API Documental

Cliente: SRES

Autor: LeafNoise Company S.A (Christian Coello Grassi)

Leafnoise Company S.A

19 de diciembre de 2016

# Introducción

El objetivo de la solución brindada es ser el único componente que realice todas las operaciones documentales y de transacciones de un expediente. Mediante este único punto de acceso se podrán gestionar todos los expedientes de forma centralizada, pudiéndose acceder al mismo desde cualquier punto de acceso necesario, gracias a ser el mismo un servicio REST.

La API REST generada utiliza como motor de persistencia una base de datos documental lo cual facilita la gestión de gran cantidad de expedientes. Para el guardado de los documentos integrantes de cada expediente se proporciono una interfaz de tal manera que su implementación pueda variar en el tiempo, actualmente siendo la misma en el *filesystem* (sistema de archivos).

# Metodología

El almacén de documentos posee como requerimiento su fácil acceso desde diversos sitios, por lo que luego del análisis se concluyó con la necesidad de que su acceso sea mediante protocolo HTTP.

El esquema de comunicación adoptado fue *Representational State Transfer* (REST - Transferencia de Estado Representacional) con utilización de objetos del tipo *JavaScript Object Notation* (JSON – Notación de Objetos de Javascript). La utilización de este esquema procede de su simplicidad, robustez y fácil despliegue.

Para el almacenaje de los expedientes se utilizó la base de datos documental MongoDB, adecuada para el almacenamiento de documentos por su tiempo de respuesta rápido, fácil implementación y distribución, presentando posibilidades de *clustering* (distribución de datos entre varios nodos para proporcionar *High Avalability* [alta disponibilidad]).

El servicio web generado se encuentra desarrollado en Java por lo que puede correr en cualquier sistema operativo que implemente su máquina virtual de Java, siendo requisito especifico de la API la utilización de la versión 1.7 o superior.

# Consideraciones tecnicas

Para la generación de los servicios REST se utilizo Spring, *framework* (marco de trabajo) de referencia para la creación de este tipo de servicios web. En el afán de mantener la simplicidad se utilizo la vertiente auto configurable de Spring llamada Springboot, logrando así mantener al mínimo la configuración necesaria para la definición de los diversos *endpoints* (extremos de servicio). Los mismo se definen en el próximo apartado.

En cuanto a la persistencia, se utilizo *Object Document Mapping* (ODM – Mapeo Documento a Objeto) para simplificar las operaciones sobre la base de datos. El *framework* elegido para llevar a cabo esta tarea fue Morphia debido a que presenta facilidades para configuración de entidades, así como también una API de sencillo uso pero a su vez poderosa.

Se utilizaron 2 colecciones para almacenar los datos, una para los expedientes y sus documentos y otra para reservar el numero documento de un expediente. La misma tiene tiempo de expiración para cada registro, de tal manera de que la reserva de un numero de documento para un expediente tiene caducidad.

# *Endpoints*

A continuación se definen los *endpoints* definidos, con sus métodos, parámetros, requerimientos y respuestas.

Creación de expediente

* Descripción: Crea un nuevo expediente (puede ser de tipo expediente o libro)
* Lógica: Crea un expediente nuevo en la colección de MongoDB
* Método: POST
* Ruta: **/api/files/manager**
* Requerimiento:

{

"type": "book",

"header":

{

"office": {"id": "H00000001", "text": "Dirección General"},

"category": {"id": "resolucion", "text": "Resolución"},

"year": 2017

}

}

* Respuesta:

{

"success": true,

"result":

{

"id": "wefafkhsejhfgefh34@#$@#4"

}

}

Listado de expedientes

* Descripción: Devuelve una lista de expedientes según su tipo, un numero de página, cantidad de resultados y orden
* Lógica: Para cada documento se necesita devolver la cantidad de actualizaciones y la fecha de la última actualización
* Método: GET
* Ruta: **/api/files/manager/<type>/<page>/<limit>/<order\_filed>/<order\_direction>**
* Respuesta:

{

"success": true,

"result":

[

{

"id": "wefafkhsejhfgefh34@#$@#4",

"type": "book",

"updates": 23,

"year": 2016,

"office": {"id": "H00000001", "text": "Dirección General"},

"last\_update": "25/12/2016",

"category": {"id": "resolucion", "text": "Resolucion"}

}

]

}

Obtención de índice de un expediente especifico

* Descripción: Obtiene el índice para un expediente especifico
* Lógica: Obtiene el expediente con todas sus actualizaciones
* Método: GET
* Ruta: **/api/files/manager/<id>**
* Respuesta:

{

"success": true,

"result":

{

"file":

{

"id": "file\_1",

"type": "book",

"updates": 23,

"year": 2016,

"office": {"id": "H00000001", "text": "Dirección General"},

"last\_update": "25/12/2016",

"category": {"id": "resolucion", "text": "Resolucion"}

},

"index":

[

{

"file\_id": "file\_1",

"doc\_id": "doc\_1"

"parent": "root",

"doc":

{

"path": "",

"name": ""

}

},

{

"file\_id": "file\_1",

"doc\_id": "doc\_2"

"parent": "doc\_1",

"doc":

{

"path": "",

"name": ""

}

}

]

}

}

Obtención de documento de expediente

* Descripción: Obtiene el archivo en formato base 64 (cadena de caracteres con formato base 64 para que después pueda ser transformado a binario)
* Lógica: Obtiene un archivo especifico de un expediente usando el id del expediente y el número de archivo. Envía de forma separada los firmantes del archivo y los adjuntos del mismo.
* Método: GET
* Ruta: **/api/files/manager/<id>/<number>**
* Respuesta:

{

"success": true,

"result":

{

"doc":

{

"b64": "",

"name": ""

},

"signers":

[

{

"name": "Persona1",

"ca": ""

},

{

"name": "Persona2",

"ca": ""

}

],

"atachments":

[

{

"id": "eefrsf",

"name": "nota2.txt",

"path": "/"

},

{

"id": "tdhdref",

"name": "nota3.xls",

"path": "/"

}

]

}

}

Reserva de siguiente número para documento de expediente

* Descripción: reserva un numero de documento en el expediente, lo bloquea e inserta en el archivo brindado el número de documento asignado
* Lógica: Bloquea el expediente para que no haya problemas con los números de documentos, escribe en el mismo el número de documento asignado, devolviéndolo en base 64. También devuelve una llave que debe ser utilizada en la posterior inserción del documento, la cual funciona como validación de inserción de documento reservado
* Método: POST
* Ruta: **/api/files/next\_number/<id>**
* Requerimiento:

{

"b64": "aefsefesf",

"file\_id": "file\_1"

}

* Respuesta:

{

"success": true,

"result":

{

"b64": "bsargaekbhAEFJHaefi",

"number": 4,

"key": "e1d23f53-006c-4731-8357-e506703523a9"

}

}

Añadir un documento al expediente

* Descripción: Añade un documento a un expediente
* Lógica: Inserta un documento en un expediente, comprobando primero que el mismo haya sido solicitado y validando con la llave entregada en la llamada de reserva. Luego de insertado el documento en el expediente, desbloquea el expediente de tal manera que se puedan insertar otros documentos.
* Método: POST
* Ruta: **/api/files/manager/<id>/<number>/<key>**
* Requerimiento:

{

"b64": "aefsefesf",

"name": "document\_1"

}

* Respuesta:

{

"success": true,

"result":

{

"file\_id": "file\_1",

"doc\_id": "doc\_2",

"parent": "doc\_1",

"payload": {},

"doc":

{

"path": "",

"name": ""

}

}

}

Adjuntar archivo a documento

* Descripción: Adjunta un archivo a un documento
* Lógica: Adjunta el archivo al documento, lo guarda y devuelve la ruta y su hash
* Método: POST
* Ruta: **/api/files/helpers/atach\_doc\_to\_pdf**
* Requerimiento:

{

"b64": "aefsefesf",

"atachments":

[

{

"id": "eefrsf",

"name": "nota2",

"extension": "txt",

"path": "/"

},

{

"id": "tdhdref",

"name": "nota3",

"extension": "xls",

"path": "/"

}

]

}

* Respuesta:

{

"success": true,

"result":

{

"doc":

{

"path": "",

"name": "",

"hash": ""

}

}

}

Convertir a pdf

* Descripción: Convierte un documento que puede estar en formato “.xls” , “.xlsx”, ”.doc”, ”.docx” o “.html” a un documento pdf
* Lógica: Convierte el documento, lo guarda y devuelve el hash y la ruta
* Método: POST
* Ruta: **/api/files/helpers/convert\_to\_pdf**
* Requerimiento:

{

"origin": "html", //.xls, .doc, .html, .xlsx, .docx

"html": "<html></html>"

}

* Respuesta:

{

"success": true,

"result":

{

"doc":

{

"path": "",

"name": "",

"hash": ""

}

}

}

Validar si el documento es pdf y en caso afirmativo chequear si se encuentra protegido

* Descripción: Dado un documento suministrado, devuelve si el mismo es pdf y en caso de que lo sea, si el mismo se encuentra protegido o no
* Lógica: Valida la cadena en base 64, detectando su formato. En caso de que el formato sea pdf, chequea si el mismo se encuentra protegido o no y lo informa
* Método: POST
* Ruta: **/api/files/helpers/validations/type\_of\_file**
* Requerimiento:

{

"b64": "ewfsef"

}

* Respuesta:

{

"success": true,

"result":

{

"pdf": true,

"protected": false

}

}

Devolver firmantes de un documento

* Descripcion: Dado un documento suministrado, devuelve los firmantes del mismo
* Logica: Se suministra un documento en base 64 y lee los firmantes del mismo, devolviendo su nombre y su certificado digital
* Metodo: POST
* Ruta: **/api/files/helpers/validations/signers**
* Requerimiento:

{

"b64": "ewfsef"

}

* Respuesta:

{

"success": true,

"result":

{

"signers":

[

{

"name": "Persona1",

"ca": ""

},

{

"name": "Persona2",

"ca": ""

}

]

}

}