Documenting Architecture Decisions

Aluno: Vitor Veiga Silva

O texto “Documenting Architecture Decisions”, de Michael Nygard, apresenta uma proposta prática para registrar decisões importantes de arquitetura em projetos de software, especialmente em contextos ágeis. O autor explica que, nas metodologias ágeis, a arquitetura não é totalmente definida no início do projeto, mas evolui à medida que o desenvolvimento avança e novas necessidades surgem. Apesar disso, é fundamental documentar as decisões arquiteturais, pois, sem esse registro, as razões por trás de determinadas escolhas podem se perder com o tempo. Isso dificulta a compreensão do sistema, prejudica a manutenção e pode levar a retrabalhos ou decisões incorretas no futuro.

Nygard argumenta que o problema não está na documentação em si, mas em produzir documentos extensos, pesados e de pouca utilidade prática. Esse tipo de material tende a ficar desatualizado rapidamente e raramente é consultado. Em vez de criar relatórios longos e complexos, o autor propõe o uso dos chamados Registros de Decisões de Arquitetura, conhecidos como ADRs (Architecture Decision Records). Esses registros são documentos curtos, objetivos e fáceis de manter, que servem para registrar decisões realmente relevantes para o projeto — aquelas que afetam a estrutura do sistema, os padrões tecnológicos, as dependências, as interfaces e os aspectos não funcionais, como desempenho, segurança e escalabilidade.

Cada ADR deve abordar apenas uma decisão central e seguir um formato simples, composto por algumas seções principais. O autor sugere que o documento contenha um título que identifique claramente a decisão, um contexto que descreva as forças e restrições que influenciaram a escolha, a decisão propriamente dita, o status (indicando se está proposta, aceita, substituída ou descontinuada) e as consequências (efeitos positivos e negativos decorrentes da decisão). Essa estrutura torna o registro claro e direto, permitindo que qualquer membro da equipe compreenda rapidamente a motivação e o impacto de cada escolha, mesmo que não tenha participado do processo decisório original.

Os ADRs devem ser armazenados junto com o código-fonte do projeto, no mesmo repositório, para que possam evoluir de forma sincronizada com o sistema. Cada novo ADR recebe um número sequencial e único. Mesmo quando uma decisão é alterada ou deixada de lado, o documento original não deve ser apagado — apenas marcado como “substituído” ou “obsoleto”. Isso preserva o histórico de decisões e garante a rastreabilidade das mudanças arquiteturais ao longo do tempo.

Nygard destaca que a adoção dos ADRs traz diversos benefícios. Eles proporcionam maior clareza e transparência sobre as decisões técnicas, facilitam a comunicação entre os membros da equipe e reduzem o risco de escolhas arbitrárias. Além disso, ajudam a manter a coerência da arquitetura ao longo do desenvolvimento e servem como fonte de aprendizado para novos

integrantes, que podem entender rapidamente por que certas decisões foram tomadas. Outro ponto importante é que os ADRs permitem revisar decisões antigas de maneira estruturada, quando o contexto muda, evitando que o projeto permaneça preso a escolhas ultrapassadas.

O autor também compartilha experiências positivas com a aplicação desse método em equipes reais. Segundo ele, tanto desenvolvedores quanto clientes reconhecem o valor dos ADRs por sua clareza e simplicidade. Um desafio observado é garantir que essas informações estejam acessíveis a pessoas fora da equipe técnica, como gestores e partes interessadas. No entanto, essa limitação pode ser superada facilmente por meio do uso de plataformas colaborativas, que permitem visualizar os documentos de forma simples e direta.

Em resumo, o artigo defende que documentar as decisões de arquitetura é uma prática essencial, mas que deve ser feita de modo leve, contínuo e objetivo. Os ADRs transformam a documentação em uma ferramenta útil e viva, que acompanha a evolução do sistema sem se tornar um peso para a equipe. Essa prática equilibra a necessidade de registro histórico com a agilidade dos métodos modernos de desenvolvimento, promovendo aprendizado, transparência e a preservação da integridade arquitetural ao longo de todo o ciclo de vida do software.