Escuela Politécnica de Ingeniería
Grado de Ingeniería Informática en Tecnologías de la Información

Tecnologías Web

Tema 4

Seguridad y aspectos legales en la Web

Índice

- *Autenticación/Autorización y Auditoría*
- *Ley de protección de datos*

Conceptos

- Autenticación (o acreditación)
  - Asegurar que el usuario es quien dice ser
- Autorización
  - Verificar que tiene acceso al recurso que solicita
- Auditoria
  - Registrar todos los accesos de usuarios a recursos

Métodos de Autenticación

- Se basan en [...] el usuario
  - algo conocido por
  - algo poseído por
  - una característica física de
  - algo que hace
- Puede haber combinaciones de varios
  - Autenticación multifactor

AAA en aplicaciones Web

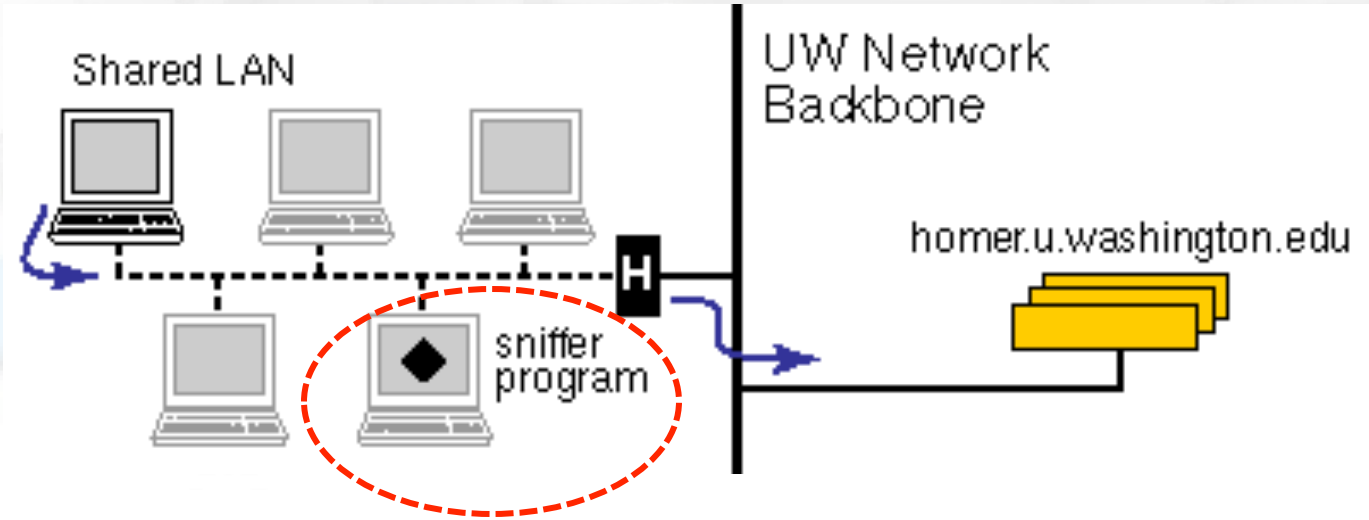
- Autenticación
  - Se delega en sistemas PAM
    - Pluggable Authentication Modules
  - Se puede emplear infraestrucutra PKI
- Autorización
  - Por contenedor, declarativa
  - Por aplicación, programática
- Auditoria
  - Por aplicación
  - Uso de frameworks

Problemas de seguridad en redes

- Suplantación de servidores
- Suplantación de la identidad de un usuario
- Tráfico entre cliente y servidor
  - Monitorizarlo
  - Corromperlo
  - Desviarlo
  - Bloquearlo
- …

Transmisión en Web

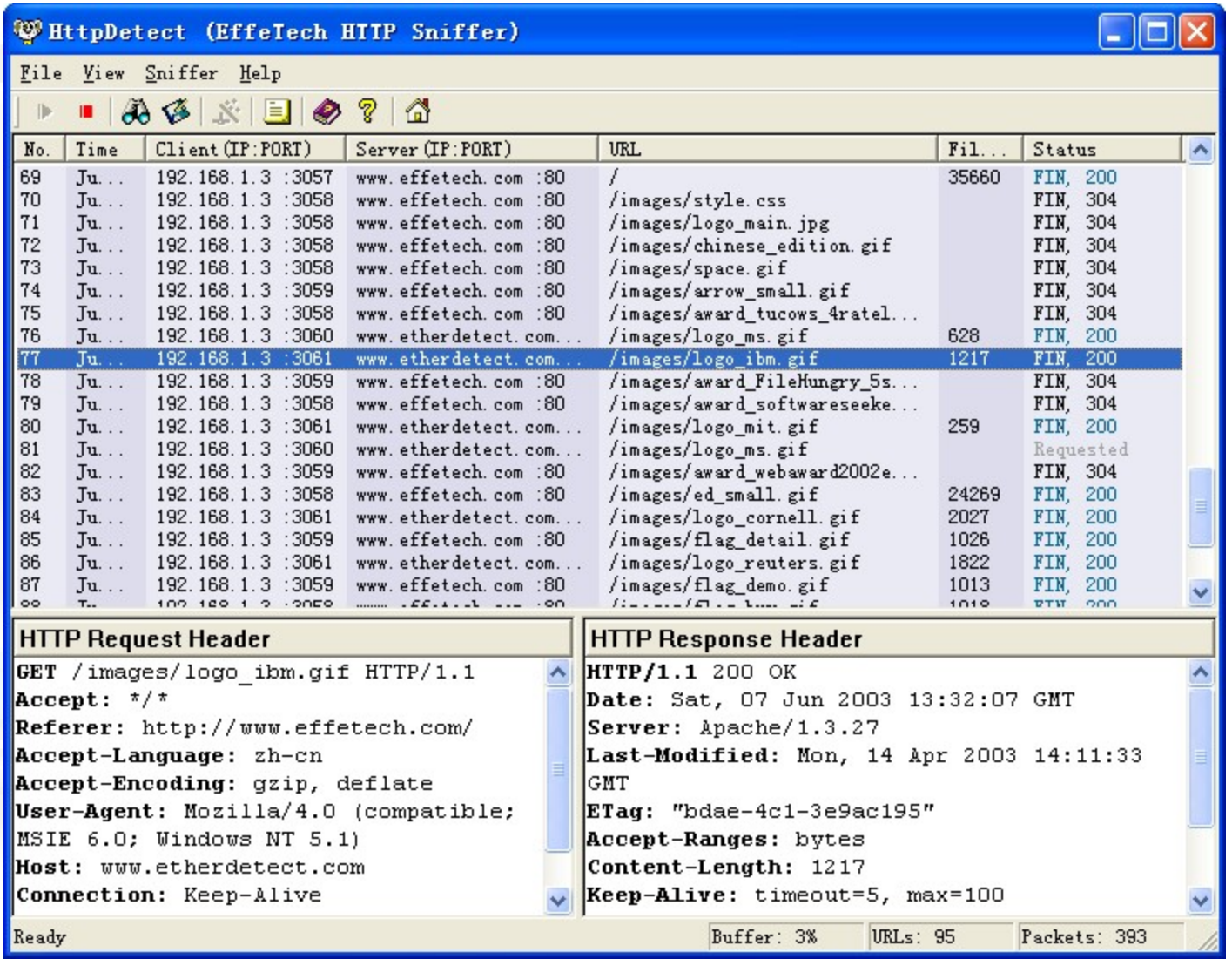
- El tráfico en Internet puede ser espiado y alterado con mucha facilidad
- ¿Cómo intercambiar datos confidenciales?



Packet Sniffing

Login y pass
van en cabeceras
HTTP. Contenidos en
HTML (texto)

Cabeceras HTTP
en texto plano

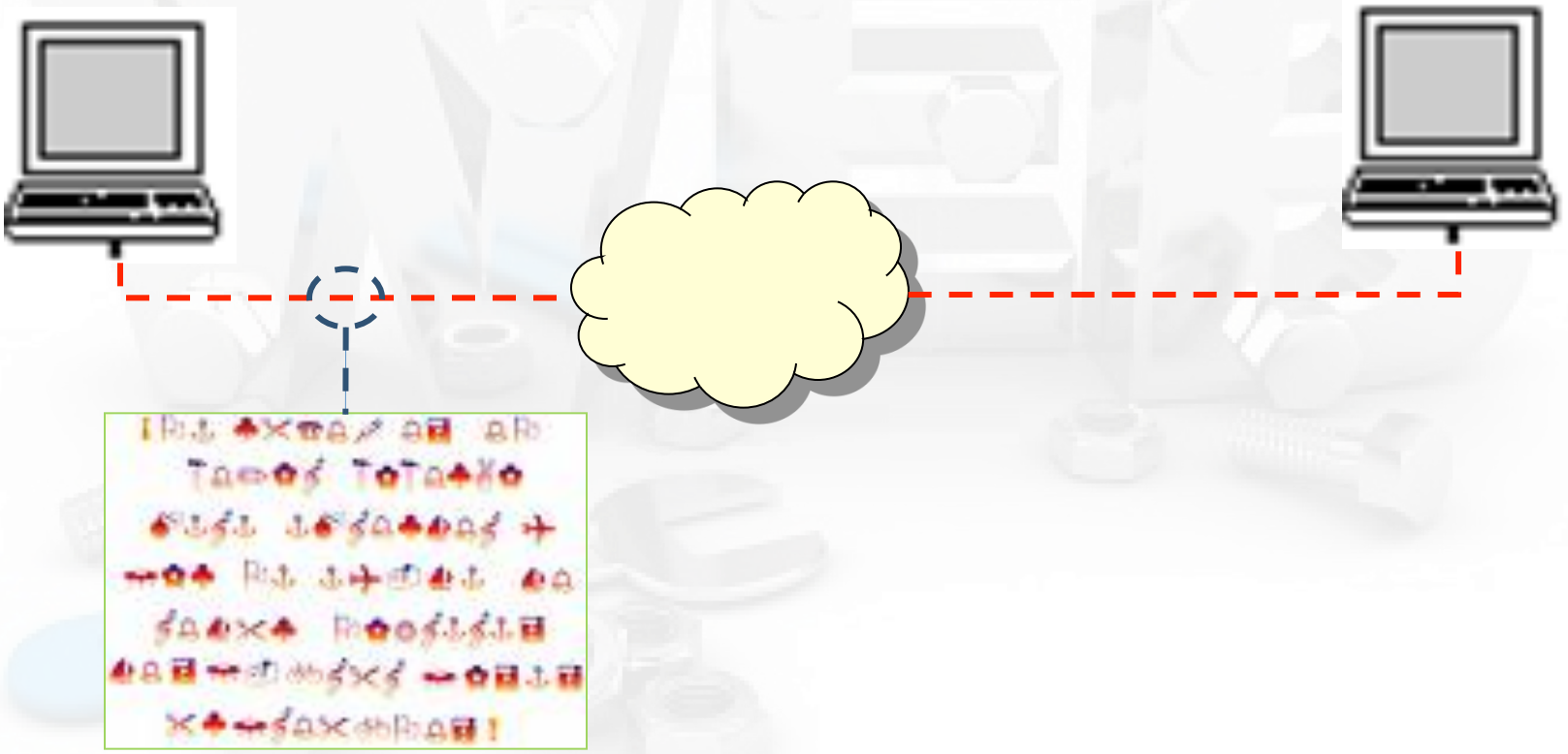


Infraestructura de Seguridad

- En inglés PKI, Public Key Infrastructure
- Objetivos de las PKI
  - Control de acceso
  - Autenticación
  - Confidencialidad
  - Integridad
  - No repudio

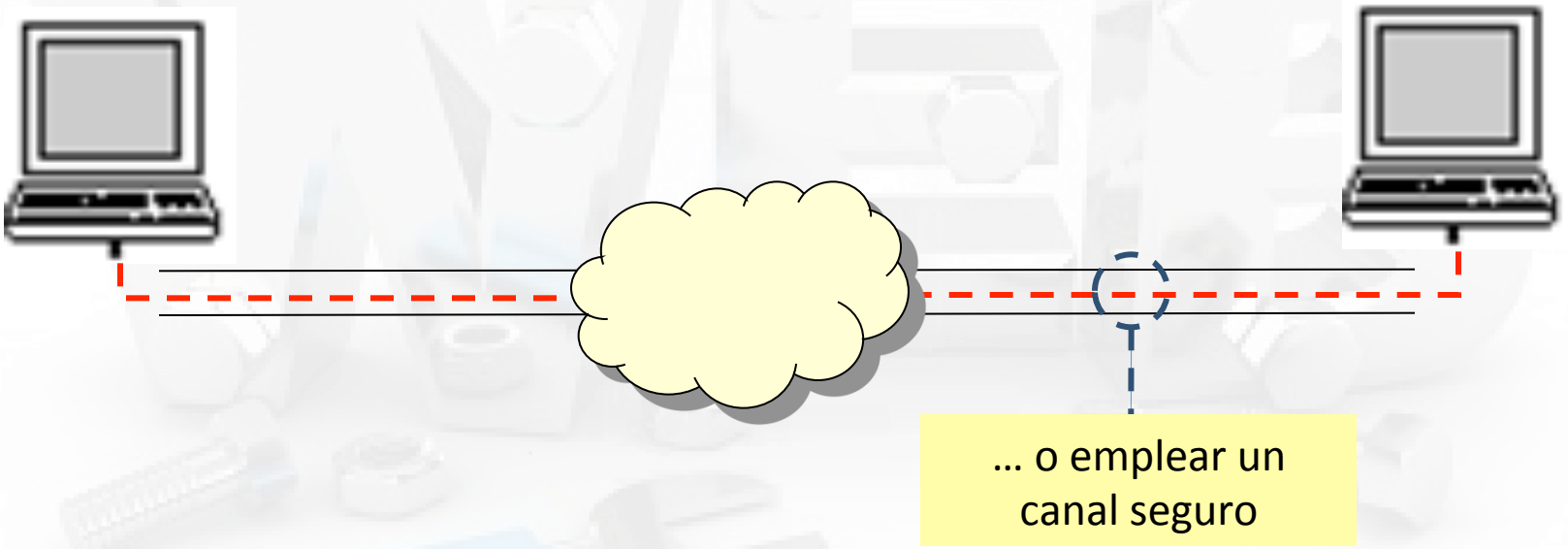
Seguridad, alternativas

- Codificar el mensaje



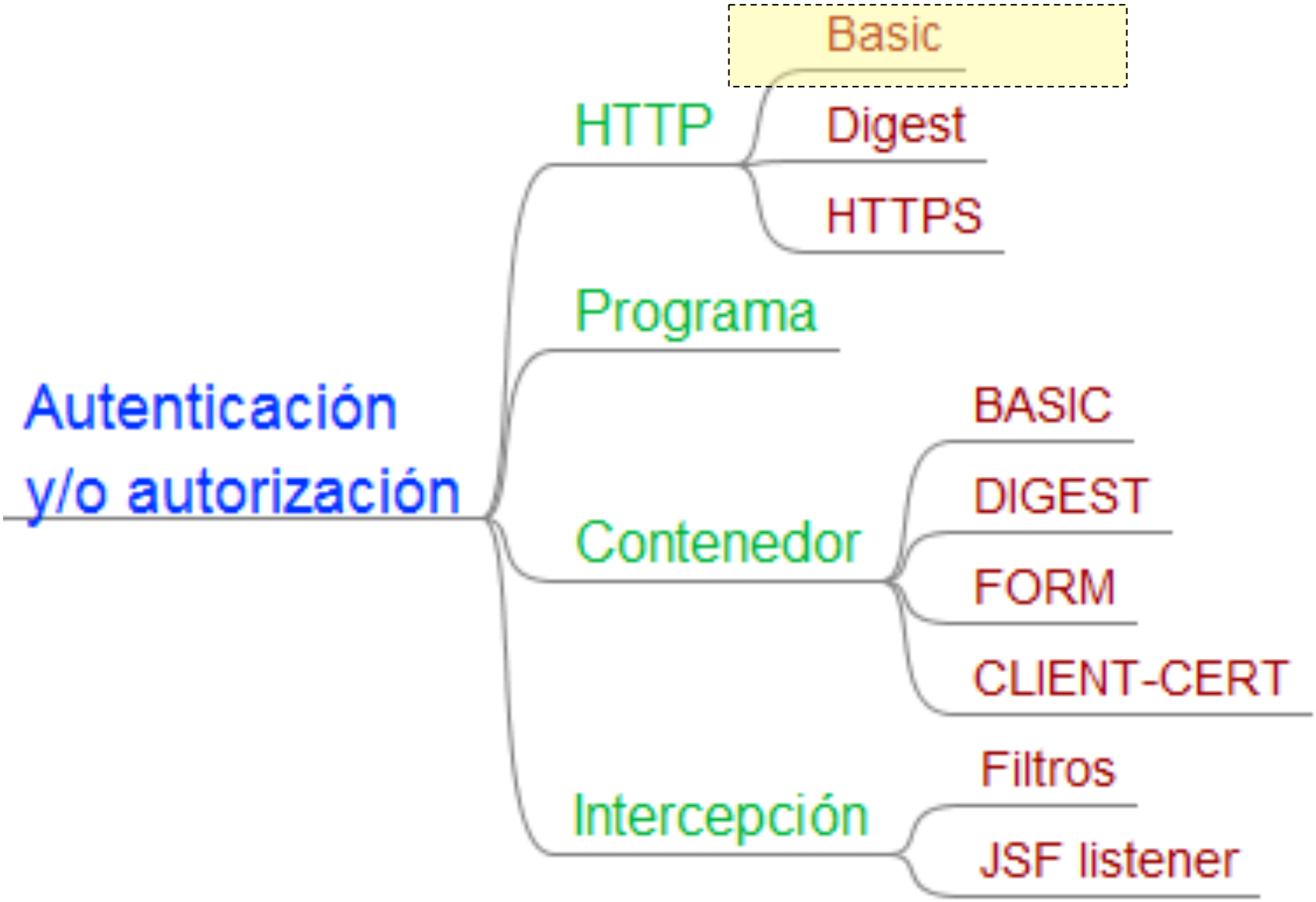
Seguridad, alternativas

- Emplear un canal seguro



Autorización

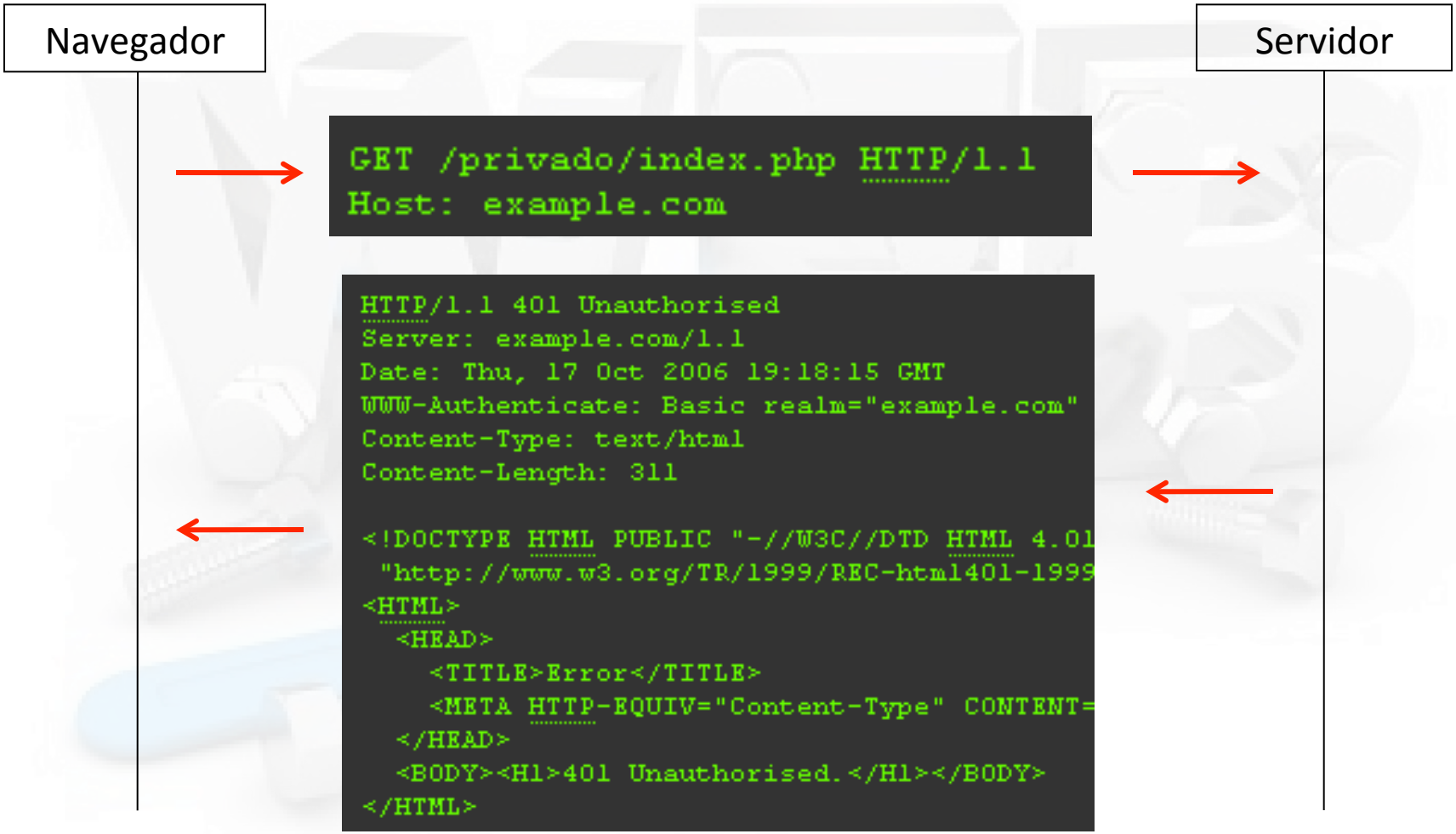
- Basada en listas de control de acceso (ACL)
  - Para cada recurso se especifica qué usuarios pueden hacer qué operaciones
- Basada en Roles
  - Rol: grupo de permisos.
  - Permiso: operación sobre un recurso.
  - A los usuarios se les asignan roles.


Autenticación
y/o autorización
HTTP
Basic
Digest
HTTPS
Programa
Contenedor
BASIC
DIGEST
FORM
CLIENT-CERT
Intercepción
Filtros
JSF listener

Autenticación HTTP Básica

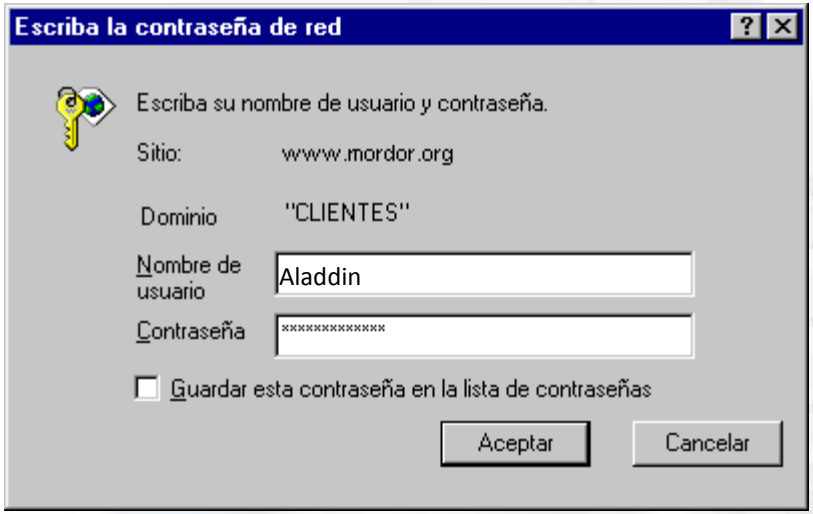
- Cuando el navegador solicita un recurso (jsp, por ejemplo), el servidor solicita usuario y contraseña, que viajan codificadas en Base-64.
- Disponible desde versión 1.0 de HTTP
- Muy simple
- Poco seguro

Autenticación HTTP Básica



Autenticación HTTP Básica

- Login: Aladdin, pass: open sesame
- “Aladdin:open sesame”
  - QWxhZGRpbjpvcGVuIHNlc2FtZQ==

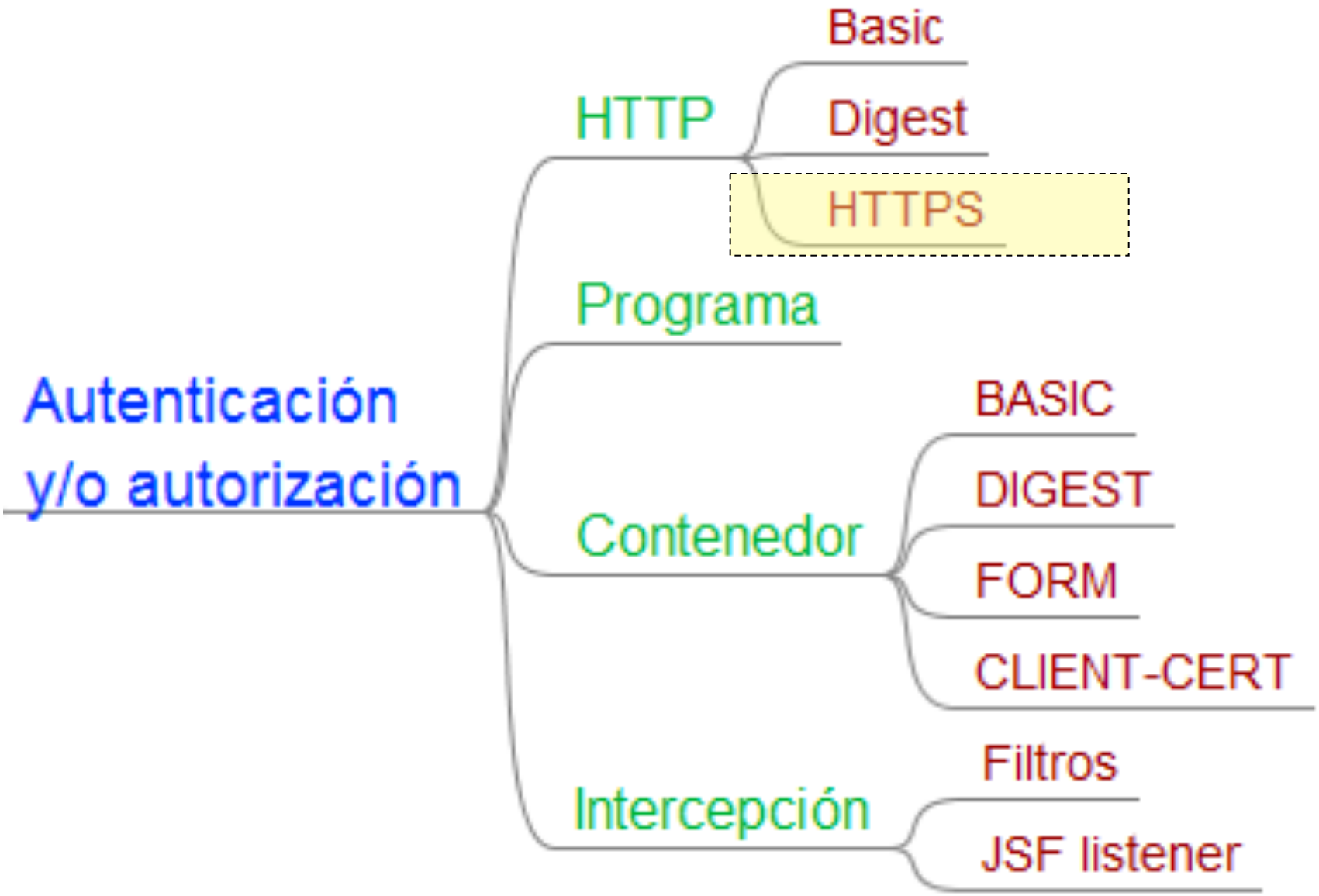


Base64

```
GET /privado/index.php HTTP/1.1
Host: example.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

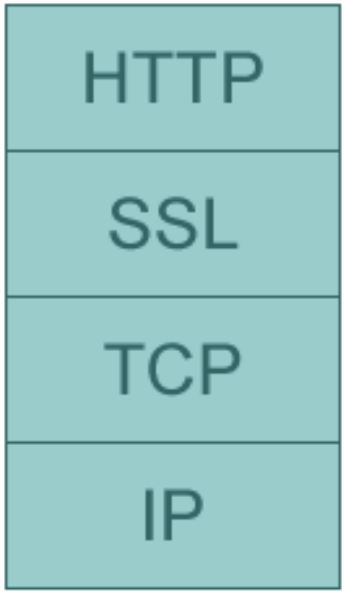
Autenticación HTTP digest (hash)

- Desde la versión 1.1 de HTTP
- La clave viaja encriptada en base a un timestamp generado por el servidor en el momento de la petición
- Más difícil de decodificar, Más seguro.
- No está totalmente difundida ni aceptada por todos los navegadores y servidores.


Autenticación
y/o autorización
HTTP
Basic
Digest
HTTPS
Programa
Contenedor
BASIC
DIGEST
FORM
CLIENT-CERT
Intercepción
Filtros
JSF listener

Seguridad para WEB (HTTPS)

- HTTPS = HTTP Secure
- HTTP sobre un protocolo de transporte seguro
  - HTTP → SSL → TCP



- SSL también es conocido como TLS según IETF

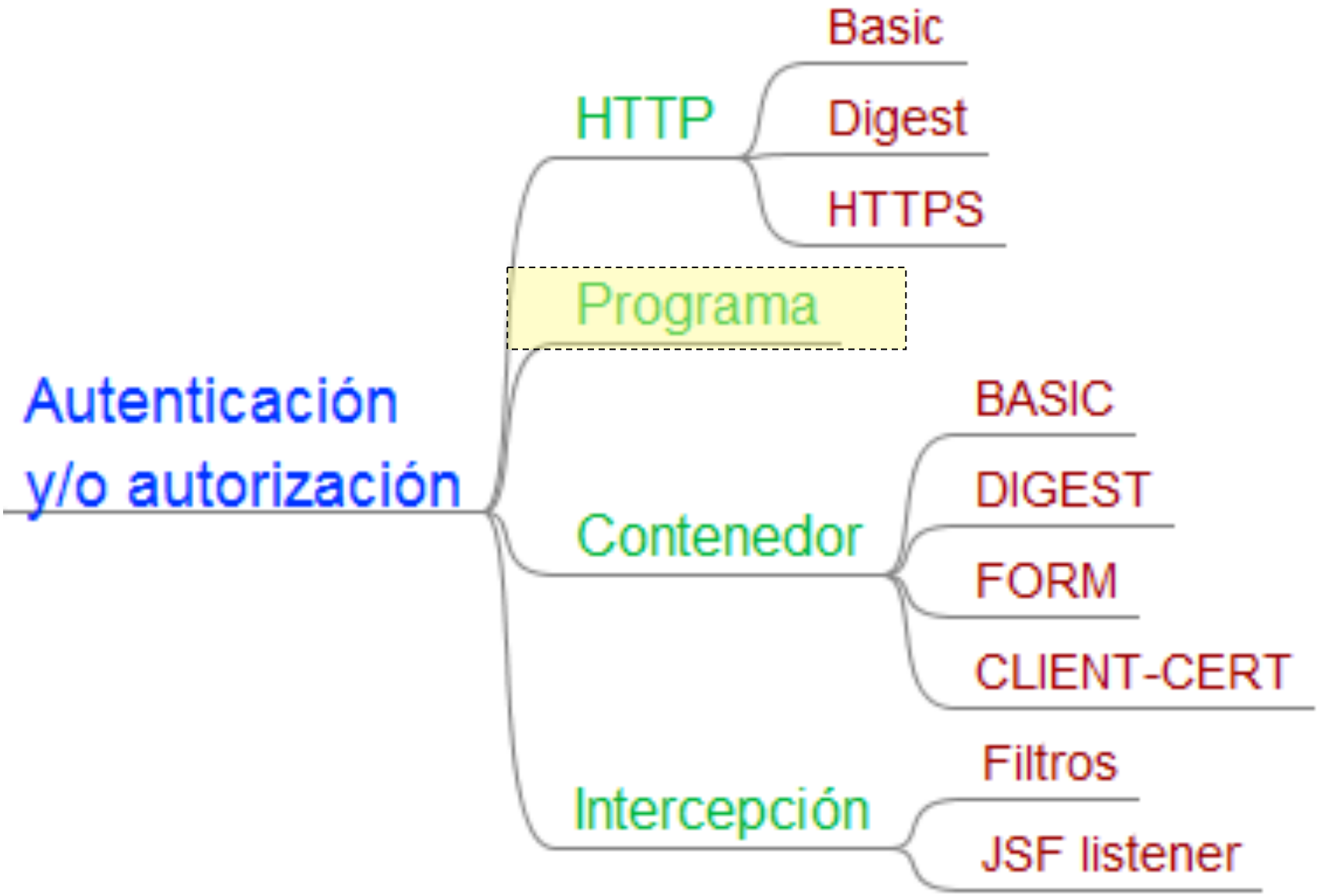
Transmisión con HTTPs

- Encripta todo el contenido de los paquetes, no sólo la clave.
- Se basa en mecanismos de Clave Pública (PKI): SSL (aka TLS)
- Requiere certificado(s) generado(s) por una autoridad certificadora (CEA)
- Hay un proceso "complicado" de *handshake* que garantiza la conexión segura.

SSL (o TLS)

- El servidor tiene un certificado
  - opcionalmente el cliente tiene el suyo
- Permite intercambio seguro de datos (p.e. claves de sesión)
- Usa algoritmos de clave simétrica y asimétrica
- Ofrece:
  - Autenticación
  - Confidencialidad
  - Integridad
  - No repudio

Del servidor siempre, del cliente si tiene certificado

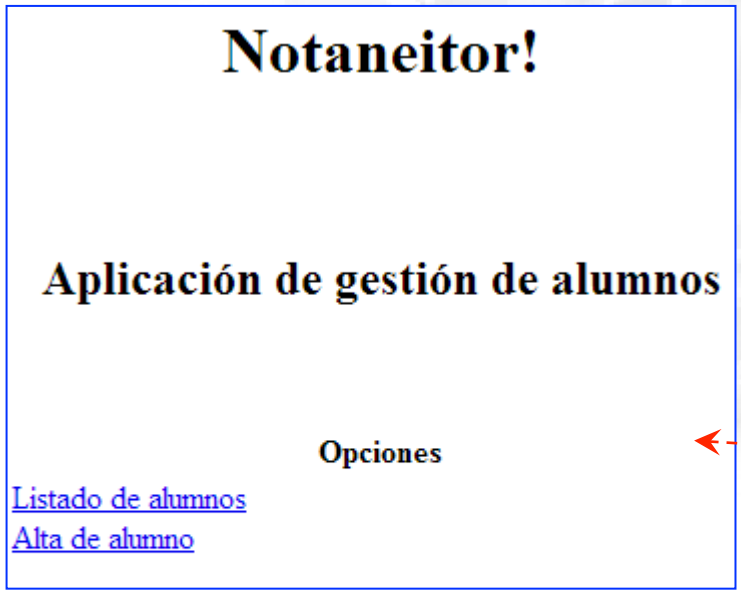

Autenticación
y/o autorización
HTTP
Basic
Digest
HTTPS
Programa
Contenedor
BASIC
DIGEST
FORM
CLIENT-CERT
Intercepción
Filtros
JSF listener

Autenticación por Programa

- Todo lo tenemos que programar
- La aplicación coteja la contraseña contra su propio sistema de control de usuarios
- Hay que basarla en protocolos seguros de comunicación como HTTPs (SSL) para evitar que nadie "escuche" la contraseña interpretando los paquetes HTTP.
- Más tedioso de programar
- Ventajas: No dependemos de la Autenticación del entorno → Más portable.


```java
@ManagedBean(name="login")
public class LoginBean implements Serializable {
    private static final long serialVersionUID = 875296

    private String name = "";
    private String password = "";

    private String result = "login_form_result_valid";

    public String verify() {

        if ( validLogin(name, password) ) {
            return "success";
        }

        result = "login_form_result_error";
        return "login";
    }
```

success



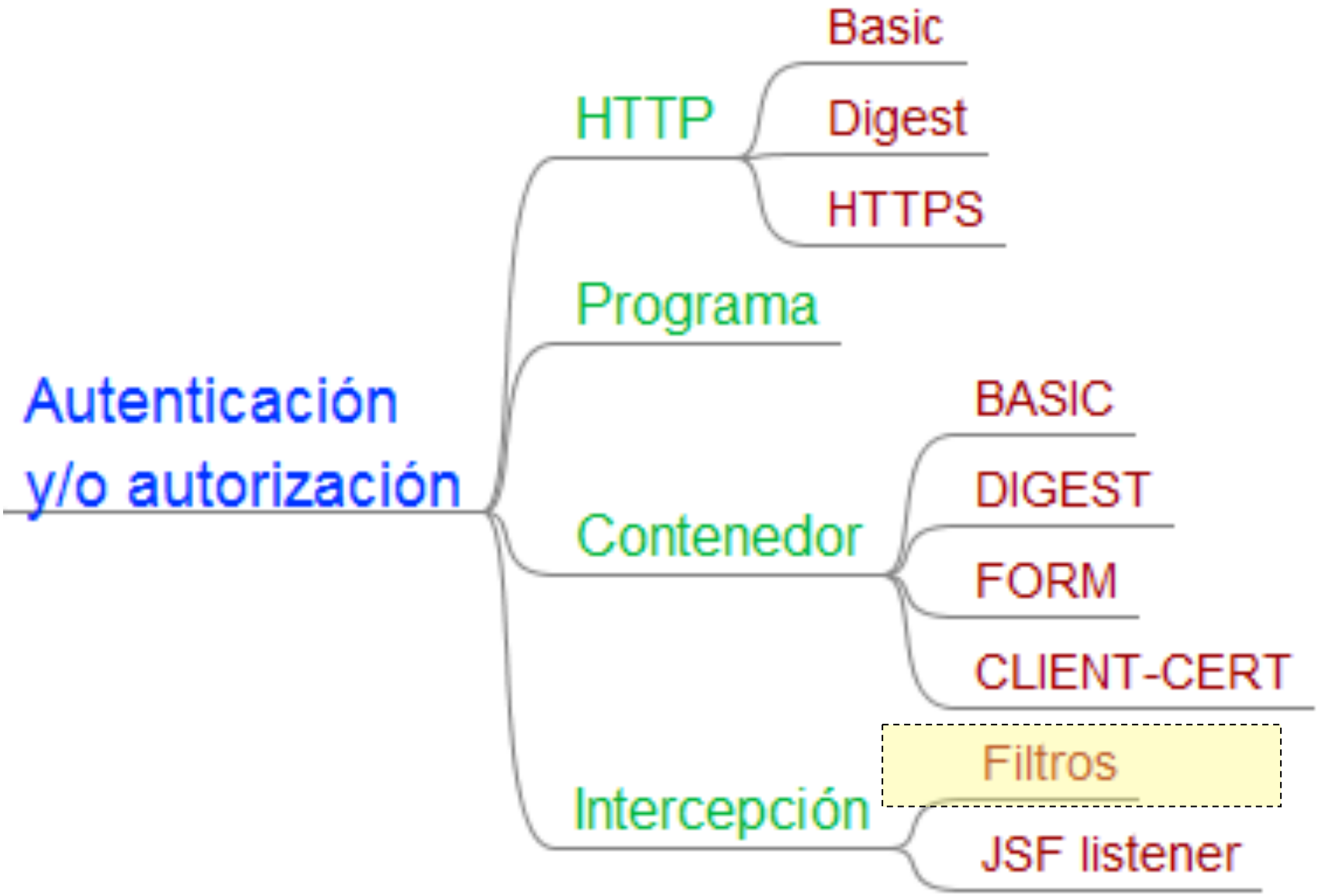
```
<ui:define name="titulo">
  #{msgs.login_form_tittle}
</ui:define>

<ui:define name="cuerpo">
<center>
      <h:form>
          <ui:include src="/snippets/login-form.xhtml"/>
          <h:panelGrid columns="2">
            <h:outputText id="msgAlta" value="#{msgs[login.result]}" />
          </h:panelGrid>
          <h:commandButton
              value="#{msgs.login_form_button_send}"
              action="#{login.verify}">
          </h:commandButton>
          <br />
      </h:form>
</center>
</ui:define>

<ui:define name="pie" />
```

```
<h:panelGrid columns="2">

    #{msgs.login_form_name_label}:
    <h:panelGroup>
        <h:inputText
            id="name"
            value="#{login.name}"
            required="true"
            requiredMessage="#{msgs.login_form_name_required}" />
        <h:message for="name" />
    </h:panelGroup>

    #{msgs.login_form_password_label}:
    <h:panelGroup>
        <h:inputSecret
            id="password"
            value="#{login.password}"
            required="true"
            requiredMessage="#{msgs.login_form_password_required}" />
        <h:message for="password" />
    </h:panelGroup>

</h:panelGrid>
```

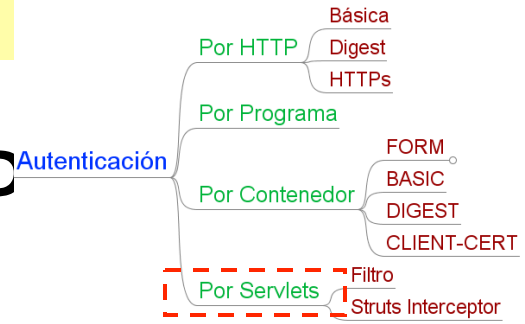
Login Form

Ejemplo por programa

```
<navigation-rule>
  <from-view-id>/index.xhtml</from-view-id>
  <navigation-case>
    <from-action>#{login.verify}</from-action>
    <from-outcome>success</from-outcome>
    <to-view-id>/main.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-action>#{login.verify}</from-action>
    <from-outcome>login</from-outcome>
    <to-view-id>/index.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

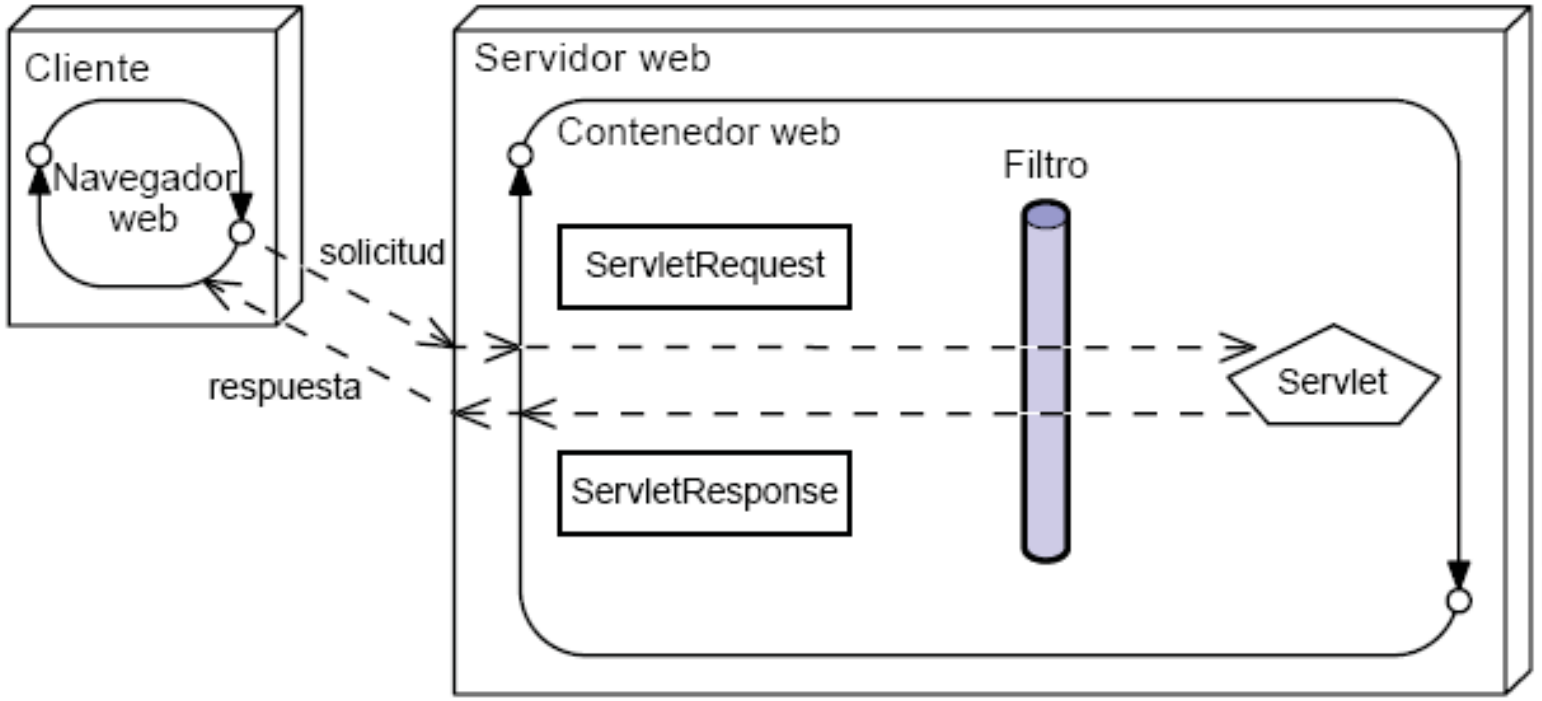

Autenticación
y/o autorización
HTTP
Basic
Digest
HTTPS
Programa
Contenedor
BASIC
DIGEST
FORM
CLIENT-CERT
Intercepción
Filtros
JSF listener

Filtros Servlet HTTP



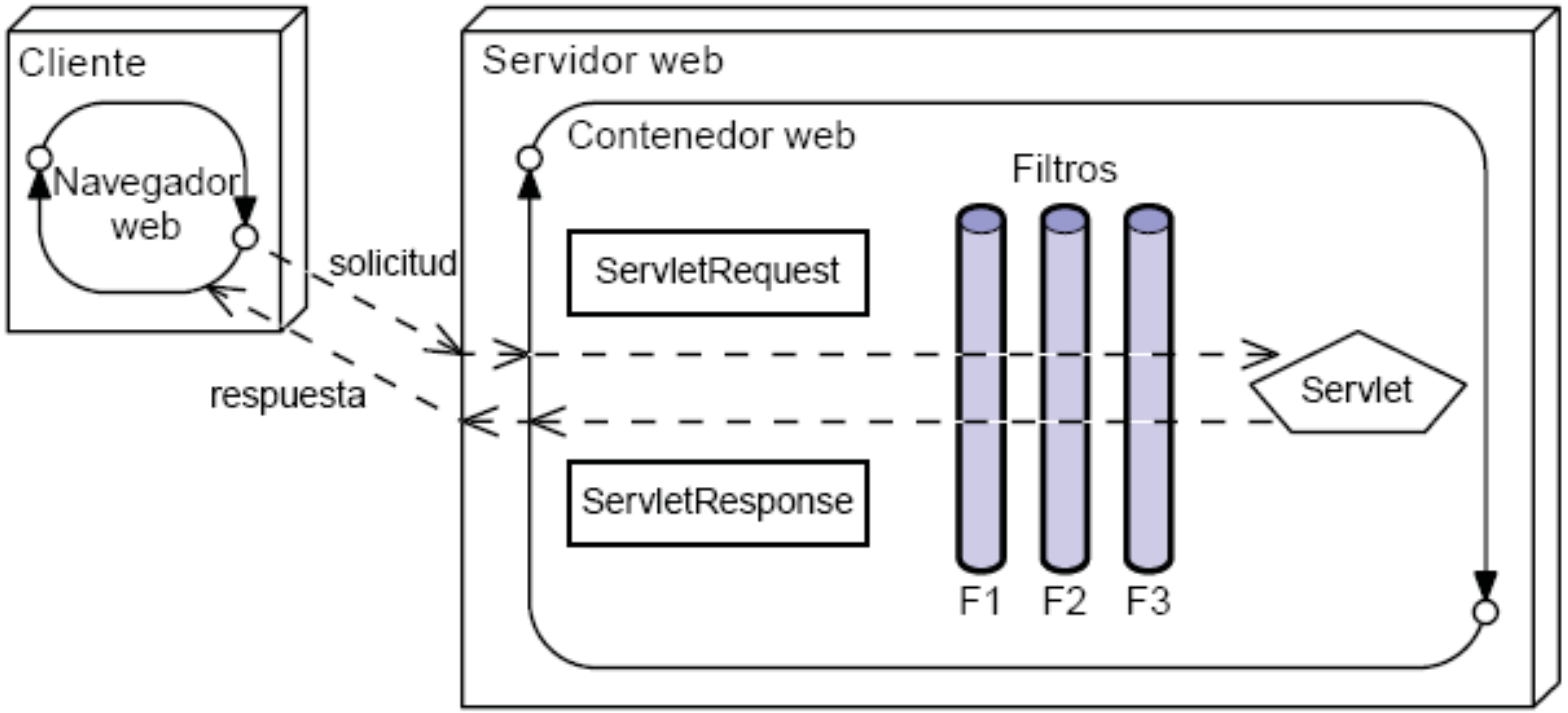
- Incorporados desde servlets 2.3.
- Interceptan la invocación del servlet
  - Antes y después de que sea invocado el propio servlet.
- Permiten examinar y modificar la request antes de que le llegue al servlet.
- Permite modificar el response y redirigir, en caso necesario, la petición a otro recurso distinto.
- Ideales para el control de acceso de usuarios autenticados

Aplicación de filtros a solicitudes entrantes



Intercepción de solicitud y respuesta mediante filtro

Cadenas de filtros



Varios filtros aplicados en cadena

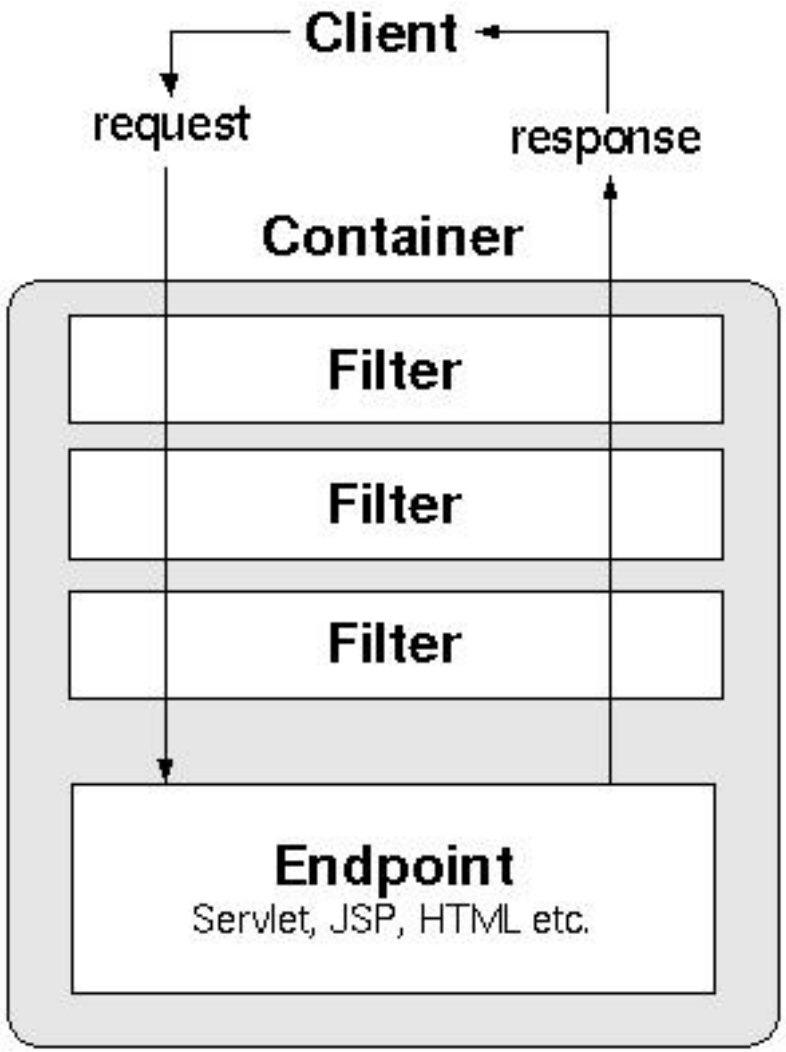
Filtros, ventaja

- Permiten hacer autenticación por programa de forma transversal a la aplicación → el código de la aplicación no se modifica
- Lo habitual es redireccionar a formulario → necesario HTTPs
  - Sigue siendo autenticación por formulario
- A parte de autenticación permiten añadir de forma genérica “aspectos” a la aplicación

Filtros Servlet HTTP

- Interfaz javax.servlet.Filter
- Tres métodos:
  - void init(FilterConfig config) throws ServletException Invocado antes de que el filtro entre en servicio. Permite configurar el filtro.
  - void destroy() Invocado cuando el filtro dejar de estar en servicio.
  - void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException: Método que implementará el filtrado.

Interfaz filter y ciclo de vida



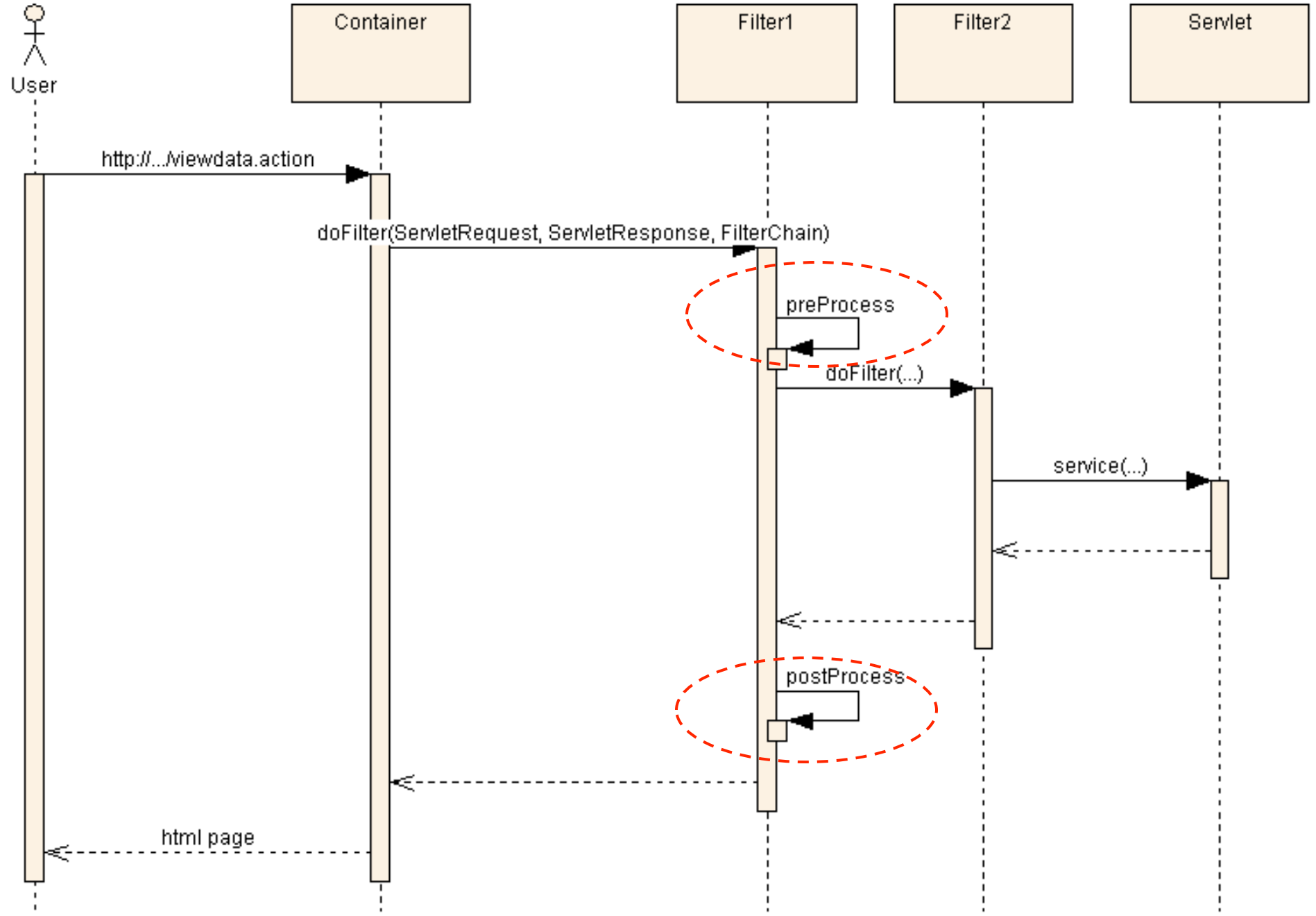
Filter
- destroy()
- doFilter(ServletRequest, ServletResponse, FilterChain)
- init(FilterConfig)

FilterConfig
- getFilterName()
- getInitParameter(String)
- getInitParameterNames()
- getServletContext()

Procesamiento típico en un filtro

- Examinar las cabeceras de request
- [Si procede]Preprocesar la petición
- Invocar al resto de la cadena de filtros
- [Si procede]Postprocesar la petición

- A la ida puede:
  - Responder él a la request y terminar proceso
  - Abortar la ejecución → UnavailableException
  - Redirigir a otra URL
  - Añadir info a session, request o contexto
- A la vuelta puede:
  - Examinar la respuesta, modificarla
  - Examinar las cabeceras añadidas


User
Container
Filter1
Filter2
Servlet
http://.../viewdata.action
doFilter(ServletRequest, ServletResponse, FilterChain)
preProcess
doFilter(...)
service(...)
postProcess
html page

Ejemplo, redirección, Notación

```
@WebFilter (    dispatcherTypes = {DispatcherType.REQUEST },
   description = "Filtro de seguridad",
   urlPatterns = { "/*" },
   initParams = {
         @WebInitParam(name = "LoginForm",
            value = "/index.xhtml",
            description = "Página de logeo")
                  }
)
public class LoginFilter implements Filter {

         ....

}
```

Ejemplo, redirección a login

```
public class LoginFilter implements Filter {
        FilterConfig config = null;
public void doFilter(ServletRequest request, ServletResponse response,
                        FilterChain chain) throws IOException, ServletException {
 // Si no es petición HTTP nada que hace
if (!(request instanceof HttpServletRequest)){
    chain.doFilter(request, response);   return;
                }
HttpServletRequest req = (HttpServletRequest) request;
// Si es la petición de login nada que hacer (si no entramos en bucle)
                String uri = req.getRequestURI().toLowerCase();
                String contextpath = req.getContextPath().toLowerCase()+"/";
                if ( uri.contains("index") || uri.equals(contextpath)){
                        chain.doFilter(request, response);
                        return;
                }
// En el resto de casos se verifica que se haya hecho login previamente
HttpServletResponse res = (HttpServletResponse) response;
HttpSession session = req.getSession();

if (session.getAttribute("LOGGEDIN_USER") == null) {
```

Ejemplo, redirección a login

```
// En el resto de casos se verifica que se haya hecho login previamente
HttpServletResponse res = (HttpServletResponse) response;
HttpSession session = req.getSession();
//Sino está logeado se le envia a la páginia de logeo (loginForm)
if (session.getAttribute("LOGGEDIN_USER") == null) {
String loginForm = config.getInitParameter("LoginForm");
// Si no hay login, redirección al formulario de login
res.sendRedirect(req.getContextPath() + loginForm);
return;
}
//Si ya está logeado se continua la cadena de logeo
chain.doFilter(request, response);
}
public void init(FilterConfig fConfig) throws ServletException {
config = fConfig;
}
}
```

Resumen Autorización

	Autenticar	Autorizar	Transporte
Protocolo HTTP	Basic Digest	----	HTTPS
Por programa	Programado	Filtros JSF listener	
Por contenedor	BASIC DIGEST FORM CLIENT-CERT		

LOPD

- Niveles de sensibilidad de los datos: bajo, medio y alto
- Formularios de autorización de incorporación de datos personales al archivo digital de la base de datos de la web comercial.
- Ejemplo: