Hexagonal Architecture

Aluno: Vitor Veiga Silva

A arquitetura hexagonal, também conhecida como Ports & Adapters, foi proposta por Alistair Cockburn em 2005 como uma forma de estruturar sistemas de software de maneira que a lógica de negócio fique isolada das dependências externas, como interface de usuário, banco de dados ou outros sistemas. A ideia central é criar uma separação clara entre o núcleo da aplicação e o mundo exterior por meio de portas (ports) e adaptadores (adapters). As portas representam os pontos de comunicação do sistema, enquanto os adaptadores traduzem as interações com tecnologias específicas. Dessa forma, o núcleo da aplicação permanece independente de detalhes técnicos, podendo ser testado e evoluído sem impacto direto nas camadas externas.

Essa proposta surgiu como uma resposta a dois problemas recorrentes no desenvolvimento de software: o forte acoplamento da lógica de negócio à interface do usuário, que dificulta testes automatizados e mudanças de interface, e a dependência direta de sistemas de persistência, que impede o funcionamento do software sem um banco de dados ativo. Ao isolar o núcleo dessas dependências, é possível executar e testar a aplicação de forma independente, trocar tecnologias externas e adicionar novas interfaces sem modificar a lógica principal.

Na representação conceitual da arquitetura, o sistema é desenhado como um hexágono, embora o número de lados não seja fixo. No centro está o núcleo da aplicação, e em cada face há uma porta que define um tipo de comunicação possível com o ambiente externo. Para cada porta podem existir múltiplos adaptadores: por exemplo, uma porta de entrada pode ser conectada tanto a uma interface gráfica quanto a uma API web ou um conjunto de testes automatizados. Já uma porta de saída pode se conectar a diferentes mecanismos de persistência, serviços externos ou simuladores de teste. Essa estrutura garante que o núcleo não precise conhecer os detalhes das tecnologias utilizadas, mantendo-se independente e modular.

Na prática, o desenvolvimento em arquitetura hexagonal costuma começar com a implementação do núcleo e de testes automatizados que interagem diretamente

com as portas, sem depender de interface de usuário ou banco de dados. Em seguida, são criados os adaptadores externos, como a interface real e os mecanismos de persistência. Essa abordagem incremental permite validar a lógica central desde o início e evita que decisões técnicas prejudiquem a clareza e a testabilidade do sistema.

Cockburn distingue ainda dois tipos de portas e adaptadores: primários (driving) e secundários (driven). As portas primárias são aquelas que comandam a aplicação, como uma interface de usuário ou um sistema externo que envia comandos. Já as portas secundárias são aquelas que a aplicação controla, como o acesso a um repositório de dados ou a comunicação com um serviço. Essa distinção ajuda a entender o fluxo de dependências e facilita a implementação de testes, já que adaptadores de teste podem dirigir o sistema pelas portas primárias e utilizar versões simuladas dos adaptadores secundários.

Os casos de uso, segundo Cockburn, devem residir dentro do núcleo da aplicação e definir o comportamento lógico do sistema sem referência a tecnologias externas. Essa separação evita a mistura entre lógica de negócio e infraestrutura, tornando o código mais coeso e menos sujeito a mudanças quando tecnologias são substituídas. Assim, é possível manter a estabilidade da aplicação mesmo quando ocorrem evoluções em suas interfaces ou meios de armazenamento.

A quantidade de portas necessárias varia conforme o tamanho e a complexidade do sistema. Cockburn observa que não existe uma regra fixa, podendo haver poucas portas amplas ou várias portas específicas, dependendo do grau de granularidade desejado. O importante é manter o equilíbrio entre modularidade e simplicidade. Em sistemas médios, geralmente duas a quatro portas são suficientes para garantir flexibilidade e clareza estrutural.

Exemplos práticos demonstram a eficácia dessa abordagem. Em um sistema de alertas meteorológicos, por exemplo, foi possível implementar portas distintas para receber dados externos, gerenciar o sistema por meio de uma interface administrativa, enviar notificações e interagir com o banco de dados. Com isso, novas formas de interação, como envio por e-mail ou integração com APIs, puderam ser adicionadas facilmente apenas criando novos adaptadores, sem modificar o núcleo do sistema.

A arquitetura hexagonal se relaciona com outros padrões conhecidos, como o Adapter, o MVC (Model-View-Controller) e os princípios de injeção e inversão de dependência. Todos compartilham a ideia de que o código de domínio não

deve depender de detalhes técnicos, e sim de abstrações. Essa filosofia reforça o conceito de independência entre camadas e permite que o software seja mais fácil de testar, manter e evoluir.

Em síntese, a arquitetura hexagonal propõe uma separação clara entre o núcleo da aplicação e suas dependências externas, promovendo modularidade, testabilidade e independência tecnológica. Essa estrutura permite construir sistemas duradouros, capazes de acompanhar mudanças de interface, persistência e comunicação sem comprometer sua essência. Ao aplicar seus princípios, desenvolvedores conseguem criar softwares mais coesos, robustos e adaptáveis às transformações inevitáveis do ambiente tecnológico.