UNIVERSIDADE DE ÉVORA

Sistemas Distribuídos

# *Web Services*

# Arquitetura REST

# Microsserviços

# Web Services

- camada de software para facilitar a interação entre cliente e servidor, tornando-a mais rica e mais estruturada.
- incluem uma API que permite aceder a serviços remotos através da rede
- independentemente da linguagem das aplicações cliente e servidor, os pedidos e respostas são *usualmente* codificados numa **R**epresentação **E**xterna de **D**ados (RED) em `XML` e transmitidos sobre HTTP
  - Outra possível RED: `JSON`
- são identificados por um URI (URL ou URN)

# Interface e formato das mensagens

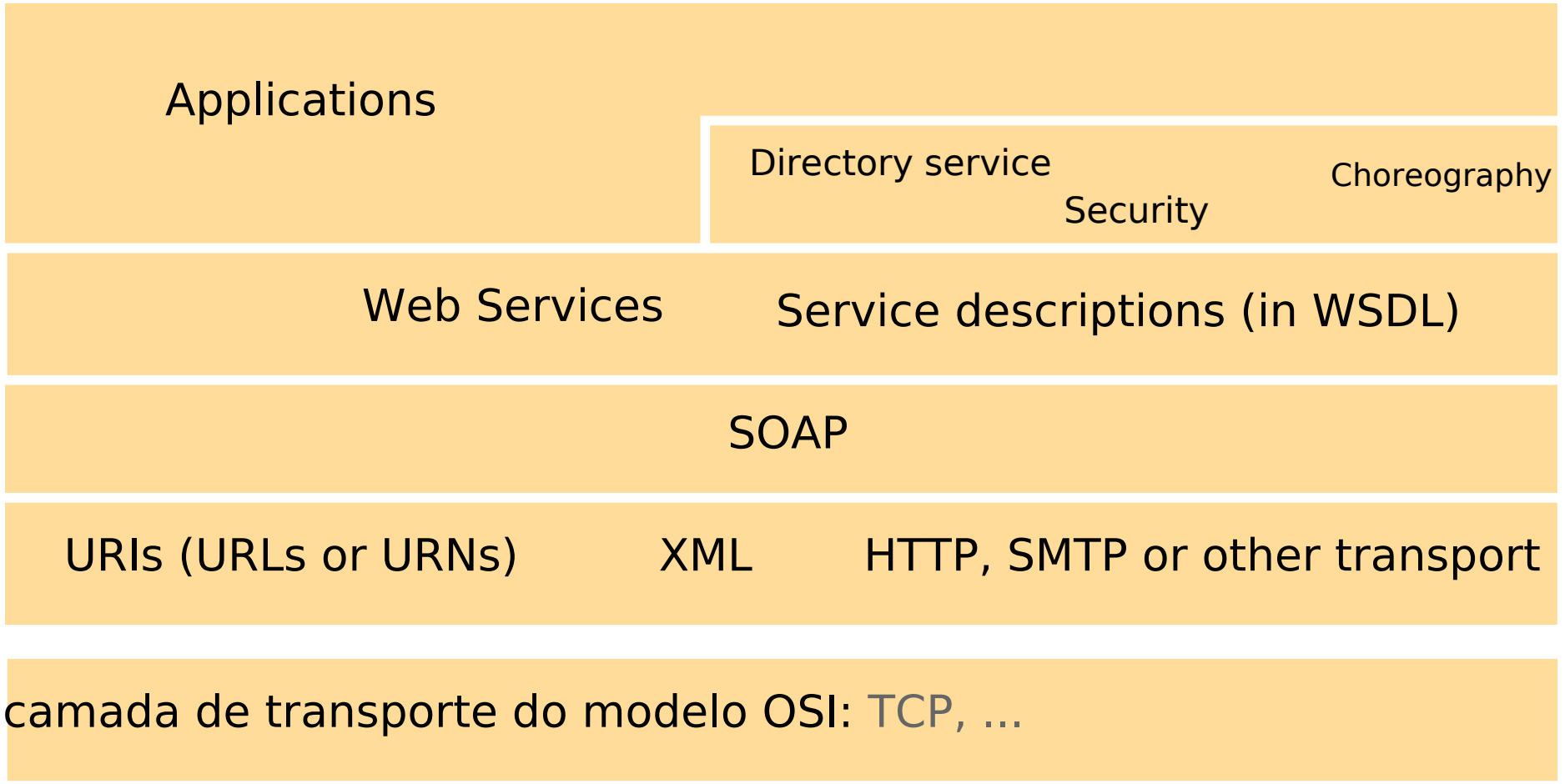
*Service Description*

- Tal como no CORBA e Java RMI, há uma descrição para a **interface** do web service. Há também a especificação do protocolo de codificação e comunicação das mensagens e a localização do Web Service (URL ou URN)
- A IDL usada é a *Web Services Description Language* (WSDL)

SOAP - Simple Object Access Protocol

- extensão de XML-RPC
- é um protocolo para troca de mensagens, usualmente sobre HTTP, que trata do correto encapsulamento dos dados em XML
- possíveis protocolos para envio das mensagens HTTP, SMTP, TCP, UDP
- não levanta problemas na presença de firewalls

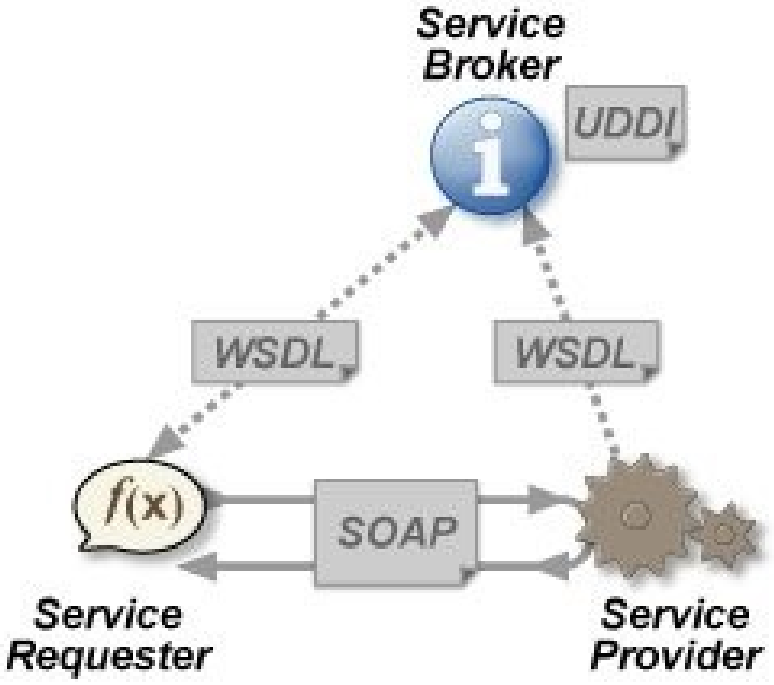
# Web Services: camada de software usada pelas aplicações



# Web Services: arquitetura

- tal como em Java RMI, o cliente do Web Service poderá consultar a descrição do serviço de nomes ou de diretoria

- **UDDI**:*Universal Description Discovery and Integration*
  - protocolo para publicar e pesquisar meta-informação sobre *web services*... permite que uma aplicação descubra e use um *web service* em tempo de execução



# Web Services

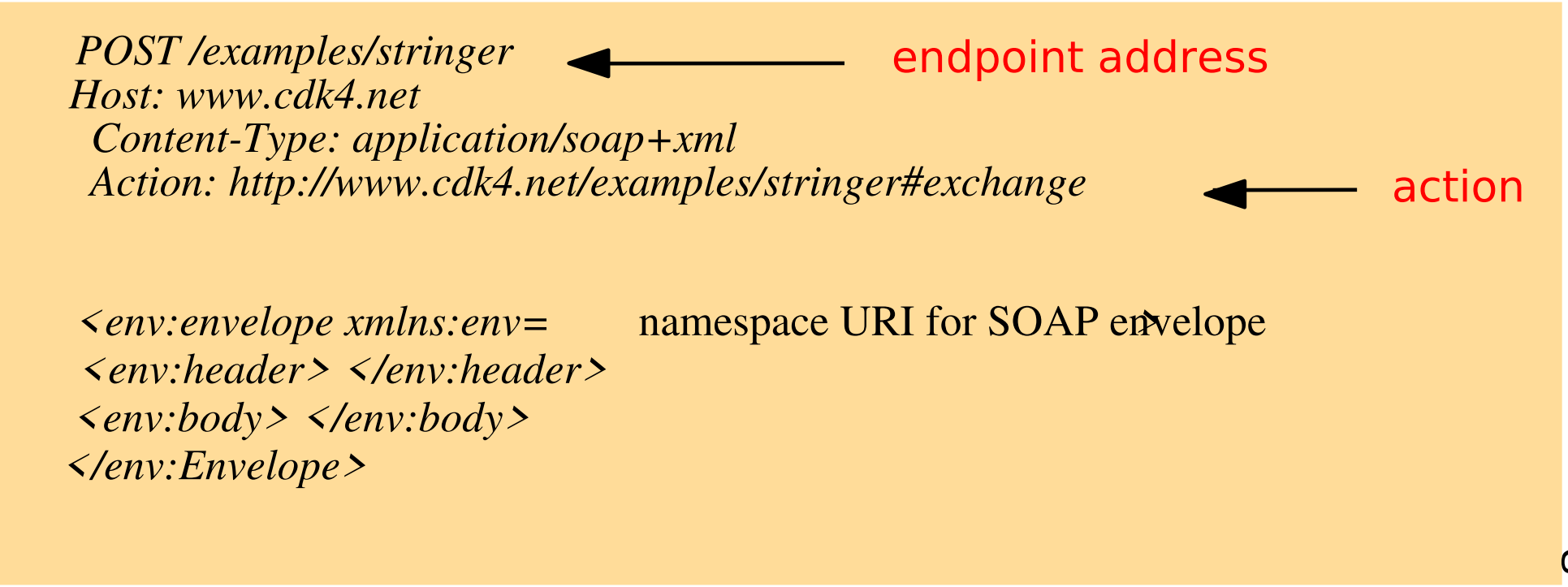
- servem de suporte à computação distribuída via internet, facilitando a cooperação de aplicações baseadas em diferentes linguagens

- os detalhes de SOAP e XML são usualmente escondidos por APIs (Java, Perl, Python, C++). A *service description* pode ser usada para gerar as rotinas de *marshalling* e *unmarshalling* de forma automática.

- diferença relativamente ao Object Model Distribuído: um *web service* é assegurado por um único objeto
  - o garbage collection neste caso não é tão crítico, porque não expõe objetos a referências remotas
  - a referência remota para o objeto não é relevante (ele é o único associado ao serviço)
  - Para lá do Web Service, podem existir muitos objetos, para funcionalidade de apoio...

- Uso de XML em SOAP e nos dados:
  - vantagem: mais legível, por humanos
  - desvantagem: processamento mais lento que formatos binários

# Figure 19.6 Use of HTTP POST Request in SOAP client-server communication



# SOAP em Java

- **JAX-WS** significa Java API for XML Web Services e inclui:
  - Java Architecture for XML Binding (JAXB)
  - SOAP with Attachments API for Java (SAAJ)

- A API do JAX-WS esconde a complexidade do protocolo SOAP
  - O programador não precisa gerar ou fazer o parse explícito de mensagens SOAP

- Web Service **Endpoint** é a implementação do Web Service
  - Fica  no servidor

# Web Services em Java: Servidor e descrição

Assumindo que se começa por desenvolver a *Service Interface* e a sua implementação (*endpoint*)

- há ferramentas para gerar automaticamente o Skeleton e a descrição do serviço em WSDL
- o serviço vai correr num Servlet Container (exemplo: Apache Tomcat)
- Depois de preparado, o web service deve estar num ficheiro ".war", que pode ser *deployed* (instalado) no *Servlet Container*
  - o dispatcher do *servlet container* identifica a operação no pedido
    - pelo header http Action
    - e invoca o método apropriado no respetivo Skeleton
- É possível gerar a classe do Proxy, para o cliente, em runtime, a partir da *service description*... mas não é sempre assim.

# Segurança no XML

funcionalidade para assinar ou para cifrar

- Para determinadas garantias de segurança, não basta proteger o canal... o próprio documento tem de incorporar alguma meta-informação (assinatura, informação de chaves...)
  - para ser validada posteriormente à cessação do canal de comunicação

- o XML permite várias formas sintáticas para os mesmos dados
  - a assinatura digital do XML é precedida de uma conversão para *Canonical XML*
    - *representação normalizada e serializada de XML*

# Alguns Web Services da Amazon

| *Web service* | *Description* |
|---|---|
| Amazon Elastic Compute Cloud (EC2) | Web-based service offering access to virtual machines of a given performance and storage capacity |
| Amazon Simple Storage Service (S3) | Web-based storage service for unstructured data |
| Amazon Simple DB | Web-based storage service for querying structured data |
| Amazon Simple Queue Service (SQS) | Hosted service supporting message queuing (as discussed in Chapter 6) |
| Amazon Elastic MapReduce | Web-based service for distributed computation using the MapReduce model (introduced in Chapter 21) |
| Amazon Flexible Payments Service (FPS) | Web-based service supporting electronic payments |

# GRID Computing

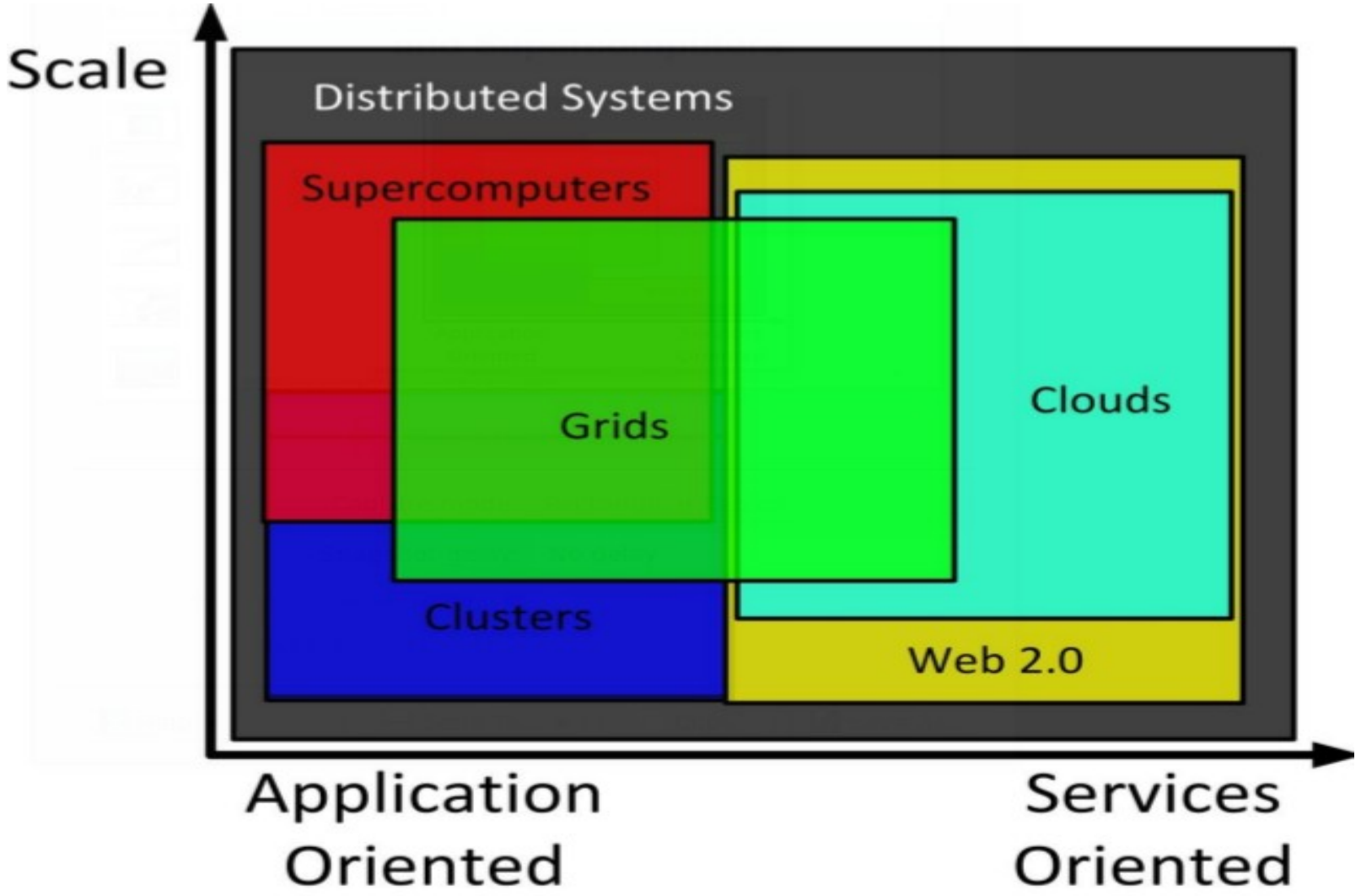
- *middleware* desenhado para permitir e optimizar a partilha de recursos (computadores, dados, sensores, software) em larga escala

- usualmente os utilizadores destes sistemas (cientistas, engenheiros) colaboram para alcançar um objetivo comum, como um estudo que requer o processamento de grandes quantidades de dados, por exemplo.

- os recursos estão alojados em computadores de diferentes plataformas, com ambiente heterogéneo

- o *middleware* **GRID pode** assentar em **Web Services**

# Grid & Cloud


Scale
Distributed Systems
Supercomputers
Grids
Clouds
Clusters
Web 2.0
Application Oriented
Services Oriented

# Grid & Cloud

## Grid

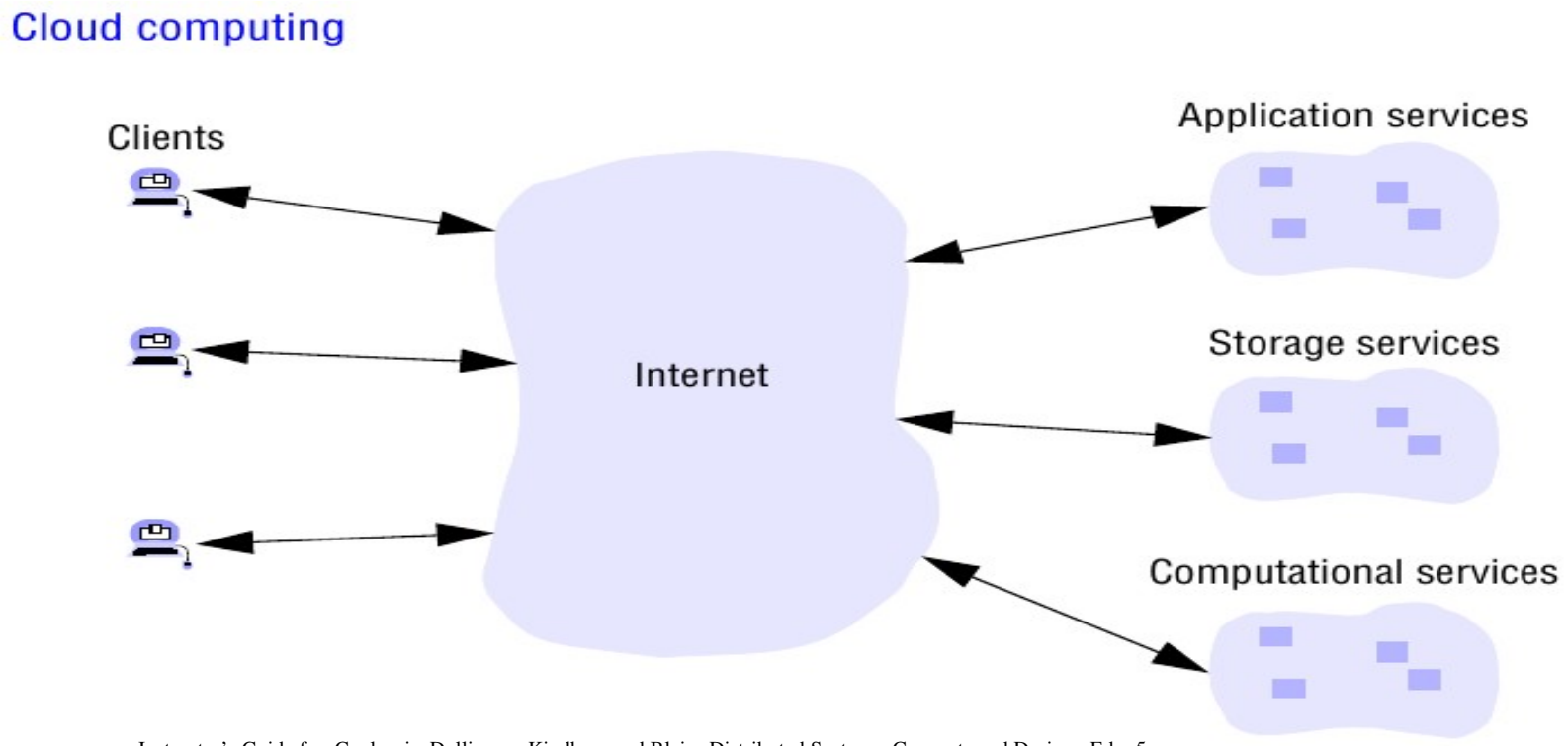
- Usualmente heterogéneas (mas não necessariamente)
- Lidar com enorme volume de dados ou processamentos complexos
- Geograficamente dispersas
- Exemplos
  - Open Science Grid (OSG) - http://www.opensciencegrid.org/
  - LHC Computing Grid (CERN)
  - Middleware: Globus Toolkit

## Cloud

- Paradigma “recente” de computação distribuída de larga escala
  - Motivações:
    - Uso mais geral (muitos ou poucos dados); maior leque de utilizadores
    - vocação para os serviços
    - **Virtualização**
    - Alocação de recursos dinâmica em função das necessidades
    - Exemplos:
      - Amazon, GoogleApps, Windows Azure

# Cloud Computing

*Computação em Nuvem* permite armazenamento e/ou computação baseado na Internet. Reduz a necessidade de armazenamento, software e capacidade de processamento do lado dos equipamentos terminais (junto do utilizador)



# Arquitetura REST

- REST - *Representational State Transfer*
  - É uma arquitetura para interação em sistemas distribuídos
  - Surgiu em 2000, da tese de doutoramento de Roy Thomas Fielding

- Objetos têm estado, cuja representação é transportada por pedidos HTTP
  - PUT; POST; GET; DELETE
  - Cada pedido tem um significado próprio (criar, alterar, consultar e apagar objetos)

- Em comparação com os SOAP based Web Services:
  - Mais leve
  - Ainda intelegível
  - Formato dos dados pode ser JSON ou XML (mantém flexibilidade)

# Arquitetura REST

- Analogia com SOAP:
  - mensagem de consulta de dados sobre o utilizador nº 12345
  - SOAP WS:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:body pb="http://www.acme.com/phonebook">
  <pb:GetUserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
 </soap:Body>
</soap:Envelope>
```

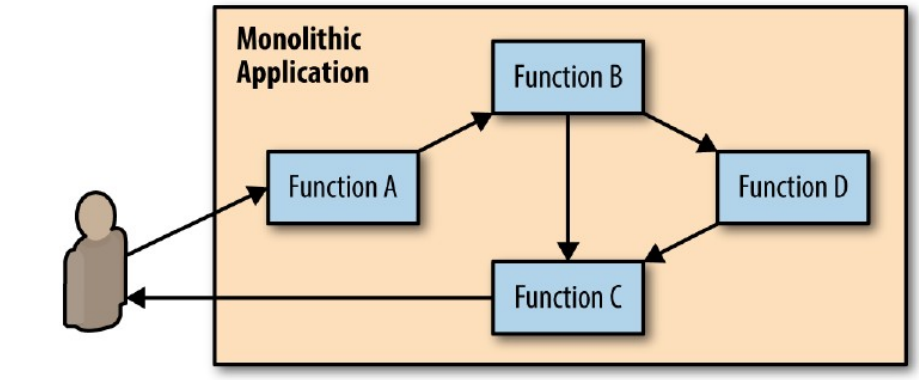
  - REST:

```
http://www.acme.com/phonebook/UserDetails/12345
```

    - O url inclui os parâmetros da consulta

# Microsserviços

- Uma abordagem de desenvolvimento de software que separa as tarefas de um processo complexo num conjunto de serviços e processos autónomos, que comunicam através de APIs, cooperando para a realização de um serviço composto.




*A monolithic service with all functions in a single container*

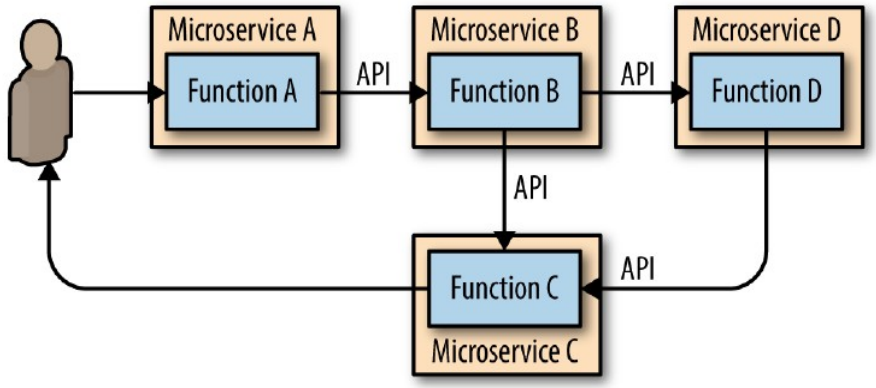



*Figure II-2. A microservice architecture with each function broken out as a separate microservice*

Imagens: Brendan Burns (2018). Designing Distributed Systems

# Microsserviços

Em comparação com sistemas monolíticos, os Microsserviços

- maior agilidade
- facilitam manutenção do software
- promover a modularização e conformidade com normas e APIs
- facilitam alterações à arquitetura, permitindo escalar de modo mais eficaz, com o reforço aplicado de forma independente em cada componente/serviço

Imagens: Brendan Burns (2018). Designing Distributed Systems