Resenha “Hexagonal Architecture” – Pedro Henrique Maia Alves

O que é a Arquitetura Hexagonal? É um estilo arquitetural proposto por Alistair Cockburn que busca tornar as aplicações mais independentes e fáceis de evoluir, isolando o núcleo da lógica de negócio das dependências externas como frameworks, bancos de dados e interfaces de usuário. Também conhecida como *Ports and Adapters*, essa arquitetura propõe que o sistema seja dividido em duas partes principais: o núcleo da aplicação, que contém suas regras e comportamentos, e os adaptadores externos, que lidam com a comunicação com o mundo exterior. O formato hexagonal é apenas uma metáfora que representa a ideia de múltiplas portas de entrada e saída, onde cada lado do hexágono simboliza um tipo de interação diferente com a aplicação, como APIs, bancos de dados ou interfaces gráficas.

Algumas características destacadas no artigo são fundamentais para compreender este conceito, e embora não exista uma definição formal e única da arquitetura hexagonal, é esperado que sistemas que a adotem apresentem, no mínimo, alguns dos princípios a seguir:

Separação entre Domínio e Infraestrutura – O domínio (ou núcleo) contém apenas as regras de negócio e não depende de frameworks, bibliotecas ou tecnologias externas. Isso garante que a aplicação possa ser testada e mantida sem a necessidade de toda a infraestrutura.

Ports (Portas) – Representam as interfaces de comunicação entre o núcleo e o mundo externo. Elas definem *o que* deve ser feito, mas não *como*.

Adapters (Adaptadores) – São implementações concretas das portas. É por meio deles que a aplicação se conecta a tecnologias específicas, como bancos de dados, APIs, filas de mensagens ou interfaces gráficas.

Testabilidade e Independência Tecnológica – Como o domínio não depende de ferramentas externas, os testes podem ser realizados com facilidade, e tecnologias podem ser substituídas sem afetar o núcleo do sistema.

Flexibilidade e Evolução – Ao isolar as regras de negócio, a arquitetura hexagonal permite que o sistema evolua ao longo do tempo, incorporando novas funcionalidades ou tecnologias sem grandes reestruturações.

Quais são as vantagens e desvantagens da adoção da arquitetura hexagonal? Entre as principais vantagens está a clara separação de responsabilidades, que torna o código mais limpo, modular e compreensível. Isso também facilita a manutenção, pois alterações em um adaptador (por exemplo, trocar o banco de dados ou o tipo

de interface) não afetam o núcleo da aplicação. Outra vantagem importante é a testabilidade, já que o domínio pode ser testado de forma isolada, garantindo maior qualidade no desenvolvimento. No entanto, essa abordagem também traz desafios. Estruturar um sistema seguindo os princípios hexagonais exige maturidade técnica e um bom entendimento do domínio, o que pode aumentar a complexidade inicial do projeto. Além disso, em sistemas pequenos, o excesso de camadas e abstrações pode gerar uma sobrecarga desnecessária.

Devemos ou não adotar a arquitetura hexagonal? Assim como no caso dos microserviços, a resposta depende do contexto. A arquitetura hexagonal se mostra extremamente útil para sistemas que precisam ser duradouros, flexíveis e independentes de tecnologias específicas, mas pode ser um exagero em projetos simples ou de curta duração. Cockburn sugere que o mais importante é compreender os princípios por trás do modelo — como a separação de responsabilidades e o isolamento do domínio — e aplicá-los conforme a necessidade, sem transformar a arquitetura em um fim em si mesma.

No meu dia a dia como desenvolvedor, ainda não utilizei especificamente a arquitetura hexagonal em um projeto prático, mas compreender seus conceitos me ajudou a enxergar de forma mais clara os trade-offs existentes entre diferentes estilos arquitetônicos. A proposta de isolamento do domínio e independência tecnológica, por exemplo, demonstra como o design do sistema pode influenciar diretamente sua testabilidade, manutenção e capacidade de evolução. Ao compará-la com arquiteturas mais tradicionais, como a em camadas ou mesmo o modelo monolítico modular, percebo que a hexagonal oferece maior flexibilidade e liberdade tecnológica, mas também exige mais disciplina, abstrações bem definidas e um esforço inicial maior na organização do código. Entender esses princípios me faz repensar a forma como estruturou aplicações, reforçando a importância de projetar sistemas que suportam mudanças e possam evoluir com menor acoplamento — algo essencial no desenvolvimento moderno de software.