

# **Resenha: Documenting Architecture Decisions (Michael Nygard, 2011)**

Link do artigo: [cognitect.com/blog/2011/11/15/documenting-architecture-decisions.html](http://cognitect.com/blog/2011/11/15/documenting-architecture-decisions.html)

## **Ideia central**

O artigo apresenta os Architecture Decision Records (ADRs) como um artefato leve e versionado para registrar decisões arquiteturais significativas. Em vez de grandes documentos que envelhecem e ninguém lê, os ADRs vivem no repositório de código, preservam o contexto e o 'porquê' das escolhas técnicas, e reduzem o risco de aceitação ou reversão cegas no futuro.

## **O que o artigo defende (e como)**

- Por que ADRs: em projetos ágeis, as decisões surgem de forma incremental. O que mais se perde é o motivo por trás das escolhas; os ADRs capturam essa motivação e suas consequências.
- O que é arquiteturalmente significativo: decisões que afetam estrutura, qualidades não funcionais, dependências, interfaces e técnicas de construção.
- Formato enxuto: cada ADR tem Título, Contexto, Decisão, Status e Consequências. É um texto curto, claro e direto.
- Governança e ciclo de vida: numerar sequencialmente (ADR-001, ADR-002...), manter no repositório e, quando uma decisão for substituída, marcar como 'superseded' apontando para o novo ADR.
- Relato de experiência: aplicação prática em projetos mostrou bom retorno para preservar intenções de longo prazo sem travar futuras mudanças.

## **Forças do texto**

- Praticidade: formato mínimo viável que qualquer time pode adotar rapidamente.
- Integração com o fluxo real: no mesmo repositório e sob controle de versão, com rastreabilidade via PRs e histórico.
- Clareza pedagógica: estrutura simples e orientada à ação, fácil de ensinar e padronizar.

## **Limitações e cuidados**

- Risco de inflar a documentação se tentar registrar toda e qualquer decisão. Foque no que é realmente significativo.
- Exige cultura de escrita: é necessário registrar contexto e consequências (positivas e negativas), não apenas atas.
- Acesso de stakeholders não técnicos pode exigir um espelho em wiki, dependendo do time.

## **Por que isso importa (para estudo e prática)**

Para quem está estudando e montando portfólio, ADRs oferecem trilha de raciocínio e transparência. Você registra por que escolheu uma tecnologia (por exemplo, Flask vs. Laravel), padrões de integração,

banco de dados (PostgreSQL) e aceita os trade-offs envolvidos. Isso facilita justificar decisões para revisores, professores, recrutadores ou colaboradores.

## Mini-guia de implementação (passo a passo)

- Criar diretório: doc/arch/ (ou docs/adr/).
- Padrão de arquivo: adr-0001-título-curto.md.
- Template mínimo: Título, Contexto, Decisão, Status (Proposed/Accepted/Superseded) e Consequências (prós, contras, impactos).
- Higiene: um ADR por decisão; se mudar, crie um novo ADR e marque o anterior como substituído (superseded).

## Exemplos práticos

- Data Science/APIs: “ADR-0005: Padronizar Pandas vs. Polars?” — decide por Pandas pelas bibliotecas do ecossistema; consequência: guias de performance e treinamento do time.
- Web/Laravel: “ADR-0007: Livewire (stateful) vs. SPA (React)” — optar por Livewire para reduzir JS custom, aceitando acoplamento ao ciclo de request do Laravel.
- Banco de dados: “ADR-0012: Postgres + UUID v7” — justifica IDs, índices e implicações de migração.

## Conclusão

O texto de Nygard inaugura um padrão disciplinado e acessível: capturar decisão, contexto e consequências dentro do repositório, com linguagem clara. Esse hábito aumenta a memória arquitetural, reduz o medo de mexer no legado e facilita a evolução consciente do software.