

# User Rules Guide

---

## O que são User Rules

---

User Rules são regras **globais** aplicadas a todos os projetos no Cursor IDE. Elas são configuradas uma única vez nas configurações do usuário e ficam sempre ativas, independente do projeto atual.

## Características das User Rules

---

### Formato

- **Texto puro:** Sem front-matter, sem metadados
- **Simples:** Direto ao ponto, sem estruturas complexas
- **Conciso:** Máximo 500-1000 caracteres para evitar token bloat

### Escopo

- **Global:** Aplicadas a todos os projetos
- **Persistente:** Sempre ativas
- **Pessoal:** Refletem preferências pessoais do desenvolvedor

### Propósito

- **Estilo de escrita:** Como o agente deve se comunicar
- **Guard-rails básicos:** Controles fundamentais de segurança
- **Preferências gerais:** Abordagens preferidas de desenvolvimento

## Como Configurar

---

### Acesso às Configurações

1. Abra o Cursor IDE
2. Vá para **Settings** (Ctrl/Cmd + ,)
3. Procure por **Rules** na barra de pesquisa
4. Cole suas regras na caixa de texto

## Exemplo de User Rules Recomendadas

Você é um agente sênior. **Prioridades:** DRY ☐ KISS ☐ YAGNI ☐ SOLID ☐ DDD/Clean.

### Execução:

- Faça apenas o próximo passo e espere "Go/No-Go".
- Patch-**only** ☐ máx. 5 arquivos por ciclo, 200 linhas/arquivo, 500 linhas **no** total.
- Nunca **presuma**: se faltar dado, dispare ASSUMPTION\_REQUEST com opções e recomendação.
- **RISCO**: se envolver secrets, migrações, deleção em massa, auth, config de produção ☐ pare e peça validação (RISK\_ALERT).
- Qualquer trabalho fora do escopo atual ☐ SCOPE\_CHANGE com impacto e alternativas.

### Qualidade:

- Sem artefatos/dummies. Tudo precisa ter uso **real**.
- Cada mudança deve declarar contrato I/O (REQUEST/RESPONSE) + critérios de aceite resumidos.
- Sem testes, sem **merge**. Cobertura alvo ☐ 80% **no** módulo tocado.

### Operação:

- Consulte MCP/Context e docs relevantes antes de decisões.
- Optimize **tokens**: respostas objetivas, diffs unificados, sem contexto inútil.

## Boas Práticas

### O que INCLUIR nas User Rules

#### Controles Fundamentais

- Execução step-by-step
- Limites de patch-only
- Protocolos de segurança (RISK\_ALERT)
- Gestão de suposições (ASSUMPTION\_REQUEST)

#### Preferências de Estilo

- Tom de comunicação preferido
- Nível de detalhamento desejado
- Formato de respostas preferido

#### Princípios de Desenvolvimento

- Hierarquia de princípios (DRY, KISS, YAGNI, SOLID)
- Abordagem arquitetural preferida
- Padrões de qualidade mínimos

### O que NÃO incluir nas User Rules

#### Detalhes Específicos de Projeto

- ☒ Configurações específicas de frameworks
- ☒ Estruturas de diretórios detalhadas
- ☒ Padrões específicos de linguagem
- ☒ Configurações de ferramentas específicas

#### Conteúdo Extenso

- ☒ Documentação completa de APIs
- ☒ Exemplos de código longos

- ❌ Checklists detalhados
- ❌ Especificações técnicas completas

## Regras Contextuais

- ❌ Regras que só se aplicam a certos tipos de arquivo
- ❌ Configurações específicas de ambiente
- ❌ Padrões que variam por projeto

## Exemplos por Perfil de Desenvolvedor

### Desenvolvedor Júnior

Você é um mentor experiente. Seja didático e explique o "porquê" das decisões.

Execução:

- Um passo por vez, aguarde confirmação.
- Máximo 3 arquivos por mudança.
- Sempre explique riscos antes de executar.
- Se não souber algo, pergunte ao invés de assumir.

Qualidade:

- Priorize legibilidade sobre performance.
- Sempre inclua testes para código novo.
- Documente decisões importantes.

### Desenvolvedor Sênior

Você é um par técnico experiente. Seja conciso e focado em resultados.

Execução:

- Step-by-step com **Go/No-Go** explícito.
- Patch-**only** 5 arquivos **max**, 200 linhas/arquivo.
- ASSUMPTION\_REQUEST para ambiguidades.
- RISK\_ALERT para operações sensíveis.

Qualidade:

- SOLID + DDD sempre.
- 80%+ cobertura de testes.
- Zero débito técnico intencional.

### Tech Lead

Você é um arquiteto sênior. Foque em decisões arquiteturais e impacto no time.

Execução:

- Considere impacto em todo o sistema.
- Documente decisões arquiteturais (ADRs).
- Avalie trade-offs de performance vs manutenibilidade.
- SCOPE\_CHANGE para mudanças que afetam outros times.

Qualidade:

- Arquitetura limpa não-negociável.
- Padrões consistentes em todo o codebase.
- Observabilidade e monitoramento obrigatórios.

## Validação e Teste

### Como Testar suas User Rules

1. **Teste em projeto simples:** Crie um projeto pequeno para testar
2. **Verifique comportamento:** O agente segue suas regras?
3. **Ajuste gradualmente:** Refine as regras baseado no comportamento
4. **Monitore token usage:** Regras muito longas consomem muitos tokens

### Sinais de User Rules Problemáticas

#### Token Bloat

- Respostas muito longas do agente
- Contexto desnecessário sendo incluído
- Performance degradada

#### Conflitos

- Agente confuso entre regras globais e específicas
- Comportamento inconsistente entre projetos
- Regras contradizendo umas às outras

#### Rigidez Excessiva

- Agente não consegue adaptar-se a contextos específicos
- Regras muito restritivas para certos tipos de projeto
- Perda de flexibilidade

## Migração de Regras Existentes


### De Regras Monolíticas para User Rules

Se você tem um arquivo grande de regras, siga este processo:

1. **Identifique o essencial:** Quais regras são realmente globais?
2. **Extraia preferências pessoais:** Estilo, tom, abordagem
3. **Mantenha controles básicos:** Segurança, qualidade mínima
4. **Mova específicos para Project Rules:** Detalhes técnicos vão para `.cursor/rules/`

### Exemplo de Migração

#### Antes (Monolítico):

Você  um agente sênior especializado em TypeScript, React, Next.js, Python, FastAPI...  
[3000+ linhas de regras específicas]

#### Depois (User Rules):

Você  um agente sênior. **Prioridades:** DRY  KISS  YAGNI  SOLID  DDD/Clean.  
[200 linhas de regras essenciais]

#### Project Rules ( `.cursor/rules/*.mdc` ):

- Regras específicas de TypeScript

- Padrões de React/Next.js
- Configurações de Python/FastAPI

## Troubleshooting

---

### Problemas Comuns

#### Agente Ignora User Rules

- **Causa:** Regras muito longas ou conflitantes
- **Solução:** Simplifique e torne mais diretas

#### Token Limit Exceeded

- **Causa:** User Rules muito extensas
- **Solução:** Mova detalhes para Project Rules

#### Comportamento Inconsistente

- **Causa:** Conflito entre User Rules e Project Rules
- **Solução:** Revise hierarquia e remova duplicações

### Debug de User Rules

1. **Teste isoladamente:** Desative Project Rules temporariamente
2. **Monitore logs:** Observe como o agente interpreta as regras
3. **Iteração gradual:** Adicione regras uma por vez
4. **Feedback direto:** Pergunte ao agente como ele interpreta suas regras

## Referências

---

- [Cursor Rules Documentation](https://docs.cursor.com/en/context/rules) (<https://docs.cursor.com/en/context/rules>)
- [Project Rules Guide](#) (project-rules-guide.md)
- [Migration Guide](#) (migration-guide.md)
- [Templates](#) (../templates/)