Hexagonal Architecture

O artigo “Hexagonal Architecture”, escrito por Alistair Cockburn em 2005, apresenta uma proposta de organização de sistemas chamada Arquitetura Hexagonal, também conhecida como Ports and Adapters. A principal ideia é permitir que a aplicação funcione de forma independente de interfaces externas, como banco de dados, interface gráfica ou frameworks, mantendo o foco na lógica de negócio.

Cockburn observa que, em muitos projetos, a lógica de negócio acaba ficando misturada com as partes externas do sistema, o que causa dificuldades de manutenção e de testes. Por exemplo, quando o banco de dados muda ou quando a interface de usuário é alterada, o código central do sistema também precisa ser modificado. Essa forte dependência entre as camadas torna o sistema rígido, frágil e difícil de evoluir com o tempo.

Para resolver esse problema, o autor propõe uma separação clara entre o “dentro” e o “fora” da aplicação. O “dentro” é o núcleo do sistema, onde estão a lógica de negócio, as regras principais e o comportamento essencial da aplicação. O “fora” é formado por tudo que interage com esse núcleo, como interfaces de usuário, bancos de dados, APIs externas, serviços de terceiros e qualquer outra tecnologia de integração.
A comunicação entre esses dois lados é feita por meio de portas (ports), que são interfaces bem definidas. Cada porta pode ter adaptadores (adapters), que servem como tradutores entre o mundo externo e o núcleo da aplicação. Esses adaptadores fazem com que o núcleo não precise saber como o mundo externo funciona, mantendo a independência e a testabilidade do sistema.

O formato hexagonal é usado como metáfora visual para representar que a aplicação pode se comunicar com o ambiente externo por diferentes lados, e não apenas pela interface de usuário ou pelo banco de dados. Cada lado do hexágono simboliza uma porta, e cada adaptador é como um “plugue” que permite a conexão entre o núcleo e o ambiente externo. Essa estrutura ajuda a enxergar o sistema de forma mais flexível e simétrica, sem a hierarquia rígida das arquiteturas em camadas tradicionais.

Uma das maiores vantagens dessa abordagem é que ela facilita os testes automatizados. Como o núcleo da aplicação não depende diretamente de infraestrutura externa, é possível testar toda a lógica de negócio utilizando adaptadores simulados (mocks), sem precisar de um banco de dados real ou de

uma interface gráfica. Isso reduz o tempo de teste e melhora a qualidade do código. Além disso, a arquitetura hexagonal torna o sistema mais flexível, permitindo substituir tecnologias, como trocar o banco de dados ou mudar a interface web, sem precisar alterar o código central.

O autor também diferencia dois tipos de adaptadores: os primários (ou driving adapters), que representam os elementos que acionam a aplicação, como usuários, testes automatizados ou sistemas externos; e os secundários (ou driven adapters), que são aqueles usados pela aplicação para executar ações externas, como salvar dados, enviar mensagens ou acessar serviços. Essa distinção ajuda a entender melhor o fluxo de comunicação dentro do sistema e deixa mais claro quem está controlando quem.

Em resumo, a Arquitetura Hexagonal busca criar sistemas mais flexíveis, testáveis e fáceis de manter, separando a lógica de negócio das dependências externas. Essa separação torna o código mais limpo e reduz o acoplamento entre as partes do sistema. O conceito proposto por Alistair Cockburn influenciou várias arquiteturas modernas, como a Clean Architecture e a Onion Architecture, que seguem os mesmos princípios de independência, isolamento e foco no domínio da aplicação.