Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

Universidad de Jaén

Escuela Politécnica Superior
de Jaén

TRABAJO FIN DE GRADO

Servicio Online de Almacenamiento Seguro

Alumno

Juan Ramón Martínez Segura

Tutor

Juan Carlos Cuevas Martínez

(Departamento de Ingeniería de Telecomunicación)

Septiembre, 2023

Universidad de Jaén

Departamento de Informática

Don Juan Carlos Cuevas Martínez , tutor del Trabajo Fin de Grado titulado: '**Servicio Online de Almacenamiento Seguro**', que presenta Don Juan Ramón Martínez Segura, otorga el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2023

El alumno:

Firmado por MARTINEZ SEGURA JUAN RAMON - ***7412** el día 08/09/2023 con un certificado

Juan Ramón Martínez Segura

El tutor:

CUEVAS MARTINEZ JUAN CARLOS - 26038239P

Firmado digitalmente por CUEVAS MARTINEZ JUAN CARLOS - 26038239P
Fecha: 2023.09.08 08:35:19 +02'00'

Juan Carlos Cuevas Martínez

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Tecnologías de la Información
Mención (solo TFG)	Sin mención
Idioma	Español
Tipo	Específico
TFT en equipo	No
Autor/a	Juan Ramón Martínez Segura
Fecha de asignación	02/11/2022
Descripción corta	El presente TFG tiene como objetivo el crear una aplicación web que permita alojar documentos en el servidor de manera que estos estén cifrados y solo accesibles a su propietario, así como verificar su integridad a través de firma digital.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
UNE-ISO 690	Estilo de referencias y citas

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
Estas normas se han utilizado como base o referencia para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como proyecto la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.	

Contenido

UNE-ISO 690 6

1 Especificación del trabajo 11

1.1 Introducción 11
1.2 Objetivos del trabajo 12
1.3 Antecedentes y estado del arte 12
1.3.1 Antecedentes 13
1.3.2 Estado del Arte 13
1.3.2.1 Cifrado de archivos 13
1.3.2.2 Firmado de archivos 14
1.4 Requisitos iniciales 15
1.5 Alcance 15
1.6 Hipótesis y restricciones 15
1.7 Estudio de alternativas y viabilidad 16
1.8 Descripción de la solución propuesta 17
1.9 Tecnologías utilizadas 17
1.10 Metodología de desarrollo de software 19
1.11 Estimación del tamaño y esfuerzo 20
1.12 Planificación temporal 20
1.13 Presupuesto 22
1.14 Normas y referencias 23
1.14.1 Mecanismos de control de calidad 23

2 Diseño inicial 25

2.1 Especificaciones del sistema 25
2.1.1 Especificación de Requisitos 25
2.1.2 Especificación de Casos de Uso 27
2.2 Análisis y diseño del sistema 29
2.2.1 Diseño de Bases de Datos 29
2.2.2 Diseño de la Interfaz 30

3 Desarrollo 34

3.1 Primera Iteración. Investigación Inicial e Instalación de las Herramientas Necesarias 34
3.1.1 Pruebas 35
3.2 Segunda Iteración. Desarrollo del Back-End 35
3.2.1 Tarea 4. Desarrollo de la funcionalidad inicial 35
3.2.1.1 Detalles de implementación 36
3.2.1.2 Pruebas 36
3.2.2 Tarea 5. Implementación de la carga de archivos 37
3.2.2.1 Pruebas 37
3.2.3 Tarea 6. Implementación de la descarga de archivos 38
3.2.3.1 Pruebas 38
3.3 Tercera Iteración. Desarrollo del Front-End 38
3.3.1 Tarea 7. Implementación de la carga/descarga de archivos 38
3.3.1.1 Detalles de implementación 38
3.3.1.2 Pruebas 39
3.3.2 Tarea 8. Implementación del cifrado de archivos 39

3.3.2.1 Detalles de implementación....40
3.3.2.2 Pruebas....40
3.3.3 Tarea 9. Integración de AutoFirma....40
3.3.3.1 Detalles de implementación....41
3.3.3.2 Pruebas....41
3.3.4 Tarea 10. Adecuación de la interfaz a los estándares actuales....41
3.3.4.1 Pruebas....42
3.3.5 Tarea 11. Conformidad con los requisitos de accesibilidad WCAG nivel AA....42
3.3.5.1 Principio 1 - Perceptible....42
3.3.5.2 Principio 2 - Operable....42
3.3.5.3 Principio 3 – Comprensible....43
3.3.5.4 Principio 4 – Robusto....43
3.4 Cuarta Iteración. Elaboración de la Documentación....43
3.5 Pruebas finales....43

4 Modelo de negocio....45

4.1 Oferta de Servicios....45
4.2 Estrategia de monetización....45

5 Conclusiones y trabajos futuros....47

6 Apéndices....49

6.1 Guía original del Trabajo Fin de Título....49
6.2 Instalación y configuración del sistema....50
6.3 Manuales de usuario....51
6.3.1 Manual de despliegue de la aplicación....51
6.3.2 Manual de usuario de la aplicación....51

7 Definiciones y abreviaturas....52

8 Bibliografía....53

Índice de ilustraciones

Ilustración 1.1 - Diagrama de Gantt del desarrollo del proyecto.....21
Ilustración 2.1 - Diagrama Entidad-Relación.....30
Ilustración 2.2 - Pantalla de Inicio de Sesión.....31
Ilustración 2.3 - Pantalla de Registro.....32
Ilustración 2.4 - Pantalla de Almacén.....33

Índice de tablas

Tabla 1.1 - Comparación Frameworks Back-End....16
Tabla 1.2 - Comparación Frameworks Front-End....17
Tabla 1.3 - Iteraciones de la planificación del desarrollo....20
Tabla 1.4 - Costes Programador Junior....22
Tabla 1.5 - Costes de equipo físico....23
Tabla 2.1 - Clasificación de requisitos de la aplicación web....26
Tabla 2.2 - Caso de uso (Autenticación)....27
Tabla 2.3 - Caso de uso (Darse de alta)....27
Tabla 2.4 - Caso de uso (Carga de archivo)....28
Tabla 2.5 - Caso de uso (Descarga de archivo)....28
Tabla 2.6 - Caso de uso (Eliminación de archivo)....29

1 ESPECIFICACIÓN DEL TRABAJO

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*".

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado 7 (Definiciones y abreviaturas).

1.1 Introducción

El Trabajo de Fin de Grado (TFG) se centrará en el desarrollo e implementación de una plataforma en línea que permita a los usuarios almacenar sus documentos electrónicos de manera segura. Se establecerán controles de acceso adecuados para garantizar la confidencialidad de los datos, y se aplicarán mecanismos de cifrado para proteger la información durante su transmisión y almacenamiento. Además contará con un servicio de firma para poder controlar la integridad de dichos ficheros gracias a la aplicación AutoFirma, la cual se describe de manera más detallada en el apartado 1.9 (Tecnologías utilizadas).

A continuación detallo la estructura del presente documento:

- El **primer apartado** trata las **bases de este trabajo**, incluyendo la investigación y revisión del estado del arte en medidas de seguridad informática en aplicaciones web.
- Los **siguientes dos apartados** describen el **desarrollo del proyecto** y tratan, entre otras cosas, tanto el análisis y diseño como la justificación de ciertas decisiones en cada uno de ellos.
- El **cuarto apartado** describe someramente el **modelo de negocio** que se podría utilizar para comercializar la aplicación resultante del proyecto.
- El **apartado número cinco** describe las **conclusiones finales** obtenidas tras el desarrollo del proyecto y algunas posibles mejoras propuestas para él.

- Para finalizar, restan los **apéndices**, las **definiciones y abreviaturas** y la **bibliografía.**

1.2 Objetivos del trabajo

Este proyecto tiene como objetivo principal proporcionar al usuario una aplicación web que le permita almacenar sus ficheros con una capa adicional de seguridad. Esta capa debe asegurar la integridad de dichos ficheros, su autoría y restringir el acceso a todo usuario que no sea el que lo ha almacenado.

Concretamente, los objetivos son los siguientes:

- El desarrollo de la aplicación web completa que solo permita el acceso a los usuarios autenticados.
- Proporcionar los medios para que los archivos que se cargan al servidor estén cifrados adecuadamente, y para añadir una firma digital que permita verificar su integridad.
- Permitir al usuario añadir, descargar y eliminar archivos.
- Que el acceso al servicio sea mediante una interfaz intuitiva y de diseño actual.
- Prestar atención a diversos aspectos de accesibilidad y protección de datos, como la posible gestión de cookies.

1.3 Antecedentes y estado del arte

En la era digital actual, el almacenamiento y la transmisión segura de datos han cobrado una relevancia fundamental debido al crecimiento exponencial de la información digital y a las crecientes preocupaciones sobre la privacidad y la seguridad en línea. Con la proliferación de dispositivos conectados y la adopción generalizada de servicios en la nube, la necesidad de salvaguardar la integridad y la confidencialidad de los archivos se ha vuelto crucial. En este contexto, los sistemas de almacenamiento en línea que ofrecen cifrado y firmado de archivos han emergido como una solución clave para abordar estas preocupaciones.

1.3.1 Antecedentes

En los primeros días de la informática, la seguridad de los datos no era una prioridad, ya que las redes eran limitadas y aisladas. Sin embargo, con la creciente interconexión de sistemas y la evolución de amenazas cibernéticas más sofisticadas, la necesidad de proteger la información se ha convertido en una cuestión apremiante. Los sistemas tradicionales de almacenamiento y transferencia de datos no proporcionaban suficientes salvaguardias contra ataques y accesos no autorizados, lo que ahora supondría un riesgo grave de seguridad.

1.3.2 Estado del Arte

En respuesta a dichas demandas de seguridad, han surgido varios servicios de almacenamiento en línea que incorporan técnicas avanzadas de cifrado y firmado de archivos, como **Tresorit** o **pCloud**. Estos servicios se esfuerzan por garantizar que los datos almacenados en sus plataformas no solo estén protegidos contra el acceso no autorizado, sino también contra posibles modificaciones maliciosas o no deseadas.

1.3.2.1 Cifrado de archivos

En primer lugar existen dos modos de cifrar archivos, el cifrado simétrico y el asimétrico. La principal diferencia entre ellos es que, como su nombre indica, en el cifrado simétrico se utiliza la misma clave para el cifrado que para el descifrado, mientras que en el cifrado asimétrico son distintas. En el caso del almacenamiento de archivos el modo de cifrado más adecuado es el **simétrico**, por lo que no se entrará en detalle en el cifrado asimétrico, más adecuado para el envío de archivos a otros remitentes distintos a uno mismo.

El cifrado simétrico de archivos es una técnica esencial para garantizar la confidencialidad de la información almacenada en línea. Los servicios de almacenamiento seguro utilizan algoritmos criptográficos robustos como por ejemplo el algoritmo “Advanced Encryption Standard” (**AES**) para cifrar los datos antes de ser transferidos o almacenados en servidores remotos. Esto significa que incluso si un tercero no autorizado accede a los archivos, no podrá descifrar su contenido sin la clave de descifrado correspondiente.

El funcionamiento de AES es el siguiente: Hay un número determinado de rondas de encriptación, dependiendo de la variante del algoritmo. AES es uno de los denominados algoritmos de cifrado en bloques debido a que la información no se estructura de manera secuencial, sino en matrices (denominadas bloques). En cada una de estas rondas se le aplican a todos los bloques una serie de procesos matemáticos, de los cuales uno de ellos esta directamente ligado a la clave de cifrado.

Otros algoritmo de cifrado simétrico digno de mención es **Blowfish,** que opera en bloques al igual que AES, pero con bloques de menor tamaño, una posible causa de su poca aceptación y uso en la actualidad.

1.3.2.2 Firmado de archivos

Además del cifrado, el firmado de archivos es otra práctica importante en el contexto de la seguridad de datos. El firmado de archivos implica utilizar una firma digital para autenticar la fuente de un archivo y verificar que no ha sido alterado desde que se creó. De esta manera, se puede detectar cualquier modificación no autorizada en los archivos almacenados y se puede garantizar la integridad de los mismos.

En el ámbito de la administración pública española, la herramienta de firma digital mas ampliamente utilizada es **AutoFirma**, precisamente la herramienta de la que hago uso en este proyecto. Esta aplicación ha sido desarrollada por el Ministerio de Hacienda y Administraciones Públicas y no requiere de ningún pago para su descarga o su uso.

En el ámbito global una de las aplicaciones más utilizadas es **DocuSign eSignature.** Esta aplicación está enfocada a su uso empresarial, y ofrece compatibilidad con la mayoría de dispositivos y tiene integrado un sistema de envío para equipos de trabajo.

En conclusión, el avance tecnológico y la creciente necesidad de seguridad en línea han llevado al desarrollo de servicios de almacenamiento seguro en línea que incorporan cifrado y firmado de archivos. Estas soluciones buscan abordar las preocupaciones sobre la privacidad y la integridad de los datos en un entorno digital cada vez más complejo y conectado.

1.4 Requisitos iniciales

Los requisitos iniciales son los siguientes, en consonancia con los objetivos del proyecto:

- Debe proporcionar un **mecanismo de autenticado** robusto y seguro para que cada usuario pueda acceder únicamente a sus ficheros.
- Debe **cifrar los ficheros** en el servidor con una clave derivada de una contraseña que proporciona el usuario a la hora de subir cada uno de ellos.
- El usuario debe poder **firmar los ficheros** para garantizar su autenticidad y que no han sido alterados.
- El usuario debe poder **descargar tanto el fichero como la firma** asociada y ambos a la vez.
- La **interfaz** de la aplicación debe ser **intuitiva y fácil de utilizar** para el usuario.

1.5 Alcance

Teniendo en cuenta que el objetivo principal del proyecto es la obtención de una aplicación web completa, el alcance de este trabajo es precisamente ese, que la aplicación web sea completamente funcional, dando respuesta a todos los requisitos iniciales propuestos en el apartado anterior (1.4). Quedarían fuera del alcance del proyecto todas las mejoras aplicables a este que no sean permitidas por las restricciones formuladas en el apartado siguiente (1.6), las cuales menciono de manera breve en el apartado 5 (Conclusiones y trabajos futuros).

1.6 Hipótesis y restricciones

El Trabajo de Fin de Grado se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

1.7 Estudio de alternativas y viabilidad

A continuación se efectúa una comparativa entre los frameworks web enfocados al **back-end** que he sopesado utilizar.

Framework	Características
Django	- Basado en Python. - Fácil de aprender y no requiere de mucho código en comparación con otros frameworks. - Fomenta el patrón Modelo-Vista-Controlador (MVC). - Comunidad activa que contribuye al desarrollo, documentación, etc. - Sistema de autenticación incorporado. - Altamente escalable.
Laravel	- Basado en PHP. - Código abierto. - Arquitectura modular para la construcción de aplicaciones escalables. - Fomenta el patrón MVC. - Sistema de autenticación incorporado. - Menos herramientas incorporadas que sus competidores en esta tabla.
Ruby on Rails	- Basado en Ruby. - Muy adaptado a metodologías ágiles. - Altamente escalable. - Comunidad activa que contribuye al desarrollo, documentación, etc. - Desarrollo veloz pero bajo rendimiento. -Fomenta el patrón MVC.

Tabla 1.1 - Comparación Frameworks Back-End

Puesto que estos tres frameworks tienen prestaciones muy similares, he decidido utilizar **Django**, debido a que Python es el lenguaje con el que más experiencia tengo. También ha tenido mucha relevancia en la elección la facilidad de aprendizaje de este framework.

También es necesario efectuar una comparación entre los frameworks dedicados al front-end que me he planteado utilizar.

Framework	Características
React	- Basado en Javascript. - Reusabilidad de componentes. - Alto rendimiento. - Solo provee utilidad para el front-end.
Bootstrap	- Basado en CSS y Javascript. - Código abierto - Reusabilidad de componentes. - Ligero y fácil de integrar. - Alta compatibilidad con todos los navegadores usuales.
Angular	- Basado en Typescript. - Reusabilidad de componentes. - Amplia comunidad para el aprendizaje y apoyo. - Curva de dificultad elevada.

Tabla 1.2 - Comparación Frameworks Front-End

Tras el estudio de estas alternativas me decanté por utilizar **Bootstrap** sobre todo debido a su ligereza y facilidad de uso, ya que la aplicación del proyecto no demanda grandes detalles en cuanto a la interfaz de usuario.

1.8 Descripción de la solución propuesta

La solución propuesta para dar respuesta a este proyecto es una aplicación web cuyo front-end será desarrollado mediante el framework Bootstrap y cuyo back-end será desarrollado mediante Django. En cuanto a los requisitos funcionales más concretos, el cifrado de archivos se efectuará según el algoritmo **AES** con clave de **256** bits y en formato Cipher-Block Chaining (**CBC**) y la firma se efectuará utilizando la aplicación AutoFirma y concretamente se utilizará el algoritmo de firma “XML Advanced Electronic Signatures (**XAdES**). En cuanto al sistema de autenticación se utilizará el que viene incorporado con Django, puesto que no considero que se necesite ninguna funcionalidad adicional.

1.9 Tecnologías utilizadas

A continuación se enumeran las tecnologías utilizadas en el desarrollo del proyecto junto con una breve descripción de ellas y del por qué de su elección.

- HyperText Markup Language (**HTML**) es el lenguaje básico dentro de las páginas web, por lo tanto se ha utilizado para determinar el contenido de la aplicación web. Es el lenguaje con el que en este proyecto se construyen las plantillas que forman la parte visible de la aplicación.
- Cascading Style Sheets (**CSS**) es también básico a la hora de determinar la apariencia de las páginas web, lo que hace su uso prácticamente obligatorio. Este lenguaje no solo permite cambiar el aspecto y el color de los elementos HTML, sino también su disposición en la ventana.
- **Django** es un framework web enfocado al back-end de alto nivel basado en el lenguaje de programación Python. Este framework trae utilidades que aligeran la carga del desarrollo del proyecto y aconseja y fomenta el uso del patrón **Modelo-Vista-Controlador.** En cuanto a la seguridad, punto crucial de este proyecto, Django proporciona mecanismos para evitar ciertos problemas usuales de seguridad, como el Cross-site scripting (XSS), la inyección SQL, además del uso de tokens para evitar la falsificación de petición en sitios cruzados (**CSRF**).
- **Bootstrap** es un framework web enfocado al front-end de alto nivel basado en los lenguajes de programación HTML, CSS y Javascript. Sus principales características son la ligereza, la compatibilidad con todos los navegadores debido a su amplio uso y lo sencillo que es integrarlo y usarlo en un proyecto. Este framework también permite dotar a la aplicación web de responsividad, permitiendo que se adapte a diferentes tamaños de ventana y pantalla, con lo que se consigue una experiencia de usuario más uniforme.
- Para el proceso de firmado de ficheros he hecho uso de la aplicación **AutoFirma**. Esta Aplicación de firma electrónica desarrollada por el Ministerio de Hacienda y Administraciones Públicas de España tiene como funciones principales firmar ficheros y verificar las firmas de éstos. El motivo de su elección es que consta de un manual de integración muy detallado y fácil de entender que me permitió integrarlo en mi aplicación web de manera sencilla y rápida. Otra característica muy relevante de esta herramienta es que debido a su uso generalizado la administración de este

país, resulta muy fiable y recibe actualizaciones de manera muy regular. Esta herramienta también consta de multitud de opciones y formatos de firma como **CAdES, XAdES y PAdES**.

- **Javascript** es un lenguaje de alto nivel, dinámico e interpretado. Se utiliza principalmente en la parte asociada al navegador de las aplicaciones web (Front-End) aunque también tiene ciertos usos en documentos PDFs y widgets de escritorio. En este caso se ha utilizado para la integración de la ya mencionada aplicación AutoFirma en la aplicación web y para ciertas funcionalidades del front-end, como el cifrado y descifrado de los archivos y la aparición de mensajes de alerta en el navegador, entre otros.

1.10 Metodología de desarrollo de software

Para llevar a cabo este proyecto he decidido utilizar la metodología de software denominada **desarrollo incremental**.

Esta metodología tradicional se caracteriza por ir construyendo el producto de manera progresiva. Esto significa que cada funcionalidad se va añadiendo en una nueva etapa incremental de manera que toda la funcionalidad anterior ya estará completamente operativa. Esta manera de trabajar ofrece ciertas ventajas, como que el usuario pueda empezar a utilizar el producto aunque aún falten ciertas funcionalidades o la capacidad de ver resultados de forma rápida, ya que al terminar la primera etapa ya obtienes un producto funcional. Otra ventaja que posee esta metodología es que puede ser menos lineal que las demás metodologías tradicionales, esto se debe a que, en nuestro caso, ya que los requisitos iniciales son bastante disjuntos, se puede comenzar una nueva etapa sin necesidad de haber terminado la anterior. A pesar de ser una ventaja interesante, en este proyecto no se prevé la necesidad de hacerlo de este modo.

Debido a la ya mencionada separación entre los requisitos iniciales me ha parecido interesante utilizar esta metodología e ir diseñando, desarrollando y probando de manera incremental y secuencial cada una de las funcionalidades del proyecto. Por este motivo y por tener la capacidad de mostrar cada una de las funcionalidades tras su implementación he elegido esta metodología de desarrollo.

1.11 Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFG, no existen restricciones de tipo económico, sino de tipo temporal. Ya que la asignatura consta de 12 créditos ECTS y cada uno de éstos créditos equivale a 25 horas de trabajo aproximadamente, tenemos que la duración aproximada del proyecto es de **300 horas**.

En cuanto al esfuerzo, se dispone de tan un solo efectivo (la persona autora del trabajo).

1.12 Planificación temporal

El proceso de desarrollo de la aplicación consta de las tres iteraciones descritas en la tabla 1.3, donde se detallan sus correspondientes fechas de inicio y fin. Para mayor detalle se encuentra la ilustración 1.1, donde se pueden ver con más detalle las tareas y subtareas correspondientes a cada iteración y las semanas que les corresponden.

Iteración	Fecha de Inicio	Fecha de Finalización	Número de tareas	Número de subtareas
Iteración 1. Investigación inicial e instalación de las herramientas necesarias.	14/02/2023	27/02/2023	3	0
Iteración 2. Desarrollo del Back-End	27/02/2023	07/04/2023	3	3
Iteración 3. Desarrollo del Front-End	10/04/2023	16/06/2023	5	0
Iteración 4. Elaboración de la documentación	19/06/2023	07/07/2023	1	0

Tabla 1.3 - Iteraciones de la planificación del desarrollo

Nombre de la tarea	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9	Semana 10	Semana 11	Semana 12	Semana 13	Semana 14	Semana 15	Semana 16	Semana 17	Semana 18	Semana 19	Semana 20	Semana 21	Semana 22
Desarrollo TFG																						
Desarrollo de la Aplicación Web																						
Iteración 1. Investigación inicial e instalación de las herramientas necesarias																						
Tarea 1. Investigación Frameworks Back-End																						
Tarea 2. Investigación Frameworks Front-End																						
Tarea 3. Instalación de Python, Django, Bootstrap y puesta a punto del proyecto																						
Iteración 2. Desarrollo del Back-End																						
Tarea 4. Desarrollo de la funcionalidad inicial																						
Tarea 4.1 Puesta a punto del autenticado																						
Tarea 4.2 Diseño e implementación del modelo																						
Tarea 4.3 Implementación del esqueleto de la aplicación																						
Tarea 5. Implementación de la carga de archivos.																						
Tarea 6. Implementación de la descarga de archivos.																						
Iteración 3. Desarrollo del Front-End																						
Tarea 7. Implementación de la carga/descarga de archivos																						
Tarea 8. Implementación del cifrado de archivos																						
Tarea 9. Integración de AutoFirma																						
Tarea 10. Adecuación de la interfaz a los estándares actuales																						
Tarea 11. Conformidad con los requisitos de accesibilidad WCAG nivel AA																						
Iteración 4. Elaboración de la Documentación																						
Tarea 12. Elaboración de la documentación																						

Aplicación
Iteración
Tarea
Subtarea
Cada celda simboliza una semana de trabajo

Ilustración 1.1 - Diagrama de Gantt del desarrollo del proyecto

Utilizando este diagrama podemos ver con detalle cada una de las tareas, su duración y semanas correspondientes. También podemos ver que su duración total se prevé que sea de **22 semanas**, expandiéndose a lo largo de 5 meses. El trabajo aproximado que se realiza a la semana es de **3 horas diarias de lunes a viernes**, lo que resulta en las 300 horas de duración aproximada que debe tener el trabajo.

1.13 Presupuesto

En este apartado se realiza un estudio del coste del proyecto. Partimos de que el equipo de desarrollo esta integrado por una única persona, y que la duración estimada debe estar cerca de las 300 horas, las correspondientes a los 12 créditos ECTS de la asignatura.

Los gastos del proyecto se pueden dividir en costes de hardware, costes indirectos y costes de personal. No aparecen costes relacionados con licencias software ni similares puesto que todas las herramientas utilizadas son gratuitas.

El cálculo del **coste invertido en personal** se ha efectuado asumiendo que la única persona que pertenece al equipo de desarrollo es un **Programador Junior** (Área 3, Grupo E y Nivel 1) según el convenio (Disposición 3156 del BOE núm. 57, 2018). Por lo tanto el coste correspondiente sería como se detalla a continuación en la tabla 1.4.

Salario base	Plus convenio	Salario total	Cuantía Mensual (14 pagas)	Tiempo de Contrato	Cuantía total
13827.66€	973€	14800.66€	1057.19€	5 meses	5.285,95 €

Tabla 1.4 - Costes Programador Junior

Respecto a los **costes de equipo o hardware**, los detallo en la tabla 1.5 que se encuentra a continuación. El único hardware necesario para el proyecto es un ordenador personal así que se toma el utilizado para el desarrollo como referencia para calcular el presupuesto.

Recurso	Uso	Precio	Tiempo de uso	Amortización mensual
HP 15S-fq2092ns Intel Core i7-1165G7/8GB/512GB SSD/15.6"	Utilizado para desarrollo	698,86€	5 meses	139,77 € / mes

Tabla 1.5 - Costes de equipo físico

Para finalizar, hay que tener en cuenta los **costes indirectos** asociados al proyecto. Se ha estimado que estos costes supondrán aproximadamente un 10% del resto del presupuesto del proyecto, con lo que nos quedan **598,48€** destinados a los costes indirectos

1.14 Normas y referencias

A continuación, se presentan las normas, reglamentos y referencias de cualquier tipo que han sido de aplicación en la elaboración del trabajo y/o en la puesta en práctica del mismo.

1.14.1 Mecanismos de control de calidad

El aseguramiento de la calidad en la redacción del trabajo implica la verificación de la completitud (falta de omisiones), la integridad de la documentación, así como una redacción clara, concisa y entendible por todos los participantes e interesados en dicho trabajo.

Al estar el trabajo desarrollado por una sola persona y verificado por un/a tutor/a, no es necesario llevar un documento de control de edición y revisión de la documentación. De esta forma, se han utilizado mecanismos básicos para la verificación de la integridad y completitud, que incluyen un control de versiones a nivel de documento y un control sencillo de la trazabilidad de especificaciones y requisitos.

2 DISEÑO INICIAL

2.1 Especificaciones del sistema

Este apartado se ha dividido en dos secciones con el objetivo de abordar los casos de uso y la especificación de requisitos de forma separada.

2.1.1 Especificación de Requisitos

A continuación, en la tabla 2.1 se detallan los requisitos de la aplicación web del proyecto, estrechamente ligados a los requisitos iniciales que se tratan en la sección 1.4.

Requisito	Clasificación	Tipo
La aplicación web debe tener una interfaz intuitiva y sencilla.	No funcional	Usabilidad
La interfaz de la aplicación web debe ser estéticamente agradable y moderna	No funcional	Usabilidad
Debe proporcionar un mecanismo de autenticado robusto y seguro	Funcional	Perspectiva funcional
La aplicación debe permitir al usuario cargar nuevos ficheros	Funcional	Perspectiva funcional
La aplicación debe mostrar todos los ficheros que el usuario tiene almacenados	Funcional	Perspectiva de comportamiento
Debe cifrar los ficheros en el navegador web mediante una clave derivada de una contraseña que proporcione el usuario	Funcional	Perspectiva funcional
El usuario debe poder firmar los ficheros	Funcional	Perspectiva funcional
El usuario debe poder descargar tanto los ficheros como sus firmas asociadas	Funcional	Perspectiva funcional
El usuario debe poder eliminar ficheros de la nube	Funcional	Perspectiva funcional
La interfaz de la aplicación debe cumplir los requisitos de accesibilidad WCAG nivel AA	No funcional	Usabilidad

Tabla 2.1 - Clasificación de requisitos de la aplicación web

2.1.2 Especificación de Casos de Uso

En las siguientes tablas se encuentran descritos los casos de uso contemplados para la aplicación web del proyecto.

Atributo	Contenido
Nombre	Autenticación
Descripción	El usuario se identifica frente al sistema
Actores involucrados	Usuario sin identificar, Sistema
Precondiciones	-
Postcondiciones	El sistema reconoce al usuario y le muestra sus ficheros cargados
Flujo básico	(1) Usuario accede a la aplicación web. (2) Usuario decide si registrarse o darse de alta. (3) En ambos casos Usuario rellena el formulario. a. El sistema reconoce al Usuario. b. El sistema no reconoce al Usuario. (4) El sistema muestra el almacén de archivos del Usuario.

Tabla 2.2 - Caso de uso (Autenticación)

Atributo	Contenido
Nombre	Darse de alta
Descripción	El usuario se da de alta en el sistema
Actores involucrados	Usuario sin identificar, Sistema
Precondiciones	-
Postcondiciones	El sistema reconoce al usuario y le muestra su almacén de archivos (Estará vacío pues se acaba de dar de alta)
Flujo básico	(1) Usuario accede a la aplicación web. (2) Usuario accede al formulario de registro. (2) Usuario rellena el formulario. a. El sistema valida los datos introducidos. b. El sistema avisa al Usuario para que corrija algún campo erróneo. (3) El sistema muestra el almacén de archivos del Usuario.

Tabla 2.3 - Caso de uso (Darse de alta)

Atributo	Contenido
Nombre	Carga de un archivo
Descripción	El usuario almacena un archivo nuevo en la aplicación
Actores involucrados	Usuario identificado, Sistema
Precondiciones	El Usuario debe haberse identificado.
Postcondiciones	Se añade el archivo al modelo de datos. El sistema muestra el almacén de archivos del Usuario, con el nuevo archivo en él.
Flujo básico	(1) Usuario abre el formulario de carga de archivos (2) Usuario rellena el formulario a. El sistema valida los datos introducidos. b. El sistema avisa al Usuario para que corrija algún campo erróneo. (3) El sistema lanza AutoFirma en la maquina del Usuario (4) El Usuario elige el certificado a utilizar. (5) El archivo y su firma correspondiente se añaden al almacén de archivos.

Tabla 2.4 - Caso de uso (Carga de archivo)

Atributo	Contenido
Nombre	Descarga de un archivo
Descripción	El usuario descarga un archivo del almacén a su equipo.
Actores involucrados	Usuario identificado, Sistema
Precondiciones	El Usuario debe haberse identificado.
Postcondiciones	El archivo se descarga a través del navegador en el equipo del Usuario.
Flujo básico	(1) Usuario presiona en el botón correspondiente a la descarga del archivo deseado. (2) El sistema pide al Usuario la contraseña correspondiente para efectuar el descifrado del archivo. (3) El Usuario introduce la contraseña. (4) El navegador web descarga el archivo deseado del almacén.

Tabla 2.5 - Caso de uso (Descarga de archivo)

De manera análoga a este caso de uso se efectúan la descarga de la firma correspondiente al archivo y de ambos a la vez, solo cambiando el botón que se pulsa.

Atributo	Contenido
Nombre	Eliminación de un archivo
Descripción	El usuario elimina un archivo del almacén.
Actores involucrados	Usuario identificado, Sistema
Precondiciones	El Usuario debe haberse identificado.
Postcondiciones	El archivo desaparece del modelo de datos, y por lo tanto del almacén que se le muestra al Usuario.
Flujo básico	(1) Usuario presiona en el botón correspondiente a la eliminación del archivo deseado. (2) El Sistema pide una confirmación al Usuario. a. Usuario confirma la eliminación. (3) El sistema elimina el archivo del modelo de datos. b. Usuario rectifica la eliminación. (3) No sucede nada.

Tabla 2.6 - Caso de uso (Eliminación de archivo)

2.2 Análisis y diseño del sistema

Una vez que contamos con ambas especificaciones de manera detallada, podemos considerar que el análisis ha sido concluido, con lo que podemos proceder al diseño de la aplicación.

El diseño de la aplicación está fuertemente condicionado por el framework que he utilizado para el back-end de ésta: **Django**. Este framework te anima a usar el patrón **MVC**, y concretamente a crear una clase de Python (el lenguaje en el que está basado) para cada vista de la aplicación. Además, al tener claro que se va a utilizar el sistema de autenticación que viene integrado con Django, es posible aprovechar esa parte del diseño del modelo, y construir directamente sobre ella.

2.2.1 Diseño de Bases de Datos

Como se menciona anteriormente, para el modelo de datos de la aplicación se ha utilizado el que trae Django incorporado: la tabla llamada **User** que aparece en el diagrama posterior (Ilustración 2.1 - Diagrama Entidad-Relación). Para simplificar el diagrama se han omitido los atributos que no he utilizado.

Para modelizar los ficheros se ha utilizado una relación de uno a muchos desde dicha tabla a la tabla denominada **UserFile**, puesto que, naturalmente, un usuario puede almacenar diversos archivos. Esta tabla tiene una referencia a la tabla User, el archivo y su firma asociada y diversos metadatos.

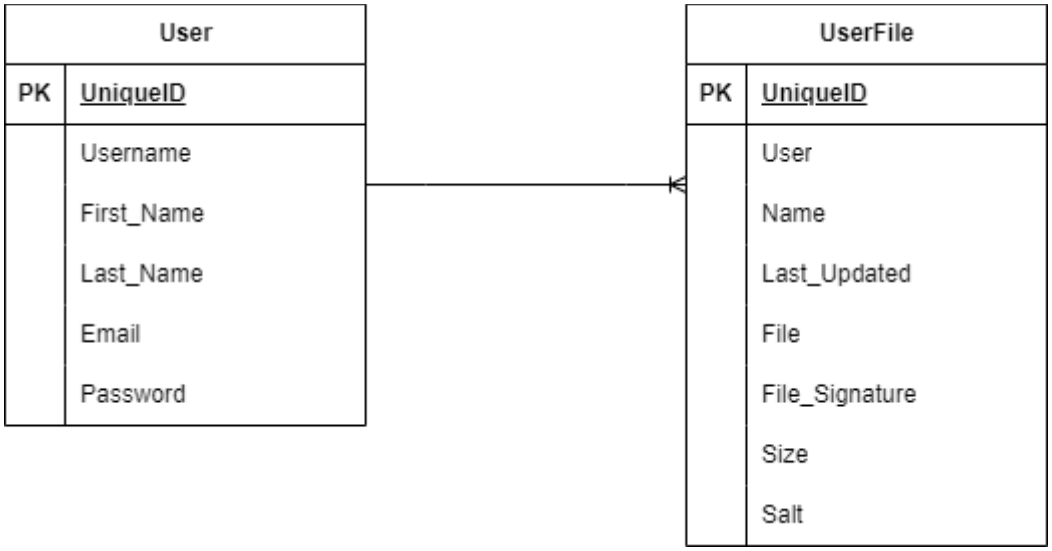

Ilustración 2.1 - Diagrama Entidad-Relación

2.2.2 Diseño de la Interfaz

Para mostrar el diseño de la interfaz a continuación muestro los storyboards de la aplicaciones y detallo como se relacionan las distintas pantallas entre sí y como moverse de una a otra.

Ilustración 2.2 - Pantalla de Inicio de Sesión

Esta es la primera pantalla que mostrará la aplicación cuando accedes a ella. Al pulsar sobre el hipervínculo “Registrarse” el sistema te envía a la pantalla asociada a la ilustración 2.3.

Al rellenar el formulario de inicio de sesión con datos correctos y pulsar sobre el botón “Iniciar sesión” el sistema te llevará a la pantalla correspondiente a la ilustración 2.4.

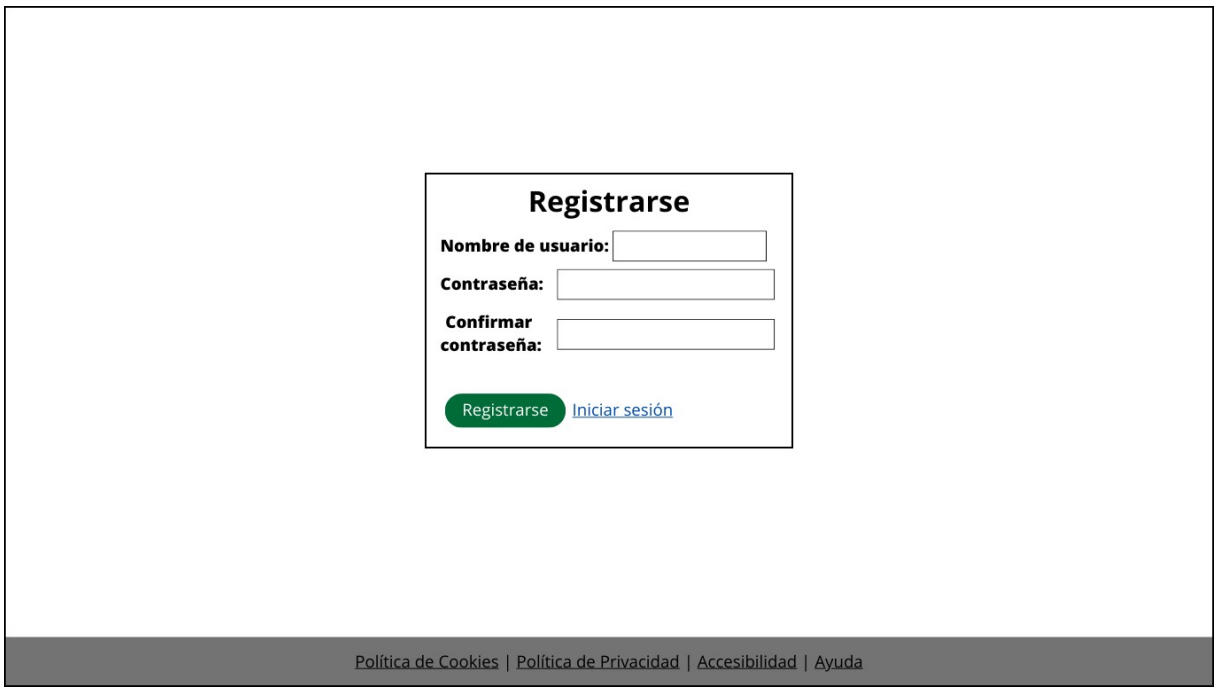

Ilustración 2.3 - Pantalla de Registro

Esta pantalla te devuelve a la anterior pulsando sobre el hipervínculo “Iniciar sesión” (Ilustración 2.2). De manera análoga a la anterior, si se rellena el formulario y se presiona sobre “Registrarse” aparecerá la pantalla de Almacén (Ilustración 2.4).

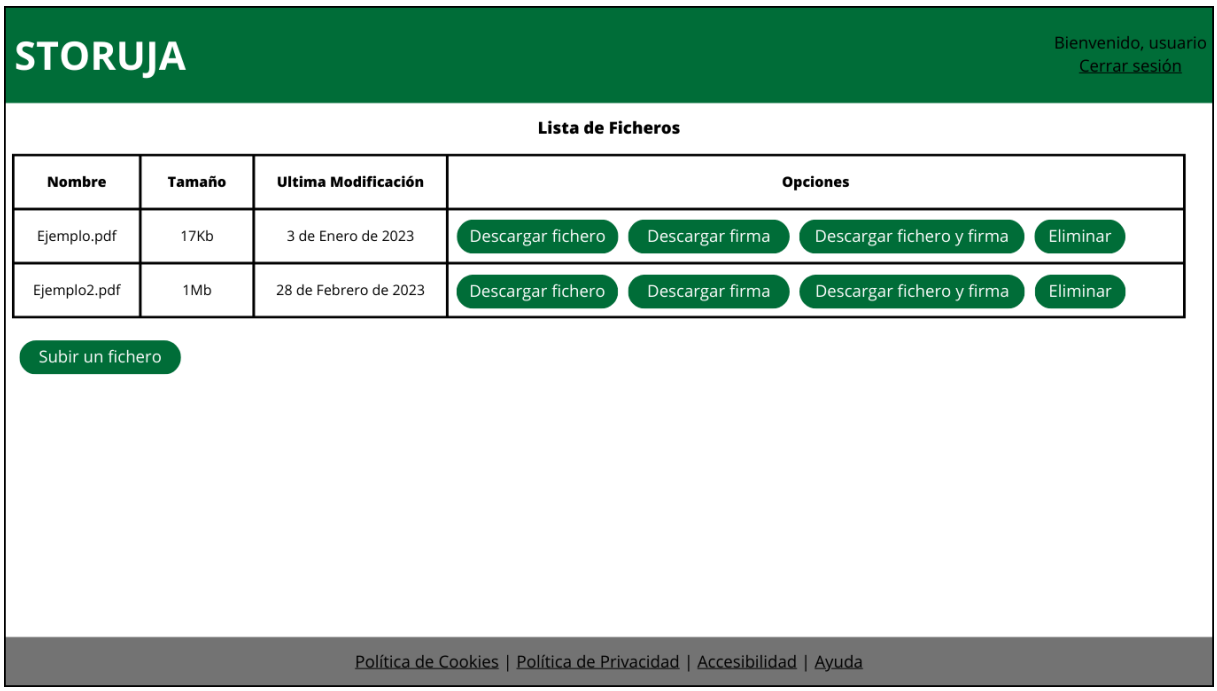

Ilustración 2.4 - Pantalla de Almacén

En esta pantalla aparecerán todos los ficheros del usuario y las opciones asociadas a cada uno de ellos. Para volver a la pantalla de Inicio de sesión (Ilustración 2.2) el usuario debe pulsar sobre “Cerrar sesión”.

3 DESARROLLO

3.1 Primera Iteración. Investigación Inicial e Instalación de las Herramientas Necesarias

La primera Iteración del proyecto consistió en la investigación y elección de los frameworks a utilizar y en su puesta a punto para poder comenzar el desarrollo propiamente dicho en la siguiente iteración. Debido a la brevedad de las tareas de esta iteración no se ha llevado a cabo la división en subsecciones que aparece en las demás iteraciones de la documentación.

El primer paso natural de este proyecto es llevar a cabo una investigación inicial de los dos ámbitos (Front y Back End) que compondrán la aplicación para conocer cuales son los frameworks más utilizados actualmente en el ámbito empresarial. Una vez hecho esto se buscó información sobre cada uno de ellos en profundidad para saber cual de ellos era el idóneo para el proyecto. En el apartado Estudio de alternativas y viabilidad se encuentra la disertación correspondiente a los frameworks que parecían mas adecuados para el proyecto y el porqué de la elección de Django y Bootstrap.

En cuanto a la instalación de estos frameworks se refiere, hay que hacer lo siguiente, teniendo en cuenta que en este caso se han instalado en una máquina **Windows**:

1. Instalación de **Python**: Para instalar Python en Windows solo hay que acceder a la Microsoft Store mediante el menú Inicio, buscar allí “Python” y elegir la versión que se desee, en mí caso la versión **3.9**. Esta instalación incluye el gestor de paquetes para Python **PIP**, que se utilizará posteriormente para la instalación de Django y diversas librerías adicionales que he utilizado.

2. Instalación de **Django**: Para instalar Django basta con haber concluido el paso anterior, y en la terminal de windows escribir “*pip install Django*“. Esto instalará la última version de Django disponible.

3. Instalación de **Bootstrap**: Existen diversas formas de utilizar Bootstrap, puedes descargar todo el código para el desarrollo o puedes utilizarlo mediante un Content Delivery Network (**CDN**). Se ha optado por esta

segunda opción puesto que de este modo si el usuario ya ha descargado los datos correspondientes a Bootstrap de ese mismo CDN, cosa muy usual ya que es ampliamente utilizado, tendrá dichos datos en la caché de su navegador y por lo tanto el rendimiento se verá ampliamente mejorado.

Para utilizar este CDN basta con añadir dos líneas en la etiqueta “head” del archivo HTML utilizado, que importarán en tiempo de ejecución los archivos necesarios.

3.1.1 Pruebas

Antes de continuar con la siguiente iteración se ha comprobado que todo estaba correctamente instalado.

En el caso de Python basta con introducir en la terminal de windows “*python –version*”.

Para Django se comprueba de manera análoga escribiendo en la terminal “*django-admin –version*”.

Para comprobar que la instalación de Bootstrap ha sido satisfactoria basta con abrir el archivo HTML en el navegador y ver que ha cambiado visualmente frente al estado previo a la instalación.

3.2 Segunda Iteración. Desarrollo del Back-End

Las tareas que componen esta iteración se han tratado como iteraciones per se, es decir, para cada una de ellas se han hecho las pruebas pertinentes y no se ha comenzado una tarea hasta terminar la anterior, sin embargo se ha tomado la decisión de denominarlas como tareas para tener un nivel superior de organización (Front-End, Back-End, Documentación) y que quede más clara la estructura del proyecto.

3.2.1 Tarea 4. Desarrollo de la funcionalidad inicial

Esta tarea engloba todo el desarrollo necesario para, a partir de una instalación limpia de Django, tener un esqueleto para la aplicación al que empezar a añadirle la funcionalidad requerida.

En este caso este desarrollo consiste en preparar el **sistema de autenticación**, el **modelo**, es decir, la base de datos y el **esqueleto de la**

aplicación. Estas tres tareas están muy relacionadas entre sí, puesto que para implementar completamente el sistema de autenticación es necesario tener cierta parte del modelo de datos implementada y las vistas de inicio de sesión y de registro también (Parte del esqueleto de la aplicación). Sin embargo, al estar este sistema ya incorporado en el framework, la parte del modelo de datos relativa a la autenticación también está incorporada y lista para su uso.

3.2.1.1 Detalles de implementación

El primer paso del desarrollo es crear un proyecto de Django, para lo que se utiliza la orden de terminal "*django-admin startproject nombreProyecto*".

Posteriormente hay que crear las vistas correspondientes a las pantallas de inicio de sesión y registro. Para esto debemos crear las correspondientes clases de Python, que heredaran de la clase de Django "*View*" y los formularios de registro y login, los cuales vienen ya incorporados, por lo que solo es necesario importarlos.

Por último, es necesario crear las plantillas HTML asociadas a dichas vistas. Estas se crearán de manera muy esquemática, lo justo para permitir probar la funcionalidad, ya que la siguiente iteración será la que se centrará en que las vistas sean estéticamente adecuadas. En esta etapa también se crea la vista Almacén (Ilustración 2.4 - Pantalla de Almacén) con su correspondiente plantilla (En la que en este momento del desarrollo solo muestra el nombre del usuario), para poder saber si se ha efectuado el inicio de sesión con éxito.

A pesar de no ser necesario para el sistema de autenticación, en esta tarea se definió el modelo de datos completo de la aplicación. Para ello es necesario crear una subclase de la clase "*Model*" de Django, con los atributos necesarios. En el apartado Diseño de Bases de Datos se puede observar con detalle el diseño de esta tabla.

3.2.1.2 Pruebas

Para probar este sistema, se creó un súper usuario mediante línea de comandos introduciendo en la terminal de Windows: "*python manage.py createsuperuser*" con la terminal ubicada en la raíz del proyecto. Después de rellenar el formulario que aparece ya disponemos del súper usuario. Se ha creado un súper usuario y no un usuario corriente para poder acceder a la interfaz de administración

de Django, la cual nos permite visualizar el contenido de las tablas de las bases de datos entre otras cosas, lo cual es muy útil en los procesos de depuración y pruebas.

Una vez creado solo resta ver que el inicio de sesión se efectúa correctamente, introduciendo las credenciales en la vista correspondiente y viendo que te redirige a la vista Almacén, donde debe aparecer el nombre de usuario correspondiente a las credenciales introducidas.

Para comprobar que el registro funciona, se rellenó el formulario correspondiente y mediante la interfaz de administración antes mencionada se comprobó que en la tabla de usuarios todos los datos aparecían correctamente y en su atributo correspondiente.

3.2.2 Tarea 5. Implementación de la carga de archivos

Para implementar la carga de archivos se añadió el formulario correspondiente a la funcionalidad, en este caso, un selector de ficheros para el fichero a cargar, en HTML <input type='file'> y un cuadro de texto para la contraseña correspondiente al fichero. Este formulario se asoció a la tabla "*UserFile*" del modelo de datos creado en la tarea anterior. Los demás atributos de dicha tabla se rellenan sin necesidad de añadirlos al formulario, puesto que la mayoría son metadatos. En este paso del proceso de desarrollo el atributo correspondiente a la sal para la función de derivación de claves no se rellena, puesto que esa funcionalidad aún no está presente.

Los archivos se almacenan directamente en un campo de la base de datos, llamado "*File*".

3.2.2.1 Pruebas

Para probar la carga de archivos, es necesario autenticarse como usuario y cargar varios archivos. Utilizando la herramienta de administración mencionada anteriormente es posible observar que todos los atributos de la base de datos eran correctos.

Para comprobar que la gestión de los archivos de cada usuario era correcta fue necesario autenticarse con otro usuario distinto, ver que no podía visualizar los archivos del otro usuario y subir varios archivos más, comprobando que el funcionamiento era el esperado.

3.2.3 Tarea 6. Implementación de la descarga de archivos

Esta tarea consistió en añadir código en la vista correspondiente al almacén para que al pulsar los botones se sirvieran los archivos correspondientes, tanto el fichero en sí como su firma asociada.

3.2.3.1 Pruebas

Puesto que posteriormente los archivos se deben descifrar en el navegador, no se pueden enviar directamente del servidor, lo cual es lo usual. Por lo tanto para probar esta parte de la funcionalidad basta con cerciorarse de que el front-end recibe los datos correctamente, ya que esa es la parte que se encarga de la creación del archivo.

3.3 Tercera Iteración. Desarrollo del Front-End

De manera análoga a la iteración anterior, las tareas pertenecientes a ésta también son tratadas como iteraciones propiamente dichas. En esta iteración es necesario tener claro el flujo de operaciones en la carga y descarga de los archivos. En el caso de la carga el navegador firma el archivo, lo cifra y lo manda al servidor. Y en la descarga el navegador manda el archivo cifrado y la sal, el navegador lo descifra y crea el fichero en el equipo.

3.3.1 Tarea 7. Implementación de la carga/descarga de archivos

Esta tarea consistió en proporcionar las herramientas necesarias para poder efectuar la tarea posterior: el cifrado y descifrado de los archivos. Para ello fue necesario añadir código Javascript que pidiera la contraseña al usuario en la descarga, por ejemplo. Para la correcta organización del código fue necesario configurar los archivos estáticos en el proyecto.

3.3.1.1 Detalles de implementación

Como el front-end debe encargarse del descifrado del archivo, la descarga se implementó de manera que el front-end recibe el contenido del archivo y fabrica el fichero para la descarga. Para ello se crea una clase "*Blob*" de Javascript con el

contenido que se recibe del servidor y con el tipo "*application/text*". Posteriormente se crea una URL a este blob utilizando la función "*URL.createObjectURL()*" y se crea un elemento HTML temporal al que se le añaden dos atributos mediante la funcion ".*setAttribute()*": Dicha URL y el atributo "*download*" con el nombre del fichero. Este elemento se pulsa mediante la función ".*click()*", lo que origina el fichero en descargas y se elimina.

3.3.1.2 Pruebas

Para probar esta funcionalidad se utilizaron los ficheros de prueba de las dos tareas anteriores. Se descargaron esos ficheros comprobando que se descargaban adecuadamente. Puesto que aún la funcionalidad de firma no estaba operativa se simuló con archivos de prueba.

3.3.2 Tarea 8. Implementación del cifrado de archivos

El cifrado del archivo se realiza en el cliente, de manera que el servidor no conozca la clave, ésta no viaja por la red y el archivo en sí viaja ya cifrado. Para el cifrado de datos se le pide al usuario una contraseña de la cual se deriva la clave de cifrado. Este proceso se realiza mediante el algoritmo Password-Based Key Derivation Function 2 (**PBKDF2**) implementado en este proyecto mediante la Aplication Programming Interface (API) de Javascript llamada Web Crypto, concretamente en su interfaz "SubtleCrypto".

Esta función, a parte de recibir la contraseña también recibe una secuencia de bytes aleatorios denominados **sal** cuya función se especifica en el RFC en el que se introduce dicha función: "Una sal en la criptografía basada en contraseñas ha servido tradicionalmente al propósito de producir un gran conjunto de claves correspondientes a una determinada contraseña, y se elige una de ellas aleatoriamente determinada por la sal." **[1]**

Por lo tanto dicha sal debe ser guardada en la base de datos puesto que es imprescindible para volver a generar la misma clave en el cifrado y en el descifrado.

La clave de cifrado que se obtiene de este algoritmo es de **256 bits**, que posteriormente se le pasa como argumento al algoritmo de cifrado **AES**, implementado mediante la misma API anterior. En la especificación del algoritmo de cifrado AES se detalla: "Los cifradores en bloque AES-128, AES-192 y AES-256 difieren en tres aspectos. La longitud de la clave, el numero de rondas de cifrado y la

especificación de la recursión dentro de KEYEXPANSION().” **[2]**. Podemos concluir por lo tanto que a mayor longitud de clave mayor seguridad criptográfica, lo que motiva la elección de **AES-256**.

3.3.2.1 Detalles de implementación

Las funciones criptográficas utilizadas son las siguientes:

- **“*window.crypto.subtle.deriveKey()*”**: A esta función recibe la sal como argumento y como contiene distintas implementaciones de derivación de claves hay que indicarle que la deseada es "*PBKDF2*" en otro de sus argumentos.
- “**window.crypto.subtle.encrypt()**”: De manera similar a la función anterior, contiene distintos algoritmos de cifrado, por lo que hay que explicitar en uno de sus argumentos “AES-CBC”. También hay que proporcionarle el vector de inicialización, la clave obtenida en la función anterior y los datos a cifrar.
- “**window.crypto.subtle.decrypt()**”: De manera análoga a la anterior, recibe los mismos parámetros salvo que en este caso recibe el texto a descifrar.

El vector de inicialización (16 bytes generados aleatoriamente) debe ser el mismo en el encriptado que en el desencriptado por lo que se ha decidido colocarlo al principio del fichero encriptado, de manera que cuando se reciba el archivo para desencriptar, los primeros 16 bytes corresponden a este vector y los demás son los datos correspondientes al fichero.

3.3.2.2 Pruebas

Para llevar a cabo las pruebas correspondientes a esta parte realizaba la carga de archivos autenticado como usuario, se comprobaba que el fichero se encontraba encriptado y posteriormente se descargaba, viendo que el fichero se encontraba igual que antes de la carga.

3.3.3 Tarea 9. Integración de AutoFirma

El primer paso para integrar AutoFirma en mi aplicación web fue estudiar su documentación, ya que era la primera vez que trabajaba con esta aplicación.

La integración se realiza mediante código Javascript**,** puesto que es el navegador web quien llama a la aplicación AutoFirma en el equipo del usuario.

De todos los algoritmos de firmado que ofrece AutoFirma he decidido utilizar **xAdES**, concretamente una **firma despegada**, para así obtener un fichero aparte del original donde se encuentra la firma. Sin embargo, como se ha mencionado anteriormente ésta no es la única manera de firmar que ofrece esta herramienta, por ejemplo, según el manual de integración: ”es posible realizar las firmas XAdES en cuatro modos diferentes: Enveloping (Por defecto), Enveloped, Internally Detached (también referida en este documento como Detached) y Externally Detached” **[3]**. y como se describe en el apartado 5 (Conclusiones y trabajos futuros) se podría configurar la aplicación web y la herramienta para que el usuario pudiera elegir el tipo de firma y el formato deseados.

3.3.3.1 Detalles de implementación

Para implementar esta parte de la aplicación es necesario importar el fichero “Autoscript.js”, el cual nos da toda la funcionalidad relacionada con AutoFirma en Javascript. Concretamente la función utilizada para firmar el documento es “*Autoscript.sign()*”. Dicha función recibe como argumentos el modo y formato de firma además de los datos a firmar*,* entre otros.

3.3.3.2 Pruebas

Para probar la firma de ficheros, se realizó la carga de ficheros autenticado como usuario, accedía a la firma asociada al fichero subido y mediante la propia aplicación AutoFirma trataba de validarla.

3.3.4 Tarea 10. Adecuación de la interfaz a los estándares actuales

Para el desarrollo de la interfaz se aprovechó la previa configuración de los archivos estáticos en el proyecto para añadir también los archivos **CSS**.

El primer objetivo de esta tarea era colocar los elementos de la manera adecuada, centrándolos para dar un aspecto mas ordenado.

Posteriormente se adecuó la página dotándola de la fuente (“*Montserrat*”) y la paleta de colores que utiliza la Universidad de Jaén.

3.3.4.1 Pruebas

Para probar esto lo único necesario era ir refrescando las páginas en las que se estaba trabajando para ver como se muestran al usuario. Es importante que este refresco sea sin tener en cuenta la caché ya que al efectuar cambios en los archivos estáticos podrían no verse al refrescar de la manera usual.

3.3.5 Tarea 11. Conformidad con los requisitos de accesibilidad WCAG nivel AA

Para llevar a cabo esta tarea únicamente hubo que tener presente el listado de requisitos e ir comprobándolos uno a uno, y si no se cumplía dicho requisito se modificaba el código para asegurar su cumplimiento.

A continuación enumero los requisitos de primer nivel, divididos por principios, que aparecen en dicho listado junto a una breve descripción de ellos.

3.3.5.1 Principio 1 - Perceptible

- **Alternativas de texto**: Proporcionar una alternativa para todo contenido no textual de manera que se pueda modificar, en caso de ser necesario, a Braille, audio, etc.
- **Elementos basados en tiempo**: Proporcionar alternativas para los elementos basados en tiempo como el audio o el vídeo, como ofrecer subtítulos, lenguaje de signos y descripción de audio.
- **Adaptabilidad**: El contenido debe ser presentado de distintos modos sin perder información ni estructura.
- **Distinguible**: Es necesario facilitar el ver y oír el contenido, y concretamente distinguir el fondo del primer plano de la aplicación.

3.3.5.2 Principio 2 - Operable

- **Accesible mediante teclado**: Toda la funcionalidad debe ser accesible mediante el teclado.
- **Suficiente tiempo**: Se debe proporcionar al usuario del tiempo suficiente para leer y usar el contenido.
- **Convulsiones y reacciones físicas**: No diseñar el contenido de manera que cause convulsiones y reacciones físicas en personas vulnerables.

- **Navegable**: Proporcionar maneras para ayudar a los usuarios a navegar, encontrar contenido y determinar donde se encuentran.
- **Modalidades de introducción de datos**: Facilitar a los usuarios operar a través de otros medios a parte del teclado.

3.3.5.3 Principio 3 – Comprensible

- **Legible**: El contenido textual debe ser legible y comprensible.
- **Predecible**: La aplicación web debe parecer y operar de maneras predecibles.
- **Asistencia en la introducción de datos**: Ayudar a los usuarios a evitar y corregir errores.

3.3.5.4 Principio 4 – Robusto

- **Compatible**: Maximizar la compatibilidad con agentes del usuario actuales y futuras, incluyendo tecnologías asistenciales.

3.4 Cuarta Iteración. Elaboración de la Documentación

Esta iteración comprende tanto la elaboración del manual de usuario presente en la aplicación web como la del presente documento.

El manual de usuario se diseñó teniendo en mente que debe ser una referencia rápida en la que el usuario debe poder encontrar lo que busca rápidamente, por lo tanto, es necesario que este esté bien desglosado y estructurado.

En cuanto al presente documento, se ha elaborado teniendo en cuenta todas las decisiones tomadas y todos los hitos del desarrollo del proyecto de manera que queden reflejados, ya que el objetivo de esta memoria es que quede documentado el proceso completo.

3.5 Pruebas finales

Una vez finalizado el desarrollo se volvieron a realizar todas las pruebas que se han mencionado en este apartado, para comprobar que todas las funcionalidades implementadas se mantenían en correcto funcionamiento.

Además de éstas, se efectuaron pruebas funcionales también denominadas **End-To-End** que consisten en la simulación de escenarios de uso usuales de los usuarios. En estas pruebas se debe tener muy presente la especificación de requisitos ya que el grado de completitud de éstos es lo que nos dará el nivel de adecuación de la aplicación.

A continuación describo los escenarios usados para dichas pruebas:

- El usuario se autentica, descarga la firma asociada a un fichero (cargado previamente) y cierra sesión
- El usuario se autentica, elimina un fichero, carga uno diferente, lo descarga introduciendo una contraseña errónea, lo intenta de nuevo con la contraseña adecuada y cierra sesión.

Puesto que los casos de uso de la aplicación no son muy numerosos no es posible simular escenarios mucho más complejos, sin embargo este es un buen modo de probar el conjunto de la aplicación con todas sus funcionalidades.

4 MODELO DE NEGOCIO

El éxito de cualquier servicio en línea depende en gran medida de la creación de un modelo de negocio sólido que permita no solo ofrecer un servicio de calidad, sino también garantizar la **sostenibilidad financiera** a largo plazo. En el caso de un servicio de almacenamiento seguro en línea, el modelo de negocio debe ser cuidadosamente diseñado para equilibrar la seguridad, la accesibilidad y la viabilidad económica.

4.1 Oferta de Servicios

El servicio de almacenamiento seguro en línea se basa en la premisa fundamental de proporcionar a los usuarios la capacidad de almacenar, acceder y compartir sus archivos de manera segura y conveniente. La oferta de servicios debe incluir características clave que resuelvan las preocupaciones de seguridad y privacidad de los usuarios, al tiempo que les brinden una experiencia de usuario intuitiva y accesible. Entre las características esenciales se encuentran:

- **Cifrado Robusto:** El servicio debe implementar algoritmos de cifrado sólidos para proteger los datos almacenados y transmitidos. Los archivos deben cifrarse antes de salir del dispositivo del usuario y solo ser descifrados cuando el usuario los accede.
- **Firmado Digital:** La función de firmado digital garantiza que los archivos no hayan sido modificados desde su creación y que provienen de una fuente confiable. Esto refuerza la integridad de los datos almacenados.
- **Almacenamiento Redundante**: Los archivos deben almacenarse en servidores redundantes y geográficamente distribuidos para garantizar la disponibilidad y resistencia a fallos.
- **Interfaz de Usuario Amigable**: La plataforma debe ofrecer una interfaz de usuario intuitiva que permita a los usuarios cargar, organizar y acceder a sus archivos de manera eficiente.

4.2 Estrategia de monetización

Para lograr la viabilidad financiera, el modelo de negocio puede basarse en diversas estrategias de monetización. He sopesado las siguientes opciones:

- **Suscripciones de Pago**: Ofrecer diferentes niveles de suscripción con características y capacidades incrementales. Los usuarios pueden elegir entre planes gratuitos con características básicas y planes premium que ofrecen mayor capacidad de almacenamiento, velocidad y características avanzadas de seguridad.
- **Modelo Freemium**: Ofrecer una versión gratuita con funcionalidades básicas y limitadas, y luego incentivar a los usuarios a actualizar a planes pagados para acceder a características más avanzadas y mayor capacidad de almacenamiento.
- **Publicidad Dirigida**: En el caso de una versión gratuita, se podría generar ingresos a través de publicidad dirigida en la plataforma.

En este caso se ha decidido que la opción más adecuada es la opción Freemium, con un modelo básico gratuito y diferentes opciones de pago que permitan mejorar la capacidad de almacenamiento y las velocidades de carga y descarga.

5 CONCLUSIONES Y TRABAJOS FUTUROS

En este TFG se ha desarrollado un servicio web dedicado al almacenamiento remoto de archivos cifrados y firmados digitalmente por el usuario. Para determinar el grado de satisfacción de dicho desarrollo es necesario tener en cuenta el nivel de cumplimiento de los requisitos propuestos que aparecen en el apartado 2.1.1 (Especificación de Requisitos). Se considera que dicho grado de cumplimiento es alto, puesto que todos los requisitos funcionales han sido verificados. Sin embargo es difícil evaluar el grado de cumplimiento de los requisitos no funcionales de índole subjetiva como “La interfaz de la aplicación web debe ser estéticamente agradable y moderna” o “La aplicación web debe tener una interfaz intuitiva y sencilla”. En el caso de la primera de las citadas se considera que la completitud es suficiente pero no total, puesto que la aplicación es visualmente agradable pero podría tener un acabado mucho más profesional. En cuanto a la intuitividad y sencillez sí que se piensa que el grado de completitud es el máximo puesto que la interfaz carece de complicaciones innecesarias y es lo más simple posible.

A continuación se proponen algunas posibles líneas de mejora para esta aplicación, que debido a las limitaciones de este trabajo no se han podido implementar.

- **Autenticación en dos pasos**: Para aumentar la seguridad en el proceso de autenticación se puede añadir un factor de autenticación más a parte de la contraseña, como recibir un código por SMS a un teléfono asociado a la cuenta o un correo electrónico.
- Capacidad de **compartir archivos**: También considero que sería una buena adición que el usuario pudiera compartir sus archivos para que otros usuarios pudieran descargarlos.
- **Buscador de archivos**: Para los usuarios que tengan muchos archivos en el almacén sería buena idea incluir un buscador para facilitar la tarea de encontrar un archivo concreto.
- **Software propio de firma:** Para evitar depender del software externo AutoFirma se podría implementar, por medio de alguna librería o desde cero, el proceso de firmado, dotando al sistema de mayor libertad y mayor grado de personalización.

- **Árbol de directorios personalizados**: Para ayudar al usuario a organizar mejor sus archivos, se le podría permitir la creación de directorios en el almacén y su re-ordenamiento.

6 APÉNDICES

6.1 Guía original del Trabajo Fin de Título

Tutor del TFG: Juan Carlos Cuevas Martínez

Modalidad: Proyecto de Ingeniería | **Tipo**: TFG Específico

Número máximo de estudiantes: 1 (1 asignados)

Idioma: Castellano

El presente TFG tiene como objetivo el crear una aplicación web que permita alojar documentos en el servidor de manera que estos estén cifrados y solo accesibles a su propietario, ási como verificar su integridad a través de firma digital.

Conocimientos Previos

- Protocolo HTTP y tecnologías web.
- Técnicas de protección y verificación de integridad de la información a través de suites de cifrado.

Objetivos del TFG

- Desarrollar una aplicación web (back-end y front-end) que permita solamente el acceso al servicio solamente a usuarios autenticados.
- El servicio proporcionará los medios para que los archivos que se suban al servidor están cifrados adecuadamente y se añada una firma digital que permita verificar posteriormente su integridad.
- El servicio de almacén permitirá añadir o eliminar archivos, así como descargarlos.
- El front-end permitirá el acceso al servicio a través de tecnologías que proporciones una interfaz intuitiva y de diseño actual.
- Se deberán observar aspectos de accesibilidad y protección de datos como la posible gestión de cookies.

Metodología a Desarrollar

- Estudio del proceso a implementar.
- Estudio preliminar de las tecnologías.

- Especificación detallada de requisitos.
- Determinación de las tecnologías a emplear.
- Diseño preliminar.
- Re-definición de requisitos y modificación del diseño.
- Diseño definitivo.
- Implementación de la aplicación:
  - Vista administrador: creación y gestión de usuarios.
  - Vista de usuario: acceso y gestión del almacén seguro.
- Pruebas y corrección de errores.
- Elaboración de la documentación.

Documentos y Formatos de Entrega

Los establecidos por la normativa y además:

- Manual de despliegue de la aplicación.
- Manual de usuario de la aplicación.

6.2 Instalación y configuración del sistema

Para poder instalar todo el sistema se deben seguir los siguientes pasos:

- Instalación de Python y Django como se detalla en el apartado Primera Iteración. Investigación Inicial e Instalación de las Herramientas Necesarias.
- Usando el gestor de paquetes PIP, es necesario instalar la librería “Werkeuz”. Esto se hace mediante el comando “pip install“ seguido del nombre de la librería.
- Como el código de AutoFirma impide que se ejecute en la IP 127.0.0.1, es necesario modificar el archivo hosts de Windows, que se encuentra en la ruta C:\Windows\System32\drivers\etc\hosts. Basta con añadir la línea “127.0.0.1 storuja.com”. En lugar de storuja.com se puede colocar el nombre que se desee.
- Una vez todas las librerías están instaladas, para lanzar la aplicación es necesario ejecutar el comando “python manage.py runserver_plus --cert-

file cert.pem --key-file key.pem" con la terminal de Windows ubicada en la raíz del proyecto.

- Una vez completado el comando anterior, si en el navegador web colocamos el nombre de dominio colocado en el archivo hosts de Windows ya se accede a la aplicación web del proyecto.

6.3 Manuales de usuario

6.3.1 Manual de despliegue de la aplicación

Puesto que la aplicación no se ha desplegado y sólo se ha ejecutado en local no existe un manual de despliegue de ésta, sin embargo, para desplegar la aplicación los pasos a seguir son los siguientes:

- Configurar un servidor tipo Apache, Nginx o similares.
- Instalar en la máquina en la que se ha instalado dicho servidor todo lo necesario mencionado en el apartado anterior: Python, Django, las librerías, etc.
- Establecer el ajuste de Django DEBUG a false.
- Cambiar el ajuste de Django SECRET_KEY a una variable de entorno o leída de un archivo de sólo servicio.
- Activar el servidor y acceder a la dirección IP pública que se haya configurado previamente.

6.3.2 Manual de usuario de la aplicación

El manual de usuario de la aplicación está incluido en ella para facilitar al usuario la consulta de éste. Para acceder a él hay que pulsar sobre la palabra **Ayuda** en la barra inferior de la aplicación.

7 DEFINICIONES Y ABREVIATURAS

En este apartado se describen los términos y abreviaturas utilizadas en el presente documento.

- **Front-End**: Parte de la aplicación web que se ejecuta en el navegador web y con la que interactúa el usuario.
- **Back-End**: Parte de la aplicación web que se ejecuta en el servidor, y que responde a las peticiones del front-end.
- **AES**: Algoritmo de cifrado basado en un esquema de cifrado por bloques adoptado como estándar por el gobierno de los Estados Unidos.
- **MVC**: Patrón de diseño software enfocado a las aplicaciones web que establece una distinción entre el modelo de datos, las vistas y el controlador que une funcionalmente las anteriores.
- **XadES**: Formato de firma electrónica avanzada basado en XML.
- **CAdES**: Formato de firma electrónica avanzada basado en sintaxis de mensaje encriptado.
- **PAdES**: Formato de firma electrónica avanzada enfocado a archivos PDF.
- **PBKDF2**: Función para derivar una clave de cifrado a partir de una contraseña en texto plano.
- **API**: Código que permite a diferentes aplicaciones comunicarse entre ellas.

8 BIBLIOGRAFÍA

[1] KALISKI, Burt. (2000). *PKCS #5: Password-Based Cryptography Specification Version 2.0 .* RFC 2898. https://www.doi.org/10.17487/RFC2898

[2] National Institute of Standards and Technology. (2001). *Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication (FIPS) NIST FIPS 197-upd1. https://doi.org/10.6028/NIST.FIPS.197-upd1

[3] Secretaría General De Administración Digital. (2021). *Manual Integrador 1.7.0. https://administracionelectronica.gob.es/ctt/resources/Soluciones/138/Descargas/Manual%20Integrador%201-7-0.pdf?dIniciativa=138&idElemento=18913*