uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Informática

2018-2019

Trabajo Fin de Grado

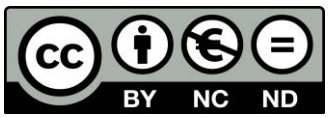
"Uso de checksum en la descarga online de aplicaciones"

Jorge Platas Feced

Tutora:

Lorena González Manzano

4 de Julio a las 15:00 en Salón de Grados en Colmenarejo



AGRADECIMIENTOS

Antes de comenzar con el proyecto, me gustaría dedicar este apartado para mencionar a aquellas personas que han hecho posible o más sencilla la carrera.

Mención especial para mi familia, ya que son los primeros que han hecho posible esto, ofreciéndome una de las mejores educaciones desde el inicio hasta la etapa universitaria incluida, siendo consciente el gran esfuerzo económico y psicológico que esto ha supuesto.

Por supuesto, una de las mejores partes de la vida universitaria ha sido conocer a mis compañeros con los que he creado un gran grupo de amigos y nos hemos ayudado entre todos a sacar adelante esta difícil carrera. Esto es una de las mejores cosas que me llevo conmigo mismo ya que sin ellos esto no habría sido igual, y de los que he aprendido muchísimo durante estos 4 años.

Y no puedo finalizar este apartado sin hacer una mención especial a mi tutora, ya que tengo mucho que agradecerla. Fue mi profesora en segundo de carrera en la asignatura de Criptografía y Seguridad Informática, momento en el que empecé a interesarme por la ciberseguridad. Además, me saqué un curso sobre ciberseguridad en la que ella participaba como profesora, por lo que decidí contactar con ella para pedirla consejo e información, y ella siempre me ofreció su conocimiento y ayuda sobre el tema. Teniendo esto en mente, tuve claro que quería realizar mi Trabajo de Fin de Grado con ella. Durante estos 4 meses ha sido de gran ayuda ya que me lo ha explicado todo siempre con un gran nivel de detalle, y lo mejor de todo, me ha respondido en cuestión de horas para comentarme posibles mejoras sobre mi trabajo y dudas que me han ido surgiendo.

Con mi trabajo, esfuerzo y dedicación, junto con las personas que he mencionado anteriormente, he sido capaz de finalizar mis estudios y poder realizar este proyecto en el que he invertido tanto empeño.

RESUMEN

Hoy en día, los usuarios descargan numerosas aplicaciones en sus ordenadores confiando en que aquello que están descargando se corresponde con el software original y no está corrupto o dañado de manera intencional o accidental. Por ello, los usuarios son responsables de lo que descargan e instalan en sus ordenadores. Puede darse la situación en la que un usuario sea víctima de un ataque informático en la que el atacante haga creer al usuario que está accediendo a la página oficial, y se descarga un programa maligno o diferente al original.

Por este motivo, se ha decidido estudiar qué información proporcionan las páginas de descargas para verificar que lo que el usuario descarga es correcto y no ha sido modificado. También, se analiza y se compara qué soluciones y herramientas existentes hay en la actualidad para verificar dicha comprobación.

Para verificar que las aplicaciones se corresponden con el software original, y este no ha sido modificado, los desarrolladores de aplicaciones proporcionan para ello el checksum del archivo, pretendiendo garantizar la integridad de los datos. Este proceso consiste en comparar el checksum proporcionado por los desarrolladores con el checksum generado del archivo descargado.

La solución propuesta para realizar la verificación del archivo es una extensión para el navegador Google Chrome. Esta aplicación realiza de manera automática la verificación buscando coincidencia entre los checksums generados del archivo y los ofrecidos en la página de descarga.

Palabras clave

checksum, hash, descarga online, seguridad, integridad, extensión web

ABSTRACT

Nowadays, users download numerous applications to their computers trusting that what they are downloading corresponds to the original software and is not corrupted or damaged intentionally or accidentally. Therefore, users are responsible for what they download and install on their computers. The situation may arise in which a user is the victim of a computer attack in which the attacker makes the user believe that he or she is accessing the official page, and a malignant or different program is downloaded from the original.

For this reason, it has been decided to study what information the download pages provide to verify that what the user downloads is correct and is not modified. Also, it is analyzed and compared what existing solutions and tools are currently available to verify this verification.

In order to verify that the applications correspond to the original software, and this has not been modified, the application developers provide the file's checksum, trying to guarantee the integrity of the data. This process consists of comparing the checksum provided by the developers with the checksum generated from the downloaded file.

The proposed solution to perform file verification is an extension for the Google Chrome browser. This application automatically performs the verification looking for a match between the checksums generated from the file and those offered on the download page.

Keywords

Checksum, hash, online download, security, integrity, web extension

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. CONCEPTOS PREVIOS

El objetivo de esta sección es explicar los conceptos que se consideran necesarios para la comprensión del documento. Es aconsejable su lectura antes de comenzar a leer los siguientes apartados.

- Criptografía: es la técnica de cifrar y codificar los datos con el objetivo de protegerlos para que solo puedan ser accedidos por las personas autorizadas.

- Integridad: en el ámbito de la seguridad de la información hace referencia a la capacidad de garantizar que los datos no han sido modificados desde su creación. El objetivo es asegurar que la información no ha sido alterada (por accidente o alguien no autorizado) y por lo tanto garantizar su validez [1].

- Función hash: también conocido como función resumen en español, y función digest, es una función criptográfica resultado de aplicar un algoritmo matemático el cual a partir de un bloque de datos produce una salida alfanumérica única de longitud fija. Es irreversible, es decir, a partir del resultado de una aplicar una función hash no se puede obtener el dato original. Además, es independiente del tamaño de la entrada ya que el resultado es un resumen de los datos, y genera una salida de tamaño fijo actuando como identificador único de los datos de entrada. Estas funciones son utilizadas para verificar la integridad de los datos tal que, si se aplica una función hash sobre un dato determinado, y se modifica su valor y se vuelve a aplicar otra vez la misma función hash, se obtendrían 2 valores distintos, pudiendo así comprobar que estos han sido modificados. A continuación, se van a detallar una serie de funciones hash de interés para el caso de estudio.
    - MD5 (Message-Digest Algorithm 5): produce un código hash de 128 bits, representado por 32 símbolos hexadecimales. En 2004 se publicó el descubrimiento de colisiones (cuando se introducen dos entradas de datos distintas y producen la misma salida) para este hash, por lo que se declaró como “roto”, dando lugar a ser uno de los menos seguros [2].
    - SHA-1 (Secured Hash Algorithm 1): produce un código hash de 160 bits, representado por 40 símbolos hexadecimales, basado en las mismas técnicas que MD5. Gran utilizado para firmas electrónicas (con la combinación de otros

algoritmos). A pesar de ser considerado más seguro que MD5, en 2017 se encontraron las primeras colisiones de este [3].

- SHA-256 (Secured Hash Algorithm 256): produce un código hash de 256 bits, representado por 64 símbolos hexadecimales. Uno de los más seguros, siendo de los más usados por el equilibrio que tiene entre seguridad y coste computacional de generación [4].
- SHA-512 (Secured Hash Algorithm 512): produce un Código hash de 512 bits. El más seguro de todos los mencionados hasta ahora, y tomará más importancia de cara al futuro.

TABLA 1.1. COMPARACIÓN DE ALGORITMOS.

Algoritmo	**Tamaño salida bits**	**Número símbolos hexadecimal**	**Iteraciones**	**Operaciones**	**Colisiones**
MD5	128	32	64	+, and, or, xor, rot	Sí
SHA-1	160	40	80	+, and, or, xor, rot	Sí
SHA-256	256	64	64	+, and, or, xor, shr, rot	No
SHA-512	512	128	80	+, and, or, xor, shr, rot	No

Fuente: Wikipedia

- Checksum: también conocido como suma de verificación o suma de chequeo, es el resultado de aplicar una función hash con el objetivo de verificar la integridad de los datos comprobando si estos han sufrido alguna modificación.

- Dirección IP: identificador unívoco para cada dispositivo electrónico conectado a la red. Está compuesto por 4 conjuntos de número entre 0 y 255 separados por puntos entre ellos, por ejemplo: “192.168.1.1”.

- Host: conocido también como huésped o anfitrión, es un término informático que se utiliza para referenciar a cualquier aparato electrónico que ofrece servicios y está conectado a la red.

- Dominio: identificador unívoco de una página web. Suele estar compuesto por 3 partes. Empieza con “www” seguido del nombre de la web u organización, y el tipo de organización. Por ejemplo: “www.google.com”.

- Servidor: máquina física conectada a la red que atiende peticiones de otros, llamados clientes, ofreciendo distintos servicios. Por ejemplo, queremos descargos una imagen contenida en una determinada página web. Nosotros como cliente mandamos una petición o solicitud al servidor de la página web, y este procesa la solicitud y nos proporciona el contenido deseado.

- Proxy: un servidor intermediario entre la conexión cliente (origen) y servidor (destino) comúnmente utilizado para encubrir la dirección IP cliente, por distintos fines que se consideren, por ejemplo, imaginemos que ciertas direcciones IPs de Rusia no tienen permitido el acceso a unas páginas del gobierno de España. Utilizando un proxy podría realizar las peticiones desde otra dirección IP y así poder acceder a dicho contenido.

- Mockup: en el ámbito del diseño informático sirve para mostrar y evaluar el diseño de las interfaces que se van a desarrollar con el objetivo de dar una imagen aproximada de cómo será la interfaz final del sistema.

- Protocolo: en términos de la informática y las telecomunicaciones, este término hace referencia al conjunto de reglas y normas para establecer la comunicación. Los protocolos pueden estar regidos por unos estándares determinados. En este caso de estudio se mencionan 2 tipos de protocolos:

    - Http (Hypertext Transfer Protocol): es un protocolo de comunicación entre cliente y servidor en el que este primero es el que inicia la conexión a través de peticiones, y este último envía de vuelta mensajes llamados respuestas. Es usado mayoritariamente para el envío de documentos HTML, imágenes, vídeos y otros datos entre servidores [6]. La información que viaja entre ambas conexiones está sin cifrar, es decir, los datos viajan en claro. Esto supone que este protocolo no se use para envío de información privada o sensible.
    - Https (Hypertext Transfer Protocol Secure): es un protocolo de comunicación entre cliente y servidor, pero, a diferencia de http, la conexión es segura, ya que los datos viajan cifrados con el fin de que si alguien se interpone en la conexión no pueda obtener en claro los datos de la comunicación.

2. INTRODUCCIÓN

En este primer capítulo se realiza una breve presentación acerca de por qué se ha elegido este tema para desarrollar el Trabajo de Fin de Grado, y qué solución se trata de aportar para mejorar el problema actual.

2.1 Problemática y necesidad

Hoy en día cualquier usuario instala y prueba distintas aplicaciones en sus terminales, ya sea en el ordenador, en el móvil o en la tablet. Para ello, en algunos casos, se invierte tiempo comparando las distintas opciones en función de las necesidades, verificando opiniones, funcionalidades que ofrecen, precio etc.

Cuando un usuario realiza la descarga de la aplicación, accede a la tienda oficial de aplicaciones, o a la página web del desarrollador correspondiente, y obtiene el producto que necesita. Por lo general, el usuario accede a la sección de descargas y selecciona el programa que se dese. Puede darse el caso de que dicho software haya sido intercambiado por personas ajenas malintencionadas, y el usuario obtenga un programa malicioso o diferente al esperado, o en el mejor de los casos, que algún archivo del software ha sido modificado o dañado durante el proceso de descarga.

Para evitar hechos como los anteriormente mencionados, los desarrolladores facilitan el checksum, que es un conjunto de caracteres alfanuméricos, resultado de aplicar una función hash, con el objetivo de comprobar que los datos no hayan sufrido modificaciones. Esta comprobación se realiza comparando el checksum proporcionado en la página de descarga con el checksum generado que se obtiene del archivo descargado, ya sea con alguna aplicación o herramienta online. Esta verificación sirve para cumplir con uno de los principios de la seguridad de la información, la integridad, que es la capacidad de garantizar que los datos no han sido modificados de manera accidental o malintencionada previniendo así cambios no autorizados [1].

El problema que esta verificación presenta es que el usuario es quien tiene que realizar dicha comprobación de manera manual, por lo que solamente el pequeño porcentaje de aquellos usuarios que son conscientes de la existencia de dicho checksum, simplemente

por cuestiones de tiempo o despreocupación, no realizan la correspondiente comprobación.

Por este motivo se ha considerado necesario realizar una aplicación que automatice dicho proceso de comprobación y, de ser necesario, alertar al propio usuario cuando no se pueda garantizar la integridad de un determinado software. Para ello, primero se van a estudiar los tipos de aplicaciones existentes, y ver cómo se puede ofrecer algo nuevo o mejorado al usuario.

2.2 Motivación de trabajo

Es un hecho que en la actualidad la seguridad informática juega un papel fundamental. En ocasiones, la gente se preocupa en que los desarrolladores se encarguen de asegurar las aplicaciones con más y mejores medidas de seguridad como ofrecer más factores de autenticación u ofreciendo una mayor protección de sus datos. A pesar de esto, los usuarios tienen la responsabilidad de saber lo que se está descargando y el uso que se hacen de las distintas aplicaciones.

Por otro lado, el comportamiento por defecto del usuario es depositar la confianza en las páginas oficiales y populares. Esto es correcto ya que se supone que la información y el software que ofrecen es el oficial y válido. En cambio, un usuario puede ser víctima de un ataque en el que se le redirija a una página que se haga pasar por la oficial, pudiendo así ofrecer software maligno u obtener información sensible del usuario.

La motivación del desarrollo de este trabajo viene dada por la importancia que tiene comprobar todo aquello que se descarga de Internet. No es suficiente obtener los programas de páginas que se consideran fiables, sino que es necesario verificar que el programa que se está descargando no está corrupto, ya sea de manera intencionada o no. Por ello, se pretende concienciar y proporcionar una solución fácil para el usuario, y así garantizar uno de los pilares fundamentales de la seguridad informática, como es la integridad.

2.3 Objetivos

Como finalidad de este trabajo, en primer lugar, se va a estudiar qué medidas ofrecen las páginas de descarga de aplicaciones para comprobar la integridad del software que ofrecen. Una vez analizado esto, el siguiente paso será analizar las aplicaciones existentes para facilitar dicha verificación haciendo un balance de las facilidades que estas ofrecen y aquellos puntos nuevos que se pueden ser desarrollados y mejorados.

La idea principal de la aplicación que se va a desarrollar consiste en una extensión para un navegador del ordenador. El objetivo es que la aplicación realice de manera automática la verificación de la integridad de los archivos que se descarguen, con el fin de mostrar al usuario un mensaje informativo acerca del resultado. En caso de que la verificación sea satisfactoria se mostrará un mensaje informando al usuario acerca de ello; en caso contrario, en el que no se haya podido comprobar la verificación del archivo, se notificará dicho problema y los peligros que estos puedan suponer.

La decisión de los mensajes a mostrar, como del navegador sobre el que se desarrollará la aplicación, será decisión en una fase más avanzada después de haber realizado el estudio oportuno.

2.4 Metodología

Para el desarrollo del proyecto se ha seguido una metodología en cascada. Esta también es conocida como secuencial o ciclo de vida de un programa puesto que para el desarrollo de las distintas etapas del proyecto las anteriores han de considerarse como finalizadas. Las etapas se han dividido en tareas para el desarrollo completo del proyecto.

La elección de esta metodología ha sido por los siguientes factores:

- La inversión inicial en la fase de estudio ayuda a prevenir grandes cambios en la etapa de desarrollo.
- Desde el inicio se tiene clara la funcionalidad principal de la aplicación. Además, este proyecto se considera estable en términos de que los requisitos del sistema no tendrán una gran variación a lo largo del tiempo.
- Resulta sencilla la manera de estructurar y explicar el proyecto, por lo que facilita su entendimiento.

- No se tiene la necesidad de ir entregando pequeñas versiones al cliente. Se ha considerado correcto que la aplicación cumpla satisfactoriamente las funcionalidades al final del proyecto. Además, el proyecto no sufre de modificaciones no previstas ante las necesidades que puedan surgir del cliente, ya que se parte de un plan bien marcado tras la fase de estudio.

Una vez mencionado las razones principales de los beneficios de esta metodología para el desarrollo de este proyecto, es importante resaltar que en algunas ocasiones se dará el caso en el que se tenga que modificar alguna tarea anterior por posibles cambios o actualizaciones del proyecto. Con esto se asegura que todas las fases tengan coherencia y sean acumulativas, y estén relacionadas y actualizadas.

3. ESTADO DEL ARTE

En este apartado se va a estudiar la situación actual en la que se encuentra el problema que se va a desarrollar. Esto servirá para obtener una visión general y para poder analizar qué ofrecen las alternativas de hoy en día para dicho problema y, así, poder saber cuáles son las deficiencias actuales para determinar mejor la solución, mejorando lo existente y aportando las novedades que se consideren.

3.1 Situación actual

Antes de comenzar con el análisis de las aplicaciones existentes en la actualidad, se va a realizar una comparativa de las distintas páginas online en las que se puede descargar programas, con el objetivo de comparar qué hashes proporcionan, y qué supone esto, además de otros datos de seguridad importantes a tener en cuenta. Para ello, se han seleccionado una serie de aplicaciones populares del 2019 [7] y se han añadido otras por propia elección para analizar un mayor número de aplicaciones.

A continuación, se muestra la Tabla 3.1 compuesta por las siguientes columnas:

- Nombre: identificador del programa.
- Descripción: breve información acerca de las funciones principales.
- Protocolo página programa: protocolo de transferencia que utiliza la página de descarga de la aplicación. Puede ser http o https.
- Protocolo página checksum: protocolo de transferencia que utiliza la página que ofrecen el hash correspondiente al programa descargado. Puede ser http o https. En caso de no proporcionar dicho hash se ha dado el valor de un guion rojo.
- Host: relación entre la página de descarga y la página que ofrece el hash, contemplando si ambos están ubicados en el mismo servidor, o si pertenecen a distintos dominios. En caso de no proporcionar información acerca del checksum, se da el valor de un guion rojo.
- MD5, SHA1 Y SHA256 y SHA512: los 4 tipos de funciones hash que se han analizado. El caso afirmativo es representado con un tick verde y, en caso contrario, una cruz roja.

TABLA 3.1. ANÁLISIS DE PROGRAMAS.

Nombre	Descripión	Protocolo página programa	Protocolo página checksum	Host	MD5	SHA1	SHA256	SHA512
Adobe Acrobat Reader	Visor PDF	https	-	-	✗	✗	✗	✗
Audacity	Editor de vídeo y audio	https	https	Diferente dominio	✗	✗	✓	✗
Avast	Antivirus	https	-	-	✗	✗	✗	✗
AVG	Antivirus	https	-	-	✗	✗	✗	✗
Code Blocks	Entorno desarollo C y C++	http	http	Mismo servidor	✓	✓	✓	✗
CrypTool	Programa algoritmos criptográficos	https	https	Diferente dominio	✗	✗	✓	✗
Dropbox	Alojamiento de archivos en la nube	https	-	-	✗	✗	✗	✗
Eclipse	Entorno desarrollo de Java	https	https	Mismo servidor	✗	✗	✗	✓
FL Studio	Editor de audio	https	https	Mismo servidor	✗	✗	✓	✗
GIMP	Editor imágenes digitales	https	https	Diferente dominio	✗	✗	✓	✗
Microsoft Office	Suite ofimática para Windows	https	-	-	✗	✗	✗	✗
Oracle SQL Developer	Entorno de desarrollo bases de datos	http	https	Diferente dominio	✓	✓	✗	✗
Photoshop	Editor gráfico	https	-	-	✗	✗	✗	✗
Putty	Cliente SSH	https	-	-	✗	✗	✗	✗
PyCharm	Entorno desarrollo de Python	https	https	Diferente dominio	✗	✗	✓	✗
RStudio	Computación estadística y gráficos	https	https	Diferente dominio	✓	✗	✗	✗
Skype	Comunicador de voz y vídeo	https	-	-	✗	✗	✗	✗
TeamViewer	Acceso remoto a otro equipo	https	-	-	✗	✗	✗	✗
Ubuntu	Sistema operativo	http	http	Mismo servidor	✗	✗	✓	✗
uTorrent	Descargador de archivos	https	-	-	✗	✗	✗	✗
VirtualBox	Software de virtualización	https	https	Mismo servidor	✓	✗	✓	✗
VLC	Reproductor multimedia	https	https	Diferente dominio	✗	✗	✓	✗
WinRAR	Compresor de datos	https	-	-	✗	✗	✗	✗

Antes de comenzar con el análisis de los datos expuestos en la Tabla 3.1 es importante aclarar cómo se han obtenido los datos de la columna Host. Para determinar si el checksum se encuentra en el mismo host que el software a descargar se han obtenido las direcciones IPs al acceder a las distintas páginas con DNSlytics [9]. De este modo,

estarían en el mismo host si el checksum y el software a descargar tuviesen la misma dirección IP. En caso contrario podría decirse que están en distintos servidores. Para esto se ha supuesto que no se hace uso de un proxy puesto que no se puede saber con certeza, por lo que esto podría variar algo los datos mostrados en la tabla.

Una vez explicado el significado de las columnas y sus correspondientes dominios, se procede al análisis de los datos y a obtener una conclusión de ellos.

En primer lugar, el dato a destacar sería que, de los 23 programas elegidos a estudiar, 11 de ellos no facilitan ningún tipo de hash para poder comprobar la integridad de la descarga. Esto supone casi el 48% de los programas elegidos, lo que representa prácticamente la mitad de ellos. Esto supone un gran problema de seguridad, ya que no habría manera de comprobar la integridad del archivo descargado, por lo que si un usuario se descarga uno de estos programas tendrá que confiar que aquello que se va a instalar, es la versión original, y no está corrupto. Ahora se van a analizar los 12 programas restantes que sí que ofrecen algún hash. Por otro lado, 3 de los 12 programas utilizan el protocolo http, lo que significa que no está cifrada la conexión entre usuario y servidor, por lo que los datos viajan en claro (sin cifrar). Esto podría dar lugar al problema conocido en seguridad como Man-In-The-Middle (hombre en el medio en español) en el que un atacante podría interponerse en la conexión entre usuario y servidor, e ir enviando la información que considere al usuario, haciéndose pasar por la página a la que ha accedido el usuario. Esto permitiría al atacante enviar al usuario un archivo de descarga diferente al de la página web oficial de origen, o cambiar el valor del checksum. Otro factor de riesgo importante que se puede observar es que 5 de los 12 programas que ofrecen el hash el archivo descargable y su correspondiente hash de comprobación en el mismo servidor. Esto supone un riesgo en cuanto a que, si dicho servidor se ve comprometido por un atacante, este tendría acceso a ambos recursos. En cambio, si ambos recursos estuviesen ubicados en distintos servidores, el atacante tendría que obtener el acceso a ambos recursos de manera independiente, lo que sería más complicado. Por último, quedaría analizar los checksums que estas páginas ofrecen. De los 12 programas que lo ofrecen, 4 de ellos proporcionan MD5, 2 ofrecen SHA1, 9 de ellos proporcionan SHA256 y 1 el SHA512. De este último dato se puede concluir que 9 de 12, es decir, el 75% de estos programas proporcionan SHA256, siendo este uno de los más seguros junto con SHA512.

A grandes rasgos, se puede concluir que gran parte de los programas a día de hoy, siguen sin proporcionar el checksum, con los riesgos que esto conlleva. De aquellos que sí que

lo ofrecen, algunos programas utilizan protocolo http, pero se trata de casos puntuales, por lo que la gran mayoría utilizan https, ya que es algo básico para garantizar el cifrado entre las conexiones. Casi la mitad de estos alojan el archivo descargable en el mismo servidor que el propio checksum, lo cual quizás sea por simplicidad a la hora de distribuir los datos, aunque esto supone un problema de seguridad. Por último, una minoría de estos programas ofrecen el checksum MD5, lo que no termina de ser del todo seguro, ya que este se ha demostrado que es más vulnerable a ataques. Es aconsejable que estos programas ofrezcan otras alternativas, como SHA256 o SHA512. Como punto positivo, SHA256, es proporcionado por 9 de los 12 programas que ofrecen la verificación.

A modo de ampliación del estudio, se ha decidido investigar si las aplicaciones con más descargas y usuarios ofrecen los hashes de sus archivos de descarga [10]. Estas aplicaciones son WhatsApp, Facebook, Twitter etc. Estas aplicaciones tienen millones de usuarios. Por ello, podría darse que dichos usuarios descarguen alguna de estas aplicaciones en su ordenador. En tal caso, este tipo de aplicaciones no ofrece ningún tipo de hash de comprobación. Esto supone un gran riesgo de seguridad debido a la gran cantidad de personas que utilizan a diario estas aplicaciones y, como mínimo, debería proporcionarse la posibilidad de verificar que lo que se descarga no está corrupto y se corresponde con el software original.

3.2 Aplicaciones existentes

A continuación, se van a detallar y comparar aquellas soluciones existentes para el proceso de verificación de la integridad. Así se obtiene una visión más detallada de lo que estos programas ofrece y así poder determinar sus ventajas y desventajas para desarrollar la aplicación de maneras más acertada.

- **MD5SUM**

Es una herramienta online gratuita. Ofrece la posibilidad de calcular el hash MD5 de un archivo. La forma de realizar dicho cálculo es seleccionando el archivo que se quiera comprobar, y arrastrarlo sobre la página web. Automáticamente proporciona el hash calculado del fichero. Dicha página es http, por lo que es menos segura.

La ventaja es que se puede realizar la generación del hash desde el navegador web, por lo que no habría problemas de compatibilidad entre sistemas operativos y se evitaría la necesidad de ser instalado.

Como desventaja, solo ofrece la posibilidad de obtener el hash MD5 y, además, solo permite utilizar un archivo con tamaño máximo de 500Mb [11]. Ofrecer solo este algoritmo es limitante, ya que al ser este el más débil y menos seguro, muchas páginas de descarga de software ya no lo proporcionan.

A continuación, se muestra en la Figura 3.1 el resultado tras haber arrastrado un fichero de prueba.

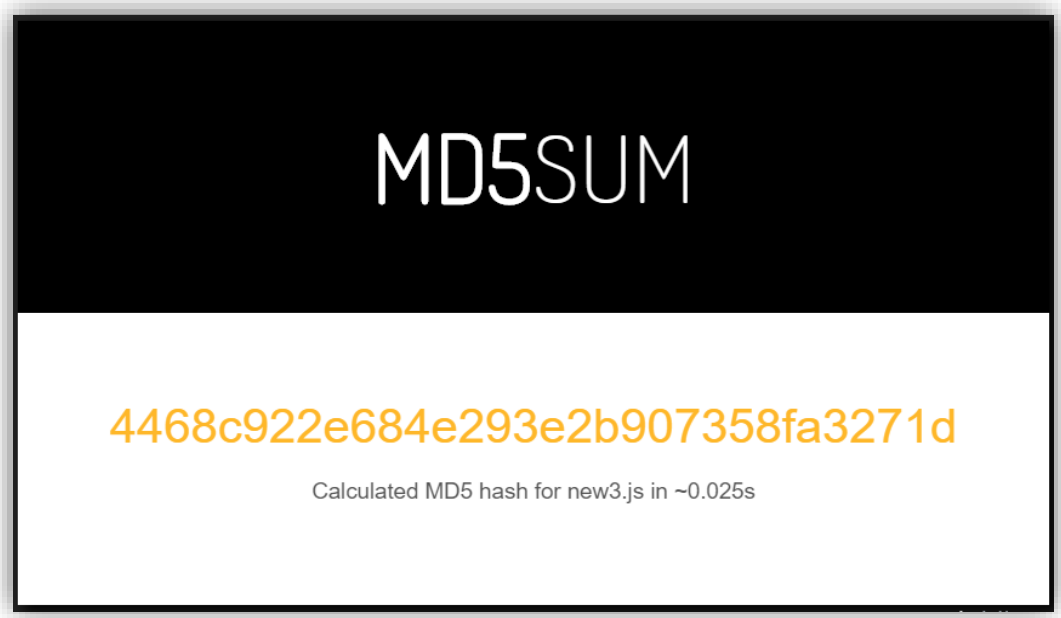


Figura 3.1. Aplicación MD5SUM [11].

- **MD5 & SHA Checksum Utility**

Esta herramienta está disponible solo para Windows. Tiene una versión gratuita, y otra de pago. La versión gratuita, que ha sido la utilizada, genera las siguientes funciones hash: MD5, SHA-1, SHA-256, SHA-384 y SHA-512. Ofrece distintos hashes los cuales, además, suelen ser proporcionados en las páginas de descargas. Para poder utilizar esta herramienta es necesario descargarse un archivo ejecutable de la página web del desarrollador. Para su utilización se tiene que seleccionar el archivo que se desea verificar, y automáticamente muestra los distintos hashes calculados mencionados anteriormente. Por último, ofrece la posibilidad de introducir en un campo el valor del hash que

proporciona la página de descarga, y así comprobar y mostrar un mensaje emergente con el correspondiente resultado (si coinciden o no) [12].

Como desventajas serían la limitación de estar solo disponible para Windows, y la necesidad de descargar el archivo ejecutable.

En la Figura 3.2 se muestra el resultado tras seleccionar ejecutable para instalar la aplicación de Eclipse.

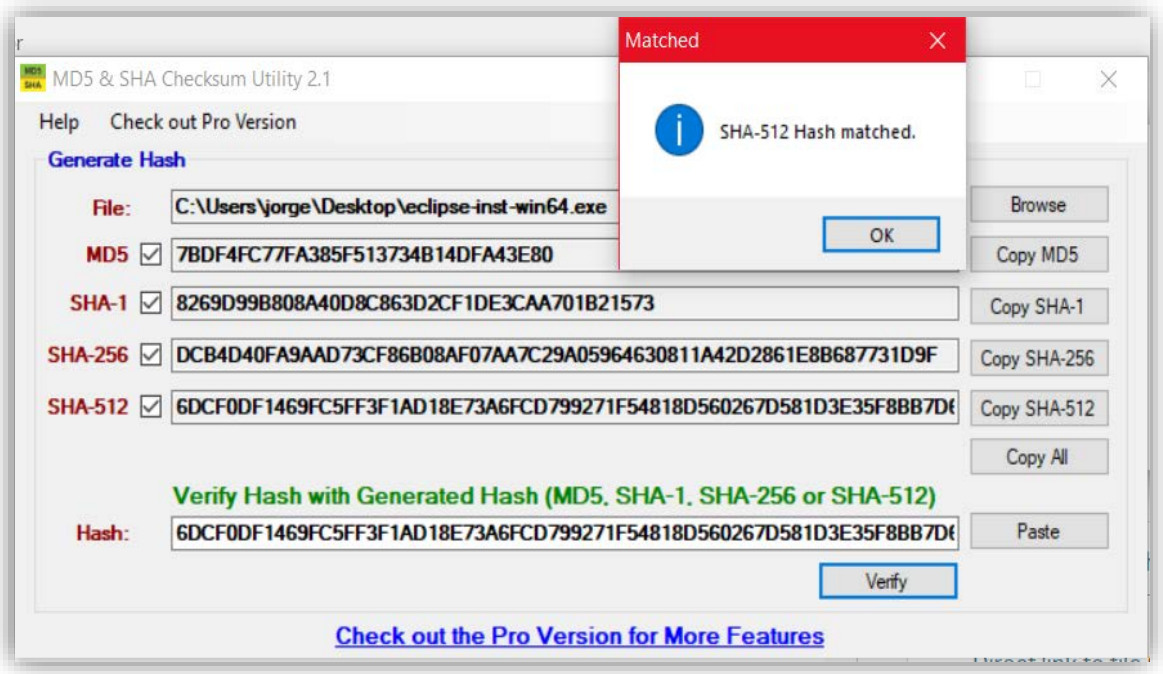


Figura 3.2. Aplicación MD5 & SHA Checksum Utility [12].

- **IgorWare Hasher**

Esta herramienta es gratuita y compatible solo con Windows. Ofrece la generación de los hashes SHA-1, MD5 y CRC32. De igual modo que el programa anterior, se ejecutaría el archivo descargado del programa, en el que se selecciona el archivo que se desea comprobar. Proporciona el cálculo de los distintos hashes del archivo en la parte inferior. En la parte superior, ofrece la posibilidad de introducir en los campos el hash a comprobar. También, ofrece la posibilidad de introducir dicho hash a través de un archivo externo, por si fuese el caso [13].

Como desventaja, mencionada anteriormente, los cifrados que proporcionan son de bajo nivel de seguridad por lo que lo limita bastante su uso hoy en día.

En la Figura 3.3 se puede observar el resultado obtenido tras seleccionar el archivo de instalación de VirtualBox.

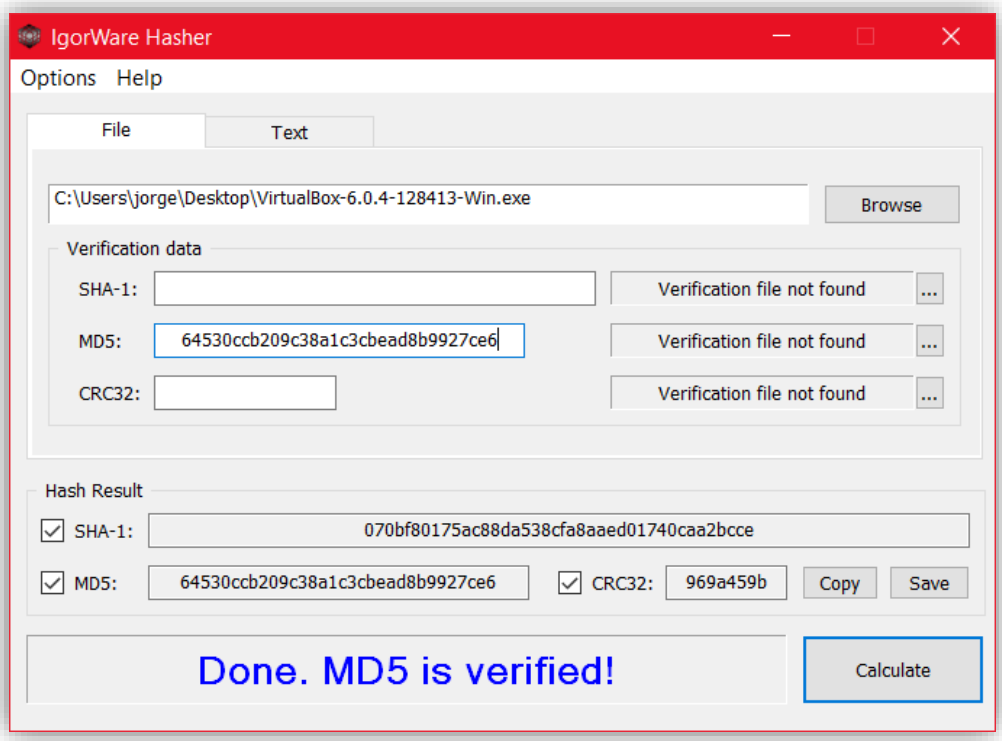


Figura 3.3. Aplicación IgorWre Hasher [13].

- **Hash Generator**

Este software es gratuito y solo compatible con Windows. Proporciona un archivo de descargar el cual tiene que ser instalado en el ordenador. Ofrece la posibilidad de utilizar un archivo o introducir el texto. Posteriormente, genera 11 tipos de hashes [14].

Este programa tiene la ventaja de que ofrece gran variedad de funciones hash, en cambio, esto es algo contraproducente ya que, por este gran número de alternativas, el proceso de cálculo es algo lento teniendo que esperar alrededor de unos 30 segundos para que finalice el proceso. Además, algunos de ellos son poco seguros, por lo que pocos desarrolladores lo ofrecen en sus páginas. Por último, no ofrece la posibilidad de comparar el hash original del archivo, por lo que tendría que ser comprobado con otra herramienta externa o por uno mismo.

El resultado de este programa se muestra en la Figura 3.4 tras seleccionar el archivo de instalación de Eclipse.

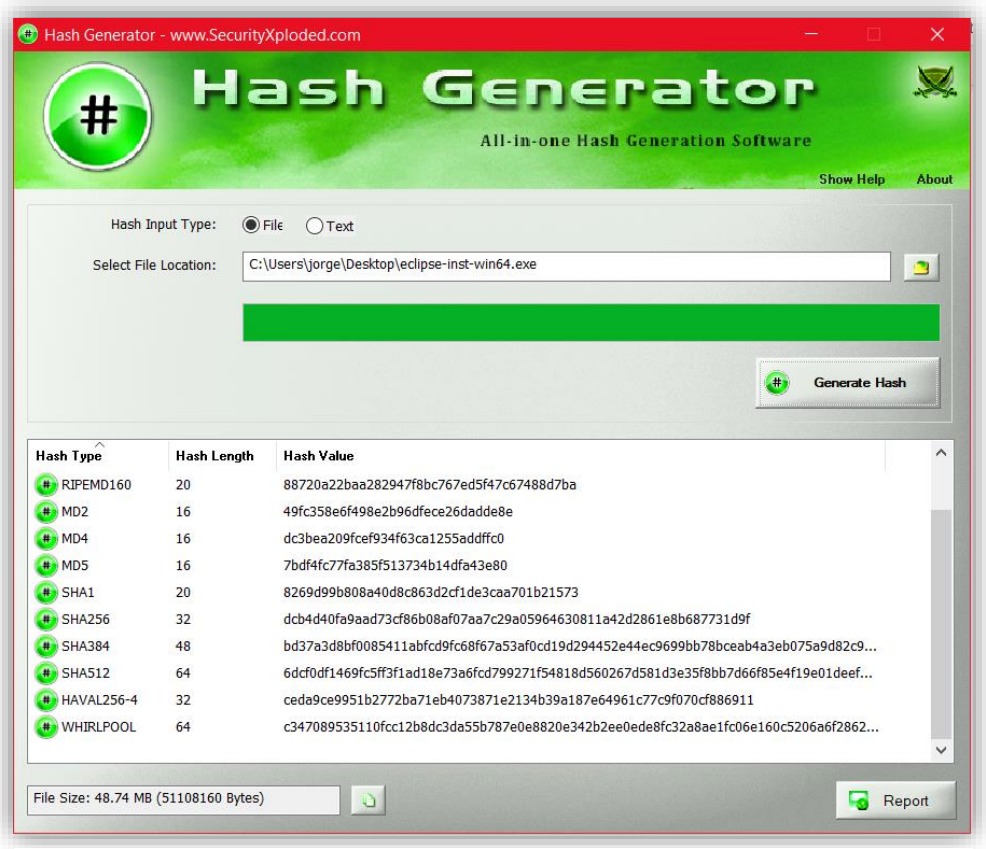


Figura 3.4. Aplicación Hash Generator [14].

- **Download and Verify Checksum**

Es una extensión que está disponible para Google Chrome, por lo que sería compatible con cualquier sistema operativo en el que esté instalado este navegador. Esta aplicación ha sido encontrada en un estudio [8] por lo que no está disponible en la plataforma de Google Chrome (tienda oficial de descargas) y está sin terminar, por lo que tiene ciertas limitaciones. Una vez agregada dicha extensión al navegador, cuando detecta una descarga, manera automática calcula los hashes MD5, SHA1 y SHA256, y muestra una alerta en la que informa si el hash generado coincide con alguno de los que se encuentran en la misma página de descarga.

Como ventaja ofrece al usuario la automatización del proceso de verificación. En cambio, solamente funciona para aquellos archivos y hashes que se encuentren en la misma página de descarga.

Para probar este programa se ha descargado un archivo de prueba proporcionado en dicho estudio. El resultado se muestra en la Figura 3.5.

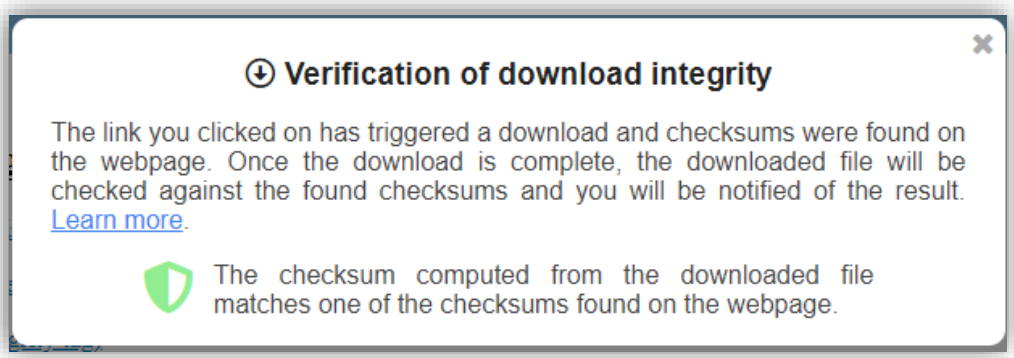


Figura 3.5. Aplicación Download and Verify Checksum [8].

Los programas de estudio que se han seleccionado han sido aquellos que han aparecido en las primeras búsquedas en Internet, y los que recomendaban en algún blog [15]. Este suele ser el proceso por defecto que realiza un usuario al descargar una aplicación sin conocer el nombre previamente. Las debilidades encontradas en la mayoría de estos programas son los siguientes:

1. No ofrecen todos los algoritmos que proporcionan las páginas de descarga.
2. Se proporciona en gran mayoría hashes poco seguros.
3. Se necesita hacer la comprobación manualmente.
4. No son compatibles con todos los sistemas operativos.
5. Poseen una interfaz poco atractiva.

El último programa (Download and Verify Checksum) sí que cumple los puntos 2 y 3, pero el resto no, por lo que este último sería la excepción (además que no es tan accesible para los usuarios puesto que pertenece a un estudio y no aparece en las primeras búsquedas y recomendaciones en Google).

En conclusión, existen distintas aplicaciones para comprobar la integridad de los archivos descargados. No todas calculan los hashes que se ofrecen en las páginas de descargas, y muchos de estos son pocos seguros. Exceptuando la última aplicación analizada, el resto sólo están disponibles para Windows lo que supone una gran limitación y, además, para realizar la comprobación de la integridad se requiere la acción del usuario. Por ello, se ha considerado que la mejor alternativa es el desarrollo de una extensión para el navegador del ordenador para evitar problemas de compatibilidad. La aplicación calculará los

checksums md5, sha1, sha256 y sha512 ya que son los que ofrecen las páginas de descarga en su mayoría. Sha512 solamente se ofrece en 1 de los programas, pero se considera una buena inversión a futuros ya que alguno de estos hashes, como md5 y sha1, son menos seguros y es posible que en un futuro dejen de ofrecerse. Por último, el proceso de verificación de integridad se realizará de manera automática y transparente, lo que significa que no precisará de la acción del usuario.

4. ANÁLISIS

En este apartado se pretende exponer los objetivos generales de la aplicación, así como las funcionalidades que esta debe cumplir junto con una visión de alto nivel del sistema.

4.1 Perspectiva general de la aplicación

Como se comentó en el apartado 2.3 (Objetivos), la idea es desarrollar una extensión para el navegador de Internet que verifique automáticamente la integridad del archivo descargado. A grandes rasgos, se decidió desarrollar una extensión por las siguientes razones:

- Se evitan problemas de compatibilidad con sistemas operativos.
- Se tiene acceso directo a la página de descarga y a las descargas del usuario.
- Automatizar el proceso de verificación para evitar que tenga que hacerlo el usuario.

La primera elección para la creación de la aplicación es decidir para qué navegador se va a desarrollar dicho software. Para ello, se va a estudiar cuál ha sido más utilizado durante el último año. A continuación, se va a mostrar 2 gráficos en la Figura 4.1 que se han realizado a partir de la información obtenida de W3Counter [16]. La información que se va a comparar es la cantidad de tráfico que recibe cada uno de los navegadores.

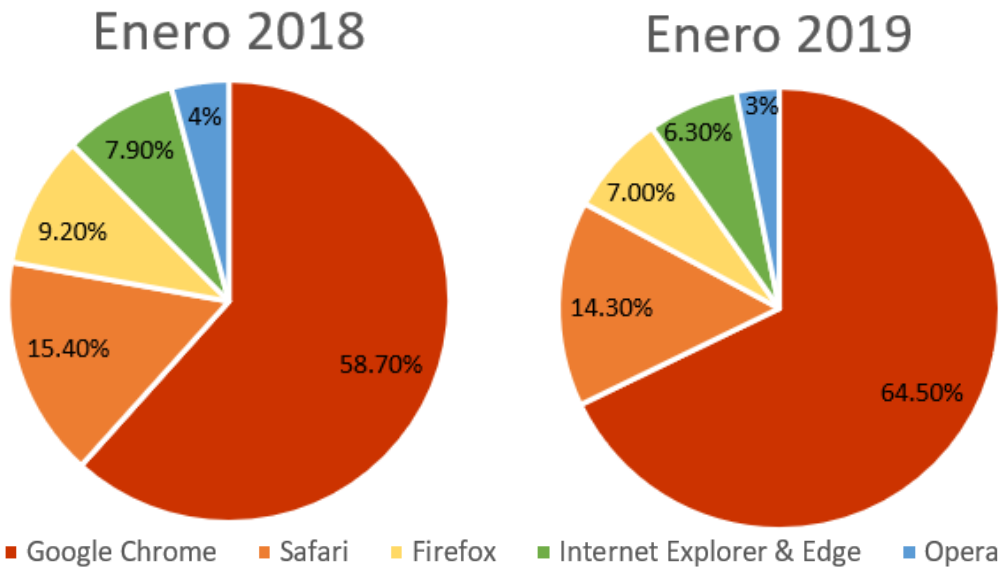


Figura 4.1. Comparación del tráfico en los navegadores [16].

En primer lugar, en vez de estudiar los navegadores más usados del 2018 (puesto que al ser del año anterior puede haber modificaciones) y del 2019 (tan solo llevamos unos meses de este año), se ha elegido analizarlos en el mismo mes de ambos años, por lo que se correspondería con 1 año entero. Como se puede observar en la Figura 4.1, el dato más relevante es que el navegador predominante en ambos diagramas es Google Chrome, con una gran diferencia. En ambos años representa más de la mitad, incrementando su tráfico de datos casi un 6% (del 58,7% en 2018 al 64,5% en 2019). Finalmente, se puede concluir de manera clara que Google Chrome es el navegador más utilizado, y lo que es interesante también es que este ha incrementado su tráfico por lo que podría continuar siendo así en los siguientes años y ser una inversión acertada. Por lo tanto, como el objetivo es proporcionar nuestra aplicación al mayor número de usuarios, Google Chrome será el elegido.

Una vez elegido el navegador, el siguiente paso es detallar las funcionalidades que la aplicación va a tener. Aun habiéndolo mencionado anteriormente, es de especial importancia dejar claro que nuestro software tiene verificar la integridad de forma transparente, es decir, sin que el usuario tenga que interactuar con la aplicación. Por lo tanto, la aplicación será la encargada de realizar el proceso en segundo plano (ejecutar cálculos y comprobaciones sin que el usuario lo vea) y finalmente muestre el mensaje correspondiente con el resultado final.

La aplicación existente más similar a la que se va a desarrollar es la mostrada en el apartado 3.2 (el análisis de aplicaciones existentes), llamada “Download and Verify Checksum”. Como se demostró en su momento, esta era capaz de realizar la comprobación automática, pero tan solo era capaz de hacerlo cuando el archivo a descargar se encuentra exactamente en la misma página que el checksum proporcionado por los desarrolladores. Como es de suponer, esto es un factor muy limitante ya que no todas las páginas de descarga lo hacen así, sino que en ocasiones ofrecen dichos hashes en links externos.

Para poder dar una solución más avanzada e intentar mejorar el problema limitante anterior, se ha decido lo siguiente: en primer lugar, se van a generar los hashes MD5, SHA1, SHA256, y se va a añadir SHA512, siendo este uno de los menos proporcionados hoy en día, pero puede ser una inversión a futuros puesto que es de los más seguros, y lo normal es que los más débiles y los que se han encontrado colisiones (MD5 y SHA1) dejen de ser utilizados. Por otro lado, haciendo un estudio de las páginas que ofrecen el

checksum en páginas diferentes a las de la descarga del archivo, todas ellas tienen algo en común: indican en qué enlace están esos hashes. Con lo cual, se ha decidido ampliar la funcionalidad de nuestra aplicación, la cual se encargará de buscar en la página de descarga el checksum, y si no lo encuentra, lo buscará en links a otras páginas a las que se hacen referencia desde la página de descarga. Esto supone un gran avance ya que por lógica la información que se suele aportar para verificar la integridad se hace en el momento de la descarga. Aun así, esto tiene el límite de que no se abarcan todos los links de una página web entera puesto que tan solo se accede a aquellos referenciados en la página de descargas.

4.2 Tecnologías para el desarrollo

Como se ha concluido anteriormente, la extensión se va a desarrollar para el navegador web Google Chrome. Esta plataforma impone una estructura específica del proyecto a seguir y una serie de tecnologías, por lo que no existe la posibilidad o alternativa de elegir otras distintas.

Como Google Chrome es un navegador web, estas tecnologías están basadas en el desarrollo web. Estas son:

- HTML (HyperText Markup Language): enfocado al desarrollo de la página web, como su descripción, estructura y elementos.
- CSS (Cascading Style Sheets): sirve para declarar el estilo y la presentación de la página web.
- JavaScript: lenguaje de programación web que se encarga de añadir funcionalidades y efectos de forma dinámica.

Por último, como en los 2 últimos años he realizado algunos cursos online sobre desarrollo web y, además, he desarrollado algunas páginas web puedo decir que estoy familiarizado con estas tecnologías, por lo que facilitará bastante la etapa de desarrollo.

4.3 Requisitos de software

4.3.1 Justificación de la clasificación de requisitos

La lógica de la aplicación va a ser desarrollada de manera autónoma, es decir, sin la obtención de parámetros ni la acción del usuario (será necesario que el usuario inicie una descarga). El usuario interactúa con el sistema solamente en el caso que seleccione el botón de cerrar la ventana de mensajes. Por lo tanto, se van a considerar todos los requisitos como requisitos del sistema.

4.3.2 Justificación de la plantilla de los requisitos

Para detallar las distintas funcionalidades, se ha optado por la representación en una tabla que contiene las siguientes columnas:

- ID: código de identificación único (RFXX para los requisitos funcionales y RNFXX para los no funcionales donde XX es el número del requisito)
- Nombre: título descriptivo.
- Descripción: detalle de funcionalidades y requerimientos.
- Prioridad: importancia según se considere. Dominio: Alta/Media/Baja
- Estado: situación en la que se encuentra. Para los requisitos funcionales el dominio es Propuesto/En desarrollo/Desarrollado. Para requisitos no funcionales el dominio es Cumplido/No cumplido

Para una mejor comprensión del conjunto de requisitos funcionales y no funcionales del sistema se ha elegido representarlos en la Tabla 4.1 y Tabla 4.2 en hojas horizontales.

4.3.3 Requisitos funcionales

TABLA 4.1. REQUISITOS FUNCIONALES.

ID	Nombre	Descripción	Prioridad	Estado
RF01	Capturar archivo descargado	La aplicación tiene que obtener el archivo descargado por el usuario. Está acción se realiza una vez finalizada la descarga.	Alta	Desarrollado
RF02	Comprobar tipo de archivo.	La aplicación tiene que obtener la extensión del archivo descargado y verificar si se corresponde con una válida. Las extensiones aceptadas son las siguientes: exe, iso, msi, zip, tar, tar.gz, tar.xz, tar.bz2, deb, rpm y dmg.	Alta	Desarrollado
RF03	Generar hashes	La aplicación tiene que generar los distintos hashes del archivo descargado. Estos son: MD5, SHA1, SHA256 Y SHA512.	Alta	Desarrollado
RF04	Comprobar hashes en página de descarga	La aplicación tiene que buscar en la página de descarga si se encuentra alguno de los hashes generados anteriormente. En tal caso, se mostrará un mensaje indicando que se ha podido verificar satisfactoriamente la integridad del archivo descargado.	Alta	Desarrollado
RF05	Comprobar hashes en páginas referenciadas	La aplicación tiene que obtener cada una de las páginas a las que se hace referencia en la página de descarga para así poder comprobar si se encuentra alguno de los hashes generados anteriormente. En caso afirmativo, se mostrará el mensaje correspondiente. En caso contrario, se alertará al usuario del resultado.	Alta	Desarrollado
RF06	Mostrar ventana emergente	La aplicación tiene que mostrar la ventana emergente tras obtener un archivo con extensión aceptada. Durante el resto de la ejecución mostrará los mensajes correspondientes cuando finalice cada ejecución.	Alta	Desarrollado
RF07	Cerrar ventana emergente	La aplicación tiene que permitir al usuario cerrar la ventana emergente.	Alta	Desarrollado

4.3.4 Requisitos no funcionales

TABLA 4.2. REQUISITOS NO FUNCIONALES.

ID	Nombre	Descripción	Prioridad	Estado
RNF01	Aplicación compatible con Google Chrome	La aplicación tiene que ser compatible con el navegador Google Chrome.	Alta	Cumplido
RNF02	Velocidad de ejecución	La aplicación tiene que mostrar la ventana emergente al usuario en menos de 2 segundos tras haber finalizado la descarga.	Alta	Cumplido
RNF03	Aplicación desarrollada en HTML, CSS y JavaScript	La aplicación tiene que ser desarrollado con las siguientes tecnologías: HTML, CSS y JavaScript.	Alta	Cumplido
RNF04	Privacidad	La aplicación no recogerá datos personales, ni de navegación del usuario, puesto que no es necesario para el funcionamiento del software.	Alta	Cumplido
RNF05	Idioma español	La aplicación tiene que desarrollada en español.	Alta	Cumplido

4.4 Diagrama de alto nivel

Para el entendimiento de cómo funciona la aplicación, en este subapartado se va a explicar el transcurso de la ejecución y su orden, mostrado en la Figura 4.2. La lógica de la aplicación se divide en pequeñas funcionalidades. El proceso comienza cuando un usuario inicia una descarga. La aplicación queda a la espera de que esta termine, y cuando finaliza, se obtiene y se verifica si el archivo descargado cumple con las extensiones de interés que se han marcado anteriormente. Cuando se da el caso de ser un archivo dentro de los aceptados, se muestra la ventana emergente con el mensaje de que se ha detectado una nueva descarga. A continuación, se informa al usuario que comienza el cálculo de los diferentes hashes del archivo obtenido. Finalmente, se realizan las comprobaciones para verificar si alguno de los hashes generados coincide con el que se encuentra en la página web de la que se obtiene el archivo, o en alguna de las páginas que se referencian desde la página de descarga. Cuando esta última comprobación termina, se informa al usuario del resultado de la verificación de la integridad de su descarga.

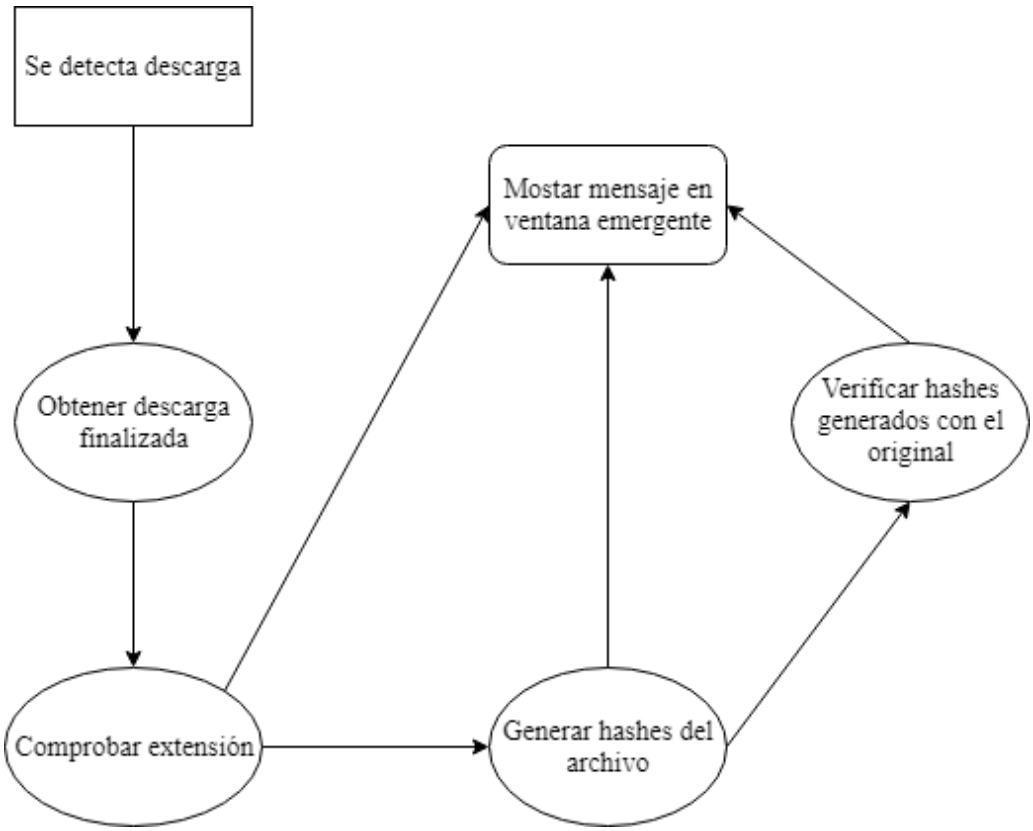


Figura 4.2. Diagrama de alto nivel.

5. DISEÑO

En esta sección se va a profundizar en los distintos aspectos expuestos en el capítulo de análisis. De esta manera se obtendrá una compresión más detallada de la función que desempeña cada componente, la relación entre ellos, y las clases que estos poseen.

5.1 Diagrama de componentes

Partiendo del proceso explicado de la lógica de la aplicación, a continuación, se van a mostrar los componentes que forman el sistema en la Figura 5.1. Para ello se han utilizado nombres en inglés, ya que es una buena práctica a la hora de la programación hacerlo en dicho idioma por cuestión de estándares. El diagrama de componentes es el siguiente:

Figura 5.1. Diagrama de componentes.

Como se puede observar, el sistema contendrá tan solo 2 componentes que llevarán a cabo toda la lógica del programa. En primer lugar, el componente Background es el encargado de realizar en segundo plano distintas funcionalidades: principalmente, es el encargado de estar pendiente de detectar cuándo comienza una descarga para así empezar con el hilo de ejecución. Además, será el encargado de generar los hashes correspondientes y, finalmente, de comprobar si alguno de estos hashes coincide con alguno de los proporcionados en la página de la descarga. Por último, este componente cada vez que finaliza una de sus acciones se encarga de comunicárselo al componente Content. La función principal de este otro componente es mostrar al usuario los distintos mensajes, en forma de ventana emergente. Los mensajes que muestra dependen de los que recibe del otro componente. Es importante resaltar lo siguiente: se le ha otorgado a este componente la funcionalidad de que, en caso de que no se encuentre coincidencia entre los hashes generados y los encontrados en la página de descarga, este se encarga de buscar si alguno de estos hashes se encuentra en alguna de las páginas que se hacen referencia en la página de descarga. Esta función es más sencilla realizarla con este

componente puesto que tiene acceso directo a la página desde la que el usuario realiza la descarga.

Es importante aclarar la flecha discontinua del segundo componente, Content, dirigida hacia el componente Background. Esto es debido a que existe una relación de dependencia ya que para que el componente Content realice sus operaciones, este depende de una serie de mensajes que recibe de Background. Esto quiere decir que Content está a la espera de Background para ir desempeñando sus tareas en función del mensaje que reciba.

5.2 Diagrama de clases

A continuación, se muestra con más detalle en la Figura 5.2 la composición del sistema. Para ello, se indican las clases pertenecientes, con sus atributos y métodos correspondientes.

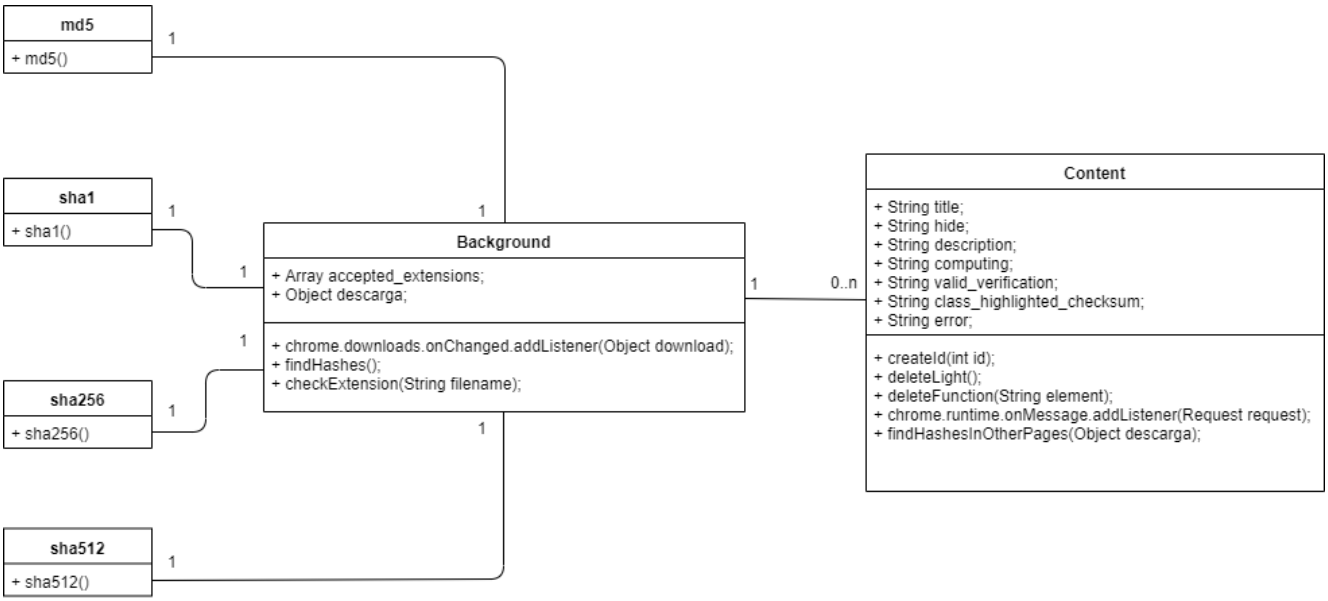


Figura 5.2. Diagrama de clases.

- Background: está a la espera de detectar nueva descarga. En primer lugar, comprueba la validez de la extensión, y posteriormente genera los respectivos hashes (utilizando las clases MD5, SHA1, SHA256 Y SHA512). Por último, comprueba si coincide alguno de estos con los encontrados en las páginas de descargas. Esta clase envía varios mensajes a la clase Content tras finalizar las operaciones correspondientes.

- md5, sha1, sha256, sha512: las clases encargadas de generar los hashes del archivo descargado.
- Content: principalmente el responsable de la creación del mensaje emergente que se muestra al usuario. Esta clase está a la espera de los mensajes que recibe de la clase Background. En función de estos, decide qué mensaje hay que mostrar en cada momento. También se encarga de recorrer las páginas proporcionadas en la página de descarga para la comprobación de hashes.

5.3 Diagrama de casos de uso

l siguiente diagrama mostrado en la Figura 5.3 describe el comportamiento del sistema frente a las acciones del usuario. Como se puede observar, el usuario interacciona con el sistema en 2 situaciones. La primera iniciando la descarga lo cual produce que la aplicación comience a ejecutarse. La segunda es cerrando la ventana emergente producida por la aplicación cuando muestra los mensajes correspondientes.

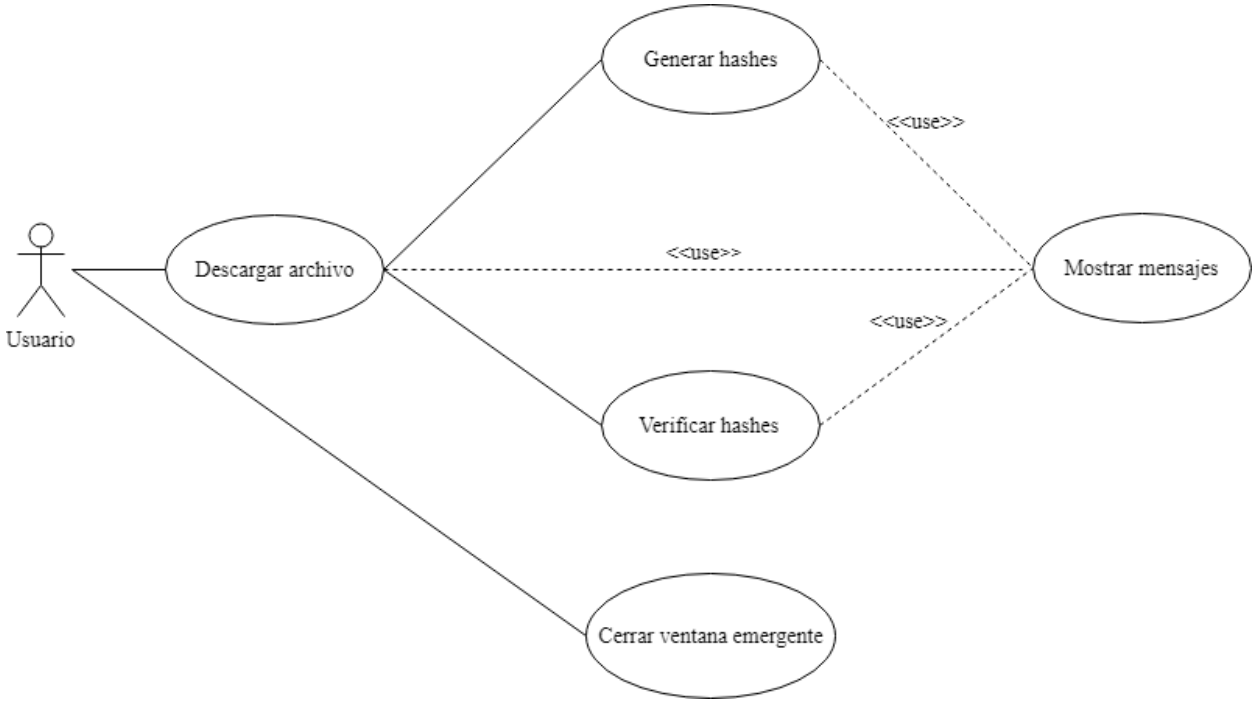


Figura 5.3. Diagrama de casos de uso.

5.3.1 Casos de uso alto nivel

En este apartado se muestra de manera más detallada los casos de uso expuestos en el apartado anterior. Para ello se va a hacer uso de la plantilla de la Tabla 5.1:

TABLA 5.1. PLANTILLA DE CASOS DE USO.

ID	Código de identificación único con el formato CUXX donde XX se corresponde con el número de caso de uso correspondiente.
Nombre	Título descriptivo.
Descripción	Descripción detallada de las funcionalidades y cómo y cuándo es invocado.
Actores	Aquellos que invocan e interactúan, pudiendo ser Usuario y/o Sistema.
Precondiciones	Condiciones previas para que se ejecute.
Postcondiciones	Resultado tras su ejecución
Flujo normal	Acción habitual o por defecto del sistema
Flujo alternativo	Acción o acciones que se dan cuando se produce una ejecución diferente al flujo normal del sistema.

TABLA 5.2. CASO DE USO CU01.

ID	CU01
Nombre	Descargar archivo.
Descripción	El caso de uso es invocado cuando el usuario descarga un archivo. Cuando la descarga finaliza el sistema verifica si la extensión pertenece a aquellas consideradas como válidas. En caso contrario finaliza la ejecución.
Actores	Usuario, Sistema.
Precondiciones	El usuario utiliza navegador Google Chrome.
Postcondiciones	Obtención de archivo descargado por parte del sistema.
Flujo normal	1. Obtener archivo de descarga. 2. Verificar extensión. 3. Extensión aceptada. 4. Enviar al encargado de mostrar mensajes que la descarga ha finalizado. 5. Se continua con ejecución del programa.
Flujo alternativo	1. Obtener archivo de descarga. 2. Verificar extensión.

	3. Extensión rechazada. 4. Finalización del programa.

TABLA 5.3. CASO DE USO CU02.

ID	CU02
Nombre	Generar hashes.
Descripción	El caso de uso es invocado cuando la descarga posee extensión válida. El sistema genera los hashes md5, sha1, sha256, sha512.
Actores	Sistema.
Precondiciones	Descarga verificada.
Postcondiciones	Generación de los 4 hashes.
Flujo normal	1. Enviar al encargado de mostrar mensajes que se están calculando los hashes. 2. Generar hash MD5. 3. Generar hash SHA1. 4. Generar hash SHA256. 5. Generar hash SHA512.
Flujo alternativo	1. Enviar al encargado de mostrar mensajes que se están calculando los hashes. 2. Se produce error al generar alguno de ellos. 3. Enviar al encargado de mostrar mensajes el error de cálculo hashes. 4. Finalización del programa.

TABLA 5.4. CASO DE USO CU03.

ID	CU03
Nombre	Verificar hashes.
Descripción	El caso de uso es invocado cuando se han generado los hashes del archivo descargado. El sistema comprueba si coincide alguno de estos en la página de descarga, o en alguna de las páginas referenciadas en esta.
Actores	Sistema.
Precondiciones	Hashes del archivo generados.
Postcondiciones	Verificación de la integridad finalizada.

Flujo normal	1. Enviar al encargado de mostrar mensajes que se están verificando los hashes. 2. Obtener HTML de la página de descarga. 3. Buscar si alguno de los hashes generados se encuentra en dicha página. 4. Se encuentran coincidencia. 5. Enviar al encargado de mostrar mensajes que se ha encontrado coincidencia entre los hashes y se ha verificado la integridad del archivo. 6. Finalización del programa.
Flujo alternativo	• Caso válido 1. Enviar al encargado de mostrar mensajes que se están verificando los hashes. 2. Obtener HTML de la página de descarga. 3. Buscar si alguno de los hashes generados se encuentra en dicha página. 4. No se encuentra coincidencia 5. Obtener todas las páginas referenciadas en la página de descarga. 6. Recorrer cada página y comprobar coincidencia. 7. Se encuentra coincidencia. 8. Enviar al encargado de mostrar mensajes que se ha encontrado coincidencia entre los hashes y se ha verificado la integridad del archivo. 9. Finalización del programa. • Caso no válido 1. Enviar al encargado de mostrar mensajes que se están verificando los hashes. 2. Obtener HTML de la página de descarga. 3. Buscar si alguno de los hashes generados se encuentra en dicha página. 4. No se encuentra coincidencia 5. Obtener todas las páginas referenciadas en la página de descarga. 6. Recorrer cada página y comprobar coincidencia. 7. No se encuentra coincidencia. 8. Enviar al encargado de mostrar mensajes que no se ha encontrado coincidencia entre los hashes y no se puede garantizar la integridad del archivo. 9. Finalización del programa.

TABLA 5.5. CASO DE USO CU04.

ID	CU04
Nombre	Mostrar mensajes.
Descripción	El caso de uso es invocado cuando el sistema recibe el mensaje correspondiente para realizar las funciones de mostrado de información.
Actores	Sistema.
Precondiciones	Recibe mensaje.
Postcondiciones	Mostrado de mensajes en la ventana emergente.
Flujo normal	1. Recibir mensaje. 2. Verificar que el mensaje corresponde con alguno de los que espera recibir. 3. El mensaje corresponde con alguno de los aceptados. 4. Realizar función necesaria. 5. Mostrar ventana emergente con mensajes. Finalización del programa.
Flujo alternativo	1. Recibir mensaje. 2. Verificar que el mensaje corresponde con alguno de los que espera recibir. 3. El mensaje no corresponde con ninguno de los aceptados. 4. Mostrar por consola que se ha recibido un mensaje desconocido. 5. Finalización del programa.

TABLA 5.6. CASO DE USO CU05.

ID	CU05
Nombre	Cerrar ventana emergente.
Descripción	El caso de uso es invocado cuando el sistema recibe la acción de que el usuario ha hecho seleccionado en el botón de cerrar la ventana emergente.
Actores	Usuario, Sistema.
Precondiciones	Se ha seleccionado botón de cerrar ventana emergente.

Postcondiciones	Cerrado de ventana emergente.
Flujo normal	1. Se muestra ventana emergente. 2. Seleccionar botón de cerrar ventana. 3. Se cierra la ventana. 4. Finalización del programa.
Flujo alternativo	- No contemplado.

5.4 Diagramas de secuencia

En esta sección se va a representar los diagramas de secuencia correspondientes a cada de uso. Para ello, se expone los diagramas oportunos para el flujo normal y el flujo alternativo del sistema, con una breve explicación de cada uno de ellos para hacer más sencilla su interpretación.

5.4.1 Descargar archivo (CU01)

- Flujo normal

Para el proceso de descarga de un archivo, el usuario lo inicia seleccionando una con una extensión válida reconocida por el sistema. En tal caso, Background es el encargado de comprobar dicha validez, y cuando termina esta verificación, envía el mensaje descargado" a Content para que este último muestre por pantalla la ventana emergente con el mensaje inicial. En la Figura 5.4 se puede observar dicho proceso.

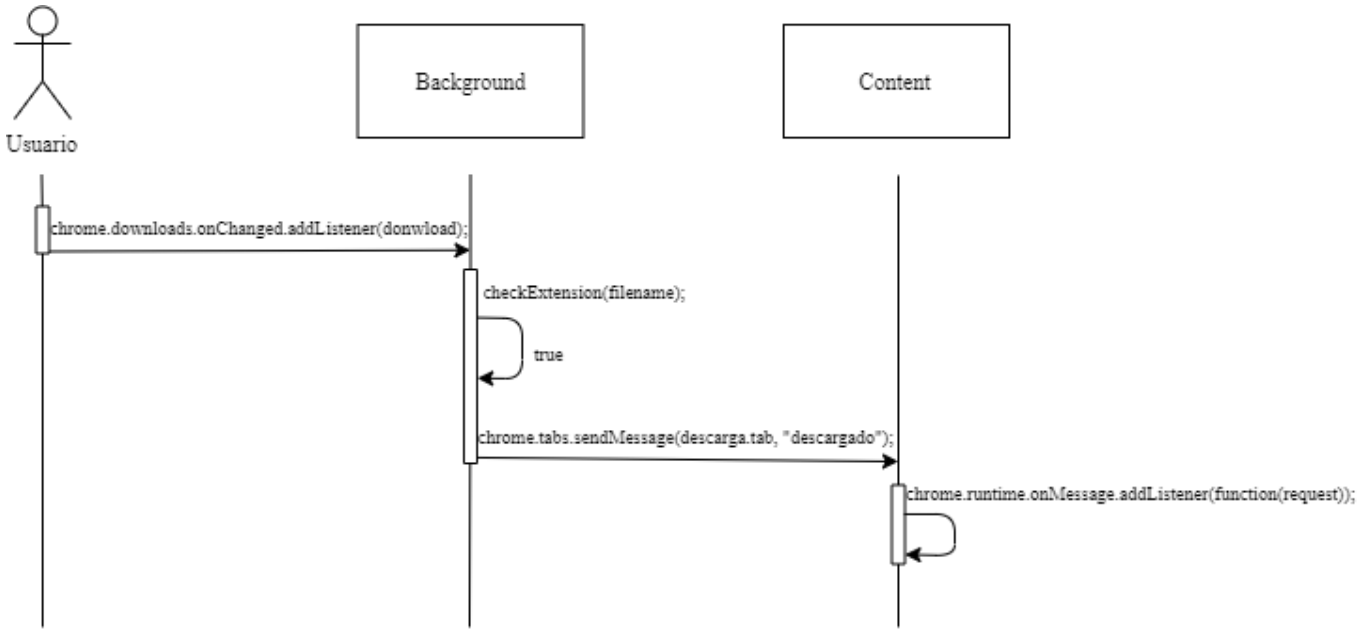


Figura 5.4. Diagrama de secuencia descargar archivo flujo normal.

- Flujo alternativo

En cambio, en la Figura 5.5, se puede ver como el usuario selecciona un archivo a descargar con una extensión no válida para el sistema. En esta situación, Background descarta el archivo finalizando así la ejecución del programa sin mostrar ningún mensaje.

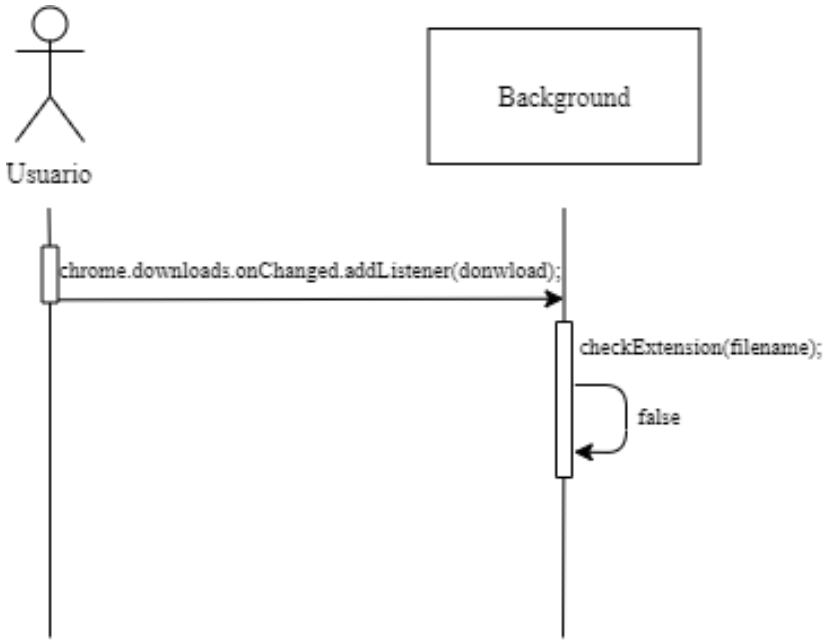


Figura 5.5. Diagrama de secuencia descargar archivo flujo alternativo.

5.4.2 Generar hashes (CU02)

- Flujo normal

El proceso de generar los hashes, mostrado en la Figura 5.6, comienza con los pasos anteriormente citados, en el que el usuario inicia una nueva descarga con una extensión aceptada verificada por Background, y mostrando el mensaje la ventana emergente con el mensaje inicial por parte de Content. Ahora Background se encarga de enviar a Content el mensaje “calculando” para que este muestre al usuario que comienza dicho proceso. Después, Background llama a las funciones correspondientes de las clases md5, sha1, sha256 y sha512 para generar los hashes del archivo descargado, finalizando aquí esta ejecución.

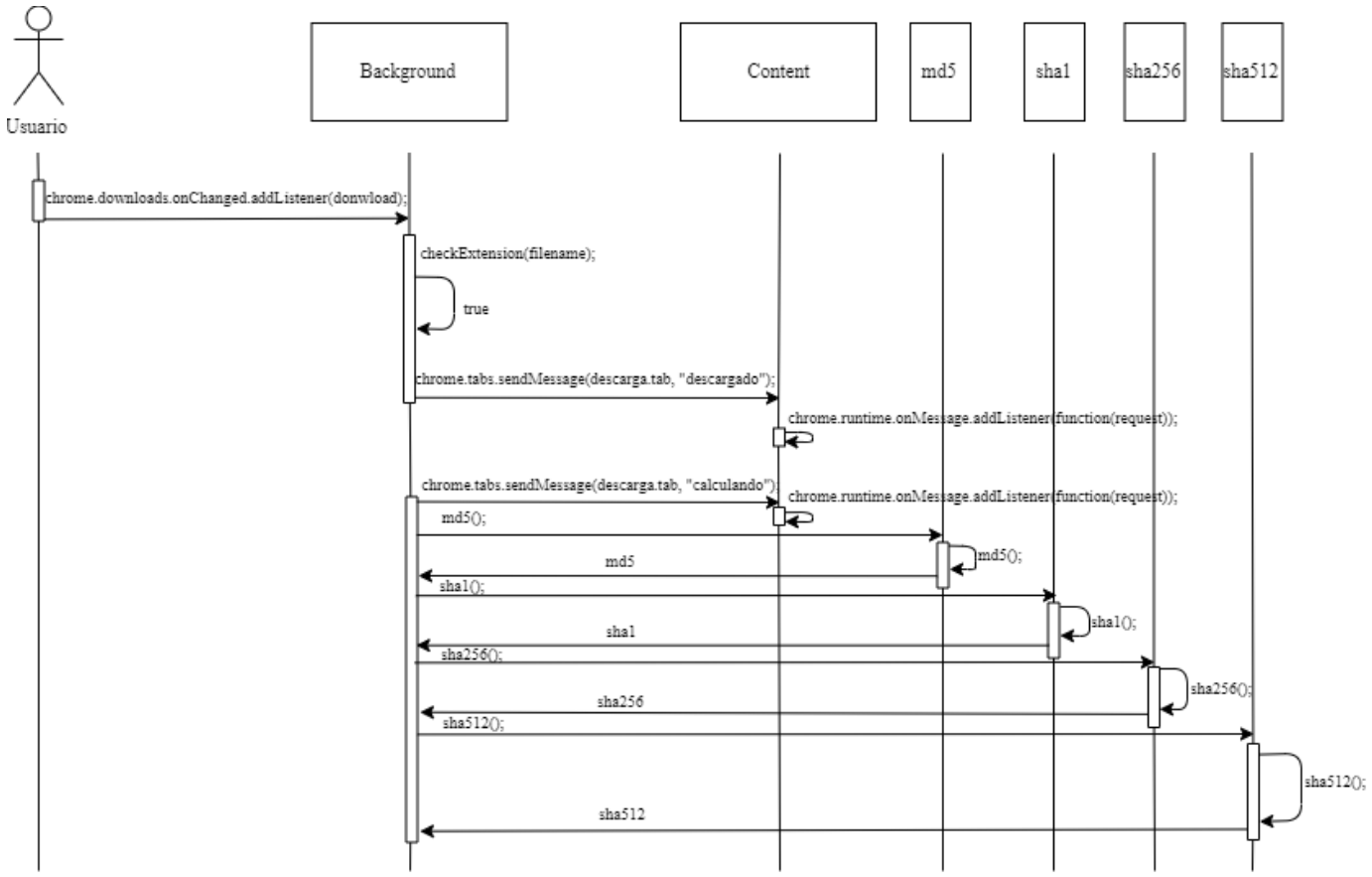


Figura 5.6. Diagrama de secuencia generar hashes flujo normal.

- Flujo alternativo

De igual manera que el proceso descrito en el flujo normal, pero se ha representado en la Figura 5.7 la acción en la que se produce algún error en el cálculo del hash md5, por lo que esta clase devuelve un error, y Background se lo comunica a Content enviando "error". Este último muestra al usuario que ha habido un error en el proceso de cálculo de hashes.

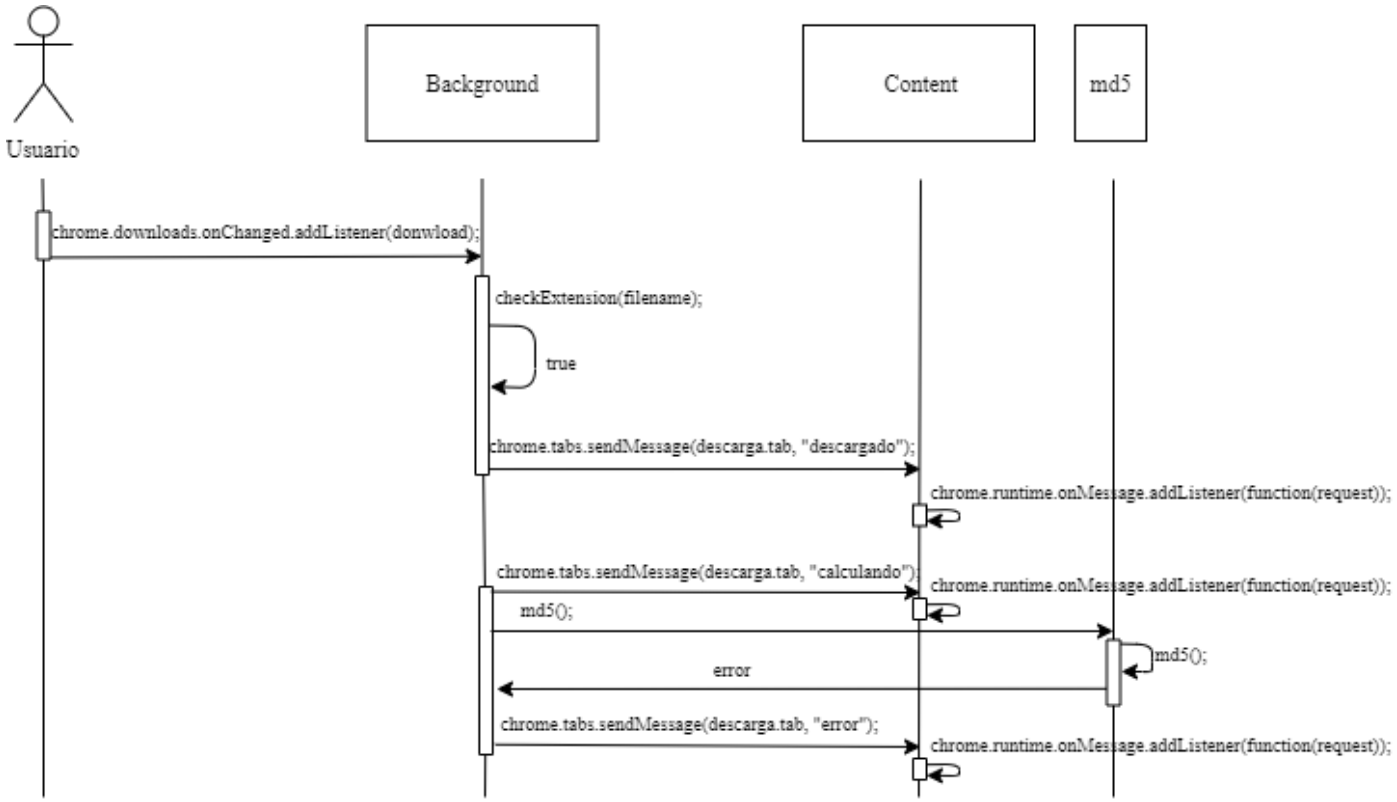


Figura 5.7. Diagrama de secuencia generar hashes flujo alternativo.

5.4.3 Verificar hashes (CU03)

- Flujo normal

Para el proceso de verificar la integridad del archivo descargado partimos del diagrama de secuencia de Generar hashes de la Figura 5.6 donde el sistema, Background, se encarga de buscar si alguno de los hashes generados anteriormente se encuentra en la página de descarga. En caso de ser cierto, envía el mensaje "verificado" a Content y esta muestra y finaliza la ejecución con el mensaje de verificación satisfactoria, mostrado en la Figura 5.8.

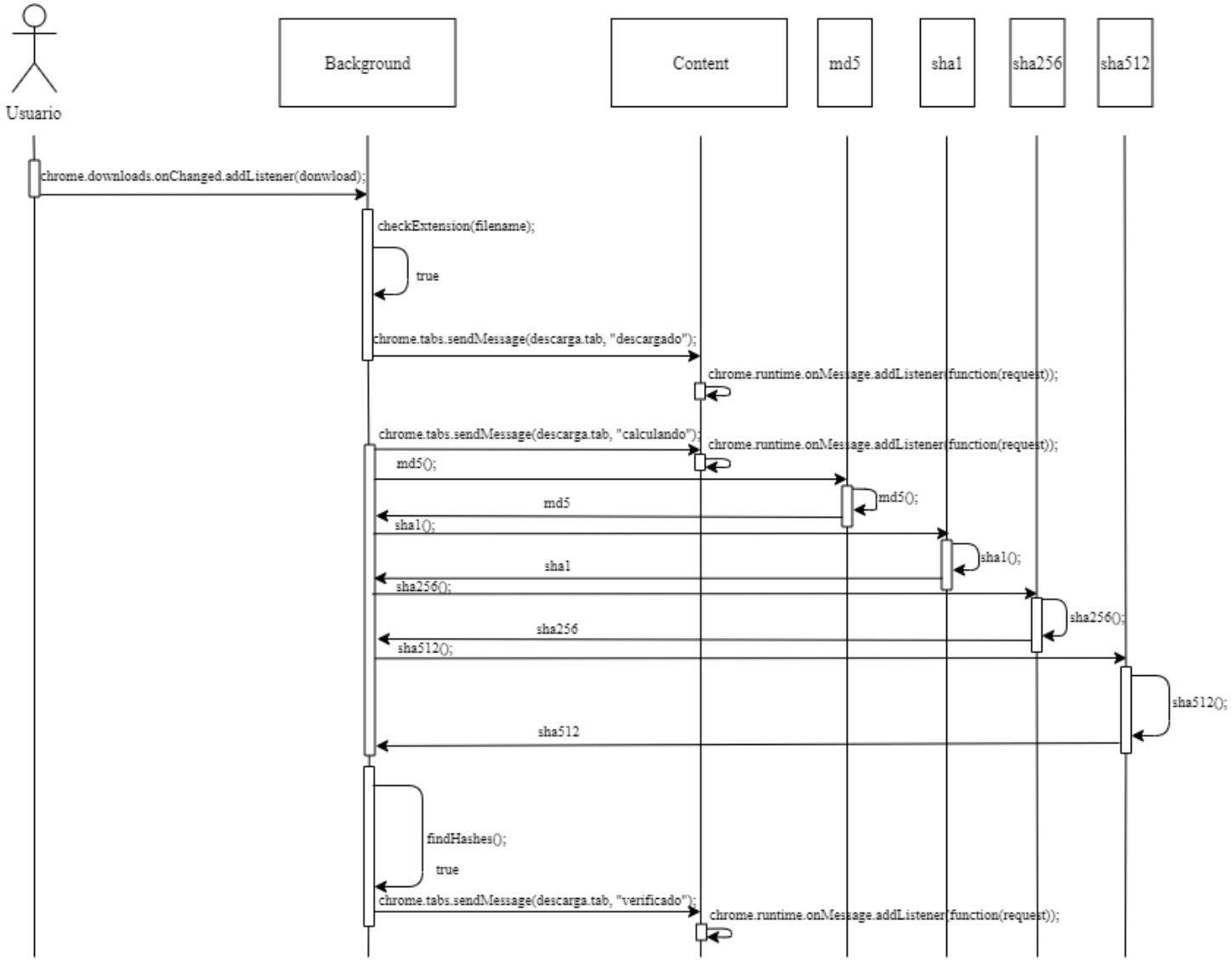


Figura 5.8. Diagrama de secuencia verificar hashes flujo normal.

- Flujo alternativo válido

El siguiente diagrama de la Figura 5.9 se corresponde con el caso de no haber encontrado alguno de los hashes generados en la página de descarga (mostrado en el flujo normal). Cuando esto sucede, Background envía el mensaje “noverificado” a Content y este se encarga de realizar un bucle con el que visita cada una de las páginas referenciadas en la página de descarga buscando si existe alguna coincidencia entre los hashes generados y el contenido de la página que se está analizando. En caso afirmativo Content muestra el mensaje de verificación satisfactoria y finaliza la ejecución.

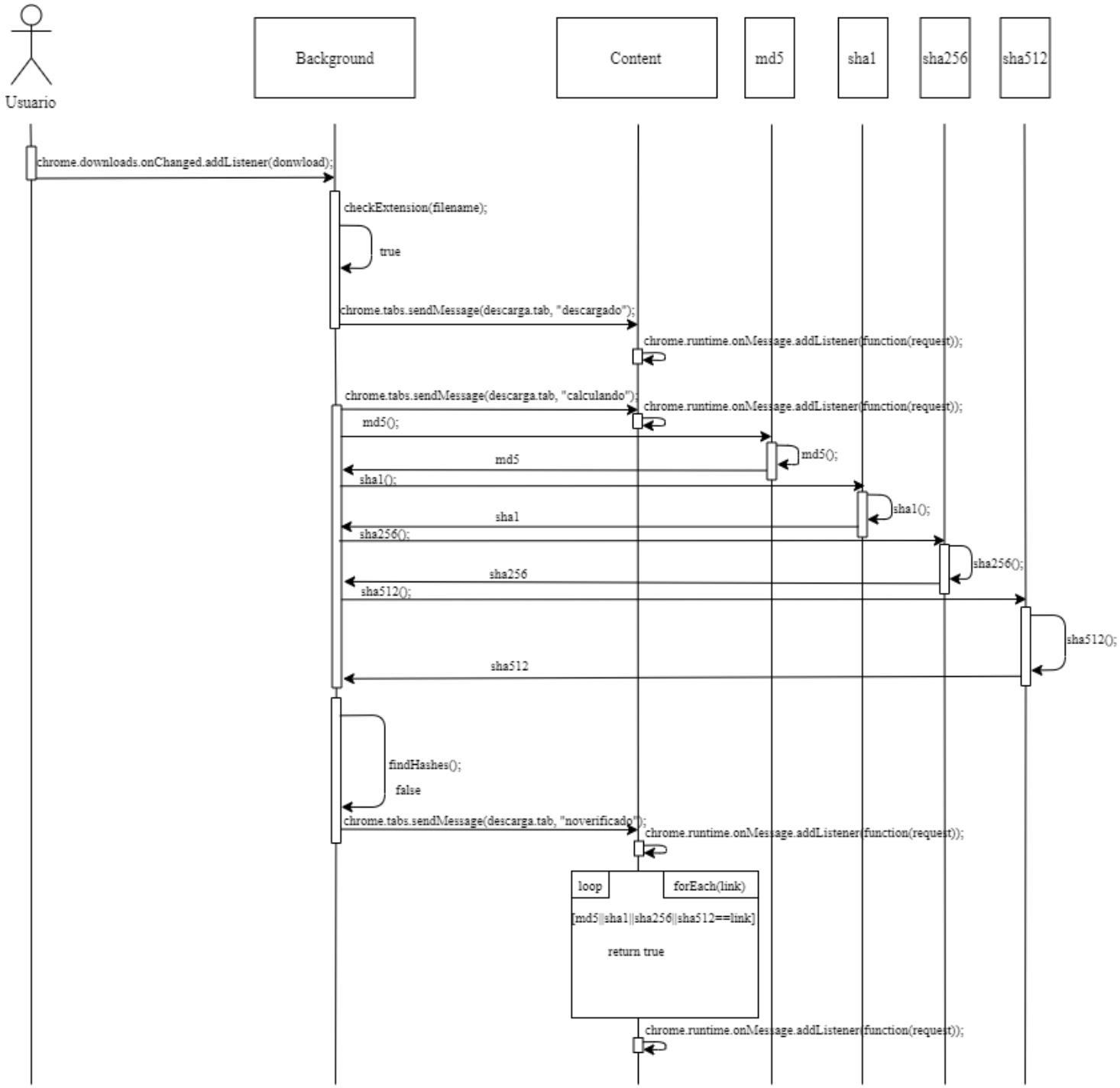


Figura 5.9. Diagrama de secuencia verificar hashes flujo alternativo válido.

- Flujo alternativo no válido

Esta situación es la misma que el flujo alternativo válido a diferencia que el bucle de comprobación de los hashes en las páginas referenciadas no encuentra coincidencia y por lo tanto Content muestra que no se ha podido verificar la integridad del archivo. Dicha acción se representa en la Figura 5.10.

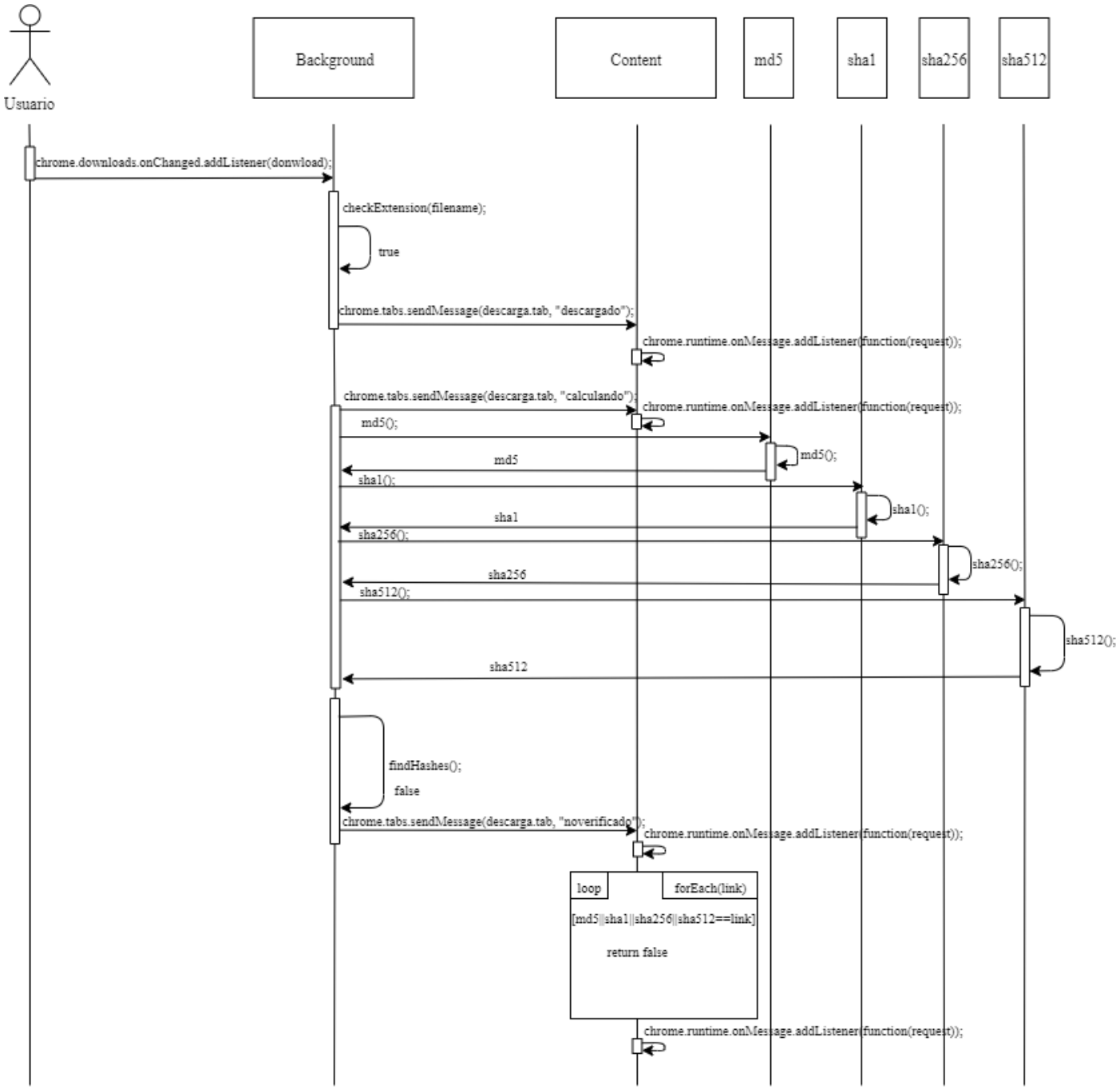


Figura 5.10. Diagrama de secuencia verificar hashes flujo alternativo no válido.

5.4.4 Mostar mensajes (CU04)

- Flujo normal

Para este diagrama valdría cualquiera de los diagramas de secuencia anteriores, por ejemplo, la Figura 5.8 en los que se hace uso de la función chrome.runtime.onMessage.addListener() de la clase Content para mostrar los mensajes correspondientes. Content recibiría un mensaje reconocido por parte de Background y se lo mostraría al usuario.

- Flujo alternativo

El diagrama de secuencia de la Figura 5.11 representa la acción en la que el usuario comienza con la descarga del archivo, se inicia la ejecución de la aplicación y Background envía a Content un mensaje "test" el cual desconoce, y Content al recibirlo imprimiría por consola que se corresponde con un mensaje desconocido.

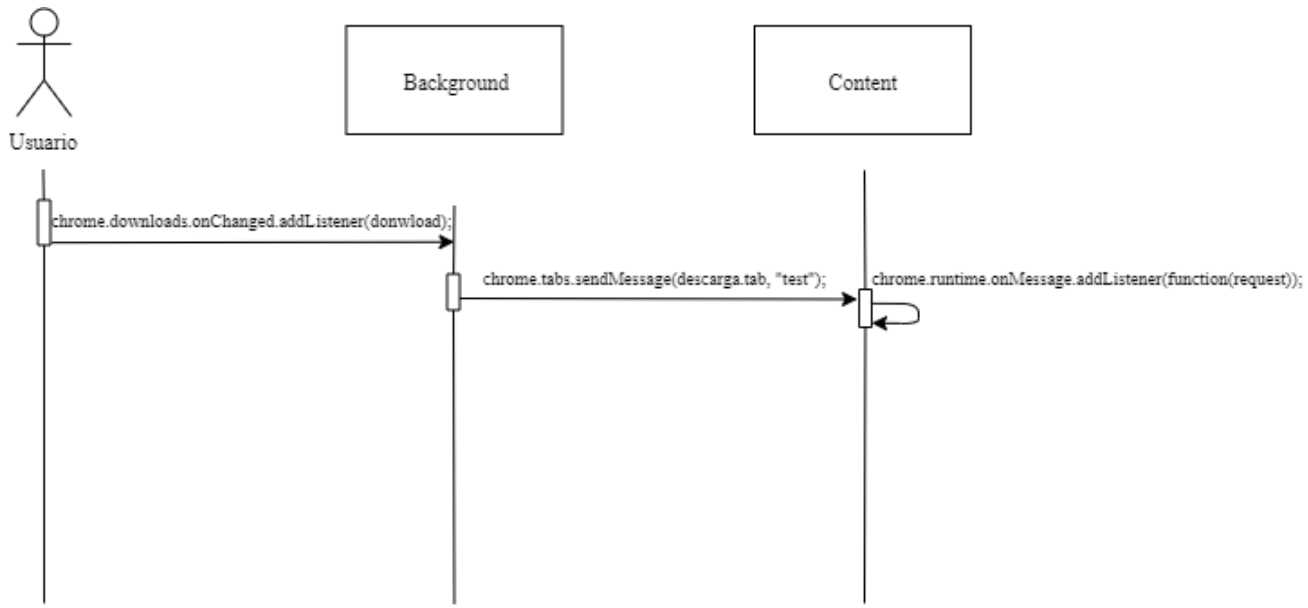


Figura 5.11. Diagrama de secuencia mostrar mensajes.

5.4.5 Cerrar ventana emergente (CU05)

- Flujo normal

El diagrama de la Figura 5.12 representa el proceso de cuando un usuario inicia una descarga y espera a que termine todo el proceso de verificación. En este caso se ha elegido la situación en el que se ha podido verificar la integridad satisfactoriamente en la página de descarga. Por último, el usuario selecciona el botón de cerrar ventana y es la clase Content quien recibe dicha acción y se encarga de ocultarla.

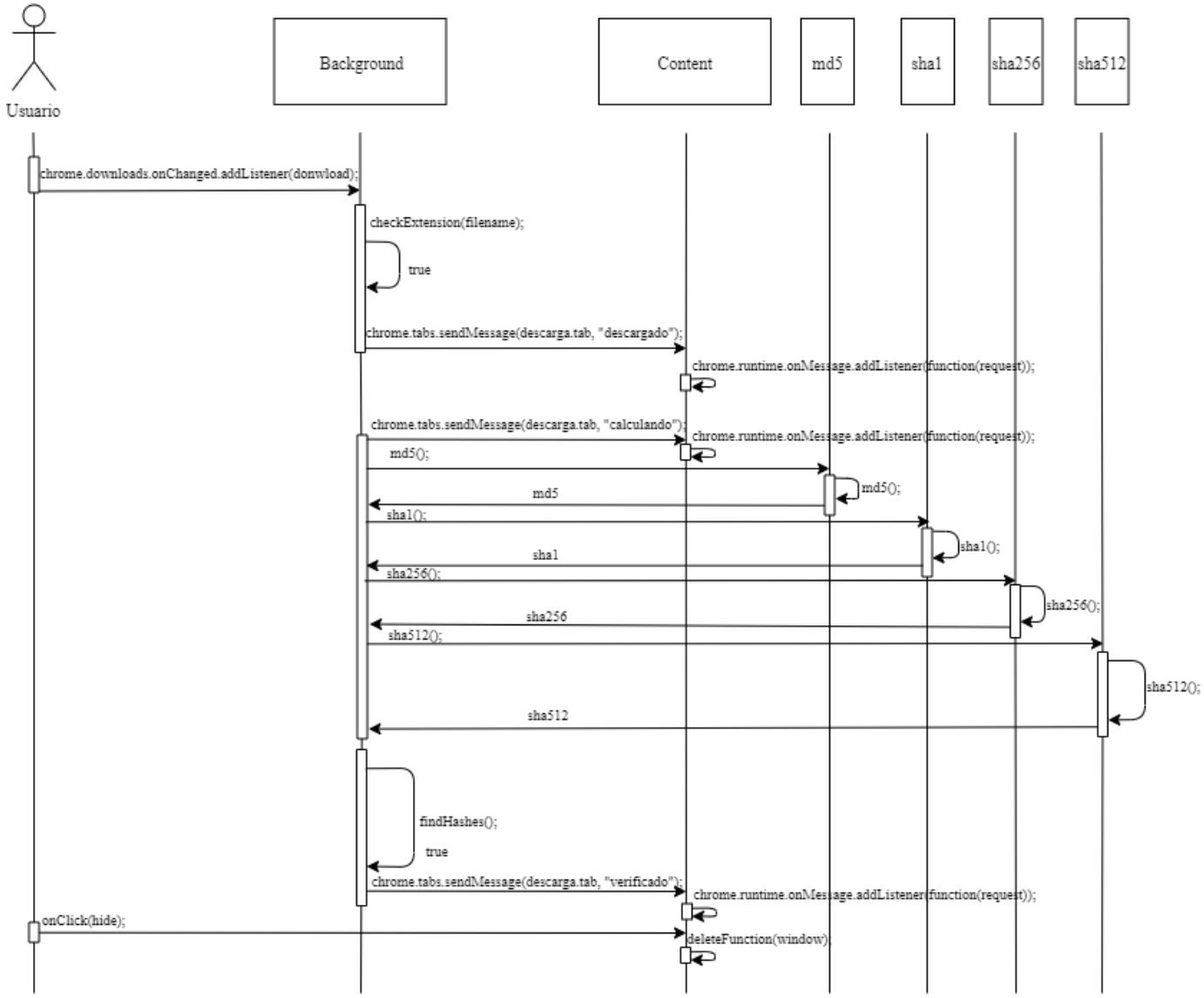


Figura 5.12. Diagrama de secuencia cerrar ventana emergente.

5.5 Diseño de la ventana emergente

Como el objetivo del sistema es alertar e informar al usuario tras una serie de comprobaciones, es importante definir de forma correcta la interfaz a mostrar.

En primer lugar, se decidió que los mensajes a mostrar fuesen con ventana emergente ya que así se asegura que el usuario dirige su atención tras la ejecución del programa. Posteriormente, se tiene que elegir correctamente los mensajes informativos teniendo en cuenta los siguientes puntos:

- La ventana emergente se debe mostrar en el centro de la pantalla.
- Resaltar la ventana emergente oscureciendo la ventana del navegador excepto el mensaje mostrado.
- Menor cantidad de mensajes posibles.
- Mensajes cortos.
- Interfaz agradable y atractiva.
- Utilizar iconos dedicando el color verde para caso satisfactorio y rojo para el contrario.

Los puntos anteriores se consideran relevantes ya que como se ha estudiado en la carrera principalmente en la asignatura de Interfaces de Usuario, es de gran importancia orientar la interfaz del sistema pensando cómo se puede hacer más usable, atractiva y sencilla para el usuario final.

Para el diseño de la interfaz se ha utilizado la herramienta Balsamiq [17] para realizar los distintos mockups que se van a exponer a continuación.

5.5.1 Mockup del proceso de cálculo y verificación

Este diseño se corresponde una vez el sistema ha comprobado que el archivo descargado contiene una extensión válida. En tal caso se muestra el mensaje inicial, en el que se describe que se ha detectado la descarga y que se mostrará el resultado una vez terminada la verificación. Para informar al usuario de que se está realizando un proceso se ha decidido añadir un nuevo mensaje en el que se dice que se están calculando y verificando los checksums con una rueda en movimiento (spinner en inglés) para que el usuario sepa que todavía no ha finalizado la ejecución. El resultado se muestra a continuación en la Figura 5.13.

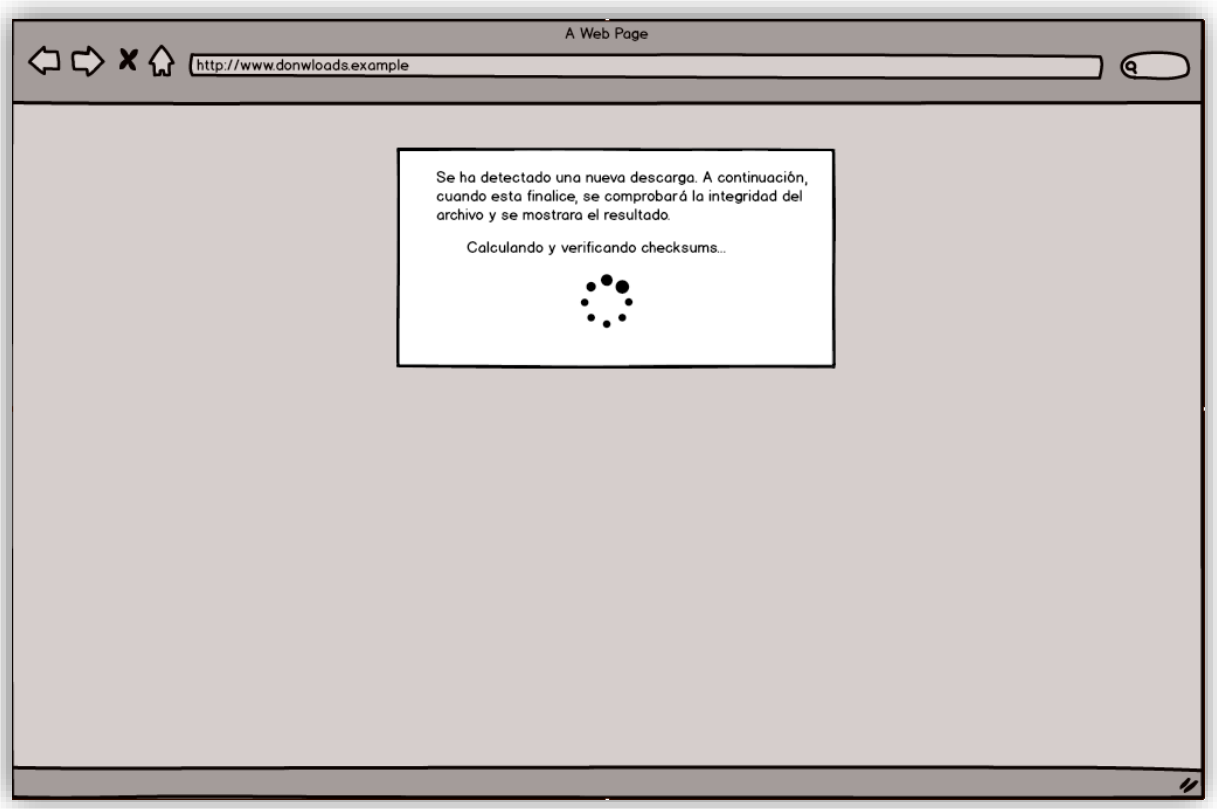


Figura 5.13. Mockup proceso de cálculo y verificación.

5.5.2 Mockup de verificación satisfactoria

En caso de que se haya encontrado alguna coincidencia entre los hashes generados y alguno de los proporcionados en la página de descarga o en alguno de los links referenciados, se muestra al usuario que el checksum coincide. Además, se añade el icono de tick con círculo verde lo que se reconoce fácilmente como algo satisfactorio o correcto. El resultado sería el mostrado en la Figura 5.14.

Figura 5.14. Mockup verificación satisfactoria.

5.5.3 Mockup de verificación desfavorable

Cuando no se ha encontrado ningún hash, o no coinciden con ninguno de los generados, se informará al usuario de que no ha podido ser verificada la integridad de la descarga. En tal caso se muestra el icono de una cruz roja haciendo referencia a algo incorrecto o erróneo para alertar de dicha situación al usuario. El resultado sería como el mostrado en la Figura 5.15.

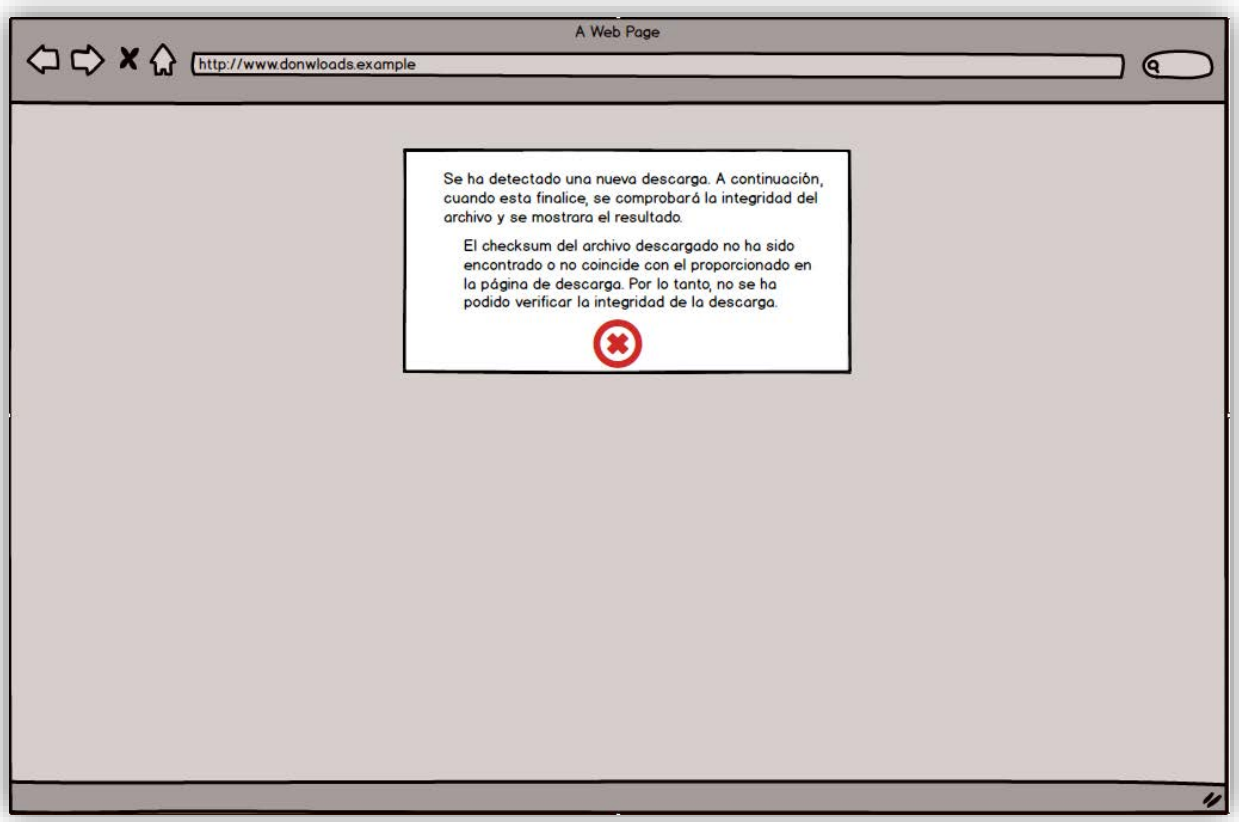


Figura 5.15. Verificación desfavorable.

6. IMPLEMENTACIÓN

6.1 Decisión de implementación

Para el desarrollo de la aplicación, como se mencionó anteriormente, la aplicación se ha desarrollado para ser compatible con Google Chrome. Por ello, ha habido que seguir las tecnologías impuestas por dicha plataforma (HTML, CSS y JavaScript). Además, se ha utilizado la API de Google Chrome para utilizar ciertas funcionalidades como obtener las descargas y obtener información de la ventana desde la que se realiza las acciones en el navegador.

Como información adicional, se ha desarrollado el software en Visual Studio Code, un editor de código fuente que ofrece muchas facilidades a la hora de programar. Esto fue decisión propia por la buena experiencia obtenida durante los años de carrera, pero se podría haber elegido cualquier otro editor de código incluso un editor de texto como, por ejemplo, Notepad++.

6.2 Explicación breve del código

Se ha considerado necesario el desarrollo de este punto para poder explicar más a bajo nivel la lógica y el funcionamiento de la aplicación. Aun así, se va a exponer solamente aquellas funciones que se consideran más importantes.

A grandes rasgos, la aplicación desarrollada cuenta con un fichero de configuración llamado *manifest.json* que se encuentra en la raíz del programa. En la raíz también se encuentran los iconos con distintas resoluciones con las que se quiere dar la imagen del sistema. El proyecto posee las siguientes carpetas:

- css: contiene el archivo *style.css* donde se definen los estilos de la ventana emergente.
- fonts: contiene *fontawesome.css* y *otherfont.css*. El primero importa una serie de clases utilizada para los iconos de la ventana emergente y el segundo para definir el tipo de letra que se muestra.
- scripts: aquí se encuentran los archivos que aportan la lógica al sistema. En primer lugar, el conjunto de clases *md5.js*, *sha1.js*, *sha256* y *sha512* para la generación

de hashes. También *background.js* y *content.js* los cuales se van a explicar más con más detalle posteriormente.

Como se ha estado mencionado en los apartados anteriores de este documento, la lógica de la aplicación reside principalmente en los 2 archivos *background.js* y *content.js*. Las funciones principales de cada uno de ellos son las siguientes:

- Background
  - chrome.downloads.onChanged.addListener(): recibe como argumento el archivo de descarga. Esta es la función principal de esta clase. En primer lugar, llama a una función para comprobar si la extensión del archivo es válida. En caso afirmativo almacena datos de interés de la descarga, y genera los hashes MD5, SHA1, SHA256 Y SHA512. Una vez terminado este proceso llama a la función encargada de verificar si alguno de estos se encuentra en la página de descarga. En caso negativo, envía el mensaje correspondiente a la clase Content para que haga la comprobación en las páginas referenciadas. También se encarga de enviar los mensajes oportunos explicados anteriormente en la finalización de cada función.
  - checkExtension(): recibe el nombre de un archivo y comprueba si la extensión de este corresponde con las consideradas como válidas.
  - findHashes(): se encarga de comprar si alguno de los hashes generados anteriormente se encuentra en la página de descarga del archivo. Devuelve “true” en caso afirmativo.

- Content
  - chrome.runtime.onMessage.addListener(): la función principal de esta clase, encargada de recibir los mensajes de la clase Background. Los mensajes reconocidos son “descargado”, “calculando”, “verificado”, “noverificado”, y “error”. Cada tipo de mensaje que recibe muestra en la ventana emergente la información oportuna.
  - findHashesInOtherPages(): recibe la descarga, y obtiene cada página web referenciada desde la página de descarga para verificar si alguno de los hashes generados anteriormente se encuentra en alguna de estas páginas. Devuelve true en caso afirmativo.

6.3 Aspectos adicionales

Es importante aclarar una serie de aspectos acerca de la aplicación. Como se ha citado en el apartado anterior, existe en el proyecto el archivo de configuración *manifest.json*. Para el correcto funcionamiento de la aplicación ha sido necesario definir en este archivo una serie de permisos que se pueden observar en la Figura 6.1.

```
{
    "name": "Verificador de Checksum",
    "version": "1.0",
    "manifest_version": 2,
    "description": "Aplicación para verificar la integridad de la descarga de archivos.",
    "icons": {
       "16": "icon16.png",
       "48": "icon48.png",
       "128": "icon128.png"
            },
    "background": {
        "persistent": false,
        "scripts": [
            "scripts/md5.js",
            "scripts/sha1.js",
            "scripts/sha256.js",
            "scripts/sha512.js",
            "scripts/background.js"
            ]
        },
    "content_scripts": [{
        "matches": ["<all_urls>"],
        "css": ["css/style.css", "fonts/otherfont.css", "fonts/fontawesome.css"],
        "js": ["scripts/content.js"]
        }],
    "permissions": [
        "tabs",
        "downloads",
        "downloads.open",
        "http://*/*", "https://*/*", "file://*"
    ],
    "web_accessible_resources": [
        "css/*",
        "scripts/*",
        "fonts/*",
        "webfonts/*"]
}
```

Figura 6.1. Archivo de configuración manifest.json.

Los permisos son declarados en el apartado “permissions”. El primero, “tabs”, permite obtener información de las ventanas abiertas en el navegador. El siguiente, “downloads”, permite obtener información y gestionar las descargas, con la ayuda de “downloads.open” que ofrece el permiso de poder abrirlas. Por último, se define que se tiene acceso a páginas que utilizan protocolo http y https, y también a la ruta de file://* lo que corresponde al acceso de archivos del sistema del usuario.

La aplicación ha sido dotada de los permisos citados anteriormente con el único objetivo de cumplir la comprobación de la integridad. Por ello, estos permisos no son utilizados para obtener información privada del usuario ya que los datos tratados son la obtención del archivo y generación de los correspondientes hashes, sin manipular ni obtener más información acerca de ellos.

7. EVALUACIÓN DEL SISTEMA

Con el objetivo de mostrar el funcionamiento del sistema, en este apartado se va a exponer 3 casos del resultado de la ejecución de la aplicación. En primer lugar, se realizará la descarga de una aplicación que ofrezca el checksum de verificación en la misma página en la que se realiza la descarga. En segundo lugar, será el caso de que dicho checksum se encuentre la alguna página externa, es decir, en alguna que se referencie desde la página de descarga. Por último, se va a mostrar el resultado que muestra el sistema cuando se descargue una aplicación que no ofrezca checksum de verificación.

7.1 Checksum en página de descarga

Para esta demostración se va a descargar la aplicación VLC, un reproductor multimedia. Dicho software se obtiene en el siguiente enlace: https://www.videolan.org/vlc/index.es.html

Cuando la descarga finaliza, se muestra la ventana emergente de la Figura 7.1 en la que se muestra que se están generando y verificando los hashes.

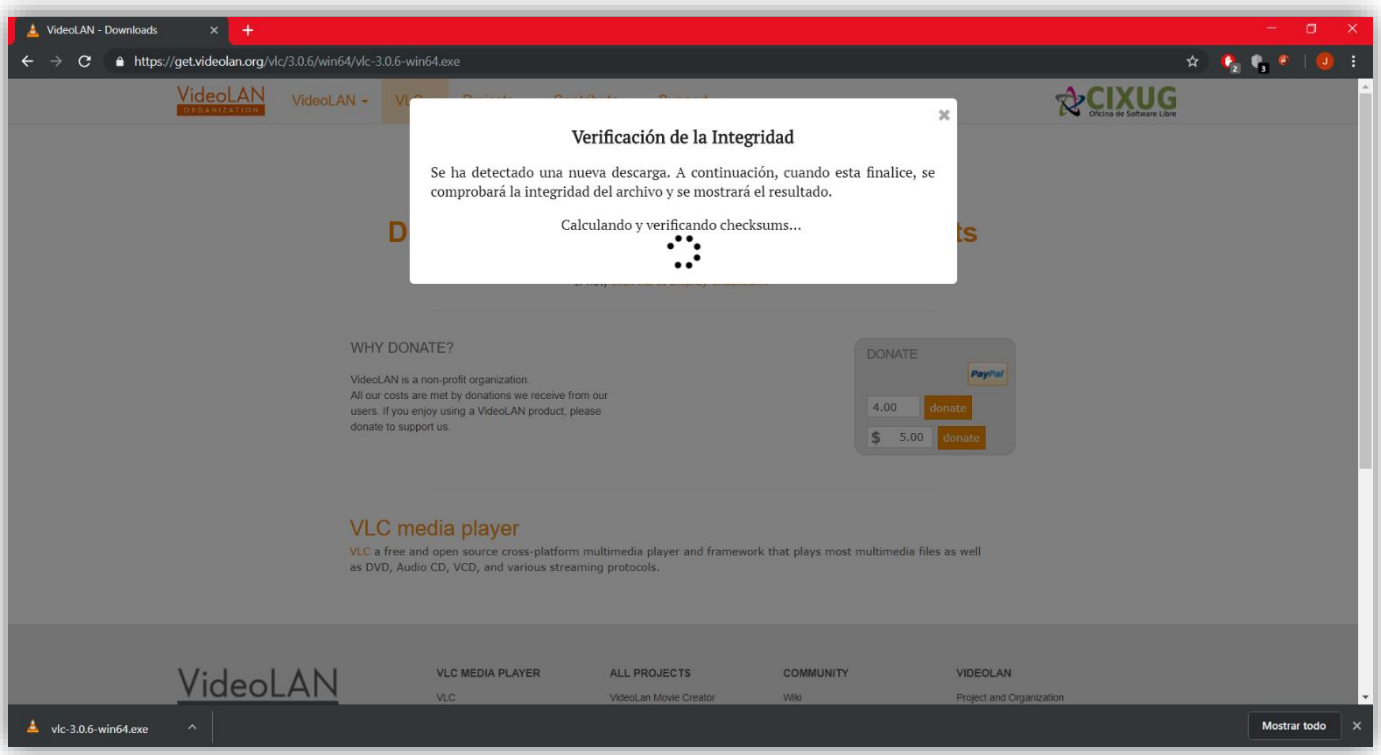


Figura 7.1. Inicio de verificación de VLC.

Cuando dicho proceso termina, se muestra en la Figura 7.2 el resultado de la verificación satisfactoria del archivo.

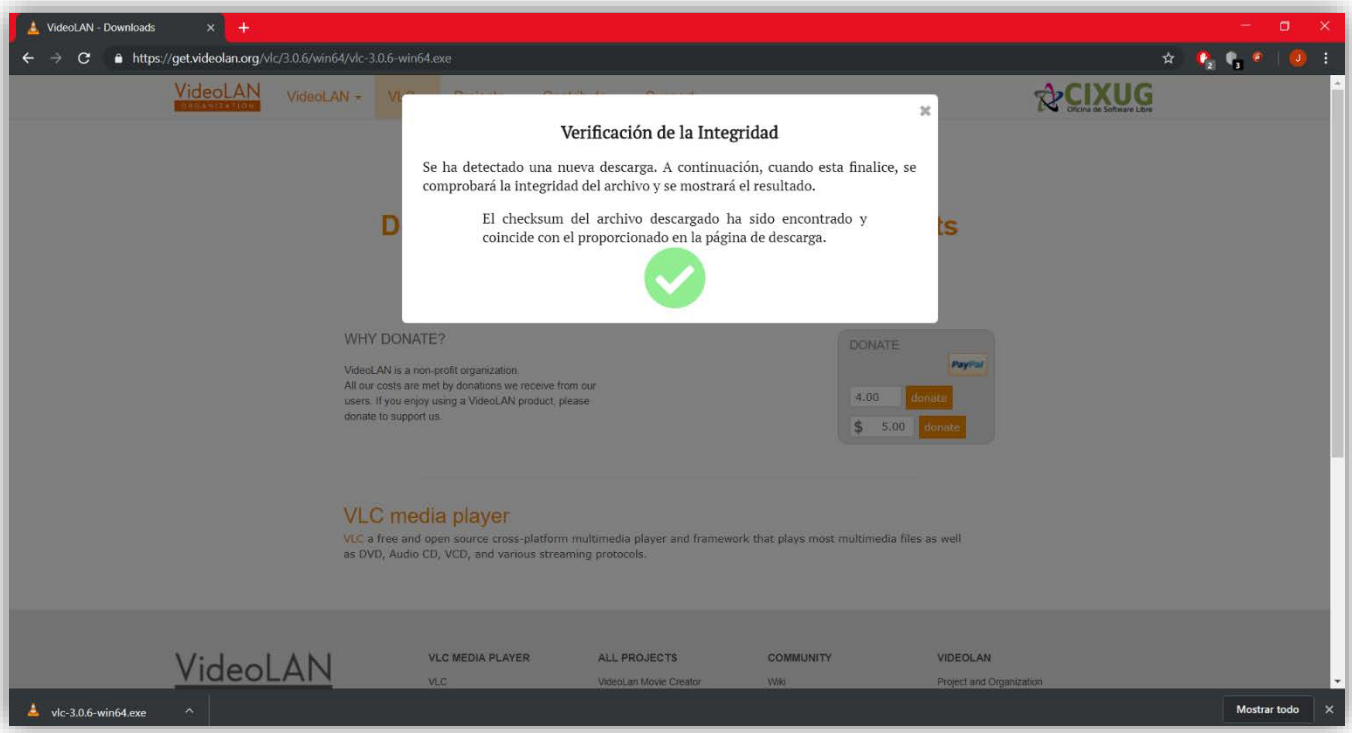


Figura 7.2. Resultado de verificación de VLC.

7.2 Checksum en página externa

En este segundo escenario se ha elegido VirtualBox, aplicación para virtualización. El archivo de descarga se obtiene de: https://www.virtualbox.org/wiki/Downloads

De igual modo que antes, se muestra el mensaje de la Figura 7.3 al usuario, cuando la descarga ha finalizado, en el que se le indica que se está generado y calculando los hashes.

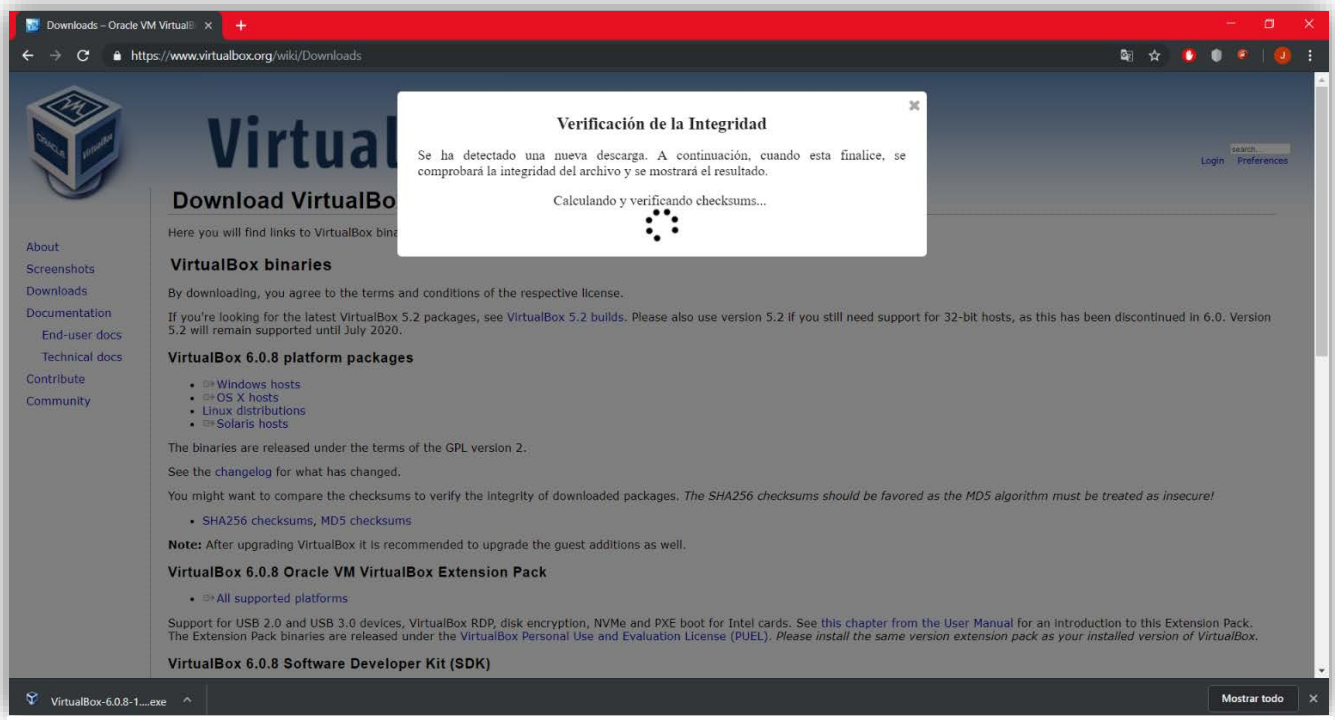


Figura 7.3. Inicio de verificación de VirtualBox.

Como se tiene que acceder a todos los links de la página hasta que encuentra coincidencia entre alguno de los hashes generados y los ofrecidos en alguno de los links de la página de descarga, la ejecución tarda unos segundos más. Finalmente, se muestra la verificación satisfactoria de la descarga en la Figura 7.4.

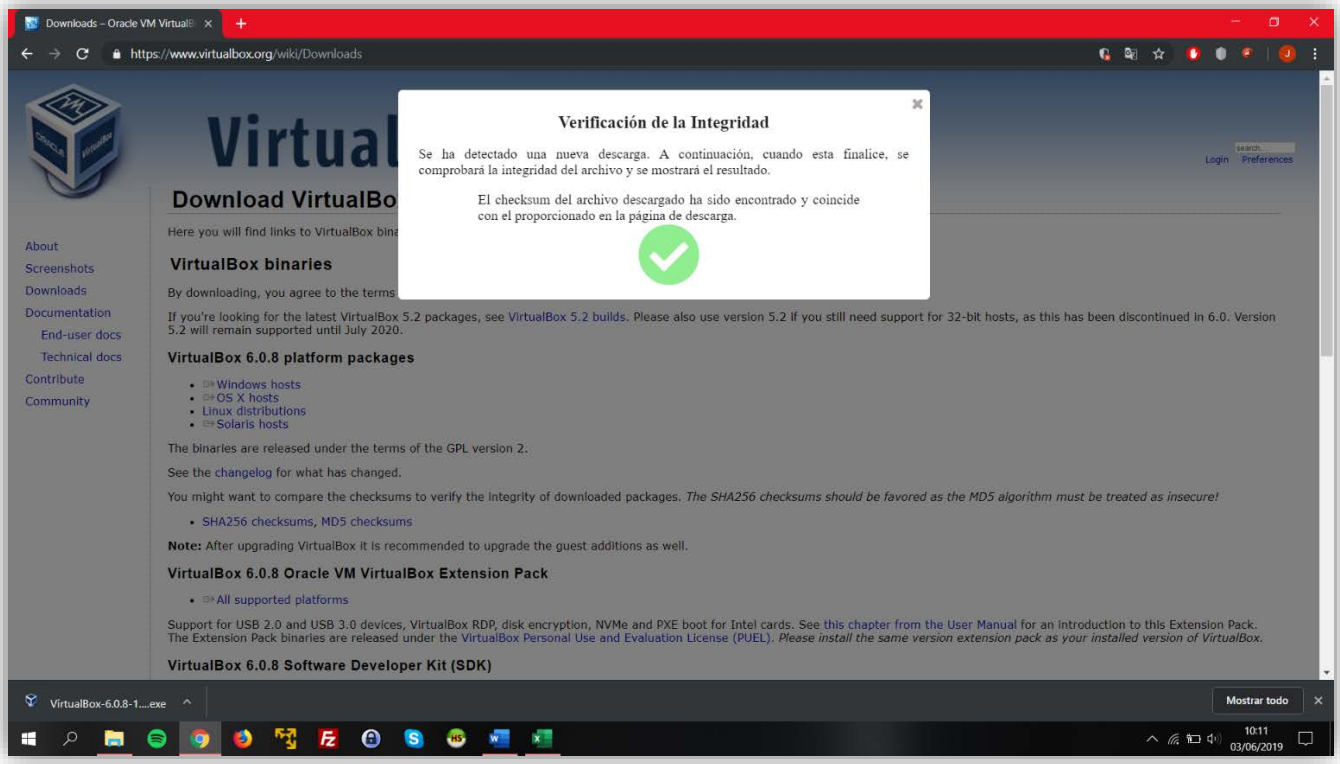


Figura 7.4. Resultado de verificación VirtualBox.

7.3 Checksum no encontrado

Por último, se va a descargar el antivirus Avast. Dicho software no ofrece checksum de verificación, y se puede obtener en: https://www.avast.com

Como en casos anteriores, se muestra el mensaje inicial en la Figura 7.5.

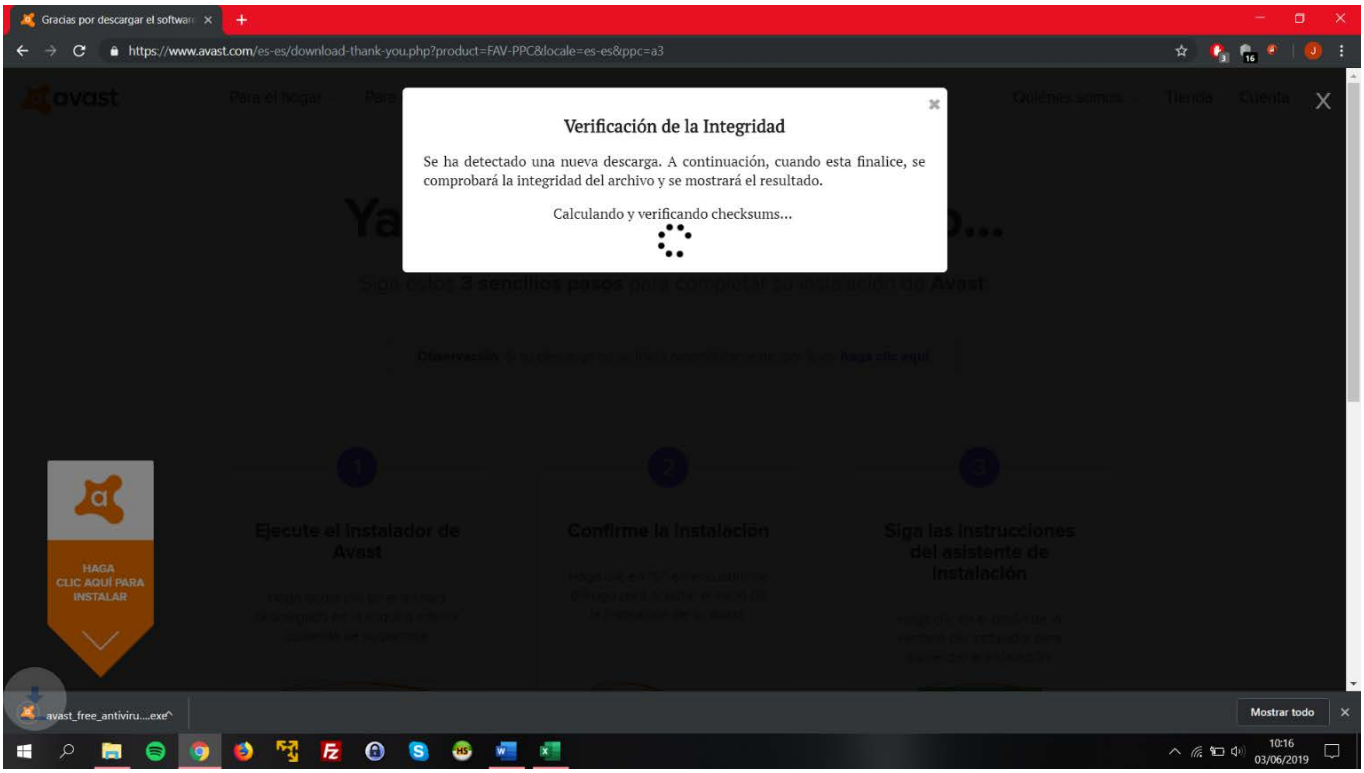


Figura 7.5. Inicio de verificación de Avast.

Al no encontrar ningún checksum, ni en la página de descarga, ni en páginas externas referenciadas desde de esta, se muestra en la Figura 7.6 el resultado de que no se ha podido verificar la integridad del archivo descargado.

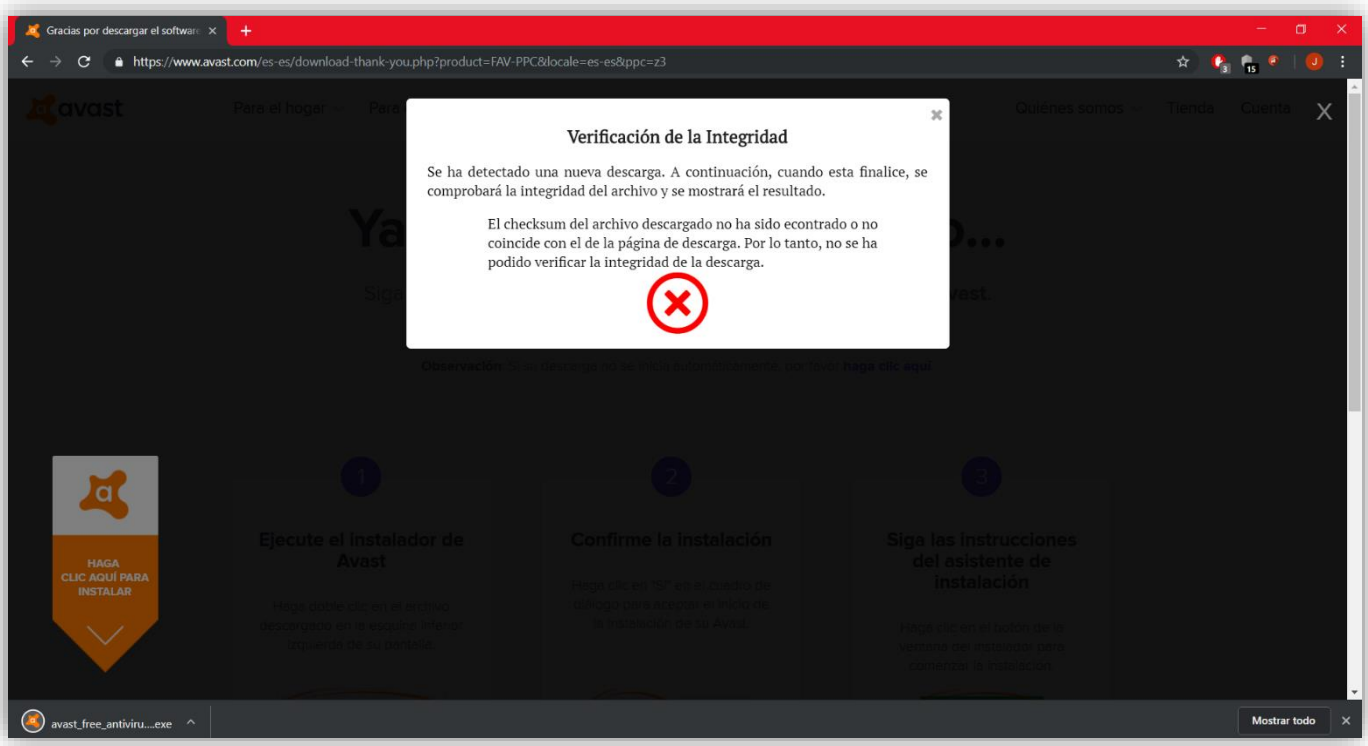


Figura 7.6. Resultado de verificación de Avast.

8. PRUEBAS DE ACEPTACIÓN

Una vez se han definido las funcionalidades y características expuestas en los apartados 5 y 6 (Análisis y Diseño), en esta sección se va a realizar el plan de pruebas para determinar si el sistema finalmente cumple con los objetivos propuestos inicialmente.

La Tabla 8.1 contiene el plan de pruebas, formada por los siguientes campos:

- ID: identificador único con formato PXX donde XX se corresponde con el número de prueba correspondiente.
- Requisitos implicados: aquellos que son probados o forman parte del proceso.
- Descripción: objetivo e idea general que se desea comprobar. Se detalla también la salida deseada tras la realización de la prueba.
- Salida obtenida: resultado que muestra la aplicación tras ser ejecutada.
- Superada: Sí, en caso de haber superado la prueba, No, en caso contrario.

TABLA 8.1. PLAN DE PRUEBAS.

ID	Requisitos implicados	Descripción	Salida obtenida	Superada
P01	RF01 RF02 RF06	Descargar un archivo con extensión válida. El sistema debe aceptarlo, y mostrar la ventana emergente con mensaje de que se ha detectado una nueva descarga.	Se muestra la ventana emergente indicando que se ha detectado nueva descarga.	Sí
P02	RF01 RF02	Descargar un archivo con extensión no válida. El sistema debe rechazarlo y no mostrar la ventana emergente.	No se muestra la ventana emergente.	Sí
P03	RF01 RF02 RF03 RF06	Descargar un archivo con extensión válida y generar los hashes. El sistema debe mostrar la ventana emergente con el mensaje de calculando hashes.	Se muestra la ventana emergente con el mensaje de calculando hashes.	Sí
P04	RF01 RF02 RF03 RF04 RF05 RF06	Descargar un archivo con extensión válida que posea el checksum en la página de descarga. El sistema debe mostrar la ventana emergente con el mensaje de que el checksum generado coincide con el de la página de descarga.	Se muestra la ventana emergente con el mensaje de que el checksum coincide con el de la página de descarga.	Sí

P05	RF01 RF02 RF03 RF04 RF05 RF06	Descargar un archivo con extensión válida que no posea checksum para verificar. El sistema debe mostrar la ventana emergente con el mensaje de que checksum del archivo no se ha encontrado o no coincide con el de la página de descarga.	Se muestra la ventana emergente con el mensaje de que el checksum del archivo no se ha encontrado o no coincide con el de la página de descarga.	Sí
P06	RF01 RF02 RF03 RF04 RF05 RF06	Descargar un archivo con extensión válida que posea el checksum en una página externa a la página de descarga del archivo. El sistema debe mostrar la ventana emergente con el mensaje de que el checksum generado coincide con el de la página de descarga.	Se muestra la ventana emergente con el mensaje de que el checksum coincide con el de la página de descarga.	Sí
P07	RF01 RF02 RF06 RF07	Descargar un archivo con extensión válida y presionar botón para cerrar ventana emergente según aparezca. El sistema debe cerrar la ventana y no volver a mostrarla.	Se cierra la ventana, pero vuelve a mostrarse cuando tiene que mostrar un mensaje nuevo.	No
P08	RF01 RF02 RF03 RF04 RF05 RF06 RF07	Descargar un archivo con extensión válida y presionar botón para cerrar ventana emergente al finalizar el programa. El sistema debe cerrar la ventana y no volver a mostrarla.	Se cierra la venta y no vuelve a mostrarse.	Sí

Como se puede observar en la Tabla 8.1, la columna de requisitos implicados es acumulativa, es decir, al ir probando las funcionalidades de la aplicación de menos a más, se van implicando más requisitos con carácter acumulativo pues es un proceso incremental del requisito RF01 al RF05. El requisito RF06 (mostrar ventana emergente) empieza a realizar su función a partir del requisito RF02, a la vez que el requisito RF07 (cerrar ventana emergente) podrá ser ejecutado una vez que el requisito de mostrar ventana emergente (RF06) haya sido ejecutado. De esta tabla el dato más importante es la última columna “Superada”, en la que se indica si la prueba responde correctamente a la funcionalidad propuesta. Por ello, se puede concluir que de todas las pruebas planteadas todas han sido superadas a excepción de la prueba P07, que se corresponde con la prueba de cerrar ventana cuando programa no ha finalizado. Lo que debería ocurrir es que la ventana se cerrase y no volviese a parecer. En cambio, al cerrarla, cuando el sistema tiene que mostrar un mensaje nuevo vuelve a aparecer la ventana.

9. PLANIFICACIÓN Y ENTORNO SOCIO-ECONÓMICO

9.1 Planificación del proyecto

Esta sección tiene como objetivo principal establecer un plan inicial de asignación de tareas para una mejor organización y desarrollo del proyecto. Finalmente, cuando este se termine, se comparará con la planificación real que se ha llevado a cabo y así analizar las desviaciones del proyecto.

9.1.1 Planificación inicial

Con el propósito de cumplir todas las etapas del desarrollo del proyecto se ha considerado necesario hacer un estudio previo de la asignación del tiempo estimado que se va a dedicar a cada etapa. Para ello, en primer lugar, se va a determinar qué duración va a requerir cada tarea en función de lo que se estima según su complejidad y tamaño. Finalmente, para una mejor organización y visualización, se representará dicha asignación de cada fase en un diagrama de Gantt.

TABLA 9.1. PLANIFICACIÓN INICIAL.

Tarea	Días	Inicio	Fin
Planificación	3	11-Feb	13-Feb
Introducción	4	14-Feb	17-Feb
Estado del arte	14	18-Feb	03-Mar
Análisis	17	04-Mar	20-Mar
Diseño	29	21-Mar	18-Apr
Implementación	32	19-Apr	20-May
Evaluación del sistema	1	21-May	22-May
Pruebas de aceptación	10	23-May	01-Jun
Conceptos previos	2	02-Jun	03-Jun
Conclusiones	4	04-Jun	07-Jun
Trabajo Fin de Grado	116	11-Feb	07-Jun

La Tabla 9.1 está compuesta por las tareas a realizar, seguido de la duración en días estimados junto con la fecha de inicio y fin de cada una. Para la diferenciación de tareas se han representado cada una de ella con un color distinto (que se corresponde con el

color en el diagrama de Gantt). Como se puede observar, la planificación inicial del proyecto tiene una duración de 116 días, finalizando el día 7 de Junio. Con esta estimación se terminaría el proyecto 10 días antes de la fecha de entrega oficial (máximo el 17 de Junio), por lo que es bastante positivo en cuanto a que se dispone de un cierto margen en caso de sufrir algunos imprevistos en el desarrollo del proyecto. De igual modo, si se finalizase en la fecha estimada, se contaría con 10 días para el mantenimiento y revisión del proyecto lo cual sería resultaría bastante beneficioso. Por último, se ha estimado que las fases de mayor duración son el Estado del Arte, Análisis, Diseño y la Implementación. Estas tareas son consideradas la base del proyecto ya que abarca desde la fase de estudio del problema hasta su implementación, por lo que la gran mayoría del esfuerzo estarán enfocadas en estas.

A continuación, en la Figura 9.1 se muestra el diagrama de Gantt que representa la Tabla 9.1 explicada anteriormente. Se puede observar cómo se irá desarrollando cada tarea de manera secuencial según finaliza la anterior. Para ello, se ha establecido en el eje horizontal superior la secuencia de las fechas con diferencia de 10 días. Además, para facilitar la interpretación, se han añadido entre las líneas principales (líneas verticales naranjas) líneas secundarias (líneas verticales grises) para delimitar cada día entre las fechas distintas fechas.

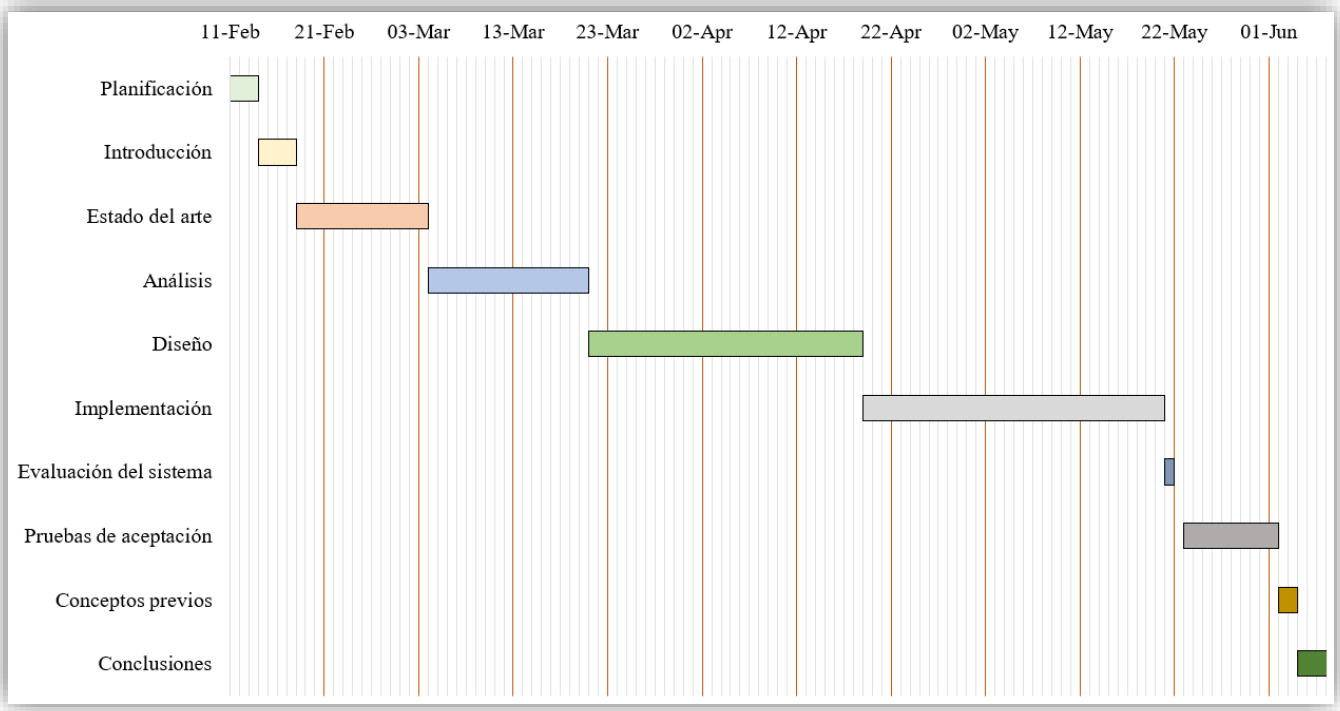


Figura 9.1. Diagrama de Gantt de planificación inicial.

9.1.2 Planificación real

A continuación, se muestra el desarrollo real que se ha llevado a cabo para la realización del proyecto. La Tabla 9.2 representa la asignación real, con su duración en días y fechas correspondientes. A diferencia de la planificación inicial, se han repetido tareas como se explicó en la metodología puesto que en el desarrollo del proyecto se encuentran novedades y cambios que se consideran importantes y han de ser actualizados. Estas han sido Introducción, Análisis, Diseño e Implementación.

Se han utilizado la misma gama de colores para diferenciar las distintas tareas del proyecto.

TABLA 9.2. PLANIFICACION REAL.

Tarea	Días	Inicio	Fin
Planificación	3	11-Feb	13-Feb
Introducción	2	14-Feb	15-Feb
Estado del arte	21	16-Feb	08-Mar
Análisis	14	09-Mar	22-Mar
Diseño	16	23-Mar	07-Apr
Implementación	19	08-Apr	26-Apr
Análisis	3	27-Apr	29-Apr
Diseño	6	30-Apr	05-May
Implementación	6	06-May	11-May
Evaluación del sistema	1	12-May	12-May
Pruebas de aceptación	8	13-May	20-May
Introducción	2	21-May	22-May
Conceptos previos	2	23-May	24-May
Conclusiones	4	25-May	28-May
Trabajo Fin de Grado	107	11-Feb	28-May

Como se puede observar, la duración real final del proyecto ha sido de 107 días, 9 días menos que lo estimado, finalizando el 28 de Mayo.

El diagrama de Gantt de la planificación real se muestra a continuación en la Figura 9.2, con el mismo formato que el diagrama de Gantt de la planificación inicial en la Figura 9.2.

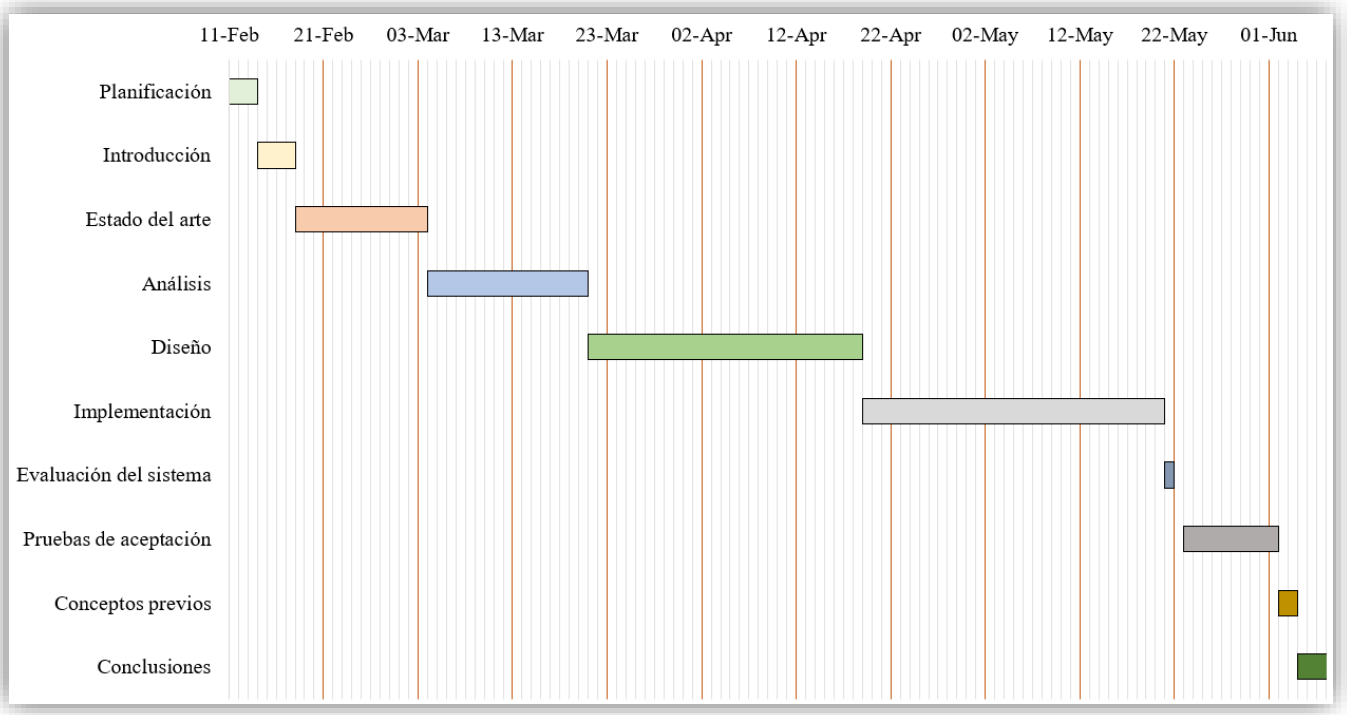


Figura 9.2. Diagrama de Gantt de planificación real.

9.1.3 Comparación de las planificaciones

Como en los apartados anteriores se han realizado las planificaciones iniciales y reales, ahora es necesario estudiar cuáles han sido las diferencias entre ambas para así resaltar las desviaciones y poder concretar algunas causas de estas y tenerlas en cuenta en futuros proyectos.

La Tabla 9.3 compara ambas planificaciones en las que se tiene en cuenta la duración en días de cada tarea en ambos casos, y mostrando la diferencia junto a la variación que esta supone. No se ha tenido en cuenta para esta comparación la repetición de tareas que se ha llevado a cabo en la planificación real, sino el conjunto total de días. Analizando los resultados, en primer lugar, el Estado del Arte ha requerido de 1 semana más de trabajo en comparación con lo estimado, en un total de 21 días. Tanto en las tareas de Diseño como la de Implementación se han invertido 7 días menos en cada una de ellas. Por último, se invirtió en las Pruebas de aceptación del sistema 8 días en lugar de 10.

Las desviaciones del proyecto son fruto de que se consideró que algunas tareas requerían más tiempo del estimado y finalmente estas fueron resueltas con mayor sencillez que lo pensado, como lo son el Diseño y la Implementación. Esta última tarea mencionada, la

Implementación, necesitó menos tiempo debido a que se estaba familiarizado con las tecnologías usadas y esto en un principio no se sabía ya que faltaba la fase de estudio previo y el Análisis para saber las tecnologías que se iban a usar. En cambio, el Estado del Arte requirió de unos días más para su finalización, ya que algunos de los programas que se probaron no eran lo esperado y se trató de invertir el mayor número de horas necesarias en este apartado para prevenir posibles errores por desconocimiento en etapas futuras.

En conclusión, donde mejor se puede observar el resultado y comparación final de ambas planificaciones es en la diferencia en la última fila en la que se muestra que la duración del proyecto se estimaba 116 días y, finalmente, se invirtieron 107, es decir, 9 días menos. Esto resulta bastante beneficioso ya que si, por el contrario, se hubiese dado el escenario de que se hubiese requerido de 9 días adicionales para la finalización del proyecto, se habría apurado demasiado a la fecha límite de entrega. Por lo tanto, estos días adicionales de margen se han podido invertir en pequeñas correcciones y mantenimiento del documento.

TABLA 9.3. COMPARACIÓN DE PLANIFICACIONES.

Tarea	Inicial	Real	Diferencia	Variación
Planificación	3	3	0	0.00%
Introducción	4	4	0	0.00%
Estado del arte	14	21	7	33.33%
Análisis	17	17	0	0.00%
Diseño	29	22	-7	-31.82%
Implementación	32	25	-7	-28.00%
Evaluación del sistema	1	1	0	0.00%
Pruebas de aceptación	10	8	-2	-25.00%
Conceptos previos	2	2	0	0.00%
Conclusiones	4	4	0	0.00%
Trabajo Fin de Grado	116	107	-9	**-8.41%**

9.2 Entorno socio-económico

A continuación, se va a detallar el presupuesto de la elaboración del proyecto y el impacto socio-económico que se espera resultado del proyecto.

9.2.1 Presupuesto

Los costes que se van a calcular a continuación no tienen incluido el IVA, por lo que será añadido al final en la suma de todos los costes.

- Costes derivados del capital humano

En primer lugar, como se ha detallado en el apartado de planificación, se han dedicado 107 días para la elaboración del proyecto. Se estima que de media se han podido invertir unas 4 horas diarias, lo que daría un total de 428 horas. Por otro lado, el salario percibido por hora de trabajo sería de 20€. Siguiendo la siguiente fórmula:

$$Coste = Horas_{totales} \; x \; Salario_{porHora} = 8.560€$$

Esta cifra es considerada el importe bruto a lo que habría que añadirle el 21,30% para el pago de la Seguridad Social, dando así un total de 10.382,28€.

- Costes derivados de los equipos informáticos

En este caso se tiene en cuenta los equipos tecnológicos utilizados para el desarrollo del proyecto, tanto para la investigación como para la implementación. Estos han sido el teléfono móvil, el portátil personal y 2 monitores para trabajar con más pantallas. En la Tabla 9.4 se muestran los costes estimados para la utilización de los terminales con el precio medio aproximado actual en el mercado. El tiempo de uso está expresado en meses, y se considera que todos los equipos han sido usados durante los 5 meses del desarrollo del proyecto. El tiempo de depreciación también está representado en meses, siendo este el tiempo que se estima que vayan a durar estos equipos de cara al futuro. Se ha estimado que el ordenador portátil va a durar 5 años (60 meses), al igual que el monitor principal. El teléfono móvil solamente 2 años (24 meses) mientras que el monitor secundario 3 años (36 meses). Por lo tanto, el coste total imputable es de 241,847€.

TABLA 9.4. COSTES DE EQUIPOS INFORMÁTICOS.

Equipo	Descripción	Coste	Uso	Tiempo depreciación	Coste imputable
Ordenador portátil: Lenovo Ideapad 520-ikb	· Procesador Intel Core I7 Octava Generación · 8 GB de memoria RAM DDR4 · 256 GB de memoria SSD SATA · 2 GB tarjeta gráfica NVIDIA GEFORCE MX150	849 €	5	60	70.750 €
Teléfono móvil: Huawei P20 Pro clt-l09	· Procesador Kirin 970 · 128 GB de almacenamiento · 6 GB de memoria RAM	559 €	5	24	116.458 €
Monitor principal: LG 34UC79G-B	· 34 pulgadas · 144 Hz LED IPS · Resolución 2560 x 1080	489 €	5	60	40.750 €
Monitor secundario: ACER 246HL	· 24 pulgadas · 60 Hz LCD LED · Resolución 1920 x 1080	100 €	5	36	13.889 €
				Coste total imputable	**241.847 €**

- Costes derivados del uso del software

En la Tabla 9.5 se muestra el resumen de los gastos requeridos del uso del software para el desarrollo del proyecto. Como se puede observar no ha habido costes porque el sistema operativo Windows 10 Home ya va venía incluido en la compra del ordenador portátil. Además, Microsoft Office ha sido utilizado con la licencia proporcionada por la Universidad Carlos III de Madrid. El resto del software es gratuito.

TABLA 9.5. COSTES DE SOFTWARE.

Producto	Coste
Windows 10 Home	0 €
Visual Studio Code	0 €
Google Chrome	0 €
Microsoft Office	0 €
Draw.io	0 €
HTML	0 €
JavaScript	0 €
CSS	0 €
Total	0 €

- Costes derivados del entorno de trabajo

Para este caso, como se ha desarrollado el proyecto desde casa, los gastos han sido de electricidad e Internet. Estos han sido calculados de la siguiente manera:

- Electricidad

Se ha calculado la potencia de cada equipo informático por hora. Posteriormente, se ha multiplicado por el número de horas totales, y por el precio del kilovatio (KW, que es 0,143889), obtenido del plan estable de Iberdrola [18]. La fórmula utilizada para el cálculo de la potencia ha sido la siguiente: *P(Potencia) = V(Tensión) x I(Intensidad)*.

- Internet

La opción que se ha considerado más coherente ha sido la de contratar una tarifa móvil con datos para hacer uso de Internet y utilizar este como Wifi. Esto es debido a que la contratación de fibra óptica en la mayoría de las compañías telefónicas requiere una permanencia de 12 meses, costando al mes unos 40€, dando lugar a un total de 480€ anuales. En cambio, el plan que se ha elegido ha sido la contratación de una línea móvil con la compañía O2 [19] que cuesta 20€ mensuales con IVA incluido y ofrece 20GB de navegación sin contrato de permanencia, con lo que se podría pagar mes a mes durante los 5 meses del proyecto.

En la Tabla 9.6 y 9.7 se puede ver el desglose de cada coste explicado anteriormente, dando así lugar a un coste total entre ambos cargos de 110,462€.

TABLA 9.6. COSTES DE ELECTRICIDAD.

Equipo Informático	Potencia	Horas	Precio KW	Coste
Ordenador portátil	65	428	0.143889	4,003 €
LG 34UC79G-B	64.98	428	0.143889	4,002 €
ACER 246HL	39.9	428	0.143889	2,457 €
			Total	**10,462 €**

TABLA 9.7. COSTE DE INTERNET.

Compañía	Datos móviles	Precio	Meses	Coste
O2	20 GB	20 €	5	**100 €**

- Presupuesto final

A continuación, se muestra el presupuesto final a entregar al cliente por el desarrollo del proyecto. Para ello, se han agrupado todos los costes calculados en los apartados anteriores, sumándole un 10% del riesgo por imprevistos que se puedan dar de cara al futuro. Este valor ha sido estimado de acuerdo a los valores e interpretaciones utilizadas durante la carrera en proyectos similares. El beneficio estimado a obtener para este proyecto es del 12%, de igual modo que se ha aprendido durante la carrera con unos valores similares. Por último, se añade el 18% correspondiente del IVA ya que este no se ha incluido en el desglose de cada apartado anterior.

Reuniendo el conjunto de gastos totales del proyecto el presupuesto final que se entrega al cliente se muestra en la Tabla 9.8, dando lugar a un coste total del proyecto de 15.605,516€.

TABLA 9.8. PRESUPUESTO FINAL.

Concepto	Coste
Salario	10,382.280 €
Equipos informáticos	241.847 €
Software	0.000 €
Entorno de trabajo	110.462 €
Total Coste sin riesgo	**10,734.589 €**
Riesgo (10%)	1,073.459 €
Total Coste sin beneficios	**11,808.048 €**
Beneficios (12%)	1,416.97 €
Total costes sin IVA	**13,225.014 €**
IVA (18%)	2,380.50 €
Total	**15,605.516 €**

9.2.2 Impacto socio-ecomómico

La aplicación desarrollada tiene 2 pilares fundamentales: verificar la integridad de los archivos que los usuarios descargan desde el navegador web y, además, realizar este proceso de manera transparente, es decir, sin ser necesario la interacción al usuario. Esto produce el siguiente resultado:

- Se proporciona un nivel más de seguridad en la descarga de aplicaciones verificando su integridad.
- A medida que avance el tiempo la aplicación obtendrá más valor ya que a medida que pasan los días los desarrolladores tratan de proporcionar más seguridad a sus aplicaciones por lo que comenzarán a incluir el checksum para ser verificadas. De igual modo, más usuarios serán consciente de la importancia de comprobar aquello que se descargan.
- Se aumentará el número de usuarios que puedan realizar dicha verificación puesto que no habrá que tener conocimientos técnicos sobre qué hay que comprobar ni dónde ya que este proceso lo realiza la aplicación por el usuario.
- El uso de esta aplicación podrá evitar la instalación en algunos casos de programas malignos por lo que se ofrecerá una capa más de seguridad en los ordenadores de los usuarios y evitar comprometer su información personal.
- El impacto medioambiental es nulo ya que no se hace uso materiales naturales y el uso de la aplicación no afecta negativamente al ambiente.

10. MARCO REGULADOR

La aplicación que se ha desarrollado es una extensión compatible para Google Chrome. Dicha aplicación tiene permisos para obtener las descargas que realiza el usuario cuando tiene activada esta aplicación en el navegador. Además, cuenta con el permiso correspondiente para acceder a la ubicación del archivo en el ordenador del usuario. Esto es necesario para obtener el archivo y poder comprobar el tipo que es, y así poder realizar la verificación de la integridad del mismo. Por lo tanto, todo usuario que haga uso del navegador de Google Chrome está expuesto y acepta las Condiciones de Servicio de Google Chrome. Estas se pueden encontrar en https://www.google.com/intl/es-419/chrome/privacy/eula_text.html .

Además, los archivos obtenidos que los usuarios descargan están regidos por la *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales* por lo que no se hace un uso indebido de ellos o diferente al establecido en este documento.

Para el desarrollo de este proyecto se han aceptado los términos y condiciones de los distintos softwares y tecnologías utilizados. El software utilizado ha sido el siguiente:

- Windows 10 Home: sistema operativo sobre el que se ha trabajado para el desarrollo de la aplicación y del documento actual. Para ello, se cuenta con una licencia proporcionada con la compra del ordenador portátil utilizado.
- Visual Studio Code: editor de código fuente desarrollado por la empresa Microsoft. La versión utilizada ha sido la 1.33.1 gratuita para el público.
- Google Chrome: navegador web de código cerrado y disponible gratuitamente.
- Word y Excel: 2 productos de Microsoft Office utilizados para la realización del documentos y creación de algunas tablas, gráficos y diagramas. Estos programas han sido obtenidos a través de la licencia académica proporcionada por la Universidad Carlos III de Madrid.
- Draw.io: herramienta utilizada para la creación de diagramas en el apartado de Análisis. Es gratuita y está disponible a través de la web.
- HTML, JavaScript y CSS: las 3 tecnologías utilizadas para el desarrollo de la aplicación. Todas ellas están ligadas al Consorcio WWW (W3C), organización dedicada a la estandarización de las tecnologías dedicadas a la web. Para la

utilización de estas tecnologías no se ha seguido ningún estándar puesto que el desarrollo de la aplicación no va dirigida a un grupo de trabajo o usuarios.

Se ha obtenido información de otros autores para la fase de estudio y elaboración de este proyecto, por lo que ha sido necesario realizar las correspondientes citaciones a las fuentes de las que se ha obtenido la información. Por ello, este documento cumple con el *Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia*

11. CONCLUSIONES Y LÍNEAS FUTURAS

11.1 Conclusiones sobre el proyecto

El objetivo de este subapartado es analizar y concluir los resultados que se han obtenido del proyecto, además de algunos problemas encontrados.

11.2 Resultados obtenidos

Se ha desarrollado una extensión para Google Chrome la cual proporciona al usuario el proceso de verificación de la integridad de los archivos que se descarga de manera automática. Dicha verificación se ha realizado generando los hashes del archivo que el usuario descarga, y comprobando si en la página de descarga se encuentra alguno de ellos o, en caso contrario, en alguna de las páginas referenciadas.

La gran ventaja de esta aplicación es que es compatible para cualquier ordenador siempre y cuando tenga instalado el navegador Google Chrome. Esto proporciona la facilidad de evitar problemas de compatibilidad entre distintos sistemas operativos. Además, se ha conseguido evitar que el usuario tenga que hacer de manera manual dicha comprobación.

Recordando las alternativas estudiadas en el apartado de Estado del Arte, la mayoría de ellas requerían la interacción del usuario para la verificación de la integridad y, además, solo eran compatibles con Windows. En cambio, con la extensión desarrollada se han conseguido principalmente 2 ventajas. En primer lugar, se generan los distintos tipos de hashes que ofrecen los desarrolladores en las páginas de descarga (MD5, SHA1, SHA256 Y SHA512), y se realiza el proceso de verificación de la integridad de manera automática y transparente, es decir, no se requiere la interacción del usuario. Además, se ha añadido la funcionalidad adicional de que la verificación no se realice solo en la página de descarga, sino que también acceda a los links que esta página contiene para poder verificar un mayor número de páginas posibles.

11.3 Dificultades encontradas

A pesar de haber cumplido con los objetivos propuestos para el desarrollo del proyecto, también se han encontrado una serie de dificultades, mayoritariamente en la etapa de implementación. Estas son las siguientes:

- En primer lugar, como la aplicación se ha desarrollado para ser compatible con Google Chrome, hubo que ceñirse a las tecnologías impuestas de esta plataforma. Con JavaScript, la depuración y las pruebas del funcionamiento del código ha sido algo complejo e incómodo ya, por ejemplo, para mostrar el valor de determinadas variables se tuvo que mostrar en mensajes emergentes o por la consola de la ventana en la que se realiza la descarga.
- La última versión de Google Chrome actualiza dinámicamente la extensión cuando detecta cambios en el código. En cambio, en varias ocasiones esto no ha funcionado así y ha llevado a la confusión. Esto es debido a que cuando se realizaban modificaciones en el código y se probaba de nuevo la aplicación, no se obtenía el resultado esperado puesto que el navegador no había actualizado correctamente la nueva versión del código.
- Se han encontrado pocos ejemplos de la API de Google Chrome utilizada por lo que en ocasiones resultó complicado poner en contexto ciertas funcionalidades.
- Por último, antes de comenzar con el proyecto pensé que dispondría de mayor tiempo para su realización. En cambio, algunas de las asignaturas del último cuatrimestre de la carrera y un examen de inglés que he tenido que realizar me han quitado más tiempo del esperado, por lo que ciertas tareas del proyecto se retrasaron en algún momento.

11.4 Conclusiones personales

A modo de resumen final, considero que ha sido una de las mejores maneras de abarcar la gran mayoría de conceptos aprendidos durante la carrera en un mismo proyecto. He podido solidificar muchos conceptos clave, y poner en práctica todo lo aprendido. Además de haber aplicado conocimientos para programar, ha sido de gran importancia aquellos aplicados y adquiridos en las asignaturas de Ingeniería de Software para planificar el proyecto y poder especificar todos los documentos necesarios sobre el proyecto.

Ha sido la primera vez que me he tenido que enfrentar individualmente, junto con la supervisión de mi tutora, a la realización entera de un proyecto. Esto ha supuesto asumir una gran responsabilidad pero que he sabido llevar con facilidad, compatibilizando al mismo tiempo la realización de las últimas asignaturas de la carrera, y del trabajo como becario informático en la misma universidad.

Con el trabajo de fin de grado he podido darme cuenta también de la gran falta de concienciación y responsabilidad de muchas empresas. Esto es debido a que la gran mayoría que ofrecen aplicaciones para descargar de Internet no proporcionan el checksum para verificar la integridad. Ofrecer dicho dato no supondría ningún coste para estas empresas, ya que estas invierten grandes cantidades de dinero para asegurar sus aplicaciones y demás servicios, y facilitar el checksum sería tan sencillo como utilizar algún programa (externo o interno) para obtener el resultado y velar por la integridad de los archivos que ofrecen a los usuarios.

11.5 Líneas futuras

De cara al futuro, sería positivo ofrecer otra serie de características:

- Introducir más idiomas, principalmente el inglés, para poder ofrecer la extensión a un mayor número de personas.
- Optimizar el proceso de acceso a los links de las diferentes páginas para que el tiempo de verificación sea menor y poder ofrecer al usuario una respuesta aún más rápida.
- Considerar la posibilidad de ofrecer esta aplicación para el resto de los navegadores. Esto no se ha realizado en este estudio puesto que se ha demostrado que Google Chrome a día de hoy sigue siendo el que abarca la gran mayoría del mercado.
- También convendría probar con más páginas de descarga ya que en ciertas ocasiones la aplicación deja de funcionar, pero no se ha conseguido encontrar el patrón que hace que deje de funcionar, ya que sucede muy de vez en cuando y no en los mismos casos (aparentemente).

BIBLIOGRAFÍA

[1] Toro, R. Confidencialidad, integridad y disponibilidad en los SG-SSI. PMG SSI - ISO 27001. https://www.pmg-ssi.com/2018/02/confidencialidad-integridad-y-disponibilidad/ [Accessed 20 May. 2019].

[2] II), M. MD5: vulnerabilidades y evoluciones (y II). Blog.elevenpaths.com. https://blog.elevenpaths.com/2013/11/md5-vulnerabilidades-y-evoluciones-y-ii.html [Accessed 4 May. 2019].

[3] RedesZone. La colisión SHA1 es real y ya se ha tomado su primera víctima. https://www.redeszone.net/2017/02/27/la-colision-sha1-real-ya-se-ha-tomado-primera-victima/ [Accessed 23 May. 2019].

[4] Academy, B. SHA-256, el algoritmo que usa la red Bitcoin | Bit2Me AcademyBit2Me Academy. https://academy.bit2me.com/sha256-algoritmo-bitcoin/ [Accessed 23 May. 2019].

[5] Es.wikipedia.org. SHA-2. https://es.wikipedia.org/wiki/SHA-2 [Accessed 24 May. 2019].

[6] Documentación web de MDN. Generalidades del protocolo HTTP. https://developer.mozilla.org/es/docs/Web/HTTP/Overview [Accessed 24 May. 2019].

[7] Ellis, C. and Turner, B. Best open source software of 2019TechRadar. https://www.techradar.com/best/best-open-source-software [Accessed 19 Feb. 2019].

[8] Towards Usable Checksums: Automating the Integrity Verification of Web Downloads for the Masses). https://serval.unil.ch/resource/serval:BIB_9BD511E5C0D0.P001/REF [Accessed 18 Feb. 2019].

[9] Dnslytics.com. Online investigation tool - Reverse IP, NS, MX, WHOIS and Search Tools. https://dnslytics.com/ [Accessed 20 Feb. 2019].

[10] Marketing 4 Ecommerce - Tu revista de marketing online para e-commerce. Cuáles son las redes sociales con más usuarios del mundo (2019). https://marketing4ecommerce.net/cuales-redes-sociales-con-mas-usuarios-mundo-2019-top/ [Accessed 26 Feb. 2019].

[11] Md5.jrham.es. md5sum - online md5summer. http://md5.jrham.es/ [Accessed 28 Feb. 2019].

[12] Evansville Web Design & Development. How to verify MD5, SHA1, and SHA256 Checksum on Windows. https://bhoover.com/how-to-verify-checksum-windows/ [Accessed 28 Feb. 2019].

[13] Jerosimić, I. Download IgorWare Hasher. Igorware.com. https://www.igorware.com/hasher/download [Accessed 29 Feb. 2019].

[14] Talekar, N. Hash Generator: Free All-in-one Tool to Generate Hash MD5/SHA1/SHA256/SHA512/BASE64/LM/NTLM/CRC32 | www.SecurityXploded.com. Securityxploded.com. https://securityxploded.com/hashgenerator.php [Accessed 29 Feb. 2019].

[15] Nextofwindows.com. https://www.nextofwindows.com/5-ways-to-generate-and-verify-md5-sha-checksum-of-any-file-in-windows-10 [Accessed 29 Feb. 2019].

[16] W3counter.com. W3Counter: Free Web Stats and Website Widgets. https://www.w3counter.com/ [Accessed 11 Mar. 2019].

[17] Balsamiq.com. Balsamiq. Rapid, effective and fun wireframing software. https://balsamiq.com/ [Accessed 22 Apr. 2019].

[18] Iberdrola.es. Plan estable para hogares - IBERDROLA. https://www.iberdrola.es/luz/plan-estable [Accessed 31 May. 2019].

[19] O2. O2 | 20GB y llamadas ilimitadas para tu tarifa móvil. https://o2online.es/movil/ [Accessed 31 May. 2019].

ANEXO 1: RESUMEN EN INGLÉS

1. Introduction

Every day, people use different applications in their terminals, whether it is in mobile phones, computers or tablets. In addition, many users want these applications to be each day more secure in order to protect their personal data. On the other hand, what must be clear in the first place is that the main responsible for the applications that are downloaded are the users themselves. Therefore, it must be verified what is downloaded in our computer equipment to protect our information.

In the case of the online download of applications on computers, some developers provide the checksum, which is a sequence of alphanumeric characters resulting from applying a hash function on the desired file. The result after applying these hash functions is unique, so it identifies the input file. The objective is to guarantee one of the fundamental pillars of information security, such as integrity, which corresponds to the ability to guarantee that the data has not been modified by any unauthorized person.

2. Objetives

To develop this project, first of all, it will be studied the current situation of the problem in order to conclude how many pages the checksum offers and what type they are. In addition, existing applications will be compared to facilitate integrity verification.

Once the results have been obtained after the study phase, it will be considered which solution is the most appropriate to solve the current problem and be able to develop the application with greater accuracy.

3. Actual situation and solutions

In order to study what the application download pages offer, it will be used
Table A1.1 with a set of programs that are the most used in 2019.

Table A1.1 is formed by the following:

- Name: program identifier.
- Description: brief information about the main functions.

- Program page protocol: transfer protocol that uses the download page of the application. It can be http or https.
- Checksum page protocol: transfer protocol that uses the page that offers the hash corresponding to the downloaded program. It can be http or https. In case of not providing said hash, the value of a red script has been given.
- Host: relationship between the download page and the page that offers the hash, considering if both are located on the same server, or if they belong to different domains. In case of not providing information about the checksum, the value of a red script is given.
- MD5, SHA1 and SHA256 and SHA512: the 4 types of hash functions that have been analyzed. The affirmative case is represented with a green tick and, if not, a red cross.

From table A1.1 the following information can be obtained:

- 11 of the 23 programs studied do not provide any type of hash to verify integrity. Almost half of them.
- Of the 12 programs that do offer some checksum 3 of them use the http protocol, so the data between client and server travel without encrypting. This can lead to the problem known as Man-In-The-Middle in which the attacker could interpose between the client-server connection and send the information considered by posing as the official page.
- 5 of the 12 programs that offer the hash have the file to be downloaded and the checksums on the same server, so if an attacker obtains access to said server, he could manipulate both data.
- Regarding the checksums offered by the 12 programs, 4 of them provide MD5, 2 offer SHA1, 9 SHA256 and 1 SHA512. The important thing here is that 9 of the 12 programs, that is, 75%, offer SHA256 being this one of the most important and secure.

TABLE A1.1. PROGRAM ANALYSIS.

Name	Description	Program page protocol	Checksum page protocol	Host	MD5	SHA1	SHA256	SHA512
Adobe Acrobat Reader	PDF viewer	https	-	-	✗	✗	✗	✗
Audacity	Video and audio editor	https	https	Different domain	✗	✗	✓	✗
Avast	Antivirus	https	-	-	✗	✗	✗	✗
AVG	Antivirus	https	-	-	✗	✗	✗	✗
Code Blocks	C and C++ development environment	http	http	Same server	✓	✓	✓	✗
CrypTool	Cryptography algorithm program	https	https	Different domain	✗	✗	✓	✗
Dropbox	Cloud storage	https	-	-	✗	✗	✗	✗
Eclipse	Java development environment	https	https	Same server	✗	✗	✗	✓
FL Studio	Audio editor	https	https	Same server	✗	✗	✓	✗
GIMP	Image editor	https	https	Different domain	✗	✗	✓	✗
Microsoft Office	Office programs for Windows	https	-	-	✗	✗	✗	✗
Oracle SQL Developer	Data base development environment	http	https	Different domain	✓	✓	✗	✗
Photoshop	Graphic editor	https	-	-	✗	✗	✗	✗
Putty	SSH client	https	-	-	✗	✗	✗	✗
PyCharm	Python development environment	https	https	Different domain	✗	✗	✓	✗
RStudio	Statistical computing and graphics	https	https	Different domain	✓	✗	✗	✗
Skype	Video chat and voice call	https	-	-	✗	✗	✗	✗
TeamViewer	Remote desktop between computers	https	-	-	✗	✗	✗	✗
Ubuntu	Operating system	http	http	Same server	✗	✗	✓	✗
uTorrent	File downloader	https	-	-	✗	✗	✗	✗
VirtualBox	Virtualization software	https	https	Same server	✓	✗	✓	✗
VLC	Multimedia player	https	https	Different domain	✗	✗	✓	✗
WinRAR	Trialware file archiver	https	-	-	✗	✗	✗	✗

Broadly speaking, it can be concluded that a large part of the programs today still does not provide the checksum, with the risks that this entails. Of those who do, a minority uses the http protocol, but these are specific cases, so the vast majority use https, since it is basic to ensure encryption between connections. Half of these host the downloadable file on the same server as the checksum itself, which may be for simplicity when

distributing the data, although this is a security problem. Finally, some of these programs offer the MD5 checksum, which does not end up being completely safe, as this has been shown to be more vulnerable to attacks. As a positive point, SHA256, is provided by 9 of the 12 programs that offer verification.
As an extension of the study, it has been decided to investigate whether the applications with more downloads and users offer the hashes of their download files [10]. These applications are WhatsApp, Facebook, Twitter etc. These applications have millions of users. Therefore, it could be that these users download any of these applications on their computers. In such a case, this type of application does not offer any type of checksum. This poses a great security risk due to the large number of people who use these applications on a daily basis, and, at a minimum, the possibility should be provided to verify that what is downloaded is not corrupt and corresponds to the original software.

Once concluded with what kind of security measures are offered by the download pages, it will be studied what applications exist today to help verify the integrity of the downloads, thus being able to conclude with the advantages they offer and the points to be improved.

The programs studied are the following:

- MD5SUM
- MD5 & SHA Checksum utility
- Igorware Hasher
- Hash generator
- Download and Verify Checksum

The conclusions after having studied and tested these applications are the following:

1. They do not offer all the algorithms that the download pages provide.
2. Mostly unsafe hashes are provided.
3. You need to do the check manually.
4. They are not compatible with all operating systems.
5. They have an unattractive interface.

The last program (Download and Verify Checksum) does meet points 2 and 3, but the rest does not, so the latter would be the exception (besides it is not so accessible to users

since it belongs to a study and does not appear in the first searches and recommendations in Google).

4. Analysis

4.1 Web browser

After having studied the current situation and the existing applications, it has been decided to develop an extension for a web browser. First, it will determine which of these is the best option based on the amount of traffic they receive, obtaining this information from W3Counter [16].

In the first place, instead of studying the most used browsers of 2018 (since there may be modifications from the previous year) and 2019 (we only have a few months of this year), it has been chosen to analyze them in the same month of both years, so it would correspond to a whole year. As you can see in the Figure A1.1, the most relevant data is that the predominant browser in both diagrams is Google Chrome, with a big difference. In both years it represents more than half, increasing its data traffic by almost 6% (from 58.7% in 2018 to 64.5% in 2019). Finally, it can be clearly concluded that Google Chrome is the most used browser, and what is also interesting is that it has increased its traffic so it could continue to be like this in the following years and be a wise investment. Therefore, as the goal is to provide our application to the largest number of users, Google Chrome will be chosen.

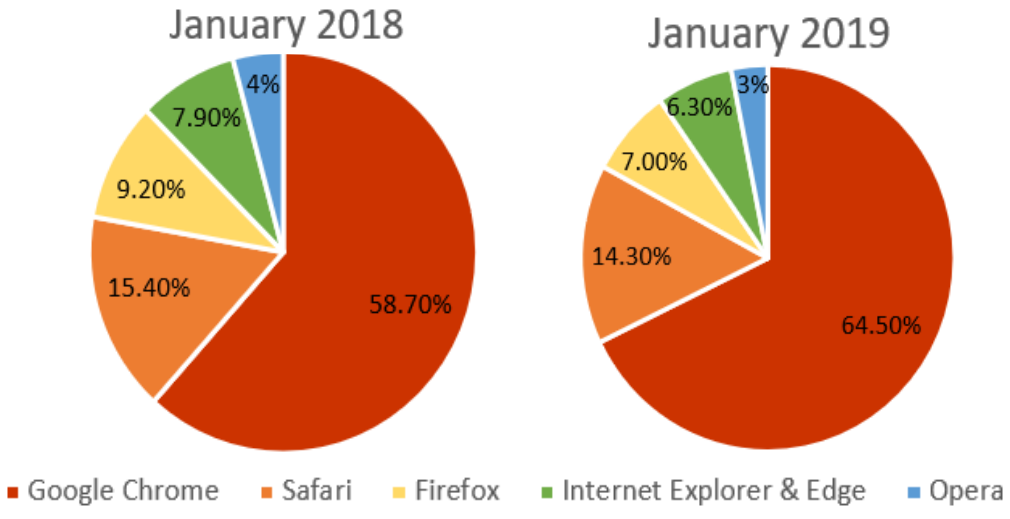


Figure A1.1. Traffic comparison between web browsers.

4.2 Technologies

As an extension for Google Chrome is going to be developed, this platform imposes a specific structure of the project to be followed and a series of technologies, so there is no possibility or alternative to choose different ones.

Since Google Chrome is a web browser, these technologies are based on web development. These are:

- HTML (HyperText Markup Language): focused on web development, such as its description, structure and elements.
- CSS (Cascading Style Sheets): used to declare the style and presentation of the web page.
- JavaScript: web programming language for adding functionalities and effects dynamically.

Finally, as I have done some online courses and web development projects in the last 2 years, I can say that I am familiar with these technologies, which will greatly facilitate the development stage.

4.3 High level diagram

In the diagram of Figure A1.2 the process of the operation of the application is shown. The process begins when a user initiates a download. The application is waiting for it to end and, when it is finished, it is obtained and verified if the downloaded file complies with the extensions of interest that have been previously marked. When it is the case of being a file within the accepted, the popup window is shown with the message that the download has finished. Next, the user is informed that begins the calculation of the different hashes of the obtained file. Finally, the checks are made to verify if any of the generated hashes match the one found on the web page from which the file is obtained, or on any of the pages that are referenced from the download page. When this last check is completed, the user is informed of the result of the verification of the integrity of his download.

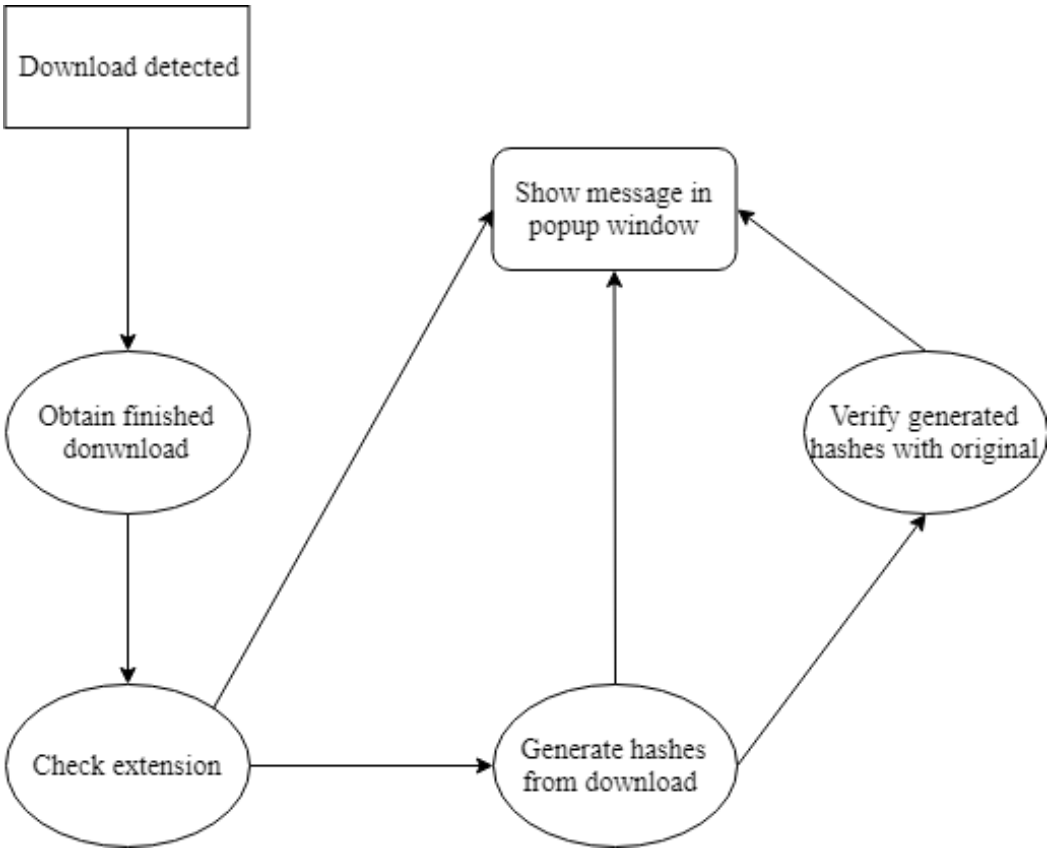


Figure A1.2. High level diagram.

4.4 System requirements

In this section it will show in Table A1.2 and Table A1.3 the functional and non-functional requirements of the application.

4.4.1 Functional requirements

TABLE A1.2. FUNCTIONAL REQUIREMENTS.

ID	Name	Description	Priority	State
FR01	Obtain file downloaded	The application must obtain the file downloaded by the user. This action is performed once the download is complete.	High	Developed
FR02	Check file type	The application must obtain the extension of the downloaded file and verify if it corresponds to a valid one. The extensions accepted are the following: exe, iso, msi, zip, tar, tar.gz, tar.xz, tar.bz2, deb, rpm and dmg.	High	Developed
FR03	Generate hashes	The application must generate the different hashes of the downloaded file. These are: MD5, SHA1, SHA256 AND SHA512.	High	Developed
FR04	Check hashes in download page	The application must search if any of the hashes generated previously are in the download page. In this case, a message will be displayed showing that the integrity of the downloaded file has been satisfactorily verified.	High	Developed
FR05	Check hashes in external pages	The application must verify if any of the hashes generated previously are in any of the referenced pages in the download page. If any hashes are found, the popup window will show the corresponding message. Otherwise, the user will be alerted that the integrity verification was unsuccessful.	High	Developed
FR06	Show popup window	The application must show the popup window after obtaining a file with an accepted extension. During the rest of the execution it will show the corresponding messages for each process.	High	Developed
FR07	Close popup window	The application must allow the user to close the popup window.	High	Developed

4.4.2 Non-functional requirements

TABLE A1.3. NON-FUNCTIONAL REQUIREMENTS.

ID	Name	Description	Priority	State
NFR01	Application compatible with Google Chrome	The application must be compatible with Google Chrome.	High	Satisfied
NFR02	Execution speed	The application must show the popup window to the user in less than 2 seconds after the download has finished.	High	Satisfied
NFR03	Application developed with HTML, CSS and JavaScript	The application must be developed with the following technologies: HTML, CSS and JavaScript.	High	Satisfied
NFR04	Privacy	The application will not collect personal or navigation user data.	High	Satisfied
NFR05	Spanish language	The application must be developed in Spanish.	High	Satisfied

5. Tests and results

In this section it will expose the different tests that have been done to evaluate if the system complies with the proposed objectives. Table A1.4 shows the results. The table is formed by:

- ID: unique identifier with TXX where XX corresponds to the corresponding test number.
- Requirements involved: those that are tested or are part of the process.
- Description: objective and general idea that it wants to be checked. The desired output after the test is also detailed.
- Output obtained: result that shows the application after being executed.
- Exceeded: Yes, in case of having passed the test, No, otherwise.

TABLE A1.4. TEST SET.

ID	Requirements involved	Description	Output obtained	Exceeded
T01	FR01 FR02 FR06	Download a file with a valid extension. The system must accept it and show the popup window with a message that a new download has been detected.	The popup window is displayed showing download detected message.	Yes
T02	FR01 FR02	Download a file with an invalid extension. The system must reject it and show the popup window.	The popup window is not displayed.	Yes
T03	FR01 FR02 FR03 FR06	Download a file with valid extension and generate the hashes. The system must show the popup window with the message of calculating hashes.	The popup window is displayed showing calculating hashes message.	Yes
T04	FR01 FR02 FR03 FR04 FR05 FR06	Download a file with valid extension which has the checksum on the download page. The system must show the popup window with the message that the generated checksum matches the one on the download page.	The popup window is displayed showing checksum matches message.	Yes
T05	FR01 FR02 FR03 FR04 FR05 FR06	Download a file with a valid extension that does not have checksum to verify. The system must show the popup window with the message that the file's checksum has not been found or does not match the one on the download page.	The popup window is displayed showing checksum does not match or it has not been found message.	Yes

T06	FR01 FR02 FR03 FR04 FR05 FR06	Download a file with a valid extension that has the checksum on an external page external of the download page of the file. The system must show the popup window with the message that the generated checksum matches the one on the download page.	The popup window is displayed showing checksum matches message.	Yes
T07	FR01 FR02 FR06 FR07	Download a file with valid extension and press button to close popup window when it appears. The system must close the window and not show it again.	The window closes, but it reappears when the application has to show a new message.	NO
T08	FR01 FR02 FR03 FR04 FR05 FR06 FR07	Download a file with a valid extension and press the button to close the popup window at the end of the program execution. The system must close the window and not show it again.	The window closes and does not show again.	Yes

As can be seen, in Table A1.4 the column of requirements involved is cumulative, that is, as the functionalities of the application are tested from less to more, more requirements are involved with a cumulative nature as it is an incremental process. From this table the most important data is the last column "Exceeded", in which it is indicated if the test responds correctly to the due and proposed functionality. Therefore, it can be concluded that of the 8 tests presented all have been overcome except for the T07 test, which corresponds to the window closing test when the program has not finished. What should happen is that the window closes and does not appear again. On the other hand, when closing it, when the system has to show a new message, the window reappears.

6. Conclusions

6.1 Results obtained

An extension has been developed for Google Chrome which provides the user with the process of verifying the integrity of the files that is downloaded automatically.

The great advantage of this application is that it is compatible for any computer as long as the Google Chrome browser is installed. This provides the facility to avoid compatibility problems between different operating systems. In addition, it has been possible to avoid that the user has to do this verification manually.

In summary, two main advantages have been achieved: generate all the checksums offered by the application developers and prevent the user from having to carry out the verification manually since this is done automatically by the application.

6.2 Future Works

Looking forward, it would be interesting to offer features:

- Add more languages, mainly English, in order to offer the extension to a greater number of people.
- Optimize the access process to the links of the different pages in order to make the verification faster and offer the user an even faster response.
- Consider the possibility of offering this application for the rest of the web browsers. This has not been done in this study since it has been shown that Google Chrome today remains the one that covers the vast majority in the market.
- It would also be convenient to try more download pages since sometimes the application stops working, but it has not been possible to find the pattern that makes it stop working, since it happens very occasionally and not in the same cases (apparently).

6.3 Personal conclusion

As a final summary, I think it has been one of the best ways to cover the vast majority of concepts learned during the race in the same project. It has been able to solidify many key concepts and put into practice everything learned. In addition to having applied knowledge to program, it has been of great importance those applied and acquired in the Software Engineering courses to plan the project and be able to specify all the necessary documents about it.

It was the first time that I had to face individually, together with the supervision of my tutor, the entire realization of a project. This has meant assuming a great responsibility but that I have known how to take with ease, making compatible at the same time the realization of the last subjects of the career, and of the work as a computer scholar in the same university.

With the end of grade work, I have also been able to realize the great lack of awareness and responsibility of many companies. This is because the vast majority that offer applications for downloading from the Internet do not provide the checksum to verify the integrity. Offering this data would not entail any cost for these companies, since they invest large amounts of money to ensure their applications and other services, and facilitate the checksum would be as simple as using a program (external or internal) to obtain the result and ensure the integrity of the files they offer to users.

ANEXO 2: MANUAL DE USUARIO

En principio la aplicación no va a ser publicada en la plataforma de extensiones para Google Chrome ya que es la primera versión y se desearía mejorarla en algunos aspectos como, por ejemplo, ofrecer más idiomas. Aun así, no se descarta esta opción en futuras versiones.

Como en un principio esta aplicación y este documento tiene un fin académico se ha considerado útil explicar en unos simples pasos cómo instalar la extensión en el navegador de Google Chrome y poder disfrutar de las ventajas que esta ofrece.

1. Configuración de la extensión

1. Obtener el archivo comprimido del proyecto tfg_extension.zip y descomprimirlo.
2. Acceder al navegador Google Chrome e instalar la extensión
3. Seleccionar el icono de Ajustes en la parte superior derecha del navegador, como en la Figura A2.1.

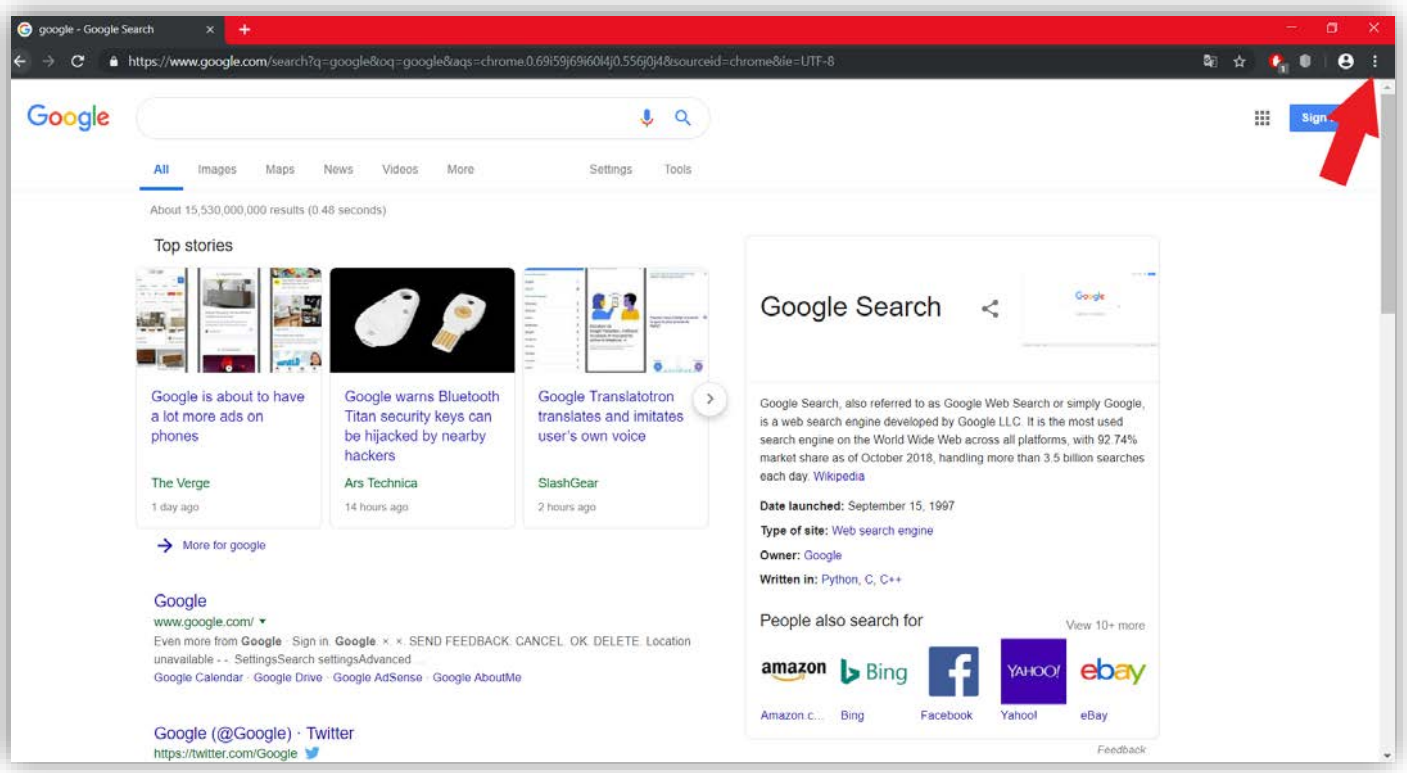


Figura A2.1. Manual de usuario seleccionar ajustes.

4. Dentro de Ajustes, seleccionar Más herramientas → Extensiones, mostrado en la Figura A2.2.

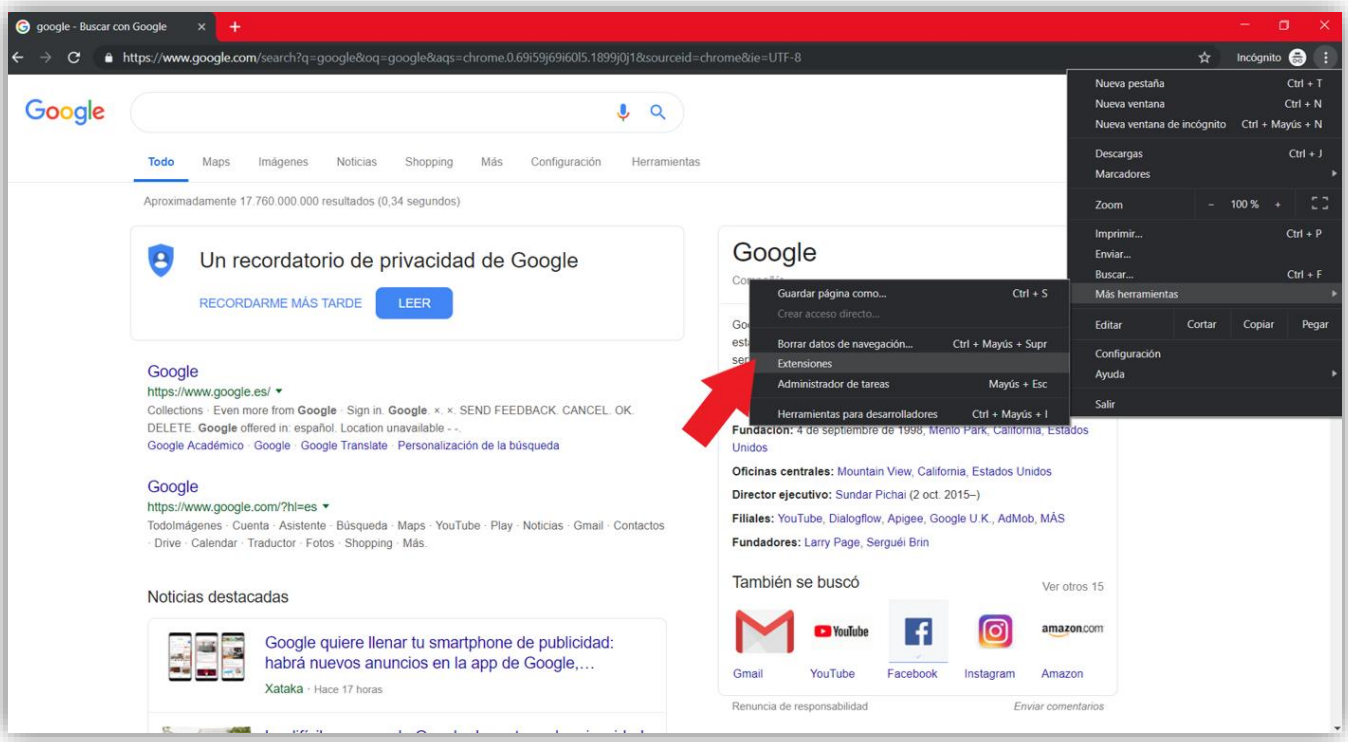


Figura A2.2. Manual de usuario seleccionar extensiones.

5. En esta nueva ventana, en la parte superior derecha, si no está activado el Modo de desarrollador, activarlo como se muestra en la Figura A2.3.

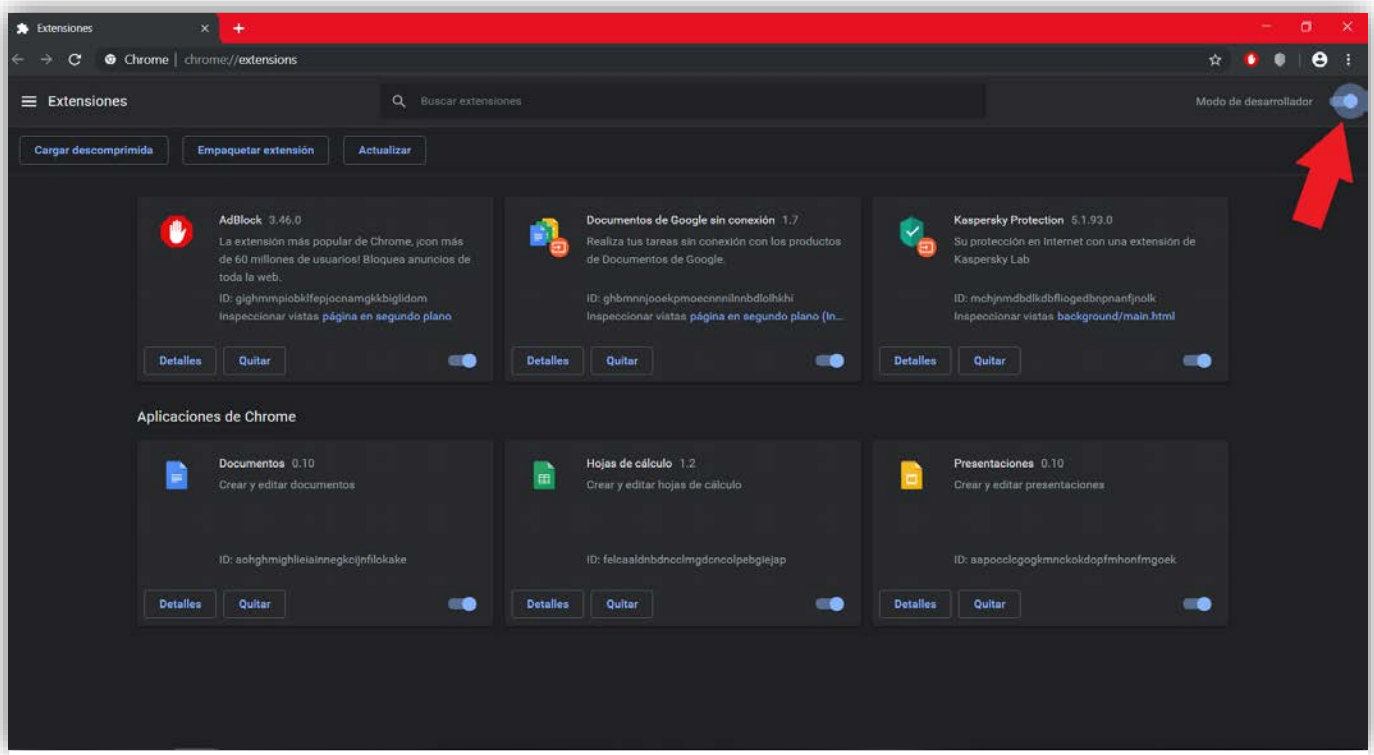


Figura A2.3. Manual de usuario activar modo de desarrollador.

6. En la Figura A2.4 se muestra el proceso de seleccionar Cargar descomprimida en la que se abre una ventana para elegir la carpeta de la extensión, pero ya descomprimida.

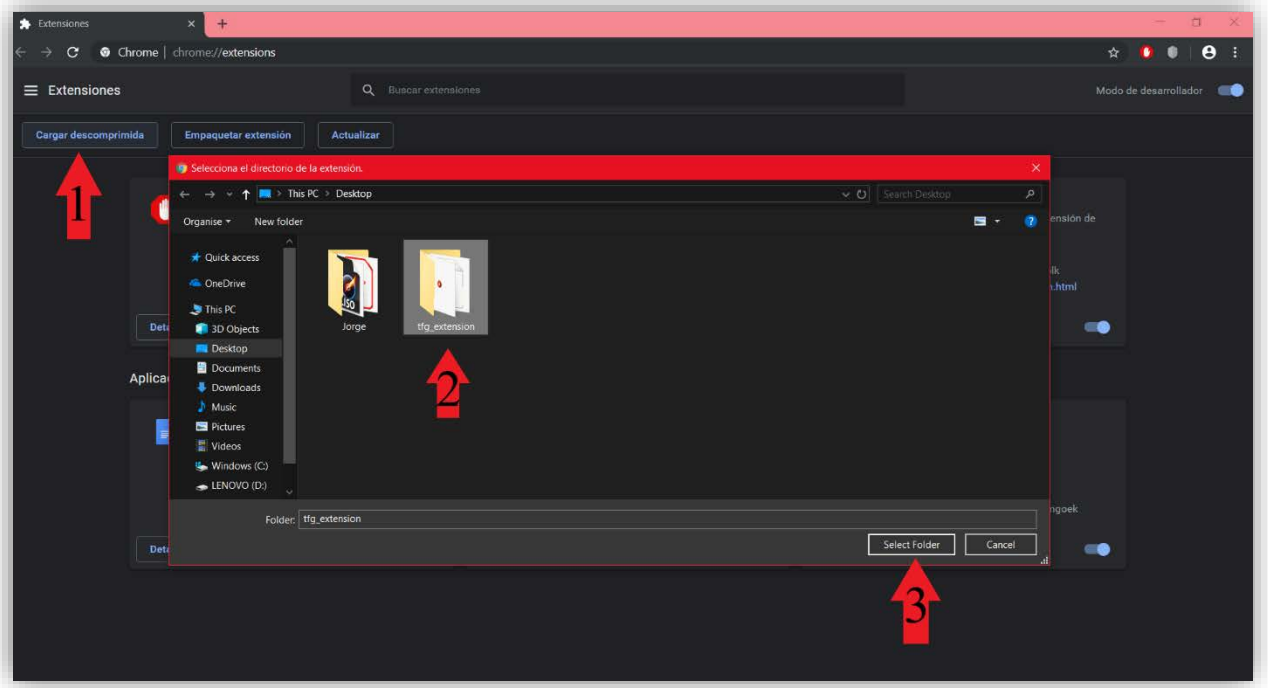


Figura A2.4. Manual de usuario seleccionar extensión.

7. Una vez seleccionado la carpeta, Google Chrome añade esta extensión junto al resto, como se muestra en la Figura A2.5.

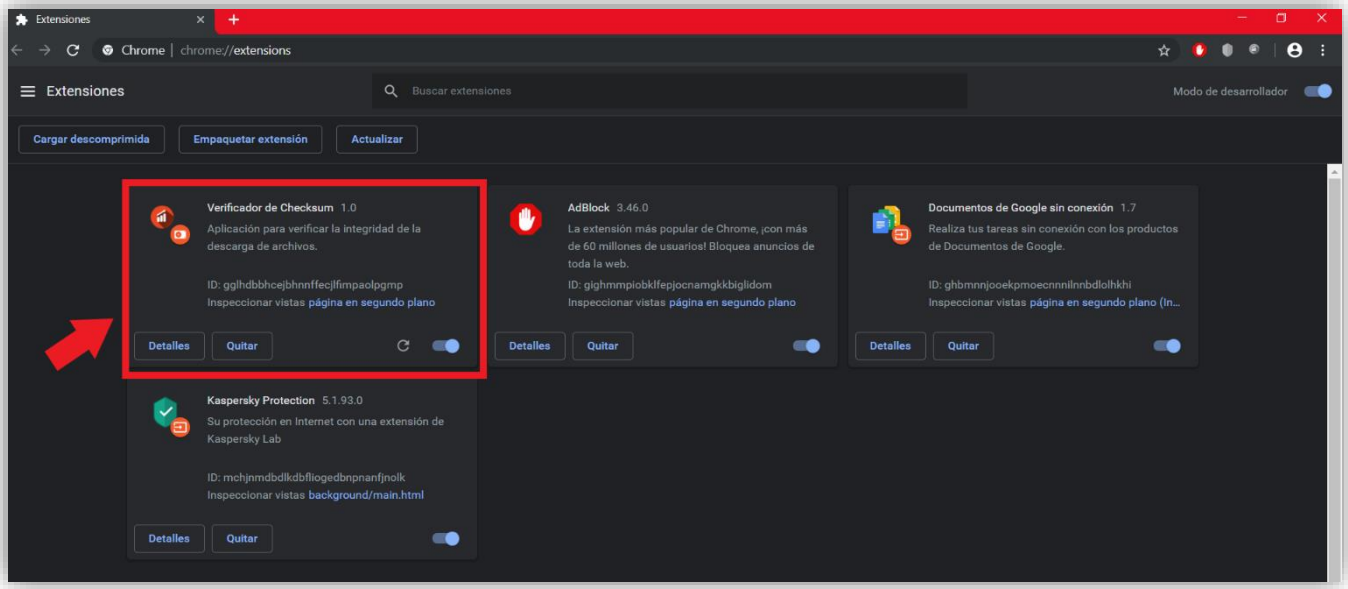


Figura A2.5. Manual de usuario extensión añadida.

2. Probar extensión

Para hacer una pequeña demostración del funcionamiento de la aplicación se van a mostrar 3 ejemplos. En el primero se va a probar con una aplicación que ofrece el checksum en la misma página de descarga. En segundo lugar, otra aplicación que ofrece dicho checksum en otra página diferente a la de la descarga. Por último, el caso de que la aplicación no ofrezca el hash de verificación.

2.1 Checksum en página de descarga

En este caso se va a obtener la herramienta Cyptool la cual ofrece el checksum en la misma página que en la descarga de la aplicación (obtenida en https://www.cryptool.org/de/ct1-downloads).

1. Visitar página de descarga y seleccionar versión en español. Esperar a la finalización del proceso de descarga, como es mostrado en la Figura A2.6.

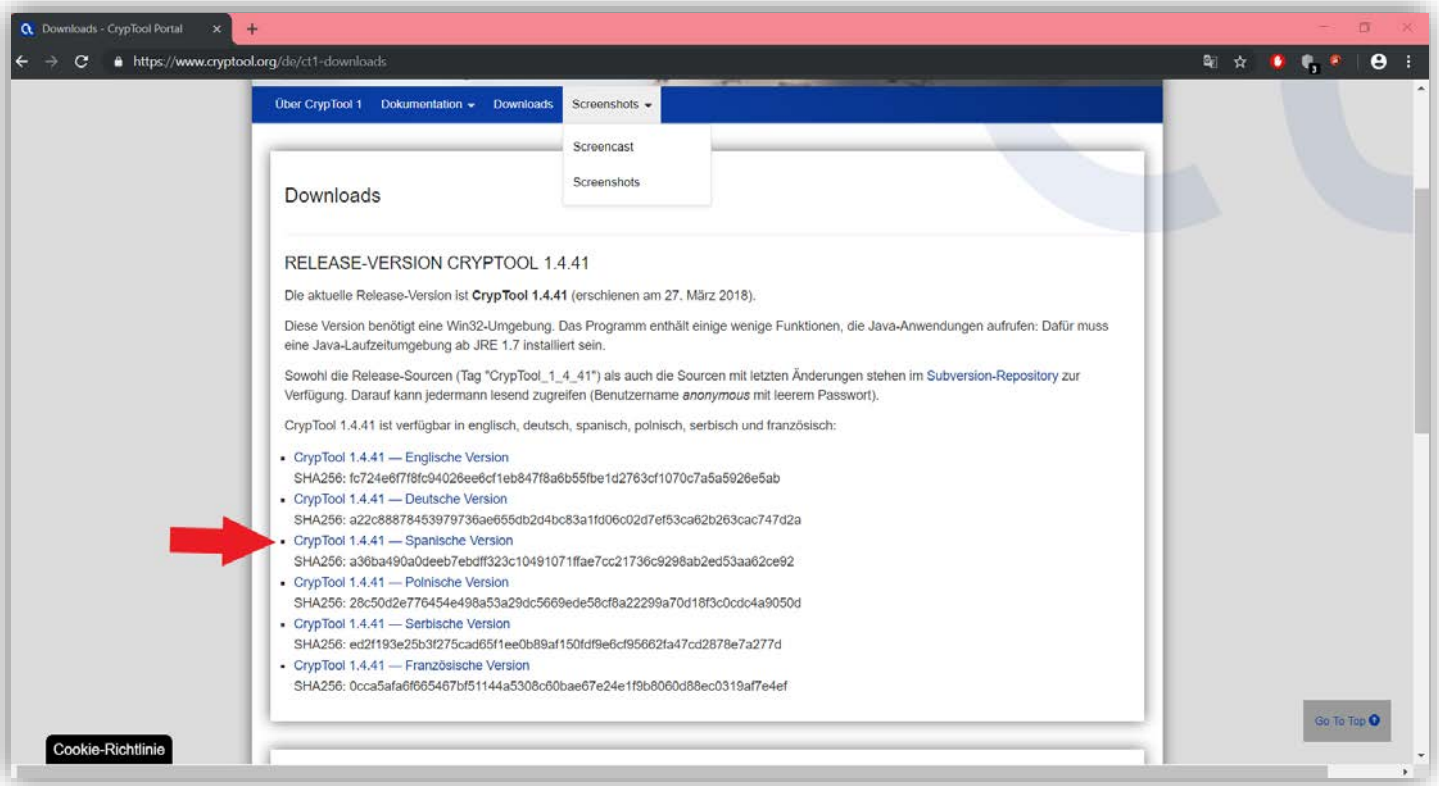


Figura A2.6. Manual de usuario descarga de Cyrptool.

Como se puede observar en esta imagen, para cada archivo (con idioma distinto) se ofrece su correspondiente hash de verificación distinto, puesto que son archivos diferentes.

2. Cuando la descarga finaliza, se obtiene el resultado de que se ha encontrado coincidencia, como se muestra en la Figura A2.7, ya que el checksum se ofrece debajo de la descarga (SHA256).

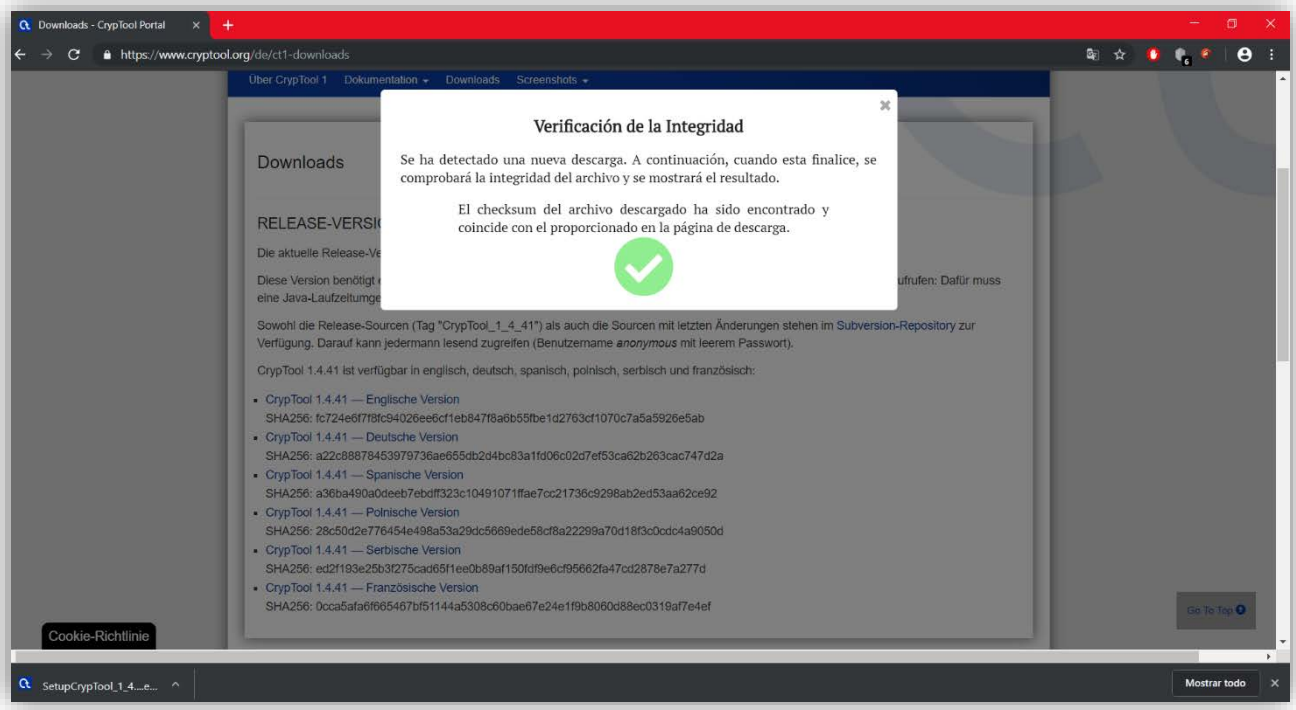


Figura A2.7. Manual de usuario verifiación de Cryptool.

2.2 Checksum en página diferente

En esta demostración se ha elegido VirtualBox la cual ofrece los hashes MD5 y SHA256 en 2 páginas distintas (se puede ver ejemplo en Figura 40). (obtenida en https://www.virtualbox.org/wiki/Downloads)

1. En la Figura A2.8 se muestra la página de descarga de VirtualBox. Seleccionar versión para Windows en este caso. Esperar a finalización del proceso de descarga. La Figura A2.9 muestra lo que contiene la página con los checksums SHA256.

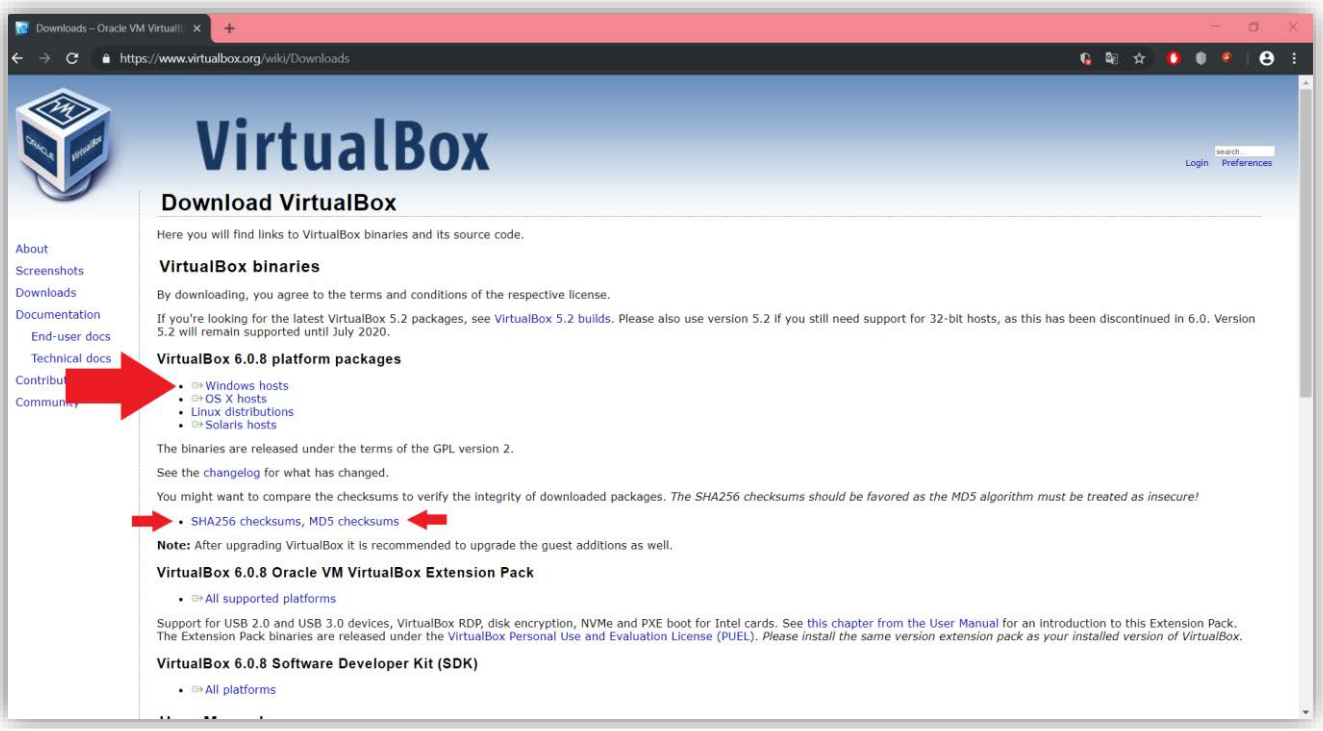


Figura A2.9. Manual de usuario descarga de VirtualBox.

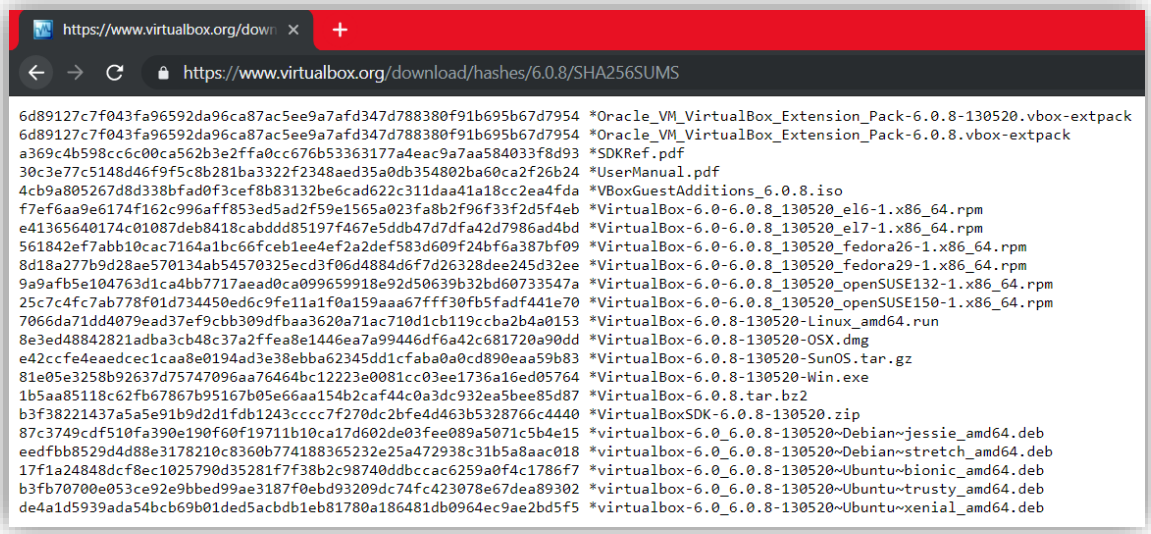


Figura A2.8. Manual de usuario SHA256 VirtualBox

2. Cuando finaliza la descarga se obtiene el mensaje mostrado en la Figura A2.10 de que la verificación ha sido satisfactoria, ya que como se puede observar en la Figura A2.8, los hashes SHA256 y MD5 son proporcionados en páginas distintas.

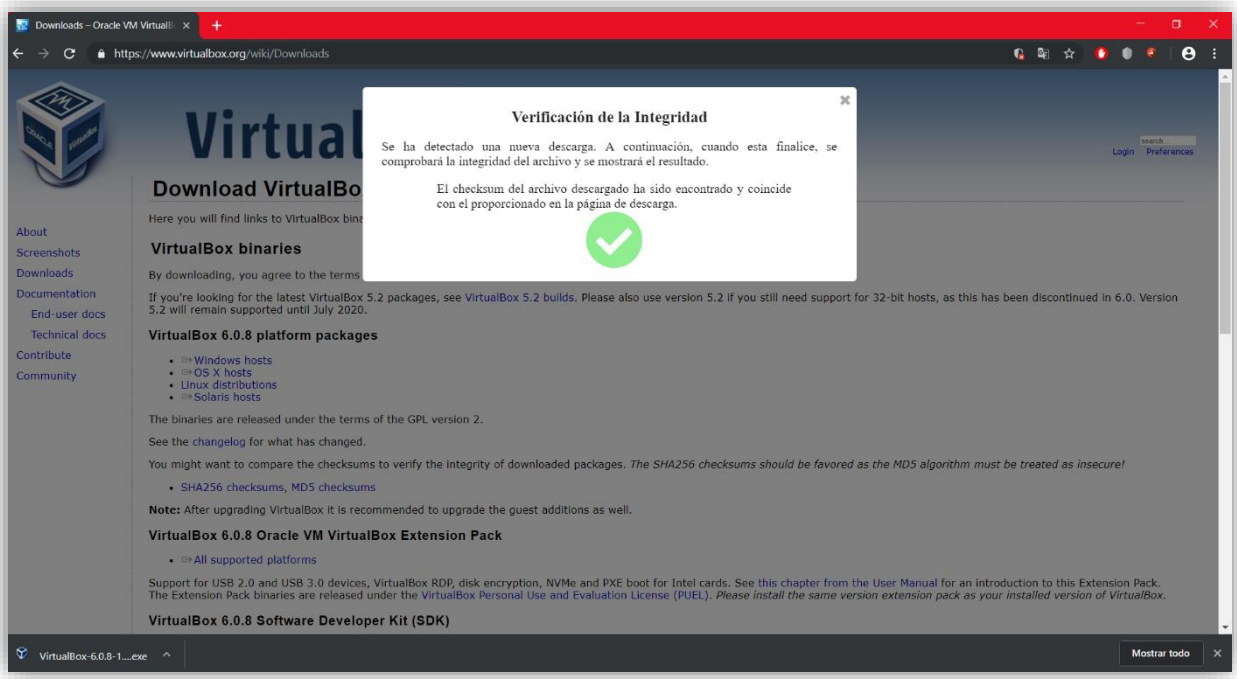


Figura A2.10. Manual de usuario verificación VirtualBox.

2.3 Checksum no disponible

Por último, se va a probar a descargar Avast la cual no proporciona el checksum ni en la página de descarga ni en otras páginas (obtenida en https://www.avast.com).

1. Entrar en la página de descarga de Avast y seleccionar el archivo como se indica en la Figura A2.11.

Figura A2.11. Manual de usuario descarga de Avast.

2. Como era de esperar, se obtiene el mensaje mostrado en la Figura A2.12 de que no se ha podido verificar la integridad puesto que no ofrece ningún checksum en ninguna página.

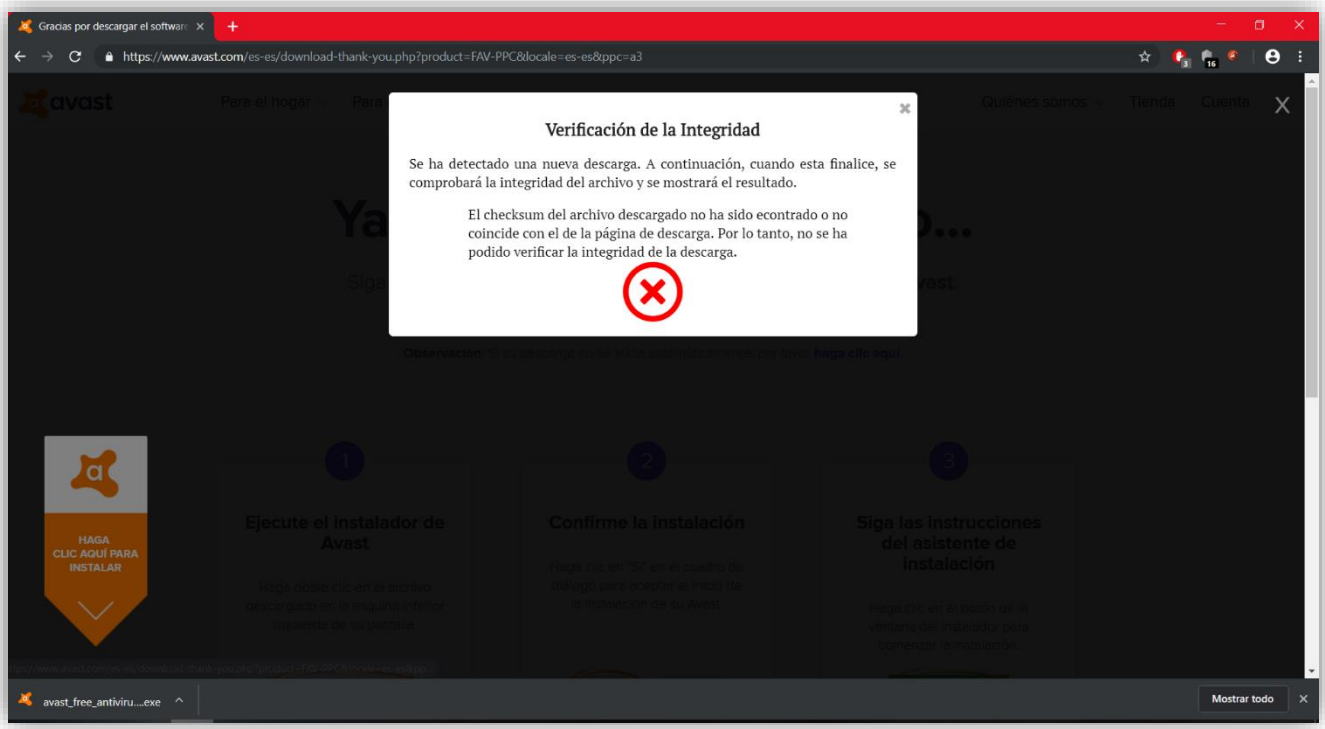


Figura A2.12. Manual de usuario verificación de Avast.