

Instituto Tecnológico de Buenos Aires

escuela de ingeniería y gestión

Gazzeth: Una plataforma de noticias descentralizada contra la desinformación

Autores:
Agustín Dammiano
Alan Donoso Naumczuk

Tutor:
Dra. Alicia Mon

trabajo final presentado para la obtención del título de ingeniero en informática

Noviembre 2021

Resumen

La desinformación es un fenómeno que perjudica a sociedades de todo el mundo. Los servicios de *fact-checking* no logran resolver este problema de manera efectiva, en especial por sus características centralizadas y su falta de transparencia. Este documento describe Gazzeth, una plataforma de noticias descentralizada, *permissionless*, resistente a censura, transparente y global.

Palabras clave: *desinformación, noticias falsas, aplicación descentralizada, descentralización, blockchain, Web 3.0, Ethereum.*

Abstract

Disinformation is a phenomenon that harms societies all around the world. Fact-checking services fail to solve this problem effectively, especially because of their centralized characteristics and their lack of transparency. This document describes Gazzeth, a decentralized, permissionless, censorship-resistant, transparent and global news platform.

Keywords: *disinformation, fake news, decentralized application, decentralization, blockchain, Web 3.0, Ethereum.*

Agradecimientos a los Cypherpunks por promover el avance tecnológico y de la criptografía como herramienta social, a Satoshi Nakamoto por inventar Bitcoin y así inspirar nuevos desarrollos tales como Ethereum, a cuyos creadores también dedicamos esta sección, que hoy permite el desarrollo del proyecto que describe este documento.

Thanks to the Cypherpunks for promoting the advance of technology and cryptography as a social tool, to Satoshi Nakamoto for inventing Bitcoin and inspiring new developments such as Ethereum, to whose creators we also dedicate this section because their invention allowed the development of the project that this document describes.

Índice

1. Introducción **1**

2. Estado del arte **2**
2.1. Puntos focales 2
2.2. Kleros 3

3. Alcance del proyecto **4**
3.1. Funcionamiento básico 5

4. Requisitos **6**
4.1. Requisitos funcionales 6
4.2. Requisitos no funcionales 7

5. Atributos de calidad **8**
5.1. Seguridad 8
5.2. Transparencia 8
5.3. Disponibilidad 8

6. Solución propuesta **9**
6.1. Arquitectura 9
6.2. Usuarios 10
6.3. Tópicos 10
6.4. Publicaciones 11
6.4.1. Formato 11
6.4.2. Almacenamiento 11
6.5. Diseño de mecanismos 12
6.5.1. Depósitos y penalizaciones 12
6.5.2. Recompensas y el token GZT 13
6.6. Reglas para elección del voto 15
6.7. Commitment-Scheme 15
6.7.1. Esquema clásico 16
6.7.2. Esquema propuesto 17
6.7.3. Desventajas 20
6.8. Sybil-resistance 20
6.8.1. Ataque Sybil en Gazzeth 21
6.8.2. Solución 21
6.8.3. Desventaja 21
6.9. Apelaciones 22
6.9.1. Justificación del voto 23
6.10. Gobernanza 23
6.11. Fases para lanzamiento justo 25
6.11.1. Fase I: Lanzamiento inicial 26
6.11.2. Fase II: Curado de tópicos 26
6.11.3. Fase III: Lanzamiento final 27

6.11.4. Conclusiones sobre el mecanismo planteado 27
6.12. Notificaciones 27
6.13. Cliente 28
6.13.1. Tipos de clientes 28
6.13.2. API abierta 29

7. Prueba de concepto **30**
7.1. Protocolo 30
7.1.1. Herramientas utilizadas 30
7.1.2. Implementación 31
7.1.3. Red 33
7.2. Almacenamiento de noticias 33
7.3. API abierta 33
7.4. Cliente 34
7.4.1. Implementación 34
7.4.2. Estados de la noticia 35
7.4.3. Despliegue 36

8. Trabajo futuro **36**

9. Conclusión **37**

Anexos **40**

A. Costos de transacción **40**

B. Manual de uso **42**
B.1. Barra de navegación 42
B.2. Listado de noticias 42
B.3. Conectar una billetera 42
B.4. Manejar suscripciones como jurado 44
B.5. Publicar una noticia 45
B.6. Ver una noticia 46
B.7. Emitir voto 47
B.8. Revelar un voto 48
B.9. Distribuir ganancias 49

1. Introducción

La desinformación es un fenómeno que ocurre a nivel global, donde los medios, con distintos fines e intereses, apelan a las emociones de los ciudadanos en lugar de poner los hechos en el centro de la escena. Tener control sobre la opinión pública parece ser más valioso que informar con la verdad.

A principios del siglo XXI surgen en Internet las redes sociales, aplicaciones de mensajería, blogs, portales de noticias digitales y demás plataformas conformando lo que se conoce como *Web 2.0*. Esto aceleró drásticamente la velocidad de propagación de información y, en consecuencia, es cada vez más fácil esparcir noticias falsas causando impacto en la sociedad.

El periodismo independiente, que intenta mantenerse alejado de influencias de gobiernos, corporaciones u otras entidades poderosas, no logra tener la exposición que tienen los medios tradicionales.

En algunos países surgieron plataformas de *fact-checking* para resolver este problema. La idea es que un grupo de especialistas tome noticias relevantes o polémicas y se encargue de verificarlas, lanzando un nuevo artículo donde se exponen, con justificación, los puntos ciertos y/o falsos de cada una. Si bien en cada país en donde se implementó ha dado diferentes resultados, en general, el nivel de confiabilidad y adopción no fue alto.

En búsqueda de mejorar los servicios de verificación de información para aumentar su confiabilidad y adopción surgen algunas preguntas... ¿Cómo lograr que este servicio sea un bien público para todas las personas del mundo? ¿Cómo asegurar que funcione indiferentemente del tipo de gobierno, sea más o menos autoritario? ¿Qué ocurriría si una entidad intenta censurar el servicio? ¿Y si es el gobierno quien decide censurarlo? ¿Qué ocurriría si alguna entidad soborna a quienes se encargan de realizar las validaciones con el fin de obtener un determinado resultado? ¿Y si en vez de sobornados son coaccionados? ¿Quién financia el servicio? ¿Cómo encontrar una fuente de financiamiento que no implique una reducción en el nivel de confianza sobre el mismo? ¿Quién decide qué publicaciones o qué hechos validar? ¿Qué ocurre si uno o varios individuos no están de acuerdo con el resultado de una validación? ¿Tienen los mecanismos para hacer algo al respecto con impacto en la opinión pública? ¿Se puede saber cómo fue el proceso de validación de cada noticia? ¿Está a disposición para ser auditado por quienes consumen el resultado final? ¿Cómo se logra involucrar a los consumidores de noticias en el proceso?

El lanzamiento de Ethereum, en 2015, facilitó el desarrollo de protocolos descentralizados, brindando una arquitectura distribuida, segura y pública, dando lugar a un nuevo abanico de ideas y soluciones a los problemas.

Este documento presenta Gazzeth, una plataforma de noticias descentralizada que busca mitigar la desinformación dando soluciones a las problemáticas planteadas que los servicios de fact-checking actuales no logran solventar.

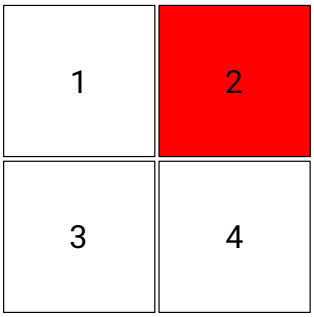
2. Estado del arte

2.1. Puntos focales

Se conoce como *Punto focal*[1], o *Schelling point* en honor a quien definió el concepto, a la opción que los humanos tienden a elegir por defecto cuando deben cooperar en ausencia de comunicación.

Esta solución suele ser escogida por sobre las demás por percibirse como la más intuitiva, ya sea por algún aspecto de su simetría o por parecer la más natural, y, por lo tanto, si el resto de las personas con las que se debe cooperar son racionales, se espera, con un grado alto de probabilidad, que la vean de esta manera también.

Por ejemplo, se pone a dos personas, Alicia y Bob, a jugar un juego sin ningún tipo de comunicación entre ellos. En el juego se les presenta la siguiente imagen:



Se les pide a los jugadores que elijan uno de los cuadrados indicándoles que, si su elección coincide con la del otro participante, serán premiados con 1 BTC cada uno. Por el contrario, si no eligen la misma opción, no se llevarán ningún premio. Así se ve la matriz de pagos del juego descripto:

		Alicia			
		1	2	3	4
Bob	1	$(1,1)$	$(0,0)$	$(0,0)$	$(0,0)$
	2	$(0,0)$	$(1,1)$	$(0,0)$	$(0,0)$
	3	$(0,0)$	$(0,0)$	$(1,1)$	$(0,0)$
	4	$(0,0)$	$(0,0)$	$(0,0)$	$(1,1)$

Nótese que no existe un cuadrado cuya elección represente una salida mejor o peor que las demás. No hay una respuesta correcta, sólo basta que Alicia y Bob coincidan en ella para llevarse el premio.

Quizás el 3 es el número favorito de Alicia y, por lo tanto, ella hubiera elegido el cuadrado con dicho número si se tratara de gustos. Sin embargo, dado que es racional y quiere ganar el premio, teniendo en cuenta que debe coincidir

con Bob, a quien también asume racional y, por lo tanto, espera que también quiera obtener el premio, lo más probable es que Alicia elija el cuadrado con el número 2, porque este es de un color distinto al resto, esperando que Bob también encuentre esto como una característica relevante y, en consecuencia, coincida en la elección. Esto es lo que hace que el cuadrado con el número 2 sea el punto focal de este juego.

¿Qué ocurriría si en lugar de cuadrados en una imagen tuvieran que escoger una fecha del año? Nuevamente, cualquier fecha, en tanto coincidan en ella, sería igual de correcta. Sin embargo, es probable que, en ausencia de comunicación, se elija el 1 de enero, por percibirse la primera fecha del año como la solución más intuitiva.

Este tipo de experimentos podrían aplicarse en verificación de noticias. Se les da a Alicia y a Bob una misma noticia pidiéndoles que la clasifiquen como *Verdadera* o *Falsa*. Sólo en caso de coincidir en dicha clasificación se llevarán el premio. La matriz de pagos de este experimento se ve así:

		Alicia	
		V	F
Bob	V	$(1,1)$	$(0,0)$
	F	$(0,0)$	$(1,1)$

Si Alicia y Bob no se conocen, no conocen las preferencias y sesgos del otro, y no tienen forma de comunicarse entre sí, se espera que tomen como estrategia común tratar de validar la noticia de forma objetiva, esperando que el otro participante haga lo mismo y de esta manera intentar coincidir en la clasificación elegida.

Existe la posibilidad de que ambos jugadores, por diferentes motivos, lleguen a clasificaciones distintas. El punto focal en este experimento es la estrategia elegida: la búsqueda de la objetividad.

Si este experimento se lleva a cabo con un número mayor de participantes y se ajustan aún más los incentivos económicos, por ejemplo agregando penalizaciones en caso de no coincidencia, es probable que la elección mayoritaria represente la clasificación objetiva para la noticia provista.

2.2. Kleros

Kleros[2] es un protocolo descentralizado de justicia y resolución de disputas que corre principalmente en Ethereum[3]. Este hace uso de mecanismos de incentivos, basados en penalizaciones y recompensas, para buscar la honestidad como punto focal en los jurados que actúan en las disputas que se ejecutan en su plataforma.

El protocolo cuenta con distintas cortes dependiendo de la temática a juzgar y requiere que los jurados se inscriban como tal en alguna de ellas dejando PNK, el token de Kleros, como garantía.

Luego, cuando se crea una disputa en una corte, los jurados son elegidos al azar dentro de los disponibles en ella, siendo la probabilidad de ser escogidos proporcional a la cantidad de PNK bloqueado en la correspondiente corte.

Los jurados deben votar eligiendo alguna de las opciones disponibles en la disputa, por ejemplo fallando a favor o en contra de quien la creó. Un jurado podría salir sorteado más de una vez para una misma disputa, en ese caso su voto tiene el peso equivalente a la cantidad de veces que haya sido elegido y también su recompensa o penalización será proporcional a ello.

Una vez finalizada la votación, parte del PNK de los jurados que votaron de forma minoritaria se reparte entre los que votaron mayoritariamente, siendo el valor votado por estos últimos el tomado como resolución de la disputa.

Además, se debe pagar una tarifa por el trabajo de los jurados, que será repartida entre los que votaron de forma mayoritaria. Quien pague esta tarifa dependerá de cómo esto se haya configurado, la cual, por ejemplo, sirve para que cuando todos los jurados voten de igual manera se garantice que igualmente obtengan un pago por su trabajo.

Kleros propuso una idea para combatir las noticias falsas utilizando su protocolo[1]. Esta se basa en crear una lista de noticias falsas, o múltiples de ellas clasificadas por idioma, región u otras características, donde cualquier usuario puede agregar noticias, o partes de ellas, que considera falsas dejando un depósito en garantía de ello. Luego, habría una ventana de tiempo en la cual cualquier otro usuario puede desafiar el agregado de dicha noticia a la lista.

Si no ocurre ningún desafío, la noticia queda en la lista de noticias falsas y se devuelve el depósito al usuario que la agregó. Por otro lado, si un usuario crea un desafío, deberá dejar un depósito y automáticamente se crearía una disputa en una corte de validación de noticias de Kleros.

A través del mecanismo explicado anteriormente, la corte resolvería la disputa indicando si la noticia es falsa o no. En caso de determinarse falsa, se devuelve el depósito al usuario que agregó la noticia a la lista y se utiliza el depósito del desafiante para pagar la tarifa de la corte. Si se determina verdadera ocurre lo contrario.

3. Alcance del proyecto

Este proyecto es de índole académica y experimental. El objetivo de este documento es detallar de forma teórica la propuesta de una plataforma de noticias descentralizada, gobernada por sus usuarios, *permissionless*, neutral, resistente a censura, transparente y global. Luego, describir aspectos relevantes de la prueba de concepto realizada, cuyo fin es demostrar la viabilidad del desarrollo del mismo.

Es relevante destacar por qué se decidió plantear la plataforma como un nuevo protocolo en lugar de seguir la propuesta de Kleros o hacer uso de este.

En primer lugar, si bien Kleros funciona muy bien como servicio de resolución de disputas donde las partes involucradas están dispuestas a pagar por ello, la

[1] https://blog.kleros.io/can-kleros-fight-fake-news/

forma en que Kleros plantea combatir las *fake news* mediante su protocolo tiene los incentivos desalineados. El usuario que se encarga de agregar noticias falsas a la lista, actividad que debiera ser incentivada, no obtiene nada a cambio como recompensa y, por el contrario, además de gastar en costos de transacción, corre el riesgo de estar equivocado y perder el depósito que puso en garantía.

Por otro lado, Gazzeth busca ser una plataforma de noticias, esto incluye tanto noticias falsas como ciertas, lo importante es que cada una sea clasificada como tal y que los periodistas tengan incentivo a publicar en ella. Que en la plataforma puedan encontrarse todo tipo de noticias permitiría que lectores de medios tradicionales puedan migrar hacia ella sin problemas.

3.1. Funcionamiento básico

A continuación se lista de forma simplificada el funcionamiento del protocolo a modo de brindar una visión general del mismo que facilite el abordaje sobre los distintos aspectos de su diseño que se detallarán en las próximas secciones del documento.

- Se crean tópicos por temática y región donde se publicarán noticias.
- Usuarios se suscriben a los tópicos en rol de jurado. Dejan un depósito en garantía de su honestidad, objetividad y capacidad de validación en dicho tópico.
- Cuando se llega a una determinada cantidad de jurados en un tópico, puede comenzar a publicarse noticias en él.
- Cualquiera puede publicar una noticia en un tópico en rol de periodista. Al publicar deja un depósito en garantía de la veracidad de lo publicado.
- Jurados son seleccionados al azar dentro del *pool* de jurados del tópico.
- Los jurados votan el estado de la noticia.
- Los jurados que votan como la minoría son penalizados mediante la pérdida de su depósito. Los que votan como la mayoría lo mantienen y además ganan una recompensa.
- El periodista pierde su depósito si la noticia no fue validada como cierta. Caso contrario, recupera el depósito y obtiene una recompensa.

4. Requisitos

4.1. Requisitos funcionales

- **Crear y cerrar tópico:** Se debe poder crear tópicos indicando la información relevante al mismo (región, temática, duración de la votación, valor de depósito, etc.). Luego debe poder cerrarse. Ambas acciones deben surgir de la propuesta de un usuario y decidirse a través del consenso teniendo en cuenta la reputación de cada uno.

- **Editar atributos de un tópico:** Se debe poder editar atributos relevantes al tópico a través del consenso de la plataforma.

- **Suscribirse y desuscribirse a tópico como jurado:** Un usuario debe poder suscribirse a un tópico creado en rol de jurado, eligiendo la cantidad máxima de noticias que estaría dispuesto a validar simultáneamente en dicho tópico y depositando dinero, proporcional a la cantidad elegida, en garantía de su competencia en dicho tópico y de su objetividad y honestidad a la hora de validar.

  Además, un usuario que se encuentra suscripto a un tópico debe poder desuscribirse del mismo eligiendo una nueva cantidad de noticias a validar en simultáneo menor a la actual, recuperando así, proporcionalmente, el depósito. Para desuscribirse completamente, elige cero como cantidad y recupera la totalidad del depósito. No puede elegir una cantidad menor a la cantidad de noticias en las que está actuando como jurado al realizar la desuscripción.

- **Ver suscripciones a tópicos:** Se debe poder ver a qué tópicos se está suscripto y la cantidad de noticias a validar simultáneamente elegida en cada uno.

- **Publicar noticia en un tópico:** Un usuario debe poder realizar una publicación en un tópico ejerciendo el rol de periodista. Publicar debe requerir un depósito de dinero en garantía de la veracidad de lo publicado. Al publicarse la noticia se selecciona, al azar, los jurados del tópico que se encargarán de validarla.

- **Votar estado de la noticia:** Un usuario seleccionado como jurado debe poder votar acerca de la veracidad de la noticia. El voto debe ser confidencial hasta que la fase de emisión de los mismos termine. Luego, cada usuario debe poder revelar su voto para contabilizarlo y, opcionalmente, proveer una justificación del mismo.

- **Iniciar una instancia de apelación:** Al finalizar una votación, debe haber una ventana de tiempo en la cual cualquier usuario que no esté de acuerdo con el resultado obtenido pueda iniciar una instancia de apelación, depositando un monto en garantía de que su reclamo es certero. Dicha instancia dispara una nueva votación, cuyo valor de salida se impone por

sobre el de las anteriores y es el que será tenido en cuenta a la hora de ejecutar penalizaciones y recompensas.

- **Ejecutar penalizaciones y recompensas:** Al finalizar una votación, si pasó el tiempo para creación de nuevas apelaciones, se debe poder ejecutar la obtención de recompensas a los jurados que votaron correctamente y la penalización en los depósitos a los que no. De igual modo, el periodista deberá recuperar su depósito si la noticia se consideró verdadera, o perderlo en caso contrario.
- **Notificar al usuario:** El usuario jurado deberá ser notificado cuando sea elegido para ejercer dicho rol en alguna publicación, tanto para emitir el voto, para revelarlo y cuando la votación haya concluido. Esto último también debe ser notificado al usuario autor de la noticia.
- **Listado de noticias y filtros:** Un usuario debe poder listar todas las noticias, viendo una síntesis de la misma, con su estado de validación asociado. Además, debe poder aplicar filtros sobre las mismas, por ejemplo, por tópico, por estado de validación y/o por autor.
- **Ver noticia en detalle:** Un usuario debe poder seleccionar una noticia listada para poder leerla en detalle, donde también debe poder apreciarse el estado de validación asociado a la misma.
- **Ver detalles de la validación de una noticia:** Un usuario debe poder ver el detalle de la votación efectuada para validar una noticia. El valor de cada uno de los votos emitidos y sus respectivas justificaciones.

4.2. Requisitos no funcionales

- **Global:** Proporciona una solución que no se limita a ningún país o región en particular.
- **Descentralizada:** No existe una autoridad o entidad central que la regule.
- **Transparente:** Permite ver y auditar en detalle cada uno de los procesos involucrados.
- **Resistente a censura:** No pueden alterarse ni eliminarse registros ya procesados. Esto incluye las noticias, sus votaciones de validación, suscripciones de jurados, etc.
- **Disponible:** Que el sistema esté siempre operando.
- ***Permissionless*:** Cualquiera que siga las reglas puede participar (publicando y/o validando noticias), sin excepciones.
- **Accesible:** Las barreras de entrada deben ser lo más bajas posibles.

- ***Trustless*:** Que los participantes no necesiten conocer a otros participantes o confiar en ellos, sino que sólo requieran confiar en los mecanismos de la plataforma.
- **Confidencial en los votos:** Durante la etapa de emisión de votos, estos deben mantenerse de forma confidencial, evitándose el cálculo de resultados parciales.

5. Atributos de calidad

5.1. Seguridad

El protocolo incentiva a los usuarios a comportarse de la manera deseada mediante *diseño de mecanismos*, planteando penalizaciones y recompensas, buscando la objetividad como punto focal en la validación de noticias.

Es importante asegurar que los votos sean confidenciales durante la fase de emisión de los mismos, evitando el cálculo de resultados parciales, de lo contrario el mecanismo se vería comprometido, y, por lo tanto, la confianza en los resultados de los procesos de la plataforma.

Además, es importante que ni los depósitos de los usuarios ni los archivos de las noticias puedan ser comprometidos, que los usuarios no puedan ser suplantados, que cada voto, y por lo tanto el resultado de la votación, no pueda ser manipulado ni eliminado.

5.2. Transparencia

La transparencia es una propiedad clave que una entidad que tome la responsabilidad de validar información debe cumplir si quiere que la sociedad la adopte como tal. Sin transparencia, pueden surgir sospechas acerca de los procesos y, como consecuencia de esto, perder credibilidad en los resultados de los mismos tornando a la entidad prácticamente irrelevante. Es por esto que la plataforma debe ser absolutamente transparente en cada uno de ellos.

5.3. Disponibilidad

Como se mencionó anteriormente, la plataforma funciona a través de recompensas y penalizaciones. No votar implica una penalización económica, y no se puede votar si el sistema se encuentra caído.

La validación de noticias puede comprometer a entidades relevantes y/o poderosas guiándolas a organizar ataques que afecten la disponibilidad de la plataforma para evitar así el correcto proceso de validación.

Además, la plataforma busca proporcionar una solución global. Esto implica estar operando, como mínimo, en los horarios con mayor flujo de noticias de cada una de las zonas horarias.

6. Solución propuesta

A continuación se expondrá el diseño de la plataforma, comenzando por su arquitectura y luego detallando los distintos aspectos del diseño de su protocolo.

6.1. Arquitectura

Dados los atributos de calidad a priorizar, se optó por construir la plataforma sobre una blockchain. En particular, se decidió utilizar Ethereum dado que es la blockchain pública con soporte de *smart contracts* con mayor adopción y descentralización, permitiendo así cumplir con los atributos de seguridad, transparencia y disponibilidad. Además, su comunidad y efecto de red son grandes ventajas a la hora de generar adopción y aprovechar la componibilidad de protocolos que, como se verá a lo largo del documento, permitió resolver distintas problemáticas.

El protocolo de Gazzeth estará implementado como varios contratos inteligentes diseñados de tal forma de cumplir con los requisitos no funcionales planteados anteriormente.

Luego se contará con uno o varios clientes que permitan interactuar con el protocolo de manera amigable.

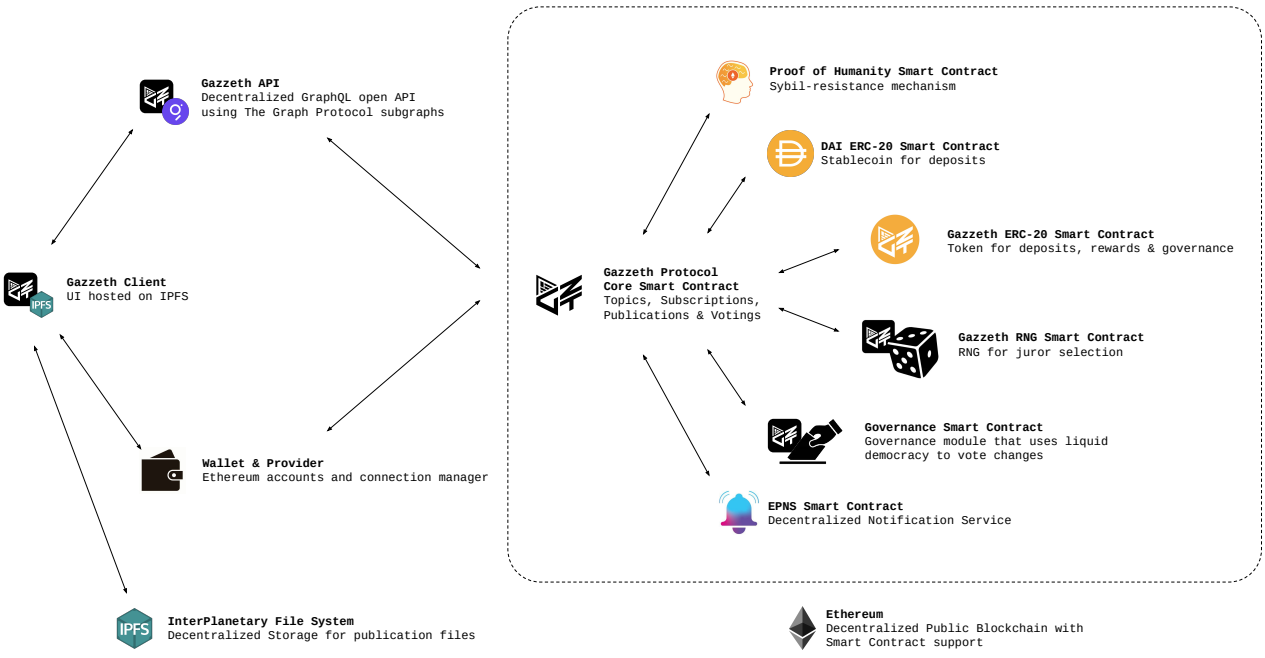


Figura 1: Arquitectura de la solución propuesta.

6.2. Usuarios

Cada usuario puede tomar tres roles distintos, no mutuamente excluyentes, dentro de la plataforma:

- **Periodista (o Autor):** Usuario que publica noticias en la plataforma.
- **Jurado (o Validador):** Usuario que valida noticias publicadas en la plataforma.
- **Lector (o Consumidor):** Usuario que consume noticias de la plataforma.

6.3. Tópicos

Los tópicos surgen para permitir categorizar las publicaciones según temática y región. De esta forma, permite a los usuarios ejercer su rol de jurado sólo en tópicos que contengan publicaciones vinculadas a temáticas de las cuales se consideran capaces de verificar y en un idioma que comprenden, así se permite proveer una solución a nivel global.

Los tópicos se identifican mediante una cadena de caracteres, para la cual se sugiere la utilización del siguiente esquema:

$$\texttt{region}_1\texttt{/region}_2\texttt{/.../region}_n\texttt{/category}_1\texttt{/category}_2\texttt{/.../category}_m$$

Por ejemplo, para información vinculada a discursos públicos de políticos argentinos podría crearse el tópico:

```
Argentina/Politica/Discursos
```

Nótese que la región de un tópico podría no ser un país, permitiendo ser un continente, provincia, estado, combinación de ambos u otras. Incluso hay casos donde la información podría no estar vinculada a una región particular, como por ejemplo los *airdrops*[2] de *tokens* de protocolos que corren en Ethereum. Para este caso se recomienda acordar un lenguaje común, como el inglés, y luego entonces generar el tópico:

```
Worldwide/Ethereum/Tokens/Airdrops
```

Un tópico sólo puede activarse a través de mecanismos de gobernanza, que se explicarán más adelante en este documento, y sólo pueden publicarse noticias en él cuando el mismo supere determinado umbral de jurados disponibles para elección.

[2]En el contexto de protocolos que corren en blockchain, un airdrop refiere al proceso de distribución de un token, generalmente utilizado para gobernanza, donde la elegibilidad para su obtención suele estar ligada a ciertas condiciones como, por ejemplo, la previa participación o uso del protocolo en cuestión.

6.4. Publicaciones

6.4.1. Formato

Las publicaciones son representadas por un archivo de texto en lenguaje Markdown, la estandarización del formato permite a los clientes del protocolo procesar el archivo y asegurar a los usuarios que cuando un archivo no es comprensible es porque no cumple las reglas del protocolo.

Se decidió la utilización del lenguaje Markdown dado que provee versatilidad al autor para estructurar el artículo, permitiendo formatear el texto (tanto su estilo como su tamaño), agregar tablas, items, imágenes y demás. Es fácil de aprender, el archivo final es liviano y es fácil de convertir a otros formatos como, por ejemplo, HTML.

En particular, se propone respetar una estructura al comienzo del archivo: título, copete y una imagen que represente la noticia, todo separado por una línea en blanco. Luego continuar con el cuerpo de la noticia.

De esta forma, los clientes que interactúen con el protocolo pueden aprovechar dicha estandarización para interpretar fácilmente la información y utilizarla, por ejemplo, para *renderizar* las noticias en una vista preliminar o *feed*.

A continuación la estructura propuesta:

```
Titulo

Copete

![](UrlImagenPrincipal)

Cuerpo de la noticia
```

6.4.2. Almacenamiento

Los archivos que representan publicaciones deben ser almacenados de forma distribuida. Recaer en un servidor central haría perder a todo el sistema las propiedades de descentralización y resistencia a censura deseadas, dado que el servidor que hostea los archivos podría eliminarlos (ya sea en su totalidad, o selectivamente) o ser sometido a ataques.

Una primera aproximación a la solución sería almacenar los datos en la blockchain, sin embargo es costoso, haciendo que una transacción de publicación tenga un costo elevado que la torne prácticamente inviable para el uso que se requiere.

Es por esto que se decidió que las publicaciones sean almacenadas en el InterPlanetary File System (IPFS)[4], un storage distribuido. La información se guarda en un *Merkle Directed Acyclic Graph*. Almacenar un archivo en IPFS de forma exitosa retorna un multi-hash del nodo del grafo mencionado, utilizándolo como identificador del recurso (content identifier, CID)[3], el cual sirve para

[3] https://docs.ipfs.io/concepts/content-addressing

recuperar el archivo posteriormente.

Se decidió que sea dicho CID el que se almacene en la blockchain como parte del protocolo, asociado a la publicación, lo cual es mucho más económico, y permite mantener las propiedades deseadas. IPFS garantiza que la información se mantiene inmutable, es decir, los archivos de las noticias no podrían ser modificados.

6.5. Diseño de mecanismos

6.5.1. Depósitos y penalizaciones

Los periodistas deben realizar un depósito de un cierto monto en DAI[5], o el 80 % de su valor equivalente en GZT (token de la plataforma que se presentará en el próximo apartado de esta sección), al publicar en garantía de que lo que se está publicando es una noticia verídica y relevante al tópico en cuestión. En caso de que la publicación no sea considerada como verdadera luego del proceso de validación, el periodista perderá la totalidad del depósito realizado. De esta forma, no sólo se desincentiva al periodista a publicar noticias con información falsa, sino que también se lo incentiva a citar fuentes y redactar la noticia de tal forma que facilite a los jurados verificar la veracidad de la misma y así minimizar el riesgo de pérdida del depósito.

De manera similar, los jurados deben realizar un depósito de un monto en DAI, o el 80 % de su equivalente en GZT, para suscribirse en dicho rol en algún tópico y así estar disponibles para ser elegidos para validar noticias. Dicho depósito está en garantía de la competencia y objetividad del usuario para validar noticias del tópico al que se suscribe. En caso de que el voto emitido al validar la noticia no coincida con el voto mayoritario, el usuario pierde el depósito. De esta forma, se desincentiva al usuario a suscribirse como jurado a tópicos donde no se considera competente y se lo incentiva a investigar en profundidad la noticia que le toca validar, analizando cuidadosamente el voto a emitir sobre la misma.

Dado que se desea que Gazzeth sea lo más accesible posible a todo tipo de usuario, utilizar monedas o tokens no estables para los depósitos, tales como ETH o GZT, podría desincentivar la adopción de la plataforma para usuarios que prefieran no estar expuestos a la volatilidad. Por ejemplo, un usuario que se suscribe como jurado realizando su depósito, a menos que haya validado noticias de forma incorrecta, debería poder decidir retirarse de la plataforma sin que esto implique pérdidas económicas (medidas en dinero fiat y exceptuando el costo de transaccionar en la blockchain). Esto podría no cumplirse con un token volátil, dado que el mismo podría bajar de precio durante el período en que se encontró utilizado como depósito.

Es por esto que se decidió brindar una alternativa estable para los depósitos a través de la stablecoin DAI[4]. La decisión de restringir los depósitos en tokens volátiles sólo a GZT, y que su utilización como depósito requiera un bloqueo

[4]Si bien una alternativa al token GZT hubiera sido necesaria dado que el mismo carece de preminado, esta alternativa podría haber sido temporal y no haberse tratado de una stablecoin.

de un valor 20 % menor que el requerido en la alternativa estable, tiene como objetivo incentivar la demanda sobre el token; en la siguiente sección se detalla por qué esto es deseable.

6.5.1.1. Servicio de fact-checking pago Nótese que un usuario en rol de autor, en lugar de actuar como periodista redactando una noticia verídica y publicándola en Gazzeth a cambio de la recompensa, podría tomar una noticia que le parece sospechosa (quizás de un tópico del cual no se considera apto para validar) y someterla a validación de jurados del correspondiente tópico, esperando que la misma sea falsa y asumiendo de antemano la probable pérdida del depósito realizado al publicar, utilizando así la plataforma como servicio de *fact-checking* pago.

6.5.2. Recompensas y el token GZT

Cuando la publicación de un usuario periodista es validada como cierta por los jurados, éste no sólo recupera su depósito, sino que además es recompensado a través de la obtención de un Gazzeth, token de la plataforma, cuyo símbolo es GZT, forma en la que se lo referirá de aquí en adelante. De igual modo, los jurados que votan mayoritariamente, no sólo mantienen su depósito intacto, sino que además son recompensados ganando un GZT cada uno.

El token GZT representa la reputación[5] de cada usuario en la plataforma y es un *equity token* dado que sirve como token de gobernanza del protocolo. Cumple con el estándar de token fungible ERC-20[6], para ser compatible con *wallets* y protocolos de Ethereum.

Se espera que la mayoría de las veces[7] que se valide una noticia publicada en la plataforma haya usuarios recompensados mediante GZT, los cuales serán emitidos especialmente para ser otorgados a dichos usuarios, aumentando así la base monetaria de GZT y, por lo tanto, diluyendo la reputación del resto de los usuarios (los no involucrados en el proceso de publicación y validación de esta noticia) de la plataforma. De aquí, se desprenden dos maneras de aumentar o mantener la reputación en el sistema:

- **Participando activamente en la plataforma:** Ya sea publicando noticias ciertas o validándolas correctamente.
- **Comprando la reputación de otros usuarios:** Dado que el token GZT es un ERC-20, puede transferirse de una cuenta a otra, operar en otros protocolos y/o *exchanges*, permitiendo que los usuarios le asignen un valor de mercado y lo comercien.

Lo interesante de que el token GZT pueda ser adquirido a través del comercio es que permite a los usuarios lectores de la plataforma obtener poder

[5] Exceptuando el hecho de que, en general, se concibe la reputación como un bien no transferible.

[6] https://eips.ethereum.org/EIPS/eip-20

[7] Existen casos donde no surge un voto mayoritario en la validación y, por lo tanto, ningún usuario es recompensado.

de decisión sobre el protocolo que consumen, sin tener que participar en roles de jurado o periodista, aportando no sólo a la descentralización del mismo sino también permitiendo a periodistas y jurados convertir el trabajo invertido en la plataforma en beneficio económico.

Por otro lado, la emisión de GZT se espera que tenga dos fases. La primera, en el corto y mediano plazo, un período transitorio en el que la plataforma gane adopción y la cantidad de noticias que se publican mensualmente en ella crezca, dándose así un crecimiento en la emisión del token. La segunda, ya en el largo plazo, un período estacionario en el cual la plataforma ya está establecida y la cantidad de noticias que se publican mensualmente en ella sea prácticamente constante, llevando a que la emisión de GZT se comporte de manera similar.

Teniendo en cuenta el comportamiento esperado en la emisión de GZT y lo interesante de que éste tome valor económico, generar demanda en el token es un aspecto altamente deseable. Es por esto que se incentiva el uso del mismo para depósitos, además de requerirse para aspectos como gobernanza y apelaciones, los cuales serán presentados en detalle más adelante.

Los GZT perdidos a causa de penalizaciones serán quemados en búsqueda de reducir la oferta del token, mientras que los DAI confiscados pasarán a formar parte del tesoro del protocolo que será administrado por su comunidad a través del sistema de gobernanza.

6.5.2.1. Sobre el minado de GZT En Bitcoin cada BTC es minado a través de cómputo, y por lo tanto energía, mediante el algoritmo *Proof of Work*[8]. Haciendo una analogía con esto, se podría argumentar que cada token GZT es *minado* a través de la atención que gasta el usuario que redacta y publica la noticia, y los usuarios que validan la misma. La atención y el tiempo de vida del ser humano son recursos limitados, la *Economía de la atención* estudia sus usos como *commodity*. A este proceso de minado de tokens a través de la atención se lo conoce como *Attention Mining*[6] y ya existen proyectos que lo implementan, como Brave Browser con su Basic Attention Token (BAT)[9].

[8] https://bitcoin.org/bitcoin.pdf

[9] https://basicattentiontoken.org/static-assets/documents/BasicAttentionTokenWhitePaper-4.pdf

6.6. Reglas para elección del voto

Previamente se describió el diseño de mecanismos del protocolo y las consecuencias que tiene el resultado de la votación tanto en jurados como periodistas. A continuación se presentan los tres tipos de voto que se encontrarán disponibles para elegir en cada votación de validación de noticia y qué condiciones se deben cumplir para la elección de cada uno de ellos:

- **Verdadera:** La publicación es una noticia, se encuentra en el tópico correcto, brindando información que no se encuentra actualmente disponible en él, y es completamente contrastable como verdadera.
- **Falsa:** La publicación es una noticia, se encuentra en el tópico correcto, brindando información que no se encuentra actualmente disponible en él, y contiene al menos un dato que no es posible contrastar como verdadero.
- **No cualificada:** Cualquier otro caso. Por ejemplo: la publicación no es una noticia, no se encuentra en el tópico correcto, no se comprende, ya se encuentra publicada una noticia que brinda la misma información, etc.

Si bien se podría haber optado por tener sólo dos tipos de voto, marcando toda noticia que no pueda considerarse verdadera como falsa, se prefirió agregar un tercer tipo de voto para indicar que la publicación no cumple los requerimientos mínimos para ser juzgada como *Verdadera* o *Falsa*.

De esta forma, por ejemplo, si alguien publica una copia exacta de una noticia que fue marcada como *Verdadera*, la copia sería marcada como *No cualificada* por encontrarse repetida. Si no se contase con este tercer tipo de voto, se generaría ambigüedad en el proceso de validación, dado que o se estaría marcando como *Falsa* una noticia esencialmente verdadera, o se estaría marcando como *Verdadera* permitiendo así un vector de ataque para obtener GZT a partir de noticias repetidas.

6.7. Commitment-Scheme

Ethereum es una blockchain pública, cualquier interacción con un *smart contract* revela información del invocador, la función invocada y los parámetros utilizados, entre otros datos. Por lo tanto, si se quiere implementar un sistema de votación en el cual no sea posible calcular resultados parciales hasta que la votación finalice, se debe realizar la misma mediante un *Commitment-Scheme*.

Estos esquemas constan de dos fases. En la primera, *Commit*, cada votante debe proveer información (*commitment*) del voto de tal forma que el mismo se mantenga confidencial. Luego, en la segunda fase, *Reveal*, cada votante revela el voto, contabilizándose sólo al comprobar que lo revelado se condice con el *commitment*.

6.7.1. Esquema clásico

6.7.1.1. Commit La solución frecuentemente aplicada utiliza como *commitment* el hash del voto concatenado con un IV[10]. Tomando en cuenta las particularidades del protocolo, se agrega el ID de la publicación, `publicationId`, y el commitment quedaría:

```
commitment := hash(publicationId||vote||IV)
```

La función de commit sería (el votante, quien invoca al método, se encuentra como una variable global, `sender`, en cada llamada):

```
Input: publicationId, commitment
commitments[publicationId][sender] := commitment
```

6.7.1.2. Reveal Luego, la función de reveal recibiría el `publicationId`, el voto y el IV utilizado, reconstruiría el commitment contabilizando el voto sólo si el commitment reconstruido coincide con el brindado previamente:

```
Input: publicationId, vote, IV
rebuiltCommitment := hash(publicationId||vote||IV)
if rebuiltCommitment == commitments[publicationId][sender]
    countVote(publicationId, sender, vote)
else
    error("Revealed data does not match commitment")
```

Donde `hash` es una función de hash, `countVote` contabiliza al votante `sender` un voto de valor `vote` sobre la publicación con ID `publicationId` y `error` revierte la transacción con el mensaje de error brindado.

6.7.1.3. Inconvenientes El esquema clásico presenta dos inconvenientes en el contexto de este proyecto.

- **Falta de estandarización:** El protocolo, al correr sobre Ethereum, será público, es decir, cualquiera podrá crear su propio cliente para interactuar con él.

  Por lo tanto, quedaría a cargo de cada cliente la generación de los IV de forma segura, pudiendo surgir alguno que permita al usuario elegir el IV, perdiendo así las propiedades de aleatoriedad requeridas, por ejemplo permitiéndole elegir siempre el mismo valor o utilizando un contador haciéndolo predecible.

  De esta forma, no podría forzarse la completa estandarización del esquema de votación, tornándose inseguro, y rompiendo el diseño de mecanismos presentado, dado que en algunos casos se podría conocer el valor de cierto voto en la fase de commit.

[10] Si no se usara el IV, dado que el `publicationId` es conocido, un atacante podría calcular `hash(publicationId||vote)` para cada valor posible de voto y así descifrar cuál fue usado.

- **Mala usabilidad:** Teniendo en cuenta que la plataforma busca minimizar fricciones para obtener la adopción de la mayor cantidad de usuarios posibles y que no lograr revelar un voto implica una penalización económica, el hecho de que el usuario deba recordar o almacenar de forma segura un IV para lograr el reveal exitosamente atenta contra la usabilidad (al ser una plataforma descentralizada, no se cuenta con un servidor que almacene el IV y, dado que se quiere preservar la identidad de los usuarios, recolectar información personal para enviárselo, como su correo electrónico, no es opción).

  Este problema de mala usabilidad va en contra del principio de diseño *Psychological Acceptability*, incentivando a ciertos usuarios a utilizar un cliente que les permita seleccionar el IV a gusto de tal manera de facilitar el proceso de reveal, vulnerando así la seguridad del esquema de votación como se mencionó en el item anterior.

6.7.2. Esquema propuesto

Ambos inconvenientes mencionados sobre el esquema clásico surgen de la existencia del IV como parte del commitment y tienen como posible consecuencia la pérdida de confidencialidad sobre algunos votos en la fase de commit. Es por esto que se propone un nuevo esquema que permite realizar el commit y el reveal prescindiendo del IV.

6.7.2.1. Explicación básica El esquema plantea que el votante genere una firma digital de su voto y envíe como commitment el hash de la misma. Es importante que para firmar se utilice el algoritmo ECDSA en su versión determinista de 256-bits, tal como lo define el RFC-6979[7], soportado por Ethereum.

Luego, en el reveal, el votante vuelve a generar la firma (por eso la importancia del determinismo en el algoritmo de firmado, de lo contrario no coincidiría con la brindada en el commit) y la revela junto al valor de su voto.

De esta forma, el smart contract del protocolo efectúa el hash sobre la firma provista y verifica que este se corresponde con el commitment enviado previamente. Luego, dado que ECDSA permite recuperar la clave pública del firmante a partir de la firma y el dato firmado correspondiente, el contrato utiliza el valor del voto junto con la firma para recuperar la clave pública del firmante y así verificar que la firma provista es válida y tiene como dato subyacente el voto indicado. Una vez efectuado esto de forma exitosa se contabiliza el voto del votante.

En el commitment, el hash es necesario dado que, gracias a su propiedad de resistencia a la preimagen, brinda confidencialidad sobre la firma. Si se enviara la firma sin hashear, un atacante podría tomarla y probar recuperar la clave pública del firmante utilizando cada valor posible de voto. Luego, el voto emitido será el que al ejecutar la recuperación del firmante dé como salida la clave pública del votante. En este apartado, cuando se hable de función de hash se estará haciendo referencia a KECCAK-256, soportada por Ethereum.

6.7.2.2. Commit Para definir el esquema se utilizó el *estándar de firmado y hasheado de estructuras tipadas* propuesto en el EIP-712[8] ya que considera algunos aspectos de seguridad adicionales, como el *domain separator*, y, al ser un estándar, es tenido en cuenta por las distintas *wallets* haciendo su uso amigable al humano, por ejemplo, mostrando de forma amigable la estructura y contenido de la información a firmar.

La estructura tipada de voto se definió como:

```
struct Vote {
    uint256 publicationId;
    uint8 vote;
    uint256 nonce;
}
```

Por lo tanto, el *typehash*, que es una constante que identifica el tipo de la estructura a firmar, queda definido como:

```
VOTE_TYPEHASH := hash("Vote(uint256 publicationId,uint8 vote,uint256 nonce)")
```

Luego, el *hashStruct*, que es el hash del *typehash* junto a los valores de cada campo de la estructura[11], queda definido como:

```
hashStruct := hash(VOTE_TYPEHASH||publicationId||vote||nonce)
```

Y la información que quedará para firmar es:

```
data := hash(0x19||0x01||DOMAIN_SEPARATOR||hashStruct)
```

Donde el primer byte está para evitar que la firma sea una posible transacción válida (dado que el 0x19 no es un valor válido de inicio en el encoding RLP, utilizado por las transacciones de Ethereum), el segundo byte indica la versión, tal como lo define el EIP-191, y `DOMAIN_SEPARATOR` es el *hashStruct* de la siguiente estructura utilizando los datos correspondientes al contrato del protocolo de Gazzeth:

```
struct EIP712Domain {
    string name;
    string version;
    uint256 chainId;
    address verifyingContract;
}
```

Esto aportará información como la *address* del contrato y la blockchain para la cual se está generando la firma, haciendo así que el contrato sólo esté aceptando firmas que fueron generadas exclusivamente para ser utilizadas en él.

[11] El EIP-712 define una función de encoding para cada valor según su tipo. En el caso de los enteros deben ser extendidos a 256-bits respetando su signo y dispuestos en big endian. Esto fue omitido para simplificar la notación de la definición.

Por último, el commitment será el hash de la firma ECDSA determinista de la información definida anteriormente:

```
signature := ecdsaSign(data)
commitment := hash(signature)
```

Y la función de commit, teniendo en cuenta que los *nonces* se encuentran en 0 por defecto, sería:

```
Input: publicationId, commitment, nonce
if nonce >= nonces[publicationId][sender]
    commitments[publicationId][sender] := commitment
    nonces[publicationId][sender] := nonce + 1
else
    error("Nonce must be greater than the last one")
```

Se decidió utilizar un nonce porque, dado que se permitirá al usuario cambiar su voto, intercambiando el commitment por otro (siempre y cuando se encuentre en el período de tiempo correspondiente a la fase de commit), se quiere evitar revelar patrones de comportamiento.

Sin el nonce, si un usuario emite un voto v_0, luego se arrepiente y emite un voto distinto v_1 y luego vuelve a arrepentirse emitiendo un voto v_2 igual a v_0, como la información firmada en los votos v_0 y v_2 es la misma (igual publicación y valor de voto), sus correspondientes commitments serán idénticos.

El nonce hace que la información a firmar en cada voto sea distinta indiferentemente del valor del voto y, por lo tanto, para el atacante, todos los commitments se vean distintos, evitando así encontrar patrones en los mismos.

6.7.2.3. Reveal El usuario vuelve a generar la firma mediante el mismo `publicationId`, `nonce` y `vote` que utilizó para el commit. La función de reveal queda:

```
Input: publicationId, vote, signature
if nonces[publicationId][sender] == 0
    error("Missing vote commitment")
if commitments[publicationId][sender] != hash(signature)
    error("Revealed values do not match commitment")
nonce := nonces[publicationId][sender] - 1
hashStruct := hash(VOTE_TYPEHASH||publicationId||vote||nonce)
data := hash(0x19||0x01||DOMAIN_SEPARATOR||hashStruct)
if ecdsaRecoverSigner(signature, data) != sender
    error("Invalid signature")
countVote(publicationId, sender, vote)
```

La función `ecdsaRecoverSigner`, que recibe la firma y la información firmada y retorna el firmante, debe fallar si la firma no cumple lo que indica el Apéndice F, "Signing Transactions", del Yellow Paper de Ethereum[9].

6.7.2.4. Sobre los inconvenientes planteados Nótese que con el esquema propuesto ya no se necesita recordar una secuencia aleatoria (IV) para lograr el reveal de forma exitosa. El `publicationId` es conocido y el `nonce` es obtenido del smart contract a partir del votante y el `publicationId`. Estos datos serán provistos por el cliente y, por lo tanto, el único dato que debe recordar el votante es el valor del voto que emitió para dicha publicación en la fase de commit, solucionándose así el problema de usabilidad planteado como segundo inconveniente.

Por otro lado, ahora el esquema de votación se encuentra estandarizado. Cualquier cliente que no respete el esquema EIP-712 planteado estará generando votos inválidos, quedando así solucionado el primer inconveniente planteado.

6.7.3. Desventajas

Si bien las votaciones en blockchain pueden proveer interesantes atributos como correctitud, transparencia y censorship-resistance, hay otros aspectos relevantes a las votaciones que, al momento de la creación de este documento, no se brindan[10], ni siquiera en el último esquema de votación planteado:

- **Privacidad:** Que no puedas *ver* quién votó y quién no, ni qué votó cada uno. De esta forma se busca evitar que la gente elija su voto en base a potenciales repercusiones ligadas a su identidad como, por ejemplo, discriminación o repudio social por el voto escogido.
- **Resistencia a coerciones:** Que no puedas *probarle* a alguien cómo votaste, incluso aunque quisieras hacerlo. Así se evitan coerciones como, por ejemplo, sobornos o amenazas.

Proveer este tipo de características es complejo en blockchain, pero más aún en protocolos como Gazzeth donde hay una recompensa o penalización en base al voto elegido. Gazzeth sólo provee privacidad durante la fase *commit* de la votación para evitar la utilización de resultados parciales y provee apelaciones (se verá más adelante) que permiten mitigar coerciones como sobornos.

6.8. Sybil-resistance

Un Ataque Sybil[11] se da cuando un sistema es corrompido por una entidad (o entidades) que logra crear y controlar múltiples identidades en él, aparentando ser independientes, permitiendo a través de estas una manipulación no deseada del mismo. Los sistemas que no son vulnerables a este tipo de ataque se consideran *sybil-resistant*.

En general, contar con una entidad central que tenga la autoridad de certificar identidades soluciona este inconveniente. Sin embargo, esto no es deseable en redes o protocolos descentralizados, tal como es el caso de Gazzeth.

6.8.1. Ataque Sybil en Gazzeth

Se cuenta con un tópico que tiene n jurados inscriptos, siendo n mayor a la cantidad mínima de jurados que debe haber inscriptos para poder publicar noticias en dicho tópico.

Si en el tópico se publican noticias cuya verificación afecta de forma negativa la imagen pública de cierta entidad (podría ser por ejemplo un gobierno o una empresa), dicha entidad podría realizar un Ataque Sybil, creando $m > n$ cuentas y suscribiéndolas como jurado al tópico.

Ahora esta entidad tendría control de más de la mitad de los jurados del tópico y, como los jurados se eligen de forma aleatoria, tendría una probabilidad mayor al 50 % de tener mayoría en la votación y elegir el estado de las noticias a su gusto.

6.8.2. Solución

Es importante tener en cuenta que no es deseable aumentar demasiado el valor de depósito para ser jurado ya que se tornaría inaccesible para una gran parte de las personas. Dado que el impacto de las noticias en la sociedad es relevante, una entidad poderosa (o conjunto de ellas) podría tener suficientes incentivos como para costear el ataque (de hecho, que la noticia fuese verificada objetivamente podría implicarle una pérdida económica mayor que la ejecución del mismo). Es por esto que el costo del ataque se consideró irrelevante a la hora de evitar el mismo.

Como solución se plantea integrar un protocolo Proof of Personhood[12] que garantice que cada jurado esté asociado a *un y sólo un* ser humano existente.

Para esto se propone utilizar Proof of Humanity[13][14], protocolo descentralizado que mantiene un registro de humanos *on-chain*, donde cada uno está vinculado a una única dirección Ethereum.

De esta forma, un usuario sólo debe poder suscribirse como jurado si su dirección de Ethereum se encuentra registrada en Proof of Humanity. En caso contrario, la transacción es revertida con error. Como consecuencia de esto, cada jurado está asociado a un humano diferente y, por lo tanto, cada voto también. Así se evita el ataque descripto anteriormente y se convierte a Gazzeth en un protocolo *sybil-resistant*.

6.8.3. Desventaja

Las soluciones que proveen identidad descentralizada se enfrentan a un trilema[15]: auto-soberanía[16], *sybil-resistance* y privacidad. Actualmente no existe una solución que logre cubrir los tres pilares.

Proof of Humanity falla en preservar la privacidad. Si bien permite usar pseudónimos evitando proveer información personal (nombre, edad, nacionalidad, etc.), requiere un video en el cual quien se registra debe exponer su rostro mientras muestra su dirección de Ethereum y recita una frase.

Las herramientas de reconocimiento facial ya son lo suficientemente poderosas como para que entidades que tienen asociados rostros con información

sensible, como empresas detrás de redes sociales o gobiernos, logren identificar personas a través de imágenes o videos.

Ya se ha visto manifestantes utilizando mecanismos para evitar el reconocimiento facial gubernamental por temor a las consecuencias de sus protestas[12][13]. Una herramienta que busca ser una fuente de verdad sobre la información, donde muchas veces esta puede afectar de forma negativa a alguna entidad poderosa, debe proveer privacidad sobre la identidad de los participantes para evitar que puedan ser coaccionados o censurados.

6.8.3.1. Optimismo de cara al futuro Por lo expuesto anteriormente es claro que, a la fecha que se escribe este documento, integrar Proof of Humanity a un protocolo en el cual la privacidad es crítica representa una desventaja. Sin embargo, el protocolo lleva menos de un año desde su lanzamiento. Ya han surgido propuestas para utilizar *zero-knowledge proofs*, permitiendo crear un esquema donde humanos registrados puedan utilizar una dirección nueva, demostrando que la misma pertenece a alguna identidad del registro sin revelar a cuál[14]. Esto expondría qué humanos se encuentran en el registro pero no podría saberse quién está actuando como jurado en Gazzeth.

6.9. Apelaciones

Luego de que finalice una votación habrá un período de tiempo previo a la ejecución de las penalizaciones y recompensas en el cual se podrá apelar en caso de que algún usuario no se encuentre conforme con el resultado obtenido, disparando así una nueva votación, pero esta vez con el doble de jurados que la votación original.

Para esto, se deberá bloquear el equivalente en GZT a un cierto monto alto fijado en moneda fiat (por ejemplo, en dólares estadounidenses o euros), que podrá ser provisto por diferentes usuarios (crowdfunding), sin ningún requisito especial sobre ellos, para permitir que usuarios lectores también puedan participar de este proceso con los GZT que adquirieron.

Se podrá apelar tantas veces como se desee (es decir, apelar una apelación) siempre y cuando se encuentre disponible el número de jurados en el tópico correspondiente a la noticia y la misma se encuentre en el período de tiempo donde se permitan apelaciones (esta ventana de tiempo comienza cada vez que finaliza una votación, ya sea la original o la de una apelación). Es por esto que mientras más jurados haya suscriptos en un tópico (y mayor cantidad de suscripciones de cada uno en él) más confiable será el mismo, ya que garantizará la posibilidad de realizar una mayor cantidad de apelaciones.

[12] https://www.cbc.ca/news/world/hong-kong-protest-lasers-facial-recognition-technology-1.5240651

[13] https://www.theatlantic.com/technology/archive/2019/08/why-hong-kong-protesters-are-cutting-down-lampposts/597145/

[14] https://geek.sg/blog/privacy-for-public-registries%E2%80%8A-%E2%80%8Aproof-of-humanity-x%C2%A0veilos

Tanto el monto a bloquear en GZT como la cantidad de jurados requeridos en la votación se duplicarán en cada apelación realizada sobre una misma noticia. Este comportamiento exponencial busca mitigar el escenario en el que un atacante inicia la apelación y soborna a los jurados para que voten a su gusto.

Una vez finalizada una votación, si el tiempo para apelar finaliza, se ejecutarán todas las penalizaciones y recompensas, teniendo en cuenta como resultado correcto el de la última votación. Las apelaciones que hayan sido ejecutadas para modificar un resultado distinto al obtenido en la última votación, serán consideradas apelaciones correctas, en caso contrario serán consideradas incorrectas.

Quienes hayan aportado GZT para realizar una apelación correcta lo recuperarán con un incremento porcentual sobre el mismo (para esto se emitirá GZT). Por otro lado, se perderá todo el GZT utilizado para ejecutar apelaciones incorrectas.

6.9.1. Justificación del voto

Cuando se revela el voto, opcionalmente, debe poder brindarse una justificación del mismo. Dicha justificación quedará almacenada en la blockchain asociada al voto revelado. Dado que el almacenamiento en blockchain es costoso, si la justificación es extensa se propone almacenar la misma en IPFS y luego brindar el CID como justificación a través del formato *ipfs://<cid>*, de esta forma el cliente podría *parsearlo*, obtener el correspondiente archivo y renderizar su contenido como justificación.

La justificación es un aspecto clave para brindar transparencia y permitir a usuarios lectores comprender las razones detrás de la clasificación de la publicación. Además, el usuario jurado está incentivado a proveerla, dado que esta podría servir para evitar una apelación en su contra o para generar y/o ganar una apelación en favor suyo.

6.10. Gobernanza

Gazzeth se presenta como una plataforma descentralizada donde sus cambios y mantenimiento se deciden a través del consenso de los participantes de su comunidad mediante un mecanismo conocido como gobernanza (*governance*) conformando estos así una *Organización Autónoma Descentralizada* (DAO, por sus siglas en inglés).

La gobernanza del protocolo se plantea como una *democracia líquida*[15] de voto ponderado (*token weighted*) a través del token GZT (quienes tengan más GZT tendrán más poder de decisión). Si bien esto permitiría que unas pocas entidades con mucho poder económico adquieran la mayoría del GZT convirtiendo la gobernanza en una plutocracia, las características dilutivas del token GZT y su falta de minado previo (*premine*), permiten que una comunidad de usuarios activos e interesados en la plataforma eviten la concentración de poder en la misma.

[15] Cada votante puede elegir emitir su voto, como una democracia directa, o delegarlo a otro actor (o actores) para que voten por él, como una democracia representativa.

Para abrir cualquier tipo de propuesta se debe poseer una cierta cantidad de tokens GZT. De este modo, se evita el *spam* y se busca que las propuestas provengan de participantes de la comunidad.

Por otro lado, para que una propuesta quede aprobada, no sólo debe obtener el voto mayoritario acerca de su aprobación sino que también la cantidad de peso en votos debe superar cierto umbral. De esta forma se evita que se aprueben propuestas donde hubo poca participación. A su vez, las propuestas aprobadas tendrán un retraso a la hora de ejecutarse, permitiendo a los usuarios que estén en desacuerdo, por ejemplo, vender sus tokens, retirarse de la plataforma o realizar un *fork* del protocolo.

El peso de los votos de cada usuario en cada propuesta de gobernanza será calculado como $min(GZT_{start}, GZT_{end})$, donde GZT_{start} son los GZT que tenía el usuario al momento de la creación de la propuesta y GZT_{end} los GZT que tenía el usuario cuando finaliza el período de votación de la propuesta.

Hay ciertas acciones ya predeterminadas que deben realizarse vía gobernanza y cuyos cambios estarán contemplados en la implementación del protocolo:

- **Creación de tópicos:** Para crear un nuevo tópico debe realizarse una propuesta justificando la existencia del mismo y sus parámetros.
- **Desactivación de tópicos:** Si la existencia de un tópico deja de tener sentido o está siendo utilizado de manera inadecuada, puede proponerse su desactivación.
- **Edición de tópicos:** Las reglas sobre el contenido de un tópico pueden ser modificadas a través de una propuesta.
- **Modificación de montos de depósito:** Dado que los montos de depósito están expresados en DAI, y por tanto ligados al dólar estadounidense, se requerirá actualizar los montos periódicamente (por ejemplo, anualmente) para mantener el valor con respecto a su inflación. Además, puede plantearse su modificación por otros motivos y en tópicos particulares.
- **Modificación de monto de recompensa:** Puede plantearse una modificación en la cantidad de GZT a recibir como recompensa. Además, puede proponerse recompensar a los periodistas con un monto diferente al de los jurados.
- **Modificación de tiempo de fase de emisión y/o revelación de voto:** Puede sugerirse una modificación tanto en el tiempo que debe durar la fase de emisión de votos como en la de revelación de los mismos. Este cambio podría proponerse sólo para algún tópico en particular.
- **Modificación de monto para primera apelación:** Se puede proponer una modificación en la cantidad de GZT que se requiere bloquear para ejecutar una primera apelación sobre una noticia.

- **Cantidad de jurados requeridos para la validación de una noticia:** Se puede modificar la cantidad de jurados que deben juzgar una noticia. Este cambio podría proponerse sólo para algunos tópicos.
- **Cantidad de jurados disponibles para permitir publicar:** Se puede modificar la cantidad de jurados que debe haber disponibles en un tópico para permitir la publicación de una nueva noticia. Este cambio podría proponerse sólo para algunos tópicos.
- **Parámetros de gobernanza:** Tanto la duración de la votación, el umbral de peso de votos a superar y la cantidad de GZT a tener para realizar una propuesta son parámetros modificables a través de la propia gobernanza.

Sin embargo, el proceso de gobernanza no se limita únicamente a los cambios mencionados anteriormente. Cualquier tipo de modificación sobre el protocolo puede proponerse y derivar en nuevas versiones del mismo. Cambios distintos a los anteriormente listados probablemente requieran modificar parte de la implementación del protocolo[16].

Por ejemplo, podría decidirse que una porción de los DAI que conforman parte del tesoro de la DAO sean intercambiados por otro token. O, aprovechando la componibilidad de Ethereum, podría plantearse que los DAI del tesoro junto con los depositados en la plataforma fueran, a su vez, depositados en algún protocolo de *yield farming* permitiendo así generar retornos que permitieran crear algún programa de *staking* de GZT a cambio de una porción de los DAI generados o utilizarse para ser intercambiados por GZT incrementando así su demanda.

6.11. Fases para lanzamiento justo

Uno de los aspectos deseados a la hora de lanzar el protocolo Gazzeth es que sea justo desde entonces. Es por esto que el token GZT carece de minado previo: para minar GZT se requiere publicar y validar noticias.

Sin embargo, para publicar noticias se requiere tener al menos un tópico activo y, como se mencionó anteriormente, los tópicos se crean a través de la gobernanza utilizando el token GZT. Es evidente que con los mecanismos presentados hasta el momento existen problemas para poner el protocolo en marcha.

Permitir un período en el cual los tópicos puedan ser creados sin ninguna restricción ni mecanismo especial, pondría la plataforma en una situación de vulnerabilidad: por ejemplo, podría crearse un tópico cuya descripción invite a votar toda publicación como si fuera cierta y, de esta forma, si un grupo suficiente de usuarios cooperan para realizar el ataque, podrían minar grandes cantidades de GZT fácil y rápidamente, obteniendo poder mayoritario en la DAO, evitando que el tópico malicioso pueda ser desactivado, entre otras acciones indeseadas. Esto simplemente sabotearía el lanzamiento de la plataforma,

[16] Visitar "The State of Smart Contract Upgrades" del Blog de OpenZeppelin para ver patrones de actualización de smart contracts. `https://blog.openzeppelin.com/the-state-of-smart-contract-upgrades`

haciendo que esta y todos los tokens GZT que se hayan generado carezcan de valor.

Por otro lado, definir tópicos iniciales de manera arbitraria, por ejemplo por parte de los creadores del protocolo, no cumpliría los aspectos de justicia y neutralidad deseados para el lanzamiento del mismo.

Es por esto que se plantea un mecanismo en el cual se divide al protocolo en tres fases en búsqueda de mitigar las vulnerabilidades del lanzamiento sin sacrificar las características deseadas en el mismo.

6.11.1. Fase I: Lanzamiento inicial

En la primera fase del protocolo cualquier usuario puede crear un tópico.

Los jurados requerirán Proof of Humanity y no podrán suscribirse para validar más de una noticia en simultáneo en un mismo tópico. Además, si bien podrán darse de baja de la validación de noticias, esto no liberará su depósito de DAI bloqueado.

Los periodistas no desbloquearán su depósito de DAI cuando su noticia sea validada como cierta, sólo podrán reutilizarlo como depósito para una nueva publicación.

Tanto periodistas como jurados, en lugar de GZT, obtendrán como recompensa *puntos de tópico*, cada uno representado como un token GTP (por sus siglas *Gazzeth Topic Points*). Dicho token no será transferible, por lo que no seguirá el estándar ERC-20. A su vez, cada GTP tendrá como estado interno el ID del tópico en el que se encuentra la publicación que permitió la emisión del mismo. Este token no podrá utilizarse en la Fase I, tanto los depósitos como la activación de apelaciones se harán a través de DAI.

Luego de un período de tiempo posterior a que un segundo tópico emita su primer GTP, terminará la Fase I y se dará comienzo a la Fase II del protocolo.

6.11.2. Fase II: Curado de tópicos

Durante la Fase II no se podrán crear más tópicos, no se podrán suscribir jurados, ni realizar publicaciones ni apelaciones.

En esta fase se activará una votación por tópico, en cada una de ellas se deberá decidir si su tópico subyacente debe permanecer o no en la tercera fase del protocolo. Todas las votaciones se darán en simultáneo durante el período de tiempo que dure la segunda fase.

Cada votación será democracia directa de voto ponderado sobre el token GTP, sin mínimo de votos, contabilizando sólo los GTP que no hayan sido emitidos en el tópico subyacente. Además, en cada votación sólo podrán emitir voto los usuarios que no hayan participado en él (ni como periodista, ni como jurado).

Los tópicos de la Fase I que no hayan alcanzado el umbral de jurados mínimos para permitir publicaciones en el mismo serán descartados, sus DAI bloqueados serán confiscados pasando a formar parte del tesoro del protocolo.

Lo mismo ocurrirá con los tópicos cuya votación termine indicando el descarte del mismo, o en empate (incluso cuando se trate de cero votos), al finalizar la segunda fase.

Por otro lado, los tópicos cuya votación indique permanencia continuarán existiendo en la Fase III y cada GTP emitido en ellos podrá ser convertido a GZT para ser utilizado en dicha fase.

6.11.3. Fase III: Lanzamiento final

Al finalizar la Fase II habrán quedado tópicos establecidos y GZT en circulación, sentando así las condiciones iniciales del protocolo de manera descentralizada. El protocolo estará listo para funcionar con las reglas planteadas a lo largo del documento.

6.11.4. Conclusiones sobre el mecanismo planteado

Se considera que el mecanismo de fases presentado permitiría lanzar el protocolo de forma relativamente justa: descentralizada y permissionless.

Sin embargo, se debe tener en cuenta que factores como la asimetría de información evitan lanzamientos plenamente justos y que, si bien se consideran mitigados, existen riesgos de que el lanzamiento igualmente se vea saboteado.

La diferencia se encuentra en que el mecanismo de fases presentado evita que un tópico sea aprobado por quienes participaron en él, dificulta la realización del ataque previamente descripto donde se toma control de la DAO mediante un tópico malicioso y, en caso de ser ejecutado, reduce las chances de que el resultado del mismo sea exitoso.

Además, los ataques destinados a sabotear el lanzamiento se ven desalentados dado que posiblemente impliquen grandes pérdidas económicas para la entidad que los financie y, posteriormente, se contará con suficiente información como para realizar un *fork* que sea considerado legítimo y comience desde la Fase III.

6.12. Notificaciones

Los jurados deben suscribirse en algún tópico y esperar a ser elegidos de forma aleatoria para juzgar una publicación. Dado que no emitir o revelar voto por parte de un jurado seleccionado implica una penalización económica, es importante que los mismos sean notificados, como mínimo, sobre los siguientes eventos:

- Cuando son elegidos para juzgar una publicación.
- Cuando la votación de la publicación donde fueron seleccionados pasó de fase de emisión a fase de revelación.

Sin embargo, como se mencionó anteriormente, es importante que la plataforma preserve la identidad de los participantes por lo que se debe evitar el pedido

de información personal como dirección de correo o teléfono celular, y se desea mantener todos los componentes de la plataforma de manera descentralizada.

Es por esto que para proveer notificaciones se propone integrar Ethereum Push Notification Service (EPNS)[17], un protocolo de notificaciones descentralizado. El mismo permite crear canales de notificaciones y emitirlas on-chain a través de eventos, que luego pueden ser indexados y enviados al usuario interesado de la forma que cada cliente implemente[17].

Los destinatarios de las notificaciones serán una o múltiples direcciones de Ethereum, dependiendo de cómo se genere el canal, siendo ésta la única información que se requiere de los destinatarios.

Además, EPNS permite elegir una serie de parámetros sobre las notificaciones, uno de ellos es el momento en que la notificación debe ser mostrada al usuario. Esto es relevante a la hora de emitir las dos notificaciones mencionadas anteriormente, ya que se crearán al mismo tiempo, cuando los jurados sean seleccionados. La primera será de visualización inmediata, para notificar su participación como jurado, y la segunda tendrá configurado un *offset* para el tiempo de visualización de la misma, que es el tiempo de duración de la fase de emisión de votos.

6.13. Cliente

Para que Gazzeth sea adoptado por la sociedad debe ser inclusivo y accesible, teniendo barreras de entrada bajas. Para eso, es importante que exista por lo menos un cliente que permita consumir el protocolo e interactuar con él, a través de una interfaz de usuario, de manera amigable.

6.13.1. Tipos de clientes

El lanzamiento del protocolo debe ser acompañado por el de un cliente que permita visualizar todas las noticias asociadas a su estado de validación y realizar distintos tipos de filtros sobre ellas. Además, debe permitir publicar noticias, suscribirse o desuscribirse como jurado en los distintos tópicos, votar en las publicaciones donde se ha sido elegido como jurado y participar de la gobernanza.

Para esto, se propone que el cliente sea una aplicación web, dado que es accesible desde distintos tipos de dispositivos, hosteada en IPFS para mantener la descentralización y evitar la censura. Sin embargo, cualquiera podría descargar el código de este cliente y ejecutarlo de forma local o hostearlo en algún proveedor centralizado de su preferencia.

Es importante destacar que como el protocolo es público, y está totalmente desligado del cliente, cualquiera podría crear su propia versión, ya sea modificando la provista por los desarrolladores del protocolo o creando una desde cero. Esto permitiría modificar la forma en que se visualizan las noticias, agregar nuevos tipos de filtros, modificar las reglas de elección del estado a asociar

[17] EPNS provee sus clientes oficiales, siendo uno de ellos una aplicación móvil que tiene tanto soporte en Android como en iOS, y permite recibir las notificaciones generadas on-chain en forma de *push notification* en el teléfono móvil.

a cada noticia (por ejemplo visualizando como verdaderas sólo las publicaciones que tengan un porcentaje de votos *Verdadera* superior al 90 %), hacer versiones específicas para ciertos dispositivos o plataformas, entre otras cosas.

6.13.2. API abierta

Para poder crear un cliente que cumpla con los requisitos mencionados anteriormente se requiere poder realizar consultas sobre la información almacenada en los smart contracts del protocolo de Gazzeth.

Si bien los contratos del protocolo podrían proveer formas de realizar accesos aleatorios a publicaciones a través de su ID, no lograrían proveer una forma de realizar consultas sobre la totalidad de los datos utilizando filtros, offset, paginado, ordenamiento y otros aspectos deseables a la hora de consultar grandes volúmenes de información.

Esto podría ser resuelto de forma relativamente simple montando un servidor centralizado que se encargue de indexar los datos de las publicaciones y exponga una API que permita realizar las consultas deseadas sobre dicha información. Sin embargo, se desea que todos los componentes de la arquitectura se mantengan descentralizados, de lo contrario este servidor sería un único punto de falla para dejar a los clientes fuera de servicio y por lo tanto que muchos usuarios pierdan la posibilidad de interactuar con el protocolo.

Para no depender de un indexador centralizado se decidió utilizar el protocolo The Graph[18][19], que permite descentralizar la capa de indexado y consulta de información, a través de actores con distintos roles e incentivos, generando una red de indexadores que exponen una API abierta la cual puede ser consultada a través de GraphQL.

7. Prueba de concepto

A continuación se describirá la prueba de concepto desarrollada. Para que la misma fuera viable en los plazos deseados para este trabajo, se desarrolló directamente la Fase III del protocolo y se evitó la implementación de los módulos de gobernanza y notificaciones. Además, no cuenta con apelaciones y los depósitos sólo son permitidos en DAI.

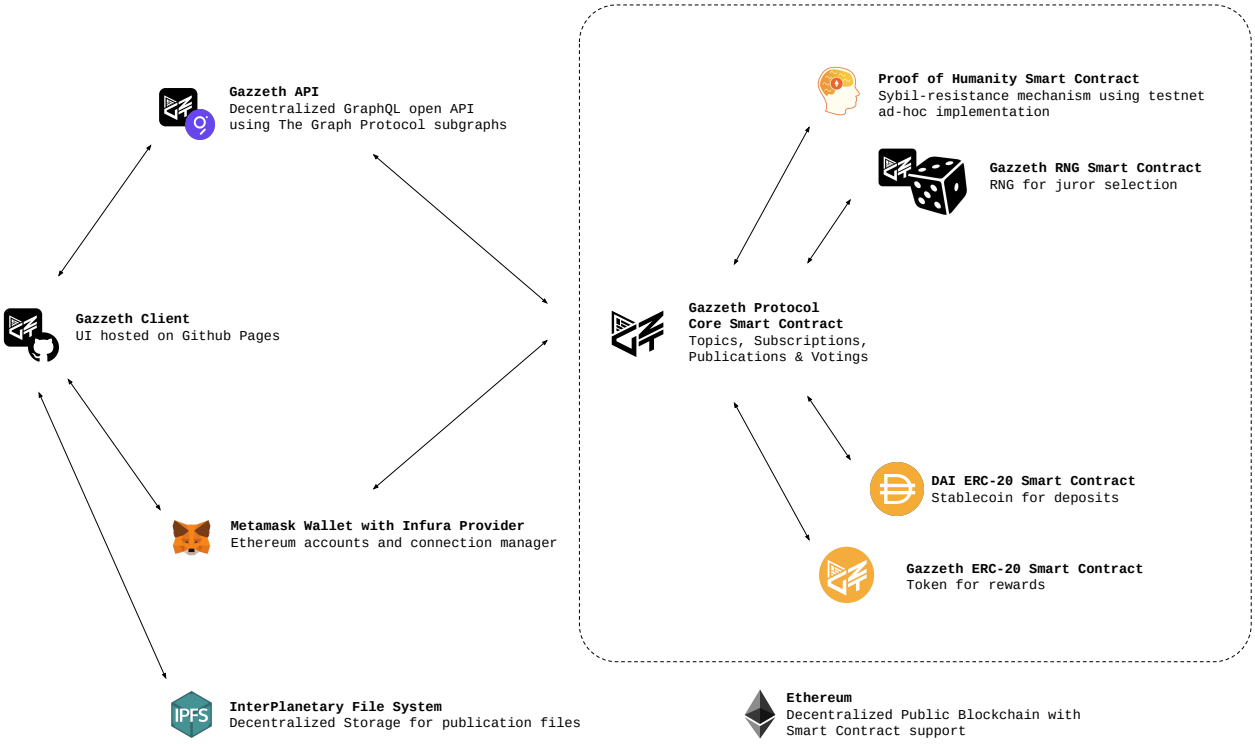


Figura 2: Arquitectura de la prueba de concepto

7.1. Protocolo

7.1.1. Herramientas utilizadas

Los smart contracts del protocolo fueron implementados en el lenguaje Solidity, versión 0.7.6, utilizando las librerías de OpenZeppelin que, además de ahorrar tiempo de desarrollo, brindan código estandarizado y seguro.

Además, se utilizó el entorno de desarrollo Hardhat que facilitó automatizar distintas tareas como deployments, verificaciones de código fuente en exploradores de blockchain, tests y debugging. Para esta última tarea también se hizo uso de Tenderly.

7.1.2. Implementación

Para la prueba de concepto se desarrollaron cuatro smart contracts, cada uno representando un módulo del protocolo. A continuación se describe el rol de cada uno de ellos.

7.1.2.1. Core del protocolo Este smart contract[18] es el principal del protocolo Gazzeth. Cuenta con las funcionalidades para suscribirse y desuscribirse como jurado en un cierto tópico, realizar una publicación, votar a través del commitment-scheme definido previamente, ejecutar penalizaciones y recompensas *minteando* GZT, entre otras.

Este contrato se integra con los otros tres que forman parte de la prueba de concepto y, adicionalmente, con el contrato de DAI, dado que es el token utilizado para los depósitos. En la prueba de concepto se utilizó un contrato de DAI que ya estaba desplegado en Ropsten.

Estas dependencias son configuradas a través del constructor del contrato, junto con otros parámetros como el número de jurados mínimos para poder realizar una publicación en un tópico, el número de jurados elegidos en cada votación, el costo de los depósitos, tiempo de duración de cada fase de la votación y cantidad de GZT a dar como recompensa.

Además, este smart contract emite los eventos que luego se utilizan para indexar la información de la blockchain y generar la API abierta a través de The Graph.

7.1.2.2. Generador de números aleatorios Este smart contract[19] tiene como finalidad generar números aleatorios, que luego son utilizados por el contrato core del protocolo para seleccionar los jurados.

El contrato implementa una interfaz de una única función que tiene la siguiente firma:

```
function getRandomNumbers(uint256 _quantity) external returns (uint256[] memory);
```

La función recibe la cantidad de números aleatorios deseados y retorna un arreglo, de dicha longitud, de enteros aleatorios. Luego, cada vez que se publica una noticia, el contrato core del protocolo pide a este módulo tantos números aleatorios como jurados necesite seleccionar y transforma dichos números en índices que utilizará para elegir a los jurados dentro de una colección.

La implementación de este módulo, para la prueba de concepto, hace uso del hash de los últimos bloques de la red como fuente de aleatoriedad. Dado que el smart contract core integra este módulo a través de su interfaz, en el futuro pueden explorarse distintas implementaciones buscando obtener mayor seguridad y aleatoriedad, por ejemplo a través de oráculos y *Verifiable Delay Functions*, y adaptarlas a la firma de la interfaz para evitar cambios en la implementación del contrato core.

[18] https://github.com/gazzeth/protocol/blob/master/contracts/Protocol.sol
[19] https://github.com/gazzeth/rng/blob/master/contracts/Rng.sol

7.1.2.3. Token GZT Este es el smart contract[20] del token de la plataforma, GZT. Este contrato implementa el estándar de token fungible ERC-20 y el EIP-2612[21] que introduce la función `permit`, la cual permite ejecutar meta-transacciones equivalentes a la función `approve` del ERC-20 a partir de una firma EIP-712.

Si bien la función `permit` tiene varias utilidades, algunas de ellas detalladas en el EIP-2612, en el contexto de este proyecto se encuentra especialmente útil para implementar depósitos de tokens en un contrato inteligente en una única transacción[22], estrategia que se utilizó para los depósitos de DAI en el contrato core del protocolo, dado que mejora la experiencia de usuario.

Además de los estándares mencionados anteriormente, implementa las funciones `setProtocolContractAddress`[23], `mint` y `burn`. La primera utilizada para configurar la address del contrato core, que será la única que podrá usar la función `mint`, la cual permite emitir nuevos GZT hacia una dirección en particular; esto es utilizado por el protocolo para emitir recompensas. Por último, la función `burn` permite destruir o "quemar" tokens GZT, ésta puede ser ejecutada por el contrato core, por el dueño de los GZT a destruir o por alguien que tenga aprobada la extracción de dicha cantidad de tokens sobre la address de la cual serán quemados.

7.1.2.4. Proof of Humanity Este contrato[24] es una implementación de Proof of Humanity ad-hoc para probar su integración en testnet. El objetivo es tener un contrato que cumpla la interfaz de Proof of Humanity del cual el desarrollador tenga control total. De esta forma, se puede dar de alta o baja direcciones de Proof of Humanity simulando los distintos posibles estados del registro sin la necesidad de realizar el proceso oficial ni depender de ningún tercero.

La interfaz de Proof of Humanity sólo cuenta con la siguiente función:

```
function isRegistered(address _address) external view returns (bool);
```

Como se puede apreciar, la función recibe una address y retorna un booleano indicando si la misma está dada de alta o no.

El protocolo de Gazzeth requiere estar registrado en Proof of Humanity para poder suscribirse como jurado exitosamente, sin embargo, si éste es dado de baja de Proof of Humanity luego de haberse inscripto como jurado, el contrato core de Gazzeth no será alertado al respecto y el jurado podría seguir cumpliendo su rol como tal. Para evitar esto, la implementación de la función de reveal del esquema de votación que se realizó requiere que el jurado se encuentre en dicho

[20] https://github.com/gazzeth/protocol/blob/master/contracts/Gazzeth.sol

[21] https://eips.ethereum.org/EIPS/eip-2612

[22] El contrato que extrae el monto lo puede hacer a través de `permit` y `transferFrom` en una única transacción, en lugar de el usuario tener que primero ejecutar una transacción de `approve` y luego interactuar con el contrato para que éste ejecute el `transferFrom`.

[23] Para la prueba de concepto, `setProtocolContractAddress` puede ser invocada una única vez, sólo por la address que realiza el despliegue del contrato.

[24] https://github.com/gazzeth/proof-of-humanity-testnet/blob/master/contracts/ProofOfHumanity.sol

momento registrado en Proof of Humanity para ser ejecutada exitosamente. De esta forma, el jurado está incentivado a desuscribirse como jurado o a renovar su registro en Proof of Humanity, dado que siempre que sea escogido para validar una publicación en Gazzeth no podrá concretar exitosamente la fase de votación y como consecuencia perderá su depósito.

Este smart contract permite configurar a través de su constructor si se quiere permitir auto-registros y si se quiere habilitar super-usuarios, es decir, usuarios con control total de los registros.

7.1.3. Red

Los smart contracts fueron desplegados en la red de prueba, o *testnet*, Ropsten. Las testnets son utilizadas para probar tanto cambios en los nodos, como cambios en los smart contracts, previamente a ser lanzados a la red productiva. Si bien existen varias redes de prueba, se decidió utilizar Ropsten porque es de tipo Proof of Work, al igual que la red productiva, en lugar de ser Proof of Authority como la mayoría del resto de las testnets.

7.2. Almacenamiento de noticias

Cabe destacar que por default IPFS tiene un *garbage collector* que, después de un tiempo, eliminaría las noticias subidas. Por esto se utilizó una funcionalidad de *pinning* que permite que la misma persista en el tiempo. En particular, se utilizó el servicio de *IPFS pinning* de Infura.

7.3. API abierta

Tal como se describió en el diseño de la solución, la API abierta es provista a través del protocolo The Graph. Para la prueba de concepto se utilizó el *hosted service*, que es un servicio de *hosting* que provee The Graph para desplegar el subgrafo creado. Este servicio es centralizado y sirve para realizar pruebas del subgrafo en etapas de desarrollo, previo a desplegarlo en la versión productiva y descentralizada.



Figura 3: Diagrama de las entidades del esquema GraphQL definido y sus relaciones

7.4. Cliente

A continuación se brinda información sobre el desarrollo del cliente que permite al usuario interactuar con los contratos inteligentes de Gazzeth desplegados en la blockchain.

7.4.1. Implementación

El cliente es una aplicación web de una sola página (*single page application*). Por esta razón, el contenido de la página sólo es cargado por el usuario una sola vez.

Para el desarrollo de este se utilizó el framework React, utilizando el lenguaje TypeScript, el cual permite definir componentes que se pueden reutilizar

y componer para generar las vistas del cliente, renderizando y actualizando lo que corresponde de manera eficiente. Se utilizó la librería Material-UI como base para crear los componentes y la librería ethers.js para comunicarse con los smart contracts.

Es importante destacar que se utiliza la *arquitectura limpia*. En esta se definen distintas capas y sus interfaces dejando el código de estas desacoplado, lo cual permite realizar cambios en el mismo sencillamente. Por un lado se tiene los repositorios, los cuales interactúan con sistemas externos (como una API o, en este caso, los contratos de Gazzeth en la blockchain). Luego están los casos de uso, los cuales representan una funcionalidad del usuario e incluyen las reglas de negocio. Finalmente está la vista, la cual consiste de los componentes de React con los que interactuará el usuario.

Los componentes del cliente cuentan con internacionalización permitiendo soportar distintos idiomas y facilitar el agregado de nuevos en el futuro. Por el momento cuenta con inglés, español y japonés.

En el Anexo B se encuentra el manual de usuario del cliente desarrollado.

7.4.2. Estados de la noticia

Cada noticia, una vez finalizada su validación, tendrá en el smart contract un estado de votación asociado del cual se podrá derivar la clasificación de la misma.

Como se explicó previamente, el protocolo es público y está desligado de cualquier cliente, permitiendo a cada uno de ellos elegir cómo definir la interfaz. Para demostrar esto, el cliente implementado para la prueba de concepto define sus propios estados de validación a partir de los resultados de la votación. A continuación se listan las reglas que determinan cada uno de ellos, las cuales son evaluadas para cada noticia en orden, eligiendo como estado el de la primera que se cumpla:

- Si no terminó la votación de la validación de la noticia, la misma se considera *Pendiente*.
- Si no votó más de 65 % de los jurados seleccionados, se la considera como *Votos insuficientes*.
- Si los votos *Verdadera* superan el 60 %, se considera su estado como *Verdadera*.
- Si los votos *Falsa* superan el 60 %, se considera su estado como *Falsa*.
- Si los votos *No cualificada* superan el 60 %, se considera su estado como *No cumple las normas*.
- En cualquier otro caso se considera su estado como *Sin consenso*.

7.4.3. Despliegue

Se utilizó GitHub pages para desplegar el cliente debido a que es sencillo y sin costo económico. Además, se configuró un registro de servidor DNS para que la URL `gazzeth.com` apunte al cliente desplegado.

8. Trabajo futuro

A continuación se presentan algunos puntos en los que continuar trabajando:

- **Definición de cantidades:** A lo largo del documento se ha evitado definir cantidades y montos. Por ejemplo, el monto necesario para inscribirse como jurado o publicar, el monto para crear una apelación, la cantidad de jurados mínima para comenzar a publicar en un tópico, la cantidad de jurados a seleccionar para una validación y el tiempo de duración de cada fase de votación. Se considera que definir estos valores objetivamente para que se alineen con la misión del proyecto requiere un estudio que excede el alcance de este trabajo.
- **Reducción de costos:** En el Anexo A se pueden ver los costos de interactuar con la prueba de concepto implementada. Estos costos son altos haciendo inviable el uso de la L1 de Ethereum para este proyecto. Como solución a esto se deben explorar las distintas soluciones de L2 de Ethereum, como Optimistic Rollups y ZK-Rollups, que reducen el costo de transacción y, además, optimizar el código con el mismo fin.
- **Experimentación en mainnet:** Para verificar si el proyecto efectivamente obtiene el resultado esperado debe desplegarse en mainnet, estudiar el comportamiento de los usuarios y los resultados de las validaciones que se efectúen en él.
- **Sybil-resistance con privacidad:** Es importante para Gazzeth contar con un mecanismo de sybil-resistance que provea privacidad. Para esto, se considera importante trabajar sobre Proof of Humanity para que pueda brindar dicho atributo o desarrollar algún mecanismo alternativo que cumpla con las condiciones deseables para este proyecto.
- **Buscador de texto IPFS descentralizado:** Las reglas para elección de voto incluyen que la noticia a publicar brinde información que no se encuentre en dicho momento en el tópico donde se está publicando. Actualmente no existen las herramientas para lograr esto de forma descentralizada. Se propone trabajar en un protocolo que indexe archivos IPFS y exponga un motor de búsqueda de texto para buscar sobre ellos. Podría considerarse una extensión de The Graph y, por lo tanto, proponerse como mejora del mismo a través de su gobernanza.
- **RNG:** Desarrollar un módulo de generación de números aleatorios que sea más eficiente y provea mayor seguridad.

9. Conclusión

Gazzeth logra cubrir los inconvenientes planteados al comienzo del documento, proponiendo el diseño de una plataforma de noticias que se presenta como un bien público, global, resistente a censura, sin necesidad de financiamiento de ninguna entidad, permitiendo que cualquiera pueda publicar noticias en ella, incluyendo sistema de validación de forma nativa, brindando mecanismos para auditar dicho proceso de validación y actuar en caso de que parezca incorrecto, involucrando a los distintos usuarios de la plataforma y permitiendo mecanismos democráticos para la evolución de la misma.

Este tipo de plataformas, que surgen de la creación de protocolos descentralizados, resistentes a censura, transparentes, permissionless, globales, conforman un nuevo paradigma que comienza a ser referido como *Web 3.0*.

Es importante que se ponga foco en mejorar la usabilidad de las herramientas que el usuario utiliza para interactuar con este tipo de plataformas, como las wallets, en búsqueda de bajar las barreras de entrada, que actualmente son relativamente altas y, en muchos casos, dificultan la adopción de estas tecnologías.

De cara al futuro no queda más que optimismo. La *Web 3.0* a través de su transparencia y su característica permissionless permite la componibilidad y cooperación de distintos protocolos, que junto con la competencia, producen una evolución constante del ecosistema, que se traslada en herramientas sociales cada vez más poderosas para los usuarios.

Referencias

[1] T. C. Schelling. *The strategy of conflict*. Harvard University Press.

[2] C. Lesaege, F. Ast y W. George. *Kleros*. URL: https://kleros.io/whitepaper.pdf.

[3] V. Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. URL: https://ethereum.org/en/whitepaper.

[4] J. Benet. *IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3)*. URL: https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf.

[5] MakerDAO. *The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System*. URL: https://makerdao.com/en/whitepaper.

[6] DemocracyEarth. *The Social Smart Contract*. URL: https://github.com/DemocracyEarth/paper#324-attention-mining.

[7] T. Pornin. *RFC 6979: Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)*. URL: https://tools.ietf.org/html/rfc6979.

[8] R. Bloemen, L. Logvinov y J. Evans. *EIP-712: Ethereum typed structured data hashing and signing*. URL: https://eips.ethereum.org/EIPS/eip-712.

[9] G. Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger, Appendix F*. URL: https://ethereum.github.io/yellowpaper/paper.pdf#appendix.F.

[10] V. Buterin. *Blockchain voting is overrated among uninformed people but underrated among informed people*. URL: https://vitalik.ca/general/2021/05/25/voting2.html.

[11] J. R. Douceur. *The Sybil Attack*. URL: https://www.cs.yale.edu/homes/aspnes/pinewiki/attachments/SybilAttack/sybil-attack.pdf.

[12] D. Siddarth, S. Ivliev, S. Siri y P. Berman. *Who Watches the Watchmen? A Review of Subjective Approaches for Sybil-resistance in Proof of Personhood Protocols*. URL: https://arxiv.org/pdf/2008.05300.pdf.

[13] M. Borge, E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly y B. Ford. *Kleros Documentation: Proof of Humanity*. URL: https://kleros.gitbook.io/docs/products/proof-of-humanity.

[14] S. James at Kleros Blog. *Proof of Humanity - An Explainer*. URL: https://blog.kleros.io/proof-of-humanity-an-explainer/.

[15] M. Laskus. *Decentralized Identity Trilemma*. URL: http://maciek.blog/dit.

[16] C. Allen. *The Path to Self-Sovereign Identity*. URL: https://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

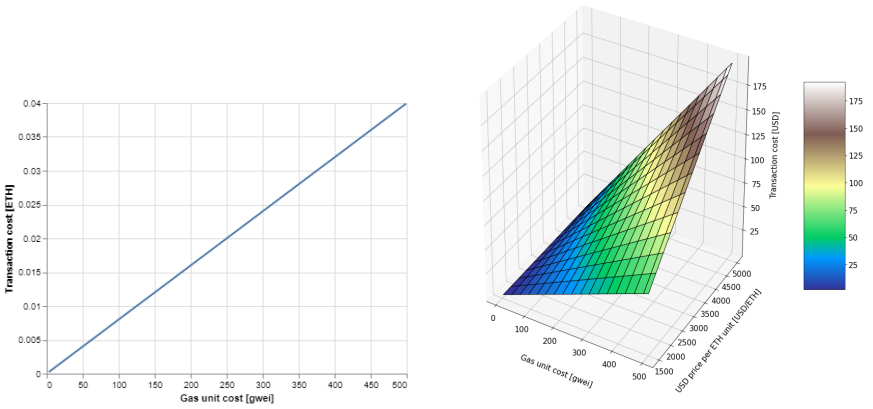
[17] R. Joshi H. Rajat. *Ethereum Push Notification Service (EPNS)*. URL: https://whitepaper.epns.io/.

[18] B. Ramirez. *The Graph Network In Depth (Part 1)*. URL: https://thegraph.com/blog/the-graph-network-in-depth-part-1.

[19] B. Ramirez. *The Graph Network In Depth (Part 2)*. URL: https://thegraph.com/blog/the-graph-network-in-depth-part-2.

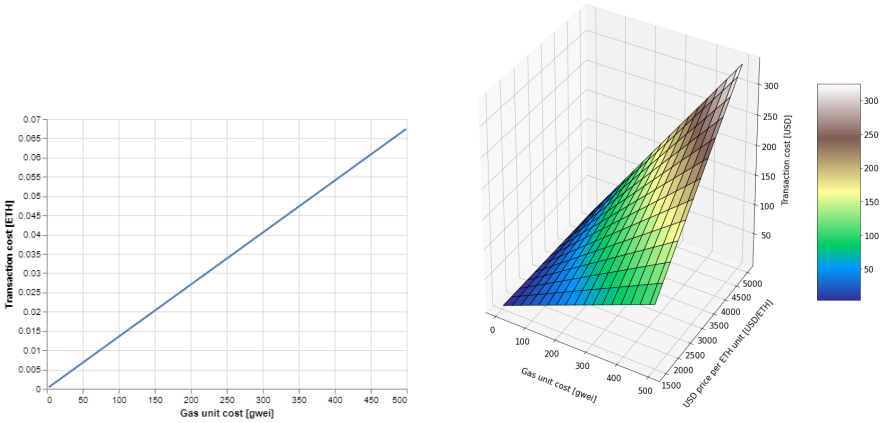
Anexos

A. Costos de transacción



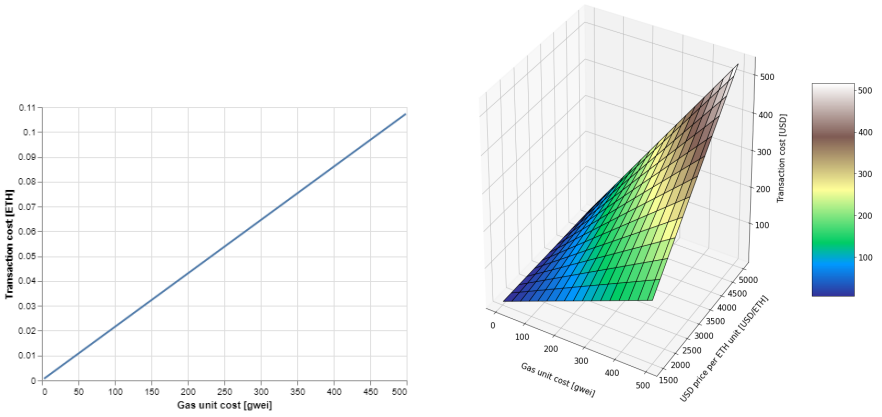
(a) Costo en ETH (b) Costo en USD

Figura 4: Costo de transacción *commit* del voto en Ethereum.



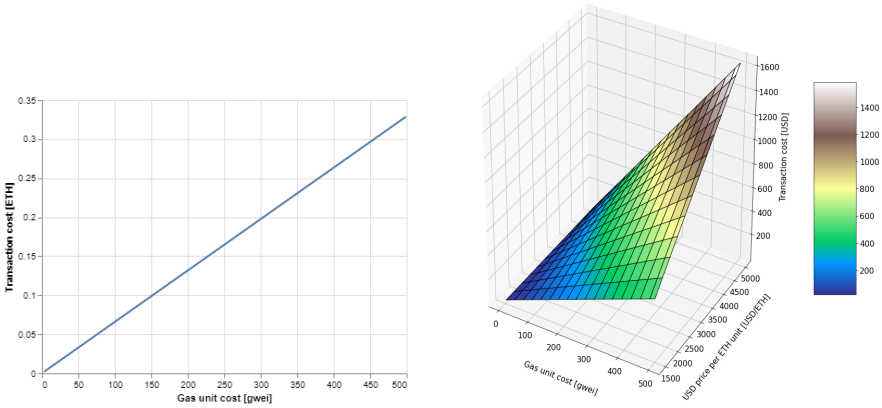
(a) Costo en ETH (b) Costo en USD

Figura 5: Costo de transacción *reveal* del voto en Ethereum.



(a) Costo en ETH (b) Costo en USD

Figura 6: Costo de transacción de voto en total (*commit* + *reveal*) en Ethereum.



(a) Costo en ETH (b) Costo en USD

Figura 7: Costo de transacción de publicación de noticia en Ethereum.

B. Manual de uso

Este anexo describe las distintas funcionalidades que provee el cliente. A continuación se muestra la página principal de la aplicación:

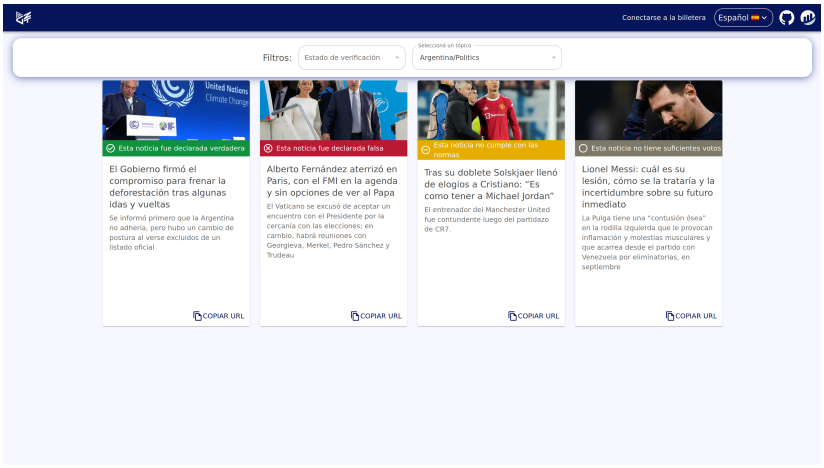


Figura 8: Pantalla principal del cliente.

B.1. Barra de navegación

Como se puede ver en la figura 8, se puede conectar una billetera, cambiar el idioma, acceder al perfil de la organización en GitHub y al contrato desplegado en la blockchain, visualizándolo a través del explorador Etherscan. Estos dos últimos botones proveen transparencia al usuario permitiendo que audite el código con el que interactúa.

B.2. Listado de noticias

Como se puede ver en la figura 8, se listan las distintas noticias. Estas pueden ser filtradas por tópico y estado de verificación. Cada noticia muestra una imagen, título, un breve resumen de la misma y su estado. Al hacer click en la tarjeta de una noticia se puede ir a ver el detalle de la misma.

B.3. Conectar una billetera

En la barra de navegación hay un botón que permite conectar una billetera (figura 8). Al cliquearlo aparecerá un listado de billeteras integradas, en este caso únicamente Metamask. Si el usuario no tiene dicha billetera instalada aparecerá un botón que lo dirigirá a la página de instalación de la misma. A continuación se pueden ver dichas pantallas:

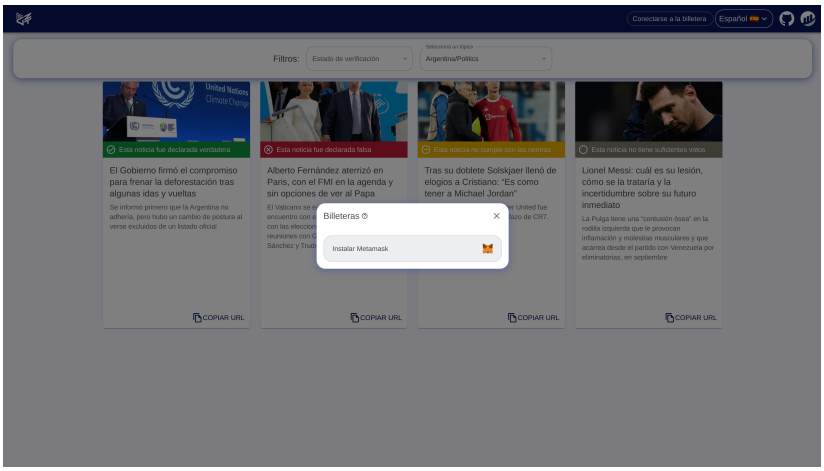


Figura 9: La billetera no está instalada.

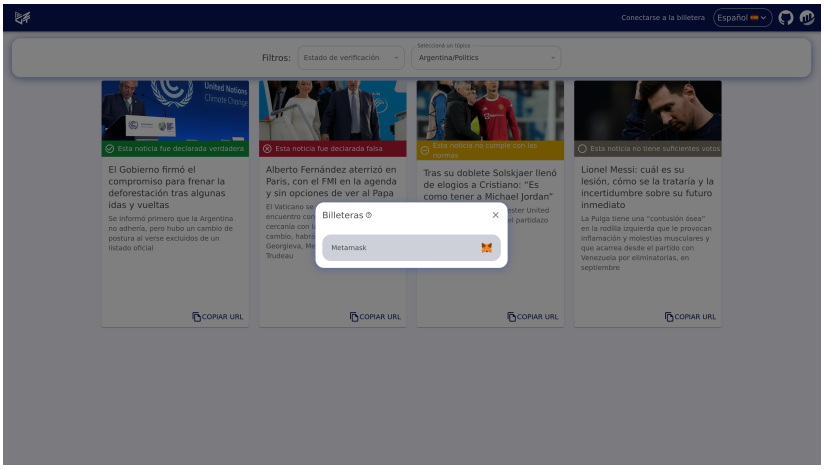


Figura 10: La billetera está instalada.

Una vez seleccionada la billetera Metamask, se mostrará una pantalla de carga y, si no se encuentra actualmente desbloqueada, aparecerá una ventana propia de la billetera pidiendo la contraseña de la misma (figura 11). Una vez se acepte la conexión, volverá a la pantalla principal y aparecerán más opciones en la barra de navegación (figura 12). Caso contrario se mostrará un error.

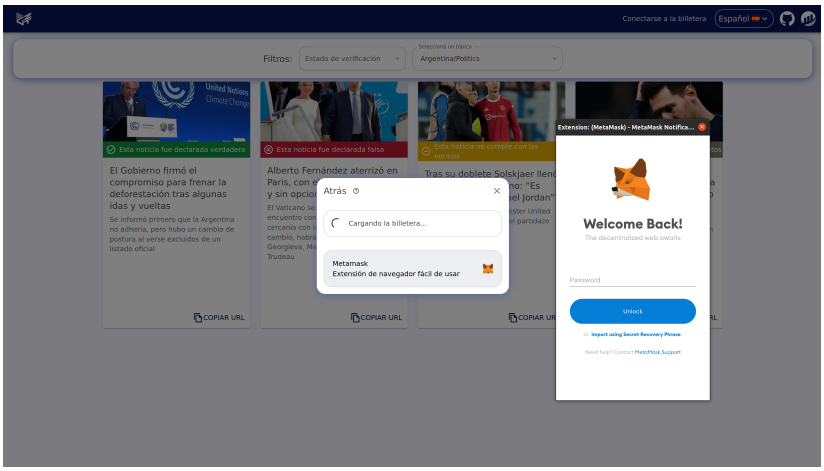


Figura 11: Conecta Metamask como billetera.

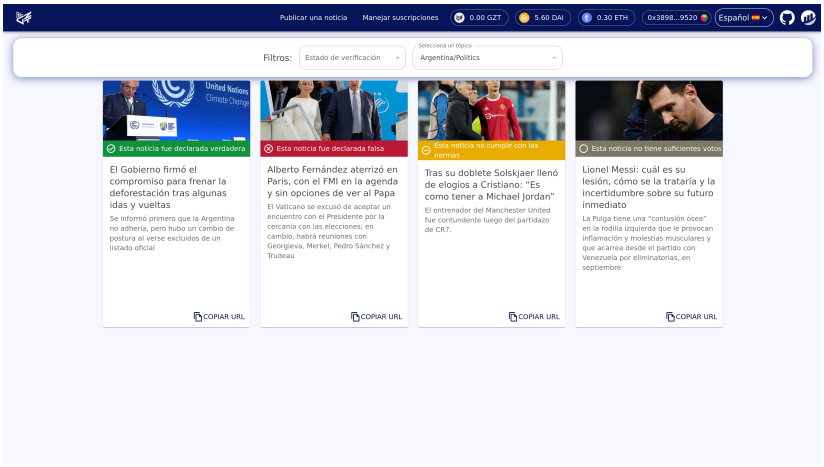


Figura 12: Pantalla principal conectada a la billetera.

B.4. Manejar suscripciones como jurado

Realizando un click en «Manejar suscripciones» en la barra de navegación se puede ir a la página para suscribirse o desuscribirse como jurado, donde se elegirán los tópicos y la cantidad de veces a ser jurado simultáneamente en cada uno. Esta página muestra la cantidad de DAI que se bloquearán, negativo en caso de liberación, como resultado de la transacción. También muestra los tópicos a los que está suscripto y la cantidad en cada uno.

Al apretar el botón de inscripción aparecerá una ventana de la billetera pidiendo que se firme la transacción a realizar. Si la transacción fue un éxito

o presentó un error es transmitido por un mensaje toast. A continuación se muestran las distintas pantallas del proceso:

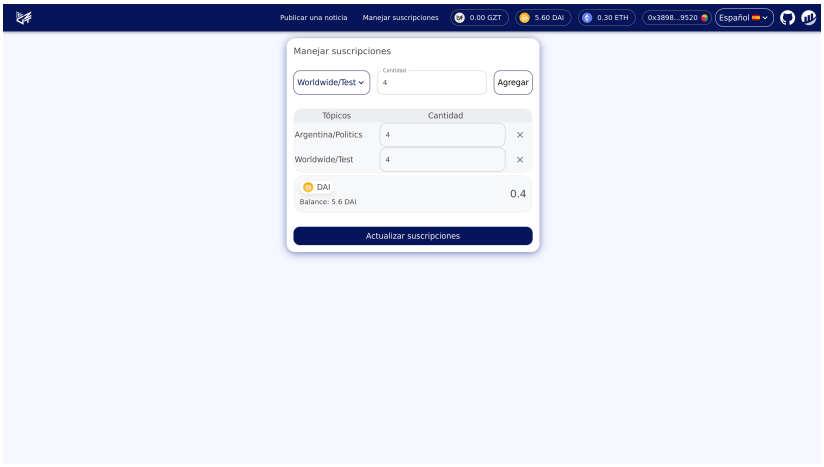


Figura 13: Formulario para suscribirse o desuscribirse como jurado.

B.5. Publicar una noticia

Se puede ir a la página de publicación de noticias haciendo click en el botón «Publicar una noticia». En esta página se elige el tópico donde publicar y se adjunta un archivo de tipo Markdown con el contenido de la misma. Cabe destacar que, si se desea que la vista resumida de la noticia (la que se encuentra en el listado de la pantalla principal) se vea correctamente, existe un formato específico que ya fue discutido en la sección 6.4.

También hay un botón para mostrar cómo se visualizaría la noticia cuando se encuentre en la vista detallada de la misma. Al apretar el botón de publicar aparecerá una ventana de la billetera pidiendo que se firme la transacción de publicación. A continuación se muestran las pantallas:

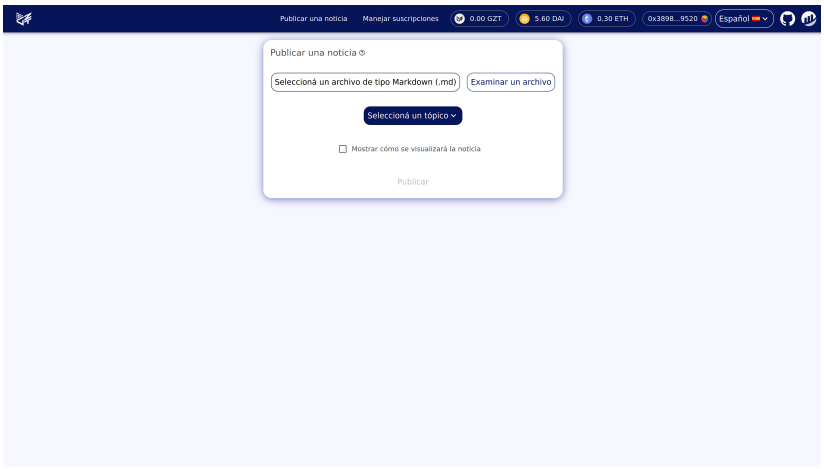


Figura 14: Formulario para subir una noticia.

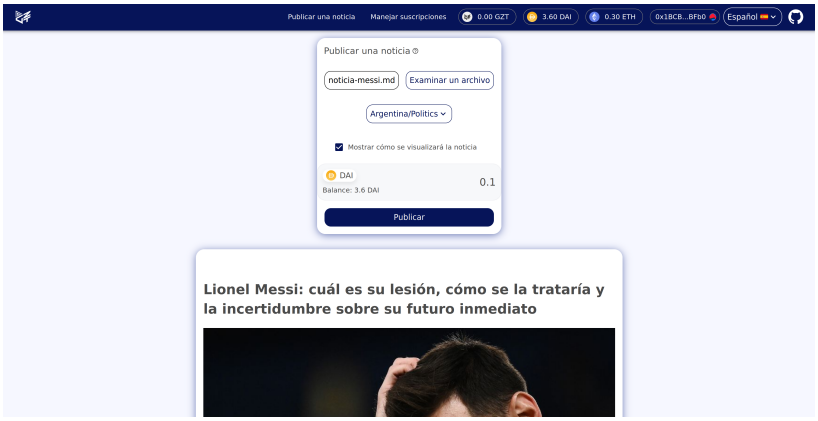


Figura 15: Formulario para subir una noticia con información.

B.6. Ver una noticia

Se puede acceder al detalle de una noticia haciendo click en ella desde la página principal o accediendo directamente a través de un enlace a la misma. En dicha pantalla se puede leer el contenido de la noticia y los resultados de su votación de validación (en el caso de que haya acabado). A continuación se muestra dicha pantalla:



Figura 16: Detalle de la noticia.

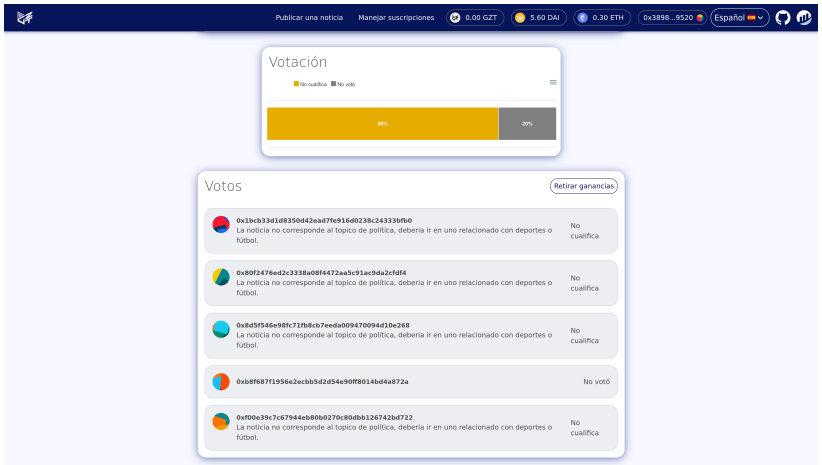


Figura 17: Votación de la noticia.

B.7. Emitir voto

El usuario puede ver si fue seleccionado como jurado haciendo click en su dirección en la barra de navegación. Esto abre una ventana que muestra información sobre la cuenta del usuario, incluyendo las publicaciones en las que está participando. Desde esta pantalla se puede acceder al formulario de votación, donde se puede emitir el voto. Al emitir el voto se pedirá que se firme la transacción con la billetera. A continuación se muestran dichas pantallas:



Figura 18: Perfil de usuario.



Figura 19: Formulario de votación.

B.8. Revelar un voto

De la misma forma que se accede a votar, se puede ir a la pantalla de revelar el voto (ver figura 18). Se debe ingresar el valor del voto que se emitió previamente y una justificación de por qué se eligió dicha opción. Al apretar el botón de revelar, se pide que se firme la transacción con la billetera. A continuación se muestra la pantalla:

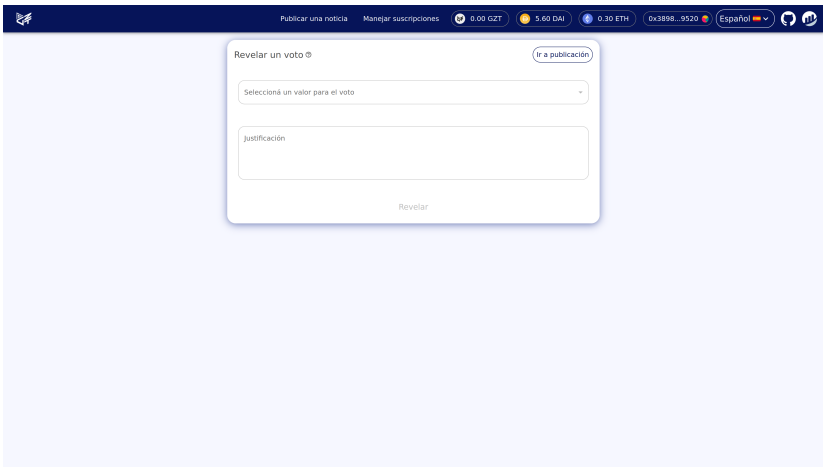


Figura 20: Formulario para revelar un voto.

B.9. Distribuir ganancias

Una vez terminada la fase de revelar los votos, cualquier persona puede realizar la distribución de las ganancias. Esta se realiza desde la pantalla de detalle de la noticia, donde aparece un botón en los detalles de la votación de la misma (ver figura 21). Al apretar el botón se pide que se firme la transacción con la billetera. Esta operación ejecuta las penalizaciones y recompensas.



Figura 21: Votación de una noticia con el botón para distribuir ganancias.