



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**  
**título del TFG**



Presentado por Nombre del alumno  
en Universidad de Burgos — 18 de febrero  
de 2025

Tutor: nombre tutor







UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Nombre del alumno, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 18 de febrero de 2025

Vº. Bº. del Tutor:

Vº. Bº. del co-tutor:

D. nombre tutor

D. nombre co-tutor





## Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

## Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

## **Abstract**

A **brief** presentation of the topic addressed in the project.

## **Keywords**

keywords separated by commas.



---

# Índice general

---

Índice general	iii
Índice de figuras	iv
Índice de tablas	v
1. Introducción	1
2. Objetivos del proyecto	3
3. Conceptos teóricos	5
3.1. Conceptos de LaTeX . . . . .	7
3.2. Secciones . . . . .	8
3.3. Referencias . . . . .	8
3.4. Imágenes . . . . .	8
3.5. Listas de items . . . . .	9
3.6. Tablas . . . . .	9
4. Técnicas y herramientas	11
5. Aspectos relevantes del desarrollo del proyecto	13
6. Trabajos relacionados	15
7. Conclusiones y Líneas de trabajo futuras	17
Bibliografía	19

---

# Índice de figuras

---

3.1. Autómata para una expresión vacía . . . . .	9
--	---

---

# Índice de tablas

---

3.1. Herramientas y tecnologías utilizadas en cada parte del proyecto	10
---	----



---

# 1. Introducción

---

Descripción del contenido del trabajo y del estructura de la memoria y del resto de materiales entregados.



---

## **2. Objetivos del proyecto**

---

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.





---

## 3. Conceptos teóricos

---

En esta sección se definen aquellos conceptos que son necesarios conocer para comprender el resto del documento.

### **Análisis estático**

Técnica de detección de *malware* que se realiza sin la necesidad de ejecutar el programa en cuestión. Este método se basa en la obtención, inspección y evaluación de las características que se pueden extraer de un archivo binario, tales como su estructura, código fuente (si está disponible), indicios de obfuscación u otras técnicas de ocultación, cadenas de texto incrustadas en este, firmas digitales, huella digital *hash o signature* del archivo, secuencias de bytes concretas, cabeceras del programa, metadatos incrustados, desensamblado del ejecutable y otras propiedades que pueden ser extraídas directamente del archivo. Las ventajas que este enfoque presenta son, su simplicidad, rapidez y bajo coste computacional, ya que no requiere de entornos de ejecución específicos ni de hardware especializado para probar el comportamiento del programa. Sin embargo, el mayor problema de este tipo de análisis es su dificultad para detectar malware que utiliza técnicas avanzadas de ofuscación o cifrado, ya que estas prácticas dificultan la extracción de información útil del binario.

### **Análisis dinámico**

Técnica de detección de *malware* que consiste en evaluar el comportamiento de un programa mediante su ejecución en un entorno controlado, con el objetivo de observar sus interacciones con el sistema operativo, los recursos de este y otros programas. En este enfoque, se monitorean actividades como

la modificación de archivos, el tráfico de red generado, la creación de procesos o la inyección de código en estos, lo cual permite identificar patrones de comportamiento asociados con programas maliciosos. A diferencia del análisis estático, el análisis dinámico ofrece una mayor precisión, ya que puede detectar comportamientos maliciosos que no son evidentes simplemente escaneando el archivo de manera estática, como el uso de técnicas de ofuscación. Sin embargo, este tipo de análisis sigue teniendo sus inconvenientes, por un lado, es más complejo, requiere de más recursos computacionales y es más costoso de implementar, dado que involucra la ejecución real del código en un entorno controlado, generalmente una máquina virtual (*sandbox*). Por otro lado, también es poco eficiente contra casos en los que el *malware* detecta el hecho de que está siendo analizado y oculta su comportamiento malicioso. Además, puede no ser adecuado para dispositivos con recursos limitados, como dispositivos IoT o móviles, debido a sus altos requerimientos de hardware y tiempo.

## Análisis híbrido

Metodo de detección de *malware* el cual combina las fortalezas tanto del análisis estático como del dinámico. En este método, el programa se ejecuta en un entorno controlado, y durante su ejecución, se realizan *dumps* de memoria de manera periódica o en respuesta a comportamientos sospechosos. Estos volcados de memoria son posteriormente analizados utilizando técnicas de análisis estático para identificar posibles patrones maliciosos, tales como la inyección de código en procesos ajenos, manipulación de memoria que no le pertenece al programa o modificaciones en partes protegidas de la memoria pertenecientes al sistema operativo. Este enfoque permite una detección más precisa de *malware* que utiliza técnicas avanzadas de ocultamiento, ya que combina la observación del comportamiento en tiempo real con la inspección detallada del estado de la memoria. Sin embargo, el análisis híbrido es el más complejo y costoso de implementar, ya que requiere tanto de infraestructura de virtualización como de herramientas para realizar un buen análisis de memoria. A pesar de todo, suele ofrecer los mejores resultados en términos de detección.

## Huella digital (*fingerprint*)

El fingerprinting o, la generación de huellas digitales de archivos, es una técnica utilizada para identificar de manera única un archivo mediante la aplicación de funciones criptográficas de *hashing*. Este proceso consiste en calcular un *hash* a partir del contenido completo del archivo utilizando

algoritmos como MD5, SHA-1, SHA-256 u otros. El resultado es una cadena de longitud fija que actúa como un identificador único para ese archivo. Cualquier modificación, por mínima que sea, en el contenido del archivo resultará en un *hash* completamente diferente, lo que permite detectar alteraciones o corrupciones en estos.

Esta técnica es muy utilizada en la verificación de la integridad de archivos, la detección de duplicados, y la identificación de malware conocido al comparar el *hash* que este genera con una base de datos de muestras previamente catalogadas. Sin embargo, una limitación importante del *fingerprinting* es su sensibilidad extrema a cambios mínimos, lo que dificulta la identificación de archivos que han sido ligeramente modificados pero que conservan una estructura o funcionalidad. Esto implica que incluso cambiar un bit en el *padding* del archivo, hace que este ya no se detecte como malware al tener una huella digital diferente.

### Huella digital difusa (*fuzzy hashing*)

El *fuzzy hashing*, o *hashing* difuso, es una técnica que extiende el concepto del *hashing* tradicional al permitir la comparación de archivos basada en similitudes parciales en lugar de una coincidencia exacta. A diferencia del *hashing* convencional, que opera sobre el archivo completo, el *fuzzy hashing* divide el archivo en bloques o segmentos y calcula un *hash* para cada uno de ellos. Este enfoque por bloques permite identificar similitudes entre archivos incluso cuando solo una porción de su contenido ha sido modificada.

El *fuzzy hashing* es particularmente útil en el análisis forense digital y la detección de *malware*, ya que permite identificar variantes de archivos maliciosos que han sido modificados para evadir su detección, pero que conservan partes significativas de su código original. Al comparar dos *hashes* difusos, es posible calcular un grado de similitud basado en la cantidad de bloques que coinciden entre ambos. Esto se logra mediante algoritmos especializados como SSDeep o TLSH, que están diseñados para generar *hashes* difusos y medir la similitud entre ellos.

## 3.1. Conceptos de LaTeX

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Algunos conceptos teóricos de L<sup>A</sup>T<sub>E</sub>X<sup>1</sup>.

## 3.2. Secciones

Las secciones se incluyen con el comando `section`.

### Subsecciones

Además de secciones tenemos subsecciones.

### Subsubsecciones

Y subsecciones.

## 3.3. Referencias

Las referencias se incluyen en el texto usando `cite` [1]. Para citar webs, artículos o libros [?], si se desean citar más de uno en el mismo lugar [?, ?].

## 3.4. Imágenes

Se pueden incluir imágenes con los comandos standard de L<sup>A</sup>T<sub>E</sub>X, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

---

<sup>1</sup>Créditos a los proyectos de Álvaro López Cantero: Configurador de Presupuestos y Roberto Izquierdo Amo: PLQuiz



Figura 3.1: Autómata para una expresión vacía

### 3.5. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

**Primer item** más información sobre el primer item.

**Segundo item** más información sobre el segundo item.

▪

### 3.6. Tablas

Igualmente se pueden usar los comandos específicos de  $\text{\LaTeX}$  o bien usar alguno de los comandos de la plantilla.

Herramientas	App	AngularJS	API REST	BD	Memoria
HTML5		X			
CSS3		X			
BOOTSTRAP		X			
JavaScript		X			
AngularJS		X			
Bower		X			
PHP			X		
Karma + Jasmine		X			
Slim framework			X		
Idiorm			X		
Composer			X		
JSON		X	X		
PhpStorm		X	X		
MySQL				X	
PhpMyAdmin				X	
Git + BitBucket		X	X	X	X
MikTeX					X
TeXMaker					X
Astah					X
Balsamiq Mockups		X			
VersionOne		X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

---

## 4. Técnicas y herramientas

---

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas. No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.





---

## 5. Aspectos relevantes del desarrollo del proyecto

---

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros<sup>3</sup>, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.



---

## 6. Trabajos relacionados

---

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.



---

## **7. Conclusiones y Líneas de trabajo futuras**

---

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.



---

# Bibliografía

---

- [1] Wikipedia. Latex — wikipedia, la enciclopedia libre. <https://es.wikipedia.org/w/index.php?title=LaTeX&oldid=84209252>, 2015. [Internet; descargado 30-septiembre-2015].