LINGUAGEM DE PROGRAMAÇÃO ORIENTADA A OBJETO II

PROFESSOR: Michel Osadon
Cursos: ADS - Análise e Desenvolvimento De Sistemas
Turma: 4° Semestre/2025
Aluno: Pedro Judah G. N. Lopes

Arquitetura Monolítica X Microsserviços

Conceitos Iniciais

- Monolito (O jeito "clássico"): Imagine que sua aplicação inteira é um único projeto gigante. A tela de login, o catálogo de produtos, o carrinho e os pedidos estão todos no mesmo código, acessando um único e enorme banco de dados.
    - Problema: Se você precisa corrigir um bug no carrinho, tem que fazer o deploy da aplicação inteira. Se o módulo de pagamento der pau, a loja toda pode sair do ar.
- Microsserviços (O jeito da imagem): Em vez de um "appzão", você tem vários "appzinhos" especialistas que conversam entre si. Cada um faz uma única coisa e faz bem feito.

Analisando a Arquitetura da Imagem

1. Frontend (Client apps):
    - São as interfaces com o usuário (feitas em Angular, mobile, etc.). Elas não sabem da complexidade do backend. Elas apenas consomem uma API.
2. API Gateway (O "Porteiro"):
    - O frontend não fala com cada microsserviço. Ele bate na porta do API Gateway. Esse cara é o único ponto de entrada, que recebe a requisição (ex: GET /produtos/123) e a redireciona para o microsserviço correto lá dentro. É uma fachada que simplifica e protege o sistema.
3. Microsserviços (Os "Especialistas"):

Cada caixinha é uma API REST independente, com sua própria lógica e seu próprio banco de dados.

- Catalog microservice: Uma API focada em CRUD de produtos.
- Ordering microservice: Outra API que só sabe lidar com pedidos.
- Basket microservice: Uma API para o carrinho. Note que ela usa Redis, um banco NoSQL de chave-valor, porque é muito mais rápido para

dados temporários como um carrinho. Isso mostra a flexibilidade de usar a melhor ferramenta para cada trabalho.
    - Identity microservice: Outra API que cuida só de autenticação (login/senha), geralmente gerando tokens (JWT).
4. Event Bus (O "Mensageiro"):
    - Este é o pulo do gato para manter os serviços independentes. Quando o serviço de Ordering finaliza um pedido, ele não chama diretamente o serviço de Estoque para dar baixa. Ele simplesmente publica uma mensagem no Event Bus: "Pedido #123 finalizado com o produto X".
    - O serviço de Estoque (que nem aparece aí, mas poderia existir) está "ouvindo" essas mensagens e, quando vê uma que lhe interessa, ele age e dá baixa no produto. Isso se chama comunicação assíncrona e desacopla os serviços.

Quadro Comparativo Final

Aspecto	Monolito	Microsserviços (a imagem)
Deploy	Sobe o projeto inteiro de uma vez.	Sobe só o microsserviço que você alterou.
Escalabilidade	Precisa de mais poder no catálogo? Tem que escalar a aplicação inteira.	O catálogo está lento na Black Friday? Você escala *apenas* o Catalog microservice.
Resiliência	Se o módulo de login quebrar, a loja toda pode parar.	Se o Ordering microservice cair, os clientes ainda podem ver produtos e montar o carrinho.
Tecnologia	Geralmente preso a uma única stack (ex: tudo em Java).	O serviço de Identity pode ser em C#, o de Catalog em Node.js. Total liberdade.