Hexagonal architecture

Kaio Souza Oliveira Mayer

Introdução

O artigo Hexagonal Architecture, de Alistair Cockburn, apresenta uma abordagem de arquitetura de software também conhecida como “Ports and Adapters” (portas e adaptadores). O principal objetivo do autor é mostrar como estruturar aplicações de modo que a lógica de negócio fique isolada de tecnologias externas (interface de usuário, banco de dados, serviços externos), tornando-a mais testável, mais modular e mais resistente a mudanças tecnológicas.

Problema abordado

Cockburn aponta que muitas arquiteturas tradicionais, especialmente as de camadas empilhadas (UI - lógica - persistência), sofrem de dois problemas principais:

- A lógica de aplicação frequentemente “vaza” entre camadas, provocando acoplamentos indesejados e complexidade crescente.

- O modelo de camadas empurra uma visão unidimensional de arquitetura (“usuário acima”, “banco de dados abaixo”), o que falha quando há múltiplas formas de interação (por exemplo, interface humana, API externa, testes automáticos) ou múltiplas formas de persistência.
  Esses problemas dificultam evolução, manutenção, testes isolados e adaptação da aplicação a novos cenários tecnológicos.

A solução proposta: Hexagonal Architecture

O autor propõe que a aplicação seja vista como um “núcleo” (ou “interior”) que se comunica com o exterior por meio de portas (ports). Essas portas definem contratos de comunicação, sem se preocupar com detalhes de implementação. Fora das portas ficam os adaptadores (adapters), que implementam a tecnologia específica (uma interface de usuário, um cliente HTTP, uma base de dados, mocks para testes, etc.).
Visualmente, a figura do “hexágono” serve para ilustrar essa estrutura: o núcleo no centro, com múltiplas faces (portas) para conectar adaptadores externos. O nome “hexagonal” é mais uma convenção gráfica para permitir múltiplas conexões, e não significa necessariamente seis portas.
Os benefícios principais dessa organização são:

- A lógica de negócio (core) fica isolada e independente de frameworks, bases de dados ou interfaces específicas.

- Os testes podem focar no núcleo sem se preocupar com dependências externas por exemplo, usar um adaptador em memória ou mock.

- A adaptabilidade da aplicação melhora: se for necessário trocar o banco de dados, modificar a interface ou mudar um serviço externo, só o adaptador precisa ser alterado; a lógica central permanece intacta.

Esse artigo ajuda a perceber que a construção de software não é apenas sobre “funcionar agora”, mas sobre “manter-e-evoluir”. A Hexagonal Architecture nos ensina que pensar na separação de responsabilidades desde cedo colocando a lógica de negócio no centro e tecnologias nas bordas é uma prática que facilita manutenção, teste e extensibilidade.
Por exemplo, em projetos de disciplina ou em estágios, muitas vezes somos “enganados” a acoplar diretamente UI com banco ou lógica de negócio com framework específico. Após ler esse artigo, fica claro que esse tipo de acoplamento pode aumentar o custo de mudanças futuras.
Também me faz refletir sobre testes: se a lógica está bem isolada, posso escrever testes automatizados focados nela, sem depender de banco ou rede, o que aumenta a velocidade e confiabilidade dos testes iniciais.

Conclusão

O artigo Hexagonal Architecture é relevante e atual, mesmo tendo sido publicado em 2005. Ele oferece um direcionamento arquitetural simples e poderoso: isolar o “coração” da aplicação das particularidades tecnológicas externas, por meio de portas e adaptadores.
Para um futuro engenheiro de software como eu, essa leitura reforça que adotar padrões arquiteturais conscientes desde etapas iniciais de desenvolvimento ajuda a reduzir acoplamento, facilita evolução e torna o sistema mais robusto frente a mudanças. A aplicação desse padrão, inclusive, pode contribuir para tornar o código mais limpo, mais organizado e mais fácil de testar habilidades essenciais para quem ainda está na graduação, mas que já almeja trabalhar com software de qualidade nas empresas.