Universidad de las Ciencias Informáticas

Facultad 1

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

Título: Portafirmas Digital de la Universidad de las Ciencias Informáticas

Autor:

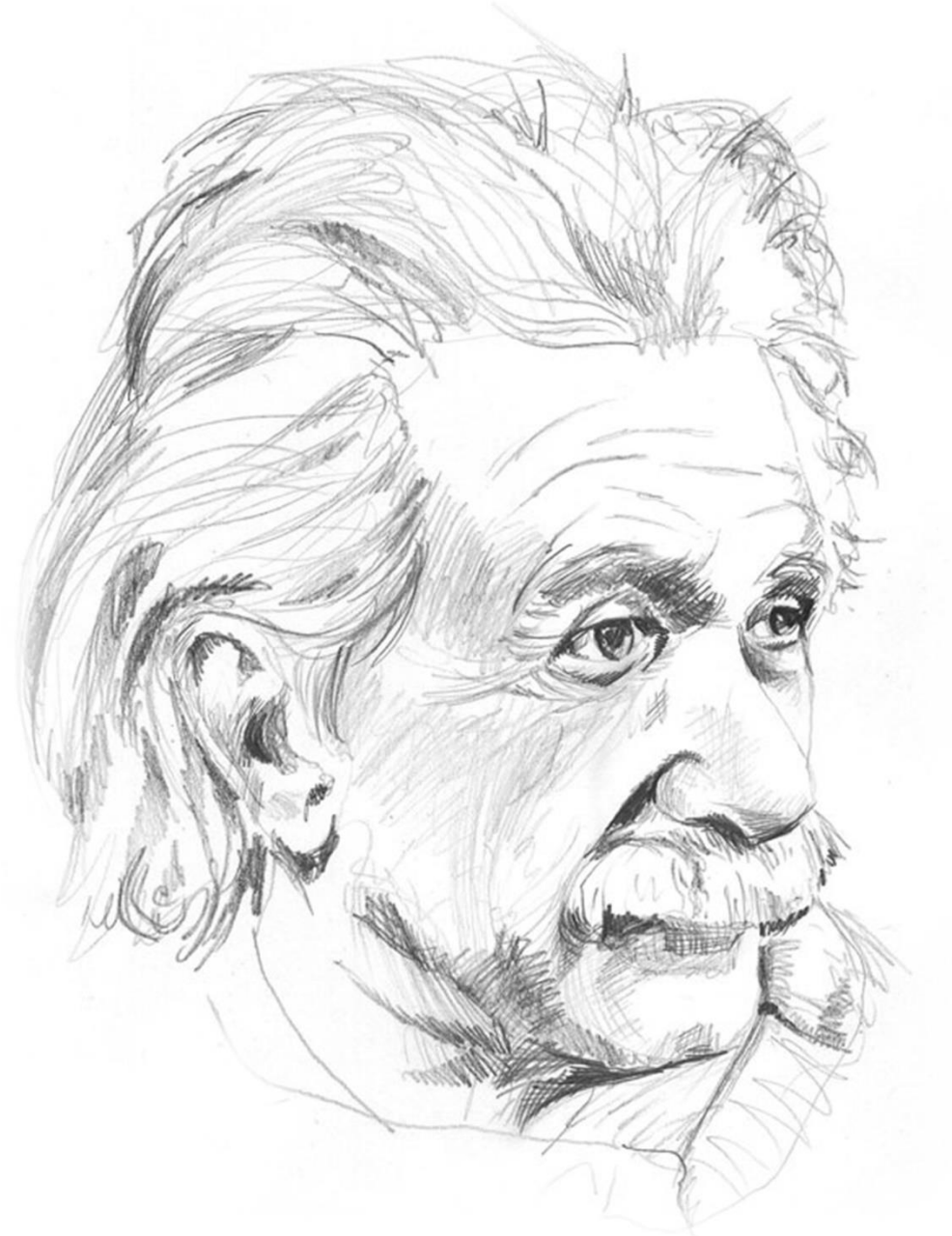
Anaili Pérez Piedra

Tutores:

Dr.C Ramón Santana Fernández

MSc. Adrian Alberto Machado Cento

La Habana, Junio 2018



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad"

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo: Anaili Pérez Piedra, con carné de identidad 95090627158, soy la autora principal del trabajo final de tesis de pregrado que se titula: “Portafirmas Digital de la Universidad de las Ciencias Informáticas”. El cual ha sido desarrollado como parte del trabajo en el centro CISED de la Facultad 1. Autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, se firma la presente declaración jurada de autoría en La Habana, a los _____ días del mes de _____ del año 2018.

_____

Autor

Anaili Pérez Piedra

_____ _____

Tutor **Tutor**

Dr.C Ramón Santana Fernández MSc. Adrian Alberto Machado Cento

DEDICATORIA

A mis padres por dedicarse a mí toda su vida, ahora me corresponde darle valor al fruto de su sacrificio.

AGRADECIMIENTOS

A mi mamá y mi papá por ser el mayor regalo que la vida me pudo dar, por confiar siempre en mí y apoyarme durante todos estos años y por ser mi inspiración en cada meta que me propongo.

A absolutamente cada miembro de mi familia, a los que tengo cerca, los que están un poco lejos y a los que ya no puedo abrazar, por vivir orgullosos de mí, por ser mi luz y brindarme el calor de un hogar.

A mi novio, que a pesar del poco tiempo que llevamos juntos, ha sabido darme la tranquilidad y ayuda que tanto necesité en este período, por su paciencia y amor en todo momento.

A los amigos de la infancia y adolescencia, esos que, aunque te demores en verlos las cosas siguen siendo iguales entre nosotros.

A los nuevos amigos que descubres en este camino, a los que te acompañan todos los días en el aula, a mi cuasi compañero de tesis y los que en momentos difíciles te extienden la mano y te brindan su corazón para siempre.

A mis tutores, en especial a Ramón por su paciencia y apoyo, a Denier y Yiriam por su ayuda.

A los profesores de todas las enseñanzas, en especial a los de esta universidad, por formarme como profesional, por sus muestras de afecto, por sus consejos y sugerencias para obtener este logro.

A la Revolución por la oportunidad de estudiar esta carrera.

A los que de alguna manera me han brindado su ayuda en lo que me ha hecho falta.

A todos muchísimas gracias.

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 análisis 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 el análisis 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 sistema que permite firmar digitalmente documentos en formato *PDF* haciendo uso de certificados digitales y efectuar el flujo de firmado desde una aplicación web que garantiza la autenticación de la información que maneja. Se muestran los resultados de un conjunto de pruebas realizadas al sistema, con el fin de entregar al cliente una solución libre y confiable, que pueda ser utilizada en la universidad.

Palabras clave: autenticación, certificados digitales, documentos, firma digital, flujo de firmado.

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN **1**

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA **6**

1.1 Conceptos asociados al dominio de la investigación 6

1.1.1 Criptografía 6

1.1.2 Llave 6

1.1.3 Criptosistema 7

1.1.4 Sistemas de cifrado híbrido 8

1.1.5 Tecnología *PKI* 9

1.1.6 Certificado Digital 9

1.1.7 Firma Digital 10

1.2 Portafirmas Digitales 12

1.2.1 Portafirmas a nivel internacional 12

1.2.2 Portafirmas a nivel nacional 14

1.3 Entorno de desarrollo de la propuesta de solución 16

1.4 Conclusiones del capítulo 22

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN **23**

2.1 Análisis 23

2.1.1 Modelo conceptual 23

2.1.2 Descripción de los elementos del modelo conceptual 24

2.1.3 Características de la propuesta de solución 25

2.2 Diseño 28

2.2.1 Requisitos funcionales 28

2.2.2 Requisitos no funcionales 30
2.2.3 Historias de usuario 31
2.2.4 Descripción de la Arquitectura de Software 33
2.2.5 Patrones de diseño 34
2.2.6 Diagrama de clases 38
2.2.7 Diagrama de despliegue 39
2.3 Conclusiones del capítulo 40
CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN 41
3.1 Diagrama de componentes 41
3.2 Estándares de codificación 42
3.3 Verificación y validación de la propuesta de solución 45
3.3.1 Pruebas funcionales 45
3.3.2 Pruebas de seguridad 49
3.3.3 Pruebas de usabilidad 50
3.4 Interfaces principales del sistema Portafirmas Digital de la UCI 52
3.5 Validación de la hipótesis (Criterio de expertos) 54
3.6 Conclusiones del capítulo 57
CONCLUSIONES GENERALES 58
RECOMENDACIONES 59
REFERENCIAS BIBLIOGRÁFICAS 60
BIBLIOGRAFÍA 66
ANEXOS 67
Anexo 1. Entrevista al cliente para conocer la necesidad del desarrollo de la propuesta de solución y definir los requisitos funcionales y no funcionales 67

Anexo 2. Historias de usuario 68

Anexo 3. Pruebas funcionales 81

Anexo 4. Encuesta para determinar el coeficiente de competencias de los expertos 85

Anexo 5. Procedimiento empleado para determinar el coeficiente de competencia de los candidatos a expertos 87

Anexo 6. Resultado del cuestionario aplicado a los candidatos a expertos para determinar su nivel de competencia 89

Anexo 7. Cuestionario a expertos 90

Anexo 8. Respuestas dadas por los expertos para cada indicador 91

ÍNDICE DE FIGURAS

Figura 1: Modelo conceptual 24
Figura 2: Arquitectura de software del Portafirmas Digital de la UCI 34
Figura 3: Código para rechazar las peticiones 35
Figura 4: Código del componente addUser 36
Figura 5: Código de definición del componente addUser 37
Figura 6: Código buscar del UserService 38
Figura 7: Diagrama de clases 39
Figura 8: Diagrama de despliegue del Portafirmas Digital 39
Figura 9: Diagrama de componentes del sistema Portafirmas Digital 42
Figura 10: Resultados de las pruebas funcionales 49
Figura 11: Resultados de las pruebas de seguridad 50
Figura 12: Interfaz de la página para crear una petición de firma 53
Figura 13: Interfaz de la página para firmar un documento 53
Figura 14: Resultados de la aplicación de la escala de *Likert* 56

ÍNDICE DE TABLAS

Tabla 1: Operacionalización de las variables....3
Tabla 2: Resumen del análisis de los sistemas homólogos....15
Tabla 3: Descripción de las peticiones....25
Tabla 4: Relación de requisitos funcionales del sistema....29
Tabla 5: HU_9 Crear petición de firma....32
Tabla 6: HU_13 Firmar documento digital....32
Tabla 7: Descripción de las variables del caso de prueba 1....46
Tabla 8: Caso de prueba del RF9_Crear petición de firma....47
Tabla 9: Resultado de las pruebas de usabilidad....51
Tabla 10: Expertos seleccionados en la validación de la investigación....55
Tabla 11: HU_1 Autenticar usuario....68
Tabla 12: HU_2 Crear usuario....68
Tabla 13: HU_3 Editar usuario....69
Tabla 14: HU_4 Eliminar usuario....69
Tabla 15: HU_5 Buscar usuario....70
Tabla 16: HU_6 Listar usuarios....70
Tabla 17: HU_7 Rechazar documento....71
Tabla 18: HU_8 Mostrar documento....71
Tabla 19: HU_10 Mostrar estado de la petición de firma....72
Tabla 20: HU_11 Buscar petición de firma....72
Tabla 21: HU_12 Listar peticiones de firma....73
Tabla 22: HU_14 Listar peticiones terminadas....73
Tabla 23: HU_15 Descargar documento....74
Tabla 24: HU_16 Listar peticiones creadas....74
Tabla 25: HU_17 Listar peticiones en espera de firma....75

Tabla 26: HU_18 Crear área 75
Tabla 27: HU_19 Editar área 76
Tabla 28: HU_20 Eliminar área 76
Tabla 29: HU_21 Buscar área 77
Tabla 30: HU_22 Listar áreas 77
Tabla 31: HU_23 Crear cargo 78
Tabla 32: HU_24 Editar cargo 78
Tabla 33: HU_25 Eliminar cargo 79
Tabla 34: HU_26 Buscar cargo 79
Tabla 35: HU_27 Listar cargos 80
Tabla 36: Descripción de las variables del caso de prueba 2 81
Tabla 37: Caso de prueba del RF13_Firmar digitalmente un documento 81
Tabla 38: Descripción de las variables del caso de prueba 3 83
Tabla 39: Caso de prueba del RF19_Crear área 83
Tabla 40: Fuentes de argumentación del conocimiento de los expertos 87
Tabla 41: Resultado de la encuesta aplicada a los candidatos a expertos para determinar nivel de competencia 89
Tabla 42: Respuestas dadas por los expertos para cada indicador 91

INTRODUCCIÓN

La información es el activo más importante en las organizaciones y garantizar su disponibilidad, confidencialidad e integridad constituye un aspecto significativo para evitar la pérdida, modificación o robo de la misma. Una de las técnicas más usadas en la actualidad para proteger la información es la criptografía. La palabra criptografía proviene en un sentido etimológico del griego Kriptos (ocultar), Graphos (escritura), lo que significaría ocultar la escritura, o en un sentido más amplio sería aplicar alguna técnica para hacer ininteligible un mensaje (Paredes 2006). En la práctica, la criptografía consiste en aplicar mecanismos matemáticos que transformen los mensajes, de manera que solo puedan ser entendidos por aquellas personas que conozcan los procedimientos para volverlos a su forma original.

Históricamente la criptografía ha transitado por tres períodos fundamentales: pre-científica, científica y de clave pública. La criptografía pre-científica se caracterizó por ser más un arte que una ciencia, pues los métodos de cifrado que se utilizaban poseían una consistente formulación algebraica, pero carecían de una sólida base matemática. En 1948, con la publicación de la Teoría de la Información por *Shannon*, comienza la etapa de la criptografía científica, más elaborada y con una amplia base matemática. Finalmente, en el año 1976 se publica el estudio realizado por *Whitfield Diffie* y *Martin Hellman* sobre la aplicación de funciones matemáticas de un solo sentido a un modelo de cifra, denominado cifrado con clave pública (Díaz 1995).

Existen dos tipos de criptografía, simétrica y asimétrica, y ambas proporcionan autenticidad, confidencialidad e integridad de la información. La criptografía asimétrica es la base de la Infraestructura de Clave Pública (*PKI*)[1], 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).

La firma digital es una de las aplicaciones de la criptografía moderna de mayor éxito en la actualidad y es usada para la tramitación de documentos a través de sistemas de información y garantiza autenticidad, integridad y no repudio de los mismos. La verificación de la firma digital permite determinar cuándo un mensaje o documento ha sido alterado y tiene la propiedad de que solo puede ser producida de manera correcta por una entidad, y ser verificada por cualquiera que reciba el mensaje o documento firmado digitalmente (Wolfgang y Wolfgang 2004).

[1] Del inglés, *Public Key Infrastructure.*

En la administración electrónica de documentos, es cada vez más frecuente el uso de la firma digital para la tramitación de documentos mediante sistemas de información, así como en gestiones de índole interna. En el marco del proceso de informatización de la Universidad de las Ciencias Informáticas (UCI) se encuentra la implantación de una *PKI* que permita la autenticación de un usuario frente a otro y usar la llave pública del otro usuario para cifrar mensajes, firmar digitalmente información y garantizar el no repudio de un envío. Debido a la diversidad de sistemas de información que gestionan y tramitan los diferentes procedimientos en la universidad, el coste de implantar la firma electrónica en cada sistema de información, tanto en tiempo como en complejidad es extremadamente elevado, ya que cada sistema de información está basado en tecnologías y arquitecturas diferentes.

A pesar de que el desarrollo de la *PKI* en la UCI ha avanzado considerablemente, no se han explotado al máximo todas las potencialidades que esta brinda. No existe una solución informática que gestione la firma electrónica de documentos electrónicos creados por distintos medios y sistemas de información de forma centralizada, provocando que la acción de firmar los documentos se realice de manera aislada al proceso que la origina.

El Sistema de Gestión Documental (eXcriba) utilizado en la universidad no cuenta con un módulo para la firma digital de documentos en su flujo de aprobación formal de los mismos. Debido a esto, los usuarios deben descargar el documento para su ordenador, ponerle su firma y enviarlo a la siguiente persona que debe firmarlo, en caso de que sea la última persona de la cadena de firmado, debe subirlo nuevamente al sistema. Todo lo anterior trae consigo un aumento del tiempo y esfuerzo empleado en la ejecución de los procesos. No es posible conocer la fecha de caducidad de los documentos a firmar, atentando la eficacia y eficiencia del trabajo. Tampoco existe la posibilidad de conocer cuántos documentos han sido firmados por una persona, dificultando el proceso de supervisión y control de los documentos firmados.

Ante esta situación se plantea 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 digital de documentos digitales. Derivándose como **campo de acción** el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.

Teniendo en cuenta lo planteado anteriormente, el **objetivo general** de esta investigación es: desarrollar el sistema Portafirmas Digital para la autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.

Para cumplir con el objetivo general expuesto, se han trazado varios **objetivos específicos**:

1. Analizar los referentes teórico-metodológicos asociados a la firma y autenticación de documentos digitales.
2. Diseñar el sistema Portafirmas Digital teniendo en cuenta las funcionalidades con las que debe contar.
3. Implementar las funcionalidades correspondientes al sistema Portafirmas Digital.
4. Validar el funcionamiento del sistema Portafirmas Digital.

Tras definir los objetivos y analizar el problema, se obtiene la siguiente **hipótesis**: el desarrollo del sistema Portafirmas Digital mejorará el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas.

Identificándose como **variable independiente**: sistema 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**
Sistema Portafirmas Digital	Facultad 1	Adecuación funcional	Completitud funcional	(%)
		Usabilidad	Reconocibilidad	(%)
			Estética de interfaz de usuario	(%)
Proceso de autenticación	Usuarios que utilizan el	Tiempo de firmado de	Alta	1-5 (seg)

de documentos digitales en la Universidad de las Ciencias Informáticas	sistema	documentos	Media	6-10 (seg)
			Baja	11-15 (seg)

Con el objetivo de cumplir con los objetivos específicos planteados se establecen las siguientes **tareas de investigación**:

1. Realización de un estudio sobre las tendencias de los portafirmas digitales existentes en la actualidad.
2. Selección de las tecnologías, herramientas y metodología de desarrollo de software que se necesitan para la propuesta de solución.
3. Modelación de los procesos asociados a la autenticación de documentos digitales en la universidad.
4. Definición de los requisitos funcionales y no funcionales del sistema.
5. Implementación del Portafirmas Digital de la UCI.
6. Realización de las pruebas de software para validar la propuesta de solución.
7. Validación de la hipótesis de investigación.

Los **métodos de investigación** utilizados para dar solución a la presente investigación son:

Teóricos:

- **Histórico-Lógico:** posibilitó realizar un estudio de los principales conceptos relacionados con la criptografía y firma digital desde su surgimiento hasta la actualidad, así como de los diferentes portafirmas digitales existentes hasta la fecha.
- **Analítico-Sintético:** permitió llevar a cabo un análisis bibliográfico para establecer las bases teóricas en relación al desarrollo de los portafirmas digitales.

- **Modelación:** facilitó la representación mediante diagramas de las características, procesos y componentes de la solución propuesta, así como la relación existente entre ellos.

Empíricos:

- **Entrevista:** se realizó a través de encuentros con el cliente para conocer la necesidad del desarrollo de la propuesta de solución y definir sus requisitos funcionales.

La presente investigación está estructurada de la siguiente forma:

Capítulo 1: Fundamentación Teórica.

En este capítulo se realiza un estudio de los referentes teórico-metodológicos asociados a la investigación. Se analizan los diferentes sistemas que manejan la firma digital. Se seleccionan la metodología, herramientas y tecnologías a utilizar en la propuesta de solución.

Capítulo 2: Análisis y diseño del sistema.

En este capítulo se documenta todo el proceso de análisis y diseño del sistema de manera más detallada. Se describen los requisitos de acuerdo a lo establecido en la metodología utilizada, diagrama de clase, diagrama de despliegue, así como los patrones de diseño utilizados.

Capítulo 3: Implementación y validación del sistema.

En este capítulo se detalla la propuesta de solución al problema planteado. Se describe la organización del sistema en un diagrama de componentes y se especifican los estándares de codificación a utilizar. Se realizan las pruebas definidas para el sistema garantizando su correcto funcionamiento y culmina con la validación de la hipótesis de investigación.

Posibles resultados:

Con la realización de la presente investigación, se pretende obtener un sistema que permita realizar el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas de una forma más rápida, sencilla e intuitiva. Se espera obtener también la descripción de requisitos del proceso de autenticación de documentos digitales y toda la documentación referente a la realización de las pruebas de software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Con el objetivo de lograr una mayor comprensión del alcance de la investigación y esclarecer su objeto de estudio, se exponen en el presente capítulo, los conceptos asociados al dominio de la investigación y se realiza un análisis del estado del arte de sistemas homólogos. Se presenta la metodología y el entorno de desarrollo a utilizar para dar solución al problema planteado.

1.1 Conceptos asociados al dominio de la investigación

Con el propósito de una mejor comprensión de los temas que serán tratados en este capítulo, directamente vinculados con el objeto de estudio de la investigación, se describen a continuación un grupo de conceptos relacionados con el dominio del problema.

1.1.1 Criptografía

Según (RAE 2017) la definición de la palabra Criptografía es: “Arte de escribir con clave secreta o de un modo enigmático”. Esta definición no es del todo adecuada en la actualidad, ya que la criptografía ha dejado de ser un arte para convertirse en una ciencia, pues tiene bases matemáticas como son: teoría de números, teoría de la complejidad algorítmica, teoría de la información y estadística (Xifré 2009).

Desde un punto de vista técnico, se puede ver a la criptografía como el estudio de técnicas matemáticas relacionadas con aspectos de la seguridad de la información, como la confidencialidad, la integridad de datos, la autenticación de entidades y la autenticación de origen de datos. La criptografía no es el único medio para proporcionar seguridad de la información, sino más bien un conjunto de técnicas con el objetivo principal de cifrar, y por tanto, proteger un mensaje o archivo por medio de un algoritmo, usando una o más llaves (Menezes, Oorschot y Vanstone 1996).

1.1.2 Llave

En criptografía, una llave es un valor que trabaja con un algoritmo criptográfico para producir un texto cifrado; las llaves son básicamente números muy grandes, y su tamaño se mide en bits. Cuanto más grande sea la llave, más seguro será el texto cifrado. Las llaves deben ser almacenadas en forma cifrada para evitar que algún intruso pueda obtenerla (Menezes, Oorschot y Vanstone 1996).

1.1.3 Criptosistema

Un criptosistema se define como una quíntupla (**M, C, K, E, D**), según (Menezes, Oorschot y Vanstone 1996):

- **M**: representa el conjunto de todos los mensajes sin cifrar (lo que se denomina texto plano) que pueden ser enviados.
- **C**: representa el conjunto de todos los posibles mensajes cifrados, o criptogramas.
- **K**: representa el conjunto de claves que se pueden emplear en el criptosistema.
- **E**: es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de M para obtener un elemento de **C**. Existe una transformación diferente E_k para cada valor posible de la clave **k**.
- **D**: es el conjunto de transformaciones de descifrado, análogo a **E**.

Existen dos tipos fundamentales de criptosistemas (López 2010):

- **Criptosistemas simétricos o de llave privada:** son aquellos que emplean la misma clave k tanto para cifrar como para descifrar. Presentan la inconveniente de que, para ser empleados en comunicaciones, la clave k debe estar tanto en el emisor como en el receptor, lo que trae consigo una transmisión de la clave de forma insegura.
    - ***AES*:** es un algoritmo de cifrado simétrico llamado *Advanced Encryption Standard* que se da a conocer en el 2000 como resultado de un concurso lanzado por el Instituto Nacional de Estándares y Tecnología (*NIST*), en busca de un sustituto para *DES* y *Triple DES*. Este algoritmo utiliza cifrado por bloques y maneja llaves de longitudes de 128, 192 y 256 bit (Mogollon 2007) y está considerado uno de los algoritmos más seguros de la actualidad.
- **Criptosistemas asimétricos o de llave pública:** son aquellos que emplean una doble clave (kp, kP), kp se conoce como clave privada y kP se conoce como clave pública. Una de ellas sirve para la transformación E de cifrado y la otra para la transformación D de descifrado. En muchos casos son intercambiables, esto es, si se emplea una para cifrar la otra sirve para descifrar y viceversa. Ofrecen un espectro superior de posibilidades, pudiendo emplearse para establecer

comunicaciones seguras por canales inseguros, puesto que únicamente viaja por el canal la clave pública, que solo sirve para cifrar o para llevar a cabo autentificaciones.

- ***RSA*:** es un algoritmo de cifrado asimétrico propuesto por *Ron **R**ivest*, *Adi **S**hamir*, y *Len **A**dleman* en 1978, es el más conocido y versátil de los algoritmos de llave pública usados actualmente. Es apropiado para las operaciones de cifrado y firma digital (Wolfgang y Wolfgang 2004). Su seguridad está basada en la dificultad de factorizar números enteros muy grandes y por el grado de avance en esta área de las matemáticas, se sugiere el empleo de llaves de al menos 1024 *bit*. Su limitación está dada en que el contenido que se desea cifrar no debe sobrepasar los 2048 *bit* equivalentes a 256 *bytes*.

1.1.4 Sistemas de cifrado híbrido

Los sistemas híbridos surgen a partir de la combinación de mecanismos de cifrado simétricos y asimétricos. Estos sistemas se basan en la utilización de una clave de sesión, que es una clave binaria, usada solo en el período de duración de la comunicación. Utilizan la velocidad de cifrado que ofrecen los mecanismos simétricos para grandes volúmenes de información y la seguridad que propician los mecanismos de cifrado asimétricos para llevar a cabo una comunicación segura por canales inseguros (Silva 2004).

Ejemplo del procedimiento de cifrado a través de criptosistemas híbridos (Silva 2004):

1. Alicia y Juan tienen sus pares de clave, una clave privada que solo ha de conocer el propietario de la misma y una clave pública que está disponible para todos los usuarios.
2. Alicia escribe un mensaje a Juan. Lo cifra con el sistema de criptografía de clave simétrica. La clave que utiliza el algoritmo simétrico es una clave de sesión que se genera aleatoriamente, que puede ser del conocimiento de Alicia o invisible a ella.
3. Para enviar o almacenar la clave de sesión de forma segura, esta es cifrada con la clave pública de Juan, utilizando mecanismos criptográficos de clave asimétrica.
4. Juan recibe el mensaje cifrado y la clave de sesión, la cual se encuentra cifrada con su clave pública. Para obtener la información contenida en el mensaje, Juan utiliza su clave privada para descifrar la clave de sesión, y una vez que haya obtenido la clave de sesión, ya puede descifrar el mensaje.

1.1.5 Tecnología *PKI*

Una *PKI* es un sistema de entrega de certificados y llaves criptográficas, lo cual posibilita la seguridad en transacciones económicas financieras y el intercambio de información sensible entre personas relativamente desconocidas (Báster 2011). El principal objetivo de un sistema *PKI* es la gestión y distribución de claves públicas, en sentido general se basa en un modelo de confianza, en el cual los usuarios se aseguran de que las claves públicas gestionadas por dicha *PKI* son auténticas (Peña et al. 2009).

Algunas de las ventajas que ofrece el uso de la *PKI* son (Rodríguez, Morales y Fabre 2016):

- Promueve flujos de trabajo eficientes y procesos de negocios más seguros y automatizados.
- Reduce los requerimientos de entrenamiento al usuario final relacionados con el uso de los servicios de seguridad.
- Reduce los costos administrativos.
- Optimiza la productividad de la fuerza de trabajo, concentrándola en su modo de producción básico y sin preocuparse de complejos procesos requeridos para garantizar la seguridad.

Fundamentalmente, en una Arquitectura de *PKI* las principales entidades son (Báster 2011):

- Una Autoridad de Certificación (*CA*) que controla el ciclo de vida de los certificados digitales.
- Autoridad de Registro (*RA*) que identifica a los usuarios y a los cuales se les entrega su respetivo certificado digital.
- Suscriptores o usuarios de los certificados.
- Repositorios que almacenen los certificados y la lista de certificado revocados (*CRL*).

1.1.6 Certificado Digital

Un certificado digital es una credencial electrónica que relaciona la información del titular de dicho certificado con su clave pública (Peña et al. 2009). Es un mecanismo que se basa en la criptografía de claves públicas o asimétricas para permitir comunicaciones seguras entre el origen y/o el destino utilizando medios de comunicaciones inseguros como Internet y permiten que la comunicación esté

certificada por un tercero de confianza que lo constituye la *CA* que garantiza la confidencialidad y el no repudio de la comunicación (Carvajal 2007).

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) para *PKI*, siendo la pieza central de esta y donde se especifican formatos de atributos estándares para certificados de claves públicas y un algoritmo de validación de la ruta de certificación («Cisco» 2016).

Entre los formatos de archivo de certificados X.509v3 que existen, en la aplicación se utilizará el que está establecido por política de la universidad que es el ***PKCS* #12**, correspondiente a la extensión ".p12". Este formato de archivo es uno de la familia de estándares llamados *Public Key Cryptography Standards* (*PKCS*) publicados por *RSA Laboratories* y 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 et al. 2014).

1.1.7 Firma Digital

Según (Zayas y Milagro 2013) la firma digital es un conjunto de datos electrónicos que identifican a una persona en concreto. Suelen unirse al documento que se envía por medio telemático, como si de la firma tradicional y manuscrita se tratara, de esta forma el receptor del mensaje está seguro de quién ha sido el emisor, así como la seguridad de que el mensaje no ha sido alterado o modificado.

La firma digital provee a los documentos las siguientes características:

- Autenticidad: el receptor del mensaje es capaz de verificar el origen del mismo.
- No repudio: el emisor del mensaje no puede negar que lo ha firmado y enviado.
- Integridad: el receptor del mensaje es capaz de verificar que el mensaje no haya sido alterado.

Aunque según (Menezes, Oorschot y Vanstone 1996), la autenticación de datos o las técnicas de autenticación de mensajes proporcionan, a una parte que recibe una garantía de mensaje, la identidad de la parte que originó el mensaje. También garantiza implícitamente integridad de datos ya que, si el mensaje se modificó durante la transmisión, el que envió el mensaje ya no sería el creador.

La firma digital involucra dos acciones: la acción de firmar y la acción de verificación de la firma. A continuación se explican estos procedimientos (Silva 2004):

- **Generación de la firma digital:** se le aplica una función resumen al documento que se quiere firmar y se obtiene una secuencia de bits que identifica de manera unívoca al documento. El código obtenido se cifra con la clave privada del emisor del documento. Finalmente, se envía el documento y el resumen del mensaje cifrado que representa la firma digital. De esta forma, el signatario protege la integridad del documento, pues le incorpora una marca que solo él es capaz de realizar.
- **Verificación de la firma:** el receptor separa el documento de la firma digital, calcula el *HASH* del documento y descifra utilizando la clave pública del emisor la firma digital enviada. Si el resultado que se obtiene por los dos caminos es el mismo, significa que el documento conserva su autenticidad e integridad.

Entre las ventajas más representativas de la firma digital se encuentran (Báster 2011):

- Permite la integridad de un documento, ya que un archivo firmado digitalmente no puede ser modificado sin dejar un rastro o huella.
- Permite dotar a la firma de una duración de muchos años, así como de una posibilidad de validación en cualquier instante de tiempo.
- Los tiempos de entrega y envío de documentos firmados se reducen considerablemente dentro de una organización.
- Permite garantizar la autoría del documento, evitando así el repudio del documento firmado.
- Permite firmar lotes considerables de documento digitales, los cuales, si fueran llevados al mundo físico, demandarían mucho más tiempo y cansancio.
- La indexación de los documentos firmados facilita la manipulación a través de un software de gestión de documentos sin necesidad de complicadas integraciones.
- Brinda una garantía legal y respaldo jurídico, ya que la firma digital tiene el mismo valor que una firma manuscrita.

1.2 Portafirmas Digitales

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. Se trata de herramientas de usuario final que utilizan y proveen servicios de autenticación y firma digital que tienen, siempre que las normativas jurídicas lo estipulen, el mismo valor que la firma manuscrita («Evaluación y revisión» 2009).

A través de la firma digital, los portafirmas digitales, incluyen en los documentos un código seguro de verificación generado electrónicamente, que permite constatar su integridad y autenticidad. Los documentos emitidos por instituciones, firmados con herramientas de este tipo pudieran tener la validez y eficacia de documentos originales. En la actualidad existen aplicaciones para la firma digital de documentos. Con el objetivo de identificar ventajas y desventajas de estas, se realiza el siguiente análisis haciendo énfasis en las características y funcionalidades de este tipo de herramientas.

1.2.1 Portafirmas a nivel internacional

En el análisis realizado en el ámbito internacional, se identificó a España como uno de los países con más experiencia en este campo y en el que más se ha difundido el uso de los portafirmas digitales en todas las áreas que utilizan firma digital de documentos. Por esta razón se estudiaron los siguientes portafirmas:

ViaFirmaInbox

ViaFirmaInbox es una aplicación web basada en los patrones de la Arquitectura Orientada a Servicios (*SOA*), 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 («Agenda de firmas electrónicas - ViafirmaInbox | Viafirma» 2017):

- Posibilita 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.
- Brinda 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 («@firma | Portal de Administración Electrónica» 2016).

Port@firmas, Universidad de Sevilla

Porta@firmas es una herramienta web 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 *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 («Port@firmas | Portal de Administración Electrónica» 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 histórico 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 («Portafirmas | Guadaltel» 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.

1.2.2 Portafirmas a nivel nacional

El uso de los portafirmas digitales es un área poco explorada en el ámbito nacional, por lo que el único sistema homólogo estudiado es el siguiente:

UCI PDFSigner

Es una aplicación de escritorio desarrollada como resultado de un trabajo de diploma del Centro de Identificación y Seguridad Digital (CISED) que permite la firma de documentos *PDF* (Bondartchuk 2009). Tiene implementadas las siguientes funcionalidades que pueden servir de apoyo a la presente investigación:

- Permite gestionar archivos *PDF* seleccionando la ruta donde se encuentran en la computadora y todos los archivos que desee firmar.

- Posibilita 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.
- Posibilita 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.

El análisis realizado sobre las aplicaciones para firmar documentos digitales, permitió realizar un resumen (ver Tabla 2) teniendo en cuenta los siguientes parámetros:

- 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 sí lo requieren.
- Web: se refiere a si la herramienta es una aplicación web o no.

Tabla 2: Resumen del análisis de los sistemas homólogos *(Elaboración propia)*

Herramienta	Licencia	Web
ViaFirmaInbox	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 estudio del estado del arte realizado, el autor arriba a las siguientes impresiones:

- Las herramientas ViaFirmaInbox 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, además de que no es una aplicación 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.
- La herramienta UCI PDFSigner a pesar de ser desarrollada por la universidad y ser pública, ya no responde a las necesidades de dicha institución debido a que no puede ser utilizada a través de la web ni integrada a otras plataformas de amplio uso en la universidad.
- Algunas de estas herramientas cuentan con un conjunto de funcionalidades que pueden ser incluidas en la propuesta de solución, tales como: firmar documentos en lote, buscar documentos, rechazar el contenido de un documento.

Por todo lo antes expuesto, se puede resumir que las herramientas analizadas no cumplen con el objetivo de la presente investigación, pues ninguna reúne todos los requisitos de ser una aplicación web, gratuita y que permita la firma digital de documentos en la universidad. Sin embargo, cada una de ellas contribuyó a una mejor comprensión de las principales funcionalidades con las que debe cumplir un sistema de este tipo.

1.3 Entorno de desarrollo de la propuesta de solución

Para el desarrollo del sistema Portafirmas Digital se utilizarán tecnologías, metodología, lenguajes y herramientas específicas, las cuales deben ser utilizadas igualmente para el diseño y desarrollo de dicho sistema. Las mismas son definidas por el proyecto de desarrollo.

Lenguaje de modelado

El lenguaje de modelado es cualquier lenguaje informático gráfico o textual que provee el diseño y construcción de estructuras y modelos siguiendo un conjunto sistemático de reglas y marcos

(«Techopedia» 2017). Como lenguaje de modelado se utiliza el **Lenguaje Unificado de Modelado** (*UML* por sus siglas en inglés) en su versión 2.1, que se define como un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. Proporciona una forma estándar de diagramar planos de un sistema, abarcando las partes conceptuales (funciones del sistema), y los objetos concretos (clases escritas en lenguajes de programación específico, esquemas de bases de datos y componentes de software reutilizables) (González, Calderón y Usano 2005).

Herramienta de modelado

Dentro de las principales herramientas para el modelado se encuentran herramientas de Ingeniería de Software Asistida por Computadora (*CASE* por sus siglas en inglés) definidas por (Araujo y Rodríguez 2014) como herramientas informáticas que asisten al diseñador en algunas de las actividades relacionadas con el desarrollo de un sistema (requisitos, análisis, diseño, codificación y pruebas).

Visual Paradigm para *UML* es una herramienta que soporta el ciclo de vida completo en el desarrollo de software: análisis y desarrollos orientados a objetos, construcción, prueba y despliegue. Permite diseñar todo tipo de diagrama de clases, código inverso, generación de código a partir de diagramas y generar documentación (Morales, Clark y Oliva 2013). Para la presente investigación, se utiliza esta herramienta en su versión 8.0.

Herramienta para el control de versiones

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante («Git - Control de versiones» 2016). Para garantizar el control de versiones en el sistema se emplea el repositorio *GitLab*, específicamente el cliente ***Git*** en su versión 2.8, el cual, según («Git - Control de versiones» 2016):

- Es un sistema de control de versiones distribuido.
- No depende de acceso a la red o un repositorio central.
- Está enfocado a la velocidad, uso práctico y manejo de proyectos grandes.

Lenguajes de programación

Un lenguaje de programación está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (González y Arzuaga 2014). El lenguaje de programación definido para el desarrollo del sistema es ***TypeScript*** en su versión 2.6.2 que es un lenguaje de programación libre y de código abierto desarrollado y mantenido por *Microsoft*. Es un superconjunto de *JavaScript*, que esencialmente añade tipado estático y objetos basados en clase. Al extender la sintaxis de *JavaScript*, cualquier código *JavaScript* existente debería funcionar sin problemas. Está pensado para grandes proyectos, los cuales a través del compilador de *TypeScript* se traducen a código *JavaScript* original («TypeScript» 2017).

CSS3 es la versión de *CSS*[2] seleccionada para la solución. Es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas e interfaces de usuario para aplicaciones web. Está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas, los colores y las fuentes. Esta separación busca mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación de características presentacionales y reducir la complejidad y la repetición de código en la estructura del documento («CSS» 2017).

HTML5 es la versión de *HTML*[3] que se empleará en el desarrollo del sistema. Es un lenguaje de marcado que establece una estructura básica y un código para la definición del contenido de una página web. Se considera el producto de la combinación de *HTML*, *CSS* y *JavaScript* que provee tres características básicas: estructura, estilo y funcionalidad respectivamente. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de *HTML5* (Gauchat 2012).

Java es el lenguaje de programación que se empleará para implementar la funcionalidad de firmar los documentos digitales, específicamente la versión 8. Se considera un lenguaje de programación de alto nivel, orientado a objetos e independiente del hardware en la que se ejecuta. Esto es posible ya que al compilar el código *Java* se genera un código intermedio, que después la Máquina Virtual de *Java* (*JVM*) se

[2] Del inglés *Cascading Stylesheets* (Hojas de estilo en cascada).

[3] Del inglés *HyperText Markup Language* (Lenguaje de marcas de hipertexto).

encarga de transformarlo en el código máquina correspondiente a cada ordenador en el que se ejecute la aplicación («Java» 2017).

Librería

Una librería es un conjunto de recursos (algoritmos) prefabricados, que pueden ser utilizados por el programador para realizar determinadas operaciones. Las declaraciones de las funciones utilizadas en estas librerías, junto con algunas macros y constantes predefinidas que facilitan su utilización, se agrupan en ficheros de nombres conocidos que suelen tener las extensiones lib, bpl, a, dll, js, etc.

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.

Marco de trabajo

Para el desarrollo de la propuesta de solución se utiliza el marco de trabajo ***Angular*** que es un *framework* para aplicaciones web mantenido por *Google* y sigue una arquitectura basada en componentes. *Angular* combina plantillas declarativas, inyección de dependencia y mejores prácticas integradas para resolver los desafíos de desarrollo. Es multiplataforma, garantiza velocidad y rendimiento en el desarrollo ya que permite crear rápidamente vistas de interfaz de usuario con una sintaxis de plantilla simple y potente («Angular» 2017). Se empleará la última versión estable que es la 5.2.0.

Formato de intercambio de datos

JSON, acrónimo de *Java Script 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 sistema.

Servidor de aplicaciones web

Para el alojamiento del sistema se emplea el servidor web *HTTP* de código abierto ***Apache*** en su versión 2.4.29 desarrollado por la *Apache Software Foundation (ASF)*. Es usado para muchas tareas donde el contenido de una aplicación web necesita ser puesto a disposición de los usuarios de forma segura y confiable («Apache» 2018).

Entorno de Desarrollo Integrado (por sus siglas en inglés *IDE*)

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 («IntelliJ IDEA» 2017). Para la implementación del sistema se utiliza el *IntelliJ IDEA* en su versión 2017.2.5.

Herramienta para las pruebas de software

Las pruebas de software, son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada (Pressman 2010). La propuesta de solución de la investigación será validada a través de varias tipos y técnicas de pruebas, algunas de ellas de forma manual y otras mediante el uso de una herramienta que permite la realización de dicha tarea. Entre las herramientas existentes, se pretende utilizar la herramienta ***Acunetix* Web *Vulnerability Scanner*** para las pruebas de seguridad. *Acunetix* realiza automáticamente auditorías a aplicaciones web comprobando vulnerabilidades de Inyección *SQL*, *Cross site scripting* y otras vulnerabilidades que puedan ser explotadas por hackers.

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, Urrea y Rebollar 2008).

Para la implementación del sistema se decide utilizar una metodología ágil debido a que según (Merchán, Urrea y Rebollar 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 y a las exigencias del mercado. Las metodologías ágiles brindan 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.

Variación *AUP* para la UCI

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 (equipo de desarrollo, recursos, etc.), 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[4], 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.

[4]Colección de buenas prácticas de desarrollo procedentes de la industria y del gobierno. Es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software (SEI 2010).

Fases de la metodología (Sánchez 2015):

- Inicio: en esta fase se llevan a cabo las actividades relacionadas con la planeación del proyecto, se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- Ejecución: se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto.
- Cierre: se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre de proyecto.

1.4 Conclusiones del capítulo

En el presente capítulo se abordaron los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, en tal sentido se puede arribar a las siguientes conclusiones:

- 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 de los sistemas homólogos, permitió identificar las tendencias en cuanto al desarrollo de herramientas informáticas para la firma digital y las deficiencias que impiden que sean utilizadas en la UCI.
- El análisis de la metodología de desarrollo, así como las herramientas, tecnologías y lenguajes de programación utilizados en su implementación, permitió especificar el ambiente de desarrollo para la propuesta de solución.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN

Una de las prioridades cuando se desea desarrollar un software, es establecer un entendimiento entre el cliente y el equipo de trabajo en relación con los objetivos a lograr, realizando un correcto análisis y diseño de dicho sistema. El objetivo de este capítulo es presentar los resultados que se obtuvieron una vez cumplidas las fases de Análisis y Diseño que propone la metodología *AUP*-UCI. Se detallan las características del sistema, se especifican los requisitos funcionales y no funcionales del mismo y se presentan los artefactos generados para dar solución al problema planteado en la investigación.

2.1 Análisis

En esta fase se presentan las características generales del sistema y se realiza un análisis de los conceptos fundamentales asociados al proceso de firmado de documentos digitales en la UCI, los cuales se relacionan a través de un modelo conceptual.

2.1.1 Modelo conceptual

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). Tiene como objetivo identificar y explicar los conceptos más significativos del “mundo real”, identificando los atributos y las asociaciones existentes entre ellos (Sommerville 2011).

Para lograr un mejor entendimiento de los procesos que requieren informatización, se realizó un modelo conceptual (ver figura 1), con un total de siete (7) clases y ocho (8) relaciones, el cual recoge y describe los conceptos más importantes dentro del contexto del sistema, así como las relaciones entre ellos.

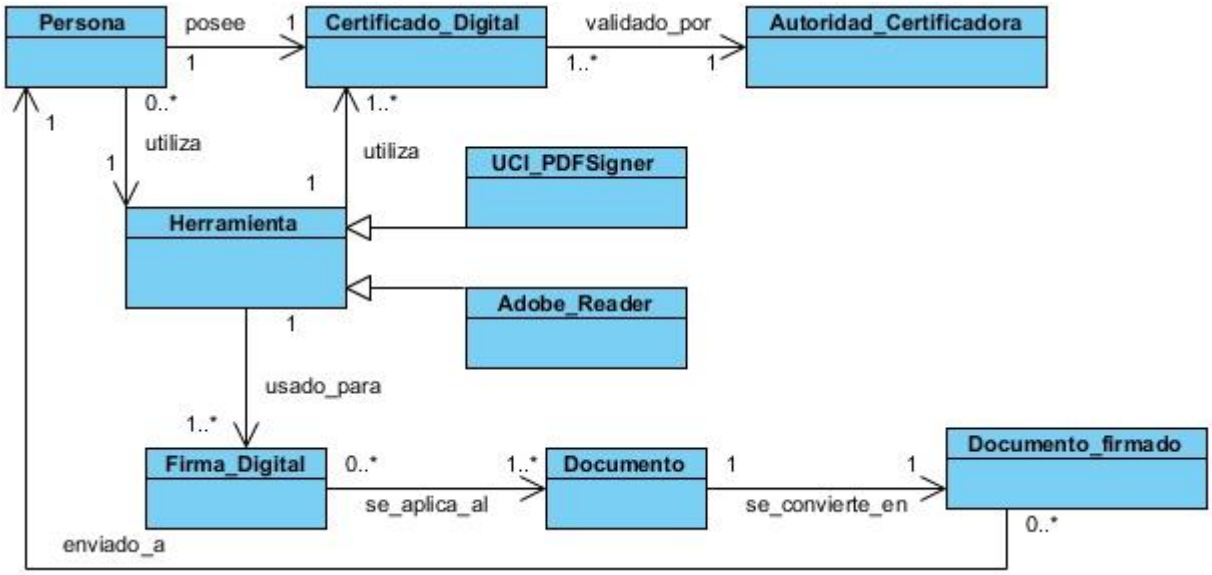


Figura 1: Modelo conceptual *(Elaboración propia)*

2.1.2 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.

2.1.3 Características de la propuesta de solución

Para dar cumplimiento al objetivo planteado se propone desarrollar un sistema basado en tecnologías web para llevar a cabo el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas. Para acceder al sistema es necesario que el usuario se autentique con rol de firmante, redactor o administrador, una vez dentro, el sistema deberá mostrar las opciones correspondientes a cada rol. Estas opciones son las siguientes:

- **Firmante:** permite el acceso a la aplicación mostrando las bandejas de peticiones “Pendientes”, “En Espera”, “Terminadas” y “Vencidas”, y a la configuración de usuario y la opción de firmar las peticiones pendientes.
- **Redactor:** muestra acceso a las bandejas permitidas al rol Firmante, a la bandeja de “Mis peticiones” y la opción de crear una nueva petición.
- **Administrador:** brinda acceso a la bandeja de peticiones en su totalidad y a las opciones de gestión de los nomencladores definidos en la aplicación.

En la tabla 3 se muestran las descripciones de las peticiones que se muestran en cada una de las bandejas de peticiones independientemente del rol del usuario que se encuentre autenticado:

Tabla 3: Descripción de las peticiones *(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 o rechazadas por el usuario autenticado.

Vencidas	Son las peticiones en las que el usuario autenticado sobrepasó la fecha de caducidad de la petición sin firmar el documento.
Mis peticiones	Son las peticiones que el usuario autenticado haya creado teniendo el rol de Redactor.

Desde las diferentes bandejas de peticiones que muestra el sistema, el usuario en dependencia del rol que ocupe en el mismo, puede acceder a un conjunto de opciones tales como:

1. En la bandeja de peticiones “Pendientes” se muestra un listado con los datos de las peticiones en las que el usuario autenticado le corresponde firmar. Estos datos son: remitente, asunto y fecha de caducidad. Las acciones que se pueden realizar sobre esos datos son: mostrar estado de la petición, mostrar el documento, rechazar el documento, firmar los documentos individualmente o en lote, buscar en el listado una petición determinada, y si el usuario tiene rol de Redactor puede crear una petición.
2. En la bandeja de peticiones “En Espera” se muestra un listado con los datos de las peticiones en las que el usuario forma parte del flujo de firmado. Estos datos son: remitente, asunto y fecha de caducidad. La acción que se puede realizar sobre estos datos es la de mostrar el estado de las peticiones.
3. En la bandeja de peticiones “Terminadas” se muestra un listado con los datos de las peticiones en las que ya el usuario firmó el documento o lo rechazó. Estos datos son: remitente, asunto, estado del documento (firmado o rechazado) y la fecha y hora en que lo firmó o rechazó. Las acciones que puede realizar sobre los datos son las de mostrar el documento, mostrar estado de la petición, descargar el documento y buscar una petición determinada.
4. En la bandeja de peticiones “Vencidas” se muestra un listado con los datos de las peticiones que sobrepasaron la fecha de caducidad sin ser firmadas por el usuario autenticado. Estos datos son: remitente, asunto y fecha de caducidad. Las acciones que se pueden realizar sobre estos datos son las de mostrar el documento y buscar una petición determinada.

5. En la bandeja “Mis peticiones”, solo visible para los usuarios con rol Redactor o Administrador, se muestra un listado con los datos de las peticiones que creó el usuario autenticado. Estos datos son: asunto y fecha de caducidad. Las acciones que se pueden realizar son: mostrar estado de la petición, mostrar documento, borrar una o varias peticiones a la vez y buscar una petición determinada.

En cada una de las acciones que se pueden realizar sobre los datos, mencionadas anteriormente, se despliega lo siguiente:

- Cuando se crea una nueva petición de firma, se introducen datos como: asunto, fecha de caducidad de la petición, se carga el documento a firmar (solo permitido el formato de extensión “.pdf”) y se crea el listado de los usuarios que deben firmar el documento.
- Para firmar un documento o varios se introduce el archivo de extensión “.p12” que contiene los certificados y su contraseña de acceso. Esta contraseña es cifrada con la clave *AesKey* generada aleatoriamente por el algoritmo simétrico *AES*. Luego se cifra esa clave *AesKey* con la clave pública del Paquete de Servicios a través del algoritmo asimétrico *RSA*. Se le envía al servicio de firma la contraseña cifrada, la clave *AesKey* cifrada y el archivo “.p12”. De esta forma se garantiza un envío seguro de los datos al Paquete de Servicios para que este firme el documento con uno de sus servicios. Una vez estos datos estén en el servicio, este descifra la clave *AesKey* con su clave privada usando *RSA* y descifra la contraseña de acceso al “.p12” usando el algoritmo *AES* con la clave *AesKey* ya descifrada, de esta manera se puede acceder al archivo “.p12” con su contraseña y proceder a firmar el documento haciendo uso de la librería *iText* de *Java*.
- Para rechazar un documento se registra la justificación del desacuerdo.

El usuario Administrador va a tener la posibilidad de gestionar los siguientes nomencladores:

- Usuario: se muestra un listado con los datos de todos los usuarios del sistema, estos datos son: usuario, rol, nombre, apellidos, correo, área, cargo y la activación. Estos usuarios se pueden editar, eliminar, buscar y crear un nuevo usuario.
- Cargo: se muestra un listado con los nombres de los cargos registrados en el sistema y la posibilidad de editarlos, eliminarlos y crear uno nuevo.

- Área: se muestra un listado con los nombres de las áreas registradas en el sistema y la posibilidad de editarlas, eliminarlas y crear una nueva.

Para ejecutar el flujo de firmado de los documentos es necesario crear una petición con los campos especificados anteriormente. Del listado de usuarios establecido que deben firmar el documento, al primero le llega la petición en su bandeja de “Pendientes” y al resto “En Espera”, este usuario puede realizar dos acciones: firmar o rechazar. En el caso de que el usuario autenticado firme, se traslada la petición para su bandeja de “Terminadas” y al segundo usuario le llega en “Pendientes” y este puede realizar cualquiera de las dos acciones, así sucesivamente ocurre con el resto de usuarios hasta que quede el documento firmado por todas las personas del listado. En cambio, si decide rechazar el documento, se cancela la petición. Todos los usuarios definidos en el flujo de firma de una petición y su usuario creador pueden consultar el estado de la misma en cualquier momento.

De esta manera queda conformada la propuesta de solución, enfocada en resolver los problemas que existen actualmente en la universidad para llevar a cabo el flujo de firmado de documentos digitales y la firma digital de esos documentos. Todo ese proceso controlado y supervisado desde una aplicación web por cada usuario registrado en el sistema. Quedando pendiente una futura integración del sistema con el sistema de gestión documental eXcriba para que sea utilizada dentro de su flujo de procesos.

2.2 Diseño

Modelar el sistema posibilita reflejar las propiedades o restricciones que este debe cumplir, a fin de satisfacer las necesidades del cliente (Oduardo y Santana 2013). En esta fase se presentan los requisitos funcionales y no funcionales del sistema y los respectivos artefactos generados, a partir de la metodología seleccionada.

2.2.1 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 requisitos funcionales también explican lo que el sistema no debe hacer (Sommerville 2011).

Después del encuentro con el cliente, se obtuvo un total de veintisiete (27) requisitos funcionales, a los cuales se les asignó una prioridad teniendo en cuenta la importancia fijada por el cliente a partir de sus necesidades. Los mismos se especifican en la tabla 4:

Tabla 4: Relación de requisitos funcionales del sistema *(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	Alta	RF24: Editar cargo	Baja
RF25: Eliminar cargo	Baja	RF26: Buscar cargo	Baja
RF27: Listar cargos	Alta		

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de recursos y del proceso de desarrollo, como impuestas por los estándares. Los requisitos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del mismo (Sommerville 2011).

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

- **Usabilidad:**

RnF 1: El sistema debe ser una aplicación web.

RnF 2: La aplicación debe mostrar una interfaz agradable e intuitiva para el usuario.

- **Seguridad:**

RnF 3: El sistema debe ser tolerante a fallos, y mostrar solo la información necesaria para orientar al usuario.

RnF 4: La contraseña de acceso a los archivos ".p12" de los usuarios debe ser enviada al Paquete de Servicios de manera cifrada.

RnF 5: Solo tendrán acceso a la aplicación las personas que se encuentren registradas en la misma.

- **Eficiencia:**

RnF 6: El sistema debe permitir que los usuarios interactúen con él de manera concurrente.

- **Funcionalidad:**

RnF 7: La aplicación debe gestionar y requerir información de usuarios para su uso.

- **Mantenibilidad:**

RnF 8: El software estará bien documentado de forma tal que contribuya al mantenimiento en caso de necesitarse.

2.2.3 Historias de usuario

La metodología *AUP*-UCI, en su escenario 4 para la disciplina Requisitos, genera como uno de sus artefactos a las Historias de Usuario (HU) (Sánchez 2015), que consiste en una técnica para encapsular los requisitos del software, a través de un conjunto de tablas que describen brevemente las características que desea el cliente para el sistema a desarrollar. Estas son lo suficientemente comprensibles y delimitadas para que los programadores puedan implementarlas.

Las HU se aplican a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente debe estar 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 HU no debe poseer demasiada información (Sánchez 2015). Se definen los siguientes parámetros para las HU orientados por el cliente:

- **Número:** número asignado a la HU.
- **Nombre:** nombre de la HU.
- **Programador:** nombre del programador responsable de la HU.
- **Iteración asignada:** iteración a la que pertenece la HU en el plan de iteraciones.
- **Prioridad:** 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.

Para la presente investigación se generaron un total de veintisiete (27) HU (ver Anexo 2), a continuación, solo se muestran dos (2) de ellas, pertenecientes a los RF 9 y RF 13 respectivamente, especificados en el sub-epígrafe 2.2.1.

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

Historia de usuario		
Número: HU_9	**Nombre:** Crear petición de firma	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite crear una nueva petición de firma en la aplicación.		
Observaciones: Para crear una petición de firma, el sistema debe mostrar un formulario en el que se ingresen datos como: asunto de la petición, fecha de caducidad, documento a adjuntar y listado de flujo de firma. 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). A esta funcionalidad solo tiene acceso el usuario que tenga el rol de redactor o administrador en el sistema.		

Tabla 6: HU_13 Firmar documento digital *(Elaboración propia)*

Historia de usuario		
Número: HU_13	**Nombre:** Firmar documento digital	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite firmar digitalmente un documento en formato ".pdf".		
Observaciones: Cada usuario del sistema tiene asociado un certificado digital validado por la Autoridad Certificadora de la UCI. En el sistema debe existir una opción de firmar el documento		

entre en el listado de peticiones pendientes de firma. Cuando el usuario presione en la opción de firmar, debe introducir su 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.

2.2.4 Descripción de la Arquitectura de Software

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 requisitos, 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.

Basada en Componente

El sistema que se diseña sigue una arquitectura basada en componentes porque este estilo arquitectónico es para aplicaciones compuestas por componentes individuales. Esta se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma en que se comparte el estado (Foukalas et al. 2005).

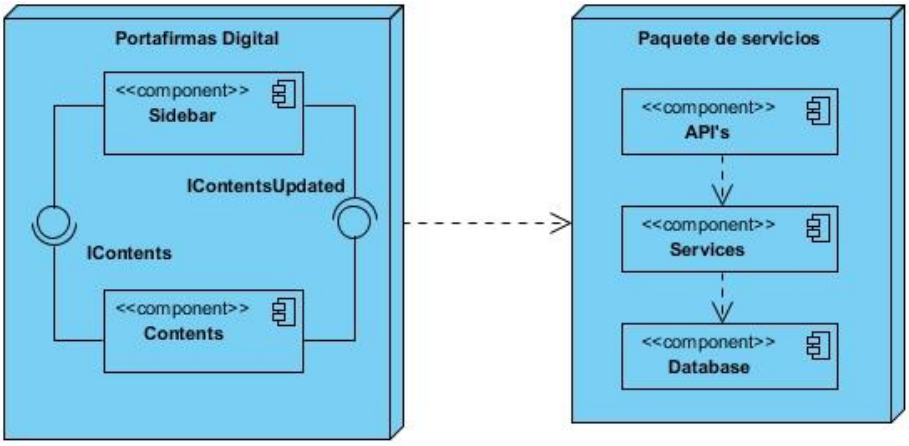


Figura 2: Arquitectura de software del Portafirmas Digital de la UCI *(Elaboración propia)*

Como se observa en la figura 2, el componente encargado de mostrar el menú lateral izquierdo del sistema es “Sidebar”, cuando el usuario seleccione una opción de ese menú, se muestra el componente “Contents” que es el encargado de proporcionar el contenido correspondiente a esa opción y actualizar el componente anterior con los datos que el usuario vaya modificando. Los datos son brindados por el Paquete de Servicios. Al ser una arquitectura basada en componentes, estos se ven como unidades independientes entre sí, que pueden mantenerse sin problemas con otros componentes, son altamente reutilizable y escalar es tan fácil como agregar más componentes a la aplicación existente. Interiormente cada componente del sistema sigue una arquitectura Modelo-Vista-Controlador (*MVC*), en la que el modelo lo constituyen las propiedades del componente, la vista es la plantilla en la que se muestra el componente y el controlador son los métodos que proporcionan la lógica del componente.

2.2.5 Patrones de diseño

Los patrones de diseño representan la descripción de un problema particular y recurrente, que aparece en contextos específicos y presenta un esquema genérico demostrado con éxito para su solución; este último se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí (Larman 2004).

En el diseño del sistema Portafirmas Digital de la Universidad de las Ciencias Informáticas se tuvieron en cuenta los siguientes patrones:

Patrones *GRASP*[5] (Patrones Generales 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:** consiste en asignar la responsabilidad de realizar una labor a la clase que tiene o puede tener los datos involucrados.

```
reject(request: Request){
    this.pendingRequestService.reject(request)
        .takeUntil(this.ngUnsubscribe) // <-- Unsubscribe when destroy emit
        .subscribe( (data) => {
            this.popupRejectService.hide();
            this.setPage(this.page);
            this.alertService.success(`La petición ha sido rechazada`);
        },
        error => {
          this.alertService.error(error);
        });
}
```

Figura 3: Código para rechazar las peticiones *(Elaboración propia)*

En la figura 3, se muestra el método **reject(request: Request)** evidenciándose el uso del patrón al darle la responsabilidad al componente **pending-request.ts** de rechazar la petición. Esa clase es la que posee toda la información necesaria para realizar la acción.

- **Creador:** consiste en asignar a una determinada clase B la responsabilidad de crear una instancia de la clase A al ocurrir alguna de las siguientes circunstancias:
    - B agrega a A,
    - B tiene los datos de inicialización de A,
    - B registra a A,
    - B utiliza estrechamente a A.

[5]*General Responsibility Assignment Software Patterns*, por sus siglas en inglés.

```
user = new User();

passwordRepeat: string = ""; //<-- Only for valid password equality

// For control if is Edit or Add action
isEdit: boolean = false;
```

Figura 4: Código del componente **addUser** *(Elaboración propia)*

Un ejemplo del empleo de este patrón en el sistema es que el componente **add-user.component.ts** tiene la responsabilidad de crear una nueva instancia de **usuario** para cumplir con la funcionalidad de adicionar un nuevo usuario.

- **Bajo acoplamiento:** consiste en asignar una responsabilidad a una clase de manera que el acoplamiento permanezca bajo. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases.

La arquitectura de software basada en componentes permite un bajo acoplamiento entre sus componentes. Estos pueden ser creados, modificados o eliminados en cualquier momento, lo que proporciona que la dependencia entre ellos sea baja.

- **Alta cohesión:** consiste en asignar una responsabilidad de manera que la cohesión permanezca alta. La cohesión es una medida de la fuerza con la que se relacionan las responsabilidades de un elemento.

La arquitectura de software basada en componentes permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Cada componente realiza solo las funcionalidades para las cuales fueron creados, colaborando entre ellos para cumplir con el resto de las funcionalidades, generando un bajo acoplamiento y fomentando la reutilización. Esto hace posible que el sistema sea flexible a cambios sustanciales con efecto mínimo.

Patrones *GoF*[6] (Patrones de la "pandilla de los cuatros"):

Los patrones *GoF* son los más utilizados en la implementación de algún software, se clasifican por su propósito en creacionales, estructurales y de composición, mientras que respecto a su ámbito se clasifican en clases y objetos (Gamma et al. 1997). Los patrones *GoF* utilizados en la propuesta de solución son los siguientes:

- ***Decorator* (Decorador):** proporciona metadatos adicionales a las clases que determinan cómo el componente se debe procesar, instanciar y usar en tiempo de ejecución.

Este patrón se utiliza en todas las clases del sistema ya que modifican las clases *JavaScript* adjuntando metadatos para que se sepa lo que significan y cómo deberían usarse. Un ejemplo de ello es la definición del componente **add-user.ts** con el decorador **@Component**.

```
@Component({
  selector: 'app-adduser',
  templateUrl: './adduser.component.html',
  styleUrls: ['./adduser.component.css']
})
```

Figura 5: Código de definición del componente **addUser** *(Elaboración propia)*

- ***Observer* (Observador):** se utiliza como soporte para que el sistema pueda comunicarse con una API[7] (Interfaz de Programación de Aplicaciones) externa accediendo a los datos brindados por el Paquete de Servicios.

El sistema constituye el "*Observer*" que se subscribe a los "Observables" del Paquete de Servicios para obtener objetos que se actualizarán cada vez que se produzca un cambio en los formularios. Un ejemplo de este patrón se evidencia en el método **search(pattern: string): Observable<HttpResponse<User[]>>** del componente servicio **user.service.component.ts** encargado de buscar un usuario específico entre los guardados en la base de datos del Paquete de Servicios para crear el flujo de firmado.

[6]Del inglés, *Gang-of-Four*

[7] Del inglés, *Application Programming Interface.* Brinda un conjunto de funciones y procedimientos para ser usados por otro software como una capa de abstracción.

```
search( pattern: string ) : Observable<HttpResponse<User[]>> {
    let params: HttpParams = new HttpParams();
    params = params.set('pattern', pattern);

    return this.http.get<User[]>(`${this.resourceUrl}/search`,
        { params: params, observe: 'response'})
        .map( (res: HttpResponse<User[]>) => this.convertArrayResponse(res) );
}
```

Figura 6: Código buscar del **UserService** *(Elaboración propia)*

- ***Singleton* (Instancia única):** se encarga de asegurar que una clase tenga una sola instancia y que proporcione un punto de acceso global a ella. El uso de este patrón es necesario cuando existen clases que tienen que gestionar de manera centralizada un recurso.

Este patrón se evidencia en el componente servicio **user.service.component.ts** que constituye un punto de acceso único del componente **users**, encargado de comunicarse con el servicio externo al sistema intercambiando los datos necesarios para llevar a cabo sus funcionalidades.

2.2.6 Diagrama de clases

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). Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones (Larman 2004).

En la figura 7 se muestran las clases que encapsulan toda la información de los objetos asociados el entorno de estudio de la propuesta de solución (*User*, *Area*, *Charge*, *Request*, *Document*) y las relaciones que se establecen entre ellos.

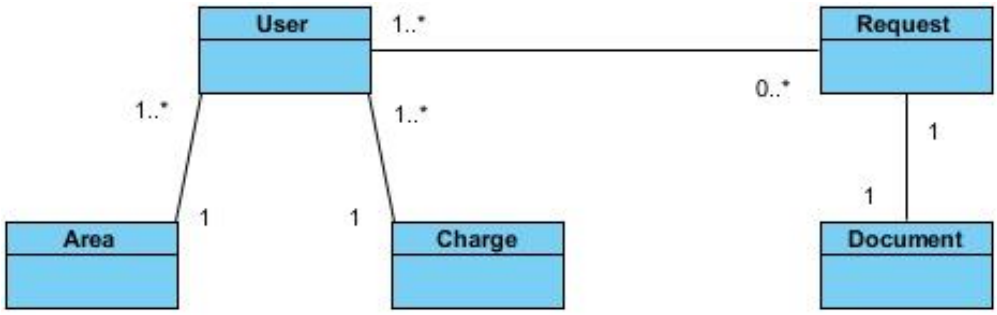


Figura 7: Diagrama de clases *(Elaboración propia)*

2.2.7 Diagrama 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). A continuación, se muestra el diagrama de despliegue propuesto para el Portafirmas Digital. El mismo, muestra la disposición física de los nodos que componen el sistema y el reparto de los componentes en dichos nodos.

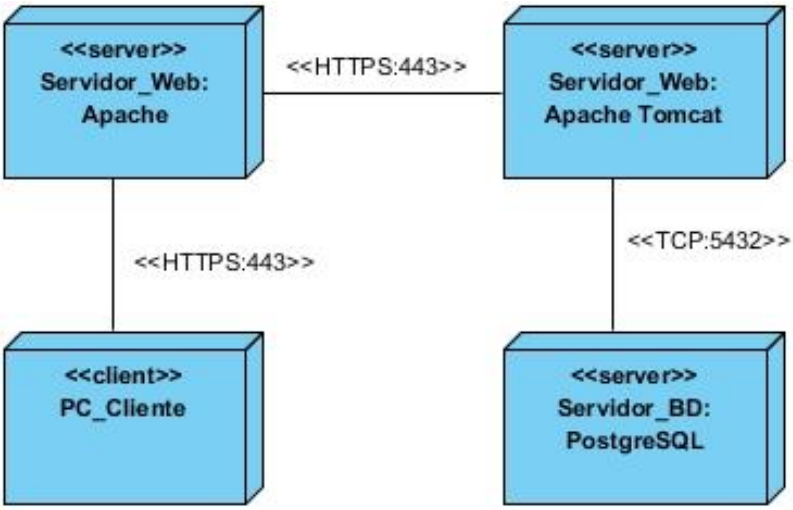


Figura 8: Diagrama de despliegue del Portafirmas Digital *(Elaboración propia)*

Descripción de elementos e interfaces de comunicación:

<<*HTTPS*>>: Protocolo Seguro de Transferencia de Hipertexto o *HTTPS*[8] empleado para establecer a través del puerto 443 la conexión segura entre el dispositivo de acceso cliente y el servidor de aplicaciones *Apache*, y entre el servidor de aplicaciones *Apache* y el servidor de aplicaciones *Apache Tomcat*.

<<*TCP*>>: Protocolo de Control de Transmisión o *TCP*[9] empleado para establecer la conexión entre el servidor de aplicaciones y el servidor de base de datos. Para el servidor de base de datos de *PostgreSQL* se define el puerto 5432. La conexión entre el servidor web y el servidor de base de datos permite dar órdenes y obtener información de esta.

2.3 Conclusiones del capítulo

Después de realizado el análisis y diseño de la propuesta de solución y haber generado los diferentes artefactos que dispone la metodología *AUP*-UCI, se puede concluir lo siguiente:

- El análisis de las características del sistema y la modelación del dominio, permitieron identificar los principales requisitos funcionales y no funcionales del sistema.
- La identificación de los patrones de diseño y el estilo arquitectónico del sistema, evidenciaron que la solución propuesta cuenta con un alto grado de resistencia ante posibles modificaciones.
- El diseño del diagrama de clases, facilitó la visión en cuanto a composición física y lógica del sistema.
- La generación de todos los artefactos requeridos por el modelo de desarrollo, documentaron la solución propuesta facilitando su posterior implementación.

[8]Del inglés, *Hypertext Transfer Protocol Secure.*

[9] Del inglés, *Transmission Control Protocol.*

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN

La etapa de implementación del software es el proceso de convertir una especificación del sistema en un sistema ejecutable (Sommerville 2011). Esta fase comprende la materialización, en forma de código, de todos los artefactos, descripciones y arquitectura propuestos en la etapa de análisis y diseño; con el objetivo de conformar el producto final requerido por el cliente (Larman 2004).

Una vez desarrollado el software, el mismo debe ser sometido a una serie de pruebas que muestren que el sistema se ajusta a su especificación y que cumple con las expectativas del usuario. Según (Sommerville 2011), esta etapa es conocida como la validación e implica una serie de procesos de comprobación, como las inspecciones y revisiones. Por lo que el presente capítulo tiene como objetivo documentar los resultados de las fases de implementación del sistema y de la estrategia de pruebas desarrollada, y validar la hipótesis de investigación.

3.1 Diagrama de componentes

El diagrama de componentes permite visualizar la estructura de un sistema informático atendiendo a las partes que lo conforman. Cada componente debe ser tratado como una unidad de composición independiente e indispensable dentro de un sistema, pudiéndose encontrar dependencias entre estos. Algunos ejemplos de componentes físicos lo constituyen los archivos, módulos, librerías, ejecutables y binarios (Sommerville 2011). Este diagrama provee una vista arquitectónica de alto nivel del sistema que ayuda a visualizar el camino de la implementación. Su realización posibilita tomar decisiones respecto a las tareas de implementación y los requisitos.

A continuación, la figura 9, muestra el diagrama de componentes del sistema Portafirmas Digital de la UCI, organizado acorde a la arquitectura basada en componentes propuesta por el *framework* Angular y descrita en el capítulo anterior.

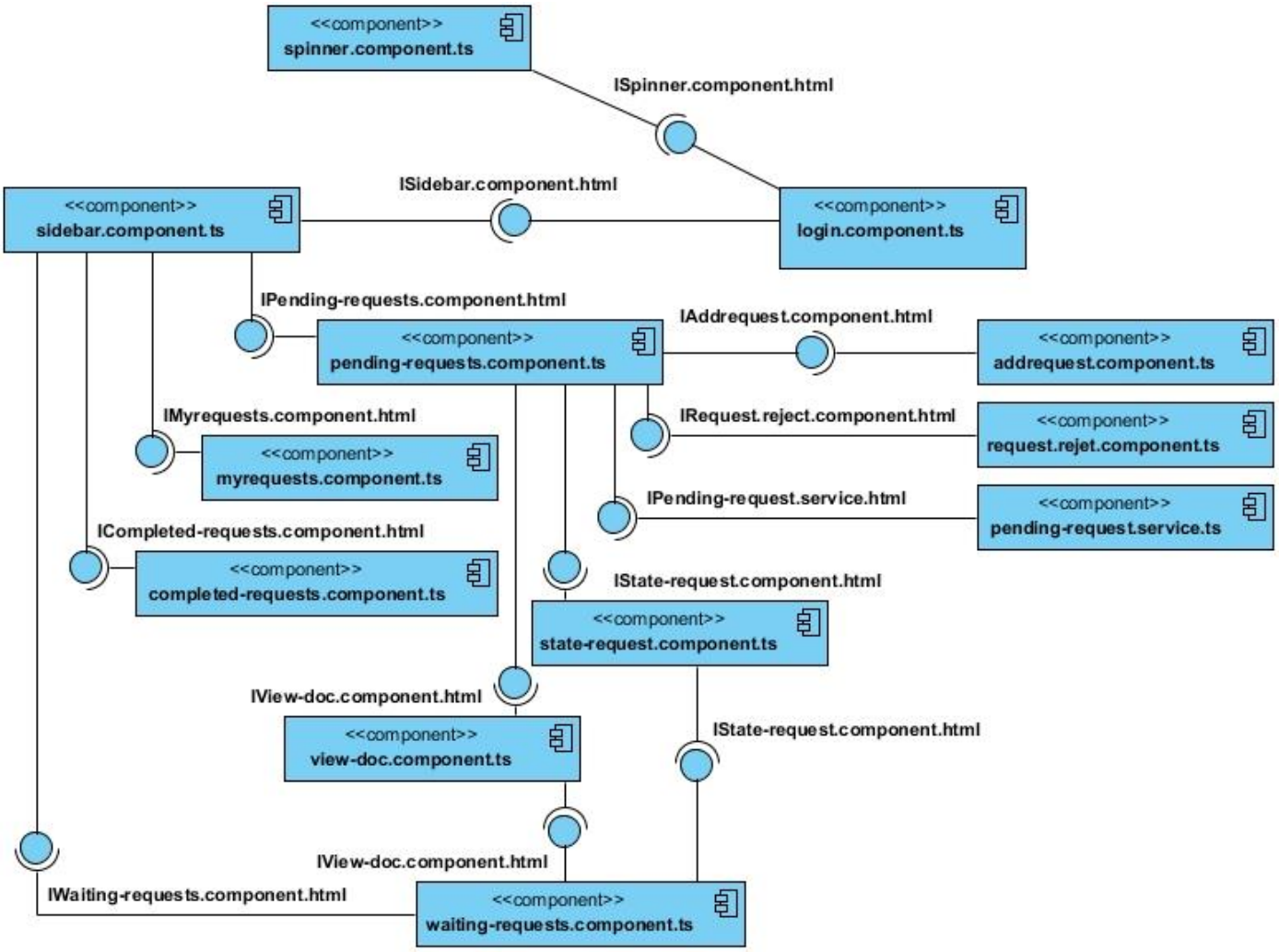


Figura 9: Diagrama de componentes del sistema Portafirmas Digital *(Elaboración propia)*

3.2 Estándares de codificación

Los estándares de codificación garantizan el mejoramiento de la comunicación en los equipos de desarrollo de software, reduce los errores de programación y mejora la calidad del software. Todo esto repercute en la competitividad de las empresas de software y en la productividad de sus trabajadores porque se mejora su facilidad de mantenimiento teniendo un gran impacto en la reducción de los costos de mantenimiento (Vera et al. 2015).

A continuación, se definen los estándares de codificación a utilizar en la implementación del sistema, basados en la guía de estilos de Angular («Angular» 2018).

- Todos los componentes, servicios y otros archivos deben cumplir con el principio de responsabilidad única para que cada uno tenga la responsabilidad sobre una sola parte de la funcionalidad proporcionada por el software y esta responsabilidad debe estar encapsulada en su totalidad por esa clase.
- Los archivos no deben exceder las 400 líneas de código.
- Las funciones deben ser pequeñas de implementación, no deben exceder las 75 líneas para promover su reutilización y mantenibilidad.
- Se deben usar nombres consistentes para los archivos siguiendo el patrón **feature.type.ts**, que describe las características del símbolo y luego su tipo.
- Debe coincidir el nombre del símbolo con el nombre del archivo.
- Se deben usar guiones para separar palabras en el nombre descriptivo.
- Se deben usar puntos para separar el nombre descriptivo del tipo.
- Se deben utilizar nombres de tipos convencionales, incluidos **.service**, **.component**, **.pipe**, **.module** y **.directive** ya que proporcionan una manera consistente de identificar rápidamente lo que está en el archivo.
- Se debe utilizar *UpperCamelCase*[10] para los nombres de las clases. Por ejemplo, el nombre de la clase ***AddUser***.
- Se debe sufijar un nombre de clase de servicio con el Servicio. Por ejemplo, la clase que obtiene los usuarios debería llamarse ***UserService***.
- Se debe poner el arranque y la lógica de inicio de la aplicación en un archivo llamado **main.ts**.
- Se debe usar camello inferior para nombrar los selectores de las directivas.

[10] Es un estilo de programación y convención del uso de la mayúscula y minúscula en los nombres de las clases, que indica que cada palabra debe ir concatenada y la primera letra de cada una en mayúscula.

- Se debe utilizar un valor de selector de elementos en minúsculas con guiones. Por ejemplo **'app-users'**.
- Se debe poner nombre a los archivos de especificación de prueba con un sufijo de **.spec**.
- Se deben nombrar los archivos de especificación de prueba de extremo a extremo (e2e) después de la característica que prueban con un sufijo de **.e2e-spec**.
- Se debe dar nombre a los módulos con el sufijo **.module.ts**. Por ejemplo **users.module.ts**.
- Se debe finalizar el nombre de archivo de un ***RoutingModule*** con **-routing.module.ts**. Por ejemplo **users-routing.module.ts**.
- Se debe usar una caja de camello inferior para nombrar propiedades y métodos.
- Se debe dejar una línea vacía entre las importaciones de terceros y las importaciones de aplicaciones.
- Se debe poner todo el código de la aplicación en una carpeta llamada **src**.
- Se debe crear una carpeta para un componente cuando tenga varios archivos adjuntos (**.ts**, **.html**, **.css** y **.spec**).
- Se deben crear carpetas con nombre para el área de características que representan.
- Se debe nombrar el módulo raíz **app.module.ts**.
- Se debe crear un módulo de características llamado ***SharedModule*** en una carpeta compartida. Por ejemplo, app/shared/**shared.module.ts** define *SharedModule*. Así los componentes, las directivas y las tuberías serán reutilizados y referenciados por los componentes declarados en otros módulos de características.
- Los servicios generalmente son singletons que se proporcionan una vez para toda la aplicación o en un módulo de características en particular.
- Se debe considerar colocar **@Input ()** o **@Output ()** en la misma línea que la propiedad que decora.

- Se debe limitar la lógica de un componente a solo la requerida para la vista. Toda otra lógica debería delegarse en los servicios.
- Se debe mover la lógica reutilizable a los servicios y mantenga los componentes simples y enfocados en su propósito previsto.
- Se debe nombrar los métodos del controlador de eventos con el prefijo **on** seguido del nombre del evento. Por ejemplo, el método **onActivate($event)**.
- Se debe poner la lógica de presentación en la clase de componente, y no en la plantilla; esto mejora la capacidad de prueba, el mantenimiento y la reutilización.
- Se deben proporcionar servicios al inyector angular en el componente superior donde se compartirán.
- Se debe utilizar el decorador de clase **@Injectable ()** en lugar del decorador de parámetros @Inject cuando se use tipos como *tokens* para las dependencias de un servicio, ya que provee una sintaxis mucho más sencilla.

3.3 Verificación y validación de la propuesta de solución

Luego del desarrollo de la propuesta de solución, se hace necesario verificar y validar el sistema implementado a través de una estrategia de pruebas que permita comprobar el cumplimiento de las especificaciones del diseño y de la codificación e identificar los posibles errores cometidos. Las pruebas de 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 (Sommerville 2011). Se diseñó una estrategia de prueba para el sistema Portafirmas Digital de la UCI, en función de garantizar su seguridad, verificar la conformidad de los requisitos establecidos y validar que el software hace lo que el usuario espera.

3.3.1 Pruebas funcionales

Las pruebas funcionales se aplican a un software determinado, con el objetivo de verificar que las funcionalidades implementadas se desempeñen de acuerdo a las especificaciones de los requisitos definidos con anterioridad. Al conocer las funciones específicas que se le asignaron a una aplicación para

su realización, pueden llevarse a cabo casos de pruebas que demuestren que cada función es completamente operativa mientras que al mismo tiempo se buscan errores en cada función (Pressman 2010). Este tipo de pruebas es basado en requisitos y se vincula a los datos de entrada.

Con el objetivo de realizar este tipo de pruebas al sistema, se diseñó un conjunto de casos de pruebas, tres (3) en total (ver Anexo 3), 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. En las celdas de la tabla del caso de prueba (ver tabla 7) se pueden encontrar los valores **V**, para datos válidos, **I**, para datos inválidos, y **N/A**, para datos a los que no es necesario proporcionarles un valor.

Tabla 7: Descripción de las variables del caso de prueba 1 *(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	Listado de flujo de firma	Campo de búsqueda	No	Permite buscar con autocompletamiento los usuarios que se insertarán automáticamente en el listado de flujo de firmado.

Tabla 8: Caso de prueba del RF9_Crear petición de firma *(Elaboración propia)*

Escenario	Descripción	1	2	3	4	Respuesta del sistema	Flujo Central
EC 1.1 Crear petición de forma correcta	Interfaz con el formulario para llenar los datos de la petición, si todos son correctos, se agrega la petición al sistema	V Petición 1	V 07/04/2018	V Acta de asamblea general.pdf	V Anaili Pérez, Adiel Gómez, María Moros	Agrega la petición y muestra un mensaje de notificación	1. Seleccionar, en la bandeja de peticiones Pendientes, la opción “Crear petición” 2. Llenar los campos correspondientes en el formulario y seleccionar la opción “Enviar”
EC 1.2 Crear petición con campos vacíos	Interfaz con el formulario para llenar los datos de la petición, si existe algún campo vacío, se muestra un mensaje pidiendo llenar el campo	I 	I 	V Balance del partido.pdf	I 	Comprueba si los campos están vacíos, si lo están, muestra un mensaje que solicita llenarlos	
EC 1.3 Crear	Interfaz con el formulario	V Petici	V 03/05/	I Resu	V Pepe	Comprueba el formato del	

petición con un documento en formato distinto a “pdf”	para llenar los datos de la petición, si se inserta un documento con formato distinto a “pdf”, se muestra un mensaje de error	ón 2	2018	men anual de econo mía.d ocx	López, Juan Prieto, Carlos Ferrer	documento, si es distinto a “pdf”, muestra un mensaje de error	

Como resultado final de las pruebas funcionales, se obtuvo, en una primera iteración, un total de nueve (9) no conformidades, divididas en cuatro (4) de ortografía, dos (2) de funcionalidad y tres (3) de validación. De estas, se resolvieron seis (6), y tres (3) quedaron pendientes. En una segunda iteración, se detectaron cinco (5) no conformidades, tres (3) de ellas pendientes de la iteración anterior y dos (2) nuevas de validación, las cuales fueron resueltas en su totalidad. En una tercera iteración no se identifican nuevas inconformidades, obteniendo, de esta manera, resultados satisfactorios. La siguiente gráfica, muestra los resultados antes descritos:

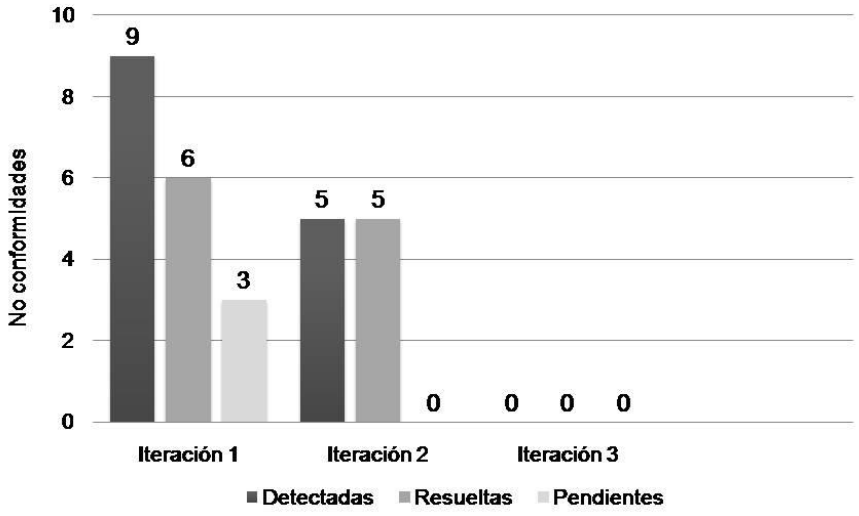


Figura 10: Resultados de las pruebas funcionales *(Elaboración propia)*

La realización de las pruebas funcionales posibilitó la detección temprana de errores en la aplicación, así como la corrección de cada uno de estos. Entre las faltas más comunes se encontraban los errores ortográficos en la interfaz de usuario y la incorrecta validación de los datos.

3.3.2 Pruebas de seguridad

Las pruebas de seguridad intentan verificar que los mecanismos de protección que se construyen en un sistema lo protegerán realmente de cualquier irrupción inapropiada. Buscan medir la confidencialidad, integridad y disponibilidad de los datos. Se diseñan para detectar las vulnerabilidades del entorno en el sistema, las comunicaciones de red que ocurren conforme los datos pasan del Paquete de Servicios al sistema y viceversa, y el entorno en el Paquete de Servicios. 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).

Para la realización de las pruebas de seguridad, se empleó la herramienta *Acunetix* Web *Vulnerability Scanner,* caracterizada en el epígrafe 1.3. En una primera iteración, se obtuvo un total de once (11) no conformidades, divididas en cinco (5) de nivel medio y seis (6) de nivel bajo. De las de nivel medio, destacó el uso del protocolo no seguro para el envío de datos, así como envío de credenciales en texto claro al Paquete de Servicios. Las de nivel bajo estuvieron relacionadas con problemas para la protección

contra ataques de fuerza bruta a la página de autenticación, así como directorios que pueden ser accesibles directamente sin pasar la autenticación y la protección de las *cookies* y las sesiones en el navegador. Todas estas deficiencias fueron corregidas en la primera iteración, y para una segunda, no se identificó ninguna nueva, por lo cual se obtuvo finalmente una herramienta que cumple con los requisitos de seguridad definidos para la misma. Los resultados antes descritos, se muestran a continuación en la siguiente gráfica:

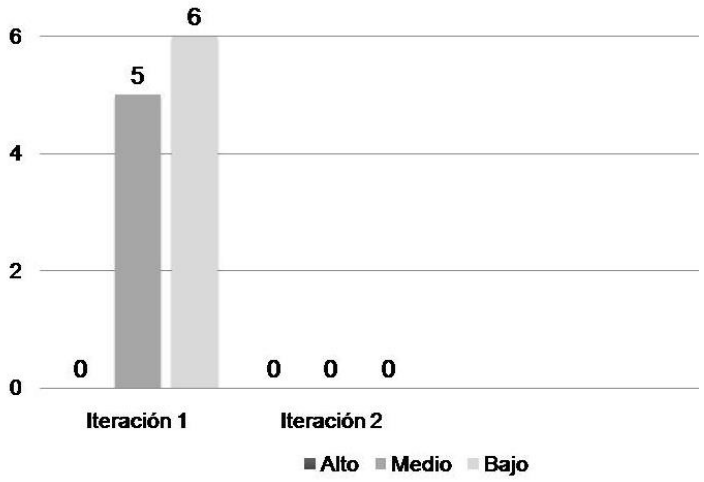


Figura 11: Resultados de las pruebas de seguridad *(Elaboración propia)*

3.3.3 Pruebas de usabilidad

La usabilidad es la capacidad de un producto de software de ser entendido, aprendido, utilizado y atractivo al usuario, cuando es utilizado bajo condiciones especificadas (Lamancha 2006). Es un atributo intangible del software, por lo que es difícil de visualizar, medir y reconocer como un factor determinante de su calidad. Una mayor atención a este elemento contribuiría a incrementar la calidad del producto percibida por el usuario, sin un aumento excesivo en el costo de desarrollo (Mascheroni et al. 2012).

Las pruebas de usabilidad intentan encontrar problemas de usabilidad del sistema en su interacción con humanos. Según el enfoque tradicional, estas pruebas se aplican sobre el producto software para garantizar o determinar si el mismo alcanza un nivel aceptable de usabilidad. Entre los modelos que presenta para evaluar se encuentran los heurísticos y los empíricos. Los primeros implican la participación

de expertos especialistas en usabilidad, mientras que los segundos constan de técnicas que implican la participación de los usuarios (Mascheroni et al. 2012).

Para la realización de las pruebas de usabilidad al sistema se hace uso de la “Lista de Chequeo de Usabilidad para sitios web”, desarrollada por los especialistas del grupo de Seguridad del Departamento de Evaluación de Productos de Software (DEPSW), perteneciente al Centro Nacional de Calidad de Software (CALISOFT). A continuación, se muestran los resultados de dichas pruebas.

En la tabla 9 se reflejan las categorías en las que se dividen los indicadores, cuántos indicadores existen por cada categoría, de ellos, cuántos se aplican a la evaluación del Portafirmas Digital (Proceden), y de los evaluados, cuántos están implementados correcta e incorrectamente en el sistema.

Tabla 9: Resultado de las pruebas de usabilidad *(Elaboración propia)*

Categorías de los indicadores	Indicadores	Proceden	Correctos	Incorrectos
Visibilidad del sistema	17	9	7	2
Lenguaje común entre sistema y usuario	11	4	4	0
Libertad y control por parte del usuario	29	22	17	5
Consistencia y estándares	33	21	17	4
Estética y diseño minimalista	18	7	7	0
Prevención de errores	8	7	4	3
Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores	11	8	1	7
Ayuda y documentación	11	0	0	0

Flexibilidad y eficiencia de uso	6	3	3	0
Total	**144**	**81**	**60**	**21**

Como se observa en la tabla, de los 144 parámetros pertenecientes a la lista de chequeo, solo proceden 81. En una primera iteración se evaluaron como correctos 60 parámetros, identificando 21 no conformidades, para un 75 % de usabilidad. Los principales problemas estuvieron relacionados con las búsquedas, el paginado de los elementos, la ayuda a los usuarios mediante documentación, problemas con los enlaces entre páginas y para enfrentar errores en el sistema. Las no conformidades se revisaron y de las 21 encontradas, fueron solucionadas 17, obteniendo un sistema con un 95 % de usabilidad. Las no conformidades que no fueron resueltas están relacionadas con la imposibilidad de volver atrás luego de realizar una acción relevante como firmar los documentos, la ausencia de ayuda en los campos de los formularios y los atajos de teclado.

3.4 Interfaces principales del sistema Portafirmas Digital de la UCI

Una vez desarrollado el Portafirmas Digital, es posible visualizar las pantallas principales del mismo, donde se observa el resultado obtenido durante la implementación de las historias de usuarios descritas en el sub-epígrafe 2.2.3.

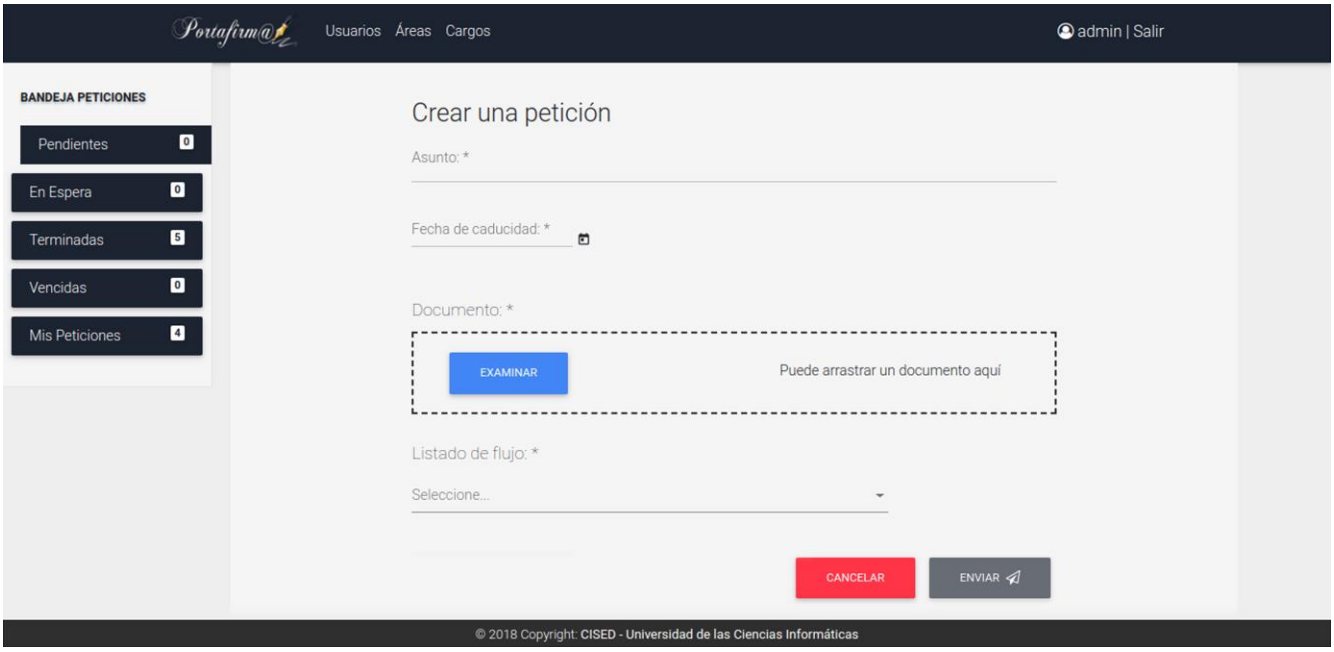


Figura 12: Interfaz de la página para crear una petición de firma *(Elaboración propia)*

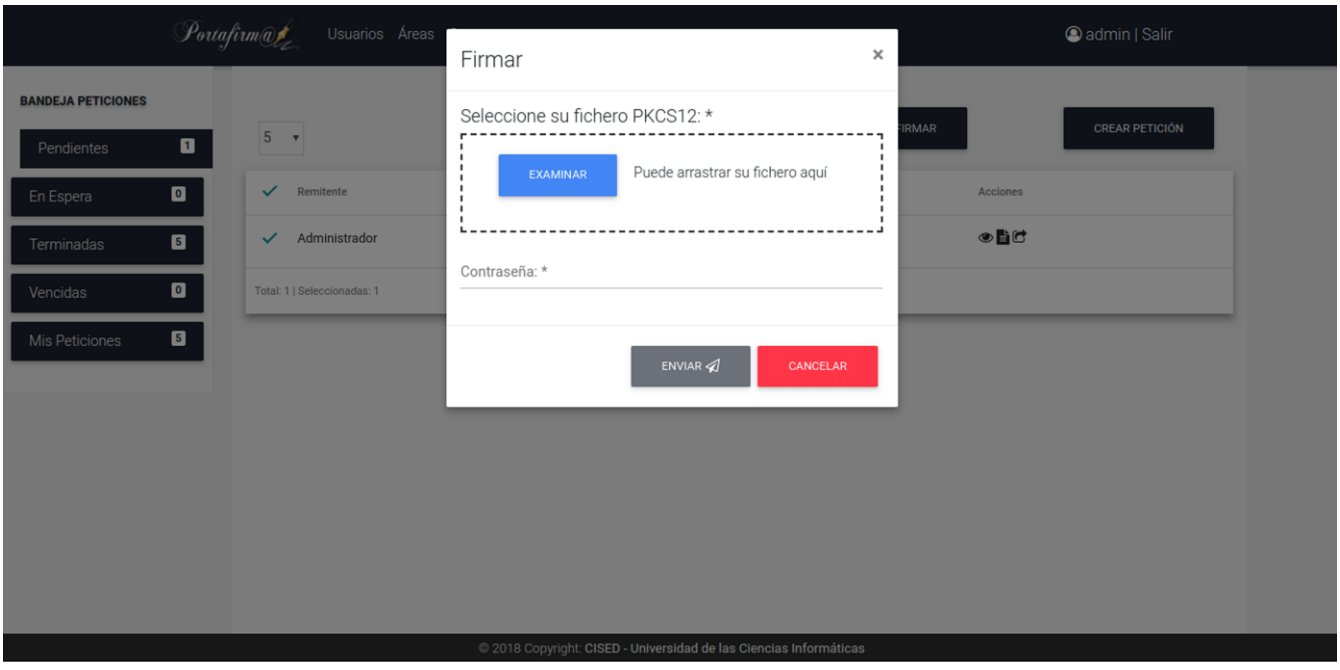


Figura 13: Interfaz de la página para firmar un documento *(Elaboración propia)*

3.5 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 seis (6) posibles expertos se seleccionaron los que su coeficiente de competencias fue igual o superior a 0,7, para un total de cuatro (4), detallados en la tabla 10. De los expertos, tres (3) obtuvieron un coeficiente de competencia alto y uno (1) de ellos medio. Los expertos poseen suficiente conocimiento en la investigación y aplicaciones web, todos graduados universitarios y una parte de ellos ha obtenido títulos de formación académica.

Tabla 10: Expertos seleccionados en la validación de la investigación *(Elaboración propia)*

Expertos	Área	Años de experiencia
MSc. Joelsy Porven Rubier	Dirección de Redes y Servicios Telemáticos	15
Ing. Oscar Lázaro Garcés Pérez	Dirección de Seguridad Informática	7
Ing. Michel James Navarro	Dirección de Seguridad Informática	6
MSc. Henry Raúl González Brito	Dirección de Seguridad Informática	12

Realización del cuestionario a los expertos: Después de contar con los cuatro (4) expertos seleccionados se les realizó un cuestionario compuesto por cuatro (4) 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 14 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 mejorar el proceso de autenticación de documentos digitales en la Universidad de las Ciencias Informáticas y se obtuvo como recomendación de seguridad que en los campos de contraseña sea obligatorio escribirla.

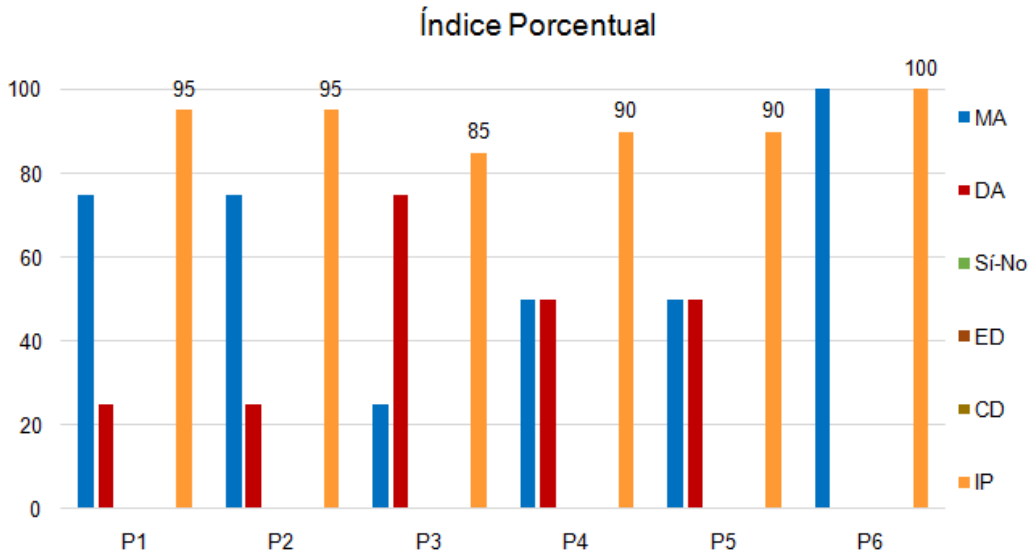


Figura 14: Resultados de la aplicación de la escala de *Likert (Elaboración propia)*

La presente investigación culmina con el desarrollo del sistema Portafirmas Digital de la Universidad de las Ciencias Informáticas, el cual mejora el proceso de autenticación de documentos digitales en dicha entidad permitiendo llevar a cabo el flujo de firmado de documentos y su firma digital desde una aplicación web de manera centralizada. Esto garantiza una mayor agilidad en sus procesos y reducción de costes por el ahorro de papel y de espacio para almacenar los documentos. El sistema fue desarrollado como

una tecnología de apoyo a todos los procesos que se lleven a cabo en la universidad e impliquen firmar digitalmente documentos.

3.6 Conclusiones del capítulo

En este capítulo se han abordado los elementos de la implementación del Portafirmas Digital de la UCI, así como las pruebas realizadas al mismo y los resultados obtenidos; lo cual permite arribar a las siguientes conclusiones:

- La elaboración del diagrama de componentes, facilitó la comprensión de la estructura general del sistema a través de sus componentes.
- El correcto uso de los estándares de codificación, permitió una mejor legibilidad y comprensión del código, facilitando su mantenimiento.
- La implementación del sistema permitió la obtención de una aplicación funcional y completamente operativa.
- La ejecución de la estrategia de pruebas especificada, permitió detectar y corregir deficiencias presentes en la solución y ofrecer una aplicación con mayor calidad, seguridad y usabilidad.
- La validación de la hipótesis a través del método de criterio de expertos, ofreció una alta valoración de la propuesta de solución.

CONCLUSIONES GENERALES

Una vez cumplidos los objetivos trazados en la presente investigación se concluye lo siguiente:

- La firma de documentos digitales desde una aplicación web mejoró el proceso de autenticación de documentos digitales en la universidad.
- La ejecución del flujo de firmado de documentos digitales desde una misma aplicación contribuyó al proceso de supervisión y control de los documentos firmados.
- 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:

- Incluir soporte para firmar digitalmente documentos electrónicos en otros formatos.
- Integrar el sistema desarrollado al sistema de gestión documental eXcriba para que sea utilizado en su flujo de procesos.
- Validar que los certificados digitales empleados para firmar los documentos sean emitidos por la Autoridad Certificadora de la Universidad de las Ciencias Informáticas.

REFERENCIAS BIBLIOGRÁFICAS

Agenda de firmas electrónicas - Viafirma Inbox | Viafirma. [en línea], 2017. [Consulta: 24 mayo 2017]. Disponible en: https://www.viafirma.com/es/inbox-agenda-firmas-electronicas.

Angular. [en línea], 2017. [Consulta: 1 diciembre 2017]. Disponible en: https://angular.io/.

Angular - Style Guide. [en línea], 2018. [Consulta: 4 abril 2018]. Disponible en: https://angular.io/guide/styleguide.

ARAUJO, P.B. y RODRÍGUEZ, S.A., 2014. Enfoque para la validación sintáctica de modelos organizacionales de Sistemas Multiagentes. *Ciencia y Tecnología*, vol. 1, no. 14, pp. 123-144. ISSN 1850-0870.

BÁSTER, D.R.L., 2011. *Infraestructura de firma y validación digital de los documentos clínicos electrónicos generados por el sistema alas HIS* [en línea]. S.l.: Universidad de las Ciencias Informáticas. [Consulta: 17 octubre 2017]. Disponible en: https://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04004_11/1/TD_04004_11.pdf.

BONDARTCHUK, R.Q., 2009. *Firma Digital de Documentos utilizando Smart Cards* [en línea]. S.l.: s.n. [Consulta: 18 mayo 2017]. Disponible en: https://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_1942_09/1/TD_1942_09.pdf.

CARVAJAL, A., 2007. PKI* y firmas digitales: aplicaciones reales. *Revista Inventum*, no. 3, pp. 13–26.

CORRALES, F.J.I., 2011. Metodología Investigativa y Tradicional., no. No. 14, pp. 8. ISSN 1988-6047.

DÍAZ, J.C.G., 1995. *Criptografía: historia de la escritura cifrada*. S.l.: Editorial Complutense. ISBN 978-84-89365-29-2.

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

@firma | Portal de Administración Electrónica. [en línea], 2016. [Consulta: 24 mayo 2017]. Disponible en: https://ws024.juntadeandalucia.es/ae/adminelec/areatecnica/afirma.

FOUKALAS, F., NTARLADIMAS, Y., GLENTIS, A. y BOUFIDIS, Z., 2005. Protocol reconfiguration using component-based design. *IFIP International Conference on Distributed Applications and Interoperable*

Systems [en línea]. S.l.: Springer, pp. 148–156. Disponible en: https://link.springer.com/content/pdf/10.1007%2F11498094_14.pdf.

GAMMA, E., HELM, R., JOHNSON, R. y VLISSIDES, J., 1997. *Design Patterns: Elements of Reusable Object-Oriented Software - DesignPatterns* [en línea]. S.l.: s.n. [Consulta: 6 marzo 2018]. Disponible en: http://www.uml.org.cn/c++/pdf/DesignPatterns.pdf.

GAUCHAT, J.D., 2012. *El gran libro de HTML5, CSS3 y Javascript* [en línea]. Barcelona: s.n. [Consulta: 23 noviembre 2017]. ISBN 978-84-267-1782-5. Disponible en: https://gutl.jovenclub.cu/wp-content/uploads/2013/10/El+gran+libro+de+HTML5+CSS3+y+Javascrip.pdf.

Git - Control de versiones. [en línea], 2016. [Consulta: 24 noviembre 2017]. Disponible en: https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones.

GONZÁLEZ, C.P. y ARZUAGA, D.D., 2014. *Módulo de edición de plantillas y recepción de órdenes de impresión para el Sistema de Personalización de Documentos de Identidad basado en tecnologías libres* [en línea]. S.l.: Universidad de las Ciencias Informáticas. [Consulta: 6 marzo 2018]. Disponible en: https://repositorio_institucional.uci.cu/jspui/bitstream/123456789/7058/1/TD_07881_15.pdf.

GONZÁLEZ, P.P., CALDERÓN, C.L.P. y USANO, R.R., 2005. Modelado de procesos y desarrollo de sistemas software: integración entre UML y EPC. *IX Congreso de Ingeniería de Organización: Gijón* [en línea]. S.l.: s.n., pp. 87. Disponible en: https://www.researchgate.net/profile/Carlos_Parra_Calderon/publication/45337484_Modelado_de_procesos_y_desarrollo_de_sistemas_software_integracion_entre_UML_y_EPC/links/02bfe5123468912dee000000/Modelado-de-procesos-y-desarrollo-de-sistemas-software-integracion-entre-UML-y-EPC.pdf.

GROSSO, A., 2011. Prácticas de software, Experiencias sobre la Ingeniería y Management del Software. [en línea]. S.l.: [Consulta: 6 marzo 2018]. Disponible en: https://ar.linkedin.com/in/andresgrosso/es?trk=public_profile_card_url.

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

Información sobre Java 8. [en línea], 2017. [Consulta: 7 mayo 2018]. Disponible en: https://www.java.com/es/download/faq/java8.xml.

IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains. [en línea], 2017. [Consulta: 1 diciembre 2017]. Disponible en: https://www.jetbrains.com/idea/.

LAMANCHA, B.P., 2006. *Proceso de testing funcional independiente* [en línea]. Montevideo, Uruguay: Universidad de la República. Disponible en: http://www.ces.com.uy/documentos/imasd/Tesis-Beatriz_Perez_2006.pdf.

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

Learn to style HTML using CSS. *Mozilla Developer Network* [en línea], 2017. [Consulta: 23 noviembre 2017]. Disponible en: https://developer.mozilla.org/en-US/docs/Learn/CSS.

LÓPEZ, M.J.L., 2010. *Criptografía y Seguridad en los Computadores*. 4ta Edición. España: s.n.

MASCHERONI, M.A., GREINER, C.L., PETRIS, R.H., DAPOZO, G.N. y ESTAYNO, M.G., 2012. Calidad de software e ingeniería de usabilidad. *XIV Workshop de Investigadores en Ciencias de la Computación* [en línea]. Red de Universidades con Carreras en Informática (RedUNCI): s.n., ISBN 978-950-766-082-5. Disponible en: http://sedici.unlp.edu.ar/bitstream/handle/10915/19202/Documento_completo.pdf?sequence=1.

MENEZES, A.J., OORSCHOT, P.C. van y VANSTONE, S.A., 1996. *Handbook of applied cryptography* [en línea]. S.l.: s.n. [Consulta: 24 mayo 2018]. Disponible en: http://labit501.upct.es/~fburrull/docencia/SeguridadEnRedes/teoria/bibliography/HandbookOfAppliedCryptography_AMenezes.pdf.

MERCHÁN, L., URREA, A. y REBOLLAR, R., 2008. Definición de una metodología ágil. [en línea], vol. 6, no. No. 1. [Consulta: 16 noviembre 2017]. ISSN 1794-192. Disponible en: https://dialnet.unirioja.es/descarga/articulo/2753852.pdf.

MOGOLLON, M., 2007. *Cryptography and Security Services: Mechanisms and Applications* [en línea]. University of Dallas, USA: Cybertech Publishing. [Consulta: 7 mayo 2018]. ISBN 978-1-59904-839-0. Disponible en: https://pdfs.semanticscholar.org/38f4/04345a2c7a887e373f2c457f4d90a9afe748.pdf.

MORALES, Y.R., CLARK, M.E.M. y OLIVA, A.T., 2013. Extensión de la herramienta Visual Paradigm para la generación de clases de acceso a datos con Doctrine 2.0. *Serie Científica de la Universidad de las*

Ciencias Informáticas [en línea], vol. 6, no. 10. ISSN 2306-2495. Disponible en: https://publicaciones.uci.cu/?journal=SC&page=article&op=view&path[]=1271.

ODUARDO, E.L. y SANTANA, Y.L.A., 2013. *Desarrollo del Módulo de gestión de reportes estadísticos para el sistema AiresProxyAudit* [en línea]. S.l.: Universidad de las Ciencias Informáticas. [Consulta: 8 diciembre 2017]. Disponible en: https://repositorio_institucional.uci.cu/jspui/bitstream/ident/8298/2/TD_06465_13.pdf.

PAREDES, G.G., 2006. Introducción a la Criptografía. *Artículos* [en línea], vol. Vol. 7, no. No. 7. ISSN 1067-6079. Disponible en: http://www.ru.tic.unam.mx/tic/bitstream/handle/123456789/1105/511.pdf?sequence=1&isAllowed=y.

PARKINSON, S., MORIARTY, K., SCOTT, M., RUSCH, A. y NYSTROM, M., 2014. PKCS# 12: Personal Information Exchange Syntax v1. 1. [en línea], Disponible en: https://pdfs.semanticscholar.org/0b19/6cc3787306230f41ad2af1a0f06cf87d74a3.pdf.

Portafirmas | Guadaltel. [en línea], 2017. [Consulta: 16 noviembre 2017]. Disponible en: http://www.guadaltel.com/producto-portafirmas.

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

PRESSMAN, R., 2010. *Ingeniería del Software. Un Enfoque Práctico*. 7ma. S.l.: McGRAW-HILL Interamericana. ISBN 978-607-15-0314-5.

RAE, A., 2017. Diccionario de la lengua española - Edición del Tricentenario. *Diccionario de la lengua española* [en línea]. [Consulta: 24 octubre 2017]. Disponible en: http://dle.rae.es/?id=BHcfHjo.

RODRÍGUEZ, A.S., MORALES, A.P. y FABRE, N.J.S., 2016. INFRAESTRUCTURA DE LLAVE PÚBLICA PARA EL INTERCAMBIO SEGURO DE INFORMACIÓN: INTEGRO-PKI. [en línea], Disponible en: http://www.informaticahabana.cu/sites/default/files/ponencias/GES66.pdf.

SÁNCHEZ, T.R., 2015. *Metodología de desarrollo para la Actividad productiva de la UCI*. 2015. S.l.: s.n.

SARMIENTO, F., Johana, 2013. Visión General de los Diagramas de Despliegue. *UML* [en línea]. [Consulta: 6 marzo 2018]. Disponible en: http://umldiagramadespliegue.blogspot.com/.

SEI, 2010. *Mejora de los procesos para el desarrollo de mejores productos y servicios*. 2010. S.l.: s.n.

PEÑA, J.C.S., MUSA, Y.N., LIMA, R.S. y SUÁREZ, A.R., 2009. Propuesta de aplicación de un sistema de Infraestructura de Clave Pública (Public Key Infrastructure« PKI») y los Certificados Digitales en la trazabilidad de productos agrícolas. *Revista Ciencias Técnicas Agropecuarias* [en línea], vol. 18, no. 4. Disponible en: http://www.redalyc.org/pdf/932/93212367015.pdf.

SILVA, N.M.M., 2004. *Estudio monográfico sobre técnicas de criptografía* [en línea]. Soyapango, El Salvador: Universidad Don Bosco. [Consulta: 3 noviembre 2017]. Disponible en: http://rd.udb.edu.sv:8080/jspui/bitstream/11715/580/1/034436_tesis.pdf.

SOMMERVILLE, I., 2011. *Ingeniería de Software*. 9na. México: s.n. ISBN 978-607-32-0603-7.

Techopedia. *Techopedia.com* [en línea], 2017. [Consulta: 23 noviembre 2017]. Disponible en: https://www.techopedia.com/definition/20810/modeling-language.

TypeScript. *TypeScript* [en línea], 2017. [Consulta: 6 marzo 2018]. Disponible en: http://www.typescriptlang.org/docs/home.html.

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]. México: Universidad Popular Autónoma del Estado de Puebla. [Consulta: 16 noviembre 2017]. Disponible en: www.eumed.net/tesis-doctorales/2014/jlcv/jlcv.zip.

VERA, D., DANILO, E., DOMÍNGUEZ VERA, V.B., RAMÍREZ, A. del Á., BARRIOS, M. y ANTONIO, J., 2015. Reglas de calidad para la codificación estandarizada en Lenguaje C: una propuesta para la enseñanza a nivel superior. *Proyectos institucionales y de vinculación*, vol. 3, no. 6, pp. 155–175. ISSN 2395-9029.

Welcome to The Apache Software Foundation! [en línea], 2018. [Consulta: 4 mayo 2018]. Disponible en: https://www.apache.org/.

WOLFGANG, R. y WOLFGANG, E., 2004. *Smart Card Handbook*. S.l.: John Wiley & Sons. ISBN 978-0-470-85669-7.

X.509v3 *Certificates for SSH Authentication* [en línea], 2016. 2016. S.l.: s.n. [Consulta: 8 diciembre 2017]. Disponible en: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3750x_3560x/software/release/15-

2_4_e/configurationguide/b_1524e_consolidated_3750x_3560x_cg/b_1524e_consolidated_3750x_3560x_cg_chapter_01010010.pdf.

XIFRÉ, S.P., 2009. *Antecedentes y perspectivas de estudio en historia de la criptografía* [en línea]. S.l.: Universidad Carlos III Madrid. [Consulta: 24 octubre 2017]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/6173/PFC_Patricia_Xifre_Solana.pdf;jsessionid=0055BA41F57F138FC16A5932BD63F9E5?sequence=1.

ZAYAS, F. y MILAGRO, Y., 2013. La firma electrónica, su recepción legal: Especial referencia a la ausencia legislativa en Cuba. *Revista IUS*, vol. 7, no. 31, pp. 104–120.

BIBLIOGRAFÍA

Aguirre, Jorge Ramió. *Libro Electrónico de Seguridad Informática y Criptografía.* 2006.

Díaz, Juan Carlos Galende. *Criptografía: historia de la escritura cifrada.* 1995.

Gamma, Erich, y otros. *Design Patterns: Elements of Reusable Object-Oriented Software – Design Patterns.* 1997.

Gauchat, Juan Diego. *El gran libro de HTML5, CSS3 y Javascript.* 2012.

Hernández Sampieri, R., Fernández-Collado, C. y Baptista Lucio, P. *Metodología de la Investigación.* 2006.

Larman, Craig. *UML y Patrones: Introducción al análisis y diseño orientado a objetos.* 2004.

López, Manuel José Lucena. *Criptografía y Seguridad en los Computadores.* 2010.

Mogollon, Manuel. *Cryptography and Security Services: Mechanisms and Applications.* 2007.

Triguero, Jesús Javier Ortega, Guerrero, Miguel Ángel López y Crespo, Eugenio C. García del Castillo. *Introducción a la criptografía: historia y actualidad.* 2005.

Menezes, Alfred J., van Oorschot, Paul C. y Vanstone, Scott A. *Handbook of applied cryptography.* 1996.

Pressman, RogerS. *Ingeniería del Software. Un Enfoque Práctico.* 2010.

Wolfgang, Rankl y Wolfgang, Effing. *Smart Card Handbook.* 2004.

Sommerville, Ian. *Ingeniería de Software.* 2011.

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 actualmente, ¿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 sistema, en cuanto a la usabilidad, seguridad, interfaz u otro aspecto que garantice su calidad?

Muchas gracias por su colaboración.

Anexo 2. Historias de usuario

Tabla 11: HU_1 Autenticar usuario *(Elaboración propia)*

Historia de usuario		
Número: HU_1	**Nombre:** Autenticar usuario	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite el acceso a la aplicación por parte de los usuarios.		
Observaciones: Para el acceso a la aplicación es necesaria la entrada de datos por parte del usuario, el cual constará de nombre de usuario y contraseña. Es necesario que los usuarios estén autenticados en el sistema para mantener un control sobre la actividad de los mismos. Además muchas de las funcionalidades del sistema utilizan los datos de los usuarios registrados para realizar sus operaciones.		

Tabla 12: HU_2 Crear usuario *(Elaboración propia)*

Historia de usuario		
Número: HU_2	**Nombre:** Crear usuario	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite insertar un nuevo usuario en la base de datos de la aplicación.		
Observaciones: Para crear un nuevo usuario en la aplicación es necesario la entrada de datos como el nombre de usuario, contraseña, nombre y apellidos de la persona, correo, área, cargo, rol y la activación. El administrador del sistema es el encargado de crear los usuarios del sistema.		

Tabla 13: HU_3 Editar usuario *(Elaboración propia)*

Historia de usuario	
Número: HU_3	**Nombre:** Editar usuario
Programador responsable: Anaili Pérez Piedra	**Iteración asignada:** 1
Prioridad en negocio: Media	
Descripción: Permite editar un usuario de la lista de usuarios registrados en la base de datos de la aplicación.	
Observaciones: En el caso que determinados datos de un usuario 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 14: HU_4 Eliminar usuario *(Elaboración propia)*

Historia de usuario	
Número: HU_4	**Nombre:** Eliminar usuario
Programador responsable: Anaili Pérez Piedra	**Iteración asignada:** 1
Prioridad en negocio: Baja	
Descripción: Permite borrar un usuario de la lista de usuarios registrados en la base de datos de la aplicación.	
Observaciones: En el caso de que un usuario fuera dado de baja de la aplicación por diversas razones, 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. A esta funcionalidad solo tiene acceso el administrador del sistema.	

Tabla 15: HU_5 Buscar usuario *(Elaboración propia)*

Historia de usuario		
Número: HU_5	**Nombre:** Buscar usuario	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Baja		
Descripción: Permite buscar un usuario en la base de datos de la aplicación.		
Observaciones: Debe existir un campo que permita buscar a un determinado usuario en la lista de usuarios registrados. A esta funcionalidad solo tiene acceso el administrador del sistema.		

Tabla 16: HU_6 Listar usuarios *(Elaboración propia)*

Historia de usuario		
Número: HU_6	**Nombre:** Listar usuarios	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite mostrar los usuarios registrados en la aplicación.		
Observaciones: Para cumplir con los requisitos funcionales RF 3 y RF 4 es necesario mostrar los usuarios registrados y las acciones que se pueden acometer sobre ellos, como es el caso de editar y eliminar. Además, debe mostrar la opción de crear un nuevo usuario y buscar usuario, correspondientes a los requisitos funcionales RF 2 y RF 5 respectivamente. A esta funcionalidad solo tiene acceso el administrador del sistema.		

Tabla 17: HU_7 Rechazar documento *(Elaboración propia)*

Historia de usuario		
Número: HU_7	**Nombre:** Rechazar documento	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite rechazar un documento de una petición de firma en la aplicación.		
Observaciones: En el caso de que un usuario no esté de acuerdo en firmar un documento, debe existir la opción de rechazarlo y registrar a través de un formulario 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 18: HU_8 Mostrar documento *(Elaboración propia)*

Historia de usuario		
Número: HU_8	**Nombre:** Mostrar documento	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite mostrar un documento de una petición de firma en la aplicación.		
Observaciones: El sistema debe brindar la opción de mostrar el documento en el listado de peticiones de firma para que el usuario pueda revisarlo antes de firmarlo.		

Tabla 19: 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: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite mostrar el estado de una petición de firma en la aplicación.		
Observaciones: Debe existir un campo que permita visualizar el estado de una petición de firma (pendiente, terminada, cancelada), mostrando la fecha y hora en que firmó cada usuario perteneciente al flujo de firmado.		

Tabla 20: HU_11 Buscar petición de firma *(Elaboración propia)*

Historia de usuario		
Número: HU_11	**Nombre:** Buscar petición de firma	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Media		
Descripción: Permite buscar una petición de firma determinada en la aplicación.		
Observaciones: Debe existir un campo que permita buscar una determinada petición de firma en la lista de peticiones de firma.		

Tabla 21: HU_12 Listar peticiones de firma *(Elaboración propia)*

Historia de usuario		
Número: HU_12	**Nombre:** Listar peticiones pendientes de firma	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite mostrar un listado de las peticiones de firma de un usuario determinado en la aplicación.		
Observaciones: Para cumplir con los requisitos funcionales RF 7, RF 8, RF 10 y RF 13 es necesario mostrar las peticiones pendientes de firma del usuario autenticado 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 22: HU_14 Listar peticiones terminadas *(Elaboración propia)*

Historia de usuario		
Número: HU_14	**Nombre:** Listar peticiones terminadas	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite listar todas las peticiones que un usuario determinado ha firmado o rechazado.		
Observaciones: La aplicación debe mostrar un listado de las peticiones que han sido firmadas o rechazadas por el usuario autenticado en la bandeja de peticiones “Terminadas”. Además, debe mostrar 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 23: HU_15 Descargar documento *(Elaboración propia)*

Historia de usuario		
Número: HU_15	**Nombre:** Descargar documento	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Media		
Descripción: Permite descargar un documento firmado o rechazado por el usuario autenticado.		
Observaciones: La aplicación debe 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 24: HU_16 Listar peticiones creadas *(Elaboración propia)*

Historia de usuario		
Número: HU_16	**Nombre:** Listar peticiones creadas	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite listar las peticiones que el usuario autenticado ha creado en el sistema.		
Observaciones: La aplicación debe mostrar un listado de las peticiones que han sido creadas por el usuario autenticado en la bandeja de “Mis peticiones”. Además, debe 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. El		

redactor y administrador del sistema son los únicos roles que puede acceder a esta bandeja de peticiones.

Tabla 25: 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: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite listar las peticiones en las que el usuario autenticado está definido como usuario firmante dentro del flujo de firma de la petición.		
Observaciones: La aplicación en la bandeja de peticiones “En Espera” debe mostrar 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 26: HU_18 Crear área *(Elaboración propia)*

Historia de usuario		
Número: HU_18	**Nombre:** Crear área	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite insertar una nueva área en la base de datos de la aplicación.		

Observaciones: El sistema debe brindar la posibilidad de insertar un área en el listado de áreas de la aplicación introduciendo el campo nombre del área. El administrador del sistema es el único que tiene permiso para acceder a esta funcionalidad.

Tabla 27: HU_19 Editar área *(Elaboración propia)*

Historia de usuario	
Número: HU_19	**Nombre:** Editar área
Programador responsable: Anaili Pérez Piedra	**Iteración asignada:** 1
Prioridad en negocio: Baja	
Descripción: Permite editar un área de la lista de áreas registradas en la base de datos de la aplicación.	
Observaciones: En el caso que los datos de un área 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 28: HU_20 Eliminar área *(Elaboración propia)*

Historia de usuario	
Número: HU_20	**Nombre:** Eliminar área
Programador responsable: Anaili Pérez Piedra	**Iteración asignada:** 1
Prioridad en negocio: Baja	
Descripción: Permite borrar un área de la lista de áreas registradas en la base de datos de la aplicación.	
Observaciones: Los usuarios que tengan asignados el área eliminada, se le asignarán un valor vacío en ese campo hasta que se decida qué área ocuparán. A esta funcionalidad solo tiene acceso el administrador del sistema.	

Tabla 29: HU_21 Buscar área *(Elaboración propia)*

Historia de usuario		
Número: HU_21	**Nombre:** Buscar área	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Baja		
Descripción: Permite buscar un área determinada en la aplicación.		
Observaciones: Debe existir un campo que permita buscar una determinada área en la lista de áreas de la aplicación. El administrador del sistema es el único que tiene permiso para acceder a esta funcionalidad.		

Tabla 30: HU_22 Listar áreas *(Elaboración propia)*

Historia de usuario		
Número: HU_22	**Nombre:** Listar áreas	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite mostrar las áreas registradas en la aplicación.		
Observaciones: Para cumplir con los requisitos funcionales RF 19 y RF 20 es necesario mostrar las áreas registradas y las acciones que se pueden acometer sobre ellas, como es el caso de editar y eliminar respectivamente. Además debe mostrar la opción de crear una nueva área y buscar un área dentro del listado de áreas correspondiente a los requisitos funcionales RF 18 y RF 21. A esta funcionalidad solo tiene acceso el administrador del sistema.		

Tabla 31: HU_23 Crear cargo *(Elaboración propia)*

Historia de usuario		
Número: HU_23	**Nombre:** Crear cargo	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite insertar un nuevo cargo en la base de datos de la aplicación.		
Observaciones: El sistema debe brindar la posibilidad de insertar un cargo en el listado de cargos de la aplicación introduciendo el campo nombre del cargo. El administrador del sistema es el único que tiene permiso para acceder a esta funcionalidad.		

Tabla 32: HU_24 Editar cargo *(Elaboración propia)*

Historia de usuario		
Número: HU_24	**Nombre:** Editar cargo	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Baja		
Descripción: Permite editar un cargo de la lista de cargos registrados en la base de datos de la aplicación.		
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 tiene permiso para modificar estos datos.		

Tabla 33: HU_25 Eliminar cargo *(Elaboración propia)*

Historia de usuario		
Número: HU_25	**Nombre:** Eliminar cargo	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Baja		
Descripción: Permite borrar un cargo de la lista de cargos registrados en la base de datos de la aplicación.		
Observaciones: Los usuarios que tengan asignados el cargo eliminado, se le asignarán un valor vacío en ese campo hasta que se decida qué cargo ocuparán. A esta funcionalidad solo tiene acceso el administrador del sistema.		

Tabla 34: HU_26 Buscar cargo *(Elaboración propia)*

Historia de usuario		
Número: HU_26	**Nombre:** Buscar cargo	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Baja		
Descripción: Permite buscar un cargo determinado en la aplicación.		
Observaciones: Debe existir un campo que permita buscar un determinado cargo en la lista de cargos de la aplicación. El administrador del sistema es el único que tiene permiso para acceder a esta funcionalidad.		

Tabla 35: HU_27 Listar cargos *(Elaboración propia)*

Historia de usuario		
Número: HU_27	**Nombre:** Listar cargos	
Programador responsable: Anaili Pérez Piedra		**Iteración asignada:** 1
Prioridad en negocio: Alta		
Descripción: Permite mostrar los cargos registrados en la aplicación.		
Observaciones: Para cumplir con los requisitos funcionales RF 24 y RF 25 es necesario mostrar los cargos registrados y las acciones que se pueden acometer sobre ellos, como es el caso de editar y eliminar respectivamente. Además debe mostrar las opciones de crear un nuevo cargo y buscar un cargo determinado correspondientes a los requisitos funcionales RF 23 y RF 26. A esta funcionalidad solo tiene acceso el administrador del sistema.		

Anexo 3. Pruebas funcionales

Tabla 36: 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 37: Caso de prueba del RF13_Firmar digitalmente un documento *(Elaboración propia)*

Caso de prueba 2: SC RF13_Firmar digitalmente un documento					
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, se firma el documento	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 presionar la opción "Firmar" 2. Llenar los campos correspondientes en el formulario y seleccionar la

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	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 38: 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 39: 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,	I	Comprueba si los campos están vacíos, si lo están, muestra un mensaje que solicita llenarlos	

	se 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	

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. Realice 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 sistema informático 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 40: 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 41: 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

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 aplicación tiene una interfaz intuitiva, amigable y fácil de utilizar.	MA ☐	ED ☐	DA ☐	CD ☐	Sí-No ☐	
2	La aplicación será de gran utilidad para llevar a cabo la firma digital de documentos en la universidad.	MA ☐	ED ☐	DA ☐	CD ☐	Sí-No ☐	
3	La aplicación es apropiada para satisfacer todas las necesidades actuales de firma de documentos digitales.	MA ☐	ED ☐	DA ☐	CD ☐	Sí-No ☐	
4	Con la aplicación es posible controlar el flujo de la firma de documentos digitales en la Universidad de las Ciencias Informáticas.	MA ☐	ED ☐	DA ☐	CD ☐	Sí-No ☐	
5	Con la aplicación se puede conocer qué persona firmó un documento digital y en qué momento lo hizo.	MA ☐	ED ☐	DA ☐	CD ☐	Sí-No ☐	
6	La aplicación permite conocer qué persona rechazó una petición y el motivo que tuvo.	MA ☐	ED ☐	DA ☐	CD ☐	Sí-No ☐	

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 42: 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