# Programación Distribuida y Tiempo Real

## Estilos arquitecturales: Capas

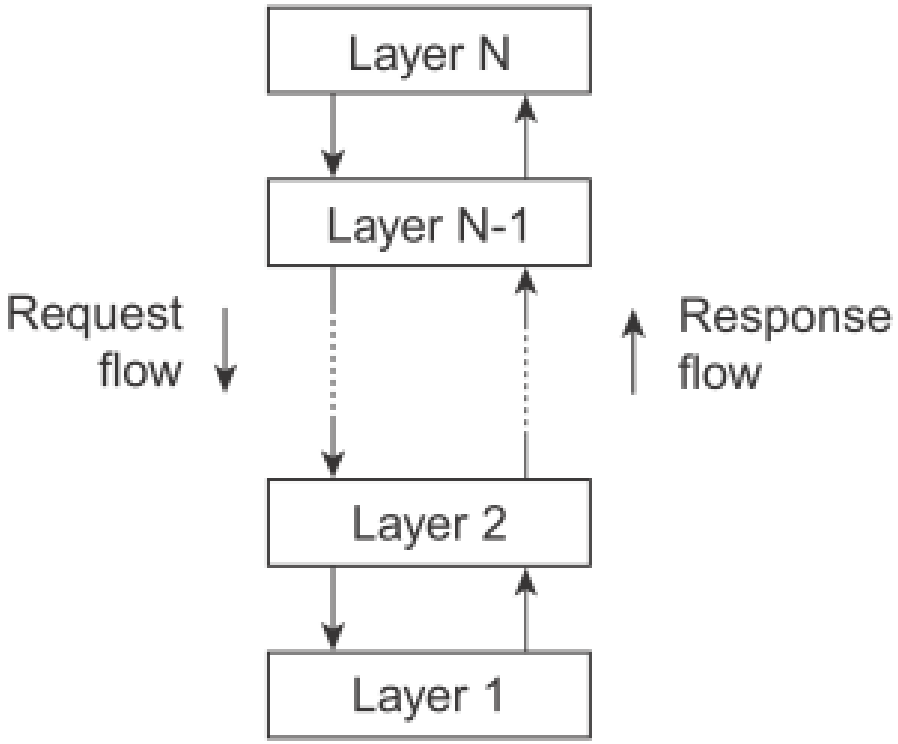
# Estilos Arquitecturales

- Los más importantes para Sistemas Distribuidos
  - Layered architectures
  - Object-based architectures
  - Resource-centered architectures
  - Event-based architectures

- Cada uno de ellos afectará
  - La programación
  - Los aspectos de tiempo real involucrados (Si los hubiera)…

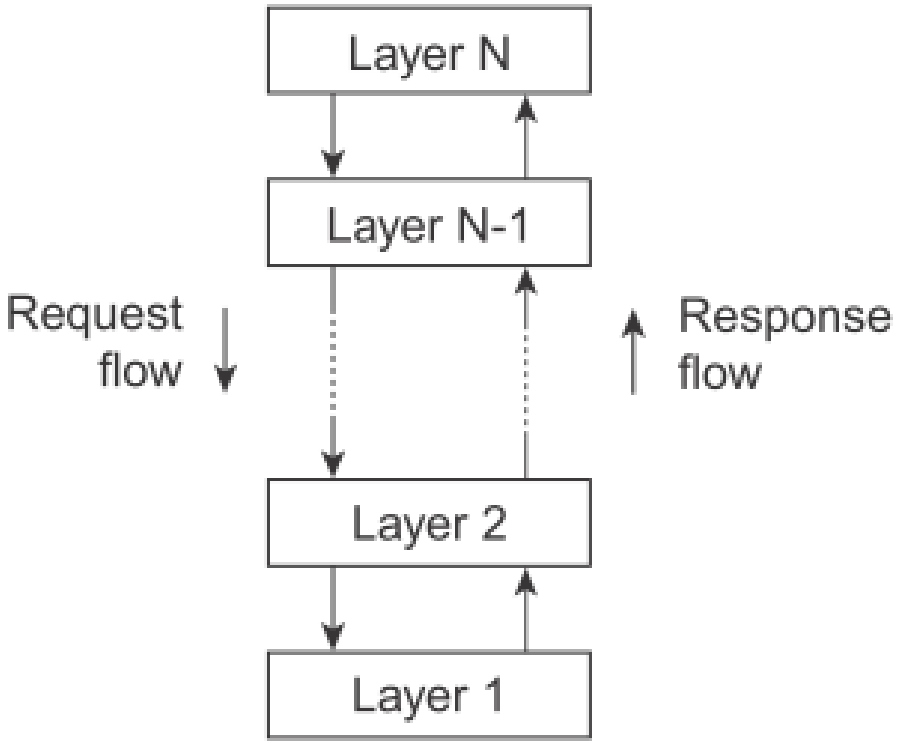
# Estilos Arquitecturales

- Layered architectures (capas)



# Estilos Arquitecturales

- Layered architectures (capas)




Componentes: capas

Interfaces: c/s

Organización: c/s, quizás "niveles de abstracción"

Comunicaciones:

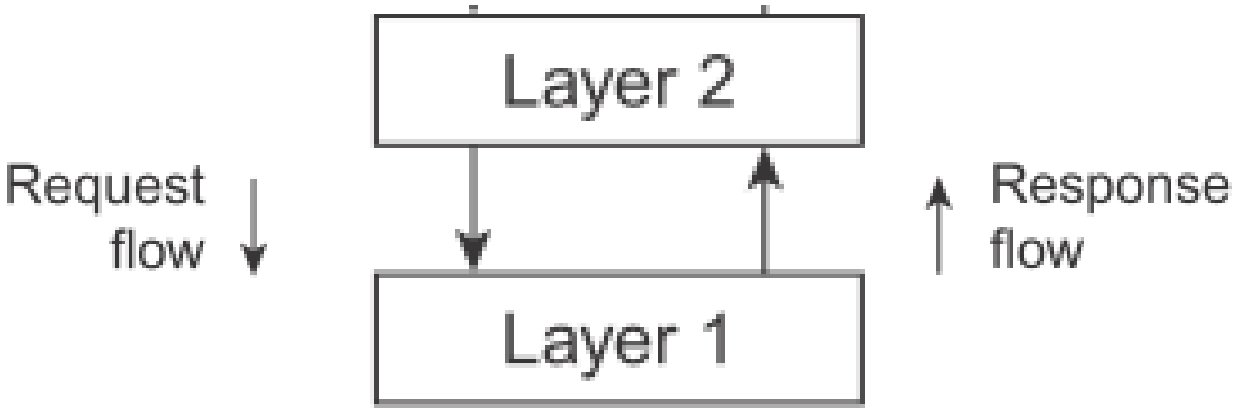
- requerimientos
- respuestas

# Capas: 2

- Dos capas: c/s
  - Una capa es solo cliente
  - Una capa es solo servidor

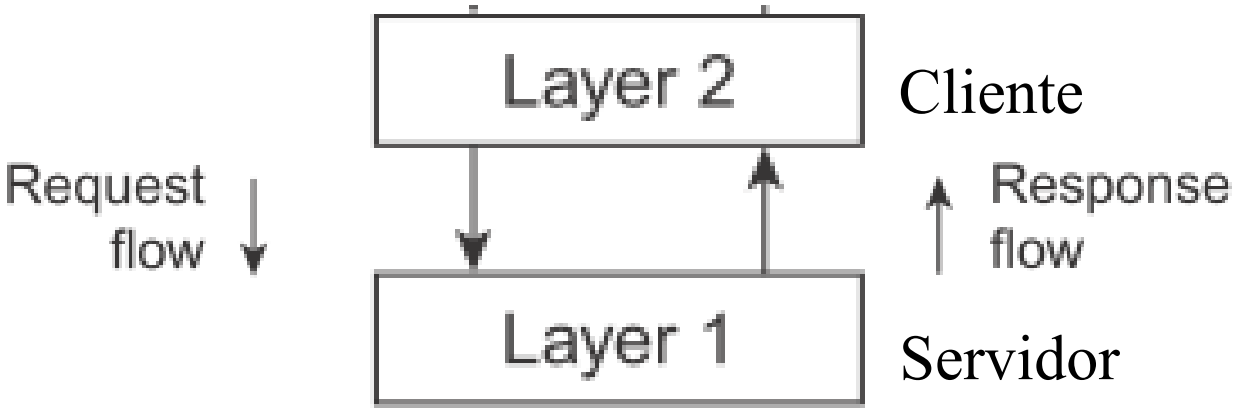
# Capas: 2

- Dos capas: c/s
  - Una capa es solo cliente
  - Una capa es solo servidor



# Capas: 2

- Dos capas: c/s
  - Una capa es solo cliente
  - Una capa es solo servidor

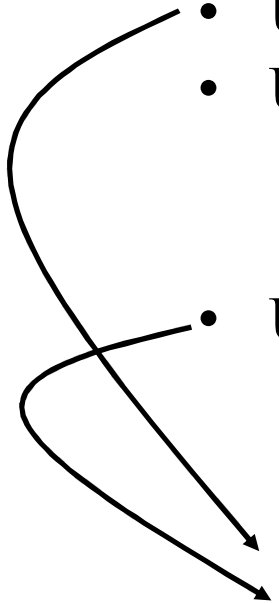


# Capas: 3

- Tres capas: usr/procesamiento/datos
  - Una capa es solo cliente: interfaz de usuario
  - Una capa “intermedia”
    - Servidor de la interfaz de usuario
    - Cliente de la capa de base de datos (persistente)
  - Una capa es solo servidor: base de datos

# Capas: 3

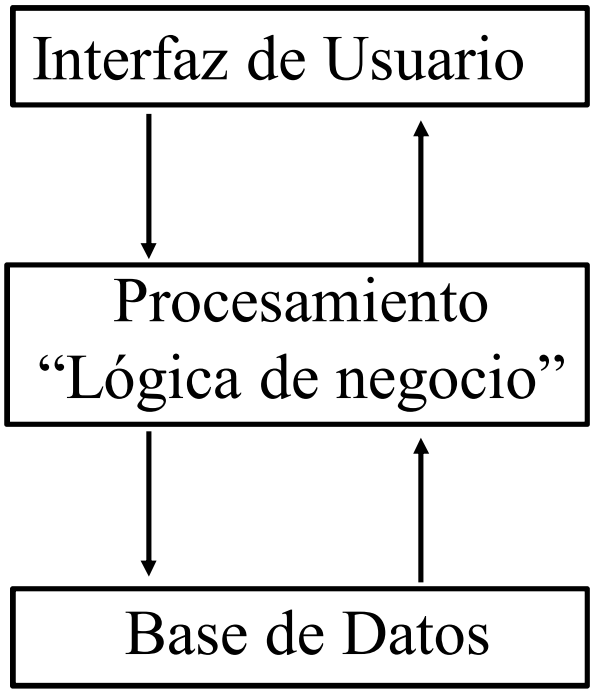
- Tres capas: usr/procesamiento/datos
  - Una capa es solo cliente: interfaz de usuario
  - Una capa “intermedia”
    - Servidor de la interfaz de usuario
    - Cliente de la capa de base de datos (persistente)
  - Una capa es solo servidor: base de datos



Relativamente “sencillas” de definir

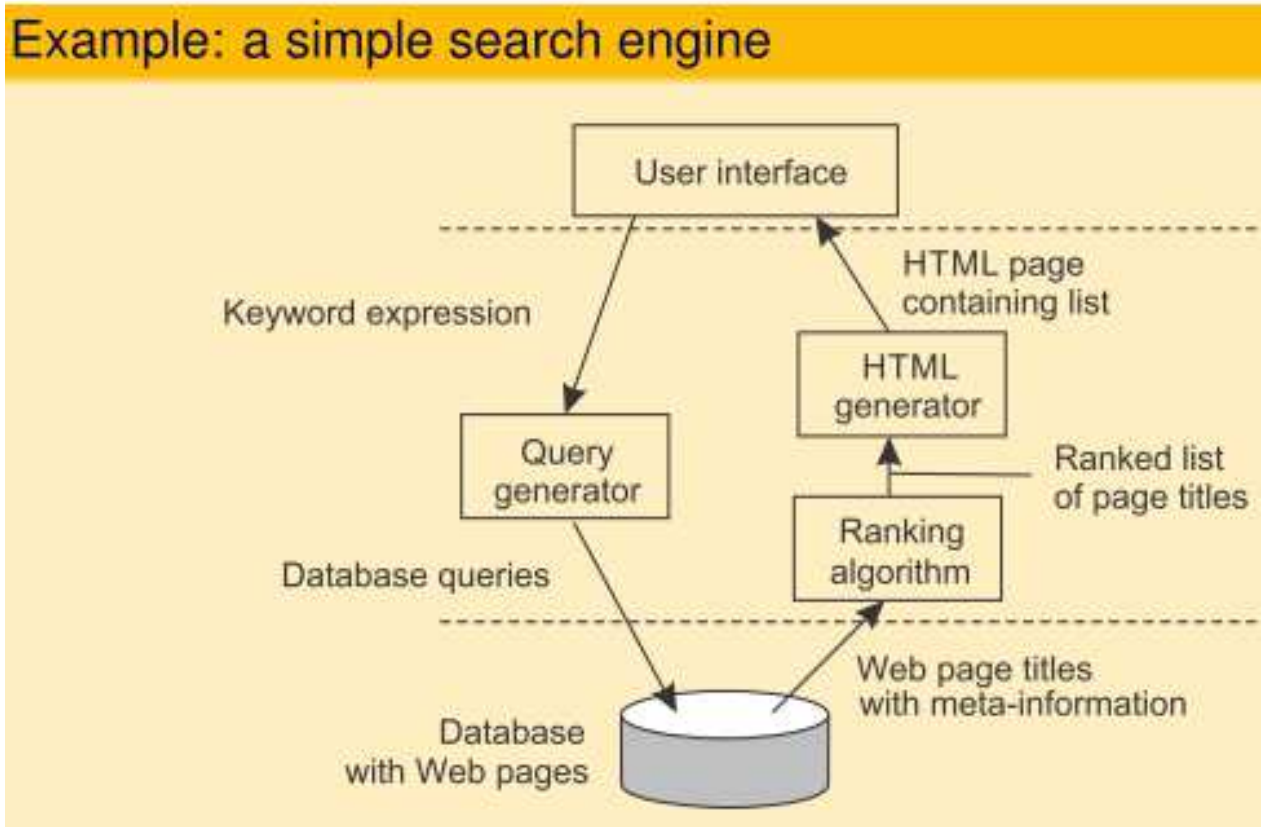
# Capas: 3

- Tres capas: usr/procesamiento/datos



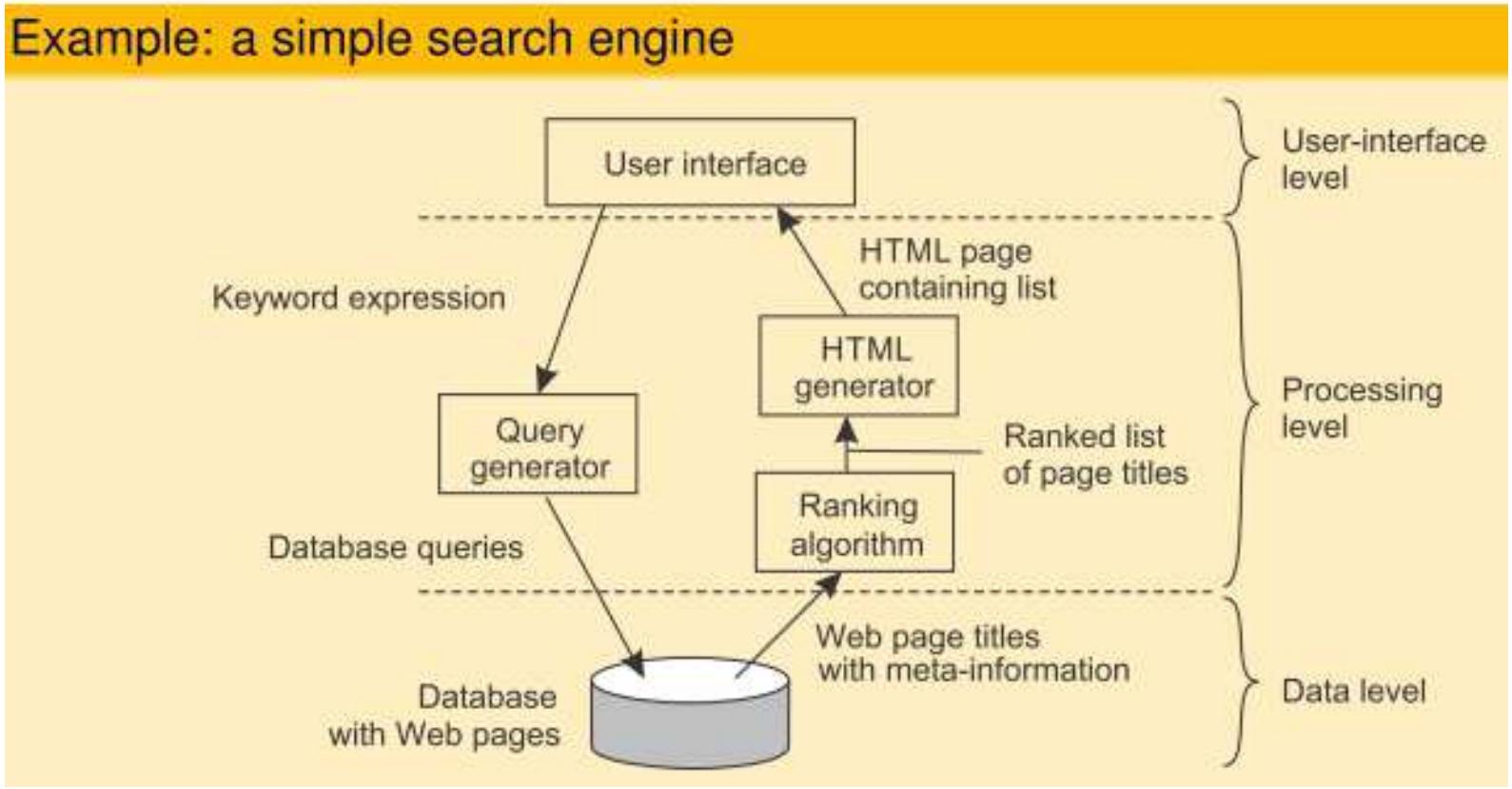
# Capas: 3

- Tres capas: usr/procesamiento/datos



# Capas: 3

- Tres capas: usr/procesamiento/datos



# Capas: 3

- La “confusión” de

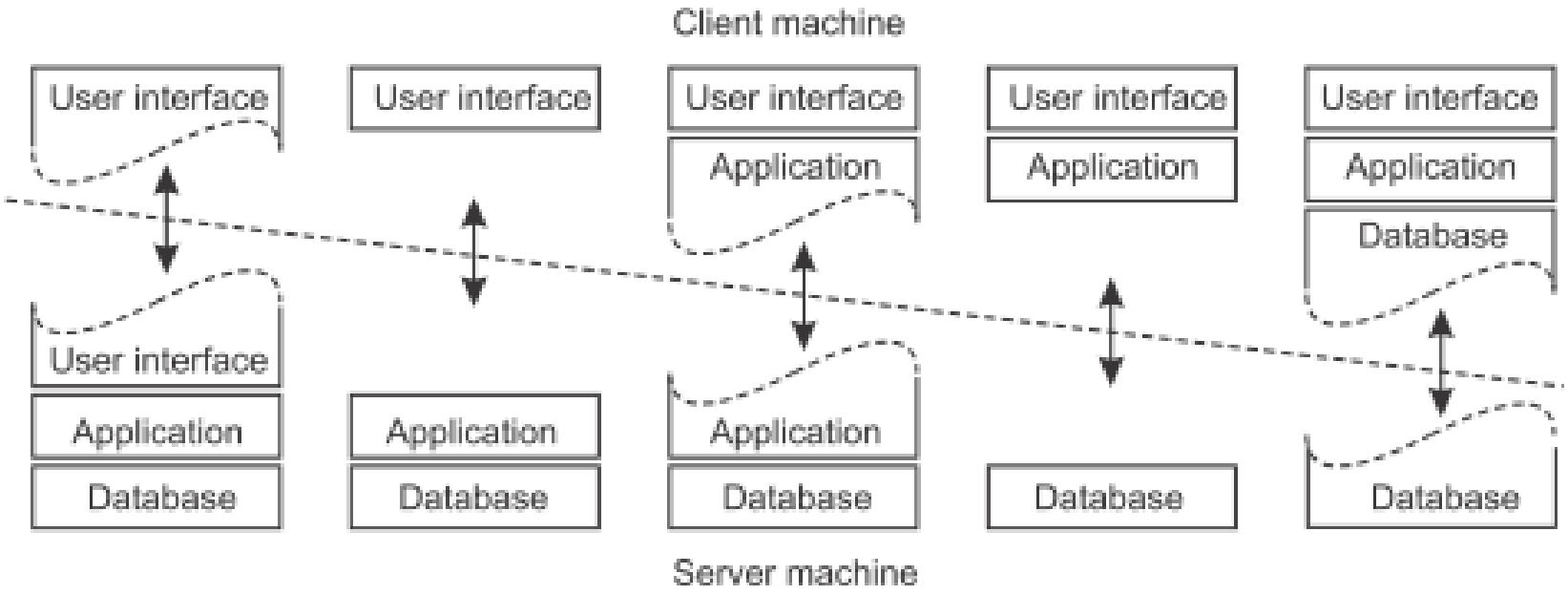
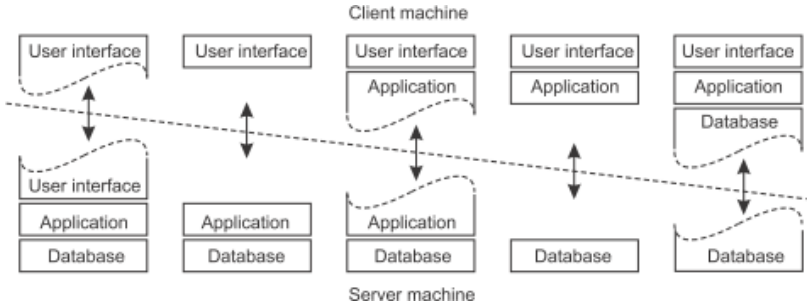



**Figure 2.16:** Client-server organizations in a two-tiered architecture.

# Capas: 3

- La “confusión” de



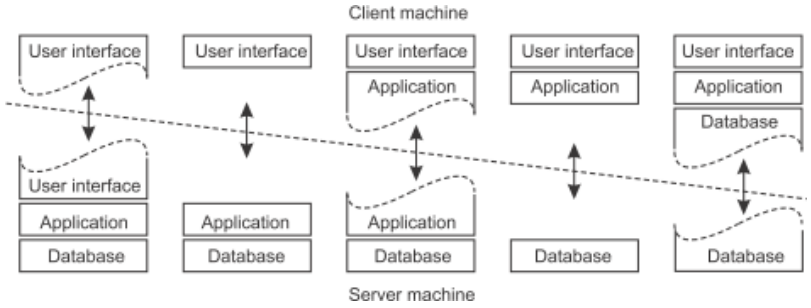

I think that Fig. 2.16 involves some conceptual problems:
1) I think that 3-layer architectural style system could be divided into 2 physical computers, but it does not make a great deal of difference if they are in 1, 2, or 3 computers from the point of view of programming, which is (al/the) most complex task of the entire system, I think. From this point of view, options b) and d) in Fig. 2.16 could be decided depending on (computing and/or communication) performance issues.
2) I think that Fig. 2.16 options c) and e) are rather far from 3 layers, because each "sublayer" in the application layer for option c) and data layer in option e) is a layer in itself because given that each sublayer is in different computers there must be a well defined communication protocol, then it involves a work at least similar to that needed for a 4 layer architectural software system.

Thus, taking into account 1) and 2) above I think fig. 2.16 is confusing.

But maybe I'm losing something...

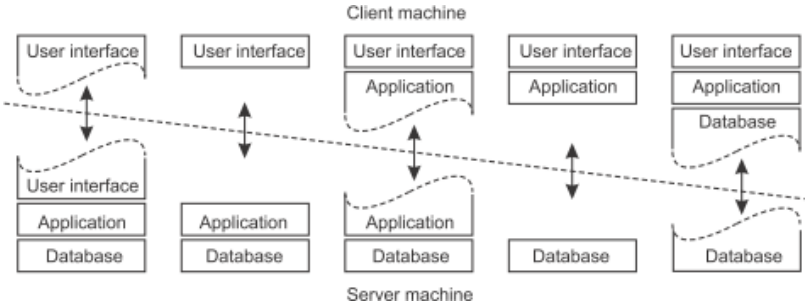
# Capas: 3

- La “confusión” de




Thanks.
Andy Tanenbaum

# Capas: 3




- La “confusión” de

Thanks.
Andy Tanenbaum

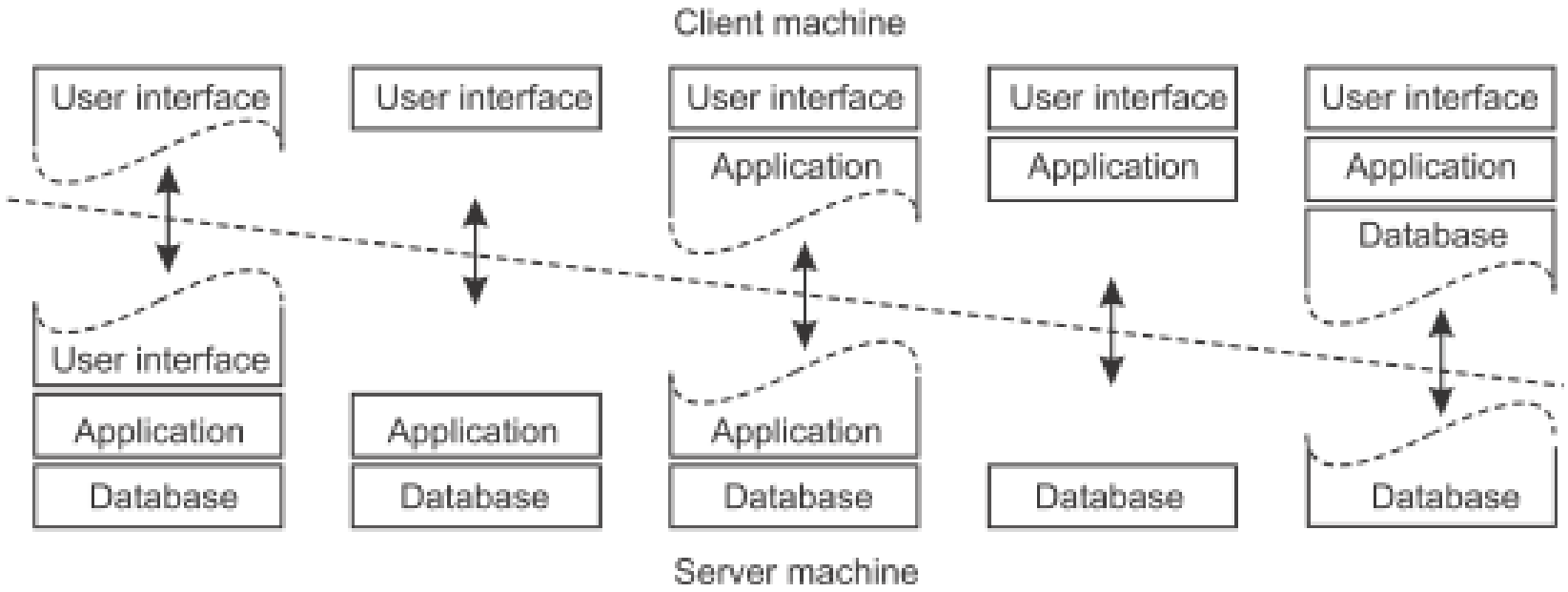
The matter to me is not whether programming is the most important/difficult aspect. What matters is the logicl divide into 3 layers, which I believe is still correct. To what extent the layers are actually separated is another. In many cases, you'll notice that they are intertwined. However, if you would think in terms of procedure calls and/or modules, you'll notice that the distinction is often explicit. Then, taking into account that modules may actually be placed on different machines without you, as a programmer, even noticing it (think of Java RMI), then suddenly Figure 2-16 starts making more sense perhaps.

All the best,

Maarten

# Capas: 3

- El “aporte”: *gordo* <===> *flaco*




==> Mayor distribución de la carga entre clientes
==> Mayor capacidad de procesamiento de los clients
==> Mejores tiempos de respuesta/interacción con el usuario

# Capas: ¿Muchas?

- Aumentar capas:
  - "Subdividir" la capa intermedia
  - Más capas, más modularización
    - Desacoplar/simplificar desarrollo, debug, etc.
  - Más capas, más interfaces
    - Mayor cantidad de datos a transferir
    - Mayor necesidad de interacción/sincronización
  - En el extremo:
    - Muchísimas intefaces
    - Solo comunicación y sincronización
  - No se conoce mucho en producción > 3 capas

# Dudas/Consultas

- Plataforma Ideas