Paquete de Servicios para el Portafirmas Digital de la Universidad de las Ciencias Informáticas

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autor:

Adiel Alfonso Cordovi

Tutores:

Dr.C Ramón Santana Fernández

Ing. Osay González Fuentes

La Habana-Junio 2018

DECLARACIÓN DE AUTORÍA

Por la presente declaro que Adiel Alfonso Cordovi es el único autor de este trabajo y como tal autoriza a la Universidad de las Ciencias Informáticas (UCI) a darle el uso que estime pertinente. Y para que así conste, firmo la presente a los __ días del mes de ___ del año 2018**.**

_____

Adiel Alfonso Cordovi

Autor

_____

Ing. Osay González Fuentes

Tutor

_____

Dr.C Ramón Santana Fernández

Tutor

DEDICATORIA

A mi familia y a todas las personas que de una forma u otra me han apoyado y contribuido a mi formación.

En especial a mis padres y mi hermano que son mi ejemplo a seguir.

AGRADECIMIENTOS

Antes que todo quiero agradecer a mis padres María y Andrés por todo el sacrificio que han hecho para que yo cumpliera mi meta.

Un agradecimiento especial a mi hermano Andreis por el apoyo que dado, y su disposición a ayudarme en cualquier momento. Eres mi modelo a seguir como profesional.

A mis suegros Gladys y Manuel, por estar siempre presentes y preocupados durante todo este proceso.

A mi novia Claudia por todo el cariño y comprensión que me ha dado. Gracias por aparecer en mi vida.

A mis tutores Ramón y Osay por todo lo que me han enseñado y el apoyo dado.

A todas esas personas que me dedicaron su tiempo y tanto me han ayudado en la realización de este ejercicio.

A José por toda su preocupación y paciencia. Gracias por todo lo enseñado.

A mis amigos y mis compañeros de aula, en especial a Yaidel, Baratutis, Pulido, Vladimir y Nardy.

Un agradecimiento especial a mi compañera Anaili por estar ahí en los momentos difíciles.

RESUMEN

La Universidad de las Ciencias Informáticas (UCI) es un centro de altos estudios que hace uso de las Tecnologías de la Información y las Comunicaciones (TIC) en el proceso de informatización de la institución y la sociedad. Actualmente se le da un uso cada vez mayor a la firma digital de documentos en los procesos que se llevan a cabo en la universidad. Debido a esto, en la presente investigación se realizó un estudio de las aplicaciones que facilitan el proceso de autenticación de documentos digitales a nivel nacional e internacional, identificándose que ninguna cumple con ser una aplicación web, de código abierto y que satisfaga las necesidades actuales de la universidad. Para ello fueron utilizados métodos teóricos y empíricos que, de manera general, permitieron conocer el estado actual del proceso en la universidad, para luego realizar una propuesta de solución que responda a esas necesidades. Luego de ser implementada la propuesta de solución se obtuvo un paquete de servicios que publica servicios mediante APIS-REST para el Portafirmas UCI. Este garantiza que el Portafirmas UCI, al consumir estos servicios, firme digitalmente documentos en formato *PDF* haciendo uso de certificados digitales y efectuar el flujo de firmado desde una aplicación web, que garantiza autenticidad, integridad y no repudio de la información que maneja. Se muestran los resultados de un conjunto de pruebas realizadas al paquete de servicios, con el fin de entregar al cliente una solución libre y confiable, que pueda ser utilizada en la universidad.

Palabras clave: autenticidad, certificados digitales, firma digital, integridad, no repudio.

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las organizaciones y empresas presentan un complejo flujo de información donde se generan diferentes tipos de documentos, estos son manejados por sistemas automatizados para así gestionar los disímiles procesos de sus diferentes departamentos. Los documentos que contienen información clasificada solo pueden ser accedidos por el personal autorizado; la modificación o consulta por una persona no autorizada puede crear brechas de seguridad y afectar así el funcionamiento de la institución. El análisis de la seguridad informática se hace imprescindible para agregarle a estos sistemas funcionalidades que minimicen los riesgos ante cualquier ataque informático y así salvaguardar los bienes informáticos de la institución.

Es necesario crear diferentes mecanismos dirigidos a garantizar la confidencialidad y autenticidad de los documentos electrónicos. Una de las áreas de la ciencia encargada de resolver estos problemas, es la criptografía, utilizada en protección de la información con la utilización de cifras o códigos al escribir datos secretos y confidenciales en documentos digitales que circulan por la red. Además de mantener la seguridad del usuario, la criptografía preserva la integridad de la web**,** la autenticación del usuario, la del remitente, el destinatario y de la actualidad del mensaje o del acceso. Existen dos tipos de criptografía: simétrica y asimétrica; ambas proporcionan autenticidad, confidencialidad e integridad a la información. La criptografía asimétrica es la base de la Infraestructura de Clave Pública (PKI), una combinación de hardware, software, políticas y procedimientos de seguridad que permiten la ejecución de operaciones criptográficas como el cifrado y la firma digital. (López, 2010)

Cuando los datos son transmitidos por la red es la firma digital la encargada de garantizar la integridad de la información, mientras el cifrado es el encargado de velar por la confidencialidad de la misma. Pero es el certificado digital, el encargado de velar por la autenticidad del emisor ante el receptor. Cada uno de ellos establecen la base para garantizar la seguridad de un sistema. La integración de sistemas independientes con infraestructuras de clave pública provee, a los mismos, de las herramientas necesarias para velar por la seguridad de la información. (EuroLogic, 2010)

En varios países como España, se utilizan herramientas llamadas portafirmas digitales como solución a los problemas de seguridad que generan la gestión de información clasificada. Los portafirmas contienen una carpeta o cartera donde se almacenan las cartas u otros documentos antes de presentarlos a la firma. En este caso de forma digital, permitiendo firmar los documentos a través de firma electrónica.

La Universidad de la Ciencias Informáticas (UCI), no solo es una universidad, sino también una gran fábrica de producción de software. Atiende contratos a organizaciones y empresas nacionales e internacionales con las cuales se maneja información referente a sus clientes. El acceso o modificación de esta información por una persona no autorizada, afectando a un cliente o a la misma institución, ocasionaría una pérdida de prestigio. La necesidad de establecer sistemas de gestión de documentos digitales, utilizando la firma digital, en la universidad es primordial para el incremento de la seguridad y eficiencia en los procesos de autenticación de documentos oficiales. El uso de la firma digital en los procesos de autenticación de documentos digitales en la universidad, es necesario, para mantener un control sobre qué persona o entidad ha modificado o generado determinada información. Cada usuario o entidad firmante en la universidad posee un certificado digital con el cual realiza la firma digital de documentos. El certificado digital es el documento emitido por la Autoridad Certificadora de la UCI que identifica a un usuario o entidad. Estos certificados son solicitados, publicados y asignados a los usuarios de forma manual, lo que trae consigo demoras e ineficiencia en la solicitud, publicación, así como en la asignación de dichos certificados solicitados.

La herramienta eXcriba, desarrollada en la universidad, es la encargada de la gestión de documentos en los diferentes proyectos. Esta no firma digitalmente documentos, dejando a un lado el tratamiento de la validez de los diferentes documentos generados tanto en los proyectos como en los procesos académicos acometidos en la institución. Las herramientas de escritorio utilizadas para firmar documentos digitales en la universidad no poseen un servidor que centralice la información persistente asociada al flujo de firmado de un documento. La situación actual de la universidad es, que ninguna de las soluciones desplegadas para la gestión de la documentación en la universidad, gestiona los procesos a autenticación de documentos digitales.

La utilización de un sistema al nivel de los portafirmas usados actualmente, como solución a los problemas existentes de autenticación de documentos digitales, sería un gran avance. Aunque para evitar brechas de seguridad existe un sistema de clave pública (*PKI*) en el nodo central. Esta herramienta permite la autenticación de un usuario frente a otro a través de claves criptográficas, que permite cifrar y descifrar mensajes y de firmar digitalmente la información. El sistema no está totalmente automatizado y presenta deficiencias: los usuarios deben descargar los documentos a firmar y luego enviarlos o a otro usuario para firmarlo. No atiende las peticiones de firmado digital, lo que ocasiona retrasos en los diferentes procesos existentes en la institución al no validar la fecha límite de la petición. Se requiere firmar digitalmente grandes

volúmenes de información de forma rápida y sencilla, para así evitar el engorroso proceso de firmar manualmente documentos por separado. Muchos usuarios firman documentos digitales por sus medios mediante diferentes herramientas como el *Adobe Reader*. Al no existir un sistema centralizado que resuelva esta problemática, no existe un control real sobre el proceso de autenticación de documentos digitales en la universidad.

Ante esta situación surge la siguiente interrogante como **problema de investigación**: ¿Cómo mejorar el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas?

Para dar solución a este problema, se tomará como **objeto de estudio** el proceso de firma de documentos digitales, donde enmarcará el **campo de acción** el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.

Teniendo en cuenta lo anteriormente planteado, el **objetivo general** de esta investigación es: Desarrollar un Paquete de Servicios para el Portafirmas Digital que permita mejorar el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.

Para cumplir con el objetivo general se derivan los siguientes **objetivos específicos:**

- ✓ Analizar los referentes teórico-metodológicos asociados a la firma digital y el proceso de autenticación de documentos digitales.
- ✓ Diseñar el Paquete de Servicios para el Portafirmas Digital teniendo en cuenta las funcionalidades con las que debe contar.
- ✓ Implementar las funcionalidades correspondientes al Paquete de Servicios para el Portafirmas Digital de la UCI.
- ✓ Verificar el funcionamiento del Paquete de Servicios para el Portafirmas UCI.
- ✓ Validar el funcionamiento del Paquete de Servicios para el Portafirmas UCI.

Para dar solución al **problema de la investigación** se define la siguiente **hipótesis:** el desarrollo del Paquete de Servicios para el Portafirmas Digital mejorará el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.

Identificándose como **variable independiente:** Paquete de Servicios para el Portafirmas Digital y como **variable dependiente:** proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.

Tabla 1: Operacionalización de las variables. (*Elaboración propia*)

Variables	Dimensiones	Indicadores	Sub-Indicadores	Unidad de medidas
Paquete de Servicios para el Portafirmas Digital.	UCI	Adecuación funcional	Pertinencia funcional	(%)
		Fiabilidad	Disponibilidad	(%)
Proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.	Usuarios que utilizan el portafirmas	Tiempo de firmado de documentos	Alta	1-5 (seg)
			Media	6-10 (seg)
			Baja	11-15 (seg)

Métodos de Investigación Científica

Entre los métodos teóricos que se emplearán para nuestra investigación está:

- ✓ **Histórico-Lógico:** Permitió a partir de la documentación consultada referente a la autenticación de documentos digitales, determinar las tendencias actuales de los portafirmas digitales.
- ✓ **Analítico-Sintético:** Posibilitó un mejor entendimiento del funcionamiento de la PKI.
- ✓ **Modelación:** Facilitó la representación mediante diagramas de las características, procesos y componentes de la solución propuesta, así como la relación existente entre ellos.
- ✓ **Hipotético-deductivo:** Permitió proponer líneas de trabajo a partir de resultados parciales obtenidos durante el proceso de investigación.

Entre los métodos empíricos:

- ✓ **Observación:** Método utilizado a lo largo del estudio del funcionamiento del proceso de autenticación de documentos digitales que se desarrolla en la UCI a partir de una planificación previa

donde se precisaron los elementos que serían objeto de observación.

- ✓ **Experimental:** método utilizado en la comprobación de la utilidad de los resultados obtenidos a partir de la introducción de datos ficticios.

El presente trabajo de diploma está compuesto por un resumen, una introducción, tres capítulos, conclusiones, recomendaciones y las referencias bibliográficas. A continuación, se describen los principales aspectos abordados en cada uno de los capítulos:

Capítulo 1: contiene una base teórica, donde su objetivo es la compresión del problema planteado. Se exponen los conceptos asociados al dominio de la investigación y se realiza un análisis de la teoría para el diseño de la propuesta de solución. Se hace referencia a las tendencias y tecnologías actuales usadas durante el desarrollo de la propuesta de solución. Queda definido el entorno de desarrollo que se utilizará para dar solución al problema planteado en la investigación.

Capítulo 2: de acuerdo a lo establecido en la metodología utilizada, se identifican los requisitos funcionales y los no funcionales, se obtienen los diagramas de clases del diseño, diagramas de despliegue, lo referente al estilo arquitectónico del paquete de servicios, así como los patrones de diseño utilizados y el entorno de despliegue propuesto para desplegarlo. Se plantea la elaboración del modelo de dominio para una mejor comprensión de los conceptos relacionados en el negocio.

Capítulo 3: se describe la arquitectura y los patrones utilizados en la implementación de la propuesta de solución, así como los diferentes diagramas utilizados para presentar la información. Se realizan las pruebas funcionales y de seguridad para comprobar el correcto funcionamiento del paquete de servicios elaborado.

Posibles resultados:

Con la realización de la presente investigación, se pretende obtener un paquete de servicios que permita brindar servicios para el Portafirmas Digital de la UCI. Estos servicios podrán ser utilizados por cualquier otro sistema desplegado en la universidad que participe en los procesos de autenticación de documentos digitales.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se aborda sobre las bases teóricas de los portafirmas y la firma digital en los procesos de autenticación de documentos digitales. Se exponen los conceptos asociados al dominio de la investigación y se realiza un análisis del estado del arte de los portafirmas a nivel nacional e internacional. Se presenta el entorno de desarrollo a utilizar para dar solución al problema planteado.

1.1 CONCEPTOS ASOCIADOS AL DOMINIO DE LA INVESTIGACIÓN

1.1.1 Criptografía

La criptografía es el estudio de las técnicas matemáticas relacionadas con aspectos de la seguridad de la información, como la confidencialidad, la integridad de los datos, la autenticación de entidades y la autenticación de orígenes de datos. La criptografía no es el único medio para proporcionar seguridad de la información, sino más bien un conjunto de técnicas. (Alfred J. Menezes, 1996)

Según (PGP Corporation, 2002) la Criptografía es la ciencia encargada de diseñar funciones o dispositivos, capaces de transformar mensajes legibles o en claro a mensajes cifrados de tal manera que esta transformación (cifrar) y su transformación inversa (descifrar) sólo pueden ser factibles con el conocimiento de una o más claves.

Claves criptográficas: Las claves criptográficas son un conjunto de datos o información manejada y gestionada por el usuario para realizar operaciones criptográficas que posibilitan las operaciones que se generan en el momento de la solicitud del certificado y quedan unidas inequívocamente al titular de las mismas. (Ugarte, 2013)

1.1.2 Criptosistemas

Criptosistemas de clave secreta o simétricos: La criptografía simétrica o de clave secreta es aquella que utiliza algún método matemático llamado sistema de cifrado para cifrar y descifrar un mensaje utilizando únicamente una clave secreta. (PGP Corporation, 2002)

Figura 1: Criptosistema simétrico (*Elaboración propia*)

Criptosistemas asimétricos: En este sistema se utilizan dos claves complementarias llamadas clave privada y clave pública. Lo que está codificado con una clave privada necesita su correspondiente clave pública para ser decodificado y viceversa, lo codificado con una clave pública sólo puede ser decodificado con una clave privada. La clave privada sólo es conocida por el propietario mientras que la clave pública se comparte. El problema que plantea este sistema es la lentitud de operación y la confianza en la clave pública. (Alfred J. Menezes, 1996)

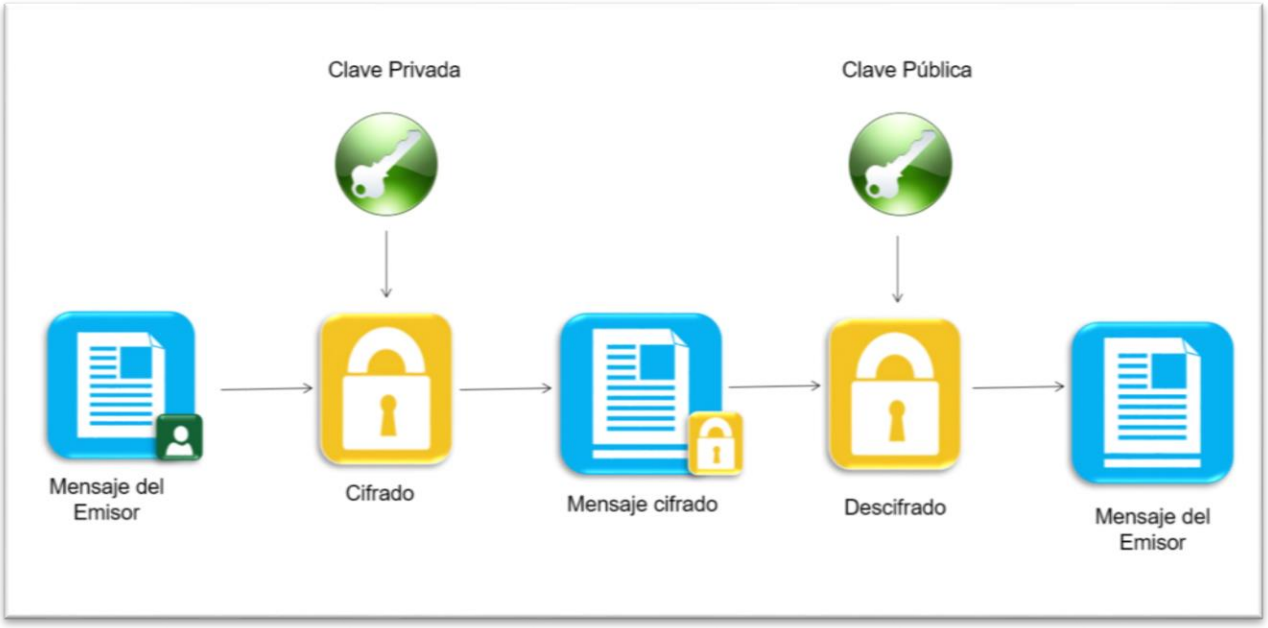

Figura 2: Criptosistema asimétrico. (*Elaboración propia*)

Funciones resumen o *hash*: Las funciones *hash* son funciones que mapean una entrada de longitud arbitraria a una cadena de longitud fija, el *hashcode*. Si estas asignaciones satisfacen algunas condiciones criptográficas adicionales, se pueden usar para proteger la integridad de la información. Otras aplicaciones criptográficas en las que las funciones *hash* son útiles son la optimización de los esquemas de firma digital, la protección de las frases de contraseña y el compromiso con una cadena sin revelarla. (K.U.Leuven, 1993)

RSA

En la criptografía *RSA*, tanto las claves públicas como las privadas pueden encriptar un mensaje; la clave opuesta a la utilizada para encriptar un mensaje se usa para descifrarla. Este atributo es una de las razones por las que *RSA* se ha convertido en el algoritmo asimétrico más utilizado: proporciona un método para garantizar la confidencialidad, integridad y autenticidad de las comunicaciones electrónicas y el almacenamiento de datos. (Stallings, 2005)

Advanced Encryption Standard (AES)

Algoritmo de cifrado en bloques capaz de manejar bloques de 128 *bits*, utilizando claves de tamaño 128, 192 y 256 *bits*. Fue pensado para ser fácil de implementar en hardware y software, así como en entornos

restringidos (por ejemplo, en una tarjeta inteligente) y ofrece buenas defensas contra diversas técnicas de ataque. (Stallings, 2005)

Hash-based message authentication code (HMAC)

El código de autenticación de mensaje basado en *hash* proporciona al servidor y al cliente una clave privada conocida solo por ese servidor específico y ese cliente específico. El cliente crea un *HMAC* único, o *hash*, por solicitud al servidor, mezclando los datos de la solicitud con las claves privadas y enviándolo como parte de una solicitud. (TechTarget, 2018)

1.1.3 Infraestructura de clave pública (*Public Key Infrastructure ,PKI*)

Certificados digitales: Un certificado de clave pública es un punto de unión entre la clave pública de una entidad y uno o más atributos referidos a su identidad. El certificado garantiza que la clave pública pertenece a la entidad identificada y que la entidad posee la correspondiente clave privada. (Talens-Oliag, 2006)

Los certificados de clave pública se denominan comúnmente Certificado Digital, *ID* Digital o simplemente certificado. La entidad identificada se denomina sujeto del certificado o subscriptor (si es una entidad legal, por ejemplo, una persona). Los certificados digitales sólo son útiles si existe alguna Autoridad Certificadora (*Certification Authority o CA*) que los valide, ya que si uno se certifica a sí mismo no hay ninguna garantía de que su identidad sea la que anuncia y, por lo tanto, no debe ser aceptada por un tercero que no lo conozca. Es importante ser capaz de verificar que una autoridad certificadora ha emitido un certificado y detectar si un certificado no es válido. Para evitar la falsificación de certificados, la entidad certificadora después de autenticar la identidad de un sujeto, firma el certificado digitalmente. Los certificados digitales proporcionan un mecanismo criptográfico para implementar la autenticación; también proporcionan un mecanismo seguro y escalable para distribuir claves públicas en comunidades grandes. (Talens-Oliag, 2006)

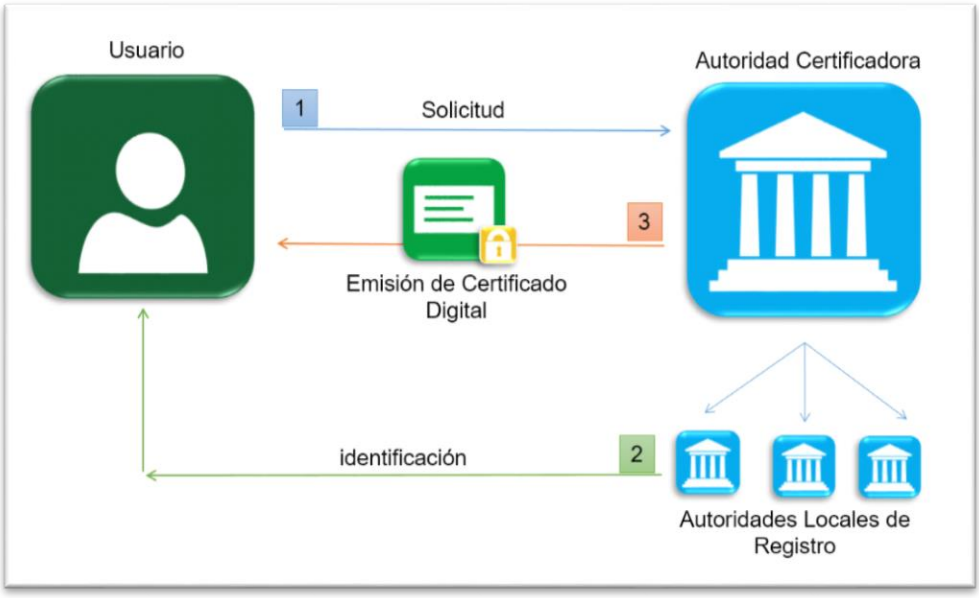

Figura 3: Proceso de Obtención de Certificados. (*Elaboración propia*)

Certificado digital: es la certificación electrónica generada por una autoridad de certificación, que vincula unos datos de verificación de firma a un signatario y confirma su identidad. El certificado tiene una validez determinada y un uso concreto. (Tarrats, 2017) Aunque también es definido como un documento, firmado electrónicamente, que autentica la relación de una clave pública con un titular participante en procesos telemáticos que precisan identificación y autenticación, confidencialidad, integridad, validez y no repudio. (Alfred J. Menezes, 1996)

El estándar más utilizado para estos certificados es el *X.509v3* como versión más actualizada de los certificados *X.509*. El certificado *X.509v3* es un estándar de la Unión Internacional de Telecomunicaciones (UIT), este es la pieza central de una *PKI* y es donde se especifican formatos de atributos estándar para certificados de claves públicas y un algoritmo de validación de la ruta de certificación. (CISCO, 2016)

Entre las extensiones de archivo de certificados *X.509v3* que existen, en el sistema se utilizará el que está establecido por política de la universidad que es el *p12.* Esta extensión corresponde al formato de archivo *PKCS #12* pertenece a la familia de estándares llamados *Public-Key Cryptography Standards (PKCS*) publicados por *RSA Laboratories*. Este estándar especifica un formato portable para almacenar y transportar certificados, claves privadas y secretos varios. Es un formato muy útil para realizar operaciones de gestión de certificados y es soportado por la mayoría de los navegadores. Tiene la ventaja de que es capaz de

almacenar el certificado con su correspondiente clave, con el certificado raíz de la autoridad certificadora y otros certificados posibles de la cadena en un único fichero. (Parkinson, 2014)

Infraestructura de Clave Pública: Una Infraestructura de Clave Pública (*PKI*) es una infraestructura que proporciona servicios de seguridad a las aplicaciones informáticas y sus usuarios mediante el uso de técnicas de clave pública de un modo transparente al usuario. Entre los beneficios más destacados que proporciona una *PKI* se incluyen el ahorro de costes y la interoperabilidad de soluciones. (Lapuente, 2011)

Aunque también puede ser definida como: Los estándares y servicios que facilitan el uso de la criptografía y los certificados en un entorno de red. (Tarrats, 2017)

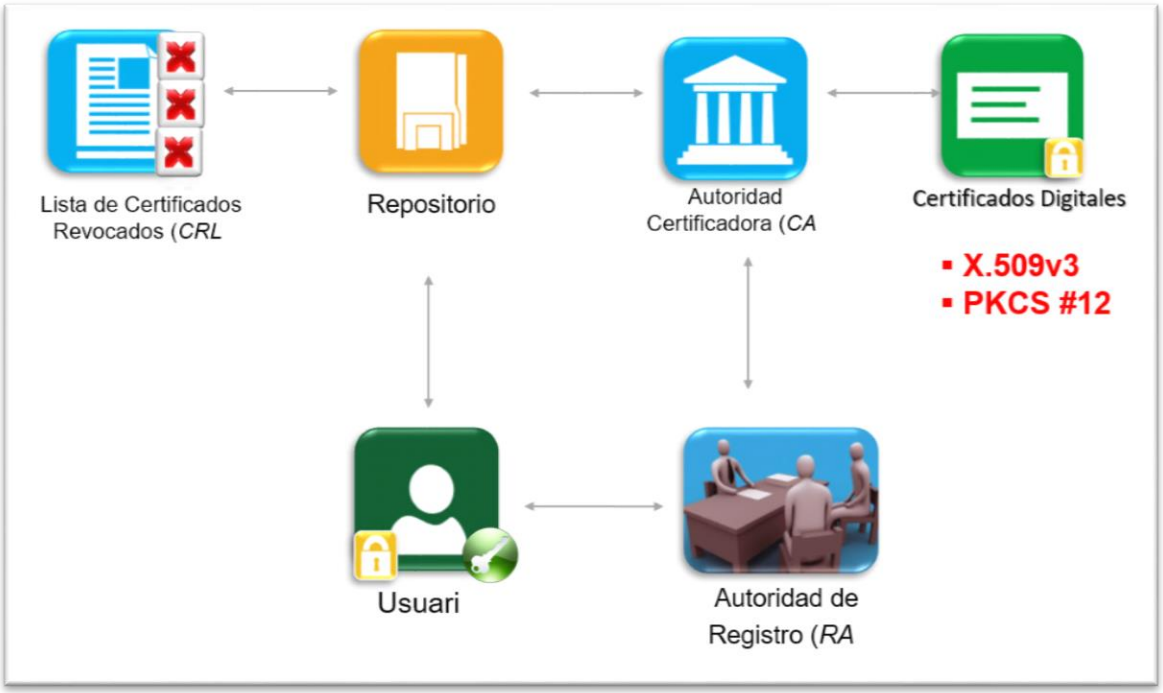

Figura 4: Infraestructura de Clave Pública. (*Elaboración propia*)

Firma electrónica: La firma electrónica es el conjunto de datos relativos a una persona consignados en forma electrónica, y que junto a otros o asociados con ellos, pueden ser utilizados como medio de identificación del firmante, teniendo el mismo valor que la firma manuscrita. Permite que tanto el receptor como el emisor de un contenido puedan identificarse mutuamente, para evitar que terceras personas

intercepten esos contenidos y que los mismos puedan ser alterados, o alguna de las partes pueda "repudiar" la información que recibió de la otra, que inicialmente fue aceptada. (Ugarte, 2013)

Esta permite la identificación del usuario firmante y ha sido creada por medios que éste mantiene bajo su exclusivo control, de manera que está vinculada únicamente al mismo y a los datos a los que se refiere, lo que permite que la detección de cualquier modificación sobre éstos. (Tarrats, 2017)

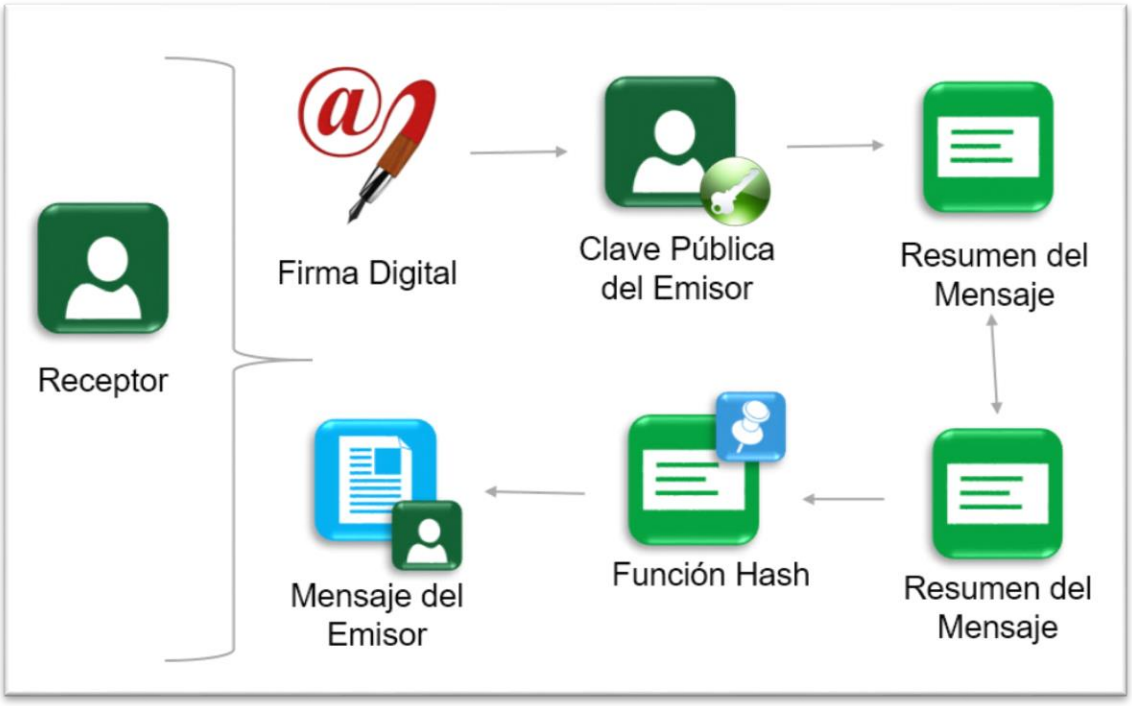

Figura 5: Proceso de Firma de un Documento Digital. (Elaboración propia)

Figura 6: Proceso de Verificar Firma de un Documento Digital. (*Elaboración propia*)

Autoridad Certificadora (*CA. Certification Authority*): La Autoridad Certificadora es una entidad de confianza cuyo objeto es garantizar la identidad de los titulares de certificados y su correcta asociación a las claves de firma electrónica. Las solicitudes por parte de los usuarios de certificados digitales son recibidas y procesadas por la autoridad certificadora la cual consulta con una autoridad de registro con el fin de determinar si acepta o rehúsa la petición de certificado. La autoridad certificadora es la encargada final de emitir el certificado, y llevar a cabo la renovación de los certificados cuando es solicitada. Gestiona la Lista de Revocación de Certificados (*CRLs. Certificates Revocation List*), esta lista se debe mantener correctamente actualizada, ya que debe contener todos los certificados para los que se solicita la revocación por causas como robo, pérdida o suspensión de identidades digitales. La autoridad de certificación es la responsable de proporcionar un servicio de *backup* y archivo seguro de claves de cifrado. (Ugarte, 2013)

Una autoridad certificadora es una organización fiable que acepta solicitudes de certificados de entidades, las valida, genera certificados y mantiene la información de su estado. (Talens-Oliag, 2006)

Las labores de una *CA* son (Talens-Oliag, 2006):

- ✓ **Admisión de solicitudes.** Un usuario rellena un formulario y lo envía a la *CA* solicitando un certificado. La generación de las claves pública y privada son responsabilidad del usuario o de un sistema asociado a la *CA*.

- ✓ **Autenticación del sujeto.** Antes de firmar la información proporcionada por el sujeto, la *CA* debe verificar su identidad. Dependiendo del nivel de seguridad deseado y el tipo de certificado se deberán tomar las medidas oportunas para la validación.
- ✓ **Generación de certificados.** Después de recibir una solicitud y validar los datos la *CA* genera el certificado correspondiente y lo firma con su clave privada. Posteriormente lo manda al subscriptor y, opcionalmente, lo envía a un almacén de certificados para su distribución.
- ✓ **Distribución de certificados.** La entidad certificadora puede proporcionar un servicio de distribución de certificados para que las aplicaciones tengan acceso y puedan obtener los certificados de sus subscriptores.
- ✓ **Anulación de certificados**. Al igual que sucede con las solicitudes de certificados, la *CA* debe validar el origen y autenticidad de una solicitud de anulación. La *CA* debe mantener información sobre una anulación durante todo el tiempo de validez del certificado original.

Autoridad de Registro (*RA. Registration Authority*): El propósito de una *RA* es verificar el contenido de los certificados en lugar de la *CA*. De igual modo que una *CA*, una *RA* es una colección de hardware, software y las personas que lo operan. Cada *CA* mantiene una lista de *RAs* acreditadas y verificando la firma de una *RA* en un mensaje una *CA* puede estar segura de que una *RA* acreditada proporcionó la información. Se trata de una entidad de confianza que gestiona el registro de usuarios y sus peticiones de certificación/revocación, así como las respuestas a dichas peticiones de certificado. La autoridad de registro es la entidad que indica a la *CA* si se debe emitir un certificado autorizando para asociar una clave pública y el titular de un certificado. (Lapuente, 2011)

Desde la Autoridad de Registro también se gestiona el ciclo de vida de un certificado:

- ✓ Revocación (por pérdida, robo o cancelación).
- ✓ Expiración (cumplida fecha de vencimiento).
- ✓ Renovación (extensión del periodo de validez del certificado, respetando el plan de claves).
- ✓ Re-emisión del par claves del usuario.
- ✓ Actualización de datos del certificado.

Listas de Revocación (*CRL*. *Certificate Revocation List*): Las *CRLs* son estructuras de datos firmadas que contienen una lista de certificados revocados. La firma digital agregada en la *CRL* proporciona mecanismos de autenticidad e integridad. Siempre que las políticas lo permitan las *CRLs* pueden ser almacenadas en memoria y facilitar la verificación de certificados *off-line*. La lista de revocación está firmada electrónicamente por la autoridad de certificación e indica los certificados que han sido revocados antes de que estos expiren. (Tarrats, 2017)

Estas son archivos que contienen las listas con los números de serie de los certificados que han sido revocados para la validación del estado del certificado firmada digitalmente por una *CA*. Una necesidad importante respecto a estas listas es su constante actualización de forma que cuando un usuario necesite descargar la lista de certificados revocados para comprobar un certificado, el listado de certificados revocados esté lo más actualizado posible.

Los certificados tienen un periodo de validez que va de unos meses a unos pocos años. Durante el tiempo que el certificado es válido la entidad certificadora que lo generó mantiene información sobre el estado de ese certificado.

La información más importante que guarda es el estado de anulación, que indica que el periodo de validez del certificado ha terminado antes de tiempo y el sistema que lo emplee no debe confiar en él. Las razones de anulación de un certificado son varias: la clave privada del sujeto se ha visto comprometida, la clave privada de la *CA* se ha visto comprometida o se ha producido un cambio en la afiliación del sujeto (por ejemplo, cuando un empleado abandona una empresa).

Para el despliegue de la infraestructura se precisan los siguientes componentes (Tarrats, 2017):

Autoridad de Certificadora (*CA*): La *CA* emite certificados para las partes que intervienen, en definitiva, da fe de quien nos presenta una clave pública es quien dice ser. La *CA* también mantiene las listas de revocación de certificados para resolver los casos de robo, pérdida o suspensión de claves privadas. La seguridad de la *CA* es crítica; un problema de seguridad que afecte a la *CA* puede afectar a toda la infraestructura existente.

Directorio: El directorio es la base de datos donde se publican los certificados, están disponibles para todas las entidades y este guarda otros datos como las listas de revocación.

Sistema de revocación de certificados: Aunque sea un servicio asociado a la autoridad de certificación, éste puede suministrarse por otra entidad.

Actualización, históricos y copias de claves: Son los componentes que permiten la renovación del certificado y el uso de claves antiguas. En los sistemas donde intervienen datos cifrados hay que suministrar el servicio de recuperación de claves.

Soporte para el no repudio: La protección de las claves privadas puede ser crítica para el no repudio de las firmas digitales realizadas. Los sistemas basados en tarjetas criptográficas son los que ofrecen las mayores garantías. Estos componentes deben existir y pueden estar gestionados por la propia entidad bancaria, un consorcio u otra entidad externa.

1.1.4 Portafirmas:

Un portafirmas es una carpeta donde se incluyen los documentos a firmar. El portafirmas digital funciona exactamente igual, pero a través de una plataforma colaborativa donde se firman los documentos de manera electrónica. El Portafirmas Digital es una plataforma centralizada, donde independientemente el origen del documento a firmar, el usuario firmante acumulará todas las peticiones de firma sin necesidad de ir accediendo a los distintos sistemas de tramitación.

No es necesario registrarse en la plataforma, cualquier persona que reciba un documento para firmar tiene a su disposición un buzón personal de firma. Pero, si imprescindible poseer un certificado digital o *DNI* electrónico para poder efectuar la firma que tendrá plena validez jurídica.

Los portafirmas digitales son herramientas destinadas a facilitar a los órganos y unidades administrativas el uso de la firma digital de documentos, procedentes de diferentes sistemas de información independientes, con la consiguiente agilización de la actividad administrativa. Son herramientas de usuario final que utilizan y proveen servicios de autenticación y firma digital. Estas verifican siempre que las normativas jurídicas estipulen el mismo valor que la firma manuscrita (Evaluación y revisión, 2009)

1.2 PORTAFIRMAS UTILIZADOS A NIVEL INTERNACIONAL

ViaFirma *Inbox*

ViaFirma *Inbox* es una aplicación web basada en los patrones de la Arquitectura Orientada a Servicios (*SOA. Service Oriented Architecture*), está desarrollada en *Java* y es multiplataforma tanto en el cliente

como en el servidor. Es un software con licencia privativa. Sus principales características son (Viafirma, 2017) :

- ✓ Posibilidad de firmar documentos desde cualquier ubicación.
- ✓ Posibilita el envío de peticiones de firmado a direcciones de correo electrónico de usuarios que no se encuentran en el sistema, realizando el registro en el momento del acceso del mismo.
- ✓ Servicios de verificación de copias auténticas mediante el uso de la firma digital.
- ✓ Facilita la auditoría del sistema.
- ✓ Permite la búsqueda de documentos.

AutoFirma (@firma)

La Plataforma @firma es la solución tecnológica en la que se basa la implementación de la plataforma de validación y firma electrónica del Ministerio de la Presidencia de España. Es un producto robusto e integral, compuesto por diferentes productos y servicios. Es una solución basada en software libre y estándares abiertos.

El componente Cliente @firma es un módulo que se distribuye de forma independiente a la Plataforma y que permite a los usuarios realizar el proceso de firma digital de documentos en sus máquinas locales. Es una herramienta de firma electrónica que se ejecuta en cliente (*PC* del usuario) basada en *java*. Esto es así para evitar que la clave privada asociada a un certificado digital tenga que viajar por la red. El cliente es distribuido bajo licencia *EUPL* (*European Public License*), *GPL* (*GNU General Public License*). No permite firmar lotes de documentos. (Port@firmas, 2017)

Port@firmas, Universidad de Sevilla

Porta@firma es una herramienta web ofrecida por la Junta de Andalucía e incorporada a la plataforma de tramitación electrónica de la Universidad de Sevilla, con la intención de realizar la firma electrónica de documentos procedentes del sistema de información ESTELA o de peticiones generadas desde el propio portafirmas por usuarios autorizados. Es compatible con los Sistemas Operativos más utilizados, *Windows* (*XP* o superior), *Linux* (*Ubuntu* y *Guadalinex*) y *Mac OS* X. Está distribuido bajo licencia de código abierto, pero solo puede firmar documentos provenientes de un sistema específico. La herramienta ofrece las siguientes posibilidades (Portafirmas, 2017):

- ✓ Posibilidad de firma masiva de documentos (hasta 50 documentos).
- ✓ Posibilidad de firmar documentos desde cualquier ubicación.
- ✓ Avisos de correo al firmante informándole si tiene algún documento pendiente de firma.
- ✓ Mayor simplicidad tanto en el acceso como en el uso.
- ✓ Posibilidad de acceso al historial de documentos firmados.

Portafirmas, Junta de Andalucía

Portafirmas es un gestor centralizado de documentos que permite la firma digital de éstos de forma sencilla por parte de los usuarios, así como la integración de la firma en terceras aplicaciones ya existentes. Es una herramienta web empleada bajo licencia privativa, sin embargo, las tecnologías empleadas en su desarrollo, así como el análisis de su arquitectura, pueden servir de apoyo para la presente investigación. La herramienta reúne las siguientes funcionalidades (Port@firmas, 2017):

- ✓ Petición de firma a un cargo o puesto de trabajo, el cual puede estar ostentado por una o varias personas.
- ✓ Posibilidad de firmar los documentos identificándose como personas o cargos en Portafirmas.
- ✓ Posibilidad de crear y modificar una línea de firma que puede ser en Cascada o Jerárquica, en Paralelo, o de Primer firmante.
- ✓ Soporta distintos formatos de firmas: *CAdES, XAdES* y *PAdES* (o *PDF*).
- ✓ Ofrece un servicio llamado “Informe de firma” que le añade al pie de un documento un detalle del proceso de firma y un código que permite la verificación del mismo.
- ✓ Rechazo fehaciente de un documento transmitiendo mediante una firma la negativa a su contenido.

1.3 PORTAFIRMAS A NIVEL NACIONAL

UCI PDFSigner (Bondartchuk, 2009)

Es una aplicación desarrollada como resultado de un trabajo de diploma del centro CISED que permite la firma de documentos *PDF*. Tiene implementadas las siguientes funcionalidades que pueden servir de apoyo a la presente investigación:

- ✓ Permite gestionar archivos *PDF* seleccionando la ruta donde se encuentran y todos los que desee

firmar.

- ✓ Brinda la posibilidad de gestionar los certificados digitales desde una ruta en la computadora o a través de una tarjeta inteligente, siendo imprescindibles en ambos casos la contraseña de acceso al certificado.
- ✓ Soporta la firma de documentos *PDF* individuales o en lotes.
- ✓ Posibilidad de gestionar la apariencia de la firma digital en el documento y el servicio de sellado de tiempo.
- ✓ Permite gestionar la configuración de certificados, de la apariencia de la firma, del servidor de sellado de tiempo, de la conexión de red y de otras generales.

1.4 RESULTADO DEL ANÁLISIS:

Comparación de los sistemas homólogos detectados en cuanto a licencia y tecnología WEB:

- ✓ **Licencia:** se refiere al estado jurídico de la aplicación en cuanto a su uso, modificación y distribución, esta puede ser pública, en aquellas que no necesitan un pago para ser utilizadas; o privativa, en aquellas que si lo requieren.
- ✓ **WEB:** se refiere a si la herramienta es una aplicación web o no.

Tabla 2: Sistemas homólogos. (*Elaboración propia*)

Herramienta	Licencia	Web
ViaFirma Inbox	privada	sí
AutoFirma	pública	no
Port@firmas, Universidad de Sevilla	pública	sí
Portafirmas, Junta de Andalucía	privada	sí
UCI PDFSigner	pública	no

A partir del análisis realizado se arriba a las siguientes conclusiones:

- ✓ Las herramientas ViaFirma *Inbox* y Portafirmas de la Junta de Andalucía son privativas y su

aplicación en Cuba resulta engorrosa a partir de los altos costos por concepto de licencia y soporte de las tecnologías que utilizan.

- ✓ La herramienta AutoFirma está desarrollada bajo licencia pública, pero es de uso privado en España y no es una herramienta web.
- ✓ La herramienta Port@firmas de la Universidad de Sevilla es pública y web, pero solo permite firmar documentos desde un sistema de información específico y no permite adaptaciones para otras instituciones.
- ✓ Cada una de las herramientas sirvieron como base para seleccionar la tecnología con la que se desarrollará el paquete de servicios y contribuyeron a una mejor comprensión de las principales funcionalidades con las que debe cumplir un paquete de servicios de este tipo.
- ✓ Ninguna de estas herramientas puede ser utilizada en la propuesta de solución al no responder a las necesidades de la universidad.

1.5 ENTORNO DE DESARROLLO DE LA PROPUESTA DE SOLUCIÓN

Durante el desarrollo del paquete de servicios se utilizarán tecnologías para el desarrollo de la propuesta de solución. A continuación, se describirán los lenguajes de programación a utilizar, los *IDEs* de desarrollo, el gestor y servidor de bases de datos. Se explica la metodología a utilizar y los estándares específicos como vía para mejorar el proceso de desarrollo.

1.5.1 Lenguajes de Modelado

UML 2.1 (*Unified Modeling Language*)

Cualquier rama de ingeniería o arquitectura ha encontrado útil desde hace mucho tiempo la representación de los diseños de forma gráfica. Desde los inicios de la informática se han estado utilizando distintas formas de representar los diseños de una forma más bien personal o con algún modelo gráfico. La falta de estandarización en la manera de representar gráficamente un modelo impedía que los diseños gráficos realizados se pudieran compartir fácilmente entre distintos diseñadores. Se necesitaba un lenguaje no sólo para comunicar las ideas a otros desarrolladores sino también para servir de apoyo en los procesos de análisis de un problema. Con este objetivo se creó el Lenguaje Unificado de Modelado (*UML*).

El lenguaje *UML* tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, hasta la implementación y configuración con los diagramas de despliegue. (El Lenguaje Unificado de Modelado, 2002)

UML es un lenguaje de modelado visual que se usa para visualizar, construir, especificar y documentar artefactos de un sistema de software. Captura decisiones y conocimientos de los sistemas que se deben construir. Se usa para diseñar, entender, mostrar, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. (Rumbaugh, 2000)

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones según (El Lenguaje Unificado de Modelado, 2002):

- ✓ **Visualizar:** *UML* permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- ✓ **Especificar:** *UML* permite especificar cuáles son las características de un sistema antes de su construcción.
- ✓ **Construir:** A partir de los modelos especificados se pueden construir los sistemas diseñados.
- ✓ **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.5.2 Lenguajes del lado del Servidor

JAVA 8

Java es un lenguaje de desarrollo de propósito general, y como tal es válido para realizar todo tipo de aplicaciones profesionales (Sun Microsystems, 2016).

Java es un lenguaje de programación de alto nivel y orientado a objetos independiente del hardware en que se ejecuta. Esto último se consigue gracias a la máquina virtual de *Java*, para que una aplicación *Java* funcione tenemos que tener instalado la *JMV* (*Java virtual machine*) y el *JRE* (*Java Runtime Enviorment*) necesario para crear aplicaciones *Java*. Al compilar el código *Java* no se genera código máquina como se hace normalmente, sino un código intermedio, esto es así ya que el código máquina depende del ordenador

en que se ejecute, por eso se genera el código intermedio y después la máquina virtual de *Java* es la encargada de transformar este código intermedio en el código máquina correspondiente para cada ordenador en el que se ejecute para lograr la total independencia del hardware. (Millet, 2012)

1.5.3 *Frameworks*

Framework

Puede definirse como una plataforma para desarrollar aplicaciones de software. Proporciona una base en la que los desarrolladores de software pueden crear programas para una plataforma específica. Esto agiliza el proceso de desarrollo ya que los programadores no necesitan reinventar la rueda cada vez que desarrollan una nueva aplicación. (TechTerms, 2013)

Spring Boot 2.0.0

Se trata de un *framework* que impulsa una metodología de trabajo ágil, eficiente y de buena praxis, lo que resulta en la creación de software de elevada calidad y mantenibilidad.

Es un *framework* de desarrollo de código libre para la plataforma *JAVA*, por lo tanto, cualquier sistema operativo con una máquina virtual de *JAVA* puede ejecutar aplicaciones desarrolladas en este *framework*. Su aspecto modular lo hace flexible y configurable para cualquier tipo de aplicación. (Faraoni, 2014)

Spring Initializr

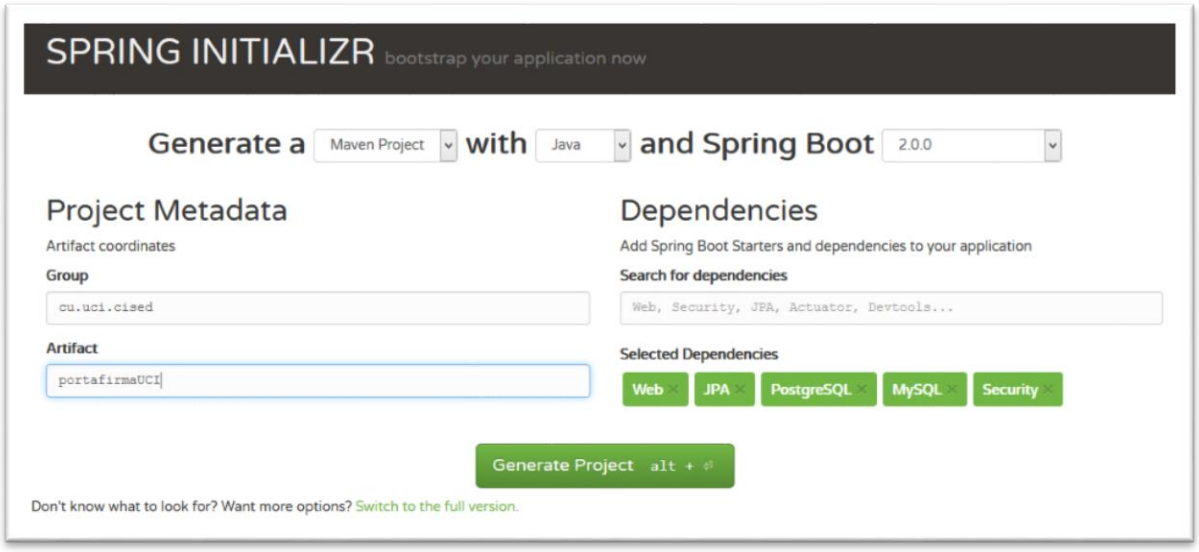

Figura 7: Interfaz de Spring Initializr. (start.spring.io)

Proporciona una interfaz de usuario web simple para configurar el proyecto que se puede usar a través de *HTTP* simple. Puede ver una instancia predeterminada en *start.spring.io*. El servicio le permite personalizar el proyecto a generar: el sistema de compilación y sus coordenadas, el idioma y la versión, el embalaje y finalmente las dependencias para agregar al proyecto. Este último es un concepto básico: basado en la versión *Spring Boot* elegida, se puede elegir un conjunto de dependencias, generalmente arrancadores *Spring Boot*, que tendrán un impacto concreto en su aplicación. (Pivotal, 2018)

1.5.4 Formato de intercambio de datos

JSON 3

Acrónimo de *JavaScript Object Notation*, es un formato de texto ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de JavaScript, aunque hoy, debido a su amplia adopción como alternativa a *XML*, se considera un formato de lenguaje independiente. Una de las ventajas de *JSON* como formato de intercambio de datos es la sencillez para escribir un analizador sintáctico (*parser*) de *JSON*. En *JavaScript*, un texto *JSON* se puede analizar fácilmente usando la función *eval()*, lo cual ha sido fundamental para que *JSON* haya sido aceptado por parte de la comunidad de desarrolladores *AJAX*, debido a la

ubicuidad de *JavaScript* en casi cualquier navegador web (Gutierrez, 2009). Por lo antes expuesto, se propone el uso del formato *JSON 3* para el desarrollo del paquete de servicios.

1.5.5 Servidores WEB

Apache Tomcat 7

Tomcat es un contenedor de *servlets* que se utiliza en la Referencia oficial de la implementación para *Java Servlet* y *Java Server Pages* (*JSP*). Las especificaciones *Java Servlet* y *Java Server Pages* son desarrolladas por *Sun Microsystems* cuyas especificaciones vienen dadas por la *JCP* (*Java Community Process*). *Apache Tomcat* es desarrollado en un entorno abierto y participatorio, bajo la licencia de *Apache Software License*. (Ajpdsoft, 2016)

JPA 5.1

La persistencia de datos es un medio mediante el cual una aplicación puede recuperar información desde un sistema de almacenamiento no volátil y hacer que esta persista. La persistencia de datos es vital en las aplicaciones empresariales debido al acceso necesario a las bases de datos relacionales. Las aplicaciones desarrolladas para este entorno deben gestionar por su cuenta la persistencia o utilizar soluciones de terceros para manejar las actualizaciones y recuperaciones de las bases de datos con persistencia. *JPA* (Keith, y otros, 2009) proporciona un mecanismo para gestionar la persistencia y la correlación relacional de objetos y funciona desde las especificaciones *EJB 3.0*.

1.5.6 Servidores de bases de Datos

Entre los sistemas de bases de datos existentes hoy en día, *Postgres SQL* juega un papel muy importante ya que es un sistema que tiene muchas cualidades que lo hacen ser una muy buena alternativa para instalar sistemas en empresas, universidades y una gran cantidad de otras aplicaciones.

Postgres SQL 9.4.5

Entre los sistemas de bases de datos existentes hoy en día, *Postgres SQL* juega un papel muy importante ya que es un sistema que tiene muchas cualidades que lo hacen ser una muy buena alternativa para instalar sistemas en empresas, universidades y una gran cantidad de otras aplicaciones.

Es un avanzado sistema de bases de datos relacionales basado en *Open Source*. Esto quiere decir que el código fuente del programa está disponible a cualquier persona libre de cargos directos, permitiendo a

cualquiera colaborar con el desarrollo del proyecto o modificar el sistema para ajustarlo a sus necesidades. *Postgres SQL* está bajo licencia *BSD.* (Denzer, 2002)

PGAdmin III

Es una aplicación gráfica para gestionar el gestor de bases de datos *Postgres SQL,* siendo la más completa y popular con licencia *Open Source*. Está escrita en *C++* usando la librería gráfica multiplataforma *wxWidgets,* lo que permite que se pueda usan en *Linux, Mac OS X* y *Windows*. Es capaz de gestionar versiones a partir de la *Postgres SQL* 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de *Postgres SQL* como *Pervasive Postgres, EnterpriseDB, Mammoth Replicator* y *SRA PowerGres*. (Ubuntu, 2014)

MySQL 7.5

MySQL Enterprise Edition incluye el conjunto más completo de funciones avanzadas, herramientas de administración y soporte técnico para alcanzar los niveles más altos de escalabilidad, seguridad, confiabilidad y tiempo de actividad de *MySQL*. Reduce el riesgo, el costo y la complejidad en el desarrollo, implementación y administración de aplicaciones *MySQL* críticas para el negocio. (Oracle, 2018)

MySQL es la base de datos de código abierto más popular del mundo. Con su rendimiento, confiabilidad y facilidad de uso comprobados, *MySQL* se ha convertido en la principal opción de base de datos para aplicaciones basadas en la Web, utilizada por propiedades web de alto perfil como *Facebook, Twitter*, *YouTube*, y los cinco principales sitios web. Es una alternativa extremadamente popular como base de datos integrada, distribuida por miles de *ISV* y *OEM*. (Oracle, 2017)

MySQL Workbench 6.2

MySQL Workbench es una herramienta visual unificada para arquitectos de bases de datos, desarrolladores y administradores de bases de datos. *MySQL Workbench* proporciona modelado de datos, desarrollo *SQL* y herramientas integrales de administración para la configuración del servidor, administración de usuarios, respaldo y mucho más. *MySQL Workbench* está disponible en *Windows, Linux* y *Mac OS X. SQL Workbench* permite a un desarrollador o arquitecto de datos diseñar visualmente, modelar, generar y administrar bases de datos. Incluye todo lo que un modelador de datos necesita para crear modelos entidad relación complejos, ingeniería directa e inversa, y también ofrece funciones clave para realizar tareas difíciles de

administración y documentación de cambios que normalmente requieren mucho tiempo y esfuerzo. (Oracle, 2018)

1.5.7 Entornos de Desarrollo

NetBeans 8.1

Es un entorno de desarrollo gratuito y de código abierto que en el momento de desarrollar la propuesta de solución está en su versión 8.5. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Da soporte a las siguientes tecnologías, entre otras: *Java, PHP, Groovy, C/C++, HTML5*. Puede instalarse en varios sistemas operativos: *Windows, Linux, Mac OS.* (Calendamaia, 2014)

IntelliJ IDEA 17.3

IntelliJ IDEA es un entorno de desarrollo visual de código abierto que permite desarrollar rápida y fácilmente aplicaciones de escritorio, móviles y web, desarrollado por *JetBrains*. Ayuda a los desarrolladores a escribir, depurar, refactorizar, probar y aprender su código. Constantemente valida la calidad del código y ofrece soluciones inmediatas para los problemas encontrados en todos los niveles, desde la instrucción individual para arquitectura global, utilizando las inspecciones de código avanzado y análisis de matriz de dependencia (JetBrains, 2018). Para la implementación del paquete de servicios se utiliza el *IntelliJ IDEA* en su versión *2017.2.5*.

1.5.8 Herramientas de Modelado

Visual Paradigm 10.1

Es una herramienta de diseño y administración de software. La herramienta está diseñada para una amplia gama de usuarios, incluidos ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, o para cualquier persona interesada en la construcción fiable de sistemas de software de gran escala con un enfoque orientado a objetos. *Visual Paradigm* admite los últimos estándares de desarrollo de software *UML, BPMN* y ágil. (Visual Paradigm Guide User's, 2017)

1.5.9 Seguridad

JWT 2.2.0

JSON Web Token (*JWT*) es un estándar abierto (*RFC 7519*) que define una forma compacta y autónoma para transmitir de forma segura información entre las partes como un objeto *JSON*. Esta información puede ser verificada y confiable porque está firmada digitalmente. Los *JWT* se pueden firmar usando un secreto (con el algoritmo *HMAC*) o un par de claves públicas/privadas usando *RSA*. Aunque los *JWT* se pueden cifrar para que también proporcionen secreto entre las partes, nos centraremos en los *tokens* firmados. Los *tokens* firmados pueden verificar la integridad de los reclamos contenidos en él, mientras que los *tokens* cifrados ocultan esos reclamos de otras partes. Cuando los *tokens* se firman usando pares de claves públicas / privadas, la firma también certifica que solo la parte que tiene la clave privada es la que la firmó. (Auth0, 2017)

1.5.10 Herramientas para las pruebas

Apache JMeter 3.2

JMeter es un proyecto de *Apache Jakarta* que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web. Puede ser utilizado como una herramienta de pruebas unitarias para conexiones de bases de datos con *JDBC, FTP, LDAP*, Servicios web, *JMS, HTTP* y conexiones *TCP* genéricas. Es clasificado como una herramienta de "generación de carga" y soporta aserciones para asegurarse que los datos recibidos son correctos, por *cookies* de hilos, configuración de variables y una variedad de reportes. (Apache Foundation, 2018)

Acunetix 9.5

Acunetix es un escáner de vulnerabilidades de aplicaciones web. La herramienta está diseñada para encontrar agujeros de seguridad en las aplicaciones web de la organización que un atacante podría aprovechar para obtener acceso a los sistemas y datos. Comprueba los sistemas en busca de múltiples vulnerabilidades incluyendo: *SQL injection Cross Site Scripting Passwords* débiles. *Acunetix* puede utilizarse para realizar escaneos de vulnerabilidades en aplicaciones web y para ejecutar pruebas de acceso frente a los problemas identificados. La herramienta provee sugerencias para mitigar las vulnerabilidades

identificadas y puede utilizarse para incrementar la seguridad de servidores web o de las aplicaciones que se analizan. (López, 2017)

1.5.11 Control de versiones

Git 2.8

Git es un sistema de control de versiones distribuidas de código abierto y gratuito diseñado para gestionar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia. *Git* es fácil de aprender y tiene una huella pequeña con un rendimiento increíblemente rápido. Supera a las herramientas de *SCM* como *Subversion*, *CVS*, *Perforce* y *ClearCase* con funciones como ramificación local barata, áreas de preparación conveniente y flujos de trabajo múltiples. (Git, 2016)

1.5.12 Librerías

En la programación, una Librería es una colección de rutinas pre compiladas que un programa puede usar. Las rutinas, a veces llamadas módulos, se almacenan en formato de objeto. Las Librerías son particularmente útiles para almacenar rutinas usadas con frecuencia porque no necesita vincularlas explícitamente a cada programa que las usa.

iText 5.0.6

Es uno de los motores de *PDF* mejor documentados y más versátiles del mundo (escrito en *Java* y *.NET*). Integra funcionalidades de *PDF* dentro de sus aplicaciones, procesos o productos. (iText, 2018)

iText es una librería *PDF* que nos permite, usando *Java*, editar o transformar documentos en formato *PDF*. La versión utilizada en la propuesta de solución para firmar digitalmente los documentos es la 5.5.9.

Lombok

Project Lombok es una biblioteca de *Java* que se conecta automáticamente a su editor y crea herramientas, condimentando su *Java*. Evita que los desarrolladores escriban otro *getter* o método igual de nuevo. (Lombok, 2018)

1.5.13 Metodología de desarrollo de software

Una metodología de desarrollo de software es un enfoque estructurado que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos. Las metodologías han evolucionado de

manera significativa en las últimas décadas, tanto así, que pueden permitir el éxito o el fracaso de muchos de los sistemas desarrollados para distintas áreas. (Valdéz, 2014)

Dentro de las metodologías de desarrollo de software se encuentran:

- ✓ **Metodología Tradicional:** impone una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un sistema más eficiente. Se centra especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. (Corrales, 2011)
- ✓ **Metodología Ágil:** es aquella que permite adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para adaptar el proyecto y su desarrollo a las circunstancias específicas del entorno. (Merchán, 2008)

Para la implementación del paquete de servicios se decide utilizar una metodología ágil debido a que según (Merchán, 2008), éstas permiten entregar productos de calidad con los costes y tiempos pactados, y las metodologías tradicionales ya no bastan para este cometido, no se adaptan a las nuevas expectativas de los usuarios ni a las exigencias del mercado. Las metodologías ágiles brindan muchas facilidades en cuanto a la documentación y el entorno cambiante que tiene el proyecto; pues el mismo se lleva a cabo por un solo desarrollador y es más importante centrarse en la correcta obtención del sistema sin importar cuán documentado esté.

Luego de haber seleccionado la metodología ágil, a continuación, se realiza un análisis de varias de ellas:

Agile Unified Process (*AUP*)

Es un acercamiento aerodinámico al desarrollo del software basado en el Proceso Unificado Racional (*RUP* por sus siglas en inglés) de la *International Business Machines Corp.* (*IBM*), basado en disciplinas y entregas incrementales con el tiempo. El ciclo de vida en proyectos grandes es serial mientras que en los pequeños es iterativo. Las disciplinas de *AUP según* (Salvador, 2015) son:

- ✓ **Modelado:** Entender el negocio de la organización, el problema de dominio que se aborda en el proyecto, y determinar una solución viable para resolver el problema de dominio.
- ✓ **Implementación:** Transformar el modelo en código ejecutable y realizar un nivel básico de pruebas individuales.
- ✓ **Prueba:** Realizar una evaluación objetiva para garantizar la calidad. Incluye la búsqueda de defectos,

validar que el sistema funciona tal como está establecido y verificar que se cumplan los requisitos.

- ✓ **Despliegue:** Realizar un plan para la presentación del sistema y ejecutarlo para hacer que el sistema se encuentre a disposición de los usuarios finales.
- ✓ **Administración de la configuración:** Realizar la gestión de acceso a los artefactos de su proyecto. Incluye el seguimiento de las versiones del artefacto en el tiempo y la gestión y control de los cambios.
- ✓ **Administración o gerencia del Proyecto:** Dirigir las actividades que se llevan a cabo en el proyecto. Incluye la gestión de riesgos, dirección de personas, garantizar la entrega a tiempo y con el presupuesto asignado.
- ✓ **Entorno:** Garantizar que las orientaciones y herramientas estén disponibles para el equipo en cualquier momento que ellos lo necesiten.

Variación *AUP* para la UCI (Escenario No 4)

La UCI desde sus inicios se caracterizó por el uso de diferentes metodologías de desarrollo entre robustas y ágiles para la realización del software en los centros de producción. A pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos la aplican en su totalidad. Según (Sánchez, 2015), al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto, exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología *AUP*, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Esta variación está unida al modelo *CMMI-DEV v1.3*, para garantizar las buenas prácticas en función de un software de calidad. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. Para el desarrollo de la propuesta de solución, se decide usar la metodología *AUP*, en su variación para la UCI, teniendo en cuenta todo lo antes expuesto.

Escenario No 4: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una Historias de Usuario (HU) no debe poseer demasiada información. Todas las disciplinas antes definidas (desde Modelado de negocio hasta Pruebas de Aceptación) se desarrollan en la Fase de Ejecución, de ahí que en la misma se realicen Iteraciones y se obtengan resultados incrementales. En una iteración se repite el flujo de trabajo de las disciplinas,

Requisitos, Análisis y diseño, Implementación y Pruebas internas. De esta forma se brinda un resultado más completo para un producto final de manera creciente. Para llegar a lograr esto, cada requisito debe tener un completo desarrollo en una única iteración.

Con la adaptación de *AUP* que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software, dando cumplimiento a las buenas prácticas que define *CMMI-DEV v1.3*. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos.

1.5.14 Disponibilidad de datos

API (Application Programming Interface)

Una *API* es un conjunto de comandos, funciones, protocolos y objetos que los programadores pueden usar para crear software o interactuar con un sistema externo. Proporciona a los desarrolladores comandos estándar, para realizar operaciones comunes, y evitar escribir el código desde cero. (TechTerms, 2013)

API-REST

Los servicios web *RESTfull* están diseñados para funcionar mejor en la Web. *Representational State Transfer* (*REST*) es un estilo arquitectónico que especifica restricciones, como la interfaz uniforme, que si se aplican a un servicio web inducen propiedades deseables, como el rendimiento, la escalabilidad y la capacidad de modificación, que permiten que los servicios funcionen mejor en la Web. En el estilo arquitectónico *REST*, los datos y la funcionalidad se consideran recursos y se accede a ellos utilizando identificadores uniformes de recursos (*URI*), generalmente enlaces en la Web. Se actúa sobre los recursos utilizando un conjunto de operaciones simples y bien definidas. El estilo arquitectónico *REST* restringe una arquitectura a una arquitectura cliente / servidor y está diseñado para usar un protocolo de comunicación sin estado, típicamente *HTTP*. En el estilo de arquitectura *REST*, los clientes y servidores intercambian representaciones de recursos mediante el uso de una interfaz y protocolo estandarizados. (Oracle, 2018)

1.6 CONCLUSIONES DEL CAPÍTULO

- ✓ Se evidencia la importancia de proteger la información, como activo más importante en una organización. Donde el uso de un portafirmas es la solución más adecuada a las condiciones actuales de la universidad.
- ✓ La definición de los principales conceptos asociados al dominio de la presente investigación y las relaciones entre estos, permitió alcanzar una mayor comprensión de la propuesta de solución.

- ✓ El análisis realizado a los sistemas homólogos, permite constatar las principales funcionalidades con las que debe contar un portafirmas, así como las tecnologías más usadas en su desarrollo.
- ✓ El estudio del ambiente de desarrollo de la propuesta de solución definido por el proyecto constató como metodología de desarrollo AUP-UCI, lenguaje de modelado *UML* con la herramienta *Visual Paradigm*, para la implementación, *JAVA* como lenguaje de programación sobre el *framework Spring Boot* apoyado en el *IDE NetBeans*, para la gestión de los datos *PostgreSQL* y como servidor de aplicaciones *Apache Tomcat.*

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA PROPUESTA SE SOLUCIÓN

En este capítulo quedan reflejadas las relaciones de los conceptos del paquete de servicios una vez definido el modelo de dominio, resultado del análisis de la situación actual en la Universidad de las Ciencias Informáticas. Se detallan las características de la propuesta de solución, se especifican los requisitos funcionales y no funcionales del mismo. Como parte de las fases de Análisis y Diseño que propone la metodología AUP-UCI, se presentan los patrones utilizados durante la implementación, el modelo de datos, el diagrama de clases del diseño y el modelo de despliegue asociados a la propuesta de solución.

2.1 MODELO CONCEPTUAL

En ingeniería de software, el modelo conceptual es un tipo de modelo relativamente poco sofisticado lo cual se hace más fácil de comprender. (Sommerville, 2011)

Dentro de las principales actividades definidas en la metodología AUP se encuentra la definición del modelo conceptual. Un modelo conceptual es una descripción del dominio de un problema real, no constituye una descripción del diseño del software. (Larman, 2004.)

Se presenta la persona como el activo principal de proceso, la cual posee un certificado digital emitido por la autoridad certificadora de la UCI, UCI-CA. El usuario con su certificado digital conjuntamente con su clave privada, firma digitalmente documentos digitales. Para firmar el usuario utiliza diferentes herramientas, como es el caso de UCI_PDFSigner o *Adobe Reader*. Luego de este proceso el documento pasa a ser un documento firmado. A continuación, para un mejor entendimiento se representa el siguiente modelo conceptual en la Figura 8.

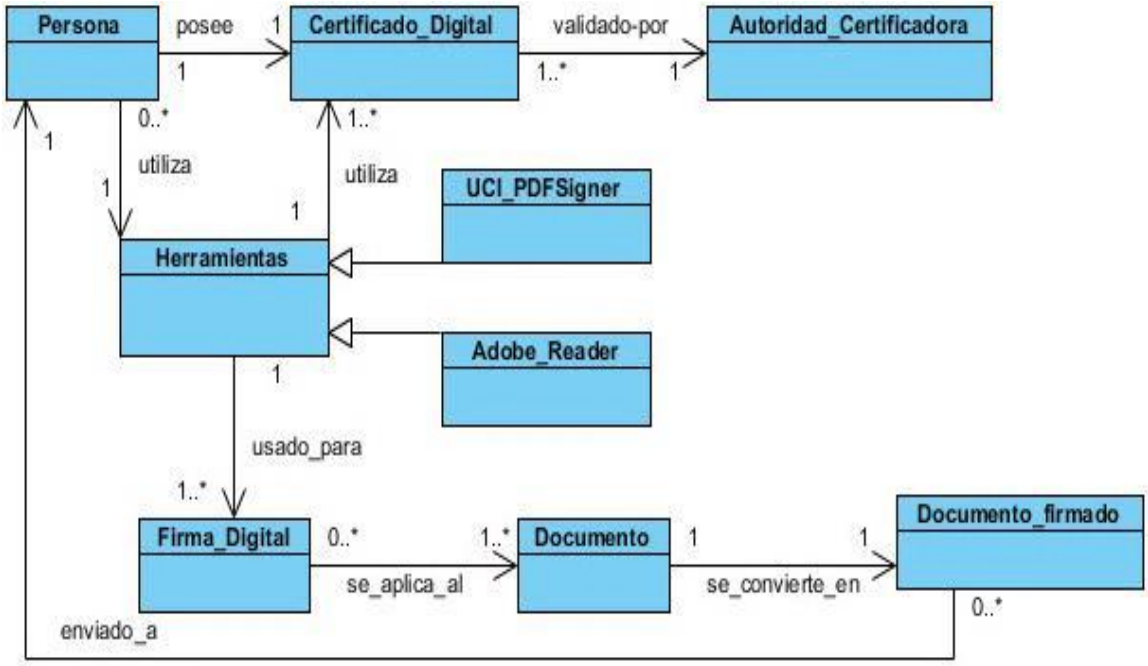

Figura 8: Modelo conceptual del Portafirmas-UCI (*Elaboración propia*)

Descripción de los elementos del modelo conceptual

- ✓ **Persona:** recurso humano que tiene la responsabilidad de firmar digitalmente documentos.
- ✓ **Certificado Digital:** credencial electrónica que contiene la información del titular de dicho certificado con su clave pública.
- ✓ **Autoridad Certificadora:** entidad encargada de controlar el ciclo de vida de los certificados y garantiza la confidencialidad y el no repudio de la comunicación.
- ✓ **Herramienta:** aplicación de escritorio utilizada para firmar digitalmente los documentos en la universidad.
- ✓ **UCI PDFSigner y Adobe Reader:** herramientas utilizadas para la firma digital de documentos.
- ✓ **Firma Digital:** acción que realiza la herramienta a los documentos, incorporándole un código seguro de verificación que permite constatar la identidad de los usuarios firmantes.
- ✓ **Documento:** archivo electrónico que necesita ser firmado en el proceso de autenticación de documentos digitales.

- ✓ **Documento firmado:** archivo electrónico que está firmado digitalmente por una o varias personas.

2.2 CARACTERÍSTICAS DE LA PROPUESTA DE SOLUCIÓN.

La solución propuesta facilita la mejora del proceso de autenticación de documentos digitales en la UCI, aportando seguridad en la gestión documental de los procesos docentes y productivos. Esta permitirá publicar mediante *APIs REST* (servicios) todos los datos necesarios para el funcionamiento del Portafirmas UCI. Estos servicios podrán ser consumidos por cualquier otro sistema que necesite los datos asociados al proceso de autenticación de documentos oficiales, por ejemplo, el Portafirmas de la UCI. El paquete de servicios solo será capaz de firmar documentos en formato *PDF*. El paquete de servicios brindará la posibilidad de firmar varios documentos a la vez (firmado por lotes). Los documentos a firmar viajarán mediante una petición de firmado, la cual tiene una lista de usuarios a notificar, esta lista estará ordenada. Ver Figura 9.

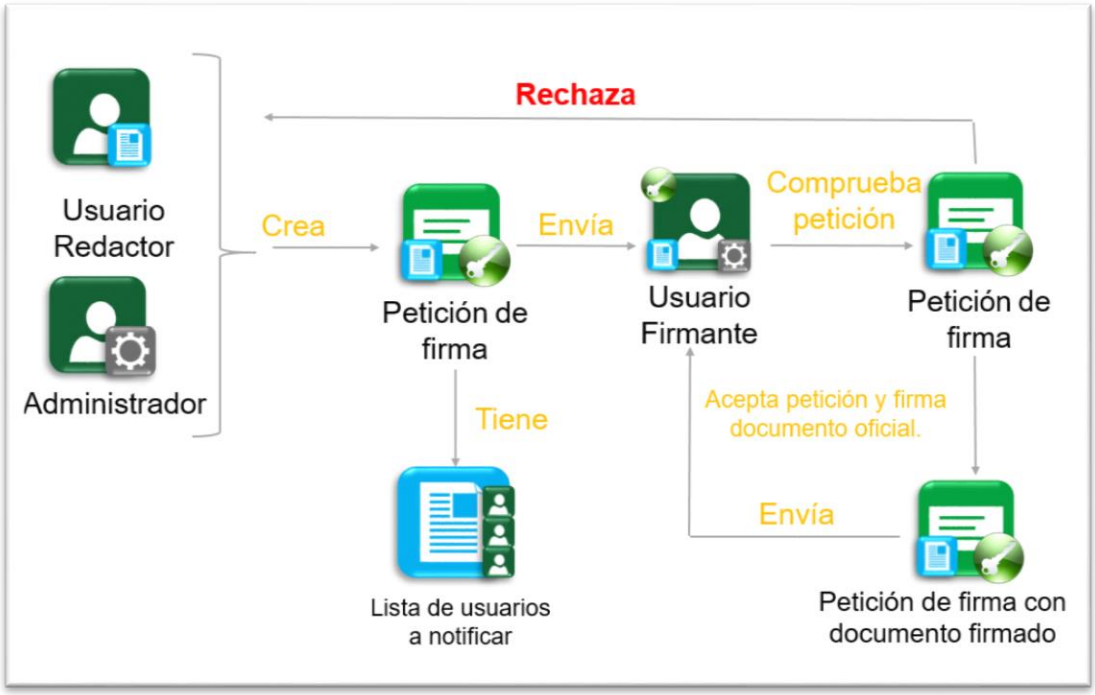

Figura 9: Descripción de la propuesta de solución.

El paquete de servicios gestiona tres tipos de roles: administrador, encargado de gestionar todas las configuraciones; el redactor y el firmante. A continuación, se explican dichos roles:

- ✓ **Firmante:** permite el acceso a los datos de las peticiones pendientes, en espera, terminadas,

vencidas, y a los datos referentes a la firma de documentos digitales de las peticiones pendientes.

- ✓ **Redactor:** permite el acceso a los datos referentes a la redacción de peticiones, permite enviar peticiones a otros usuarios del sistema. Tiene acceso a los permisos del rol firmante.
- ✓ **Administrador:** presenta todos los permisos del sistema y brinda acceso a los requisitos de administración del sistema.

Los usuarios independientemente de su rol manejan peticiones explicadas y asociadas a continuación:

Tabla 3: Descripción de las peticiones manejadas por los usuarios en el sistema. (*Elaboración propia*)

Peticiones	Descripción
Pendientes	Son las peticiones que le han enviado al usuario autenticado y están pendientes de su firma.
En Espera	Son las peticiones en las que el usuario autenticado se encuentra definido como firmante en la cadena de firmado; estas peticiones pasarán a "Pendientes" de modo automático en el momento en que los firmantes del orden anterior al usuario hayan realizado su firma.
Terminadas	Son las peticiones que ya han sido firmadas por el usuario autenticado o devueltas al remitente sin firmar.
Mis Peticiones	Son las peticiones que el usuario firmante haya enviado teniendo el rol de Redactor.
Vencidas	Son las peticiones que han caducado por el usuario autenticado.

2.3 REQUISITOS FUNCIONALES

Los requisitos funcionales son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse en situaciones específicas. En algunos casos, los requerimientos funcionales también explican lo que el sistema no debe hacer (Sommerville, 2011).

Se obtuvo un total de (27) requisitos funcionales, los cuales son mostrados en la Tabla 4:

Tabla 4: Requisitos funcionales del paquete de servicios. (*Elaboración propia*)

Requisito	Prioridad	Requisito	Prioridad
RF1: Autenticar usuario	Alta	RF2: Crear usuario	Alta
RF3: Editar usuario	Media	RF4: Eliminar usuario	Baja
RF5: Buscar usuario	Baja	RF6: Listar usuarios	Alta
RF7: Rechazar documento	Alta	RF8: Mostrar documento	Alta
RF9: Crear petición de firma	Alta	RF10: Mostrar estado de la petición de firma	Alta
RF11: Buscar petición de firma	Media	RF12: Listar peticiones pendientes de firma	Alta
RF13: Firmar documento digital (.pdf)	Alta	RF14: Listar peticiones terminadas	Alta
RF15: Descargar documento	Media	RF16: Listar peticiones creadas	Alta
RF17: Listar peticiones en espera de firma	Alta	RF18: Crear área	Alta
RF19: Editar área	Baja	RF20: Eliminar área	Baja
RF21: Buscar área	Baja	RF22: Listar áreas	Alta
RF23: Crear cargo	Alto	RF24: Editar cargo	Alta

RF25: Eliminar cargo	Baja	RF26: Buscar cargo	Baja
RF27: Listar cargos	Alta		

2.4 REQUISITOS NO FUNCIONALES

Se obtuvo un total de (8) requisitos no funcionales, distribuidos en especificaciones de confiabilidad, fiabilidad, eficiencia, funcionalidad y mantenibilidad, estos se relacionan a continuación:

Fiabilidad

RnF 1: El paquete de servicios debe ser tolerante a fallos.

Confiabilidad

RnF 2: Enviar solo la información necesaria para orientar al usuario.

RnF 3: La clave privada de los usuarios nunca debe ser almacenada en el paquete de servicios.

Eficiencia

RnF 4: Interacción concurrente por parte de los usuarios.

Funcionabilidad

RnF 5: Es necesario la información del usuario para poder utilizar la aplicación.

RnF 6: Al ocurrir un fallo en el paquete de servicios no se debe mostrar información que pueda comprometer la seguridad de la misma.

RnF 7: Luego de 5 minutos de inactividad la aplicación debe ser capaz de cerrar sesión.

Hardware de la estación servidor

RnF 8: Servidor con procesador *Core2Duo* a 3.16 *GHz* o superior, y *8Gb* de memoria *RAM*.

2.5 HISTORIAS DE USUARIOS

Unas de las técnicas utilizadas por la metodología son las Historias de Usuario (HU), según (Sánchez, 2015), en la metodología AUP-UCI, en su escenario 4 para la disciplina requisitos, genera como uno de sus artefactos a las Historias de Usuario (HU), que tienen como objetivo especificar los requisitos del software.

Las HU describen brevemente las características que desea un cliente para el sistema a desarrollar. Estas son lo suficientemente comprensibles y delimitadas para que los programadores puedan implementarlas.

Mediante los siguientes parámetros quedan recogidas la HU orientadas por el usuario:

- ✓ **Número:** Número de la HU a describir.
- ✓ **Nombre:** Nombre de la HU a describir.
- ✓ **Programador responsable:** Programador responsable de la HU.
- ✓ **Iteración asignada:** Iteración a la que pertenece la HU en el plan de iteraciones.
- ✓ **Prioridad en negocio:** Nivel de prioridad de la HU para los desarrolladores (Alta, Media, Baja).
    - **Baja:** Se le otorga a las HU que son de funcionalidades auxiliares y que son independientes del sistema.
    - **Media:** Se le otorga a las HU que son de funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
    - **Alta:** Se le otorga a las HU que son de funcionalidades fundamentales en el desarrollo del sistema.
- ✓ **Descripción:** Breve descripción de la HU.
- ✓ **Observaciones:** Aspectos importantes de interés para el cliente.

Se generaron un total de 27 HU, a continuación, se muestran la HU perteneciente al RF 9:

Tabla 5: HU_9 Crear petición de firma. (*Elaboración propia*)

Historia de usuario		
Número: HU_9	**Nombre:** Crear petición de firma	
Programador responsable: Adiel Alfonso Cordovi		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite crear una nueva petición de firma en el sistema.		

Observaciones: Para crear una petición de firma, el sistema debe obtener los datos: asunto de la petición, fecha de caducidad, documento a adjuntar y listado de flujo de firma (por usuario o por cargo). El sistema debe registrar la fecha en que se realizó la petición, la hora y el estado por el que va transitando la petición (pendiente, terminada o rechazada).

2.6 DIAGRAMA DE CLASES DEL DISEÑO

Los diagramas de clase pueden usarse cuando se desarrolla un modelo de sistema orientado a objetos para mostrar las clases en un sistema y las asociaciones entre dichas clases. (Sommerville, 2011)

En la Figura 10, se muestra el diagrama de clases del diseño de la propuesta de solución.

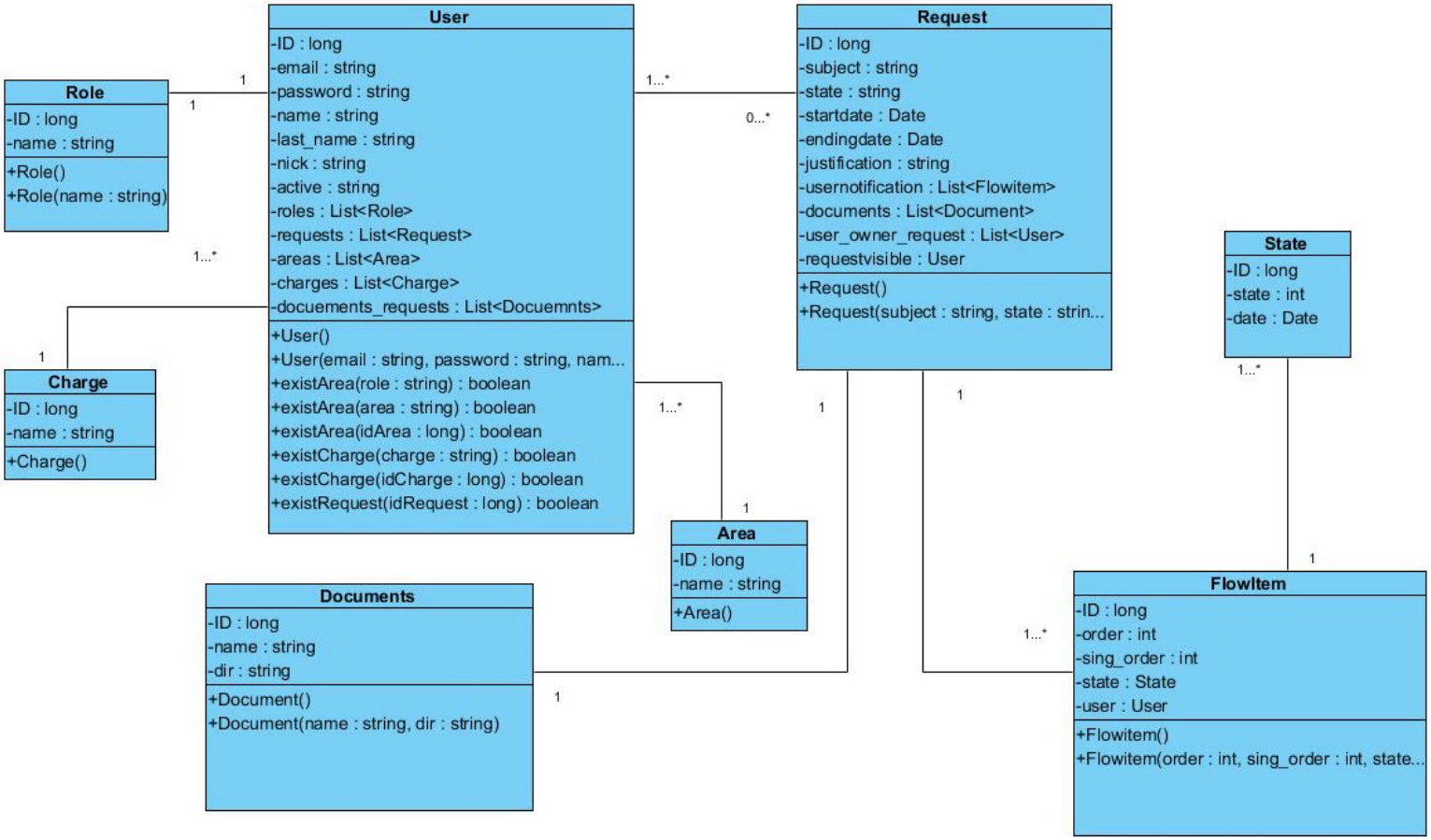

Figura 10: Diagrama de clases del diseño. (*Elaboración propia*)

2.7 ESTILO ARQUITECTÓNICO

La arquitectura de un software se encarga de entender cómo debe organizarse el sistema y cómo tiene que diseñarse la estructura global de ese sistema. Es el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos. La salida del proceso de diseño arquitectónico consiste en un modelo arquitectónico que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación. La arquitectura de software es importante porque afecta el desempeño y la potencia, así como la capacidad de distribución y mantenimiento de un sistema (Sommerville, 2011).

Según (Pressman, 2010.), la arquitectura de un sistema puede basarse en un patrón o un estilo arquitectónico particular. Un patrón arquitectónico es una descripción de una organización del sistema, captan la esencia de una arquitectura que se usó en diferentes sistemas de software.

Presentación desacoplada

El estilo de presentación separada indica cómo debe realizarse el manejo de las acciones del usuario, la manipulación de la interfaz y los datos de la aplicación. Este estilo separa los componentes de la interfaz del flujo de datos y de la manipulación.

Características

- ✓ Es un estilo, para diseñar aplicaciones, basado en patrones de diseño conocidos.
- ✓ Separa la lógica para el manejo de la interacción de la representación de los datos con que trabaja el usuario.
- ✓ Permite a los diseñadores crear una interfaz gráfica mientras los desarrolladores escriben el código para su funcionamiento.
- ✓ Ofrece un mejor soporte para el testeo ya que se pueden testear los comportamientos individuales.

¿Cuándo usarlo?

- ✓ Se desea mejorar el testeo y el mantenimiento de la funcionalidad de la interfaz.
- ✓ Se desea separar la tarea de crear la interfaz de la lógica que la maneja.
- ✓ No se desea que la interfaz contenga ningún código de procesamiento de eventos.

- ✓ El código de procesamiento de la interfaz no implementa ninguna lógica de negocio.

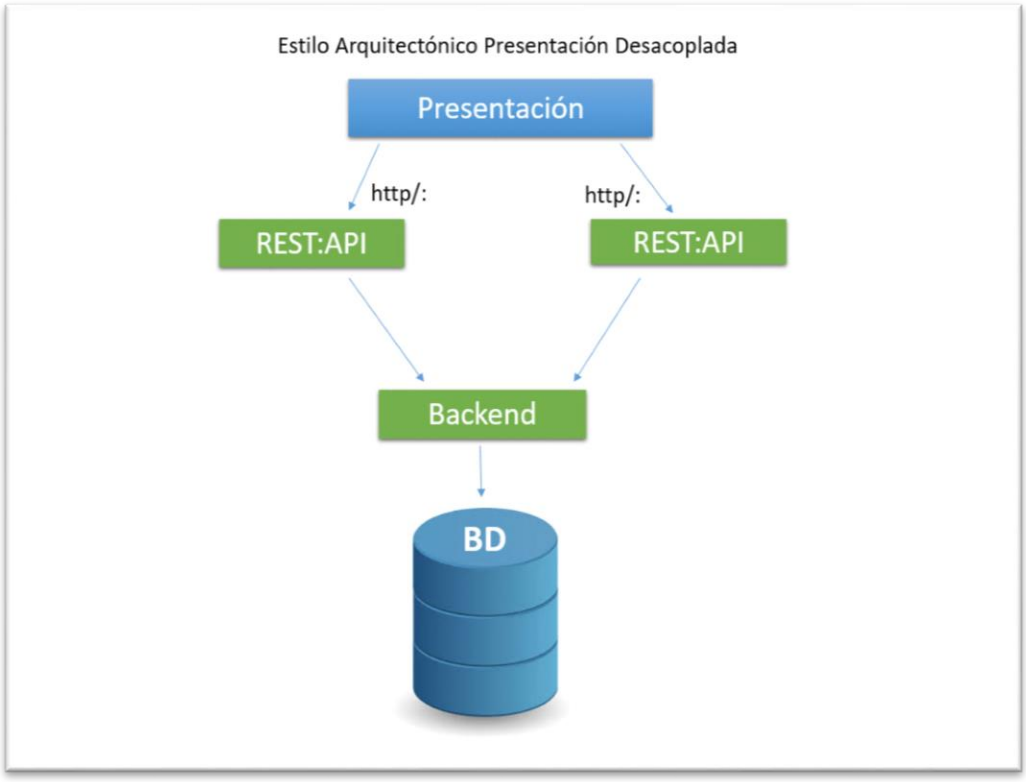

Figura 11: Arquitectura presentación desacoplada. (*Elaboración propia*)

La Figura 12, muestra los elementos del paquete de servicios organizados a partir del estilo arquitectónico antes mencionado y la utilización del *framework Spring Boot*.

En el paquete *ControllerAPIs* están publicados todos los servicios en sus respectivas *APIs*, los cuales son implementados en el paquete *ImplementServices*. Las clases de acceso a datos están almacenadas en el paquete *repository*, basadas en los modelos almacenados en el paquete *model*. Estos son los elementos básicos del estilo arquitectónico presentación desacoplada, los demás paquetes almacenan clases de configuración u otras para realizar funciones específicas.

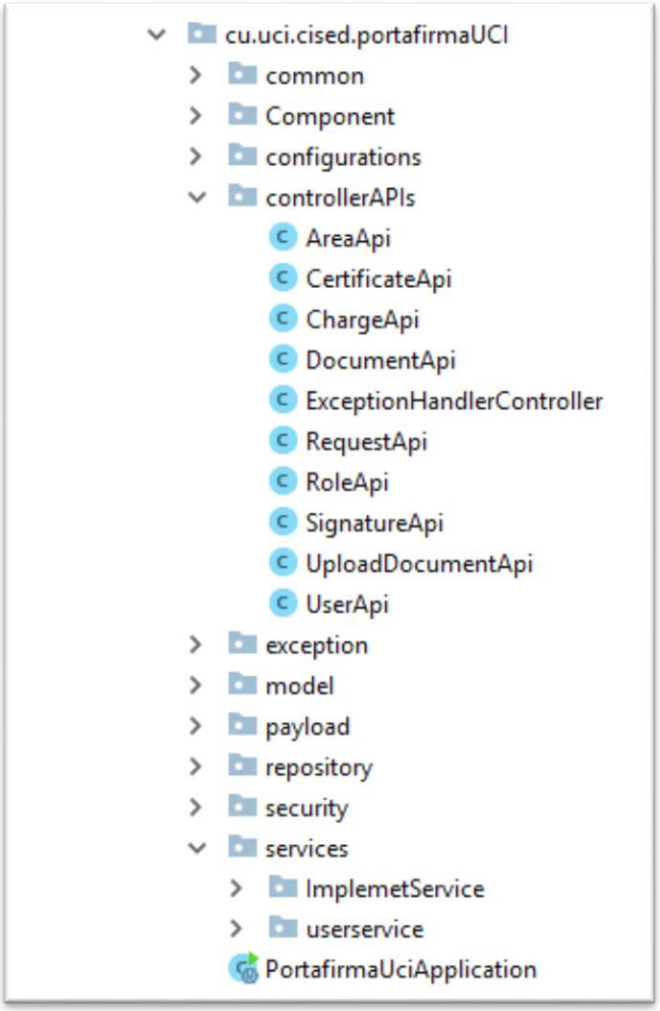

Figura 12: Aplicando Presentación desacoplada en la propuesta de solución. (Elaboración propia)

2.8 PATRONES DE DISEÑO

Los diseñadores expertos en orientación a objetos (y también otros diseñadores de software), van formando un amplio repertorio de principios generales y de expresiones que los guían a crear el software. A unos y a otras podemos asignarles el nombre de patrones, si se codifican en un formato estructurado que describe el problema y su solución, y si se les asigna un nombre. (Larman, 2004.)

2.8.1 Patrones GRASP (*Responsability Assignment Software Patterns* o *Patrones* de Software para Asignación de Responsabilidades):

Según (Grosso, 2011), son patrones basados en la asignación de responsabilidades a objetos. Es una buena práctica para el desarrollo eficaz de la Programación Orientada a Objetos (POO).

Experto.

Asignar responsabilidades al experto de la información, es decir, a la clase que tiene la información necesaria para llevar la tarea a cabo.

Hay que tener en cuenta que esto es aplicable mientras estemos considerando los mismos aspectos del sistema:

- ✓ Lógica de negocio.
- ✓ Persistencia a la base de datos.
- ✓ Interfaz de usuario.

```
@Override
@Scheduled(cron = "0 0 0 * * *")
public void expire() {
    List<Request> request=requestRepository.findAllPendingRequest();
    for (Request r: request ){
        System.out.println(">>> " + r.getSubject());
    }

    Date today = new Date();

    for (Request request1: request){
        Date endingDate = request1.getEndingDate();

        if ( endingDate.before( today ) ){
            request1.setState("expired");
            request1.setEndingDate(new Date());
            System.out.println(String.format(">>> Request %s expired", request1.getSubject()));
            request1.setEndingDate(new Date());
            requestRepository.save(request1);
        }
    }
}
```

Figura 13 : Aplicando el patrón experto, método expirar petición. (Elaboración propia)

Creador

Este patrón, como su nombre lo indica, es el que crea la guía de asignación de responsabilidades relacionadas con la creación de objetos. Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

- ✓ B contiene a A.
- ✓ B es una agregación (o composición) de A.
- ✓ B almacena a A.
- ✓ B tiene los datos de inicialización de A (datos que requiere su constructor).
- ✓ B usa a A.

```
@RequestMapping(value = "/auth/sign-up", method = RequestMethod.POST)
  @PreAuthorize("hasRole('ROLE_ADMIN')")
public ResponseEntity<?> signUp(@RequestBody UserPayload userRequest) throws EResourceAlreadyTaken {

    User user = UserPayload.create(userRequest);
    System.out.println(">>>" + user);
    // Verify if user exist
    boolean userPresent = userService.findUserByNick(user.getNick()).isPresent();
    if (userPresent) throw new EResourceAlreadyTaken("El usuario ya esta tomado");

    // Verify if email exist
    boolean emailPresent = userService.findUserByEmail(user.getEmail()).isPresent();
    if (emailPresent) throw new EResourceAlreadyTaken("El correo ya esta tomado");

    // Creating user's account
    user.setPassword(passwordEncoder.encode(user.getPassword()));
    User result = userService.saveUser(user);
    UserPayload response = UserPayload.create(result, hasPassword: false);

    return ResponseEntity.status(HttpStatus.OK).body(response);
}
```

Figura 14: Aplicando el patrón creador, método singUp. (Elaboración propia)

Controlador

Es un evento generado por actores externos. Se asocian con operaciones del sistema, operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos.

Normalmente un controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. No realiza mucho trabajo por sí mismo.

```
@CrossOrigin(origins = "*")
@RequestMapping(value = "/add/area", method = RequestMethod.POST)
@ApiOperation(value = "Add un area a un usuario.", response = User.class)
public ResponseEntity<?> addAreaUser(@RequestParam("idUser") Long idUser, @RequestParam("idArea") Long
idArea) throws URISyntaxException {
    try {
        userService.addAreaToUser(idUser, idArea);
    } catch (EUserNotFound eUserNotFound) {
        return ResponseEntity.notFound().headers(HeadersTools.createFailureAlert(ENTITY_NAME,
errorKey: "notfound", eUserNotFound.getMessage())).build();
    } catch (EAreaNotFound eAreaNotFound) {
        return ResponseEntity.notFound().headers(HeadersTools.createFailureAlert(ENTITY_NAME,
errorKey: "notfound", eAreaNotFound.getMessage())).build();
    }
    return ResponseEntity.ok().build();
}
```

Figura 15: Aplicando patrón controlador, método adicionar área a un usuario. (Elaboración propia)

Alta Cohesión

La alta cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo.

Bajo Acoplamiento

El bajo acoplamiento es una medida de fuerza con que un elemento, tiene conocimiento de, confía en, otros elementos. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. El controlador no realiza estas actividades, las delega en otras clases con las

que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

2.8.2 Patrones *GoF* (*Gang of Four* o Patrones de la pandilla de los cuatros):

Los patrones de diseño del grupo *GoF* clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

Creacionales: Tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

- ✓ ***Abstract Factory* (Fábrica abstracta):** Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.

```
@Override
public long countExpiredRequestsByUser(long userId) {
    List<Request> completedRequestByUser = requestRepository.countExpiredRequestByUser(userId);
    List<Request> filter = new LinkedList<>();
    for (Request request: completedRequestByUser ){
        List<User> collect = request.getRequestvisible().stream().filter(u -> u.getId() == userId)
                .collect(Collectors.toList());
        if ( collect.size() == 0)
            filter.add(request);
    }
    return filter.size();
}
```

Figura 16: Aplicando fábrica abstracta, método contar peticiones expiradas por usuario. (Elaboración propia)

- ✓ ***Builder* (Constructor virtual):** Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.

Estructurales: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.

- ✓ ***Adapter*** (Adaptador): Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.

```java
public interface RoleService {
    public Role findByRole(String role);
    public Role saveRole(Role role);
    public Role updateRole(Role role) throws ERoleNotFound;

   // public RoleTransferList getRole(int page, int elementsByPage);

    public List<Role> getRoles();


    public boolean delete(Long roleId) ;

    public Role get(Long roleId) throws ERoleNotFound;
}
```

Figura 17: Aplicando Patrón adapter, clase interfaz RoleService. (Elaboración propia)

***Composite* (Objeto compuesto):** Permite tratar objetos compuestos como si de uno simple se tratase.

- ✓ ***Facade* (Fachada):** Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

Comportamiento: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

- ✓ **Iterator (Iterador):** Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.

```java
for (Request request1: request){
    Date endingDate = request1.getEndingDate();

    if ( endingDate.before( today ) ){
        request1.setState("expired");
        request1.setEndingDate(new Date());
        System.out.println(String.format(">>> Request %s expired", request1.getSubject()));
        request1.setEndingDate(new Date());
        requestRepository.save(request1);
    }
```

Figura 18: Aplicando patrón iterator, recorrido sobre la entidad request. (Elaboración propia)

- ✓ ***Mediator* (Mediador):** Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
- ✓ ***Observer* (Observador):** Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.

2.9 MODELADO DE DATOS

El modelo de datos obtenido, como se observa en la Figura 19, está compuesto por (12) relaciones y (11) tablas, las cuales se explican a continuación:

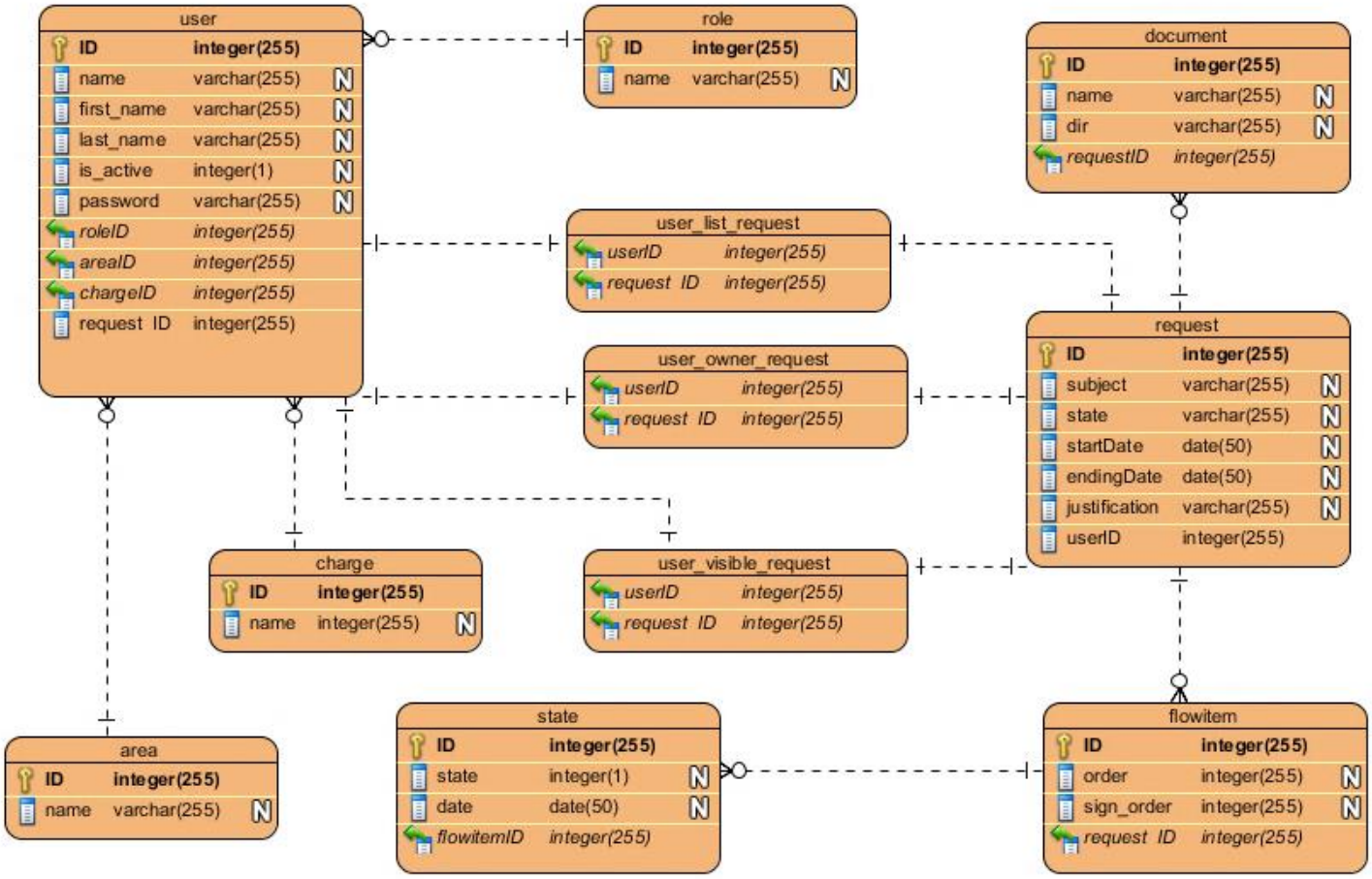

Figura 19: Modelo de datos del Paquete de Servicios para el Sistema Portafirmas-UCI. (Elaboración propia)

***user*:** Contiene los datos de acceso de un usuario en el sistema y los datos que lo identifican (*ID*), contraseña (*password*) y nombre de usuario (*name*). Contiene la referencia a la tabla cargo (*chargeID*) donde a un usuario siempre le corresponde un cargo, lo mismo sucede con área (*areaID*) y role (*roleID*).

***role*:** Contiene los datos de los diferentes roles existentes en el sistema (*name*) y el identificador de dicha tabla (*ID*).

***charge*:** Contiene los datos de los diferentes cargos existentes en el sistema (*name*) y el identificador de dicha tabla (*ID*).

***area*:** Contiene los datos de las diferentes áreas existentes en el sistema (*name*) y el identificador de dicha tabla (*ID*).

***request*:** Almacena las diferentes peticiones generadas por un usuario determinado y los datos que la identifican como fecha de creación de la petición (*startDate*), fecha de culminación de la petición (*endingDate*), si es rechazada, vencida o finalizada (*state*), la justificación de rechazo en el caso que sea rechazada (*justification*) y su respectivo identificador (*ID).*

user_*owner*_request: Contiene los datos de la referencia de un usuario que crea determinada petición, (*userID*) contra (*requestID*).

user_list_*request*: Contiene las referencias de los usuarios de la lista de usuarios a notificar de determinada petición, (*userID*) contra (*requestID*).

user_*visible*_request: Contiene las referencias de las peticiones borradas por determinado usuario, (*userID*) contra (*requestID*).

***flowitem*:** Almacena el flujo de una petición desde que es creada, validando sus diferentes estados y manejando el orden de los diferentes usuarios involucrados con dicha petición.

***state*:** Contiene los datos de las decisiones de los usuarios sobre determinadas peticiones, es decir si las han firmado, rechazado o se han vencido.

2.10 MODELO DE DESPLIEGUE

Un modelo de despliegue consiste en una representación estructural de la arquitectura del sistema desde el punto de vista de la distribución de los artefactos del software en los destinos de despliegue; definiendo a los artefactos como representaciones de elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo (Sarmiento, 2013). En la Figura 20 se muestra el diagrama de despliegue propuesto para el portafirmas digital de la UCI.

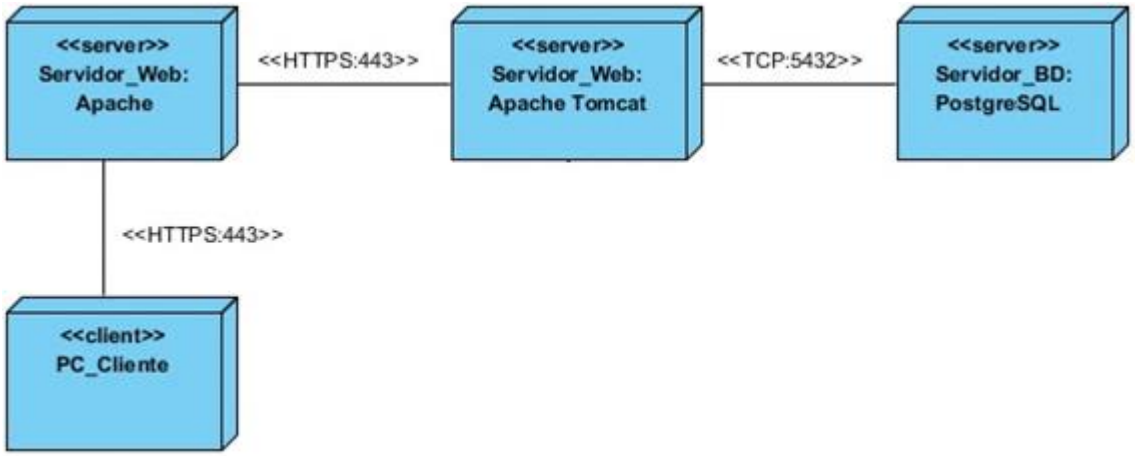

Figura 20: Diagrama de despliegue del Sistema Portafirmas-UCI. (*Elaboración propia*)

La PC_Cliente es donde el usuario interactúa con el sistema (Portafirmas UCI), el cual se conecta por el protocolo *HTTPS*:443 al Servidor_Web *Apache* donde está desplegado el cliente del sistema. El cliente del sistema se conecta por el protocolo *HTTPS*:443 al servidor de aplicaciones Servidor_Web *Apache Tomcat* donde está desplegado el paquete de servicio, el cual se conecta al Servidor_BD *PostgreSQL.* donde está la base de datos del sistema.

CONCLUSIONES DEL CAPÍTULO

- ✓ El diseño del modelo conceptual en la presente investigación posibilitó una mejor comprensión de los conceptos asociados a la problemática abordada.
- ✓ El análisis de las características del paquete de servicios permitió identificar los principales requisitos no funcionales y funcionales del Portafirmas UCI, estos últimos fueron agrupados y descritos mediante historias de usuarios.
- ✓ La utilización de los patrones de diseño y el estilo arquitectónico del paquete de servicios, evidenciaron el alto grado de resistencia ante posibles modificaciones que presenta la solución propuesta.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN

En el presente capítulo se describe la implementación de la solución propuesta, etapa donde se materializa el producto final y se cumple con los requisitos obtenidos al inicio de la investigación. Se obtiene el diagrama de componentes y se realizan las pruebas funcionales, de seguridad y rendimiento, validando y verificando la solución obtenida.

3.1 DIAGRAMA DE COMPONENTES

Un diagrama de componentes muestra los elementos de un diseño de un sistema de software. Permite visualizar la estructura de alto nivel del sistema y el comportamiento del servicio que estos componentes proporcionan y usan a través de interfaces. (Microsoft, 2018)

Lo que distingue a un diagrama de componentes de otros tipos de diagramas es su contenido. Normalmente contienen componentes, interfaces y relaciones entre ellos. Y como todos los diagramas, también puede contener paquetes utilizados para agrupar elementos del modelo.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes.

Dado que los diagramas de componentes muestran los componentes de software que constituyen una parte reusable, sus interfaces y sus interrelaciones, en muchos aspectos se puede considerar que un diagrama de componentes es un diagrama de clases a gran escala. Cada componente en el diagrama debe ser documentado con un diagrama de componentes más detallado, un diagrama de clases o un diagrama de casos de uso.

Un paquete en un diagrama de componentes representa una división física del sistema. Los paquetes se organizan en una jerarquía de capas donde cada capa tiene una interfaz bien definida. Un ejemplo típico de una jerarquía en capas de este tipo es: Interfaz de usuario; paquetes específicos de la aplicación; paquetes reusables; mecanismos claves; y paquetes hardware y del sistema operativo.

Un diagrama de componentes se representa como un grafo de componentes de software unidos por medio de relaciones de dependencia (generalmente de compilación). Puede mostrar también que un componente software contiene una interfaz, es decir, la soporta.

A continuación, en la Figura 21, se muestra el diagrama de componentes del paquete de servicios; cuya organización se encuentra acorde con el estilo arquitectónico Presentación Desacoplada propuesto por el *Framework Spring Boot* descrito en el capítulo 1 de este trabajo.

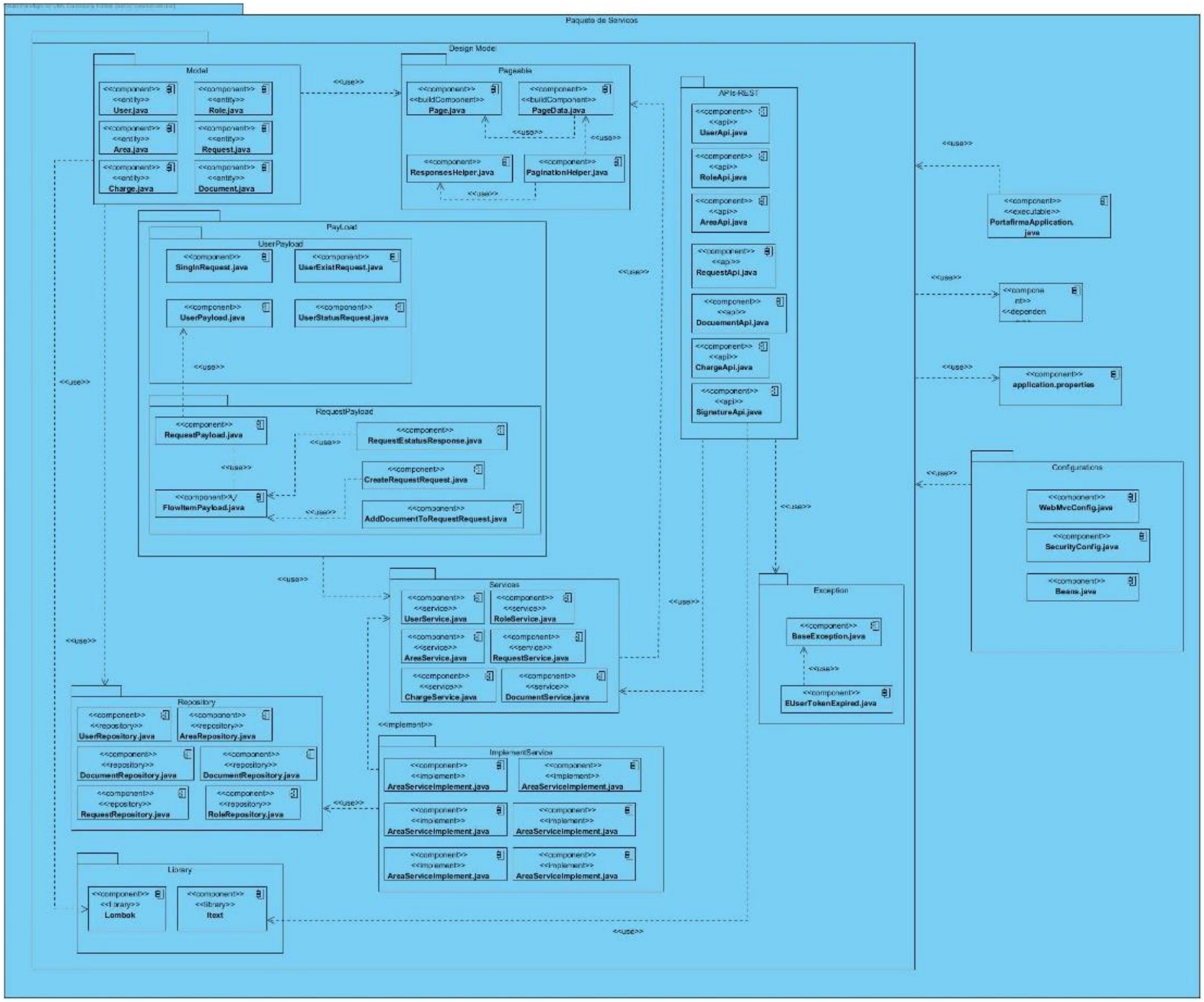

Figura 21: Diagrama de componentes del Paquete de Servicios del Portafirmas-UCI. (Elaboración propia)

A continuación, se describen los elementos que componen el diagrama de componentes mostrado.

Tabla 6: Descripción de los elementos más importantes del diagrama de componentes. (*Elaboración propia*)

Componentes		**Descripción**
Model	User.java Role.java Area.java Charge.java Request.java Document.java	Componente de las diferentes entidades del paquete de servicios.
Repository	UserRepository.java RoleRepository.java AreaRepository.java ChargeRepository.java RequestRepository.java DocumentRepository.java	Este componente es el encargado de transformar los datos de determinada entidad, a una entidad en una base de datos utilizando *JPA*.
Services	UserService.java RoleService.java AreaService.java ChargeService.java RequestService.java DocumentService.java	Este componente es una interfaz que recoge los servicios sobre determinada entidad.
	UserServiceImpl.java RoleServiceImpl.java AreaServiceImpl.java ChargeServiceImpl.java	Este componente es una clase que implementa los servicios sobre una entidad.

	RequestServiceImpl.java DocumentServiceImpl.java	
APIS-REST	UserApi.java RoleApi.java AreaApi.java ChargeApi.java RequestApi.java DocumentApi.java	Esta clase es la encargada de publicar las APIs y los respectivos servicios sobre una entidad para que sean consumidos por un cliente.
PayLoad	SingInRequest.java UserPayload.java UserExistRequest.java UserStatusRequest.java RequestPayload.java FlowItemPayload.java RequestStatusResponse.java CreateRequestRequest.java	Componente encargado en traducir una entidad en el paquete de servicios a una en el cliente y viceversa. Este componente es imprescindible en el intercambio de datos mediante JSON.
Pageable	Data.java PageData.java ResponseHelper.java PaginationHelper.java	Componente encargado en devolver listas de objetos que puedan ser paginados y ordenados por un cliente.
Exception	BaseException.java	Componente encargado en manejar las excepciones de los diferentes objetos en el paquete de servicios.

	EUserTokenExpired.java	Componente encargado en manejar los *tokens* de un usuario en el sistema.

3.2 ESTÁNDARES DE CODIFICACIÓN

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento (Microsoft, 2018). En la Tabla 7 se definen los estándares de codificación a utilizar en la implementación del paquete de servicios.

Tabla 7: Estándares de codificación a utilizar en la implementación del Portafirmas-UCI. (*Elaboración propia*)

Tipo de estándar	Descripción
Estructuración del código	✓ Las líneas de continuación deben alinearse verticalmente con el caracter que se ha utilizado (paréntesis, llaves, corchetes) ya que se utiliza *JAVA* como lenguaje de programación. ✓ La estructuración del código se realizará de forma jerárquica, cuando una línea depende de una sentencia estará ubicada dos espacios más adentro que la superior. ✓ El código en una página se organizará por bloques.
Máxima longitud de las líneas	Todas las líneas deben estar limitadas a un máximo de 100
Líneas en blanco	✓ Separar las funciones y definiciones de clases con una

	(1) línea en blanco. ✓ Las definiciones de métodos dentro de una clase deben separarse por una (1) línea en blanco. ✓ Se puede utilizar líneas en blanco escasamente para separar secciones lógicas.
Codificaciones	Utilizar la codificación *UTF*-8.
Importaciones	✓ Las importaciones deben estar en líneas unidas. ✓ Siempre deben colocarse al comienzo del archivo. ✓ Cada grupo de importaciones debe estar separado por una línea en blanco. ✓ Deben quedar agrupadas de la siguiente forma: • Importaciones de la librería estándar. • Importaciones locales de la aplicación / librerías.
Dependencias	Se utilizará el gestor de dependencia de la UCI (*NEXUS*), las dependencias quedarán agregadas en el fichero de configuración *POM* en formato *xml*.
Espacios en blanco en expresiones y sentencias	✓ Evitar utilizar espacios en blanco en las siguientes situaciones: • Inmediatamente dentro de paréntesis, corchetes y llaves. • Inmediatamente antes de una coma, un punto y coma o dos puntos. • Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función.

	• Inmediatamente antes de un corchete que empieza una indexación. • Al menos un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro. ✓ Deben rodearse con exactamente un espacio los siguientes operadores binarios: • Asignación (=). • Asignación de aumentación (+=, -=, etc.). • Comparación (==, <, >, >=, <=, !=, <>, *in*, *not in, is, is not*). • Expresiones lógicas (*and, or*, *not*). ✓ Si se utilizan operadores con prioridad diferente se aconseja rodear con espacios a los operadores de menor prioridad. ✓ No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una función o un parámetro con un valor por defecto.
Comentarios	✓ Los comentarios deben ser oraciones completas. ✓ Nunca cambiar las minúsculas y mayúsculas en los identificadores de clases, objetos, funciones, etc. ✓ Si un comentario es corto el punto final puede omitirse.
Comentarios en bloque	✓ Deben estar alineados al mismo nivel que el código a comentar. ✓ Cada línea de un comentario en bloque comienza con un numeral (#) y un espacio en blanco.

Comentarios en la misma línea	✓ Se recomienda utilizarlos escasamente. ✓ Se debe definir comenzando por un numeral (#) seguido de un espacio en blanco. ✓ Deben ubicarse en la misma línea que se desea comentar.

3.3 PRUEBAS

Durante la etapa de implementación, pueden cometerse algunos errores y pueden pasarse por alto algunos elementos que son importantes para el correcto funcionamiento del sistema. Por tal motivo, es esencial llevar a cabo la fase de validación y verificación del software, a través de varios tipos y métodos de pruebas (estrategias de pruebas).

Las pruebas del software intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el software, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o información de atributos no funcionales del programa. El software debe probarse desde dos perspectivas diferentes: la lógica interna del programa, que se comprueba utilizando técnicas de diseño de casos de prueba de caja blanca, y los requerimientos del software que se comprueban utilizando técnicas de diseño de casos de prueba de caja negra. En ambos casos, se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo. (Pressman, 2010.)

Tabla 8: Estrategia de prueba para el Portafirmas-UCI. (*Elaboración propia*)

Tipo de prueba	Técnica de prueba	Herramienta	Validación
Funcional	Casos de prueba (Caja Negra)	----------	Valida las funcionalidades diseñadas para el sistema
Seguridad	Caja Negra	Software *Acunetix*	Valida la confidencialidad, integridad y disponibilidad de los datos en el sistema

Carga y estrés	Caja Negra	Software *Apache JMeter*	Valida el comportamiento del sistema con distintos niveles de usuarios concurrentes y el consumo excesivo de sus recursos

3.3.1 Pruebas funcionales

Este tipo de pruebas tienen en cuenta el comportamiento externo del software, es decir, cómo funciona el sistema y se suelen utilizar técnicas de diseño de caja negra. Suelen ser las últimas pruebas que se realizan a la hora de dar el pase a producción a cualquier tipo de software. A este tipo de pruebas se les denomina también de Caja Negra, ya que los *Tester* se centran en analizar los datos de entrada y salida para definir unos casos de prueba que estarán listos antes del inicio de estas. (Peño, 2015)

Según (Pressman, 2010.) la prueba funcional se centra en comprobar que los sistemas desarrollados funcionan acorde a las especificaciones funcionales y requisitos del cliente, con el objetivo de validar que las funcionalidades implementadas funcionen correctamente y cumplan con los requisitos definidos con anterioridad. Para la ejecución de este tipo de pruebas, suelen emplearse dos métodos fundamentales, el método de Caja Blanca y el método de Caja Negra. Este servicio ayuda a detectar los posibles defectos derivados de errores en la fase de programación.

Tabla 9: Variables para los casos de prueba funcional gestionar petición. (*Elaboración propia*)

Variable	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Asunto	Campo de texto	No	Permite todos los caracteres.
2	Fecha de caducidad	Campo seleccionable	No	Permite seleccionar una fecha determinada en un calendario.

3	Documento	Campo de archivo de entrada	No	Permite seleccionar la ruta de un documento “pdf”.
4	Tipo de flujo	Campo seleccionable	No	Puede tomar los valores: por Usuario o por Cargo.
5	Filtrar por tipo de flujo	Campo de búsqueda	No	Permite buscar con autocompletamiento los usuarios o los cargos que se insertarán automáticamente en el listado de flujo de firmado.

Se diseñó un conjunto de casos de pruebas, (4) en total, referentes a varias de las historias de usuarios obtenidas en la fase de diseño del capítulo anterior, pertenecientes a requisitos funcionales de prioridad alta, también especificados en dicho capítulo. A continuación, se muestra uno (1) de los casos de prueba mencionados. Las celdas de la Tabla 10 contienen V (indica válido), I (indica inválido) y N/S (No es necesario llenar).

Tabla 10: Caso de prueba del RF13_Crear ejercicios de diseño. (*Elaboración propia*)

Caso de prueba 1: SC RF9_Crear petición de firma								
Condiciones de ejecución: El usuario debe tener asignado el rol necesario para acceder a la funcionalidad.								
Escenario	**Descripción**	**1**	**2**	**3**	**4**	**5**	**Respuesta del sistema**	**Flujo Central**
EC 1.1 Crear petición de	Interfaz con un formulario para llenar los datos de	V Petición 1	V 01/05/2018	V Manual_de	V por Usuario	V Anaili Pérez, Adiel	Agrega la petición.	1. Seleccionar, en el formulario

forma correcta	la petición, si todos son correctos, se agrega la petición al sistema			_preb as.pdf		Alfons o, Claudi a Gutier rez		el servicio “Crear petición” 2. Llenar los campos correspon dientes en JSON, selecciona r la opción “Enviar”
EC 1.2 Crear petición con campos vacíos	Interfaz con un formulario para llenar los datos de la petición, si existe algún campo vacío, se muestra un mensaje pidiendo llenar el campo en el caso del cliente.	I	I	V	V	I	Comprueba si los campos están vacíos, si lo están, muestra un mensaje que solicita llenarlos	
				Manu al_de _preb as.pdf	por Cargo			
EC 1.3 Crear petición con un documento en formato distinto a “pdf”	Interfaz con un formulario para llenar los datos de la petición, si se inserta un documento con formato distinto a	V	V	I	V	V	Comprueba el formato del documento, si es distinto a “pdf”, muestra un mensaje de error	
		Petici ón 2	03/05/ 2018	Manu al_de _preb as.do cx	por Usuar io	Anaili Pérez , Adiel Alfons o, Claudi a		

	“pdf”, se muestra un mensaje de error					Gutierrez		

Resultados de las pruebas:

Se detectaron en la primera iteración (8) no conformidades de funcionalidad, fueron resueltas (6) y (2) quedaron pendientes. En la segunda iteración se detectaron (6) no conformidades de funcionalidad, (2) de la iteración anterior y (4) nuevas en la actual iteración, todas fueron solucionadas. En una tercera iteración no se detectaron no conformidades.

La siguiente gráfica muestra los resultados de las pruebas funcionales:

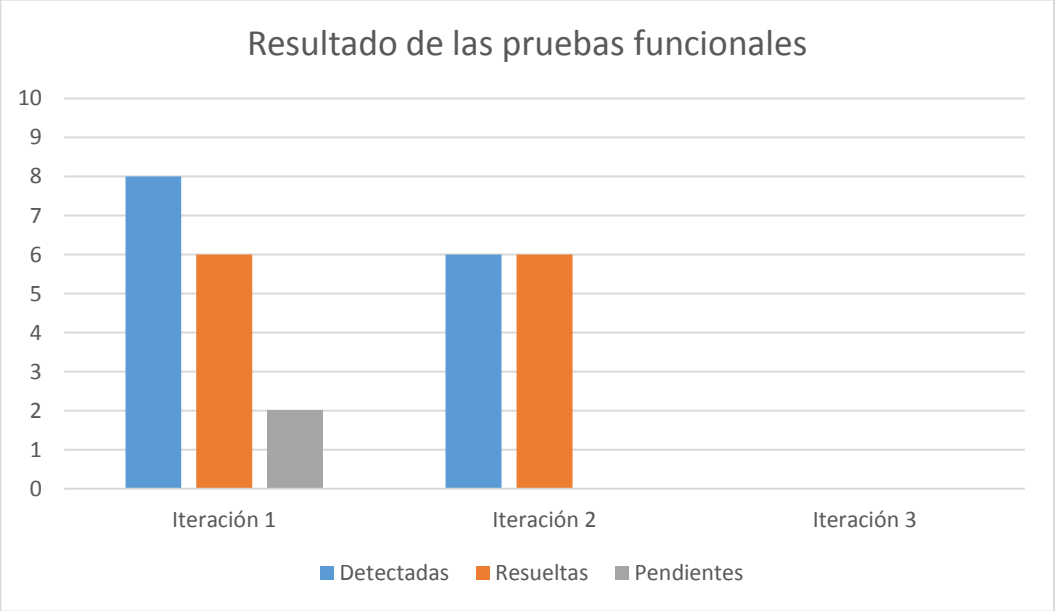

Figura 22: Resultado de las pruebas funcionales. (Elaboración propia)

3.3.2 Pruebas de seguridad

La prueba de seguridad intenta verificar que los mecanismos de protección que se construyen en un sistema lo protegerán de cualquier irrupción inapropiada. Buscan medir la confidencialidad, integridad y disponibilidad de los datos. Se diseñan para detectar las vulnerabilidades del entorno en el lado cliente, las

comunicaciones de red que ocurren conforme los datos pasan de cliente a servidor y viceversa, y el entorno del lado servidor. Cada uno de estos dominios puede atacarse, y es tarea del examinador de seguridad descubrir las debilidades que puedan explotar quienes tengan intención de hacerlo. (Pressman, 2010.)

Utilizando la herramienta *Acunetix* Web *Vulnerability Scanner.* Inicialmente se detectaron (6) de nivel medio, todas relacionadas con el uso de protocolo no seguro en el envío de datos hacia el cliente y viceversa. Se detectó (2) de nivel bajo relacionadas con el acceso a directorios sin pasar la autenticación.

Todas fueron solucionadas en una primera iteración. En una segunda iteración no se detectaron no conformidades.

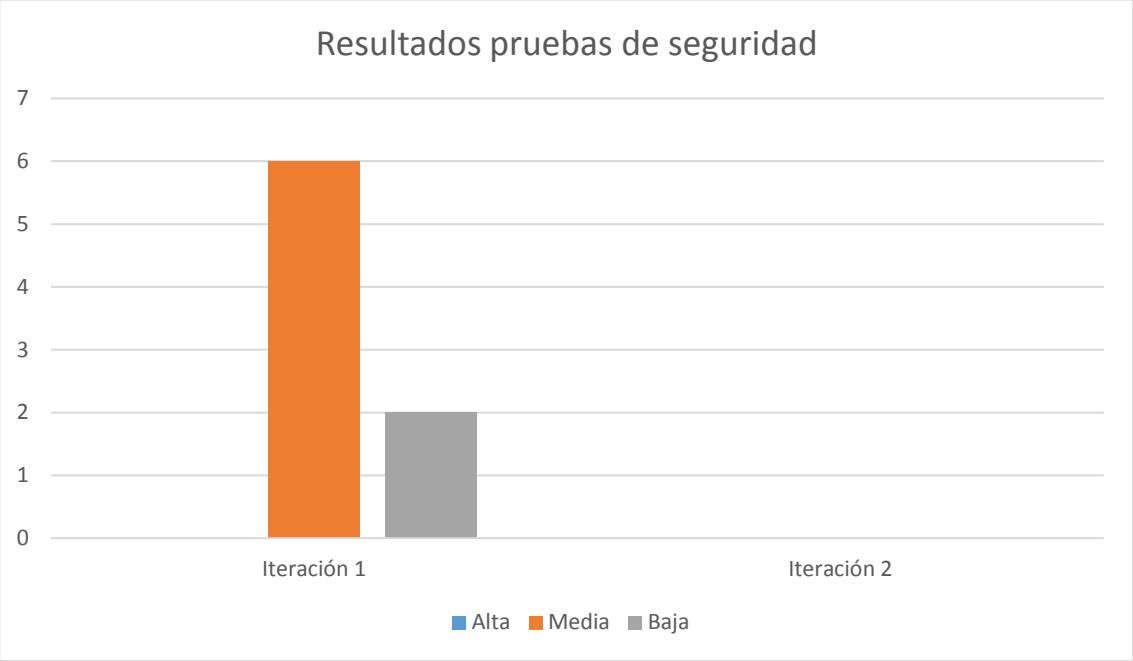

Figura 23: Resultados de las pruebas de seguridad.(Elaboración propia)

3.3.3 Pruebas de Carga y Estrés

La prueba de rendimiento se diseña para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado de rendimiento de software. Se realizan para medir la respuesta de la aplicación a distintos volúmenes de carga esperados, se centra en determinar la velocidad con la que el sistema bajo pruebas realiza una tarea en las condiciones particulares del escenario de pruebas. (Pressman, 2010.)

La prueba de carga es para determinar y validar la respuesta de la aplicación cuando es sometida a una carga de usuarios y/o transacciones que se esperan en el ambiente de producción. Estas pruebas consisten en simular una carga de trabajo similar y superior a la que tendrá cuando el sitio está en funcionamiento, con el fin de detectar si el software instalado cumple con los requerimientos de muchos usuarios simultáneos y también si el hardware (servidor y el equipamiento computacional de redes y enlace que lo conecta a Internet) es capaz de soportar la cantidad de visitas esperadas. (Pressman, 2010.)

La prueba de estrés es para encontrar el volumen de datos o de tiempo en que la aplicación comienza a fallar o es incapaz de responder a las peticiones. Son pruebas de carga o rendimiento, pero superando los límites esperados en el ambiente de producción y/o determinados en las pruebas. Las pruebas de estrés evalúan la robustez y la confiabilidad del software sometiéndolo a condiciones de uso extremas. (Pressman, 2010.)

Empleando la herramienta *Apache JMeter*, descrita en el Capítulo 1. Las pruebas se realizaron desde un ordenador con 4*GB* de *RAM*, microprocesador *Intel Core* i3 2120 con 3.16 *GHz* y sistema operativo *Windows* 10 64 *bits*. Se utilizaron las siguientes variables que miden el resultado de las pruebas de carga y estrés realizadas al paquete de servicios:

Muestra: Cantidad de peticiones realizadas para cada *URL*.

Media: Tiempo promedio en milisegundos en el que se obtienen los resultados.

Mediana: Tiempo en milisegundos en el que se obtuvo el resultado que ocupa la posición central.

Min: Tiempo mínimo que demora un hilo en acceder a una página.

Max: Tiempo máximo que demora un hilo en acceder a una página.

Línea 90 %: Máximo tiempo utilizado por el 90 % de la muestra, al resto de la misma le llevo más tiempo.

% Error: Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.

Rendimiento (*Rend*): El rendimiento se mide en cantidad de solicitudes por segundo.

***Kb/s*:** Velocidad de carga de las páginas.

Se simularon las peticiones realizadas al paquete de servicios por un total de 100, 200, 300 y 400 usuarios simultáneamente en cada caso, los cuales realizan hasta 3 peticiones por segundo. Se obtuvieron los siguientes resultados:

Tabla 11: Resultado de las pruebas de carga y estrés.(*Elaboración propia*)

Usuarios	Muestra	Media	Mediana	Min	Max	Línea 90%	%Error	*Rend*	*Kb/S*
100	300	1290	1142	166	3016	1984	0.00%	32.2	86.6
200	600	1320	1203	198	3789	2026	0.80%	66.1	162.7
300	900	4110	1019	102	4050	2200	19.22%	267.4	221.9
400	1200	12010	1256	188	4112	1562	60.62%	366.9	361.2

Los resultados de las pruebas realizadas muestran que el paquete de servicios es capaz de responder a 300 peticiones de 100 usuarios conectados simultáneamente en un tiempo promedio de 1290 milisegundos con 0 % de error, esto evidencia que el paquete de servicios puede procesar la carga esperada. Con 200 usuarios se comportó de forma media, donde no pudo atender todas las peticiones con 0.80% de error.

Esto demuestra que el módulo puede procesar la carga esperada, aunque no fue capaz de responder correctamente el 0.80% de las peticiones realizadas.

3.4 VALIDACIÓN DE LA HIPÓTESIS (CRITERIO DE EXPERTOS)

Para realizar la validación de la investigación se aplicó el método Criterio de expertos a través del cumplimiento de los pasos siguientes:

- ✓ Identificación de los posibles expertos.
- ✓ Determinación del coeficiente de competencia de los candidatos a expertos.
- ✓ Selección de los expertos.
- ✓ Realización del cuestionario a los expertos.
- ✓ Aplicación del escalamiento de *Likert*.

Identificación de los posibles expertos: Se tuvieron en cuenta, la experiencia profesional en el tema, de modo que estuvieran en capacidad de ofrecer valoraciones y hacer recomendaciones pertinentes, en relación con los aspectos que le serían consultados.

Determinación del coeficiente de competencia de los candidatos a expertos: Una vez identificados los posibles expertos, se les realizó una encuesta para valorar el grado de conocimiento y el grado de influencia que cada una de las fuentes ha tenido en ese conocimiento y criterio de los encuestados ver Anexo 4. El

procedimiento empleado para determinar el coeficiente de competencia de los candidatos a expertos, así como los resultados obtenidos pueden ser consultados en los Anexos 5 y 6 respectivamente.

Selección de expertos: De una cantidad inicial de **(6)** posibles expertos se seleccionaron los que su coeficiente de competencias fue igual o superior a 0.7, para un total de **(4)**, detallados en la Tabla 12. De los expertos, **(3)** obtuvieron un coeficiente de competencia alto y **(1)** de ellos medio.

Los expertos poseen suficiente conocimiento en la investigación y aplicaciones web, todos graduados universitarios y una gran parte de ellos ha obtenido títulos de formación académica.

Tabla 12: Expertos seleccionados en la validación de la investigación. (*Elaboración propia*)

Expertos	Entidad	Años de experiencia
Joelsy Porven Rubier	Redes y Servicios Telemáticos	8
Oscar Lázaro Garcés Pérez	Dirección de Seguridad Informática	7
Michel James Navarro	Dirección de Seguridad Informática	6
Henry Raúl González Brito	Dirección de Seguridad Informática	12

Realización del cuestionario a los expertos: Después de contar con los **(4)** expertos seleccionados se les realizó un cuestionario compuesto por **(5)** preguntas evaluativas que permitan conocer la opinión de los expertos. Los parámetros evaluativos empleados fueron: MA (muy de acuerdo), DA (de acuerdo), Sí-No (ni de acuerdo ni en desacuerdo), ED (en desacuerdo) y CD (completamente en desacuerdo). El cuestionario y las respuestas dadas por los expertos pueden ser consultadas en los Anexos 7 y 8 respectivamente. Los parámetros evaluativos se cuantificaron a través de la siguiente escala:

- ✓ Muy de acuerdo (5)
- ✓ De acuerdo (4)
- ✓ Ni de acuerdo ni en desacuerdo (3)

- ✓ En desacuerdo (2)
- ✓ Completamente en desacuerdo (1)

Aplicación del escalamiento de *Likert:* Para el procesamiento de los resultados se empleó un método que consiste en identificar la frecuencia en cada categoría de la escala de *Likert* definida en el cuestionario realizado y se calculan los por cientos de concordancia de cada categoría de acuerdo a las características propuestas por el autor. Luego se calcula en un índice porcentual (IP), que integra en un solo valor la aceptación del grupo de evaluadores sobre las características de la aplicación, como indica la siguiente fórmula:

$$IP = \frac{5(\%\ de\ MA) + 4(\%\ de\ DA) + 3(\%\ de\ Si - No) + 2(\%\ de\ ED) + 1(\%\ de\ CD)}{5}$$

La Figura 24 muestra un gráfico con el índice porcentual de los expertos en cada una de las preguntas, que como se observa sobrepasan el valor de 80. El procesamiento realizado a través del escalamiento de *Likert* evidencia que tanto los elementos teóricos de la investigación como la propuesta de solución, tienen una alta valoración por parte de los expertos. Durante el proceso se constataron criterios favorables en el uso de la aplicación para facilitar el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.

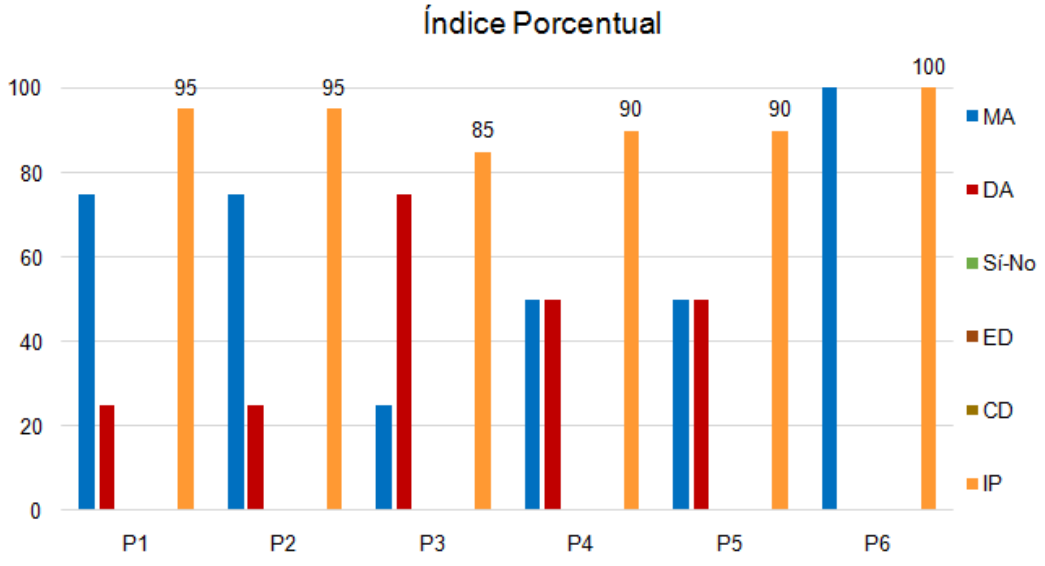

Figura 24: Resultados de la aplicación de la escala de Likert. (Elaboración propia)

CONCLUSIONES DEL CAPÍTULO

- ✓ La definición de los estándares de codificación utilizados en la implementación del paquete de servicios permitió uniformidad, claridad y fácil entendimiento del código.
- ✓ El diagrama de componentes presentado permitió una mayor apreciación de la lógica y la organización de los componentes existentes ayudando a su implementación.
- ✓ El desarrollo de las pruebas detectó no conformidades, las cuales fueron corregidas en cada iteración, asegurando una solución aceptada en todo momento de la implementación.
- ✓ El diagrama de despliegue presentado permitió una mejor representación estructural de la arquitectura del sistema definiendo a los artefactos como representaciones de elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo.

CONCLUSIONES

- ✓ La propuesta de solución mejoró la seguridad en el proceso de autenticación de documentos digitales, al mantener un mayor control sobre los documentos digitales manejados en la UCI.
- ✓ La firma de documentos digitales desde una aplicación web mejoró el proceso de autenticación de documentos digitales en la universidad.
- ✓ La firma de documentos por lote y la creación de un flujo de firma mediante peticiones, permitió la mejora del proceso de autenticación de documentos digitales en la UCI.
- ✓ El despliegue del Paquete de Servicios, permitió el consumo de los servicios publicados, a cualquier sistema que requiera información referente al proceso de autenticación de documentos digitales en la UCI.
- ✓ La posibilidad de conocer y consultar en cualquier momento la fecha de caducidad de los documentos a firmar favoreció la eficacia del proceso.
- ✓ La validación de la hipótesis de investigación, a través del método de criterio de expertos con escalamiento de *Likert*, evidenció el correcto cumplimiento de los objetivos planteados en la presente investigación.

RECOMENDACIONES

Para el desarrollo de futuras investigaciones relacionadas con la presente, se propone:

1. Incluir soporte para firmar digitalmente documentos electrónicos en otros formatos.
2. Integrar el paquete de servicios desarrollado al sistema de gestión documental eXcriba para que sea utilizado en su flujo de procesos.

REFERENCIAS BIBLIOGRÁFICAS

Ajpdsoft. 2016. Ajpdsoft. [En línea] 2016. [Citado el: 22 de Enero de 2017.] http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=769.

Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. 1996. *HANDBOOK OF APPLIED CRYPTOGRAPHY.* Massachusetts : Massachusetts Institute of Technolog, 1996.

Apache Foundation, Software. 2018. Apache JMeter. [En línea] 2018. [Citado el: 14 de Abril de 2018.] http://jmeter.apache.org//.

Auth0. 2017. Introduction to JSON Web Tokens. [En línea] 2017. [Citado el: 2 de diciembre de 2017.] https://jwt.io/introduction/.

Bondartchuk, Roberto Quiñones. 2009. *Firma Digital de Documentos utilizando Smart Cards.* La Habana : s.n., 2009. Tesis de Grado.

Calendamaia. 2014. Genbeta:dev. [En línea] enero de 2014. [Citado el: 4 de Diciembre de 2017.] https://www.genbetadev.com/herramientas/netbeans-1.

CISCO. 2016. CISCO. *CISCO.* [En línea] 2016. [Citado el: 20 de Enero de 2018.] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwjOgoW844XbAhVPmeAKHc_EDLUQFgguMAE&url=https%3A%2F%2Fwww.cisco.com%2Fc%2Fen%2Fus%2Ftd%2Fdocs%2Fswitches%2Flan%2Fcatalyst3750x_3560x%2Fsoftware%2Frelease%2F15-2_4_e%2Fconfigurationgu.

Corrales, F.J.I. 2011. *Metodología Investigativa y Tradicional.* s.l. : ISSN 1988-6047., 2011.

Denzer, Patricio. 2002. *PostgreSQL.* U.T.F.S.M. : s.n., 2002.

dtgjhtg. sferf. [En línea] [Citado el: 14 de 03 de 2005.] httpsdg.

El Lenguaje Unificado de Modelado. **Orallo, Enrique Hernández. 2002.** 2002.

EuroLogic. 2010. EuroLogic. [En línea] 15 de Enero de 2010. [Citado el: 25 de Noviembre de 2017.] www.eurologic.es/soluciones/que-es-pki.html.

Evaluación y revisión, . 2009. *Evaluación y revisión.* Sevilla, España : Universidad Pablo de Olavide, Vicerrectorado de tecnologías de la información y la comunicación, 2009.

Faraoni, Federico Julián Gutierrez. 2014. *Desarrollo de una aplicación webcon spring frameworkpara un gestor de un recetario.* Madird : Universidad Politécnica de Madrid, 2014. Proyecto de fin de Grado.

Fielding, Roy Thomas. 2000. *Architectural Styles and the Design of Network-based Software Architectures.* University of California, Irvine : s.n., 2000.

Figueroa, R.G., Solís, C.J. y Cabrera, A.A. 2007. *Metodologías tradicionales vs. metodologías ágiles.* Universidad Técnica Particular de Loja : s.n., 2007.

Git. 2016. Git - Control de versione. [En línea] 2016. [Citado el: 4 de diciembre de 2017.] https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones. .

Grosso, A. 2011. Prácticas de software, Experiencias sobre la Ingeniería y Management del Software. [En línea] 2011. [Citado el: 6 de marzo de 2018.] https://ar.linkedin.com/in/andresgrosso/es?trk=public_profile_card_url..

Gutierrez, Enmanuel. 2009. *JavaScript: Conceptos básicos y avanzados.* Barcelona: : Ediciones ENI. ISBN 978-2-7460-5220-8., 2009.

iText. 2018. Harness the power of PDF with iText. [En línea] 2018. [Citado el: 25 de Enero de 2018.] https://itextpdf.com/.

JetBrains. 2018. JetBrains. [En línea] 2018. [Citado el: 21 de Diciembre de 2017.] https://www.jetbrains.com/idea/.

K.U.Leuven. 1993. *CRYPTOGRAPHIC HASH FUNCTIONS:.* 1993.

Keith, Mike y Schnicariol., Merrick. 2009. *JPA Mastering the JAVA™ 2 Persistance API.* NY, USA : ISBN-13 (electronic): 978-1-4302-1957-6, 2009.

Lapuente, César Guzmán de. 2011. *Implantación de un sistema de certificados.* 2011.

Larman. 2004.. *UML y Patrones: Introducción al análisis y diseño orientado a objetos.* La Habana : Félix Varela, 2004.

Lombok, Project. 2018. Project Lombok. [En línea] 2018. [Citado el: 10 de Enero de 2018.] https://projectlombok.org/.

López, Manuel L José Lucena. 2010. *CRIPTOGRAFÍA Y SEGURIDAD EN COMPUTADORAS.* Jaén : s.n., 2010.

López, Marical. 2017. Acunetix Datasec. [En línea] 2017. [Citado el: 14 de Abril de 2018.] https://www.datasec-soft.com/es/acunetix. .

Merchán, L., Urrea, A. y Rebollar, R. 2008. *Definicion de una metodologia agil.* s.l. : ISSN 1794-192.. https://dialnet.unirioja.es/descarga/articulo/2753852.pdf., 2008.

Microsoft, © 2018. 2018. Microsoft Developer Netwoek. [En línea] 2018. [Citado el: 25 de 2018 de Enero.] https://msdn.microsoft.com/es-es/library/dd409390.aspx.

Millet, Juan Carlos Ceresola. 2012. *Material didáctico para la enseñanza de SPRING.* 2012.

Oracle. 2017. Oracle. [En línea] 2017. [Citado el: 2 de Diciembre de 2017.] https://www.oracle.com/mysql/index.html.

—. 2018. Oracle Help Center. [En línea] 2018. [Citado el: 2 de Diciembre de 2017.] https://docs.oracle.com/en/cloud/paas/api-platform-cloud/apfrm/index.html.

Oracle, Corporation. 2018. MaySql. [En línea] 2018. [Citado el: 3 de Diciembre de 2017.] https://www.mysql.com/products/workbench/.

Parkinson, S., Moriarty, K., Scott, M., Rusch, A. y Nystrom, M. 2014. *Personal Information Exchange Syntax.* 2014.

Peño, Jose Manuel Sánchez. 2015. *Pruebas de Software.Fundamentos teóricos.* Madrid : s.n., 2015.

PGP Corporation. 2002. *An Introduction to Cryptography.* 2002.

Pivotal. 2018. Spring. [En línea] 2018. https://spring.io/.

Port@firmas. 2017. Portal de Administración Electrónica. [En línea] 18 de mayo de 2017. https://ws024.juntadeandalucia.es/ae/adminelec/areatecnica/portafirmas. .

Portafirmas. 2017. Guadaltel. [En línea] 16 de noviembre de 2017. http://www.guadaltel.com/producto-portafirmas.

Pressman. 2010.. *Ingenieria del Software. Un Enfoque Práctico. 7ma.* s.l. : McGRAW-HILL Interamericana. ISBN 978-607-15-0314-5, 2010.

Rumbaugh, James. 2000. *EL LENGUAJE UNIFICADO DE MODELADO. MANUAL DE REFERENCIA.* 2000.

Salvador, J. 2015. METODOLOGIAS AGILES AUP. [En línea] 2015. [Citado el: 17 de novimebre de 2017.] http://www.academia.edu/7894130/METODOLOGIAS_AGILES_AUP..

Sánchez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

Sarmiento, F., Johana,. 2013. Visión General de los Diagramas de Despliegue. UML. [En línea] 2013. [Citado el: 2018 de marzo de 6.] http://umldiagramadespliegue.blogspot.com/..

Sommerville. 2011. *Ingeniería de Software.* México : ISBN 978-607-32-0603-7., 2011.

Stallings, Willians. 2005. *Stallings Cryptography and Network Security.* s.l. : Prentice Hall, 2005. ISBN-13: 978-0-13-187316-2.

Sun Microsystems, INC. 2016. *El lenguaje de Programación Java.* 2016.

Talens-Oliag, Sergio. 2006. Introducción a los certificados digitales. *InfoCentre.* [En línea] 2006. http://www.infocentre.gva.es/.

Tarrats, Jordi Buch i. 2017. *LA SEGURIDAD DE LAS TRANSACCIONES BANCARIAS EN INTERNET.* 2017.

TechTarget, . 2018. Search Security. [En línea] 2018. [Citado el: 25 de enero de 2018.] https://searchsecurity.techtarget.com/definition/Hash-based-Message-Authentication-Code-HMAC.

TechTerms. 2013. TechTerms. [En línea] 7 de Marzo de 2013. [Citado el: 10 de Noviembre de 2017.] https://techterms.com/definition/framework.

Ubuntu, Guia. 2014. Guia Ubuntu. [En línea] 30 de mayo de 2014. [Citado el: 15 de Noviembre de 2017.] www.guia-ubuntu.com.

UCI, Universidad de las Ciencias Informáticas,. 2015. *Metodología para el desarrollo de la actividad productiva en la UCI.* La Habana : Universidad de las Ciencias Informáticas, 2015.

Ugarte, Rafael Muruaga. 2013. *LA FIRMA ELECTRONICA.* 2013.

Valdéz, J.L.C. 2014. Implementación del modelo integral colaborativo (MDSIC) como fuente de innovación para el desarrollo ágil de software en las empresas de la zona centro-occidente en México. [En línea] 2014. [Citado el: 20 de Enero de 2017.] www.eumed.net/tesis-doctorales/2014/jlcv/jlcv.zip..

Viafirma, Viafirma Inbox |. 2017. Agenda de firmas electrónicas. [En línea] 2017 de mayo de 2017. [Citado el: 20 de Octubre de 2017.] https://www.viafirma.com/es/inbox-agenda-firmas-electronicas..

Visual Paradigm Guide User's, . 2017. Visual Paradigm User's Guide. [En línea] 2017. [Citado el: 23 de Octubre de 2017.] https://www.visual-paradigm.com/support/documents/vpuserguide.jsp.

Wells, Don. 2013. Extreme Programming:. [En línea] octubre de 2013. http://www.extremeprogramming.org/.

ANEXOS

ANEXO 1: ENTREVISTA AL CLIENTE PARA CONOCER LA NECESIDAD DEL DESARROLLO DE LA PROPUESTA DE SOLUCIÓN Y DEFINIR LOS REQUISITOS FUNCIONALES Y NO FUNCIONALES

Estimado especialista: Se necesita de su cooperación en una investigación para una tesis de pregrado. Por ello, sería de gran ayuda que respondiera lo siguiente:

1. ¿Considera que en la universidad el proceso de firmado de documentos digitales se realiza de la mejor manera?
2. Respecto a la manera en que se lleva a cabo el proceso de autenticación de documentos digitales en la UCI en la actualidad, ¿qué porciento de documentos se quedan sin firmar después de haber caducado la fecha establecida para ese documento?
3. ¿Cuáles son las causas que podrían estar influyendo en la descoordinación de ese proceso?
4. ¿Existe alguna herramienta en la Universidad para la firma digital de documentos?
5. ¿Cuáles son sus características?
6. A partir de estas características, ¿considera que la herramienta se ajusta a las necesidades actuales de la universidad?
7. ¿Cómo deberían firmarse digitalmente los documentos en la universidad?
8. ¿Qué otras características, considera que deba presentar el paquete de servicios, en cuanto a la disponibilidad, seguridad, fiabilidad u otro aspecto que garantice su calidad?

ANEXO 2: HISTORIAS DE USUARIO

Tabla 13: HU_1 Autenticar usuario. (*Elaboración propia*)

Historia de usuario	
Número: HU_1	**Nombre:** Autenticar usuario
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Acceso al sistema por un determinado usuario.	
Observaciones: Es necesario estar autenticado en el sistema para poder acceder al mismo, la autenticación se establecerá mediante el nombre de usuario y contraseña. El sistema luego de recibir y comprobar el usuario y la contraseña pertenecen a un usuario autenticado, devolverá los datos referentes a un usuario, para que las funcionalidades del cliente funcionen en consecuencia al usuario autenticado en ese momento, muchas solo serán accedidas por los usuarios que tengan permisos a trabajar sobre las mismas.	

Tabla 14: HU_2 Crear usuario. (*Elaboración propia*)

Historia de usuario	
Número: HU_2	**Nombre:** Crear usuario
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Insertar un nuevo usuario en la base de datos del sistema.	
Observaciones: Para crear un nuevo usuario en el sistema es necesario los datos referentes a un usuario, como el nombre de usuario, contraseña, nombre y apellidos de la persona, correo, área, cargo, rol y la activación.	

Tabla 15: HU_3 Editar usuario (*Elaboración propia*)

Historia de usuario	
Número: HU_3	**Nombre:** Editar usuario
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Media	
Descripción: Editar un usuario de la lista de usuarios registrados en la base de datos del sistema.	
Observaciones: En el caso que determinados datos de un usuario sean incorrectos, o hayan sido cambiados por determinado motivo, se hace necesario actualizarlos.	

Tabla 16: HU_4 Eliminar usuario. (*Elaboración propia*)

Historia de usuario	
Número: HU_4	**Nombre:** Eliminar usuario
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Baja	
Descripción: Borra un usuario de la lista de usuarios registrados en la base del sistema.	
Observaciones: En el caso de que un usuario fuera dado de baja de la aplicación o deje la universidad por determinada razón, debe ser eliminado de la lista de usuarios registrados, pero sus datos deben quedar guardados en la base de datos para tener un historial de sus evidencias en la aplicación.	

Tabla 17: HU_5 Buscar usuario. (*Elaboración propia*)

Historia de usuario	
Número: HU_5	**Nombre:** Buscar usuario
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Baja	

Descripción: Buscar un usuario en la base de datos del sistema.
Observaciones: Mediante un parámetro de búsqueda devolverá el usuario o los usuarios que contengan en sus datos elementos similares a ese parámetro de búsqueda.

Tabla 18: HU_6 Listar usuarios. (*Elaboración propia*)

Historia de usuario	
Número: HU_6	**Nombre:** Listar usuarios
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Devuelve los usuarios registrados en la aplicación.	
Observaciones: Para que el cliente pueda cumplir con los requisitos funcionales RF 3, RF 4 y las acciones que se pueden acometer sobre ellos, como es el caso de editar y eliminar es necesario devolver los usuarios registrados.	

Tabla 19: HU_7 Rechazar documento. (*Elaboración propia*)

Historia de usuario	
Número: HU_7	**Nombre:** Rechazar documento
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Rechazar un documento de una petición de firma en el sistema.	
Observaciones: En el caso de que un usuario no esté de acuerdo en firmar un documento, debe existir la opción de rechazarlo y registrar en el sistema su justificación. El sistema debe enviarles una notificación de correo electrónico al primer usuario firmante y al que realizó la petición de firma; además de registrar la fecha y hora del momento en que se rechazó y el usuario responsable.	

Tabla 20: HU_8 Mostrar documento. (*Elaboración propia*)

Historia de usuario	
Número: HU_8	**Nombre:** Mostrar documento
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Devuelve un documento de una petición de firma en el sistema.	
Observaciones: El sistema debe devolver el documento en el listado de peticiones de firma para que el usuario pueda revisarlo antes de firmarlo.	

Tabla 21: HU_10 Mostrar estado de la petición de firma. (*Elaboración propia*)

Historia de usuario	
Número: HU_10	**Nombre:** Mostrar estado de la petición de firma
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Devuelve el estado de una petición de firma en el sistema.	
Observaciones: Se devuelve todos los datos de la petición, para que el cliente extraiga el estado de una petición de firma (pendiente, terminada, cancelada), enviando además la fecha y hora en que firmó cada usuario perteneciente al flujo de firmado.	

Tabla 22: HU_11 Buscar petición de firma. (*Elaboración propia*)

Historia de usuario	
Número: HU_11	**Nombre:** Buscar petición de firma
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Media	
Descripción: Buscar una petición de firma determinada en el sistema.	

Observaciones: Mediante un parámetro de búsqueda brindará la posibilidad de buscar una determinada petición de firma en la lista de peticiones de firma.

Tabla 23: HU_12 Listar peticiones de firma. (*Elaboración propia*)

Historia de usuario	
Número: HU_12	**Nombre:** Listar peticiones pendientes de firma
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Devolver un listado de las peticiones de firma de un usuario determinado en el sistema.	
Observaciones: Es necesario devolver las peticiones pendientes de firma del usuario autenticado para que el cliente pueda cumplir con los requisitos funcionales RF 7, RF 8, RF 10, RF 13 y las acciones que se pueden realizar sobre ellas, rechazar documento, mostrar documento, mostrar estado de una petición de firma y firmar documento digital (.pdf) respectivamente. Además de buscar petición de firma que corresponde con el requisito funcional RF 11.	

Tabla 24: HU_13 Firmar documento digital (.pdf). (*Elaboración propia*)

Historia de usuario	
Número: HU_13	**Nombre:** Firmar documento digital (.pdf)
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Permite firmar digitalmente un documento oficial en formato “.pdf”.	
Observaciones: Cada usuario del sistema tiene asociado un certificado digital validado por la Autoridad Certificadora de la UCI. Para firmar el documento debe introducir un archivo “.p12” y la contraseña de acceso al mismo. Esta contraseña debe ser cifrada haciendo uso de la criptografía híbrida combinando los algoritmos *AES* y *RSA*. Puede firmar los documentos de manera individual o en lote.	

Tabla 25: HU_14 Listar peticiones terminadas. (*Elaboración propia*)

Historia de usuario	
Número: HU_14	**Nombre:** Listar peticiones terminadas
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Devolver todas los peticiones que un usuario determinado ha firmado o rechazado.	
Observaciones: El sistema debe devolver las peticiones que han sido firmadas o rechazadas por el usuario autenticado para que el cliente pueda mostrar un listado de dichas peticiones en la bandeja de peticiones “Terminadas”. Al cumplirse este requisito el cliente podrá realizar acciones por cada petición que permitan dar cumplimiento a los requisitos funcionales RF 8, RF 10 y RF 15; estas acciones son: mostrar documento, mostrar estado de una petición de firma y descargar documento respectivamente. También debe permitir buscar una petición en la lista de peticiones terminadas, correspondiente al requisito funcional RF 11.	

Tabla 26: HU_15 Descargar documento. (*Elaboración propia*)

Historia de usuario	
Número: HU_15	**Nombre:** Descargar documento
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Media	
Descripción: Descargar un documento firmado o rechazado por el usuario autenticado.	
Observaciones: El sistema debe devolver un documento para que el cliente pueda mostrar la opción de descargar documento como una acción en el listado de documentos terminados que se muestra en la bandeja de peticiones “Terminadas” para que dicho documento pueda continuar con su ciclo de vida en otro proceso de la universidad.	

Tabla 27: HU_16 Listar peticiones creadas. (*Elaboración propia*)

Historia de usuario	
Número: HU_16	**Nombre:** Listar peticiones creadas
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Devolver las peticiones que el usuario autenticado ha creado en el sistema.	
Observaciones: El sistema debe devolver las peticiones que han sido creadas por el usuario autenticado para que el cliente pueda mostrar un listado en la bandeja de "Mis peticiones". Luego de que el cliente reciba estos datos podrá mostrar acciones por cada petición que permitan dar cumplimiento a los requisitos funcionales RF 8 y RF 10; estas acciones son: mostrar documento y mostrar estado de una petición de firma. También debe permitir buscar una petición en la lista de peticiones creadas, correspondiente al requisito funcional RF 11.	

Tabla 28: HU_17 Listar peticiones en espera de firma. (*Elaboración propia*)

Historia de usuario	
Número: HU_17	**Nombre:** Listar peticiones en espera de firma
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Devolver las peticiones en las que el usuario autenticado está definido como usuario firmante dentro del flujo de firma de la petición.	
Observaciones: Necesario para que el cliente muestre en la bandeja de peticiones "En Espera" el listado de peticiones en espera de firma del usuario autenticado con la acción de mostrar estado de una petición correspondiente al requisito funcional RF 10. Una vez que al usuario autenticado le llegue el turno de firmar el documento dentro del flujo de firma, la petición pasa automáticamente a la bandeja de peticiones "Pendientes".	

Tabla 29: HU_18 Crear área. (*Elaboración propia*)

Historia de usuario	
Número: HU_18	**Nombre:** Crear área
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Insertar una nueva área en la base de datos del sistema.	
Observaciones: El sistema debe brindar la posibilidad de insertar un área en el listado de áreas del sistema introduciendo el nombre del área.	

Tabla 30: HU_19 Editar área. (*Elaboración propia*)

Historia de usuario	
Número: HU_19	**Nombre:** Editar área
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Baja	
Descripción: Permite editar un área de la lista de áreas registradas en la base de datos del sistema.	
Observaciones: En el caso que los datos de un área sean incorrectos, o hayan sido cambiados por determinados factores, se hace necesario actualizarlos.	

Tabla 31: HU_20 Eliminar área. (*Elaboración propia*)

Historia de usuario	
Número: HU_20	**Nombre:** Eliminar área
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Baja	
Descripción: Borra un área de la lista de áreas registradas en la base de datos del sistema.	

Observaciones: Solo se eliminaran las áreas que no tengan un usuario asignado.

Tabla 32: HU_21 Buscar área. (*Elaboración propia*)

Historia de usuario	
Número: HU_21	**Nombre:** Buscar área
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Baja	
Descripción: Buscar un área determinada en la aplicación.	
Observaciones: Mediante un parámetro se buscará una determinada área en la lista de áreas del sistema.	

Tabla 33: HU_22 Listar áreas. (*Elaboración propia*)

Historia de usuario	
Número: HU_22	**Nombre:** Listar áreas
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Devolver las áreas registradas en el sistema.	
Observaciones: Necesario para que el cliente pueda cumplir con los requisitos funcionales RF 20 y RF 21 mostrando las áreas registradas y las acciones que se pueden acometer sobre ellas, como es el caso de editar y eliminar respectivamente. Al devolver los datos referentes a las áreas el cliente debe mostrar la opción de crear una nueva área y buscar un área dentro del listado de áreas correspondiente a los requisitos funcionales RF 19 y RF 22.	

Tabla 34: HU_23 Crear cargo. (*Elaboración propia)*

Historia de usuario		
Número: HU_23	**Nombre:** Crear cargo	
Programador responsable: Adiel Alfonso Cordovi		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Insertar un nuevo cargo en la base de datos del sistema.		
Observaciones: El sistema debe brindar la posibilidad de insertar un cargo en el listado de cargos del sistema introduciendo el nombre del cargo.		

Tabla 35: HU_24 Editar cargo. (*Elaboración propia*)

Historia de usuario		
Número: HU_24	**Nombre:** Editar cargo	
Programador responsable: Adiel Alfonso Cordovi		**Iteración asignada:** 1
Prioridad en negocio: Baja		
Descripción: Permite editar un cargo de la lista de cargos registrados en la base de datos del sistema.		
Observaciones: En el caso que los datos de un cargo sean incorrectos, o hayan sido cambiados por factores ajenos, se hace necesario actualizarlos. El administrador del sistema es el único que debe tener permiso para modificar estos datos.		

Tabla 36: HU_25 Eliminar cargo. (*Elaboración propia*)

Historia de usuario		
Número: HU_25	**Nombre:** Eliminar cargo	
Programador responsable: Adiel Alfonso Cordovi		**Iteración asignada:** 1
Prioridad en negocio: Baja		

Descripción: Borrar un cargo de la lista de cargos registrados en la base de datos del sistema.
Observaciones: Solo podrán ser eliminados los cargos que no tengan un usuario asociado.

Tabla 37: HU_26 Buscar cargo. (*Elaboración propia*)

Historia de usuario	
Número: HU_26	**Nombre:** Buscar cargo
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Baja	
Descripción: Devolver un cargo determinado en el sistema.	
Observaciones: Mediante un parámetro se buscará una determinada área en la lista de áreas del sistema.	

Tabla 38: HU_27 Listar cargos. (*Elaboración propia*)

Historia de usuario	
Número: HU_27	**Nombre:** Listar cargos
Programador responsable: Adiel Alfonso Cordovi	**Iteración asignada:** 1
Prioridad en negocio: Alta	
Descripción: Devolver las cargos registrados en el sistema.	
Observaciones: Necesario para que el cliente pueda cumplir con los requisitos funcionales RF 25 y RF 26 para mostrar los cargos registrados y las acciones que se pueden acometer sobre ellos, como es el caso de editar y eliminar respectivamente. Al devolver los datos referentes a los cargos el cliente debe mostrar las opciones de crear un nuevo cargo y buscar un cargo determinado correspondientes a los requisitos funcionales RF 24 y RF 27. A esta funcionalidad solo tiene acceso el administrador del sistema.	

ANEXO 3: PRUEBAS FUNCIONALES.

Tabla 39 Descripción de las variables del caso de prueba 2. (*Elaboración propia*)

Variable	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Archivo ".p12"	Campo de archivo de entrada	No	Permite seleccionar la ruta de un certificado ".p12".
2	Contraseña	Campo de texto	No	Permite todos los caracteres.

Tabla 40: Caso de prueba del RF13_Firmar digitalmente un documento (.pdf) . (Elaboración propia)

Caso de prueba 2: SC RF13_Firmar digitalmente un documento (.pdf)					
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con cualquiera de los roles posibles para acceder a la funcionalidad.					
Escenario	**Descripción**	**1**	**2**	**Respuesta del sistema**	**Flujo Central**
EC 1.1 Firmar digitalmente un documento de manera correcta	Interfaz con un formulario para ingresar los datos necesarios para firmar el (los) documento(s) seleccionado(s), si todos son correctos,	V mi_certificado.p12	V **********	Firma el (los) documento(s) y muestra un mensaje de notificación	1. Seleccionar, en la bandeja de peticiones Pendientes, los documentos que se desean firmar y

	se firma el documento				presionar la opción “Firmar”
EC 1.2 Firmar documento con campos vacíos	Interfaz con un formulario para ingresar los datos necesarios para firmar el (los) documento(s) seleccionado(s), si existe algún campo vacío, se muestra un mensaje pidiendo llenar el campo	I	I	Comprueba si los campos están vacíos, si lo están, muestra un mensaje que solicita llenarlos	2. Llenar los campos correspondientes en el formulario y seleccionar la opción “Firmar”
EC 1.3 Firmar documento con una contraseña incorrecta	Interfaz con un formulario para ingresar los datos necesarios para firmar el (los) documento(s) seleccionado(s), si se inserta una contraseña que no corresponde con el certificado, se muestra un mensaje de error	V mi_certificado.p12	I *****	Comprueba que la contraseña sea la que corresponde con ese certificado, si es incorrecta, muestra un mensaje de error	

Tabla 41: Descripción de las variables del caso de prueba 3. (*Elaboración propia*)

Variable	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Nombre del área	Campo de texto	No	Permite solo los caracteres que sean letras, números y espacios.

Tabla 42: Caso de prueba del RF19_Crear área. (*Elaboración propia*)

Caso de prueba 3: SC RF19_Crear área				
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con el rol necesario para acceder a la funcionalidad.				
Escenario	**Descripción**	**1**	**Respuesta del sistema**	**Flujo Central**
EC 1.1 Crear un área de manera correcta	Interfaz con un formulario para ingresar el dato necesario para crear un área, si es correcto, se crea el área	V Facultad 1	Crea el área y muestra un mensaje de notificación	1. Seleccionar, en el menú superior, la opción “Áreas” y presionar el botón “Crear área” 2. Llenar los campos correspondientes en el formulario y seleccionar la opción “Enviar”
EC 1.2 Crear un área con campos vacíos	Interfaz con un formulario para ingresar el dato necesario para crear un área, si existe algún campo vacío, se	I	Comprueba si los campos están vacíos, si lo están, muestra un mensaje que solicita llenarlos	

	muestra un mensaje pidiendo llenar el campo			
EC 1.3 Crear un área que ya existe	Interfaz con un formulario para ingresar el dato necesario para crear un área, si se inserta un área que ya existe en la base de datos, se muestra un mensaje de error	V Facultad 1	Comprueba que el área que se inserte no exista en la base de datos, si existe, muestra un mensaje de error	

Tabla 43: Descripción de las variables del caso de prueba 4. (*Elaboración propia*)

Variable	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Nombre del cargo	Campo de texto	No	Permite solo los caracteres que sean letras, números y espacios.

Tabla 44: Caso de prueba del RF19_Crear área. (*Elaboración propia*)

Caso de prueba 4: SC RF19_Crear cargo				
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con el rol necesario para acceder a la funcionalidad.				
Escenario	**Descripción**	**1**	**Respuesta del sistema**	**Flujo Central**

EC 1.1 Crear un cargo de manera correcta	Interfaz con un formulario para ingresar el dato necesario para crear un cargo, si es correcto, se crea el área	V Facultad 1	Crea el cargo y muestra un mensaje de notificación	1. Seleccionar, en el menú superior, la opción “cargo” y presionar el botón “Crear área” 2. Llenar los campos correspondientes en el formulario y seleccionar la opción “Enviar”
EC 1.2 Crear un cargo con campos vacíos	Interfaz con un formulario para ingresar el dato necesario para crear un cargo, si existe algún campo vacío, se muestra un mensaje pidiendo llenar el campo	I	Comprueba si los campos están vacíos, si lo están, muestra un mensaje que solicita llenarlos	
EC 1.3 Crear un cargo que ya existe	Interfaz con un formulario para ingresar el dato necesario para crear un cargo, si se inserta un cargo que ya existe en la base de datos, se muestra un mensaje de error	V Facultad 1	Comprueba que el cargo que se inserte no exista en la base de datos, si existe, muestra un mensaje de error	

ANEXO 4: ENCUESTA PARA DETERMINAR EL COEFICIENTE DE COMPETENCIAS DE LOS EXPERTOS.

Compañero (a): _____

Usted ha sido seleccionado como posible experto para ser consultado respecto a temas relacionados a la firma digital de documentos y portafirmas digitales, con vista a la investigación que se está llevando a cabo. Agradecemos sinceramente su valiosa cooperación.

Gracias.

1. Marque con una cruz (X) en la tabla siguiente el valor que se corresponde con el grado de conocimiento que usted posee sobre criptografía y específicamente sobre "firma digital". (Escala ascendente).

0	1	2	3	4	5	6	7	8	9	10

2. Se realiza una autoevaluación del grado de influencia que cada una de las fuentes que le presentamos a continuación ha tenido en su conocimiento y criterio sobre criptografía y específicamente sobre "firma digital". Marque con una cruz (X) según corresponda en A (alto), M (medio) o B (bajo).

No	Fuente de argumento	Grado de influencia de cada una de las fuentes		
		A (alto)	M (medio)	B (bajo)
1	Estudios teóricos realizados por usted.			
2	Experiencia adquirida durante su vida profesional.			
3	Conocimiento de investigaciones y/o publicaciones nacionales e internacionales.			
4	Conocimiento propio sobre el estado del tema de investigación.			

5	Actualización en cursos de postgrado, diplomados, maestrías, doctorado, etc.			
6	Intuición.			

ANEXO 5. PROCEDIMIENTO EMPLEADO PARA DETERMINAR EL COEFICIENTE DE COMPETENCIA DE LOS CANDIDATOS A EXPERTOS

Cálculo del coeficiente de competencia de los expertos que evaluaron el paquete de servicios desarrollado.

El cálculo de dicho coeficiente se realiza de la forma siguiente:

$Kcomp = ½ (Kc + Ka)$

Donde:

Kcomp: Coeficiente de competencia.

Kc: Coeficiente de conocimiento o información que tiene el experto acerca del problema, calculado sobre la valoración del propio experto en una escala de 0 a 10 y multiplicado por 0,1.

Ka: Coeficiente de argumentación o fundamentación de los criterios del experto, obtenido como resultado de la suma de los puntos de acuerdo a la siguiente tabla patrón:

Tabla 45: Fuentes de argumentación del conocimiento de los expertos. (*Elaboración propia*)

No	Fuentes de argumentación	Alto (A)	Medio (M)	Bajo (B)
1	Análisis teóricos realizados.	0,30	0,20	0,10
2	Experiencia adquirida durante su vida profesional.	0,50	0,37	0,30
3	Conocimiento de investigaciones y/o publicaciones nacionales e internacionales.	0,05	0,04	0,03
4	Conocimiento propio sobre el estado del tema de investigación.	0,05	0,04	0,03
5	Actualización en cursos de postgrado, diplomados, maestrías, doctorado, etc.	0,05	0,04	0,03
6	Intuición.	0,05	0,03	0,02

	Total	1,00	0,70	0,50

Se plantea entonces que:

- ✓ La Competencia del experto es de Alta (A): Si *Kcomp* > 0,7
- ✓ La Competencia del experto es Media (M): Si 0,5 < *Kcomp* = < 0,7
- ✓ La Competencia del experto es Baja (B): Si *Kcomp* = < 0,5

ANEXO 6. RESULTADO DEL CUESTIONARIO APLICADO A LOS CANDIDATOS A EXPERTOS PARA DETERMINAR SU NIVEL DE COMPETENCIA.

Tabla 46: Resultado de la encuesta aplicada a los candidatos a expertos para determinar nivel de competencia.

(Elaboración propia)

Expertos	*Kc*	1.1	1.2	1.3	1.4	1.5	1.6	*Ka*	*Kcomp*	valor
1	0,8	0,10	0,50	0,05	0,03	0,04	0,02	0,74	0,77	alto
2	0,7	0,20	0,37	0,05	0,04	0,04	0,05	0,75	0,73	alto
3	0,6	0,20	0,37	0,03	0,04	0,03	0,04	0,71	0,66	medio
4	0.8	0.30	0.50	0.04	0.04	0.03	0.05	0.96	0.88	alto
5	0.4	0.10	0.37	0.03	0.03	0.03	0.02	0.58	0.49	bajo
6	0.5	0.10	0.30	0.03	0.03	0.03	0.02	0.51	0.50	bajo

ANEXO 7. CUESTIONARIO A EXPERTOS

PAQUETE DE SERVICIOS PARA EL PORTAFIRMAS DIGITAL DE LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS				
Datos del encuestado				
Entidad:				
Área:				
Nombre y Apellidos:				
Cargo o Rol:				
Nivel Escolar:	Técnico Medio 	Universitario 		
Categoría Docente:	Instructor 	Asistente 	Auxiliar 	Titular 
Categoría Científica:	Especialista 	Máster 	Doctor 	
Años de experiencia:				

#	Afirmaciones	Respuesta
1	La utilización de la propuesta de solución facilitará el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.	MA  ED  DA  CD  Sí-  No
2	El sistema será de gran utilidad para llevar a cabo la firma digital de documentos en la universidad.	MA  ED  DA  CD  Sí-  No
3	El sistema brindará la posibilidad de integrarse con cualquier tipo de presentación dando mayor seguimiento a los procesos de autenticación de documentos digitales en la universidad.	MA  ED  DA  CD  Sí-  No
4	El usuario puede cumplir su objetivo de manera rápida.	MA  ED  DA  CD 

		Sí- No	
5	El sistema será capaz de integrarse con otras soluciones existentes en la universidad.	MA DA Sí- No	ED CD
6		MA DA Sí- No	ED CD

ANEXO 8. RESPUESTAS DADAS POR LOS EXPERTOS PARA CADA INDICADOR

5: Muy de acuerdo (MA)

4: De acuerdo (DA)

3: Ni de acuerdo ni en desacuerdo (Sí-No)

2: En desacuerdo (ED)

1: Completamente en desacuerdo (CD)

Tabla 47: Respuestas dadas por los expertos para cada indicador. (*Elaboración propia*)

Experto	Indicador					
	1	2	3	4	5	6
1	4	5	4	5	4	5
2	5	5	5	4	5	5
3	5	5	4	5	5	5
4	5	4	4	4	4	5