

Instituto Tecnológico de Buenos Aires

escuela de ingeniería y gestión

Gazzeth: A decentralized news platform against disinformation

Authors:
Agustín Dammiano
Alan Donoso Naumczuk

Advisor:
Dr. Alicia Mon

final project submitted for the degree of computer engineer

November 2021

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.*

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.*

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.

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.

Contents

1 Introduction **1**

2 State of the art **2**
2.1 Focal points . . . 2
2.2 Kleros . . . 3

3 Project scope **4**
3.1 Basic operation . . . 5

4 Requirements **6**
4.1 Functional requirements . . . 6
4.2 Non-functional requirements . . . 7

5 Quality attributes **8**
5.1 Security . . . 8
5.2 Transparency . . . 8
5.3 Availability . . . 8

6 Proposed solution **9**
6.1 Architecture . . . 9
6.2 Users . . . 10
6.3 Topics . . . 10
6.4 Publications . . . 10
6.4.1 Format . . . 10
6.4.2 Storage . . . 11
6.5 Mechanism design . . . 12
6.5.1 Deposits and penalties . . . 12
6.5.2 Rewards and the GZT token . . . 13
6.6 Rules for choosing the vote . . . 15
6.7 Commitment-Scheme . . . 15
6.7.1 Classic scheme . . . 15
6.7.2 Proposed scheme . . . 17
6.7.3 Disadvantages . . . 20
6.8 Sybil-resistance . . . 20
6.8.1 Sybil Attack on Gazzeth . . . 20
6.8.2 Solution . . . 21
6.8.3 Disadvantage . . . 21
6.9 Appeals . . . 22
6.9.1 Vote justification . . . 22
6.10 Governance . . . 23
6.11 Phases for a fair launch . . . 25
6.11.1 Phase I: Initial launch . . . 25
6.11.2 Phase II: Topic curation . . . 25
6.11.3 Phase III: Final launch . . . 26

6.11.4 Conclusions about the proposed mechanism 26
6.12 Notifications 26
6.13 Client 27
6.13.1 Types of clients 27
6.13.2 Open API 28

7 Proof of concept **29**
7.1 Protocol 29
7.1.1 Tools used 29
7.1.2 Implementation 30
7.1.3 Network 32
7.2 News storage 32
7.3 Open API 32
7.4 Client 33
7.4.1 Implementation 33
7.4.2 News statuses 34
7.4.3 Deployment 34

8 Future work **35**

9 Conclusion **35**

Appendices **39**

A Transaction costs **39**

B User manual **41**
B.1 Navigation bar 41
B.2 News list 41
B.3 Connect a wallet 41
B.4 Manage juror subscriptions 43
B.5 Publish a news item 44
B.6 View a news item 45
B.7 Cast a vote 46
B.8 Reveal a vote 47
B.9 Distribute earnings 48

1 Introduction

Disinformation is a global phenomenon, where the media, with different goals and interests, appeal to the emotions of citizens instead of putting the facts at the center of the stage. Having control over public opinion seems to be more valuable than reporting the truth.

At the beginning of the 21st century, social networks, messaging applications, blogs, digital news portals and other platforms appeared on the Internet, forming what is known as the *Web 2.0*. This drastically increased the speed at which information spreads and, as a consequence, it is increasingly easy to spread fake news that has an impact on society.

Independent journalism, which tries to stay away from the influence of governments, corporations or other powerful entities, does not get the exposure that traditional media have.

In some countries, *fact-checking* platforms appeared to solve this problem. The idea is that a group of specialists takes relevant or controversial news and verifies it, releasing a new article that explains, with justification, the true and/or false points of each news item. Although the results have been different in each country where it was implemented, in general, the level of trust and adoption was not high.

When trying to improve information verification services to increase their trustworthiness and adoption, some questions arise... How can this service become a public good for all the people in the world? How can we make sure that it works regardless of the type of government, whether it is more or less authoritarian? What would happen if an entity tries to censor the service? And what if it is the government that decides to censor it? What would happen if some entity bribes the people in charge of the validations in order to obtain a certain result? And what if, instead of being bribed, they are coerced? Who funds the service? How can we find a source of funding that does not reduce the level of trust in it? Who decides which publications or facts to validate? What happens if one or more individuals disagree with the result of a validation? Do they have the mechanisms to do something about it with an impact on public opinion? Is it possible to know how the validation process of each news item was carried out? Is it available to be audited by those who consume the final result? How can news consumers be involved in the process?

The launch of Ethereum, in 2015, made it easier to develop decentralized protocols, providing a distributed, secure and public architecture, and opening up a new range of ideas and solutions to problems.

This document presents Gazzeth, a decentralized news platform that seeks to mitigate disinformation by solving the issues raised above, which current fact-checking services fail to solve.

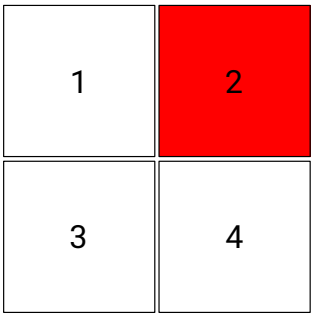
2 State of the art

2.1 Focal points

The option that humans tend to choose by default when they must cooperate in the absence of communication is known as a *Focal point*[1], or *Schelling point* in honor of the person who defined the concept.

This solution is usually chosen over the others because it is perceived as the most intuitive, either because of some aspect of its symmetry or because it seems the most natural. Therefore, if the rest of the people one must cooperate with are rational, it is expected, with a high degree of probability, that they will see it this way as well.

For example, two people, Alice and Bob, are asked to play a game without any kind of communication between them. In the game they are shown the following image:



The players are asked to choose one of the squares, and they are told that, if their choice matches the other participant's choice, they will each be rewarded with 1 BTC. On the other hand, if they do not choose the same option, they will not get any prize. This is the payoff matrix of the described game:

		Alice			
		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	$(0,0)$	$(0,0)$	$(0,0)$	$(1,1)$

Note that there is no square whose choice represents a better or worse outcome than the others. There is no correct answer; Alice and Bob only need to choose the same square to get the prize.

Perhaps 3 is Alice's favorite number and, therefore, she would have chosen the square with that number if it were a matter of taste. However, she is rational and wants to win the prize, and she must match Bob, whom she also assumes

to be rational and, therefore, expects to also want the prize. So Alice will most likely choose the square with the number 2, because its color is different from the rest, expecting Bob to also find this a relevant feature and, as a result, make the same choice. This is what makes the square with the number 2 the focal point of this game.

What would happen if, instead of squares in an image, they had to choose a date of the year? Again, any date would be equally correct as long as they both choose it. However, in the absence of communication, they would probably choose January 1st, since the first date of the year is perceived as the most intuitive solution.

This kind of experiment could be applied to news verification. Alice and Bob are given the same news item and asked to classify it as *True* or *False*. Only if their classifications match will they get the prize. The payoff matrix of this experiment looks like this:

		Alice	
		T	F
Bob	T	$(1,1)$	$(0,0)$
	F	$(0,0)$	$(1,1)$

If Alice and Bob do not know each other, do not know each other's preferences and biases, and have no way to communicate with each other, they are expected to take as a common strategy trying to validate the news objectively, expecting the other participant to do the same, and in this way try to match in the chosen classification.

It is possible that both players, for different reasons, reach different classifications. The focal point in this experiment is the chosen strategy: the search for objectivity.

If this experiment is carried out with a larger number of participants and the economic incentives are adjusted further, for example by adding penalties when the answers do not match, the majority choice will likely represent the objective classification for the given news.

2.2 Kleros

Kleros[2] is a decentralized justice and dispute resolution protocol that runs mainly on Ethereum[3]. It uses incentive mechanisms, based on penalties and rewards, to seek honesty as the focal point for the jurors who act in the disputes run on its platform.

The protocol has different courts depending on the subject to be judged, and it requires jurors to register as such in one of them, leaving PNK, the Kleros token, as collateral.

Then, when a dispute is created in a court, jurors are drawn at random among those available in it, and the probability of being chosen is proportional to the amount of PNK locked in that court.

Jurors must vote by choosing one of the options available in the dispute, for example ruling in favor of or against whoever created it. A juror could be drawn more than once for the same dispute. In that case, the weight of their vote is equal to the number of times they were chosen, and their reward or penalty will also be proportional to it.

Once the vote is over, part of the PNK of the jurors who voted with the minority is distributed among those who voted with the majority, and the value voted by the majority is taken as the resolution of the dispute.

In addition, a fee must be paid for the jurors' work, which will be distributed among those who voted with the majority. Who pays this fee depends on how it was configured. This fee makes sure, for example, that when all jurors vote the same way they still get paid for their work.

Kleros proposed an idea to fight fake news using its protocol[1]. It is based on creating a list of fake news, or multiple lists classified by language, region or other characteristics, where any user can add news, or parts of news, that they consider false, leaving a deposit as a guarantee. Then, there would be a time window in which any other user can challenge the addition of that news item to the list.

If no challenge happens, the news item stays in the fake news list and the deposit is returned to the user who added it. On the other hand, if a user creates a challenge, they must leave a deposit and a dispute would automatically be created in a Kleros news validation court.

Through the mechanism explained above, the court would resolve the dispute by indicating whether the news is false or not. If it is found to be false, the deposit is returned to the user who added the news to the list and the challenger's deposit is used to pay the court fee. If it is found to be true, the opposite happens.

3 Project scope

This project is academic and experimental. The goal of this document is to describe in theory the proposal of a decentralized news platform that is governed by its users, *permissionless*, neutral, censorship-resistant, transparent and global. Then, it describes relevant aspects of the proof of concept that was built, whose purpose is to show that its development is viable.

It is relevant to explain why we decided to design the platform as a new protocol instead of following the Kleros proposal or using it.

First, although Kleros works very well as a dispute resolution service where the parties involved are willing to pay for it, the way Kleros proposes to fight *fake news* through its protocol has misaligned incentives. The user who adds fake news to the list, an activity that should be incentivized, gets nothing in return as a reward. On the contrary, besides paying transaction costs, they risk being wrong and losing the deposit they put up as a guarantee.

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

On the other hand, Gazzeth seeks to be a news platform, which includes both fake and true news. What matters is that each news item is classified as such and that journalists have an incentive to publish on it. Having all kinds of news on the platform would allow readers of traditional media to move to it without problems.

3.1 Basic operation

The following is a simplified list of how the protocol works. It gives an overview that makes it easier to approach the different aspects of its design, which will be detailed in the following sections of the document.

- Topics are created by subject and region, where news will be published.
- Users subscribe to topics in the juror role. They leave a deposit as a guarantee of their honesty, objectivity and ability to validate in that topic.
- When a topic reaches a certain number of jurors, news can start being published in it.
- Anyone can publish a news item in a topic in the journalist role. When publishing, they leave a deposit as a guarantee of the truthfulness of what they publish.
- Jurors are randomly selected from the topic's *pool* of jurors.
- The jurors vote on the status of the news.
- Jurors who vote with the minority are penalized by losing their deposit. Those who vote with the majority keep it and also earn a reward.
- The journalist loses their deposit if the news was not validated as true. Otherwise, they get the deposit back and obtain a reward.

4 Requirements

4.1 Functional requirements

- **Create and close a topic:** It must be possible to create topics, giving the information relevant to them (region, subject, voting duration, deposit amount, etc.). Then, it must be possible to close them. Both actions must come from a user's proposal and be decided through consensus, taking into account the reputation of each user.

- **Edit topic attributes:** It must be possible to edit attributes relevant to the topic through the platform's consensus.

- **Subscribe to and unsubscribe from a topic as a juror:** A user must be able to subscribe to a created topic in the juror role, choosing the maximum number of news items they would be willing to validate at the same time in that topic and depositing money, proportional to the chosen number, as a guarantee of their competence in that topic and of their objectivity and honesty when validating.

  In addition, a user who is subscribed to a topic must be able to unsubscribe from it by choosing a new number of news items to validate at the same time that is lower than the current one, which returns part of the deposit proportionally. To unsubscribe completely, they choose zero as the number and recover the whole deposit. They cannot choose a number lower than the number of news items in which they are acting as a juror at the time of unsubscribing.

- **View topic subscriptions:** It must be possible to see which topics one is subscribed to and the number of news items to validate at the same time chosen in each one.

- **Publish a news item in a topic:** A user must be able to publish in a topic in the journalist role. Publishing must require a money deposit as a guarantee of the truthfulness of what is published. When the news is published, the topic jurors who will validate it are selected at random.

- **Vote on the status of a news item:** A user selected as a juror must be able to vote on the truthfulness of the news. The vote must be confidential until the vote casting phase ends. Then, each user must be able to reveal their vote so it is counted and, optionally, give a justification for it.

- **Start an appeal:** When a vote ends, there must be a time window in which any user who disagrees with the result can start an appeal, depositing an amount as a guarantee that their claim is correct. This appeal triggers a new vote, whose outcome overrides the previous ones and is the one taken into account when executing penalties and rewards.

- **Execute penalties and rewards:** When a vote ends, if the time for creating new appeals has passed, it must be possible to execute the rewards

for the jurors who voted correctly and the penalties on the deposits of those who did not. In the same way, the journalist must get their deposit back if the news was considered true, or lose it otherwise.

- **Notify the user:** The juror user must be notified when selected for that role in a publication, both to cast the vote and to reveal it, and when the vote has ended. The end of the vote must also be notified to the user who wrote the news.
- **News list and filters:** A user must be able to list all the news, seeing a summary of each item with its associated validation status. In addition, they must be able to filter them, for example, by topic, by validation status and/or by author.
- **View a news item in detail:** A user must be able to select a listed news item to read it in detail, where its associated validation status must also be visible.
- **View the validation details of a news item:** A user must be able to see the details of the vote held to validate a news item: the value of each cast vote and their justifications.

4.2 Non-functional requirements

- **Global:** It provides a solution that is not limited to any particular country or region.
- **Decentralized:** There is no central authority or entity that regulates it.
- **Transparent:** It allows seeing and auditing in detail each of the processes involved.
- **Censorship-resistant:** Records that were already processed cannot be changed or deleted. This includes the news, their validation votes, juror subscriptions, etc.
- **Available:** The system is always running.
- ***Permissionless*:** Anyone who follows the rules can participate (publishing and/or validating news), without exceptions.
- **Accessible:** Entry barriers must be as low as possible.
- ***Trustless*:** Participants do not need to know other participants or trust them; they only need to trust the platform's mechanisms.
- **Confidential votes:** During the vote casting stage, votes must remain confidential, so partial results cannot be calculated.

5 Quality attributes

5.1 Security

The protocol incentivizes users to behave in the desired way through *mechanism design*, setting penalties and rewards, and seeking objectivity as the focal point in news validation.

It is important to make sure that votes are confidential during the vote casting phase, so partial results cannot be calculated. Otherwise the mechanism would be compromised and, therefore, so would the trust in the results of the platform's processes.

In addition, it is important that neither the users' deposits nor the news files can be compromised, that users cannot be impersonated, and that no vote, and therefore no voting result, can be manipulated or deleted.

5.2 Transparency

Transparency is an essential property that an entity taking the responsibility of validating information must have if it wants society to accept it in that role. Without transparency, suspicions about the processes can arise and, as a consequence, the results of those processes can lose credibility, making the entity practically irrelevant. That is why the platform must be completely transparent in each of them.

5.3 Availability

As mentioned before, the platform works through rewards and penalties. Not voting implies an economic penalty, and it is not possible to vote if the system is down.

News validation can compromise relevant and/or powerful entities, which may lead them to organize attacks against the availability of the platform to prevent the validation process from working correctly.

In addition, the platform seeks to provide a global solution. This implies running, at least, during the hours with the highest news flow in each time zone.

6 Proposed solution

The following sections describe the design of the platform, starting with its architecture and then detailing the different aspects of its protocol design.

6.1 Architecture

Given the quality attributes to prioritize, we decided to build the platform on a blockchain. In particular, we decided to use Ethereum because it is the public blockchain with *smart contract* support that has the highest adoption and decentralization, which makes it possible to meet the security, transparency and availability attributes. In addition, its community and network effect are great advantages to generate adoption and to take advantage of protocol composability which, as will be seen throughout the document, made it possible to solve different problems.

The Gazzeth protocol will be implemented as several smart contracts designed to meet the non-functional requirements stated above.

Then, there will be one or more clients that allow users to interact with the protocol in a friendly way.

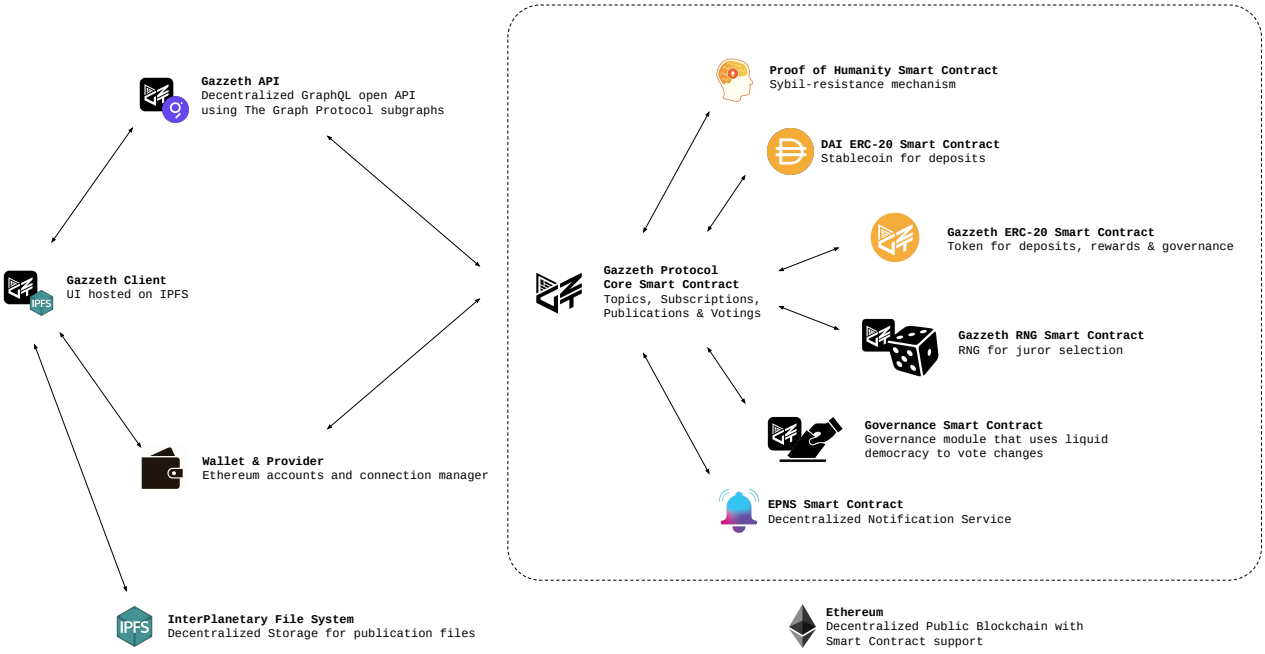


Figure 1: Architecture of the proposed solution.

6.2 Users

Each user can take three different roles within the platform, and these roles are not mutually exclusive:

- **Journalist (or Author):** User who publishes news on the platform.
- **Juror (or Validator):** User who validates news published on the platform.
- **Reader (or Consumer):** User who consumes news from the platform.

6.3 Topics

Topics exist to allow publications to be categorized by subject and region. In this way, users can act as jurors only in topics with publications about subjects they consider themselves able to verify, and in a language they understand. This makes it possible to provide a global solution.

Topics are identified by a string, for which the following scheme is suggested:

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

For example, for information about public speeches of Argentine politicians, the following topic could be created:

```
Argentina/Politics/Speeches
```

Note that the region of a topic might not be a country. It can be a continent, province, state, a combination of these, or others. There are even cases where the information might not be tied to a particular region, such as *airdrops*[2] of *tokens* of protocols that run on Ethereum. For this case it is recommended to agree on a common language, such as English, and then create the topic:

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

A topic can only be activated through governance mechanisms, which will be explained later in this document, and news can only be published in it when it exceeds a certain threshold of jurors available for selection.

6.4 Publications

6.4.1 Format

Publications are represented by a text file in the Markdown language. Standardizing the format allows protocol clients to process the file, and to assure

[2]In the context of protocols that run on a blockchain, an airdrop is the process of distributing a token, usually used for governance, where the eligibility to obtain it is usually tied to certain conditions such as, for example, previous participation in or use of the protocol in question.

users that when a file cannot be understood it is because it does not follow the protocol rules.

We decided to use Markdown because it gives the author flexibility to structure the article, allowing text formatting (both style and size), tables, items, images and more. It is easy to learn, the final file is lightweight and it is easy to convert to other formats such as HTML.

In particular, we propose following a structure at the beginning of the file: title, lead and an image that represents the news, all separated by a blank line. Then comes the body of the news.

In this way, clients that interact with the protocol can take advantage of this standardization to easily interpret the information and use it, for example, to *render* the news in a preview or *feed*.

The proposed structure is the following:

```
Title

Lead

![](MainImageUrl)

News body
```

6.4.2 Storage

The files that represent publications must be stored in a distributed way. Relying on a central server would make the whole system lose the desired decentralization and censorship-resistance properties, since the server that hosts the files could delete them (either all of them, or selectively) or be the target of attacks.

A first approach to the solution would be to store the data on the blockchain. However, this is expensive: it makes a publication transaction so costly that it becomes practically unviable for the required use.

That is why we decided to store the publications in the InterPlanetary File System (IPFS)[4], a distributed storage. The information is stored in a *Merkle Directed Acyclic Graph*. Successfully storing a file in IPFS returns a multihash of the node of that graph, which is used as the resource identifier (content identifier, CID)[3] and serves to retrieve the file later.

We decided that this CID is what gets stored on the blockchain as part of the protocol, associated with the publication. This is much cheaper and keeps the desired properties. IPFS guarantees that the information stays immutable, that is, the news files cannot be modified.

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

6.5 Mechanism design

6.5.1 Deposits and penalties

When publishing, journalists must make a deposit of a certain amount in DAI[5], or 80% of its equivalent value in GZT (the platform token, which is presented in the next part of this section), as a guarantee that what they publish is truthful news relevant to the topic in question. If the publication is not considered true after the validation process, the journalist will lose the whole deposit. In this way, the journalist is not only discouraged from publishing news with false information, but also encouraged to cite sources and to write the news in a way that makes it easier for the jurors to verify its truthfulness, which minimizes the risk of losing the deposit.

In a similar way, jurors must make a deposit of an amount in DAI, or 80% of its equivalent in GZT, to subscribe in that role to a topic and so be available to be chosen to validate news. This deposit is a guarantee of the user's competence and objectivity to validate news of the topic they subscribe to. If the vote they cast when validating the news does not match the majority vote, the user loses the deposit. In this way, the user is discouraged from subscribing as a juror to topics where they do not consider themselves competent, and is encouraged to research in depth the news they have to validate, carefully analyzing the vote to cast on it.

Since we want Gazzeth to be as accessible as possible to all kinds of users, using non-stable currencies or tokens for deposits, such as ETH or GZT, could discourage the adoption of the platform by users who prefer not to be exposed to volatility. For example, a user who subscribes as a juror and makes their deposit should, unless they have validated news incorrectly, be able to leave the platform without economic losses (measured in fiat money and excluding the cost of transacting on the blockchain). This might not hold with a volatile token, since its price could go down during the period in which it was used as a deposit.

That is why we decided to offer a stable alternative for deposits through the DAI stablecoin[4]. The decision to allow GZT as the only volatile token for deposits, and to require a value 20% lower than in the stable alternative when it is used as a deposit, aims to encourage demand for the token. The next section explains why this is desirable.

6.5.1.1 Paid fact-checking service Note that a user in the author role, instead of acting as a journalist who writes a truthful news item and publishes it on Gazzeth in exchange for the reward, could take a news item that seems suspicious to them (perhaps from a topic they do not consider themselves fit to validate) and submit it to the jurors of the corresponding topic for validation. They would expect it to be false and accept in advance the likely loss of the

[4] Although an alternative to the GZT token would have been necessary since GZT has no premine, this alternative could have been temporary and did not need to be a stablecoin.

deposit made when publishing, thus using the platform as a paid *fact-checking* service.

6.5.2 Rewards and the GZT token

When the publication of a journalist user is validated as true by the jurors, they not only get their deposit back, but are also rewarded with one Gazzeth, the platform token, whose symbol is GZT. This is how it will be called from now on. In the same way, the jurors who vote with the majority not only keep their deposit, but are also rewarded with one GZT each.

The GZT token represents the reputation[5] of each user on the platform and is an *equity token*, since it is the governance token of the protocol. It follows the ERC-20 fungible token standard[6], to be compatible with Ethereum *wallets* and protocols.

Most of the times[7] a news item published on the platform is validated, some users are expected to be rewarded with GZT. These tokens will be issued specifically to be given to those users, which increases the GZT monetary base and, therefore, dilutes the reputation of the rest of the platform users (those not involved in the publication and validation process of this news item). From this, there are two ways to increase or keep the reputation in the system:

- **Participating actively in the platform:** Either publishing true news or validating news correctly.
- **Buying the reputation of other users:** Since the GZT token is an ERC-20, it can be transferred from one account to another and be used in other protocols and/or *exchanges*, which allows users to give it a market value and trade it.

What is interesting about being able to get the GZT token through trade is that it allows reader users of the platform to gain decision power over the protocol they consume, without having to act as jurors or journalists. This not only contributes to the decentralization of the protocol, but also allows journalists and jurors to turn the work invested in the platform into economic benefit.

On the other hand, GZT issuance is expected to have two phases. The first one, in the short and medium term, is a transition period in which the platform gains adoption and the number of news items published on it each month grows, so token issuance grows too. The second one, in the long term, is a stationary period in which the platform is already established and the number of news items published on it each month is practically constant, so GZT issuance behaves in a similar way.

[5] Except for the fact that reputation is generally seen as something that cannot be transferred.

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

[7] There are cases where no majority vote comes out of the validation and, therefore, no user is rewarded.

Taking into account the expected behavior of GZT issuance, and the interest in it gaining economic value, creating demand for the token is highly desirable. That is why its use for deposits is encouraged, in addition to it being required for aspects such as governance and appeals, which will be presented in detail later.

The GZT lost because of penalties will be burned to reduce the token supply, while the confiscated DAI will become part of the protocol treasury, which will be managed by its community through the governance system.

6.5.2.1 About GZT mining In Bitcoin, each BTC is mined through computation, and therefore energy, using the *Proof of Work* algorithm[8]. By analogy, one could argue that each GZT token is *mined* through the attention spent by the user who writes and publishes the news, and by the users who validate it. Attention and human lifetime are limited resources, and the *Attention economy* studies their uses as a *commodity*. This process of mining tokens through attention is known as *Attention Mining*[6], and there are already projects that implement it, such as the Brave Browser with its Basic Attention Token (BAT)[9].

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

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

6.6 Rules for choosing the vote

The mechanism design of the protocol, and the consequences that the result of the vote has for both jurors and journalists, were described above. The following are the three types of vote that will be available in each news validation vote, and the conditions that must be met to choose each of them:

- **True:** The publication is news, it is in the correct topic, it gives information that is not currently available in that topic, and it can be fully checked as true.
- **False:** The publication is news, it is in the correct topic, it gives information that is not currently available in that topic, and it contains at least one piece of data that cannot be checked as true.
- **Unqualified:** Any other case. For example: the publication is not news, it is not in the correct topic, it cannot be understood, a news item with the same information was already published, etc.

Although we could have chosen to have only two types of vote, marking every news item that cannot be considered true as false, we preferred to add a third type of vote to indicate that the publication does not meet the minimum requirements to be judged as *True* or *False*.

In this way, for example, if someone publishes an exact copy of a news item that was marked as *True*, the copy would be marked as *Unqualified* because it is repeated. Without this third type of vote, the validation process would be ambiguous: either an essentially true news item would be marked as *False*, or it would be marked as *True*, which would open an attack vector to obtain GZT from repeated news.

6.7 Commitment-Scheme

Ethereum is a public blockchain: any interaction with a *smart contract* reveals information about the caller, the invoked function and the parameters used, among other data. Therefore, to implement a voting system in which partial results cannot be calculated until the vote ends, it must be done through a *Commitment-Scheme*.

These schemes have two phases. In the first one, *Commit*, each voter must provide information (*commitment*) about the vote in a way that keeps the vote confidential. Then, in the second phase, *Reveal*, each voter reveals the vote, which is counted only after checking that what was revealed matches the *commitment*.

6.7.1 Classic scheme

6.7.1.1 Commit The commonly used solution takes as *commitment* the hash of the vote concatenated with an IV[10]. Taking into account the particu-

[10]If the IV were not used, since the `publicationId` is known, an attacker could compute `hash(publicationId||vote)` for each possible vote value and so find out which one was used.

larities of the protocol, the publication ID, `publicationId`, is added, and the commitment would be:

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

The commit function would be (the voter, who invokes the method, is available as a global variable, `sender`, in each call):

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

6.7.1.2 Reveal Then, the reveal function would receive the `publicationId`, the vote and the IV used, and it would rebuild the commitment, counting the vote only if the rebuilt commitment matches the one provided before:

```
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")
```

Where `hash` is a hash function, `countVote` counts for the voter `sender` a vote of value `vote` on the publication with ID `publicationId`, and `error` reverts the transaction with the given error message.

6.7.1.3 Drawbacks The classic scheme has two drawbacks in the context of this project.

- **Lack of standardization:** Since the protocol runs on Ethereum, it will be public, that is, anyone will be able to create their own client to interact with it.

  Therefore, each client would be responsible for generating the IVs securely, and a client could appear that lets the user choose the IV. This would lose the required randomness properties, for example by letting the user always choose the same value or use a counter, which makes the IV predictable.

  In this way, full standardization of the voting scheme could not be enforced. This makes the scheme insecure and breaks the mechanism design presented, since in some cases the value of a certain vote could be known during the commit phase.

- **Poor usability:** The platform seeks to minimize friction to get as many users as possible, and failing to reveal a vote implies an economic penalty. So the fact that the user must remember or securely store an IV to complete the reveal harms usability (since it is a decentralized platform, there

is no server that stores the IV and, since we want to protect users' identities, collecting personal information to send it to them, such as their email address, is not an option).

This poor usability goes against the *Psychological Acceptability* design principle. It encourages some users to use a client that lets them choose the IV freely to make the reveal process easier, which breaks the security of the voting scheme as mentioned in the previous item.

6.7.2 Proposed scheme

Both drawbacks of the classic scheme come from the IV being part of the commitment, and both can lead to the loss of confidentiality of some votes during the commit phase. That is why we propose a new scheme that allows doing the commit and the reveal without the IV.

6.7.2.1 Basic explanation In this scheme, the voter generates a digital signature of their vote and sends its hash as the commitment. It is important to use the ECDSA algorithm for signing, in its deterministic 256-bit version, as defined in RFC-6979[7], which is supported by Ethereum.

Then, in the reveal, the voter generates the signature again (this is why the signing algorithm must be deterministic; otherwise the signature would not match the one given in the commit) and reveals it together with the value of their vote.

In this way, the protocol smart contract computes the hash of the given signature and checks that it matches the commitment sent before. Then, since ECDSA allows recovering the signer's public key from the signature and the corresponding signed data, the contract uses the vote value together with the signature to recover the signer's public key. This verifies that the given signature is valid and that its underlying data is the indicated vote. Once this succeeds, the voter's vote is counted.

In the commitment, the hash is necessary because, thanks to its preimage resistance property, it keeps the signature confidential. If the signature were sent without hashing it, an attacker could take it and try to recover the signer's public key using each possible vote value. The cast vote would then be the one for which the signer recovery outputs the voter's public key. In this section, every mention of a hash function refers to KECCAK-256, which is supported by Ethereum.

6.7.2.2 Commit To define the scheme, we used the *typed structured data hashing and signing standard* proposed in EIP-712[8]. It considers some additional security aspects, such as the *domain separator*, and, since it is a standard, it is supported by the different *wallets*, which makes it friendly to humans: for example, wallets show the structure and content of the information to sign in a readable way.

The typed vote structure was defined as:

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

Therefore, the *typehash*, which is a constant that identifies the type of the structure to sign, is defined as:

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

Then, the *hashStruct*, which is the hash of the *typehash* together with the values of each field of the structure[11], is defined as:

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

And the information to sign is:

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

The first byte prevents the signature from being a possible valid transaction (since 0x19 is not a valid first value in the RLP encoding used by Ethereum transactions). The second byte indicates the version, as defined in EIP-191. `DOMAIN_SEPARATOR` is the *hashStruct* of the following structure, using the data of the Gazzeth protocol contract:

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

This adds information such as the contract *address* and the blockchain for which the signature is generated, so the contract only accepts signatures that were generated to be used exclusively in it.

Finally, the commitment will be the hash of the deterministic ECDSA signature of the information defined above:

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

And the commit function, taking into account that *nonces* are 0 by default, would be:

[11] EIP-712 defines an encoding function for each value according to its type. Integers must be extended to 256 bits keeping their sign, and laid out in big endian. This was left out to simplify the notation of the definition.

```
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")
```

We decided to use a nonce because the user will be allowed to change their vote by replacing the commitment with another one (as long as it is within the time period of the commit phase), and we want to avoid revealing behavior patterns.

Without the nonce, if a user casts a vote v_0, then changes their mind and casts a different vote v_1, and then changes their mind again and casts a vote v_2 equal to v_0, the signed information in votes v_0 and v_2 is the same (same publication and vote value), so their commitments will be identical.

The nonce makes the information to sign in each vote different, regardless of the vote value. Therefore, all the commitments look different to an attacker, who cannot find patterns in them.

6.7.2.3 Reveal To reveal, the user generates the signature again. They use the same `publicationId`, `nonce` and `vote` that they used for the commit. The reveal function is:

```
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)
```

The `ecdsaRecoverSigner` function, which receives the signature and the signed information and returns the signer, must fail if the signature does not meet what Appendix F, 'Signing Transactions', of the Ethereum Yellow Paper[9] states.

6.7.2.4 About the drawbacks raised Note that with the proposed scheme there is no longer a need to remember a random sequence (IV) to complete the reveal. The `publicationId` is known, and the `nonce` is obtained from the smart contract using the voter and the `publicationId`. The client will provide these data and, therefore, the only thing the voter must remember is the value of the vote they cast for that publication in the commit phase. This solves the

usability problem raised as the second drawback.

On the other hand, the voting scheme is now standardized. Any client that does not follow the EIP-712 scheme defined above will generate invalid votes, which solves the first drawback raised.

6.7.3 Disadvantages

Although blockchain voting can provide interesting attributes such as correctness, transparency and censorship-resistance, there are other relevant aspects of voting that, at the time of writing this document, are not provided[10], not even in the last voting scheme presented:

- **Privacy:** You cannot *see* who voted and who did not, or what each person voted. This seeks to prevent people from choosing their vote based on possible consequences tied to their identity such as, for example, discrimination or social rejection because of the chosen vote.
- **Coercion resistance:** You cannot *prove* to someone how you voted, even if you wanted to. This prevents coercion such as, for example, bribes or threats.

Providing these features is complex on a blockchain, and even more in protocols like Gazzeth, where there is a reward or penalty based on the chosen vote. Gazzeth only provides privacy during the *commit* phase of the vote, to prevent the use of partial results, and it provides appeals (presented later) that help mitigate coercion such as bribes.

6.8 Sybil-resistance

A Sybil Attack[11] happens when a system is corrupted by an entity (or entities) that manages to create and control multiple identities in it, which look independent, and uses them to manipulate the system in an undesired way. Systems that are not vulnerable to this type of attack are considered *sybil-resistant*.

In general, having a central entity with the authority to certify identities solves this problem. However, this is not desirable in decentralized networks or protocols, as is the case of Gazzeth.

6.8.1 Sybil Attack on Gazzeth

Consider a topic with n registered jurors, where n is greater than the minimum number of jurors that must be registered to publish news in that topic.

If news is published in the topic whose verification harms the public image of a certain entity (for example, a government or a company), that entity could carry out a Sybil Attack, creating $m > n$ accounts and subscribing them as jurors to the topic.

Now this entity would control more than half of the jurors of the topic and, since jurors are chosen at random, it would have a probability greater than 50%

of having the majority in the vote and choosing the status of the news as it wishes.

6.8.2 Solution

It is important to keep in mind that increasing the juror deposit too much is not desirable, since it would become inaccessible to a large part of the population. Since the impact of news on society is relevant, a powerful entity (or group of entities) could have enough incentives to pay for the attack (in fact, if the news were verified objectively, the economic loss for that entity could be greater than the cost of the attack). That is why we did not consider the cost of the attack a relevant way to prevent it.

As a solution, we propose integrating a Proof of Personhood protocol[12] that guarantees that each juror is associated with *one and only one* existing human being.

For this, we propose using Proof of Humanity[13][14], a decentralized protocol that keeps an *on-chain* registry of humans, where each one is linked to a single Ethereum address.

In this way, a user can only subscribe as a juror if their Ethereum address is registered in Proof of Humanity. Otherwise, the transaction is reverted with an error. As a consequence, each juror is associated with a different human and, therefore, so is each vote. This prevents the attack described above and makes Gazzeth a *sybil-resistant* protocol.

6.8.3 Disadvantage

Solutions that provide decentralized identity face a trilemma[15]: self-sovereignty[16], *sybil-resistance* and privacy. Currently, there is no solution that covers all three pillars.

Proof of Humanity fails to preserve privacy. Although it allows using pseudonyms without giving personal information (name, age, nationality, etc.), it requires a video in which the person who registers must show their face while showing their Ethereum address and saying a phrase.

Facial recognition tools are already powerful enough for entities that have faces associated with sensitive information, such as companies behind social networks or governments, to identify people through images or videos.

Protesters have already been seen using methods to avoid government facial recognition, for fear of the consequences of their protests[12][13]. A tool that seeks to be a source of truth about information, where this information can often harm some powerful entity, must keep the identity of its participants private to prevent them from being coerced or censored.

[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/

6.8.3.1 Optimism for the future From the above, it is clear that, at the time of writing this document, integrating Proof of Humanity into a protocol where privacy is critical is a disadvantage. However, the protocol was launched less than a year ago. There are already proposals to use *zero-knowledge proofs*, which would allow a scheme where registered humans can use a new address and prove that it belongs to some identity in the registry without revealing which one[14]. This would expose which humans are in the registry, but it would not be possible to know who is acting as a juror in Gazzeth.

6.9 Appeals

After a vote ends, there will be a period of time before the penalties and rewards are executed. During this period, any user who is not satisfied with the result can appeal, which triggers a new vote, but this time with twice as many jurors as the original vote.

For this, the equivalent in GZT of a certain high amount set in fiat currency (for example, in US dollars or euros) must be locked. This amount can be provided by different users (crowdfunding), without any special requirement on them, so that reader users can also take part in this process with the GZT they acquired.

It will be possible to appeal as many times as wanted (that is, to appeal an appeal) as long as the needed number of jurors is available in the topic of the news, and the news is within the time period in which appeals are allowed (this time window starts every time a vote ends, whether it is the original one or the one of an appeal). That is why the more jurors subscribed to a topic (and the more subscriptions each one has in it), the more reliable the topic will be, since it will guarantee the possibility of more appeals.

Both the amount of GZT to lock and the number of jurors required in the vote will double with each appeal on the same news item. This exponential behavior seeks to mitigate the scenario in which an attacker starts the appeal and bribes the jurors to vote as the attacker wants.

Once a vote ends, if the time to appeal runs out, all the penalties and rewards will be executed, taking the result of the last vote as the correct one. Appeals that were made to change a result that is different from the one obtained in the last vote will be considered correct appeals; otherwise, they will be considered incorrect.

Those who contributed GZT to a correct appeal will get it back with a percentage increase (GZT will be issued for this). On the other hand, all the GZT used for incorrect appeals will be lost.

6.9.1 Vote justification

When the vote is revealed, it must be possible to optionally give a justification for it. This justification will be stored on the blockchain, associated with the

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

revealed vote. Since storage on the blockchain is expensive, if the justification is long we propose storing it in IPFS and then giving the CID as the justification using the *ipfs://<cid>* format. In this way, the client could *parse* it, get the corresponding file and render its content as the justification.

The justification is important to provide transparency and to allow reader users to understand the reasons behind the classification of the publication. In addition, the juror user has an incentive to provide it, since it could help avoid an appeal against them or help start and/or win an appeal in their favor.

6.10 Governance

Gazzeth is a decentralized platform where its changes and maintenance are decided through the consensus of its community participants, using a mechanism known as governance. In this way, the participants form a *Decentralized Autonomous Organization* (DAO).

The protocol governance is designed as a *token weighted liquid democracy*[15] using the GZT token (those who have more GZT will have more decision power). Although this would allow a few entities with a lot of economic power to acquire most of the GZT and turn governance into a plutocracy, the dilutive characteristics of the GZT token and its lack of previous mining (*premine*) allow a community of active users interested in the platform to prevent the concentration of power in it.

To open any type of proposal, a user must hold a certain amount of GZT tokens. This prevents *spam* and seeks that proposals come from participants of the community.

On the other hand, for a proposal to be approved, it must not only get the majority vote for its approval, but the total vote weight must also exceed a certain threshold. This prevents the approval of proposals with low participation. In turn, approved proposals will have a delay before they are executed, which allows users who disagree to, for example, sell their tokens, leave the platform or *fork* the protocol.

The vote weight of each user in each governance proposal will be calculated as $min(GZT_{start}, GZT_{end})$, where GZT_{start} is the GZT the user had when the proposal was created and GZT_{end} is the GZT the user had when the voting period of the proposal ends.

There are certain predefined actions that must be done through governance, and whose changes will be supported by the protocol implementation:

- **Topic creation:** To create a new topic, a proposal must be made that justifies its existence and its parameters.
- **Topic deactivation:** If a topic no longer makes sense or is being used inappropriately, its deactivation can be proposed.

[15] Each voter can choose to cast their vote, as in a direct democracy, or delegate it to another actor (or actors) to vote for them, as in a representative democracy.

- **Topic editing:** The rules about the content of a topic can be changed through a proposal.
- **Deposit amount changes:** Since deposit amounts are expressed in DAI, and therefore tied to the US dollar, the amounts will need to be updated periodically (for example, every year) to keep their value against its inflation. In addition, changes can be proposed for other reasons and in particular topics.
- **Reward amount changes:** A change in the amount of GZT received as a reward can be proposed. In addition, it can be proposed to reward journalists with a different amount than jurors.
- **Changes to the duration of the vote casting and/or reveal phases:** A change can be proposed both to how long the vote casting phase lasts and to how long the reveal phase lasts. This change could be proposed only for a particular topic.
- **First appeal amount changes:** A change can be proposed to the amount of GZT that must be locked to make a first appeal on a news item.
- **Number of jurors required to validate a news item:** The number of jurors that must judge a news item can be changed. This change could be proposed only for some topics.
- **Number of available jurors required to allow publishing:** The number of jurors that must be available in a topic to allow the publication of a new news item can be changed. This change could be proposed only for some topics.
- **Governance parameters:** The voting duration, the vote weight threshold to exceed and the amount of GZT needed to make a proposal are all parameters that can be changed through governance itself.

However, the governance process is not limited to the changes listed above. Any type of change to the protocol can be proposed and lead to new versions of it. Changes other than those listed above will probably require changing part of the protocol implementation[16].

For example, it could be decided to exchange part of the DAI in the DAO treasury for another token. Or, taking advantage of the composability of Ethereum, it could be proposed that the treasury DAI, together with the DAI deposited on the platform, be deposited in turn in some *yield farming* protocol. This would generate returns that could fund some GZT *staking* program in exchange for a portion of the generated DAI, or be exchanged for GZT, which would increase its demand.

[16] See 'The State of Smart Contract Upgrades' on the OpenZeppelin Blog for smart contract upgrade patterns. `https://blog.openzeppelin.com/the-state-of-smart-contract-upgrades`

6.11 Phases for a fair launch

One of the aspects we want when launching the Gazzeth protocol is that it is fair from the start. That is why the GZT token has no premine: to mine GZT, news must be published and validated.

However, publishing news requires at least one active topic and, as mentioned before, topics are created through governance using the GZT token. It is clear that, with the mechanisms presented so far, there are problems to get the protocol started.

Allowing a period in which topics can be created without any restriction or special mechanism would make the platform vulnerable. For example, a topic could be created whose description invites voting every publication as true. In this way, if a large enough group of users cooperate to carry out the attack, they could mine large amounts of GZT easily and quickly, gaining majority power in the DAO and preventing the malicious topic from being deactivated, among other undesired actions. This would simply sabotage the launch of the platform, making the platform and all the GZT tokens generated worthless.

On the other hand, defining initial topics arbitrarily, for example by the creators of the protocol, would not meet the fairness and neutrality we want for its launch.

That is why we propose a mechanism that divides the protocol into three phases, seeking to mitigate the launch vulnerabilities without sacrificing the characteristics we want for it.

6.11.1 Phase I: Initial launch

In the first phase of the protocol, any user can create a topic.

Jurors will require Proof of Humanity and will not be able to subscribe to validate more than one news item at the same time in the same topic. In addition, although they will be able to unsubscribe from news validation, this will not release their locked DAI deposit.

Journalists will not unlock their DAI deposit when their news is validated as true. They will only be able to reuse it as a deposit for a new publication.

Both journalists and jurors, instead of GZT, will get *topic points* as a reward, each one represented as a GTP token (*Gazzeth Topic Points*). This token will not be transferable, so it will not follow the ERC-20 standard. In turn, each GTP will have as internal state the ID of the topic of the publication that led to its issuance. This token cannot be used in Phase I: both deposits and the activation of appeals will be done with DAI.

After a period of time following the moment a second topic issues its first GTP, Phase I will end and Phase II of the protocol will start.

6.11.2 Phase II: Topic curation

During Phase II, it will not be possible to create more topics, subscribe jurors, or make publications or appeals.

In this phase, one vote per topic will be activated. In each of them, it must be decided whether its underlying topic should remain in the third phase of the protocol. All the votes will take place at the same time, during the period that the second phase lasts.

Each vote will be a direct democracy weighted by the GTP token, with no minimum number of votes, counting only the GTP that were not issued in the underlying topic. In addition, in each vote only the users who did not participate in that topic (neither as journalist nor as juror) can vote.

The Phase I topics that did not reach the minimum number of jurors needed to allow publications in them will be discarded, and their locked DAI will be confiscated and become part of the protocol treasury.

The same will happen with topics whose vote ends indicating that they must be discarded, or in a tie (even when there are zero votes), at the end of the second phase.

On the other hand, topics whose vote indicates that they must remain will continue to exist in Phase III, and each GTP issued in them can be converted to GZT to be used in that phase.

6.11.3 Phase III: Final launch

At the end of Phase II there will be established topics and GZT in circulation, which sets the initial conditions of the protocol in a decentralized way. The protocol will be ready to work with the rules presented throughout the document.

6.11.4 Conclusions about the proposed mechanism

We consider that the proposed phase mechanism would allow launching the protocol in a relatively fair way: decentralized and permissionless.

However, it must be taken into account that factors such as information asymmetry prevent fully fair launches and that, although these factors are considered mitigated, there are risks that the launch is sabotaged anyway.

The difference is that the proposed phase mechanism prevents a topic from being approved by those who participated in it, makes it harder to carry out the previously described attack, where control of the DAO is taken through a malicious topic, and, if the attack is carried out, reduces its chances of success.

In addition, attacks aimed at sabotaging the launch are discouraged, since they would probably cause large economic losses to the entity that funds them and, afterwards, there will be enough information to make a *fork* that is considered legitimate and starts from Phase III.

6.12 Notifications

Jurors must subscribe to some topic and wait to be randomly chosen to judge a publication. Since a selected juror who does not cast or reveal their vote gets

an economic penalty, it is important that jurors are notified, at least, about the following events:

- When they are chosen to judge a publication.
- When the vote of the publication where they were selected moves from the casting phase to the reveal phase.

However, as mentioned before, it is important that the platform protects the identity of its participants, so asking for personal information such as an email address or mobile phone number must be avoided. We also want to keep all the components of the platform decentralized.

That is why, to provide notifications, we propose integrating the Ethereum Push Notification Service (EPNS)[17], a decentralized notification protocol. It allows creating notification channels and emitting notifications on-chain through events, which can then be indexed and sent to the interested user in the way each client implements[17].

The recipients of the notifications will be one or multiple Ethereum addresses, depending on how the channel is set up. This is the only information required from the recipients.

In addition, EPNS allows choosing a set of parameters for the notifications. One of them is the moment when the notification must be shown to the user. This is relevant for the two notifications mentioned above, since both will be created at the same time, when the jurors are selected. The first one will be shown immediately, to notify the user of their participation as a juror, and the second one will have an *offset* configured for its display time, equal to the duration of the vote casting phase.

6.13 Client

For Gazzeth to be adopted by society, it must be inclusive and accessible, with low entry barriers. For this, it is important that there is at least one client that allows consuming and interacting with the protocol through a user interface in a friendly way.

6.13.1 Types of clients

The launch of the protocol must come with the launch of a client that allows viewing all the news with their validation status and applying different types of filters to them. In addition, it must allow publishing news, subscribing or unsubscribing as a juror in the different topics, voting in the publications where the user was chosen as a juror, and participating in governance.

For this, we propose that the client be a web application, since it is accessible from different types of devices, hosted on IPFS to keep decentralization and

[17]EPNS provides its own official clients. One of them is a mobile application for both Android and iOS, which allows receiving the notifications generated on-chain as *push notifications* on the mobile phone.

avoid censorship. However, anyone could download the code of this client and run it locally or host it on a centralized provider of their choice.

It is important to note that, since the protocol is public and fully decoupled from the client, anyone could create their own version, either by changing the one provided by the protocol developers or by creating one from scratch. This would allow changing the way news is shown, adding new types of filters, changing the rules to choose the status associated with each news item (for example, showing as true only the publications with more than 90% of *True* votes), making specific versions for certain devices or platforms, among other things.

6.13.2 Open API

To create a client that meets the requirements mentioned above, it must be possible to query the information stored in the Gazzeth protocol smart contracts.

Although the protocol contracts could provide random access to publications by their ID, they could not provide a way to query all the data using filters, offset, pagination, sorting and other features that are desirable when querying large volumes of information.

This could be solved in a relatively simple way by setting up a centralized server that indexes the publication data and exposes an API to run the desired queries on that information. However, we want all the components of the architecture to remain decentralized. Otherwise, this server would be a single point of failure that could leave the clients out of service and, therefore, leave many users unable to interact with the protocol.

To avoid depending on a centralized indexer, we decided to use The Graph protocol[18][19]. It decentralizes the indexing and querying layer through actors with different roles and incentives, creating a network of indexers that expose an open API that can be queried using GraphQL.

7 Proof of concept

The following sections describe the proof of concept that was developed. To make it viable within the time frame of this work, Phase III of the protocol was developed directly, and the governance and notification modules were not implemented. In addition, it does not include appeals and deposits are only allowed in DAI.

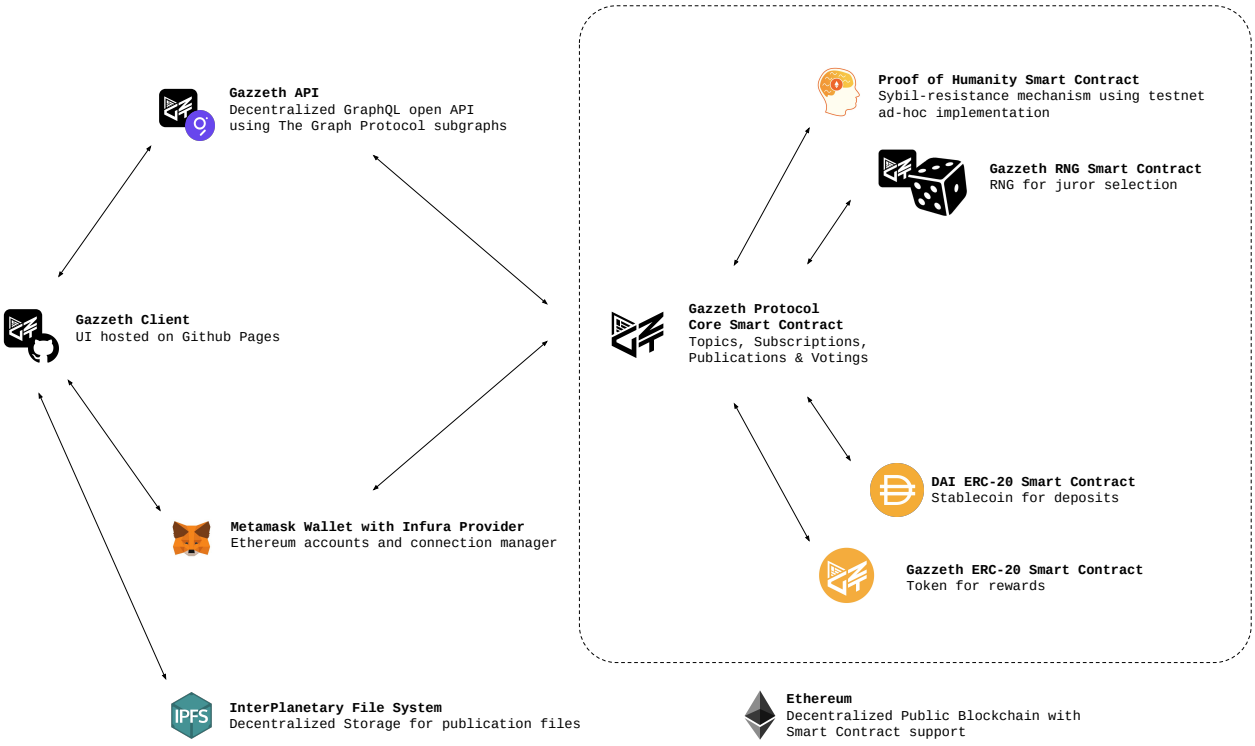


Figure 2: Architecture of the proof of concept

7.1 Protocol

7.1.1 Tools used

The protocol smart contracts were implemented in the Solidity language, version 0.7.6, using the OpenZeppelin libraries, which save development time and also provide standardized and secure code.

In addition, we used the Hardhat development environment, which helped automate different tasks such as deployments, source code verification on blockchain explorers, tests and debugging. For debugging we also used Tenderly.

7.1.2 Implementation

Four smart contracts were developed for the proof of concept, each one representing a module of the protocol. The role of each of them is described below.

7.1.2.1 Protocol core This smart contract[18] is the main contract of the Gazzeth protocol. It has the features to subscribe and unsubscribe as a juror in a certain topic, make a publication, vote through the commitment-scheme defined before, and execute penalties and rewards by *minting* GZT, among others.

This contract integrates with the other three contracts of the proof of concept and, in addition, with the DAI contract, since DAI is the token used for deposits. The proof of concept used a DAI contract that was already deployed on Ropsten.

These dependencies are configured through the contract constructor, together with other parameters such as the minimum number of jurors needed to publish in a topic, the number of jurors chosen in each vote, the deposit cost, the duration of each phase of the vote and the amount of GZT to give as a reward.

In addition, this smart contract emits the events that are then used to index the blockchain information and generate the open API through The Graph.

7.1.2.2 Random number generator This smart contract[19] generates random numbers, which the protocol core contract then uses to select the jurors.

The contract implements an interface with a single function that has the following signature:

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

The function receives the number of random numbers wanted and returns an array of random integers of that length. Then, each time a news item is published, the protocol core contract asks this module for as many random numbers as jurors it needs to select, and it turns those numbers into indexes that it uses to choose the jurors from a collection.

The implementation of this module, for the proof of concept, uses the hash of the latest blocks of the network as the source of randomness. Since the core smart contract integrates this module through its interface, different implementations can be explored in the future to get more security and randomness, for example through oracles and *Verifiable Delay Functions*, and adapted to the interface signature to avoid changes in the implementation of the core contract.

7.1.2.3 GZT token This is the smart contract[20] of the platform token, GZT. This contract implements the ERC-20 fungible token standard and EIP-2612[21], which introduces the `permit` function. This function allows executing

[18] https://github.com/gazzeth/protocol/blob/master/contracts/Protocol.sol
[19] https://github.com/gazzeth/rng/blob/master/contracts/Rng.sol
[20] https://github.com/gazzeth/protocol/blob/master/contracts/Gazzeth.sol
[21] https://eips.ethereum.org/EIPS/eip-2612

meta-transactions equivalent to the ERC-20 `approve` function from an EIP-712 signature.

Although the `permit` function has several uses, some of them described in EIP-2612, in the context of this project it is especially useful to implement token deposits into a smart contract in a single transaction[22]. This strategy was used for the DAI deposits in the protocol core contract, since it improves the user experience.

In addition to the standards mentioned above, it implements the functions `setProtocolContractAddress`[23], `mint` and `burn`. The first one is used to configure the address of the core contract, which will be the only one that can use the `mint` function. That function allows issuing new GZT to a particular address, and the protocol uses it to issue rewards. Finally, the `burn` function allows destroying or 'burning' GZT tokens. It can be executed by the core contract, by the owner of the GZT to destroy, or by someone who is approved to withdraw that amount of tokens from the address they will be burned from.

7.1.2.4 Proof of Humanity This contract[24] is an ad-hoc implementation of Proof of Humanity to test its integration on a testnet. The goal is to have a contract that follows the Proof of Humanity interface and that the developer fully controls. In this way, addresses can be registered in or removed from Proof of Humanity, simulating the different possible states of the registry, without going through the official process or depending on any third party.

The Proof of Humanity interface only has the following function:

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

As can be seen, the function receives an address and returns a boolean that indicates whether it is registered or not.

The Gazzeth protocol requires being registered in Proof of Humanity to subscribe as a juror. However, if the juror is removed from Proof of Humanity after subscribing, the Gazzeth core contract will not be notified and the juror could keep acting in that role. To prevent this, the implementation of the reveal function of the voting scheme requires the juror to be registered in Proof of Humanity at that moment for the reveal to succeed. In this way, the juror has an incentive to unsubscribe as a juror or to renew their Proof of Humanity registration, since every time they are chosen to validate a publication on Gazzeth they will not be able to complete the voting phase and, as a consequence, will lose their deposit.

This smart contract allows configuring through its constructor whether self-registration is allowed and whether super-users are enabled, that is, users with

[22] The contract that takes the amount can do it through `permit` and `transferFrom` in a single transaction, instead of the user having to first send an `approve` transaction and then interact with the contract so that it executes the `transferFrom`.

[23] For the proof of concept, `setProtocolContractAddress` can be called only once, and only by the address that deploys the contract.

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

full control over the registrations.

7.1.3 Network

The smart contracts were deployed on the Ropsten test network, or *testnet*. Testnets are used to test both node changes and smart contract changes before releasing them to the production network. Although there are several test networks, we chose Ropsten because it uses Proof of Work, like the production network, instead of Proof of Authority like most of the other testnets.

7.2 News storage

It should be noted that, by default, IPFS has a *garbage collector* that, after some time, would delete the uploaded news. That is why we used a *pinning* feature, which allows the news to persist over time. In particular, we used the *IPFS pinning* service from Infura.

7.3 Open API

As described in the solution design, the open API is provided through The Graph protocol. For the proof of concept we used the *hosted service*, which is a *hosting* service that The Graph provides to deploy the created subgraph. This service is centralized and is meant for testing the subgraph during development, before deploying it to the decentralized production version.



Figure 3: Diagram of the entities of the defined GraphQL schema and their relationships

7.4 Client

The following sections give information about the development of the client that allows the user to interact with the Gazzeth smart contracts deployed on the blockchain.

7.4.1 Implementation

The client is a *single page application*. For this reason, the user loads the page content only once.

It was developed with the React framework, using the TypeScript language. React allows defining components that can be reused and composed to build the views of the client, rendering and updating what is needed in an efficient

way. The Material-UI library was used as the base to create the components, and the ethers.js library to communicate with the smart contracts.

It is important to note that the client uses *clean architecture*. This architecture defines different layers and their interfaces, keeping their code decoupled, which makes it easy to change. On one side there are the repositories, which interact with external systems (such as an API or, in this case, the Gazzeth contracts on the blockchain). Then there are the use cases, which represent a user feature and include the business rules. Finally there is the view, which consists of the React components the user will interact with.

The client components support internationalization, which allows supporting different languages and makes it easier to add new ones in the future. At the moment, the client supports English, Spanish and Japanese.

Appendix B contains the user manual of the developed client.

7.4.2 News statuses

Each news item, once its validation is over, will have an associated voting status in the smart contract, from which its classification can be derived.

As explained before, the protocol is public and decoupled from any client, which allows each client to choose how to define the interface. To show this, the client implemented for the proof of concept defines its own validation statuses based on the results of the vote. The rules that determine each of them are listed below. They are evaluated in order for each news item, and the status of the first rule that matches is chosen:

- If the validation vote of the news has not ended, it is considered *Pending*.
- If no more than 65% of the selected jurors voted, it is considered *Insufficient votes*.
- If the *True* votes are more than 60%, its status is considered *True*.
- If the *False* votes are more than 60%, its status is considered *False*.
- If the *Unqualified* votes are more than 60%, its status is considered *Does not meet the rules*.
- In any other case, its status is considered *No consensus*.

7.4.3 Deployment

We used GitHub Pages to deploy the client because it is simple and free. In addition, we configured a DNS record so that the URL `gazzeth.com` points to the deployed client.

8 Future work

The following are some points to keep working on:

- **Definition of quantities:** Throughout the document we have avoided defining quantities and amounts. For example, the amount needed to register as a juror or to publish, the amount to create an appeal, the minimum number of jurors to start publishing in a topic, the number of jurors to select for a validation and the duration of each phase of the vote. We consider that defining these values objectively, so that they align with the mission of the project, requires a study that goes beyond the scope of this work.
- **Cost reduction:** Appendix A shows the costs of interacting with the implemented proof of concept. These costs are high, which makes using Ethereum L1 unviable for this project. To solve this, the different Ethereum L2 solutions that reduce the transaction cost, such as Optimistic Rollups and ZK-Rollups, must be explored, and the code must also be optimized for the same purpose.
- **Experimentation on mainnet:** To verify whether the project actually achieves the expected result, it must be deployed on mainnet, and the behavior of the users and the results of the validations made on it must be studied.
- **Sybil-resistance with privacy:** It is important for Gazzeth to have a sybil-resistance mechanism that provides privacy. For this, we consider it important to work on Proof of Humanity so that it can provide this attribute, or to develop some alternative mechanism that meets the conditions this project needs.
- **Decentralized IPFS text search engine:** The rules for choosing the vote include that the news to publish must give information that is not, at that moment, in the topic where it is being published. Currently there are no tools to do this in a decentralized way. We propose working on a protocol that indexes IPFS files and exposes a text search engine to search over them. It could be considered an extension of The Graph and, therefore, be proposed as an improvement to it through its governance.
- **RNG:** Develop a random number generation module that is more efficient and provides more security.

9 Conclusion

Gazzeth manages to address the problems raised at the beginning of the document. It proposes the design of a news platform that is a public good, global and censorship-resistant, that does not need funding from any entity and that

allows anyone to publish news on it. The platform includes a native validation system, provides mechanisms to audit that validation process and to act if the result seems incorrect, involves the different users of the platform, and allows democratic mechanisms for its evolution.

This type of platform, which comes from the creation of decentralized, censorship-resistant, transparent, permissionless and global protocols, forms a new paradigm that is starting to be called *Web 3.0*.

It is important to focus on improving the usability of the tools that users use to interact with this type of platform, such as wallets, to lower the entry barriers. These barriers are currently relatively high and, in many cases, make the adoption of these technologies harder.

Looking to the future, there is nothing but optimism. Through its transparency and permissionless nature, the *Web 3.0* allows the composability and cooperation of different protocols, which, together with competition, produce a constant evolution of the ecosystem. This evolution turns into increasingly powerful social tools for users.

References

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

[2] C. Lesaege, F. Ast, and 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, and 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, and 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, and 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.

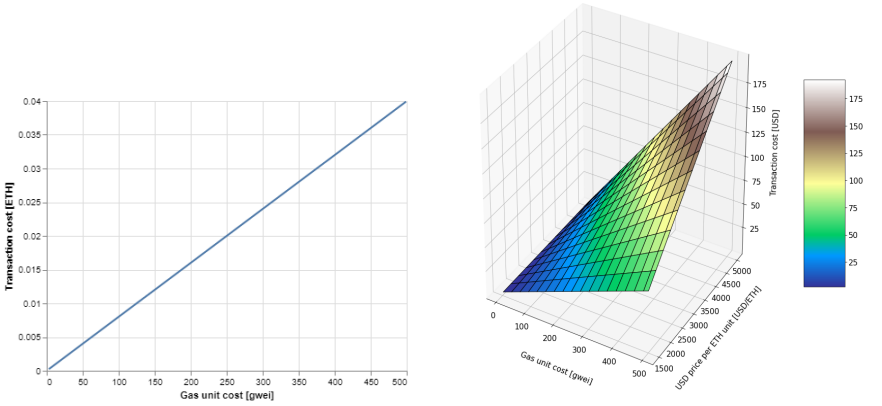
[17] R. Joshi H. Rajat. *Ethereum Push Notification Service (EPNS)*. URL: https://whitepaper.epns.io/.

[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.

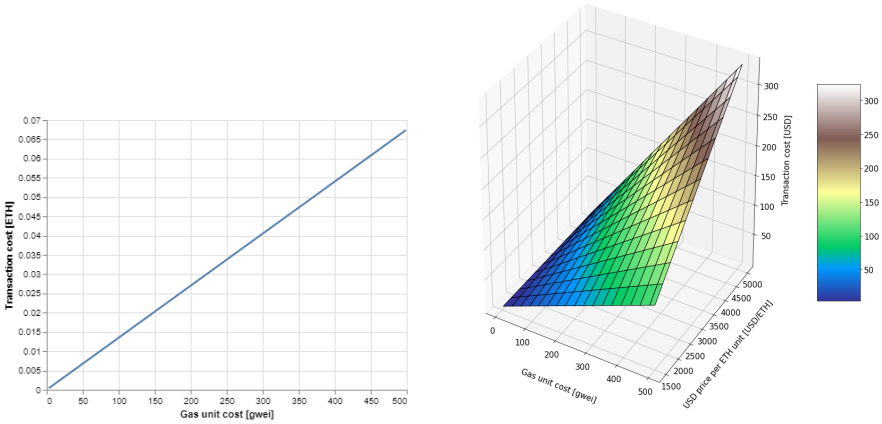
Appendices

A Transaction costs



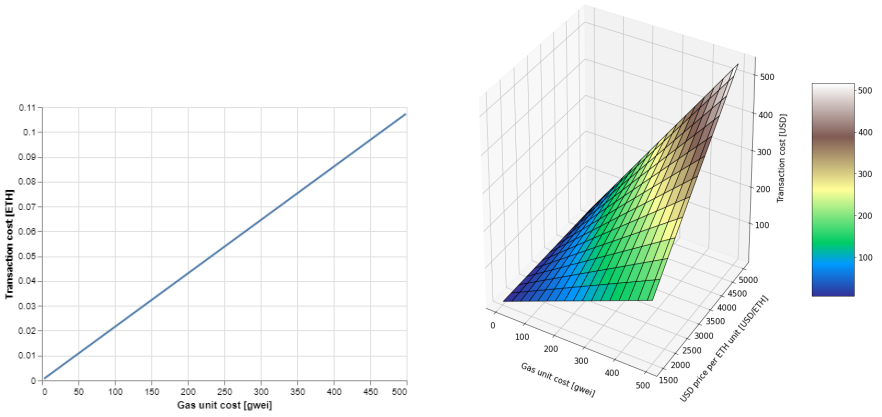
(a) Cost in ETH (b) Cost in USD

Figure 4: Cost of the vote *commit* transaction on Ethereum.



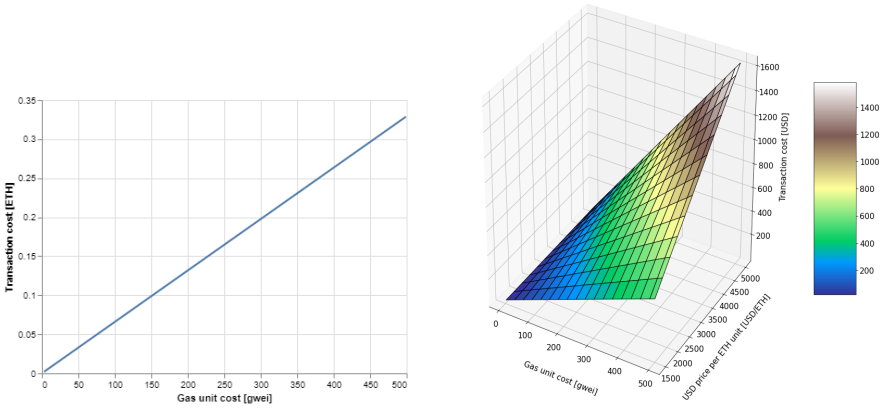
(a) Cost in ETH (b) Cost in USD

Figure 5: Cost of the vote *reveal* transaction on Ethereum.



(a) Cost in ETH (b) Cost in USD

Figure 6: Total cost of voting (*commit* + *reveal*) on Ethereum.



(a) Cost in ETH (b) Cost in USD

Figure 7: Cost of the news publication transaction on Ethereum.

B User manual

This appendix describes the different features that the client provides. The main page of the application is shown below:

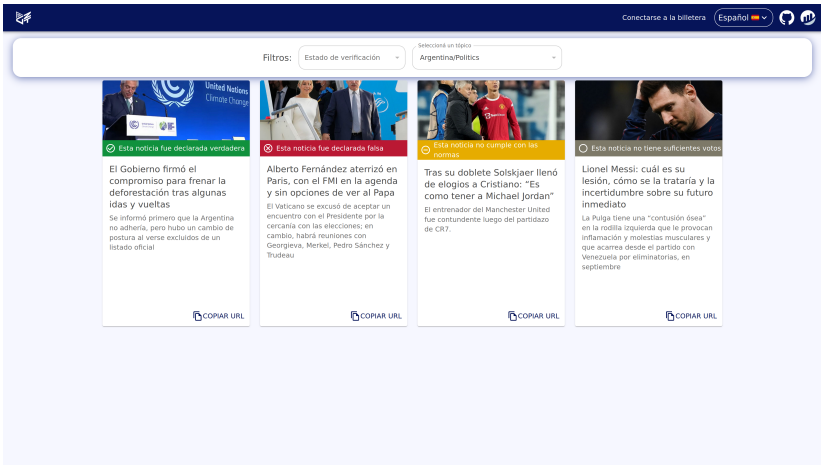


Figure 8: Main screen of the client.

B.1 Navigation bar

As can be seen in figure 8, the user can connect a wallet, change the language, and open the GitHub profile of the organization and the contract deployed on the blockchain, which is shown through the Etherscan explorer. These last two buttons provide transparency to the user, since they allow auditing the code they interact with.

B.2 News list

As can be seen in figure 8, the different news items are listed. They can be filtered by topic and verification status. Each news item shows an image, a title, a short summary and its status. Clicking on the card of a news item opens its details.

B.3 Connect a wallet

The navigation bar has a button to connect a wallet (figure 8). Clicking it shows a list of integrated wallets, in this case only Metamask. If the user does not have that wallet installed, a button will appear that takes them to its installation page. These screens are shown below:

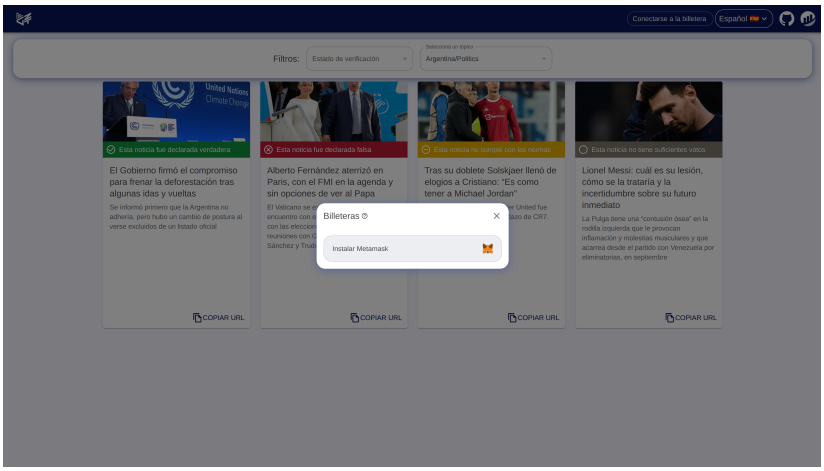


Figure 9: The wallet is not installed.

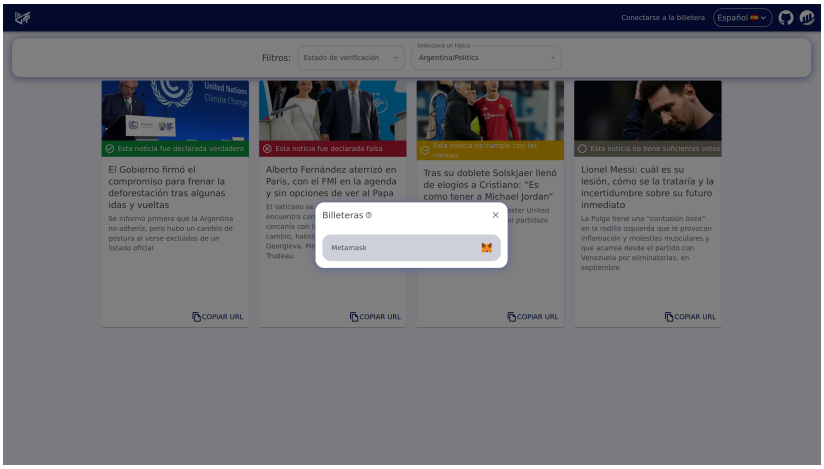


Figure 10: The wallet is installed.

Once the Metamask wallet is selected, a loading screen will be shown and, if the wallet is not currently unlocked, a wallet window will appear asking for its password (figure 11). Once the connection is accepted, the main screen will be shown again and more options will appear in the navigation bar (figure 12). Otherwise, an error will be shown.

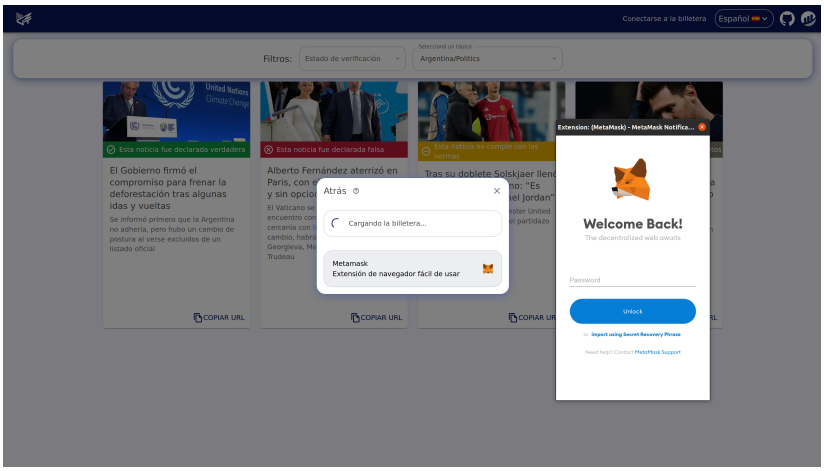


Figure 11: Connecting Metamask as the wallet.

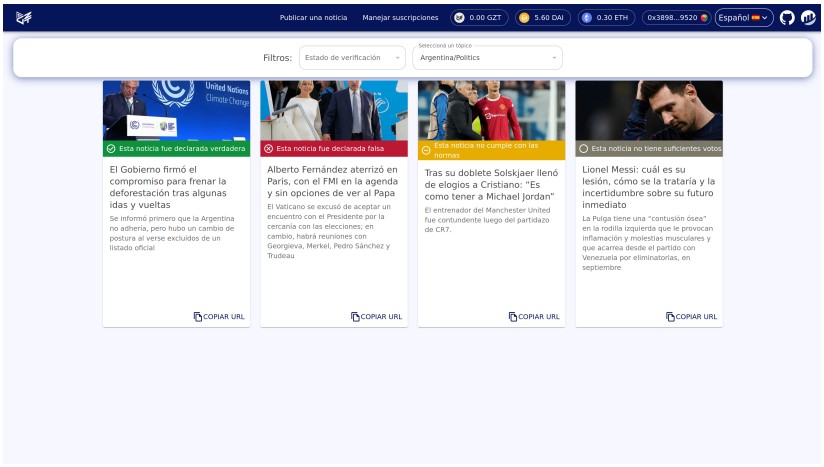


Figure 12: Main screen connected to the wallet.

B.4 Manage juror subscriptions

Clicking on "Manage subscriptions" in the navigation bar opens the page to subscribe or unsubscribe as a juror, where the user chooses the topics and the number of times to be a juror at the same time in each one. This page shows the amount of DAI that will be locked (negative if it is released) as a result of the transaction. It also shows the topics the user is subscribed to and the number in each one.

Pressing the subscription button opens a wallet window that asks to sign the transaction. A toast message shows whether the transaction succeeded or

failed. The screens of the process are shown below:

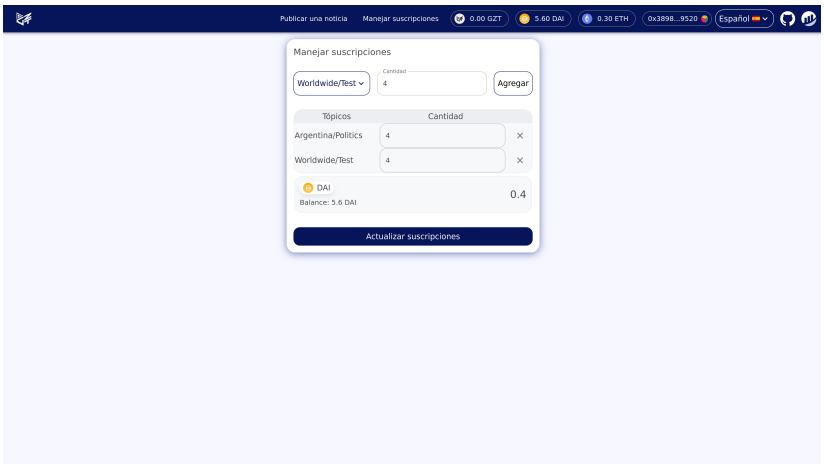


Figure 13: Form to subscribe or unsubscribe as a juror.

B.5 Publish a news item

The news publication page can be opened by clicking the "Publish news" button. On this page, the user chooses the topic to publish in and attaches a Markdown file with the content of the news. Note that, for the summary view of the news (the one in the list of the main screen) to look right, there is a specific format, which was already discussed in section 6.4.

There is also a button to show how the news would look in its detail view. Pressing the publish button opens a wallet window that asks to sign the publication transaction. The screens are shown below:

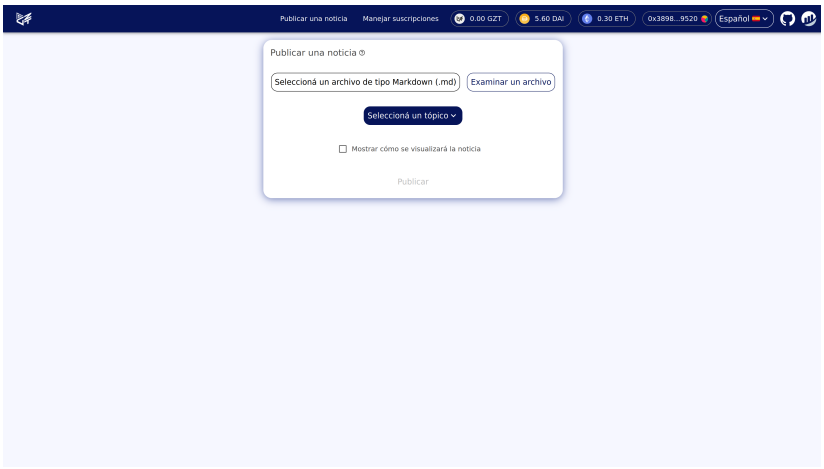


Figure 14: Form to upload a news item.

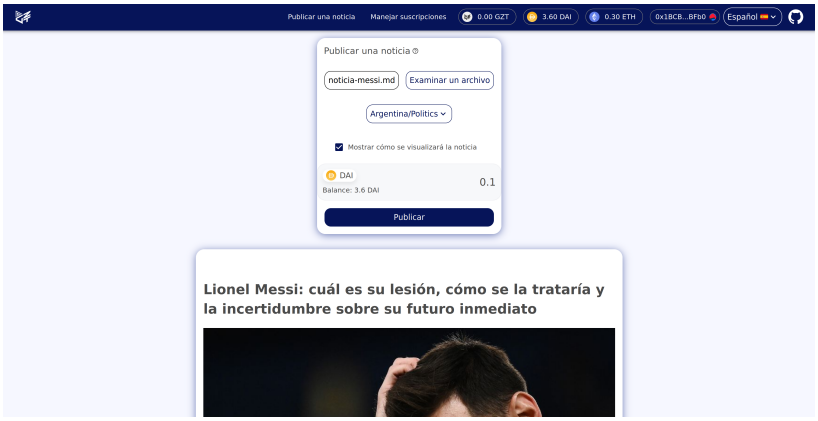


Figure 15: Form to upload a news item, with data.

B.6 View a news item

The details of a news item can be opened by clicking on it from the main page, or directly through a link to it. On that screen, the user can read the content of the news and the results of its validation vote (if the vote has ended). That screen is shown below:



Figure 16: News details.

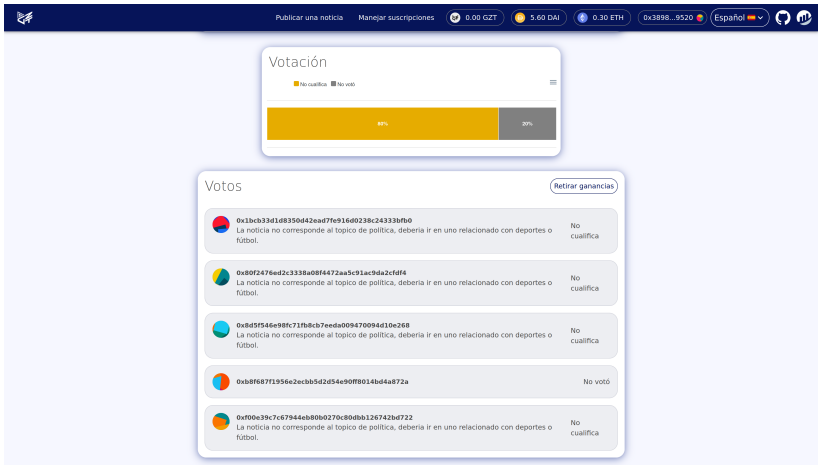


Figure 17: News vote.

B.7 Cast a vote

The user can see whether they were selected as a juror by clicking on their address in the navigation bar. This opens a window with information about the user's account, including the publications they are participating in. From this screen, the user can open the voting form, where they can cast the vote. When casting the vote, the user will be asked to sign the transaction with the wallet. These screens are shown below:



Figure 18: User profile.

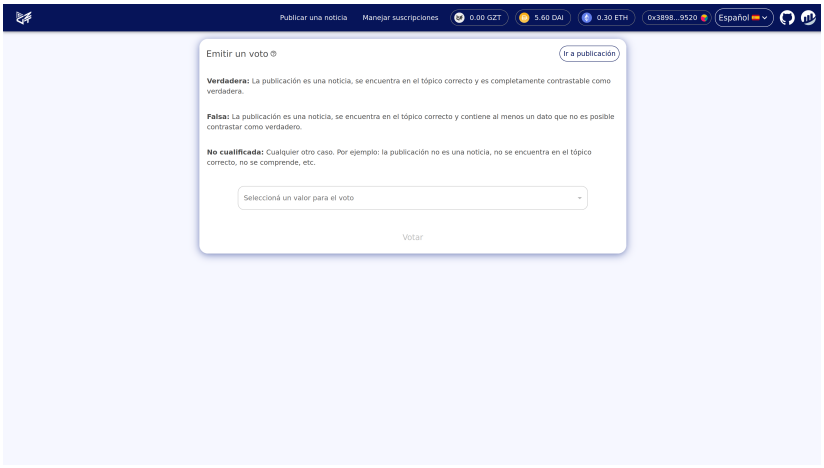


Figure 19: Voting form.

B.8 Reveal a vote

In the same way as for voting, the user can open the screen to reveal the vote (see figure 18). The user must enter the value of the vote cast before and a justification of why that option was chosen. Pressing the reveal button asks to sign the transaction with the wallet. The screen is shown below:

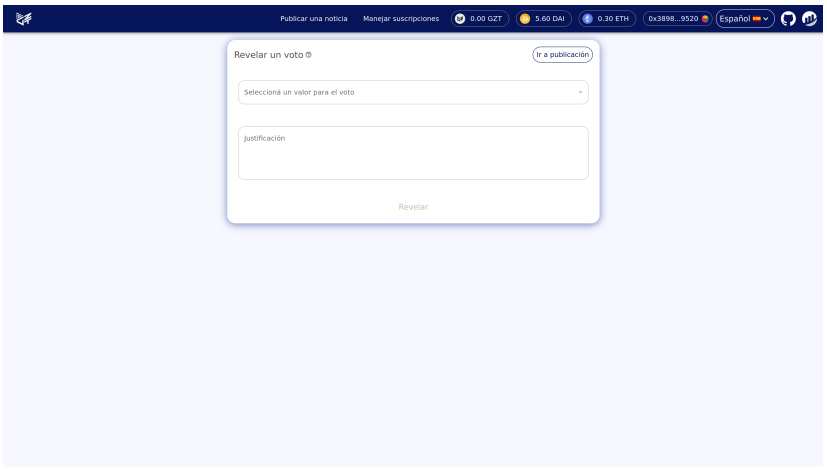


Figure 20: Form to reveal a vote.

B.9 Distribute earnings

Once the vote reveal phase is over, anyone can distribute the earnings. This is done from the news details screen, where a button appears in the vote details (see figure 21). Pressing the button asks to sign the transaction with the wallet. This operation executes the penalties and rewards.



Figure 21: Vote of a news item with the button to distribute earnings.