DGTE-002: Desarrollo de aplicaciones seguras en la plataforma JEE.

Versión 1.1

Área de Arquitectura

GERENCIA INFORMÁTICA
JOSEFA VALCÁRCEL, 44
28027-MADRID

Índice General

Índice de Ilustraciones

Índice de ejemplos de código

1 Introducción

En el contexto del Programa de Transformación Tecnológica, la Dirección General de Tráfico ha decidido tomar medidas para asegurar coherencia y homogeneidad en todo desarrollo en la plataforma Java™ Enterprise Edition (1).

A día de hoy, el desarrollo de las aplicaciones Java™ de la DGT está enmarcado dentro de los requerimientos de los departamentos de Arquitectura y Calidad indicados en el documento Guía de Desarrollo; este documento forma parte de la metodología asociada al modelo de proceso de desarrollo de la DGT.

Este documento todavía no recoge las pautas a seguir durante el desarrollo de aplicaciones seguras y las aplicaciones desarrolladas hasta la fecha han sido diseñadas con soluciones que cumplían los requerimientos de seguridad de cada una de ellas. Las soluciones realizadas, aunque válidas, no están enmarcadas dentro de una especificación de seguridad de la Dirección General de Tráfico. Actividades como la auditoría de Calidad, el mantenimiento, la auditoría de seguridad, se hacen más complejas por la heterogeneidad de las soluciones implementadas.

El impacto de la migración de parte del sistema de la DGT a otras tecnologías[1] en la situación actual, además, debería estudiarse caso por caso, para determinar cuáles son los servicios utilizados por una aplicación, a través de qué mecanismos, y por qué.

La seguridad de una aplicación es un tema amplio que abraza el estudio de vulnerabilidades de naturalezas muy distintas. Estas vulnerabilidades, además, no son debidas exclusivamente a la implementación de la lógica de una aplicación, sino también a las características de la infraestructura en la cual la aplicación se está ejecutando.

Esta especificación, dirigida a los desarrolla-dores de aplicaciones Java™, se limita a proponer soluciones adoptables a través de la codificación y la configuración de las aplicaciones.

[1] Migración de base de datos, migración de directorio LDAP, etc.

1.1 Un ejemplo típico

El caso de uso que esta especificación pretende cubrir es la gestión de la autenticación y la autorización de usuarios en una aplicación Java™ EE. Durante el desarrollo de una aplicación Java™ que requiera manejar perfiles de usuarios con privilegios distintos, las herramientas puestas a disposición por Java™ SE, se reducen esencialmente al API Java™ Authentication and Authorization Service[2]. La plataforma Java™ EE aumenta la flexibilidad de JAAS a través de la funcionalidad proporcionada por el servidor de aplicaciones, que permite utilizar el API JAAS de manera programática o de manera declarativa, es decir, a través de la configuración de los privilegios (roles en la terminología Java™ EE) necesarios para acceder a los recursos[3] disponibles en los descriptores de despliegue de la aplicación[4], sin necesidad de modificar el código existente.

El API JAAS es una API estándar que ya es parte del Java™ Development Kit[5] y por lo tanto permite escribir código sin atarse a API o funcionalidades propietarias. Sin embargo, un mecanismo de seguridad basado en roles requiere que exista un registro de usuarios a autenticar y una función que establezca el mapeo entre usuarios y roles. La no disponibilidad de uno o ambos de estos requisitos, incluso porque los usuarios a autenticar no existen en el registro de usuarios locales[6], da pie a que las aplicaciones decidan gestionar usuarios, privilegios o ambas cosas en repositorios privados, delegando las comprobaciones de seguridad y el otorgamiento de privilegios a su propia lógica. Una aplicación podría por ejemplo decidir mantener catálogos de usuarios y privilegios a ellos asociados en una base de datos de la misma aplicación. La lógica utilizada durante el proceso de autenticación y de autorización de los usuarios, además, podría implementarse en la propia aplicación y utilizarse de forma explícita.

[2] JAAS de aquí en adelante

[3] Páginas web, servicios web, EJBs, etc.

[4] La especificación Java™ EE de hecho proporciona la misma funcionalidad a través de metadatos (anotaciones) también. Los metadatos que proceden de las anotaciones se podrán sobrescribir con la información proporcionada a través del descriptor de despliegue.

[5] Desde la versión 1.4.

[6] Como es el caso de un ciudadano que se conecte a una aplicación de la Dirección General de Tráfico.

Evidentemente, una solución de ese tipo establece una dependencia entre la aplicación y los servicios[7] utilizados para almacenar y explotar esta información. La migración de un backend utilizado por esta solución, por ejemplo, comportaría como mínimo la reescritura parcial del mecanismo de autenticación y autorización. Este tipo de solución es lo que la API JAAS pretendía solucionar proporcionando un API de servicio que permitiese ser extendida para acoger soluciones no previstas por la implementación utilizada por el servidor de aplicaciones.

La adopción de una solución de este tipo su-pone para la Dirección General de Tráfico una serie de problemas:

- El código de las aplicaciones se responsabiliza de actividades críticas.
- La calidad del código, de extrema importancia para que la seguridad de las aplicaciones no se vean comprometidas, es difícil de controlar.
- Las aplicaciones no pueden compartir las informaciones de los usuarios la una con la otra impidiendo, de facto, el establecimiento de una política de single sign-on a nivel corporativo.
- Duplicación de lógica parecida.
- Aumento de la complejidad de las actividades de mantenimiento.
- Aumento de la complejidad de las actividades de migración.
- Duplicación de información en múltiples lugares del sistema.

1.2 Objetivo

El objetivo de este documento pretende llenar este vacío y solucionar los problemas descritos en esta introducción.

La Dirección General de Tráfico pretende ampliar el marco proporcionado en la Guía de desarrollo especializando la información allí recogida y proporcionar lo necesario para que las aplicaciones cumplan con los requisitos que allí se proponen.

[7] En este contexto la palabra servicio se utiliza con un sentido amplio: bases de datos, componentes, lógica de negocio, etc.

Los requerimientos de seguridad establecidos por la Dirección General de Tráfico responden a exigencias de distinta naturaleza: por un lado se necesitan recoger las normas legales existentes y por otro lado se quiere optimizar el proceso de desarrollo de las aplicaciones. Las herramientas que se pretende proporcionar son herramientas conceptuales[8] que, enmarcadas dentro de la Guía de desarrollo[9], fijen la solución a los problemas existentes. De esta manera toda aplicación utilizará el mismo conjunto de soluciones para el mismo conjunto de problemas consiguiendo:

- Evitar el gasto para estudiar, diseñar e implementar una solución a los problemas de autenticación y autorización de las aplicaciones Java™ EE.
- Evitar la proliferación de soluciones personalizadas para cada aplicación.
- Homogeneizar el conjunto de herramientas utilizadas por las aplicaciones[10].
- Utilizar oportunas capas de abstracción para desacoplar las aplicaciones de las implementaciones de estas soluciones.
- Facilitar las migraciones futuras.

Esta especificación pondrá también la base para ampliar los requisitos de seguridad y normalizarlos como estándares de la DGT a partir de su incorporación dentro de la Guía de desarrollo.

1.3 Audiencia

Este documento está dirigido a los Jefes de Proyecto, analistas y, en general, desarrolladores de aplicaciones en la plataforma Java™ EE para aplicaciones de la Dirección General de Tráfico.

El documento presupone que el lector tiene un conocimiento básico de las APIs que componen la especificación Java™ EE, con particular énfasis a la interfaz de programación JAAS. No obstante, donde se revise la necesidad, se hará explícita referencia a las secciones pertinentes de la documentación oficial de dichas APIs.

[8] Tales como estándares, APIs, patrones, modelos de programación, etc.

[9] Como documentos adjuntos.

[10] APIs, backends, etc.

1.4 Estructura del documento

Este documento está distribuido en z capítulos, con los siguientes contenidos:

- Capítulo 1: Introducción, contiene información relativa al propio documento
- Capítulo 2: Visión general, en el que se describe el ámbito de la autenticación y autorización en entornos JEE
- Capítulo 3: Protección de recursos en aplicaciones web
- Capítulo 4: Protección de un servicio web
- Capítulo 5: Integración con IBM Websphere Datapower
- Apéndice A: Proceso de autenticación de la Dirección General de Tráfico

2 Visión general

2.1 Ventajas

Como ha quedado explicado en el capítulo 1, el Departamento de Arquitectura de la Dirección General de Tráfico ha decidido publicar esta especificación con el objetivo de dar una respuesta homogénea a una serie de problemas que se pueden encontrar durante el desarrollo de una aplicación en la plataforma Java™ EE. Los objetivos que Arquitectura persigue definiendo esta especificación son los siguientes:

- La especificación para el desarrollo de aplicaciones Java™ EE que utilicen el modelo de seguridad declarativo será la referencia a seguir durante el desarrollo de aplicaciones Java™ que requieran funcionalidad de autenticación y autorización de usuarios.
- Esta especificación facilitará el análisis y el diseño de la lógica de las aplicaciones especificando las API y los patrones a utilizarse durante el empleo de las APIs de seguridad.
- Esta especificación y la utilización de los estándares que ella propone permitirá aislar las aplicaciones de las implementaciones subyacentes escogidas por el departamento de

Arquitectura. Un cambio en el registro de usuarios, por ejemplo, será aislado por la API estándar, JAAS, utilizada por las aplicaciones que no deberán modificarse para gestionar el nuevo backend.

2.2 Servicios de seguridad en la plataforma Java™ EE

La plataforma Java™ EE proporciona servicios a las aplicaciones ejecutadas en los servidores de aplicaciones que cumplen la especificación. Una categoría de servicios proporcionados por los contenedores de módulos Java™ EE son los ser-vicios de seguridad. Según la tipología de aplicación que se esté desarrollando esta tendrá requerimientos de seguridad distintos. En la plataforma Java™ EE se pueden distinguir tres niveles fundamentales en las tecnologías de seguridad que se pueden utilizar:

- Seguridad de la capa de la aplicación.
- Seguridad de la capa de transporte.
- Seguridad de la capa de mensaje.

En lo que concierne los requerimientos de la seguridad de la capa de transporte, aunque las plataformas Java™ SE y Java™ EE proporcionan mecanismos para cubrir este requisito[11], se utilizará la appliance WebSphere® DataPower. Las repercusiones de esta elección en el desarrollo de aplicaciones son nulas, en cuanto no se ven afectadas en ninguna de sus partes.

En lo que concierne la seguridad de la capa de la aplicación, se utilizará una solución hibrida: en esta capa también se utilizará la appliance WebSphere® DataPower y durante el desarrollo de las aplicaciones se requerirá exclusivamente la protección de las aplicaciones a través del mecanismo de la seguridad declarativa, tal y como se detalla más adelante en este documento.

En lo que concierne a la seguridad de la capa de mensajes esta especificación propondrá la utilización del estándar WS Security (2) y los demás estándares a éste asociados.

[11] La plataforma Java™ SE proporciona las tecnologías Java™ Secure Socket Layer Extension (JSSE) y Simple Authentication and Security Layer (SASL).

2.3 Autenticación y autorización

Durante el proceso de desarrollo de aplicaciones Java™ para la Dirección General de Tráfico, en lo que concierne los servicios de seguridad en la capa de aplicación, los equipos de desarrollo deberán preocuparse de proteger los recursos de las aplicaciones. Las dos operaciones que comúnmente deberán gestionarse son las operaciones de autenticación y de autorización.

La autenticación es el proceso a través del cual se identifica a un usuario comprobando las credenciales proporcionadas durante el proceso de login. La responsabilidad del proceso de autenticación será delegada al appliance WebSphere® DataPower y a un conjunto de servicios adicionales[12]. Los equipos de desarrollo, por lo tanto, no tendrán que gestionar este proceso de ninguna forma. Esta especificación, por lo tanto, hace explicita prohibición de que se gestione el proceso de autenticación de forma programática a través de soluciones personalizadas en el interior de la propia aplicación.

La autorización es el proceso a través del cual se comprueba que el usuario previamente identificado posee los privilegios necesarios para acceder al recurso que está solicitando. La responsabilidad de proteger adecuadamente los recursos de la aplicación recae en el equipo de desarrollo. Esta especificación propone la utilización de los servicios de seguridad proporcionados por la plataforma Java™ EE: en este caso habrá entonces que utilizar la seguridad basada en roles. La configuración de la aplicación deberá hacerse utilizando el modelo declarativo.

La utilización del modelo programático, es decir, de la utilización del API JAAS para ejecutar comprobaciones a tiempo de ejecución en la lógica de negocio de las aplicaciones, queda prohibida por esta especificación.

2.4 Conceptos básicos

Independientemente del servicio de seguridad o de la capa en la que se está aplicando la política de seguridad, ya que el objetivo explícito de esta especificación es aislar a los desarrolladores de las particularidades de los entornos de ejecución, en lo que concierne el desarrollo de aplicaciones

[12] Tales como la plataforma @firma en lo que concierne la autenticación de usuarios a través de un certificado digital.

Java™ para la Dirección General de Tráfico se re-quiere exclusivamente el conocimiento y la utilización de los conceptos básicos necesarios y suficientes para analizar los requisitos de seguridad de una aplicación y diseñar la solución adecuada.

Los conceptos definidos en la especificación Java™ EE V.5 necesarios para cumplir estos objetivos son los siguientes:

- Realm.
- Usuario.
- Grupo de usuarios.
- Rol.

El realm es el dominio en que una determinada política de seguridad es válida. Esta propiedad es gestionada por el administrador a nivel de servidor de aplicación. Esta especificación hace mención de ella exclusivamente porque el concepto de realm suele utilizarse a menudo en la literatura sobre los servicios de seguridad en la plataforma Java™ EE. Sin embargo este concepto no se utilizará durante el proceso de desarrollo de aplicaciones Java™ para la Dirección General de Tráfico.

Los conceptos de usuario y grupo de usuarios se corresponden a los familiares conceptos homónimos manejados por el sistema de autenticación y autorización de un sistema operativo UNIX®[13]. Los usuarios son las identidades que se pueden autenticar y los grupos son conjuntos de usuarios. El concepto de grupo es útil a la hora de organizar jerárquicamente los usuarios agrupándolos en conjuntos según las características que sean útiles a este fin.

[13] Los *usuarios* y los *grupos* utilizados por un servidor de aplicaciones Java™ EE en principio no son las mismas entidades manejadas por el sistema de autenticación del sistema operativo subyacente. Sin embargo en los casos en los cuales este comportamiento fuese el esperado, se puede configurar el servidor de aplicación para que utilice el mismo registro de usuarios utilizado por el sistema operativo durante el proceso de autenticación. En lo que concierne el proceso de autenticación en las aplicaciones Java™ desarrolladas por la DGT, el concepto de usuario a manejarse es el especificado por la especificación Java™ EE. En lo que conciernen la actividad de autorización no es siquiera necesario manejar el concepto de *usuario*, sino de *rol*.

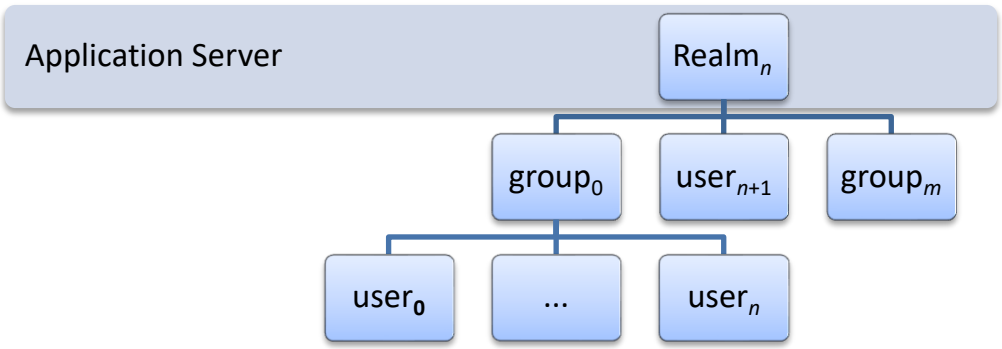

Ilustración 1. Jerarquía de usuarios y grupos en un realm

Los conceptos de realm, usuario y grupo de usuarios son conceptos básicos necesarios para aclarar cuáles son las entidades que participan en el proceso de autenticación. Por sus mismas naturalezas, durante el proceso de desarrollo no se configurarán o manejarán[14] entidades de estas categorías que son propiedades el entorno de ejecución que no están relacionadas directamente con una aplicación, como se ilustra en la figura siguiente.

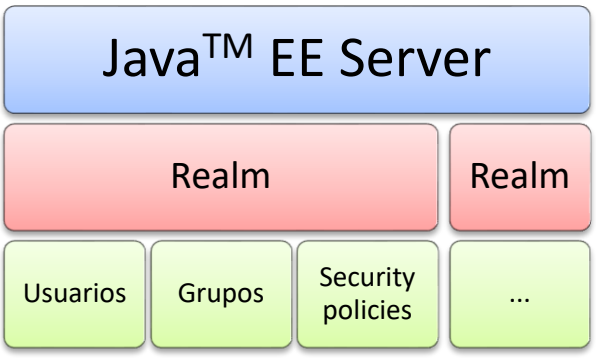

Ilustración 2. Relación entre realm, usuarios y grupos de usuarios

2.4.1 Rol

El modelo de seguridad utilizado durante la fase de autenticación por la plataforma Java™ EE es esencialmente un modelo de control de acceso basado en roles. Los roles representan un privilegio

[14] En ciertas ocasiones, un programa Java™ podrá manejar en sola lectura el Subject, que es la *representación* de un usuario autenticado proporcionada por la API JAAS.

que en la plataforma Java™ EE se traduce en el derecho de acceder a un recurso. El rol de un usuario, como sugiere su nombre, puede utilizarse para realizar la organización de los recursos protegidos de una aplicación en categorías: los ro-les. En tiempo de ejecución, los usuarios y los grupos de usuarios obtendrán el acceso al recurso protegido solicitado exclusivamente si el sujeto autenticado pertenece a unos de los roles a los cuales ha sido otorgado el acceso a dicho recurso.

A diferencia de lo que ocurre con los realms, los usuarios y los grupos, el concepto de rol está definido a nivel de aplicación y no de servidor de aplicaciones, como ilustra la figura siguiente. Como se detallará en las siguientes secciones de este documento esta característica es lo que per-mite analizar, diseñar y configurar la protección de los recursos de una aplicación aislándose de las demás aplicaciones e incluso ignorando la jerarquía de usuarios que habrá en tiempo de ejecución, ya que la función que asocia usuarios con roles también es parte de la configuración de la aplicación.

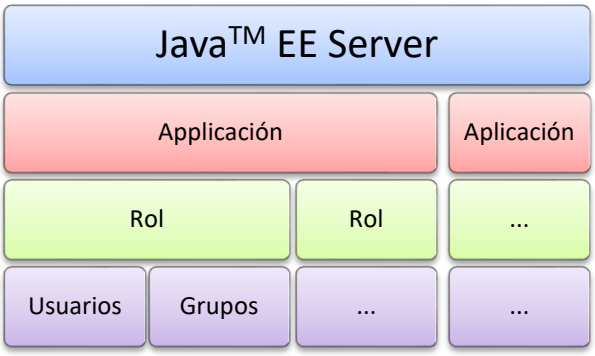

Ilustración 3. Relación entre aplicaciones y roles

El concepto de rol introduce por lo tanto un nivel de abstracción que permite independizar los privilegios a otorgarse de los objetos que representan los sujetos a autenticarse[15]. Los roles y la asociación establecida entre rol, usuarios y grupos de usuarios, son una propiedad de la configuración de una aplicación y esta característica será objeto de análisis durante el desarrollo de las aplicaciones. No es el usuario la entidad a la que se otorga un privilegio, sino al rol[16] al que un usuario está asociado en la configuración de la aplicación en cuestión.

[15] Los usuarios y los grupos de usuarios.

[16] O a uno de los roles.

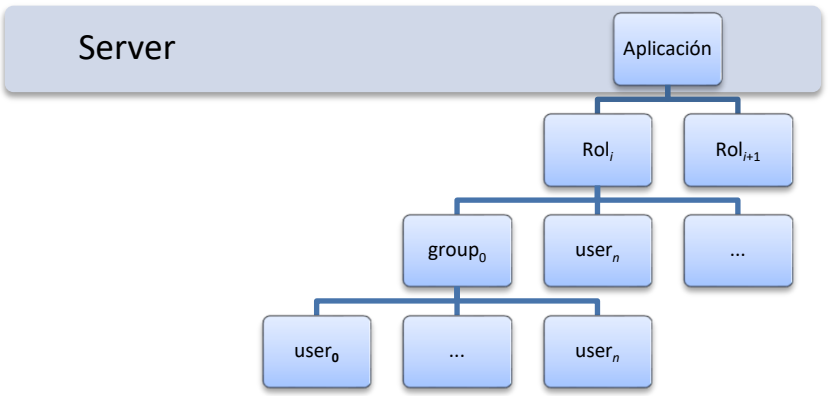

Ilustración 4. Relación entre roles, usuarios y grupos de usuarios en aplicaciones Java™ EE

2.4.2 Ventajas del control de acceso basado en roles

El desacoplamiento entre *aplicación*, *sujetos autenticados* y *privilegios* introducido por la abstracción del rol proporciona ventajas en términos de *flexibilidad* durante la configuración de la aplicación y de *aislamiento* durante el proceso de análisis de los requisitos de seguridad de una aplicación. La posibilidad de organizar los recursos de una aplicación en roles significa que durante el análisis y el diseño de una aplicación será posible organizar los usuarios autenticados en categorías, los roles, sin preocuparse de cómo en la realidad los usuarios estén organizados en el registro de usuarios. Los roles identificados, además, serán una característica de la aplicación analizada con completa independencia de las demás aplicaciones. Será responsabilidad de la lógica de asociación de roles establecer esta relación para que en tiempo de ejecución el contenedor pueda efectuar las comprobaciones necesarias durante el proceso de autorización.

Resumiendo las ventajas aportadas por la utilización del control de acceso basado en roles son, entre otras:

- Cada aplicación define el conjunto de roles necesarios para cubrir los requerimientos de protección de los recursos con una resolución tan fina como sea necesario.
- A cada rol pueden pertenecer usuarios y grupos de usuarios. Un usuario puede pertenecer a más de un rol.

- La definición del rol se efectúa a nivel de aplicación y la pareja aplicación y rol es indisoluble. El mapeo entre usuario, grupos de usuarios y roles se puede potencialmente efectuar con grano fino y a nivel de aplicación[17].

2.5 Análisis de los requerimientos de seguridad de los recursos de una aplicación

Como ha quedado explicado en las secciones precedentes de este documento, los roles representan un conjunto de privilegios en una aplicación Java™ EE. Durante el análisis de los requerimientos de seguridad de una aplicación Java™ para la Dirección General de Tráfico será entonces necesario analizar las características de una aplicación para cumplir los siguientes objetivos:

- Recopilación de los recursos que se requiere que sean protegidos.
- Identificación del conjunto de roles necesarios para cubrir los requisitos de seguridad.

La recopilación de los recursos a proteger tiene como objetivo la producción del catálogo de los recursos a protegerse a través del mecanismo de autenticación. Una vez realizado este catálogo, es necesario establecer el conjunto de roles que la aplicación necesita para proteger sus recursos manteniendo la resolución de los privilegios tan finas como se requiera.

Durante la fase de análisis es conveniente además tener en cuenta la posible evolución de la aplicación y efectuar una subdivisión de los roles funcionalmente adecuada. Reducir al mínimo el número de roles puede ser contraproducente si estos roles no corresponden a los roles funcionales identificados en fase de análisis.

2.5.1 Ejemplo

Durante la fase de análisis de una aplicación se ha detecta un conjunto de páginas a proteger. Dichos recursos pertenecen a tres categorías funcionales: páginas de *administración*, páginas para *personal* de la empresa y páginas para *usuarios genéricos*.

[17] En la solución propuesta por la Dirección General de Tráfico no se hace uso de esta posibilidad.

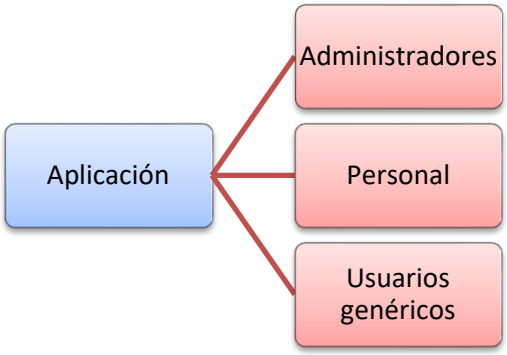

Ilustración 5 – Jerarquía de roles funcionales de una aplicación

Los roles identificados funcionalmente son candidatos ideales para ser roles de seguridad incluso en la hipótesis en que, temporalmente, los conjuntos de recursos protegidos a través de estos roles coincidan, incluso parcialmente.

El rol *personal*, por ejemplo, representa el privilegio de acceder a un conjunto de recursos, R_{dep}, como queda ilustrado en la figura siguiente:

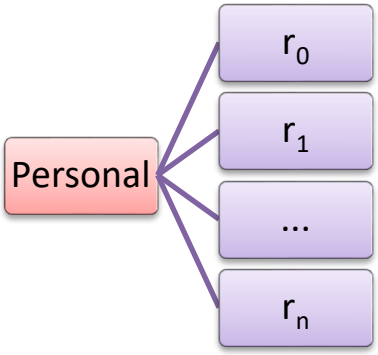

Ilustración 6 – Recursos protegidos por el rol *Personal*

El rol *usuarios genéricos* representa el privilegio de acceder a otro conjunto de recursos, R_{auth}, que en el estado actual podría coincidir con el conjunto R_{dep}:

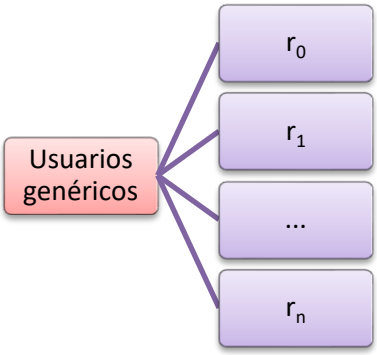

Ilustración 7 – Recursos protegidos por el rol *Usuarios genéricos*

En esta situación se podría definir un único rol de seguridad para cubrir el requisito de protección del conjunto de recursos, ya que para estos dos roles funcionales los conjuntos de recursos coinciden. Sin embargo esta elección podrá revelarse débil cuando, en el caso en que se integren nuevos recursos en la aplicación, los dos conjuntos comiencen a divergir.

2.5.2 Objetivos de la fase de análisis

Resumiendo los conceptos que han sido expuestos en estas secciones del documento, el objetivo a cumplir durante la fase de análisis funcional de la aplicación es determinar qué recursos requieren protección y qué roles es necesario definir para implementar la protección respetando el significado funcional de los roles identificados.

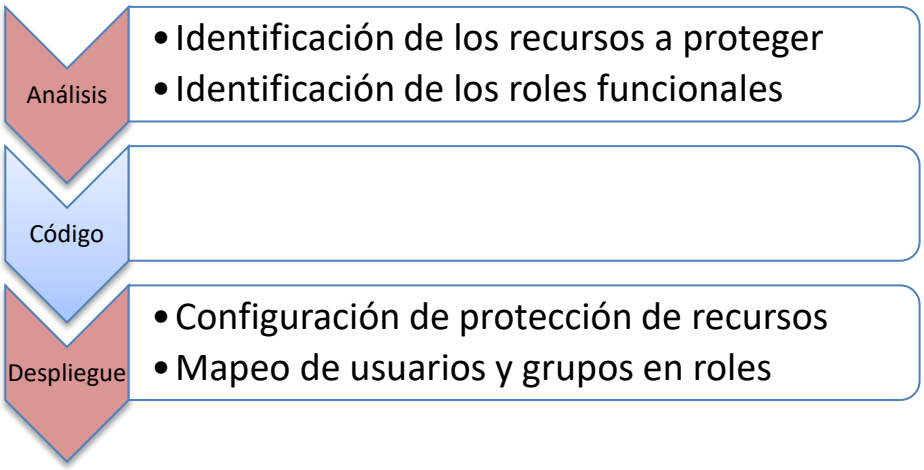

Ilustración 8 – Objetivos por fases

Como se aprecia en la ilustración precedente, el modelo de seguridad declarativa proporciona un modelo de programación simplificado donde las responsabilidades del establecimiento de las políticas de seguridad para la protección de recursos en la capa de aplicación recaen en las fases de análisis y de despliegue. Durante la fase de codificación no se requiere ninguna medida para cumplir con este objetivo con la ventaja de que el código no será contaminado por invocaciones programáticas de las APIs de seguridad con un consiguiente aumento de flexibilidad durante las fases de evolución y de mantenimiento de las aplicaciones.

3 Protección de recursos en aplicaciones web

En la plataforma Java™ EE el contenedor web es responsable del alojamiento de componentes de los siguientes tipos:

- Java™ Servlet (3).
- JavaServer™ Pages (4).
- JavaServer™ Faces.
- WebServices (5) (2).

Los recursos alojados en el contenedor web pueden utilizar los servicios de seguridad a través de distintos mecanismos:

- Modelo declarativo de la seguridad.
- Metadatos en el código de la aplicación (6).
- Modelo programático de la seguridad (7).

El modelo *declarativo* y los *metadatos*[18] en el código de la aplicación son dos mecanismos a través de los cuales se pueden aplicar políticas de seguridad a un componente desplegado en el contenedor web. Sendos métodos permiten aplicar el mismo conjunto de parámetros teniendo la

[18] La plataforma Java™ EE permite utilizar *anotaciones* para proporcionar información que en las precedentes versiones de la plataforma se podían exclusivamente proporcionar a través de los descriptores de despliegue. Sin embargo esta especificación no permite utilizar las anotaciones privilegiando la utilización de los descriptores de despliegue en su lugar.

información proporcionada a través del descriptor de despliegue prioridad sobre la información proporcionada a través de anotaciones. Los dos vehículos son complementarios, pudiendo un desarrollador aplicar valores por defecto a través de anotaciones que, en fase de despliegue, se refinan a través de las configuraciones aplicadas en el descriptor de despliegue. Sin embargo esta especificación aconseja, siempre que sea posible, evitar utilizar anotaciones en el código y favorecer las configuraciones en el descriptor de despliegue. Para obtener información detallada sobre el descriptor de despliegue de un módulo web Java™ EE referirse a la Java™ Servlet Specification (3), capítulo SRV.13 "*Deployment descriptor*".

El modelo programático permite efectuar las decisiones correspondientes en la lógica de las aplicaciones a través de un API estándar. El modelo programático puede ser útil para colmar las lagunas del modelo declarativo cuando éste no tenga la expresividad suficiente para declarar los requerimientos de seguridad que la aplicación necesita. Esta situación suele darse cuando la seguridad basada en roles no proporciona una solución que cumpla los requisitos de seguridad de la aplicación. El departamento de Arquitectura de la Dirección General de Tráfico impone a través de esta especificación la utilización del control de acceso basado en roles y por lo tanto no se tomará en consideración el modelo de seguridad programático.

3.1 Aplicación de las políticas de seguridad en el contenedor web

Las políticas de seguridad para autorizar el acceso a recursos protegidos se efectúa a nivel de contenedor web *antes* que la ejecución se ceda al componente invocado. La siguiente ilustración ejemplifica el proceso de autenticación en el contenedor web.

Cuando el contenedor web recibe una solicitud para un recurso protegido, comprueba que el usuario autenticado que está pidiendo el recurso posea un rol que le otorgue acceso a dicho recurso. En el caso en que el usuario pueda ser autorizado, el contenedor procede a pasar la solicitud HTTP al recurso solicitado, en caso contrario devolverá un error al cliente.

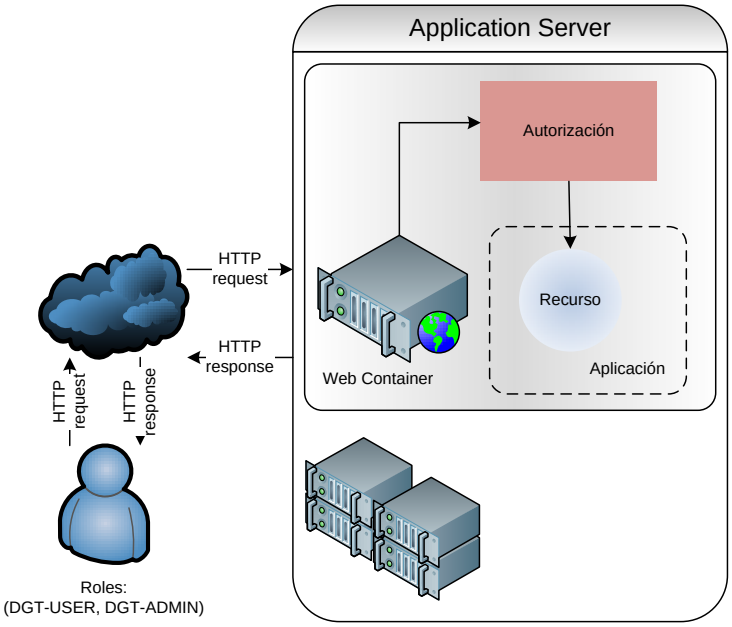

Ilustración 9 – Autorización de acceso a un recurso en un application server Java™ EE

3.2 Declaración de los roles

Como ha quedado detallado en el capítulo precedente, durante la fase de análisis habrá que definir los roles de seguridad que se necesitan para cumplir los requisitos de protección de los recursos sensibles de la aplicación web.

La plataforma Java™ EE proporciona un mecanismo para desacoplar el código y los descriptores de las aplicaciones de los parámetros de configuración establecidos por el administrador durante el despliegue y el mantenimiento del servidor de aplicaciones. Uno de los parámetros que se pueden desacoplar es el nombre de los roles de seguridad.

Esta abstracción permite a los equipos de desarrollo elegir libremente los roles utilizados en las aplicaciones dejando la posibilidad, en fase de despliegue de la aplicación, de resolver dichos nombres en los nombres que se van a configurar en el entorno de ejecución. A falta de esta configuración el servidor de aplicaciones realizará un mapeo idéntico entre los nombres de roles utilizados en la aplicación y los nombres utilizados a tiempo de ejecución, separando en espacios de nombres distintos los nombres manejados por la aplicación y los nombres manejados por el ensamblador y el administrador.

Retomando el ejemplo de la Ilustración 4, la aplicación descrita definirá tres nombres para identificar los roles funcionales que está manejando.

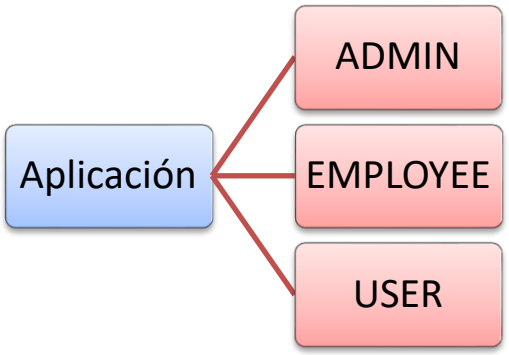

Ilustración 10 – Definición de roles para una aplicación Java™ EE

En tiempo de ensamblado y despliegue, estos nombres podrán *mapearse* a otros nombres a través de *referencias a nombres de rol*.

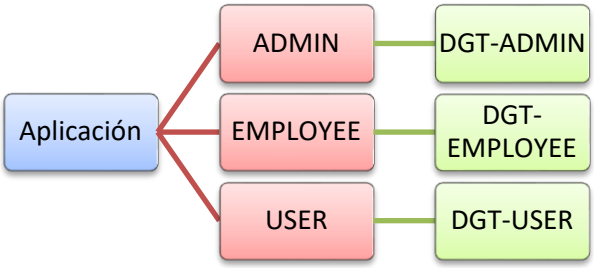

Ilustración 11 – Referencias a nombres de rol para una aplicación Java™ EE

3.2.1 Declaración de roles en el descriptor de despliegue

Para establecer una referencias a nombre de rol se utilizan los elementos **`<security-role-ref/>`** y **`<security-role/>`** del descriptor de despliegue de las aplicaciones web. La sintaxis de esto elementos es la siguiente:

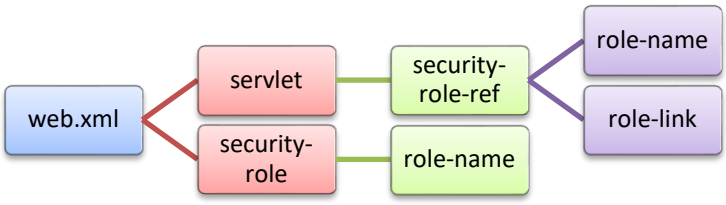

Ilustración 12 – Declaración de roles y referencias a roles en el descriptor de despliegue de una aplicación web Java™ EE

3.2.1.1 <security-role-ref/>

El elemento **`<security-role-ref/>`** se utiliza para declarar los roles de seguridad utilizados en la aplicación web.

El elemento **`<role-name/>`** se utiliza para especificar el nombre del rol en el espacio de nombres de la aplicación.

El elemento **`<role-link/>`** se utiliza para establecer el mapeo entre el nombre del rol utilizado por la aplicación y el rol declarado a través de uno de los elementos **`<security-role/>`**.

3.2.1.2 <security-role/>

El elemento **`<security-role/>`** se utiliza para declarar los roles utilizados por la aplicación que se está configurando.

El elemento **`<role-name/>`** define el nombre del role de seguridad que se está declarando.

3.2.1.3 Ejemplo de configuración

En el fragmento de código siguiente se ejemplifica la configuración del descriptor de despliegue, **`web.xml`**, para declarar una referencia a nombres de roles para un componente Java™ Servlet de ejemplo:

```
<servlet>
  <servlet-name>Nombre</servlet-name>
  <servlet-class>
    es.trafico.MyServlet
  </servlet-class>
  <security-role-ref>
```

```
    <role-name>USER</role-name>
    <role-link>DGT-USER</role-link>
  </security-role-ref>
</servlet>
<security-role>
  <role-name>DGT-ADMIN</role-name>
</security-role>
<security-role>
  <role-name>DGT-EMPLOYEE</role-name>
</security-role>
<security-role>
  <role-name>DGT-USER</role-name>
</security-role>
```

Código 1 – Declaración de roles y mapeos en el web.xml

Este fragmento de configuración declara que la aplicación web en cuestión hará referencia a tres nombres de roles: **DGT-ADMIN**, **DGT-EMPLOYEE** y **DGT-USER**. Un componente Java™ Servlet, cuya configuración es incompleta, declara una referencia entre el nombre de rol **USER** y **DGT-USER**. En lo que concierne el espacio de nombres en que vive dicho componente el rol en cuestión se nombrará **USER**. El contenedor web, en tiempo de ejecución, se encargará de resolver la referencia entre el nombre utilizado por el componente y el nombre declarado por la aplicación ensamblada, **DGT-USER**.

Representando gráficamente esta configuración, obtenemos la representación en la ilustración siguiente:

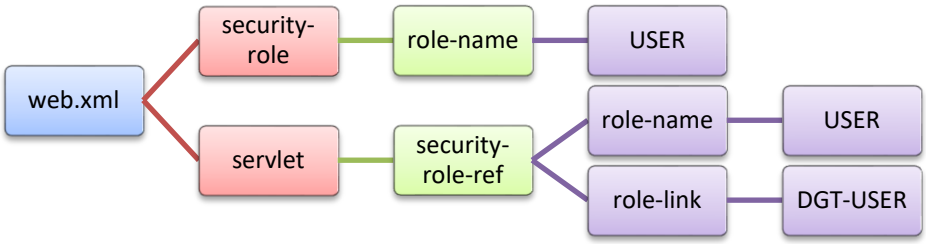

Ilustración 13 – Declaración de roles y sus mapeos

3.3 Protección de un recurso web

Una vez que se han identificado los roles en el interior de la aplicación, se puede proceder a la protección de un recurso web.

La protección de un recurso de ese tipo se puede efectuar con uno cualquiera de los métodos especificados en la introducción de este capítulo. Sin embargo, el Departamento de Arquitectura de la Dirección General de Tráfico recomienda la utilización de los descriptores de despliegue para esta categoría de parámetros de configuración. Por lo tanto, el equipo de desarrollo podrá utilizar el elemento **<security-constraint/>** del descriptor de despliegue, el file **web.xml**, para proteger los recursos sensibles de la propia aplicación. Para la sintaxis completa de este elemento, el desarrollador puede referirse a la Java™ Servlet Specification (3). En la Ilustración 14 se representa la sintaxis del elemento **<security-constraint/>** en su forma de utilización más habitual.

Para aplicar una política de seguridad que impida el acceso a un recurso web

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      Páginas de usuarios
    </web-resource-name>
    <url-pattern>/usuarios/*</url-pattern>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>DGT-USER</role-name>
  </auth-constraint>
</security-constraint>
```

Código 2 – Protección de recursos en el descriptor de despliegue de una aplicación web Java™ EE

El parámetro **<url-pattern/>** se utiliza para especificar un patrón contra el cual el contenedor compara el URL del recurso solicitado: si el URL del recurso satisface el patrón, el contenedor aplica la política de seguridad especificada. Los patrones más comúnmente utilizados son los siguientes[19]:

- **/.*/***: mapeo de rutas.
- ***\.**: mapeo de extensiones.

[19] En los ejemplos se utiliza el lenguaje formal de las expresiones regulares.

- `/`: mapeo de la servlet por defecto.
- `.*`: comparación exacta.

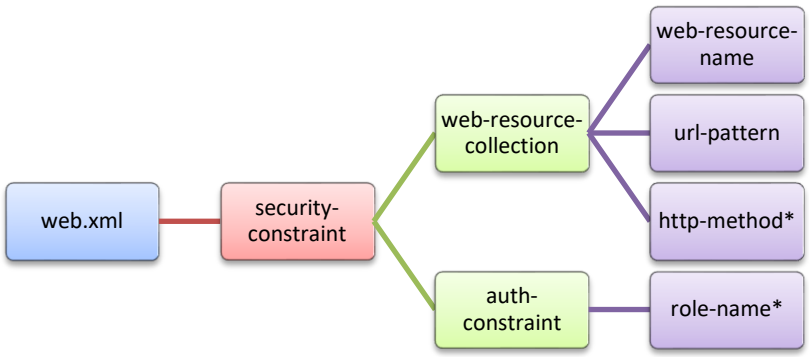

Ilustración 14 – Declaración de políticas de seguridad en el descriptor de despliegue de una aplicación web Java™ EE

Ejemplos de los precedentes patrones son los siguientes:

- Mapeo de rutas: **`/ruta/a/carpeta/*`**
- Mapeo de extensiones: **`*.jsp`**
- Mapeo del Servlet por defecto: **`/`**
- Mapeo exacto: **`cualquier/otra/cosa`**

Someramente, el algoritmo utilizado por el contenedor para resolver estos patrones es el siguiente:

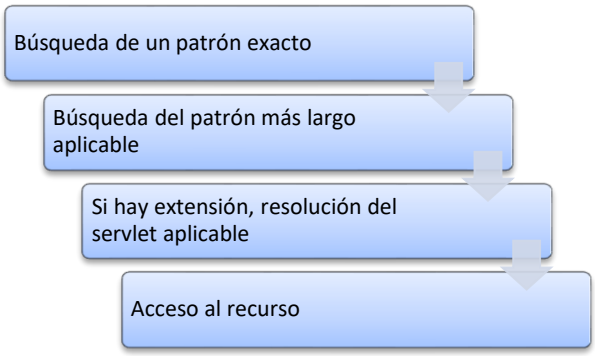

Ilustración 15 – Algoritmo de resolución de los patrones de mapeo de URL utilizado por un contenedor web de un servidor de aplicaciones Java™ EE

Para mayores detalles sobre las reglas utilizadas para hacer la comparación entre URL y el patrón especificado, consultar la Java™ Servlet Specification, capítulo SRV.11 "*Mapping Requests to Servlets*".

4 Protección de un Servicio Web

4.1 Autenticación para Web Services SOAP

La especificación Web Services for the Java EE Platform (JSR-109) (8) sólo requiere que sean soportados los siguientes mecanismos de autenticación:

- HTTP Basic Authentication.
- Symmetric HTTPS.

Esta especificación, sin embargo, opta para la utilización de la especificación WS-Security (2) para realizar las tareas de autenticación de servicios web SOAP. WS-Security es esencialmente un estándar para servicios de seguridad en la capa de mensaje. Como se puede notar en la siguiente ilustración, en el caso de los Servicios Web la tarea de autenticación es responsabilidad de la capa de mensaje y la capa de aplicación puede, si es necesario, quedarse con la gestión de las autorizaciones basadas en roles Java™ EE.

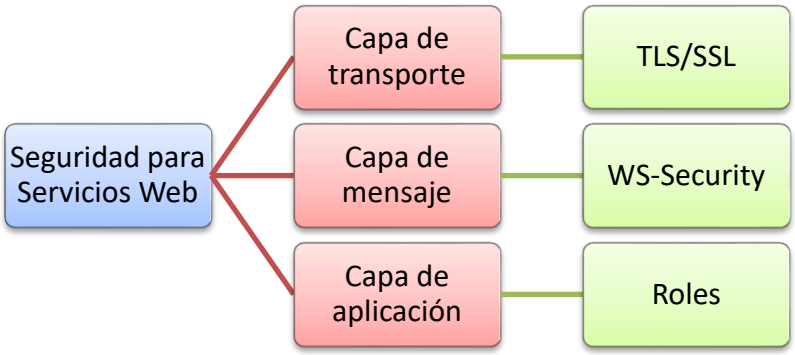

Ilustración 16 - Servicios de seguridad para Servicios Web

4.1.1 Disponibilidad de WS-Security en el entorno de ejecución

WS-Security, aunque no sea una tecnología requerida a un servidor conforme al estándar Java™ EE v. 5, será una funcionalidad adicional soportada por el servidor de aplicaciones Java™ EE utilizado en el entorno de ejecución de la Dirección General de Tráfico.

4.1.2 WS-Security

Las ventajas proporcionadas por el estándar WS-Security en la capa de mensaje son múltiples. WS-Security se utiliza para mitigar factores de riesgo comunes a los cuales los Servicios Web están sujetos:

- Riesgos asociados al proceso de autenticación.
- Riesgos asociados a la falta de integridad del mensaje.
- Riesgos asociados a la falta de confidencialidad durante el transporte del mensaje.

Estos factores de riesgo están ilustrados en la figura siguiente.

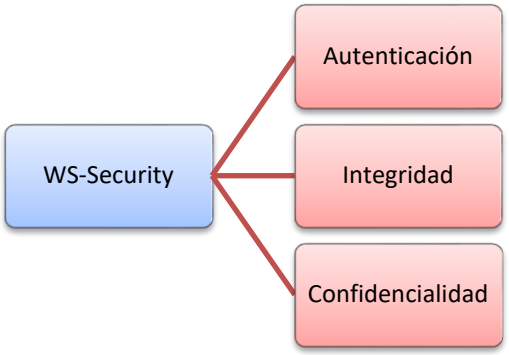

Ilustración 17 - Riesgos comunes asociados a la utilización de Servicios Web

WS-Security ayuda a mitigar estos factores aplicando medidas oportunas para contrarrestar el riesgo. En el caso de falta de *autenticación*, WS-Security proporciona un mecanismo estándar para asociar la identidad de un usuario, un token, al mensaje. En lo que concierne a la *integridad* del mensaje, WS-Security proporciona un mecanismo para que el mensaje pueda ser leído exclusivamente por su destinatario a través del mecanismo de cifrado del mensaje. En lo que concierne a la *integridad* del mensaje, WS-Security proporciona un mecanismo estándar para aplicar una firma al mensaje.

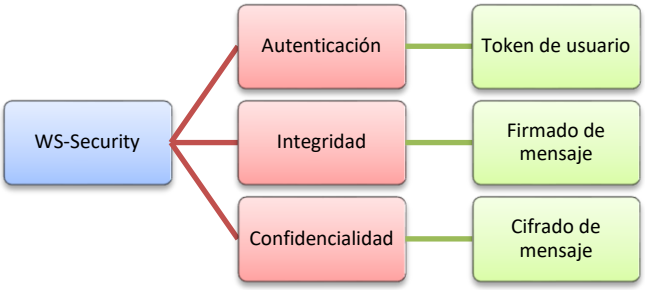

Ilustración 18 - Soluciones proporcionadas por WS-Security

Esta especificación es parte de un marco más amplio cuyo objetivo es proporcionar un modelo de seguridad para Servicios Web. El marco actualmente propuesto queda ilustrado en la figura siguiente.

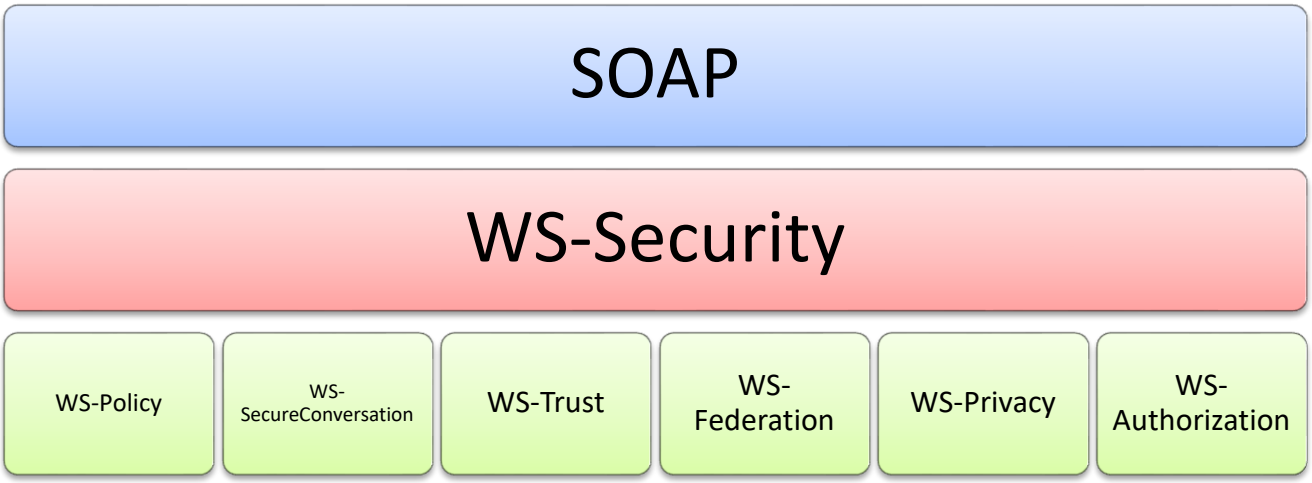

Ilustración 19 - Modelo de seguridad para Servicios Web

4.1.2.1 Autenticación con WS-Security

Esta versión de la especificación se limitará a la utilización del estándar WS-Security en lo que concierne a la funcionalidad de autenticación de usuarios.

Las especificaciones que componen este marco de seguridad son extensiones del protocolo SOAP. En el caso de WS-Security utilizado para la propagación de un token de usuario, el mensaje SOAP se modifica añadiendo una cabecera de seguridad:

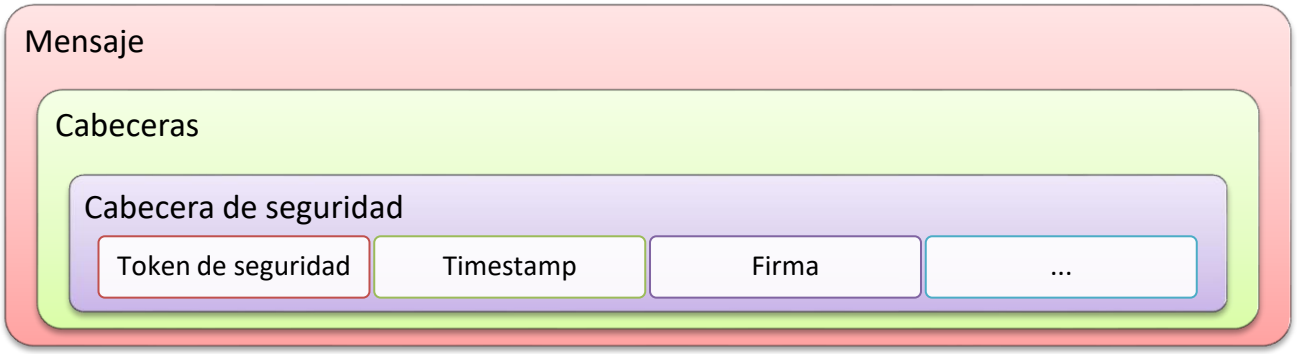

Ilustración 20 – Cabecera de seguridad WS-Security

La cabecera de seguridad es el vehículo a través del cual WS-Security proporciona al protocolo SOAP las funcionalidades extendidas de seguridad. En las cabeceras de seguridad, por lo tanto, se transportan los datos utilizados para el proceso de autenticación.

Para favorecer la interoperabilidad entre plataformas[20] el estándar WS-I Basic Profile (9) y su extensión WS-I Basic Security Profile v. 1.0 (10) permiten la utilización de un conjunto de mecanismos portables para el proceso de autenticación. En lo que concierne a esta versión de la especificación, los mecanismos utilizables son los siguientes:

- Username token.
- Certificado digital X.509.

El username token es la forma de autenticación más sencilla en la que se utiliza la pareja (nombre de usuario, contraseña) como credenciales de un usuario.

La utilización de un certificado digital es un mecanismo más complejo para el cual se puede averiguar si quien está enviando la solicitud posee la clave privada de la clave de encriptación asimétrica incluida en el certificado.

Para mayor información sobre estos dos mecanismos referirse a Username Token Profile 1.0 (11), Username Token Profile 1.1 (12), X.509 Certificate Token Profile 1.0 (11), X.509 Certificate Token Profile 1.1 (12).

[20] La interoperabilidad entre plataformas es una de las características que definen los Servicios Web. En el ámbito del desarrollo de aplicaciones Java™ para la Dirección General de Tráfico esta característica es irrenunciable.

4.1.2.2 Utilizar un mecanismo de autenticación

Como ha quedado explicado en las secciones precedentes, esta especificación permite utilizar un username token o un certificado X.509 como mecanismo de autenticación para Servicios Web desarrollados para la Dirección General de Tráfico.

4.1.2.2.1 Utilizar un username token

Para utilizar este mecanismo de autenticación es necesario insertar en las cabeceras de seguridad un elemento cuya estructura está especificada en la especificación WS-Security (12), Username token Profile 1.0. A título ilustrativo se propondrá un ejemplo de mensaje SOAP que utilice WS-Security con un username token.

El código siguiente es el mensaje de partida:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header/>
<soapenv:Body>
<!–cuerpo del mensaje -->
</soapenv:Body>
</soapenv:Envelope>
```

Código 3 – Mensaje SOAP original (sin username token)

El siguiente ejemplo es el mensaje SOAP precedente donde se ha insertado una cabecera de seguridad con un username token:

```
<soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header>
<wsse:Security soapenv:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-
secext-1.0.xsd">
<wsse:UsernameToken>
<wsse:Username>nombre</wsse:Username>
<wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wssusername-
token-profile-1.0#PasswordText">password</wsse:Password>
</wsse:UsernameToken>
</wsse:Security>
</soapenv:Header>
<soapenv:Body>
<!–cuerpo del mensaje -->
</soapenv:Body>
```

```
</soapenv:Envelope>
```

Código 4 – Mensaje SOAP con cabecera de seguridad con username token

Básicamente se trata de un elemento, **`<wsse:UsernameToken/>`**, que a su vez contiene dos elementos hijos, **`<wsse:Username/>`** y **`<wsse:Password/>`**, con los cuales se especifica la pareja (nombre, password) que constituye la credencial del usuario.

4.1.2.2.2 Utilizar un certificado X.509

Para utilizar el mecanismo de autenticación basado en certificados X.509, análogamente al ejemplo realizado para el username token, habrá que insertar en las cabeceras de seguridad un elemento oportunamente formateado.

La estructura del elemento está especificada por el estándar WS-Security (12), X.509 Token Profile 1.1.

El elemento a insertar posee la siguiente estructura:

```
<wsse:BinarySecurityToken
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soapmessage-
security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-tok
en-profile-1.0#X509v3">
<!-- token -->
</wsse:BinarySecurityToken>
```

Código 5 –Cabecera de seguridad con token de certificado X.509

En este caso el elemento padre es el elemento **`<wsse:BinarySecurityToken/>`** que contiene como valor el token mismo, que ha sido removido y comentado en ejemplo de código precedente.

4.2 Autenticación para Web Services REST

Los servidores disponibles en la Dirección General de Tráfico, se basan en una tecnología conforme al estándar Java™ EE v. 5 y por lo tanto carecen de soporte para delegar en el contenedor la seguridad de los servicios web REST, tal como disponemos para los servicios web SOAP.

Para proteger los servicios web REST, se aplicarán las directrices descritas en el capítulo 3 Protección de recursos en aplicaciones web

Los métodos de autenticación a utilizar serán los siguientes:

- HTTP Basic
- Certificado de cliente

4.2.1 HTTP Basic

En este método de autenticación, se introduce en la cabecera http "Authorización" de la petición, las credenciales del usuario de control en formato Base64.

```
Authorization: Basic aWRfdXN1YXJpbzpjb250cmFzZcOxYQ==
```

El acceso al servicio REST a través de este mecanismo de autenticación debe realizarse obligatoriamente en modo seguro (directriz que aplica al acceso de todos los recursos web)

La validación de las credenciales, debe quedar fuera del servidor de aplicaciones y se debe delegar al BUS de Integración de la DGT (IBM WebSphere® DataPower)

4.2.2 Certificado de cliente

En este método de autenticación, el servidor valida el certificado con el que se presenta el cliente que invoca al servicio durante la negociación del protocolo SSL/TLS.

Este tipo de autenticación se debe limitar su uso y queda restringido a servicios expuestos a internet o a aquellos servicios cuyos requisitos no funcionales de seguridad así lo requieran.

La validación del certificado, debe quedar fuera del servidor de aplicaciones y se debe delegar al BUS de Integración de la DGT (IBM WebSphere® DataPower), tal como se indica en el punto 6.2 Validación de un certificado

5 Integración con IBM WebSphere® DataPower

IBM WebSphere® DataPower es un dispositivo que la Dirección General de Tráfico ha adquirido para facilitar la implementación y la ejecución de algunas de las políticas de seguridad de la casa. En lo que concierne a las aplicaciones Java™ EE de la DGT, WebSphere® DataPower actuará

de forma transparente y, por lo tanto, no deberán acometerse desarrollos personalizados para integrarse con este componente de infraestructura: en su lugar, las aplicaciones deberán utilizar el modelo de seguridad basado en roles, objetivo de los capítulos precedentes, y seguir un conjunto de directrices marcadas por el Departamento de Arquitectura:

5.1 Integración de aplicaciones web

Para integrar aplicaciones web Java™ EE en el entorno de ejecución protegido por WebSphere® DataPower es necesario:

- Configurar el descriptor de despliegue del módulo web para que el tipo de autenticación sea HTTP básico.
- Solicitar la integración de la aplicación al equipo de mantenimiento de WebSphere® DataPower quién solicitará la información necesaria para cumplimentar la solicitud.

5.1.1 Configuración del mecanismo de autenticación

La configuración requerida para que un módulo web Java™ EE de la DGT se integre en el entorno de ejecución protegido por WebSphere® DataPower es que el mecanismo de autenticación configurado se HTTP básico.

La configuración del mecanismo de autenticación se efectúa en el descriptor de despliegue del módulo web tal y como se ilustra en el fragmento de código siguiente:

```
<login-config>
  <auth-method>BASIC</auth-method>
</login-config>
```

Código 6 – Configuración de la autenticación de tipo HTTP básico

5.1.2 Configuración del mecanismo de autorización

Para implementar la seguridad declarativa en el servidor de aplicaciones Websphere se deberá mapear (a través de la consola administrativa) los roles definidos en la aplicación a un grupo previamente definido en el LDAP. Los usuarios serán asignados al grupo LDAP correspondiente, para poder acceder a la aplicación con el rol/roles que le correspondan.

NO SE PERMITE definir atributos en LDAP cuyo uso exclusivo sea para realizar autorización programática.

5.2 Integración de Servicios Web

La integración de Servicios Web Services con los servicios ofrecido por WebSphere® DataPower[21] se efectúa bajo petición. El equipo de mantenimiento de WebSphere® DataPower proporcionará una plantilla de solicitud a rellenar por la aplicación especificando las características solicitadas del Servicio Web existente y los servicios que se piden a WebSphere® DataPower.

5.3 Utilización del contexto de seguridad

El entorno de ejecución pone a disposición de la aplicación un contexto de seguridad donde están almacenadas las credenciales del usuario autenticado así como otra información "adicional" que pueda haberse recuperado por los registros de usuarios que participan en la fase de autenticación[22].

5.3.1 Dependencia de las aplicaciones con los artefactos del contexto de seguridad.

Las aplicaciones Java™ EE establecen una dependencia con el artefacto que contiene el API del contexto de seguridad así como las factorías necesarias durante la recuperación de una instancia de dicho contexto.

El artefacto en cuestión es el siguiente[23]:

```
<dependency>
  <groupId>es.trafico.seguridad</groupId>
  <artifactId>dgt-iUser</artifactId>
  <version>1.0</version>
</dependency>
```

[21] Tales como firmado digital, comprobación de firma digital, transformación del mensaje, etc.

[22] Un ejemplo notable de este tipo de registros es la plataforma @firma: si un usuario se ha autenticado presentando unas credenciales comprobadas por la plataforma @firma entonces el contexto de seguridad pondrá a disposición de las aplicaciones la información devuelta por dicha plataforma.

[23] Fragmento del descriptor **`pom.xml`**. La versión más reciente debe consultarse en el repositorio de librerías.

Código 7 – Artefacto que define el API del contexto de seguridad

5.3.2 Recuperación de una referencia al contexto de seguridad

Para recuperar una instancia del contexto de seguridad se utiliza la factoría del contexto de seguridad como queda ilustrado en el fragmento de código siguiente:

```
DGTSecurityContextFactory fact = DGTSecurityContext.createSecurityContextFactory();
IDGTUser user = fact.getUser();
```

Código 8 – Recuperación de una referencia al contexto de seguridad

5.3.3 Utilización del contexto de seguridad

Una vez adquirida una referencia a una instancia de la interfaz **`IDGTUser`**, la aplicación podrá solicitar al contexto de seguridad la información requerida.

La descripción del API del contexto de seguridad es objeto de otro documento.

6 Apéndice A: Proceso de autenticación de la Dirección General de Tráfico

Este capítulo tiene como único objetivo proporcionar a los equipos de desarrollo una visión *funcional* de las operaciones realizadas durante el proceso de autenticación de un usuario en una aplicación de la Dirección General de Tráfico.

Aunque el proceso de autenticación se lleve a cabo de forma transparente para el equipo de desarrollo de las aplicaciones, el conocimiento de este proceso puede ser importante durante la fase de análisis funcional de la aplicación.

Las aplicaciones en la plataforma Java™ EE de la Dirección General de Tráfico soportan la autenticación de usuarios que utilicen un certificado digital de una autoridad de certificación aceptada por la Administración. Las autoridades de certificación admitidas, así como los casos en los cuales los

usuarios estén presentes también en el registro de usuarios de la DGT[24], son parte de la configuración del sistema de seguridad diseñado por el Departamento de Arquitectura. Puede darse la posibilidad de que un requerimiento funcional de una aplicación tenga un impacto en dicha configuración o, en el caso en que no pudiera implementarse, tenga un impacto en los requerimiento de la aplicación misma que deberá modificarse para ajustarse a los vínculos del sistema de seguridad.

6.1 El algoritmo de autenticación

Las operaciones ejecutadas por el sistema de autenticación de la Dirección de Tráfico se ilustran en la figura a continuación.

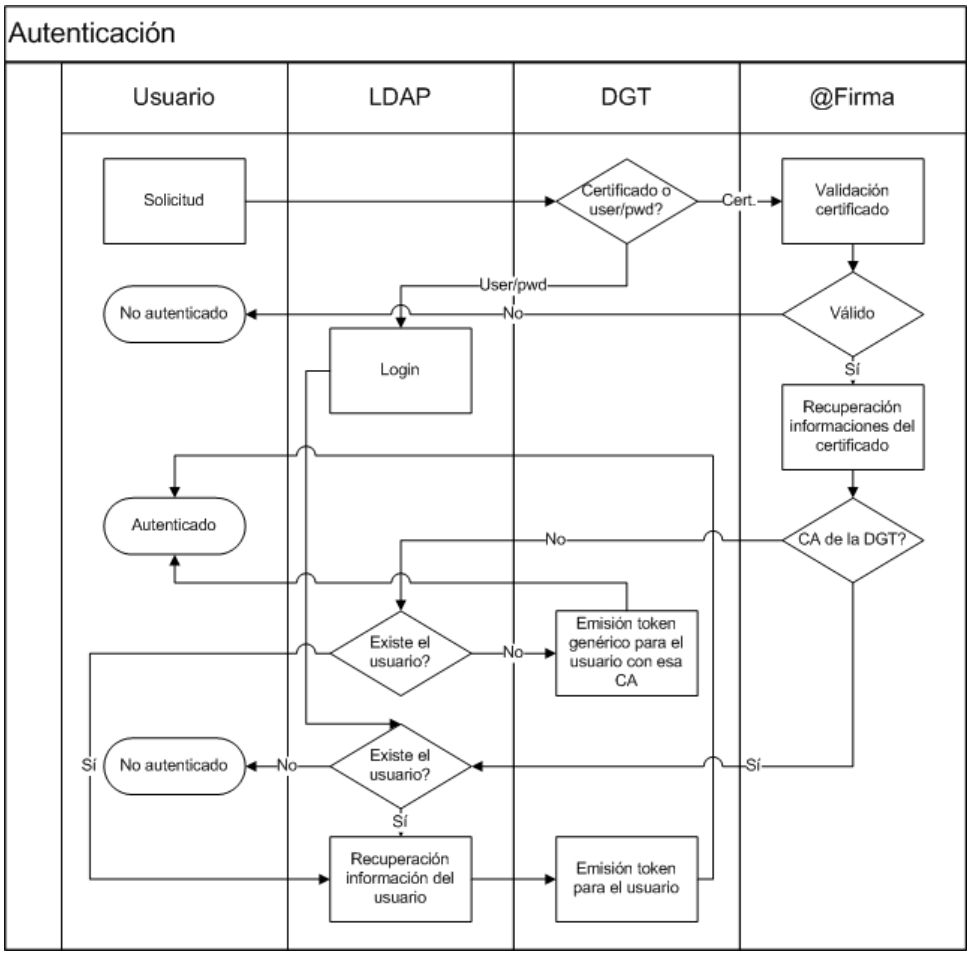

Ilustración 21 – Diagrama de flujo de alto nivel del proceso de autenticación.

[24] Un directorio LDAP.

En el algoritmo ilustrado sólo se detallan los procesos de alto nivel y se omiten los detalles de implementación. Los sistemas funcionales que participan en el proceso de autenticación son[25]:

- La plataforma @firma.
- El sistema de autenticación de la Dirección General de Tráfico.
- El registro de usuarios de la Dirección General de Tráfico, un directorio LDAP.
- El usuario.

En respuesta a la solicitud de un usuario para un recurso protegido, el sistema efectúa las siguientes operaciones:

- *Comprobación de credenciales*: Comprueba las credenciales del usuario. Si el usuario presenta como credenciales la pareja (nombre de usuario, contraseña), se efectúa un *login* comprobando la información en el registro de usuarios de la Dirección General de Tráfico. Si el usuario presenta como credenciales un certificado X.509[26] se procede a la *validación del certificado.*
- *Login*: el proceso de login a través de las credenciales básicas de un usuario se efectúa comprobando en el directorio LDAP si las credenciales del usuario son correctas. Si lo son, se procede a la recuperación de la información del usuario mientras en caso contrario el proceso de autenticación termina con un error.
- *Recuperación de la información del usuario*: después de haber efectuado el login de un usuario catalogado en el registro de usuarios de la Dirección General de Tráfico, se procede a la recuperación del subconjunto de información del usuario presente en el directorio LDAP y de interés para la arquitectura de seguridad en tiempo de ejecución. Después de este proceso se efectúa la emisión del token de seguridad del usuario.
- *Emisión del token de seguridad*: el token de usuario emitido para el usuario se asociará a la solicitud en curso y el proceso de autenticación termina con éxito.
- *Validación del certificado*: el proceso de validación del certificado efectúa las comprobaciones necesarias para averiguar si el certificado puede aceptarse como válido

[25] En el orden ilustrado.

[26] En las cabeceras de seguridad WS-Security o durante el establecimiento del canal de comunicación a través del protocolo SSL.

para la solicitud en curso. El proceso de validación se describe brevemente más adelante en este documento. Si el proceso de validación determina que un certificado es válido se procede a la recuperación de la información del certificado. En caso contrario el proceso de autenticación termina con un error.

- *Recuperación de la información del certificado*: durante este proceso se extraen del certificado las informaciones en él contenidas que la infraestructura decide propagar en la solicitud en curso y en las estructuras de datos que representan al usuario que se está autenticando. Durante este proceso se determina la autoridad certificadora (CA) que ha emitido el certificado: si ésta es la Dirección General de Tráfico se procede con el proceso de login. Si la entidad emisora no es la DGT, se procede a la emisión de un token genérico para la CA en cuestión (a no ser que el usuario del certificado estén presente en el directorio LDAP, en cuyo caso puede emitirse un token específico).
- *Emisión de un token genérico para una CA*: cuando la autoridad certificadora de un certificado no es la Dirección General de Tráfico y el usuario del certificado no se ha encontrado en el directorio LDAP se crea un token genérico cogiendo como único parámetro del proceso de creación la CA del certificado.

6.2 Validación de un certificado

El proceso de validación de un usuario, representado en la Ilustración 22, es el proceso que efectúa las comprobaciones necesarias para determinar si un certificado es válido o no. Por validez de un certificado no se entiende sólo su validez formal sino su validez como medio de autenticación para la Dirección General de Tráfico.

Desde el punto de vista formal, la validación sobre el certificado hace una serie de comprobaciones para comprobar que el certificado sea auténtico, formalmente válido y en vigor. Desde el punto de vista de la Dirección General de Tráfico se efectúan las comprobaciones necesarias para averiguar si el certificado es aceptable según las normas y las políticas de seguridad establecidas en la administración.

El proceso de validación de certificado puede someramente describirse con este algoritmo:

- Se comprueba si la CA que ha emitido el certificado es una CA aceptable por la Dirección General de Tráfico comprobando en la configuración actualmente en vigor en la Dirección General de Tráfico si la CA está dada de alta en la lista de CAs admitidas. Si la CA no puede admitirse este proceso termina con un error.
- Se comprueba la validez temporal del certificado para averiguar si está en vigor. Si no lo fuere este proceso termina con un error.
- Se comprueba la validez formal del certificado efectuando el conjunto de comprobaciones previstas por la Dirección General de Tráfico. Si el certificado resulta formalmente inválido este proceso termina con un error.
- Se invoca a la plataforma @Firma para efectuar las validaciones previstas en esta plataforma. Si la plataforma devuelve como respuesta que el certificado es inválido este proceso termina con un error.

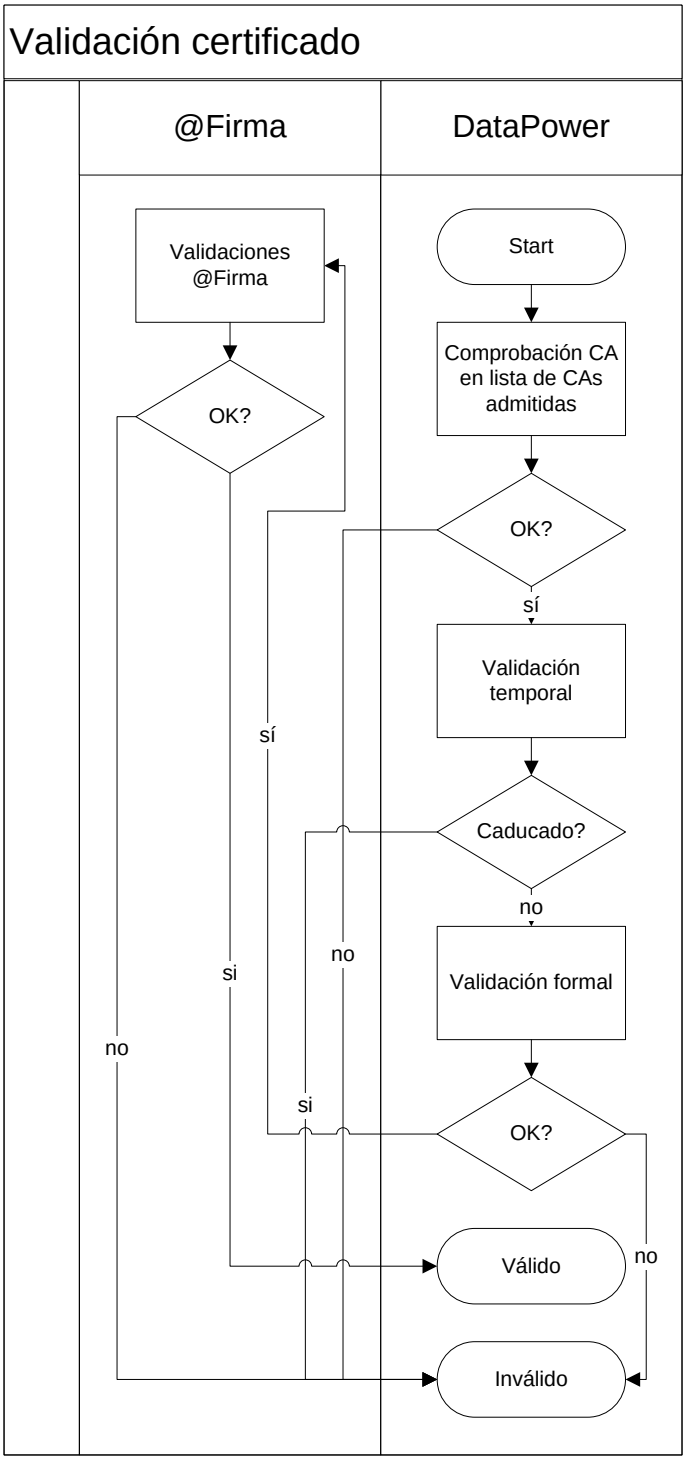

Ilustración 22 – Proceso de validación de un certificado

6.3 Proceso de login en el registro de usuarios de la DGT

Se procede al proceso de login en el registro de usuarios de la Dirección General de Tráfico exclusivamente para los usuarios que puedan estar catalogados en el directorio LDAP.

El directorio LDAP de la Dirección General de Tráfico mantiene el registro de los funcionarios de esta administración así como el registro de un subconjunto de otras identidades. Algunas de ellas corresponden a DNs *técnicos*.

El acceso al LDAP se efectúa básicamente a través de una consulta que selecciona el registro del usuario a través de su UID. En la estructura de directorio de la DGT el UID es el NIF del usuario.

7 Bibliografía

1. **Shannon, Bill (Sun Microsystems).** JSR 244: Java Platform, Enterprise Edition 5 (Java EE 5) Specification. *Java Community Process.* [En línea] http://jcp.org/en/jsr/detail?id=244.
2. **Varios.** Web Services Security. [En línea] http://www.oasis-open.org/specs/index.php.
3. **Mordani, Rajiv (Sun Microsystems, Inc.).** JSR 154: Java Servlet 2.4 Specification. *Java Community Process.* [En línea] http://jcp.org/en/jsr/detail?id=154.
4. **Delisle, Pierre (Sun Microsystems, Inc.), Luehe, Jean (Sun Microsystems, Inc.) y Roth, Mark (Sun Microsystems, Inc.).** JSR 245: JavaServer Pages 2.1. *Java Community Process.* [En línea] 11 de 05 de 2006. http://jcp.org/en/jsr/detail?id=245.
5. **Kotamraju, Jitendra (Sun Microsystems, Inc.).** JSR 224: Java API for XML-based Web Services (JAX-WS) 2.0. *Java Community Process.* [En línea] http://jcp.org/en/jsr/detail?id=224.
6. **Mordani, Rajiv (Sun Microsystems).** JSR 250: Common Annotation for the Java Platform. *Java Community Process.* [En línea] http://jcp.org/en/jsr/detail?id=250.
7. **Sun Microsystems, Inc.** Java Authentication and Authorization Service. [En línea] http://java.sun.com/javase/6/docs/technotes/guides/security/.
8. **Pandey, Dhiru (Sun Microsystems, Inc.).** JSR 109: Web Services for the Java EE Platform. *Java Community Process.* [En línea] jcp.org/en/jsr/detail?id=109.
9. **Ballinger, Keith (Microsoft Corporation), y otros, y otros.** Basic Profile Version 1.0. *Web Services Interoperability Organization.* [En línea] 16 de 04 de 2004. http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html.
10. **McIntosh, Michael (IBM), y otros, y otros.** Basic Security Profile Version 1.0. *Web Services Interoperability Organization.* [En línea] 30 de 03 de 2007. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2007-03-30.html.
11. **OASIS.** Web Services Security 1.0. *OASIS.* [En línea] http://www.oasis-open.org/specs/#wssv1.0.
12. —. Web Services Security 1.1. [En línea] http://www.oasis-open.org/specs/#wssv1.1.
13. **DeMichiel, Linda (Sun Microsystems) y Keith, Michael (Oracle).** JSR 220: Enterprise JavaBeans 3.0. *Java Community Process.* [En línea] http://jcp.org/en/jsr/detail?id=220.