

Webservices e MBaaS

Aula 5
Mark Joselli
PUCPR

Agenda de Hoje

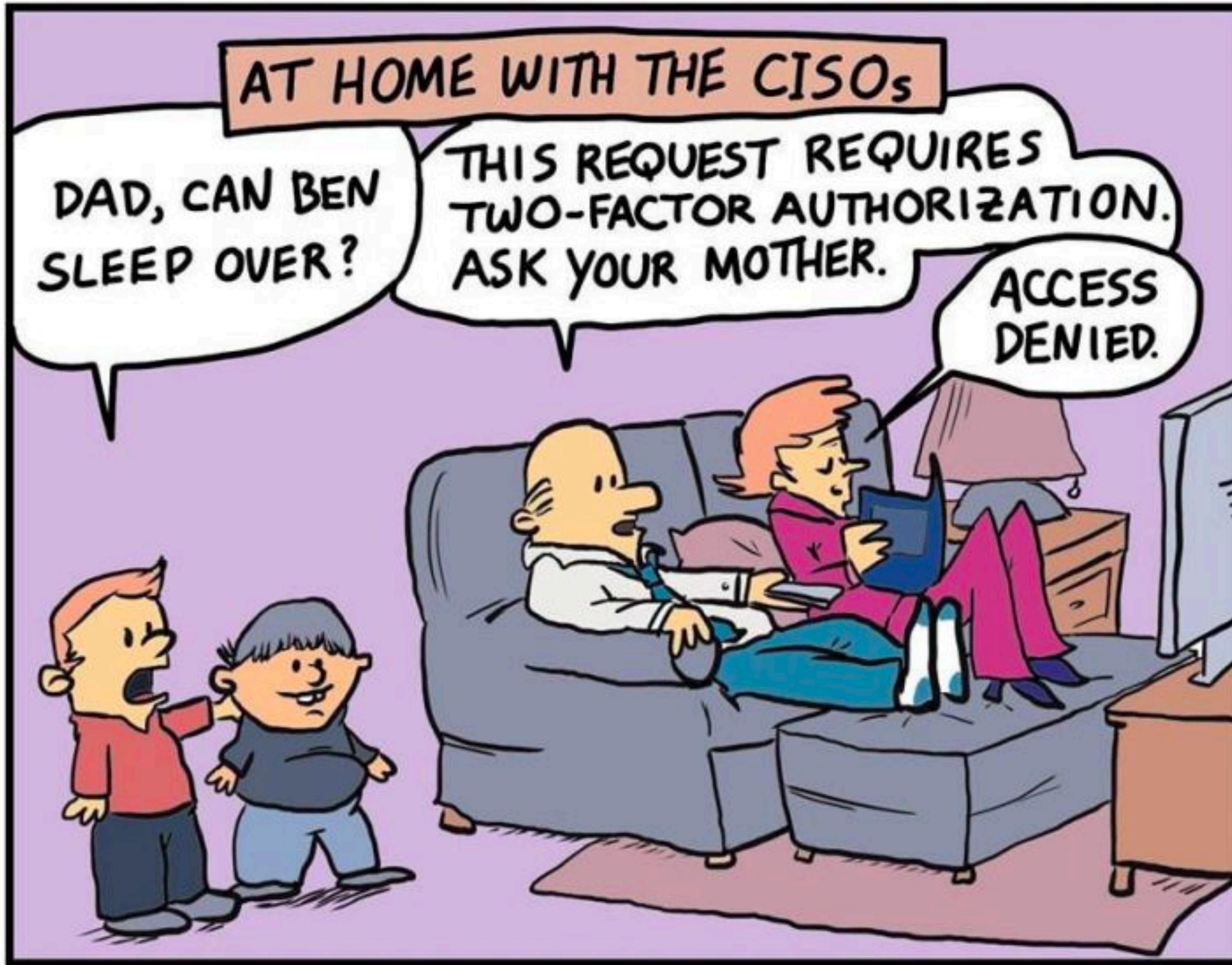
- Autenticação
- Security
- Stateless x Stateful
- Monolith x Microservices x ServerLess
- Firebase

Objetivos

- Apresentar os termos e conceitos principais desta tecnologia

Vamos Começar?

**AAA - Authentication,
Authorization and Accounting**



Authentication, Authorization and Accounting

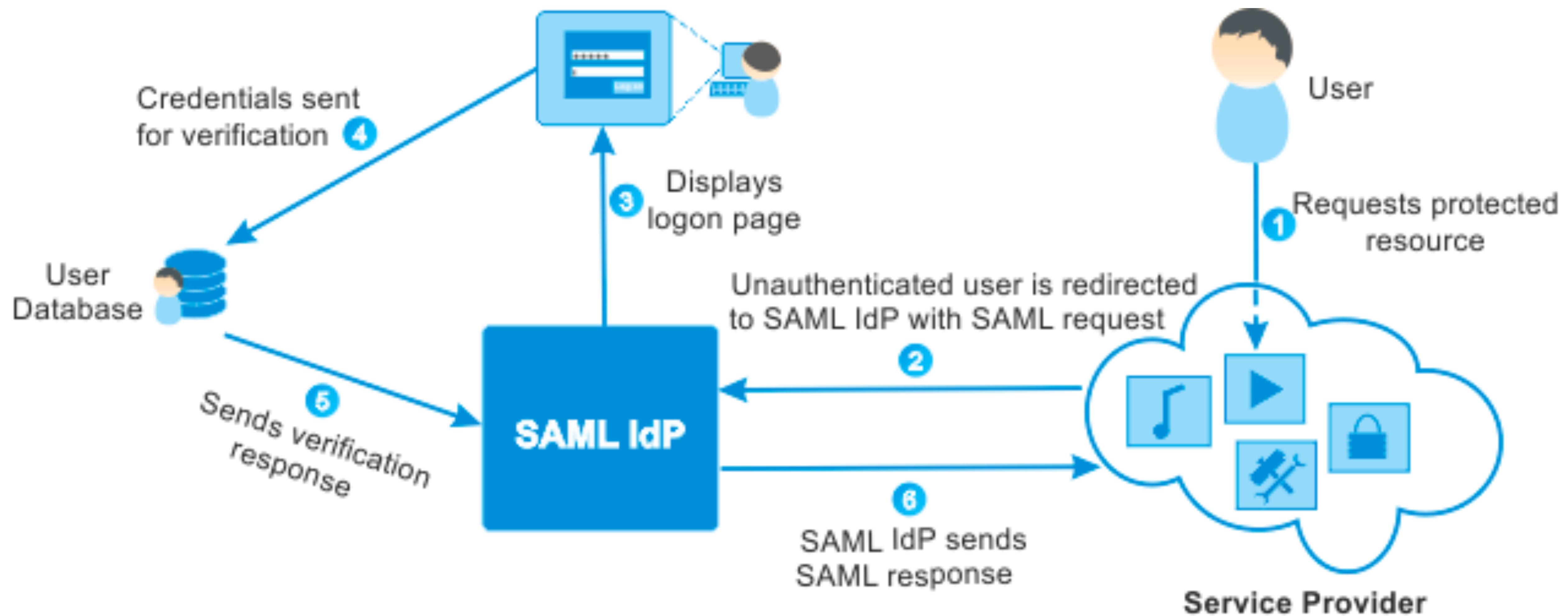
- autenticação: verifica a identidade digital do usuário de um sistema,
- autorização: garante que um usuário autenticado somente tenha acesso aos recursos autorizados
- auditoria: coleta de informações sobre o uso dos recursos de um sistema pelos seus usuários

Formas de autenticação

- HTTP Básica (tb conhecida como AuthHeaders):
 - transmissão dos dados no header por base64
 - Sem criptografia própria;
 - Normalmente a cada requisição necessita do envio dos dados do cliente;

Formas de autenticação

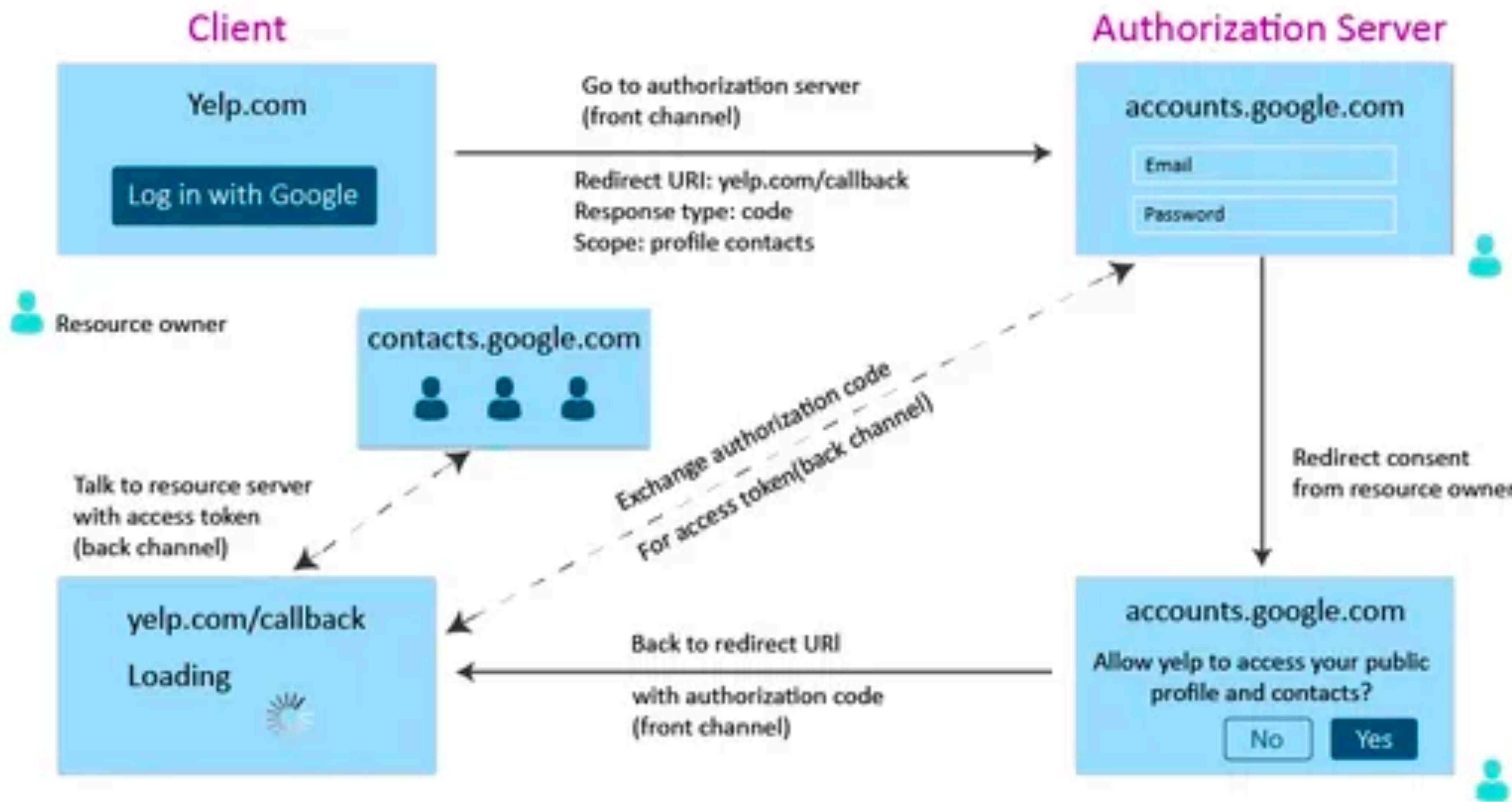
- SAML - Security Assertion Markup Language:
 - Prove um opção de SSO (SingleSignOn) usando XML;
 - Normalmente mais usado com web browsers;
 - Separa o backend em um provedor de serviços e um provedor de identidade.



Formas de autenticação

- oAuth2:
 - Na verdade é um protocolo de autorização e não autenticação.
 - Terceiriza a autenticação para um serviço de terceiros;
 - Necessita de APIs para a troca de informações;
 - Uso de tokens.

OAuth 2.0 authorization code flow



Formas de autenticação

- JWT (Json Web Tokens):
 - É com uma extensão do OAuth 2.0.
 - Formato JSON para troca de informações de forma leve, segura e contida;
 - Ele é autocontido: possui junto do token as permissões e as requisições necessárias para a requisição.



Header

```
base64enc({  
  "alg": "HS256",  
  "typ": "JWT"  
})
```

Payload

```
base64enc({  
  "iss": "toptal.com",  
  "exp": 1426420800,  
  "company": "Toptal",  
  "awesome": true  
})
```

Signature

```
HMACSHA256(  
  base64enc(header)  
  + '.',  
  base64enc(payload)  
  , secretKey)
```



The user enters in their username and password.

An authentication code is sent to the user's mobile device.

The user enters in their authentication code to log into the application.

Two Factor Authentication

2FA

Security



SQL INJECTION

- Localhost:8080/books?authors=asimov;Select * from Users;
- Filter = request[authors];
- Select * from books where author = filter
 - Select * from books where author = asimov;Select * from Users;



**Coloca novos dados para que as
consultas SQL retornem dados sigilosos.**

SQL INJECTION

SQL Injection

```
uname = request.POST['username']
passwd = request.POST['password']
sql = "SELECT id FROM users WHERE username=''" + uname + "'"
      AND password=''" + passwd + "'"
database.execute(sql)
```

```
sql = "SELECT id FROM users WHERE username=' ' AND
      password='password' OR 1='1'
```

SQL Injection

```
uname = request.POST['username']
passwd = request.POST['password']
sql = "SELECT id FROM users WHERE username=''" + uname + "'"
AND password=''" + passwd + "'"
database.execute(sql)
```

```
sql = "SELECT id FROM users WHERE username=' ' AND
password='password' OR 1='1'"
```

Para Evitar:

- Isolar a entrada
- Usar ORM (Object Relational Mapping)

Broken Authentication

- credential stuffing: o hacker tem uma lista de usuários e senhas validas de um serviço e utiliza em outros;
- Ataque de força bruta: o hacker tenta diversas combinações de login e senha (permite senhas fracas);
- Session Hijack: o hacker sequestra uma sessão valida.



Broken Authentication

- credential stuffing: o hacker tem uma lista de usuários e senhas validas de um serviço e utiliza em outros;
- Ataque de força bruta: o hacker tenta todos os possíveis combinações de senha (permite senhas fracas);
- Session Hijack: o hacker sequestra uma sessão de usuário;

Para Evitar:

- Forçar o uso de senhas complexas;
- Limitar o numero de login falhos
- Usar um hash
- Fazer um timeout nas sessões
- Usar 2FA



Broken Access Control

Broken Access Control

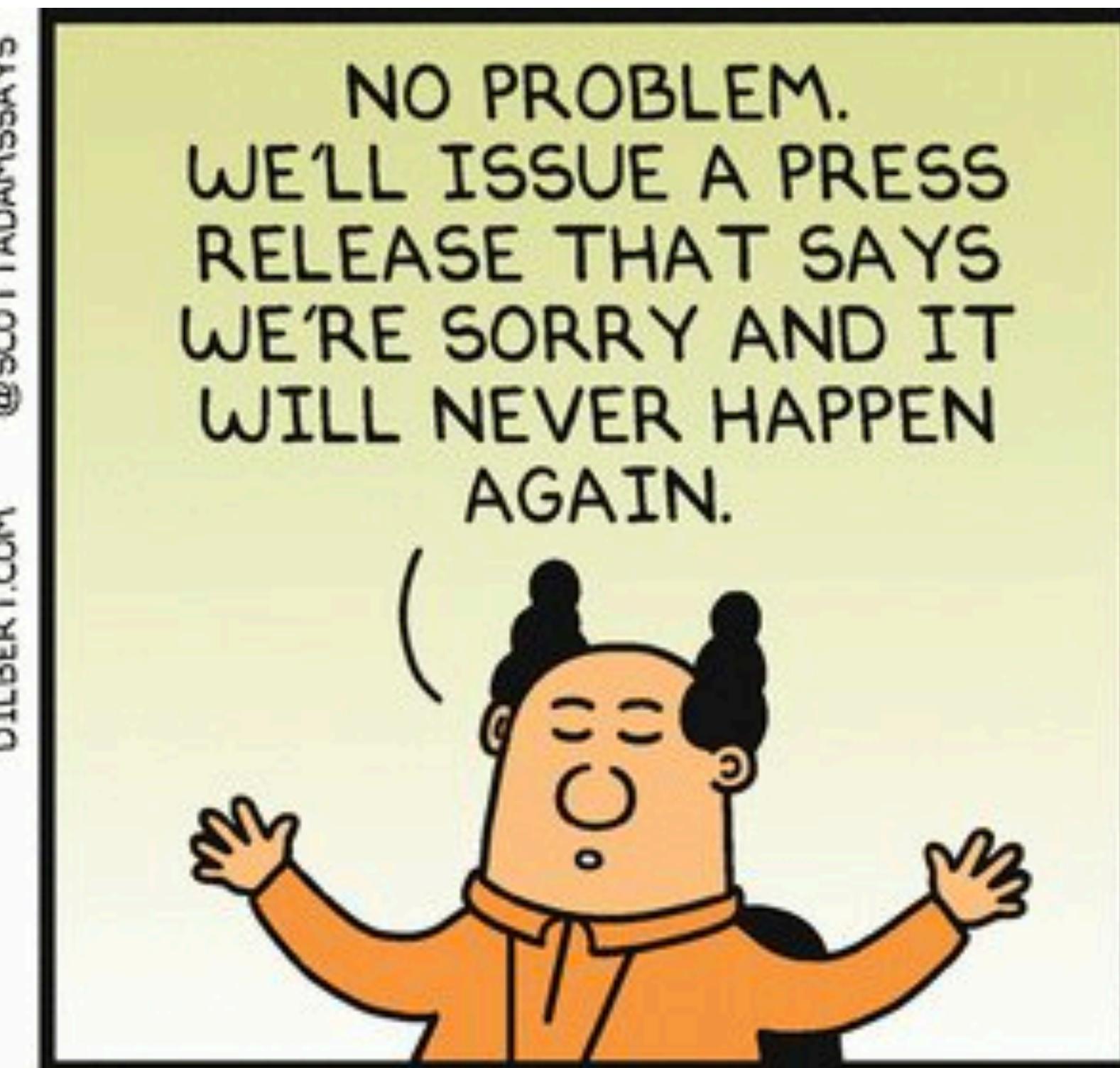
- Controle de acesso possui as regras do que o usuário pode ou não fazer;
- Tipos comuns usado por hackers:
 - CORS (Cross-Origin Resource Sharing) permite que um site acesse recursos de outro site mesmo estando em domínios diferentes.
 - Manipulação de metadata.
 - Teste de URLs (ex: <http://www.meusite.com/user/> to <http://www.meusite.com/admin>)

Broken Access Control

- Controle de acesso possui as regras do que o usuário pode ou não fazer;
- Tipos comuns usado por hackers:
 - CORS (Cross-Origin Resource Sharing) permite que um site acesse recursos de outro site mesmo estando em domínios diferentes.
 - Manipulação de metadata.
 - Teste de URLs (ex: <http://www.meusite.com/admin>)

Para Evitar:

- Fazer Deny por padrão
- Desabilitar listar os diretórios do servidor
- Invalidar os tokens no backend quando houver o logout

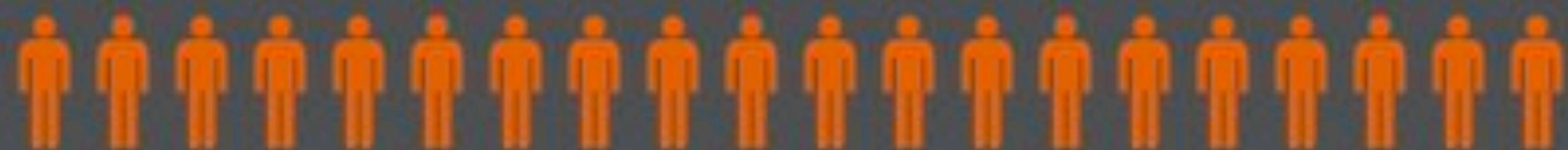


Data Breach

Roubo de informações privadas

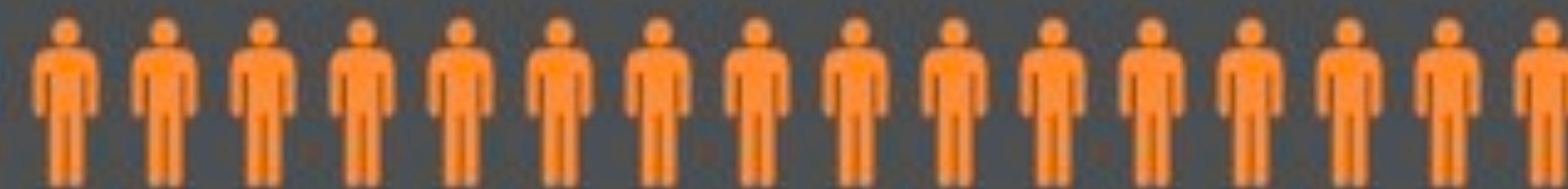
5 of The BIGGEST Data Breaches in History

2013
YAHOO



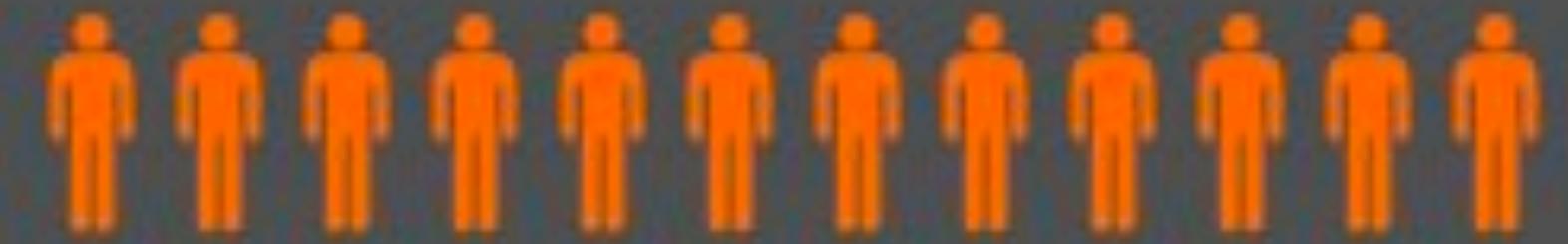
3Bn data
CAUSE: HACKING

2019
FIRST AMERICAN
FINANCIAL CORP



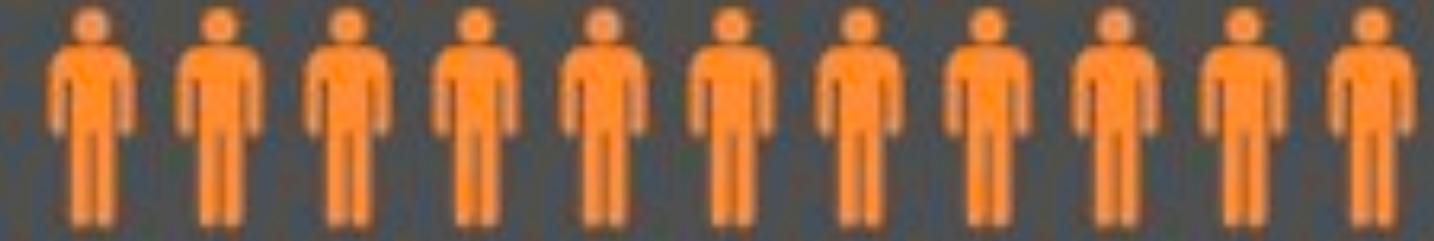
885Mn data
CAUSE: POOR SECURITY

2019
FACEBOOK



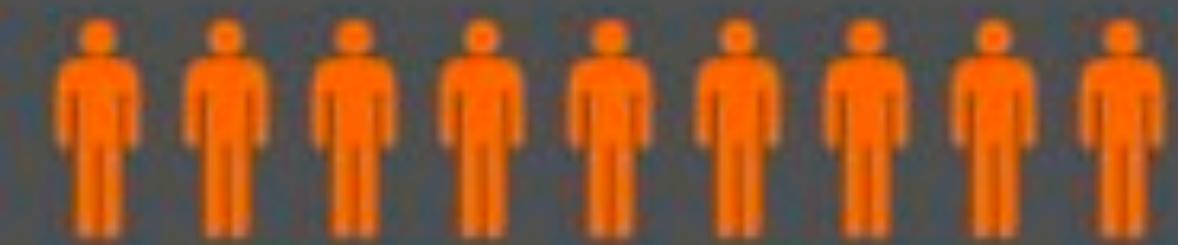
540Mn data
CAUSE: POOR SECURITY

2018
MARRIOTT



500Mn data
CAUSE: HACKING

2016
FRIEND FINDER
NETWORKS

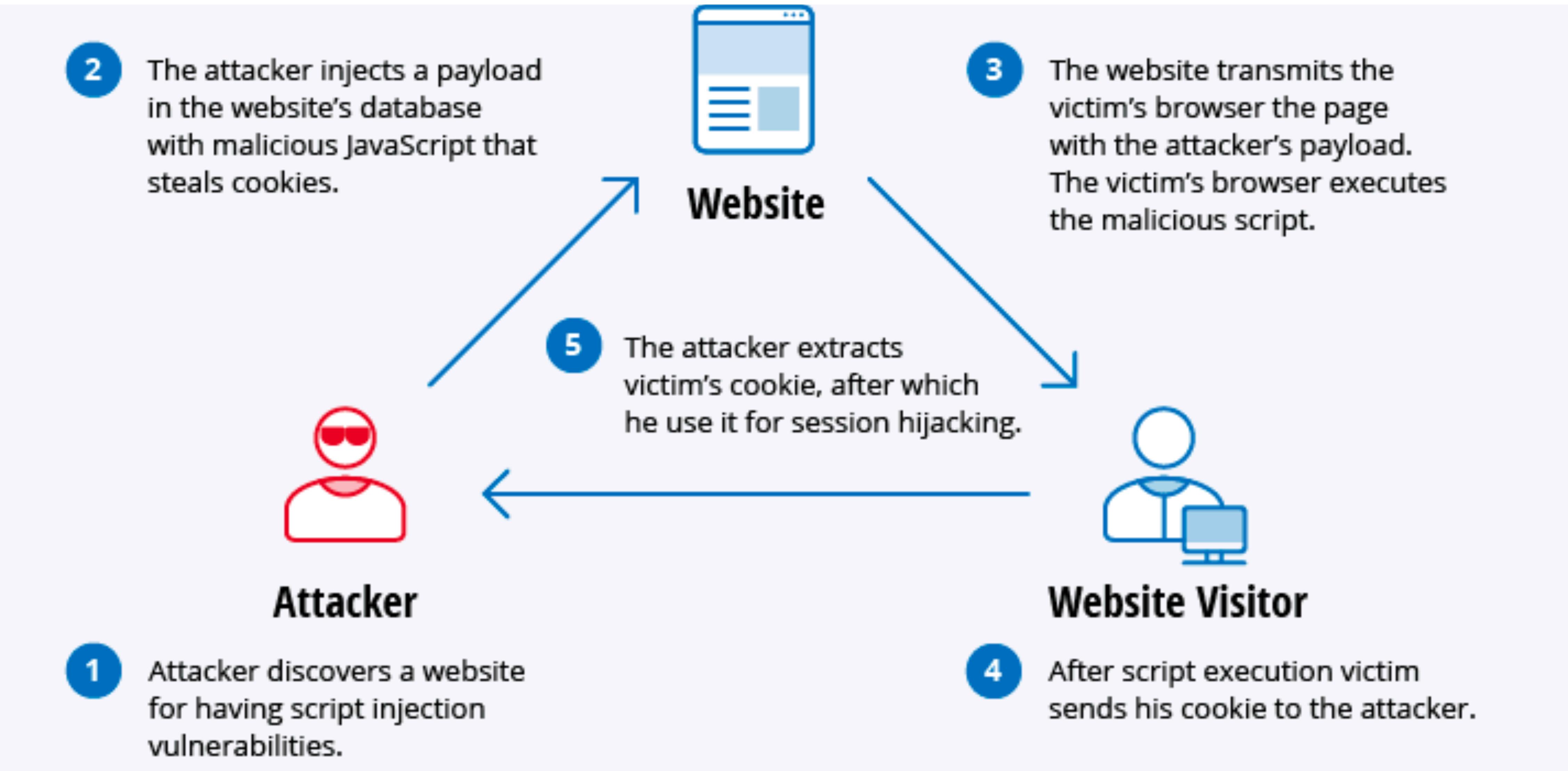


412Mn data
CAUSE: HACKING

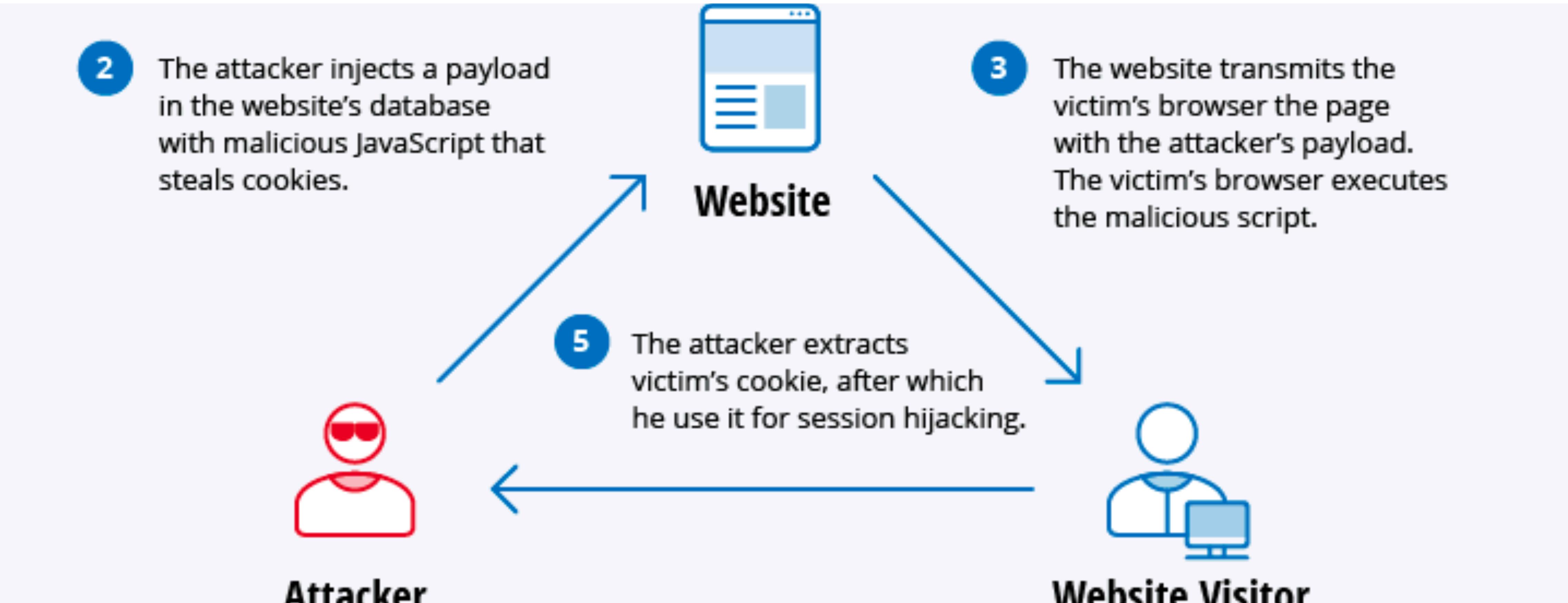
Como evitar

- Uso criptografia sempre que possível
- Identifique dados que são sensível e limite o acesso a eles;
- Uso os últimos e melhores algoritmos de criptografia





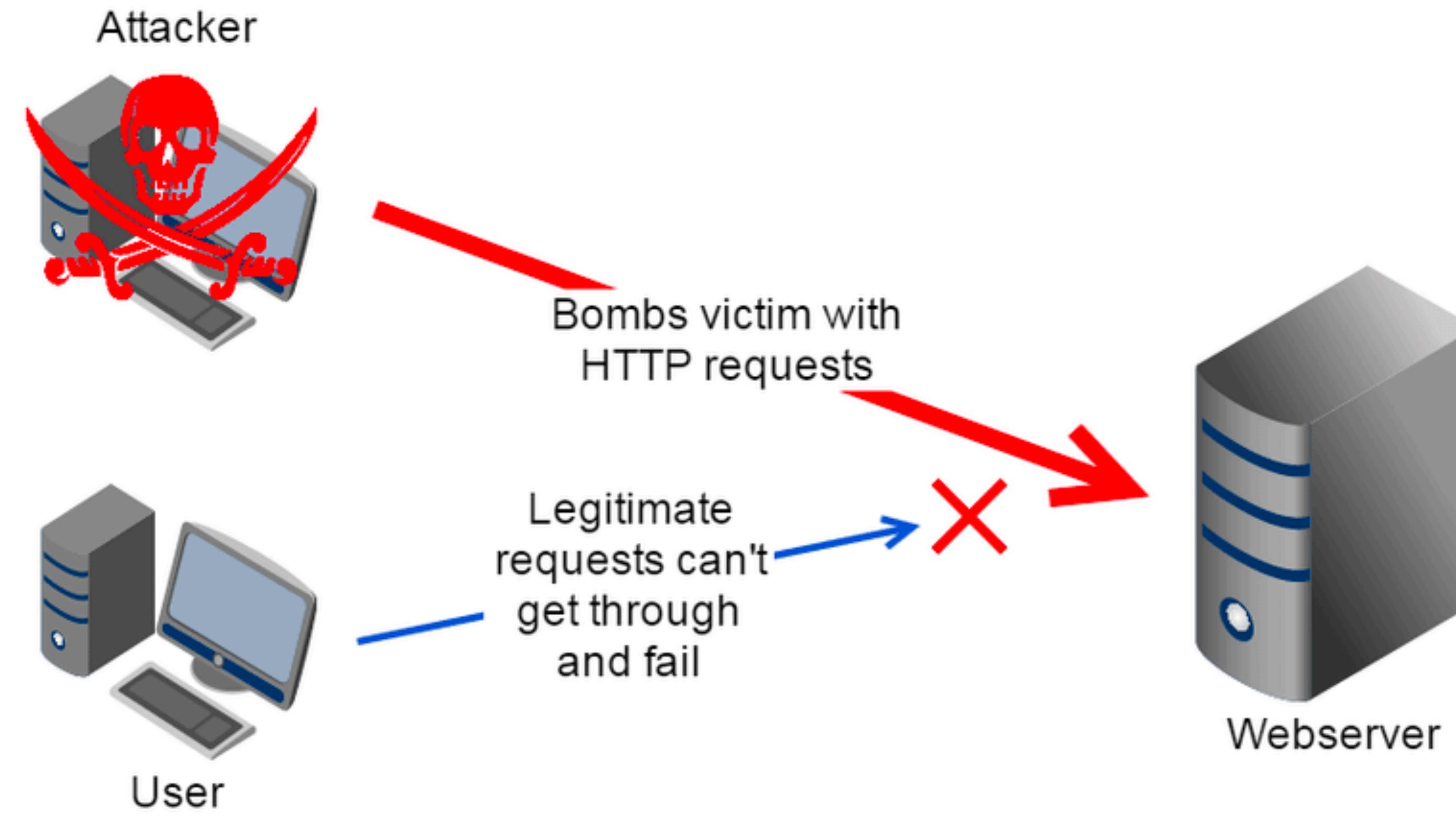
Cross-site scripting (XSS) attack



Cross-site scri

Para Evitar:

- Sempre validar o input
- Verificar o Js dos html para não ter nenhum código malicioso



(Distributed) Denial-of-service (DDoS) Attack

NETWORK



Protocolos de estado

Stateless

Stateful

Stateless

**Nenhum registro das interações anteriores são salvos no servidor.
Cada interação é tratada com base nas informações necessárias para a interação. Ex: HTTP**

Stateful

Stateless



 Simplifica a implementação do backend

 Menos memoria no servidor

 Não tem que gerenciar as sessões

Logica no cliente 😬



Logica no cliente 😬

Pode exigir request com mais informações 😳

Stateless

Servidor mantém o controle do estado de interação.
Permite que os dados passem a ser mantido entre diferentes requisições.
Ex: sessions

Stateful

Stateful



🕒 Mantém o controle do cliente durante a execução

⌚ Pode melhorar a performance



Necessidade de gerenciar as sessões



Uso maior de memória no servidor



Problemas para aumento de escala



Spring Boot

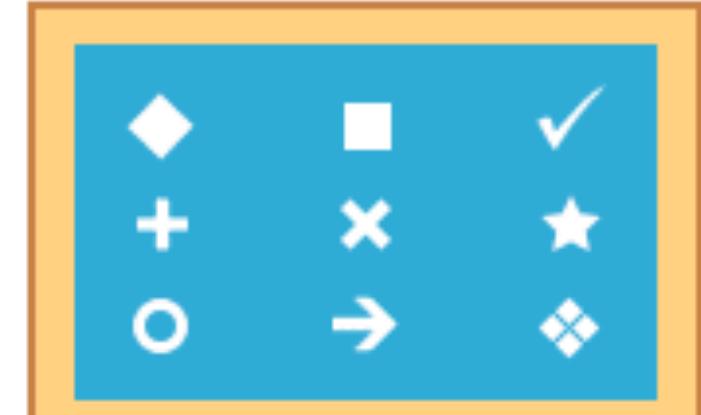
- Por padrão é Stateless, mas é possível implementar o padrão statefull

Backend Architectures

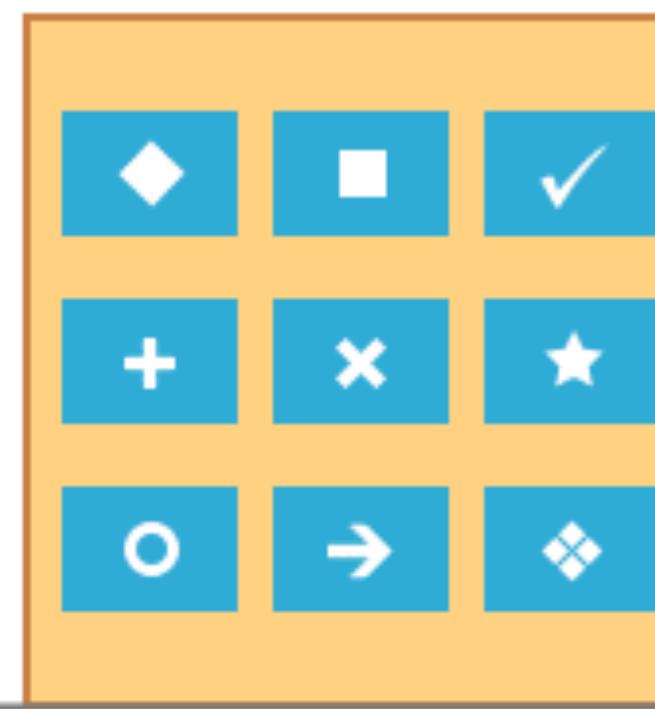
Complexity

code

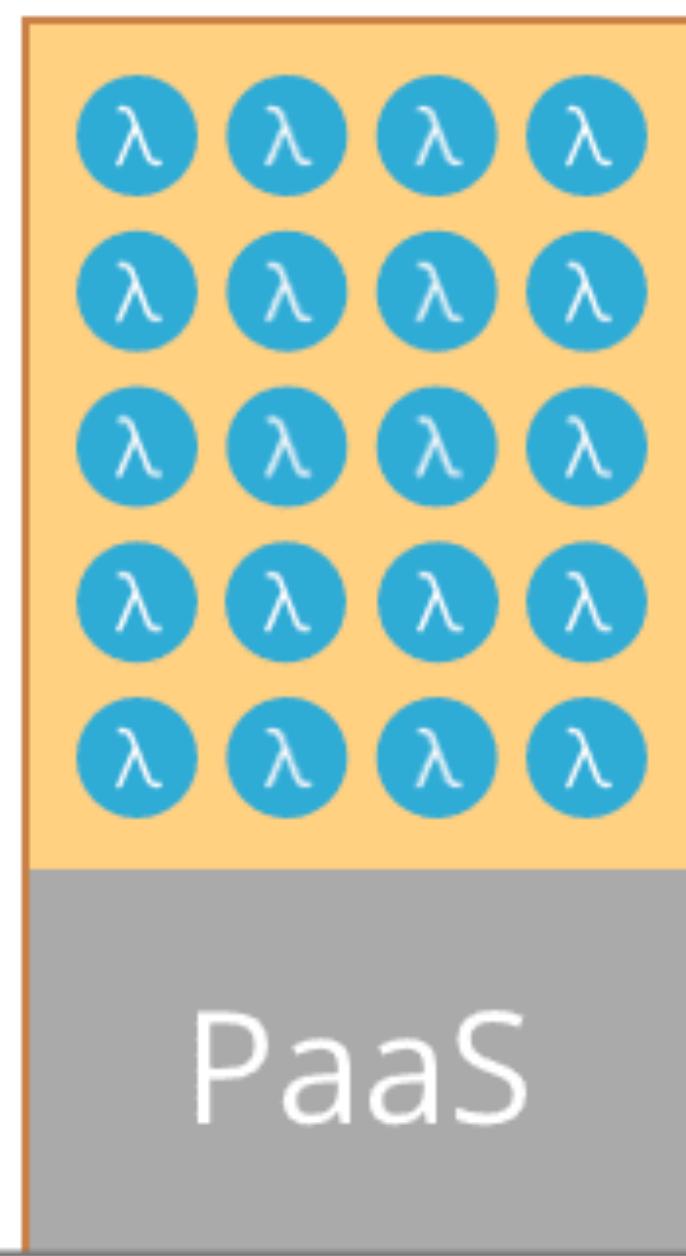
wiring



Monolith



Microservices

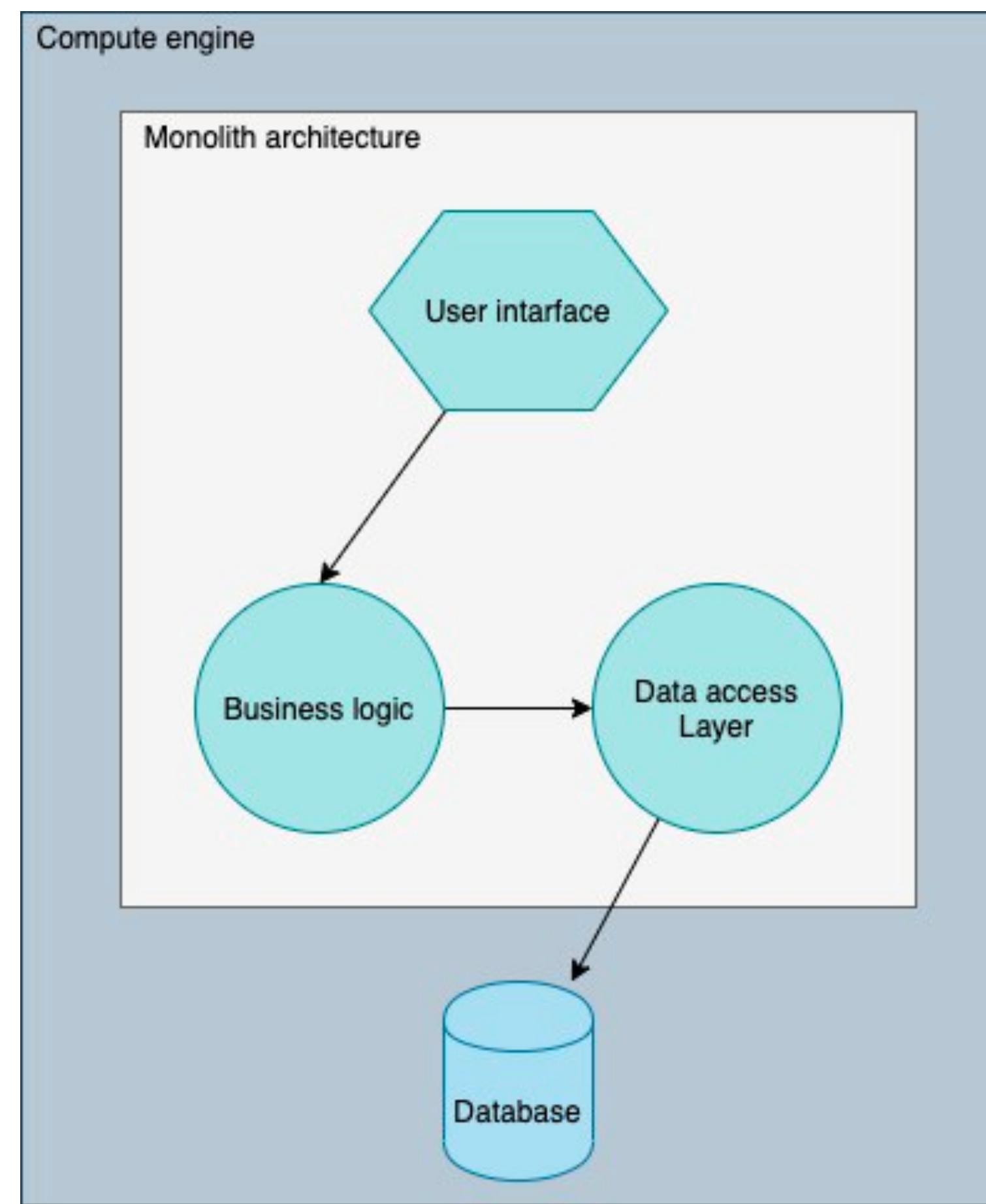


Serverless

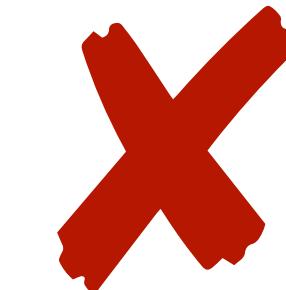
Monolith

- A arquitetura monolítica é quando toda a aplicação tem uma única base de código.
- Para tornar tudo mais gerenciável, os engenheiros dividem a aplicação em módulos de negócios ou de recursos técnicos.
- Os módulos se comunicam diretamente um com o outro chamando funções.

Monolith



Monolith



💻 simplicidade por estar tudo em
uma mesma aplicação

😊 Performance e latência

😎 Debugging

Escalabilidade 😬

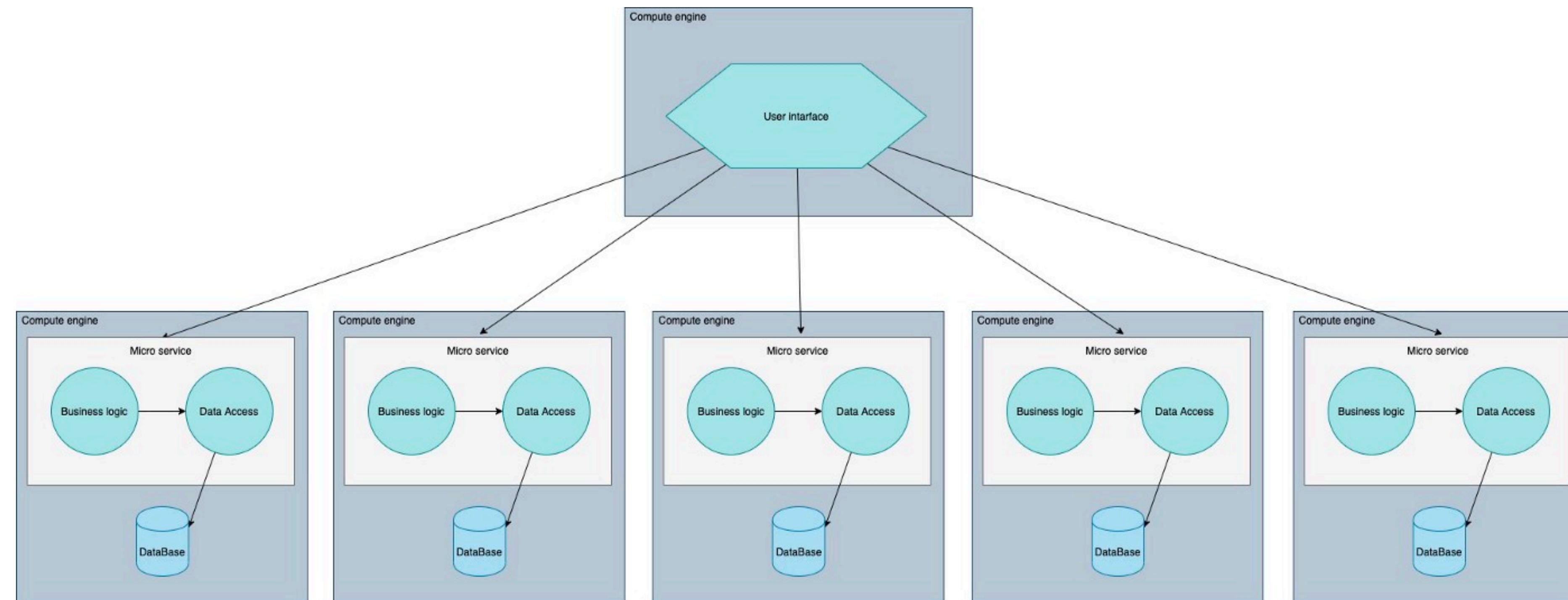
Confiabilidade 🙄

Realizar mudanças 🌎

Microserviços

- A arquitetura em microsserviços é uma variante de SOA com serviços micros usando protocolos leves.
- Um aplicativo do lado do servidor de microsserviço é uma aplicação estruturada como uma coleção de serviços em que cada serviço tem sua responsabilidade e pode se conectar a outros serviços.
- Cada microsserviço é independente, com seus próprios bancos de dados dedicados e outros recursos.
- O nome “micro” é sobre a capacidade de lógica de negócios de um único serviço e não sobre o tamanho real do serviço.
- Em comparação com uma arquitetura monolítica, cada serviço é implantado e vive separadamente.

Microserviços



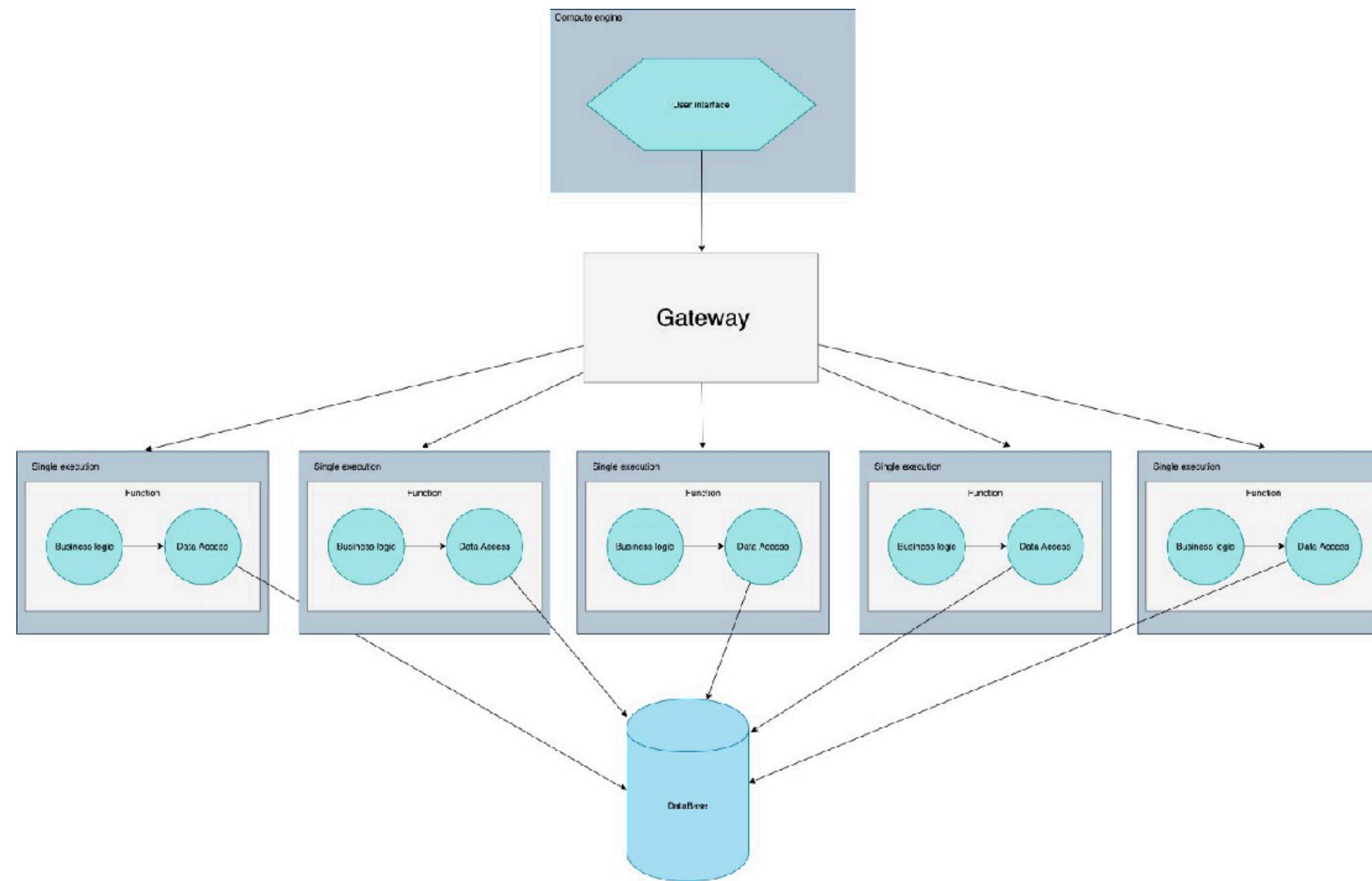
Microservices

	
 Agilidade	
 Escalabilidade	Debugging 😬
😊 Simplicidade	Deployment 🙄
😎 Desenvolvimento paralelo	Latência 
 Reusabilidade	

Serverless

- Só porque é chamado de serverless não significa que não tenha “nenhum servidor”.
- É uma abordagem que aproveita os serviços em nuvem para criar e executar serviços sem exigir gerenciamento de infraestrutura.
- Existem dois conceitos principais para engenharia de software sem servidor.
 - O primeiro é o Function as a Service (FaaS), que permite que os desenvolvedores carreguem fragmentos de funcionalidade na nuvem. Uma vez carregado, você pode executar os componentes de forma independente.
 - O segundo conceito é o Backend as a Service (BaaS). É um modelo de computação em nuvem que concede acesso a aspectos de back-end terceirizados enquanto grava e mantém apenas o serviço de front-end.

Serverless



Serverless



Dimensionamento feito pelo serviço Cloud



Menos configuração

Simplicidade de deploy e manutenção

Menor custo



Restrições de plataforma

Desenvolvimento baseado na plataforma



Reuso de código

MBaaS x Backend

MBaaS/DBaaS

Backend Server

MBaaS/DBaaS

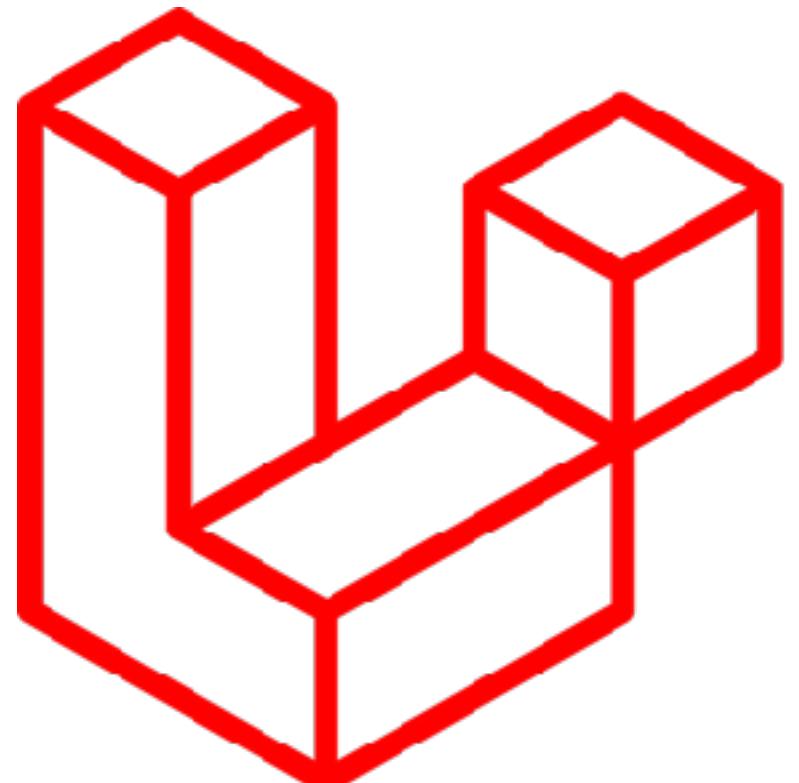
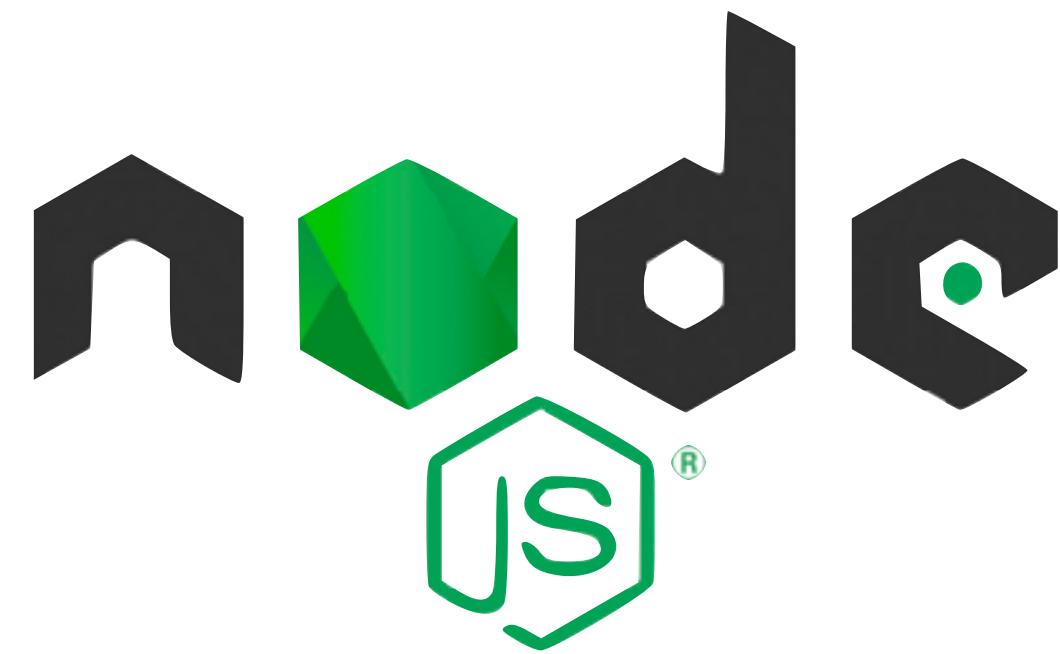
Backend Server

O back-end é o código executado no servidor, que recebe solicitações dos clientes e contém a lógica para enviar os dados apropriados de volta ao cliente.

O back-end também inclui o banco de dados, que armazenará persistentemente todos os dados do aplicativo.

Backend Server

Backend Server



Backend Server



😐 Logica no servidor



Mais Barato

😊 Mais liberdade de implementação -
MAIOR CONTROLE



Escolha da localização dos
servidores



Logica no servidor 😐

Mais Código e horas de
desenvolvimento 🤔

MBaaS/DBaaS

Backend Server

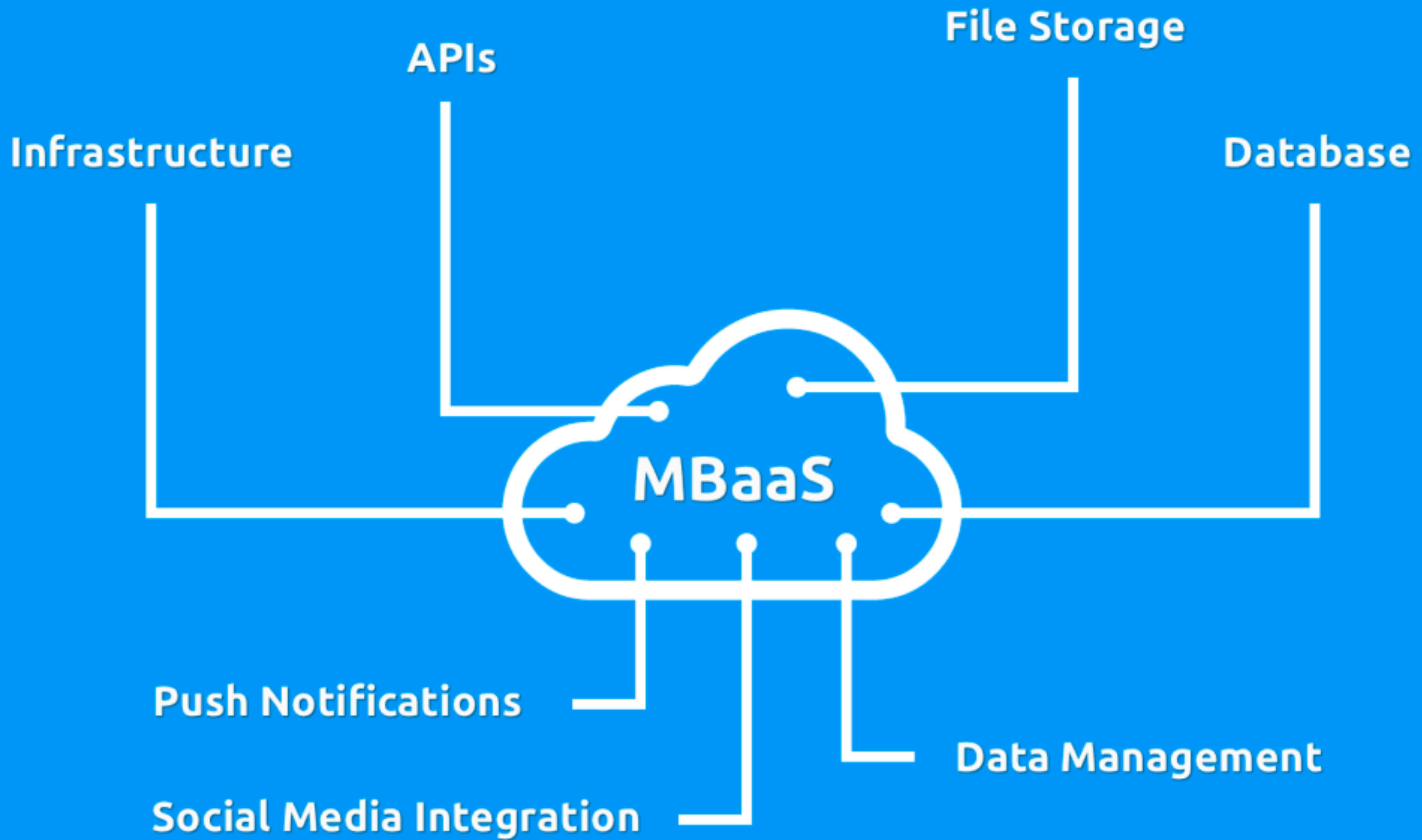
MBaaS/DBaaS

MBaaS é tem como premissa dar aos desenvolvedores fácil acesso aos serviços de back-end, os quais serão reusados em uma variedade de aplicações.

Backend Server

MBaaS/DBaaS



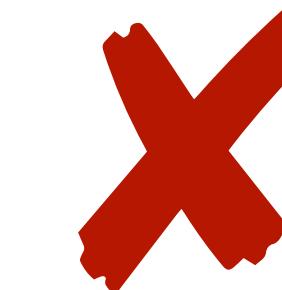


MBaaS/DBaaS



😊 Implementação +rápida (**tem AAA, BD, Push Notifications, Analytics em poucas linhas de código**)

😊 Logica no cliente



Pode ficar caro rapidamente 💰

Logica no cliente 😬

É uma caixa preta !?

Não existe tanta localização dos servidores 🌎

Projeto Final

- Vale a nota da disciplina.
- Deve ter um contexto, ser útil para algo.
- Vc tem 2 opções de projeto final:
 - Desenvolver um servidor com uso de banco de dados e testar com Swagger/Postman
 - Desenvolver um app com uso de um MBaaS com pelo menos autenticação e uso de banco de dados
- Entrega: Vídeo (entre 5 minutos e máximo de 15 minutos) demonstrando e explicando as partes principais do projeto no youtube e link do git com o código
- Pode ser em duplas
- Entrega no AVA: 16/10/2022