Protección en imágenes JPEG

16/01/2018

TRABAJO DE FINAL DE GRADO, MODALIDAD A

Estudiante: Brian Martínez Álvarez
Director/a: Silvia Llorente Viejo

FACULTAT D'INFORMÀTICA DE BARCELONA
UNIVERSIDAT POLITÈCNICA DE CATALUNYA

Resumen

La mayoría de empresas invierten gran cantidad de dinero en mejoras de seguridad en sus redes y sistemas para evitar ataques informáticos que puedan comprometer información privada o resulten en la pérdida de sus clientes.

Pese a tantas mejoras, hay elementos dentro de la informática que no protegemos adecuadamente, como es el caso de las imágenes. Compartimos cientos de imágenes entre los nuestros durante el año sin conocer qué será de esa foto una vez la hayamos enviado y sin saber que las imágenes no tienen ningún sistema de seguridad que las proteja.

Este proyecto busca dar una solución a este problema de seguridad con el que hemos vivido muchos años todos nosotros. En concreto vamos a centrarnos en el formato de imágenes JPEG mediante el uso de JPSearch, un nuevo formato de normas internacionales. Se ha desarrollado como prueba de concepto en un entorno controlado como el de una empresa o comunidad con dos objetivos principales: añadir políticas de privacidad a la imagen utilizando XACML, un lenguaje declarativo de políticas de control de acceso, y el cifrado tanto de las políticas como de los datos de la imagen. Como resultado tendremos una imagen que podrá ser compartida de forma segura.

Resum

La majoria d'empreses inverteixen gran quantitat de diners en millores de seguretat en les seves xarxes i sistemes per evitar atacs informàtics que puguin comprometre informació privada o resultin en la pèrdua dels seus clients.

Malgrat tantes millores, hi ha elements dins de la informàtica que no protegim adequadament, com és el cas de les imatges. Compartim centenars d'imatges entre els nostres durant l'any sense conèixer què serà d'aquesta foto un cop l'haguem enviat i sense saber que les imatges no tenen cap sistema de seguretat que les protegeixi.

Aquest projecte busca donar una solució a aquest problema de seguretat amb el qual hem viscut molts anys tots nosaltres. En concret ens centrarem en el format d'imatges JPEG mitjançant l'ús d'JPSearch, un nou format de normes internacionals. S'ha desenvolupat com a prova de concepte en un entorn controlat com el d'una empresa o comunitat amb dos objectius principals: afegir polítiques de privacitat a la imatge utilitzant XACML, un llenguatge declaratiu de polítiques de control d'accés, i el xifrat tant de les polítiques com de les dades de la imatge. Com a resultat tindrem una imatge que podrà ser compartida de forma segura.

Abstract

Most companies invest a lot of money in improvements of security in their networks and systems to avoid computer attacks that can compromise private information or result in the loss of their clients.

Despite many improvements, there are elements within the computer that we do not protect properly, as is the case with images. We share hundreds of images among ours during the year without knowing what will be of that photo once we have sent it and without knowing that the images do not have any security system that protect them.

This project seeks to provide a solution to this security problem which we have lived all of us for many years. In particular, let's focus on the JPEG image format by using JPSearch, a new standard international format that It has been developed as proof of concept in a controlled environment as a company or community with two main objectives: to add policies of privacy to the image using XACML, a declarative language of access control, and the encryption of both the policies and the image data. As a result, we will have an image that can be shared safely.

Glosario

Balana: aplicación de código abierto que soporta XACML en las versiones 1.0, 1.1, 2.0 y 3.0, que evalúa las peticiones de acceso contra de las políticas de privacidad.

Criptografía simétrica: Los sistemas de cifrado simétrico son aquellos que utilizan la misma clave para cifrar y descifrar un mensaje. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. El remitente cifra un mensaje usando la clave, lo envía al destinatario, y éste lo descifra con la misma clave.

Eclipse IDE: Es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones de cliente.

GitHub: Es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git.

JAVA: es un lenguaje de programación pensado como un lenguaje orientado a objetos. Es uno de los lenguajes de programación más utilizados, y se utiliza tanto para aplicaciones web como para aplicaciones de escritorio.

JPEG: Joint Photographic Experts Group, grupo de trabajo conjunto de la Organización Internacional de Normalización (ISO) y la Comisión Electrotécnica Internacional (IEC) que creó un estándar de codificación y compresión de imágenes fijas. Además de ser un método de compresión, es considerado un formato de archivo.

JPSearch: estándar, propiedad de JPEG, que proporciona un conjunto de interfaces estandarizadas para los sistemas de gestión y recuperación de imágenes digitales.

XACML: eXtensible Access Control Markup Language. Estándar que define un lenguaje declarativo de políticas de control de acceso implementado en XML y un modelo de procesamiento que describe cómo evaluar peticiones de acceso según las reglas definidas en las políticas.

Índice de tablas

Índice de figuras

Índice de contenido

1. Introducción

1.1. Descripción del proyecto

La mayoría de nosotros hemos tratado con imágenes en diferentes ámbitos y situaciones desde hace muchos años y hemos visto éstas como un simple reflejo de un momento real pasado. A diferencia de la mayoría, los ingenieros hemos visto en ellas un conjunto de bits, organizados y estructurados, formando algo que puede parecer simple.

A medida que la tecnología avanza y vamos incorporándola a nuestras vidas, hemos ido mejorando a conciencia la seguridad de nuestros sistemas y aplicaciones. La aparición de las redes sociales y de la compartición de ficheros entre amigos, familia y compañeros de trabajo ha pasado a ser un hábito para todos nosotros que a menudo olvidamos lo peligroso que puede ser.

La mayoría de empresas invierten gran cantidad de dinero en mejoras de seguridad en sus redes y sistemas para evitar ataques informáticos que puedan comprometer información privada o resulten en la pérdida de sus clientes. Pese a tantas mejoras, hay elementos dentro de la informática que no protegemos adecuadamente, como es el caso de las imágenes. Compartimos cientos de imágenes entre los nuestros durante el año sin conocer qué será de esa foto una vez la hayamos enviado y sin saber que las imágenes no tienen ningún sistema de seguridad que las proteja.

Este proyecto busca dar una solución a este problema de seguridad con el que hemos vivido muchos años todos nosotros. En concreto vamos a centrarnos en el formato de imágenes JPEG, muy conocido y usado por miles de herramientas y tecnologías como cámaras de fotografías, ordenadores, móviles, tablets, etc. La selección de este formato recae sobre todo en el uso masivo que tiene a día de hoy y la compatibilidad que presenta con las numerosas aplicaciones de los distintos sistemas informáticos.

Actualmente, las imágenes JPEG presentan un doble problema de seguridad. En primer lugar, al no estar securizadas, permiten que cualquier persona con el control de una imagen pueda ver su contenido libremente. En este caso, es posible que esa persona no sea el destinatario deseado por la persona que realizó y/o envió inicialmente la imagen. Como comentábamos anteriormente, una vez la enviamos a alguien, perdemos el control de la imagen. En segundo lugar, las imágenes JPEG pueden guardar metadatos. Los metadatos son un tipo de datos que permiten describir otros datos. En este caso, permiten describir elementos relacionados con la imagen. Estos datos pueden ser inofensivos como por ejemplo un título, las dimensiones de la imagen o el tamaño en bytes. Por otra parte, pueden contener información considerada personal o privada como la localización desde donde se realizó la imagen, el nombre del remitente o el dispositivo con el que se realizó la imagen.

1.2. Actores

En este proyecto tenemos tres actores bien diferenciados. En primer lugar encontramos al tutor o director del proyecto, Silvia Llorente, que nos ha guiado durante la realización de éste con el fin de conseguir todos nuestros objetivos. En segundo lugar, el desarrollador Brian Martínez Álvarez, que ha tenido la función de desarrollar y documentar todo el proyecto en el tiempo establecido cumpliendo los objetivos marcados previamente. Finalmente, encontramos a los usuarios de las empresas, que serán los que saquen partido de este proyecto usando las herramientas desarrolladas con el fin de mejorar su empresa.

Pese a que este proyecto puede tener un alcance global, capaz de dirigirse a cualquier imagen del planeta, a fin de experimentar su uso y desarrollo, se ha determinado usarlo como una prueba de concepto y dirigirlo a entornos de empresa o comunidades. Por tanto, su uso será dirigido a usuarios conocidos de una empresa o comunidad.

Dado que el uso de la extensión de *JPSearch* para securizar la imagen es opcional, securizar imágenes ha de ser una decisión obligatoria de la empresa para asegurar que el intercambio de imágenes entre usuarios de la misma empresa es seguro y no haciendo posible que un usuario interno envíe una imagen corporativa a un usuario externo de la empresa.

El hecho de usar *JPSearch* [1] como estándar para securizar imágenes implica un impacto en la usabilidad al usuario. Si un usuario quiere enviar una imagen segura a otro usuario, primero debe usar *JPSearch* para hacer todo el proceso de edición de metadatos de *JPSearch* y cifrado. Esto puede parecer costoso, pero permite ofrecer la confianza de seguridad y protección de los datos.

1.3. Temática

El uso de este proyecto viene motivado por mi afición a la seguridad informática y por un proyecto que realicé en la asignatura de Aplicaciones distribuidas, basado en la esteganografía de imágenes PNG. Este proyecto permite la ocultación y revelación de mensajes ocultos en imágenes PNG.

La idea de este proyecto es seguir con el camino de la seguridad informática y de la mejora de las tecnologías que a día de hoy usamos. Permitirá mejorar la seguridad y privacidad de las personas manteniendo sus imágenes a salvo de terceros.

La decisión de escoger este proyecto recae en formar parte de la investigación de mejoras en seguridad informática que siempre me ha llamado la atención. Ayudar a que la seguridad de la información progrese y que permite que usuarios de a pie puedan proteger algo tan personal como puede llegar a ser una simple fotografía.

2. Objetivo del proyecto

2.1. Formulación del problema

Como hemos comentado anteriormente, el problema que hay a día de hoy es que las imágenes se pueden intercambiar entre usuarios de forma fácil e insegura. Encontramos diversos mecanismos de protección a nivel de aplicación y de transporte pero olvidamos proteger el paquete. En este caso, al intercambiar imágenes, nos encontramos con protecciones en cuanto al medio de transporte, cifrando comunicaciones y aplicando herramientas de protección de red, pero olvidamos lo peligroso que es el mal uso que puede tener una imagen en las manos inadecuadas.

Las imágenes pueden contener información sensible guardada en sus metadatos e información privada en la propia imagen haciendo posible un futuro ataque informático o extorsión. Una simple imagen con información importante puede derivar a un gran problema de seguridad si se hace un uso fraudulento de ella. La inserción de políticas de privacidad y del cifrado de las imágenes podría dar una solución a este problema.

2.2. Objetivos

El objetivo de este proyecto es ofrecer una capa de seguridad a las imágenes JPEG de forma que al ser enviadas, sólo las personas autorizadas puedan ver los datos adecuadamente gracias a la inserción de políticas de privacidad mediante XACML [2] [3] [4] y el cifrado tanto de las políticas como de los datos de la imagen. Estas medidas ofrecen la garantía de proteger la imagen de terceros, permitiendo que sólo aquellas personas que el remitente esté interesado puedan ver el contenido.

Para ello, este proyecto propone aprovechar el estándar de JPSearch, que permite la manipulación de la imagen y la edición de metadatos específicos de JPSearch. Desde esa base, se propone la inserción de una extensión que permita añadir reglas de control de acceso usando XACML, lenguaje declarativo implementado en XML (*eXtensible Markup Language*) y un modelo de procesamiento que describe cómo evaluar peticiones de acceso según las reglas definidas en las políticas.

Una vez las políticas sean insertadas en la imagen, el usuario o los usuarios receptores deberán generar una petición de acceso, también en XACML, y juntas (política de control de acceso y petición de acceso) se envían contra el autorizador Balana, aplicación de código abierto que soporta XACML en las versiones 1.0, 2.0 y 3.0, que evalúa las peticiones de acceso según las políticas declaradas.

Con el fin de mejorar la seguridad tanto de los datos como del transporte de las imágenes, se ha añadido una capa de seguridad al proyecto, encriptando tanto las políticas insertadas en la imagen como los datos de la propia imagen. En la Figura 1 podemos ver de forma más visual cómo funciona el uso de políticas con peticiones y el autorizador:

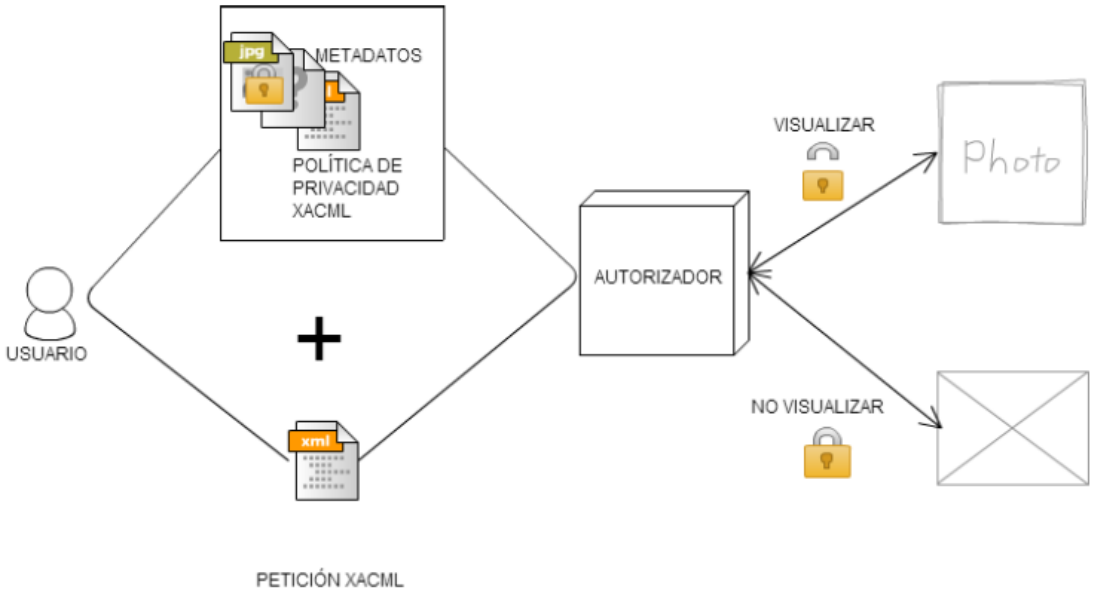


Figura 1: Esquema del funcionamiento de las reglas de autorización

Un pilar fundamental de este proyecto es conseguir securizar las imágenes JPEG siguiendo el artículo mencionado anteriormente por lo que respetaremos la estructura que propone de la imagen. Esta estructura implica añadir metadatos de JPSearch, añadir las reglas XACML en un campo APP11 y modificar la zona de datos, que actualmente forman el campo SOS (Start of Scan) de la imagen, para pertenecer también a un campo APP11. Los campos APP en imágenes JPEG, son campos de longitud variable que sirven para añadir información extra a la imagen. A la hora de descifrar la imagen, los datos volverán a su estado original (SOS) para que puedan ser leídos correctamente por los visores convencionales. En la Figura 2 podemos ver en detalle el cambio de estructura en las imágenes JPEG que propone este proyecto:

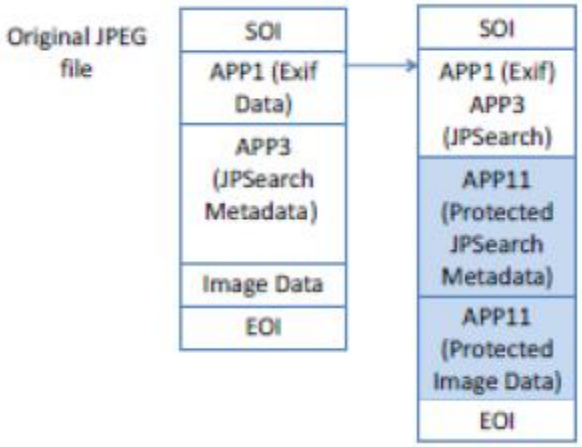


Figura 2: Cambio de estructura de la imagen

La implicación que presenta el añadir una extensión al estándar de JPSearch con el fin de usarlo, es que obligará a usar el editor de JPSearch para poder cifrar la imagen. Por lo tanto, si no se adquiere dicha herramienta, no se podrá cifrar la imagen y, por consiguiente, no se podrá proteger la imagen antes de compartirla.

Para poder cifrar y descifrar la imagen, se ha determinado usar criptografía simétrica. En este caso, es necesario que el usuario que envía la imagen cifrada, envíe por algún medio la clave de cifrado a sus receptores. La clave de cifrado y descifrado coincide, por lo que solo es necesario el uso de una clave. Como alternativa se pensó en el uso de clave asimétrica. Los usuarios contendrían una clave privada extraída a través de una clave pública. El usuario emisor cifraría la imagen con su clave privada y los receptores la descifrarían con la clave pública. Este caso también está pensado para entornos controlados como una empresa o comunidad. Por último, también se pensó en usar un servicio externo que proporcione claves y que permita identificar a los receptores con fin de descartar personas no autorizadas. Dado que la imagen puede ser enviada a varios usuarios, el uso de criptografía asimétrica lo hacía poco viable. Además, el impacto de usabilidad hace como preferente la primera opción, determinando finalmente el uso de criptografía simétrica.

Otra decisión importante es el algoritmo criptográfico que se utilice. Ha de ser lo suficientemente robusto como para determinar qué se trata de un sistema seguro. En nuestro caso se ha determinado usar AES [13] con claves de 128 bits en modo ECB.

3. Alcance del proyecto

3.1. Entorno

Este proyecto tiene como alcance cumplir sus objetivos de securizar las imágenes JPEG a la hora de distribuirlas entre usuarios de una empresa o comunidad. Por tanto, está focalizado para que se use dentro de un entorno empresarial donde el número de usuarios es conocido.

Dado que los nombres de usuarios de la empresa son conocidos, se puede cifrar las imágenes teniendo en cuenta el/los destinatario/s. Permite que el usuario emisor primero proteja la imagen añadiendo reglas XACML, seguidamente añada los metadatos que crea necesarios y finalmente cifre la imagen y las reglas, de manera que solo las personas que él crea necesario puedan ver tanto la imagen como sus metadatos.

3.2. Limitaciones

Como comentábamos anteriormente, este proyecto está pensado como una prueba de concepto dentro de un entorno controlado como una empresa, donde los usuarios son conocidos. Existen ciertas limitaciones que cabe tener en cuenta a la hora de usar las aplicaciones desarrolladas:

- **Entorno controlado**: Para que este proyecto funcione ha sido necesario limitar el entorno para que se conozca el nombre de los usuarios. Es por eso, que las claves de cifrado/descifrado y los nombres de usuarios se encuentran en bases de datos que permanecen dentro de la empresa. Más concretamente, las claves de cifrado se encuentran en un repositorio que es accesible por cualquier usuario de la empresa.

- **Fecha actual del sistema**: Uno de los parámetros que puede aparecer en una política de control de acceso es la fecha límite para ver cierto recurso. Por ejemplo, un usuario puede añadir una política que determine que el recurso al cual aplica solo pueda verse si es antes de una fecha. En este caso, cuando los receptores intenten autorizarse para acceder al recurso, se comprobará la fecha actual de sus sistemas para compararlas con la fecha que aparece en la política. Esta idea presenta un error de seguridad que puede ser vulnerado si el receptor cambia la fecha de su sistema antes de pedir la evaluación.

- **Capturas de pantalla**: Los receptores una vez reciben la imagen y son autorizados correctamente, reciben el recurso que solicitaban correctamente. En ese caso, pueden ver la imagen con un visor convencional sin ninguna limitación haciendo posible que los receptores utilicen capturas de pantalla para enviar de manera ilícita la imagen a terceros.

- **Repositorios desprotegidos:** Actualmente tanto el repositorio de claves como el de credenciales de usuario está desprotegido y por tanto cualquier usuario tendría acceso a él. Sería conveniente protegerlo y como mejora, mantenerlo en un servidor aparte y acceder a él a través de Internet.

3.3. Posibles obstáculos

Los principales problemas a los que nos enfrentaremos en la realización de este proyecto son los siguientes:

1) Tiempo limitado: Este suele ser un problema común cuando se desarrolla un proyecto. Al encontrarnos frente a un proyecto de final de carrera, no tenemos clientes que necesiten soluciones, pero tenemos un tiempo limitado fijo para la realización total del proyecto. Dado que el tiempo es muy limitado, se han eliminado algunas de las mejoras que se pensaron inicialmente y nos hemos focalizado en cumplir los objetivos básicos que exige el artículo [11].

2) Vulnerabilidades de los algoritmos de cifrado: Este proyecto es posible dado que actualmente el algoritmo de cifrado AES no es vulnerable. Si se descubriera una manera de hacerlo vulnerable y encontrar las claves secretas en tiempos razonables, este proyecto sería totalmente vulnerable y no otorgaría la protección que ofrece.

4. Estado del arte

4.1. Estado del arte

Actualmente, las imágenes JPEG no tienen ningún tipo de protección en cuanto a su distribución entre personas. No contienen ningún tipo de cifrado o autorizador que permita que terceros no puedan ver su contenido. Con el incremento de las tecnologías de comunicación y las redes sociales, esto ha pasado a ser un peligro en cuanto a la privacidad y protección de las personas que aparecen en las imágenes por lo que es necesario crear una solución a este problema.

Para resolver los problemas de seguridad y privacidad asociados a las imágenes JPEG, el Comité de Estandarización JPEG (ISO/IEC SC29/WG1) [5] puso en marcha una nueva actividad llamada *JPEG Privacy and Security* [6]. En Marzo de 2017, este comité lanzó una *Call for Proposals* [7], para dar respuesta a algunos de los casos de uso y requerimientos identificados.

El grupo de investigación DMAG-UPC [8], perteneciente al Departamento de Arquitectura de Computadores [9] de la Universitat Politècnica de Catalunya [10] participa en esta actividad, a la que ha realizado diversas contribuciones, incluyendo una respuesta a la *Call for Proposals*. En esta respuesta se incluyen algunas de las ideas presentadas en [11].

En paralelo al trabajo contribuido a JPEG, en 2016, el ex alumno Alberto Durán Montoro, presentó el trabajo de final de grado titulado *Privacidad en imágenes jpg mediante XACML* [12], que permitía añadir políticas de control de acceso a través de XACML. Posteriormente, se cifraban dichas políticas y luego se cifraba toda la imagen usando una extensión del estándar de *JPSearch* [1]. En la aplicación implementada en este proyecto, los metadatos no se cifran de forma independiente de los datos.

El proyecto que nos ocupa ha tomado como base algunas funcionalidades del proyecto de Alberto. En concreto, se ha aprovechado la inserción de metadatos *JPSearch*, el uso de políticas XACML y el cifrado de la imagen. Sin embargo, se han modificado algunos aspectos importantes de la implementación, puesto que la imagen cifrada que se generaba no cumplía con algunos de los requerimientos identificados por JPEG, descritos en [7][11]. Una de las principales modificaciones es cifrar los datos de la imagen por separado de los metadatos referentes a las políticas de control de acceso. Además, tal y como especifica [11], tanto los datos de la imagen como las políticas, deben aparecer en la sección APP11 de la imagen. Esta sección se caracteriza por tener un tamaño variable y se utiliza para guardar información extra a la imagen. En nuestro caso, usaremos dos secciones APP11, una para los datos y otra para los metadatos de las políticas de seguridad.

Dado que el cifrado de ambas secciones se hace por separado, se usaran dos claves de cifrado y descifrado diferentes y aleatorias. Una de ellas, necesaria para cifrar las políticas, se compartirá con los usuarios receptores de la imagen. La segunda clave, quedará guardada dentro de los metadatos de APP11 en el elemento *Keyword* codificada en Base64. Ningún usuario, emisor o receptor, tendrá acceso a la segunda clave por lo que solo el sistema será capaz de usarla.

Las funcionalidades implementadas en este proyecto pueden servir para solucionar algunos de los problemas de seguridad y privacidad identificados por el Comité JPEG y que actualmente siguen en proceso de estandarización, sin tener por el momento una solución definitiva.

4.2. Algoritmos criptográficos

Con el fin de incrementar la seguridad de los datos y de evitar que terceras personas puedan ver la imagen o captarla en el transporte se ha determinado proteger tanto las políticas de la imagen como los datos con encriptación simétrica (usa la misma clave para cifrar y descifrar) mediante **AES (Advanced Encryption Standard)** [13]**.** Este algoritmo de encriptación sigue un esquema de cifrado por bloques. Esto quiere decir que el cifrado se realiza bloque a bloque. Inicialmente se divide el mensaje en bloques del mismo tamaño. Acto seguido, cada bloque se va convirtiendo en un bloque del mensaje de cifrado a través un conjunto de operaciones. Para comprender mejor como funciona este algoritmo vamos a detallarlo a continuación:

AES es el sistema de cifrado de clave simétrica que remplazó a DES. Admite claves de 128, 192 y 256 bits y bloques fijos de 128 bits. AES opera en una matriz de 4x4 bytes llamada ***state*** que se inicializa con los bytes del bloque**.** Consiste en un conjunto de rondas en las cuales se realizan unas operaciones definidas en el cuerpo finito GF(2^8) i en el espacio vectorial de dimensión 4 sobre este cuerpo. El número de rondas depende de la longitud en bits de la clave dividido por 32.
Si Nk es el número de bits de la clave y Nr es el número de rondas:

$$Nk = 128/32 \rightarrow Nr = 10$$
$$Nk = 192/32 \rightarrow Nr = 12$$
$$Nk = 256/32 \rightarrow Nr = 14$$

Las operaciones que realiza en el proceso de encriptación son las siguientes:

1. Expansión de la clave usando el esquema de claves de Rijndael. [14]
2. ***AddRoundKey*****:** Cada byte de la matriz ***state*** se combina con un byte de la subclave usando la operación XOR.
3. **Rondas (Nk-1 veces):**
   - 3.1. ***SubBytes***: en este paso se realiza una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo a una tabla de búsqueda.
   - 3.2. ***ShiftRows***: en este paso se realiza una transposición donde cada fila de la matriz ***state*** es rotada de manera cíclica un número determinado de veces hacia la izquierda.
   - 3.3. ***MixColumns***: operación de mezclado que opera en las columnas de la matriz ***state***, combinando los cuatro bytes en cada columna usando una transformación
   - 3.4. ***AddRoundKey***
4. **Ronda final:**
   - 4.1. ***SubBytes***
   - 4.2. ***ShiftRows***
   - 4.3. ***AddRoundKey***

El algoritmo de AES puede usarse con varios modos de operación. En nuestro caso, por simplicidad se ha usado **ECB *(Electronic Code-Book)*** [15]**.** ECB es el modo más sencillo, en el cual los mensajes se dividen en bloques y cada uno de ellos es cifrado por separado utilizando la misma clave K. La desventaja de este método es que a bloques de texto plano o claro idénticos les corresponden bloques idénticos de texto cifrado, de manera que se pueden reconocer estos patrones como guía para descubrir el texto en claro a partir del texto cifrado.

4.3. Punto de partida

Pese a que este proyecto utiliza las bases del proyecto del ex estudiante Alberto Duran [12], el punto de partida no se encuentra ahí, pues anteriormente, en el año 2013, Nicos Demetriou, estudiante de máster de la UPC, realizó su máster tesis titulada "*Metadata Interoperability with JPSearch*" [16] en la que, entre otros, trata la edición de metadatos en imágenes JPEG y crea una aplicación en lenguaje JAVA para poder editar metadatos llamada *JPSearch Editor.*

A partir del trabajo de investigación de Nicos y del proyecto desarrollado por Alberto, se propone este proyecto para rediseñar la aplicación, con el objetivo de continuar la investigación añadiendo una extensión que permita añadir privacidad, usando el lenguaje XACML, y cifrado para proteger las imágenes en su transporte. Para poder comprender mejor el resultado final de este proyecto, vamos a explicar la aplicación inicial propuesta por Nicos.

Esta aplicación proporciona una interfaz muy sencilla de utilizar que nos permite insertar, modificar o eliminar cualquier metadato, definido en el estándar *JPSearch* de las imágenes jpg. Existen más opciones dentro de la aplicación como permitir el etiquetado de distintas partes de la imagen especificada y facilitar la creación de reglas para adaptar otros tipos de metadatos como *EBUcore*, *DublinCore*, *MPEG-7*, etc. a metadatos que cumplan el esquema *JPSearch*. La aplicación consta de diferentes interfaces que detallaremos a continuación:

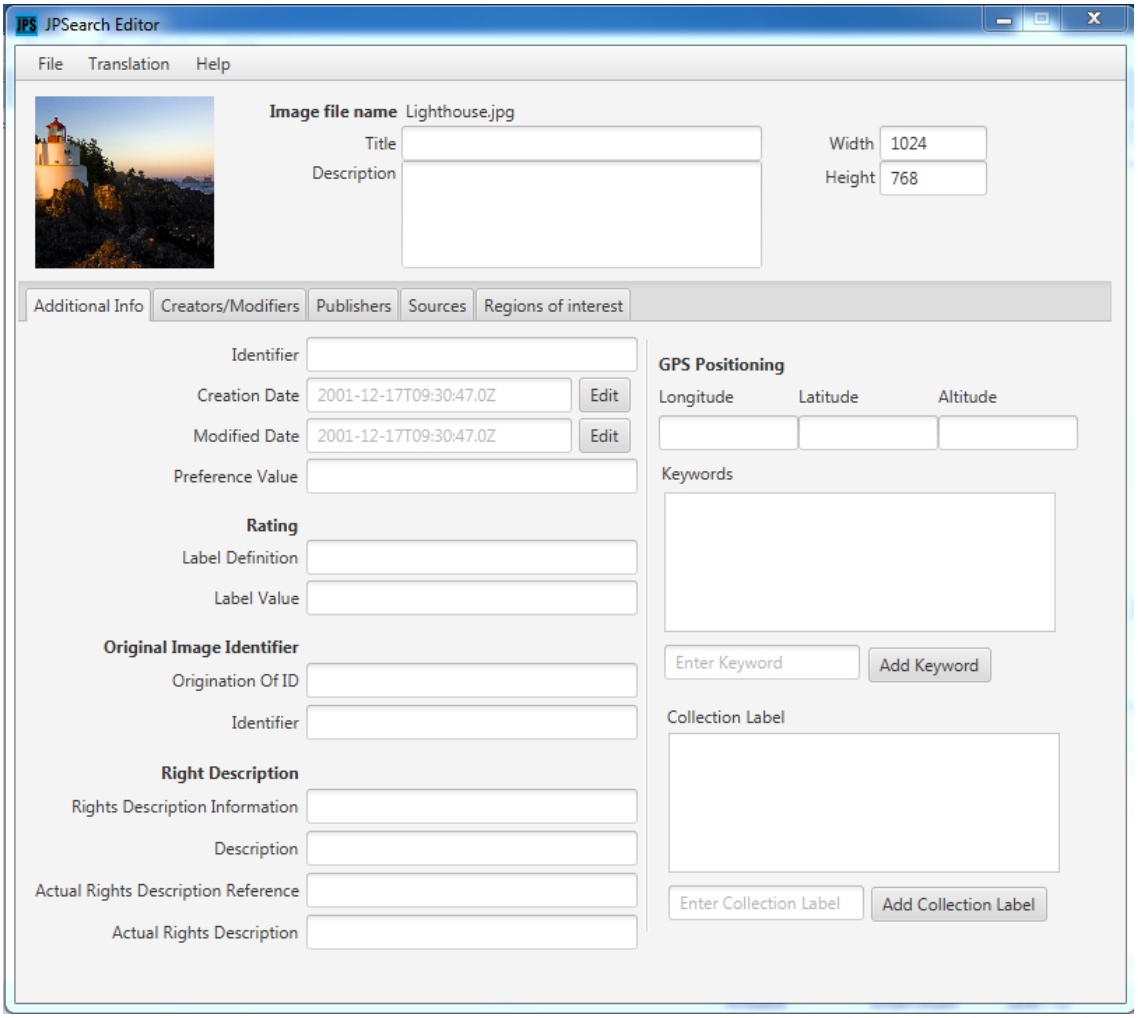


Figura 3: Diseño inicial de la pestaña Additional Info

La interfaz que se muestra en la Figura 3 permite editar la información de los metadatos básicos asociados a una imagen. Se puede modificar el identificador, la fecha de creación, la fecha de modificación, la longitud, la latitud, la altitud, las palabras clave o una colección de etiquetas entre otros. Ésta es la primera interfaz que nos mostrará el programa una vez abierto.

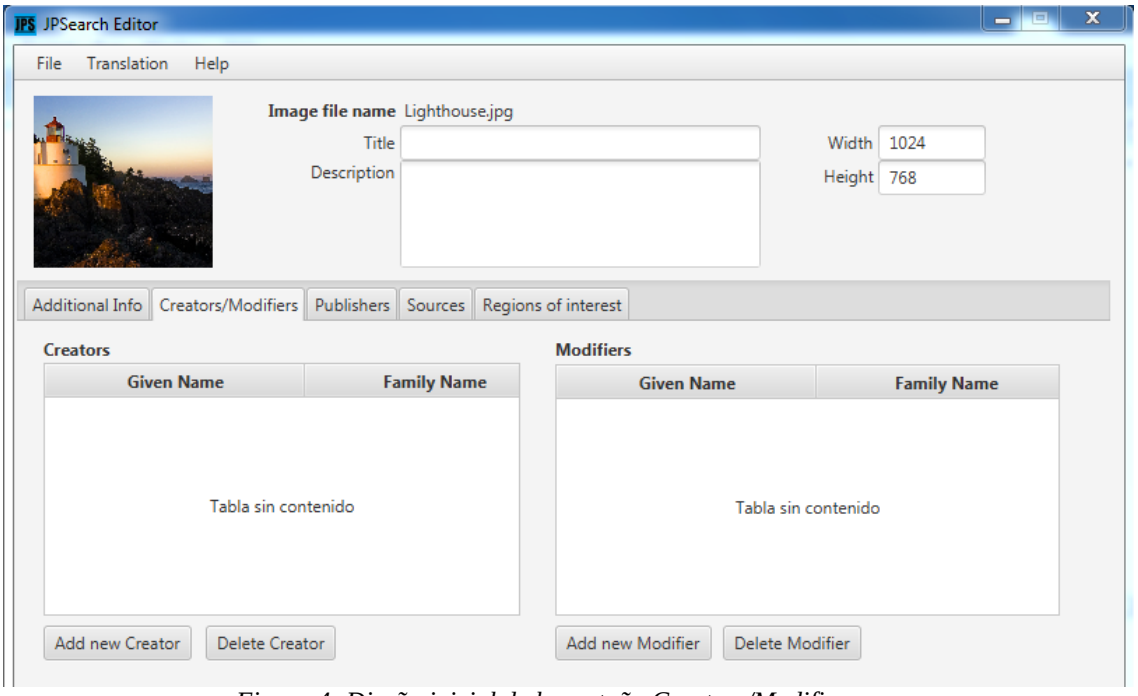


Figura 4: Diseño inicial de la pestaña Creators/Modifiers

En la Figura 4 se muestra la segunda pestaña de la aplicación. En este caso podemos editar el esquema de metadatos *PersonNameType*. Se permite la adición y eliminación de creadores y modificadores de la imagen y una pequeña información de perfil asociada.

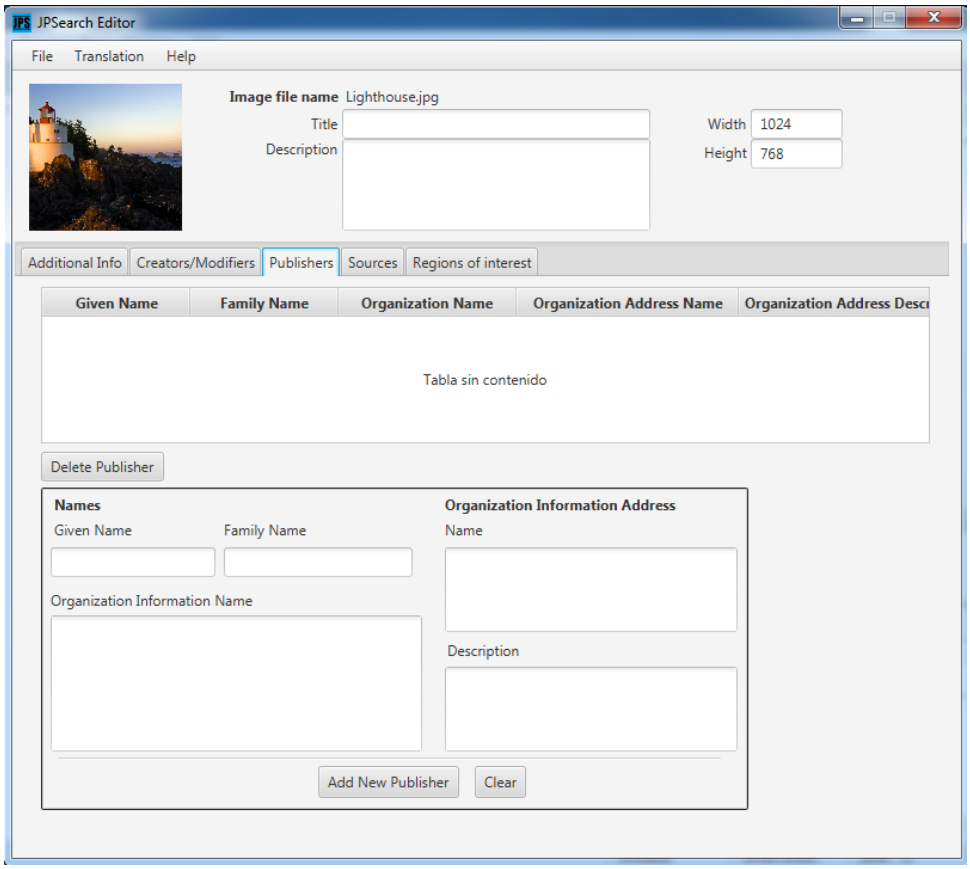


Figura 5: Diseño inicial de la pestaña Publishers

En la Figura 5 podemos observar cómo sería la pestaña *Publishers*, la cual nos permite editar el esquema *PublisherType*. Nos proporciona el formulario necesario para añadir un nuevo publicador a la imagen y un listado de los publicadores actuales; del que también podemos eliminar alguno en caso que sea necesario.

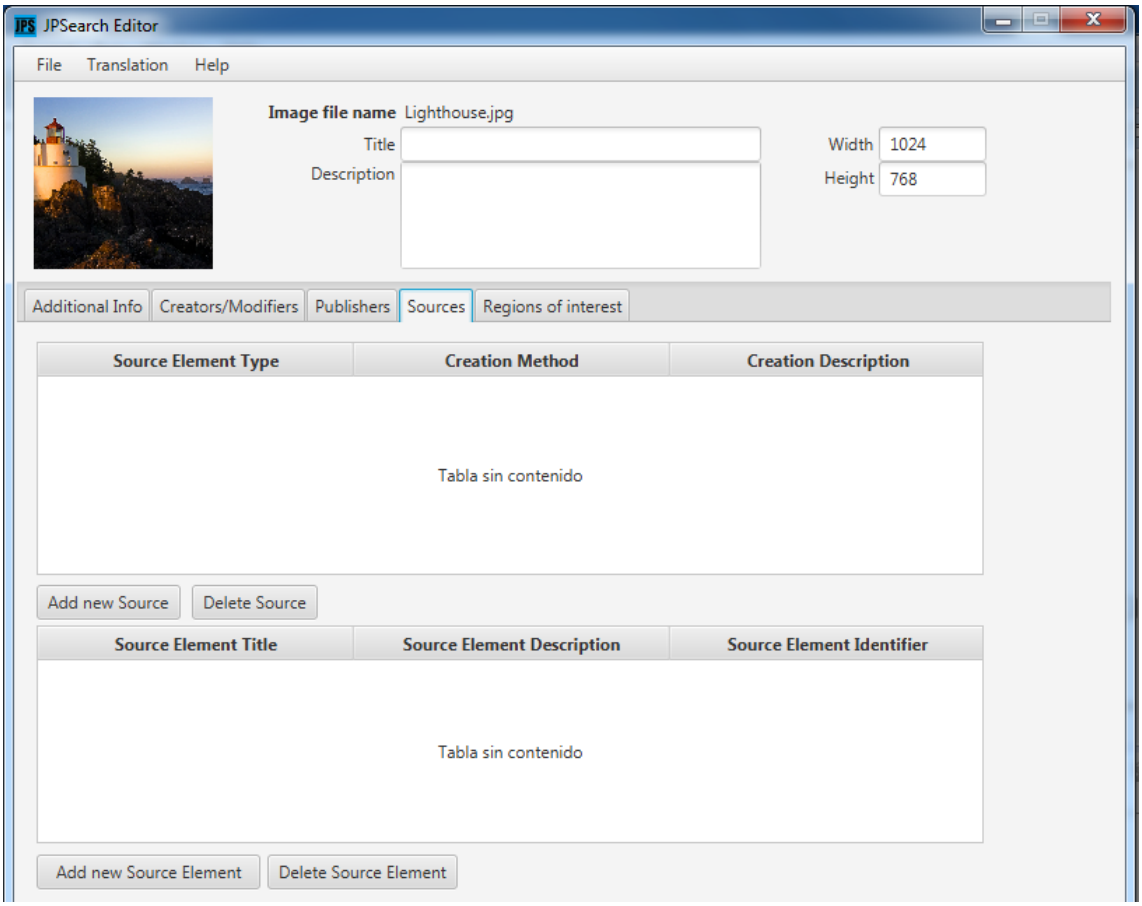


Figura 6: Diseño inicial de la pestaña Sources

En la Figura 6 podemos ver el diseño de la pestaña *Sources* que permite editar el esquema *SourceType*. Aquí podemos añadir una nueva fuente que indique el tipo de imagen que es, con qué método ha sido creada esta imagen y una descripción del dispositivo que ha creado la imagen. Una vez indicado el tipo de imagen podemos indicar el título de esta imagen, una descripción y un identificador. Por ejemplo:

<Source>
    <SourceElementType>**Óleo**</SourceElementType>
    <SourceElement>
        <SourceElementTitle>**El grito** <SourceElementTitle>
        <SourceElementDescription>
            Edvard Munch, 1893, Galería Nacional de Oslo, Noruega
            </SourceElementDescription>
    </SourceElement>
    <CreationMethod>**Óleo, temple**</CreationMethod>
    <CreationDescription>**pastel sobre cartón**</CreationDescription>
</Source>

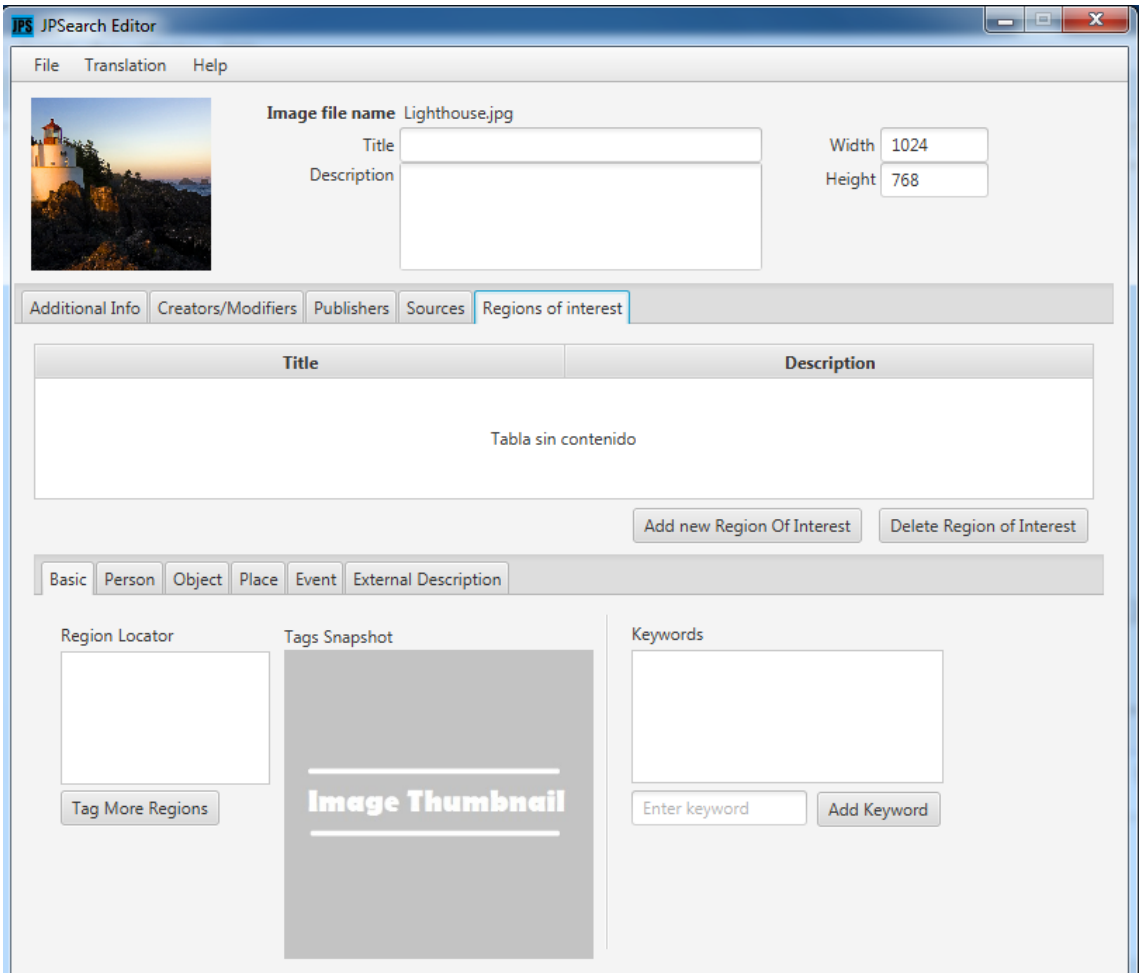


Figura 7: Diseño inicial de la pestaña Regions of Interest

En último lugar nos encontramos con la pestaña *Regions of Interest* que permite editar el esquema *JPSearchCoreType*. Se puede etiquetar partes de la imagen, marcando un trozo de imagen como región de interés, a la cual podemos asociar información como las personas que aparecen en esa parte de la imagen, los objetos, el lugar, el tipo de evento o añadir palabras clave entre otras opciones.

Alberto añadió una pestaña más a la aplicación llamada *Privacy* que permitía metadatos del tipo *Actual Rights Description Reference y Actual Rights Description*. En esta pestaña podemos añadir una referencia a una política de control de acceso, que consiste en la URL donde se encuentra dicha política, o directamente podemos poner la política en el campo *Actual Rights Description*. Cabe destacar que son excluyentes, no es posible poner ambos metadatos a la vez. Estas políticas pueden estar escritas tanto en XACML 2.0 como XACML 3.0. Podemos ver más en detalle el estilo de ésta pestaña a continuación, en la Figura 8.

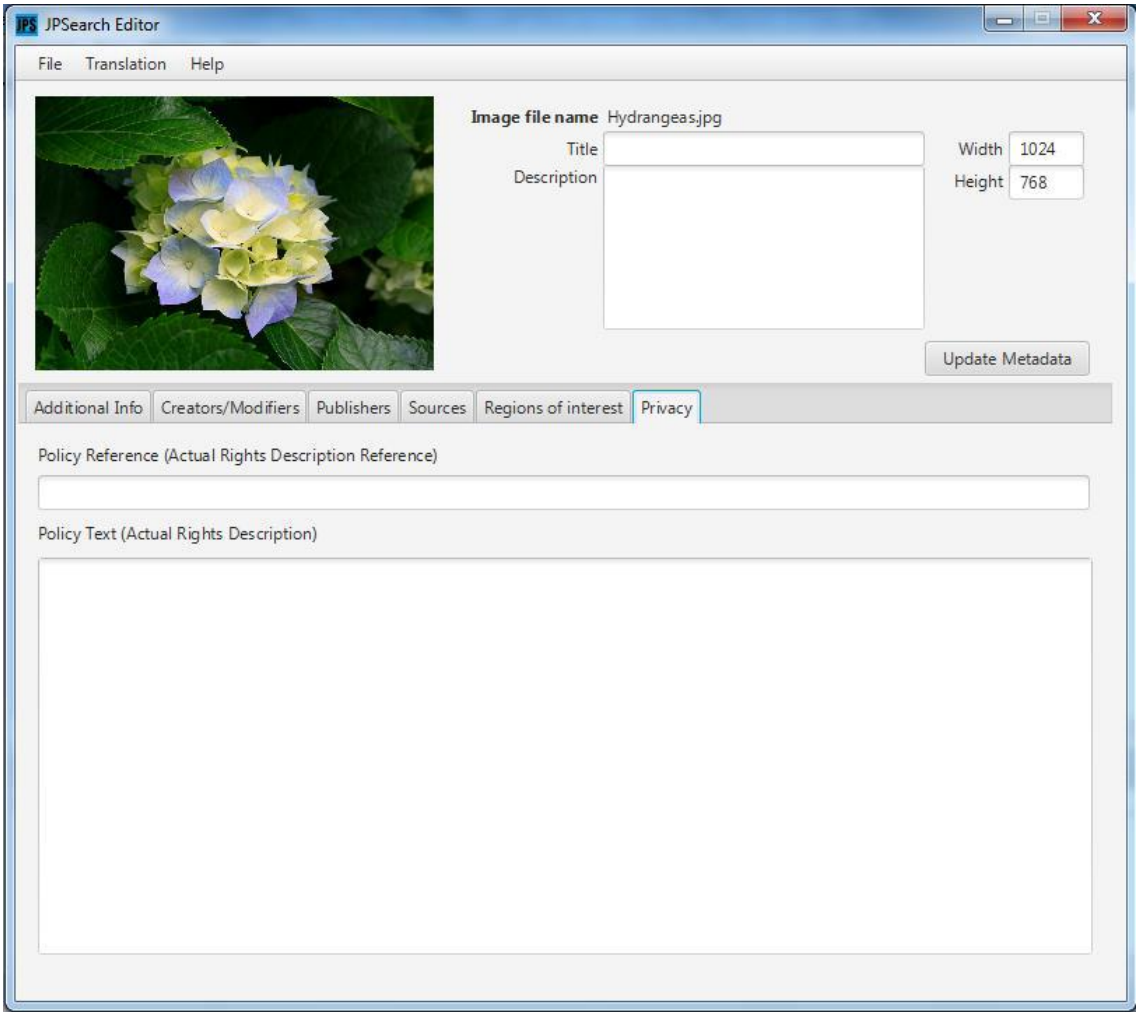


Figura 8: Diseño de la pestaña Privacy

Este proyecto toma este punto para modificar la propuesta de seguridad de Alberto y además cambiar la estructura de los datos para que se adecúen al artículo publicado por Silvia Llorente y Jaime Delgado [11].

5. Tecnologías utilizadas

5.1. *JPSearch*

El motivo de investigación de este proyecto es evaluar el efecto de incluir una política de privacidad en una imagen. De entre todos los formatos de imagen, propietarios o libres, se ha escogido *JPEG* por ser un estándar que ofrece interacción de forma estructurada a los metadatos de las imágenes mediante su *Schema*. Para ello, *JPEG* lanzó el estándar *JPSearch* que nació de la necesidad de búsqueda y recuperación de metadatos, que también se puede encontrar bajo la norma ISO/IEC 24800, con el fin de ayudar a los diferentes sistemas de gestión de imágenes a interoperar. Básicamente proporciona un *framework* abstracto, así como una arquitectura de búsqueda modular y flexible. Además de facilitar el uso y la reutilización de los metadatos.

En otras palabras, *JPSearch* es un conjunto de especificaciones que apoya el enriquecimiento de las imágenes *JPEG* o *JPEG* 2000 con el almacenamiento de metadatos en las imágenes. *JPSearch* consta de un esquema y de la construcción de ontologías, de un formato de consulta, de un formato de archivo para metadatos incrustados en los datos de imagen y de un formato de intercambio de datos para los repositorios de imágenes.

JPSearch es una especificación compuesta de seis partes. Cada una de ellas cubre diferentes aspectos del estándar, en total cooperación para completar la especificación:

1. Proporciona una visión global de *JPSearch*, explicando los componentes del *framework* y del sistema.

2. Es la base, proporcionando la estandarización de un formato independiente de la plataforma para la importación, exportación y el intercambio de ontologías. Además del registro de ontologías que podrían ser importados a un sistema compatible *JPSearch*, también incluye las funciones estándar básicas para manipular y consultar una o más ontologías en un repositorio.

3. Se centra en la funcionalidad de recuperación de imágenes entre los usuarios y los repositorios de imágenes. Más específicamente permite la expresión de criterios de búsqueda, la descripción del conjunto de resultados para la presentación preferida y finalmente define los procesos de gestión de consulta. El formato *JPSearch* de consulta (*JPQF*, *JPSearch Query Format*) es el lenguaje de consulta basado en XML que define la sintaxis de consultas intercambiada entre las aplicaciones cliente y los repositorios.

4. Especifica el formato de intercambio de metadatos de la imagen que están incrustados en *JPEG* y *JPEG-2000*.

5. Cubre el formato de intercambio de datos entre repositorios de imágenes, lo que permite una serie de funcionalidades para mejorar la portabilidad de los metadatos.

6. Proporciona el *software* de referencia que instancia la funcionalidad definida en las partes anteriores. Este software consta de cuatro *Java-basedmodules*: el módulo de traducción de metadatos que ingiere los metadatos XML traducidos en el *JPSearch Core Description* y viceversa, la validación sintáctica y semántica de una consulta de entrada *JPQF* y salida igual al módulo de códec de metadatos incrustados que incorpora la descripción XML en formato de archivo *JPSearch* y finalmente el módulo repositorio de importación y exportación que permite el intercambio de metadatos *JPSearch* entre repositorios de imágenes. La Figura 9 muestra el esquema de diagrama de las partes JPSearch.

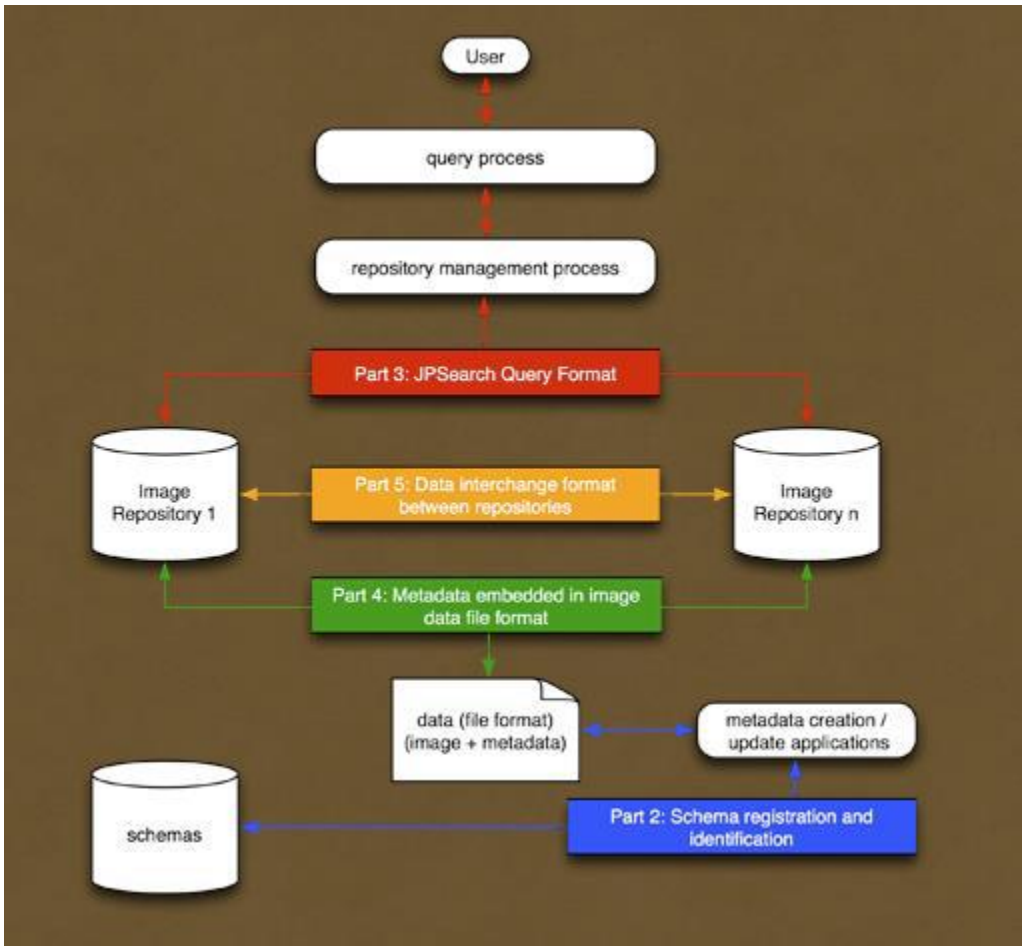


Figura 9: Esquema de la relación entre las partes de JPSearch [17]

Este proyecto se centra en la Parte 2 y en la Parte 4 de la especificación de JPSearch. La Parte 2 se utiliza para manipular los elementos definidos en los metadatos con el objetivo de encontrar la solución al problema de añadir privacidad a las imágenes.

El esquema *JPSearch Core* está compuesto por los siguientes metadatos:

Identifier	Title	Description
Width	Height	CreationDate
ModifiedDate	RightsDescription	GPSPositioning
Keyword	Source	RegionOfInterest
PreferenceValue	Rating	OriginalImageIdentifier
Creators	Modifiers	Publisher
CollenctionLabel		

Tabla 1: Elementos del esquema JPSearch Core

Para poder insertar en la imagen las políticas de control de acceso, se propone usar el elemento ***RightDescription*** *y* ***Keyword.*** El segundo elemento irá dentro del primero y nos servirá para guardar el valor de la segunda clave de cifrado que se comentará más en detalle en el siguiente apartado. La estructura que presenta el elemento *RightDescription* es la siguiente:

<RightsDescription>
<RightsDescriptionInformation>
Ubicación del estándar RightsDescription
</RightsDescriptionInformation>
<Description>
Descripción textual del estándar referenciado
</Description>
<ActualRightsDescriptionReference>
RightsDescription actual. (Política de privacidad actual). Puede estar integrada o referenciada.
</ActualRightsDescriptionReference>
<Keyword>
Value of the second Key, used to encrypt the image data
</Keyword>
</RightsDescription>

La parte 4 comentada anteriormente define cómo incluir los metadatos JPSearch en un fichero de imagen JPEG. Consiste en añadir un nuevo marcador llamado APP3 que se utiliza para contener todos los metadatos propios de *JPSearch.* En nuestro caso, los metadatos referentes a la política de control de acceso los pondremos en un marcador APP11.

Figura 10: *Estructura de la imagen con metadatos de JPSearch*

5.2. XACML

Anteriormente hemos identificado un mecanismo para insertar políticas de control de acceso en las imágenes *JPEG* mediante *JPSearch Editor*. Para construir dichas políticas es necesario un lenguaje que permita definir políticas de privacidad basadas en XML. Aquí aparece el lenguaje XACML.

Este estándar define un lenguaje declarativo de políticas de control de acceso implementado en XML y un modelo de procesamiento que describe cómo evaluar peticiones de acceso según unas reglas que previamente se han definido en políticas. La última versión, XACML 3.0, data de enero de 2013. La anterior versión, XACML 2.0, que también se emplea en este proyecto, data de febrero de 2005.

XACML es un sistema de control de acceso basado en atributos (*ABAC*, *Attribute Based Access Control*) [3], donde los atributos asociados con un sujeto o una acción o un recurso se utilizan para decidir si otorgar o denegar el acceso a un usuario a un recurso de un modo concreto. Está preparado para gestionar las decisiones de acceso tanto de forma local, como de forma remota. En general, se recomienda una separación entre el punto de uso (cliente) y el punto donde se decide el acceso (servidor) ya que de este modo es más fácil actualizar un criterio de decisión cuando en la política se producen cambios. Al separar el cliente del servidor, la autorización de políticas puede ser actualizada en tiempo real y afectar a todos los clientes con efecto inmediato. La arquitectura XACML se muestra a continuación:

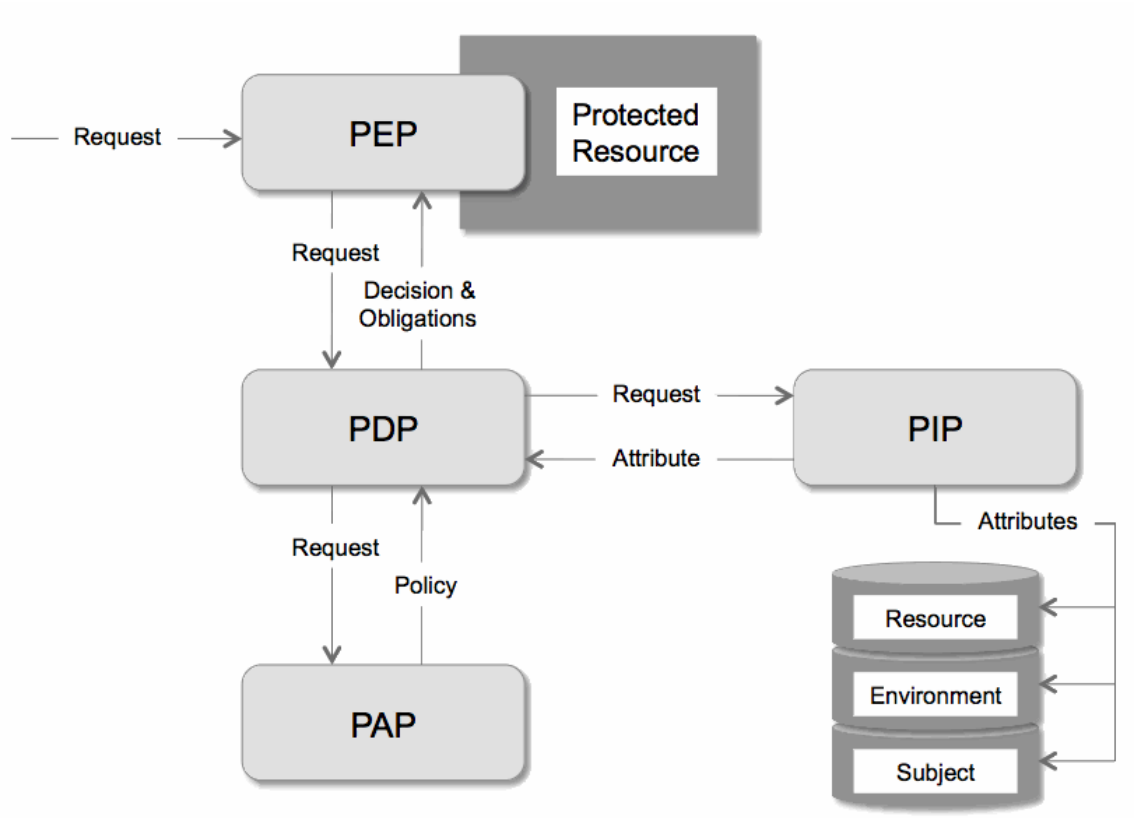


Figura 11: Componentes principales de la arquitectura XAMCL

- **PEP (*Policy Enforcement Point*):** Intercepta la petición de acceso y la deriva al PDP. Más adelante cuando éste recibe una respuesta del PDP, genera una respuesta específica para el sistema que hizo la solicitud de acceso.
- **PAP (*Policy Administration Point*):** Crea y administra las políticas de control. Envía al PDP la política que demanda.
- **PIP (*Policy Information Point*):** Envía al PDP información acerca de los recursos, el entorno y los sujetos que precisan información de los valores de determinados atributos. PIP se encarga de buscar estos valores y añadirlos a la solicitud. Por ejemplo, la fecha actual.
- **PDP (*Policy Decision Point*):** Evalúa las peticiones de acceso recibidas del PEP. Para ello solicita al PAP la política a la cual se quiere acceder y al PIP los valores de los atributos relacionados con el sujeto, el recurso y el entorno. Una vez obtiene todos estos datos toma una decisión y la envía al PEP.

XACML se estructura en 3 niveles de elementos:

- ***PolicySet***: puede contener cualquier número de elementos tipo *Policy* y *PolicySet*.
- ***Policy***: puede contener cualquier número de elementos de tipo *Rule*.
- ***Rule***: contiene la unidad básica de etiquetas que conforman una política de acceso.

Todos los elementos de tipo *PolicySets*, *Policies*, *Rules* y *Requests* utilizan los atributos *Subject*, *Resource*, *Environment* y *Action*.

- El elemento ***Subject*** representa a la entidad que realiza la petición de acceso. Un sujeto puede tener uno o más atributos.
- El elemento ***Resource*** representa los datos, el servicio o el sistema al que se quiere acceder. Un recurso puede tener uno o más atributos.
- El elemento ***Action*** define qué tipo de acceso se va a realizar sobre el recurso. Una acción puede tener uno o más atributos.
- El elemento ***Environment*** puede proporcionar información adicional de manera opcional. Por ejemplo, la fecha límite, intervalo de horas, intervalo de cantidades, etc.

Todos los elementos deben utilizar el atributo *Target* que indica el conjunto de condiciones (*Subject, Resource y Action*) que van a ir incluidos en la política de acceso y que aplican en caso de recibir peticiones de acceso.
El atributo *Condition* sólo puede ser usado por los elementos de tipo *Rule*. Este atributo puede utilizar una amplia gama de funciones para proporcionar al PDP la capacidad de comparación.

XACML define una serie de algoritmos llamados *Combining Algorithms* que ayudan a resolver conflictos en caso de haya más de un elemento de tipo *Rule* dentro de una política de acceso y que lo que indican estos *Rules* se contradigan. Estos algoritmos se identifican por los elementos *RuleCombiningAlgId* o *PolicyCombiningAlgId* y definen un procedimiento para llegar a una decisión de acceso dado los resultados individuales de evaluación de un conjunto de reglas o de políticas.

La directiva *Obligation* actúa desde el punto de decisión de políticas (PDP) hacia el punto de aplicación de políticas (PEP) e indica el comportamiento que se debe ejecutar antes o después de que se apruebe un acceso. Un acceso aprobado por el PDP puede llegar a ser realizado o no dependiendo de si el PEP es capaz de cumplir con lo establecido en la directiva. Un ejemplo de una obligación podría tener este aspecto:

Permitir acceso al recurso Noche Estrellada con atributo IDvisitante=x
if Subject match VisitanteHaPagadoEntrada
and action is view
with obligation
on Permit: do_Authorized_NocheEstrellada(*IDvisitante*, Subject, time)
on Deny: do_Unauthorized_NocheEstrellada(*IDvisitante*, Subject, time)

En los Anexos 7, 8, 9 y 10 podemos encontrar ejemplos de políticas de acceso para las versiones XACML 2.0 y XACML 3.0.

5.2.1. Diferencias entre XACML 2.0 y XACML 3.0

Las principales diferencias [18] que se aprecian entre las dos versiones son:

- Obligaciones en reglas: En XACML 2.0, las obligaciones sólo pueden añadirse a los elementos tipo *Policy* y *PolicySet*. En XACML 3.0, los elementos tipo *Rule* también pueden contener obligaciones.
- El elemento *Content*: en una solicitud XACML 2.0, sólo puede haber contenido XML dentro de la categoría de recurso como parte del elemento *ResourceContent*. En XACML 3.0, el elemento *ResourceContent* se generaliza en un elemento de contenido que se puede encontrar en cualquier categoría.
- Alcance de expresiones *XPath*: En XACML 2.0, expresiones *XPath* se aplica a la raíz de la solicitud XACML. En XACML 3.0, expresiones *XPath* se aplican a la raíz del elemento de Content.
- El elemento *Target*: En XACML 3.0 se permite realizar expresiones del tipo: “el sujeto X puede visualizar el recurso R o el sujeto S puede editar el recurso R” dentro de un mismo elemento *Target*. Mientras que en XACML 2.0 el máximo nivel de especificación de un *Target* era: “el sujeto X o el sujeto S pueden visualizar o editar el recurso R”. XACML 3.0 elimina la función disyuntiva y conjuntiva de los elementos *Resources* de XACML 2.0 e introduce los elementos *AnyOf* y *AllOf*.

5.3. JAVA

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling, de Sun Microsystems (la cual fue adquirida por la compañía Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a *bytecode* (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

5.4. XML

El lenguaje XML es el lenguaje en el que se basa XACML y que se utiliza para la generación tanto de las políticas de privacidad como de las peticiones de acceso. XML, siglas en inglés de *eXtensible Markup Language*, traducido como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

5.5. AES

Para poder cifrar tanto las políticas como los datos de la imagen, se ha optado por usar AES con claves de 128 bits. Se ha determinado usar AES por su sencillez a la hora de trabajar con las librerías que proporciona JAVA y porque actualmente se considera un algoritmo seguro. Pese a que inicialmente se pensó en la idea de usar RSA, se descartó ya que una imagen puede tener varios destinatarios en sus políticas de acceso. En estos casos, usar criptografía de clave pública lo hacía poco viable.

6. Desarrollo del proyecto

6.1. Arquitectura

El proyecto propone dos aplicaciones relacionadas entre sí, ambas necesarias para implementar el flujo que siguen las imágenes. La primera aplicación, ***JPSearch Editor*** [16] será la que se encargue de la fase de aplicar los metadatos a la imagen, permitir la inclusión de políticas de privacidad y por último, generar todo el proceso de cifrado. La segunda aplicación, ***XACML Request Generator*** está pensada para el usuario que decida acceder a un recurso, en nuestro caso, una imagen cifrada. Sus funciones serán las de comprobar los datos del usuario creando una petición en XACML y descifrando el contenido de la imagen.

La aplicación de ***JPSearch Editor*** con la que continuamos este proyecto está escrita en JAVA y contiene el siguiente grupo de paquetes:

- **exif**: Contiene las clases necesarias para almacenar los metadatos de tipo exif.
- **gui**: Contiene las clases relacionadas con la interfaz de usuario y la definición de cada pantalla.
- **jpsearch**: Contiene las clases que definen, almacenan e interactúan con los metadatos JPEG.
- **jpsearchcore**: Contiene una clase por cada uno de los tipos de metadatos definidos en el estándar JPSearch.
- **translation**: Contiene las clases necesarias para la traducción de metadatos.
- **criptography**: Contiene las clases necesarias para la encriptación de las reglas y de los datos de la imagen.

El desarrollo de esta aplicación se ha basado en rehacer el paquete ***criptography*** que había anteriormente para que se adapten a la nueva estructura propuesta y se ha modificado algunas de las clases que aparecen en ***gui, jpsearch y jpsearchcore.*** En concreto, se ha modificado el tratamiento de la inserción y actualización de metadatos, y la inserción de políticas en la zona marcada por APP11.

Por otra parte, la aplicación ***XACML Request Generator*** contiene la misma estructura de paquetes, pero, almacena la lógica principal en el paquete por defecto (***default package***). Se ha modificado todo el proceso de descifrado de la aplicación, implementado la extracción de políticas en una imagen y la creación de un repositorio de claves público para todos los usuarios.

En los Anexos 1, 2, 3, 4 se muestran los diagramas de paquetes y clases de este proyecto.

6.2. Conexión entre aplicaciones

La comunicación entre ambas aplicaciones es fundamental para el funcionamiento adecuado del proyecto. Esta comunicación representa el flujo que sigue la imagen desde que se introducen metadatos y política hasta que se visualiza:

1) En primer lugar, es necesario facilitar la aplicación ***JPSearch Editor*** al usuario remitente, que le permitirá editar los metadatos e incluir una posible política de privacidad en lenguaje XACML a la imagen.
2) El segundo paso es guardar los metadatos y la política en la imagen. En el momento en que el usuario haya insertado los datos, y acceda al botón ***Update Metadata*** todos los datos se guardaran en la imagen y ésta se cifrará. Primero se cifrará la política con una clave. Esta clave será la que se comparta al usuario receptor.
3) Seguidamente se cifrará los datos de la imagen con una segunda clave. Esta segunda clave no será accesible por ningún usuario y quedará guardada en el campo *Keyword* dentro del esquema de *RightDescription*. Se le informará al usuario que el cifrado se ha realizado con éxito y se le facilitará el nombre del fichero que contiene la clave a compartir. Esta clave se guardará en el repositorio de claves público al cual podrá acceder el receptor en el proceso de descifrado. Se creará el fichero *encrypted*.jpg en la misma ruta que la imagen original. Ésta imagen es la que el usuario emisor compartirá con sus receptores. Podemos ver un ejemplo del mensaje en la Figura 12:

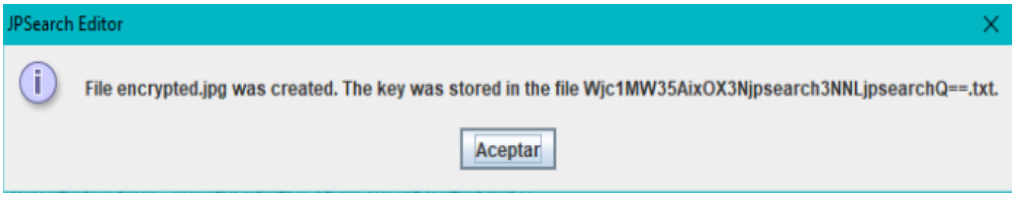


Figura 12: Fichero cifrado con éxito

4) Se facilita la aplicación ***XACML Request Generator*** a los usuarios receptores. Esta aplicación será capaz de generar una petición de acceso a un recurso.
5) El usuario receptor inicia sesión en la aplicación y rellena el formulario necesario para la creación de la petición de acceso. Los campos necesarios para el formulario son: recurso y clave de descifrado. Los campos opcionales son: rol del usuario y país.
6) El usuario envía la petición de acceso y la aplicación descifra la política con la clave que indicaba el receptor, la cual se ha usado en el formulario anterior. Se obtiene la segunda clave necesaria para descifrar los datos. Se comprueba que la petición es válida según las políticas de la imagen gracias al autorizador Balana y se revisa la respuesta que nos ofrece
7) Se descifra los datos de la imagen con la segunda clave.
8) Finalmente, si la respuesta es *Permit* se informa al usuario de que el fichero *decrypted.jpg* se ha creado en el Escritorio correctamente. En cualquier otro caso, se informa de que no ha sido posible ya que su petición no se aplica correctamente según las políticas y, por tanto, no se le devuelve el fichero descifrado.

Podemos ver un esquema más visual en la Figura 13:

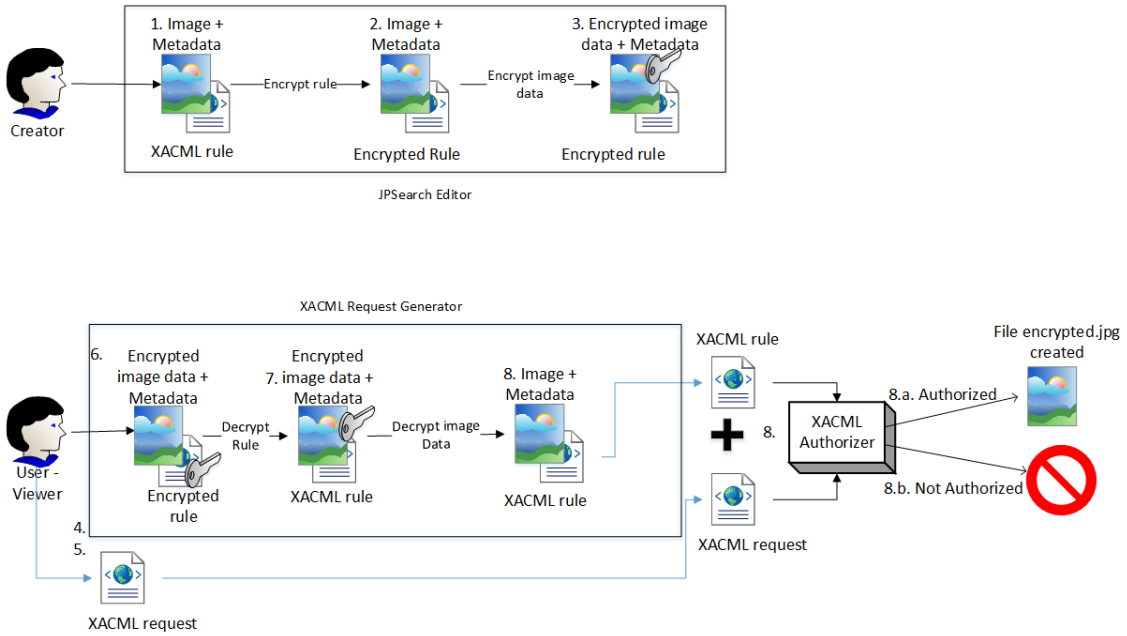


Figura 13: Esquema general del flujo de una imagen entre aplicaciones

6.3. JPSearch Editor

Tal y se comenta en el apartado punto de partida, se han realizado diversas modificaciones a la aplicación ***JPSearch Editor*** para que se adapte a las nuevas necesidades. Los cambios que se han realizado son:

- Incorporar el sector de metadatos APP11 que contiene los metadatos de privacidad. Concretamente se han situado por encima de los metadatos propios de *JPSearch* del sector APP3.
- Se ha modificado el sistema de cifrado anterior para que primero cifre la política de privacidad y luego los datos de la imagen. Cabe destacar que el segundo cifrado no consiste en cifrar al completo la imagen, sino en cifrar los datos posteriores a los metadatos.
- La estructura de *Right Description* ahora contiene el elemento *Keyword* donde se almacena la segunda clave necesaria para cifrar los datos de la imagen.
- Se ha añadido toda la lógica necesaria para tratar recoger las políticas e insertarlas en la imagen con la nueva estructura propuesta.
- Mensajes descriptivos para informar al usuario en caso de que algo no funcione correctamente.
- Control de errores que provocaban el mal funcionamiento de la aplicación.

6.3.1. Implementación

Pese a haber modificado muchas clases, las funciones principales y que más importancia han tenido a la hora de desarrollar este proyecto han sido las siguientes:

- Modificar la manera de abrir imágenes para controlar si una imagen es válida o no.
- Crear un nuevo sistema para insertar, detectar y obtener metadatos de *RightDescription* en la sección APP11.
- Nuevo sistema de cifrado basado en AES con claves aleatorias de 128 bits, cifrando los datos de la imagen y las políticas por separado.

En primer lugar, tenemos a *handleOpen();* Esta función se ejecuta al iniciar la aplicación cuando se quiere cargar una imagen o al abrir una imagen diferente.

```
3103    /**
3104     * Opens a FileChooser to let the user select an address book to load.
3105     * @throws Exception
3106     */
3107    @FXML
3108    private void handleOpen() throws Exception {
3109        if(jpsC != null && !saved){
3110            int n = JOptionPane.showConfirmDialog(null, "You have some changes that have not been saved. " +
3111                    "Are you sure you want to load a new photo?", "JPSearch Editor", JOptionPane.YES_NO_OPTION);
3112            if(n == JOptionPane.NO_OPTION) {
3113                //
3114                return;
3115            }
3116        }
3117        FileChooser fileChooser = new FileChooser();
3118        FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("JPEG files (*.jpg)", "*.jpg");
3119        fileChooser.getExtensionFilters().add(extFilter);
3120
3121        if(inputImageFile != null){
3122            File existDirectory = inputImageFile.getParentFile();
3123            fileChooser.setInitialDirectory(existDirectory);
3124        }
3125
3126        //Show open file dialog
3127        inputImageFile = fileChooser.showOpenDialog(null);
3128        if(inputImageFile != null){
3129            //TODO --> reconocer si el fichero entrante ya esta cifrado
3130            Path p = inputImageFile.toPath();
3131            byte[] data = Files.readAllBytes(p);
3132            final StringBuilder builder = new StringBuilder();
3133            for(byte b : data) {
3134                builder.append(String.format("%02x", b));
3135            }
3136            String ss = builder.toString();
3137            String sub = ss.substring(ss.length()-4, ss.length());
3138            if (!sub.equals("ffd9")) { //is not a valid JPG file or it is already encrypted
3139                JOptionPane.showMessageDialog(null, "File invalid. This file may already be encrypted.", "Warning", JOptionPane.WARNING_MESSAGE
3140                updatebutton.setDisable(true);
3141
3142            }
3143            else {
3144                updatebutton.setDisable(false);
3145                clearAllFields();
3146                lblFileName.setText(inputImageFile.getName());
3147                String s = inputImageFile.toURI().toString();
3148                actualImage = new Image(s);
3149                image = new JPSearchParser(inputImageFile.getAbsolutePath());
3150                if(image.getJpsearchData() == null){
3151                    int n = JOptionPane.showConfirmDialog(null, inputImageFile.getName() +
3152                            " does not contain JPSearch metadata! Will you add new metadata? If you press YES, the image file will be modified.'
3153                            "Warning", JOptionPane.YES_NO_OPTION);
3154                    jpsC = new JPSearchCore();
3155                    imgNumDnaI.setImage(actualImage);
3156                    readWidthHeight();
3157                    menuItemUpdate.setText("Add JPS Metadata");
3158                    menuItemUpdate.setDisable(false);
3159                    menuItemSaveXMLToFile.setDisable(false);
3160                    menuShowXML.setDisable(false);
3161                    menuItemSaveToDB.setDisable(false);
3162                    setDisableInterface(false);
3163                    menuItemOpenXMLFile.setDisable(false);
3164                    saved = true;
3165                    if(n == JOptionPane.YES_OPTION){
3166                        updatebutton.setDisable(false);
3167                        handleUpdate();
3168                    }
3169                    else updatebutton.setDisable(true);
3170                    return;
3171                }
3172                //Image contains JPSearch metadata
3173                try{
3174                    jpsC = JPSearchMetadataHandler.unmarshal(image.getXmlMetadata());
3175                }
3176                catch(Exception e){
3177                    JOptionPane.showMessageDialog(null, "Could not load image metadata. Maybe something went wrong from the previous save?", "JI
3178                    setDisableInterface(true);
3179                    return;
3180                }
3181                currentXML = image.getXmlMetadata();
3182                showMetadata();
3183                JOptionPane.showMessageDialog(null, inputImageFile.getName() + " was loaded successfully!", "JPSearch Editor", JOptionPane.INFO
3184                setDisableInterface(false);
3185                saved = true;
3186            }
3187        }
3188    }
3189
```

Figura 14: Función handleOpen();

Primero se mira que la imagen cargada tenga cambios sin guardar y se informa al usuario en caso afirmativo. Seguidamente, comprueba si la imagen cargada no es válida o ya está cifrada. En ese caso se avisa al usuario de que no

ha seleccionado una imagen válida. Una imagen *JPEG* se caracteriza por siempre finalizar con el marco EOI determinado por los bytes 0xFF, D9. Una vez cargada la imagen, se comprueba si tiene ya metadatos de *JPSearch*. En casi afirmativo se cargan y se rellenan los campos correspondientes. En otro caso, se le informa al usuario de que esa imagen no contiene metadatos y se le ofrece la posibilidad de añadirlos. Si el usuario contesta que no, el botón de *Update Metadata* quedará deshabilitado. Finalmente, se informa al usuario si ha habido algún error o por lo contrario, se ha cargado correctamente la imagen.

En segundo lugar tenemos la función *handleUpdate()*. Esta función se ejecuta cuando el usuario desea salvar los cambios realizados en la imagen.

```
3196    /**
3197     * Updates Metadata of the file
3198     * @throws Exception
3199     */
3200    @FXML
3201    private void handleUpdate() throws Exception{
3202        if(inputImageFile != null) {
3203            if (!StringUtils.isBlank(txtRDardr.getText()) && !StringUtils.isBlank(txaRDard.getText())) {
3204                JOptionPane.showMessageDialog(null, "Only 'Policy Reference' or 'Policy Text' must be filled. Please choose or
3205            }
3206            else {
3207                prepareIdentifier();
3208                prepareCreatorsandModifiers();
3209                preparePublisher();
3210                prepareDescription();
3211                prepareDates();
3212                prepareTitle();
3213                preparePreferredValue();
3214                prepareRating();
3215                prepareOrigImageIdentifier();
3216                prepareGPSPositioning();
3217                prepareSource();
3218                prepareRegionOfInterest();
3219                prepareWidthHeight();
3220                prepareRightDescription();
3221                writeXML2File();
3222                saved = true;
3223            }
3224        }
3225    }
3226
```

Figura 15: Función handleUpdate();

Primero comprueba que los campos *ActualRightsDescriptionReference* y *ActualRightsDescription* no estén rellenados a la vez. En ese caso se informa al usuario que solo puede rellenar uno de ellos. A continuación se actualizan todos los metadatos de la imagen. Estas llamadas actúan como *wrappers* para facilitar la comunicación entre los metadatos en alto y bajo nivel. Finalmente se llama a la función *writeXML2File();* encargada de guardar los metadatos en la imagen y llamar a las funciones necesarias para generar la lógica de la inserción y cifrado de las políticas y los datos. En concreto, entre otras acciones, llamará a la función setRD(); encargada de insertar los metadatos de privacidad en la imagen y cifrar toda la información tanto de la política como de los datos.

```
3035    private void setRD(String fileName, String RdXML, boolean crypto) throws IOException, NoSuchAlgorithmException, NoSuchPaddingException, Inva
3036        Path path = Paths.get(image.getJPSearchFile().getAbsolutePath());
3037        byte[] bytes = Files.readAllBytes(path);
3038        String clearData = StringEscapeUtils.unescapeXml(RdXML);
3039        byte[] result;
3040        if(crypto) { //update the metadata
3041            //Here we must decide if we want to encrypt also the app3 metadata. Actually this program does not do it.
3042            byte[] encryptedRD = EncryptDecryptString.encryptRD(clearData, key); //RightDescriptions encrypted
3043            try{
3044                byte[] encryptedData = EncryptFile.encryptData(bytes, key2); //image data encrypted
3045
3046
3047            StringBuilder builder = new StringBuilder();
3048            for(byte b : encryptedData) {
3049                builder.append(String.format("%02x", b));
3050            }
3051            String im = builder.toString();
3052            File res = new File (image.getJPSearchFile().getParent() + "/encrypted.jpg");
3053            result = new JPSearchRewriter().updateAPP11Segment(encryptedRD, bytes, im, true);
3054            foss = new FileOutputStream(res);
3055            foss.write(result);
3056            String s = Base64.getEncoder().encodeToString(key.getEncoded());
3057            String aux = s;
3058            aux = aux.replace("/","jpsearch");
3059            fkey = new File("Keys\\" + aux + ".txt");
3060            foss = new FileOutputStream(fkey);
3061            foss.write(s.getBytes("utf-8"));
3062
3063
3064            JOptionPane.showMessageDialog(null, "File encrypted.jpg was created. The key was stored in "
3065                    + "the file " + aux + ".txt.", "JPSearch Editor", JOptionPane.INFORMATION_MESSAGE);
3066            }
3067            catch(Exception e){
3068                JOptionPane.showMessageDialog(null, "The file couldn't be encrypted. Check your data and try again.", "JPSearch Editor", JOptio
3069            }
3070        }
3071        //image loaded as the first time (without app3,app11 metadata)
3072        StringBuilder builder = new StringBuilder();
3073        for(byte b : bytes) {
3074            builder.append(String.format("%02x", b));
3075        }
3076        String s = builder.toString();
3077        String sec = s.split("ffe3",2)[1];
3078        String app3 = sec.split("ffdb",2)[0]; //app3 data
3079        File res = new File(image.getJPSearchFile().getAbsolutePath());
3080        //insert the app11 and app3 data to the image
3081        result = new JPSearchRewriter().updateAPP11Segment(clearData.getBytes("utf-8"), bytes, app3, false);
3082        foss = new FileOutputStream(res);
3083        foss.write(result); //the image is updated with the information of the metadata and RightDescriptions.
3084
3085    }
```

Figura 16: Función setRD();

Esta función inicialmente obtiene los datos de la imagen a ser tratada. En caso de que el tercer parámetro sea cierto, significa que el usuario quiere actualizar los datos de la imagen y, por tanto, es necesario incorporar la política de privacidad y los datos de la imagen cifrados. Es aquí donde podremos crear el fichero *encrypted.jpg*. Cabe destacar que antes de cifrar los datos es necesario hacer un cambio de marcos, para aplicar la estructura de la imagen propuesta. Este cambio consiste en cambiar los marcos SOS en marcos APP11. Si el cifrado y el fichero *encrypted.jpg* se han realizado correctamente se informará al usuario facilitándole el fichero que se encuentra en el repositorio que contiene la clave utilizada para descifrar las políticas. En caso de error, también se informará al usuario. Por otro lado, si el tercer parámetro es falso, querrá decir que la imagen se carga por primera vez y que no contiene metadatos ni políticas. En este caso se inserta la estructura de los datos *RightDescription* sin contenido dentro de sus elementos.

En tercer y penúltimo lugar encontramos la función updateAPP11Segment(); que contiene toda la lógica necesaria para insertar en el sitio adecuado los metadatos referentes a la privacidad y que por tanto aparecen en una sección diferente de los metadatos de *JPSearch*. Esta función contiene 4 parámetros (de izquierda a derecha): datos en bytes de la política de privacidad, datos en bytes de la imagen completa, datos en formato hexadecimal de los datos de la imagen (no contiene los datos de los metadatos APP11) y si la imagen va a ser cifrada o no. El resultado es el conjunto de bytes que contiene la imagen inicial una vez se ha añadido los metadatos APP3 y APP11.

```
192     /*                                  app11         all       data image*/
193     public byte[] updateAPP11Segment(byte[] bdata, byte[] bytes, String picture, boolean crypto) throws IOException
194     {
195         String marker = "ffeb";
196         StringBuilder sb = new StringBuilder();
197         sb.append(Integer.toHexString(bdata.length + 2)); //data + dimension
198         while (sb.length() < 4) {
199             sb.insert(0, '0'); // pad with leading zero if needed
200         }
201         String length = sb.toString();
202          //marker + len + newdata (RightDescription)
203         StringBuilder builder = new StringBuilder();
204         for(byte b : bdata) {
205             builder.append(String.format("%02x", b));
206         }
207         String data_xml = builder.toString();
208         String res = marker + length + data_xml; //app11 constcuido
209 
210         builder = new StringBuilder();
211         for(byte b : bytes) {
212             builder.append(String.format("%02x", b));
213         }
214         String s = builder.toString(); // data from initial file with jpsearch metadata
215         String sec = s.split("ffdb",2)[1];  //after  ffdb
216         String prim = s.split("ffe3",2)[0]; //before  ffe3
217 
218         String resultat;
219         if(!crypto)
220             resultat = prim + res + "ffe3" + picture + "ffdb" + sec;
221         else {
222             String aux = s.split("ffe3",2)[1];
223             aux = aux.split("ffdb",2)[0];
224             resultat = prim + res + "ffe3" + aux + "ffdb" + picture;
225         }
226 
227         byte[] imageBytes = DatatypeConverter.parseHexBinary( resultat );
228         return imageBytes;
229     }
```

Figura 17: Función updateAPP11Segment();

Finalmente, en cuarto lugar, encontramos la lógica necesaria para cifrar la política, *encryptRD()*, y los datos, *encryptData()*. Antes de cifrar los datos es necesario cambiar los marcos 0xFFDA pertenecientes al SOS por 0xFFEB perteneciente a APP11. Como se ha comentado anteriormente en el apartado de algoritmos criptográficos, se ha usado para ambos casos criptografía de clave privada o simétrica usando AES con claves de 128 bits y modo de operación ECB.

```
19     public static SecretKey getAESKey() throws NoSuchAlgorithmException{
20           KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
21           keyGenerator.init(128);
22           SecretKey key = keyGenerator.generateKey();
23           return key;
24     }
25 
26     public static byte[] encryptRD(String data, SecretKey key) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, Ille
27         Cipher aes = Cipher.getInstance("AES/ECB/PKCS5Padding");
28         aes.init(Cipher.ENCRYPT_MODE, key);
29         byte[] datab = data.getBytes("utf-8");
30         return aes.doFinal(datab);
31     }
32 
```

Figura 18: Función encryptRD();

```
14     //cifra datos imagen
15     public static byte[] encryptData(byte[] data, Key key) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalB
16         final StringBuilder builder = new StringBuilder();
17         for(byte b : data) {
18             builder.append(String.format("%02x", b));
19         }
20         String s = builder.toString();
21         //We replace SOS segments to APP11 segments
22         String d = s.split("ffdb",2)[1];
23         d = d.replace("ffda", "ffeb");
24 
25         byte[] datab = d.getBytes("utf-8");
26         Cipher aes = Cipher.getInstance("AES/ECB/PKCS5Padding");
27         aes.init(Cipher.ENCRYPT_MODE, key);
28         return aes.doFinal(datab);
29     }
30 
```

Figura 19: Función encryptData();

6.4. XACML Request Generator

Esta aplicación tiene como objetivo generar una política de acceso según los parámetros indicados por el usuario, autorizar al usuario a obtener acceso al recurso que se demande y descifrar los datos de la política y de los datos de la imagen.

Primero se muestra un formulario muy simple de inicio de sesión en el que se reclama un usuario y una contraseña. Los datos de usuarios y contraseñas se guardan en un repositorio en claro llamado *users.db*. Es importante remarcar que este repositorio no es seguro ya que un usuario puede abrirlo y ver su contenido. En la Figura 20 se puede ver cómo sería el formulario de inicio:

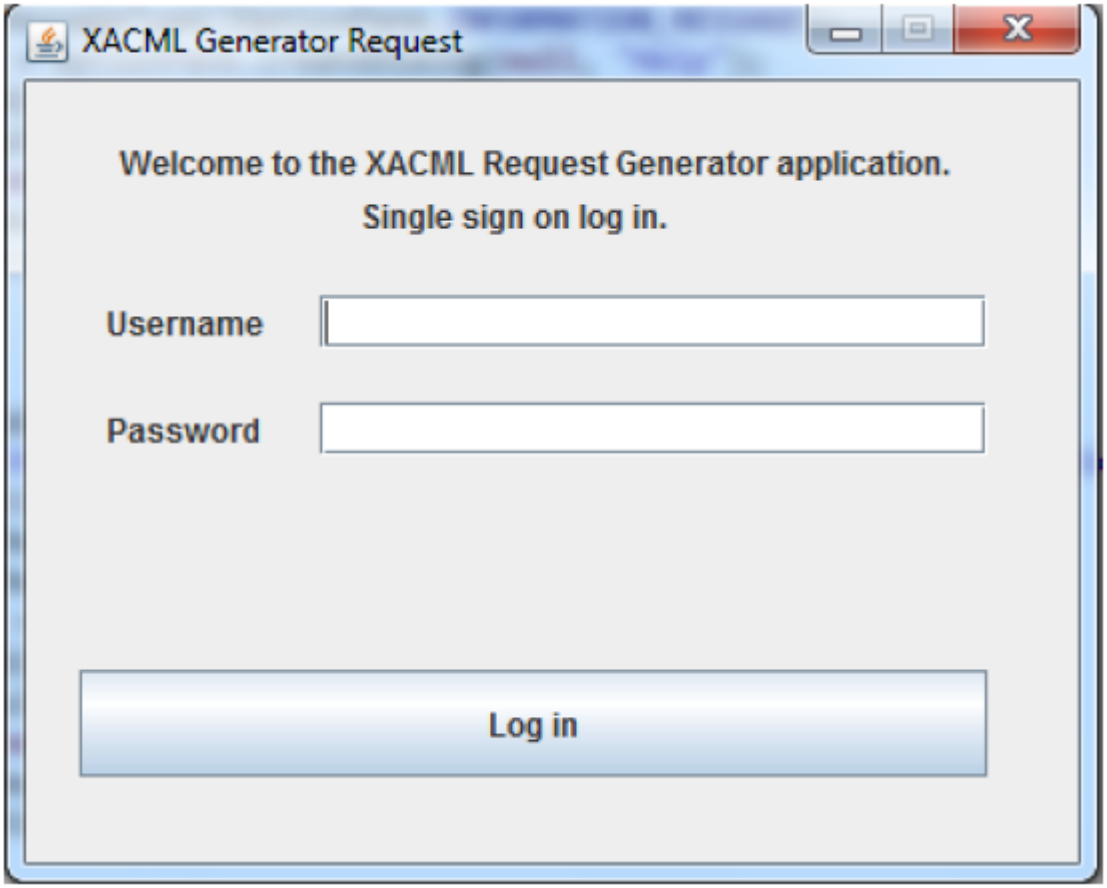


Figura 20: Formulario de inicio de sesión

Para poder ejecutar la aplicación durante la fase de desarrollo, se utilizaron algunos ejemplos de usuario y contraseña como brian:alumne o Silvia:67890.

Este proceso nos permitirá identificar al usuario que quiere acceder al recurso y por tanto, a la hora de autorizar tendremos el nombre del usuario que quiere hacer la petición. Esto es importante si en la política queda remarcado el nombre de usuarios que pueden acceder a un recurso. Sin este paso, no sería posible controlar quién intenta acceder a una imagen cifrada. Es por tanto necesario que los usuarios protejan bien sus credenciales y no las compartan con nadie.

Una vez se ha realizado el proceso de inicio de sesión se muestra una interfaz con otro formulario. En este caso, nos encontramos con el formulario que nos permitirá rellenar la petición.

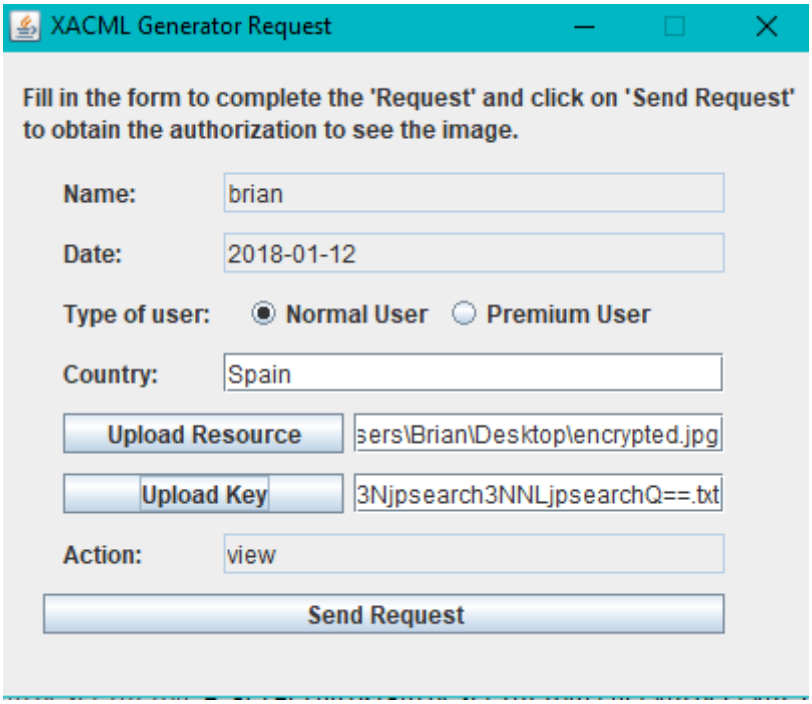


Figura 21: Formulario de los datos de la petición de acceso

Podemos observar que tanto el campo nombre y fecha no son editables y ya aparecen completos una vez se alcanza esta interfaz. La fecha coincide con la fecha actual del sistema. Una alternativa a esta implementación seria acceder a un servidor externo para obtener la fecha actual ya que coger la fecha del sistema puede ser un fallo de seguridad y puede ser cambiada antes de abrir esta aplicación.

A continuación, encontramos los campos *Type of user* y *Country*. Estos campos son opcionales pero se recomienda completarlos ya que pueden ser parámetros que se requieran en la política de privacidad. Finalmente, aparece el botón de *Upload Resource* y *Upload Key* que nos permitirá escoger el recurso al cual acceder y la clave para descifrar correspondientemente.

Para terminar, al apretar el botón de *Send Request*, se hará toda la lógica de la aplicación para generar la petición de acceso y evaluar su resultado. Tanto si el resultado es positivo o negativo, se informará al usuario con el mensaje pertinente.

6.4.1. Implementación

A continuación se detalla las funciones principales que se han desarrollado para esta aplicación, pese a que otras funciones han sufrido algunos cambios necesarios para adaptarse al nuevo proyecto:

- Añadir la posibilidad de insertar una clave de seguridad en el formulario.

- Nuevo diseño de descifrado de imagen y política de privacidad.
- Nuevo sistema de obtención de metadatos de *RightDescription*.

En primer lugar encontramos *bsend.addActionListener(new ActionListener())* que responde al evento asociado al clicar el botón de *Send Request.*

```
191        bsend.addActionListener(new ActionListener() {
192            public void actionPerformed(ActionEvent ae) {
193                String resultat = "";
194                String ress = "";
195                try {
196                    resultat = new EncryptDecryptString().decryptRD(res.getText(), keyres.getText());
197                    //TODO Revisar reglas
198
199                    key_data = getKeyData(EncryptDecryptString.rightsDesciptions);
200                    String before_data =  resultat.split("ffdb",2)[0] + "ffdb";
201                    String only_data = resultat.split("ffdb",2)[1];
202
203                    byte[] u = EncryptDecryptString.hexStringToByteArray(only_data); //bytes data cifrado
204                    byte[] data_decrypted = EncryptFile.decryptData(u, key_data); //bytes_data descifrado
205                    final StringBuilder builderr = new StringBuilder();
206                      for(byte b : data_decrypted) {
207                          builderr.append(String.format("%02x", b));
208                    }
209                    String s = JPSearchRewriter.hex2String(builderr.toString());
210                    String newS = changeSegmentsApp11(s);
211                    ress = before_data + newS;
212
213
214                } catch (InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException
215                        | IllegalBlockSizeException | BadPaddingException | IOException e1) {
216                    JOptionPane.showMessageDialog(null, "There was an error to decrypt the file. Check that the key is correct and try again.", "JPS
217                }
218                new JPSearchParser(res.getText());
219                String policy = null, request = null;
220                //Extract Policy from image metadata
221                String actualRightDescription = getActualRightDescription(EncryptDecryptString.rightsDesciptions);
222                String referenceRightDescription = getReferenceRightDescription(EncryptDecryptString.rightsDesciptions);
223                String language = getLanguage(EncryptDecryptString.rightsDesciptions);
224                if(actualRightDescription.equals("") && referenceRightDescription.equals("")) {
225                    FileOutputStream fos;
226                    try {
227                        fos = new FileOutputStream("C:\\Users\\Brian\\Desktop\\decrypted.jpg");
228                        fos.write(EncryptDecryptString.hexStringToByteArray(ress));
229                        JOptionPane.showMessageDialog(null, "The file decrypted.jpg was created successfully at Desktop folder.", "JPSearch Editor",
230                    } catch (IOException ee) {
231
232                    }
233                }
234                else if(!actualRightDescription.equals("") || !referenceRightDescription.equals("")) {
235                    //Make Request from input data
236                    ArrayList<String> features = new ArrayList<String>();
237                    features.add(name.getText());
238                    features.add(date.getText()); //TODO get from ntp?
239                    features.add(country.getText());
240                    String resource = "urn:mimage:";
241                    String[] parts = res.getText().split("\\\\");
242                    for (int i = 0; i < parts.length; i++) {
243                        if (i == parts.length-1) resource = resource + parts[i];
244                    }
245                    features.add(resource);
246                    features.add(action.getText());
247                    for (int i = 0; i < features.size(); i++) System.out.println(features.get(i));
248                    Boolean nU = normalUser.isSelected();
249                    try {
250                        policy = DocumentHelper.createPolicy(actualRightDescription);
251                        request = DocumentHelper.createRequest(features, nU, language);
252
253                    } catch (Exception e) {
254                        JOptionPane.showMessageDialog(null, "The metadata of the image is empty or not complete.");
255                    }
256
257                    //Send xml files to Balana's Authoritzator
258                    System.setProperty(FileBasedPolicyFinderModule.POLICY_DIR_PROPERTY, "support/policy/");
259                    System.setProperty(ConfigurationStore.PDP_CONFIG_PROPERTY, "support/config/config_rbac.xml");
260                    Balana balana = Balana.getInstance();
261                    PDP pdp = new PDP(balana.getPdpConfig());
262                    request = "C:" + System.getProperty("file.separator") +
263                            "Users" + System.getProperty("file.separator") +
264                            "Brian" + System.getProperty("file.separator") +
265                            "Desktop" + System.getProperty("file.separator") + request + ".xml";
266
267                    String response = "";
268                    try {
269                        response = pdp.evaluate(getFile(request, StandardCharsets.UTF_8));
270                    } catch (IOException e) {
271                        e.printStackTrace();
272                    }
273                    Document document = DocumentHelper.convertStringToDocument(response);
274                    Element root = document.getDocumentElement();
275                    NodeList nList = root.getElementsByTagName("Result");
276                    Node nNode = nList.item(0);
277                    Element eElement = (Element) nNode;
278                    String result = eElement.getElementsByTagName("Decision").item(0).getTextContent();
279                    if (result.equals("Permit")) {
280                        try {
281                            FileOutputStream fos = new FileOutputStream("C:\\Users\\Brian\\Desktop\\decrypted.jpg");
282                            fos.write(EncryptDecryptString.hexStringToByteArray(ress));
283                        } catch (IOException e) {
284                            e.printStackTrace();
285                        }
286                        JOptionPane.showMessageDialog(null, "Authorization completed. The file decrypted.jpg was saved in the Desktop folder.");
```

```
287                 }
288                 else if (result.equals("Deny")) {
289                     JOptionPane.showMessageDialog(null, "You don't have authorization to see the image.");
290                 }
291                 else if (result.equals("NotApplicable")) {
292                     JOptionPane.showMessageDialog(null, "Your request is not applicable for the policy included in this image. Try again with ot
293                 }
294                 else {
295                     JOptionPane.showMessageDialog(null, "The answer to your request is indeterminate. An error has occurred when your request wa
296                 }
297             }
298         }
299     });
300 }
301
```

Figura 22: Función bsend.addActionListener(new ActionListener());

Inicialmente se descifra la política de seguridad con la clave que ha introducido el usuario en el formulario anterior. Este paso nos ofrece como resultado la política de seguridad en texto plano. Dado que podemos leerla, se obtiene la segunda clave necesaria para descifrar los datos de la imagen y que se encuentra en el campo *Keywords* de la estructura de *Right Description*. Seguidamente, se obtiene los datos descifrados de la imagen usando la clave obtenida anteriormente. En el caso de haber algún fallo en este paso, ya sea porque ha habido un error o porque la clave usada no es la correcta, se le informará al usuario. A continuación, se genera la petición de acceso usando los datos introducidos en el formulario anterior y se envía a evaluar. La lógica descrita que sigue el autorizador es la descrita en el apartado Estado del arte donde se describe el funcionamiento de XACML (PEP, PAP, PIP, …). Finalmente se evalúa la respuesta. Si la respuesta es *Permit*, se genera el fichero *decrypted.jpg* en el Escritorio y se informa al usuario. En cualquier otro caso, se avisa al usuario que la autorización no ha sido satisfactoria y se indica el porqué. Si la imagen no contiene políticas de privacidad, se genera automáticamente el fichero resultante sin pasar por el evaluador.

En segundo lugar nos encontramos con las funciones necesarias para el descifrado de los datos y de la política: *decryptRD()* y *decryptData()*.

```
27  //Decrypt RightDescriptions from the image.
28  public static String decryptRD(String pathsrc, String pathkey) throws NoSuchAlgorithmException, NoSuchPaddingException,
29      try{
30          File k = new File(pathkey);
31          Path path = Paths.get(pathsrc);
32          byte[] bytes = Files.readAllBytes(path); //bytes del fichero a descifrar
33          StringBuilder builder = new StringBuilder();
34          for(byte b : bytes) {
35              builder.append(String.format("%02x", b));
36          }
37          String data = builder.toString();
38          String first = data.split("ffeb", 2)[0];
39          String rd = data.split("ffeb",2)[1];
40          rd = rd.split("ffe3",2)[0]; //right description
41
42          //String dimension = rd.substring(0,4);
43          rd = rd.substring(4, rd.length());
44
45          //solo fijarme en los bytes del RD
46          FileInputStream fisTargetFile = new FileInputStream(k);
47          String targetFileStr = IOUtils.toString(fisTargetFile, "UTF-8");
48
49          byte[] decodedKey = Base64.getDecoder().decode(targetFileStr);
50          SecretKey originalKey = new SecretKeySpec(decodedKey, 0, decodedKey.length, "AES");
51          byte[] ex = hexStringToByteArray(rd);
52
53          Cipher aes = Cipher.getInstance("AES/ECB/PKCS5Padding");
54          aes.init(Cipher.DECRYPT_MODE, originalKey);
55          byte[] res = aes.doFinal(ex); //rd decrypted
56
57          StringBuilder sb = new StringBuilder();
58            sb.append(Integer.toHexString(res.length + 2)); //data + dimension
59            while (sb.length() < 4) {
60                sb.insert(0, '0'); // pad with leading zero if needed
61            }
62            String length = sb.toString();
63          String resultat = first + "ffeb" + length + bytesToHex(res).toLowerCase() + "ffe3" + data.split("ffe3", 2)[1]
64
65          rightsDesciptions = JPSearchRewriter.hex2String(bytesToHex(res).toLowerCase());
66          return resultat;
67      }
68      catch(Exception e) {
69          JOptionPane.showMessageDialog(null, "There was an error to decrypt the file. Check that the key is correct an
70      }
71      return null;
72
73  }
```

Figura 23: Función decryptRD();

Esta función cuenta con dos parámetros. El primero indica la ruta de la imagen a descifrar y el segundo la ruta donde se encuentra el fichero que contiene la clave de cifrado.

Inicialmente, obtiene los bytes de la imagen a descifrar y, una vez convertidos a hexadecimal, obtiene solamente la información relevante a los metadatos APP11 que contienen la política de privacidad. Descifra la política y la retorna en texto plano.

```
14      //decrypt data
15      public static byte[] decryptData(byte[] data, SecretKey key) throws NoSuchAlgorithmException, NoSuchPaddingException, In
16          try{
17              Cipher aes = Cipher.getInstance("AES/ECB/PKCS5Padding");
18
19          aes.init(Cipher.DECRYPT_MODE, key);
20          return aes.doFinal(data);
21          }
22          catch(Exception e) {
23              JOptionPane.showMessageDialog(null, "There was an error to decrypt the file. Check that the key is correct and t
24          }
25          return null;
26      }
27  }
28
```

Figura 24: Función decryptData();

La función que se detalla arriba en la Figura 24 muestra el proceso de descifrado necesario para obtener los bytes en claro de los datos de la imagen. Estos datos no contienen ni metadatos APP3, APP11 ni de JFIF.

7. Metodología

7.1. Método de trabajo

Para el desarrollo del proyecto se ha utilizado una metodología de cascada. Esta metodología consiste en un proceso de diseño secuencial usado para el desarrollo de *software*. El proceso pasa por las distintas fases una por una (Figura 25). Las fases son: análisis, diseño, implementación, pruebas y mantenimiento. En la fase de análisis se decide que es lo que se debe hacer. Luego se diseña cómo será el sistema. A continuación se lleva a cabo la implementación del sistema, que luego se debe probar para comprobar que todo funciona correctamente. La fase de mantenimiento no la llevaremos a cabo, ya que se sale del alcance del proyecto.

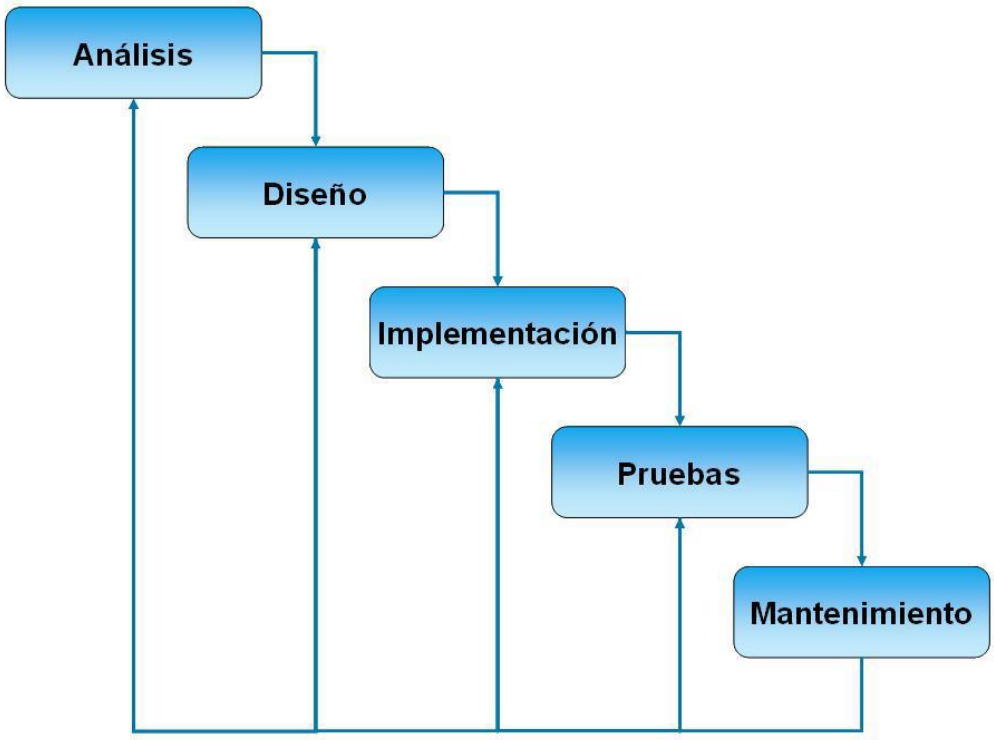


Figura 25: Metodología en cascada

7.2. Herramientas

Para poder tener un control de versiones de nuestro código, se ha usado *Git*. *Git* junto con su repositorio online, *GitHub*, que nos ha permitido controlar las versiones que se han ido haciendo del proyecto. Podemos acceder a dichas versiones desde cualquier ordenador sin importar la plataforma.

En cuanto a la documentación, se ha usado *Microsoft Word*, disponible para *Windows* y *Macintosh* y *Adobe Reader* para convertir esta documentación al formato *PDF*.

Para el desarrollo, se ha escogido *Java* como lenguaje de programación junto con *Eclipse IDE*, software que nos permite desarrollar el código del proyecto, compilarlo, ejecutarlo y crear el ejecutable final.

7.3. Validación

Para la validación, se han llevado a cabo varias reuniones con el director del proyecto con el fin de enseñar los progresos y comprobar si van en buen camino. Además, se han realizado diversas pruebas para ver si el *software* que se está desarrollando se ejecuta según lo esperado y sin errores a medida que iba avanzando el proyecto y se iba completando etapas.

8. Planificación temporal del proyecto

8.1. Planificación general

8.1.1. Instalación y comprensión de *JPSearch*

Requisitos

Para poder instalar *JPSearch* ha sido necesario obtener todo el proyecto de *JPSearch* modificado por Alberto Durán Montoro en su proyecto de final de grado. Este proyecto fue desarrollado con el IDE de Eclipse por lo que se ha usado el mismo IDE con el fin de obtener la mayor compatibilidad posible.
El lenguaje usado ha sido Java, por lo que será necesario usar dicho lenguaje y sus librerías.

Explicación

El proyecto debe seguir el trabajo de *JPSearch* teniendo en cuenta las modificaciones que usó Alberto para hacer su trabajo de final de grado. Por tanto, ha sido necesario comprender todo el código y valorar qué partes son necesarias y cuáles no. Seguidamente se ha modificado el código según el orden de las tareas.

Al ser un código extenso y no comentado totalmente, ha sido necesario un buen tiempo para poder entenderlo. Esta fase es importante ya que una mala comprensión del código puede implicar una extensión del proyecto más larga de lo normal o hacer trabajo redundante.

Tiempo estimado

Se estima que se ha necesitado una semana y media (10-11 días) en instalar el proyecto, probar su uso y comprender el código.

8.1.2. Cambios en la estructura e inserción de reglas XACML

Requisitos

Para esta etapa ha sido necesario comprender perfectamente la estructura de las imágenes JPEG. Entender su funcionamiento, el orden de la estructura y los límites que marcan cada una de sus secciones. Para poder aplicar dichos cambios al proyecto, será necesario haber superado el apartado 2 (Instalación y Comprensión de JPSearch).

Explicación

Las imágenes JPEG tienen una estructura que comprende diversas zonas. Cada zona se usa para un objetivo concreto. Para poder colocar las zonas, se usan marcos. Estos marcos indican en qué bytes comienza una zona. Su estructura siempre es 0xFF, 0xXX, donde XX indica el byte de la nueva zona. Algunas partes de la estructura son opcionales, como algunos metadatos que pueden no aparecer. Otras, en cambio, son globales, es decir, son necesarias en todas las imágenes. Como necesarias encontramos, por ejemplo, el SOI (Start of image) y el EOI (End of image) que empiezan respectivamente en 0xFF, D8 y 0xFF, 0xD9.

Los metadatos de *JPSearch* actualmente se guardan en una zona llamada APP3 y los datos en SOS, determinado por el marco 0xFF, DA. El objetivo de esta tarea es cambiar de posición los datos para ponerlos en APP11, según indica el artículo escrito por Silvia Llorente y Jaime Delgado. Además, permite la inserción de las reglas XACML en una zona APP11 distinta a los datos. Dos características importantes de las zonas APPx son que su tamaño es variable y que se instalan en el marco 0xFF, 0xEn donde n indica el número de APP. En nuestro caso, APP11, queremos que se instalen en 0xFF, 0xEB.

Tiempo estimado

Se trata de una etapa compleja ya que requiere comprender bien la estructura y la manipulación de bytes. Se ha estimado un tiempo de 2 semanas (14 días).

8.1.3. Cifrado y descifrado de reglas XACML y datos

Requisitos

Para poder realizar esta etapa ha sido necesario haber realizado el apartado 3 (Cambios en la estructura e inserción de reglas XACML) ya que es necesario que la estructura esté terminada. Para el cifrado es necesario determinar qué bytes se van a cifrar. Hacer esta etapa antes del apartado 3 haría que se tuviera que modificar completamente posteriormente.

Explicación

Para poder mantener un intercambio de imágenes seguro, se ha determinado usar criptografía de clave simétrica, es decir, con el uso de una clave tanto para el cifrado como para el descifrado. Se ha determinado el uso de AES de 128 bits con modo ECB como algoritmo de cifrado y descifrado.

Primero se cifrarán las reglas XACML con una clave aleatoria. Ésta clave será la que se comparta con los receptores y se autogenera cada vez que se cifra una imagen. Seguidamente se cifrará la parte de datos con otra clave. Ésta clave se almacenará en las reglas XACML en el campo keyword codificada en base64. El usuario receptor nunca tendrá acceso a esta clave. Dicha clave es diferente a la primera y solo será utilizada por la herramienta de descifrado sin control del usuario.

La primera clave, que permite cifrar y descifrar las reglas XACML, se almacenará en una base de datos pública para los usuarios de la empresa. La aplicación informará al emisor qué clave se ha de usar y el emisor advertirá al receptor de cuál es la clave correcta.

Las reglas XACML pueden contener información de la fecha límite y de los usuarios que pueden descifrar la imagen. Aunque el usuario receptor tenga la clave y sea un usuario autorizado, si la fecha excede de la fecha impuesta, se rechazará el permiso a la imagen. La aplicación de descifrado usará la clave seleccionada de la base de datos e identificará al receptor con un previo paso de inicio de sesión. Si todo es correcto, descifrará las reglas XACML y seguidamente descifrará los datos con la clave almacenada en la imagen.

Para poder determinar si la fecha es la correcta, actualmente se utiliza la fecha del sistema. Esto puede ser vulnerable si el usuario receptor modifica previamente la fecha de su sistema. Como alternativa, se ha pensado en consultar la fecha actual a NTP (*Network Time Servers*). De esta forma el usuario no podría engañar a la aplicación.

Tiempo estimado

Para poder realizar esta etapa será necesaria la comprensión del algoritmo de cifrado, la gestión de los bytes cifrados, el descifrado, las reglas XACML y las bases de datos. Esta etapa ha sido la más compleja de todas y se estima que se ha necesitado un tiempo de 1 mes (30 días).

8.1.4. Fase de pruebas

Requisitos

Para poder realizar esta etapa es necesario que el proyecto esté terminado. Necesita tener una versión beta del proyecto para testear posibles fallos. De todas formas, se pueden aplicar pruebas específicas al terminar cada tarea.

Explicación

El objetivo de esta etapa es la detección y corrección de posibles fallos que alteren el flujo adecuado del programa. Se han realizado muchas pruebas necesarias para determinar si la fase de desarrollo ha terminado y se puede considerar válido. Además se ha realizado un seguimiento de las tareas que inicialmente se buscaba realizar y las que finalmente se han realizado.

Tiempo estimado

Esta etapa ha dependido de los fallos encontrados y no se ha alargado más de lo esperado. Se ha estimado una longitud de una semana y media o dos semanas (de 10-11 a 14 días).

8.1.5. Memoria del proyecto

Requisitos

Esta etapa se ha teniendo en cuenta a medida que avanzaba el proyecto pero se considera que la documentación formal se realizará una vez se haya terminado la parte de desarrollo. Es por esta razón que se coloca en último lugar ya que requiere que toda la manipulación de código se realice y funcione antes de documentarlo.

Explicación

La memoria del proyecto tiene como objetivo documentar todo el proyecto desde elementos técnicos a elementos históricos. Esta memoria puede contener explicaciones de datos o posibles implementaciones que han sido relevantes para el desarrollo del proyecto pero que no se han llevado a cabo.

Tiempo estimado

Teniendo en cuenta el tiempo usado en documentar durante la fase de GEP y la fase de documentación después de realizar la de desarrollo, se considera un tiempo estimado de 3 a 4 semanas (de 21 a 30 días).

9. Recursos

9.1. Recursos personales

Para el desarrollo de este proyecto, solo ha sido necesario el uso de un trabajador que realiza diversas funciones con el fin de terminar el proyecto en el plazo establecido.

El tiempo estimado de dedicación semanal es de aproximadamente 30 horas, aunque es este número puede variar según la carga de trabajo y el flujo del proyecto.

9.2. Recursos materiales

- **Ordenador *MacBook Pro* con pantalla retina de 13 pulgadas**: Herramienta hardware para el desarrollo de este proyecto. Se usa para todas las tareas incluyendo desde la programación hasta la memoria.
- ***Eclipse IDE***: *Software* necesario para el desarrollo de la parte *software* del proyecto. Permite escribir el código, compilarlo y ejecutarlo.
- ***Git y Github***: Herramienta software para el control de versiones del proyecto y servidor en el que se guarda el código, respectivamente.
- **Correo electrónico *Gmail***: Herramienta de comunicación que se usa para contactar con el director del proyecto.
- **Conexión a Internet**: Hardware necesario para llevar a cabo parte de la investigación del proyecto. También nos proporciona acceso a otras herramientas como *Gmail* y *Github*.

10. Calendario del proyecto

10.1. Estimación de horas por tarea

Se le asigna una estimación de horas por cada tarea para poder planificar y controlar el tiempo del trabajador. Posteriormente, nos permitirá comparar la estimación con la realidad. En los Anexos 1 y 2 podremos encontrar diagramas de Gantt que muestre la duración de las tareas y sus precedencias de forma más gráfica.

Tarea	Responsable	Horas
Instalación y comprensión de *JPSearch*		45
1. Instalación en Eclipse	Desarrollador	3
2. Corrección de errores	Desarrollador	5
3. Comprensión del código	Desarrollador	37
Cambios en la estructura e inserción de reglas XACML		60
1. Estructura de la imagen	Jefe de Proyecto	10
2. Inserción de reglas XACML	Desarrollador	15
3. Modificación de los metadatos	Desarrollador	35
Cifrado y descifrado de reglas XACML y datos		128
1. Cifrado de la imagen	Jefe de proyecto	50
2. Validación de reglas XACML y autorización	Desarrollador	48
3. Descifrado de la imagen	Desarrollador	30
Fase de pruebas		60
1. Prueba de *JPSearch* base	Ingeniero QA	10
2. Prueba de la nueva estructura	Ingeniero QA	10
3. Prueba reglas XACML	Ingeniero QA	20
4. Prueba criptográfica	Ingeniero QA	20
Memoria	Jefe de Proyecto	90
Total		**383**

Tabla 2: Estimación en horas de las distintas tareas del proyecto

11. Gestión económica del proyecto

En este apartado vamos a analizar el coste económico del proyecto. Para poder hacerlo de una forma más estructurada, vamos a dividir los costes según su naturaleza. Cabe tener en cuenta que este proyecto no se basa en la producción de diversas unidades, por lo que sólo se tendrá en producción un solo producto en una sola unidad. Finalmente, veremos los posibles imprevistos que podemos encontrarnos en la realización del proyecto así como los costes de contingencia. Todos los costes que se presenten serán con IVA del 21% incluido.

11.1. Estimación de costes

11.1.1. Costes de recursos humanos

Los costes de recursos humanos dependerán del número de horas que se dediquen al proyecto ya que solo contamos con un trabajador que realiza todas las tareas. Los costes se han basado en los resultados que ofrece la empresa JobTonic en su página web [19].

Rol	Horas estimadas	Salario (€/h)	Coste estimado(€)
Jefe de proyecto	150	16	2400
Desarrollador	173	9	1557
Ingeniero QA	60	19	1140
Total	**383**		**5097**

Tabla 3: Coste de recursos humanos

Tarea	Horas	Recurso	Salario (€/h)	Coste (€)
Instalación y comprensión de *JPSearch*	45			
1. Instalación en Eclipse	3	Desarrollador	9	27
2. Corrección de errores	5	Desarrollador	9	45
3. Comprensión del código	37	Desarrollador	9	333
Cambios en la estructura e inserción de reglas XACML	60			
1. Estructura de la imagen	10	Jefe de Proyecto	16	160
2. Inserción de reglas XACML	15	Desarrollador	9	135
3. Modificación de los metadatos	35	Desarrollador	9	315
Cifrado y descifrado de reglas XACML y datos	128			
1. Creación de certificados con par de claves	40	Jefe de proyecto	16	640
2. Herramienta generadora de certificados	48	Desarrollador	9	432
3. Modificación del código del proyecto	30	Desarrollador	9	270
Fase de pruebas	60			
1. Prueba de *JPSearch* base	10	Ingeniero QA	19	190
2. Prueba de la nueva estructura	10	Ingeniero QA	19	190
3. Prueba de cifrados	20	Ingeniero QA	19	380
4. Prueba criptográfica	20	Ingeniero QA	19	380
Memoria	90	Jefe de Proyecto	16	1440
Total	**383**			**4937**

Tabla 4: Costes directos por actividad

11.1.2. Costes de recursos materiales

En este apartado se incluirán los costes de los recursos materiales de *hardware* y *software* necesarios para el proyecto. Estos gastos son considerados indirectos ya que no dependen del proyecto. Estos gastos se tendrán pese a no realizar un proyecto y son proporcionales al tiempo de uso.

Producto	Unidades	Coste por unidad	Coste estimado (€)
MacBook Pro 13" Pantalla Retina	1	200	200
Impresiones en papel del TFG	5	50	250
Total			**550**

Tabla 5: Costes indirectos por recursos materiales

11.1.3. Costes generales indirectos

En esta sección encontramos todos aquellos costes indirectos que no pertenecen a ninguna de las dos clasificaciones anteriores, pero que son necesarios para el uso diario y, por supuesto, para realizar el proyecto. En este caso vamos a suponer que la duración del proyecto dura aproximadamente un cuatrimestre (4 meses).

Producto	Precio (€/mes)	Coste estimado (€)
Internet	50	200
Local	450	1800
Luz	40	160
Agua	15	60
Gas	20	80
Combustible	60	240
Total	**635**	**2540**

Tabla 6: Costes indirectos generales

11.1.4. Contingencia

Dado que puede surgir algún problema durante la realización del proyecto se ha considerado aplicar un porcentaje del 15% al coste estimado de cada recurso con el fin de solventar dichos problemas. Solo se usará la cantidad de la contingencia si es necesario.

Producto	Porcentaje	Precio (€)	Coste (€)
Recursos humanos	15%	4937	740
Recursos materiales	15%	550	82
Recursos generales	15%	2540	381
Total			**1213**

Tabla 7: Costes de contingencia

11.1.5. Imprevistos

En este apartado se tienen en cuenta los posibles imprevistos y/o contratiempos que pueden surgir en el desarrollo del proyecto. En nuestro caso se contemplan dos imprevistos principalmente:

- **Avería del ordenador**: En caso de tener una avería del ordenador con el que se va a realizar todo el proyecto hay que tener en cuenta de que el tiempo es muy ajustado por lo que no podemos plantear la posibilidad de repararlo. En este caso, se tendría que comprar un ordenador nuevo con las mismas características. Su coste, tal y como hemos comentado anteriormente, rondaría los 1600€. Teniendo en cuenta que el ordenador es relativamente nuevo, la probabilidad de que esto ocurra es muy baja (5%).

- **Retraso en el desarrollo**: Es posible que la estimación de horas realizada anteriormente sea demasiado justa y se deba alargar la duración de las actividades. Esto es algo que puede suceder con frecuencia, pese a que se ha estimado al alta. Se le asigna una probabilidad del 35%. Podemos suponer que este imprevisto supondrá una duración extra de 15 días.

Producto	Probabilidad	Unidades	Precio (€)	Coste (€)
Avería ordenador	5%	1	200	10
Retraso desarrollo	35%	60 (horas)	9 €/hora	189
Total				**199**

Tabla 8: Costes de imprevistos

11.1.6. Presupuesto final

Para terminar, se va a realizar un presupuesto final teniendo en cuenta todos los costes estimados anteriormente por secciones. Este proyecto no pretende obtener un beneficio económico.

Concepto	Coste (€)
Recursos humanos	4937
Recursos materiales	550
Costes generales	2540
Contingencia	1213
Imprevistos	199
Total de impuestos (21%, IVA)	**1982**
Total	**9439**

Tabla 9: Presupuesto final del proyecto

11.2. Control de gestión

Para la realización de este proyecto hay que tener en cuenta que algunos recursos ya fueron comprados anteriormente y que los recursos generales seguirán existiendo pese a que no se realice este proyecto. Por tanto, lo único que podemos gestionar y controlar son los recursos humanos.

Con el fin de gestionar los costes, al final de cada tarea se comprobará cuantas horas se han dedicado para terminarla y se comparará con la previsión inicial. Esto nos permitirá comprobar el ritmo de trabajo y tomar acciones al respecto. Si el coste real del proyecto va ascendiendo al estimado, será necesario usar el fondo de contingencia para cubrir los gastos extras. Finalmente se podrá calcular las desviaciones que ha supuesto el proyecto y comparar con las estimaciones.

Para poder tener mejor control, podemos usar diversos indicadores que nos permitan evaluar cómo va el proyecto:

- Desviaciones en la realización de tareas (en coste): (coste estimado – coste real)* consumo horas real
- Desviaciones de un recurso *hardware* (en coste): (coste estimado – consumo real) * coste real
- Desviaciones en la realización de tareas (en horas): (consumo estimado – consumo real) * coste real
- Desviaciones totales en la realización de tareas: (coste estimado total – coste real total)
- Desviaciones totales de recursos: (coste estimado total – coste real total)

11.3. Comparación con estimación inicial

La previsión inicial se ha diferenciado con el resultado final fundamentalmente por el conjunto de horas que se han dedicado al proyecto. La previsión inicial contaba con 297 horas dedicadas a la realización de este proyecto, las cuales se ha visto en la versión final que han sido 383 horas. Podemos ver una comparativa de horas más gráfica en los Anexos 1 y 2 con los Diagramas de Gantt de ambas versiones.

Esta diferencia ha marcado un presupuesto totalmente diferente en ambas versiones. Inicialmente se contaba con un presupuesto aproximado de 6464€ y la versión final indica un presupuesto de 9439€, una diferencia muy notable que puede cambiar un proyecto por completo. Cabe tener en cuenta que las remuneraciones de los trabajadores se ha modificado para que sea lo más real posible en esta versión final.

En cuanto a la metodología, costes generales indirectos y costes materiales, las desviaciones e imprevistos, ambas versiones han seguido con el mismo contenido y, por tanto, no han cambiado en el presupuesto final.

12. Sostenibilidad y compromiso social

12.1. Matriz de sostenibilidad

En la siguiente tabla o matriz de sostenibilidad se muestra la valoración final de la sostenibilidad del proyecto, en base a las consideraciones de los tres puntos anteriores. Las puntuaciones individuales van de 0 a 10 y la total de 0 a 30. La matriz de sostenibilidad sólo muestra los resultados para la Puesta a Punto del Proyecto (PPP), que comprende su planificación y parte del desarrollo.

Aspecto	Valoración
Ambiental	3
Económico	10
Social	9
Total	**22**

Tabla 10: Matriz de sostenibilidad

12.2. Ambiental

Este proyecto requiere de un ordenador para realizar todo el desarrollo por lo que el impacto ambiental en cuanto a materiales preciosos y energía, podría ser prejudicial.

El objetivo del proyecto es el uso de las herramientas desarrolladas, por cada usuario de la empresa por lo que sería necesario tantos ordenadores como usuarios en la empresa. Esto puede suponer un impacto ambiental muy grande ya que una empresa puede suponer millones de usuarios.

Cabe tener en cuenta que todo el desarrollo del proyecto se realizará con un ordenador que ya se utilizaba anteriormente con fines personales por lo que no habrá un impacto extra durante la realización del proyecto.

Una mejora a este proyecto sería mantener todas las herramientas en la nube, pero no reduciría el número de usuarios y, por tanto, el número de máquinas sería el mismo.

12.3. Económico

Teniendo en cuenta los costes estimados anteriormente por secciones, se puede apreciar que los mayores costes se encuentran en los recursos humanos y los generales. En el caso de los costes generales, se podría buscar una solución más eficiente en cuanto a transporte o gastos de vivienda, pero pueden ser muy complejos de cambiar. En el caso de los costes de recursos humanos, dependen directamente del número de horas dedicadas a la realización del proyecto, por lo que la única manera de reducir este coste es reducir las horas de producción.

Si se reduce las horas de producción, sería necesario ampliar el personal y por consiguiente, aumentar los recursos materiales.

12.4. Social

El aspecto social es muy importante en este proyecto ya que permitirá aumentar la confianza y seguridad de las empresas. El intercambio de imágenes puede ser un riesgo para la empresa ya que puede derivar en ataques informáticos dirigidos que pongan en riesgo información confidencial, dinero o llegar a perjudicar a alguno de sus usuarios.

Este proyecto puede suponer un incremento en la fiabilidad de las relaciones entre usuarios de la empresa y en posible futuro, las relaciones entre diferentes empresas.

13. Identificación de leyes y regulaciones

En este proyecto, enmarca un papel fundamental la seguridad de la información. En nuestro caso, las imágenes y la información que contienen son los principales datos que es necesario que estén seguros de personas no autorizadas.

En la **LPI (Ley de Propiedad Intelectual)** podemos encontrar el artículo 1 [20] que dice:

"*La propiedad intelectual de una obra literaria, artística o científica corresponde al autor por el solo hecho de su creación."*

Más concretamente, el artículo 10.1 [21] de la **LPI** señala:

"*Son objeto de propiedad intelectual todas las creaciones originales literarias, artísticas o científicas expresadas por cualquier medio o soporte, tangible o intangible, actualmente conocido o que se invente en el futuro."*

Además, el artículo 128 [22] de la **LPI** indica específicamente sobre las fotografías que:

"*Quien realice una fotografía u otra reproducción obtenida por procedimiento análogo a aquélla, cuando ni una ni otra tengan el carácter de obras protegidas en el Libro I, goza del derecho exclusivo de autorizar su reproducción, distribución y comunicación pública, en los mismos términos reconocidos en la presente Ley a los autores de obras fotográficas."*

Los dos artículos anteriores intentan asegurar cuáles son nuestros derechos frente al uso y creación de fotografías.

A nivel internacional encontramos el artículo 12 de la **Declaración de los derechos humanos** (DUDH), sobre el derecho a la privacidad:

"Nadie será objeto de injerencias arbitrarias en su vida privada, su familia, su domicilio o su correspondencia, ni de ataques a su honra y su reputación. Toda persona tiene derecho a la protección de la ley contra tales injerencias o ataques."

El artículo 12 se refiere al derecho a la intimidad y defiende que nadie tiene derecho a la intromisión en la vida privada, en domicilio, correspondencia, honra o reputación. Por tanto si lo aplicamos a este proyecto, ya hemos visto ejemplos en los que se han publicado imágenes de la vida privada de los individuos sin consentimiento de estos.

El mismo texto se incluye en el artículo 17 del **Pacto internacional de los derechos civiles y políticos** (PIDCP).

A nivel europeo se encuentra el artículo 8 del **Convenio para la protección de los derechos humanos y libertades fundamentales** (CEDH), sobre el derecho a la vida privada y familiar:

1. Toda persona tiene derecho al respeto de su vida privada y familiar, de su domicilio y de su correspondencia.

2. No podrá haber injerencia de la autoridad pública en el ejercicio de este derecho, sino en tanto en cuanto esta injerencia esté prevista por la ley y constituya una medida que, en una sociedad democrática, sea necesaria para la seguridad nacional, la seguridad pública, el bienestar económico del país, la defensa del orden y la prevención del delito, la protección de la salud o de la moral, o la protección de los derechos y las libertades de los demás."

Este artículo es casi una extensión del artículo 12 de los **Derechos humanos**, incluyendo además a la administración pública, la cual tampoco podrá atentar contra el derecho a la vida privada, a no ser que conlleve un peligro para el resto de individuos.

También a nivel europeo encontramos los artículos 7 y 8 de la **Carta de los derechos fundamentales de la unión europea** (CDFUE):

> **Artículo 7**: Respeto a la vida privada y familiar
> Toda persona tiene derecho al respeto de su vida privada y familiar, de su domicilio y de sus comunicaciones.
>
> **Artículo 8**: Protección de datos de carácter personal
> *1. Toda persona tiene derecho a la protección de los datos de carácter personal que la conciernan.*
>
> *2. Estos datos se tratarán de modo leal, para fines concretos y sobre la base del consentimiento de la persona afectada o en virtud de otro fundamento legítimo previsto por la ley. Toda persona tiene derecho a acceder a los datos recogidos que la conciernan y a obtener su rectificación.*
>
> *3. El respeto de estas normas estará sujeto al control de una autoridad independiente.*

En estos artículos que son más recientes se empiezan a intuir cambios en la legislación para incluir todo tipo de comunicaciones y no sólo la correspondencia escrita. De este modo se pueden incluir también las nuevas tecnologías.

Por último nos encontramos dentro del marco nacional con el artículo 18 de la **Constitución española** (CE):

> 1. *Se garantiza el derecho al honor, a la intimidad personal y familiar y a la propia imagen.*
>
> *2. El domicilio es inviolable. Ninguna entrada o registro podrá hacerse en él sin consentimiento del titular o resolución judicial, salvo en caso de flagrante delito.*
>
> *3. Se garantiza el secreto de las comunicaciones y, en especial, de las postales, telegráficas y telefónicas, salvo resolución judicial.*

4. La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos.

En especial dentro de este artículo se encuentra el punto 4 en el que se aclara que se limitará el uso de la informática para proteger el derecho a la privacidad.

También dentro del marco nacional, la **Ley Orgánica de 1982** establece diversos artículos que extienden lo ya garantizado en el artículo 18 de la **CE**:

Artículo 1:

1. El derecho fundamental al honor, a la intimidad personal y familiar y a la propia imagen, garantizado en el artículo 18 de la Constitución, será protegido civilmente frente a todo género de intromisiones ilegítimas, de acuerdo con lo establecido en la presente ley orgánica.

2. Cuando la intromisión sea constitutiva de delito, se estará a lo dispuesto en el Código penal. No obstante, serán aplicables los criterios de esta ley para la determinación de la responsabilidad civil derivada del delito.

3. El derecho al honor, a la intimidad personal y familiar y a la propia imagen es irrenunciable, inalienable e imprescriptible. La renuncia a la protección prevista en esta ley será nula, sin perjuicio de los supuestos de autorización o consentimiento a que se refiere el artículo segundo de la esta ley.

Artículo 2:

1. La protección civil del honor, la intimidad y de la propia imagen quedará delimitada por las leyes y por los usos sociales atendiendo al ámbito que, por sus propios actos, mantenga cada persona reservado para sí misma o su familia.

2. No se apreciará la existencia de intromisión ilegítima en el ámbito protegido cuando estuviere expresamente autorizado por ley o cuando el titular del derecho hubiese otorgado al efecto su consentimiento expreso.

3. El consentimiento a que se refiere el párrafo anterior será revocable en cualquier momento, pero habrán de indemnizarse, en su caso, los daños y perjuicios causados, incluyendo en ello expectativas justificadas.

Artículo 7: Tendrán consideración de intromisiones ilegítimas en el ámbito de la protección delimitado por el artículo segundo de esta ley:

1. El emplazamiento en cualquier lugar de aparatos de escucha, de filmación, de dispositivos ópticos o de cualquier otro medio apto para grabar o reproducir la vida íntima de las personas.

2. La utilización de aparatos de escucha, dispositivos ópticos o de cualquier otro medio para el conocimiento de la vida íntima de las personas o de manifestaciones o cartas privadas no destinadas a quien haga uso de tales medios, así como su grabación, registro o reproducción.

3. La divulgación de hechos relativos a la vida privada de una persona o familia que afecten a su reputación y buen nombre, así como la revelación o publicación del contenido de cartas, memorias u otros escritos personales de carácter íntimo.

4. La revelación de datos privados de una persona o familia conocidos a través de la actividad profesional u oficial de quien los revela.

5. La captación, reproducción o publicación por fotografía, filme, o cualquier otro procedimiento, de la imagen de una persona en lugares o momentos de su vida privada o fuera de ellos, salvo los casos previstos en el artículo octavo, dos.

6. La utilización del nombre, de la voz o de la imagen de una persona para fines publicitarios, comerciales o de naturaleza análoga.

7. La divulgación de expresiones o hechos concernientes a una persona cuando la difame o la haga desmerecer en la consideración ajena.

Artículo 8:

1. No se reputarán, con carácter general, intromisiones ilegítimas las actuaciones autorizadas o acordadas por la Autoridad competente de acuerdo con la ley, ni cuando predomine un interés histórico, científico o cultural relevante.

2. En particular, el derecho a la propia imagen no impedirá:
a) Su captación, reproducción o publicación por cualquier medio, cuando se trate de personas que ejerzan un cargo público o una profesión de notoriedad o proyección pública y la imagen se capte durante un acto público o en lugares abiertos al público.
b) La utilización de la caricatura de dichas personas, de acuerdo con el uso social
c) La información gráfica sobre un suceso o acaecimiento público cuando la imagen de una persona determinada aparezca meramente accesoria.

En caso del no cumplimiento de estos artículos las penas se regulan a través del código penal en los siguientes artículos: el artículo 197.1 del Código Penal [23] que trata sobre el descubrimiento y revelación de secretos y que castiga con penas de prisión de 1 a 4 años y multa, "el apoderamiento de documentos o efectos personales y la interceptación de comunicaciones". El delito se produce cuando un tercero se hace con documentos o efectos personales de alguien. Por otro lado, también podría aplicarse el artículo 197.2 del Código Penal que trata sobre el delito contra los secretos informáticos, que castiga con entre 2 y 5 años de cárcel por apoderarse, utilizar o modificar, sin autorización y en perjuicio de un tercero, datos de carácter personal o familiar con el propósito de vulnerar y descubrir la intimidad de otro, o el artículo 197.3 del Código Penal con prisión de 2 a 5 años la "difusión, revelación o cesión de datos, hechos o imágenes" obtenidos en las conductas del artículo 197.1 y 197.2 citados.

Por último, y refiriéndome a la parte más técnica del proyecto también se ha de valorar que este proyecto también está regulado por el estándar *JPSearch*, ya que los metadatos de las imágenes deben seguir la estructura que marca el esquema de *JPSearch* para que la imagen se considere parte de JPEG.

En general podemos observar como desde la legislación internacional a la nacional este proyecto que se centra en la privacidad ayudaría en el cumplimiento de las leyes citadas anteriormente, por tanto considero que mi proyecto contribuye a un beneficio social, ya que ayuda al complimiento de uno de los derechos fundamentales [24].

14. Conclusiones y trabajo futuro

14.1. Conclusiones

La realización de este proyecto ha sido muy grata ya que pese a su dificultad, ha servido para mejorar mis cualidades a la hora de desarrollar y aprender tecnologías que no había usado anteriormente, como es el caso de XACML. Por otro lado, este proyecto contaba con una motivación personal ya que me apasiona el sector de la seguridad informática y todo lo que conlleva. Pese a haber contactado con varios profesores y tener varios proyectos para escoger, me decidí sin duda por este proyecto por su impacto social y por el alcance que puede llegar a tener.

Ha sido un camino difícil ya que se partía de una aplicación existente con mucho contenido. No ha sido fácil poder comprender el código y continuar el trabajo añadiendo modificaciones y mejoras. Pese a que inicialmente parecía una tarea casi imposible de realizar, poco a poco vi cómo se podía avanzar e ir resolviendo los problemas y dificultades que iban apareciendo.

Inicialmente fue muy complejo pensar y desarrollar una solución final, ya que la amplitud que ofrece el título de este proyecto hacía que existieran muchas posibilidades que desarrollar. Se pensó en utilizar clave pública con certificados, usar reconocimiento facial, implantar medidas de seguridad como el factor de doble autentificación para iniciar sesión en la segunda aplicación, desarrollar la segunda aplicación como una aplicación web, entre otras.

Pese a haber realizado multitud de proyectos en asignaturas durante la carrera, éste en concreto ha sido con diferencia el más complicado. No solo por su contenido sino también por todo lo que conlleva desarrollar un producto de este tamaño, en un periodo de tiempo relativamente corto y luego poder documentarlo de la forma más detallada posible.

Sin duda, este proyecto me ha servido mucho para mejorar, para entender en cierta medida cómo funciona el desarrollo real de un proyecto grande y, por supuesto, me servirá como una lección de experiencia que me ayudará en un futuro en mi trabajo.

Finalmente, concluir que pese a las dificultades y los imprevistos se ha conseguido realizar los objetivos marcados para este proyecto.

14.2. Trabajo futuro

Este proyecto muestra una posible solución de seguridad para aplicar en imágenes JPEG. Hay que tener cuenta que JPEG sigue trabajando en encontrar una solución definitiva al problema de seguridad en sus imágenes y ésta es solo una posible implementación que puede trabajar a su vez con otras posibles propuestas.

Es necesario que se siga trabajando en mejorar la seguridad en las imágenes JPEG y seguir con la evolución del estándar JPEG, *Privacy and Security* [7], para ver cómo aplicar los resultados obtenidos en este proyecto.

15. Referencias

[1] JPSearch Registration Authority. (2012, February). What is JPSearch? http://dmag1.ac.upc.edu/jpsearch-ra/jpsearch.jsp

[2] Lesch, S. Brossard, D. (2015, December 10). XACML https://en.wikipedia.org/wiki/XACML

[3] OASIS Standard. (2013, 22 January). eXtensible Access Control Markup Language (XACML) Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[4] Brossard, D. (2010, May 23). Stackoverflow - Who uses XACML? http://stackoverflow.com/questions/2893247/who-uses-xacml

[5] Joint Picture Experts Group, https://jpeg.org/index.html

[6] JPEG Privacy & Security Abstract and Executive Summary https://jpeg.org/items/20150910_privacy_security_summary.html

[7] ISO/IEC JTC 1/SC29/WG1N75007, JPEG Privacy and Security Call for Proposals, Australia, Marzo 2017.

[8] Distributed Multimedia Applications Group (DMAG), http://dmag.ac.upc.edu

[9] Departament Arquitectura de Computadors (DAC), http://www.ac.upc.edu

[10] Universitat Politècnica de Catalunya (UPC), http://www.upc.edu

[11] Jaime Delgado, Silvia Llorente, Improving privacy in JPEG images, http://ieeexplore.ieee.org/document/7574676/ , Multimedia & Expo Workshops (ICMEW), 2016 IEEE International Conference on, Julio 2016.

[12] Durán, A., Llorente, S. (supervisor), Privacidad en imágenes jpg mediante XACML (en español), Trabajo de final de grado, Barcelona School of Informatics (FIB), http://upcommons.upc.edu/handle/2117/82555 , Enero 2016.

[13] Encriptación con AES https://es.wikipedia.org/wiki/Advanced_Encryption_Standard

[14] Bonadero, J. Bria, O. Liberatori, M. Villagarcía wanza, H. Expansión de la clave Rijndael: Diseño y optimización en VHDL http://www.iberchip.net/iberchip2005/articles/14/14--eljuje-Texto%20completo%20Expansion%20de%20clave%20en%20Rijndael.pdf

[15] Métodos de operación AES https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

[16] Demetriou, N. (2013, July 10). Metadata Interoperability with JPSearch http://www.slideshare.net/nikgt/metadata-interoperability-with-jpsearch

[17] JPSearch Overview
https://www.dimis.fim.uni-passau.de/iris/index.php?view=jpsearch

[18] Brossard, D. (2012, 27 April). Differences between XACML 2.0 and XACML 3.0
https://wiki.oasis-open.org/xacml/DifferencesBetweenXACML2.0AndXACML3.0

[19] Remuneraciones por posición, http://www.jobtonic.es/

[20] Órgano Ministerio de Cultura, Publicación en BOE el 22 Abril de 1996, Articulo1, http://noticias.juridicas.com/base_datos/Admin/rdleg1-1996.l1t1.html#a1

[21] Órgano Ministerio de Cultura, Publicación en BOE el 22 Abril de 1996, Artículo 10.1, http://noticias.juridicas.com/base_datos/Admin/rdleg1-1996.l1t2.html#a10

[22] Órgano Ministerio de Cultura, Publicación en BOE el 22 Abril de 1996, Artículo 128, http://noticias.juridicas.com/base_datos/Admin/rdleg1-1996.l2t5.html#a128

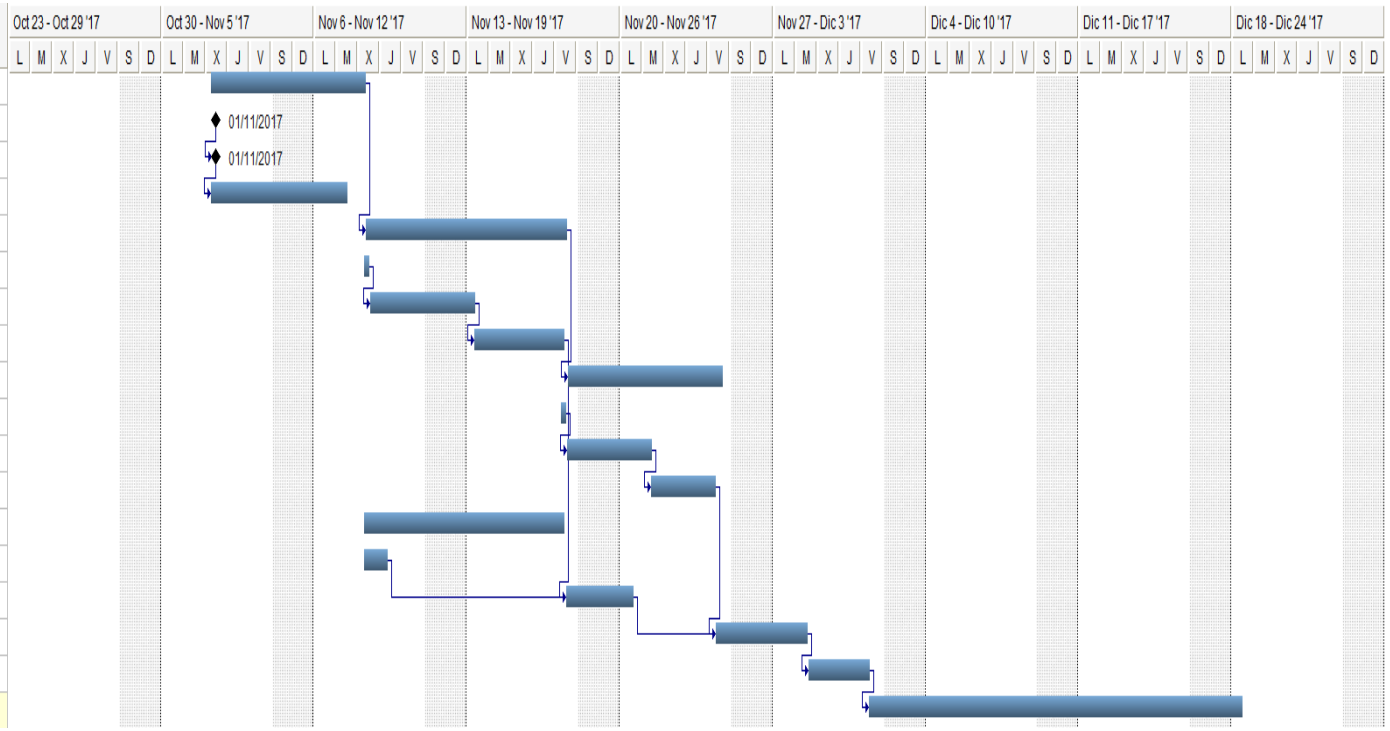
[23] Código Penal y legislación complementaria
https://www.boe.es/legislacion/codigos/codigo.php?id=38&modo=1&nota=0&tab=2

[24] Pacto Internacional de Derechos Económicos, Sociales y Culturales,
http://www.derechoshumanos.net/normativa/normas/1966-PactoDerechosEconomicosSocialesyCulturales.htm

Anexos

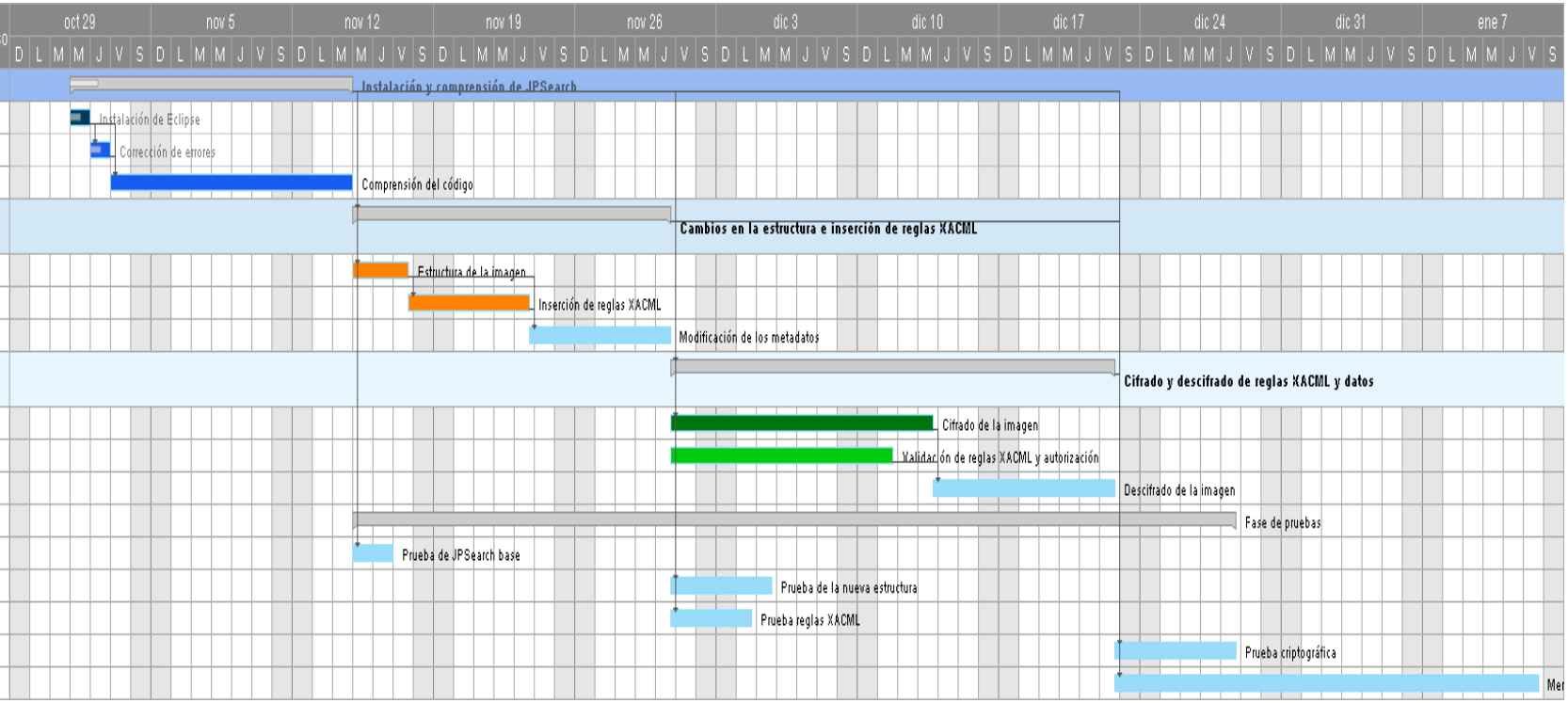
Anexo 1: Diagrama de Gantt Inicial

	Nombre	Duración	Inicio	Fin
1	Instalación y comprensión de JPSearch	42h	01/11/2017	08/11/2017
2	Instalación en Eclipse	0h	01/11/2017	01/11/2017
3	Corrección de errores	0h	01/11/2017	01/11/2017
4	Comprensión del código	37h	01/11/2017	07/11/2017
5	Cambio de formato de los metadatos de JPSearch	60h	08/11/2017	17/11/2017
6	Estructura de la imagen	5h	08/11/2017	08/11/2017
7	Cambio de estructura	20h	08/11/2017	13/11/2017
8	Modificación de los metadatos	35h	13/11/2017	17/11/2017
9	Certificado autofirmado y par de claves	42h	17/11/2017	24/11/2017
10	Creación de certifiados con par de claves	5h	17/11/2017	17/11/2017
11	Herramienta generadora de certificados	14h	17/11/2017	21/11/2017
12	Modificación del código del proyecto	23h	21/11/2017	24/11/2017
13	Fase de pruebas	60h	08/11/2017	17/11/2017
14	Prueba de JPSearch base	10h	08/11/2017	09/11/2017
15	Prueba de la nueva estructura	10h	17/11/2017	20/11/2017
16	Prueba de certificados y herramienta generadora	20h	24/11/2017	28/11/2017
17	Prueba criptográfica	20h	28/11/2017	01/12/2017
18	Memoria	90h	01/12/2017	18/12/2017

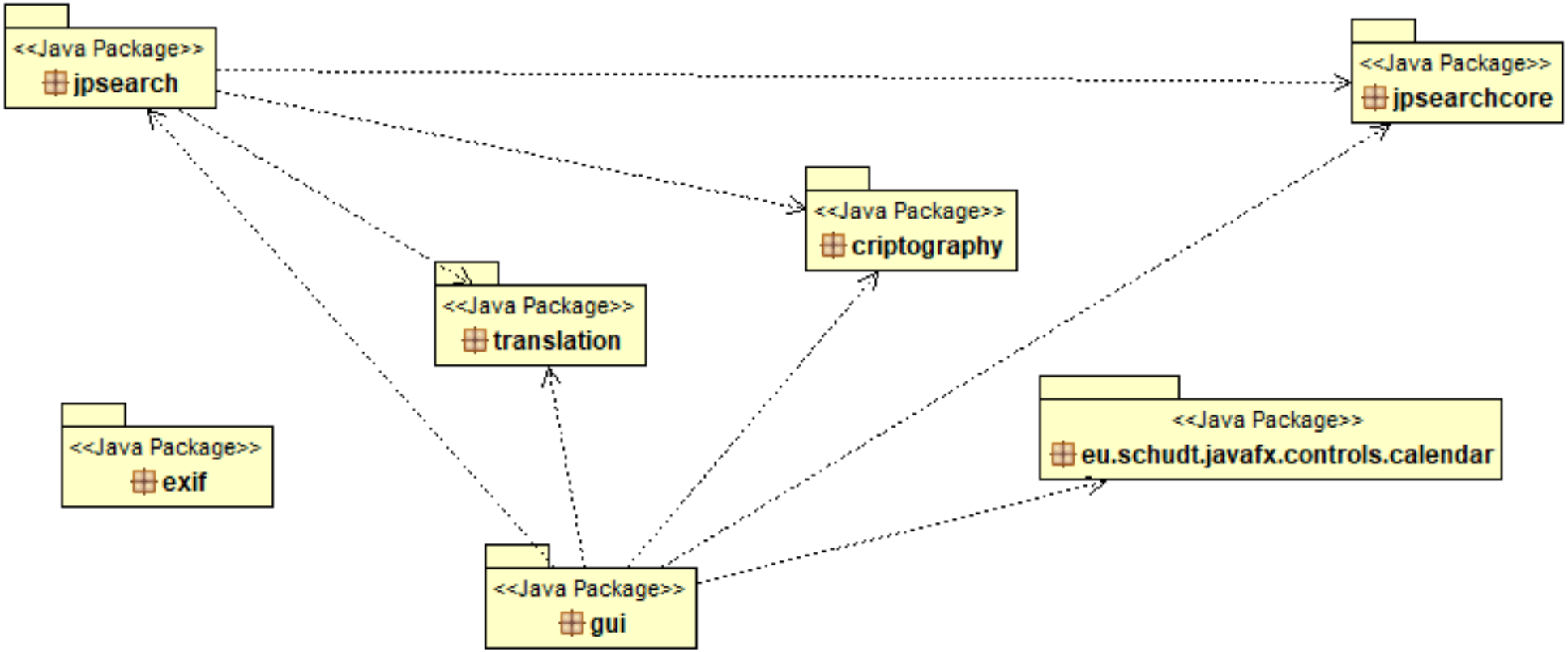


Anexo 2: Diagrama de Gantt final

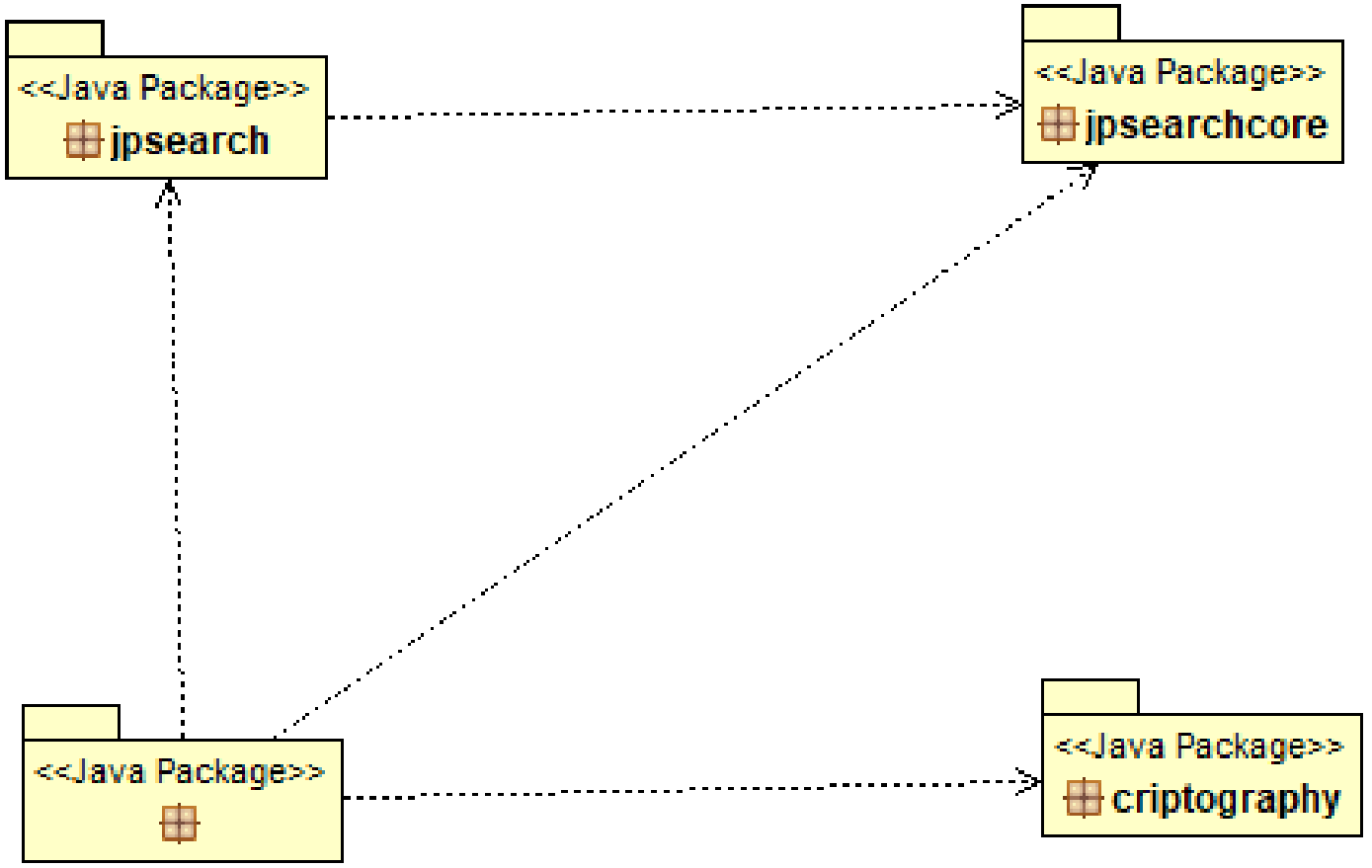
	Nombre de la tarea	Fecha de Ini	Fecha final	Predeceso
1	**Instalación y comprensión de JPSearch**	**01/11/17**	**14/11/17**	
2	Instalación de Eclipse	01/11/17	01/11/17	
3	Corrección de errores	02/11/17	02/11/17	2
4	Comprensión del código	03/11/17	14/11/17	2; 3
5	**Cambios en la estructura e inserción de reglas XACML**	**15/11/17**	**30/11/17**	**1**
6	Estructura de la imagen	15/11/17	17/11/17	1
7	Inserción de reglas XACML	17/11/17	23/11/17	6
8	Modificación de los metadatos	23/11/17	30/11/17	6; 7
9	**Cifrado y descifrado de reglas XACML y datos**	**30/11/17**	**22/12/17**	**1; 5**
10	Cifrado de la imagen	30/11/17	13/12/17	1; 5
11	Validación de reglas XACML y autorización	30/11/17	11/12/17	
12	Descifrado de la imagen	13/12/17	22/12/17	10; 11
13	**Fase de pruebas**	15/11/17	28/12/17	
14	Prueba de JPSearch base	15/11/17	16/11/17	1
15	Prueba de la nueva estructura	30/11/17	05/12/17	5
16	Prueba reglas XACML	30/11/17	04/12/17	5
17	Prueba criptográfica	22/12/17	28/12/17	9
18	**Memoria**	22/12/17	12/01/18	1; 5; 9



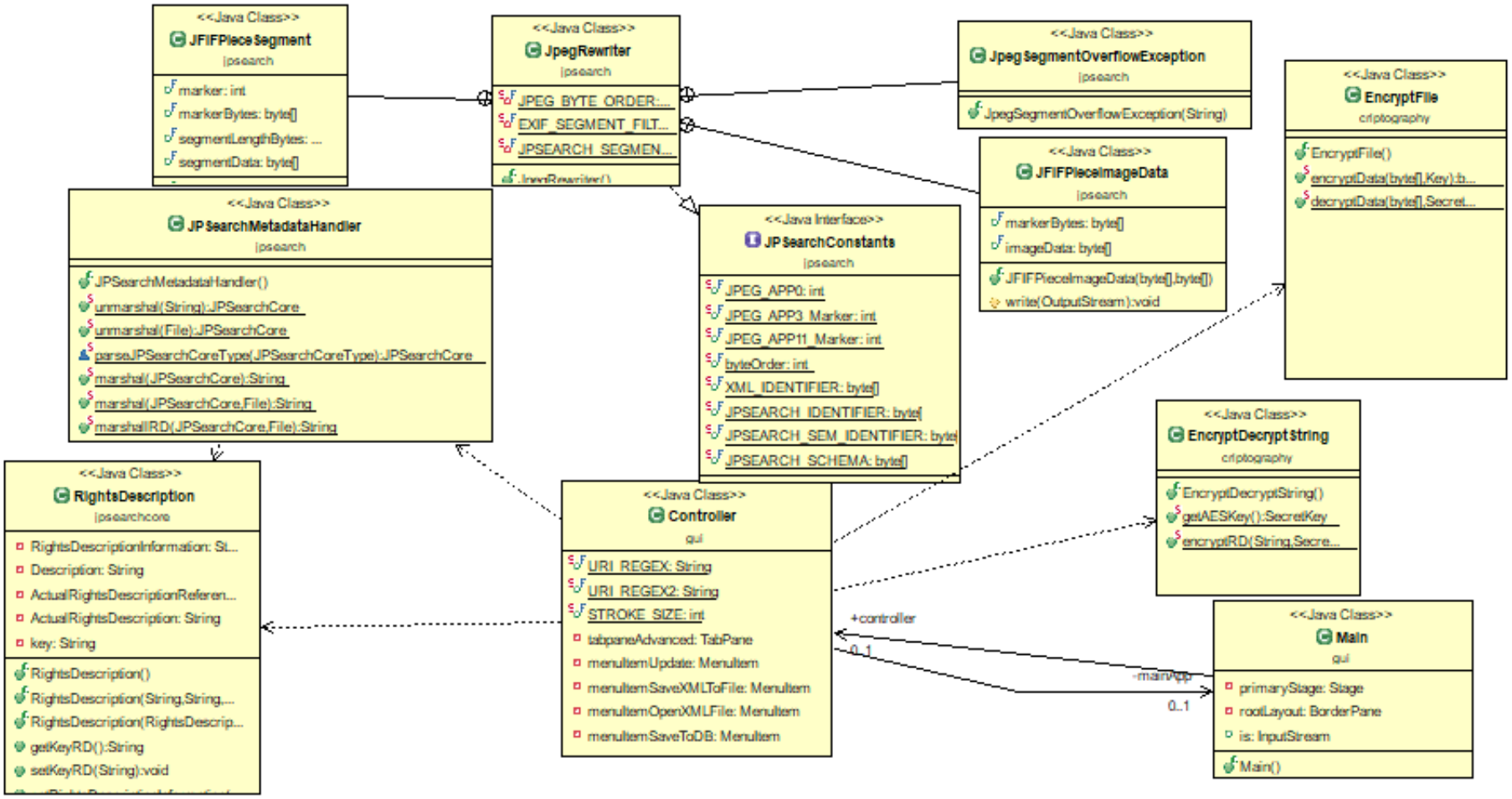
Anexo 3: Diagrama de paquetes *JPSearch Editor*



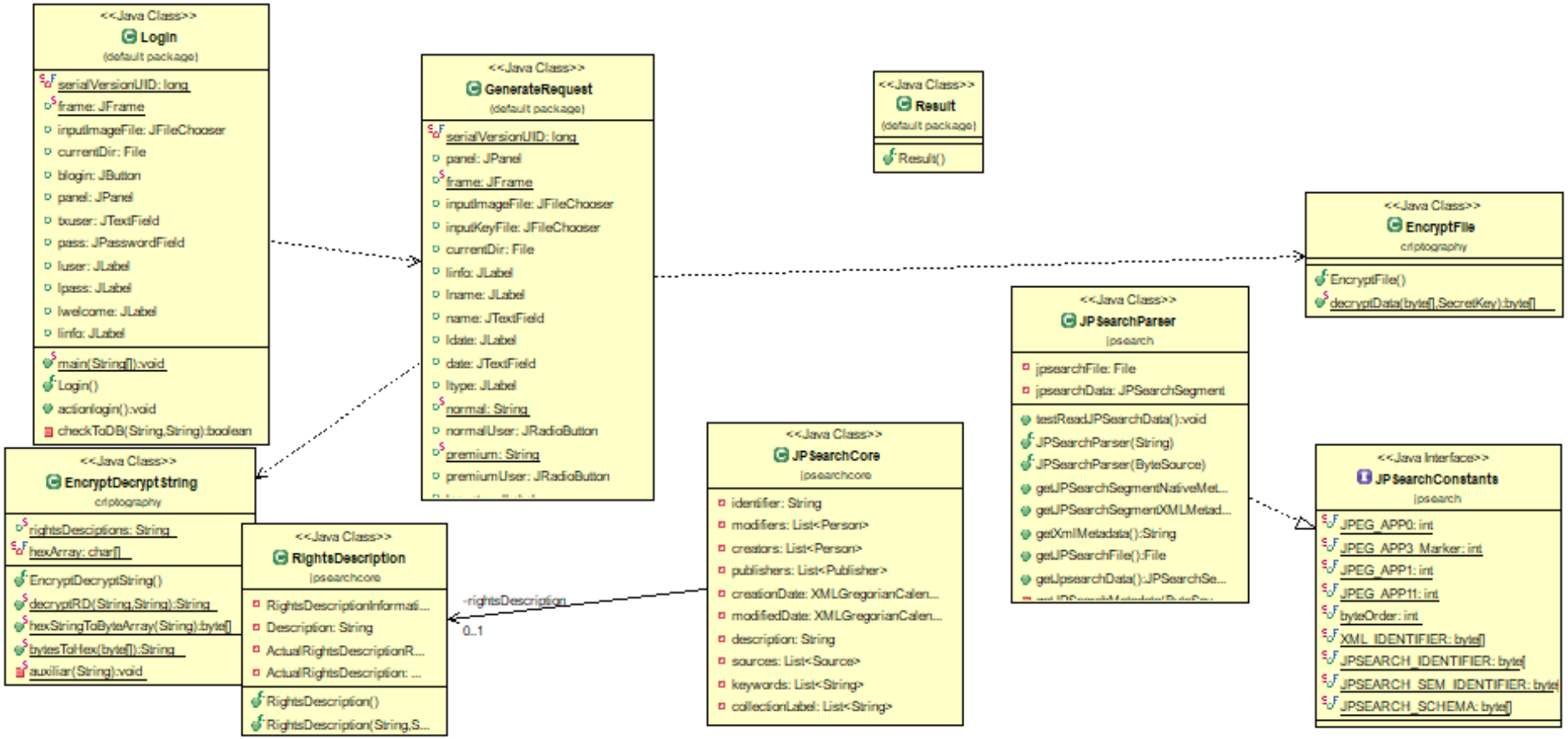
Anexo 4: Diagrama de paquetes *XACML Request Generator*



Anexo 5: Diagrama de clases *JPSearch Editor*



Anexo 6: Diagrama de clases *XACML Request Generator*



Anexo 7: Ejemplo de política de privacidad XACML 2.0

La política de privacidad incluida a continuación describe en lenguaje XACML 2.0 que la imagen titulada Desert.jpg puede ser visualizada por cualquier usuario antes de final de año, en concreto, antes del 01/01/2018.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="urn:isdcm:policyid:1" RuleCombiningAlgId="urn:oasis:names:tc:XACML:1.0:rule-combining-
algorithm:first-applicable">
   <Description>Desert.jpg</Description>
   <Rule Effect="Permit" RuleId="urn:oasis:names:tc:XACML:2.0:ejemplo:Desert">
      <Description>Cualquier usuario puede visualizar la imagen urn:mimage:Desert.jpg antes de final de
año</Description>
      <Target>
         <Subjects>
            <AnySubject />
         </Subjects>
         <Resources>
            <Resource>
               <ResourceMatch MatchId="urn:oasis:names:tc:XACML:1.0:function:regexp-string-match">
                  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:mimage:Desert.jpg</AttributeValue>
                  <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:XACML:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" />
               </ResourceMatch>
            </Resource>
         </Resources>
         <Actions>
            <Action>
               <ActionMatch MatchId="urn:oasis:names:tc:XACML:1.0:function:string-equal">
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">view</AttributeValue>
                  <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:XACML:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" />
               </ActionMatch>
            </Action>
         </Actions>
      </Target>
```

```
      <Condition FunctionId="urn:oasis:names:tc:XACML:1.0:function:and">
         <Apply FunctionId="urn:oasis:names:tc:XACML:1.0:function:date-less-than-or-equal">
            <Apply FunctionId="urn:oasis:names:tc:XACML:1.0:function:date-one-and-only">
               <SubjectAttributeDesignator AttributeId="date" DataType="http://www.w3.org/2001/XMLSchema#date" />
            </Apply>
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date">2018-01-01</AttributeValue>
         </Apply>
      </Condition>
   </Rule>
   <Rule RuleId="urn:oasis:names:tc:XACML:2.0:FinalRule" Effect="Deny" />
</Policy>
```

Anexo 8: Ejemplo de política de privacidad XACML 3.0

La política de privacidad incluida a continuación describe en lenguaje XACML 3.0 que la imagen titulada Desert.jpg puede ser visualizada por cualquier usuario antes de final de año, en concreto, antes del 01/01/2018.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:XACML:3.0:core:schema:wd-17" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
PolicyId="urn:isdcm:policyid:1" RuleCombiningAlgId="urn:oasis:names:tc:XACML:1.0:rule-combining-algorithm:first-
applicable" Version="1.0" xsi:schemaLocation="urn:oasis:names:tc:XACML:3.0:core:schema:wd-17 http://docs.oasis-
open.org/XACML/3.0/XACML-core-v3-schema-wd-17.xsd">
   <Description>Desert.jpg</Description>
   <Rule Effect="Permit" RuleId="urn:oasis:names:tc:XACML:2.0:ejemplo:Desert">
      <Description>Cualquier usuario puede visualizar la imagen urn:mimage:Desert.jpg antes de final de
año</Description>
      <Target>
         <AnyOf>
            <AllOf>
               <!-- Which resource -->
               <Match MatchId="urn:oasis:names:tc:XACML:1.0:function:regexp-string-match">
                  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:mimage:Desert.jpg</AttributeValue>
                  <AttributeDesignator AttributeId="urn:oasis:names:tc:XACML:1.0:resource:resource-id"
Category="urn:oasis:names:tc:XACML:3.0:attribute-category:resource"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false" />
               </Match>
               <!-- Which action -->
               <Match MatchId="urn:oasis:names:tc:XACML:1.0:function:string-equal">
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">view</AttributeValue>
                  <AttributeDesignator AttributeId="urn:oasis:names:tc:XACML:1.0:action:action-id"
Category="urn:oasis:names:tc:XACML:3.0:attribute-category:action" DataType="http://www.w3.org/2001/XMLSchema#string"
MustBePresent="false" />
               </Match>
            </AllOf>
         </AnyOf>
      </Target>
```

```
      <Condition>
         <Apply FunctionId="urn:oasis:names:tc:XACML:1.0:function:date-less-than-or-equal">
            <Apply FunctionId="urn:oasis:names:tc:XACML:1.0:function:date-one-and-only">
               <AttributeDesignator AttributeId="accessDate" Category="urn:oasis:names:tc:XACML:3.0:date"
DataType="http://www.w3.org/2001/XMLSchema#date" MustBePresent="false" />
            </Apply>
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date">2018-01-01</AttributeValue>
         </Apply>
      </Condition>
   </Rule>
   <Rule RuleId="urn:oasis:names:tc:XACML:2.0:FinalRule" Effect="Deny" />
</Policy>
```

Anexo 9: Ejemplo de petición de acceso XACML 2.0

La petición de acceso incluida a continuación describe en lenguaje *XACML 2.0* que el usuario brian, en fecha 29/01/2018, siendo un usuario de perfil normal (no Premium), ubicado en España, desea visualizar la imagen titulada Desert.jpg.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:XACML:1.0:context" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Subject>
      <Attribute AttributeId="urn:oasis:names:tc:XACML:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>brian</AttributeValue>
      </Attribute>
      <Attribute AttributeId="date" DataType="http://www.w3.org/2001/XMLSchema#date">
         <AttributeValue>2018-01-29</AttributeValue>
      </Attribute>
      <Attribute AttributeId="profile" DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>Normal User</AttributeValue>
      </Attribute>
      <Attribute AttributeId="urn:oasis:names:tc:XACML:2.0:subject:country"
DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>Spain</AttributeValue>
      </Attribute>
   </Subject>
   <Resource>
      <Attribute AttributeId="urn:oasis:names:tc:XACML:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>urn:mimage:Desert.jpg</AttributeValue>
      </Attribute>
   </Resource>
   <Action>
      <Attribute AttributeId="urn:oasis:names:tc:XACML:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>view</AttributeValue>
      </Attribute>
   </Action>
</Request>
```

Anexo 10: Ejemplo de petición de acceso XACML 3.0

La petición de acceso incluida a continuación describe en lenguaje *XACML 3.0* que el usuario brian, en fecha 29/01/2018, siendo un usuario de perfil normal (no Premium), ubicado en España, desea visualizar la imagen titulada Desert.jpg.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:XACML:3.0:core:schema:wd-17" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" CombinedDecision="true" ReturnPolicyIdList="true"
xsi:schemaLocation="urn:oasis:names:tc:XACML:3.0:core:schema:wd-17 http://docs.oasis-open.org/XACML/3.0/XACML-core-
v3-schema-wd-17.xsd">
   <Attributes Category="urn:oasis:names:tc:XACML:1.0:subject-category:access-subject">
      <Attribute AttributeId="urn:oasis:names:tc:XACML:1.0:subject:subject-id" IncludeInResult="false">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">brian</AttributeValue>
      </Attribute>
   </Attributes>
   <Attributes Category="urn:oasis:names:tc:XACML:3.0:date">
      <Attribute AttributeId="accessDate" IncludeInResult="false">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date">2018-01-29</AttributeValue>
      </Attribute>
   </Attributes>
   <Attributes Category="urn:oasis:names:tc:XACML:3.0:role">
      <Attribute AttributeId="role" IncludeInResult="false">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Normal User</AttributeValue>
      </Attribute>
   </Attributes>
   <Attributes Category="urn:oasis:names:tc:XACML:3.0:country">
      <Attribute AttributeId="country" IncludeInResult="false">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Spain</AttributeValue>
      </Attribute>
   </Attributes>
   <Attributes Category="urn:oasis:names:tc:XACML:3.0:attribute-category:resource">
      <Attribute AttributeId="urn:oasis:names:tc:XACML:1.0:resource:resource-id" IncludeInResult="false">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">urn:mimage:Desert.jpg</AttributeValue>
      </Attribute>
   </Attributes>
   <Attributes Category="urn:oasis:names:tc:XACML:3.0:attribute-category:action">
```

```
      <Attribute AttributeId="urn:oasis:names:tc:XACML:1.0:action:action-id" IncludeInResult="false">
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">view</AttributeValue>
      </Attribute>
   </Attributes>
</Request>
```