

ACH 2147 — DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO DISTRIBUÍDOS

ARQUITETURAS

Daniel Cordeiro

14 e 16 de março de 2018

Escola de Artes, Ciências e Humanidades | EACH | USP

- Estilos arquiteturais
- Arquiteturas de software
- Arquiteturas versus middleware
- Sistemas distribuídos autogerenciáveis

Ideia básica

Um estilo é definido em termos de:

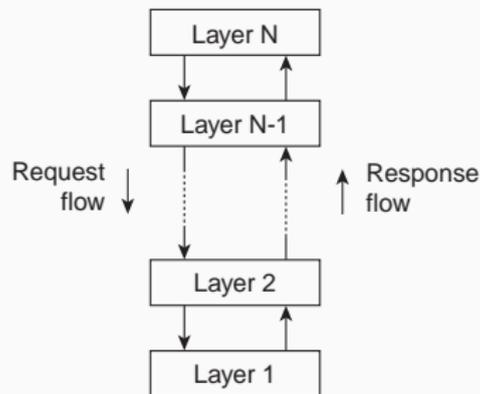
- componentes (substituíveis) com interfaces bem definidas
- o modo como os componentes são conectados entre si
- como os dados são trocados entre componentes
- como esses componentes e conectores são configurados conjuntamente em um sistema

Conector

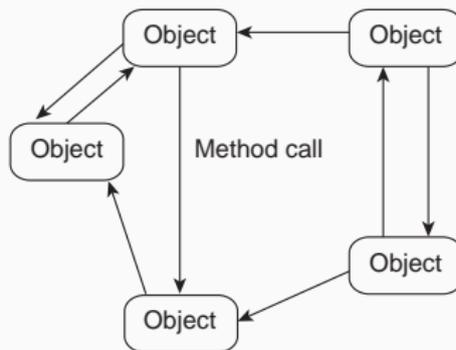
Um mecanismo que intermedeia comunicação, coordenação ou cooperação entre componentes. Exemplo: recursos para chamadas de procedimento (remotos), mensagens ou *streaming*.

Ideia básica

Organize em componentes **logicamente diferentes** e os distribua entre as máquinas disponíveis.



(a)



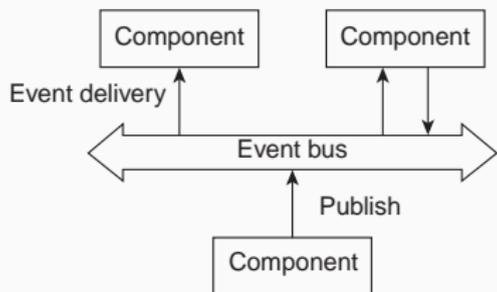
(b)

(a) Estilo em camadas é usado em sistemas cliente-servidor

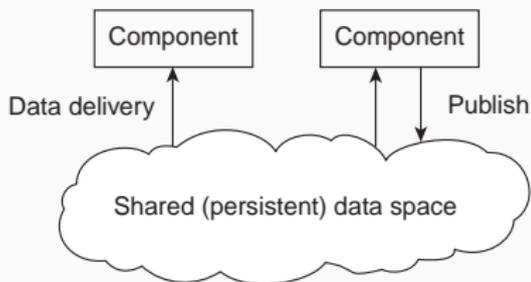
(b) Estilo orientado a objetos usado em sistemas de objetos distribuídos.

Observação

Desacoplar processos no **espaço** (anônimos) e **tempo** (assíncronos) pode levar a estilos diferentes.



(a)



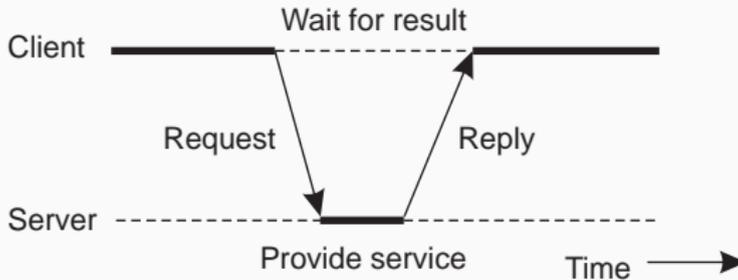
(b)

(a) Publish/subscribe [desacoplado no **espaço**]

(b) Espaço de dados compartilhados [desacoplado no **espaço** e **tempo**]

Características do modelo Cliente-Servidor

- Existem processos que oferecem serviços (**servidores**)
- Existem processos que usam esses serviços (**clientes**)
- Clientes e servidores podem estar em máquinas diferentes
- Clientes seguem um modelo requisição/resposta ao usar os serviços



Visão tradicional em três camadas

- A **camada de apresentação** contém o necessário para a aplicação poder interagir com o usuário
- A **camada de negócio** contém as funções de uma aplicação
- A **camada de dados** contém os dados que o cliente quer manipular através dos componentes da aplicação

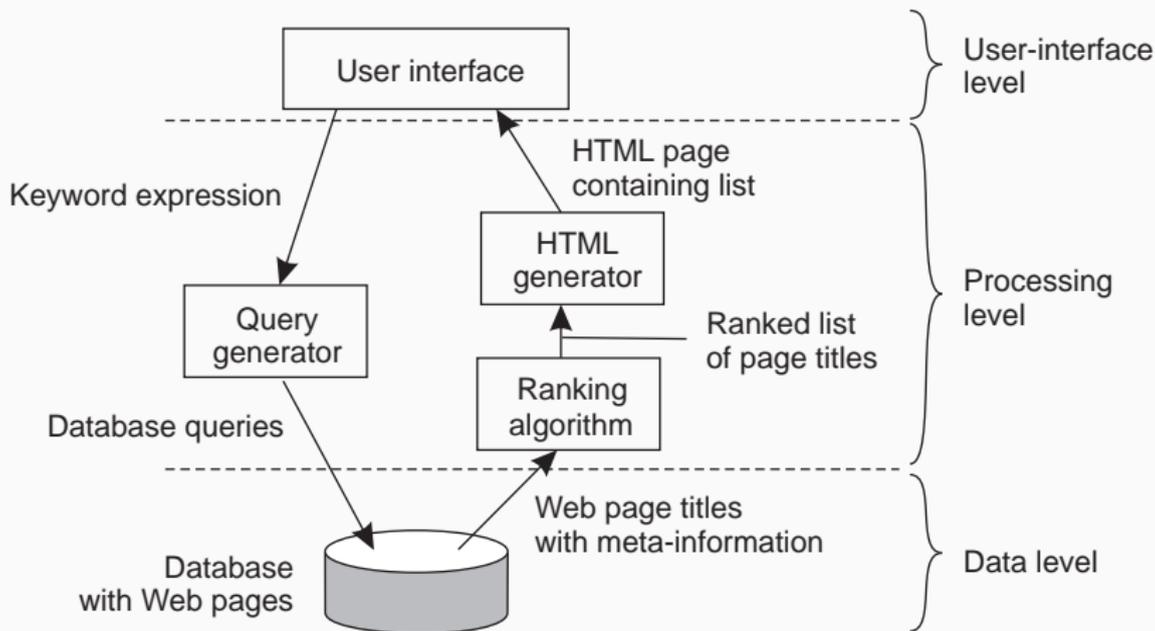
Visão tradicional em três camadas

- A **camada de apresentação** contém o necessário para a aplicação poder interagir com o usuário
- A **camada de negócio** contém as funções de uma aplicação
- A **camada de dados** contém os dados que o cliente quer manipular através dos componentes da aplicação

Observação

Estas camadas são encontradas em muitos sistemas de informação distribuídos, que usam tecnologias de bancos de dados tradicionais e suas aplicações auxiliares.

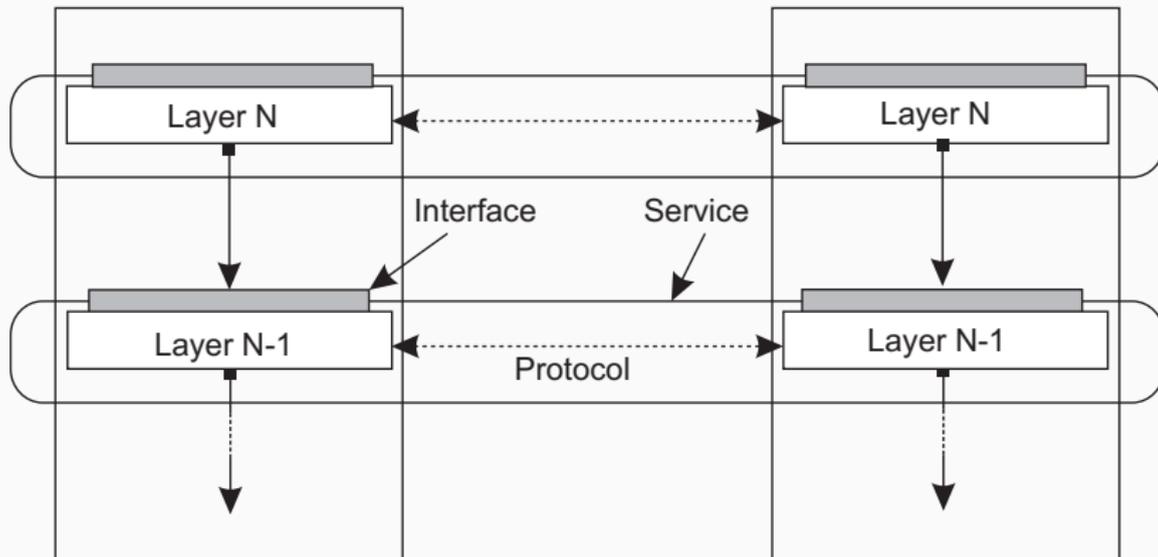
ARQUITETURAS MULTICAMADA



EXEMPLO: PROTOCOLOS DE COMUNICAÇÃO

Protocolo, serviço, interface
Party A

Party B



Servidor

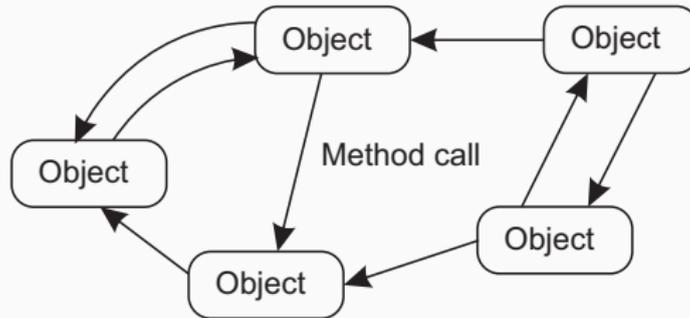
```
from socket import *
s = socket(AF_INET, SOCK_STREAM)
s.bind((HOST, PORT))
s.listen()
(conn, addr) = s.accept() # returns new socket and addr. client
while True:                # forever
    data = conn.recv(1024) # receive data from client
    if not data: break     # stop if client stopped
    conn.send(str(data)+"*") # return sent data plus an "*"
conn.close()               # close the connection
```

Cliente

```
from socket import *
s = socket(AF_INET, SOCK_STREAM)
s.connect((HOST, PORT)) # connect to server (block until accepted)
s.send('Hello, world') # send some data
data = s.recv(1024)    # receive the response
print data              # print the result
s.close()               # close the connection
```

Essência

Componentes são objetos, conectados entre si usando chamadas de procedimentos. Objetos podem ser colocados em máquinas diferentes; chamadas, por tanto, devem executar usando a rede.



Encapsulamento

Dizemos que um objeto *encapsula dados* e oferece *métodos para os dados* sem revelar sua implementação.

Vê um sistema distribuído como uma coleção de recursos que são gerenciados individualmente por componentes. Recursos podem ser adicionados, removidos, recuperados e modificados por aplicações (remotas).

1. Recursos são identificados usando um único esquema de nomeação
2. Todos os serviços oferecem a mesma interface
3. Mensagens enviadas de ou para um serviço são auto-descritivas
4. Após a execução de uma operação em um serviço, o componente esquece tudo sobre quem chamou a operação

Operações básicas

Operação	Descrição (veja https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods)
GET	Recupera o estado de um recurso usando um tipo de representação
POST	Cria um novo, ou envia uma entidade, a um dado recurso
DELETE	Apaga um recurso
PUT	Substitui um recurso ao transferir um novo estado

Essência

Objetos (arquivos) são armazenados em **buckets** (diretórios). Buckets não podem ser colocados dentro de outros buckets. Operações em **ObjectName** em **BucketName** requerem o seguinte identificador:

```
http://BucketName.s3.amazonaws.com/ObjectName
```

Operações típicas

Todas as operações são realizadas com requisições HTTP:

- Criar um bucket/objeto: **PUT** + URI
- Listar objetos: **GET** em um nome de bucket
- Ler um objeto: **GET** em uma URI completa

ARQUITETURAS MULTICAMADA

uma camada: configurações de terminal burro/mainframe

duas camadas: configuração cliente–servidor único.

três camadas: cada camada em uma máquina separada

Configurações tradicionais em duas camadas:

