



## Componente Documental w@ndA

### Jornada 1

- **Índice**

- **Descripción y alcance del proyecto w@rdA**
- **Introducción a los gestores documentales. Características**
- **Instalación y configuración de w@rdA**
- **El cliente web w@rdA. Conceptos y Funcionamiento**
- **Casos Prácticos**



# Solución adoptada

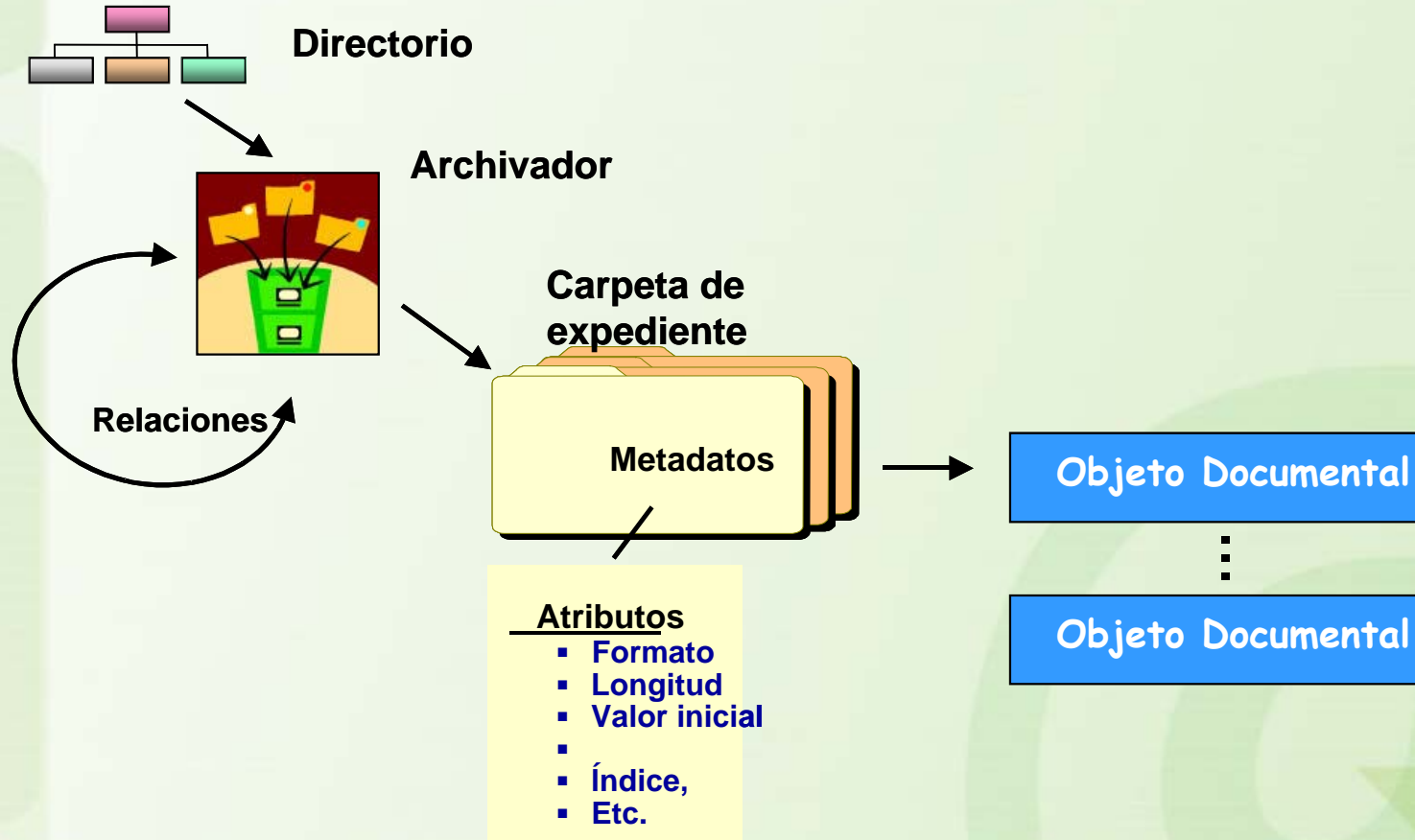
- **Producto gestor documental. Puede funcionar como una aplicación independiente.**
- **Desarrollo de los Servicios Web necesarios para cubrir las necesidades del resto de componentes w@ndA.**

# Gestores Documentales

- **Sistemas de información que son capaces de gestionar Documentos electrónicos.**
  - Información estructurada (Relacional)
  - Información no estructurada
    - Texto Libre
    - Documentos
  - Aportan un interfaz y características predefinidos (especializados)
  - Permiten modificar el modelo de datos o interfaz (genéricos)
  - Aportan herramientas para su integración en procesos de negocio (EDM....)

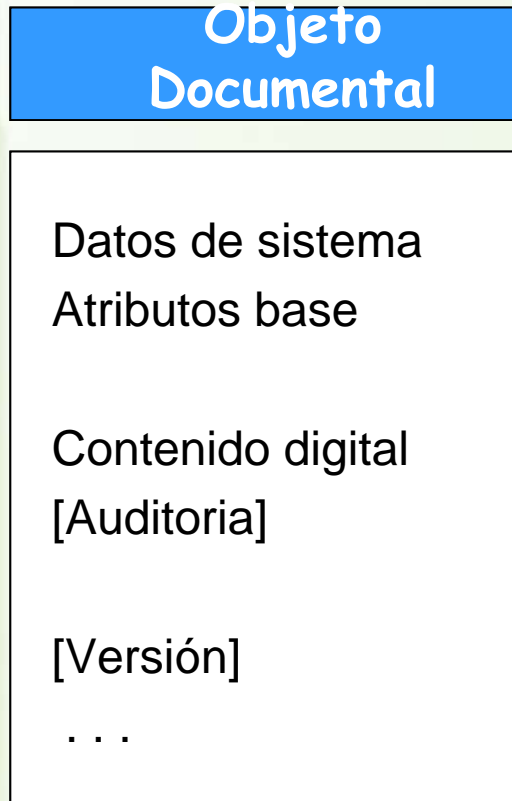
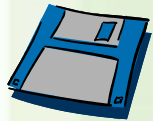
# Funcionalidades Básicas (I)

## • Caracterización de documentos

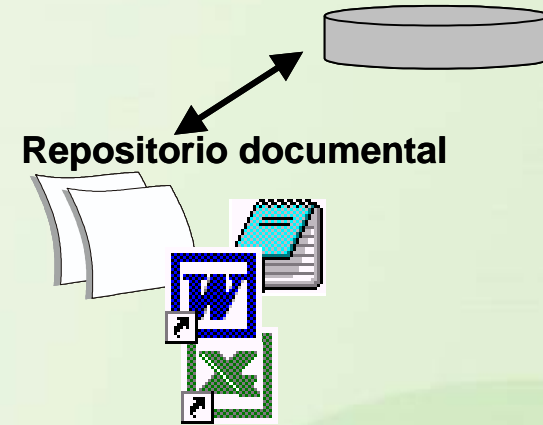


# Funcionalidades Básicas (II)

## • Objeto Documental



Acceso vía FTP,  
FileSystem,  
Api Centera,  
etc

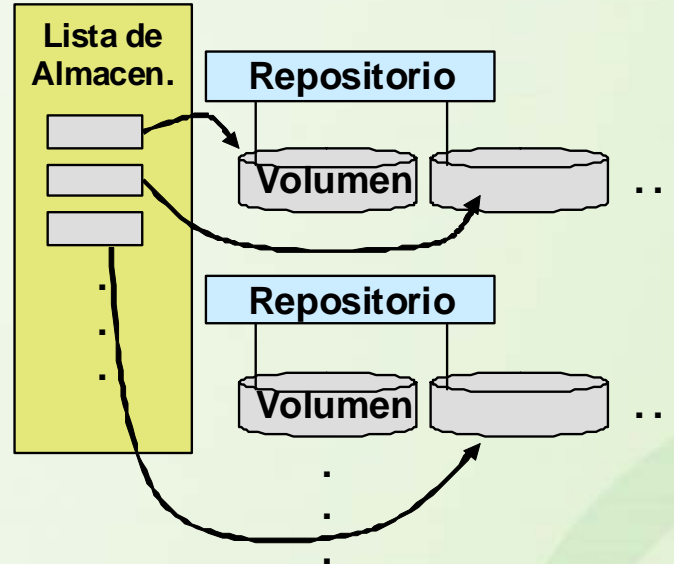
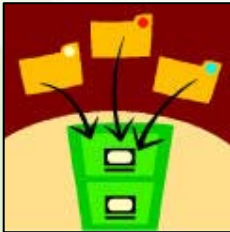




# Funcionalidades Básicas (III)

## • Almacenamiento de la Información

Archivador



- Múltiples sistemas de almacenamiento (FTP, PFS, ...)
- Múltiples ubicaciones (Servidores, Recursos de Red , iScsi, Nas, ...)



# Funcionalidades Básicas (y IV)

## • Utilización y administración

- Clientes Windows y Web (Servidores MS y Java)
  - Incluye visualizador para tipos no soportados nativamente por el cliente (TIFF, BMP.)
- API Windows (COM+) y Java
  - Construcción de Clientes “ad hoc”
  - Integración con aplicaciones (embebido)
- Herramientas de administración
  - Caracterización de la información (estructuras documentales)
  - Gestión del Almacenamiento
  - Gestión de Seguridad (usuarios y grupos, permisos, auditoria, ...)

# Arquitectura

## • Componentes

- Web Engine
- Servlet Engine (J2EE)
- Servidor de documentos (Unix, Linux, Windows,..)
- Servidor de base de datos y motor documental (Oracle, SQLServer, DB2 )
- Cliente Navegador de Internet (IE , Mozilla)
- Clientes 2 capas
- Aplicaciones consumidoras de Web Services.

# Configuración

- **Cliente w@rdA**
  - Desplegar la aplicación web (war)
  - Configurar el origen de datos (JDBC)
  - Configurar Struts
  - Configurar log4j
- **API Java**
  - Configurar el origen de datos (JDBC)
  - Configurar log4j

# Configuración JDBC

- **Fichero de Configuración de base de datos (/WEBINF/classes)**
- **leciTd\_DbConn\_Cfg.xml**

```
<Config>  
  <Pooling>Y</Pooling>  
  <DataSource>jdbc/oracle</DataSource>  
  <User></User>  
  <Password></Password>  
</Config>
```

- **Crear un DataSource de nombre 'jdbc/oracle' para el Contexto wardA o Global.**

- **En en el caso de que no se use un DataSource (pool de conexiones)**

```
<Config>  
  <Pooling>N</Pooling>  
  <DataSource></DataSource>  
  <Driver>oracle.jdbc.driver.OracleDriver</Driver>  
  <Url>jdbc:oracle:thin:@host:port:sid</Url>  
  <User>idoc891</User>  
  <Password>aWRvYzgz5MQ==</Password>  
</Config>
```

# Configuración Struts

- **Fichero de Configuración Struts (/WEBINF)**
- **struts-config.xml**

```
<controller maxFileSize="10M" tempDir="C:/WARDA_HOME/TMP/" />
<plug-in className="ieci.tecdoc.mvc.plugin.ConfigPlugin" >
<set-property property="httpServer" value="localhost" />
<set-property property="httpPort" value="70" />
<set-property property="system_config_name" value="IeciTd" />
<set-property property="ldap_config_name" value="" />
<set-property property="exception_handle" value="development" /> <!-- development, produccion -->
<set-property property="lockTimeout" value="30" /> <!-- Tiempo máximo de bloqueo de una carpeta -->
<set-property property="vldTablePageSize" value="10" /> <!-- Tamaño de la página de la tabla de validacion -->
<set-property property="searchResultsPageSize" value="17" />
<set-property property="archiveRelationPath" value="C:/WARDA_HOME/RELATIONS/" />
</plug-in>
```

# Configuración log4j

- Fichero de Configuración Struts (/WEBINF/classes)
- log4j.xml
- Configurar los 'appenders' con la ruta correcta y level de los logs

# Inicialización BD

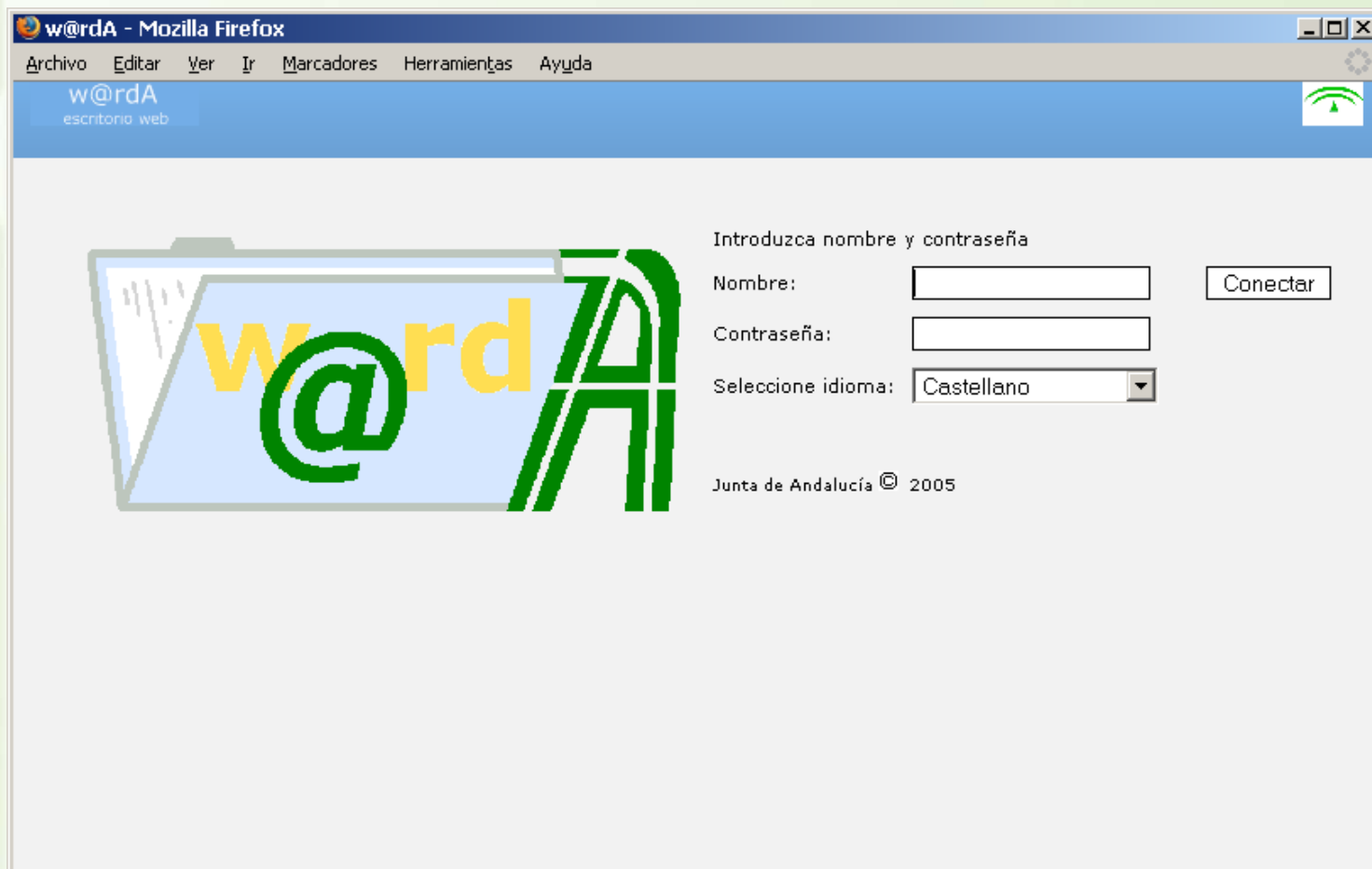
- **Script de creación e inicialización de tablas (Oracle)**
  - Estructuras básicas
  - Archivadores Iniciales (integración [w@ndA](#))
- **Aplicación de inicialización**
  - En la actualidad es una aplicación windows (ODBC)



# Cliente web w@rdA

- **Autenticación**
- **Seleccionar Archivadores**
- **Buscar Carpetas**
- **Consultar Carpetas**
  - Datos
  - Documentos
- **Modificar Carpetas**
  - Datos
  - Documentos
- **Añadir Carpetas**
  - Datos
  - Documentos
- **Eliminar Carpetas**

# Autenticación



w@rdA - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

w@rdA  
escritorio web

Introduzca nombre y contraseña

Nombre:

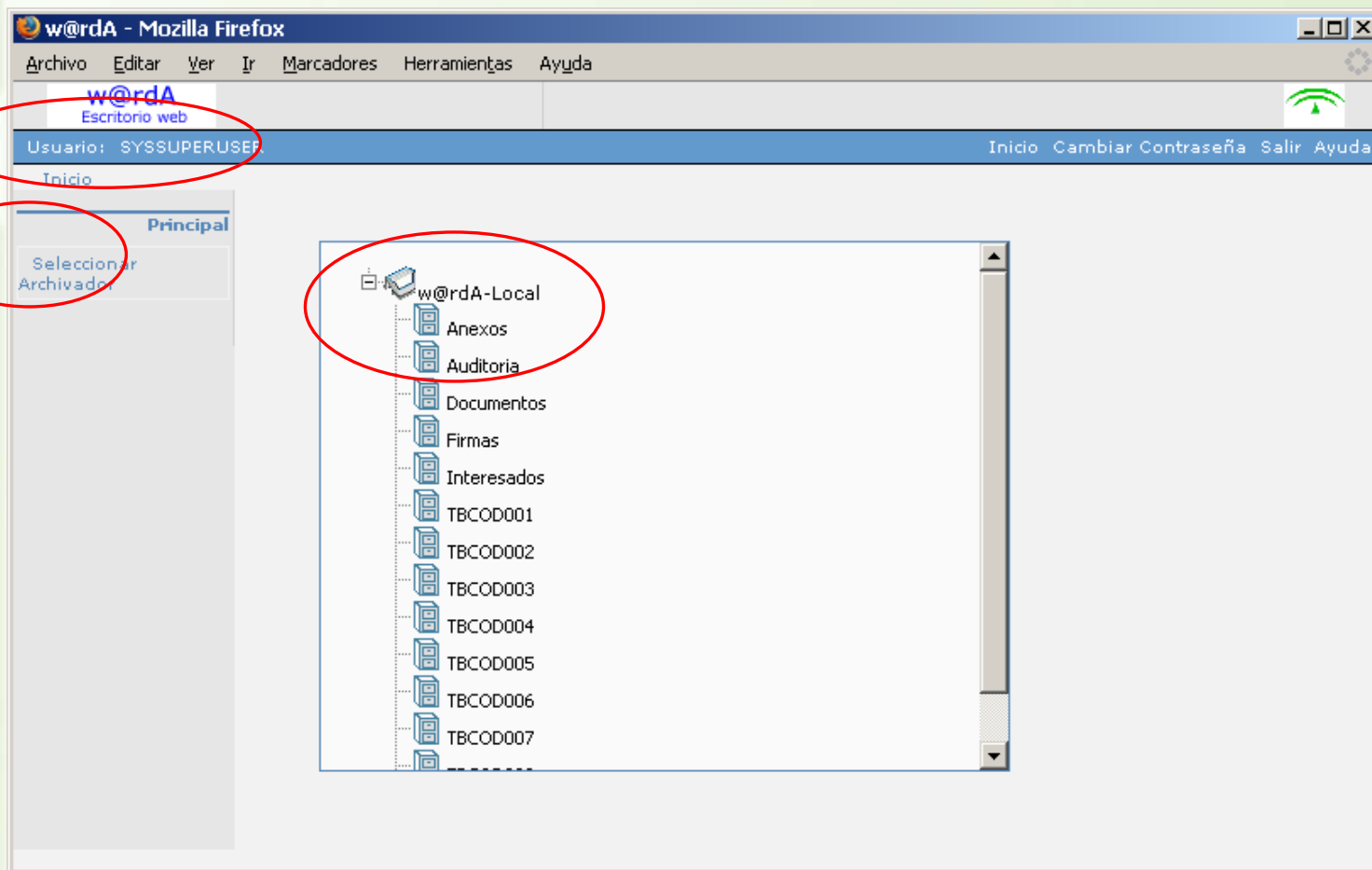
Contraseña:

Seleccione idioma:

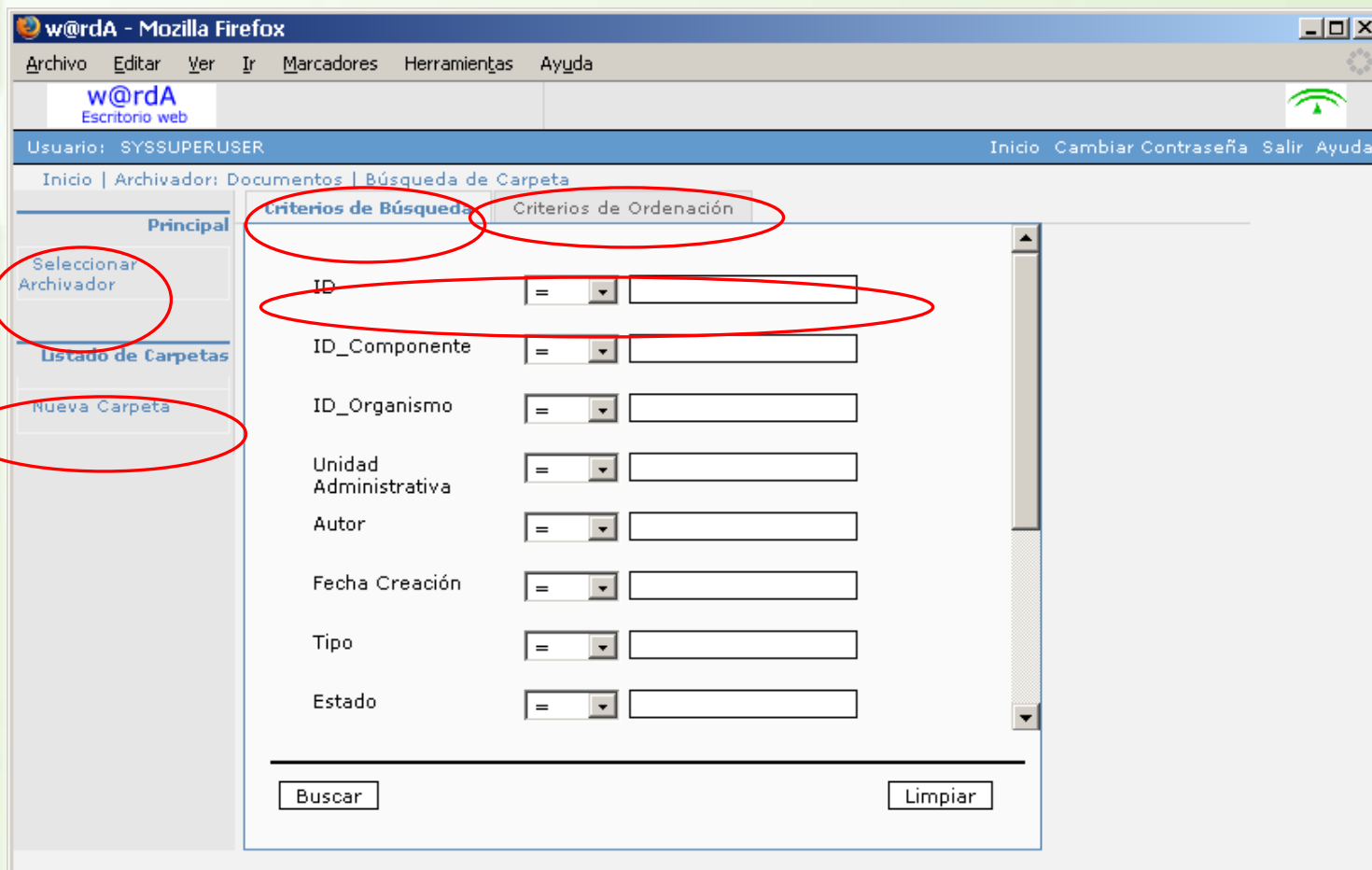
Conectar

Junta de Andalucía © 2005

# Archivadores



# Búsquedas



w@rdA - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

w@rdA  
Escritorio web

Usuario: SYSSUPERUSER Inicio Cambiar Contraseña Salir Ayuda

Inicio | Archivador: Documentos | Búsqueda de Carpeta

Criterios de Búsqueda Criterios de Ordenación

Principal

Seleccionar Archivador

Listado de Carpetas

Nueva Carpeta

ID = [dropdown] [input]

ID\_Componente = [dropdown] [input]

ID\_Organismo = [dropdown] [input]

Unidad Administrativa = [dropdown] [input]

Autor = [dropdown] [input]

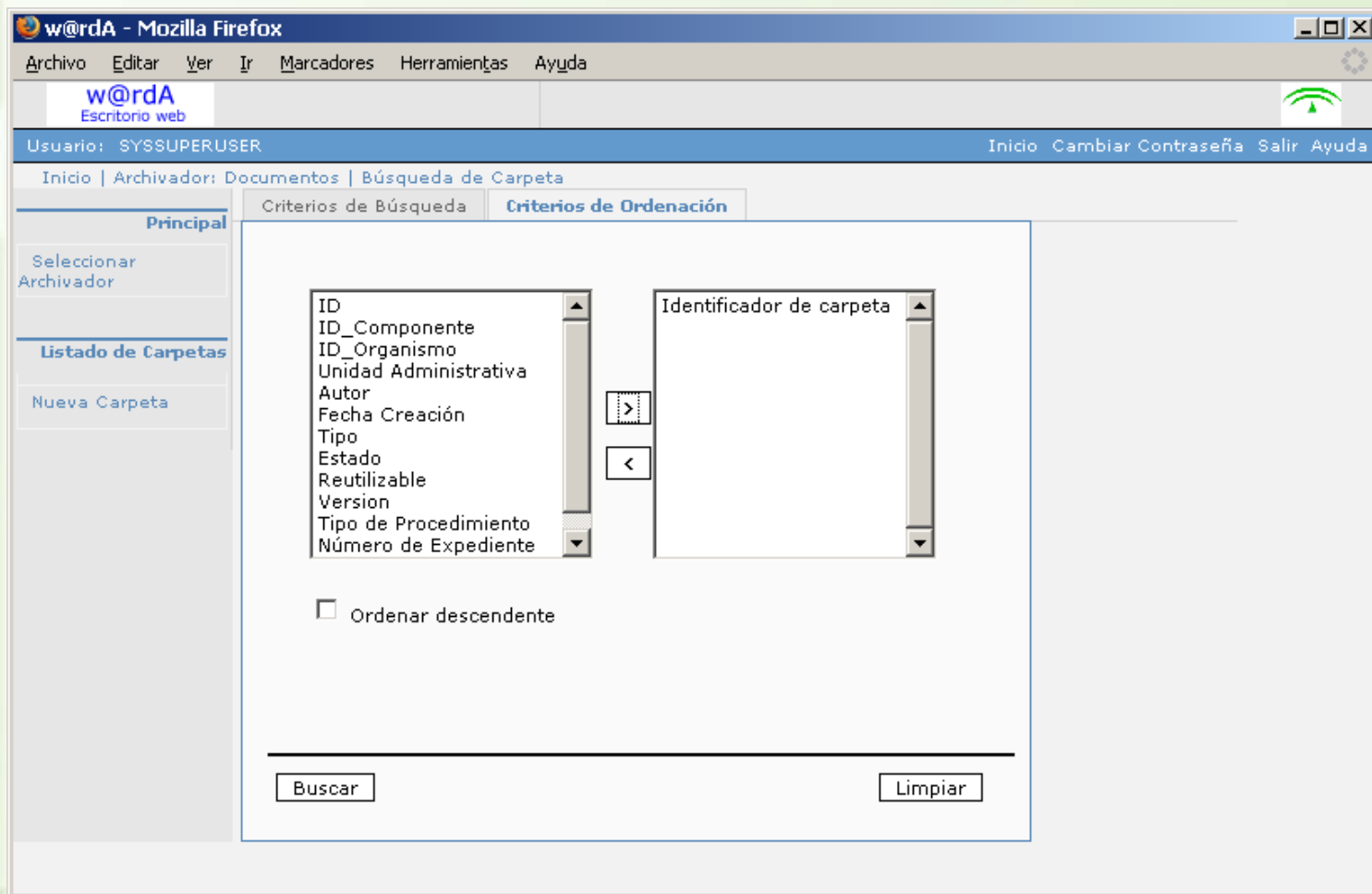
Fecha Creación = [dropdown] [input]

Tipo = [dropdown] [input]

Estado = [dropdown] [input]

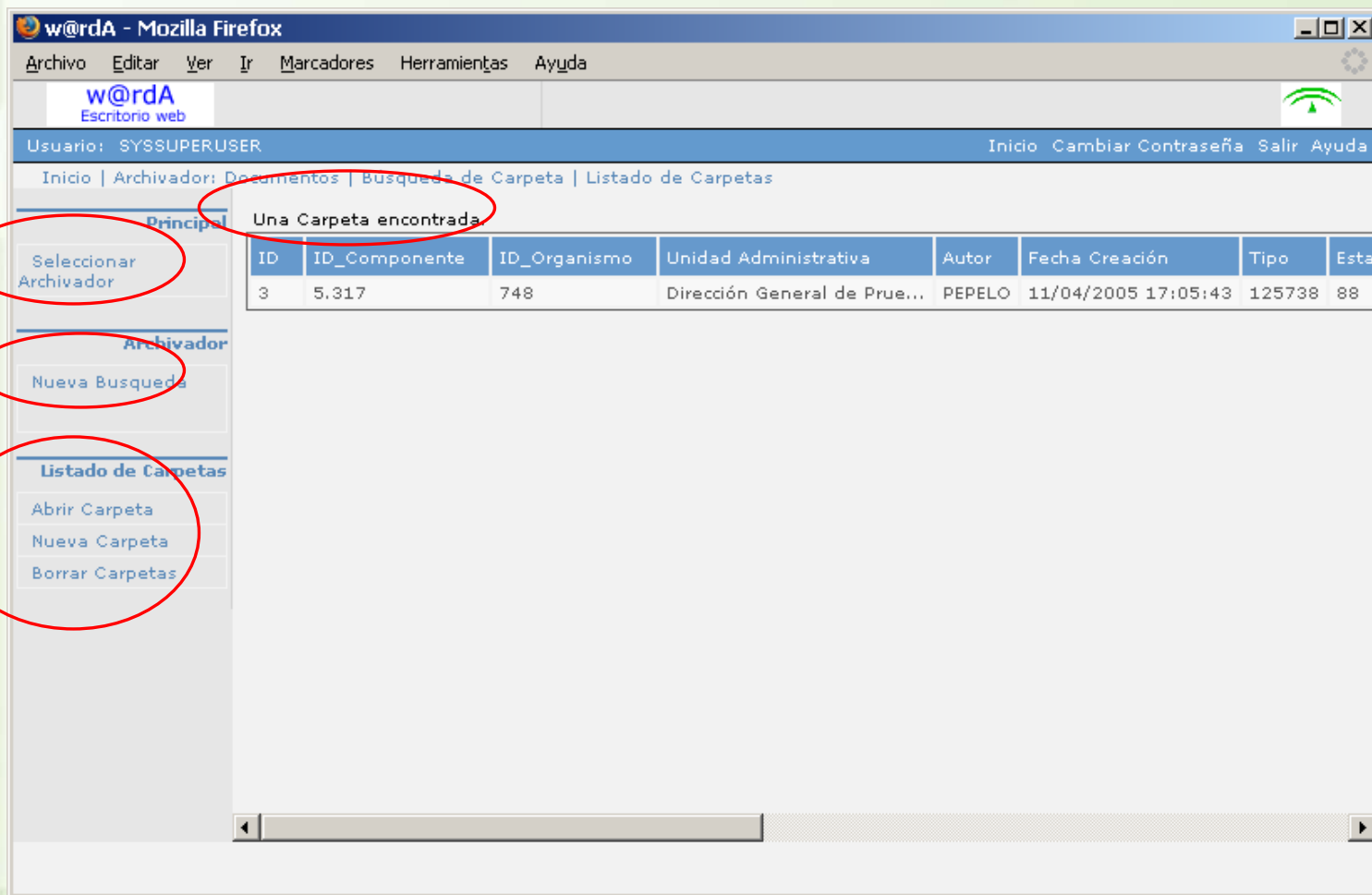
Buscar Limpiar

# Búsquedas



The screenshot shows a web browser window titled "w@rdA - Mozilla Firefox". The address bar contains "w@rdA Escritorio web". The user is logged in as "SYSSUPERUSER". The main content area is titled "Inicio | Archivador: Documentos | Búsqueda de Carpeta". It features two tabs: "Criterios de Búsqueda" and "Criterios de Ordenación", with the latter being active. The interface includes a sidebar with "Principal", "Seleccionar Archivador", "Listado de Carpetas", and "Nueva Carpeta". The main area contains two lists of search criteria: "ID", "ID\_Compone", "ID\_Organismo", "Unidad Administrativa", "Autor", "Fecha Creación", "Tipo", "Estado", "Reutilizable", "Version", "Tipo de Procedimiento", and "Número de Expediente". A "Identificador de carpeta" field is also present. Below the lists are buttons for ">" and "<". A checkbox for "Ordenar descendente" is unchecked. At the bottom, there are "Buscar" and "Limpiar" buttons.

# Resultados



w@rdA - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

w@rdA  
Escritorio web

Usuario: SYSSUPERUSER Inicio Cambiar Contraseña Salir Ayuda

Inicio | Archivador: Documentos | Busqueda de Carpeta | Listado de Carpetas

Una Carpeta encontrada

ID	ID_Componente	ID_Organismo	Unidad Administrativa	Autor	Fecha Creación	Tipo	Esta
3	5.317	748	Dirección General de Prue...	PEPELO	11/04/2005 17:05:43	125738	88

Seleccionar Archivador

Archivador

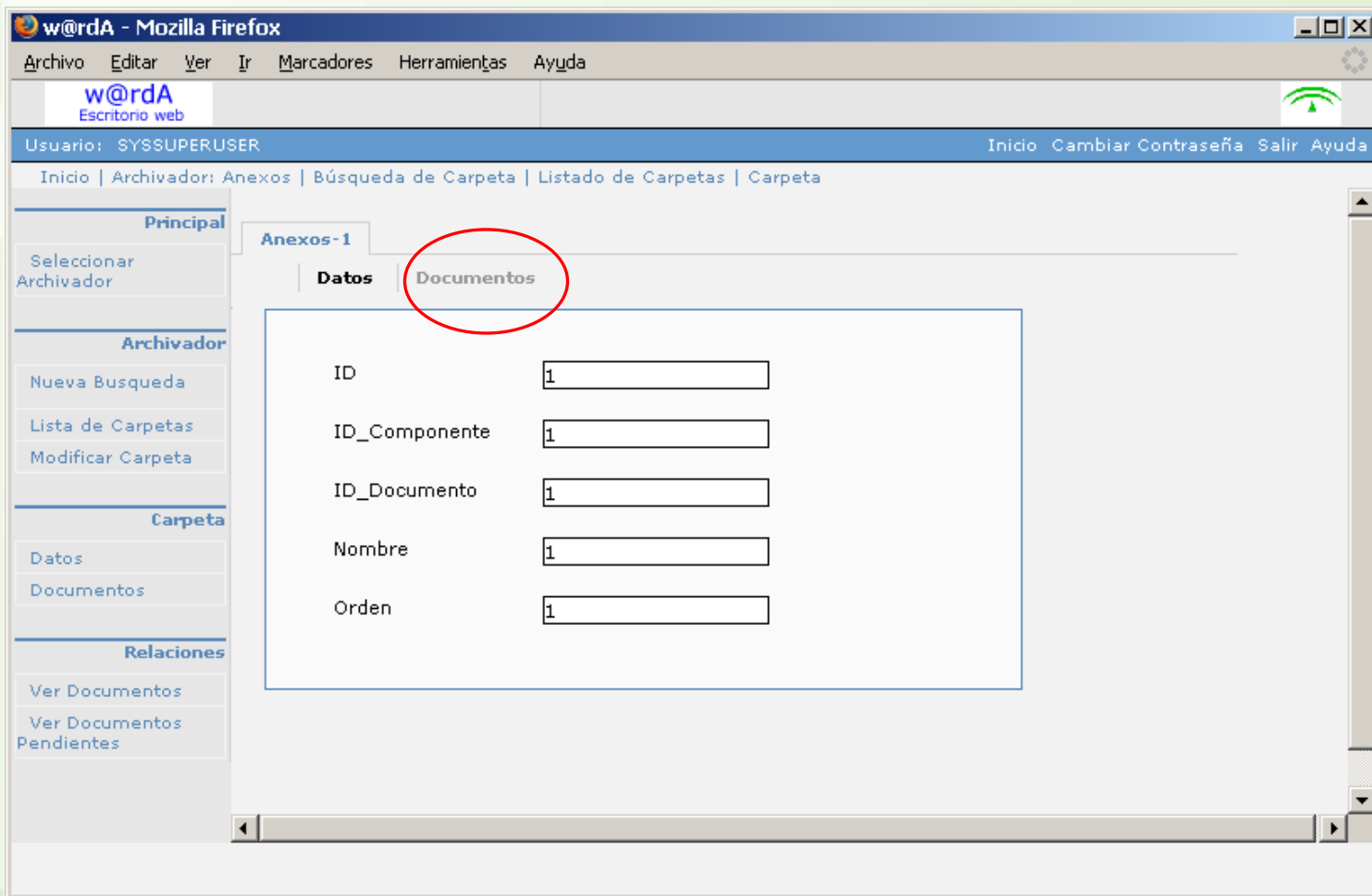
Nueva Busqueda

Listado de Carpetas

Abrir Carpeta

Nueva Carpeta

Borrar Carpetas



w@rdA - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

w@rdA  
Escritorio web

Usuario: SYSSUPERUSER Inicio Cambiar Contraseña Salir Ayuda

Inicio | Archivador: Anexos | Búsqueda de Carpeta | Listado de Carpetas | Carpeta

**Principal**

- Seleccionar Archivador

**Archivador**

- Nueva Búsqueda
- Lista de Carpetas
- Modificar Carpeta

**Carpeta**

- Datos
- Documentos**

**Relaciones**

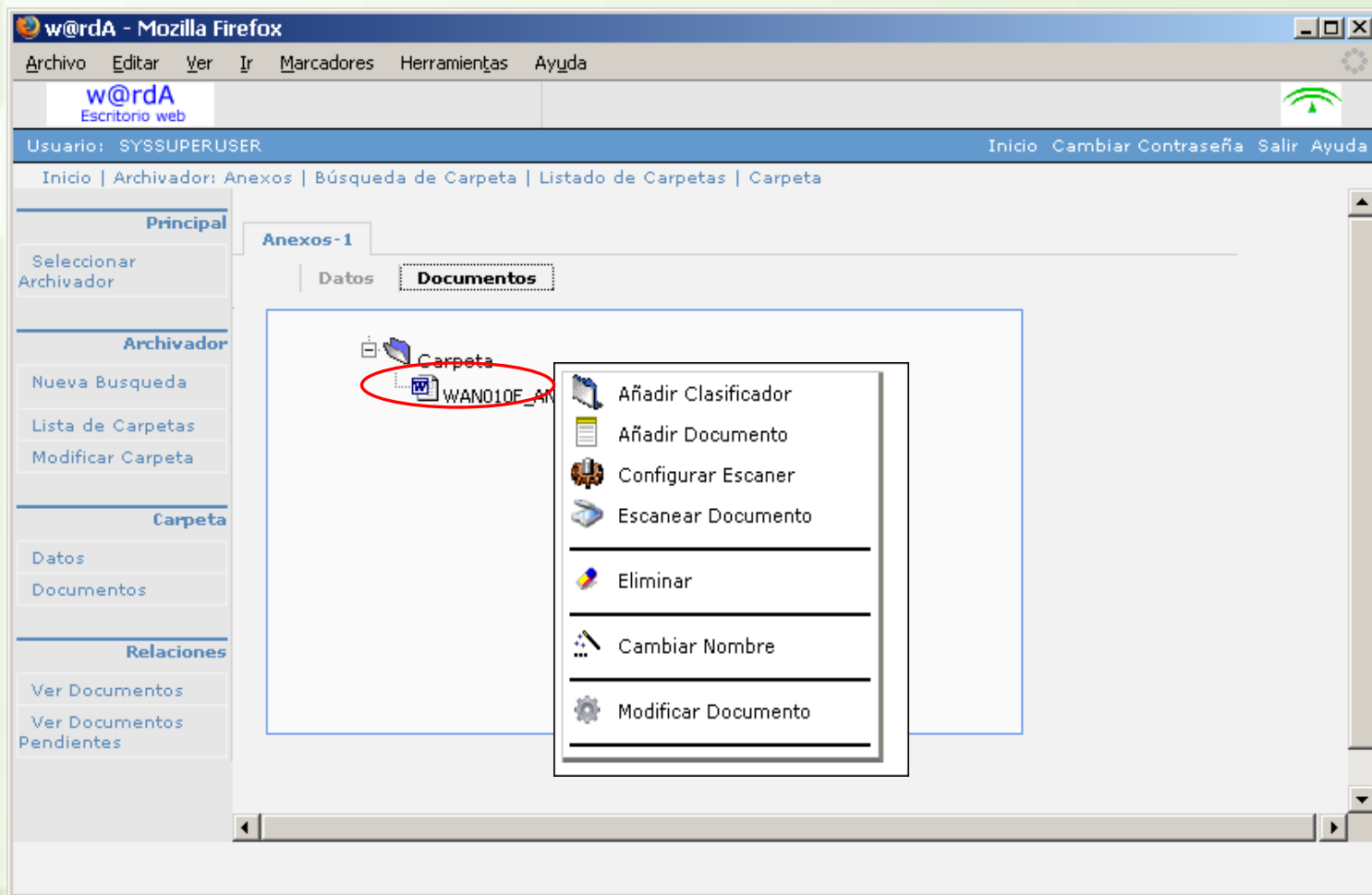
- Ver Documentos
- Ver Documentos Pendientes

**Anexos-1**

**Datos** **Documentos**

ID	<input type="text" value="1"/>
ID_Componente	<input type="text" value="1"/>
ID_Documento	<input type="text" value="1"/>
Nombre	<input type="text" value="1"/>
Orden	<input type="text" value="1"/>





w@rdA - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

w@rdA  
Escritorio web

Usuario: SYSSUPERUSER Inicio Cambiar Contraseña Salir Ayuda

Inicio | Archivador: Anexos | Búsqueda de Carpeta | Listado de Carpetas | Carpeta

**Principal**

- Seleccionar Archivador

**Archivador**

- Nueva Búsqueda
- Lista de Carpetas
- Modificar Carpeta

**Carpeta**

- Datos
- Documentos

**Relaciones**

- Ver Documentos
- Ver Documentos Pendientes

Anexos-1

Datos **Documentos**

Carpeta

WAN10E.M

- Añadir Clasificador
- Añadir Documento
- Configurar Escaner
- Escanear Documento
- Eliminar
- Cambiar Nombre
- Modificar Documento

# Prácticas

- **Autenticación**
- **Seleccionar Archivadores**
- **Buscar Carpetas**
- **Consultar Carpetas**
  - Datos
  - Documentos
- **Modificar Carpetas**
  - Datos
  - Documentos
- **Añadir Carpetas**
  - Datos
  - Documentos
- **Eliminar Carpetas**
- **<http://10.244.60.215:8080/wardACurso>**



## Componente Documental w@ndA

### Jornada 2

- **Índice**

- **Introducción a la Administración w@rdA**
- **Administración de Usuarios (ACS)**
- **Administración de Almacenamiento**
- **Administración de Archivadores**
- **Casos Prácticos**

# Introducción

- **El orden lógico de administración es:**
  - **Administración de usuarios**
    - **Definición de Departamentos, Usuarios y Grupos**
      - **Asignación de Perfiles y Derechos**
  - **Administración del Almacenamiento**
    - **Definición de Repositorios**
    - **Definición de Volúmenes**
    - **Definición de Listas**
  - **Administración de los Archivadores**
    - **Definición de Archivadores**
    - **Asignación de Permisos**

# Conceptos del 'ACS'

- **Departamentos:**
  - Representa la estructura organizativa
  - Si w@rdA está configurado para trabajar con LDAP no existe equivalencia
  - Estructura Jerárquica
  - Es posible asignar 'Administradores' a Departamentos
- **Usuarios:**
  - Los usuarios solo pueden pertenecer a un departamento
  - En el caso de LDAP los usuarios 'cuelgan' de Unidades Organizativas
- **Grupos:**
  - Representan agrupaciones funcionales no organizativas
  - La equivalencia en LDAP son los 'groupOfUniqueNames'. En el LDAP de la J.A. No hay definidos ninguno.
  - Un usuario puede pertenecer a mas de un grupo.

# Departamentos

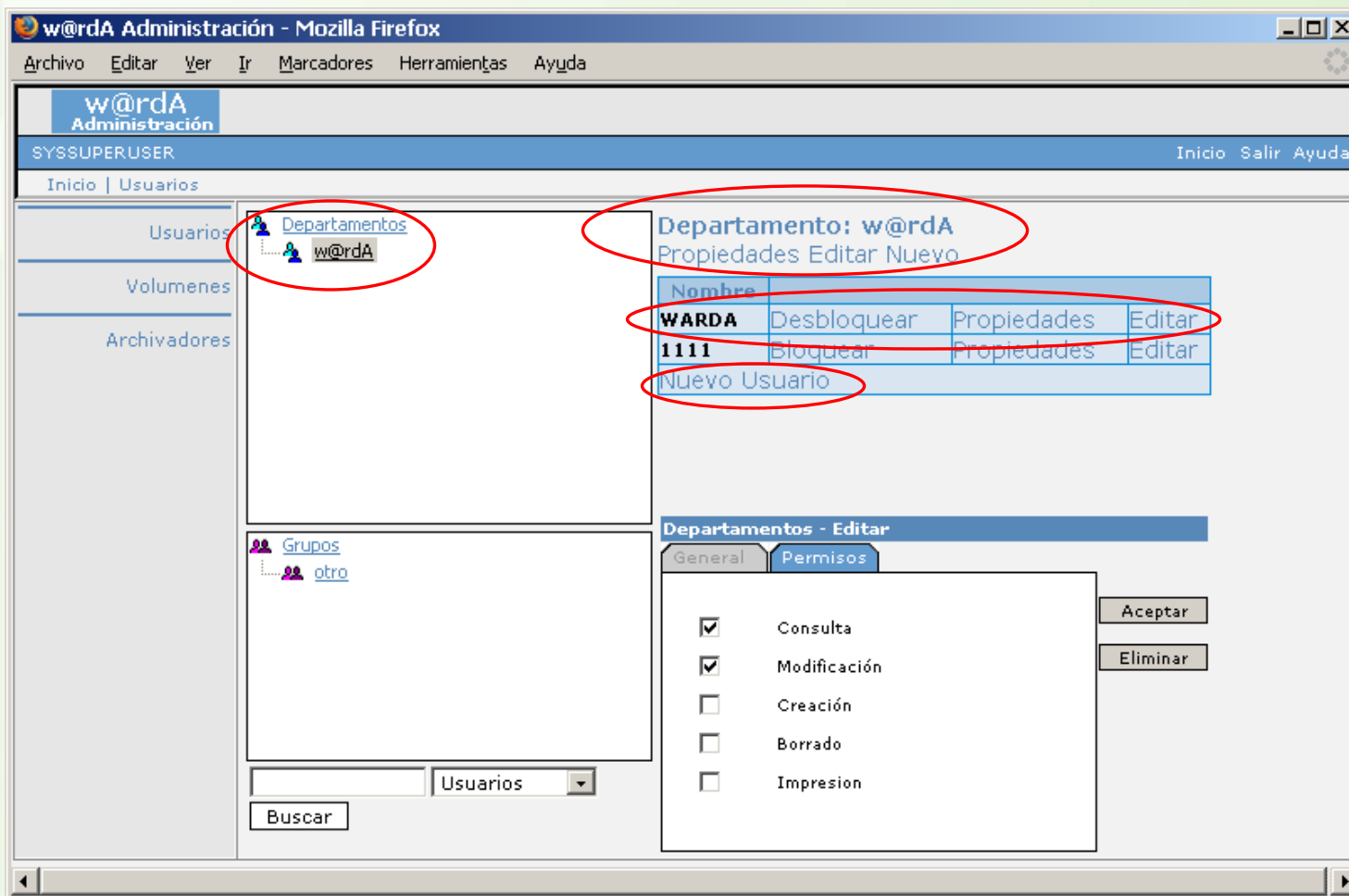
- **Atributos básicos**
  - ID (interno de utilidad en el API)
  - Nombre
  - Descripción
  - Administrador
- **Atributos de autoría (usuario y fecha)**
  - Creación
  - Modificación
- **Permisos Genéricos**

Departamento - Propiedades	
General	
Nombre:	w@rdA
Id:	1
Administrador:	SYSSUPERUSER
-----	
Descripción	<input type="text"/>
-----	
Creado:	20-4-2005 por SYSSUPERUSER
Modificado:	16-6-2005 por SYSSUPERUSER

Departamentos - Editar	
General	Permisos
<input checked="" type="checkbox"/>	Consulta
<input checked="" type="checkbox"/>	Modificación
<input type="checkbox"/>	Creación
<input type="checkbox"/>	Borrado
<input type="checkbox"/>	Impresión



# Departamentos



w@rdA Administración - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

w@rdA Administración

SYSSUPERUSER Inicio Salir Ayuda

Inicio | Usuarios

Usuarios

Departamentos

w@rdA

Volumenes

Archivadores

Grupos

otro

Departamento: w@rdA

Propiedades Editar Nuevo

Nombre			
WARDA	Desbloquear	Propiedades	Editar
1111	Bloquear	Propiedades	Editar
Nuevo Usuario			

Departamentos - Editar

General Permisos

Consulta

Modificación

Creación

Borrado

Impresión

Aceptar

Eliminar

Usuarios

Buscar

- **Atributos básicos**
  - ID (interno de utilidad en el API)
  - Nombre
  - Descripción
  - Estado
  - Contraseña
    - Política de contraseñas
- **Atributos de autoría (usuario y fecha)**
  - Creación
  - Modificación
- **Perfiles**
  - En los distintos módulos de w@rdA
- **Permisos Genéricos**

- **Sistema**
  - Superusuario
- **Archivadores**
  - Superusuario
  - Administrador
  - Estándar (cliente)
  - Sin derechos
- **Usuarios**
  - Superusuario
  - Administrador
  - Estándar (solo consulta)
  - Sin derechos
- **Almacenamiento**
  - Superusuario

# Resumen

**Usuario - Editar**

General | Perfiles | Permisos | Grupos

Nombre:

Contraseña:

Confirmación:

Inicialización obligatoria de la contraseña

Comprobar caducidad de la contraseña

Descripción:

General | **Perfiles** | Permisos | Grupos

Contexto:

▼

- Sistema
- Administración de Usuarios
- Invesdoc
- Administrador de volúmenes
- Administrador

Estándar

Sin derechos

**Usuario - Editar**

General | Perfiles | **Permisos** | Grupos

Consulta

Modificación

Creación

Borrado

Impresión

General | Perfiles | Permisos | **Grupos**

Grupo 2

Grupo 1



- **Atributos básicos**
  - ID (interno de utilidad en el API)
  - Nombre
  - Descripción
- **Atributos de autoría (usuario y fecha)**
  - Creación
  - Modificación
- **Permisos Genéricos**

# Almacenamiento

- Repositorios
  - Servidores
- Volúmenes
  - Directorios o carpetas dentro de los servidores
- Listas
  - Relación de Volúmenes.
  - Una lista puede contener volúmenes de distintos repositorios

# Repositorios

- **Atributos básicos**
  - ID (interno de utilidad en el API)
  - Nombre
  - Tipo (PFS o FTP)
  - Descripción
- **Atributos de autoría (usuario y fecha)**
  - Creación
  - Modificación
- **Detalles**
  - Ruta de acceso
  - Estado
    - Disponible
    - Solo lectura

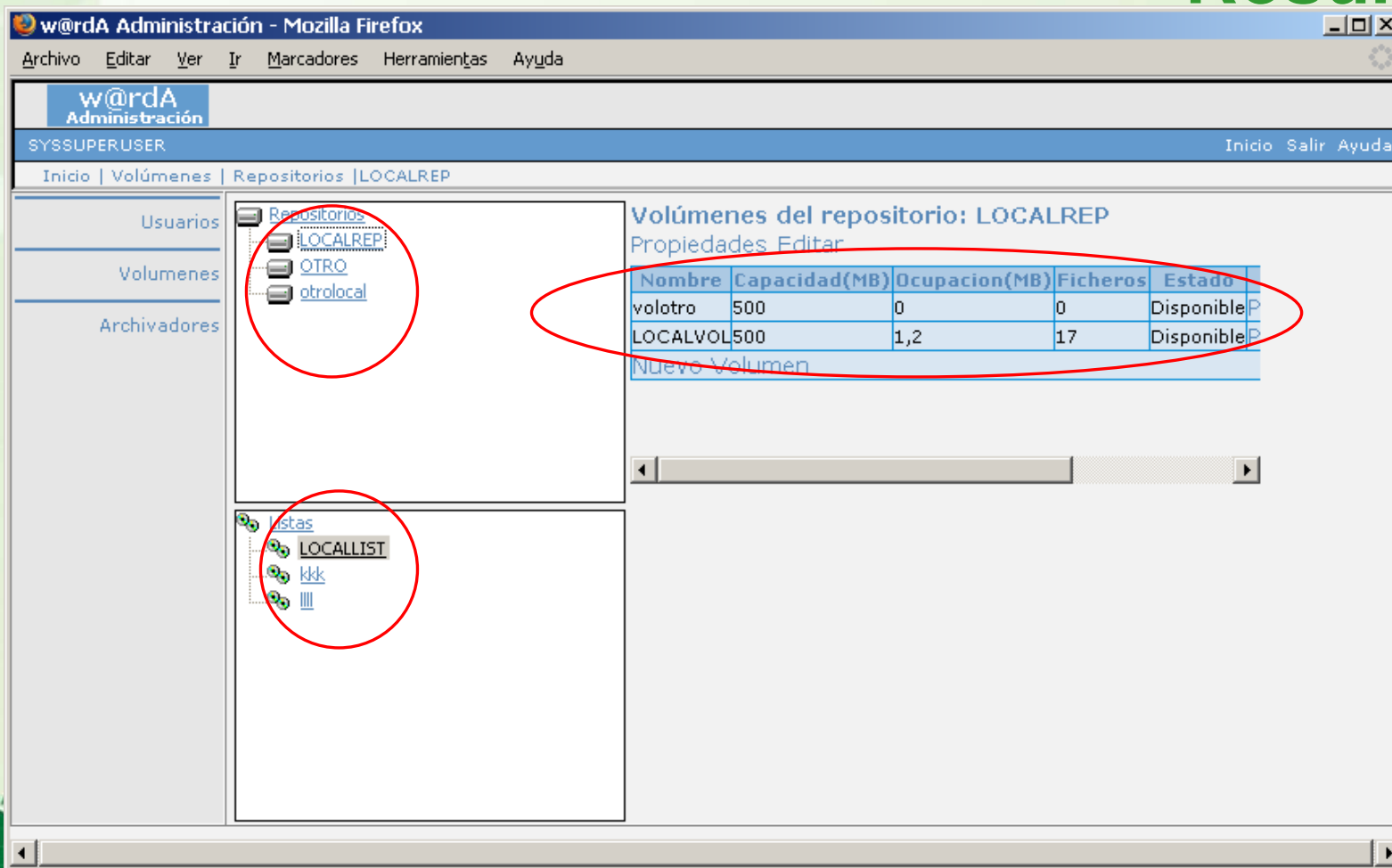


# Volúmenes

- **Atributos básicos**
  - ID (interno de utilidad en el API)
  - Nombre
  - Repositorio
  - Descripción
- **Atributos de autoría (usuario y fecha)**
  - Creación
  - Modificación
- **Detalles**
  - Listas
  - Ruta
  - Capacidad, Ocupación y Archivos
  - Estado
    - Disponible
    - Solo lectura
    - Lleno

- **Atributos básicos**
  - ID (interno de utilidad en el API)
  - Nombre
  - Descripción
- **Atributos de autoría (usuario y fecha)**
  - Creación
  - Modificación

# Resumen



w@rdA Administración - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

w@rdA Administración

SYSSUPERUSER Inicio Salir Ayuda

Inicio | Volúmenes | Repositorios | LOCALREP

Usuarios

Volúmenes

Archivadores

Repositorios

- LOCALREP
- OTRO
- otrolocal

Volúmenes del repositorio: LOCALREP

Propiedades Editar

Nombre	Capacidad(MB)	Ocupacion(MB)	Ficheros	Estado
volotro	500	0	0	Disponible
LOCALVOL	500	1,2	17	Disponible

Nuevo Volumen

Listas

- LOCALLIST
- kkk
- lll

# Archivadores

- **Directorios**
  - Su única utilidad es la de organizar los archivadores
- **Archivadores**
  - Entidad donde es almacenada la información en w@rdA.
  - Suelen contener información homogénea
  - Tienen asociado:
    - Un modelo de datos
    - Una lista de almacenamiento
    - Una configuración de acceso (permisos)

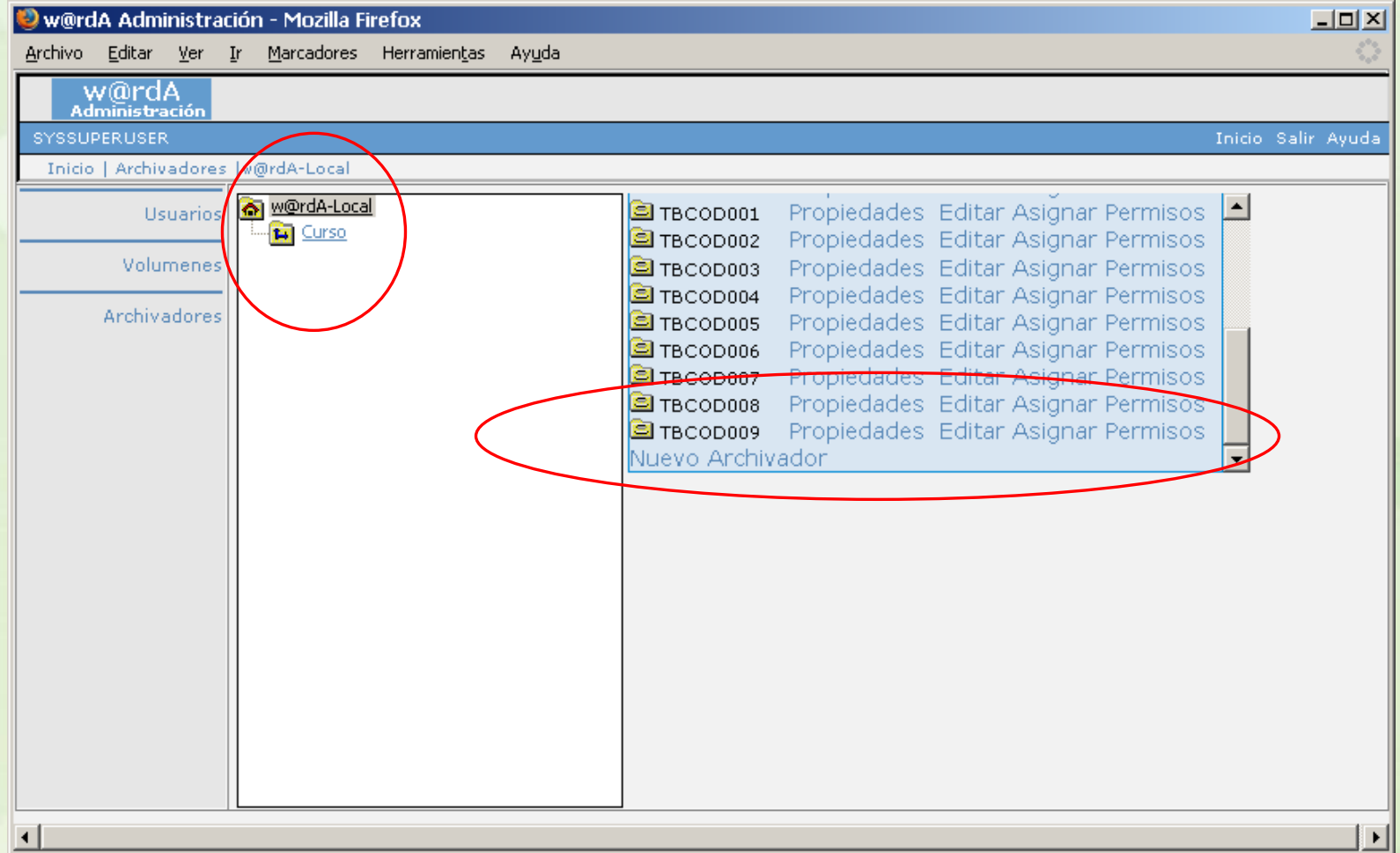
# Directorios

- **Atributos básicos**
  - ID (interno de utilidad en el API)
  - Nombre
  - Descripción
- **Atributos de autoría (usuario y fecha)**
  - Creación
  - Modificación

# Archivadores

- **Atributos básicos**
  - ID (interno de utilidad en el API)
  - Nombre
  - Descripción
- **Atributos de autoría (usuario y fecha)**
  - Creación
  - Modificación
- **Modelo de datos**
  - Campos
    - Nombre
    - Tipo
    - Longitud
    - Descripción
    - Otros
  - Lista de Almacenamiento

# Resumen(I)



w@rdA Administración - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

w@rdA Administración

SYSSUPERUSER Inicio Salir Ayuda

Inicio | Archivadores | w@rdA-Local

Usuarios

Volumenes

Archivadores

w@rdA-Local

Curso

TBCOD001	Propiedades	Editar	Asignar	Permisos
TBCOD002	Propiedades	Editar	Asignar	Permisos
TBCOD003	Propiedades	Editar	Asignar	Permisos
TBCOD004	Propiedades	Editar	Asignar	Permisos
TBCOD005	Propiedades	Editar	Asignar	Permisos
TBCOD006	Propiedades	Editar	Asignar	Permisos
TBCOD007	Propiedades	Editar	Asignar	Permisos
TBCOD008	Propiedades	Editar	Asignar	Permisos
TBCOD009	Propiedades	Editar	Asignar	Permisos
Nuevo Archivador				





# Resumen(II)

http://localhost:70 - Mozilla Firefox

Archivadores - Editar

General Campos Indices Carpetas

Nombre:

Descripción:

Administrador:  Lista de volúmenes Asociada:

Aceptar Cancelar Eliminar

http://localhost:70 - Mozilla Firefox

Archivadores - Editar

General Campos Indices Carpetas

Campos

Nombre	Tipo	Longitud
ID	Entero largo	-
ID_Componte	Entero largo	-
ID_Organismo	Texto	16
Unidad Administrativa	Texto	150

Detalle

Aceptar Cancelar Eliminar

http://localhost:70 - Mozilla Firefox

Archivadores - Editar

General Campos Indices Carpetas

Definición

Nombre	Unico
ID_DOCUMENTO	<input checked="" type="checkbox"/>

Añadir Índice

Campos

Nombre	Tipo	Longitud
<input checked="" type="checkbox"/> ID	Entero largo	-
<input checked="" type="checkbox"/> ID_Componte	Entero largo	-
<input type="checkbox"/> ID_Organismo	Texto	16

Aceptar Cancelar Eliminar

http://localhost:70 - Mozilla Firefox

Archivadores - Editar

General Campos Indices Carpetas

Campos - Selección

Nombre	Tipo	Longitud
<input type="checkbox"/> Id. de carpeta	-	-
<input type="checkbox"/> ID	Entero largo	-
<input type="checkbox"/> ID_Componte	Entero largo	-
<input type="checkbox"/> ID_Organismo	Texto	16
<input type="checkbox"/> Unidad Administrativa	Texto	150
<input type="checkbox"/> Autor	Texto	30
<input type="checkbox"/> Fecha Creación	Fecha y hora	-

Identificación

Aceptar Cancelar Eliminar

# Prácticas(I)

- **Crear un departamento**
  - **Nombre:** Iniciales de cada usuario+\_dpt
  - **Descripción:** Libre
  - **Permisos:** Todos
- **Crear un usuario ‘estandar’:**
  - **Nombre:** Iniciales+\_user
  - **Descripción , contraseña:** Libres
  - **Sin configuración de contraseñas**
  - **Perfiles:** Defecto
  - **Permisos:** Todos
- **Crear un usuario ‘administrador’:**
  - **Nombre:** Iniciales+\_admin
  - **Descripción , contraseña:** Libres
  - **Sin configuración de contraseñas**
  - **Perfiles:** Máximos

# Prácticas(II)

- **Crear un Grupo**
  - **Nombre:** Iniciales de cada usuario+\_grp
  - **Descripción:** Libre
  - **Permisos:** Todos
  - **Asociar el usuario 'estándar' propio al grupo**
- **Crear un repositorio**
  - **Tipo 'PFS'**
  - **Nombre:** Iniciales de cada usuario+\_rep
  - **Ruta de acceso 'c:\ Iniciales de cada usuario+\_rep'**
- **Crear un volumen**
  - **Perteneciente al Repositorio anterior**
  - **Tamaño: 100 Mb**

# Prácticas(III)

- **Crear un directorio**
  - **Nombre: Iniciales+\_dir**
- **Crear un archivador**
  - **Nombre: Iniciales+\_arch**
  - **Campos: Libre**
  - **Índices: Libre**
- **Asignarle al usuario propio xxxx\_usr todos los derechos sobre el archivador**
- **Conectarse al cliente y probar su funcionamiento**
- **Hacer al usuario propio xxx\_usr administrador del departamento xxx\_dpto**
- **Conectarse a la administración y probar el nuevo administrador del departamento.**



# Servicios web w@rdA

Curso para desarrolladores. Parte 3

# Índice

- Descripción del sistema w@rdA
- Configuración del servicio web
- WSDL
- Tipos de datos complejos
- Operaciones de los servicios web
- Mensajes de error
- Caso práctico: invocación desde un cliente Java

# Descripción del sistema w@rdA

## Objetivo

**Ser el almacén de los documentos que participan en la tramitación, tanto los generados por el motor de tramitación durante la ejecución de los trámites, como los documentos generados externamente y que se anexan a los trámites de los procedimientos.**



# Descripción del sistema w@rdA (y 2)

## Dos partes diferenciadas

- Núcleo de almacenamiento y gestión propiamente dicho (base de datos y repositorios).
- Interfaz de acceso desde el sistema w@ndA (servicios web).

# Descripción del sistema w@rdA (y 3)

## w@ndA

Otros componentes w@ndA

## w@rdA

Interfaz de acceso (servicios web)

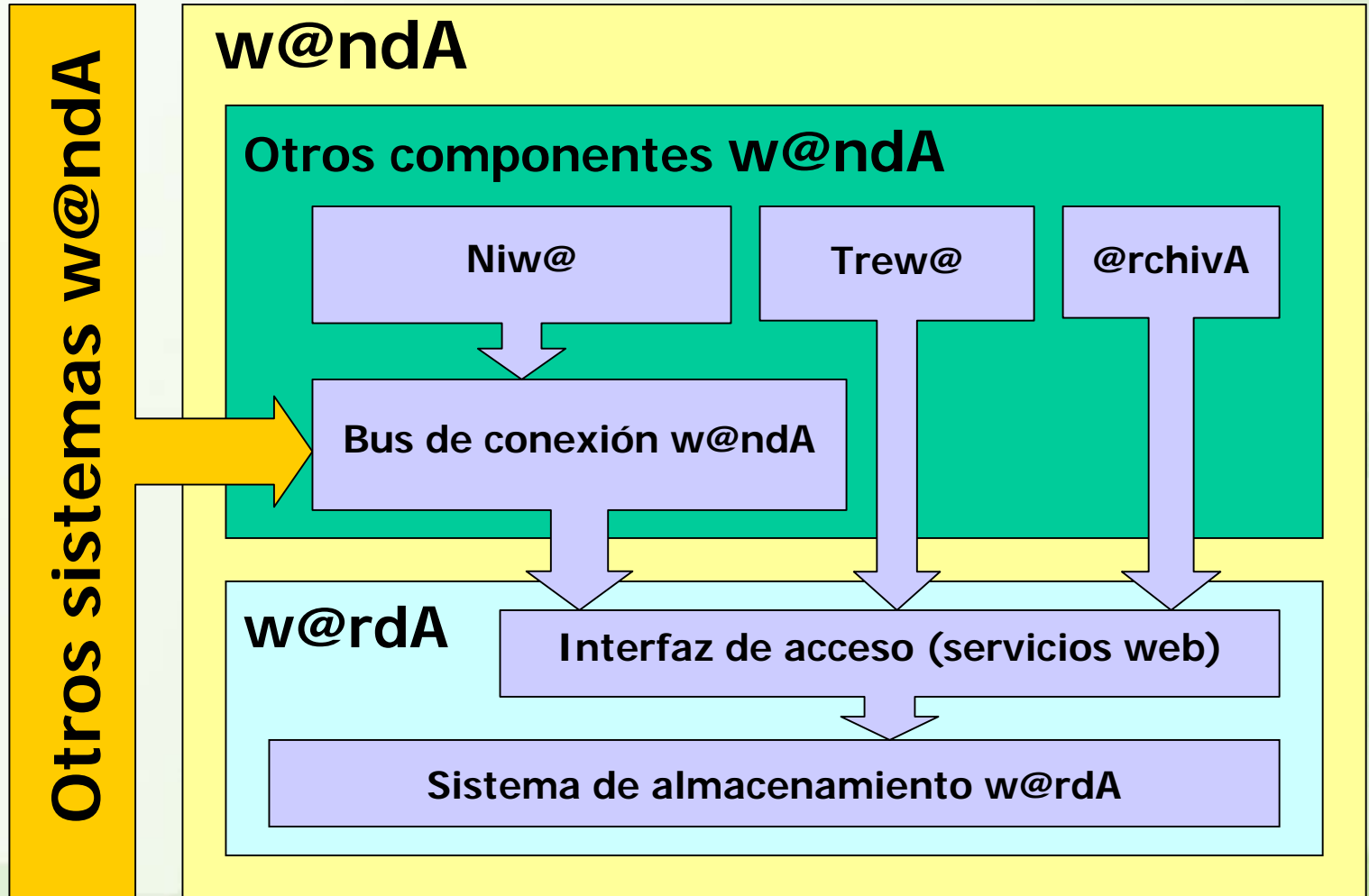
Sistema de almacenamiento  
(componente documental)

# Descripción del sistema w@rdA (y 4)

## Acceso a los datos del componente documental

- Los componentes de un mismo sistema w@ndA accederán bien directamente, o bien a través del bus de conexión, a los servicios web w@rdA.
- Los componentes de otros sistemas w@ndA tendrán que acceder forzosamente a través del bus de conexión.

# Descripción del sistema w@rdA (y 5)

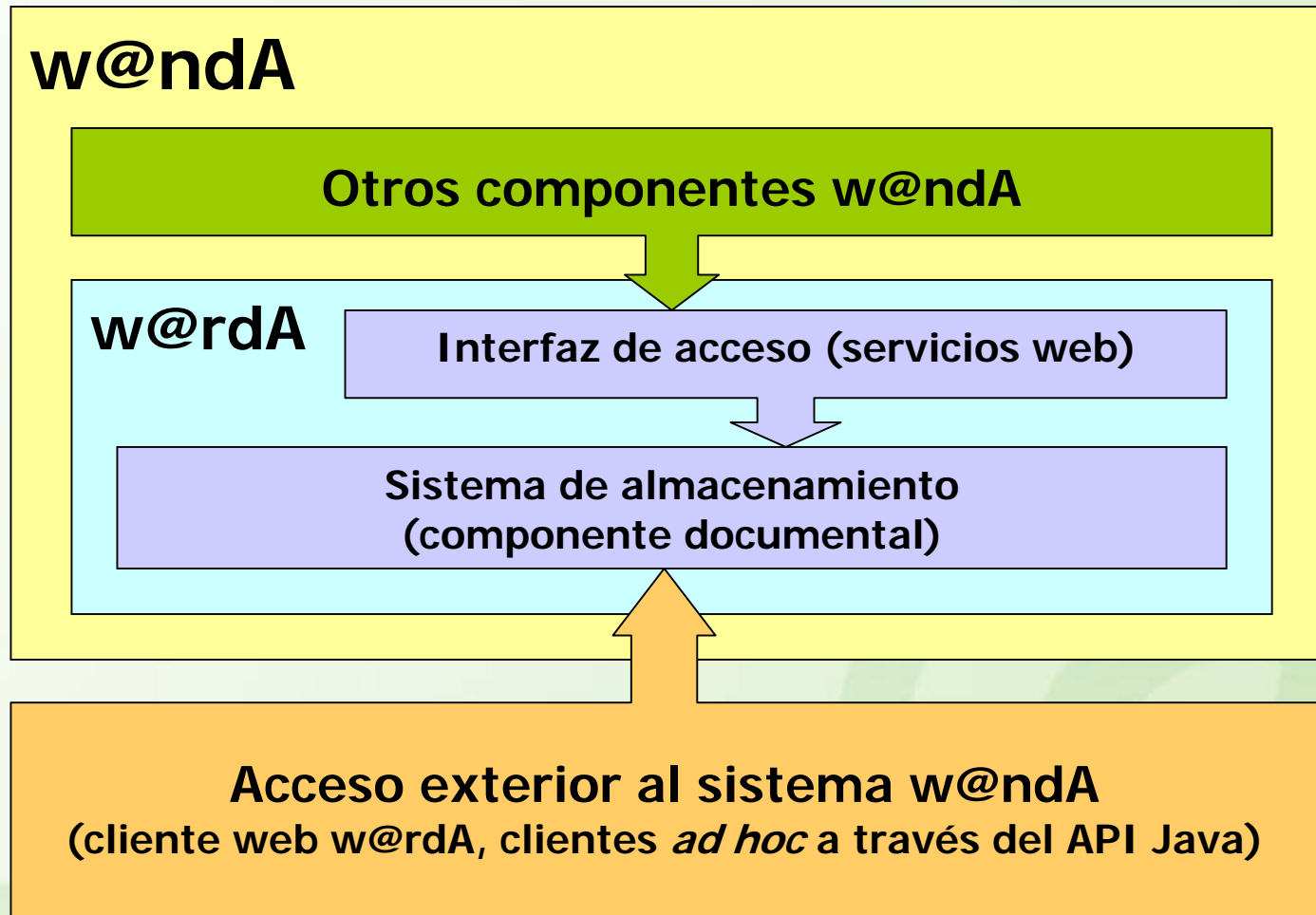


# Descripción del sistema w@rdA (y 6)

## Acceso a los datos del componente documental

- Cliente web w@rdA.
- Otros clientes web, clientes Java pesados, servlets, etc, desarrollados haciendo uso del API Java del componente.

# Descripción del sistema w@rdA (y 7)



# Descripción del sistema w@rdA (y 8)

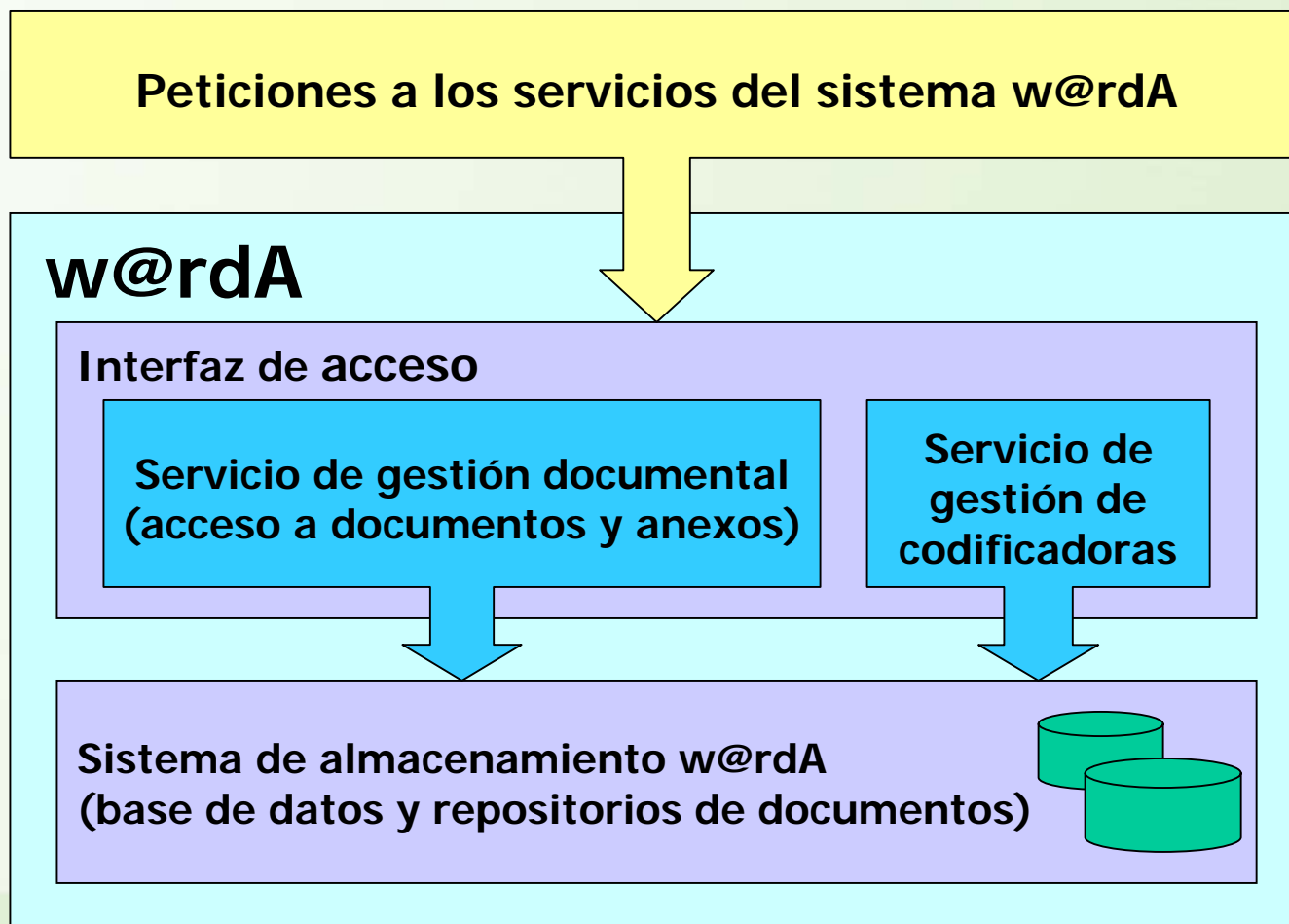
## Servicios web w@rdA

Componen la interfaz de acceso al sistema de gestión documental w@rdA desde las demás aplicaciones w@ndA:

- Servicio de gestión documental.
- Servicio de gestión de tablas.



# Descripción del sistema w@rdA (y 9)



# Modelo de datos w@rdA para w@ndA

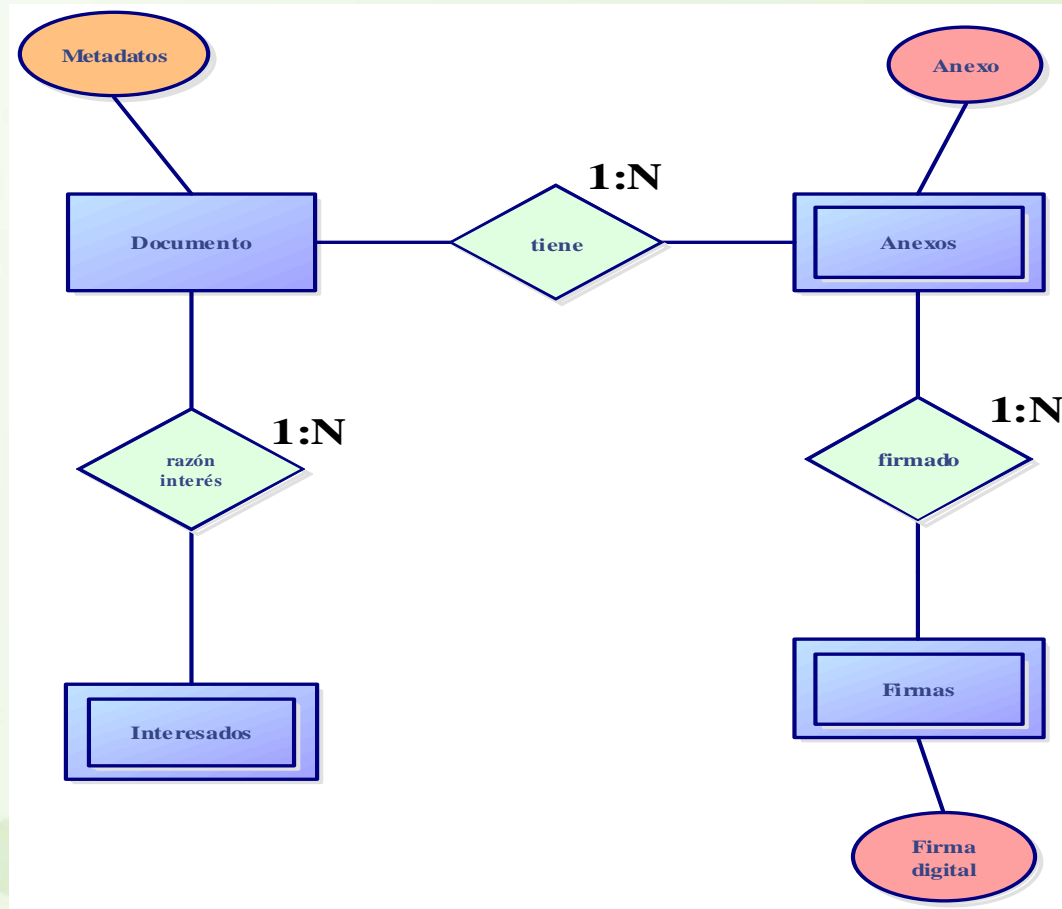
## Entidades para w@ndA

- **Documentos:** información de cabecera, o metadatos, de un documento.
- **Anexos:** metadatos y datos del anexo, propiamente dicho.
- **Firmas:** datos de la firma, y firma electrónica de un anexo.
- **Interesados:** datos básicos de interesados en los documentos.



# Mod. de datos w@rdA para w@ndA (y 2)

## Modelo E-R w@rdA para w@ndA



# Mod. de datos w@rdA para w@ndA (y 3)

## Archivadores para w@ndA

Un archivador por cada entidad:

- Documentos
- Anexos
- Firmas
- Interesados

Y uno por tabla codificadora (o maestra, si hubiere)

# WSDL

**WSDL: *Web Services Description Language*.**  
Lenguaje de descripción de servicios web.  
Estándar adoptado por el W3C.

## Documentos WSDL ¿qué son?

Documentos XML que cumplen Schema XML (no DTD):

- Describen el API pública de la interfaz w@rdA
- Uno por cada servicio web w@rdA
  - Servicio de gestión documental (wardA)
  - Servicio de gestión de tablas (codetables)

# WSDL (y 2)

## ¿Cómo obtener los documentos WSDL actualizados?

- Servicio de gestión documental

[http://\[host\]:\[port\]/wardAService/services/wardA?WSDL](http://[host]:[port]/wardAService/services/wardA?WSDL)

- Servicio de gestión documental (codetables)

[http://\[host\]:\[port\]/wardAService/services/codetables?WSDL](http://[host]:[port]/wardAService/services/codetables?WSDL)

donde:

host: IP o nombre del host servidor

port: puerto de escucha del servidor web

# Tipos de datos complejos

Los tipos de datos usados en los mensajes de entrada y salida de las operaciones (funciones o métodos) de los servicios.

Hay cierto paralelismo con las entidades del modelo de datos:

- Documentos
- Anexos
- Firmas
- Interesados



# Tipos de datos complejos (y 2)

## BusObject

Datos comunes para control de la comunicación y auditoría

Campos	Descripción
<b>fechaActual</b>	Fecha y hora en curso
<b>idPrimitiva</b>	Identificador de la primitiva utilizada
<b>idSistema</b>	Identificador del sistema que realiza la llamada
<b>nombrePrimitiva</b>	Nombre de la primitiva
<b>rol</b>	Rol del usuario en el sistema destino
<b>timeOutRespuesta</b>	Tiempo de espera en milisegundos en caso de llamadas asíncronas
<b>usuario</b>	Usuario que realiza la llamada

# Tipos de datos complejos (y 3)

## Interesado

## Datos del interesado en un documento

Campos	Descripción
<b>ciwa</b>	Código de identificación w@nda del interesado
<b>nif</b>	Documentación del interesado
<b>nombre</b>	Nombre y apellidos o razón social del interesado
<b>razonInteres</b>	Código de la razón de interés

# Tipos de datos complejos (y 4)

## Firma

## Datos de firmas digitales de los anexos

Campos	Descripción
<b>cargo</b>	Cargo del firmante
<b>datosFirma</b>	Hash de la firma
<b>fecha</b>	Fecha y hora de firmado
<b>id</b>	Id. de la firma asignado por el sistema
<b>idAplicacion</b>	Id. de la aplicación de firmas
<b>idTransaccion</b>	Id. de la transacción de firma
<b>nif</b>	DNI/NIF del firmante
<b>nombre</b>	Nombre del firmante
<b>pkcs7</b>	Firma digital en formato de array de bytes
<b>servidor</b>	Servidor donde se firma el anexo

# Tipos de datos complejos (y 5)

## Anexo

## Metadatos de los anexos e información de las firmas asociadas

Campos	Descripción
<b>idAnexo</b>	Id. de anexo asignado por el sistema en el momento de su creación
<b>nombre</b>	Nombre completo del anexo (incluida su extensión)
<b>orden</b>	Orden de creación del anexo
<b>firmas</b>	Tipo Firma [ ]. Firmas asociadas al anexo.



# Tipos de datos complejos (y 7)

...continuación

Campos	Descripción
<b>tipo</b>	Tipo de documento
<b>estado</b>	Código de estado actual del documento
<b>reutilizable</b>	Indica si el documento puede ser usado por más de un expediente (S) o no (N)
<b>codigoProcedimiento</b>	Id. de tipo de procedimiento seguido en el expediente
<b>descripcionExpediente</b>	Descripción breve del objeto del expediente
<b>numeroExpediente</b>	Numero de expediente al que pertenece el documento, asignado por el autor
<b>version</b>	Versión del documento
<b>interesados</b>	Tipo Interesado[ ]. Interesados en el documento
<b>anexos</b>	Tipo Anexo [ ]. Anexos asociados al documento



# Tipos de datos complejos (y 8)

## Registro

## Datos para alta o modificación de registros de tablas codificadoras

Campos	Descripción
<b>codigo</b>	Código a añadir o modificar
<b>obsoleto</b>	Indicador de obsolescencia del código: N = en uso S = obsoleto
<b>valor</b>	Descripción del código



# Operaciones de los WS w@rdA para w@ndA

## Servicios web w@rdA

### Servicio de gestión documental:

- Procesos que afectan tanto a los datos estructurados del Documento (Metadatos o Cabecera) como a los datos no estructurados del mismo (Anexos).

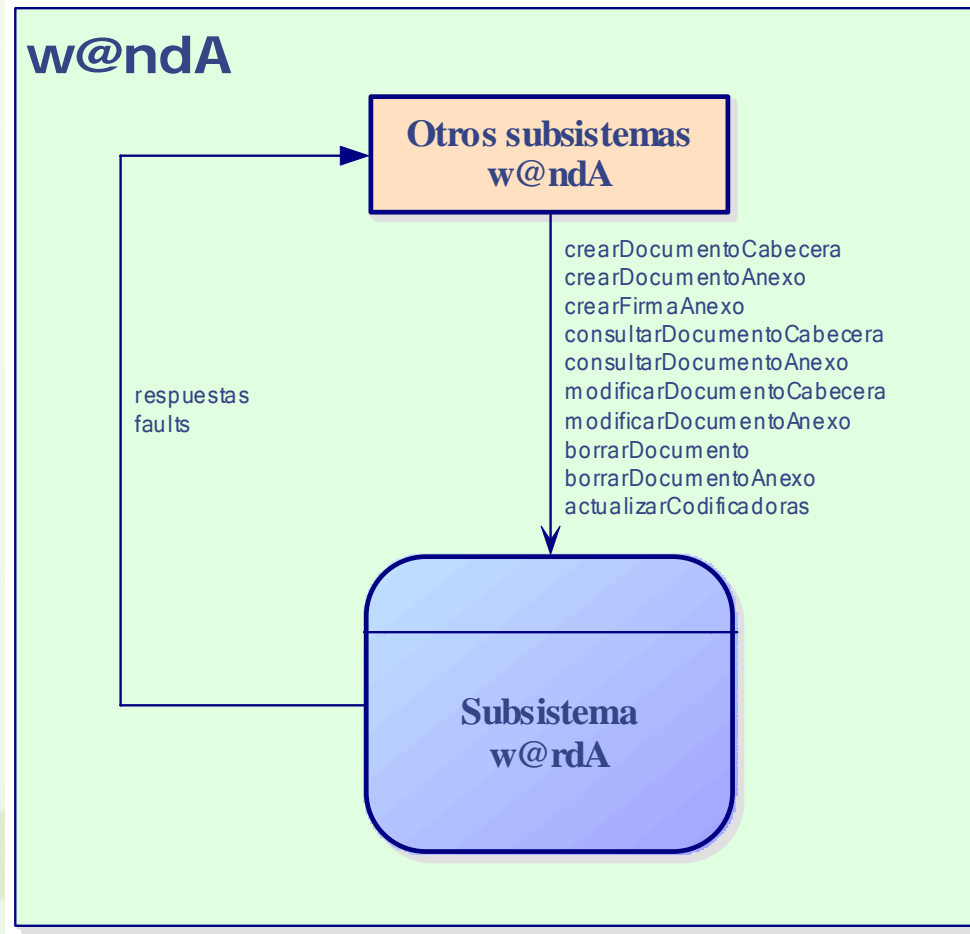
### Servicio de gestión de tablas:

- Procesos que afectan a las tablas auxiliares del sistema w@rdA (codificadoras y maestras en su caso).

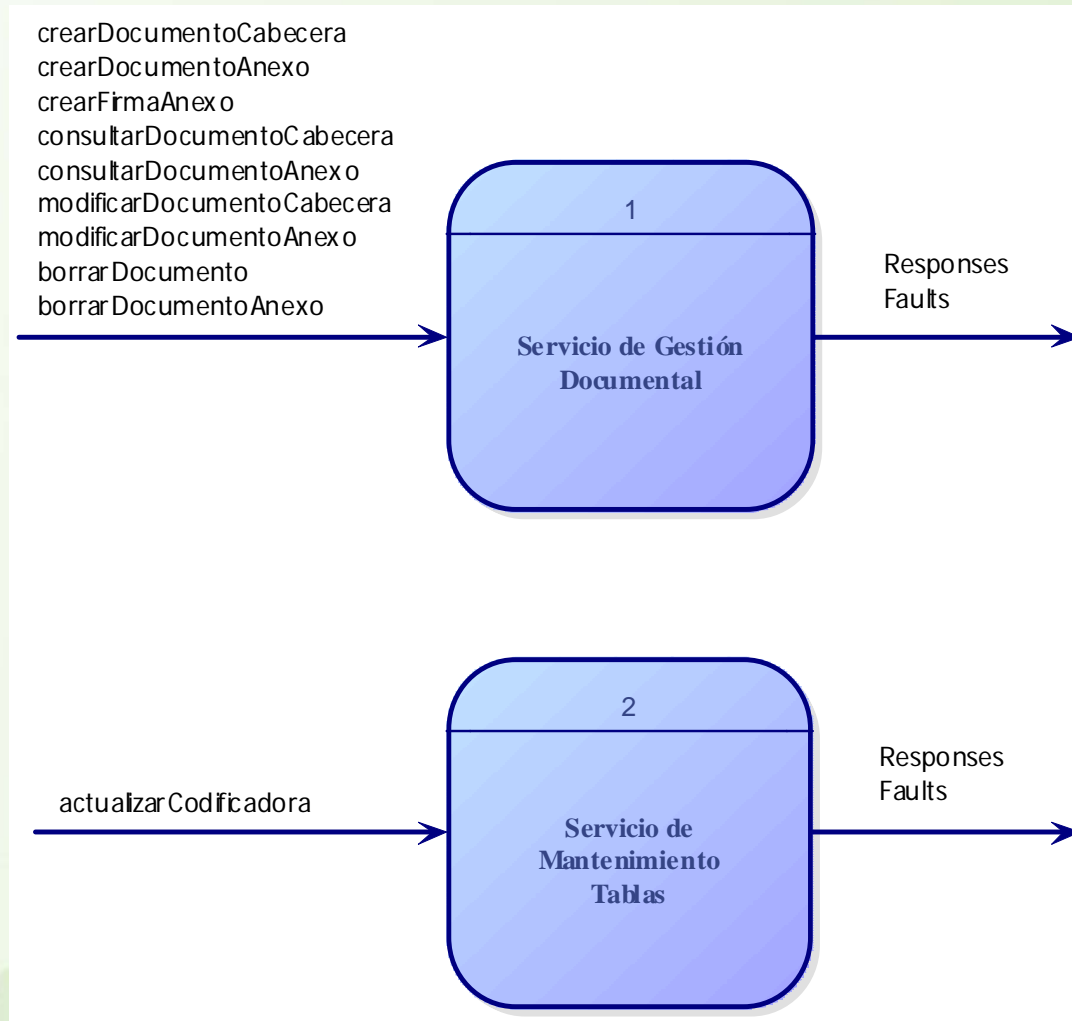


# Operaciones de los WS w@rdA (y 2)

## Mensajes a w@rdA desde w@ndA



# Operaciones de los WS w@rdA (y 3)

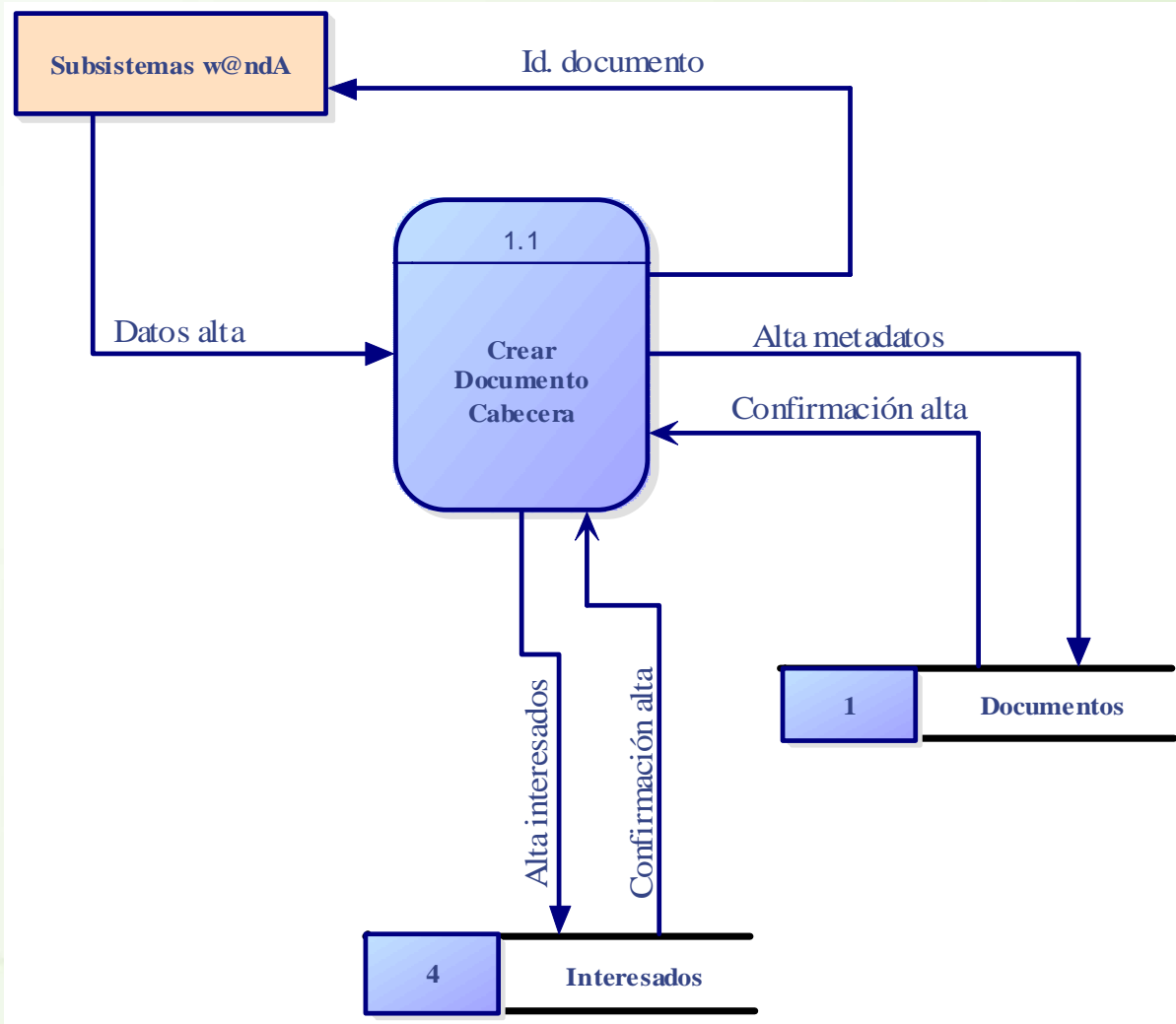


# Operaciones de los WS w@rdA (y 4)

## crearDocumentoCabecera

Crea un nuevo documento en el sistema w@rdA con sus metadatos asociados, e incorpora la información básica de los interesados en el documento.

# Operaciones de los WS w@rdA (y 5)



# Operaciones de los WS w@rdA (y 6)

Mensaje: **crearDocumentoCabeceraRequest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idOrganismo</b>	Código del organismo que crea el documento.
<b>unidadProductora</b>	Nombre de la unidad administrativa que genera el documento.
<b>fechaCreacion</b>	Fecha y hora de creación del documento.
<b>tipo</b>	Clave del tipo de documento.
<b>estado</b>	Clave del estado del documento.
<b>reutilizable</b>	Indicada si el documento puede ser utilizado por más de un expediente administrativo, en cuyo caso no se podrá eliminar.
<b>version</b>	Versión y revisión del documento.
<b>tipoProcedimiento</b>	Tipo de procedimiento que se sigue en el expediente que da lugar a la creación del documento.
<b>numeroExpediente</b>	Número de expediente asignado por el autor.
<b>descripcionExpediente</b>	Descripción breve del expediente.
<b>interesados</b>	Tipo Interesado[]. Relación de datos de interesados en el documento.



# Operaciones de los WS w@rdA (y 7)

Mensaje: **crearDocumentoCabeceraResponse** Sentido: **Salida**

**idDocumento**

ID del nuevo documento.

Formato idDocumento: [id sistema] # [id único] #

**8549#2398098#**

En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.



# Operaciones de los WS w@rdA (y 8)

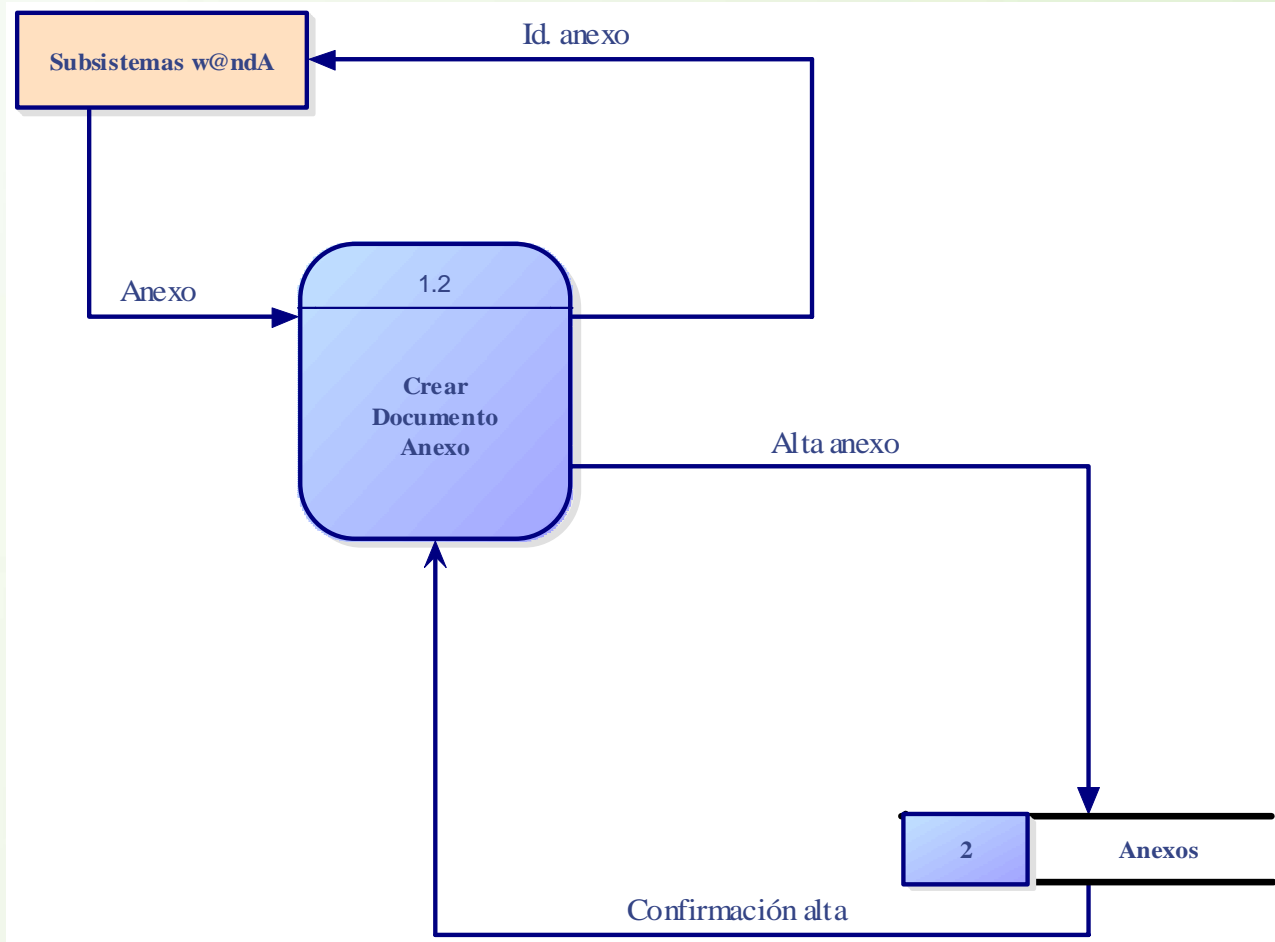
`crearDocumentoAnexo`  
`crearDocumentoAnexoBA`

Asocian y almacenan un anexo para un documento.

En el primer caso el fichero electrónico viaja como un *attachment* fuera del cuerpo principal del mensaje de entrada.

En el segundo el fichero electrónico viaja como un *bytearray* dentro del cuerpo principal del mensaje de entrada.

# Operaciones de los WS w@rdA (y 9)



# Operaciones de los WS w@rdA (y 10)

Mensaje: **crearDocumentoAnexoRequest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idDocumento</b>	ID del documento al que se va a asociar el anexo.
<b>nombre</b>	Nombre del anexo con extensión.
<b>anexo</b>	Datahandler (apuntador) al fichero electrónico que constituye el anexo propiamente dicho.

Mensaje: **crearDocumentoAnexoBARquest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idDocumento</b>	ID del documento al que se va a asociar el anexo.
<b>nombre</b>	Nombre del anexo con extensión.
<b>anexo</b>	Array de bytes con el fichero electrónico que constituye el anexo propiamente dicho.

# Operaciones de los WS w@rdA (y 11)

Mensaje: **crearDocumentoAnexoResponse** Sentido: **Salida**

<b>idAnexo</b>	ID del anexo creado.
----------------	----------------------

Mensaje: **crearDocumentoAnexoBAResponse** Sentido: **Salida**

<b>idAnexo</b>	ID del anexo creado.
----------------	----------------------

Formato idAnexo: [id sistema] # [id documento] # [id único] #

**8549#2398098#214236#**

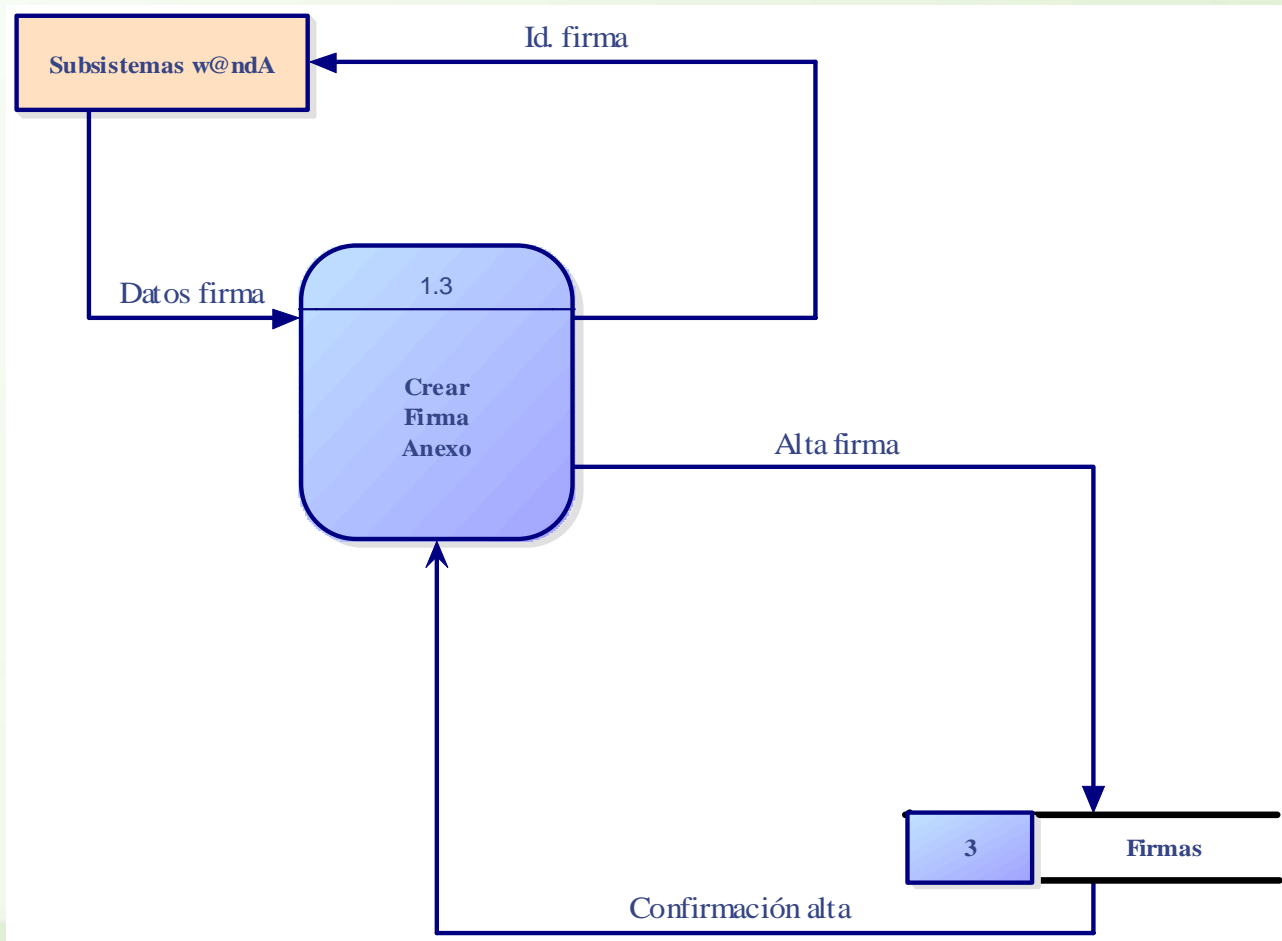
En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.

# Operaciones de los WS w@rdA (y 12)

## crearFirmaAnexo

Crea una firma de un anexo de un documento. La firma electrónica viaja dentro del cuerpo del mensaje como un *bytearray*.

# Operaciones de los WS w@rdA (y 13)



# Operaciones de los WS w@rdA (y 14)

Mensaje: <b>crearFirmaAnexoRequest</b> Sentido: <b>Entrada</b>	
<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idAnexo</b>	ID del anexo al que asociar la firma.
<b>nif</b>	NIF o DNI del firmante.
<b>nombre</b>	Nombre y apellidos del firmante.
<b>cargo</b>	Cargo que ocupa el firmante.
<b>fecha</b>	Fecha y hora de la firma.
<b>idAplicacion</b>	ID de la aplicación de firmas.
<b>servidor</b>	Servidor de firmas.
<b>idTransaccion</b>	ID de la transacción de firma.
<b>datosFirma</b>	Hash de la firma.
<b>pkcs7</b>	Array de bytes con el fichero de firma digital.



# Operaciones de los WS w@rdA (y 15)

Mensaje: `crearDocumentoFirmaResponse` Sentido: **Salida**

**idFirma**

ID de la firma creada.

**Formato idFirma: [id sistema] # [id documento] # [id anexo] # [id único] #**

**8549#2398098#214236#2133416#**

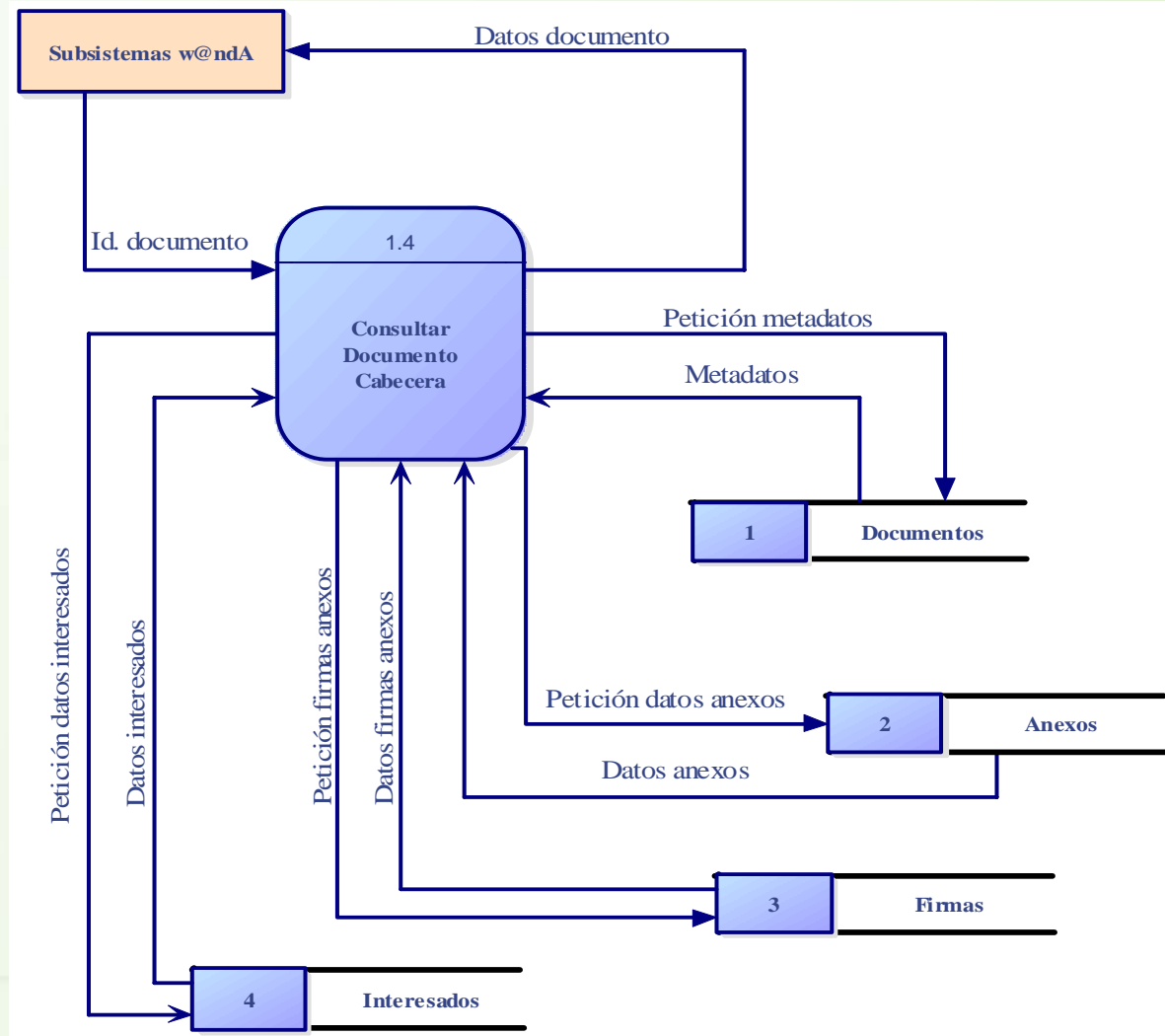
En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.

# Operaciones de los WS w@rdA (y 16)

**consultarDocumentoCabecera**

**Recupera los metadatos de un documento, junto con la información de interesados, anexos y firmas asociados al mismo (excluidos los ficheros electrónicos).**

# Operaciones de los WS w@rdA (y 17)



# Operaciones de los WS w@rdA (y 18)

Mensaje: **consultarDocumentoCabeceraRequest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idDocumento</b>	ID del documento cuyos metadatos y datos asociados se desean recuperar.

Formato idDocumento: [id sistema] # [id único] #

**8549#2398098#**

# Operaciones de los WS w@rdA (y 19)

Mensaje: **consultarDocumentoCabeceraResponse** Sentido: **Salida**

**documento**

Tipo Documento. Datos del documento solicitado.

En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.

# Operaciones de los WS w@rdA (y 20)

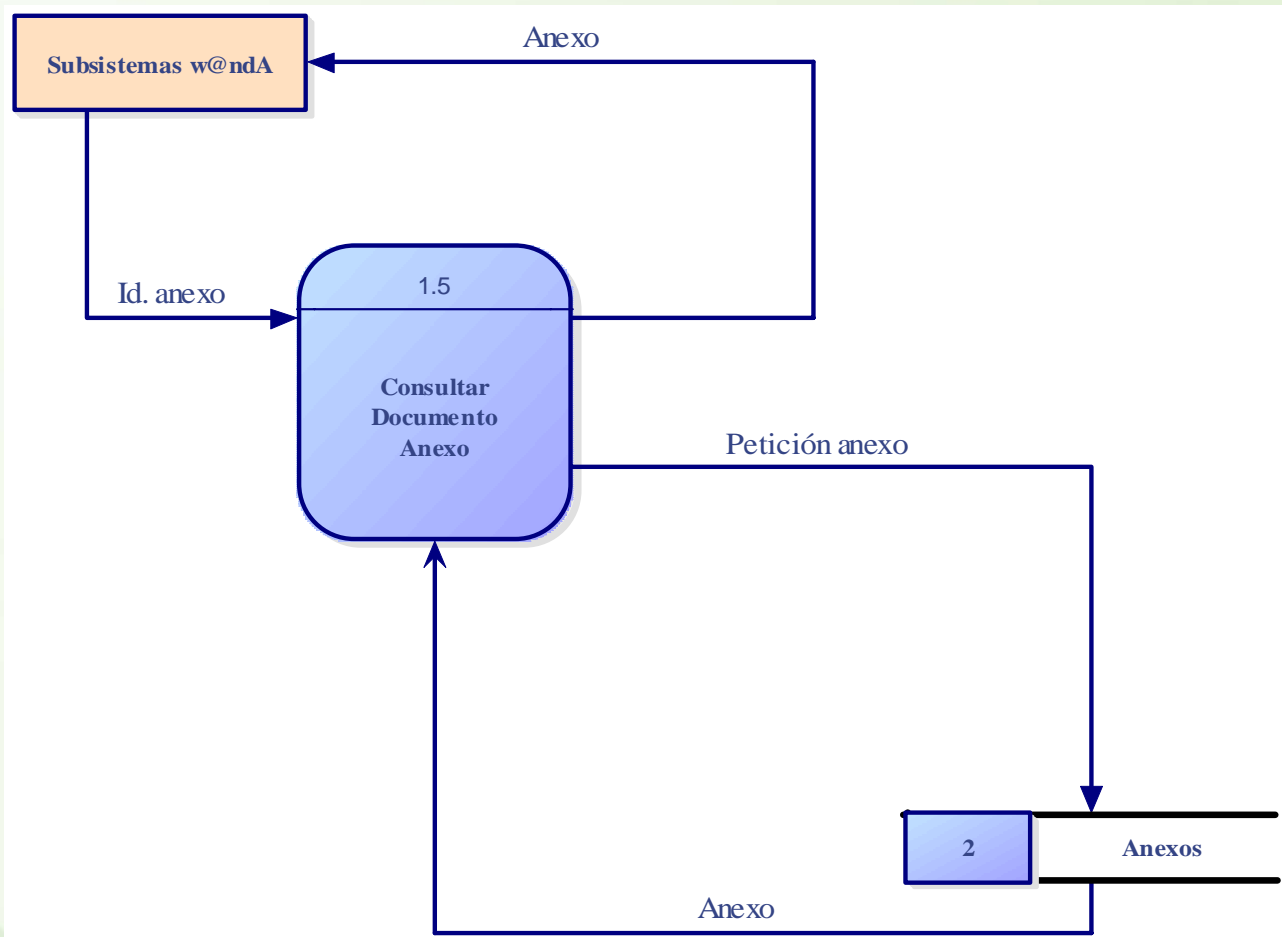
**consultarDocumentoAnexo**  
**consultarDocumentoAnexoBA**

**Recuperan el fichero electrónico correspondiente a un anexo de un documento.**

**En el primer caso el fichero electrónico retorna como un *attachment* fuera del cuerpo principal del mensaje de salida.**

**En el segundo el fichero electrónico retorna como un *bytearray* dentro del cuerpo principal del mensaje de salida.**

# Operaciones de los WS w@rdA (y 21)





# Operaciones de los WS w@rdA (y 22)

Mensaje: **consultarDocumentoAnexoRequest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idAnexo</b>	ID del anexo a recuperar.

Mensaje: **consultarDocumentoAnexoBAREquest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idAnexo</b>	ID del anexo a recuperar.

Formato idAnexo: [id sistema] # [id documento] # [id único] #

**8549#2398098#214236#**

# Operaciones de los WS w@rdA (y 23)

Mensaje: **consultarDocumentoAnexoResponse** Sentido: **Salida**

**anexo**

DataHandler al anexo solicitado.

Mensaje: **consultarDocumentoAnexoBAResponse** Sentido: **Salida**

**anexo**

Array de bytes con el anexo solicitado.

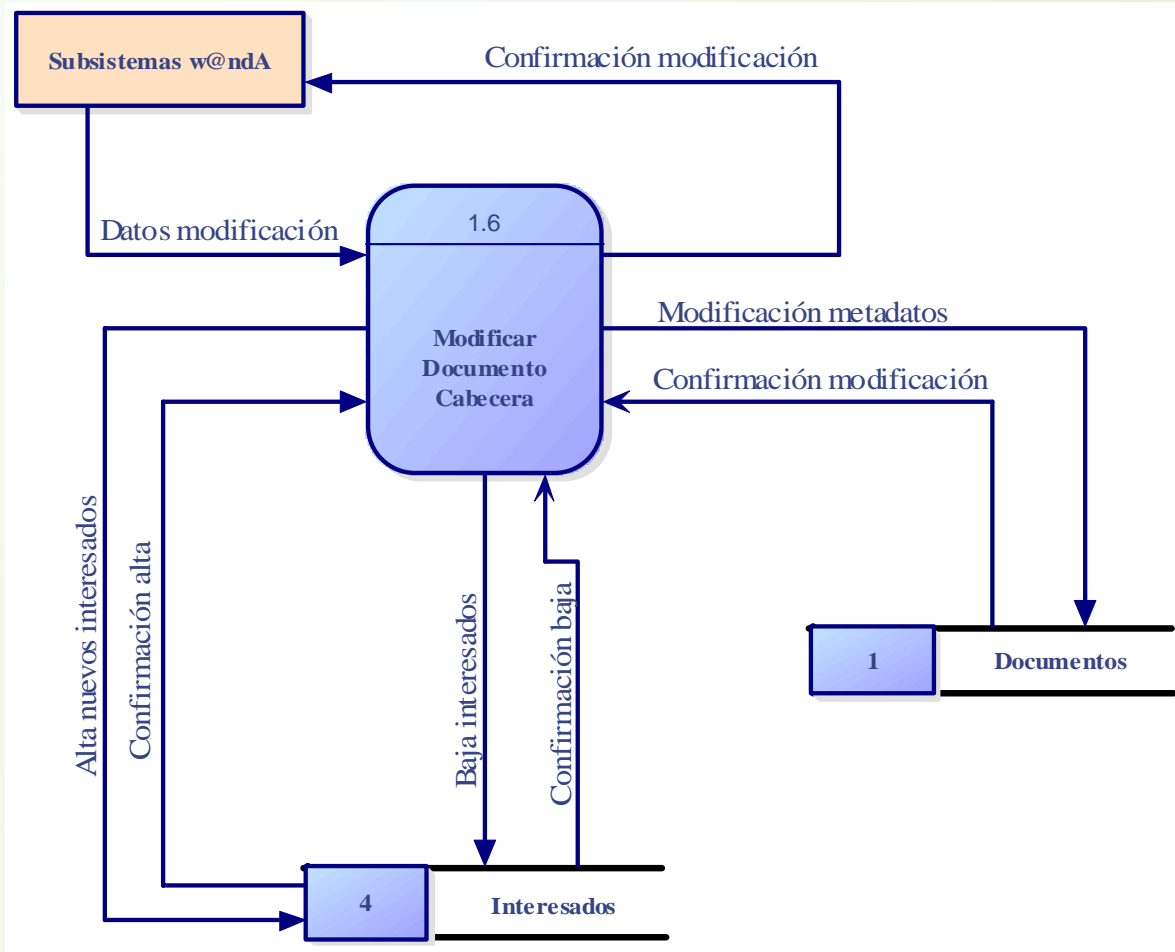
En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.

# Operaciones de los WS w@rdA (y 24)

**modificarDocumentoCabecera**

**Modifica metadatos de un documento. Si se envían interesados sustituye los interesados existentes por los nuevos.**

# Operaciones de los WS w@rdA (y 25)



# Operaciones de los WS w@rdA (y 26)

Mensaje: <b>modificarDocumentoCabeceraRequest</b> Sentido: <b>Entrada</b>	
<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idDocumento</b>	ID del documento que se quiere modificar.
<b>estado</b>	Clave del estado del documento.
<b>reutilizable</b>	Indicador de si el documento puede ser utilizado por más de un expediente administrativo, en cuyo caso no se podrá eliminar.
<b>version</b>	Versión y revisión del documento.
<b>tipoProcedimiento</b>	Código de tipo de procedimiento que se sigue en el expediente que da lugar a la creación del documento.
<b>numeroExpediente</b>	Número de expediente asignado por el autor.
<b>descripcionExpediente</b>	Descripción breve del expediente.
<b>Interesados</b>	Tipo Interesado [ ]. Datos de personas con interés en el documento. Si no se envía no se modifican los existentes.



# Operaciones de los WS w@rdA (y 27)

Mensaje: **modificarDocumentoCabeceraResponse** Sentido: **Salida**

**confirmacion**

Cadena OK.

En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.

# Operaciones de los WS w@rdA (y 28)

**modificarDocumentoAnexo  
modificarDocumentoAnexoBA**

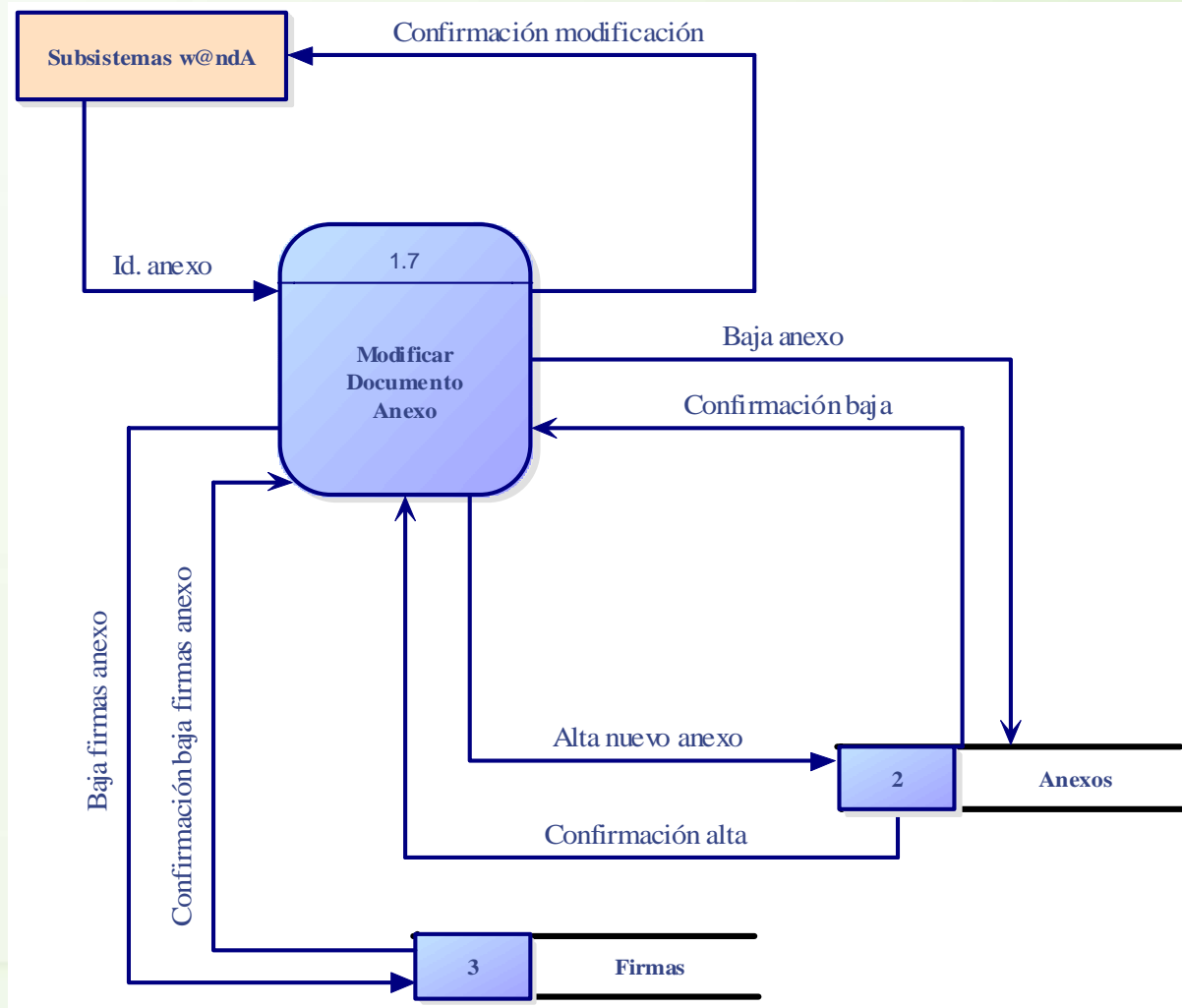
Reemplaza el anexo de un documento. Las firmas asociadas al anterior anexo dejan de tener validez y, por tanto, son eliminadas.

En el primer caso el fichero electrónico viaja como un *attachment* fuera del cuerpo principal del mensaje de entrada.

En el segundo el fichero electrónico viaja como un *bytearray* en el cuerpo principal del mensaje de entrada.



# Operaciones de los WS w@rdA (y 29)



# Operaciones de los WS w@rdA (y 30)

Mensaje: **modificarDocumentoAnexoRequest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idAnexo</b>	ID del anexo a sustituir.
<b>anexo</b>	Datahandler (apuntador) al fichero electrónico que sustituirá al anexo existente.

Mensaje: **modificarDocumentoAnexoBARquest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idAnexo</b>	ID del anexo a sustituir.
<b>anexo</b>	Array de bytes con el fichero electrónico que sustituirá al anexo existente.



# Operaciones de los WS w@rdA (y 31)

Mensaje: **modificarDocumentoAnexoResponse** Sentido: **Salida**

**confirmacion**

Cadena OK.

Mensaje: **modificarDocumentoAnexoBAResponse** Sentido: **Salida**

**confirmacion**

Cadena OK.

En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.

# Operaciones de los WS w@rdA (y 32)

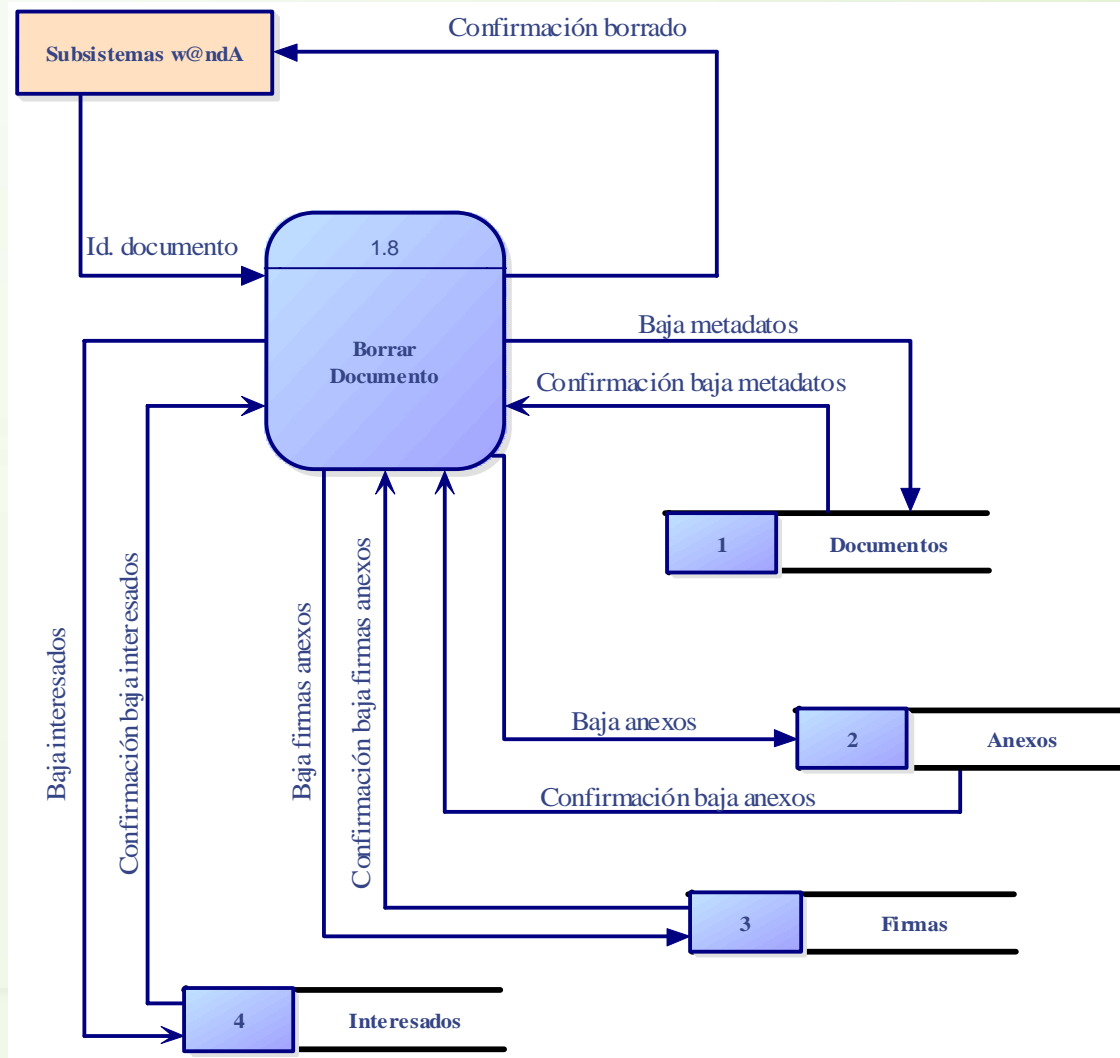
## borrarDocumento

Elimina los metadatos del documento y todos los demás datos asociados al mismo (interesados y anexos correspondientes con sus firmas).

Se aplica si el documento no es reutilizable.



# Operaciones de los WS w@rdA (y 33)



# Operaciones de los WS w@rdA (y 34)

Mensaje: **borrarDocumentoRequest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idDocumento</b>	ID del documento cuyos metadatos y datos asociados se desean eliminar.

Formato idDocumento: [id sistema] # [id único] #

**8549#2398098#**

# Operaciones de los WS w@rdA (y 35)

Mensaje: **borrarDocumentoResponse** Sentido: **Salida**

**confirmacion**

Cadena OK.

En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.

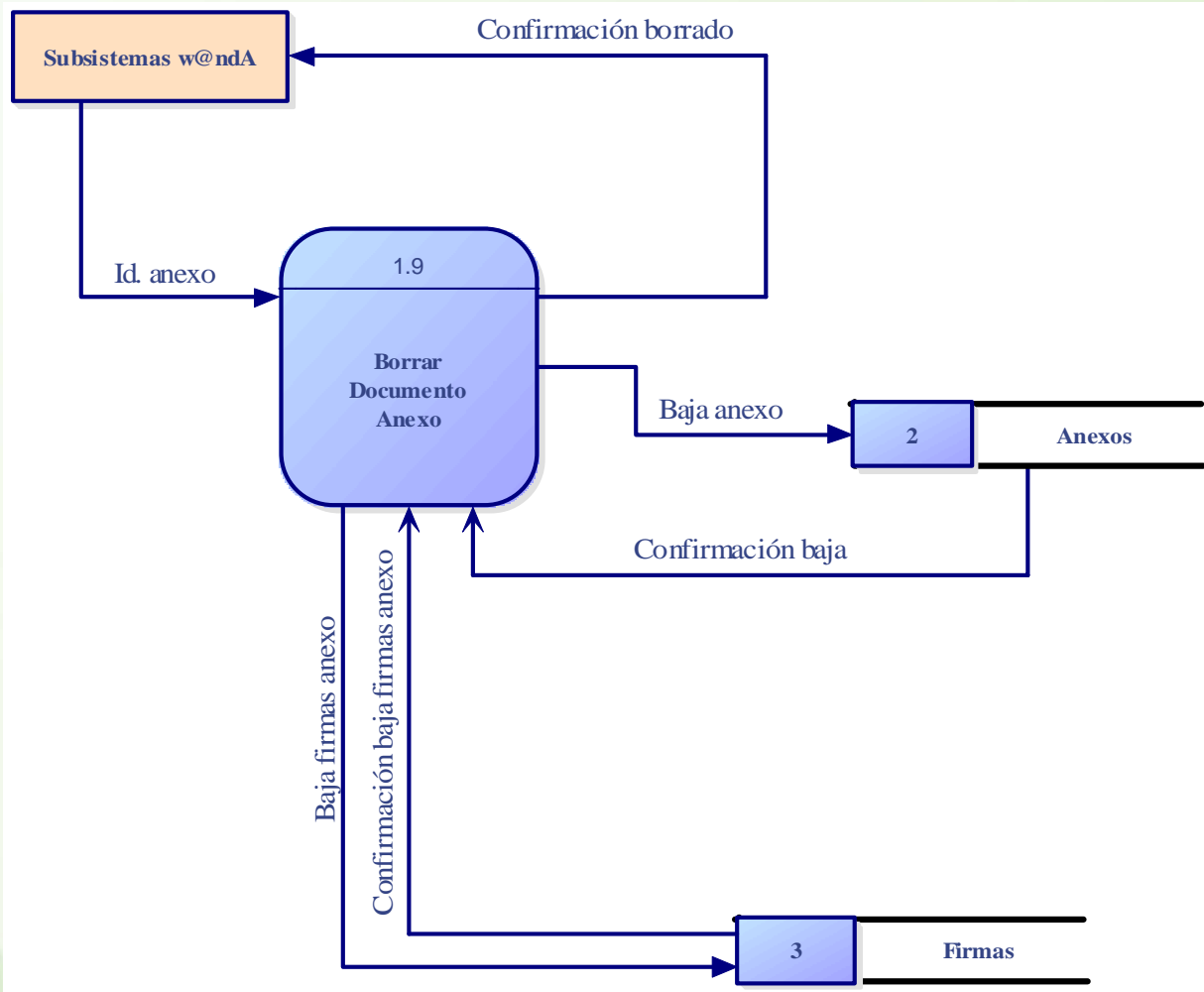


# Operaciones de los WS w@rdA (y 36)

## borrarDocumentoAnexo

Elimina un anexo correspondiente a un documento, junto con sus firmas asociadas.

# Operaciones de los WS w@rdA (y 37)



# Operaciones de los WS w@rdA (y 38)

Mensaje: **borrarDocumentoAnexoRequest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idAnexo</b>	ID del anexo que se desea eliminar.

Formato idAnexo: [id sistema] # [id documento] # [id único] #

**8549#2398098#214236#**

# Operaciones de los WS w@rdA (y 39)

Mensaje: **borrarDocumentoAnexoResponse** Sentido: **Salida**

**confirmacion**

Cadena OK.

En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.

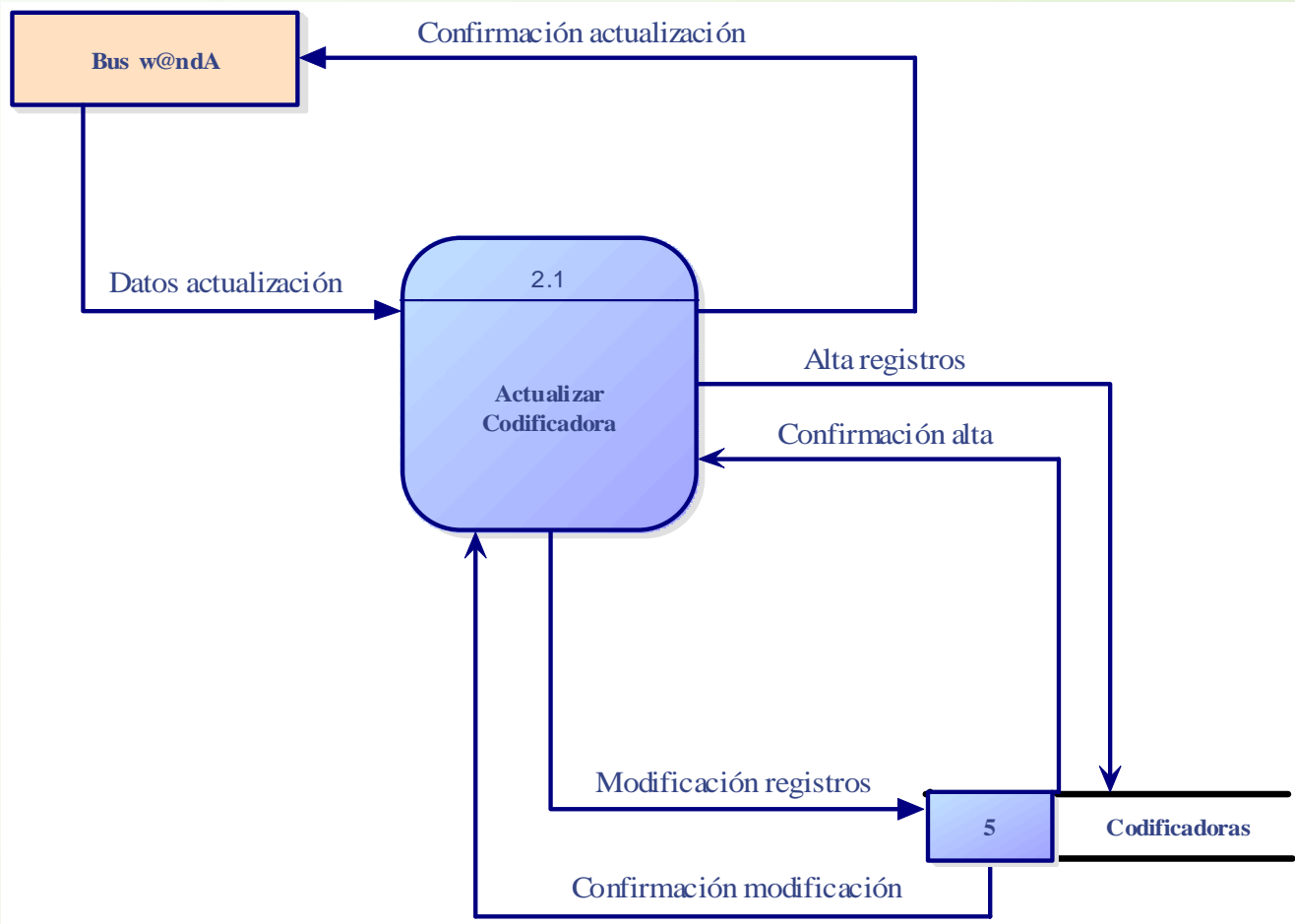
# Operaciones de los WS w@rdA (y 40)

## actualizarCodificadora

Mantenimiento de las tablas codificadoras mediante la modificación de los registros existentes y la inserción de los nuevos.

Si un registro no existe se añade a la tabla indicada, en caso contrario se actualizan la descripción y el indicador de obsolescencia.

# Operaciones de los WS w@rdA (y 41)



# Operaciones de los WS w@rdA (y 42)

Mensaje: **actualizarCodificadoraRequest** Sentido: **Entrada**

<b>busObject</b>	Tipo BusObject. Datos para control del mensaje y auditoría.
<b>idTabla</b>	Identificador de la tabla a actualizar
<b>registros</b>	Tipo Registro[ ]. Datos registros a dar de alta o modificar.



# Operaciones de los WS w@rdA (y 43)

Mensaje: **actualizarCodificadoraResponse** Sentido: **Salida**

**idTabla**

Identificador de la tabla actualizada

En caso de que se produzca algún fallo el sistema devolverá un mensaje de error de tipo *fault*.

# Mensajes de error de los WS w@rdA

Para todas las operaciones, en caso de error en cualquiera de los niveles de proceso del sistema, se genera un mensaje de error de estructura común, del tipo *fault* (fallo) definido por la especificación SOAP.

**SOAP: Simple Object Access Protocol.** Lenguaje en el que se apoya la transmisión de los mensajes en los servicios web. Proporciona un mecanismo simple y ligero para el intercambio de información estructurada y tipada entre equipos usando XML, en un entorno distribuido y descentralizado.

# Mensajes de error de WS w@rdA (y 2)

Se pueden generar tres clases de faults:

- Faults generados por Apache Axis
- Faults generados por los servicios
- Faults generados por el núcleo documental

# Mensajes de error de WS w@rdA (y 3)

## Faults generados por Apache Axis:

- Errores de conexión
- Errores por operaciones no cubiertas por los servicios
- Errores de tipo en los campos de los mensajes
- Errores por valores no validos (fechas incorrectas, campos numéricos con caracteres alfabéticos, etc.)

# Mensajes de error de WS w@rdA (y 4)

## Faults generados por los servicios:

- Errores de validación de datos
- Errores de proceso de los servicios web

# Mensajes de error de WS w@rdA (y 5)

## Faults generados por el núcleo documental:

- Errores de acceso a la base de datos
- Errores de proceso del núcleo
- Errores por parámetros válidos

# Mensajes de error de WS w@rdA (y 6)

Mensaje: <b>AxisFault</b> Sentido: <b>Salida</b>	
<b>faultcode</b>	Código que identifica el error.
<b>faultstring</b>	Error representado en forma de cadena de caracteres.
<b>faultactor</b>	Causante del error.
<b>detail</b>	De aparecer, contiene información específica de los errores generados por el servicio. El subcampo <i>errorcode</i> muestra el código de error de servicio y el subcampo <i>errordesc</i> muestra la descripción de dicho error.



# Config. de los servicios web

## Posibilidades de configuración:

- Configuración de base de datos w@rdA
- Registro de log
- Parámetros para el servicio web

# Config. de los servicios web (y 2)

## Fichero de configuración de base de datos ( /WEB-INF/classes/leciTd\_DbConn\_Cfg.xml )

- Con pool de conexiones (usando un DataSource para el contexto wardAService o Global)

```
<Config>  
  <Pooling>Y</Pooling>  
  <DataSource>jdbc/oracle</DataSource>  
  <User></User>  
  <Password></Password>  
</Config>
```

- Sin pool de conexiones (driver JDBC)

```
<Config>  
  <Pooling>N</Pooling>  
  <DataSource></DataSource>  
  <Driver>oracle.jdbc.driver.OracleDriver</Driver>  
  <Url>jdbc:oracle:thin:@host:port:sid</Url>  
  <User>usuario</User>  
  <Password>aWRvYzgz5MQ==</Password>  
</Config>
```

# Config. de los servicios web (y 3)

**Fichero de configuración del registro de log de los servicios web ( /WEB-INF/classes/log4j.xml )**

- Configurar los ‘appenders’ con la ruta apropiada
- Configurar el nivel de salida (level) de los logs
  - DEBUG
  - INFO
  - WARNING
  - ERROR
  - FATAL

# Config. de los servicios web (y 4)

## Fichero de parametrización del servicio web ( /WEB-INF/classes/warda.properties )

- Ids. de archivadores
- Ids. de campos de archivadores
- Datos tablas de códigos

En principio no es necesario modificarlo, pero si se hacen cambios en la estructura de los archivadores (nuevos campos, modificación de tipos, etc.) entonces habrá que tener en cuenta los posibles cambios.

# FIN



# API Java w@rdA

Curso para desarrolladores. Parte 4

# Índice

- **API w@rdA: introducción**
- **Clase de configuración**
- **Clases de login**
- **Clases de archivadores**
- **Clases de carpetas**
- **Otras clases (campos, documentos, clasificadores, enlaces y búsqueda)**
- **Caso práctico: invocación desde un cliente Java**



# API w@rdA: introducción

## Dos paquetes básicos:

- `ieci.tecdoc.core.db`

Conexión a la base de datos, usado tanto por el API de cliente como por el API de administración

- `ieci.tecdoc.sbo.idoc.api`

Clases que forman el núcleo del API cliente

# Clase de configuración

**Class DbConnectionConfig (paquete `ieci.tecdoc.core.db`)**

**Clase que contiene la configuración de conexión con la base de datos.**

**Se presentan dos opciones de conexión:**

- **Conexión mediante driver jdbc**
- **Conexión usando pool de conexiones**

# Clase de configuración (y 2)

**Constructor para conexión mediante driver jdbc:**

```
DbConnectionConfig(java.lang.String drvClsName,  
                   java.lang.String url,  
                   java.lang.String user,  
                   java.lang.String pwd)
```

Donde:

drvClsName - nombre de la clase del driver de base de datos

url - url de base de datos

user - usuario de base de datos

pwd - contraseña de base de datos

# Clase de configuración (y 3)

**Constructor para conexión mediante pool de conexiones:**

```
DbConnectionConfig(java.lang.String ctxName,  
                   java.lang.String user,  
                   java.lang.String pwd)
```

Donde:

ctxName - nombre del dataSource

user - usuario de base de datos

pwd - contraseña de base de datos

# Clase de configuración (y 4)

## Métodos de la clase DBConnectionConfig:

- `public java.lang.String getPwd()`  
Devuelve la contraseña de base de datos.
- `public java.lang.String getUrl()`  
Devuelve la contraseña de base de datos.
- `public java.lang.String getUser()`  
Devuelve el usuario de base de datos.
- `public boolean isCntByDriver()`  
true si la conexión es por driver.
- `public void setCtxName(java.lang.String ctxName)`  
Establece el nombre del datasource.

# Clase de configuración (y 5)

## Métodos de la clase DBConnectionConfig (continuación):

- `public void setDrvClsName(java.lang.String drvClsName)`  
Establece el nombre de la clase del driver de base de datos.
- `public void setPwd(java.lang.String pwd)`  
Establece el valor de la contraseña de base de datos.
- `public void setUrl(java.lang.String url)`  
Establece el valor de la url de conexión a base de datos.
- `public void setUser(java.lang.String user)`  
Establece el valor del usuario de base de datos.



# Clases de login

## Class Login (paquete `ieci.tecdoc.sbo.idoc.api`)

**Gestor de acceso. Conjunto de métodos para realizar login sobre la aplicación w@rdA.**

**Proporciona la funcionalidad básica para establecer una sesión.**

**La sesión puede establecerse contra diferentes sistemas:**

- **Estándar:** los usuarios se encuentran registrados en el sistema w@rdA
- **Ldap:** los usuarios se encuentran registrados en un directorio



# Clases de login (y 2)

## Constructores:

**Login()** throws `java.lang.Exception`

**Login**(`java.lang.String configDir`)  
throws `java.lang.Exception`

Donde:

`configDir` – directorio donde se encuentra la configuración de la base de datos ( `leciTd_DbConn_Cfg.xml` )

# Clases de login (y 3)

## Métodos de la clase Login:

- `public void setConnectionConfig(DbConnectionConfig dbConnConfig) throws java.lang.Exception`

Fija los parámetros de conexión a la base de datos que utilizan los métodos de esta clase. configuración corresponderá con la de la base de datos donde se encuentren los usuarios w@rdA.

- `public int getLoginMethod() throws java.lang.Exception`

Devuelve el método bajo el cual se establece la sesión. Los métodos vienen definidos en la clase LoginMethod:

- LoginMethod.STANDARD
- LoginMethod.LDAP

# Clases de login (y 4)

## Métodos de la clase Login (continuación):

- `public AcsAccessObject doLoginStd(  
    java.lang.String name,  
    java.lang.String pwd,  
    int cntsTriesNum)  
    throws java.lang.Exception`

Lleva a cabo el establecimiento de sesión estándar, devolviendo la referencia a un objeto de tipo `AcsAccessObject` que contiene información básica del usuario.

Donde:

name - nombre del usuario

pwd - contraseña del usuario

cntsTriesNum - número de intentos en el establecimiento de sesión

# Clases de login (y 5)

## Métodos de la clase Login (continuación):

- `public AcsAccessObject doLoginLdap(  
    java.lang.String name,  
    java.lang.String pwd,  
    int cntsTriesNum)  
    throws java.lang.Exception`

Lleva a cabo el establecimiento de sesión contra directorio LDAP, devolviendo la referencia a un objeto de tipo `AcsAccessObject` que contiene información básica del usuario.

Donde:

name - nombre del usuario

pwd - contraseña del usuario

cntsTriesNum - número de intentos en el establecimiento de sesión

# Clases de login (y 6)

## Class AcsAccessObject (paquete ieci.tecdoc.sbo.idoc.api)

Esta clase contiene la información sobre el usuario que ha realizado un login en w@rdA.

Se utiliza para chequear los permisos que el usuario posee sobre archivadores, carpetas, etc.

# Clases de login (y 7)

## Métodos de la clase `AcsAccessObject` :

- `public ieci.tecdoc.sbo.acs.base.AcsAccessToken  
    getAccessToken()`

Devuelve el `AcsAccessToken` con la información de acceso del usuario.
- `public int getUserId()`

Devuelve el identificador de usuario.
- `public  
    iecei.tecdoc.core.collections.IeciTdLongIntegerArrayList  
    getGroupIds()`

Devuelve una lista con los identificadores de los grupos a los que pertenece el usuario.



# Clases de login (y 8)

## Métodos de la clase `AcsAccessObject` (continuación) :

- `public int getDeptId()`

Devuelve el identificador de departamento al que pertenece el usuario.

- `public java.lang.String getProfile()`

Devuelve el perfil del usuario. Los valores posibles están recogidos en la clase `AcsProfile`:

- `AcsProfile.SYS_XSUPERUSER`
- `AcsProfile.SYS_SUPERUSER`
- `AcsProfile.IDOC_SUPERUSER`
- `AcsProfile.IDOC_MANAGER`
- `AcsProfile.IDOC_STANDARD`
- `AcsProfile.IDOC_NONE`



# Clases de archivadores

## Class Archive (paquete `ieci.tecdoc.sbo.idoc.api`)

Esta clase actúa como una fachada para los archivadores.

Proporciona un conjunto de métodos sencillos para realizar operaciones sobre archivadores.

# Clases de archivadores (y 2)

## Constructores:

**Archive()** throws `java.lang.Exception`

**Archive**(`java.lang.String configDir`)

throws `java.lang.Exception`

## Donde:

`configDir` – ruta donde se encuentra la configuración de la base de datos  
(`leciTd_DbConn_Cfg.xml`)

# Clases de archivadores (y 3)

## Métodos de la clase Archive:

- `public void setConnectionConfig(DbConnectionConfig dbConnConfig) throws java.lang.Exception`

Fija los parámetros de conexión a la base de datos que utilizan los métodos de esta clase. La configuración corresponderá con la de la base de datos donde se encuentren los usuarios w@rdA.

Donde:

dbConnConfig - Configuración de la conexión de base de datos

# Clases de archivadores (y 4)

## Métodos de la clase Archive (continuación):

- `public boolean canLoadArchive(AcsAccessObject acs, int archId)`  
`throws java.lang.Exception`

Devuelve true si el usuario tiene permisos de acceso sobre el archivador.

Donde:

acs - Objeto AcsAccessObject con los permisos del usuario. Si se recibe null no se chequean permisos  
archId - Identificador del archivador

# Clases de archivadores (y 5)

## Métodos de la clase Archive (continuación):

- `public ArchiveObject loadArchive(AcsAccessObject acs, int archId)`  
`throws java.lang.Exception`

Crea un objeto `ArchiveObject` con la información de un archivador concreto. Antes de crearlo verifica si el usuario que trata de obtener dicho `ArchiveObject` tiene permisos sobre el archivador.

Donde:

`acs` - Información sobre el usuario que ha realizado un login y que se utiliza para chequear permisos de consulta sobre el archivador. Si este valor es null, no se realiza ningún chequeo de permisos.

`archId` - Identificador del archivador



# Clases de archivadores (y 6)

**Class ArchiveObject (paquete `ieci.tecdoc.sbo.idoc.api`)**

**Contiene toda la información relevante de un archivador.**

**Se utiliza para chequear los permisos que el usuario posee sobre archivadores, carpetas, etc.**



# Clases de archivadores (y 7)

## Métodos de la clase `AcsAccessObject` :

- `public ieci.tecdoc.sbo.idoc.archive.base.ArchiveToken  
getArchiveToken()`

Devuelve el `ArchiveToken` con la información del archivador.

- `public int getId()`

Devuelve el identificador del archivador.

- `public java.lang.String getName()`

Devuelve el nombre del archivador.



# Clases de carpetas

## Class Folder (paquete `ieci.tecdoc.sbo.idoc.api`)

Esta clase actúa como una fachada para las carpetas.

Proporciona un conjunto de métodos sencillos para realizar operaciones sobre carpetas.

# Clases de carpetas (y 2)

## Constructores:

**Folder()** throws `java.lang.Exception`

**Folder**(`java.lang.String configDir`)  
throws `java.lang.Exception`

Donde:

`configDir` – ruta donde se encuentra la configuración de la base de datos  
(`leciTd_DbConn_Cfg.xml`)

# Clases de carpetas (y 3)

## Métodos de la clase Folder:

- `public void setConnectionConfig(DbConnectionConfig dbConnConfig) throws java.lang.Exception`

Fija los parámetros de conexión a la base de datos que utilizan los métodos de esta clase. La configuración corresponderá con la de la base de datos donde se encuentren los usuarios w@rdA.

Donde:

dbConnConfig - Configuración de la conexión de base de datos

# Clases de carpetas (y 4)

## Métodos de la clase Folder (continuación):

- ```
public boolean canLoadFolder(AcsAccessObject acs,  
                             ArchiveObject arch,  
                             int fdrId)  
  
    throws java.lang.Exception
```

Devuelve true si el usuario tiene permisos de acceso sobre la carpeta.

Donde:

- acs - Objeto AcsAccessObject con los permisos del usuario Si se recibe null no se chequean permisos
- arch - archivador al que pertenece la carpeta
- fdrId - Identificador de la carpeta



# Clases de carpetas (y 5)

## Métodos de la clase Folder (continuación):

- `public FolderFieldObjects fetchFolderValues(  
AcsAccessObject acs,  
ArchiveObject arch,  
int fdrId)  
throws java.lang.Exception`

Obtiene la información sobre los campos que forman la carpeta.

Donde:

- acs - Objeto AcsAccessObject con los permisos del usuario. Si se recibe null no se chequean permisos
- arch - archivador al que pertenece la carpeta
- fdrId - identificador de la carpeta

# Clases de carpetas (y 6)

## Métodos de la clase Folder (continuación):

- ```
public FolderObject loadFolder( AcsAccessObject acs,  
                                int userId,  
                                ArchiveObject arch,  
                                int fdrId)  
  
    throws java.lang.Exception
```

Creará un objeto FolderObject con la información de los campos de una carpeta concreta. Antes de crearlo verifica si el usuario que trata de obtener dicho FolderObject tiene permisos sobre el archivador, lanzando una IeciTdException en caso de que el usuario no tenga permisos de acceso sobre la carpeta

Donde:

- acs - Objeto AcsAccessObject con los permisos del usuario. Si se recibe null no se chequean permisos
- userId - identificador de usuario
- arch - archivador al que pertenece la carpeta
- fdrId - identificador de la carpeta



# Clases de carpetas (y 7)

## Métodos de la clase Folder (continuación):

- ```
public FolderObject loadFolder( AcsAccessObject acs,  
                                int userId,  
                                ArchiveObject arch,  
                                int fdrId,  
                                FolderFieldObjects fldsValues)  
    throws java.lang.Exception
```

Crea un objeto FolderObject de forma similar al anterior, pero valores de los campos de una carpeta recuperados con el método fetchFolderValues.

Donde:

acs - Objeto AcsAccessObject con los permisos del usuario. Si se recibe null no se chequean permisos

userId - identificador de usuario

arch - archivador al que pertenece la carpeta

fdrId - identificador de la carpeta

fldsValues - lista de campos que forman la carpeta





# Clases de carpetas (y 8)

## Métodos de la clase Folder (continuación):

- `public boolean canRemoveFolder(AcsAccessObject acs,  
ArchiveObject arch,  
int fdrId)  
  
throws java.lang.Exception`

Devuelve true si el usuario tiene permisos para eliminar la carpeta especificada.

Donde:

- acs - Objeto AcsAccessObject con los permisos del usuario. Si se recibe null no se chequean permisos
- arch - archivador al que pertenece la carpeta
- fdrId - identificador de la carpeta

# Clases de carpetas (y 9)

## Métodos de la clase Folder (continuación):

- ```
public boolean removeFolder(AcsAccessObject acs,  
                                int userId,  
                                ArchiveObject arch,  
                                int fdrId)  
  
                                throws java.lang.Exception
```

Elimina la carpeta indicada, si el usuario tiene permiso para ello, lanzando una `IeciTdException` en caso contrario.

Donde:

- acs - Objeto `AcsAccessObject` con los permisos del usuario. Si se recibe null no se chequean permisos
- userId - identificador de usuario
- arch - archivador al que pertenece la carpeta
- fdrId - identificador de la carpeta

# Clases de carpetas (y 8)

## Métodos de la clase Folder (continuación):

- `public boolean canCreateFolder(AcsAccessObject acs, ArchiveObject arch)`  
`throws java.lang.Exception`

Devuelve true si el usuario tiene permisos para crear una carpeta en un determinado archivador.

Donde:

- acs - Objeto AcsAccessObject con los permisos del usuario. Si se recibe null no se chequean permisos
- arch - archivador al que pertenece la carpeta

# Clases de carpetas (y 9)

## Métodos de la clase Folder (continuación):

- `public FolderObject newFolder(AcsAccessObject acs, ArchiveObject arch) throws java.lang.Exception`

Verifica si el usuario tiene permisos para crear una carpeta en un determinado archivador, y si es así devuelve un FolderObject con el esqueleto de dicha carpeta.

Donde:

- acs - Objeto AcsAccessObject con los permisos del usuario. Si se recibe null no se chequean permisos
- arch - archivador al que pertenece la carpeta

# Clases de carpetas (y 10)

## Métodos de la clase Folder (continuación):

- ```
public void createFolder(int userId,  
                          ArchiveObject arch,  
                          FolderObject fdr)  
  
                          throws java.lang.Exception
```

Creará una carpeta en el archivador especificado si el usuario tiene permisos para ello. Lanza una `IOException` en caso de que el usuario no tenga permisos para crear la carpeta. Los datos de los campos de la carpeta deben ser los que van a insertarse en la base de datos. No realiza ningún tipo de validación sobre los datos de la carpeta.

Donde:

`userId` - identificador de usuario  
`arch` - archivador al que pertenece la carpeta  
`fdr` - carpeta que se va a crear

# Clases de carpetas (y 11)

## Métodos de la clase Folder (continuación):

- `public boolean canEditFolder(AcsAccessObject acs, ArchiveObject arch, int fdrId)`  
`throws java.lang.Exception`

Devuelve true si el usuario tiene permisos para modificar una carpeta en un determinado archivador.

Donde:

acs - Objeto AcsAccessObject con los permisos del usuario. Si se recibe null no se chequean permisos

arch - archivador al que pertenece la carpeta

fdrId - identificador de la carpeta



# Clases de carpetas (y 12)

## Métodos de la clase Folder (continuación):

- ```
public void editFolder(AcsAccessObject acs,  
                      int userId,  
                      ArchiveObject arch,  
                      int fdrId)  
    throws java.lang.Exception
```

Prepara una carpeta para su modificación en el archivador especificado si el usuario tiene permisos para ello, bloqueándola por el usuario para que esta pueda ser modificada. Lanza una `IeciTdException` si el usuario no tiene permisos para modificar o la carpeta se encuentra bloqueada por otro usuario.

Donde:

`acs` - objeto `AcsAccessObject` con los permisos del usuario. Si se recibe `null` no se chequean permisos  
`userId` - identificador de usuario  
`arch` - archivador al que pertenece la carpeta  
`fdrId` - identificador de la carpeta



# Clases de carpetas (y 13)

## Métodos de la clase Folder (continuación):

- ```
public void terminateEditFolder(int userId,  
                                ArchiveObject arch,  
                                int fdrId)  
    throws java.lang.Exception
```

Termina el modo edición de la carpeta para el usuario. Es decir, desbloquea la carpeta (si es que realmente estaba bloqueada por dicho usuario). Lanza una `lecITdException` en caso de que el usuario no tenga permisos para modificar

Donde:

`userId` - identificador de usuario

`arch` - archivador al que pertenece la carpeta

`fdrId` - identificador de la carpeta que se quiere desbloquear

# Clases de carpetas (y 14)

## Métodos de la clase Folder (continuación):

- ```
public void storeFolder(int userId,
                        ArchiveObject arch,
                        FolderObject fdr)
                        throws java.lang.Exception
```

Almacena una carpeta que estaba siendo editada en base de datos, lanzando una `IeciTdException` si el usuario no tiene previamente bloqueada la carpeta mediante el método `editFolder`.

Donde:

`userId` - identificador de usuario

`arch` - archivador al que pertenece la carpeta

`fdr` – información de las modificaciones que se quieren realizar

# Clases de carpetas (y 15)

## Métodos de la clase Folder (continuación):

- `public byte[] retrieveFolderDocumentFile(  
AcsAccessObject acs,  
ArchiveObject arch,  
FolderObject fdr,  
int docId)  
throws java.lang.Exception`

Obtiene el contenido en bytes de un documento de la carpeta especificada, lanzando una excepción si se produce un error al leer el fichero

Donde:

acs - Objeto AcsAccessObject con los permisos del usuario. Si se recibe null no se chequean permisos

arch - archivador al que pertenece la carpeta

fdr - carpeta de donde se va a recuperar el documento

docId - identificador del documento

# Clases de carpetas (y 16)

## Class FolderObject (paquete `ieci.tecdoc.sbo.idoc.api`)

Esta clase modeliza una carpeta de un archivador.

Proporciona funcionalidad para trabajar con:

- Campos (Fields)
- Documentos (Documents)
- Clasificadores (Dividers)
- Enlaces (Links)

# Clases de carpetas (y 17)

## Métodos de la clase FolderObject:

- `public int getId()`

Obtiene el identificador de la carpeta.

- `public boolean isNew()`

Devuelve true si una carpeta es nueva o false si no lo es.

# Clases de carpetas (y 18)

## Métodos de la clase FolderObject (continuación) :

- `public java.lang.Object getFieldValue(int fldId)  
throws java.lang.Exception`

Retorna la referencia de un objeto que contiene el valor del campo. Lanza una excepción si el campo es multivalor.

Donde:

fldId - identificador del campo



# Clases de carpetas (y 19)

## Métodos de la clase FolderObject (continuación):

- ```
public java.util.ArrayList getFieldValues(int fldId)
    throws java.lang.Exception
```

Retorna la referencia a una lista de objetos que contienen los valores asociados a un campo **multivalor**. Lanza una excepción si el campo no es multivalor.

Donde:

fldId - identificador del campo

Los 9 tipos de campos w@rdA son encapsulados en los siguientes objetos:

| Tipo campo w@rdA | Objeto Java | Tipo campo w@rdA | Objeto Java |
|------------------|-------------|------------------|-------------|
| Fecha            | Date        | Short int        | Short       |
| Fecha/Hora       |             | Long int         | Integer     |
| Hora             |             | Short dec        | Float       |
| Texto corto      | String      | Long dec         | Double      |
| Texto largo      |             |                  |             |



# Clases de carpetas (y 20)

## Métodos de la clase FolderObject (continuación):

- `public void setFieldValue(int fldId, Object val)  
throws java.lang.Exception`

Asigna un nuevo valor a un campo de la carpeta. Lanza una excepción si el campo es multivalor.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

fldId - identificador del campo

val – Objeto con valor asociado al campo

# Clases de carpetas (y 21)

## Métodos de la clase FolderObject (continuación):

- `public void addFieldValue(int fldId, Object val)  
throws java.lang.Exception`

Añade un nuevo valor a la colección de valores de un campo **multivalor** de la carpeta. Lanza una excepción si el campo no es multivalor.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

fldId - identificador del campo

val – Objeto con valor asociado al campo

# Clases de carpetas (y 22)

## Métodos de la clase FolderObject (continuación):

- `public void removeFieldValues(int fldId)  
throws java.lang.Exception`

Elimina todos los valores de un campo **multivalor** de la carpeta. Lanza una excepción si el campo no es multivalor.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

fldId - identificador del campo

# Clases de carpetas (y 23)

## Métodos de la clase FolderObject (continuación):

- `public String getDocumentTreeRootName()`  
`throws java.lang.Exception`

Obtiene el nombre de la etiqueta asignada al elemento raíz del árbol de clasificadores y documentos de la carpeta.

# Clases de carpetas (y 24)

## Métodos de la clase FolderObject (continuación):

- `public FolderDividerObject getDivider(int divId)`  
`throws java.lang.Exception`

Recupera la información sobre un clasificador de la carpeta.

Donde:

`divId` - identificador del clasificador

- `public int getDividerId(String name)`  
`throws java.lang.Exception`

Obtiene el identificador de un clasificador a partir de su nombre. Se devuelve una excepción si no existe dicho clasificador.

Donde:

`name` - nombre del clasificador

# Clases de carpetas (y 25)

## Métodos de la clase FolderObject (continuación):

- `public FolderDividerObjects getAllDividers()`  
`throws java.lang.Exception`  
Recupera la colección con todos los clasificadores de la carpeta.
- `public FolderDividerObjects getRootDividerChildren()`  
`throws java.lang.Exception`  
Recupera la colección con los clasificadores hijos del nodo raíz de clasificadores-documentos de la carpeta.



# Clases de carpetas (y 26)

## Métodos de la clase FolderObject (continuación):

- `public FolderDividerObjects getDividerChildren(  
int parentDivId)  
throws java.lang.Exception`

Recupera la colección con los clasificadores hijos de un clasificador dado.

Donde:

parentDivId - identificador del clasificador



# Clases de carpetas (y 27)

## Métodos de la clase FolderObject (continuación):

- `public int addRootDivider(String name)`

`throws java.lang.Exception`

Añade un nuevo clasificador cuyo padre es el elemento raíz del árbol de clasificadores-documentos de la carpeta. Se produce una excepción si ya existe un clasificador hermano con el mismo nombre.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

name – nombre de nuevo del clasificador

# Clases de carpetas (y 28)

## Métodos de la clase FolderObject (continuación):

- `public int addDivider(String name, int parentDivId)`  
`throws java.lang.Exception`

Añade un nuevo clasificador al clasificador padre especificado. Se produce una excepción si ya existe un clasificador hermano con el mismo nombre o si no se encuentra el clasificador padre.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

name – nombre de nuevo del clasificador

parentDivId - identificador del clasificador padre

# Clases de carpetas (y 29)

## Métodos de la clase FolderObject (continuación):

- `public void renameDivider(int divId, String name)`  
`throws java.lang.Exception`

Renombra un clasificador de la carpeta. Se produce una excepción si ya existe un clasificador hermano con el mismo nombre o si el nuevo nombre del clasificador excede 32 caracteres.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

divId - identificador del clasificador

name – nuevo nombre del clasificador

# Clases de carpetas (y 30)

## Métodos de la clase FolderObject (continuación):

- `public void removeDivider(int divId)`  
`throws java.lang.Exception`

Elimina un clasificador de la carpeta. Se produce una excepción si el clasificador tiene hijos.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

divId - identificador del clasificador

# Clases de carpetas (y 31)

## Métodos de la clase FolderObject (continuación):

- `public FolderDocumentObject getDocument(int docId)`  
`throws java.lang.Exception`

Recupera la información sobre un documento de la carpeta. Lanza una excepción si no se encuentra el documento.

Donde:

docId - identificador del documento

- `public int getDocumentId(String name)`  
`throws java.lang.Exception`

Obtiene el identificador de un documento a partir de su nombre o devuelve -1 si no se encuentra.

Donde:

name - nombre del documento

# Clases de carpetas (y 32)

## Métodos de la clase FolderObject (continuación):

- `public FolderDocumentObjects getAllDocuments()`

`throws java.lang.Exception`

Recupera la colección con todos los datos de los documentos de la carpeta.

- `public FolderDocumentObjects getRootDocumentChildren()`

`throws java.lang.Exception`

Recupera la colección con los documentos hijos del nodo raíz de clasificadores-documentos de la carpeta.



# Clases de carpetas (y 33)

## Métodos de la clase FolderObject (continuación):

- `public FolderDocumentObjects getDocumentChildren(  
int parentDivId)  
throws java.lang.Exception`

Recupera la colección con los datos de los documentos hijos de un clasificador dado.

Donde:

parentDivId - identificador del clasificador padre



# Clases de carpetas (y 34)

## Métodos de la clase FolderObject (continuación):

- ```
public int addRootDocument(String name,  
                            String fileExt,  
                            String pathDocumentFile)  
    throws java.lang.Exception
```

Añade un nuevo documento cuyo padre es el elemento raíz del árbol de clasificadores-documentos de la carpeta. Se produce una excepción si nombre del documento ya existe o tiene más de 32 caracteres.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

name – nombre de nuevo del documento

fileExt - extensión del documento

pathDocumentFile - path del documento

# Clases de carpetas (y 35)

## Métodos de la clase FolderObject (continuación):

- ```
public int addDocument(String name,  
                        String parentDivId,  
                        String fileExt,  
                        String pathDocumentFile)  
    throws java.lang.Exception
```

Añade un nuevo documento al clasificador padre especificado. Se produce una excepción si nombre del documento ya existe o tiene más de 32 caracteres.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

name – nombre de nuevo del clasificador  
parentDivId - identificador del clasificador padre  
fileExt - extensión del documento  
pathDocumentFile - path del documento

# Clases de carpetas (y 36)

## Métodos de la clase FolderObject (continuación):

- `public void renameDocument(int docId, String name)`  
`throws java.lang.Exception`

Renombra un documento de la carpeta. Se produce una excepción si no se encuentra el documento, ya existe un documento hermano con el mismo nombre o si el nuevo nombre excede 32 caracteres.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

docId - identificador del documento

name – nuevo nombre del documento

# Clases de carpetas (y 37)

## Métodos de la clase FolderObject (continuación):

- `public void removeDocument(int docId)`  
`throws java.lang.Exception`

Elimina un documento de la carpeta. Se produce una excepción si no se encuentra el documento.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

docId - identificador del documento a borrar

# Clases de carpetas (y 38)

## Métodos de la clase FolderObject (continuación):

- `public FolderFdrLinkObjects getAllFolderLinks()`

`throws java.lang.Exception`

Recupera la colección con todos los accesos directos a otras carpetas de la carpeta.

- `public FolderFdrLinkObjects getRootFolderLinkChildren()`

`throws java.lang.Exception`

Recupera la colección con los accesos directos a otras carpetas hijos del nodo raíz de clasificadores-documentos de la carpeta.

# Clases de carpetas (y 39)

## Métodos de la clase FolderObject (continuación):

- `public FolderFdrLinkObjects getFolderLinkChildren(  
int parentDivId)  
throws java.lang.Exception`

Recupera la colección con los datos de los accesos directos a carpetas hijos de un clasificador dado.

Donde:

parentDivId - identificador del clasificador padre



# Clases de carpetas (y 40)

## Métodos de la clase FolderObject (continuación):

- `public FolderFdrLinkObject getFolderLink(int fdrLinkId)`  
`throws java.lang.Exception`

Recupera la información sobre un acceso directo de la carpeta. Lanza una excepción si no se encuentra el enlace.

Donde:

fdrLinkId - identificador del acceso directo





# Clases de campos

**Class FolderFieldObjects ([ieci.tecdoc.sbo.idoc.api](http://ieci.tecdoc.sbo.idoc.api))**

**Esta clase encapsula los valores de los campos de una carpeta.**

# Clases de campos (y 2)

## Métodos de la clase FolderFieldObjects :

- `public ieci.tecdoc.sbo.idoc.folder.base.FolderTokenFlds  
getFieldsToken()`

Devuelve los campos de una carpeta.

- `public java.lang.Object getFieldValue(int fldId)  
throws java.lang.Exception`

Retorna la referencia de un objeto que contiene el valor del campo. Lanza una excepción si se produce un error obteniendo el valor.

Donde:

fldId - identificador del campo

# Clases de campos (y 3)

## Métodos de la clase FolderFieldObjects (continuación):

- `public java.util.ArrayList getFieldValues(int fldId)  
 throws java.lang.Exception`

Retorna la referencia a una lista de objetos que contienen los valores asociados al campo. Lanza una excepción si se produce un error obteniendo los valores.

Donde:

fldId - identificador del campo

Los 9 tipos de campos w@rdA son encapsulados en los siguientes objetos:

| Tipo campo w@rdA | Objeto Java | Tipo campo w@rdA | Objeto Java |
|------------------|-------------|------------------|-------------|
| Fecha            | Date        | Short int        | Short       |
| Fecha/Hora       |             | Long int         | Integer     |
| Hora             |             | Short dec        | Float       |
| Texto corto      | String      | Long dec         | Double      |
| Texto largo      |             |                  |             |

# Clases de documentos

## Class FolderDocumentObjects (ieci.tecdoc.sbo.idoc.api)

Esta clase encapsula la lista de documentos que forman parte del árbol de nodos de la carpeta.

# Clases de documentos (y 2)

## Métodos de la clase FolderDocumentObjects :

- `public int count()`

Devuelve el numero de documentos de la lista.

- `public FolderDocumentObject get(int index)`

Devuelve el documento con el índice especificado.

Donde:

index - índice del documento

# Clases de documentos (y 3)

**Class FolderDocumentObject (ieci.tecdoc.sbo.idoc.api)**

Contiene la información referente a un documento de una carpeta.

# Clases de documentos (y 4)

## Métodos de la clase FolderDocumentObject :

- `public int getId()`

Devuelve el identificador del documento.

- `public String getName()`

Devuelve el nombre del documento.

- `public int getParentId()`

Devuelve el identificador del elemento padre del documento.

- `public String getFileExt()`

Devuelve la extensión del fichero asociado al documento.

- `public int getParentId()`



# Clases de documentos (y 5)

## Métodos de la clase FolderDocumentObject (continuación) :

- `public int getSortOrder()`

Devuelve el número de orden del documento respecto a sus documentos hermanos en el clasificador.

- `public void replaceFile(String pathFile, String fileExt)`  
`throws java.lang.Exception`

Reemplaza el fichero asociado al documento.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

pathFile - path del nuevo documento

fileExt - extension del nuevo documento

# Clases de clasificadores

**Class FolderDividerObjects (ieci.tecdoc.sbo.idoc.api)**

**Esta clase encapsula un árbol de clasificadores para una carpeta.**

# Clases de clasificadores (y 2)

## Métodos de la clase FolderDividerObjects :

- `public int count()`

Devuelve el numero de clasificadores de la lista.

- `public FolderDividerObject get(int index)`

Devuelve el clasificador cuya posición en la lista es la del índice especificado.

Donde:

index – posición del clasificador dentro de la lista

# Clases de clasificadores (y 3)

**Class FolderDividerObject (ieci.tecdoc.sbo.idoc.api)**

**Contiene la información referente a un clasificador de carpeta.**

# Clases de clasificadores (y 4)

## Métodos de la clase FolderDividerObject :

- `public int getId()`

Devuelve el identificador del clasificador.

- `public String getName()`

Devuelve el nombre del clasificador.

- `public int getParentId()`

Devuelve el identificador del elemento padre del clasificador.

# Clases de enlaces

## Class FolderFdrLinkObjects (ieci.tecdoc.sbo.idoc.api)

Esta clase encapsula una colección de enlaces a carpetas y los métodos para obtener cada uno de sus enlaces.

# Clases de enlaces (y 2)

## Métodos de la clase FolderFdrLinkObjects :

- `public int count()`

Devuelve el numero de accesos directos de la lista.

- `public FolderFdrLinkObject get(int index)`

Devuelve el acceso directo cuya posición en la lista es la especificada por el parámetro *index*.

Donde:

*index* - índice del documento



# Clases de enlaces (y 3)

**Class FolderFdrLinkObject (ieci.tecdoc.sbo.idoc.api)**

**Contiene la información referente a un acceso directo de una carpeta.**

# Clases de enlaces (y 4)

## Métodos de la clase FolderFdrLinkObject :

- `public int getId()`

Devuelve el identificador del acceso directo.

- `public String getName()`

Devuelve el nombre del acceso directo.

- `public int getParentId()`

Devuelve el identificador del elemento padre del acceso directo.

# Clases de enlaces (y 5)

## Métodos de la clase FolderFdrLinkObject (continuación) :

- `public int getSrvArchId()`

Devuelve el identificador del archivador al que pertenece la carpeta asociada al acceso directo.

- `public String getSrvArchName()`

Devuelve el nombre del archivador al que pertenece la carpeta asociada al acceso directo.

- `public int getSrvFdrId()`

Devuelve el identificador de la carpeta asociada al acceso directo.

# Clases de enlaces (y 5)

## Métodos de la clase FolderFdrLinkObject (continuación) :

- `public int getSortOrder()`

Devuelve el número de orden del documento respecto a sus documentos hermanos en el clasificador.

- `public void replaceFile(String pathFile, String fileExt)  
throws java.lang.Exception`

Reemplaza el fichero asociado al documento.

**Los cambios no son efectivos hasta que se llama a uno de los métodos *storeFolder* o *createFolder*.**

Donde:

pathFile - path del nuevo documento

fileExt - extension del nuevo documento

# Clases de búsqueda

## Class FolderSearch (ieci.tecdoc.sbo.idoc.api)

Esta clase actúa como Gestor de búsquedas de carpetas.

Proporciona un conjunto de métodos para la búsqueda de carpetas de un archivador que cumplan criterios relacionales o documentales.

# Clases de búsqueda (y 2)

## Constructores:

**FolderSearch()** throws `java.lang.Exception`

**FolderSearch**(`java.lang.String configDir`)  
throws `java.lang.Exception`

## Donde:

`configDir` – ruta donde se encuentra la configuración de la base de datos  
(`leciTd_DbConn_Cfg.xml`)

# Clases de búsqueda (y 3)

## Métodos de la clase FolderSearch:

- `public void setConnectionConfig(DbConnectionConfig dbConnConfig) throws java.lang.Exception`

Fija los parámetros de conexión a la base de datos que utilizan los métodos de esta clase. La configuración corresponderá con la de la base de datos donde se encuentren los usuarios w@rdA.

Donde:

dbConnConfig - Configuración de la conexión de base de datos



# Clases de búsqueda (y 4)

## Métodos de la clase FolderSearch (continuación):

- ```
public ieci.tecdoc.sbo.idoc.folder.search.FolderSearchResult  
    executeQuery(AcsAccessObject acs,  
                ArchiveObject arch,  
                FolderSearchQueryObject query)  
    throws java.lang.Exception
```

Realiza una búsqueda de carpetas dentro de un archivador, devolviendo la referencia a una colección que contiene los resultados de la búsqueda (con ids de carpetas).

Donde:

- acs - Opcional. Contiene información del usuario que realiza la búsqueda. Si se pasa, sólo se obtienen carpetas que, además de cumplir los filtros de la búsqueda, sean visibles para dicho usuario
- arch - información del archivador sobre el cual se van a buscar carpetas.
- query - contiene los filtros que se van a aplicar a la búsqueda

# Clases de búsqueda (y 5)

## Métodos de la clase FolderSearch (continuación):

- ```
public ieci.tecdoc.sbo.idoc.folder.search.FolderSearchResult  
    executeQuery(AcsAccessObject acs,  
                ArchiveObject arch,  
                String qual)  
    throws java.lang.Exception
```

Realiza una búsqueda de carpetas dentro de un archivador, devolviendo la referencia a una colección que contiene los resultados de la búsqueda (con ids de carpetas).

Donde:

- acs - Opcional. Contiene información del usuario que realiza la búsqueda. Si se pasa, sólo se obtienen carpetas que, además de cumplir los filtros de la búsqueda, sean visibles para dicho usuario
- arch - información del archivador sobre el cual se van a buscar carpetas.
- qual - condición sql que define los filtros de búsqueda de carpetas

# Clases de búsqueda (y 6)

## Métodos de la clase FolderSearch (continuación):

- ```
public FolderFieldObjects getFolderValues(  
    ArchiveObject arch,  
    FolderSearchResult rs,  
    int idx)  
    throws java.lang.Exception
```

Devuelve los valores de los campos asociados a la carpeta i-ésima de una colección de carpetas obtenida como resultado de una búsqueda dentro de un archivador.

Donde:

arch - información del archivador al cual pertenece la carpeta

rs - resultado de una búsqueda de carpetas sobre un archivador

idx - índice de la carpeta sobre la cual obtener los valores de sus campos



# Clases de búsqueda (y 7)

## Class FolderSearchQueryObject (ieci.tecdoc.sbo.idoc.api)

Clase que encapsula los filtros de búsqueda de carpetas dentro de un archivador y el conjunto de campos por los cuales se ordena los resultados de una búsqueda.

# Clases de búsqueda (y 8)

## Constructores:

```
FolderSearchQueryObject(ArchiveObject arch)  
throws java.lang.Exception
```

## Donde:

arch - archivador sobre el que se va a realizar la búsqueda

# Clases de búsqueda (y 9)

## Métodos de la clase FolderSearchQueryObject :

- `public java.lang.String getSqlWhere(int dbEngine)`  
`throws java.lang.Exception`

Devuelve la condición sql que se utilizará como filtro en la búsqueda de carpetas dentro de un archivador.

Donde:

dbEngine - tipo de base de datos (Oracle, SQL\_Server)

- `public java.lang.String getSqlOrderBy()`  
`throws java.lang.Exception`

Devuelve la cláusula ORDER BY que define la ordenación de las carpetas resultado de una búsqueda.



# Clases de búsqueda (y 10)

## Métodos de la clase FolderSearchQueryObject (continuación) :

- `public java.lang.String getSqlQual(int dbEngine)`

`throws java.lang.Exception`

Devuelve una cadena que representa las condiciones de búsqueda de carpetas dentro de un archivador y la cláusula ORDER BY para la ordenación de los resultados de la búsqueda.

Donde:

dbEngine - tipo de base de datos (Oracle, SQL\_Server)





# Clases de búsqueda (y 12)

## Métodos de la clase FolderSearchQueryObject (continuación) :

- `public void addSearchOrder(int fldId,  
boolean desc)  
throws java.lang.Exception`

Añade un campo dentro del conjunto de campos por los cuales se va a ordenar la búsqueda.

Donde:

fldId - identificador del campo

desc - la búsqueda es descendente

# FIN