Front-end, back-end e persistência de dados

A tão famosa arquitetura em camadas ou n-tier architecture não é um conceito novo, muito pelo contrário! Um exemplo dessa arquitetura, no caso uma arquitetura em 3 camadas, pode ser resumida na figura abaixo (que irei traduzir em seguida):

Layer	Responsibilities
Presentation	Provision of services, display of information (e.g., in Windows or HTML, handling of user request (mouse clicks, keyboard hits), HTTP requests, command-line invocations, batch API)
Domain	Logic that is the real point of the system
Data Source	Communication with databases, messaging systems, transaction managers, other packages

Traduzindo:

- Camada de Apresentação: Literalmente é a parte visual da aplicação/sistema, pode ser o que você vê num app do seu smartphone, pode ser um app do seu tablet, um site da internet (sim, isso inclui o site do Google entre outros), ou seja, algo da aplicação que permite a interação com o usuário, ou seja, eu e você! É através dessa camada que digitamos coisas, clicamos em botões, pressionamos o dedo em cima para algo acontecer

- Camada de Domínio: É aqui que desenvolvemos todas as regras de negócio que a aplicação/sistema terá, então uma vez que você digitou ou clicou em algo na camada de apresentação, este envia essas informações (do que foi digitado ou onde foi clicado) à camada de domínio, por sua vez a camada de domínio é

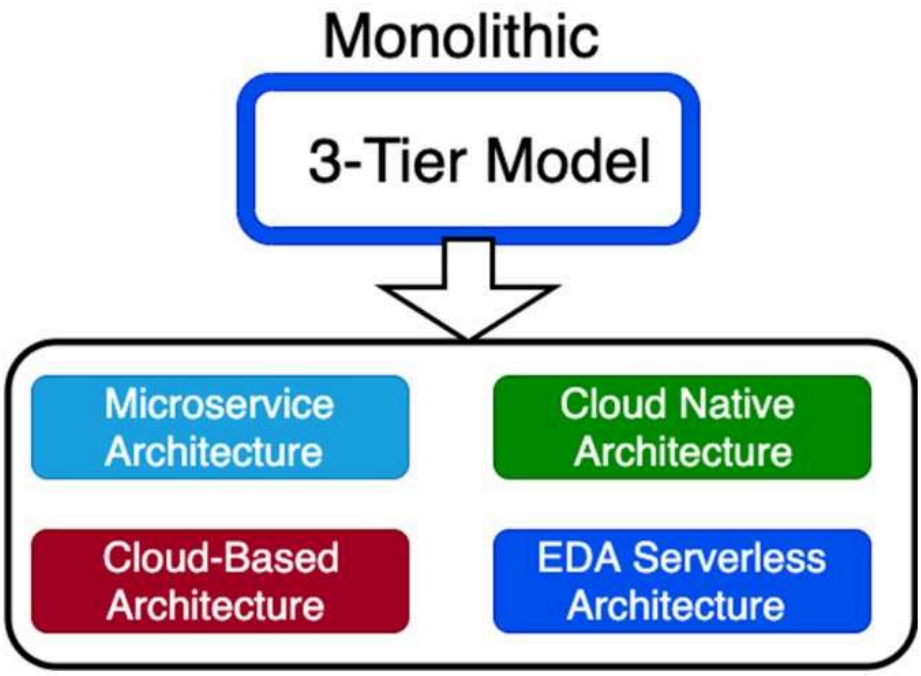
onde desenvolvemos as regras para saber o que fazer com a informação que veio, com o que foi digitado, com o que foi clicado, é aqui que está o coração da aplicação. Sem essa camada de domínio, a camada de apresentação não tem muita função, pois lá ele só coleta essas entradas que o usuário faz ao interagir com a aplicação. A camada de domínio é apartada, ou seja, totalmente separada da camada de apresentação, e ela também não faz nada sozinha se não houver alguma entrada do usuário

- Camada de Dados: Se na camada de apresentação temos a interação com o usuário para pegar as entradas como algo digitado ou clicado, e na camada de domínio temos todas as regras de negócio para saber o que fazer com essas entradas, na camada de dados vamos dizer que é onde guardamos informações importantes ao negócio. Então é na camada de dados que salvamos o cadastro de um cliente, que salvamos a compra de algum produto feita por um cliente, que consultamos como está a parte logística para a entrega de algum produto ao cliente. É aqui que estão todos os dados e somente esta camada é responsável por isso.

A maior vantagem desta arquitetura é separar a responsabilidade de uma solução, neste caso em camadas totalmente apartadas. Uma camada será responsável pela interação com o usuário (apresentação), outra para tratar todas as regras de negócio da aplicação (domínio) e a última para tratar todos as informações inerentes ao negócio (dados).

Inclusive em uma plataforma distribuída, as camadas são apartadas à nível de infraestrutura também! Então é comum termos um servidor/container exclusivo para a aplicação de front-end (camada de apresentação), outro servidor/container exclusivo para a aplicação de back-end (camada de domínio) e outro servidor/container exclusivo para o banco de dados (camada de dados).

A vantagem nesta segregação de responsabilidades é inclusive segregar times por tipo de conhecimento (front-end, back-end e dados, neste exemplo), ao invés de imbuir um único desenvolvedor com todas essas especializações. Além disso segregamos as camadas para garantirmos maior segurança das aplicações, o front-end nunca faria uma comunicação direta com o banco de dados, evitando risco de exposição das informações que valem ouro para as empresas. À partir daqui conseguimos falar de uma evolução na forma de arquitetar soluções utilizando microserviços.

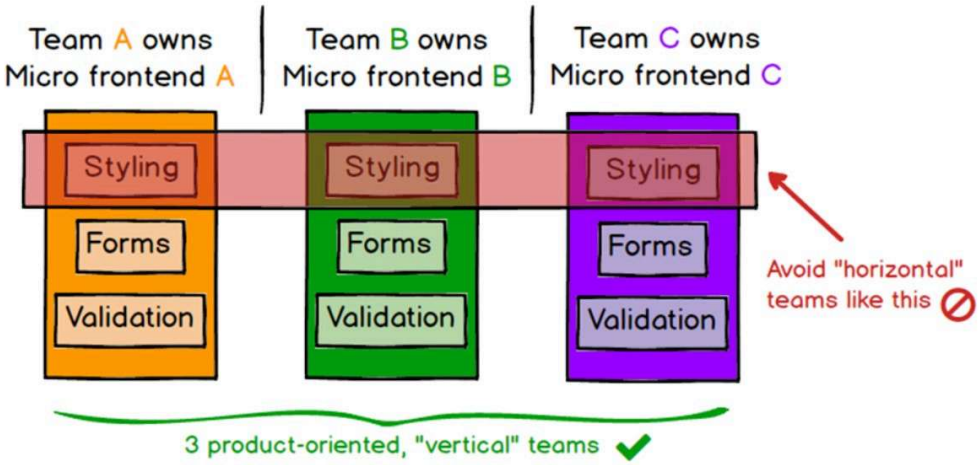


Sobre a definição do que é um microserviço, podemos afirmar que:

> Microservices are independently releasable services that are modeled around a business domain. A service encapsulates functionality and makes it accessible to other services via networks—you construct a more complex system from these building blocks. One microservice might represent inventory, another order management, and yet another shipping, but together they might constitute an entire ecommerce system. Microservices are an architecture choice that is focused on giving you many options for solving the problems you might face. (FOWLER, 2021, n.p.)

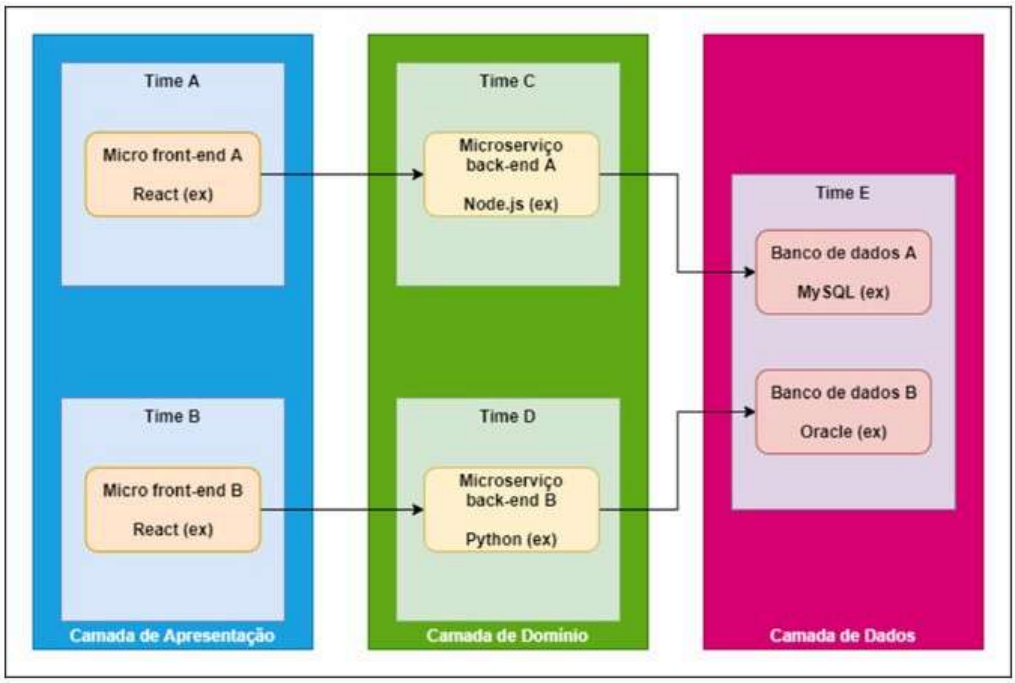
Traduzindo e resumindo: são serviços independentes e reutilizáveis acessíveis via rede (seja uma rede privada de uma empresa ou pela internet). A composição de vários microserviços pode representar um ecossistema de um sistema robusto (como um e-commerce completo).

Essa evolução para o uso de microserviços resolve um problema comum da arquitetura em 3 camadas: apesar de segregadas, as aplicações (front-end e back-end) tornavam-se algo muito difícil de dar manutenção, pela característica mais monolítica. Então além de segregar o front-end do back-end e do banco de dados, também segregamos a nível de aplicação através de vários micro front-ends (aqui expande-se o conceito de microserviços para o front-end) e os vários back-ends (em microserviços) para que o conjunto da obra torne-se o sistema completo para uso.



Na figura acima vemos um exemplo do uso deste conceito de micro front-end onde cada micro front-end é uma aplicação que tem seu objetivo e seu time responsável, supondo que o micro front-end A seja um cadastro de clientes, o micro front-end B seja um cadastro de produtos e o micro front-end C seja a composição para vendas de produtos à clientes, olhando para o back-end também haveria segregação semelhante: o back-end A poderia ser uma API para cadastrar cliente, com um banco de dados com informações de clientes, o back-end B

poderia ser uma outra API (até com uma outra tecnologia) para cadastrar produtos, com um outro banco de dados totalmente separado para guarda das informações do produtos e um terceiro back-end para tratar a venda de produtos aos clientes, com outro banco de dados totalmente segregado também!



Como parte do material de apoio, há exemplos de código-fonte de front-end, back-end e scripts para criar um banco de dados de exemplo. Para acessar o repositório do Gitlab acesse o seguinte link que lá mesmo há as instruções: https://gitlab.com/evertonjuniti/descomplica

Atividade Extra

Para se aprofundar no assunto sobre arquitetura em três camadas desta aula, leia o artigo: “Arquitetura de três camadas (tiers)”, de 28 de outubro de 2020, pela IBM Cloud Education

Link do artigo: https://www.ibm.com/br-pt/cloud/learn/three-tier-architecture

Referência Bibliográfica

- FOWLER, Martin. Patterns of Enterprise Application Architecture. 1.ed. Addison-Wesley Professional: 2002.

- SARASWATHI, Ravi. Four Architecture Choices for Application Development in the Digital Age, 6 de Janeiro. Disponível em https://www.ibm.com/cloud/blog/four-architecture-choices-for-application-development. Acesso em 22 de maio de 2022.

- NEWMAN, Sam. Building Microservices: Designing Fine-Grained Systems. 2.ed. O’Reilly Media: 2021.

- JACKSON, Cam. Micro Frontends, 19 de junho de 2019. Disponível em https://martinfowler.com/articles/micro-frontends.html. Acesso em 22 de maio de 2022.

Ir para exercício