JUNTA DE ANDALUCIA

CONSEJERÍA DE ECONOMÍA, HACIENDA Y ADMINISTRACIÓN PÚBLICA

Conector genérico para ContentManager

Trew@

Dirección General de Política Digital

Consejería de Economía Hacienda y Administración Pública

Versión:1.0.0

Fecha: 28/01/19

HOJA DE CONTROL

Título	Manual implementación conector genérico		
Entregable	Manual implementación conector genérico		
Nombre del Fichero	TRW_OTR_Manual_Implementacion_Conector_ContentManager_v01r00.odt		
Autor	CEHAP		
Versión/Edición	v01r00	**Fecha Versión**	28/01/19
Aprobado por		**Fecha Aprobación**	28/01/2019
		Nº Total Páginas	-

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Área	Fecha del Cambio
v01r00	Versión inicial	UTE	UTE	28/01/19

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos	Cargo	Área
Manuel Perera Domínguez	Jefe de Servicio	Servicio de Coordinación de Administración Electrónica
Francisco González Guillén	Director de Proyecto	Servicio de Coordinación de Administración Electrónica
Francisco Mesa Villalba	Director de Proyecto	Servicio de Coordinación de Administración Electrónica

Pedro José Casanova Luis	Jefe de Proyecto	UTE

ÍNDICE

indice

Índice de tablas

Índice de ilustraciones

1 INTRODUCCIÓN

1.1 Objeto

El propósito del presente documento es mostrar la versatilidad que ofrece *Trew@* a la hora de realizar una conexión con cualquier gestor documental.

Además, presentar de forma técnica las configuraciones e implementaciones necesarias para poder realizar esta conexión.

Por último, se mostrará como ejemplo la implementación de un conector para el gestor documental *Consigna*.

1.2 Alcance

Este documento va dirigido a personas titulares de los servicios y unidades TIC de la Junta de Andalucía y sus entidades instrumentales con el objetivo de presentar la versatilidad que ofrece *Trew@*.

2 DESCRIPCIÓN GENERAL DEL SISTEMA

Un gestor documental es un repositorio externo que permite almacenar, administrar y controlar el flujo de documentos.

En la actualidad el motor de tramitación conocido como *Trew@* puede almacenar documentos en su propia base de datos y también en un gestor documental para satisfacer las necesidades del cliente.

Cada organismo puede decidir en qué repositorio externo realizar el seguimiento, la gestión y el almacenamiento de sus propios documentos sin influir en el funcionamiento de *Trew@* o prescindir de este y realizarlo en la propia base de datos de *Trew@*.

3 ARQUITECTURA LÓGICA DEL SISTEMA

En este apartado se presenta la estructura de *Trew@*, es decir, las interfaces y clases implementadas para dar forma al conector genérico y la configuración que habría que realizar en la Herramienta de Administración de *Trew@* para el uso de un nuevo conector para ***ContentManager***.

Posteriormente se mostrará cómo se ha implementado el conector con *Consigna* y que funciones permite realizar.

La explicación de la implementación de un nuevo conector para ***ContentManager*** la dividiremos en módulos “trewa-api” y “trewa-contentmanager”, dentro de ellos explicaremos las clases e interfaces que son necesarias implementar.

3.1 Módulo trewa-api

Dentro del módulo “trewa-api” estudiaremos la clase ***ContentManagerFacade*** y la interfaz ***ContentManager.***

3.1.1 ContentManagerFacade

ContentManagerFacade es una clase que se encuentra en el módulo “trewa-api” en la ruta “src/main/java” dentro del paquete “trewa.comp.contentmanager”.

Algunos de los métodos implementados en la Api de *Trew@* como crearDocumentoCabecera, crearDocumentoAnexo, creaFirmaAnexo, modificarDocumentoCabecera, borrarDocumento, consultarDocumento, crean una instancia de la clase ***ContentManagerFacade*** para poder hacer uso de los métodos de esta.

En esta clase los métodos implementados se encargan de obtener los datos del componente de *Trew@* almacenarlos en un properties y posteriormente crear la configuración con esos datos.

```
/**
 * Elimina el documento indicado.
 *
 * @param idDocumento uuid del documento.
 * @throws ContentManagerException
 */
public void borrarDocumento(String idDocumento)
        throws ContentManagerException {
    manager.remove(idDocumento, "default", (Object[])null);
}
```

Ilustración 1: Llamada método save de ContentManager.

ContentManagerFacade se encarga de realizar llamadas a métodos de la interfaz ***ContentManager*** mediante una instancia de esta.

A continuación, se presenta la interfaz ***ContentManager.***

3.1.2 ContentManager

La interfaz ***ContentManager*** se encuentra en el mismo paquete que el ***ContentManagerFacade*** "trewa.comp.contentmanager", esta interfaz declara métodos necesarios para trabajar con el repositorio externo.

ContentManager es implementada por la clase ***AbstractContentManager.***

3.2 Módulo Trewa-contentmanager

En el módulo "trewa-contentmanager" es donde existen mayor número de cambios a la hora de implementar el nuevo conector. Dentro de "trewa-contentmanager" realizaremos una implementación de las clases ***AbstractContentManagerStrategy*** y ***AbstractContentManager*** y las interfaces ***ContentManagerCmisApi*** y ***ContentManagerStrategy.***

Antes de comenzar a tratar directamente cada una de ellas, se presenta un diagrama de clases en el que se recoge la información de relevancia sobre el módulo "trewa-contentmanager" y como se relaciona con el módulo "trewa-api".

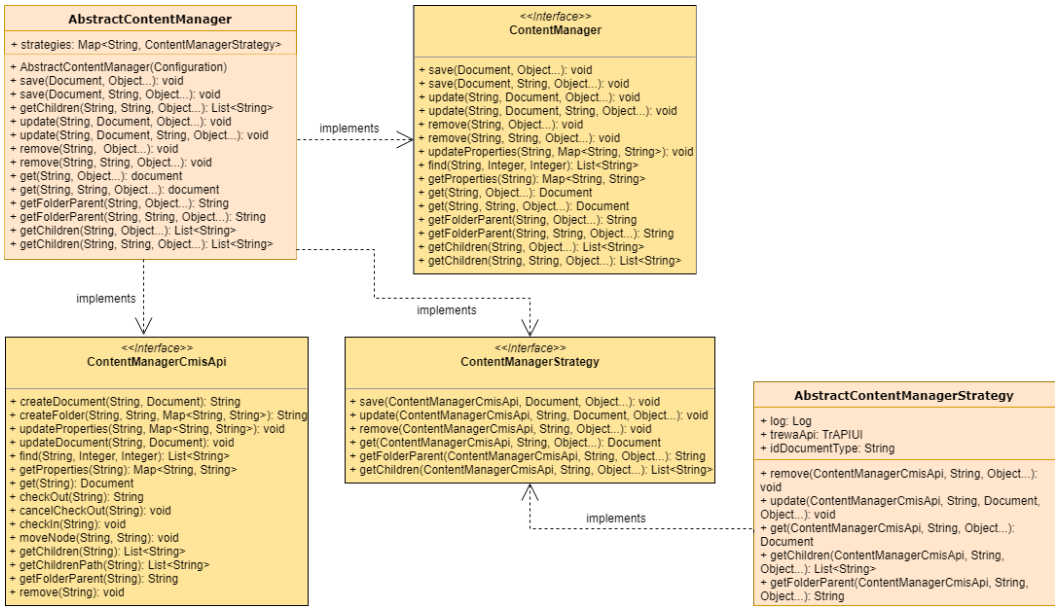


Ilustración 2: Diagrama de clases módulo trewa-contentmanager.

En el diagrama anterior se puede observar la estructura del módulo “trewa-contentmanager”, dentro de este se encuentran las clases ***AbstractContentManager***, ***AbstractContentManagerStrategy*** junto con interfaces ***ContentManagerCmisApi*** y ***ContentManagerStrategy***.

La relación entre ambos módulos se realiza por medio de la interfaz ***ContentManager*** y la clase ***AbstractContentManager***.

3.2.1 AbstractContentManager

La clase ***AbstractContentManager*** sirve para relacionar el módulo con la API de T*rew@* implementando la interfaz de ***ContentManager***.

Esta clase se encuentra en la ruta “src/main/java” en el paquete “trewa.contentmanager” y no solo implementa la interfaz ***ContentManager*** también la interfaz ***ContentManagerCmisApi.***

```
/**
 * The Class AbstractContentManager.
 */
public abstract class AbstractContentManager implements ContentManager,
        ContentManagerCmisApi, Serializable {
```

Ilustración 3: AbstractContentManager implementación.

Aquellos métodos que se encarga de implementar la clase ***AbstractContentManager*** se pueden observar en el diagrama de clases anterior.

```
/**
 * Insert a document using the default strategy.
 *
 * @param document document to insert.
 * @param strategyKey Identifier of the strategy to use.
 * @param parameters Array of objects properties used in the
 *            strategy implementation..
 * @throws ContentManagerException If something go wrong.
 */
public void save(Document document, String strategyKey,
        Object... parameters) throws ContentManagerException {

    if (strategies != null && strategies.containsKey(strategyKey)) {
        ContentManagerStrategy strategy = strategies.get(strategyKey);

        strategy.save(this, document, parameters);

    } else {
        throw new ContentManagerException(
                "The chosen strategy cannot be found, key: " + strategyKey);
    }
}
```

Ilustración 4: Ejemplo método ContentManagerStrategy.

Esta clase utiliza aquellos métodos declarados en la interfaz ***ContentManagerStrategy*** pasándole como uno de sus parámetros la referencia “this” la cual siempre apunta al objeto sobre el que se ejecuta el método.

3.2.2 ContentManagerCmisApi

Esta interfaz es implementada como se comentó anteriormente por la clase ***AbstractContentManager***.

ContentManagerCmisApi es una interfaz que basada en el estándar *JSR-170,* este estándar para interfaces de programación de aplicaciones de plataforma Java se usa para acceder a los repositorios de contenido de manera uniforme. Permite el acceso a cualquier repositorio compatible de manera neutral, por ello es necesaria su implementación en ***AbstractContentManager.***

3.2.3 ContentManagerStrategy

ContentManagerStrategy es una interfaz en la que se encuentran declarados los siguientes métodos:

- *Save*
- *Update*
- *Remove*
- *Get*
- *getFolderParent*
- *getChildren*

```
/**
 * Method invoked before a document will be saved in the repository. This
 * method must return the repository node id where the document will be
 * saved.
 *
 * @param contentManager Initialized manager to work with the repository.
 * @param document Document to insert.
 * @param parameters Parameters to be used by the strategy.
 * @throws ContentManagerException the content manager exception
 */
public void save(ContentManagerCmisApi contentManager, Document document,
        Object... parameters) throws ContentManagerException;
```

Ilustración 5: Ejemplo declaración método contentManagerStrategy.

Como vemos, la imagen superior, presenta la declaración de uno de los métodos de la interfaz ***ContentManagerStrategy,*** el primer elemento pasado por parámetro es una instancia de ***ContentManagerCmisApi***.

La implementación de los métodos declarados en la interfaz ***ContentManagerStrategy*** depende del gestor documental con el que se realice la conexión, es decir, cada repositorio externo con el que se quiera realizar una conexión debe tener su propia Strategy (***ConsignaStrategyImpl***, ***DefaultStrategyImpl***, etc.). En esta clase se implementarán los métodos anteriores apoyándose en los métodos proporcionados por la API del gestor documental para cumplir con las funcionalidades que esté te permite realizar.

3.3 Arquitectura conector consigna

3.3.1 ConsignaService

Para explicar cómo se ha realizado la conexión con C*onsigna* primero es necesario indicar que este repositorio posee una arquitectura REST en el diseño de su API, debido a que utiliza directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre sus datos.

ConsignaService implementa todas las funcionalidades que te permite realizar la API de *Consigna*. Sus funciones son las siguientes:

- *Autenticación*
- *subir documentos*
- *descargar documentos*
- *borrar fichero*
- *modificar fichero*
- *obtener información del documento*
- *obtener documento.*

3.3.1.1 Obtener documento

Método	*obtenerDocumento*
Parámetros recibidos	• *String fid* • *String usuario* • *String password* • *String url*
Devuelve	*inputStream.*
Petición	***HTTP POST.***
Tipos de respuestas	➔ *200 OK.* ➔ *404 Not Found* ➔ *400 BadRequest*
Observaciones	*Se dedica a obtener un fichero identificado por el fid que se le pase.*

Tabla 1: método obtenerDocumento

3.3.1.2 Subir documento

Método	*subirFichero*
Parámetros recibidos	• *String usuario* • *String password* • *String url* • *String expiracion.* • *String descripcion.* • *String pass* • *String uri*
Devuelve	*String.*
Petición	***HTTP POST.***
Tipos de respuestas	➔ *201 CREATED.* ➔ *404 Not Found* ➔ *400 BadRequest*
Observaciones	*El objetivo de este método es almacenar un fichero en consigna.*

Tabla 2: método subirFichero

3.3.1.3 Borrar archivo

Método	*borrarFichero*
Parámetros recibidos	• *String usuario* • *String password* • *String url* • *String fid*
Devuelve	*Void.*
Petición	***HTTP DELETE.***
Tipos de respuestas	➔ *204 NO CONTENT.* ➔ *404 Not Found*
Observaciones	*El objetivo de este método es eliminar un documento de consigna.*

Tabla 3: método borrarFichero

3.3.1.4 Modificar fichero

Método	*modificarFichero*
Parámetros recibidos	• *String usuario* • *String password* • *String url* • *String fid* • *String pass* • *String descripcion*
Devuelve	*Void.*
Petición	***HTTP PUT.***
Tipos de respuestas	➔ *201 CREATED.* ➔ *400 BadRequest* ➔ *404 Not Found*
Observaciones	*El objetivo de este método es realizar modificaciones en la descripción o en la contraseña de un fichero almacenado en consigna.*

Tabla 4: método modificarFichero

3.3.1.5 Obtener información de un fichero

Método	*obtenerInfoDocumentos*
Parámetros recibidos	• *String usuario* • *String password* • *String url*
Devuelve	*List<InfoDocConsigna>*
Petición	***HTTP GET.***
Tipos de respuestas	➔ *200 OK.*
Observaciones	*El objetivo de este método es obtener información de todos los documentos almacenados en consigna.*

Tabla 5: método obtenerInfoDocumentos

3.3.1.6 Descargar documento

Método	*descargarDocumento*
Parámetros recibidos	• *String usuario* • *String password* • *String url* • *String fid* • *String pass.*
Devuelve	*Byte[]*
Petición	***HTTP POST.***
Tipos de respuestas	➔ *200 OK.* ➔ *400 BadRequest* ➔ *404 Not Found*
Observaciones	*El objetivo de este método es obtener un fichero almacenado en consigna.*

Tabla 6: método descargarDocumento

3.3.2 ConsignaServiceImpl

ConsignaServiceImpl se encuentra en el módulo "trewa-contentmanager" alojado en la ruta "src/main/java" dentro del paquete "trewa.contentmanager.impl".

Dentro de los diferentes métodos implementados en esta clase se realizan llamadas a ***ConsignaService***, debido a que en ***ConsignaServiceImpl*** se encuentran implementadas cada una de las funcionalidades ofrecidas por la API de *Consigna*.

3.3.2.1 Update properties

Método	*updateProperties*
Parámetros recibidos	• *String fid* • *Map<String, String> properties*
Devuelve	*void*
Llamadas a métodos de ConsignaService	• *modificarFichero*

Tabla 7: método updateProperties

3.3.2.2 Get properties

Método	*getProperties*
Parámetros recibidos	• *String fid*
Devuelve	*Map<String, String>*
Llamadas a métodos de ConsignaService	• *obtenerInfoDocumento*

Tabla 8: método getProperties

3.3.2.3 Create document

Método	*createDocument*
Parámetros recibidos	• *String fid* • *Document document*
Devuelve	*String*
Llamadas a métodos de ConsignaService	• *subirFichero*

Tabla 9: método createDocument

3.3.2.4 Update document

Método	*updateDocument*
Parámetros recibidos	• *String fid* • *Document document*
Devuelve	*void*
Llamadas a métodos de ConsignaService	• *borrarFichero* • *subirFichero*

Tabla 10: método updateDocument

3.3.2.5 Get

Método	*get*
Parámetros recibidos	• *String fid* •
Devuelve	*Document*
Llamadas a métodos de ConsignaService	• *descargarDocumento*

Tabla 11: método get

3.3.2.6 Remove

Método	*remove*
Parámetros recibidos	• *String fid* •
Devuelve	*void*
Llamadas a métodos de ConsignaServic e	• *borrarFichero*

Tabla 12: método remove

3.3.3 ConsignaStrategyImpl

Tras conocer la arquitectura que posee la API de *Consigna* se tratará la implementación realizada en ***ConsignaStrategyImpl.***

Cada conector a implementar para el uso de un gestor documental en *Trew@* debe tener su propia implementación de la interfaz ***AbstractContentManagerStrategy***. Todo conector debe implementar los mismos métodos de esta clase de estrategia ***save, get, update*** adaptándose a las funcionalidades aportadas por la API del gestor documental, es decir, apoyandonse en los métodos implementados en las clases ***ConsignaServiceImpl*** y ***ConsignaService***.

SAVE

Este método recibe los siguientes parámetros:

```
public void save(ContentManagerCmisApi contentManager, Document document,
        Object... parameters) throws ContentManagerException {
```

Ilustración 6: Cabecera método save.

El método te permite guardar documentos en *Consigna*. El almacenamiento se realiza en dos pasos (existen 2 llamadas al método a la hora de guardar un documento en *Consigna*). Estas dos llamadas son necesarias porque la arquitectura de *Trew@* está orientada hacia *Alfresco*. Según los servicios publicados por *Alfresco* es necesario una primera llamada para crear una carpeta y posteriormente una segunda llamada en la que crea y añade un documento a la carpeta que posteriormente es subida.

Consigna al no proveer de los mismos servicios que *Alfresco* en la primera llamada al método no puede crear carpetas (no contiene esta funcionalidad), pese a no crear la carpeta es necesario la devolución de un *identificador* debido a que la arquitectura para reproducir el comportamiento durante el primer paso que aporta *Alfresco*, como se explica en el siguiente párrafo.

En la segunda llamada al método, comprueba si existe un identificador y si es así llama al método ***update*** implementado en el fichero ***ConsignaStrategyImpl.***

Dentro del método ***save*** también existe la posibilidad de que el documento recibido como parámetro tenga contenido y dependiendo de si tiene identificador o no se llamará al método ***updateDocument*** o ***createDocument*** del fichero ***ConsignaServiceImpl.***

La última posibilidad dentro del método será que el documento recibido no tenga contenido pero si firma y por tanto realiza una llamada a ***createDocument*** de la clase ***ConsignaServiceImpl*.**

UPDATE

El método ***update*** recibe los siguientes parámetros.

```
public void update(ContentManagerCmisApi contentManager, String uuid,
        Document document, Object... parameters)
        throws ContentManagerException {
```

Ilustración 7: Cabecera método update.

Update solo realiza una llamada al método ***updateProperties*** de la clase ***ConsignaServiceImpl***.

GET

*El método **get** recibe los siguientes parámetros.*

```
public Document get(ContentManagerCmisApi contentManager,
        String identifier, Object... parameters)
        throws ContentManagerException {
```

Ilustración 8: Cabecera método get.

Get solo realiza una llamada al método ***get*** de la clase ***ConsignaServiceImpl***.

4 CONFIGURACIÓN DEL SISTEMA

En este apartado se presentará la configuración a realizar para un nuevo conector de gestor documental en *Trew@*.

Una vez escojamos el sistema en el que se quiera crear el nuevo conector, accedemos a ***Configuración*/*Componentes*** en el menú situado a la izquierda.

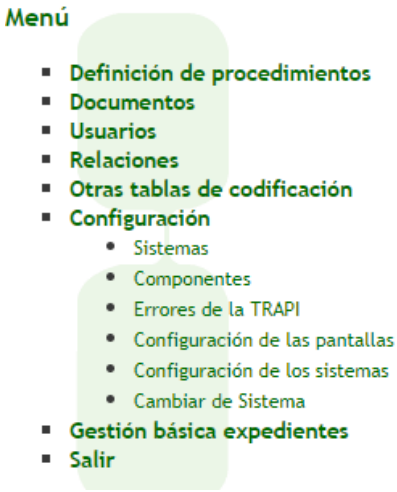
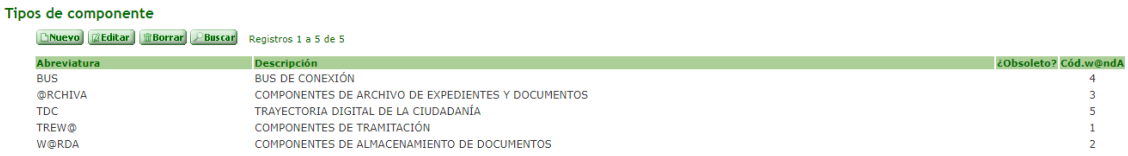


Ilustración 9: Menú principal Trew@.

Dentro de esta pestaña necesitaremos seleccionar el tipo de componente, para crear un conector con un gestor documental tenemos que acceder a ***W@rDA*** (componentes de almacenamiento de documentos).



Abreviatura	Descripción	¿Obsoleto?	Cód.w@ndA
BUS	BUS DE CONEXIÓN		4
@RCHIVA	COMPONENTES DE ARCHIVO DE EXPEDIENTES Y DOCUMENTOS		3
TDC	TRAYECTORIA DIGITAL DE LA CIUDADANÍA		5
TREW@	COMPONENTES DE TRAMITACIÓN		1
W@RDA	COMPONENTES DE ALMACENAMIENTO DE DOCUMENTOS		2

Ilustración 10: Tipos de componentes.

Una vez seleccionado el tipo de componente ***W@RDA*** tendremos que crear un nuevo componente rellenando los valores de la pestaña “Componente” y “Datos del componente”.

Por último en la configuración de Trew@ tenemos que acceder mediante el menú a ***Configuración/Sistemas*** elegir el sistema y seleccionar el componente en el que queremos guardar los documentos.

En el siguiente apartado se trata la configuración necesaria para configurar *Consigna* como gestor documental.

4.1 Configuración en Trew@ para consigna

La configuración específica de la pestaña "componente" para *Consigna* será la siguiente.

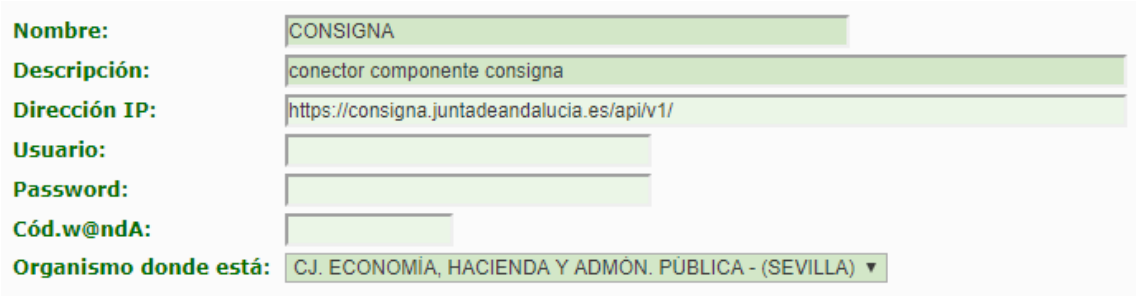


Ilustración 11: Componente.

- Dirección IP, es la url a la que se realizarán peticiones para obtener datos o ejecutar operaciones sobre ellos.
- Usuario y Password tienen que ser rellenados con los datos de acceso al gestor documental.

La pestaña "Datos del componente" tendrá que configurarse de la siguiente forma.

Atributo	Valor
CLASS	trewa.contentmanager.impl.ConsignaServiceImpl
IMPL	consignaws
PROTOCOLO	https
STRATEGYIMPL	trewa.contentmanager.strategy.ConsignaStrategyImpl
TEMPFILE	/opt/ficheros_prueba/

Ilustración 12: Datos del componente.

- *CLASS,* es de vital importancia, ya que configura la ruta donde se encuentra la implementación de las funcionalidades que ofrece el gestor documental. Por tanto una vez configurada esta ruta cuando se llame a los métodos que proporciona la API de consigna dentro del fichero ConsignaStrategyImpl estos harán referencia a la clase ConsignaServiceImpl donde se encuentran implementados estos métodos.
- *PROTOCOLO,* configura el protocolo con el que se realizarán las peticiones a la API de Consigna
- *STRATEGYIMPL,* configura el fichero ConsignaStrategyImpl como la estrategia a seguir.

• *TEMPFILE,* configura el directorio en el que se almacenará temporalmente el fichero que se desea subir a consigna.

5 ANEXOS

5.1 ANEXO 1. Tipificación código de respuesta.

200 OK	*La petición ha resultado satisfactoria.*
201 CREATED	*La petición ha resultado satisfactoria y se ha creado un recurso.*
204 NO CONTENT	*La petición ha resultado satisfactoria y el cliente no tiene que salir de la página actual.*
400 BadRequest	*Errores de validación.*
404 Not Found	*No existe el fichero.*

Tabla 13: Posibles respuestas

5.2 ANEXO 2. Tipificación parámetros.

String *usuario*	*Credencial acceso a consigna.*
String *password*	*Credencial acceso a consigna.*
String *url*	*/api/v1/ficheros.*
String *fid*	*Identificador fichero.*
String pass	*Contraseña del fichero.*
String descripcion	*Texto descriptivo de cada fichero.*
String uri	*Ruta donde devolver el fichero.*
String expiracion.	*Tiempo de expiración.*

Tabla 14: Tipificación parámetros