Nome: Leandro Alencar Pereira Clemente

Em projetos ágeis, a arquitetura precisa ser descrita e definida de forma diferente dos métodos tradicionais. Nem todas as decisões são tomadas no início, e muitas surgem ao longo do desenvolvimento. As metodologias ágeis não são contrárias à documentação, mas sim àquela que não agrega valor. Documentos extensos, desatualizados e difíceis de ler acabam sendo ignorados pelas equipes. Por isso, a proposta é produzir registros curtos, claros e úteis, que possam ser facilmente atualizados e consultados por todos os envolvidos no projeto. A documentação arquitetural deve ter como foco a comunicação e a manutenção do conhecimento, não a burocracia.

Um dos principais desafios durante o ciclo de vida de um projeto é compreender o motivo por trás de certas decisões arquiteturais. Quando novos integrantes entram na equipe, podem se deparar com escolhas que lhes parecem confusas ou questionáveis. Sem acesso ao raciocínio original, esses profissionais podem cair em dois erros comuns: aceitar cegamente decisões antigas, mesmo que o contexto tenha mudado, ou modificá-las sem entender suas consequências, correndo o risco de prejudicar o sistema. Ambas as atitudes comprometem a evolução saudável do projeto. A solução proposta para evitar esses problemas é manter um registro das decisões arquiteturais significativas, garantindo a rastreabilidade das motivações, dos contextos e dos impactos de cada escolha.

Esses registros são chamados de Architecture Decision Records, ou simplesmente ADRs. Cada ADR documenta uma decisão arquitetural importante, especialmente aquelas que influenciam a estrutura, os requisitos não funcionais, as dependências, as interfaces ou as técnicas de construção do sistema. Os ADRs são pequenos arquivos de texto, geralmente escritos em formato Markdown, e armazenados no repositório do projeto, em um diretório específico, como “doc/arch/”. Eles são numerados sequencialmente e mantidos mesmo quando uma decisão é substituída, de forma que o histórico completo do raciocínio técnico é preservado.

A estrutura de um ADR é simples e direta. Cada documento deve conter um título curto e descritivo, indicando claramente o tema da decisão, como “ADR 1: Implantação em Ruby on Rails 3.0.10”. Em seguida, vem a seção de contexto, que descreve as forças e fatores que influenciaram a decisão, sejam eles tecnológicos, políticos, sociais ou internos ao projeto. Essa parte deve ser escrita de maneira neutra e objetiva. A próxima seção, denominada decisão, descreve qual foi a escolha feita pela equipe, escrita em voz ativa e em frases completas. Depois, há o campo de status, que indica se a decisão está proposta, aceita, obsoleta ou foi substituída. Por fim, a seção de consequências apresenta os efeitos decorrentes da decisão, incluindo tanto os aspectos positivos quanto os negativos, já que ambos impactam o futuro do projeto.

Um ADR deve ter no máximo duas páginas e ser escrito como se fosse uma conversa com um futuro desenvolvedor que precisará entender o raciocínio da equipe. O estilo deve ser claro e coeso, com frases completas e bem estruturadas. Listas e tópicos são aceitáveis apenas como recurso visual, não como substitutos de um texto articulado. O objetivo é que qualquer pessoa que leia o documento compreenda rapidamente o que foi decidido, por que a decisão foi tomada e quais são seus impactos.

O uso de ADRs traz diversos benefícios. Cada registro documenta uma decisão arquitetural importante e suas implicações, tornando o histórico do projeto mais transparente e compreensível. Além disso, os ADRs criam uma relação entre decisões passadas e futuras, em que as consequências de um registro frequentemente se tornam o contexto para o próximo. Esse encadeamento é comparável à “linguagem de padrões” proposta por Christopher Alexander, em que decisões de grande escala definem o espaço para decisões de menor escala. Com isso, o conhecimento arquitetural passa a ser acumulativo e evolutivo, acompanhando as mudanças do projeto sem se perder com o tempo.

Outro benefício é a acessibilidade da informação. Com os ADRs armazenados no repositório de código, todos os desenvolvedores e stakeholders podem acessar facilmente as decisões, mesmo que a equipe sofra alterações ao longo do tempo. Assim, evita-se a confusão comum em projetos em que ninguém sabe ao certo o motivo de certas escolhas técnicas. As motivações permanecem claras e documentadas, e o momento de revisar uma decisão se torna evidente à medida que o contexto do projeto evolui. Dessa forma, os ADRs ajudam a equilibrar estabilidade e flexibilidade arquitetural.

A experiência prática relatada pelos autores mostra que o uso dos ADRs tem se mostrado muito positivo. Desde sua adoção, vários projetos experimentaram a técnica, e os feedbacks de desenvolvedores e clientes foram favoráveis. Os profissionais afirmam que o formato oferece clareza sobre o raciocínio técnico e facilita a compreensão das intenções de longo prazo da arquitetura. Os ADRs também se mostraram eficazes para registrar planos de reestruturação futura, permitindo que a equipe atualize o sistema de forma planejada e consciente, sem comprometer o que já foi construído.

Uma possível crítica é que manter os ADRs junto ao código pode dificultar o acesso de pessoas que não utilizam sistemas de controle de versão, como gerentes de projeto ou clientes. No entanto, isso é facilmente resolvido com o uso de plataformas como o GitHub, que permitem visualizar arquivos Markdown de maneira amigável, semelhante a páginas de wiki, e compartilhar links diretos para as versões mais recentes dos registros.