 

**Kit de integración de firma con FIRe**

Manual de desarrollo de un sistema gestor de sesiones

Versión: 2.4

|  |  |  |  |
| --- | --- | --- | --- |
| **CONTROL DE VERSIONES** | | | |
| Título | |  | |
| Autor | | Secretaría General de Administración Digital  Ministerio de Asuntos Económicos y Transformación digital | |
| Fecha versión 2.4 | | 08 de octubre de 2020 | |
|  | | | |
| Versión | Fecha | Responsable | Cambios introducidos |
| 2.4 | 08-10-2020 | SGAD | Creación del documento |

ÍNDICE

[1 Objeto del documento 4](#_Toc57912544)

[2 Introducción 5](#_Toc57912545)

[3 Sistema de gestión de sesiones 6](#_Toc57912546)

[3.1 Clases de gestión incorporadas 6](#_Toc57912547)

[3.1.1 Sistema de ficheros 7](#_Toc57912548)

[3.1.2 Base de datos 7](#_Toc57912549)

[3.2 Desarrollo de nuevas clases de gestión 7](#_Toc57912550)

[3.2.1 SessionsDAO 8](#_Toc57912551)

[3.2.2 TempDocumentsDAO 9](#_Toc57912552)

[3.2.3 Alta de las nuevas clases de gestión de sesiones 11](#_Toc57912553)

[3.3 Configuración de las clases de gestión de sesiones 11](#_Toc57912554)

# Objeto del documento

El presente manual detalla el funcionamiento del sistema de compartición de sesiones de FIRe para entornos balanceados, describe los mecanismos incorporados de forma nativa en FIRe y el cómo desarrollar, integrar y configurar nuevos mecanismos.

# Introducción

FIRe es un sistema para la generación de firmas electrónicas con certificado de usuario. Las aplicaciones web que deseen integrar la firma electrónica de datos como parte de su flujo de operación pueden utilizar FIRe para tal fin.

FIRe se compone principalmente de un API cliente (Componente distribuido) y unos servicios de firma en servidor (Componente Central). Por medio de este API cliente es posible invocar a funciones para la firma de uno o varios documentos (firma de lote) y posteriormente recuperar el resultado de estas operaciones. La principal ventaja de FIRe es que permite a los usuarios firmar tanto con sus certificados locales como con sus certificados en la nube sin que la aplicación que lo utiliza tenga que definir flujos de trabajo distintos para cada una de estas opciones.

Para que el componente central admita las peticiones realizadas por una aplicación, esta deberá haberse registrado en el sistema. Este registro puede hacerlo un administrador a través del módulo de administración de FIRe, mediante el cual dará de alta la nueva aplicación, establecerá el certificado con el que deberá autenticarse y obtendrá como resultado el código alfanumérico que deberá utilizar el componente distribuido como identificador de aplicación (AppId).

Toda operación de firma de firma simple u operación masiva realizada por FIRe se corresponde con una transacción. Esta transacción consta de múltiples llamadas realizadas por el API cliente al componente central de FIRe. En la primera llamada de una transacción, el componente central le asigna un identificador de transacción y este deberá seguir utilizándose en cada una de las llamadas subsiguientes.

Dada la lógica de la operativa de FIRe, es común que en un entorno balanceado las distintas operaciones de una transacción se distribuyan a nodos distintos de la red. En ese caso, posible que un nodo no disponga de los datos de una transacción ya iniciada o que otro no disponga de los datos cargados en una llamada anterior a otro nodo. Para solventar esta problemática, se ha desarrollado un sistema de compartición de sesiones mediante el cual se puede mantener una copia de los datos de todas las transacciones de forma que todos los nodos balanceados tienen acceso a ellas.

# Sistema de gestión de sesiones

El sistema de gestión de sesiones permite comprobar si la transacción a la que pertenece una petición recibida está cargada en memoria. En caso de estarlo, comprueba si esa petición es la que se esperaba. En caso de no estar cargada o si no era una petición esperada, se comprueba si la sesión se inició en otro nodo del entorno o si alguna de las operaciones se ha realizado previamente en otro nodo. Cuando el sistema detecta que el nodo no dispone de toda la información de la sesión, acude al conjunto de sesiones para recuperar esta información.

Hay dos tipos de datos que se pueden almacenar en el conjunto de sesiones:

* Los datos de configuración y resultados de la sesión.
* Los documentos firmados y las firmas generadas.

La principal diferencia entre estos tipos de datos es el tamaño que pueden adquirir. Mientras que los datos de configuración y los resultados obtenidos son siempre de pequeño tamaño, los documentos enviados a firmar y las firmas generadas pueden ser de gran tamaño.

El componente central de FIRe permite configurar mediante qué mecanismo se compartirá cada uno de estos tipos de datos. Para ello, el sistema de gestión de sesiones utiliza una clase de gestión distinta para cada uno de estos tipos de datos y permite configurarlas por separado desde el fichero de configuración del componente central.

El propio sistema se encarga de borrar los documentos a firmar y la configuración de la operación en el momento en el que se generan las firmas y las firmas generadas y la sesión en el momento en el que se recuperan. De esta forma, se evita que queden datos residuales en el sistema de compartición que se haya seleccionado. Adicionalmente, cada cierto número de operaciones, se revisan los datos almacenados en el sistema de compartición. En caso de detectar que estos recursos perteneciesen a transacciones ya caducadas, se eliminaran del sistema.

Cabe destacar que, si se despliega el componente central de FIRe en un único nodo, se recomienda no activar el sistema de compartición de sesiones, ya que este obligaría al servidor a almacenar una copia de los datos en un recurso compartido del que nunca se leerán.

## Clases de gestión incorporadas

FIRe incorpora nativamente mecanismos para la compartición de las sesiones mediante:

* Sistema de fichero:
  + Permite definir un directorio en una unidad de red compartida por todos los nodos.
* Base de datos:
  + Permite almacenar los datos en una base de datos accesible por todos los nodos del sistema balanceado.
  + Este sistema es apropiado tiene tiempos de guardado y carga reducidos, pero un coste alto de memoria para el guardado de los documentos a firmar y las firmas generadas.
  + Las clases incorporadas por FIRe para el guardado de los datos de sesión en disco son:
    - es.gob.fire.server.services.internal.sessions.DBSessionsDAO
    - es.gob.fire.server.services.internal.sessions.DBTempDocumentsDAO

### Sistema de ficheros

Este sistema permite definir que la información se comparta entre todos los nodos a través de un directorio en una unidad de red compartida (la misma en la que se almacenarán los datos temporales de la aplicación).

El compartir los datos a través de una unidad de red requiere mayor tiene de guardado y carga, pero no tiene límite de tamaño, lo que lo hace especialmente apropiado para la compartición de los documentos a firmar y las firmas generadas.

Las clases incorporadas por FIRe para el guardado de los datos de sesión en disco son:

* es.gob.fire.server.services.internal.sessions.FileSystemSessionsDAO
* es.gob.fire.server.services.internal.sessions.FileSystemTempDocument

#### Requisitos

Este sistema requiere disponer de una unidad de red compartida a la que tengan acceso todos los nodos balanceados en los que se haya desplegado el componente central de FIRe.

### Base de datos

Este sistema permite que la información se comparta entre todos los nodos a través de la base de datos de FIRe (la misma utilizada por el módulo de administración).

El compartir los datos a través de base de datos permite acceder rápidamente a la información compartida, pero puede tener un alto coste en memoria para el guardado de los documentos a firmar y las firmas generadas (datos binarios de gran tamaño).

Las clases incorporadas por FIRe para el guardado de los datos de sesión en base de datos son:

* es.gob.fire.server.services.internal.sessions.DBSessionsDAO
* es.gob.fire.server.services.internal.sessions.DBTempDocumentsDAO

#### Requisitos

Este sistema requiere que se haya desplegado la base de datos de FIRe y que se hayan creado tablas de base de datos específicas para este fin. Consulte el manual de instalación y despliegue de FIRe para más referencias.

## Desarrollo de nuevas clases de gestión

Un desarrollador puede implementar su propio mecanismo de guardado de los datos de sesión y utilizarlo para el guardado de los datos de configuración y los resultados de las operaciones, para el guardado de los documentos a firmar y las firmas generadas, o para todo ello.

Para el desarrollo de un mecanismo personalizado, el integrador deberá implementar las siguientes interfaces Java en una clase o dos separadas:

* es.gob.fire.server.services.internal.sessions.SessionsDAO
  + Interfaz de gestión de la configuración y resultados de firma.
* es.gob.fire.server.services.internal.sessions.TempDocumentsDAO
  + Interfaz de gestión de los documentos a firmar y firmas generadas.

Es posible crear un mecanismo de guardado que sólo se utilice para guardar los datos de sesión o los documentos temporales almacenados. Para ello se podría implementar sólo una de las interfaces y luego configurar en el componente central que se utilice la nueva clase compartir lo que corresponda y otra clase, que utilice otro mecanismo, para gestionar el resto.

Las mencionadas interfaces se encuentran en el módulo “fire-signature” del componente central de FIRe. Para poder implementar las clases necesarias, será necesario importar este módulo en el proyecto en el que se desarrolle.

### SessionsDAO

Las clases que implementen esta interfaz deberán implementar los siguientes métodos:

|  |
| --- |
| /\*\*  \* Guarda una sesión.  \* **@param** session Sesión a guardar.  \* **@param** created Indica si la sesión se está creando ({@code true})  \* o si ya lo estaba ({@code false}).  \*/  **void** saveSession(FireSession session, **boolean** created);  /\*\*  \* Comprueba la existencia de una sesión.  \* **@param** id Identificador de la sesión.  \* **@return** {@code true} si la sesión existe, {@code false} en caso contrario.  \* **@throws** SessionException Cuando no se haya podido comprobar la existencia de la  \* sesión.  \*/  **boolean** existsSession(String id) **throws** SessionException;  /\*\*  \* Recupera una sesión. Esto implica construir el propio objeto de  \* sesión y renovar su periodo de validez.  \* **@param** id Identificador de la sesión.  \* **@param** session Sesión web en la que se debe sustentar la sesión.  \* **@return** Sesión recuperada o {@code null} si no existe la sesión o no pudo  \* cargarse.  \*/  FireSession recoverSession(String id, HttpSession session);  /\*\*  \* Elimina una sesión.  \* **@param** id Identificador de la sesión.  \*/  **void** deleteSession(String id);  /\*\*  \* Elimina las sesiones caducadas.  \* **@param** expirationTime Tiempo en milisegundos que puede haber transcurrido desde  \* su último uso antes de considerarse caducada.  \* **@throws** IOException Cuando ocurre algún error al borrar los ficheros.  \*/  **void** deleteExpiredSessions(**long** expirationTime) **throws** IOException;  /\*\*  \* Recupera el gestor de documentos temporales asociado a un gestor de sesiones.  \* **@return** Gestor de documentos temporales o {@code null} si no tuviese uno asociado.  \*/  TempDocumentsDAO getAssociatedDocumentsDAO(); |

Los métodos a implementar son:

* saveSession
  + Permite crear o actualizar una sesión.
* existsSession
  + Permite comprobar si una sesión ya existe en el sistema compartido.
* recoverSession
  + Permite recuperar los datos de una sesión.
* deleteSession
  + Permite borrar una sesión.
* deleteExpiredSessions
  + Permite ejecutar un proceso que repasa todas las sesiones actualmente abiertas, comprueba si están caducadas y las borra en caso afirmativo.
* getAssociatedDocumentsDAO
  + Permite conocer cuál es la clase que implementa TempDocumentsDAO que debería usarse cuando se cargase este gestor de sesiones. Esta es la clase que se usará cuando el administrador no indique expresamente el uso de otra clase. La clase indicada por este método no tiene porqué utilizar el mismo mecanismo de compartición que esta clase.

### TempDocumentsDAO

Las clases que implementen esta interfaz deberán implementar los siguientes métodos:

|  |
| --- |
| /\*\*  \* Comprueba la disponibilidad de un documento.  \* **@param** id Identificador del documento.  \* **@return** {@code true} si existe el documento, {@code false} en caso contrario.  \* **@throws** IOException Cuando ocurre un error que hace fallar la operación.  \*/  **boolean** existDocument(String id) **throws** IOException;  /\*\*  \* Almacena un documento.  \* **@param** id Identificador del documento o {@code null} si el documento no tenía un  \* identificador y se le debe asignar.  \* **@param** data Contenido del documento.  \* **@param** newDocument {@code true} si el documento no existía previamente,  \* {@code false} si sí existía.  \* **@return** Identificador del documento almacenado.  \* **@throws** IOException Cuando ocurre un error que hace fallar la operación.  \*/  String storeDocument(String id, **byte**[] data, **boolean** newDocument) **throws** IOException;  /\*\*  \* Recupera un documento.  \* **@param** id Identificador del documento.  \* **@return** Contenido del documento.  \* **@throws** IOException Cuando ocurre un error que hace fallar la operación.  \*/  **byte**[] retrieveDocument(String id) **throws** IOException;  /\*\*  \* Elimina un documento si existe.  \* **@param** id Identificador del documento.  \* **@throws** IOException Cuando ocurre un error que hace fallar la operación.  \*/  **void** deleteDocument(String id) **throws** IOException;  /\*\*  \* Recupera el contenido de un documento y luego lo elimina.  \* **@param** id Identificador del documento.  \* **@return** Contenido del documento.  \* **@throws** IOException Cuando ocurre un error que hace fallar la operación.  \*/  **byte**[] retrieveAndDeleteDocument(String id) **throws** IOException;  /\*\*  \* Elimina todos los documentos caducados.  \* **@param** expirationTime Último momento del tiempo en el que se debió  \* modificar el documento para no considerarse caducado.  \* **@throws** IOException Cuando ocurre un error que hace fallar la operación.  \*/  **void** deleteExpiredDocuments(**long** expirationTime) **throws** IOException; |

Los métodos a implementar son:

* existDocument
  + Permite comprobar si existe un documento.
* storeDocument
  + Permite almacenar un documento o actualizarlo si ya existiese.
* retrieveDocument
  + Permite recuperar el contenido del documento.
* deleteDocument
  + Permite eliminar un documento.
* retrieveAndDeleteDocument
  + Permite cargar el contenido de un documento y, acto seguido, borrar le documento.
* deleteExpiredDocuments
  + Permite ejecutar un proceso que recorre los documentos compartidos actualmente, comprueba si están caducados y los borra en caso afirmativo.

### Alta de las nuevas clases de gestión de sesiones

Para dar de alta la clase gestora, el administrador deberá:

1. Empaquetar la/s clase/s gestora/s en un archivo JAR.
2. Introducir el JAR empaquetado dentro del directorio de bibliotecas del WAR del componente central (fire-signature.war\WEB-INF\lib).

Una vez importado el JAR con las nuevas clases en el componente central de FIRe, podremos modificar el fichero de configuración del componente central para que utilice las nuevas clases.

## Configuración de las clases de gestión de sesiones

El componente central de FIRe permite definir qué clases de gestión de sesiones debe utilizar por medio del fichero de configuración config.properties. Las propiedades que nos permite configurar estas clases son:

* sessions.dao
  + Configura la clase que debe gestionar la compartición de los datos de sesión de FIRe (configuración de las operaciones de firma y resultados de las operaciones).
  + El valor de esta propiedad debe ser siempre el nombre completo de una clase que implemente la interfaz SessionsDao.
  + Esto solo debe usarse cuando se despliegue el componente central en varios nodos balanceados y no se compartan los objetos en memoria entre ellos.
  + Valor por defecto, ninguno.
  + Valores disponibles de forma nativa:
    - es.gob.fire.server.services.internal.sessions.FileSystemSessionsDAO
      * Compartición de sesiones a través de disco.
      * Al usar este valor, el valor por defecto para la propiedad “sessions.documents.dao” pasará a ser “es.gob.fire.server.services.internal.sessions.FileSystemTempDocumentsDAO”.
    - es.gob.fire.server.services.internal.sessions.DBSessionsDAO
      * Compartición de sesiones a través de base de datos.
      * Requiere que se haya creado en el esquema de bases de datos las tablas concretas para el uso de esta funcionalidad.
      * Al usar este valor, el valor por defecto para la propiedad “sessions.documents.dao” pasará a es “es.gob.fire.server.services.internal.sessions.DBTempDocumentsDAO”.
  + Si se han importado en el componente central de FIRe clases propias para la compartición de sesiones, puede establecerse en esta propiedad el nombre completo de la clase encargada de la gestión de las sesiones (nombre de paquete y de la clase).
* sessions.documents.dao
  + Configura la clase que debe gestionar la compartición de documentos temporales entre nodos balanceados.
  + El valor de esta propiedad debe ser siempre el nombre completo de una clase que implemente la interfaz TempDocumentsDao.
  + Esto solo debe usarse cuando se despliegue el componente central en varios nodos balanceados y no se compartan los objetos en memoria entre ellos.
  + Puede omitirse el valor de esta propiedad cuando se establezca un valor en la propiedad “sessions.dao” y se desee utilizar el mecanismo de compartición de documentos asociado por defecto para este.
  + Valores disponibles:
    - es.gob.fire.server.services.internal.sessions.FileSystemTempDocumentsDAO
      * Guardado de datos en disco a través del directorio temporal configurado en la propiedad "temp.dir"
    - es.gob.fire.server.services.internal.sessions.DBTempDocumentsDAO
      * Guardado de temporales en base de datos. Para la conexión con la base de datos se utiliza la propiedad "bbdd.conn".
      * Requiere que se haya creado en el esquema de bases de datos las tablas concretas para el uso de esta funcionalidad.
  + Si se han importado en el componente central de FIRe clases propias para la compartición de sesiones, puede establecerse en esta propiedad el nombre completo de la clase encargada de la gestión de los datos a firmar a las firmas generadas (nombre de paquete y de la clase).

La configuración deberá establecerse en todos los nodos y, una vez hecho, es necesario reiniciar los servidores de aplicaciones para que los cambios surtan efecto.

#### Configuración de ejemplo

Un ejemplo de fichero de configuración sería:

|  |
| --- |
| # Configura el gestor de sesiones  sessions.dao=es.organismo.fire.sessions.MiGestor  # Configura el gestor para la compartición de documentos  sessions.documents.dao=es.gob.fire.server.services.internal.sessions.FileSystemTempDocumentsDAO |

En el ejemplo anterior, se ha configurado como gestor de sesiones el nombre de una clase propia que deberemos haber incluido en el WAR del componente central tal como se describe en el apartado 3.2.3 Alta de las nuevas clases de gestión de sesiones. Por otro lado, se ha configurado como gestor para la compartición de documentos la clase para la compartición de los documentos a través del sistema de ficheros que ya se incorpora con FIRe.