WIKIPEDIA

Arquitetura hexagonal (software)

A **arquitetura hexagonal** , ou arquitetura de **portas e adaptadores** , é um padrão de arquitetura usado no projeto de software . Ele visa a criação de componentes de aplicação fracamente acoplados que podem ser facilmente conectados ao seu ambiente de software por meio de portas e adaptadores . Isso torna os componentes intercambiáveis em qualquer nível e facilita a automação de testes. [1]

Conteúdo

Origem

A arquitetura hexagonal foi inventada por Alistair Cockburn na tentativa de evitar armadilhas estruturais conhecidas no projeto de software orientado a objetos , como dependências indesejadas entre camadas e contaminação do código da interface do usuário com lógica de negócios , e publicada em 2005. [2]

O termo "hexagonal" vem das convenções gráficas que mostram o componente do aplicativo como uma célula hexagonal . O objetivo não era sugerir que haveria seis fronteiras/portas, mas deixar espaço suficiente para representar as diferentes interfaces necessárias entre o componente e o mundo externo. [1]

Princípio

A arquitetura hexagonal divide um sistema em vários componentes intercambiáveis fracamente acoplados, como o núcleo do aplicativo, o banco de dados, a interface do usuário, scripts de teste e interfaces com outros sistemas. Essa abordagem é uma alternativa à arquitetura tradicional em camadas.

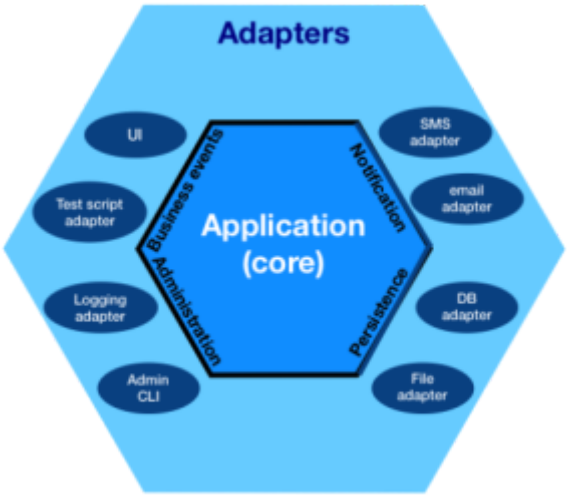
Cada componente é conectado aos outros por meio de várias "portas" expostas. A comunicação através dessas portas segue um determinado protocolo dependendo de sua finalidade. Portas e protocolos definem uma API abstrata que pode ser implementada por qualquer meio técnico adequado (por exemplo , invocação de método em uma linguagem orientada a objetos , chamadas de procedimento remoto ou serviços da Web).

A granularidade das portas e seu número não são restritos:

- uma única porta pode, em alguns casos, ser suficiente (por exemplo, no caso de um simples consumidor de serviço);
- normalmente, existem portas para fontes de eventos (interface do usuário, alimentação automática), notificações (notificações de saída), banco de dados (para fazer a interface do componente com qualquer SGBD adequado) e administração (para controlar o componente);

- em um caso extremo, pode haver uma porta diferente para cada caso de uso , se necessário.

Os adaptadores são a cola entre os componentes e o mundo exterior. Eles adaptam as trocas entre o mundo externo e as portas que representam os requisitos do componente interno do aplicativo. Pode haver vários adaptadores para uma porta, por exemplo, se os dados puderem ser fornecidos por um usuário por meio de uma GUI ou interface de linha de comando, por uma fonte de dados automatizada ou por scripts de teste.



Exemplo de arquitetura hexagonal

Críticas

O termo "hexagonal" implica que existem 6 partes no conceito, enquanto existem apenas 4 áreas-chave. O uso do termo vem das convenções gráficas que mostram o componente do aplicativo como uma célula hexagonal . O objetivo não era sugerir que haveria seis fronteiras/portas, mas deixar espaço suficiente para representar as diferentes interfaces necessárias entre o componente e o mundo externo. [3]

De acordo com Martin Fowler , a arquitetura hexagonal tem a vantagem de usar semelhanças entre a camada de apresentação e a camada de fonte de dados para criar componentes simétricos feitos de um núcleo cercado por interfaces, mas com a desvantagem de esconder a assimetria inerente entre um provedor de serviços e um consumidor de serviços que seriam melhor representados como camadas. [4]

Evolução

Segundo alguns autores, a arquitetura hexagonal está na origem da arquitetura de microsserviços . [5]

Variantes

A arquitetura onion proposta por Jeffrey Palermo em 2008 é semelhante à arquitetura hexagonal: ela também externaliza a infraestrutura com interfaces apropriadas para garantir um baixo acoplamento entre a aplicação e o banco de dados. [6] Decompõe ainda mais o núcleo da aplicação em vários anéis concêntricos usando inversão de controle . [7]

A arquitetura clean proposta por Robert C. Martin em 2012 combina os princípios da arquitetura hexagonal, a arquitetura onion e diversas outras variantes; Ele fornece níveis adicionais de detalhes do componente, que são apresentados como anéis concêntricos. Ele isola adaptadores e interfaces (interface de usuário, bancos de dados, sistemas externos, dispositivos) nos anéis externos da arquitetura e deixa os anéis internos para casos de uso e entidades [8] ’ . [9] A arquitetura limpa usa o princípio da inversão de dependências com a regra estrita de que as dependências só devem existir entre um anel externo para um anel interno e nunca o contrário.

Veja também

- Padrões de arquitetura
- Camada (design orientado a objetos)
- Diagrama de estrutura composta

- Análise e design orientados a objetos

Referências

1. Alistair, Cockburn (2005-04-01). "Arquitetura hexagonal" (https://alistair.cockburn.us/hexagonal-architecture/). *alistair.cockburn.us*. Recuperado 2020-11-18 .
2. Stenberg, janeiro (2014-10-31). "Explorando a Arquitetura Hexagonal" (https://www.infoq.com/news/2014/10/exploring-hexagonal-architecture/) . *InfoQ* . Recuperado 2019-08-12 . (https://www.infoq.com/news/2014/10/exploring-hexagonal-architecture/)
3. Alistair, Cockburn (2005-04-01). "Arquitetura hexagonal" (https://alistair.cockburn.us/hexagonal-architecture/) . *alistair.cockburn.us* . Recuperado 2020-11-18 . (https://alistair.cockburn.us/hexagonal-architecture/)
4. Fowler, Martin (2003). *Padrões de arquitetura de aplicativos corporativos* . Addison-Wesley. pág. 21. ISBN 0-321-12742-0. OCLC 50292267 (https://www.worldcat.org/oclc/50292267) .
5. Rajesh RV (2017). *Microsserviços do Spring 5.0: crie microsserviços escalonáveis com Reactive Streams, Spring Boot, Docker e Mesos* (segunda ed.). Publicação Packt. págs. 13–14. ISBN 978-1-78712-051-8. OCLC 999610958 (https://www.worldcat.org/oclc/999610958) .
6. Jeffrey, Palermo (2008-07-29). "A Arquitetura Cebola: parte 1" (https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/) . *Programação com Palermo* . Recuperado 2019-08-12 . (https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/)
7. Chatekar, Suhas (2015). *Aprendendo NHibernate 4: explore todo o potencial do NHibernate para construir um código robusto de acesso a dados* . Publicação Packt. págs. 249-250. ISBN 978-1-78439-206-2. OCLC 937787252 (https://www.worldcat.org/oclc/937787252) .
8. Martin, Robert, C. (2012-08-12). "A Arquitetura Limpa | Blog Clean Coder" (https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html) . *blog.cleancoder.com* . Recuperado 2019-08-12 . (https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html)
9. Martin, Robert C. (2017). *Arquitetura limpa: um guia do artesão para estrutura e design de software* . Prentice Hall. ISBN 978-0-13-449416-6. OCLC 1004983973 (https://www.worldcat.org/oclc/1004983973) .