Software as a Service (SaaS) e Service-Oriented Architecture (SOA)

ACH2006 – Engenharia de Sistemas de Informação
SIN5005 – Tópicos em Engenharia de Software

Daniel Cordeiro

Escola de Artes, Ciências e Humanidades | EACH | USP

PERGUNTA

Em geral, qual afirmação sobre a relação entre os custos de consertar bugs no software e de melhorar o software é a mais precisa?

- $(Consertar um bug) $\geq \approx 2\times$ $(Melhorar)
- $(Consertar um bug) $\approx$ $(Melhorar)
- $(Melhorar) $\approx 2\times$ $(Consertar um bug)
- $(Melhorar) $\approx$ 2–4$\times$ $(Consertar um bug)

PERGUNTA

Em geral, qual afirmação sobre a relação entre os custos de consertar bugs no software e de melhorar o software é a mais precisa?

- $(Consertar um bug) $\geq \approx 2\times$ $(Melhorar)
- $(Consertar um bug) $\approx$ $(Melhorar)
- $(Melhorar) $\approx 2\times$ $(Consertar um bug)
- **$(Melhorar) ≈ 2–$4\times$ $(Consertar um bug)**

Manutenção vs. desenvolvimento: gasto com TI pelo governo dos EUA em 2010–2017

Figura 1: Fonte: *Government Accountability Office (GAO) report GAO-16-696T, Federal Agencies Need to Address Aging Legacy Systems, Testimony Before the Committee on Oversight and Government Reform, House of Representatives.* Publicado em 26 de maio de 2016.

Código legado vs. código belo

- Código legado: software antigo que continua a satisfazer as necessidades do cliente, mas que é difícil de evoluir devido a um projeto deselegante ou tecnologia antiquada
  - 60% do custo de manutenção do software causado pela adição de novas funcionalidades a código legado
  - 17% do custo é usado na correção de bugs
- **Problema vital** mas ignorado na maioria dos cursos de ES
- Diferente de código belo: satisfaz o cliente e é fácil de evoluir

QUALIDADE DE SOFTWARE

- Qualidade do produto (em geral): “adequação para o uso”
  - Valor de negócio para o cliente && para o fabricante
  - Garantia de qualidade: processos/padrões
- Qualidade do software:
  1. Satisfaz as necessidades do cliente: facilidade de usar, dá as respostas corretas, não para de funcionar (*crash*), etc.
  2. É fácil de ser depurado e melhorado pelos desenvolvedores
- Software QA: garante a qualidade e melhora os processos na organização do software

GARANTIAS

- Verificação: você construiu a coisa da maneira *certa*?
  - você seguiu a especificação?
- Validação: você construiu a coisa *certa*?
  - era isso o que o cliente queria?
  - a especificação estava certa?
- Hardware: geralmente foca a verificação
- Software: geralmente foca a validação
- Testes garantem a Qualidade do Software

TESTAR EXAUSTIVAMENTE É IMPRATICÁVEL

- Dividir para conquistar: realize testes diferentes em cada fase do desenvolvimento do software
  - o nível superior não refaz os testes do nível inferior
- *Cobertura*: várias medidas que indicam o quanto do código foi exercitado pelo conjunto de testes

Tipos de testes

Teste de sistema ou aceitação verifica se o programa integrado cumpre a especificação

Teste de integração verifica se as interfaces entre diferentes unidades tem as mesmas hipóteses; se elas se comunicam corretamente

Módulo ou teste funcional verifica diversas unidades individuais

Teste de unidade verifica se um único método faz aquilo que é esperado

Arquiteturas de Aplicações SaaS

O QUE ELES TÊM EM COMUM?

Chrome

INTRODUCING
amazon echo

SHRINK-WRAPPED SOFTWARE (SWS)

Software “embalado”

- cliente específico disponibilizado como um binário, upgrades frequentes
  - precisam trabalhar com muitos tipos de hardware diferentes, S.O., bibliotecas, ...
  - difíceis de manter
  - precisa de muito teste de compatibilidade em cada lançamento

Alternativa: aplicação centralizada em servidor, thin client

- busca, e-mail, comércio eletrônico, redes sociais, vídeos, etc.
- mais recentemente engloba ferramentas de produtividade (Documentos Google/Office 365), finanças, IDEs (Codenvy)
- diversos pesquisadores acreditam que esse seja o futuro dos softwares (eu inclusive)

POR QUE SAAS > SWS ?

1. Com SaaS não há mais preocupações sobre características de hardware ou S.O.
2. Dados são armazenados de forma segura e confiável nos servidores
3. É fácil fazer com que grupos interajam com os mesmos dados
4. Se a quantidade de dados for muito grande ou se mudar com muita frequência, é mais fácil manter uma única cópia centralizada
5. Uma cópia do software em um único ambiente de hardware/S.O. $\Rightarrow$ ausência de problemas de compatibilidades $\Rightarrow$ é possível fazer testes beta de novas funcionalidades em 1% dos usuários
6. Uma cópia $\Rightarrow$ simplicidade nos upgrades para desenvolvedores e evita requisições de upgrades dos usuários

E os apps nativos para smartphones (Android, iOS, ...)?

Vantagens

- Usa capacidades extras de hardware não disponíveis em HTML 5
- *Pode ser* mais rápido... ou não — muitos são apps HTML 5 que rodam em um "contêiner" nativo
- Sua marca está na tela principal do usuário
  - mas você também pode conseguir isso com o uso dos favoritos

Desvantagens

- Mais difíceis de manter
- Upgrades voltam a se tornar um problema do usuário

Pergunta:
Será que podemos projetar um software de forma que possamos recombinar módulos independentes para oferecer a muitos apps sem envolver muita programação?

> *[Jeff Bezos, CEO da Amazon] realized long before the vast majority of Amazonians that Amazon needs to be a platform.*
>
> — Steve Yegge, Googler, ex-Amazon, em seu blog em 2011

Mensagem do CEO em 2002: A Amazon deve usar SOA!

1. *Todas as equipes, de agora em diante, exibirão seus dados e funcionalidade através de interfaces dos serviços.*
2. *As equipes devem se comunicar entre si através dessas interfaces.*
3. *Não haverá nenhuma outra forma de comunicação permitida entre processos: nenhuma conexão direta, nenhuma leitura direta do estoque de dados de outras equipes, nenhum modelo de memória compartilhada e nenhuma excessão de qualquer outra natureza. A única comunicação permitida é através das chamadas às interface dos serviços através da rede.*
4. *Não importa de quais tecnologias elas se utilizam. HTTP, CORBA, Pub/Sub, protocolos customizados – não importa. O Bezos não se importa.*

Mensagem do CEO em 2002: A Amazon deve usar SOA!

1. *Todas as equipes, de agora em diante, exibirão seus dados e funcionalidade através de interfaces dos serviços.*
2. *As equipes devem se comunicar entre si através dessas interfaces.*
3. *Não haverá nenhuma outra forma de comunicação permitida entre processos: nenhuma conexão direta, nenhuma leitura direta do estoque de dados de outras equipes, nenhum modelo de memória compartilhada e nenhuma excessão de qualquer outra natureza. A única comunicação permitida é através das chamadas às interface dos serviços através da rede.*
4. *Não importa de quais tecnologias elas se utilizam. HTTP, CORBA, Pub/Sub, protocolos customizados – não importa. O Bezos não se importa.*

Mensagem do CEO em 2002: A Amazon deve usar SOA!

1. *Todas as equipes, de agora em diante, exibirão seus dados e funcionalidade através de interfaces dos serviços.*
2. *As equipes devem se comunicar entre si através dessas interfaces.*
3. *Não haverá nenhuma outra forma de comunicação permitida entre processos: nenhuma conexão direta, nenhuma leitura direta do estoque de dados de outras equipes, nenhum modelo de memória compartilhada e nenhuma excessão de qualquer outra natureza. A única comunicação permitida é através das chamadas às interface dos serviços através da rede.*
4. *Não importa de quais tecnologias elas se utilizam. HTTP, CORBA, Pub/Sub, protocolos customizados – não importa. O Bezos não se importa.*

Mensagem do CEO em 2002: a Amazon deve usar SOA!

1. *Todas as equipes, de agora em diante, exibirão seus dados e funcionalidade através de interfaces dos serviços.*
2. *As equipes devem se comunicar entre si através dessas interfaces.*
3. *Não haverá nenhuma outra forma de comunicação permitida entre processos: nenhuma conexão direta, nenhuma leitura direta do estoque de dados de outras equipes, nenhum modelo de memória compartilhada e nenhuma excessão de qualquer outra natureza. A única comunicação permitida é através das chamadas às interface dos serviços através da rede.*
4. *Não importa de quais tecnologias elas se utilizam. HTTP, CORBA, Pub/Sub, protocolos customizados – não importa. O Bezos não se importa.*

Mensagem do CEO em 2002: A Amazon deve usar SOA! (II)

5. *Todas as interfaces dos serviços, sem exceção, devem ser projetadas a partir do zero para serem exteriorizadas. Isto é, a equipe precisa planejar e projetar para ser capaz de exibir a interface para desenvolvedores do mundo exterior. Sem exceções.*
6. *Qualquer um que não fizer isso será despedido.*
7. *Obrigado; Tenham um bom dia!*

Mensagem do CEO em 2002: a Amazon deve usar SOA! (II)

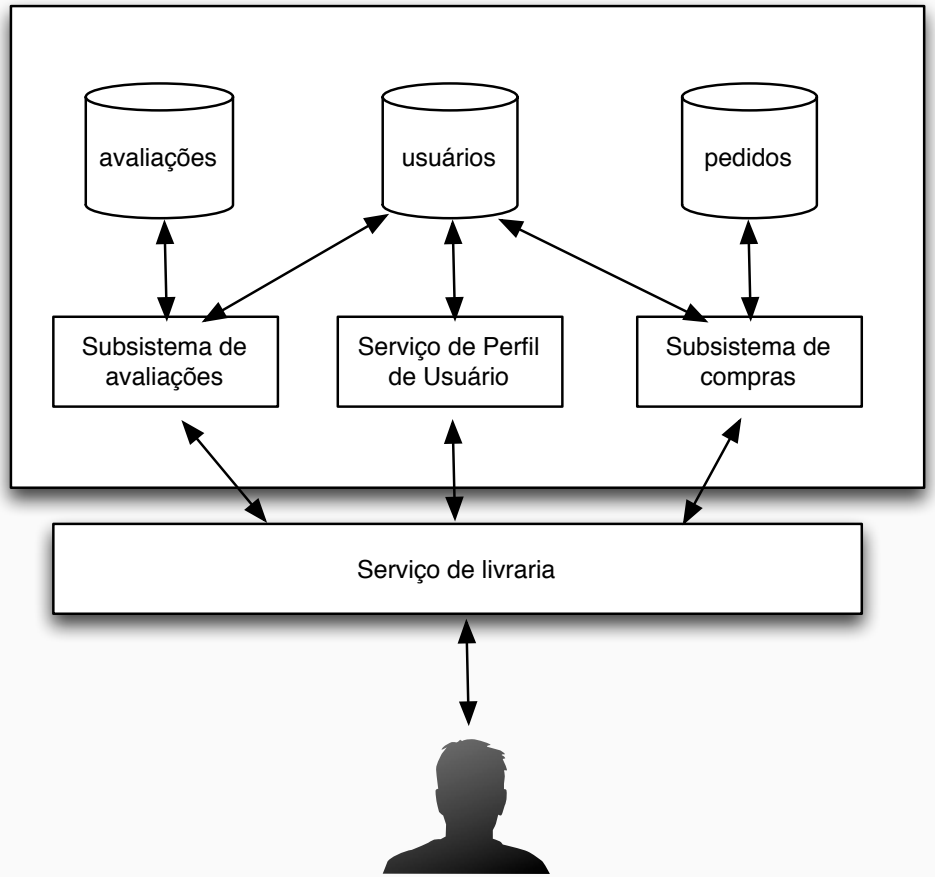
5. *Todas as interfaces dos serviços, sem exceção, devem ser projetadas a partir do zero para serem exteriorizadas. Isto é, a equipe precisa planejar e projetar para ser capaz de exibir a interface para desenvolvedores do mundo exterior. Sem exceções.*
6. *Qualquer um que não fizer isso será despedido.*
7. *Obrigado; Tenham um bom dia!*

Mensagem do CEO em 2002: a Amazon deve usar SOA! (II)

5. *Todas as interfaces dos serviços, sem exceção, devem ser projetadas a partir do zero para serem exteriorizadas. Isto é, a equipe precisa planejar e projetar para ser capaz de exibir a interface para desenvolvedores do mundo exterior. Sem exceções.*
6. *Qualquer um que não fizer isso será despedido.*
7. *Obrigado; Tenham um bom dia!*

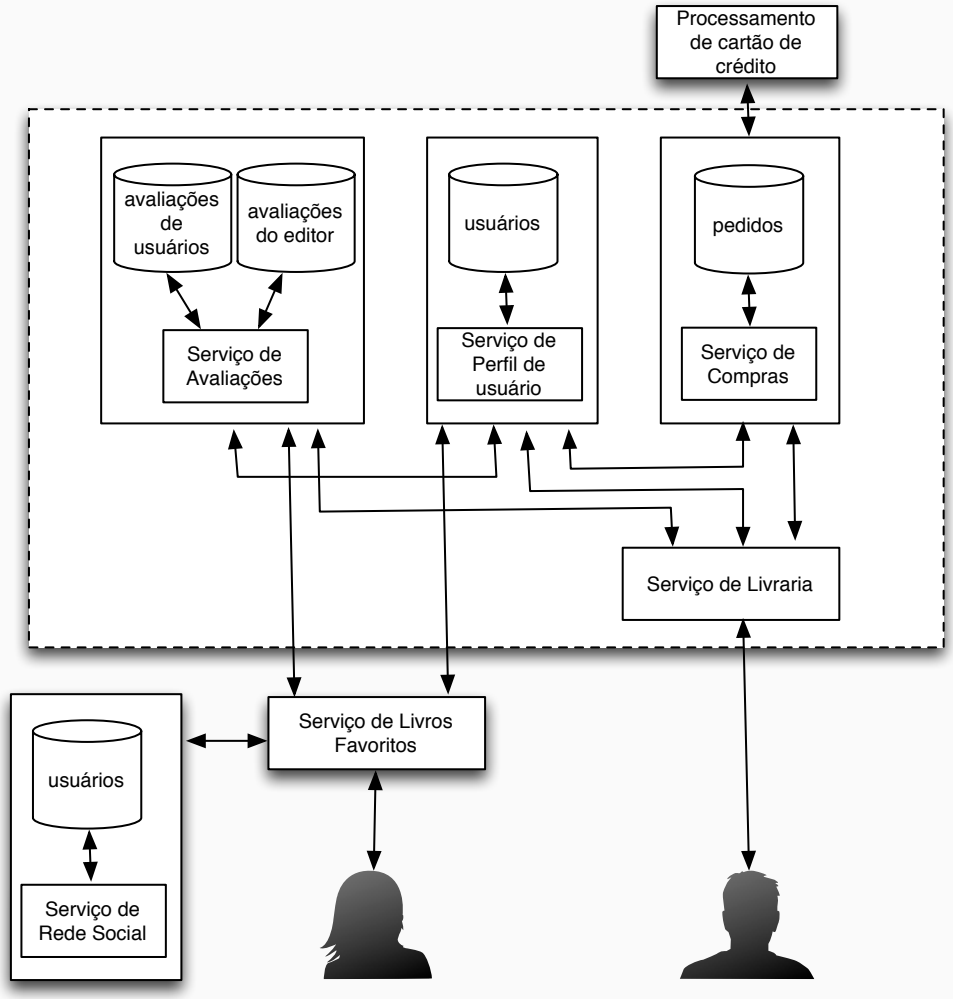
LIVRARIA: SILO

- Subsistemas internos podem compartilhar dados diretamente
- Todos os subsistemas acessados por uma única API



LIVRARIA: SOA

- Subsistemas independentes, como se estivessem em *datacenters* diferentes
- Pode ser recombinado para criar novos serviços (Serviço de Livros Favoritos)



Computação em nuvem

QUAL O HARDWARE IDEAL PARA SAAS?

- Amazon, Google, Microsoft... desenvolveram hardware para executar SaaS
- O que eles usam? Mainframes? Supercomputadores?
- Como que desenvolvedores independentes de software podem construir aplicativos SaaS e serem competitivos sem investir em hardware especializado com fazem as grandes empresas?

QUAL O HARDWARE IDEAL PARA SAAS?

SaaS dependem de 3 coisas da infraestrutura:

1. **Comunicação**
   Permitem os seus consumidores interajam com o serviço
2. **Escalabilidade** (*scalability*)
   Flutuação na demanda + possibilidade de adicionar mais serviços para lidar com crescimento rápido dos usuários
3. **Confiabilidade** (*dependability*)
   Serviço & comunicação disponíveis 24x7

SERVIÇOS EM AGLOMERADOS

Aglomerados de computadores (*clusters*): computadores comuns (*commodities*) conectados por *switches* Ethernet comuns

1. Mais escalável do que servidores convencionais
2. Mais barato do que servidores convencionais
    - 20x mais barato que um grande servidor equivalente
3. Confiabilidade obtida com uso extensivo de redundância
4. Poucos operadores para milhares de servidores
    - Seleção cuidadosa de hardware/software idênticos
    - Monitores de máquinas virtuais simplificam a operação

ARMAZENS DE COMPUTADORES

- Aglomerados cresceram de 1.000 servidores para 100.000 graças a demanda dos clientes para apps SaaS
- Economia de escala derrubaram o preço de grandes *datacenters* em fatores de 3x a 8x
    - Comprar, abrigar, operar 100.000 vs. 1.000 computadores
- *Datacenters* tradicionais usavam 10% – 20%
- Ganham $ oferecendo computação utilitária (pague apenas pelo que usa) a preços menores do que os que os clientes conseguiriam

Computação utilitária / plataformas públicas de computação em nuvem

- Oferecem poder computacional, armazenamento, comunicação por centavos/hora
- Não há vantagens ao aumentar a escala:

  1.000 computadores @ 1 hora
  = 1 computador @ 1.000 horas
- Usuário da nuvem tem ilusão de escalabilidade infinita
  - existem tantos computadores quanto você puder pagar
- Maiores exemplos: Amazon Web Services, Google App Engine, Microsoft Azure, UOL Cloud e Locaweb

EXEMPLOS DE PREÇOS NA AWS

Região: Leste dos EUA (Norte da Virgínia)

	vCPU	ECU	Memória (GiB)	Armazenamento da instância (GB)	Uso do Linux/UNIX
Uso geral – geração atual					
a1.medium	1	N/D	2 GiB	Somente EBS	0,0255 USD por hora
a1.large	2	N/D	4 GiB	Somente EBS	0,051 USD por hora
a1.xlarge	4	N/D	8 GiB	Somente EBS	0,102 USD por hora
a1.2xlarge	8	N/D	16 GiB	Somente EBS	0,204 USD por hora
a1.4xlarge	16	N/D	32 GiB	Somente EBS	0,408 USD por hora
a1.metal	16	N/D	32 GiB	Somente EBS	0,408 USD por hora
t3.nano	2	Variável	0,5 GiB	Somente EBS	0,0052 USD por hora
t3.micro	2	Variável	1 GiB	Somente EBS	0,0104 USD por hora
t3.small	2	Variável	2 GiB	Somente EBS	0,0208 USD por hora
t3.medium	2	Variável	4 GiB	Somente EBS	0,0416 USD por hora
t3.large	2	Variável	8 GiB	Somente EBS	0,0832 USD por hora
t3.xlarge	4	Variável	16 GiB	Somente EBS	0,1664 USD por hora
t3.2xlarge	8	Variável	32 GiB	Somente EBS	0,3328 USD por hora

COMPUTAÇÃO UTILITÁRIA

- 72º lugar na lista TOP 500 dos supercomputadores mais rápidos do mundo em 2012
  - 532 computadores, 17000 cores = 240 TeraFLOPS
  - $ 1300 por hora
  - 240º lugar na lista de jun/2016
- FarmVille no AWS
  - antes, o maior jogo online tinha 5M usuários
  - 4 dias: 1M; 2 meses: 10M; 9 meses: 75M
- IBM Watson: 90 servidores IBM Power 750
  - 3,5 GHz 8 cores/servidor
  - 90 por $ 2,4 / hora = $ 200/hora

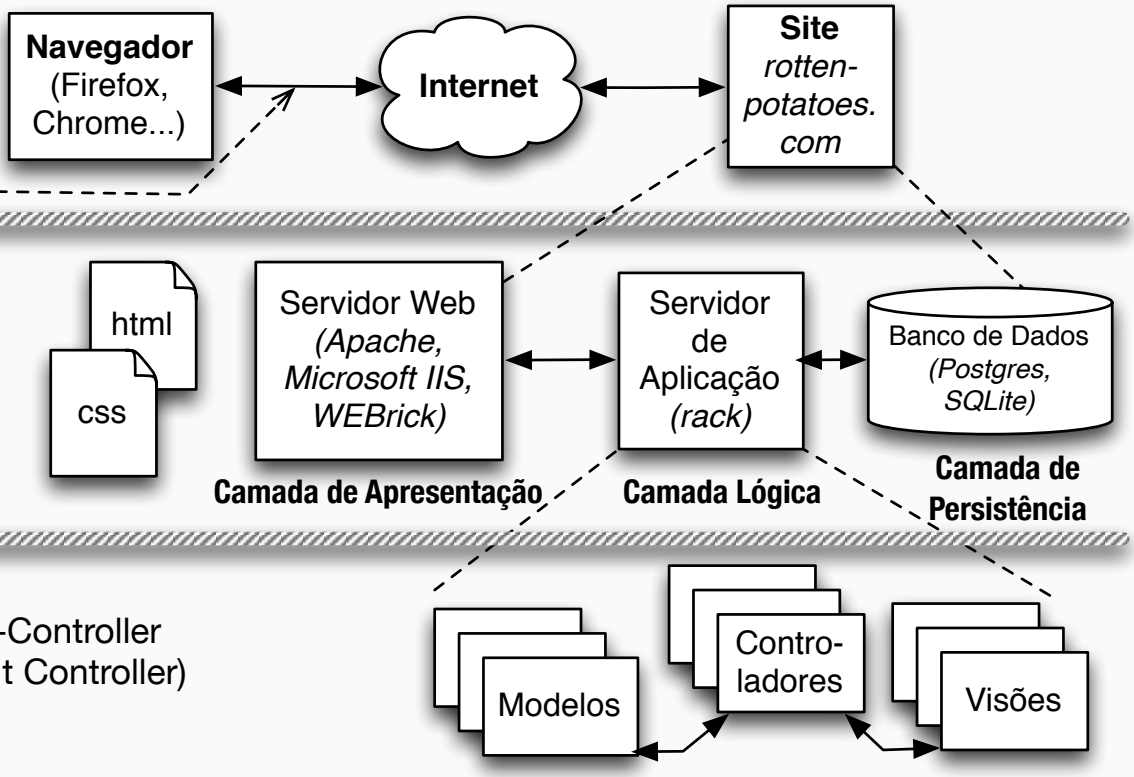
A Arquitetura de Aplicações SaaS

§2.1 100.000 pés
• Cliente-servidor (vs. P2P)

§2.2 50.000 pés
• HTTP e URIs

§2.3 10.000 pés
• XHTML e CSS

§2.4 5.000 pés
• Arquitetura de 3 camadas
• Escalabilidade horizontal



§2.5 1.000 pés—Model-View-Controller
(vs. Page Controller, Front Controller)

§2.6 500 pés: Modelos Active Record (vs. Data Mapper)
§2.7 500 pés: Controladores REST (*Representational State Transfer* para ações auto-contidas)
§2.8 500 pés: Template View (vs. Transform View)

• **Active Record** • **REST** • **Template View**
• Data Mapper • Transform View

VISÃO A 100.000 PÉS

- A web segue uma arquitetura cliente–servidor
- Processos fundamentalmente orientados a requisições–resposta

PROTOCOLOS TCP/IP

- Endereços IP (*Internet Protocol*) identificam uma interface física de rede com quatro octetos (`200.144.183.244`, 32 bits) ou com oito grupos de quatro dígitos hexadecimal (`2001:12d0:c000:91::41`, 128 bits)
  - o endereço especial `127.0.0.1` (`0:0:0:0:0:0:0:1`) aponta para "este computador", chamado de `localhost`, mesmo se não estiver conectado a Internet!
- TCP/IP (*Transmission Control Protocol / Internet Protocol*)
  - IP: não há garantias, pacotes são enviados tão bem quanto possível (*best-effort*) de um endereço IP para outro
  - TCP: torna o IP confiável ao detectar problemas no envio de pacotes (que não chegaram, que chegam fora de ordem, erros de transmissão, lentidão na rede, etc.) e se recuperar desses problemas
  - *Portas* TCP permitem múltiplas aplicações TCP no mesmo computador

VISÃO A 100.000 PÉS

- A web segue uma arquitetura cliente–servidor
- Processos fundamentalmente orientados a requisições–resposta
- DNS (*Domain Name System)* é outro tipo de servidor que mapeia nomes a endereços IP

- Protocolo ASCII de requisição–resposta para transferência de informação na Web
- Requisições HTTP incluem:
  - método de requisição (GET, POST, etc.)
  - Uniform Resource Identifier (URI)
  - versão do protocolo HTTP entendida pelo cliente
  - cabeçalhos — informação extra referente à requisição
- Resposta HTTP do servidor:
  - versão do protocolo & código de status
  - cabeçalhos da resposta
  - corpo da resposta

Códigos de status HTTP

2xx tudo ocorreu bem

3xx o recurso foi movido

4xx problema no acesso

5xx erro no servidor

COOKIES

- Obs: HTTP é *stateless*
- Problema antigo: como guiar um usuário “através” de uma sequência de páginas?
  - usar o IP para identificar o usuário? Ruim, usuários compartilham um mesmo IP em uma rede compartilhada
  - embutir informação do usuário na URI? Ruim, quebra os caches
- Rapidamente substituído por *cookies*; assista: `http://screencast.saasbook.info`
- Arcabouços como o Rails gerenciam a adulteração de cookies para você

USOS DE COOKIES

- A maioria dos sites perceberam que manter um estado para cada usuário poderia ser útil para muitas coisas:
  - personalização (“My Yahoo!”)
  - rastreamento de clicks/fluxo
  - autenticação (logado ou não)
  - Quais desses podem ser implementados no lado do usuário? Quais não devem ser e por quê?
- Regra de ouro: não confie no cliente! Deve ser possível identificar alterações nos cookies