No artigo *Documenting Architecture Decisions*, Michael Nygard propõe uma abordagem prática e enxuta para lidar com um dos pontos mais sensíveis em projetos de software: a documentação das decisões arquiteturais. Ao contrário da prática comum de produzir documentos longos e complexos, que raramente são lidos ou atualizados, o autor defende o uso de registros simples e diretos chamados ADRs — *Architecture Decision Records*. Esses registros têm como objetivo documentar, de forma clara e objetiva, cada decisão arquitetural relevante tomada durante o ciclo de vida do projeto, oferecendo contexto, justificativas e consequências associadas.

O texto parte da realidade dos projetos ágeis, nos quais a arquitetura não deve ser pensada como algo fixo, planejado 100% no início, mas como algo que evolui conforme o projeto avança e as necessidades mudam. Nygard reconhece que nem todas as decisões são tomadas no início, e muitas surgem conforme novas informações, tecnologias ou restrições aparecem. A metodologia ágil, segundo ele, não é contrária à documentação, mas à documentação que não tem valor — ou seja, documentos grandes demais, desatualizados e que não servem ao time. Nesse cenário, ADRs surgem como uma solução prática: pequenos arquivos de texto, geralmente em Markdown, que registram uma decisão por vez, com foco em ser útil para a equipe técnica atual e futura.

Cada ADR contém apenas algumas seções essenciais: um título descritivo, o contexto que levou à tomada de decisão (incluindo fatores técnicos, sociais ou políticos), a decisão propriamente dita, o status (se está proposta, aceita ou obsoleta), e as consequências, que incluem tanto os impactos positivos quanto negativos daquela escolha. Esses documentos são armazenados no repositório do projeto, versionados junto ao código-fonte, o que facilita o acesso por qualquer membro da equipe, inclusive novos integrantes que entrarem no projeto. Com isso, evita-se a situação comum em que decisões antigas são seguidas ou revertidas sem que se saiba o porquê — algo que pode causar insegurança, retrabalho ou até falhas técnicas.

O artigo também destaca que uma das maiores dificuldades nos projetos é justamente lembrar, com clareza, o motivo pelo qual certas decisões foram tomadas. Quando isso não está documentado, os novos membros de um time acabam se vendo diante de dois caminhos ruins: aceitar cegamente o que foi feito antes (mesmo que o contexto tenha mudado), ou mudar algo sem entender as consequências (correndo o risco de comprometer funcionalidades importantes, como requisitos não-funcionais). Com os ADRs, essas armadilhas são evitadas, pois há um histórico claro e acessível sobre as escolhas arquiteturais e seus motivos.

Na prática do mercado de tecnologia, essa abordagem se mostra extremamente aplicável. Imagine uma equipe de desenvolvimento de uma startup que começa com

MongoDB por causa da flexibilidade do schema, mas que, meses depois, cogita migrar para PostgreSQL por questões de desempenho. Sem um ADR que documente a decisão inicial, qualquer alteração pode ser feita sem levar em conta as motivações do passado. Mas com um ADR, a equipe entenderá que a escolha original foi baseada em agilidade, mudanças frequentes e flexibilidade — e poderá avaliar melhor se ainda faz sentido manter ou mudar a tecnologia.

Além disso, em ambientes corporativos com times ágeis, rotatividade de desenvolvedores e múltiplas squads, é comum que decisões técnicas fiquem “na cabeça” de quem as tomou — o que não escala bem. Os ADRs ajudam a preservar esse conhecimento, garantindo continuidade técnica mesmo com mudanças de equipe. Essa prática é especialmente útil em projetos que envolvem arquitetura distribuída, como microserviços, onde cada decisão de integração, linguagem, banco de dados ou padrão de comunicação pode impactar todo o sistema.

Michael Nygard encerra o artigo reforçando que os ADRs já estavam sendo aplicados em projetos reais, com retorno positivo por parte de clientes e desenvolvedores. A leveza do formato, a clareza do conteúdo e a acessibilidade por estar versionado junto ao código fazem dos ADRs uma alternativa viável e eficaz frente à documentação tradicional. Mais do que isso, os ADRs representam uma mentalidade de trabalho: não basta tomar boas decisões técnicas, é preciso garantir que elas possam ser compreendidas, avaliadas e, se necessário, revisitadas com segurança no futuro.

Em um cenário onde a agilidade não deve ser inimiga da documentação, a proposta de ADRs se encaixa perfeitamente. Trata-se de uma prática que une simplicidade e profundidade, dando ao time de desenvolvimento ferramentas reais para tomar decisões conscientes e sustentáveis ao longo de todo o ciclo de vida do software. Ao adotar ADRs, empresas ganham não apenas em organização, mas em maturidade técnica, comunicação entre equipes e capacidade de adaptação a novos contextos — elementos fundamentais no mercado de tecnologia atual.