

MÁSTER EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DECENTRALIZED APPLICATIONS’ SAFETY AND SECURITY

Autor: Miguel Oleo Blanco

Director: Yue Duan

Co-Director: Sajad Meisami

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Decentralized applications’ safety and security

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/2023 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Miguel Oleo Blanco Fecha: ……/ ……/ ……

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Yue Duan Fecha: ……/ ……/ ……



GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DECENTRALIZED APPLICATIONS’ SAFETY AND SECURITY

Autor: Miguel Oleo Blanco

Director: Yue Duan

Co-Director: Sajad Meisami

Madrid

**Agradecimientos**

Esta sección es opcional

**DECENTRALIZED APPLICATIONS’ SAFETY AND SECURITY**

**Autor: Oleo Blanco, Miguel**

Director: Duan, Yue.

Co-Director: Meisami, Sajad

**RESUMEN DEL PROYECTO**

En este trabajo se investigará sobre la seguridad de las aplicaciones distribuidas o DApps. Para ello se realizará un estudio previo sobre los distintos mecanismos de seguridad que la cartera digital Metamask proporciona a las aplicaciones. Con los resultados obtenidos, se creará una herramienta de análisis estático del código para determinar la seguridad de estas.

**Palabras clave**: Blockchain, DApps, Ciberseguridad, Criptomonedas, Testing.

1. **Introducción**

Desde la aparición de la tecnología de la cadena de bloques, muchos problemas de seguridad han surgido. Estos problemas normalmente vienen vinculado a estafas y fallos de seguridad vinculados a las aplicaciones descentralizadas que trabajan sobre el blockchain. Debido a esto, hay una necesidad por parte del usuario de fiarse de estas aplicaciones que, finalmente, pueden resultar maliciosas. Por ello, con este trabajo, se busca elaborar una herramienta que, analizando el código fuente de dichas aplicaciones, determine si es seguro de utilizar. Para ello, es necesaria una investigación previa sobre la implementación de los mecanismos de seguridad y su funcionamiento.

1. **Definición del proyecto**

El principal objetivo de este proyecto es la creación de una herramienta que, analizando el código fuente de una aplicación distribuida especificada, determine su seguridad. Para ello, como se ha citado en la sección de la introducción, hay una fase previa de investigación. Es esta fase primero se estudió el lenguaje de programación más usado entre las 100 DApps más usadas.

Una vez estudiado el lenguaje de programación, se estudian los diferentes métodos/paquetes para implementar el blockchain con las aplicaciones. En esta fase se concluye que, independientemente del paquete, es la propia wallet la que gestiona los permisos y verificaciones. Debido a esto, se procederá a estudiar en profundidad Metamask, ya que es la principal wallet. Estas wallets incorporan una serie de APIs propias y heredadas de Ethereum (o de otras criptomonedas). Para comprender en detalle el funcionamiento de estas APIs, se ha creado una tabla con los 70 diferentes métodos, los parámetros obligatorios y opcionales, si es necesaria confirmación y el resultado de este. En esta fase, se concluye que la documentación, al no haber una organización principal detrás de definición de estos métodos, las implementaciones pueden tener pequeñas diferencias con las recomendaciones de Ethereum (EIP). Una vez detectadas las APIs necesarias de confirmación, se realiza un estudio en profundidad de las mismas. Para ello, se analiza el código fuente de estas y se compara con el proporcionado por la documentación de Metamask.

Una vez entendido el código, se procede a buscar la implementación de tres métodos de firma en el listado de las 100 DApps. Estos tres métodos son *eth\_sign, eth\_signtypeddata* y *personal\_sign*, haciendo hincapié en el segundo. Para el estudio del método *eth\_signtypeddata* se ha creado una aplicación para test en la que se han probado distintos contenidos para determinar la seguridad de este método. Con estas pruebas se ha logrado determinar la estructura mínima necesaria para que la firma resultante al ejecutar este método sea exitosa. También, se ha logrado terminar que, debido a la flexibilidad que ofrece este método, no se establecen parámetros obligatorios ni se comprueban dichos parámetros.

CONTINUAR

1. **Descripción del modelo/sistema/herramienta**

La metodología empleada para el correcto desarrollo del proyecto se basa en la metodología *Agile*. En esta metodología se crean muchos *Sprints* (pequeñas iteraciones del trabajo), consiguiendo reiterar el trabajo del trabajo e incorporar e incrementar el contenido de este. Esta metodología es la óptima para este tipo de trabajos ya que, como se trata de un trabajo de investigación, es difícil planificar todo desde un principio, ya que hay muchas incógnitas que surgen a lo largo del proyecto. Para aplicar esta metodología en el proyecto, se ha optado por planificar al menos una reunión por semana (dos preferiblemente). En dichas reuniones, se comentan los resultados obtenidos durante la semana y se discute y fijan los objetivos a obtener para la siguiente reunión.

Como se ha citado previamente, se trata de un trabajo de investigación. Esto conlleva que la mayor parte del proyecto se centra en conocer la herramienta que se emplea y el objetivo. Debido a esto, no es hasta el final del proyecto donde se obtiene un resultado mesurable. Otro de los objetivos del proyecto, es contribuir a la comunidad científica con la publicación de un *paper* con los resultados obtenidos. Este último objetivo, por motivos de tiempos, se ejecutará pasada la defensa de este trabajo.

En cuanto a las herramientas que se utilizan a lo largo del desarrollo del trabajo, hay una gran variedad. Cada herramienta proporciona una utilidad distinta (el cual se citará a continuación) al desarrollo del proyecto:

* Metamask [1]: Esta es la principal herramienta del proyecto. Esta es una cartera digital de criptomonedas que se conecta a las distintas aplicaciones descentralizadas y proporciona toda la seguridad, lógica y funciones a estas. De esta herramienta, se investigan las funciones de seguridad y aquellos métodos que requieren confirmación del usuario. Estos últimos son los más importantes, ya que se tratan de métodos con datos sensibles y por donde se introducen la gran mayoría de ataques.
* DappRadar [2]: Esta es una página web que nos proporciona información sobre las DApps con más tráfico (las más usadas) del internet. Esta herramienta nos proporciona la información necesaria para investigar el lenguaje de programación más usado de las aplicaciones. Este listado se ha utilizado para llevar a cabo la investigación sobre la implementación de los distintos métodos estudiados.
* Google Drive: Esta herramienta se ha incorporado al proyecto para compartir la información que se va generando. Nos proporciona una plataforma para crear documentos y compartir con el resto de los miembros del proyecto. Principalmente se ha hecho uso de las hojas de cálculo tipo Excel para recoger toda la información sobre los distintos métodos de Metamask estudiados. Esto nos permite listar toda la información de forma organizada.
* Github [3]: Esta plataforma aporta un repositorio virtual en internet donde tener disponible todo el material empleado. La principal diferencia con Google Drive, es que esta plataforma esta enfocada al código fuente. También proporciona funciones específicas para equipos de desarrollo.
* Node.js [4]: Este es un entorno de desarrollo y ejecución para JavaScript. La gran mayoría de aplicaciones descentralizadas, están desarrolladas en JavaScript o TypeScript (lenguaje basado en JavaScript). Esta herramienta se ha empleado para el desarrollo de una aplicación propia para test que se realizan a lo largo del proyecto.
* CONTINUAR

1. **Resultados**

* Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud.



Ilustración 1: DApp desarrollada para las pruebas

1. **Conclusiones**

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud.

1. **Referencias**

[1] The crypto wallet for Defi, Web3 Dapps and NFTs | MetaMask. (s. f.). <https://metamask.io/>

[2] DappRadar - The World’s Dapp Store | Blockchain Dapps Ranked. (s. f.). DappRadar. <https://dappradar.com/>

[3] GitHub: Let’s build from here. (s. f.). GitHub. <https://github.com/>

[4] Node.js. (s. f.). Node.js. <https://nodejs.org/es>

**DECENTRALIZED APPLICATIONS’ SAFETY AND SECURITY**

**Author: Oleo Blanco, Miguel.**

Supervisor: Duan, Yue.

Co-Supervisor: Meisami, Sajad

**ABSTRACT**

This is a research project regarding the security of the distributed apps of the blockchain, also known as DApps. The first phase objective is to research the different security methods provided by the blockchain Metamask to the DApps. With the results obtained, a static análisis tool will be created to run over DApps source codes and determine their security.

**Keywords**: Blockchain, DApps, Cybersecurity, Cryptocurrencies, Testing.

1. **Introduction**

Since the appeal of the Blockchain technology, many security problems have emerged. These issues are usually linked to phishing attacks, scams and implementation faults with the decentralized apps that works on top of the blockchain. Due to this, there is an increasing need for the users to check for the security of those apps, which can be malicious. This Project seeks to create a tool to check the source code of the apps in order to determine if they are safe to use by the final user. To achieve this, an investigation and research should be made over the topic of the security of the dapps and the blockchain wallets and how those methods are implemented on real world decentralized applications. Lastly, a tool will be created to achieve it.

1. **Project Definition**

The main goal of the project is creating a tool that determines whether a decentralized application is safe to use or not. This will be achieved by analyzing the source code of the application. To achieve this, all the work mentioned on the Introduction should be made, this includes all the research before the implementation of the tool. The research phase starts with the investigation of the most common programming languages used by the DApps by analyzing the ranking of the top 100 used DApps.

Once that the most common programming languages is defined, different methods and packages used to implement the blockchain into the applications, are researched. In this phase, it is conclude that, independently of the package used, the wallet is what handles all the logic, methods, permissions and verifications of the application. Due to this, the next step is deeply studying in detail Metamask, due to the fact that it is the most used blockchain wallet. Metamask implements 70 different APIs, al lof these methods are studied in detail and documented into a Google Spreadsheet with the required and optional parameters, whether the user needs to confirm or not, and the return of the method. This phase concludes with the documentation that was obtained during the process. There are no organizations behind the blockchain, so the documentation, in most cases differs from one site to another. For that reason, the documentation obtained during the research, is key for the success of the project. In the documentation all of this is stated with the differences that were found in between the experiments we carried out and the documentation form the Ethereum Implementation Proposals (EIP). Once all the APIs that need user confirmation, are documented, a deeper study is made into those. This includes mainly studying the source code and implementation provided by Metamask, and comparing it with the documentation obtained and provided.

Once the coded is understood, the next step is to look for the implementation of three of the main APIs into the top 100 Dapps. These three methods are *eth\_sign*, *eth\_signtypeddata* and *eth\_personal\_sign*. In this project, the second method is the one that we focus the research, by implementing a test DApp that allows us to try out how the wallet checks all the parameters and data. Thanks to these tests, it is determined that this method is very powerful, modern and specially flexible. Most of the parameters are optional, which makes the method very flexible, but also very hard to make it secure, as the wallet cannot check all the data, only the definition of the structure and types.

CONTINUAR

.

1. **Description of the model/system/tools**

The methodology behind this project is key to the correct development of itself and it is based on the Agile methodology. With this methodology there are many Sprints (small iterations of the work). With this methodology, the work is iterative, and we can implement new features on top of the previous sprints. This way of working is optimum for this type of research projects, due to the fact that we keep incorporating things that we find during the research. To implement this, several meetings will be held each week to discuss the results obtained during that week and the new objectives will be set.

As mentioned before, this is a research project. That means that most of the duration of itself is occupied by the research part. Another important part is understanding the tools that will be used to research and to obtain the results. The main objective of this project is to create a paper in which all the results obtained will be mentioned and explained for the community. Due to timming, this last objectived might be reached after the defense of the project.

There are several tools that will be used during the project to achieve the objectives. Each tool will provide a unique utility (that will be mentioned) to the correct development of the project.

* Metamask [1]: This is the main tool used in the project. This is a digital wallet of the blockchain that provides all the interface to the users. This wallet also provides logic, utility and security to the decentralized applications. There will be investigated all the security features and methods that this wallet provides, specially the ones that required user confirmation. This methods are the most sensitive ones because they work with private data and this is the entry points of many attacks.
* DappRadar [2]: This is a website that provides information about DApss with the most traffic (the most used) of the internet. This tool provides the information needed to research about the most used coding language for DApss. Also we have created a database with the top 200 used dapps to research about the main methods that we cover in this project.
* Google Drive: This tool is used to share all the information that is being generated during the project. This tool provides a platform to create and share documents with the other team members. Mainly, we have used the spreadsheets to create and structure the database of the DApps and APIs. This is very useful to structure all the data.
* Github [3]: This platform provides a virtual repository on the internet in which we can store, share and collaborate in coding project. The main difference with Google Drive is that, this platform is focused on source codes. It also provides some specific features for coders.
* Node.js [4]: This is a development and execution environment for JavaScript. The majority of the apps are coded using JavaScript or TypeScript (it is a modified version of JavaScript). This tool have been used to create a test DApps to test different methods and carry on the security tests to obtained all the data beforehand.
* CONTINUAR

1. **Resultados**

* Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud.



Ilustración 2: DApp desarrollada para las pruebas

1. **Conclusiones**

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud.

1. **Referencias**

[1] The crypto wallet for Defi, Web3 Dapps and NFTs | MetaMask. (s. f.). <https://metamask.io/>

[2] DappRadar - The World’s Dapp Store | Blockchain Dapps Ranked. (s. f.). DappRadar. <https://dappradar.com/>

[3] GitHub: Let’s build from here. (s. f.). GitHub. <https://github.com/>

[4] Node.js. (s. f.). Node.js. <https://nodejs.org/es>

***Índice de la memoria***

Capítulo 1. Introducción 5

Motivación del proyecto 5

Ejemplo de cómo documentar el trabajo 6

Figuras 6

Ecuaciones 8

Elementos de numeración 8

Tablas 9

Código fuente de programas 9

Actualización de campos 9

Pie de página y encabezado 10

Capítulo 2. Descripción de las Tecnologías 11

Capítulo 3. Estado de la Cuestión 13

Capítulo 4. Definición del Trabajo 14

4.1 Justificación 14

4.2 Objetivos 14

4.3 Metodología 15

4.4 Planificación y Estimación Económica 16

Capítulo 5. Sistema/Modelo Desarrollado 18

5.1 Análisis de los lenguajes empleados 18

5.2 Implementación del blockchain 19

5.3 API de Metamask 21

5.4 API Ether.js y Web3.js 24

5.5 Problemas en la documentación 26

Capítulo 6. Análisis de Resultados 27

Capítulo 7. Conclusiones y Trabajos Futuros 28

Capítulo 8. Bibliografía 29

ANEXO I 31

***Índice de ilustraciones***

[Ilustración 1: DApp desarrollada para las pruebas 11](https://upcomillas-my.sharepoint.com/personal/201706428_alu_comillas_edu/Documents/IIT/TFM/Report%20DOCS/TFM%20Miguel%20Oleo%20Blanco.docx#_Toc137571993)

[Ilustración 2: DApp desarrollada para las pruebas 16](https://upcomillas-my.sharepoint.com/personal/201706428_alu_comillas_edu/Documents/IIT/TFM/Report%20DOCS/TFM%20Miguel%20Oleo%20Blanco.docx#_Toc137571994)

[Ilustración 3: Cronograma del proyecto en formato Gantt 16](https://upcomillas-my.sharepoint.com/personal/201706428_alu_comillas_edu/Documents/IIT/TFM/Report%20DOCS/TFM%20Miguel%20Oleo%20Blanco.docx#_Toc137571995)

[Ilustración 4: Ejemplo de múltiples carteras. Uniswap. 20](#_Toc137571996)

[Ilustración 5: Ejemplo de prueba en Metamask Playground [15] 22](#_Toc137571997)

[Ilustración 6: Imagen de la documentación creada de Metamask 22](#_Toc137571998)

[Ilustración 7: Popup de confirmación wallet\_switchEthereumChain 23](https://upcomillas-my.sharepoint.com/personal/201706428_alu_comillas_edu/Documents/IIT/TFM/Report%20DOCS/TFM%20Miguel%20Oleo%20Blanco.docx#_Toc137571999)

[Ilustración 8: Documentación creada de Ether.js 25](#_Toc137572000)

[Ilustración 9: Documentación creada de Web3.js 25](#_Toc137572001)

***Índice de tablas***

[Tabla 1. Tabla de datos 9](#_Toc137571965)

[Tabla 2: Desglose de costes estimados 17](#_Toc137571966)

[Tabla 3: Porcentajes de utilización según lenguaje de programación 18](#_Toc137571967)

[Tabla 4: Comparación Web3.js y Metamask 20](#_Toc137571968)

# Introducción

## Motivación del proyecto

La tendencia en todos los mercados es de, que el poder no recaiga sobre una misma entidad. Esto también ha pasado en el mundo tecnológico y financiero. Debido a esta tendencia, ha aparecido la tecnología blockchain. Esta tecnología permite crear una red descentralizada en la que se validan transacciones (monetarias o no). La principal ventaja es que, al ser un sistema descentralizado, no tenemos esa figura de una entidad reguladora detrás. Esta tecnología es compleja de entender, pero sencillamente es un conjunto de miles de máquinas, que validan y anotan transacciones que ocurren en esta red. Para que no haya fraudes, estas validaciones son comprobadas por muchos nodos y son matemáticamente complejas de obtener. También se mantiene un historial de todas las transacciones.

Desde que esta tecnología se estableció, han aparecido muchas aplicaciones que trabajan sobre el blockchain. Las principales aplicaciones son las relacionadas con la compraventa de criptomonedas. Este tipo de aplicaciones, como puede ser Binance [1], mueven una gran cantidad de capital. En 2022, el periódico El País indicó *“Las criptomonedas movieron unos 60.000 millones de euros en España”* [2].

Como todo mercado en el que hay un gran movimiento de capital y con una tecnología tan reciente, aparecen una gran cantidad de ataques que explotan las vulnerabilidades, con fin de sustraer de forma ilícita dichas monedas. Históricamente, han ocurrido una serie de ataques, como el que ocurrió en una plataforma de compraventa de NFTs (tokens no fungibles), donde se robaron fácilmente 310.00 euros [3]. En la sección del estado del arte se comentará más en profundidad este aspecto de la tecnología.

Este proyecto busca acabar con este tipo de fraude. Para ello, se va a investigar a fondo todos los aspectos relacionados con la seguridad de estas aplicaciones del blockchain, para que el usuario conozca la seguridad de dicha aplicación. Para ello, este proyecto tiene una gran fase de investigación para acotar los métodos de seguridad que se deben trabajar y determinar el tipo de ataques que se pueden llevar a cabo.

Para finalizar el proyecto, con toda la información recopilada de la fase de investigación y una serie de pruebas realizadas, se analizará el código fuente de una serie de aplicaciones en busca de las vulnerabilidades. Con los resultados obtenidos del análisis estático del código fuente, se creará una herramienta en la que el usuario indique la aplicación que quiere analizar y esta le indique si es segura de usar o no. Como último objetivo, la presentación de un *paper* donde se recojan todos los resultados obtenidos y que se publicará a la comunidad científica. Este último objetivo se llevará a cabo después de la defensa de este proyecto.

## Ejemplo de cómo documentar el trabajo

Este capítulo tiene como propósito explicar el uso de esta plantilla para su utilización en la redacción de la memoria de un Trabajo Fin de Grado/Master (TFG/TFM) de la Escuela Técnica Superior de Ingeniería (ICAI) de la Universidad Pontificia Comillas de Madrid.

El uso de esta plantilla no es obligatorio. Se ha preparado con el doble objetivo de ahorrar a los alumnos el desarrollo de una plantilla propia y que desarrollen una memoria con un estilo formal.

Al abrir esta plantilla con Word se crea un nuevo documento basado en esta plantilla y con un texto que explicar su uso (el que estás leyendo ahora mismo). Este texto deberás irlo eliminando a medida que escribes tu memoria.

## Figuras

Generalmente la memoria incluirá un gran número de figuras. Algunas serán creación del alumno y otras provendrán de otras fuentes.

Aunque MS-Word permite dibujar, no es el entorno más adecuado. PowerPoint es una herramienta de Office más amigable para dibujar figuras. Se recomienda dibujar las figuras en PowerPoint y luego copiarlas y pegarlas en Word como metaarchivo mejorado. Esto mismo es aplicable a figuras creadas en otros entornos u obtenidas de otras fuentes.

Para evitar que las figuras “floten” dentro del documento Word es conveniente cambiar su formato. Para ello hay que pinchar en ellas con el botón derecho y seleccionar Formato de imagen. Saldrá un cuadro de diálogo con varias pestañas. Selecciona la pestaña Diseño y dentro de ella elige la opción "En línea con el texto" (Figura 1). Este cuadro de diálogo te permite también cambiar el tamaño de la figura.

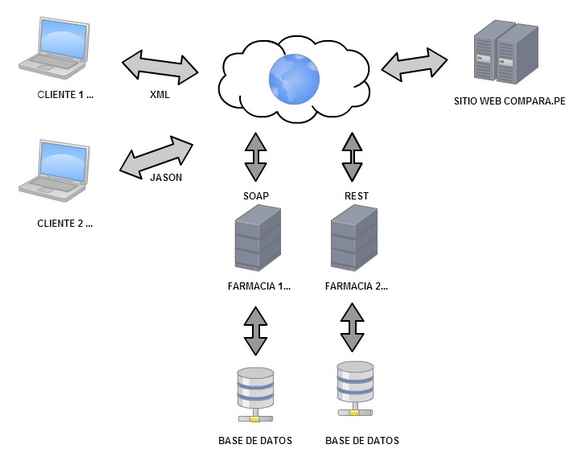


Figura 1. Diagrama de Arquitectura del Sistema [5]

Debajo de las figuras conviene poner un pie en el que se indique el orden de la figura y una descripción. Para ello se utiliza el menú Referencias → Títulos → Insertar Título. En aquellas figuras que no sean creación del alumno se deberá indicar la fuente entre paréntesis.

**Importante: No se inserta figura si no se referencia en ningún sitio. Es necesario explicar las figuras. Se recomienda que de cada idea importante se haga una figura “más vale una imagen que mil palabras”.**

Para referenciar una figura se hace (Referencias → Títulos →Referencia Cruzada → Sólo rótulo y número) en el lugar que se quiere insertar.

## Ecuaciones

Word tiene incorporado un editor de ecuaciones razonablemente bueno. Para insertar una ecuación hay que utilizar el menú Insertar → Objeto…→ Microsoft Editor de ecuaciones 3.0.

Después de escribir la ecuación con el editor de ecuaciones se hace (Referencias → Títulos → Insertar Título → E.) a la izquierda de la misma. (Ojo: esto sólo es necesario en caso de que en algún sitio se vaya a referenciar la ecuación). Ejemplo:

E. 1 

Para referenciar una ecuación se hace (Referencias → Títulos →Referencia Cruzada → E → Sólo rótulo y número) en el lugar que se quiere insertar. Ejemplo, la ecuación E. 1*...*

## Elementos de numeración

En estos elementos se incluyen los capítulos, secciones, subsecciones, ... y bibliografía.

Para introducir una referencia bibliográfica se elige el estilo “referencias”. Se pone primero el autor, luego el título, libro o revista donde se encuentra o congreso donde se presentó, año, paginas, volumen, etc:

Para cambiar de sección seleccionar el estilo correspondiente "Titulo","Titulo 1",....

Para referenciar un elemento de numeración se hace (Referencias → Títulos →Referencia Cruzada → Elem de numeración → Número de párrafo). Ejemplo. Como se describió en [1].... Como se comentó en la sección 1.4.... Como se apuntó en el Capítulo 1. ...

**Se debe seguir el mismo esquema para las figuras**, puesto que son elementos numerados.

## Tablas

Las tablas tienen que tener el formato Clásico 1 (Pinchar en Tabla → Diseño → Tabla clásica 1). Los elementos pueden estar centrados o como convenga. Después de insertar la tabla se hace (Referencias → Títulos → Tabla). Ejemplo

|  |  |
| --- | --- |
|  | Adadasd |
| Ad | 12 |

Tabla 1. Tabla de datos

Para hacer referencia a una tabla se hace (Referencias → Títulos →Referencia Cruzada → Tabla → Sólo rótulo y número) en el lugar que se quiere referenciar.

## Código fuente de programas

Usar estilo "Código Fuente":

var map;

function initialize() {

map = new google.maps.Map(document.getElementById('map-canvas'), {

zoom: 8,

center: {lat: -34.397, lng: 150.644}

});

}

google.maps.event.addDomListener(window, 'load', initialize);

## Actualización de campos

Los títulos de secciones, figuras, tablas, etc. así como los índices en realidad son campos que tienen código detrás. En ocasiones pueden no estar actualizados. Por ejemplo, los índices no se actualizan automáticamente a medida que se escribe el documento, sino que hay que actualizarlo cuando se desee.

Para actualizar un campo basta seleccionarlo con el ratón y pinchar con el botón derecho. Al hacerlo debe aparecer un cuadro de diálogo que incluye la opción "Actualizar campos". Se pincha en ella y el campo queda actualizado.

## Pie de página y encabezado

No olvidar actualizar los encabezados.

# Descripción de las Tecnologías

Para llevar a cabo el correcto desarrollo de este trabajo, se han empleado una serie de herramientas que nos facilitan el trabajo de investigación y la obtención de resultados. Las principales herramientas que se emplean se centran en la parte de investigación, ya que es la fase del proyecto más amplia.

La principal herramienta de este proyecto es Metamask [4]. Esta herramienta proporciona a los usuarios una interfaz y pasarela entre aplicaciones y el blockchain. Contiene una cartera virtual con las criptomonedas y tokens, las cuales se usan para interactuar con las aplicaciones descentralizadas. Uno de los principales usos de esta herramienta es facilitar a los usuarios el uso de las aplicaciones, gestionando todos las funciones de la blockchain. También facilita el trabajo de los desarrolladores de dichas aplicaciones, ya que implementa una serie de mecanismos y lógica. Mucha de esta lógica está disponible de forma de APIs (interfaz de programación de aplicaciones), por lo que los desarrolladores solo se limitan a hacer llamadas a dichas APIs y Metamask gestiona el resto. Esta herramienta es el principal objetivo de investigación del proyecto.

Otra de las herramientas principales del proyecto es CodeQL [5]. Esta es una herramienta de GitHub [6] que permite buscar vulnerabilidades en aplicaciones dentro de una base de datos especificada. Esta herramienta trabaja con análisis estático de código, el cual será explicado detalladamente durante el desarrollo de este trabajo. En este proyecto, se emplea esta herramienta para obtener resultados sobre la seguridad de las aplicaciones. Esta herramienta es muy útil ya que, el código fuente de las aplicaciones que se investigan, no es viable de estudiar de forma manual. Debido a ello, hay una necesidad intrínseca de emplear una herramienta de análisis como es CodeQL.

Otra de las herramientas principales es GitHub [6]. Esta herramienta proporciona un repositorio virtual en internet, donde compartir el código generado para el proyecto. Específicamente, para este proyecto, se trata de una herramienta muy útil, ya que se compone de varios miembros. Esta herramienta es muy usada en equipos de desarrollo por su facilidad de trabajar sobre el mismo repositorio y gracias a las funciones que proporciona para la gestión de los cambios y versionado.

Por otro lado, y también relacionado con el código, se empleará NodeJS [7]. Esta herramienta proporciona un entorno de desarrollo y ejecución de aplicaciones JavaScript. Principalmente, esta herramienta será usada para la creación de una aplicación descentralizada, en la que se llevarán a cabo pruebas.

Por último, se emplearán otra serie de herramientas conocidas por todos, por lo que no se citarán en detalle. Entre estas herramientas están Google Drive, que nos permite crear documentaciones y compartirlas de forma eficiente con el resto de los miembros. También se emplean códigos de programación como pueden ser JavaScript o TypeScript. Estos dos lenguajes en realidad parten de uno mismo y, como se citará en este proyecto, son los más empleados en la industria de las aplicaciones descentralizadas del blockchain.

# Estado de la Cuestión

Desde la aparición de la tecnología blockchain, ha habido una gran serie de ataques y robos de una gran suma de dinero a las aplicaciones descentralizadas. Un ejemplo de estos ataques puede ser el robo de NFTs por un valor de 310.00 euros [3]. Debido a esto hay una necesidad global por parte de la industria de trabajar por la solución y securización de las aplicaciones (ya que la tecnología de blockchain ya es segura de por sí).

Debido a esto, desde hace años se está haciendo un gran esfuerzo por parte de la industria por introducir nuevos mecanismos que acaben con las estafas, robos de transacciones y de las claves privadas de los usuarios. Las principales novedades de seguridad han sido introducidas por las distintas carteras de criptomonedas, como puede ser Metamask. Esta última ha ido variando sus mecanismos de seguridad a la vez que se detectaban los fallos de seguridad. Todos estos mecanismos modernos serán citados a lo largo de este proyecto.

Cada vez más, la responsabilidad recae sobre el usuario, ya que la tecnología actual de estas aplicaciones no es capaz de asegurar que nunca van a ser atacadas. Hay una serie de recomendaciones las cuales el usuario debe de seguir, al igual que una serie de acciones que nunca se deben de realizar, como compartir claves.

Ninguna plataforma y, mucho menos si se mueven grandes cantidades de dinero, debería recaer toda la responsabilidad sobre el usuario. La plataforma debería de aportar toda la seguridad que dispone en sus manos. Debido a esto, con este trabajo se busca mejorar y facilitar al usuario información sobre la seguridad de las aplicaciones. De esta forma (ya que las aplicaciones nunca serán 100% seguras), el usuario podrá comprobar si la aplicación es segura de utilizar (por parte de la plataforma). Se le proporcionará al usuario una herramienta que indicando el código fuente de la aplicación, esta herramienta lo analizará y avisará al usuario sobre el nivel de seguridad de esta aplicación.

# Definición del Trabajo

## Justificación

Según lo mencionado en las secciones de introducción y del estado del arte, la principal motivación de este proyecto es acabar con las estafas y robos que ocurren en las aplicaciones descentralizadas del blockchain y en las cuales se desperdician millones de dólares al año por este motivo. Un claro ejemplo es uno de los robos más recientes, en el que se robaron al instante 24 millones de euros en criptomonedas [8]. En la era digital en la que nos encontramos no se puede tolerar un “mercado” con tantos fallos de seguridad.

La herramienta que se va a emplear en este trabajo permite al usuario determinar la seguridad de dichas aplicaciones. Con ello, se pretende que el usuario final tenga un mayor conocimiento a priori sobre la aplicación, debido a que la información sobre la seguridad no suele ser pública en muchos casos.

Gracias al estudio que se lleva a cabo en este proyecto, también se generará unos resultados estadísticos los cuales se presentarán a la comunidad científica, ya que son datos representativos de este mercado.

## Objetivos

Como ya se ha citado de forma reiterada, el principal objetivo del proyecto es que el usuario final de las aplicaciones del blockchain, tenga conocimiento sobre la seguridad de la aplicación que va a usar. Para ello, el principal tangible de este proyecto será la creación de una herramienta que sea capaz de determinar el nivel de seguridad de la misma y se la presente al usuario.

El segundo objetivo de este proyecto es que, con la información recabada de analizar una serie de aplicaciones con la herramienta, generar un estudio sobre la seguridad del mercado de aplicaciones descentralizadas. Este estudio se presentará en formato *paper* a la comunidad científica después de la defensa de este trabajo.

Debido al enfoque de investigación de este proyecto, hay una serie de objetivos que se han ido generando mientras se trabajaba por conseguir el objetivo principal. Uno de estos subobjetivos más importantes es la creación de una documentación detallada sobre los métodos que proporciona Metamask. Aunque este sea un objetivo secundario, es de igual importancia, debido a que la documentación disponible por Metamask y Ethereum es escasa y en muchos casos, no concuerdan entre ellos.

## Metodología

Este es un trabajo puramente de investigación como se citó en el anexo B. Se debe de realizar una fase muy extensa de investigación sobre la API de Metamask, estudio de las DApps con más tráfico, De estas aplicaciones también se investiga el lenguaje de programación más empleado y la presencia de algunas APIs críticas para la seguridad de la misma.

Al tratarse de una investigación en profundidad, es necesario también contrastar la información con la documentación disponible de las distintas organizaciones (Metamask, Ethereum, etc). Esto también conlleva que surjan muchos imprevistos o que se encuentren otros objetivos durante el desarrollo de este.

Como ya se ha citado anteriormente, la metodología que se va a seguir para el correcto desarrollo de este proyecto es *Agile*. Esta metodología nos permite ir iterando el producto final en pequeños periodos de tiempo también conocidos como *Sprints*. Esta metodología es clave para un proyecto de desarrollo por lo citado, se va iterando sobre el mismo e incorporando *features* nuevas. Debido a esto, en un trabajo como este, da cabida a imprevistos o nuevos objetivos mientras se realiza la investigación.

## Planificación y Estimación Económica

En cuanto a la planificación del proyecto, se ha seguido la misma citada en el anexo B. A continuación, se muestra el cronograma en formato Gantt. En este se puede observar como la gran mayoría de *tasks* son de corta duración y tienen una gran fase de investigación previa. Para finalizar, se acaba con el desarrollo de la herramienta y la evaluación de la misma para concluir con unos datos empíricos que poder compartir con la comunidad científica.

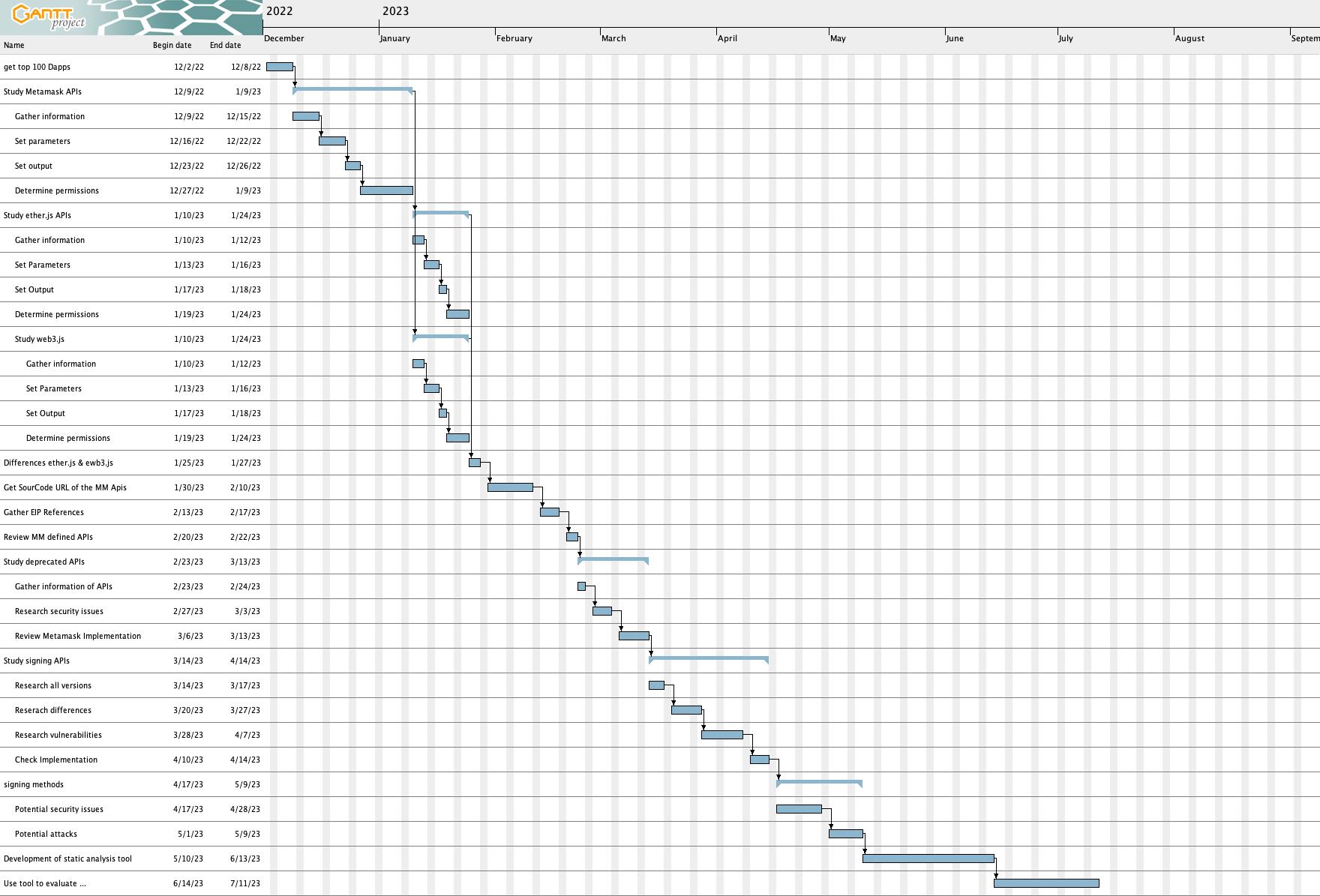


Ilustración 3: Cronograma del proyecto en formato Gantt

En cuanto a la estimación económica, este proyecto no conlleva ningún coste directo. El único coste que se podría llegar a contemplar es el pago de una serie de ingenieros de software para el mantenimiento de la herrmanienta. También se podría contemplar el coste de puesta en marcha y mantenimiento de un pequeño servidor y página web donde contener la aplicación para su descarga online. Si tenemos en cuenta estos costes, podemos desglosar el coste total anual en:

|  |  |  |
| --- | --- | --- |
|  | **Coste estimado** | **Coste total** |
| **Ingeniero de SW** | 35.000 | 36.000 |
| **Mantenimiento Servidor y Web** | 1.000 | 36.000 |

Tabla 2: Desglose de costes estimados

# Sistema/Modelo Desarrollado

En este capítulo se van a tratar todos los temas relacionados con el propio desarrollo del proyecto para alcanzar los resultados. Este se va a estructurar en:

* Análisis de los lenguajes empleados

## Análisis de los lenguajes empleados

La primera actividad a realizar en el trabajo es determinar el lenguaje de programación más empleado entre las aplicaciones del blockchain. Para ello y empleando la página de DappRadar [9], se ha creado un documento para analizar el lenguaje de las 100 aplicaciones con más tráfico de usuarios en internet. A partir de este documento se obtiene una estadística sobre los lenguajes de programación, el cual concluya que el más usado es TypeScript con un 52%.

|  |  |
| --- | --- |
| **Lenguaje de programación** | **Porcentaje** |
| TypeScript | 52% |
| JavaScript | 29% |
| Desconocido | 15% |
| Python | 3% |
| Cadence | 1% |

Tabla 3: Porcentajes de utilización según lenguaje de programación

Como se observa en la tabla, se puede concluir que prácticamente todas las aplicaciones están escritas en TypeScript y JavaScript. Estos dos lenguajes son prácticamente idénticos, ya que TypeScript está basado en JavaScript y añade una serie de modificaciones. Este dato será muy importante más adelante en el proyecto para cuando se inicie la fase de análisis del código.

## Implementación del blockchain

Hay distintos métodos para implementar los métodos del blockchain en una aplicación. Todos estos tienen sus ventajas y desventajas y se comentarán en esta sección.

La primera forma es implementar todos los métodos de una cartera del blockchain (Metamask, Binance, etc). Este método requiere que cada uno de los métodos que se definen en las documentaciones de estas carteras, sean implementados. Si se emplea esta opción, se nos ofrece el “poder” de poder controlar todos los aspectos de las aplicaciones, siendo los desarrolladores los que limitan la funcionabilidad de esta. Por otro lado, el principal problema es que, si queremos que la aplicación soporte más de una *wallet*, hay que programar la aplicación con tantas versiones como *wallets*, lo cual no es muy eficiente y mantenible.

La otra opción disponible es la programación de todos estos métodos a través de las librerías Web3.js [10] o Ether.js [11]. Estas dos son las librerías principales estudiadas en este trabajo, ya que se investiga únicamente aplicaciones que trabajan sobre el blockchain de Ethereum. Todas las funciones citadas que se debían implementar de las *wallets*, también se deben de implementar con estas librerías. La diferencia es que, no se implementarán las de cada cartera en específico, si no la función genérica que proporcionan las librerías. Debido a esta diferencia, se proporciona mucha flexibilidad a la hora de la programación de la aplicación ya que, estas librerías aceptan el uso de cualquier cartera compatible. Es decir, se programa la aplicación con los métodos genéricos de las librerías y se le permite al usuario conectar su cartera.

Otra ventaja y la cual es clave para el desarrollo de este proyecto es que, cuando se ha programado la aplicación con estas librerías y se conecta una cartera, los métodos de las librerías traspasan la responsabilidad a la cartera. Es decir, de cara al usuario es como si se hubiese programado con la cartera directamente. Esto también implica a los métodos que requieren permisos del usuario, como se citará a posteriori. Gracias a estas librerías se consigue que, con una sola versión de la aplicación, se soporten varias carteras. A continuación, se muestra un pequeño ejemplo de un mismo método en Web3.js comparado con Metamask (este último solo muestra el contenido de la petición API). Este método se usa para solicitar información para saber si se está minando criptomonedas o no:

|  |  |
| --- | --- |
| **Web3.js** | **Metamask** |
| Web3.eth.isMining([callback]) | {  "jsonrpc": "2.0",  "method": "eth\_mining",  "params": [],  "id": 0  } |

Tabla 4: Comparación Web3.js y Metamask

Un ejemplo visual de cómo son las aplicaciones que utilizan estas librerás puede ser Uniswap [12], la cual es la DApp con más tráfico de internet. Esta aplicación proporciona a los usuarios una plataforma donde intercambiar, comprar o vender criptomonedas. Como se puede observar en la siguiente imagen, esta plataforma acepta conectar distintas carteras y, como ya se ha citado, esto se consigue con las librerías Ether.js y Web3.js:

Captura de pantalla de un teléfono celular

Descripción generada automáticamente

Ilustración 4: Ejemplo de múltiples carteras. Uniswap.

En el caso de esta aplicación, si se investiga el código fuente [13], se observa que la librería empleada para la implementación de los métodos del blockchain es Web3.

Una vez visto que, independientemente del lenguaje de implementación del blockchain, la lógica que predomina es el de la cartera, se procederá a estudiar los métodos proporcionados por Metamask.

## API de Metamask

Esta es una de las fases más importantes del proyecto, ya que al estudiar los métodos que proporciona Metamask, también investigaremos cuales de ellos están relacionados con la seguridad.

Para la obtención de la información, se ha empleado la documentación que Metamask pone a disposición de usuarios y desarrolladores. Esta documentación es escasa y poco precisa, como se citará más adelante. Estas dos fuentes de documentación son, la página oficial de la documentación de Metamask [14] y una página oficial [15] para probar los distintos métodos. En cuanto a la primera documentación, es muy escasa, ya que solo recoge algunos métodos muy selectos. Entre estos métodos están los restringidos, los cuales son los métodos que requieren permisos definidos en el EIP-2255 [16] (los documentos tipo EIP serán comentado más adelante). Debido a esto, los métodos de esta documentación son muy limitados, e incluso, muchos no están bien desarrollados. Por otro lado, la pagina de pruebas de Metamask contiene casi todos los métodos disponibles. En esta página también se explica brevemente la funcionalidad de los métodos, estructura de los parámetros de entrada y de la respuesta, así como cuales de ellos son obligatorios y cuales opcionales.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración 5: Ejemplo de prueba en Metamask Playground [15]

Con la información recabada de estas dos documentaciones, se ha creado un documento tipo Excel que contienen todos los métodos que proporciona Metamask (y los heredados de Ethereum), una breve descripción, los parámetros obligatorios, los parámetros opcionales, si hace falta confirmación por parte del usuario y el formato de la respuesta.

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

Ilustración 6: Imagen de la documentación creada de Metamask

Como se puede observar se han insertado también las direcciones al repositorio de GitHub de Metamask [17] y algunas columnas extra que se comentarán más adelante. El principal objetivo de esta fase es documentar de forma clara y sencilla todos los métodos que se proporcionan en la API de Metamask. También en esta fase es muy importante determinar cuál de estos métodos son los que requieren permisos o confirmación por parte del usuario. Como ya se ha citado, estos son los métodos objetivo del estudio de este proyecto, ya que esos métodos son los que hacen que una aplicación sea segura o no.

Para determinar que métodos requieren permisos o confirmación, primero se investiga en el *Playground* de Metamask [15] donde, al ejecutar la prueba de ese método, si aparece el *popup* de la extensión de Metamask, se puede determinar si es necesaria la confirmación. Un ejemplo de ello es el método *wallet\_switchEthereumChain* que, al ejecutarlo, nos salta la pestaña de Metamask para aprobar el cambio de la cadena.

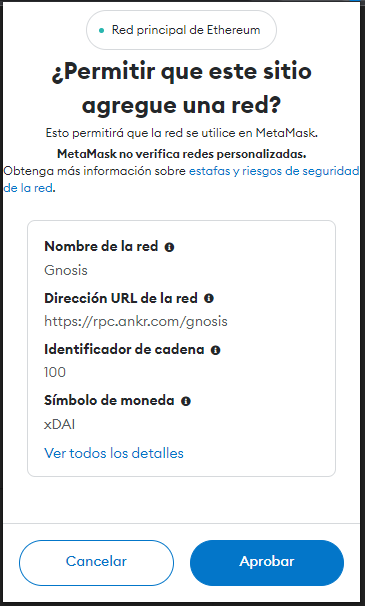


Ilustración 7: Popup de confirmación wallet\_switchEthereumChain

Hay otros de estos métodos los cuales no es tan sencillo determinar si necesitan confirmación o no. En estos casos, se debe analizar la implementación de dichos métodos en el repositorio de GitHub de Metamask [17]. Por ello, como se citó anteriormente, a la documentación se añadió la dirección web de donde están implementados dichos métodos dentro del código fuente y especialmente la documentación del EIP.

El EIP (*Ethereum Improvement Proposals*) es la documentación detrás de Ethereum como “organización”. En los EIP se encuentran sobre todo los métodos que se implementan en Ethereum así como las buenas prácticas, ejemplos y en algunos casos, el código completo.

Para determinar si un método requiere permisos por parte del usuario a partir del EIP, se debe de leer con detalle toda la especificación. En ella puede aparecer mencionado que el usuario debe de confirmar o incluso, puede que se tenga que determinar a partir del código. En muchos casos, viendo como se genera el resultado de del método. De esta última forma, el resultado muchas veces se especifica cuando:

* El proceso es correcto y el usuario acepta.
* El usuario rechaza la petición de permisos.
* Otros casos.

Un ejemplo claro de esto es el método *wallet\_addEthereumChain*. Al consultar el EIP número 3085 [18], que es el relacionado con este método, se explica tanto en texto como en la sección del resultado, que el usuario debe aceptar o rechazar y en cuanto al resultado del método, especifica cómo debe ser en caso de rechazo o aceptación.

## API Ether.js y Web3.js

Al igual que ocurre con Metamask, las librerías de Ether y Web3 también porporcionan una documentación de los métodos que se ofrecen a los programadores. Como ya se ha citado anteriormente, en las aplicaciones programadas con estas librerías, realmente la responsabilidad sigue recayendo sobre la lógica de la cartera que el usuario connecte. Debido a esto, la lógica y complejidad del desarrollo de aplicaciones con Ether.js o Web3.js, es mucho menor.

La documentación de estas dos librerías es mucho más extensa y mucho más fácil de entender de cara a los programadores (esto también se debe a que son menos complejas). Cono se ha mostrado anteriormente, el aspecto de estos métodos no son como los de Metamask (que son más peticiones API), si no que son más parecidos a una función o método de un lenguaje de programación orientado a objetos.

En este proyecto no se ha estudiado la seguridad de estas dos librerías, si no que el trabajo se centra en las APIs de Metamask. Debido a que se puede vincular la cartera de Metamask con aplicaciones desarrolladas en Ether y Web3, con estudiar Metamask, estudiamos también estas dos últimas librerías. Debido a esto, en este apartado, se ha documentado método a método, al igual que en la sección 5.3, con parámetros obligatorios, opcionales, retorno del método, etc. Lo más importante de esta sección es documentar los métodos que corresponden a los de Metamask. Por ello, en el documento tipo Excel, se han creado tres hojas de cálculo (una para Metamask, otra para Ether.js y otra para Web3.js), y los números de columna concuerdan con los métodos. Es decir, el método de metamask en la fila X, corresponde con los mismos métodos, pero de EtherJS y Web3JS en la misma fila X.

Como se puede observar en las siguientes ilustraciones, la estructura de la documentación sigue el mismo formato que el visto en la sección 5.3.

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

Ilustración 8: Documentación creada de Ether.js

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Ilustración 9: Documentación creada de Web3.js

## Problemas en la documentación

Durante los procesos de documentación de los distintos métodos (específicamente los de Metamask), ha surgido un problema no esperado y que es digno de un estudio aparte. Al crear la documentación citada en el apartado 5.3, se iba contrastando la información, que hay que recalcar que era escasa, de la documentación de Metamask, con la extensa documentación del EIP. Este tipo de “choques” de las dos documentaciones, normalmente afecta a la seguridad de las aplicaciones.

Durante este proceso, se ha concluido que en muchos casos, lo que se implementa no es lo que se recomienda. Un claro ejemplo de esto es el método *wallet\_addEthereumChain*. En este método, Metamask y el EIP especifican una serie de parámetros obligatorios y opcionales. En concreto este ejemplo es muy bueno ya que, en el EIP, estos parámetros opcionales, cita que deberían de ser obligatorios (aunque no estén especificados como tal). Esto se debe a que esto se trata de buenas prácticas, lo que no es obligatorio (aunque prácticamente lo sea).

Esto ocurre en muchos métodos. Debido a esto, el primer resultado que se crea con este proyecto es la documentación creada durante la fase de investigación. Todas estas diferencias se han añadido a la hoja de Excel donde están contenidos todos los métodos de Metamask. Por ejemplo, se puede ver en la siguiente imagen un ejemplo de la documentación que se ha creado. Este método es *wallet\_watchAsset*, donde metamask especifica todos los parámetros como opcionales, mientras que en el EIP (en rojo), especifica dos parámetros como obligatorios:

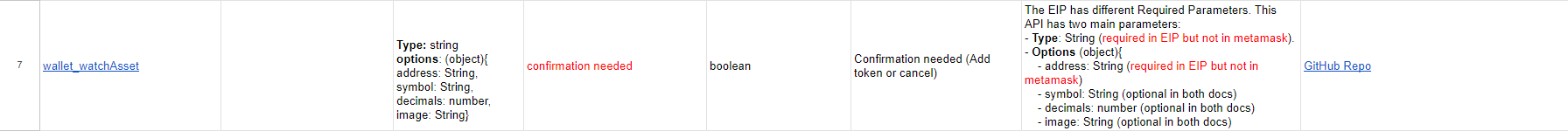


Ilustración 10: Ejemplo de colisión de la documentación en wallet\_watchAsset

## Métodos de firma

En el blockchain es necesaria la implementación de distintos métodos de firma. Esto se debe a que, por seguridad, toda la información que se quiere transmitir a través de esta cadena de bloques se firma por la seguridad e integridad de los datos, y para confirmar que el emisor es el esperado.

Con el paso de los años, se han ido descubriendo nuevas formas más seguras de firmas y lo mismo ha ocurrido. Metamask [19] tiene distintos métodos de firma que se han ido implementando con el tiempo. Los principales son:

* *Eth\_sign*
* *Eth\_personalSign*
* *Eth\_sigTypedData\_v4*

### *eth\_sign*

El primer método es el menos seguro, ya que en el *popup* que se le presenta al usuario, el mesaje de la firma no es legible. Por ello, muchos ataques se basaban en presentarle al usuario esta firma, la cual como no se puede leer, lo firmaban pensando que es una transacción legítima, cuando en realidad la aplicación les hace firmar otra cosa. Debido en el momento en el que se inició este estudio, este método no se recomendaba. Ahora en el momento que se está realizando este documento, el propio Metamask a catalogado este método de firma como deprecado. Para saber más de estos métodos, ver Anexo X. Como se aprecia en la siguiente imagen, el mensaje a firmar es una cadena hexadecimal que surge de realizar el hash de la firma, por lo que el usuario no lo puede entender.

Debido a la inseguridad de esto, Metamask en su propia aplicación, en los ajustes avanzados, ahora permite desactivar las peticiones a este método. Gracias a esto y, a que se disponen de los otros métodos que se citarán a continuación, Metamask no dejará a las aplicaciones usar este tipo de firma [20]. En general, las aplicaciones ya usan los otros métodos más modernos, por lo que la aplicación funcionará sin problema. Aun así, muchas aplicaciones siguen empleando el método *eth\_sign* como método de firma por lo que, en caso de tenerlo desactivado, estas aplicaciones no dispondrían de toda la lógica, haciendo que parte de su funcionamiento sea inutilizado.

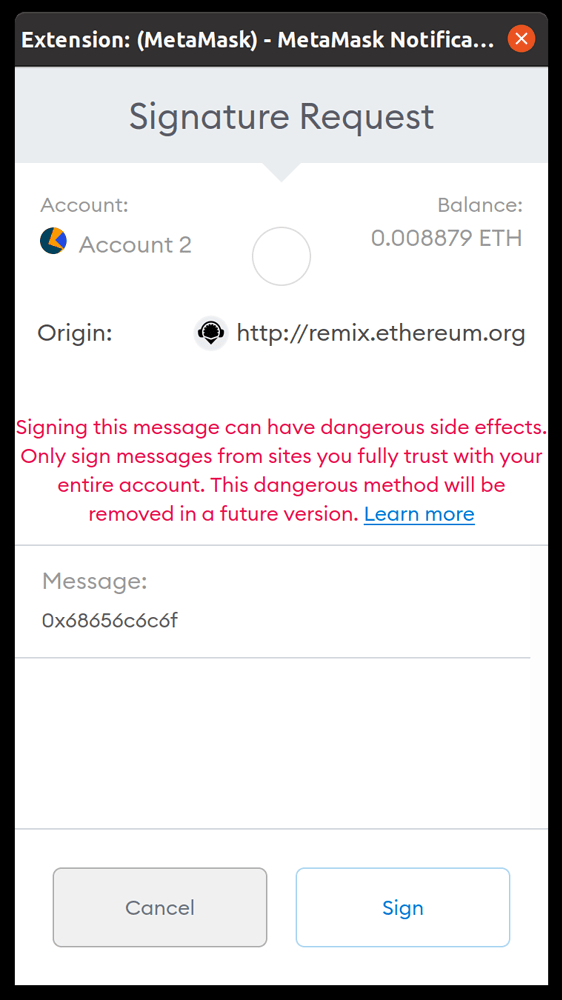


Ilustración 11: Ejemplo de eth\_sign

### eth\_personalSign

Este fue el método que Metamask introdujo para solucionar que el mensaje de la firma no sea legible. Esta función es la forma más fácil de implementar. El principal problema de este método y el siguiente es que, como el mensaje si es legible, hace que sea menos eficiente para ser procesado en el blockchain.

Este método en específico es poco eficiente y por ello surgió el que se comentará en la siguiente sección. Aun así, muchas aplicaciones todavía usan este método debido a que es muy simple. A continuación, se muestra el aspecto del mensaje que se le presenta al usuario al ejecutar este método:

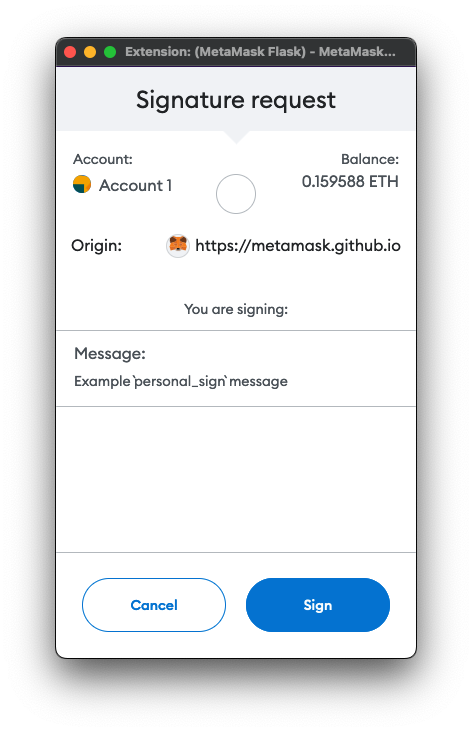


Ilustración 12: Ejemplo de eth\_personalSign

### Eth\_signTypedData\_v4

Este es el método más moderno implementado en Metamask. Este es el método en teoría más seguro y flexible. Esto último se debe a que, permite varias direcciones de envío, varias de destino y múltiples parámetros. Más adelante, se comentará los problemas que conlleva tener tanta flexibilidad. A continuación, se muestra el aspecto del *popup*:

Como se puede observar, en la confirmación se incluye unos datos estructurados (from, wallets, etc). Este método es el que implementa de forma más eficiente el hecho de que sea legible por parte del usuario. Esto conlleva que el código de implementación es más complejo.

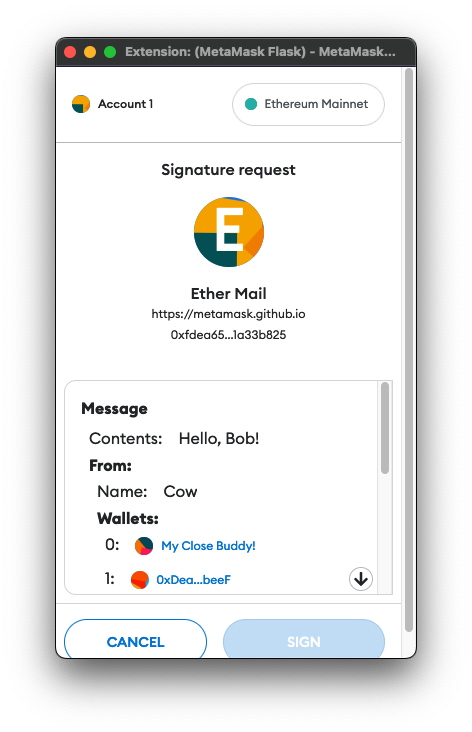


Ilustración 13: Ejemplo de eth\_signTypedData\_v4

#### Estudio de la implementación

Debido a que este método es el más moderno y tiene más seguridad de los tres, se procederá a estudiar en detalle su implementación y posibles problemas de seguridad. Para ello, primero se empieza por estudiar cómo debe implementarse y definirse.

Para entender en profundidad como funciona y como se implementa este método, se debe consultar el EIP-712 [21][22]. En estas documentaciones hay 4 pasos muy concretos que se van a comentar a continuación:

1. Diseñar la estructura de los datos: En este paso, se va a definir qué datos se van a firmar. Este método requiere que todos los datos estén estructurados y con la definición de sus tipos. La mejor forma de entender este concepto es viendo un ejemplo. A continuación, se muestra un archivo JSON (solo la parte correspondiente a este punto) que contiene todo lo necesario para poder ser usado en el método *eth\_signTypedData\_v4*.

"message": {

"contents": "Hello, Bob!",

"from": {

"name": "Cow",

"wallets": [

"0xCD2a3d9F938E13CD947Ec05AbC7FE734Df8DD826",

"0xDeaDbeefdEAdbeefdEadbEEFdeadbeEFdEaDbeeF"

]

},

"to": [

{

"name": "Bob",

"wallets": [

"0xbBbBBBBbbBBBbbbBbbBbbbbBBbBbbbbBbBbbBBbB",

"0xB0BdaBea57B0BDABeA57b0bdABEA57b0BDabEa57",

"0xB0B0b0b0b0b0B000000000000000000000000000"

]

}

]

},

Este código muestra la creación de la estructura de los datos y su definición (asignación de valores). También hay que definir los tipos de estas estructuras y sus variables, definido en types:

"types": {

"Group": [

{

"name": "name",

"type": "string"

},

{

"name": "members",

"type": "Person[]"

}

],

"Mail": [

{

"name": "from",

"type": "Person"

},

{

"name": "to",

"type": "Person[]"

},

{

"name": "contents",

"type": "string"

}

],

"Person": [

{

"name": "name",

"type": "string"

},

{

"name": "wallets",

"type": "address[]"

}

]

La asignación de los tipos es muy importante por lo que se explicará más adelante. También es muy importante ver que, todas las variables que se han establecido, tienen un tipo de variable asignada en la estructura Types.

1. En el segundo paso hay que diseñar el *domain separator*. Este es una estructura de datos, al igual que los del paso 1, que sirve para diferenciar la función de firma de dicha aplicación con otras que puedan existir (evita conflictos de funciones similares entre aplicaciones). A continuación se muestra la creación de la estructura del separador de dominio, sus asignaciones de valores y la estructura y definición de los tipos:

"domain": {

"chainId": "0x1",

"name": "Ether Mail",

"verifyingContract": "0xCcCCccccCCCCcCCCCCCcCcCccCcCCCcCcccccccC",

"version": "1"

}

"types": {

"EIP712Domain": [

{

"name": "name",

"type": "string"

},

{

"name": "version",

"type": "string"

},

{

"name": "chainId",

"type": "uint256"

},

{

"name": "verifyingContract",

"type": "address"

}

],

}

Al igual que antes, es muy importante que todas las variables usadas en el separador estén bien definidas con sus tipos. En el EIP-712 se especifica que, este separador debe de contener las variables:

* + ChainID: este parámetro contiene el identificador de la red en la que trabajan los contratos.
  + Name: Nombre que se le da al separador.
  + VerifyingContract: este contiene la dirección del contrato solidity que va a encargarse de certificar y validar las firmas de este método.
  + Version: Version empleada.
  + Salt: coste de la firma en tipo salt (esto es opcional).

1. En este paso se define y se programa el código encargado de realizar la firma.
2. Por último, se crea el código encargado de verificar el contrato y autentificar el código de firma.

Otro campo que se añade en el JSON que se firma es el *primaryType*. Este campo determina, de la lista de las variables, cual de ella es la principal. La estructura que se define como primaria, es la que se muestra en la parte superior (y principal) del *popup* que se le presenta al usuario. Debido a esto, es muy importante esta clase, ya que es la información que el usuario usará para determinar si todo está correcto y proceder a su firma.

### Creación de la base de datos

Uno de los objetivos principales de este trabajo es realizar un estudio en detalle sobre la seguridad de las aplicaciones reales que los usuarios usan alrededor del mundo todos los días. Para poder llevar a cabo este estudio, primero se debe de recabar información de estas. Por ello, para realizar el estudio sobre los tres métodos mencionados, ha habido la necesidad de crear una base de datos de la cual partir.

Esta base de datos se ha creado de forma manual, obteniendo las aplicaciones más usadas de internet. Para obtener este listado, se ha empleado la página de DappRadar [9]. Como ya se ha citado anteriormente, esta página nos proporciona un listado con las aplicaciones descentralizadas que más tráfico generan en internet. Este dato es muy importante, ya que las que más tráfico generan, son las más usadas (no hay aplicaciones que aporten una lógica tan distinta como para aportar mucho tráfico con pocos usuarios).

Gracias a esta página, se han obtenido alrededor de 800 aplicaciones. Sobre todas las aplicaciones obtenidas para el estudio, se ha anotado en un documento tipo Excel sus repositorios de código (GitHub), y la presencia de estos tres métodos. Para esto último, se ha tenido que analizar todo el código fuente de cada una de las aplicaciones en busca de estos tres métodos de uno en uno. En esta fase quiero recalcar el gran trabajo que hay detrás de esta fase, la cual es muy importante para el correcto desarrollo del proyecto, pero que no es un objetivo. A continuación, se muestra un pantallazo del aspecto de esta base de datos:

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 14: Ejemplo de la documentación de aplicaciones y los métodos

En las columnas de los métodos, los números azules que apareces en la tabla son el número de línea donde aparecen dichos métodos. Como estos repositorios tienen muchos archivos, y en muchos pueden aparecer, cada número tiene un hipervínculo al archivo.

Para la búsqueda de las líneas en las que aparece cada método, se empezó realizando una búsqueda con VSCode. Debido al tamaño de la base de datos, se ha creado un programa Shell que, recibe como entrada el nombre del archivo que contiene una lista de todas las direcciones de los repositorios. Posteriormente, pregunta por el nombre del archivo de salida, que contendrá el archivo y línea en la que se han detectado los métodos. Por último, pregunta si la petición de buscar los métodos se realiza en el contrato de verificación (solidity) o en el código fuente, como es este caso.

Esta herramienta se puede encontrar en el repositorio creado para este proyecto []. Para que esta herramienta funcione de forma correcta, también se debe de instalar una dependencia de GitHub [23].

# Análisis de Resultados

Destacar los resultados más relevantes del proyecto y hacer un análisis crítico de los mismos. También es un capítulo obligatorio y clave.

# Conclusiones y Trabajos Futuros

Comentar las conclusiones del proyecto, destacando lo que se ha hecho, dejando claros qué objetivos se han cubierto y cuáles son las aportaciones hechas.

# Bibliografía

1. Binance. (s. f.). *Binance – Exchange de criptomonedas para bitcoin, Ethereum y altcoins*. <https://www.binance.com/es>
2. Sánchez, Á. (2022, 26 abril). Las criptomonedas movieron unos 60.000 millones de euros en España el año pasado. *El País*. <https://elpais.com/economia/2022-04-26/las-criptomonedas-movieron-unos-60000-millones-de-euros-en-espana-el-ano-pasado.html>
3. Vanguardia, L. (2022, 10 agosto). Un hacker roba un NFT por valor de 310.000 euros y lo acaba vendiendo por casi la mitad. *La Vanguardia*. <https://www.lavanguardia.com/tecnologia/20220810/8459460/roban-nft-valor-310-000-euros-hacker-acaba-vendiendo-mitad-pmv.html>
4. The crypto wallet for Defi, Web3 Dapps and NFTs | MetaMask. (s. f.-b). <https://metamask.io/>
5. *CodeQL*. (s. f.). <https://codeql.github.com/>
6. *GitHub: Let’s build from here*. (s. f.-b). GitHub. <https://github.com/>
7. *Node.js*. (s. f.-b). Node.js. <https://nodejs.org/es>
8. Echarri, M., Echarri, M., & Echarri, M. (2022, 27 julio). El chico de 15 años que robó 24 millones en criptomonedas : Ellis Pinsky cuenta su historia pero nadie sabe si creerle. *El País*. <https://elpais.com/icon/2022-07-27/el-chico-de-15-anos-que-robo-24-millones-en-criptomonedas-ellis-pinsky-cuenta-su-historia-pero-nadie-sabe-si-creerle.html>
9. DappRadar - The World’s Dapp Store | Blockchain Dapps Ranked. (s. f.). DappRadar. <https://dappradar.com/>
10. web3.js - Ethereum JavaScript API — web3.js 1.0.0 documentation. (s. f.). <https://web3js.readthedocs.io/en/v1.10.0/>
11. *Ether.js Documentation*. (s. f.). <https://docs.ethers.org/v5/>
12. *Home | Uniswap Protocol*. (s. f.). Uniswap Protocol. <https://uniswap.org/>
13. *Uniswap Labs*. Código Fuente de Uniswap (s. f.). GitHub. <https://github.com/Uniswap>
14. Home | MetaMask developer documentation. (s. f.). <https://docs.metamask.io/>
15. *Metamask Playground* (De Metamask [Metamask]). (s. f.). Playground Metamask. <https://metamask.github.io/api-playground/api-documentation/>
16. Proposals, E. I. (2019). EIP-2255: Wallet Permissions System. *Ethereum Improvement Proposals*. <https://eips.ethereum.org/EIPS/eip-2255>
17. *MetaMask*. Repositorio de código de Metamask (s. f.). GitHub. <https://github.com/MetaMask>
18. Proposals, E. I. (2020). EIP-3085: Wallet Add Ethereum Chain RPC Method (`wallet\_addEthereumChain`) [DRAFT]. *Ethereum Improvement Proposals*. <https://eips.ethereum.org/EIPS/eip-3085>
19. *Sign data | MetaMask developer documentation*. (s. f.). <https://docs.metamask.io/wallet/how-to/sign-data/>
20. *What is «eth\_sign» and why is it a risk?* (2023, 12 mayo). MetaMask. <https://support.metamask.io/hc/en-us/articles/14764161421467-What-is-eth-sign-and-why-is-it-a-risk->
21. Proposals, E. I. (2017). EIP-712: Typed structured data hashing and signing. *Ethereum Improvement Proposals*. <https://eips.ethereum.org/EIPS/eip-712>
22. Jie, K. W. (2020, 20 enero). EIP712 is here: What to expect and how to use it - MetaMask - Medium. *Medium*. <https://medium.com/metamask/eip712-is-coming-what-to-expect-and-how-to-use-it-bb92fd1a7a26>
23. *Installation instructions for GitHub dependencies*. (s. f.). GitHub CLI. <https://cli.github.com/manual/installation>

# ANEXO I

….NOTA: Esta parte hace referencia a posibles Anexos opcionales de la memoria que se quieran añadir (código fuente, hojas de características, planos, etc.). NO se refiere a los documentos Anexos que se indican en la normativa (Anexo A, Anexo B, etc.)