# Aplicaciones Web

## Introducción

David Cabrero Souto

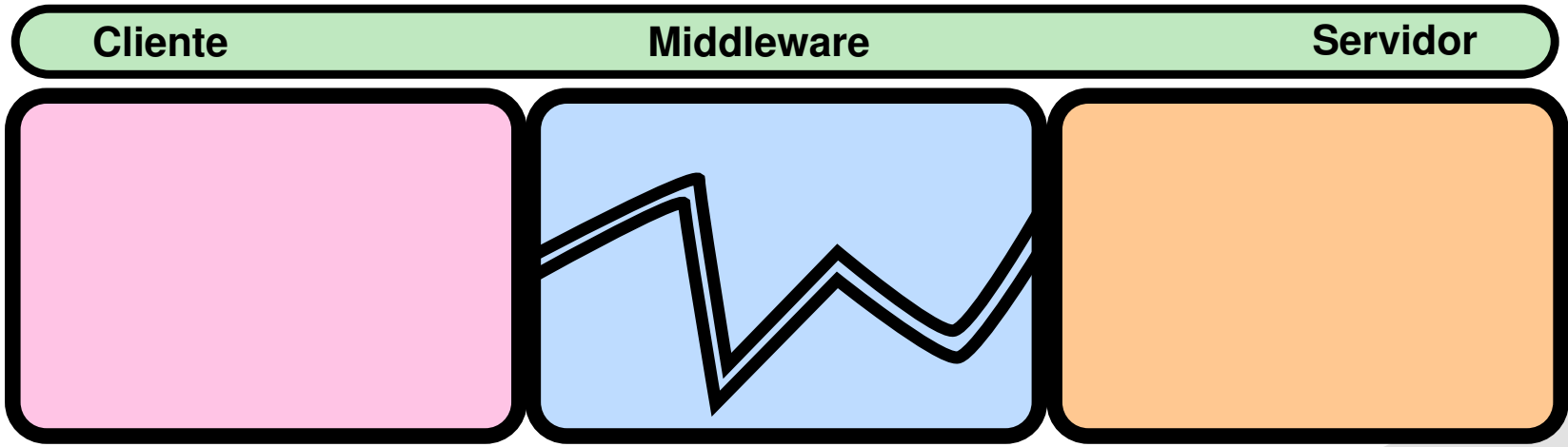
Grupo MADS (http://www.grupomads.org/)
Universidade da Coruña

# Ingredientes principales

- Arquitectura Cliente/Servidor
- Protocolos y estándares Web
  - HTTP, W3C

# Cliente/Servidor: Introducción

- Tres componentes:
  - Cliente
  - Servidor
  - Middleware (límites difusos)
    - Comunicaciones (síncrono o asíncrono)
    - Integración
    - Transparencia (topología, representación datos, . . . )
    - Seguridad




- Cardinalidad: 1 servidor, n clientes
- Varios tipos y clasificaciones

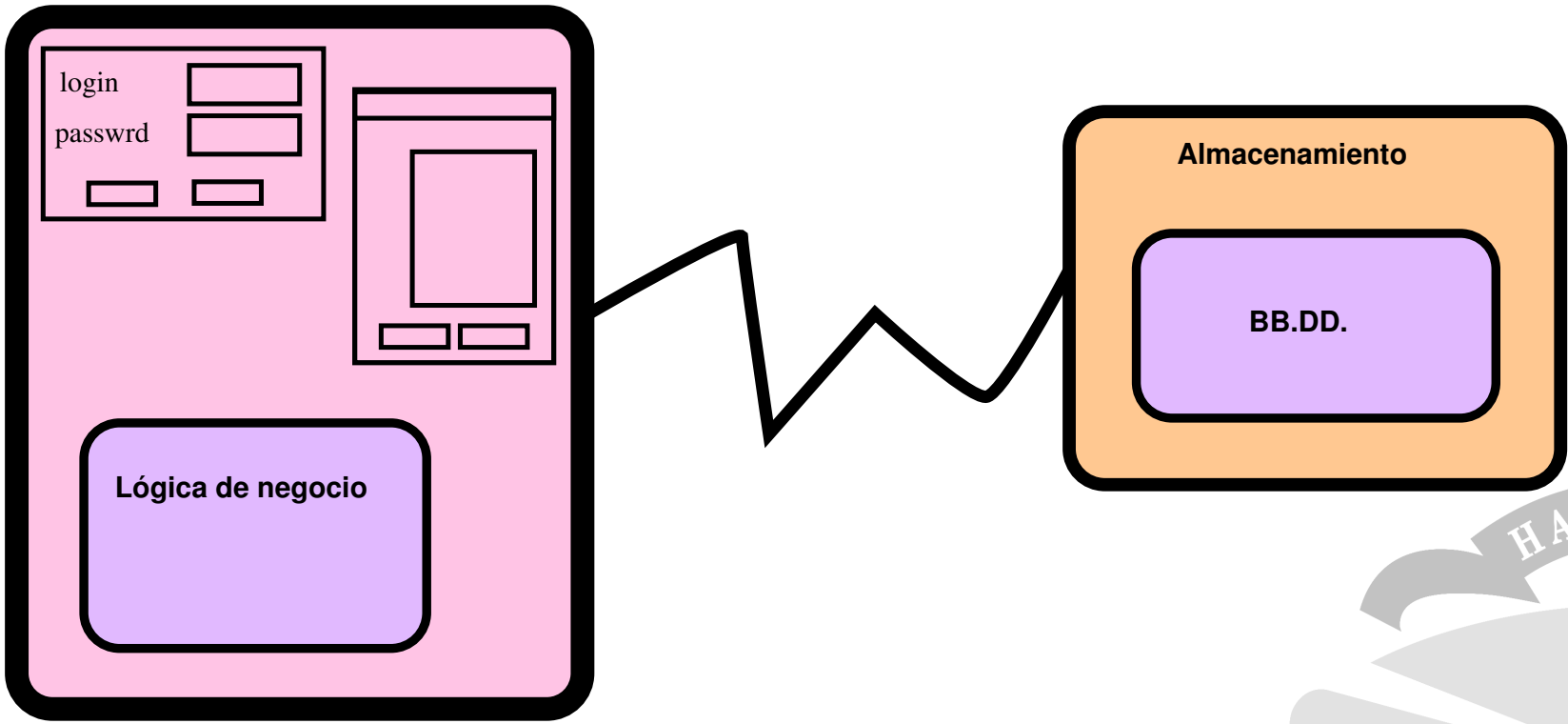
# C/S: clasificación (cliente grueso vs. delgado)

- Cliente grueso vs. Cliente delgado

# C/S: clasificación (cliente grueso vs. delgado)

- Cliente grueso vs. Cliente delgado

  Ejemplo de cliente grueso:



# C/S: clasificación (cliente grueso vs. delgado)

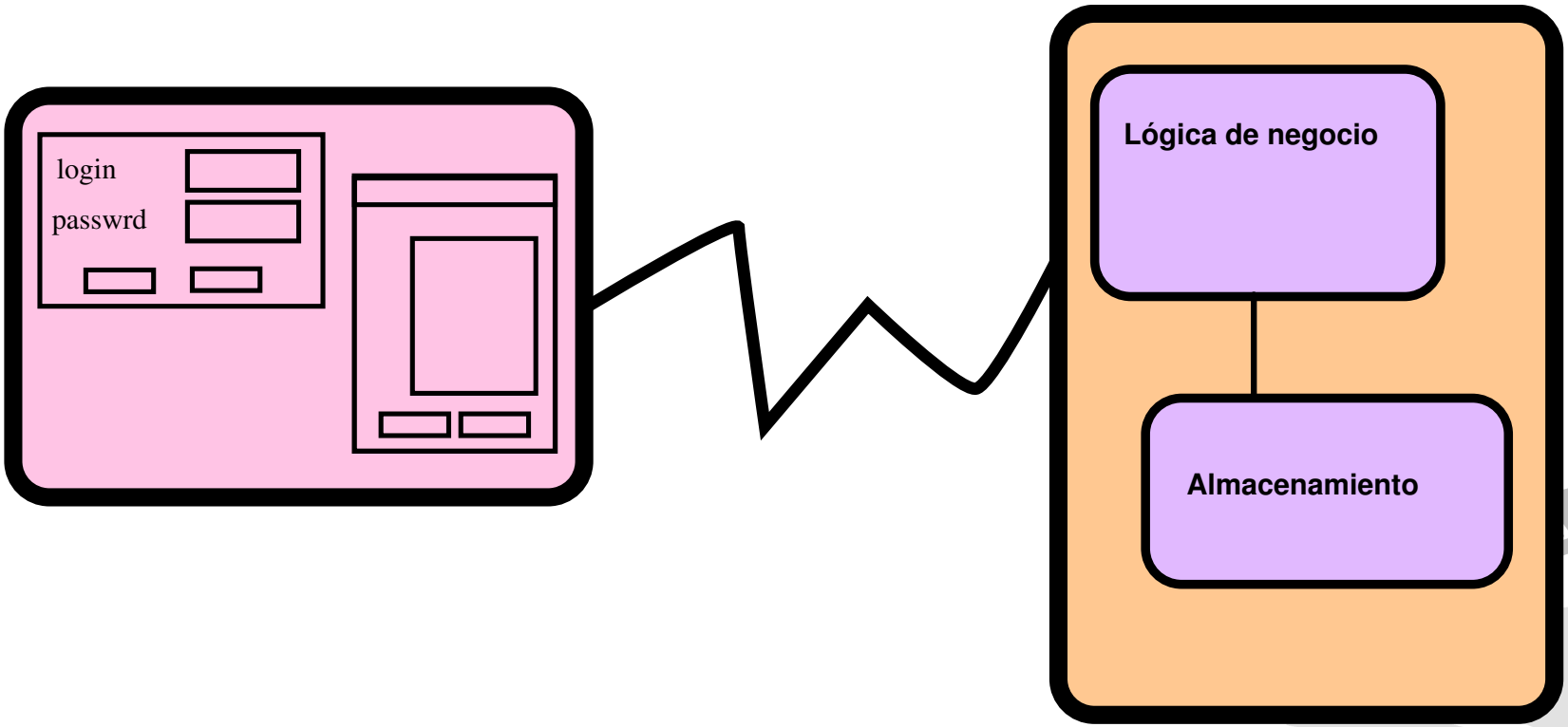
- Cliente grueso vs. Cliente delgado

  Cliente grueso. Ventajas e inconvenientes:
  - Menos carga en el servidor
  - Mayores requisitos hardware en cliente
  - Cliente menos portable
  - Más tráfico de red
  - Administración más compleja

# C/S: clasificación (cliente grueso vs. delgado)

- Cliente grueso vs. Cliente delgado

  Ejemplo de cliente delgado:



# C/S: clasificación (cliente grueso vs. delgado)

- Cliente grueso vs. Cliente delgado

  Cliente delgado. Ventajas e inconvenientes:

  - Servidor muy cargado
  - Mayores requisitos hardware en servidor (escalabilidad)
  - Cliente más portable
  - Menos tráfico de red
  - Administración menos compleja

# C/S: clasificación (2,3,n-tier)

- 2 niveles (2-tier).

  *Ejemplos: Los vistos en cliente grueso/delgado.*

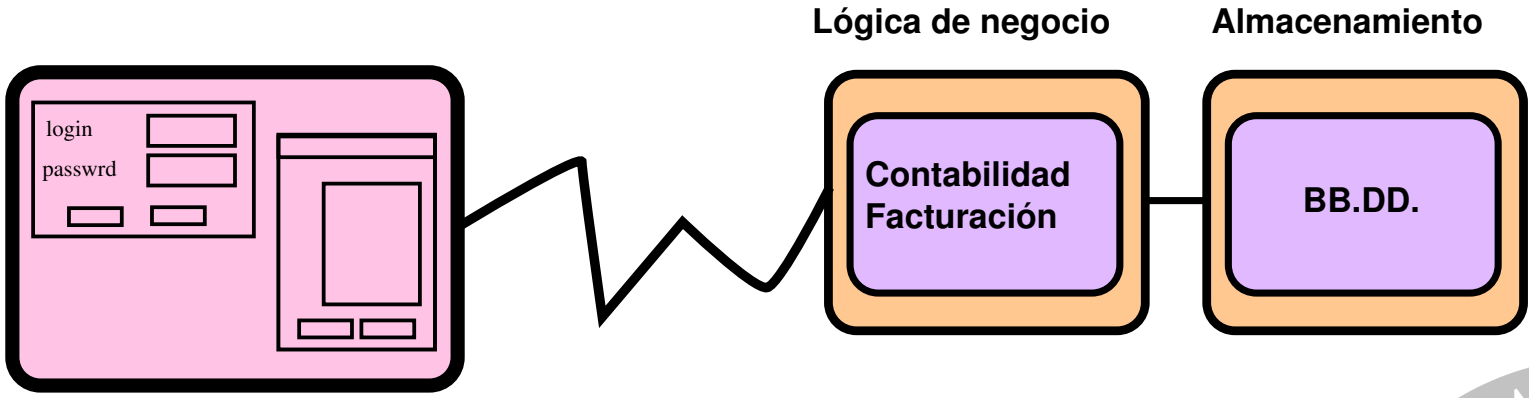
- 3 niveles (3-tier).
  - La aplicación se divide en partes (niveles).
- n niveles (n-tier).
  - Generalización de 3-tier.
  - Mejor aproximación a aplicaciones complejas.
  - Ventajas, inconvenientes:

# C/S: clasificación (2,3,n-tier)

- 2 niveles (2-tier).

  *Ejemplos: Los vistos en cliente grueso/delgado.*

- 3 niveles (3-tier).
  - La aplicación se divide en partes (niveles).
  - Ejemplo:




- n niveles (n-tier).
  - Generalización de 3-tier.
  - Mejor aproximación a aplicaciones complejas.
  - Ventajas, inconvenientes:

# C/S: clasificación (2,3,n-tier)

- 2 niveles (2-tier).

  *Ejemplos: Los vistos en cliente grueso/delgado.*

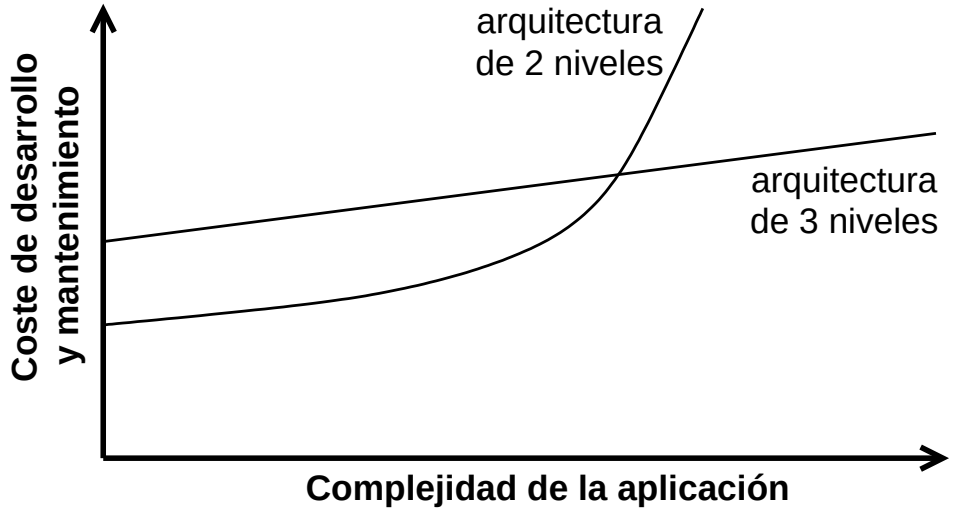
- 3 niveles (3-tier).
  - La aplicación se divide en partes (niveles).
- n niveles (n-tier).
  - Generalización de 3-tier.
  - Mejor aproximación a aplicaciones complejas.
  - Ventajas, inconvenientes:
    - Diseño, desarrollo más complejos, más infraestructura.
    - Flexibilidad (cambios, nueva funcionalidad, . . . ).
    - Escalabilidad.

# C/S: clasificación (2,3,n-tier)

- 2 niveles (2-tier).

  *Ejemplos: Los vistos en cliente grueso/delgado.*

- 3 niveles (3-tier).
  - La aplicación se divide en partes (niveles).
- n niveles (n-tier).
  - Generalización de 3-tier.
  - Mejor aproximación a aplicaciones complejas.
  - Ventajas, inconvenientes:



# Aplicaciones Web

- Originalmente pensada para intercambio de documentos
- Arquitectura Cliente/Servidor
  - Cliente delgado
- Protocolos y estándares Web
  - IETF: HTTP/1.0 (RFC 1945), HTTP/2.0 (RFC 2068)
  - W3C: HTML 4.0, XHTML, CSS

# Protocolo HTTP

- Modelo petición/respuesta (RPC)
- Protocolo sin estado
- Habitual, pero no obligatorio: TCP/IP (80, 443)
- Se establece una conexión, y se intercambia petición de recurso, respuesta

# Protocolo HTTP. Peticiones

- Texto ASCII, *Human readable*.
- Varias líneas de texto (1..*n*).
- Primera línea (obligatoria):

| *Petición* | *URI* | *Versión* |
|---|---|---|
| *GET* | */index.html* | *HTTP/1.1* |

- Existen varios tipos de petición: GET, POST, HEAD, ...
- Siguientes líneas: *cabeceras*
- Una cabecera por línea. Formato: `nombre: valor`
- En HTTP/1.0 las cabeceras son opcionales.
- En HTTP/1.1 algunas cabeceras son obligatorias.
- Ejemplo: `host: www.grupomads.org`
- Las cabeceras terminan con una línea en blanco.
- Opcionalmente puede haber un *cuerpo* de la petición con datos arbitrarios.

# Protocolo HTTP. Peticiones

- Texto ASCII, *Human readable*.
- Varias líneas de texto (1..*n*).
- Primera línea (obligatoria):

| *Petición* | *URI* | *Versión* |
|---|---|---|
| *GET* | */index.html* | *HTTP/1.1* |

- Existen varios tipos de petición: GET, POST, HEAD, ...
- Siguientes líneas: *cabeceras*
- Una cabecera por línea. Formato: `nombre: valor`
- En HTTP/1.0 las cabeceras son opcionales.
- En HTTP/1.1 algunas cabeceras son obligatorias.
- Ejemplo: `host: www.grupomads.org`
- Las cabeceras terminan con una línea en blanco.
- Opcionalmente puede haber un *cuerpo* de la petición con datos arbitrarios.

# Protocolo HTTP. Peticiones

- Texto ASCII, *Human readable*.
- Varias líneas de texto (1..*n*).
- Primera línea (obligatoria):

| *Petición* | *URI* | *Versión* |
|---|---|---|
| *GET* | */index.html* | *HTTP/1.1* |

- Existen varios tipos de petición: GET, POST, HEAD, ...
- Siguientes líneas: *cabeceras*
- Una cabecera por línea. Formato: `nombre: valor`
- En HTTP/1.0 las cabeceras son opcionales.
- En HTTP/1.1 algunas cabeceras son obligatorias.
- Ejemplo: `host: www.grupomads.org`
- Las cabeceras terminan con una línea en blanco.
- Opcionalmente puede haber un *cuerpo* de la petición con datos arbitrarios.

# Protocolo HTTP. Peticiones

- Texto ASCII, *Human readable*.
- Varias líneas de texto (1..*n*).
- Primera línea (obligatoria):

| *Petición* | *URI* | *Versión* |
|---|---|---|
| *GET* | */index.html* | *HTTP/1.1* |

- Existen varios tipos de petición: GET, POST, HEAD, ...
- Siguientes líneas: *cabeceras*
- Una cabecera por línea. Formato: `nombre: valor`
- En HTTP/1.0 las cabeceras son opcionales.
- En HTTP/1.1 algunas cabeceras son obligatorias.
- Ejemplo: `host: www.grupomads.org`
- Las cabeceras terminan con una línea en blanco.
- Opcionalmente puede haber un *cuerpo* de la petición con datos arbitrarios.

# Protocolo HTTP. Respuestas

- Similar a las peticiones.
- Primera línea:

| *Versión* | *Código* | *Descripción* |
|---|---|---|
| *HTTP/1.0* | *200* | *OK* |

- Cabeceras. Importante, p.e. `Content-type: text/html`
- Línea en blanco y a continuación el recurso solicitado.

# Protocolo HTTP. Respuestas

- Similar a las peticiones.
- Primera línea:

| *Versión* | *Código* | *Descripción* |
|---|---|---|
| *HTTP/1.0* | *200* | *OK* |

- Cabeceras. Importante, p.e. `Content-type: text/html`
- Línea en blanco y a continuación el recurso solicitado.

# Protocolo HTTP. Respuestas

- Similar a las peticiones.
- Primera línea:

| *Versión* | *Código* | *Descripción* |
|---|---|---|
| *HTTP/1.0* | *200* | *OK* |

- Cabeceras. Importante, p.e. `Content-type: text/html`
- Línea en blanco y a continuación el recurso solicitado.

# Protocolo HTTP. Extensión

- El protocolo HTTP se puede extender de varias formas.
  - Nuevos tipos de peticiones.
    Ejemplo: DAV (PROPFIND, LOCK, WRITE, COPY, . . . )
  - Añadiendo información en las cabeceras.
    Ejemplos: autenticación, cookies, . . .

# Protocolo HTTP. Extensión

- El protocolo HTTP se puede extender de varias formas.
  - Nuevos tipos de peticiones.
    Ejemplo: DAV (PROPFIND, LOCK, WRITE, COPY, . . . )
  - Añadiendo información en las cabeceras.
    Ejemplos: autenticación, cookies, . . .

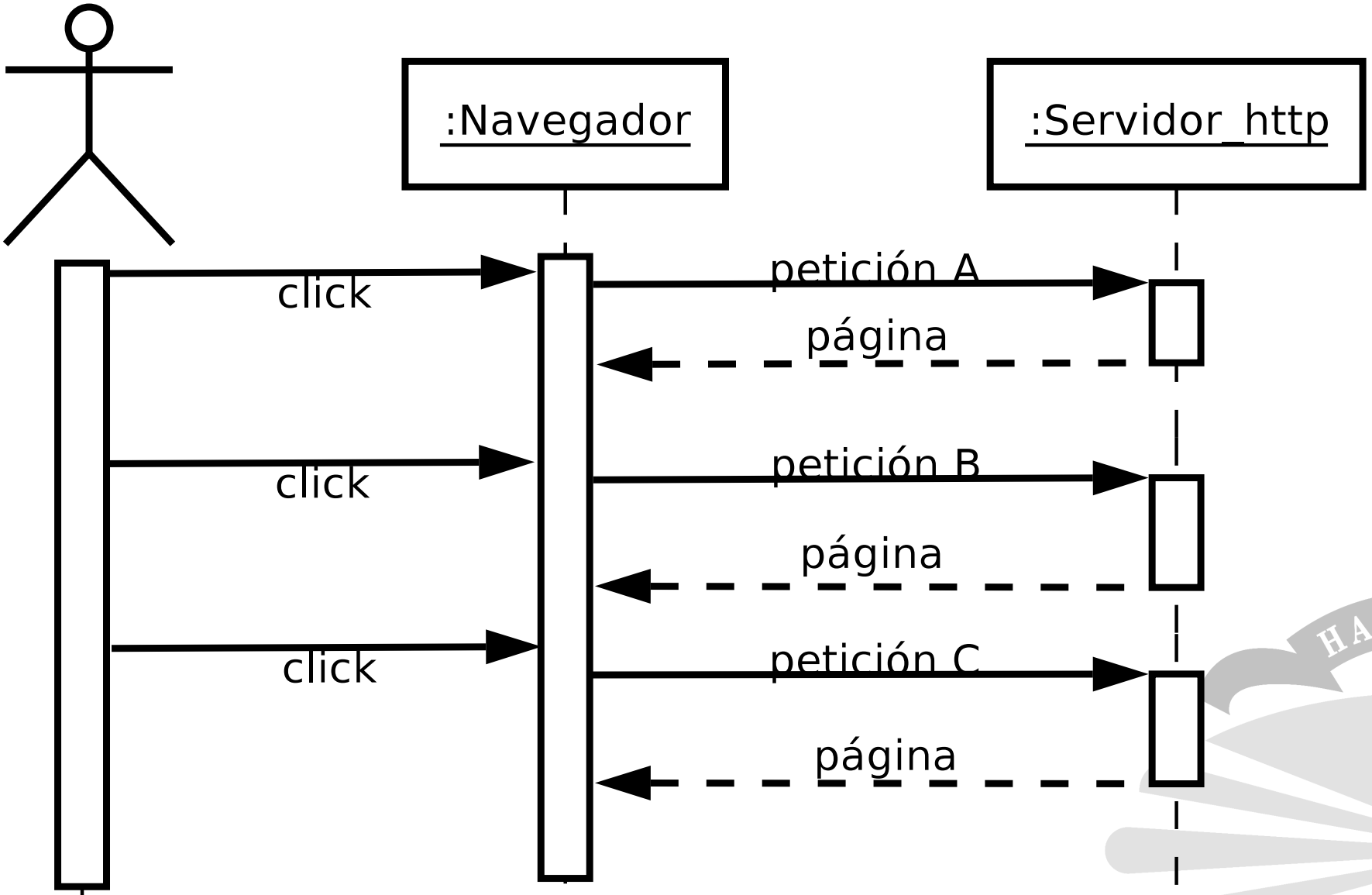
# Modelo de aplicaciones web

- Servidor:
  1. Recibir petición.
  2. Procesar petición.
  3. Generar respuesta (i.e. página web).
- Cliente:
  - Muestra páginas.
  - Recibe entrada del usuario.

# Modelo de aplicaciones web

- Servidor:
  1. Recibir petición.
  2. Procesar petición.
  3. Generar respuesta (i.e. página web).
- Cliente:
  - Muestra páginas.
  - Recibe entrada del usuario.

# Modelo de aplicaciones web (cont.)



# Modelo de aplicaciones web (cont.)

- Cuestiones adicionales:
  - Mapear URIs al API del servidor.
  - Paso de parámetros.
  - Implementar estado.
- Relación con servidor web.
  - Es habitual que un servidor web reciba todas las peticiones.
  - Delega las peticiones correspondientes a la aplicación:
    - CGI
    - Integración con el servidor web (p.e. módulos apache)
    - Contenedor de aplicaciones (p.e. tomcat)
  - Evitamos implementar el protocolo HTTP.

# Modelo de aplicaciones web (cont.)

- Cuestiones adicionales:
  - Mapear URIs al API del servidor.
  - Paso de parámetros.
  - Implementar estado.
- Relación con servidor web.
  - Es habitual que un servidor web reciba todas las peticiones.
  - Delega las peticiones correspondientes a la aplicación:
    - CGI
    - Integración con el servidor web (p.e. módulos apache)
    - Contenedor de aplicaciones (p.e. tomcat)
  - Evitamos implementar el protocolo HTTP.

# CGI

- CGI (Common Gateway Interface).
- Protocolo para intercambio de información entre el servidor web y una aplicación externa.
- Paso 1: del servidor a la aplicación externa.
  - La aplicación se lanza en un proceso nuevo.
  - Parte de la información en las variables de entorno.
    La especificación CGI define un conjunto de variables, tanto obligatorias como opcionales.
    Ejemplos: `SERVER_SOFTWARE`, `PATH_INFO`, ...
  - Parte de la información en la entrada estándar de la aplicación.
    El cuerpo de las peticiones POST y PUT.
- Paso 2: de la aplicación externa al servidor.
  - La respuesta se escribe en la salida estándard.
  - La respuesta comenzará por una cabecera HTTP.
    - `Content-type`
    - `Location`
    - `Status`

# CGI

- CGI (Common Gateway Interface).
- Protocolo para intercambio de información entre el servidor web y una aplicación externa.
- Paso 1: del servidor a la aplicación externa.
  - La aplicación se lanza en un proceso nuevo.
  - Parte de la información en las variables de entorno.
    La especificación CGI define un conjunto de variables, tanto obligatorias como opcionales.
    Ejemplos: `SERVER_SOFTWARE`, `PATH_INFO`, ...
  - Parte de la información en la entrada estándar de la aplicación.
    El cuerpo de las peticiones POST y PUT.
- Paso 2: de la aplicación externa al servidor.
  - La respuesta se escribe en la salida estándard.
  - La respuesta comenzará por una cabecera HTTP.
    - `Content-type`
    - `Location`
    - `Status`

# CGI

- CGI (Common Gateway Interface).
- Protocolo para intercambio de información entre el servidor web y una aplicación externa.
- Paso 1: del servidor a la aplicación externa.
  - La aplicación se lanza en un proceso nuevo.
  - Parte de la información en las variables de entorno.
    La especificación CGI define un conjunto de variables, tanto obligatorias como opcionales.
    Ejemplos: `SERVER_SOFTWARE`, `PATH_INFO`, ...
  - Parte de la información en la entrada estándar de la aplicación.
    El cuerpo de las peticiones POST y PUT.
- Paso 2: de la aplicación externa al servidor.
  - La respuesta se escribe en la salida estándard.
  - La respuesta comenzará por una cabecera HTTP.
    - `Content-type`
    - `Location`
    - `Status`

# CGI

- CGI (Common Gateway Interface).
- Protocolo para intercambio de información entre el servidor web y una aplicación externa.
- Paso 1: del servidor a la aplicación externa.
  - La aplicación se lanza en un proceso nuevo.
  - Parte de la información en las variables de entorno.
    La especificación CGI define un conjunto de variables, tanto obligatorias como opcionales.
    Ejemplos: `SERVER_SOFTWARE`, `PATH_INFO`, ...
  - Parte de la información en la entrada estándar de la aplicación.
    El cuerpo de las peticiones POST y PUT.
- Paso 2: de la aplicación externa al servidor.
  - La respuesta se escribe en la salida estándard.
  - La respuesta comenzará por una cabecera HTTP.
    - `Content-type`
    - `Location`
    - `Status`

# Mapear URIs

- Si usamos un servidor web.
  - Depende del servidor web.
  - Esperable: establecer correspondencias entre prefijos de los URIs y aplicaciones externas o módulos del servidor.
- Si no usamos un servidor web.
  - Responsabilidad de nuestra aplicación.

# Mapear URIs

- Si usamos un servidor web.
  - Depende del servidor web.
  - Esperable: establecer correspondencias entre prefijos de los URIs y aplicaciones externas o módulos del servidor.
- Si no usamos un servidor web.
  - Responsabilidad de nuestra aplicación.

# Mapear URIs

- Si usamos un servidor web.
  - Depende del servidor web.
  - Esperable: establecer correspondencias entre prefijos de los URIs y aplicaciones externas o módulos del servidor.
- Si no usamos un servidor web.
  - Responsabilidad de nuestra aplicación.

# Mapear URIs

- Si usamos un servidor web.
  - Depende del servidor web.
  - Esperable: establecer correspondencias entre prefijos de los URIs y aplicaciones externas o módulos del servidor.
- Si no usamos un servidor web.
  - Responsabilidad de nuestra aplicación.

# HTTP: paso de parámetros

- Tres métodos fundamentales.
  - Incluirlos como parte del URI en la petición

    ```
    GET /app/func/arg1/arg2 HTTP/1.0
    ```

  - Añadirlos al URI en la petición

    ```
    GET /app/fun?name1=value1&name2=value2
    HTTP/1.0
    ```

  - Usar el método POST
    - Los parámetros van en el cuerpo de la petición.
    - Del lado del cliente (navegador web) implica usar FORMS.

# HTTP: paso de parámetros

- Tres métodos fundamentales.
  - Incluirlos como parte del URI en la petición

    ```
    GET /app/func/arg1/arg2 HTTP/1.0
    ```

  - Añadirlos al URI en la petición

    ```
    GET /app/fun?name1=value1&name2=value2
    HTTP/1.0
    ```

  - Usar el método POST
    - Los parámetros van en el cuerpo de la petición.
    - Del lado del cliente (navegador web) implica usar FORMS.

# HTTP: paso de parámetros

- Tres métodos fundamentales.
  - Incluirlos como parte del URI en la petición

    ```
    GET /app/func/arg1/arg2 HTTP/1.0
    ```

  - Añadirlos al URI en la petición

    ```
    GET /app/fun?name1=value1&name2=value2
    HTTP/1.0
    ```

  - Usar el método POST
    - Los parámetros van en el cuerpo de la petición.
    - Del lado del cliente (navegador web) implica usar FORMS.

# HTTP: paso de parámetros

- Tres métodos fundamentales.
  - Incluirlos como parte del URI en la petición

    ```
    GET /app/func/arg1/arg2 HTTP/1.0
    ```

  - Añadirlos al URI en la petición

    ```
    GET /app/fun?name1=value1&name2=value2
    HTTP/1.0
    ```

  - Usar el método POST
    - Los parámetros van en el cuerpo de la petición.
    - Del lado del cliente (navegador web) implica usar FORMS.