3001/api-docs > /dev/null

DOCS\_STATUS=$?

kill $DEV\_PID 2>/dev/null

if [ $DOCS\_STATUS -ne 0 ]; then

echo "❌ Swagger UI não responde - CORRIGIR ANTES DE PUSH"

exit 1

fi

echo "✅ VALIDAÇÃO COMPLETA - PODE FAZER PUSH!"

**📊 Ordem de Execução:**

TypeCheck → Lint → Tests → Golden Files → Build → E2E → Swagger → ✅ APROVADO

↓ ↓ ↓ ↓ ↓ ↓ ↓

< 1min < 1min 2-5min 1-2min 2-3min 3-5min < 1min

**⏱️ Tempo Total Estimado:** 10-15 minutos

**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!

**💡 Rationale:**

* **Git local** é o sistema de backup
* **Histórico completo** de todas as mudanças
* **Sem poluição** no repositório
* **Sem conflitos** de arquivos duplicados

**REGRA #4: DOCUMENTAÇÃO VIVA OBRIGATÓRIA**

**OBRIGATÓRIO**: Toda funcionalidade documentada durante implementação.

**Tipos de Documentação:**

| **Tipo** | **Quando** | **Onde** | **Exemplo** |
| --- | --- | --- | --- |
| **JSDoc/TSDoc** | Funções públicas | Código-fonte | /\*\* @param {Decimal} pv \*/ |
| **OpenAPI** | Endpoints API | openapi-3.1\_finmath-v1.0.yaml | Schema de request/response |
| **ADRs** | Decisões arquiteturais | docs/adr/ | ADR-002: Escolha de IRR/Brent |
| **Swagger UI** | Em desenvolvimento | http://localhost:3001/api-docs | Interface interativa |
| **README** | Módulos/pacotes | Raiz de cada pacote | Como usar o módulo |

**📝 Exemplo Completo (H9 - Price):**

/\*\*

\* Calcula o PMT (Payment - Prestação) de um empréstimo Price.

\*

\* O sistema Price utiliza prestações constantes ao longo do período,

\* calculadas através da fórmula do Valor Presente de uma Anuidade.

\*

\* @param {Decimal} pv - Valor presente (principal)

\* @param {Decimal} rate - Taxa de juros por período (decimal, ex: 0.02 para 2%)

\* @param {number} n - Número de períodos

\* @returns {Decimal} Valor da prestação constante

\*

\* @example

\* ```typescript

\* const pmt = calculatePMT(

\* new Decimal('10000'), // R$ 10.000,00

\* new Decimal('0.02'), // 2% ao mês

\* 12 // 12 meses

\* );

\* // Retorna: Decimal('946.56')

\* ```

\*

\* @see {@link https://www.bcb.gov.br/content/estabilidadefinanceira/Documents/Cidadania\_financeira/RDC\_Res4935\_art5\_V.pdf | Resolução BCB 4.935}

\* @see {@link ./docs/cet-sot/guia-cet-v1.0.md#price | Guia CET — SoT §4.2}

\*/

export function calculatePMT(

pv: Decimal,

rate: Decimal,

n: number

): Decimal {

// Implementação...

}

**🔄 WORKFLOW POR SPRINT**

**Fase 1: Início da Sprint**

#!/bin/bash

# inicio-sprint.sh

echo "🚀 INICIANDO SPRINT X"

# 1. Sincronizar com GitHub

echo "🔄 Sincronizando com GitHub..."

git fetch origin

git pull origin main

if [ $? -ne 0 ]; then

echo "❌ Falha ao sincronizar - resolver conflitos"

exit 1

fi

# 2. Criar branch da sprint

SPRINT\_NUM=$1

git checkout -b "sprint-${SPRINT\_NUM}"

echo "✅ Branch sprint-${SPRINT\_NUM} criada"

# 3. Limpar backups físicos

echo "🧹 Limpando backups físicos..."

find . \( -name "\*bak\*" -o -name "\*backup\*" -o -name "\*.save" \) -type f -delete

# 4. Verificar dependências

echo "📦 Verificando dependências..."

pnpm install

# 5. Verificar ambiente

echo "🔍 Verificando ambiente..."

pnpm -F @finmath/engine typecheck

pnpm -F @finmath/engine lint

pnpm -F @finmath/engine test

pnpm -F @finmath/engine build

if [ $? -eq 0 ]; then

echo "✅ Ambiente pronto para desenvolvimento!"

echo "📋 Histórias desta sprint: H${SPRINT\_NUM}x"

echo ""

echo "🎯 Próximos passos:"

echo " 1. Ler backlog da sprint (docs/backlog/sprint-${SPRINT\_NUM}.md)"

echo " 2. Implementar histórias uma por vez"

echo " 3. Fazer commits locais frequentes"

echo " 4. Rodar testes após cada história"

else

echo "❌ Ambiente com problemas - corrigir antes de começar"

exit 1

fi

**📊 Checklist de Início:**

□ Git sincronizado com origin/main

□ Branch sprint-X criada

□ Backups físicos limpos

□ Dependências atualizadas (pnpm install)

□ Type check: ✅ PASSOU

□ Lint: ✅ PASSOU

□ Tests: ✅ PASSOU

□ Build: ✅ PASSOU

□ Backlog da sprint lido

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

# ======================================

# PADRÃO DE COMMIT POR HISTÓRIA

# ======================================

# Exemplo: Implementando H9 (Price)

# 1. Implementar motor

cd packages/engine/src/amortization

# ... escrever código ...

# 2. Implementar testes

cd ../../test/unit/amortization

# ... escrever testes ...

# 3. Implementar golden files

cd ../../golden/sprint2

# ... criar golden files ...

# 4. Rodar testes localmente

cd ../../../..

pnpm -F @finmath/engine test:unit

pnpm -F @finmath/engine test:property

pnpm -F @finmath/engine test:golden

# 5. Commit local (SE E SOMENTE SE testes passarem)

git add packages/engine/src/amortization/price.ts

git add packages/engine/test/unit/amortization/price.test.ts

git add packages/engine/test/golden/sprint2/PRICE\_\*.json

git commit -m "feat(H9): Implementa Price PMT e cronograma

- Motor: calculatePMT com decimal.js

- API: POST /api/price

- Testes: 8 unitários + 3 propriedade + 5 golden files

- Ajuste final: saldo\_n <= 0.01

- Cobertura: 87%

DoD: 5/5 critérios atendidos

Referências:

- Guia CET — SoT (§4)

- ADR-001 (decimal.js)

- Contratos de API — FinMath (§2.1)"

# 6. Continuar com próxima história...

# Repetir processo para H10, H11, etc.

**🔁 Ciclo de Desenvolvimento:**

Implementar → Testar → Commit → Próxima História

↓ ↓ ↓

Código Unit/Prop Git local

Golden

**📊 Frequência de Commits:**

* **Mínimo:** 1 commit por história
* **Recomendado:** 1 commit por funcionalidade dentro da história
* **Máximo:** Sem limite (commits frequentes são encorajados)

**💡 Boas Práticas:**

# ✅ BOM: Commits incrementais

git commit -m "feat(H9): Implementa calculatePMT"

git commit -m "feat(H9): Adiciona testes unitários PMT"

git commit -m "feat(H9): Adiciona golden files PRICE"

git commit -m "feat(H9): Implementa ajuste final"

# ❌ RUIM: Commit único gigante

git commit -m "feat(H9): Implementa tudo do Price"

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

#!/bin/bash

# finalizar-sprint.sh

SPRINT\_NUM=$1

echo "🏁 FINALIZANDO SPRINT ${SPRINT\_NUM}"

# 1. Garantir que estamos na branch correta

CURRENT\_BRANCH=$(git branch --show-current)

if [[ ! $CURRENT\_BRANCH =~ ^sprint-[0-9]+$ ]]; then

echo "❌ Não está em uma branch de sprint"

exit 1

fi

# 2. Limpar backups físicos (última vez)

echo "🧹 Limpeza final de backups físicos..."

find . \( -name "\*bak\*" -o -name "\*backup\*" -o -name "\*.save" \) -type f -delete

# 3. VALIDAÇÃO ANTI-REGRESSÃO COMPLETA

echo "🔍 ========================================="

echo "🔍 VALIDAÇÃO ANTI-REGRESSÃO COMPLETA"

echo "🔍 ========================================="

echo ""

echo "📊 Etapa 1/9: Type Check..."

pnpm -F @finmath/engine typecheck || { echo "❌ Type check falhou"; exit 1; }

echo "✅ Type check passou!"

echo ""

echo "📊 Etapa 2/9: Linting..."

pnpm -F @finmath/engine lint || { echo "❌ Lint falhou"; exit 1; }

echo "✅ Lint passou!"

echo ""

echo "📊 Etapa 3/9: Testes Unitários..."

pnpm -F @finmath/engine test:unit || { echo "❌ Testes unitários falharam"; exit 1; }

echo "✅ Testes unitários passaram!"

echo ""

echo "📊 Etapa 4/9: Testes de Propriedade..."

pnpm -F @finmath/engine test:property || { echo "❌ Testes de propriedade falharam"; exit 1; }

echo "✅ Testes de propriedade passaram!"

echo ""

echo "📊 Etapa 5/9: Testes de Integração..."

pnpm -F @finmath/engine test:integration || { echo "❌ Testes de integração falharam"; exit 1; }

echo "✅ Testes de integração passaram!"

echo ""

echo "📊 Etapa 6/9: Golden Files..."

pnpm -F @finmath/engine test:golden || { echo "❌ Golden Files falharam"; exit 1; }

echo "✅ Golden Files passaram!"

echo ""

echo "📊 Etapa 7/9: Build..."

pnpm -F @finmath/engine build || { echo "❌ Build falhou"; exit 1; }

pnpm -F @finmath/api build || { echo "❌ Build API falhou"; exit 1; }

echo "✅ Build passou!"

echo ""

echo "📊 Etapa 8/9: Testes E2E..."

pnpm -F @finmath/api test:e2e || { echo "❌ E2E falharam"; exit 1; }

echo "✅ E2E passaram!"

echo ""

echo "📊 Etapa 9/9: Swagger UI..."

timeout 10s pnpm -F @finmath/api dev &

DEV\_PID=$!

sleep 5

curl -s http://localhost:3001/api-docs > /dev/null

DOCS\_STATUS=$?

kill $DEV\_PID 2>/dev/null

if [ $DOCS\_STATUS -ne 0 ]; then

echo "❌ Swagger UI não responde"

exit 1

fi

echo "✅ Swagger UI acessível!"

echo ""

echo "🎉 ========================================="

echo "🎉 VALIDAÇÃO COMPLETA - SPRINT APROVADA!"

echo "🎉 ========================================="

# 4. Coletar métricas

echo ""

echo "📊 Coletando métricas da sprint..."

TOTAL\_COMMITS=$(git log sprint-${SPRINT\_NUM} --oneline | wc -l)

FEAT\_COMMITS=$(git log sprint-${SPRINT\_NUM} --oneline | grep 'feat(' | wc -l)

FIX\_COMMITS=$(git log sprint-${SPRINT\_NUM} --oneline | grep 'fix(' | wc -l)

GOLDEN\_FILES=$(find packages/engine/test/golden/sprint${SPRINT\_NUM} -name "\*.json" 2>/dev/null | wc -l)

echo " • Total de commits: ${TOTAL\_COMMITS}"

echo " • Features (feat): ${FEAT\_COMMITS}"

echo " • Correções (fix): ${FIX\_COMMITS}"

echo " • Golden Files novos: ${GOLDEN\_FILES}"

# 5. Merge na main (local)

echo ""

echo "🔀 Fazendo merge na main..."

git checkout main

git merge "sprint-${SPRINT\_NUM}" --no-ff -m "chore: Merge Sprint ${SPRINT\_NUM}

Histórias implementadas:

$(git log sprint-${SPRINT\_NUM} --oneline | grep 'feat(' | sed 's/^/- /')

Validação anti-regressão: ✅ PASSOU

- Type Check: ✅

- Lint: ✅

- Unit Tests: ✅

- Property Tests: ✅

- Integration Tests: ✅

- Golden Files: ✅ (${GOLDEN\_FILES} arquivos)

- Build: ✅

- E2E: ✅

- Swagger UI: ✅

Commits: ${TOTAL\_COMMITS} (${FEAT\_COMMITS} features, ${FIX\_COMMITS} fixes)

motorVersion: $(grep '"version"' packages/engine/package.json | head -1 | cut -d'"' -f4)

Data: $(date +%Y-%m-%d)"

# 6. PUSH FINAL PARA GITHUB

echo ""

echo "📤 ========================================="

echo "📤 PUSH FINAL PARA GITHUB"

echo "📤 ========================================="

git push origin main

if [ $? -eq 0 ]; then

echo ""

echo "✅ ========================================="

echo "✅ SPRINT ${SPRINT\_NUM} PUBLICADA COM SUCESSO!"

echo "✅ ========================================="

echo ""

echo "🔗 GitHub: https://github.com/PrinceOfEgypt1/fin-math"

echo "📊 Commits publicados: ${TOTAL\_COMMITS}"

echo "📁 Golden Files: ${GOLDEN\_FILES}"

echo ""

else

echo "❌ Falha no push - verificar conectividade"

exit 1

fi

# 7. Deletar branch local da sprint

git branch -d "sprint-${SPRINT\_NUM}"

echo "🎉 SPRINT ${SPRINT\_NUM} CONCLUÍDA E PUBLICADA!"

echo "🚀 Pronta para iniciar Sprint $((SPRINT\_NUM + 1))"

**📋 CHECKLISTS OPERACIONAIS**

**Checklist Diário (Durante a Sprint)**

#!/bin/bash

# checklist-diario.sh

echo "📅 CHECKLIST DIÁRIO FINMATH"

echo "================================"

echo ""

# 1. Backups físicos

echo "1. Limpeza de backups físicos:"

find . \( -name "\*bak\*" -o -name "\*backup\*" -o -name "\*.save" \) -type f

BACKUP\_COUNT=$(find . \( -name "\*bak\*" -o -name "\*backup\*" -o -name "\*.save" \) -type f | wc -l)

if [ $BACKUP\_COUNT -eq 0 ]; then

echo " ✅ Nenhum backup físico encontrado"

else

echo " ❌ ${BACKUP\_COUNT} backups físicos encontrados - LIMPAR!"

echo " 💡 Executar: find . \( -name '\*bak\*' -o -name '\*backup\*' \) -type f -delete"

fi

# 2. Commits locais

echo ""

echo "2. Status do Git:"

if git diff-index --quiet HEAD --; then

echo " ✅ Nenhuma mudança não commitada"

else

echo " ⚠️ Mudanças não commitadas encontradas"

echo " 💡 Executar: git add . && git commit -m 'descrição'"

fi

# 3. Testes

echo ""

echo "3. Status dos testes:"

pnpm -F @finmath/engine test:unit --silent > /dev/null 2>&1

if [ $? -eq 0 ]; then

echo " ✅ Testes unitários: PASSANDO"

else

echo " ❌ Testes unitários: FALHANDO - CORRIGIR!"

fi

# 4. Documentação

echo ""

echo "4. Documentação:"

UNDOC\_FUNCS=$(grep -r "export function" packages/engine/src --include="\*.ts" | grep -v "/\\*\\*" | wc -l)

if [ $UNDOC\_FUNCS -eq 0 ]; then

echo " ✅ Todas funções documentadas"

else

echo " ⚠️ ${UNDOC\_FUNCS} funções sem JSDoc - DOCUMENTAR!"

fi

# 5. OpenAPI

echo ""

echo "5. OpenAPI atualizado:"

LAST\_API\_CHANGE=$(git log -1 --format=%cd packages/api/src/routes/\*.ts 2>/dev/null)

LAST\_OPENAPI\_CHANGE=$(git log -1 --format=%cd openapi-3.1\_finmath-v1.0.yaml 2>/dev/null)

if [ "$LAST\_API\_CHANGE" == "$LAST\_OPENAPI\_CHANGE" ]; then

echo " ✅ OpenAPI sincronizado"

else

echo " ⚠️ OpenAPI pode estar desatualizado - REVISAR!"

fi

# 6. Golden Files

echo ""

echo "6. Golden Files:"

pnpm -F @finmath/engine test:golden --silent > /dev/null 2>&1

if [ $? -eq 0 ]; then

echo " ✅ Golden Files: PASSANDO"

else

echo " ❌ Golden Files: FALHANDO - CORRIGIR!"

fi

echo ""

echo "================================"

echo "Se algum item está ❌ ou ⚠️ → CORRIGIR ANTES DE CONTINUAR"

**📊 Saída Esperada:**

📅 CHECKLIST DIÁRIO FINMATH

================================

1. Limpeza de backups físicos:

✅ Nenhum backup físico encontrado

2. Status do Git:

✅ Nenhuma mudança não commitada

3. Status dos testes:

✅ Testes unitários: PASSANDO

4. Documentação:

✅ Todas funções documentadas

5. OpenAPI atualizado:

✅ OpenAPI sincronizado

6. Golden Files:

✅ Golden Files: PASSANDO

================================

Se algum item está ❌ ou ⚠️ → CORRIGIR ANTES DE CONTINUAR

**Checklist de História (Ao Completar H9, H10, etc.)**

# ✅ CHECKLIST - HISTÓRIA Hxx CONCLUÍDA

## Implementação

- [ ] Motor implementado (packages/engine/src/)

- [ ] Testes unitários (≥3 casos)

- [ ] Testes de propriedade (invariantes)

- [ ] Golden File criado/atualizado

- [ ] API implementada (se aplicável)

- [ ] Validação Zod (se aplicável)

## Documentação

- [ ] JSDoc em funções públicas

- [ ] OpenAPI atualizado (se API)

- [ ] ADR criado (se decisão estrutural)

- [ ] Changelog da história no commit

## Qualidade

- [ ] `pnpm -F @finmath/engine test:unit` passa

- [ ] `pnpm -F @finmath/engine lint` passa

- [ ] `pnpm -F @finmath/engine build` passa

- [ ] Golden Files passam

- [ ] Cobertura mantida (≥80%)

## Git

- [ ] Commit local feito

- [ ] Mensagem convencional (feat/fix/docs)

- [ ] Nenhum .bak commitado

## Definition of Done (DoD)

- [ ] Critério 1: Funcionalidade implementada

- [ ] Critério 2: Testes passando

- [ ] Critério 3: Documentação completa

- [ ] Critério 4: Code review (pair programming)

- [ ] Critério 5: Golden Files validados

---

\*\*Se todos [x] → HISTÓRIA CONCLUÍDA!\*\*

**Checklist Final de Sprint (Antes do Push)**

# 🏁 CHECKLIST FINAL - SPRINT X

## Validação Anti-Regressão

- [ ] `pnpm -F @finmath/engine typecheck`: PASSOU

- [ ] `pnpm -F @finmath/engine lint`: PASSOU

- [ ] `pnpm -F @finmath/engine test:unit`: PASSOU

- [ ] `pnpm -F @finmath/engine test:property`: PASSOU

- [ ] `pnpm -F @finmath/engine test:integration`: PASSOU

- [ ] `pnpm -F @finmath/engine test:golden`: PASSOU (X/X)

- [ ] `pnpm -F @finmath/engine build`: PASSOU

- [ ] `pnpm -F @finmath/api build`: PASSOU

- [ ] `pnpm -F @finmath/api test:e2e`: PASSOU

- [ ] Swagger UI: ACESSÍVEL (http://localhost:3001/api-docs)

## Documentação

- [ ] README.md atualizado

- [ ] OpenAPI sincronizado

- [ ] ADRs criados (se necessário)

- [ ] Changelog da sprint documentado

- [ ] JSDoc em todas funções públicas

- [ ] Comentários em lógica complexa

## Golden Files

- [ ] Novos GFs revisados por especialista

- [ ] motorVersion atualizado (se necessário)

- [ ] Tolerâncias respeitadas (≤0.01)

- [ ] Todos GFs da sprint passando

- [ ] GFs antigos ainda passando (anti-regressão)

## Limpeza

- [ ] Nenhum arquivo .bak no projeto

- [ ] Nenhum console.log não intencional

- [ ] Nenhum TODO/FIXME crítico

- [ ] Imports não utilizados removidos

- [ ] Código comentado removido

## Git

- [ ] Todos os commits na branch sprint-X

- [ ] Merge na main feito localmente

- [ ] Mensagem de merge descritiva

- [ ] Histórico limpo (sem commits duplicados)

- [ ] Branch atualizada com main antes do merge

## Métricas

- [ ] Cobertura de testes ≥ 80%

- [ ] Complexidade ciclomática < 10 (por função)

- [ ] Todas histórias da sprint concluídas

- [ ] 0 issues críticos no lint

- [ ] 0 vulnerabilidades críticas (npm audit)

---

\*\*Se TODOS [x] → PODE FAZER PUSH PARA GITHUB!\*\*

**🚨 PROIBIÇÕES ABSOLUTAS**

**❌ NUNCA Fazer**

# ========================================

# PROIBIÇÕES ABSOLUTAS - NUNCA EXECUTAR

# ========================================

# 1. NUNCA criar backups físicos

cp file.ts file.ts.bak # ❌ PROIBIDO

cp file.ts file.backup.ts # ❌ PROIBIDO

mv file.ts file.ts.save # ❌ PROIBIDO

tar -czf backup.tar.gz src/ # ❌ PROIBIDO

# 2. NUNCA fazer push durante a sprint

git push origin sprint-2 # ❌ PROIBIDO (só ao final!)

git push -f origin main # ❌ PROIBIDO SEMPRE!

git push --force # ❌ PROIBIDO SEMPRE!

# 3. NUNCA commit sem testes

git commit -m "fix: corrige bug" # ❌ SEM TESTES = PROIBIDO

git commit --no-verify # ❌ PROIBIDO (bypassa hooks)

# Correto:

pnpm -F @finmath/engine test || exit 1

git commit -m "fix: corrige bug" # ✅ CORRETO

# 4. NUNCA push sem validação completa

git push origin main # ❌ SEM VALIDAÇÃO = PROIBIDO

# Correto:

./finalizar-sprint.sh # ✅ CORRETO (valida antes)

# 5. NUNCA usar float para cálculos

const result = 10000 \* 0.025; # ❌ PROIBIDO

const rate = 0.1 + 0.2; # ❌ PROIBIDO (0.30000000000000004)

# Correto:

import { Decimal } from 'decimal.js';

const result = new Decimal('10000').mul('0.025'); # ✅ CORRETO

const rate = new Decimal('0.1').plus('0.2'); # ✅ CORRETO

# 6. NUNCA duplicar lógica de CET

// Implementar CET aqui... # ❌ PROIBIDO

# Correto:

// Ver: Guia CET — SoT # ✅ CORRETO (fonte única)

import { calculateCET } from '@/cet';

# 7. NUNCA commit de Golden File sem revisão

git add PRICE\_001.json && git commit # ❌ SEM REVISÃO = PROIBIDO

# Correto:

# 1. Revisar com especialista financeiro

# 2. Validar cálculos manualmente

# 3. Aprovar por escrito

# 4. Então commitar # ✅ CORRETO

# 8. NUNCA alterar Golden Files antigos sem motivo

sed -i 's/946.56/946.57/' PRICE\_001.json # ❌ PROIBIDO

# Correto:

# 1. Identificar motivo da mudança

# 2. Documentar no ADR

# 3. Atualizar motorVersion

# 4. Recalcular com npm run golden:recalc # ✅ CORRETO

# 9. NUNCA commitar node\_modules ou dist/

git add node\_modules/ # ❌ PROIBIDO

git add packages/\*/dist/ # ❌ PROIBIDO

# Correto: Garantir que .gitignore está configurado

# 10. NUNCA usar any em TypeScript

function calc(x: any): any { # ❌ PROIBIDO

return x \* 2;

}

# Correto:

function calc(x: number): number { # ✅ CORRETO

return x \* 2;

}

# 11. NUNCA usar console.log em produção

console.log('Debug:', value); # ❌ PROIBIDO em prod

# Correto:

import { logger } from '@/logger';

logger.debug('Debug:', value); # ✅ CORRETO

# 12. NUNCA fazer merge manual sem testar

git checkout main

git merge sprint-2 # ❌ SEM TESTES = PROIBIDO

# Correto:

./finalizar-sprint.sh # ✅ CORRETO (testa antes)

**✅ SEMPRE Fazer**

# ========================================

# BOAS PRÁTICAS - SEMPRE EXECUTAR

# ========================================

# 1. SEMPRE limpar backups físicos diariamente

find . \( -name "\*bak\*" -o -name "\*backup\*" \) -type f -delete

# 2. SEMPRE commits locais frequentes

git add .

git commit -m "feat(H9): Implementa PMT"

# 3. SEMPRE testar antes de commit

pnpm -F @finmath/engine test && \

git add . && \

git commit -m "feat(H9): Implementa PMT"

# 4. SEMPRE validação completa antes de push final

./finalizar-sprint.sh # Executa todas as validações

# 5. SEMPRE usar decimal.js para cálculos

import { Decimal } from 'decimal.js';

import { toDecimal, round2 } from '@/decimal';

const result = new Decimal('10000').mul('0.025');

const rounded = round2(result);

# 6. SEMPRE consultar Guia CET para mudanças de CET

# Ver: docs/cet-sot/guia-cet-v1.0.md

# 7. SEMPRE documentar durante implementação

/\*\*

\* Calcula PMT do Price.

\* @param {Decimal} pv - Valor presente

\* @returns {Decimal} Prestação

\*/

export function calculatePMT(pv: Decimal): Decimal {

// ...

}

# 8. SEMPRE atualizar OpenAPI ao criar/modificar endpoints

# Arquivo: openapi-3.1\_finmath-v1.0.yaml

# 9. SEMPRE criar ADR para decisões arquiteturais importantes

# Exemplo: ADR-002: Escolha de decimal.js

# Localização: docs/adr/ADR-002-decimal-js.md

# 10. SEMPRE rodar linter antes de commit

pnpm -F @finmath/engine lint --fix

git add .

git commit

# 11. SEMPRE usar convenção de commit

# feat(HXX): descrição

# fix(HXX): descrição

# docs(HXX): descrição

# test(HXX): descrição

# refactor(HXX): descrição

# chore: descrição

# 12. SEMPRE revisar diff antes de commit

git diff

git add -p # Adiciona interativamente

git commit

**📝 TEMPLATES DE COMMIT**

**Commit de História (Durante Sprint)**

git commit -m "feat(H9): Implementa Price PMT e cronograma

- Motor: calculatePMT com decimal.js

- API: POST /api/price

- Testes: 8 unitários + 3 propriedade + 5 golden files

- Ajuste final: saldo\_n <= 0.01

- Cobertura: 87%

DoD: 5/5 critérios atendidos

Referências:

- Guia CET — SoT (§4)

- ADR-001 (decimal.js)

- Contratos de API — FinMath (§2.1)

Impacto:

- Novos arquivos: 3

- Linhas adicionadas: 245

- Golden Files: PRICE\_001 a PRICE\_005"

**📊 Estrutura:**

[tipo]([escopo]): [título curto]

← linha em branco

[corpo detalhado] ← descrição completa

- [item 1] ← bullet points

- [item 2]

- [item 3]

← linha em branco

[metadados] ← referências e métricas

**Commit de Correção (Durante Sprint)**

git commit -m "fix(H9): Corrige arredondamento em ajuste final

Problema:

- Resíduo de 0.02 em alguns casos

- Ocorria quando principal não é múltiplo exato de PMT

- Golden File PRICE\_003 falhando

Solução:

- Aplicar round2 antes de comparação final

- Ajuste: saldo\_n <= 0.01 (tolerância aumentada)

- Validação adicional em testes de propriedade

Testes:

- Adiciona PRICE\_006 (caso extremo)

- Adiciona teste de propriedade para resíduo

- Todos golden files passam agora (6/6)

Closes: #12 (bug identificado em teste de propriedade)

Impacto:

- Arquivos modificados: 2

- Linhas alteradas: 15

- Golden Files afetados: 6"

**Commit de Documentação (Durante Sprint)**

git commit -m "docs(H9): Adiciona JSDoc completo em calculatePMT

Documentação adicionada:

- Descrição detalhada da função

- Parâmetros com tipos e descrições

- Retorno documentado

- Exemplo de uso completo

- Links para Guia CET e Resolução BCB

OpenAPI:

- Atualiza schema de request

- Atualiza schema de response

- Adiciona exemplos de uso

- Documenta códigos de erro

Referências:

- Contratos de API — FinMath (§1)

- Guia CET — SoT (§4.2)

Impacto:

- Arquivos modificados: 3

- Linhas adicionadas: 85

- Cobertura de documentação: 100%"

**Commit de Merge (Final da Sprint)**

git commit -m "chore: Merge Sprint 2

Histórias implementadas:

- H9: Price (PMT/Cronograma/Ajuste Final)

- H10: Day Count (30/360, ACT/365, pró-rata)

- H11: SAC (Cronograma)

- H12: CET Básico (tarifas t0)

- H13: Exportações (CSV/PDF)

- H21: Snapshots (hash + motorVersion)

- H22: Validador (upload CSV + diffs)

Validação anti-regressão: ✅ PASSOU

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

• Type Check: ✅ 0 erros

• Lint: ✅ 0 warnings

• Unit Tests: ✅ 127/127 passou (85% cobertura)

• Property Tests: ✅ 45/45 passou

• Integration Tests: ✅ 23/23 passou

• Golden Files: ✅ 10/10 verdes

• Build Engine: ✅ sucesso

• Build API: ✅ sucesso

• E2E Tests: ✅ 15/15 passou

• Swagger UI: ✅ acessível

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

Golden Files:

- Novos: PRICE\_001 a PRICE\_005

- Novos: SAC\_001 a SAC\_005

- Total: 10 arquivos

- Status: Todos validados por especialista

Commits da Sprint:

- Total: 47 commits

- Features (feat): 35

- Correções (fix): 8

- Documentação (docs): 4

Versão:

- motorVersion: 0.2.0 (atualizado)

- API version: 1.0.0

Métricas de Qualidade:

- Cobertura de testes: 85% → 87% (+2%)

- Complexidade ciclomática média: 3.2

- Tempo de execução dos testes: 8.5s

- Tamanho do build: 245KB (gzip)

Revisão:

- Code review: ✅ aprovado

- Especialista financeiro: ✅ aprovado (2025-10-15)

- Testes manuais: ✅ aprovados

- Validação de integração: ✅ aprovada

Próxima Sprint: Sprint 3

- Histórias planejadas: H14-H19, H23

- Foco: SAC completo, Price avançado, CET intermediário

- Data prevista: 2025-11-01"

**🎯 CASOS ESPECIAIS**

**1. Hotfix Crítico (Fora da Sprint)**

#!/bin/bash

# hotfix-critico.sh

echo "🚨 HOTFIX CRÍTICO - PROCESSO EXCEPCIONAL"

# 1. Criar branch de hotfix

git checkout main

git pull origin main

git checkout -b hotfix-critical-$(date +%Y%m%d)

echo "⚠️ Branch criada: hotfix-critical-$(date +%Y%m%d)"

# 2. Implementar correção mínima

echo "📝 Implementar APENAS a correção mínima necessária"

echo " - Não adicionar features"

echo " - Não refatorar código não relacionado"

echo " - Foco EXCLUSIVO no bug crítico"

# ... implementação ...

# 3. Validação anti-regressão (completa)

echo "🔍 Validando correção..."

pnpm -F @finmath/engine test:unit || { echo "❌ Testes falharam"; exit 1; }

pnpm -F @finmath/engine test:golden || { echo "❌ Golden Files falharam"; exit 1; }

pnpm -F @finmath/engine build || { echo "❌ Build falhou"; exit 1; }

# 4. Commit e push

git add .

git commit -m "hotfix: Corrige bug crítico de precisão em CET

Problema:

- Bug crítico: arredondamento incorreto em IOF diário

- Impacto: cálculos de CET divergindo até 0.05%

- Afeta: todos cálculos de CET com IOF diário

- Severidade: CRÍTICA (produção)

Causa Raiz:

- IOF diário não estava sendo arredondado antes da soma

- Acumulação de erros de precisão float

- Propagação para cálculo final de CET

Solução:

- Aplicar round2 após cada cálculo de IOF diário

- Arredondar antes de somar ao fluxo

- Validação adicional no teste de propriedade

Validação:

- CET\_002 atualizado (delta < 0.01 restaurado)

- Adiciona teste de regressão CET\_010

- Todos golden files passam (3/3)

Testes:

✅ Unit tests: 127/127 passou

✅ Golden Files: 11/11 passou (incluindo novo)

✅ Property tests: 45/45 passou

✅ Build: sucesso

Urgência: CRÍTICA

- Afeta cálculos de clientes

- Deploy imediato necessário

- Requer rollback se não aprovado

Revisor:

- Especialista Mat. Financeira: ✅ aprovado

- Code review: ✅ aprovado

- Testes manuais: ✅ aprovados

Referências:

- Issue: #45

- Guia CET — SoT (§8.3)

- Resolução BCB 4.935 (Art. 5º)"

# 5. Push direto (exceção!)

git push origin hotfix-critical-$(date +%Y%m%d)

echo "✅ Hotfix pushed"

echo "📋 Próximo passo: Abrir PR IMEDIATO para review"

echo "⚠️ Marcar PR como: URGENTE | HOTFIX | PRODUÇÃO"

# 6. Após merge do PR

echo "🔀 Após merge do PR:"

echo " 1. Fazer merge também na branch sprint atual"

echo " 2. Notificar equipe"

echo " 3. Documentar no changelog"

echo " 4. Atualizar motorVersion (patch)"

**📊 Critérios para Hotfix:**

* ✅ Bug CRÍTICO em produção
* ✅ Afeta cálculos financeiros
* ✅ Impacto financeiro > R$ 1.000
* ✅ Violação de normas BCB
* ❌ Features novas (esperar sprint)
* ❌ Refactoring (esperar sprint)
* ❌ Performance (esperar sprint)

**2. Golden File Não Passa (Durante Sprint)**

#!/bin/bash

# debug-golden-file.sh

echo "🔍 DEBUGGING GOLDEN FILE FAILURE"

# 1. Identificar qual GF falhou

echo "📊 Executando golden:verify..."

pnpm -F @finmath/engine test:golden

# Output esperado:

# ❌ PRICE\_001 FALHOU

# Esperado: 946.56

# Encontrado: 946.57

# Delta: 0.01 (tolerância: 0.01)

# 2. Investigar causa

echo ""

echo "🔎 Investigando causa da falha:"

echo " A) Mudança intencional de algoritmo?"

echo " B) Bug introduzido no código?"

echo " C) Erro no Golden File original?"

echo ""

read -p "Qual a causa? (A/B/C): " CAUSA

case $CAUSA in

A)

echo ""

echo "✅ Mudança intencional detectada"

echo ""

echo "📋 Processo para atualizar Golden File:"

echo " 1. Recalcular GF:"

echo " pnpm -F @finmath/engine golden:recalc -- --file=PRICE\_001"

echo ""

echo " 2. Revisar com especialista:"

echo " - Validar cálculos manualmente"

echo " - Comparar com referências (BCB, livros)"

echo " - Documentar mudança"

echo ""

echo " 3. Aprovar por escrito:"

echo " - Email do especialista aprovando"

echo " - Ou comentário no PR"

echo ""

echo " 4. Atualizar motorVersion (se necessário):"

echo " - Patch: correção de bug (0.2.0 → 0.2.1)"

echo " - Minor: funcionalidade nova (0.2.0 → 0.3.0)"

echo " - Major: breaking change (0.2.0 → 1.0.0)"

echo ""

echo " 5. Documentar em ADR:"

echo " - Criar ADR-XXX-mudanca-algoritmo-price.md"

echo " - Explicar motivo da mudança"

echo " - Impacto nos cálculos"

echo ""

echo " 6. Commit:"

echo " git add packages/engine/test/golden/sprint2/PRICE\_001.json"

echo " git add packages/engine/package.json # se motorVersion mudou"

echo " git add docs/adr/ADR-XXX-mudanca-algoritmo-price.md"

echo " git commit -m 'fix(H9): Atualiza algoritmo Price para maior precisão"

echo ""

echo " - Algoritmo anterior: [descrever]"

echo " - Algoritmo novo: [descrever]"

echo " - Motivo: [explicar]"

echo " - Golden File: PRICE\_001 atualizado"

echo " - motorVersion: 0.2.1"

echo " - Aprovação: Especialista João Silva (2025-10-15)'"

;;

B)

echo ""

echo "❌ Bug detectado no código"

echo ""

echo "📋 Processo para corrigir bug:"

echo " 1. Identificar local do bug:"

git diff HEAD~5 packages/engine/src/amortization/price.ts

echo ""

echo " 2. Corrigir código"

echo " 3. Rodar testes novamente:"

echo " pnpm -F @finmath/engine test:unit"

echo " pnpm -F @finmath/engine test:golden"

echo ""

echo " 4. Verificar que GF passa:"

pnpm -F @finmath/engine test:golden -- --file=PRICE\_001

echo ""

echo " 5. Commit:"

echo " git add packages/engine/src/amortization/price.ts"

echo " git commit -m 'fix(H9): Corrige cálculo PMT"

echo ""

echo " - Bug: [descrever bug]"

echo " - Causa: [explicar causa]"

echo " - Solução: [descrever solução]"

echo " - Golden Files: PRICE\_001 agora passa'"

;;

C)

echo ""

echo "⚠️ Erro no Golden File original"

echo ""

echo "📋 Processo para corrigir GF:"

echo " 1. Validar cálculo manualmente:"

echo " - Usar calculadora financeira HP-12C"

echo " - Ou Excel com fórmulas"

echo " - Ou Python decimal.Decimal"

echo ""

echo " 2. Confirmar que código está correto"

echo " 3. Recalcular GF:"

echo " pnpm -F @finmath/engine golden:recalc -- --file=PRICE\_001"

echo ""

echo " 4. Revisar com especialista"

echo " 5. Commit:"

echo " git add packages/engine/test/golden/sprint2/PRICE\_001.json"

echo " git commit -m 'fix(test): Corrige Golden File PRICE\_001"

echo ""

echo " - Problema: GF original estava incorreto"

echo " - Valor esperado: 946.56"

echo " - Valor correto: 946.57"

echo " - Validação: HP-12C + especialista"

echo " - Aprovação: Especialista João Silva (2025-10-15)'"

;;

\*)

echo "❌ Opção inválida"

exit 1

;;

esac

echo ""

echo "✅ Processo documentado"

echo "📋 Siga os passos acima para resolver"

**📊 Decisão Tree:**

GF falhou?

│

├─ Mudança intencional? → Atualizar GF + motorVersion + ADR

│

├─ Bug no código? → Corrigir código + testar

│

└─ Erro no GF original? → Validar + recalcular GF

**3. Conflito de Merge (Início de Sprint)**

#!/bin/bash

# resolver-conflito.sh

echo "🔀 RESOLVENDO CONFLITO DE MERGE"

# 1. Detectar conflito

git pull origin main

if [ $? -ne 0 ]; then

echo "❌ Conflito detectado ao fazer pull"

# 2. Estratégia: Abortar e refazer

echo "📋 Estratégia: Abortar merge e refazer clean"

# 3. Fazer backup local

BACKUP\_BRANCH="backup-local-$(date +%Y%m%d-%H%M%S)"

git branch $BACKUP\_BRANCH

echo "✅ Backup criado: $BACKUP\_BRANCH"

# 4. Abortar merge

git merge --abort

echo "✅ Merge abortado"

# 5. Forçar sincronização

echo "🔄 Forçando sincronização com origin/main..."

git fetch origin

git reset --hard origin/main

echo "✅ Sincronizado com origin/main"

# 6. Verificar estado

echo "🔍 Verificando estado..."

pnpm install

pnpm -F @finmath/engine test

pnpm -F @finmath/engine test:golden

if [ $? -eq 0 ]; then

echo "✅ Estado limpo - pronto para continuar"

else

echo "❌ Estado com problemas - investigar"

exit 1

fi

# 7. Reaplicar mudanças locais (se necessário)

echo ""

echo "📋 Mudanças locais estão no backup:"

echo " Branch: $BACKUP\_BRANCH"

echo " "

echo " Para reaplicar mudanças:"

echo " 1. git diff $BACKUP\_BRANCH > mudancas.patch"

echo " 2. Revisar mudanças.patch manualmente"

echo " 3. Aplicar mudanças relevantes manualmente"

echo " 4. NÃO usar git apply (pode reintroduzir conflitos)"

echo ""

echo "⚠️ IMPORTANTE: Reaplicar MANUALMENTE com CUIDADO"

echo " - Copiar código linha por linha"

echo " - Testar após cada mudança"

echo " - Verificar que testes passam"

else

echo "✅ Pull bem-sucedido - sem conflitos"

fi

# 8. Deletar backup após verificação

read -p "Deletar branch de backup $BACKUP\_BRANCH? (s/n): " DELETE\_BACKUP

if [ "$DELETE\_BACKUP" == "s" ]; then

git branch -D $BACKUP\_BRANCH

echo "✅ Backup deletado"

else

echo "✅ Backup mantido: $BACKUP\_BRANCH"

fi

**📊 Fluxo de Resolução:**

Conflito detectado

↓

Criar backup local

↓

Abortar merge

↓

Reset hard origin/main

↓

Testar estado limpo

↓

Reaplicar mudanças MANUALMENTE

↓

Testar novamente

↓

Deletar backup (opcional)

**📊 MÉTRICAS DE PROCESSO**

**KPIs de Workflow (Por Sprint)**

#!/bin/bash

# metricas-sprint.sh

SPRINT\_NUM=$1

echo "📊 ========================================="

echo "📊 MÉTRICAS DE WORKFLOW - SPRINT ${SPRINT\_NUM}"

echo "📊 ========================================="

# 1. Commits

echo ""

echo "📈 COMMITS:"

TOTAL\_COMMITS=$(git log sprint-${SPRINT\_NUM} --oneline | wc -l)

FEAT\_COMMITS=$(git log sprint-${SPRINT\_NUM} --oneline | grep 'feat(' | wc -l)

FIX\_COMMITS=$(git log sprint-${SPRINT\_NUM} --oneline | grep 'fix(' | wc -l)

DOCS\_COMMITS=$(git log sprint-${SPRINT\_NUM} --oneline | grep 'docs(' | wc -l)

TEST\_COMMITS=$(git log sprint-${SPRINT\_NUM} --oneline | grep 'test(' | wc -l)

REFACTOR\_COMMITS=$(git log sprint-${SPRINT\_NUM} --oneline | grep 'refactor(' | wc -l)

echo " • Total: ${TOTAL\_COMMITS}"

echo " • Features (feat): ${FEAT\_COMMITS} (${FEAT\_COMMITS}/${TOTAL\_COMMITS} = $((FEAT\_COMMITS \* 100 / TOTAL\_COMMITS))%)"

echo " • Correções (fix): ${FIX\_COMMITS}"

echo " • Documentação (docs): ${DOCS\_COMMITS}"

echo " • Testes (test): ${TEST\_COMMITS}"

echo " • Refatoração (refactor): ${REFACTOR\_COMMITS}"

# 2. Qualidade

echo ""

echo "🎯 QUALIDADE:"

# Cobertura de testes

COVERAGE=$(pnpm -F @finmath/engine test:coverage --silent | grep 'All files' | awk '{print $10}')

echo " • Cobertura de testes: ${COVERAGE}"

# Golden Files

GOLDEN\_FILES=$(find packages/engine/test/golden/sprint${SPRINT\_NUM} -name "\*.json" 2>/dev/null | wc -l)

GOLDEN\_PASS=$(pnpm -F @finmath/engine test:golden --silent 2>&1 | grep 'PASS' | wc -l)

echo " • Golden Files: ${GOLDEN\_PASS}/${GOLDEN\_FILES} passando"

# Lint warnings

LINT\_WARNINGS=$(pnpm -F @finmath/engine lint 2>&1 | grep 'warning' | wc -l)

echo " • Lint warnings: ${LINT\_WARNINGS}"

# TypeScript errors

TS\_ERRORS=$(pnpm -F @finmath/engine typecheck 2>&1 | grep 'error TS' | wc -l)

echo " • TypeScript errors: ${TS\_ERRORS}"

# 3. Tempo

echo ""

echo "⏱️ TEMPO:"

# Datas

FIRST\_COMMIT\_DATE=$(git log sprint-${SPRINT\_NUM} --reverse --format=%cd --date=short | head -1)

LAST\_COMMIT\_DATE=$(git log sprint-${SPRINT\_NUM} --format=%cd --date=short | head -1)

echo " • Início: ${FIRST\_COMMIT\_DATE}"

echo " • Fim: ${LAST\_COMMIT\_DATE}"

# Calcular duração (simplificado)

FIRST\_TIMESTAMP=$(date -d "${FIRST\_COMMIT\_DATE}" +%s)

LAST\_TIMESTAMP=$(date -d "${LAST\_COMMIT\_DATE}" +%s)

DURATION\_DAYS=$(( (LAST\_TIMESTAMP - FIRST\_TIMESTAMP) / 86400 ))

echo " • Duração: ${DURATION\_DAYS} dias"

# 4. Produtividade

echo ""

echo "🚀 PRODUTIVIDADE:"

# Commits por dia

COMMITS\_PER\_DAY=$(echo "scale=1; ${TOTAL\_COMMITS} / ${DURATION\_DAYS}" | bc)

echo " • Commits/dia: ${COMMITS\_PER\_DAY}"

# Features por dia

FEAT\_PER\_DAY=$(echo "scale=1; ${FEAT\_COMMITS} / ${DURATION\_DAYS}" | bc)

echo " • Features/dia: ${FEAT\_PER\_DAY}"

# 5. Linhas de código

echo ""

echo "📏 LINHAS DE CÓDIGO:"

# Adições

LINES\_ADDED=$(git log sprint-${SPRINT\_NUM} --numstat --pretty="%H" | awk 'NF==3 {plus+=$1} END {print plus}')

echo " • Linhas adicionadas: ${LINES\_ADDED}"

# Remoções

LINES\_REMOVED=$(git log sprint-${SPRINT\_NUM} --numstat --pretty="%H" | awk 'NF==3 {minus+=$2} END {print minus}')

echo " • Linhas removidas: ${LINES\_REMOVED}"

# Líquido

LINES\_NET=$((LINES\_ADDED - LINES\_REMOVED))

echo " • Líquido: ${LINES\_NET}"

# 6. Arquivos

echo ""

echo "📁 ARQUIVOS:"

FILES\_CHANGED=$(git log sprint-${SPRINT\_NUM} --numstat --pretty="%H" | awk 'NF==3' | wc -l)

echo " • Arquivos modificados: ${FILES\_CHANGED}"

# 7. Complexidade

echo ""

echo "🧮 COMPLEXIDADE:"

# Complexidade ciclomática média (aproximação via contador de ifs/loops)

AVG\_COMPLEXITY=$(grep -rE '(if|for|while|switch)' packages/engine/src --include="\*.ts" | wc -l)

TOTAL\_FUNCTIONS=$(grep -rE '(function|const.\*=.\*=>)' packages/engine/src --include="\*.ts" | wc -l)

AVG\_PER\_FUNCTION=$(echo "scale=1; ${AVG\_COMPLEXITY} / ${TOTAL\_FUNCTIONS}" | bc)

echo " • Complexidade média/função: ${AVG\_PER\_FUNCTION}"

# 8. Testes

echo ""

echo "🧪 TESTES:"

# Testes unitários

UNIT\_TESTS=$(grep -r 'test(' packages/engine/test/unit --include="\*.test.ts" | wc -l)

echo " • Testes unitários: ${UNIT\_TESTS}"

# Testes de propriedade

PROPERTY\_TESTS=$(grep -r 'fc.assert' packages/engine/test/property --include="\*.test.ts" | wc -l)

echo " • Testes de propriedade: ${PROPERTY\_TESTS}"

# Tempo de execução

TEST\_TIME=$(pnpm -F @finmath/engine test --silent 2>&1 | grep 'Time:' | awk '{print $2}')

echo " • Tempo de execução: ${TEST\_TIME}"

# 9. Build

echo ""

echo "🏗️ BUILD:"

# Tamanho do build

BUILD\_SIZE=$(du -sh packages/engine/dist 2>/dev/null | awk '{print $1}')

echo " • Tamanho: ${BUILD\_SIZE}"

# Tempo de build

BUILD\_START=$(date +%s)

pnpm -F @finmath/engine build --silent > /dev/null 2>&1

BUILD\_END=$(date +%s)

BUILD\_TIME=$((BUILD\_END - BUILD\_START))

echo " • Tempo de build: ${BUILD\_TIME}s"

# 10. Resumo

echo ""

echo "📊 ========================================="

echo "📊 RESUMO"

echo "📊 ========================================="

echo " ✅ Sprint ${SPRINT\_NUM} concluída em ${DURATION\_DAYS} dias"

echo " ✅ ${FEAT\_COMMITS} features implementadas"

echo " ✅ ${GOLDEN\_FILES} Golden Files criados"

echo " ✅ ${COVERAGE} cobertura de testes"

echo " ✅ ${LINT\_WARNINGS} warnings de lint"

echo " ✅ ${TS\_ERRORS} erros TypeScript"

echo ""

echo "📊 ========================================="

**📊 Exemplo de Saída:**

📊 =========================================

📊 MÉTRICAS DE WORKFLOW - SPRINT 2

📊 =========================================

📈 COMMITS:

• Total: 47

• Features (feat): 35 (74%)

• Correções (fix): 8

• Documentação (docs): 4

• Testes (test): 0

• Refatoração (refactor): 0

🎯 QUALIDADE:

• Cobertura de testes: 87%

• Golden Files: 10/10 passando

• Lint warnings: 0

• TypeScript errors: 0

⏱️ TEMPO:

• Início: 2025-10-01

• Fim: 2025-10-15

• Duração: 14 dias

🚀 PRODUTIVIDADE:

• Commits/dia: 3.4

• Features/dia: 2.5

📏 LINHAS DE CÓDIGO:

• Linhas adicionadas: 3452

• Linhas removidas: 456

• Líquido: 2996

📁 ARQUIVOS:

• Arquivos modificados: 89

🧮 COMPLEXIDADE:

• Complexidade média/função: 3.2

🧪 TESTES:

• Testes unitários: 127

• Testes de propriedade: 45

• Tempo de execução: 8.5s

🏗️ BUILD:

• Tamanho: 245KB

• Tempo de build: 12s

📊 =========================================

📊 RESUMO

📊 =========================================

✅ Sprint 2 concluída em 14 dias

✅ 35 features implementadas

✅ 10 Golden Files criados

✅ 87% cobertura de testes

✅ 0 warnings de lint

✅ 0 erros TypeScript

📊 =========================================

**🆕 GESTÃO DE DEPENDÊNCIAS**

**Atualização de Dependências**

#!/bin/bash

# atualizar-dependencias.sh

echo "📦 ATUALIZANDO DEPENDÊNCIAS"

# 1. Verificar versões atualizadas disponíveis

echo "🔍 Verificando versões disponíveis..."

pnpm outdated

# 2. Atualizar dependências dev (mais seguro)

echo ""

echo "🔄 Atualizando dependências de desenvolvimento..."

pnpm update --dev

# 3. Rodar testes

echo ""

echo "🧪 Rodando testes..."

pnpm -F @finmath/engine test

if [ $? -ne 0 ]; then

echo "❌ Testes falharam após atualização de dev deps"

echo "💡 Reverter: git checkout packages/\*/package.json pnpm-lock.yaml"

exit 1

fi

# 4. Atualizar dependências de produção (CUIDADO!)

read -p "Atualizar dependências de produção? (s/n): " UPDATE\_PROD

if [ "$UPDATE\_PROD" == "s" ]; then

echo ""

echo "⚠️ Atualizando dependências de produção..."

echo " ATENÇÃO: Pode quebrar compatibilidade!"

pnpm update

# Rodar bateria completa de testes

echo ""

echo "🧪 Rodando bateria completa..."

pnpm -F @finmath/engine test

pnpm -F @finmath/engine test:golden

pnpm -F @finmath/engine build

if [ $? -ne 0 ]; then

echo "❌ Testes falharam após atualização de prod deps"

echo "💡 Reverter: git checkout packages/\*/package.json pnpm-lock.yaml"

exit 1

fi

echo "✅ Dependências atualizadas com sucesso!"

echo "📋 Commit as mudanças:"

echo " git add packages/\*/package.json pnpm-lock.yaml"

echo " git commit -m 'chore: Atualiza dependências'"

fi

# 5. Verificar vulnerabilidades

echo ""

echo "🔒 Verificando vulnerabilidades..."

pnpm audit

CRITICAL\_VULNS=$(pnpm audit --json | jq '.metadata.vulnerabilities.critical')

if [ $CRITICAL\_VULNS -gt 0 ]; then

echo "❌ ${CRITICAL\_VULNS} vulnerabilidades críticas encontradas!"

echo "💡 Corrigir: pnpm audit fix"

else

echo "✅ Nenhuma vulnerabilidade crítica"

fi

**📦 Política de Dependências:**

* **Produção**: Atualizar apenas em sprints dedicadas
* **Desenvolvimento**: Atualizar semanalmente
* **Vulnerabilidades críticas**: Atualizar IMEDIATAMENTE
* **Major versions**: Criar ADR antes de atualizar
* **Testes**: SEMPRE rodar bateria completa após atualização

**🆕 AMBIENTE DE DESENVOLVIMENTO**

**Setup Completo do Ambiente**

#!/bin/bash

# setup-ambiente.sh

echo "🚀 SETUP COMPLETO DO AMBIENTE FINMATH"

# 1. Node.js

echo "📦 Verificando Node.js..."

NODE\_VERSION=$(node -v | cut -d'v' -f2 | cut -d'.' -f1)

if [ $NODE\_VERSION -lt 18 ]; then

echo "❌ Node.js 18+ é necessário (encontrado: v$(node -v))"

echo "💡 Instalar: https://nodejs.org/"

exit 1

fi

echo "✅ Node.js $(node -v)"

# 2. pnpm

echo ""

echo "📦 Verificando pnpm..."

if ! command -v pnpm &> /dev/null; then

echo "❌ pnpm não encontrado"

echo "💡 Instalar: npm install -g pnpm"

exit 1

fi

echo "✅ pnpm $(pnpm -v)"

# 3. Git

echo ""

echo "📦 Verificando Git..."

if ! command -v git &> /dev/null; then

echo "❌ Git não encontrado"

echo "💡 Instalar: https://git-scm.com/"

exit 1

fi

echo "✅ Git $(git --version)"

# 4. Clonar repositório (se necessário)

echo ""

echo "📁 Verificando repositório..."

if [ ! -d ".git" ]; then

echo "❌ Não está em um repositório Git"

read -p "Clonar repositório? (s/n): " CLONE

if [ "$CLONE" == "s" ]; then

git clone https://github.com/PrinceOfEgypt1/fin-math.git

cd fin-math

echo "✅ Repositório clonado"

else

echo "❌ Setup cancelado"

exit 1

fi

else

echo "✅ Repositório Git encontrado"

fi

# 5. Instalar dependências

echo ""

echo "📦 Instalando dependências..."

pnpm install

if [ $? -ne 0 ]; then

echo "❌ Falha ao instalar dependências"

exit 1

fi

echo "✅ Dependências instaladas"

# 6. Configurar Git hooks

echo ""

echo "🔧 Configurando Git hooks..."

cat > .git/hooks/pre-commit << 'EOF'

#!/bin/bash

echo "🔍 Pre-commit hook: Validando código..."

# Lint

pnpm -F @finmath/engine lint

if [ $? -ne 0 ]; then

echo "❌ Lint falhou - corrigir antes de commitar"

exit 1

fi

# Type check

pnpm -F @finmath/engine typecheck

if [ $? -ne 0 ]; then

echo "❌ Type check falhou - corrigir antes de commitar"

exit 1

fi

echo "✅ Validação passou"

EOF

chmod +x .git/hooks/pre-commit

echo "✅ Git hooks configurados"

# 7. Criar estrutura de diretórios (se necessário)

echo ""

echo "📁 Verificando estrutura de diretórios..."

DIRS=(

"packages/engine/test/unit"

"packages/engine/test/property"

"packages/engine/test/integration"

"packages/engine/test/golden"

"packages/engine/test/e2e"

"docs/adr"

"docs/api"

"docs/guides"

)

for DIR in "${DIRS[@]}"; do

if [ ! -d "$DIR" ]; then

mkdir -p "$DIR"

echo " • Criado: $DIR"

fi

done

echo "✅ Estrutura de diretórios OK"

# 8. Verificar ambiente

echo ""

echo "🔍 Verificando ambiente..."

pnpm -F @finmath/engine typecheck

pnpm -F @finmath/engine lint

pnpm -F @finmath/engine test

pnpm -F @finmath/engine build

if [ $? -eq 0 ]; then

echo ""

echo "✅ ========================================="

echo "✅ AMBIENTE PRONTO PARA DESENVOLVIMENTO!"

echo "✅ ========================================="

echo ""

echo "📋 Próximos passos:"

echo " 1. Criar branch da sprint: git checkout -b sprint-X"

echo " 2. Ler backlog: cat docs/backlog/sprint-X.md"

echo " 3. Começar desenvolvimento!"

echo ""

else

echo ""

echo "❌ Ambiente com problemas - verificar logs acima"

exit 1

fi

**🔧 Ferramentas Recomendadas:**

* **IDE**: VSCode com extensões:
  + ESLint
  + Prettier
  + TypeScript
  + GitLens
* **Terminal**: iTerm2 (Mac) ou Windows Terminal
* **Git Client**: GitKraken ou SourceTree (opcional)
* **API Testing**: Postman ou Insomnia

**🆕 CI/CD PIPELINE**

**GitHub Actions Workflow**

# .github/workflows/ci.yml

name: CI Pipeline

on:

pull\_request:

branches: [main]

push:

branches: [main]

jobs:

validate:

runs-on: ubuntu-latest

strategy:

matrix:

node-version: [18.x, 20.x]

steps:

- name: 📥 Checkout

uses: actions/checkout@v3

- name: 📦 Setup Node.js ${{ matrix.node-version }}

uses: actions/setup-node@v3

with:

node-version: ${{ matrix.node-version }}

- name: 📦 Setup pnpm

uses: pnpm/action-setup@v2

with:

version: 8

- name: 📦 Install dependencies

run: pnpm install --frozen-lockfile

- name: 🔍 Type Check

run: pnpm -F @finmath/engine typecheck

- name: 🔍 Lint

run: pnpm -F @finmath/engine lint

- name: 🧪 Unit Tests

run: pnpm -F @finmath/engine test:unit

- name: 🧪 Property Tests

run: pnpm -F @finmath/engine test:property

- name: 🧪 Integration Tests

run: pnpm -F @finmath/engine test:integration

- name: 🧪 Golden Files

run: pnpm -F @finmath/engine test:golden

- name: 🏗️ Build

run: |

pnpm -F @finmath/engine build

pnpm -F @finmath/api build

- name: 📊 Coverage

run: pnpm -F @finmath/engine test:coverage

- name: 📤 Upload Coverage

uses: codecov/codecov-action@v3

with:

files: ./packages/engine/coverage/lcov.info

flags: engine

- name: ✅ All Checks Passed

run: echo "✅ Pipeline concluída com sucesso!"

**📊 Pipeline Stages:**

Checkout → Setup → Install → Type Check → Lint → Tests → Build → Coverage → ✅

1s 5s 30s 10s 5s 2min 20s 10s

**⏱️ Tempo Total:** ~3-4 minutos

**🆕 TROUBLESHOOTING**

**Problemas Comuns e Soluções**

**1. "pnpm install" Falha**

# Problema: Dependências não instalam

# Solução 1: Limpar cache

pnpm store prune

rm -rf node\_modules

rm pnpm-lock.yaml

pnpm install

# Solução 2: Verificar Node.js

node -v # Deve ser 18+

# Solução 3: Verificar conectividade

ping registry.npmjs.org

**2. Testes Falhando**

# Problema: Testes que passavam agora falham

# Solução 1: Limpar build anterior

pnpm -F @finmath/engine clean

pnpm -F @finmath/engine build

pnpm -F @finmath/engine test

# Solução 2: Verificar Golden Files

pnpm -F @finmath/engine test:golden -- --verbose

# Solução 3: Verificar dependências

pnpm outdated

pnpm install

**3. TypeScript Errors**

# Problema: Erros de tipo que não existiam antes

# Solução 1: Limpar cache TypeScript

rm -rf packages/\*/tsconfig.tsbuildinfo

pnpm -F @finmath/engine typecheck

# Solução 2: Verificar versão TypeScript

pnpm list typescript

# Solução 3: Reinstalar @types/\*

pnpm install -D @types/node @types/jest

**4. Build Falha**

# Problema: Build não completa

# Solução 1: Verificar espaço em disco

df -h

# Solução 2: Limpar dist/

rm -rf packages/\*/dist

pnpm -F @finmath/engine build

# Solução 3: Build verbose

pnpm -F @finmath/engine build --verbose

**5. Git Conflicts**

# Problema: Conflitos ao fazer merge/pull

# Ver seção: Casos Especiais → Conflito de Merge

./resolver-conflito.sh

**6. Swagger UI Não Abre**

# Problema: http://localhost:3001/api-docs não responde

# Solução 1: Verificar porta

lsof -i :3001

# Se ocupada, matar processo:

kill -9 <PID>

# Solução 2: Verificar build da API

pnpm -F @finmath/api build

pnpm -F @finmath/api dev

# Solução 3: Verificar OpenAPI file

cat openapi-3.1\_finmath-v1.0.yaml | head -20

**7. Performance Lenta**

# Problema: Testes muito lentos

# Solução 1: Rodar testes em paralelo

pnpm -F @finmath/engine test -- --maxWorkers=4

# Solução 2: Rodar apenas testes modificados

pnpm -F @finmath/engine test -- --onlyChanged

# Solução 3: Cachear resultados

pnpm -F @finmath/engine test -- --cache

**🆕 INTEGRAÇÃO COM FERRAMENTAS**

**Postman Collection**

# Importar coleção no Postman

# Arquivo: FinMath\_API\_v1\_postman\_collection.json

# Ou usar via CLI:

newman run FinMath\_API\_v1\_postman\_collection.json \

--environment FinMath\_API\_Local\_postman\_environment.json

**Insomnia Collection**

# Importar no Insomnia

# Arquivo: FinMath\_API\_v1\_insomnia.json

**VSCode Tasks**

// .vscode/tasks.json

{

"version": "2.0.0",

"tasks": [

{

"label": "🧪 Test",

"type": "shell",

"command": "pnpm -F @finmath/engine test",

"group": {

"kind": "test",

"isDefault": true

}

},

{

"label": "🏗️ Build",

"type": "shell",

"command": "pnpm -F @finmath/engine build",

"group": {

"kind": "build",

"isDefault": true

}

},

{

"label": "🚀 Dev Server",

"type": "shell",

"command": "pnpm -F @finmath/api dev",

"isBackground": true

}

]

}

**📚 REFERÊNCIAS RÁPIDAS**

**Comandos Essenciais**

# Desenvolvimento

pnpm -F @finmath/engine dev # Modo desenvolvimento

pnpm -F @finmath/api dev # API em dev (porta 3001)

# Testes

pnpm -F @finmath/engine test # Todos os testes

pnpm -F @finmath/engine test:unit # Só unitários

pnpm -F @finmath/engine test:golden # Só golden files

# Build

pnpm -F @finmath/engine build # Build do motor

pnpm -F @finmath/api build # Build da API

# Qualidade

pnpm -F @finmath/engine lint # Linter

pnpm -F @finmath/engine typecheck # Type checking

# Git

git status # Status

git add . # Stage all

git commit -m "feat(HX): desc" # Commit

git push origin sprint-X # Push (só ao final!)

**Arquivos Importantes**

| **Arquivo** | **Descrição** | **Localização** |
| --- | --- | --- |
| **Backlog** | Histórias da sprint | docs/backlog/sprint-X.md |
| **OpenAPI** | Contrato da API | openapi-3.1\_finmath-v1.0.yaml |
| **Guia CET** | Source of Truth de CET | docs/cet-sot/guia-cet-v1.0.md |
| **ADRs** | Decisões arquiteturais | docs/adr/ADR-XXX-titulo.md |
| **Golden Files** | Testes de regressão | packages/engine/test/golden/ |

**Links Úteis**

* **GitHub**: https://github.com/PrinceOfEgypt1/fin-math
* **Swagger UI**: http://localhost:3001/api-docs
* **BCB Resolução 4.935**: https://www.bcb.gov.br/...
* **decimal.js Docs**: https://mikemcl.github.io/decimal.js/

**🎓 CHANGELOG DO DOCUMENTO**

**v1.1 (2025-10-23)**

* 🆕 Adicionada seção: Gestão de Dependências
* 🆕 Adicionada seção: Ambiente de Desenvolvimento
* 🆕 Adicionada seção: CI/CD Pipeline
* 🆕 Adicionada seção: Troubleshooting
* 🆕 Adicionada seção: Integração com Ferramentas
* ✨ Expandida seção: Métricas de Processo (script completo)
* ✨ Expandida seção: Casos Especiais (hotfix, golden files, conflitos)
* 📝 Melhorada formatação geral
* 📝 Adicionados mais exemplos práticos
* 📝 Adicionadas referências rápidas

**v1.0 (2025-10-11)**

* 🚀 Criação inicial
* 📋 Adaptação para workflow "push ao final da sprint"
* ✅ Regras adaptadas: trabalho local + validação final
* 🔄 Scripts: inicio-sprint.sh + finalizar-sprint.sh
* 📝 Templates de commit específicos
* 🚨 Proibições e casos especiais documentados

**📞 SUPORTE**

Em caso de dúvidas ou problemas não cobertos neste documento:

1. **Verificar documentação técnica**: docs/
2. **Consultar ADRs**: docs/adr/
3. **Verificar issues no GitHub**: [Issues](https://github.com/PrinceOfEgypt1/fin-math/issues)
4. **Abrir nova issue** com:
   * Descrição do problema
   * Passos para reproduzir
   * Logs relevantes
   * Ambiente (OS, Node.js version, etc.)

**📌 Este documento é VIVO e deve ser atualizado conforme o projeto evolui.**

**Última atualização:** 23/Outubro/2025  
**Próxima revisão:** Ao final de cada sprint  
**Mantenedor:** FinMath Core Team

**🎯 FIM DO DOCUMENTO**