THIAGO ADRIANO

PÓS TECH

SOFTWARE ARCHITECTURE

ARQUITETURA DE SOFTWARE

AULA 05

SUMÁRIO

O QUE VEM POR AÍ?

Nesta aula você vai aprender sobre uma das arquiteturas mais utilizadas na atualidade: a arquitetura hexagonal.

HANDS ON

No primeiro vídeo desta aula, os professores apresentaram o que é a Arquitetura Hexagonal e, por meio de alguns exemplos, demonstraram como esta abordagem pode ajudar no desenvolvimento e manutenção do seu sistema.

No vídeo seguinte, os docentes falaram sobre porta, adaptadores e os atores na arquitetura hexagonal, por meio de alguns exemplos de código.

No próximo vídeo, aplicamos os conceitos e criamos os atores conduzidos para utilização de cache na aplicação e persistência de dados para entidade Usuários.

E, no vídeo final desta aula, os professores apresentaram o conceito de Use Case (que é muito conhecido em outras arquiteturas, como na Clean Architeture), por meio de um exemplo prático, para demonstrar como persistir os dados de um usuário.

SAIBA MAIS

ARQUITETURA HEXAGONAL

Um dos princípios básicos da engenharia de software é o “Separation of Concerns (SoC)”, ou Separação de Conceitos, em português. Ele visa separar preocupações, ou seja, modularizar uma solução de forma que cada módulo esteja focado em resolver apenas um único problema.

No desenvolvimento de software, podemos afirmar que Separação de Conceitos se refere a um princípio de projeto que permite a separação de um sistema em diferentes áreas, seções ou módulos. Esses conceitos, quando combinados, atingem um objetivo principal que, na maioria das vezes, é um software, uma página da web ou um hardware. A Separação de Conceitos normalmente ocorre em softwares de grande porte ou em projetos que são desenvolvidos em mais de uma linguagem de programação.

Agora, trazemos este princípio para um padrão de arquitetura com o qual você provavelmente já trabalhou ou pelo menos teve contato, o Model View Controler (MVC). Ele separa a arquitetura do código em três camadas, cada uma com uma responsabilidade específica. Enquanto a camada View apresenta os dados para o usuário, a camada Model trata o processamento de dados (regras de negócio) e a camada Controller faz o intermédio entre ambas.

Esse é um exemplo de padrão que utiliza o SoC e que demonstra de forma prática como ele ajuda a elaborar uma arquitetura flexível, manutenível e sustentável.

Agora partindo para o tópico desta aula, o SoC foi um dos princípios adotados pelo Dr Alistair Cockburn, em meados dos anos 1990, para a proposta de criação da Arquitetura Hexagonal.

Ele postou um artigo na primeira wiki, chamada WikiWikiWeb, apresentando esta arquitetura com alguns princípios que têm como foco modularizar uma solução em três áreas distintas e isoladas:

- Centro do hexágono.
- Lado superior esquerdo, fora do hexágono.

- Lado inferior direito, fora do hexágono.

Os objetivos de uma arquitetura Hexagonal são parecidos com os de uma arquitetura limpa, que visa construir sistemas que favorecem a reusabilidade de código, alta coesão, baixo acoplamento, independência de tecnologia e que sejam mais fáceis de serem testados.

E aqui chegamos a um dos pontos mais importantes, a parte de testes. Esta parte é muito importante para garantir a qualidade do software e a confiabilidade dele, mas, infelizmente, em algumas abordagens nós não conseguimos realizar todos os testes básicos necessários que estão presentes na pirâmide de testes da figura 1 – “Exemplo de fluxo de cache”, por conta de acoplamentos entre camadas e frameworks.

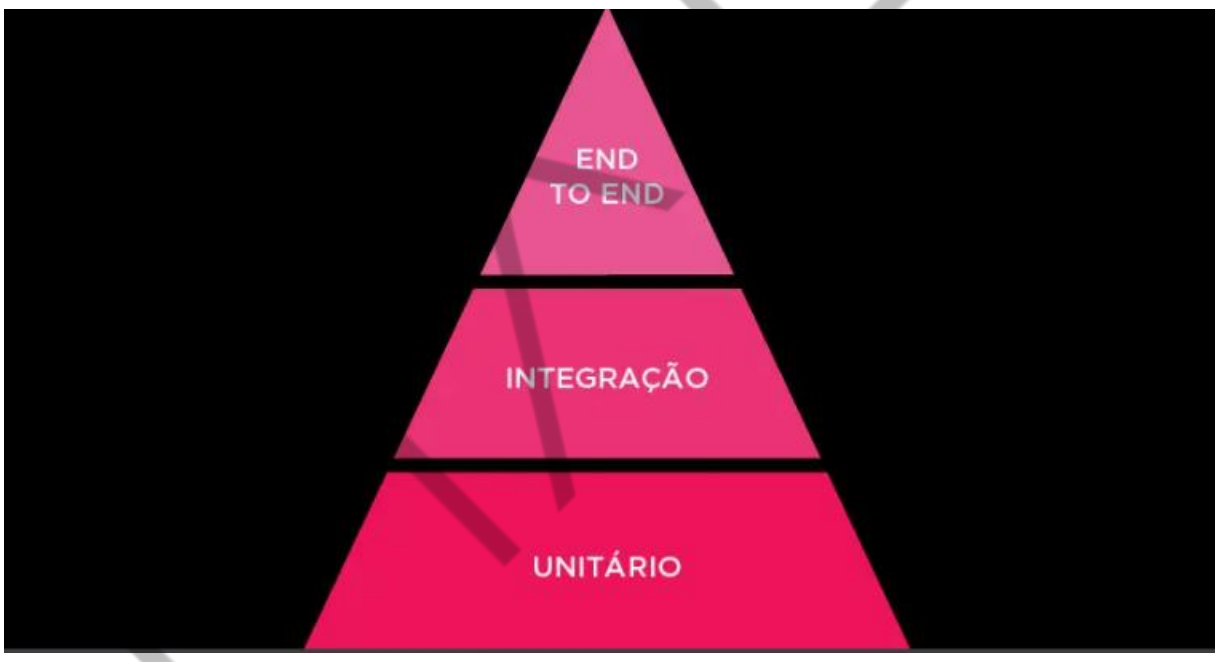

Figura 1 – Exemplo de fluxo de cache
Fonte: https://natanpf.com/2021/05/27/piramide-de-testes/ (2021), adaptada por FIAP (2023)

O foco da arquitetura hexagonal é isolar toda a lógica da aplicação no centro do hexágono, e que esta camada não tenha dependência de outras camadas ou tecnologias, tais como: qual é o banco de dados que está sendo utilizado na aplicação, ou qual é o front end que está consumindo as nossas informações.

Consequentemente, mudanças de tecnologia podem ser feitas sem impactar as classes de domínio ou casos de uso, como geralmente são adotados nesta arquitetura, permitindo que eles sejam compartilhados por mais de uma tecnologia.

Por exemplo, um sistema pode ter diversas interfaces (forma de apresentar os dados) como Web, mobile ou uma API, e trocá-las sem impactar a camada de domínio que fica no centro do hexágono.

Para ficar mais claro, a seguir você tem o desenho original desta arquitetura proposta pelo Dr Alistair:

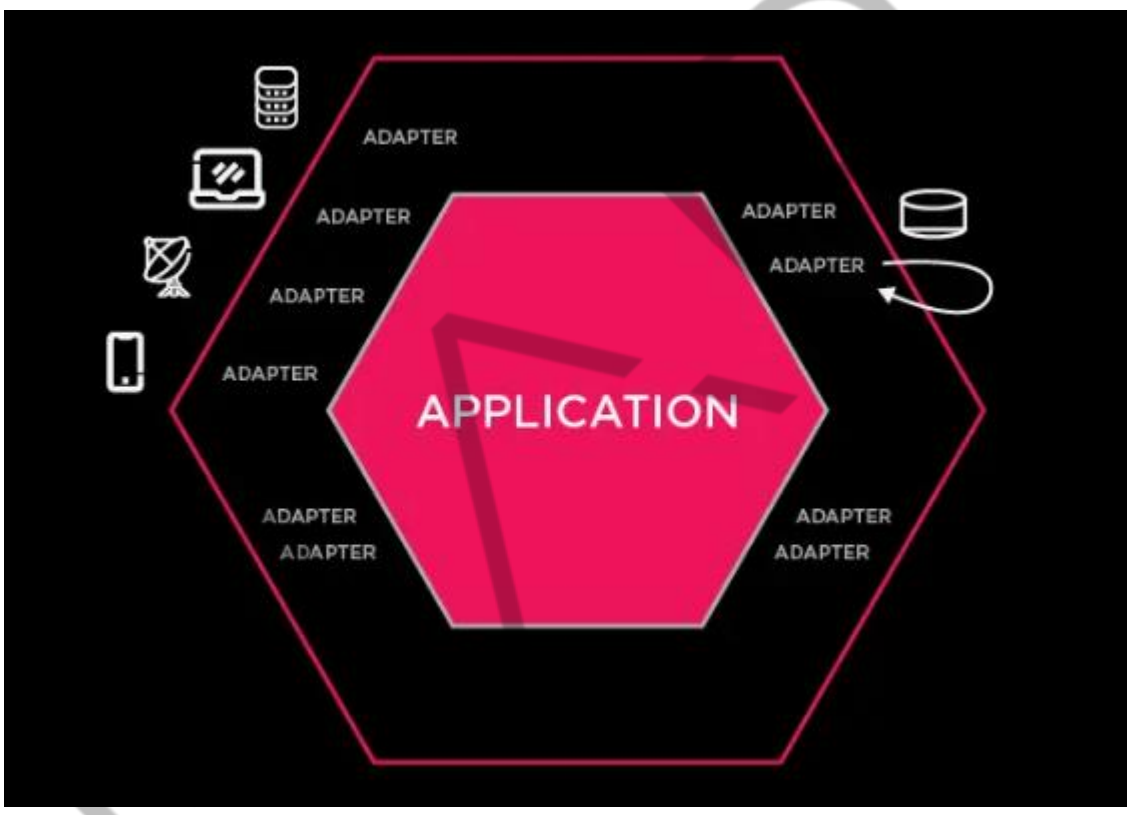

Figura 2 – Exemplo de fluxo de cache (2)
Fonte: https://alistair.cockburn.us/hexagonal-architecture/ (2023), adaptada por FIAP (2023)

Note que, no centro do hexágono, nós temos uma camada chamada de Application, ou no português, aplicação, que não tem relação com a camada de Application da arquitetura de camadas que veremos mais adiante. A diferença entre elas é o local onde fica o motivo principal do nosso software existir, o nosso core principal, ou como muitos arquitetos e desenvolvedores chamam, a nossa camada de domínio, o coração do nosso sistema.

Fora do hexágono, nós temos os adaptadores, que podem ser vistos como plugues para que as tecnologias que estão fora do hexágono cheguem à camada de aplicação por meio de algumas portas, que conheceremos ainda nesta aula.

Para ilustrar um adaptador, nós podemos utilizar a seguinte figura de um adaptador de tomada ou, como chamamos em algumas cidades do Brasil, “Benjamin” ou “T”:

Figura 3 – Exemplo de fluxo de cache
Fonte: interNEED (2023), adaptada por FIAP (2023)

Um ponto importante de se destacar antes de avançarmos para o próximo assunto desta aula, seria que o nome hexagonal não possui nenhuma relação com o número 6, e nem mesmo com o hexágono. A relação está com a figura geométrica, o polígono. Ou seja, também poderia ser representada por um pentágono ou heptágono, pois a ideia é representar algo que possua diversos lados, que são as diferentes implementações e tecnologias que uma aplicação pode utilizar para expor seus resultados, podemos dizer que os lados representam a camada de serviço e suas implementações.

Na entrevista “Hexagonal Architecture (Ports and Adapters) with Alistair Cockburn”, para um canal do youtube, o Dr. Alistair justifica o uso de um hexágono para o nome desta arquitetura da seguinte forma:

> **Cada face do hexágono representa um motivo pelo qual o sistema deve se comunicar com o mundo exterior. É por isso que são hexágonos concêntricos e não círculos concêntricos.**

PORTAS E ADAPTADORES

Em 2005, o Dr. Alistair tentou renomear o nome de sua arquitetura para Arquitetura Baseada em Portas e Adaptadores, pois ele reparou que as faces do hexágono [interno] são portas… e os objetos entre os dois hexágonos são adaptadores e, portanto, o padrão arquitetural consiste em uma Arquitetura Baseada em Portas e Adaptadores.

Na mesma entrevista ao canal do Youtube mencionada acima, O Dr Alistair comentou que, para ele o nome ficou como Portas e Adaptadores, mesmo que a tentativa de renomear não tenha dado tão certo e a arquitetura continue sendo mais conhecida como hexagonal.

Portanto, é importante destacarmos a definição do que seria uma porta e um adaptador. Vamos lá?

Uma porta nada mais é que um conceito lógico que define um ponto de entrada e saída da aplicação. Este termo se refere às interfaces usadas para comunicação com os casos de uso. Veja que, "interface" aqui significa interface de programação, por exemplo, uma interface do C#.

```
public interface ICadastroPessoa{}
```

De acordo com o professor Marco Tulio Valente, no livro Engenharia de Software Moderna, existem dois tipos de portas:

- Portas de entrada: são interfaces usadas para comunicação de fora para dentro, isto é, quando uma classe externa precisa chamar um método de

um caso de uso. Logo, essas portas declaram os serviços providos pelo sistema, isto é, serviços que o sistema oferece para o mundo exterior.

- Portas de saída: são interfaces usadas para comunicação de dentro para fora, quando uma classe de domínio precisa chamar um método de uma classe externa. Logo, essas portas declaram os serviços requeridos pelo sistema, isto é, serviços do mundo exterior que são necessários para o funcionamento do sistema como a interação com um banco de dados.

É importante destacar que as portas são independentes de tecnologia. Portanto, elas estão localizadas no centro do hexágono.

Por outro lado, os sistemas externos utilizam algumas tecnologias como: comunicação por meio do protocolo HTTP em uma aplicação REST, gRPC, GraphQL ou de bancos de dados como SQL, noSQL ou até mesmo uma interação com o usuário por meio de um aplicativo web, mobile etc.

Essas tecnologias utilizadas pelo lado externo do hexágono são os adaptadores.

A função de um adaptador é implementar a conexão entre uma porta e outros serviços externos. Na arquitetura hexagonal, nós devemos ter pelo menos dois adaptadores para uma porta, um para teste e um outro para a implementação real. Um exemplo desta implementação poderia ser uma porta para cache e dois adaptadores, um implementando o Redis (ferramenta de cache) e uma outra para o cache em memória, que pode ser utilizada para os testes unitários.

ATORES

Fora do hexágono, nós temos qualquer coisa do mundo real com qual o aplicativo interage. Essas coisas incluem seres humanos, outros aplicativos ou qualquer dispositivo de hardware ou software. Eles são chamados de atores. Existem dois tipos de atores, os condutores e os conduzidos:

- **Atores condutores:** são os que recebem chamadas de métodos vindos de fora do sistema e encaminham essas chamadas para os métodos adequados das portas de entrada. Por exemplo: uma requisição HTTP.

- **Atores conduzidos:** são os que recebem chamadas vindas de dentro do sistema, ou seja, dos casos de uso, e as direcionam para um sistema externo como: um banco de dados, um serviço de envio SMTP ou qualquer outro serviço de terceiros.

Para melhor compreensão, vamos conhecer melhor cada um desses atores.

Os atores condutores ficam ao lado esquerdo do hexágono. Eles são responsáveis por conduzir a interação com o centro do hexágono por meio de algumas ações que têm a finalidade de atingir um determinado objetivo como: suítes de testes, aplicação front-end, aplicativo mobile etc.

Verifique a figura 4 – “Exemplo de fluxo e cache”, onde temos uma ilustração que demonstra algumas das tecnologias que são mais utilizadas em nosso dia a dia para exemplificar o que seriam esses atores condutores:

Figura 4 – Exemplo de fluxo de cache
Fonte: elaborada pelo autor (2023)

O primeiro ator que nós temos na imagem está referenciando uma suíte de testes; o segundo, uma API, podendo ser RESTFul, GraphQL etc.; o terceiro, um aplicativo mobile; depois temos um ícone para exemplificar um Terminal ou CMD no caso do Windows; e, por fim, um ator para referenciar páginas WEB.

Diferente dos atores condutores que ficam ao lado esquerdo, os atores conduzidos ficam ao lado inferior direito, lado em que nós temos os serviços intercambiáveis e flexíveis que fornecem a comunicação com a infraestrutura que a solução precisa para existir.

Neste lado nós temos os detalhes infraestruturais, como: o código que interage com o banco de dados, faz chamadas para o sistema de arquivos, chamadas HTTP e outros aplicativos dos quais depende.

Diferente dos atores condutores, que acionam o hexágono por meio de interfaces, neste lado a interação é acionada pelo centro do hexágono, por meio das portas secundárias, que têm a responsabilidade de fornecer funcionalidades necessárias ao hexágono para processar a lógica de negócios, como: acesso ao banco de dados, serviços web http, stmp, sistema de arquivos etc.

Existem dois tipos de atores conduzidos:

- **Repositório (Repository):** é aquele com o objetivo de enviar e receber informações. Por exemplo: um banco de dados ou qualquer outro dispositivo de armazenamento.
- **Destinatário (Recipient):** é aquele com o objetivo de somente enviar informações e esquecer. Por exemplo: um envio de e-mail SMTP.

Figura 5 – Exemplo de fluxo de cache
Fonte: elaborada pelo autor (2023)

Agora, partindo para o centro do hexágono, nele estão localizados os modelos, domínios e regras de negócios de seu software. Nele nós temos a parte mais importante do nosso sistema, onde ficam os códigos relacionados à lógica de negócio do contexto da solução.

Nesta camada, nós temos o código que abstrai as regras de negócio do mundo real em modelo de objetos. Essa é a parte que nós devemos isolar de todas as outras partes (lado esquerdo e direito do hexágono).

Essa camada do hexágono deve ser totalmente agnóstica a qualquer tecnologia, framework e infraestrutura relacionada a interfaces gráficas, interfaces de comunicação e dispositivos externos do mundo real.

Entretanto, ele pode ter dependências de frameworks de serviços gerais como: Logg, IoC e etc.

Agora que já conhecemos as três áreas desta arquitetura, é muito importante você entender como funciona o seu fluxo de execução.

Figura 6 – Exemplo de fluxo de cache
Fonte: elaborada pelo autor (2023)

Analisando a figura 6 – “Exemplo de fluxo de cache”, nós temos: ao lado esquerdo, os atores condutores com as tecnologias que implementam os adaptadores

que acionam o centro do hexágono por meio de algumas portas; ao lado direito, os atores conduzidos, que têm as suas portas e seus adaptadores que são acionados por meio das portas do centro do hexágono.

Então, como funciona o nosso fluxo de execução?

Na arquitetura hexagonal, o fluxo de execução segue um padrão de entrada e saída, onde a lógica de negócio é isolada e independente de tecnologia, e a comunicação com as interfaces externas é feita através de portas e adaptadores.

O fluxo de execução em uma arquitetura hexagonal pode ser descrito da seguinte forma:

1. Uma solicitação é recebida por um adaptador externo, que é responsável por adaptar os dados recebidos para a estrutura esperada pela camada de negócios.
2. O adaptador envia a solicitação para uma porta na camada de negócios, que contém a lógica de negócio da aplicação.
3. A porta executa a lógica de negócio e interage com as entidades e serviços da aplicação para processar a solicitação.
4. A porta envia a resposta para outro adaptador externo, que é responsável por adaptar a resposta para a estrutura esperada pela interface externa que originou a solicitação.
5. O adaptador externo envia a resposta de volta para a interface externa, que apresenta a resposta para o usuário.

Analisando esse cenário, nós temos um problema. O centro do hexágono está dependendo do lado direito, fazendo com que os atores conduzidos não dependam de ninguém e o centro do hexágono dependa totalmente dele.

Esse foi um dos problemas que o Dr. Alistair encontrou no início da modelagem desta arquitetura, dentro do seu fluxo de execução. Para resolver, ele implementou dentro do fluxo o padrão de arquitetura Inversão de Controle (IoC).

Para quem está tendo o primeiro contato com IoC nesta aula, segue um breve resumo abaixo:

Inversão de controle é um princípio de design de programas de computadores onde a sequência (controle) de chamadas dos métodos é invertida em relação à programação tradicional, ou seja, ela não é determinada diretamente pelo programador.

Este controle é delegado a uma infraestrutura de software muitas vezes chamada de contêiner ou a qualquer outro componente que possa tomar controle sobre a execução. Esta é uma característica muito comum a alguns frameworks.

Em outras palavras, IoC é um padrão utilizado em projetos orientado a objetos, que utilizam conceitos como interface, herança e polimorfismo.

O QUE VOCÊ VIU NESTA AULA?

A arquitetura hexagonal, que também é conhecida como portas e adaptadores, foi proposta pelo Dr. Alistair em meados dos anos 1990, com a finalidade de diminuir o acoplamento entre as camadas de uma aplicação utilizando o princípio de engenharia de software (SoC):

- Nessa arquitetura nós temos três áreas: centro do hexágono, lado esquerdo e lado direito.
- A separação em camadas ajuda nos testes.
- Na camada de aplicação nós temos as regras de negócio, que podem ser definidas por meio de casos de uso.
- Ao lado de fora do hexágono, nós temos os adaptadores que se comunicam com o centro do hexágono por meio de portas que são interfaces de linguagem de programação.
- Conhecemos os tipos de portas e adaptadores.
- Conhecemos os dois tipos de atores, os condutores e os conduzidos.
- Passamos pelo fluxo de execução.
- Introdução ao IoC.

REFERÊNCIAS

BRANAS, Rodrigo. Hexagonal Architecture (Ports and Adapters) with Alistair Cockburn // Live #98. Youtube, 27 de mai de 2022. Disponível em: <**https://www.youtube.com/watch?v=AOIWUPjal60**>. Acesso em: 10 de mar de 2023.

TULIO, Marco Valente. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**. [S.l]: Editora Independente, 2020.

SHRIVASTAVA, Saurabh. SRIVASTAV, Neelanjali. **Solutions Architect's Handbook - Second Edition: Kick-start your career as a solutions architect by learning architecture design principles and strategies**. [S.l]: Packt Publishing, 2022.

PALAVRAS-CHAVE

Arquitetura. Ports and Adapters. Arquitetura Hexagonal.