Electronic voting system using blockchain

by

Eduardo FURTADO SA CORREA
Supervised by Prof. Dr. Kaiwen ZHANG

PROJECT PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE WITH PROJECT
M.Eng.

MONTREAL, JULY 31, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

TABLE OF CONTENTS

Page

CHAPTER 1

INTRODUCTION

This document is a report for the project "Voting system using blockchain" realized as a synthesis project for the title of master in software engineering. The system is aimed toward being used by the Student Association of the École de technologie supérieure (AÉÉTS).

This report goes through the research and development process, including details about the analysis, the design, the implementation, the testing, the deployment, the maintenance of the system and also the results and costs.

The developed system was deployed to three different blockchain platforms, and results were collected by running many simulations with different settings with a simulator module.

The results when using the Matic blockchain is that such blockchain is ideal to host such system and that the system could be adopted by the AÉÉTS if a graphical user interface is also developed.

This report is organized as follows:

- Chapter 1 has a brief description of the project;
- Chapter 2 exposes the objectives of the project;
- Chapter 3 describes the context of the project, discussing the concepts of voting and Blockchain;
- Chapter 4 describes the system by listing the functional requirements, the use cases, the properties, the limitations, the hypotheses, the dependencies and the non-functional requirements;
- Chapter 5 shows the system design and how it was implemented and deployed;
- Chapter 6 shows the results obtained by simulating elections and the cost analysis;
- Chapter 8 exposes how the project was realized and what were the challenges overcome and lessons learned;
- Chapter 9 exposes the conclusions of the project.

A video-demo of the project is available at: https://youtu.be/NORpjwKhPkc

CHAPTER 2

OBJECTIVES

2.1 Main objective

The main objective of this project is to deliver an electronic voting system that uses the blockchain technology and that can be used by the University for voting on things like the election of officers of the Student Association of the École de Technologie Supérieure (AÉÉTS) or in referendums.

2.2 Secondary objective

The secondary objective is to turn the theory seen during the master studies in software engineering into practice, by delivering a complete software engineering project, while solidifying and amplifying the author's knowledge and skills related to blockchain, distributed systems, infrastructures and automation.

2.3 Other objectives

Another objective of the project is to have everything open-source and available for anyone that wants to fork it or continue development. In fact, after delivery of the project, the author intends to publish the project in hopes of finding someone to develop the missing modules of the system, which is the graphical user interface and a voting obfuscating mechanism.

CHAPTER 3

CONTEXT

When groups of humans need to make collective decisions, they often vote for it. According to Behlendorf (2018), there are three phases of an election, as it can be seen in figure 3.1: what happens before voting, during the actual voting and then after the voting is done.

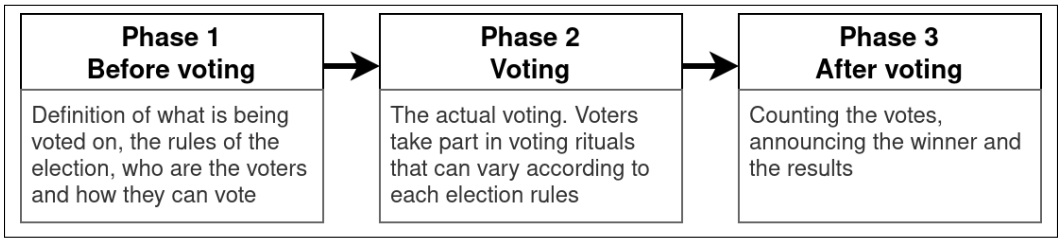

Figure 3.1 The three phases of an election

Voting can happen in different ways, for example on a small organization with a few dozen people could require unanimity to make a decision, whereas countries with hundreds of thousands of citizens could hold democratic elections based only on the count of votes or a medium-sized organization could require the majority of votes to make a decision.

In some cases people vote by simply raising their hands, or vocalizing their choices. In some other cases, people go to specific places and write down in pieces of paper what they want and then someone else looks at all the papers to count the votes.

Normally, elections with secret ballots need:

1. Enforced secrecy: each voter casts their ballot secretly and there is no way to know how they voted.
2. Individually verifiable: each voter must gain trust that their vote counted correctly.
3. Globally verifiable: all voters must gain trust that all votes counted correctly and all votes are from those that had the right to vote.

Voting is mandatory in some places and optional in others. Even when it's mandatory, people can still not vote if they justify their absence. In many cases people don't vote because of logistical issues, such as the election centre being too far or having to wait in line for their turn Yavuz, Koç, Çabuk & Dalkılıç (2018). Another common reason is that people might feel that their vote doesn't count because of the lack of transparency in the voting system implemented.

There are many successful examples of online polls and questionnaires, but not for online elections for governments, mainly because official elections is an essential element of democracy, and they are required to be robust and provide transparency and anonymity.

Currently, conducting elections is expensive because of all the logistical aspects and manual work involved in counting the votes.

Paper based voting is very inefficient to count votes because it's a manual process that doesn't scale. Likewise, paper voting has the advantage that tampering the votes also needs to be done manually, which also do not scale well. This doesn't mean that tampering can't happen in large scales, if the results of the counting of votes is manipulated, instead of the votes.

Other problems of paper-based voting include

- long lines that sometimes are faced by voters;
- a long, error-prone and expensive vote counting process;
- the results are not capable of being audited;
- expensive because many people have to work to make it happen;
- often voters can't vote because they can't physically attend a voting venue.

However, paper-based voting is still the norm. Various electronic voting solutions have been proposed but none has been widely adopted, because there are still open problems with it, as exposed in section 3.5.

3.1 Justification

Voting is a fundamental element of democracy. It is a simple concept that is easy to understand and take part, but it can become very complex when deployed in the real world and logistical (scaling), ethical and trust aspects are considered.

There are two key parts of voting, which are anonymity of the voters and trust in the systemScott (2014)Scott & Elliott (2019).

Anonymity refers to the identity of voters, it should not be disclosed. This guarantees that nobody can be bribed or threatened for their vote Scott & Elliott (2019). Trust in the system refers to fact that the system needs to be trusted by everyone using it and that every vote counts Scott & Elliott (2019). Transparency helps by making it so that everyone should be able to check the results, so people can trust the numbers and not someone. Robustness helps by making it so that that fraud can't happen, the system behaves well even if you give it input that is not meant to be given to it. An open-source solution that can be audited by everyone also helps. Finally, to trust the system users must also easily understand how it works Scott & Elliott (2019).

Currently the most common implementation of voting is using paper-based systems, that required physical presence of the voters, are hard to scale and need a great deal of trust in a system that is governed by humans and as such is prone to human error.

Blockchains are capable of eliminating central entities in systems, and for the case of voting, it would mean that trust would be deposited in a transparent software system, and not on the people running an election. Electronic voting using the blockchain would also allow for easy access to voters while having the potential of being easily scalable.

3.2 Motivation

With democratic elections there are always questions open related to these requirements:

How honest are the people responsible for the voting system? How can people trust the system?

How can we be sure that nobody is manipulating the results by changing votes before they are counted?

How can we verify the transparency of the system while protecting the identity of voters?

How can we assure that a voter has the right to vote?

And there is also a problem with scaleability of the voting systems that can be very costly. So having 10 voters can be easy, cheap and fast to count votes, but what if we're talking about millions of voters?

So, an improved voting solution should directly tackle these questions to improve on the currently deployed systems.

Currently, in general, elections:

1. They lack transparency
2. They are expensive to run in large scales
3. It's hard to ask for votes, in the sense that there can't be an election every now and then
4. In most cases, physical presence is required to vote
5. Voters have to trust a central authority to make sure they do the right thing
6. No one can be 100% sure that no fraud is committed

Even though e-voting systems are being studied intensively, very few implementations are reliable and in use today.

So a secure electronic voting system could greatly benefit society.

3.3 Anticipated Benefits and Industrial Impact

The core values of this project is openness, transparency, simplicity and security, which are intended to establish trust in the electronic voting system by making it completely easily auditable by anyone interested, as well as allowing contributions by anyone.

Therefore, it has the potential to impact the life of the students and employees of the university that are involved in the political matters of the school.

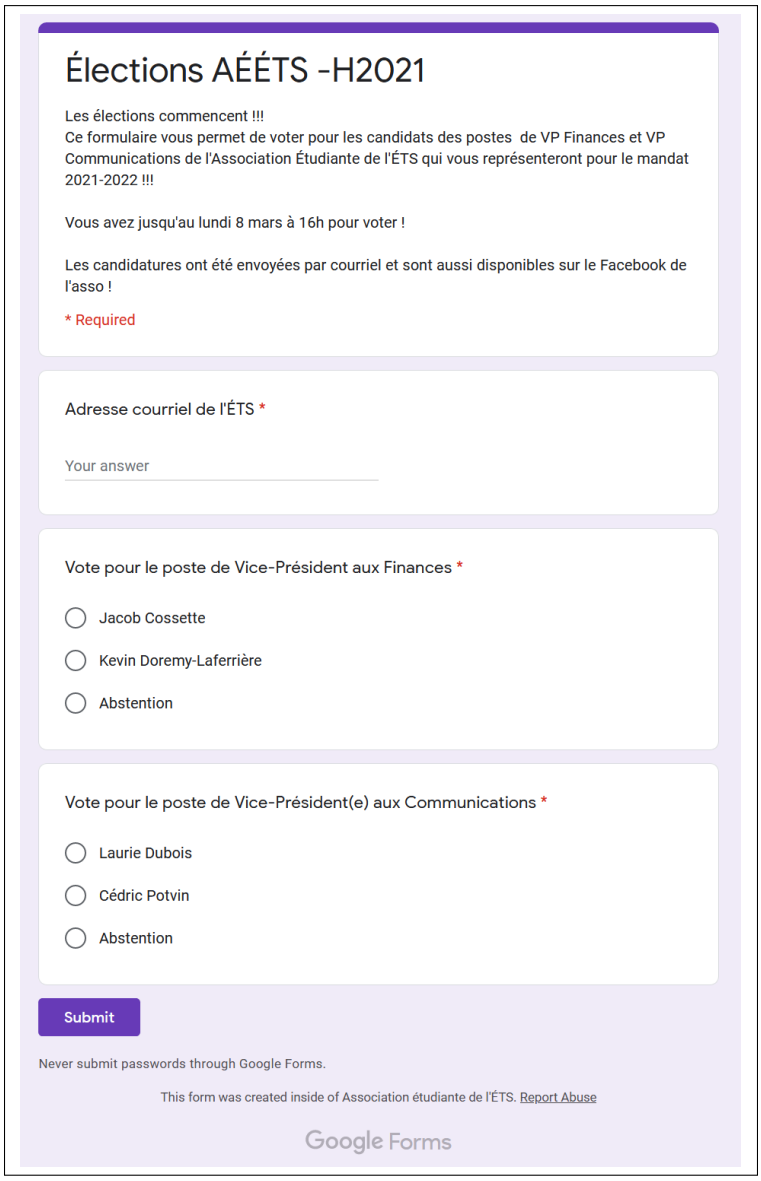

Figure 3.2 Currently the AÉÉTS are using Google forms

Currently, the elections from the AÉÉTS are being run using google forms as can be seen in Figure 3.2.

This project aims to make use of the state of the art of research done in the subject to deliver an open source solution that can be ported to other contexts outside of voting for the school's elections. Therefore, the project has the potential to improve how voting is done in many different contexts in society. This would add value to society, positively impacting people's lives by allowing them easier, more reliable and more secure access to democracy.

According to Brock (2017), humans are more and more comfortable with using their smartphones, so it makes sense to develop the app with mobile in mind.

According to the book by Patrick (2016): "It will take time. People have to be comfortable and confident in the system they use. Candidates will also need time to adopt a new way of thinking. Every election has multiple candidates, each of whom are sure they are going to win. When the election is over, the losers will be quick to challenge the voting system as the reason for their loss. Ultimately, voters and candidates will realize the blockchain will lead to fair and honest elections because the power of the blockchain ledger will ensure votes are counted correctly and cannot be altered. The result is an honest and fair election".

Therefore, small steps are a good way to make the blockchain technology reach official country elections.

This project is a step towards that goal. By launching an electronic voting system in the small scale of the context of a university, which counts as a test of how this kind of system would perform in the real world and also increases the adoption of a system built using blockchain technology, because small iterations on the evolution of these systems are fundamental to raise trust in the system.

3.4 Key concepts of Blockchain based voting

3.4.1 Blockchain

Blockchain is the technology behind the famous cryptocurrency Bitcoin, but the technology is capable of more than just cryptocurrencies (Korth, Sudarshan & Abraham Silberschatz (2010)). It can be used to run code, store data and it has interesting features: decentralized control, immutability and transfer of digital assets.

This makes this technology an ideal candidate for data storage for an electronic voting system.

The Ethereum platform is an example of a blockchain implementation other than Bitcoin, which can be seen as a global distributed computer capable of executing code from smart contracts deployed by anyone.

These smart contracts are what we call the code of an autonomous voting system that can run elections or referendums without the need of a central authority to decide which proposal wins the voting.

Fundamentally, a blockchain provides an alternative data format for storing data, and it can be seen as a database, coupled with a paradigm for transaction processing that enables a high level of decentralization(Korth *et al.* (2010)).

To understand how a blockchain can be used a database, one should understand how the blockchain technology works on the surface.

They are made up of "blocks," which contain transaction data and other related information, and these blocks are chained together in a data structure.

This means that every blockchain platform must have a mechanism that decides which blocks to add the chain and how to resolve potentially conflicting information, because it is a distributed system in which many, if not all, nodes validate the transactions.

In other words, blockchains use consensus to settle upon and finalize data that is replicated across all nodes.

So, to explain blockchain differently, you can think of blockchains as if they were ledgers holding records of a company's financial transactions. This ledger can be a physical notebook where the company writes their income and expenses.

Just like each page in a ledger holds a certain number of transactions, so do blocks in a blockchain. Also, blockchains are known for being secure because the actual chain of blocks, or ledger holding the transactions is not only distributed across many nodes, but it's also decentralized, meaning that there are no nodes that are more important than others. So blockchains are often referred to as distributed ledgers.

Since each node has its own copy of the ledger, it's very hard for someone to change a transaction without all the other nodes noticing.

A transaction can be anything from financial data to actual code that nodes can execute.

Once a node receives a transaction, it validates it, adds it to a block and then distributes it to all the other nodes who then accept or refuse the block. If the block is consensually accepted, it is added to the chain.

It's possible to see a blockchain as a database through the analogy that if a database stores data in a base, blockchain stores data in a chain.

3.4.2 Consensus

Blockchains are essentially distributed databases and the nodes of the network must be able to reach an agreement on the current state of the system. This is done by the use of consensus mechanisms, also known as consensus protocols or consensus algorithms (Sam Richards,Ryan Cordell (2020)).

Since the whole blockchain can be replicated across all participating nodes, each time a new block is added, all nodes must eventually agree on which node can propose a new block and also agree on the block (Korth *et al.* (2010)).

Consensus mechanisms are designed to make systems secure by preventing certain kinds of attacks that would allow an attacker to take control of the system and, for example, decide what blocks to add to the blockchain (Sam Richards (2020)).

There are many different consensus mechanisms with different properties.

3.4.2.1 Proof-of-work

Most notably used by Bitcoin and other cryptocurrencies, in which participants of the network compete to create new blocks filled with processed transactions. The winner is the one who is the fastest to solve a mathematical puzzle (the work done in proof of work) and then they share the new block with the rest of the network (Korth *et al.* (2010))(Sam Richards (2020)).

To participate in the competition, one must invest in hardware that needs electricity to run. They receive rewards for completing the puzzles, which is the incentive for people to take part in this competition (Korth *et al.* (2010))(Sam Richards (2020)).

The system is designed in such a way that it would not be economically viable for an attacker to take control of more than 51% of the network to control it, but at the same time, it creates a cost for everyone using the system (Korth *et al.* (2010))(Sam Richards (2020)).

Moreover, transfers have a cost for the users of the system, and their effect is subject to the latency of the network: the time it takes for the puzzle to be solved and the block propagated across the entire network (Korth *et al.* (2010)). There are ethical concerns about the usage of this consensus mechanism, as it has a considerable carbon footprint on the planet, and even consumes more energy than a country like Chile (Stoll, Klaaßen & Gallersdörfer (2019)).

3.4.2.2 Proof-of-Stake

Like Proof-of-work, Proof-of-stake is also designed to make it so that an attacker would need to control 51% of the system, but the way it works is completely different. A validator is chosen randomly to create new blocks, share them with the network and earn the rewards. Plus, instead of needing to do intense computational work to solve a puzzle, a validator only need to stake cryptocurrency on the network that they lose if they do not act honestly (Sam Richards (2020)).

3.4.3 Immutability

Immutable simply means unchangeable. It refers to the fact that once data is added to a block of a blockchain is extremely hard to be changed, nearly impossible. And with every new block added, the difficulty increases greatly because every block has a reference to the previous blocks, so to tamper with a new block means having to tamper with all the blocks that came after it, something that requires an enormous computational power and would probably cost more than the gains of tampering with such systems (Jacobsen, Sadoghi, Tabatabaei, Vitenberg & Zhang (2018))(Miles (2017)).

3.4.4 Decentralization

"In a public blockchain, control of the blockchain is by majority consensus with no central controlling authority. In a permissioned blockchain, the degree of central control is limited, typically only to access authorization and identity management. All other actions happen in a decentralized manner." (Korth *et al.* (2010)). In other words, in a decentralized distributed system, there are no nodes that are more important than others. In a perfect decentralized system, all nodes have the same power.

It might be tempting to mix the concept of centralization and distribution, but they are different:

- A centralized and non-distributed database must be in a computer, which will be THE server where this database is deployed.

- A distributed but centralized database will have a larger server than others in the same network of computers where the database is deployed.
- Centralization can be reduced by making it possible to have more than one very important server.
- A completely decentralized network is so that each node is as important as any other node.

A completely decentralized distributed system like a blockchain is very secure, because the data in the chain is extremely hard to be hacked as each node in the database will resist any unauthorized change (DevTeam.Space (2020)).

3.4.5 Use cases for blockchain systems

Decentralized applications have the power to disrupt industries by providing autonomous systems that don't need a central authority governing the systems. It's the early days of this movement towards decentralized applications, so there are many possibilities of use cases: "BigchainDB is for developers and organizations looking for a queryable database with blockchain characteristics such as decentralization, immutability and the ability to treat anything stored in the database as an asset. Whether it's atoms, bits or bytes of value, any real-world blockchain application needs performance." (BigchainDB (2020)).

The following list is a quick overview of some of the possible use cases for blockchains:

- Intellectual Property – attribution of work and authorship are easy to do because of the immutability and public nature of blockchains like Ethereum. Music authors, painters of painting or art in general can use this feature, academic certificates, transcripts and others (Korth *et al.* (2010)).
- Asset management – Just like with intellectual property, ownership records can be tracked using a blockchain enabling verifiable access to ownership records and signatures. The real-state industry, for example, can get huge benefits from this (Korth *et al.* (2010)).
- Supply chain – linking products to suppliers and all steps in the middle is easy to be done because of the transparency and trust involved in blockchain systems (Korth *et al.* (2010)).

- Identity – Personal data, medical records and credentials can be stored in blockchains (Korth *et al.* (2010)).
- Government – The automation aspect of blockchain can move trust from people to a system that doesn't require humans to function (Korth *et al.* (2010)). Actually, blockchain systems can be applied in any area of governments: "identity services, inland revenue, company formation and land registration. Having transparency in government services offers the promise of reduced corruption, stronger land and property rights, efficient taxation, and sovereign personal identity" (BigchainDB (2020)).
- Currency exchange – Transferring money over borders without having to pay taxes to a central controlling authority that doesn't do much other than make a profit on transactions (Korth *et al.* (2010)).
- Voting – Most notably for this project, many if not all of the current problems with traditional voting seem to be able to be dealt with by the usage of blockchain technologies: openness, transparency, simplicity and security, which are intended to establish trust in the electronic voting system by making it completely easily auditable by anyone interested, as well as allowing contributions by anyone.

3.5 Problems with electronic voting

The motivation to tamper with elections can be immense. If we're talking about official elections of a country, billions of dollars could be involved Scott (2014).

Therefore, every possible problem with electronic voting should be identified and studied extensively so they can be solved in order to make them as secure or more than traditional non-electronic election systems.

3.5.1 De-Ritualization

Going to a place to vote and, once at the place, following security procedures to be able to vote can be seen as a ritual that creates the experience of voting and make it be something serious

for the voter. This aspect can be lost with the ease of being able to vote using a mobile phone device, for example.

To counter this problem, the authentication process on the secure electronic voting system could be ritualistic:

- Making the whole process take some time and require preparation
- Taking pictures of documents and of the voter
- Asking the voter to repeat certain phrases

3.5.2 Changing votes can be easy to scale

This is by far the biggest problem with electronic voting when compared to regular voting. Many of the other problems of electronic voting are also shared problems with paper voting, but with some differences. For example, the hardware used to vote electronically could be a smartphone, a device that can be a target from attackers, just like a pen used in paper-based voting could also be a target of attackers. In traditional paper voting, changing votes is hard to scale, because an attacker would need to edit many pieces of papers to change a vote, whereas with electronic voting it could be possible to change many votes at once with one action Scott (2014).

To counter this problem, a cryptographic algorithm such as RSA could be used to make each vote depend on private-key and public-key encryption. This would make it harder to scale votes by not allowing many votes to be changed with one single action, but even if many actions are required, they could still be automated.

To solve the automation problem a possibility is to have electronic voting working together with a paper voting system, using the latter as validation.

3.5.3 Software vulnerabilities

Vulnerabilities in the software could be exploited by attackers.

To counter this, all software used should be open source so it's easy to audit, and the treatment of possible vulnerabilities should be public.

Also, extensive testing and validation should be done on the software.

3.5.4 Software distribution

The distribution of the software can be exploited by attackers that distribute fake versions of the software.

Voters could be instructed on how to manually check that they have a legit copy of the software on their hardware. One possibility is to have a checksum on the blockchain that can be used to validate against a local checksum of the installed software.

Manual effort for this is necessary, as to not require trust in the checking software.

The platforms in which the voting software run can also be a problem. For this, no proprietary software should be involved.

3.5.5 Hardware vulnerabilities

Vulnerabilities in the software could be exploited by attackers.

Ideally, open hardware should be required to use such a system, but this is far from being a reality nowadays. Therefore, hardware validation could be a possibility, that is, only allowing certain validated hardware to participate in the system.

3.5.6 Hardware requirements

Different kinds of hardware could be used by such a system

To counter this, the system should be able to run in most popular hardware.

The goal of a secure electronic voting system is not to make voting convenient so people can vote from anywhere, but it is to make it more secure and robust than traditional paper voting, while the convenience is somewhat of a welcome side effect. However it's not mandatory, and as so, specialized standard hardware could be a requirement, instead of using hardware like most smartphones.

3.6 Other efforts towards secure electronic voting

This section is meant to provide a broad array of different implementations of electronic voting systems, as well as publications and popular sources of information about electronic voting.

3.6.1 US secure electronic voting patent

The United States Postal Service (USPS) filled a patent Goswami, Dhananjay, Lagneaux, M., Venkataraman, Swaminathan, Henry, Wendy, Shrestha, Aashish, Dearing & M. (2020) that contains 47 pages, mostly with diagrams, showing the components of a proposed mail-in voting system that also has an electronic element to it.

The architecture can be seen in Figure 3.3.

The system is said to separate voter identity and the vote to ensure anonymity, assures that no fraud would occur.

3.6.2 Towards Secure E-Voting Using Ethereum Blockchain

The paper Yavuz *et al.* (2018) is part of a broader research related to e-voting systems.

The results shown by the paper are not a perfect solution, but they are promising and a step towards electronic voting.

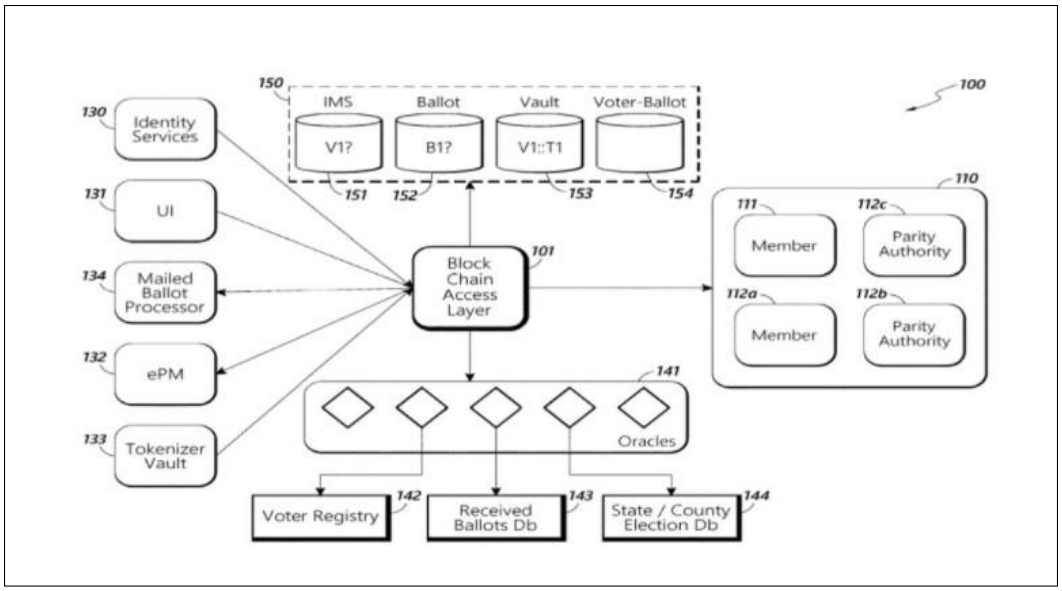

Figure 3.3 Architecture of the voting system patent filled by USPS. Source: Durden (2020).

The proposed implementation provided by the paper, which shows how they implemented a voting system with Ethereum, a decentralized open source blockchain featuring smart contract functionality.

The authors designed the e-voting system to be reliable, so that the whole election process is transparent, votes are immutable, people authenticate using credentials and the platform has provability so that only real people can vote.

It's also designed to be distributed meaning that there is no need for a central authority.

The authors suggest that operations should happen in realtime. Smart contracts written in the solidity programming language are executed by the blockchain nodes every 15 seconds and the transactions containing the votes are added to blocks in exchange of ethers, the cryptocurrency of the Ethereum network.

3.6.3 Electronic urn used in Brazil

Brazil uses an electronic urn instead of paper for their elections.

It can be seen as an improvement over paper-based voting, because even if people still have to physically attend the voting venue, counting the votes can be done automatically, instead of manually counting papers one by one.

However, one can argue that it's even easier to manipulate the results of elections because it still requires trust in the administrators of the system and people can also attack the electronic urns, as demonstrated by Professor Pedro Rezende (Santana & Rezende (2002), Rezende (2014)).

3.6.4 Electronic voting in Estonia

"Estonia is the only country in the world that relies on Internet voting in a significant way for legally binding national elections" Springall, Finkenauer, Durumeric, Kitcat, Hursti, MacAlpine & Halderman (2014a).

In Estonia there is a very robust and reliable centralized e-voting system, where voting is done online Yavuz *et al.* (2018).

Citizens are provided card readers that they use to read their IDs to authenticate with the system and vote.

It puts citizens close the decisions made by their government, by allowing them to digitally create petitions and proposals for acts and laws at the parliament's website.

But it's not a perfect system. Since it's centralized, it creates a single point of failure, that can be the target of distributed denial of service attacks, administrators of the system still have power to act maliciously. However, the system has been tested by independent researchers that uncovered the architectural limitations and vulnerabilities of the system, resulting in important lessons for

Estonia and any other entity looking to adopt an electronic voting system Springall, Finkenauer, Durumeric, Kitcat, Hursti, MacAlpine & Halderman (2014b).

It's also not scalable, because of the need for the card readers. The system has seen a weak penetration of only 30% among the population that could vote and Estonia has a relatively small population of 1.3 million people, which is 1000 times smaller than the population of China.

3.6.5 Alaska about to use blockchain in voting statewide

According to the news article by Major (2021), Alaska is the first US state on a course to use blockchain in voting statewide: there is a proposal (Legislature (2021)) for it under Senate Bill 39 by Wasilla Republican Senator Mike Shower.

CHAPTER 4

REQUIREMENTS

4.1 Functional requirements

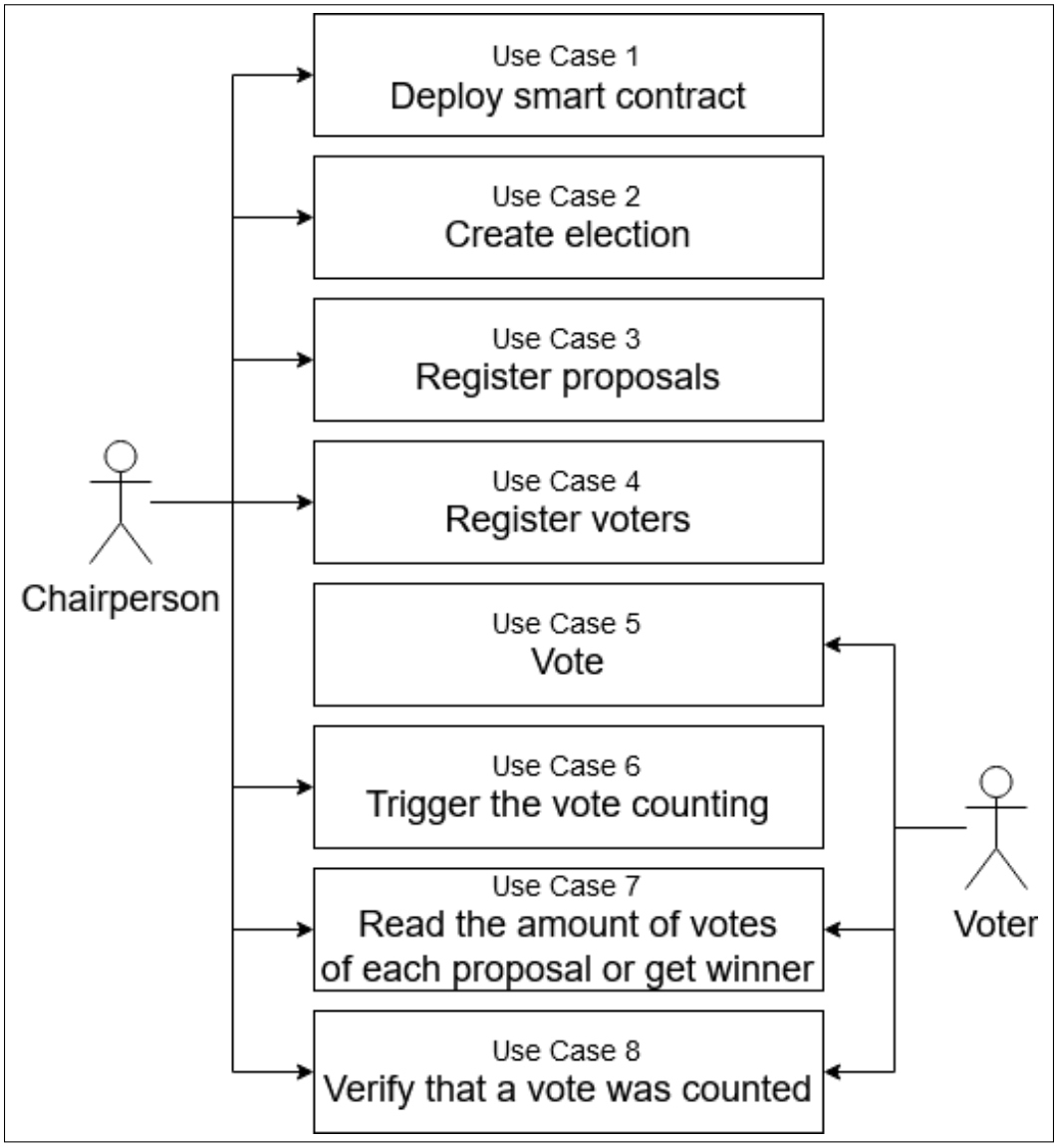

Figure 4.1 Diagram with the use cases of the system

The functionalities are resumed in figure 4.1. The functionality of the system as a whole consists in to provide a voting system that can be used by the Student Association (AÉÉTS) to run elections or referendums, while allowing the public to verify the results.

4.1.1 [FR01] Functional Requirement 1

The main functionality of the system is to allow the users to start and participate in an election or referendum. To do this, a chairperson must start an election, which means it's the person that will pay for the costs of using the system deployed in a blockchain platform.

Elections, proposals and voters are registered in the system and then voting can happen while the period to do so is active.

After elections are run, the chairperson can initiate the counting of votes, which is done totally by the system, but needs someone to pay for the cost of running the code in the blockchain platform where the system is deployed.

4.1.2 [FR02] Functional Requirement 2

A secondary functionality of the system is to allow the users to view and validate the results of an election or referendum, but not be able to tell which proposal received the vote of a voter. To do this, users can interact with the system to browse past elections, that presents the data in an objective way.

4.1.3 UC01 - Authenticate voter

The system should be able to verify the identity of a voter.

4.1.4 UC02 - Verify right to vote

The system should be able to verify that an authenticated voter has the right to vote.

4.1.5 UC03 - Authenticate chairperson

The system should be able to identify the identity of a chairperson.

4.1.6 UC04 - Initiate election

The system should enable a chairperson to define an election by setting the initial and end dates as well as defining who can vote and what is being elected by the voting.

4.1.7 UC05 - Add chairperson

The system should be able to identify the identity of a chairperson.

4.1.8 UC06 - Cast vote

The system should persist the vote of an authenticated voter that has the right to vote to cast a vote in an election they can vote according to the election constraints.

4.2 Use cases

4.2.1 Use Case 1: Deploy smart contract

Actors: Chairperson.

Level: Organizers of an election.

Stakeholders and interests: The chairperson wants to be able to use the system to run elections.

Preconditions: The chairperson must have a valid blockchain address and enough funds to run the deployment.

Guarantees in case of success: The smart contract is deployed to a target blockchain network.

Nominal case: The user uses the mobile or web app to do the deployment.

Alternative case: None

List of data and technology variants: Three blockchains are available for the deployment:

- Ethereum;
- Binance Smart Chain;
- Matic.

Frequency of occurrence: Rare. Can actually be done only once, unless there are updates to the code of the smart contracts.

4.2.2 Use Case 2: Create election

Actors: Chairperson.

Level: Organizers of an election.

Stakeholders and interests: The chairperson wants to be able to register a new election in the system.

Preconditions:

- The smart contract must be deployed (Use case 1);
- The chairperson must have a valid blockchain address and enough funds to execute the transactions with the blockchain.

Guarantees in case of success: An election is created in the system.

Nominal case: The user uses the mobile or web app to complete a form with the parameters of the election and create it.

Alternative case: A function of the smart contract is called using the simulator module.

List of data and technology variants:

- Name of the election;
- Description of the election;
- Start time of the election;
- End time of the election.

Frequency of occurrence: 1 to 4 times per year.

4.2.3 Use Case 3: Register proposals

Actors: Chairperson.

Level: Organizers of an election.

Stakeholders and interests: The chairperson wants to be able to register proposals to an existing election so that the registered proposal can receive votes.

Preconditions:

- An election must be created (Use case 2);
- The voting period of the election must not have started yet;
- The chairperson must have a valid blockchain address and enough funds to execute the transactions with the blockchain.

Guarantees in case of success: A proposal is registered in an election so that it can receive votes.

Nominal case: The user uses the mobile or web app to browse and select an election, allowing them to complete a form with the parameters of the proposal to add it to the selected election after the identity of the chairperson is validated.

Alternative case: A function of the smart contract is called using the simulator module.

List of data and technology variants:

- ID of the target election;
- Name of the proposal;

- Description of the proposal.

Frequency of occurrence: At least once per registered election. According to data from previous elections ran by the AÉÉTS, it's around 7 proposals per election.

4.2.4 Use Case 4: Register voters

Actors: Chairperson.

Level: Organizers of an election.

Stakeholders and interests: The chairperson wants to be able to register voters so they can vote in an election registered in the system.

Preconditions:

- A proposal must have been added to an election (Use case 3);
- The voting period of the election must not have started yet;
- The chairperson must know the wallet address of the voter being registered;
- The chairperson must have a valid blockchain address and enough funds to execute the transactions with the blockchain.

Guarantees in case of success: A voter is granted rights to vote in a target election.

Nominal case: The user uses the mobile or web app to browse and select an election, allowing them to complete a form with the parameters of the voter to give the voter the right to vote in the selected election after the identity of the chairperson is validated.

Alternative case: A function of the smart contract is called using the simulator module.

List of data and technology variants:

- ID of the target election;
- Wallet address of the voter.

Frequency of occurrence: At least once per registered election. According to data from previous elections ran by the AÉÉTS, it's around 350 voters per election.

4.2.5 Use Case 5: Vote

Actors: Voter.

Level: Electors of an election.

Stakeholders and interests: The voter wants to cast their vote to a target proposal in a target election.

Preconditions:

- The voter must have the received the right to vote in the given election (Use case 4);
- A proposal must have been added to an election (Use case 3);
- The voting period of the election must be active: started but not finished;
- The voter must have enough funds available in their wallet to pay for the transaction.

Guarantees in case of success: The voter's vote is cast to a target proposal in a target election.

Nominal case: The user uses the mobile or web app to browse and select an election, allowing them to browse and select a proposal to vote for it after the identity of the voter is validated.

Alternative case: A function of the smart contract is called using the simulator module.

List of data and technology variants:

- ID of the target election;
- ID of the target proposal.

Frequency of occurrence: At least once per registered election. According to data from previous elections ran by the AÉÉTS, it's around 350 voters per election.

4.2.6 Use Case 6: Register voters

Actors: Chairperson.

Level: Organizers of an election.

Stakeholders and interests: The chairperson wants to be able to start the voting counting of an election so the winner can be determined.

Preconditions:

- The voting period of the election must be finished;
- Votes must have been cast to the proposals of the election (Use case 5);
- The chairperson must have a valid blockchain address and enough funds to execute the transactions with the blockchain.

Guarantees in case of success: A voter is granted rights to vote in a target election.

Nominal case: The user uses the mobile or web app to browse and select a finished election, allowing them to start the execution of the vote-counting function of the smart contract, after the identity of the chairperson is validated.

Alternative case: A function of the smart contract is called using the simulator module.

List of data and technology variants:

- ID of the target election.

Frequency of occurrence: Once per registered election.

4.2.7 Use Case 7: Read the number of votes of each proposal or get winner

Actors: Voter or Chairperson.

Level: Electors and organizers of an election.

Stakeholders and interests: The Voter or the Chairperson wants to know who won or how many votes each proposal received.

Preconditions:

- The vote counting of the target election must have been finished (Use case 6);

Guarantees in case of success: The system responds with a message containing the counted votes of each proposal of a target election.

Nominal case: The user uses the mobile or web app to browse and select a finished election, allowing them to view how many votes each proposal received and which one was the winner.

Alternative case: A function of the smart contract is called using the simulator module.

List of data and technology variants:

- ID of the target election;

Frequency of occurrence: As many times as a voter wants to check the results from an election, which could be many times per day or simply once per election.

4.2.8 Use Case 8: Verify that a vote was counted

Actors: Voter or Chairperson.

Level: Electors and organizers of an election.

Stakeholders and interests: The Voter or the Chairperson wants to verify that the vote of a voter has been counted.

Preconditions:

- The vote counting of the target election must have been finished (Use case 6);

Guarantees in case of success: The system responds with "yes", "no" or a significant error message.

Nominal case: The user uses the mobile or web app to browse and select a finished election, allowing them to verify if the vote of a target voter has been counted.

Alternative case: A function of the smart contract is called using the simulator module.

List of data and technology variants:

- ID of the target election;
- Wallet address of a target voter.

Frequency of occurrence: As many times as a user wants to check if the vote that was cast by a voter has been counted in a finished election.

Note: It should not be possible to know which proposal received the vote of the voter.

4.3 Users of the system

As it can be seen in figure 4.1, there are two kinds of users in the system:

- Chairperson: The individual representing the organization of an election or referendum, which is responsible for paying for the costs of transactions in the blockchain. It's important to note that this actor doesn't have the authority to influence the counting of the votes, and its sole role in it all is deciding what is being voted on, who can vote and pay for the costs;
- Voter: Any person that can vote in an election or referendum. These can be the students or employees of the university.

4.4 Properties of the system

4.4.1 Public confidence

According to the European Parliamentary Research Service (Boucher (2018)), for any voting system, public confidence is crucial, and to achieve this the following is a must:

- The inner workings of the system have to be understandable by the users;
- Trustworthy, meaning that any election results should be seen as fair and valid.

Since there is a learning curve related to blockchain, these properties are harder to be attained in the large scales of governmental voting, but it's more feasible in smaller organizations, such as within the ÉTS community.

4.4.2 Anonymity

Nobody can be paid, bribed, threatened or forced to vote a certain way. To accomplish this, there should be no way of identifying who is the voter of a vote.

4.4.3 Trust

Ideally, nobody should have to be trusted for elections to run.

Humans can be bribed, threatened or be incompetent, so the system should not rely on humans Scott (2014).

Ideally, 0 people should be trusted. If this is not possible, many people are better than a few if a consensus mechanism is put into place, as to make such point of failure harder to be rigged.

The system should also be as decentralized as possible, so there is no concentration of power.

4.5 Limitations

The system must be available for free in a web browser and in the main mobile platforms (Android and iOS), so it's necessary to use technologies to develop in these platforms without introducing differences in the functionalities available.

The system will store personal user data, of which it must be encrypted to be stored.

4.6 Hypotheses and dependencies

The hypotheses assumed and the dependencies of this project are the following:

- The deployment of the system will be made on the Ethereum platform;
- The author is technically competent to implement all that is proposed;
- The Web, Android and iOS platforms are regularly updated. The blockchain platforms can also receive updates from time to time. Major changes between the different versions may occur and impact the performance and usability of the system. The voting system updates must allow it to remain functional by following the evolution of these platforms;
- A Minimum Viable Product is intended to be delivered, which means not all cases might be covered, such as a complete authentication process that includes different method of identification;
- It's assumed that it's possible to make every vote count without revealing the identity of who cast each of the votes, or which proposal received the vote of a voter;
- It's assumed that it's possible for the system to run an automated user validation process that verifies the identity of the users before they interact with the system to write data to the blockchain, for example to cast their vote;
- The whole project should be open source and the whole stack used to develop should also be open source.

4.7 Non-functional requirements

4.7.1 Usability

[NFR-U01] The application must be available in French and English.

[NFR-U02] All contents of the public parts of the application must be accessible with 3 clicks (keys) or less.

[NFR-U03] Forms must be validated by the application before submission and errors in fields must be highlighted in the field with a clear explanation of what is wrong and how to fix it.

[NFR-U04] The list of elections and referendums should be sorted by date, from most recent to oldest.

[NFR-U05] In the user's graphical interface, juxtaposed colors shall have a contrast ratio of at least 4.

[NFR-U06] A new user shall be able to use the system to perform the use cases without having to read any documentation.

[NFR-U07] A user must be able to see the details of any election registered in the system.

[NFR-U08] Before any operation that requires funds to run, the identity of the user must be verified.

4.7.2 Performance

[NFR-P01] The application shall be able to support 1000 concurrent users without performance degradation.

[NFR-P02] Any operation shall be completed in less than 10 seconds. A dialog box shall appear to show the status and progress of the process.

[NFR-P03] Each click/touch shall have visual feedback displayed within 0.1 seconds.

4.7.3 Databases

[NFR-DB01] The application must use a relational database system only when there is a clear justification that the data cannot or should not be stored in a blockchain.

[NFR-DB02] The database system solution and the blockchain (which can also be seen as a database) must be open source.

[NFR-DB03] All personal data of the users shall be encrypted. An exception is the user's email address, so that the user can be identified and also contacted by support. These facts should be displayed in the public sections of the application, as they can be considered a golden standard for user privacy.

[NFR-DB04] Intermediate results of calculations should not be kept in the database or blockchain, only the final results.

[NFR-DB05] Non-personal user data should never be deleted unless specifically requested by the user.

[NFR-DB06] The results of past elections or referendums should never be deleted.

4.7.4 Reliability

[NFR-R01] If the API cannot communicate with the blockchain or the database, after the timeout period, the user should be informed of the technical problems and asked to wait while the system automatically tries to connect again.

[NFR-R02] If the user interface cannot communicate with the API or the blockchain, after the timeout period, the user shall be informed of the technical problems and asked to wait while the system automatically tries to connect again.

4.7.5 Availability

[NFR-A01] The system must be operational at all times, which means that service-related problems must be communicated to the user in a transparent manner and the system must retry the connection automatically at regular intervals.

[NFR-A02] Updates should not disrupt system operation, so techniques such as blue/green deployment should be used.

[NFR-A03] The system shall strive to achieve 99.99% availability per year.

4.7.6 Documentation and help

[NFR-D01] The application must have a public section that explains the system, such as a white paper, and also help users to use the system. This section must be accessible even after the user has logged in.

[NFR-D02] The documentation for all the modules of the system must be publicly available in the respective repository, as it's an open-source project.

4.7.7 Security

[NFR-Sec01] The system must provide password-protected access to the application's functionality, i.e., everything except the public section of the application (the public section is like a white paper).

[NFR-Sec02] All data must be transmitted in encrypted form (e.g., with HTTPS).

[NFR-Sec03] User passwords must not be accessible by users, administrators, or even developers.

[NFR-Sec04] All personal user data that is stored by the system should also be encrypted.

[NFR-Sec05] The database should only be able to communicate (send, respond or receive requests) with the API.

[NFR-Sec06] The API must limit the number of requests, in order to protect itself from DoS and brute force attacks.

[NFR-Sec07] The API shall have a mechanism to defend against DDoS attacks.

[NFR-Sec08] The software must be resistant to cross-site scripting injection attacks.

[NFR-Sec09] The whole stack used by the system must be open source and ideally non-proprietary programming language.

[NFR-Sec10] Before each release, the software must be scanned by a tool that can report known vulnerabilities in the libraries used by the system. GitHub has such a tool, for example.

[NFR-Sec11] Before any operation that has a cost or that can affect the results of an election or referendum, the identity of the user must be verified.

4.7.8 Maintainability

[NFR-M01] When the code is analyzed by a static software quality analysis tool such as SonarQube, CodeScene or Understand, the maintainability score must be at least 75%.

4.7.9 Portability

[NFR-P01] The web and mobile application shall be written in a cross-platform programming language that allows for a single code base for Android and iOS, such as the Dart (Flutter) programming language.

[NFR-P02] The API and database shall be deployable on Unix systems or using a container solution such as Docker.

[NFR-P03] The mobile application must be compatible with Android Jelly Bean, v16, 4.1.x or later, and iOS 8 or later operating systems.

[NFR-P04] The mobile application must be compatible with mobile hardware on iOS devices (iPhone 4S or later) and Android ARM devices.

[NFR-P05] The user interface is supposed to be available for Web, Android and iOS. It should look consistent across all platforms.

4.7.10 Accessibility

[NFR-Acc01] The software shall be WCAG 2.0 Level AA compliant.

4.7.11 System monitoring

[NFR-SM01] The web or mobile application must be able to tell if there is a problem connecting to the API, or from the API to the database, or from the API to the blockchain, or from the application to the blockchain and inform the user of the problem while automatically trying connections.

4.7.12 Logging

[NFR-L01] All accesses (authentication attempts) to the software are logged. Upon successful login, the user shall be informed of previous failed login attempts.

4.7.13 Recoverability

[NF-R01] It shall be possible to recover the entire system from a backup copy that is less than one year old.

4.7.14 Interoperability

[NFR-I01] Dates shall be in accordance with ISO 8601.

[NFR-I02] The gender of a user shall conform to ISO 5218 with the addition of the gender "non-binary".

CHAPTER 5

DESIGN AND IMPLEMENTATION

5.1 Architectural view

The voting system is separated in modules. They can be seen on Figure 5.1 and the following sections goes into details about their design.

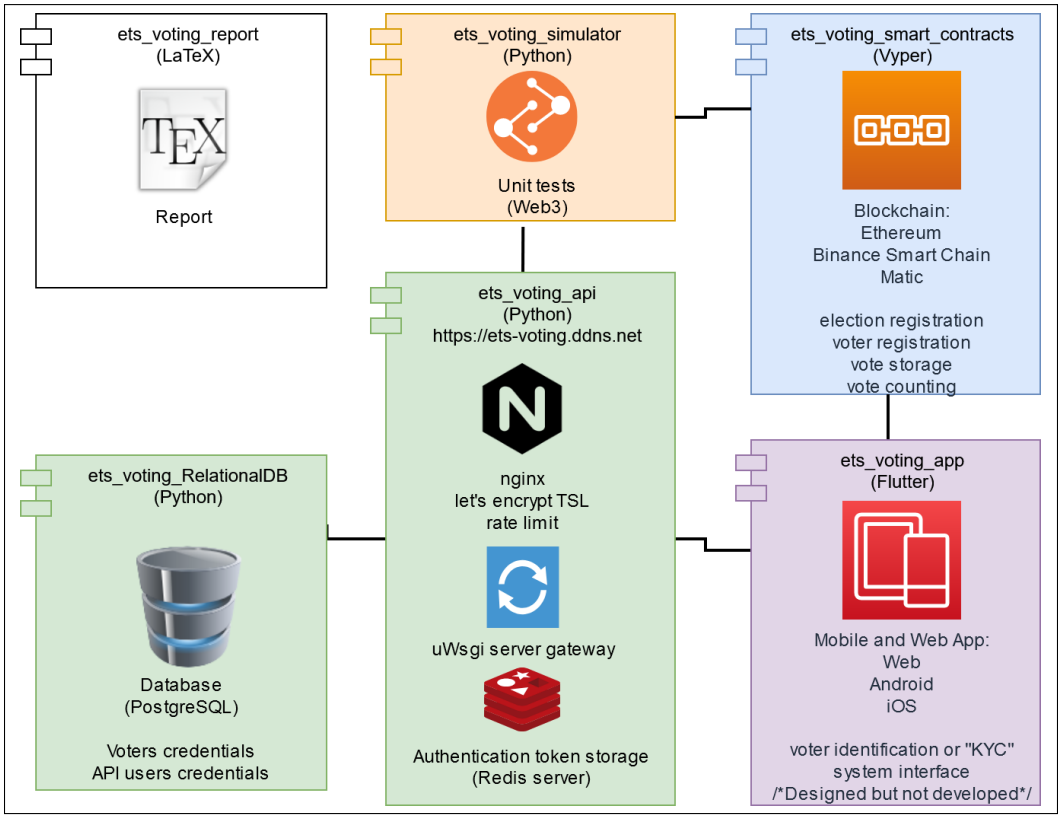

Figure 5.1 High-level view of the architecture showing the main modules

Each module has with their own open source repository which can be accessed by contributors wanting to improve the system or simply fork it to use it in another project. Also there is more documentation available such as deployment instructions.

- Smart contracts: https://github.com/eduardoxfurtado/ets_voting_smart_contracts
- Simulator: https://github.com/eduardoxfurtado/ets_voting_simulator

- API gateway: https://github.com/eduardoxfurtado/ets_voting_api
- Relational Database: https://github.com/eduardoxfurtado/ets_voting_relationaldb
- Mobile and Web app: https://github.com/eduardoxfurtado/ets_voting_app
- The report you are reading: https://github.com/eduardoxfurtado/ets_voting_report

The Restful API and relational database represent a centralized part of the system infrastructure. This centralized element is necessary because not everyone can participate in the elections, only those that are involved with the ÉTS.

Also, it's not a good idea to store private information on a blockchain, which justifies the API layer.

The voting app module was designed, but not developed, given that such development requires technical skills that are outside of the expertise of the author of this project.

There is a repository for the report, which is the document you are currently reading and that was written in LaTeX.

The smart contracts module contains the decentralized application that runs the elections.

The simulation module automatically tests the system by simulating an election and then displays the results and costs. Log files are also created during the execution.

An important design decision is that the voters do not need to worry about having a wallet with funds to be able to vote. Since the user identification is a centralized process, it was also decided that their private keys would be kept stored in the centralized infrastructure, which increases the convenience in exchange for lowering the security of the system.

It's convenient because they don't need to worry about funds to vote or about losing access to their accounts. The funds are sent by the system to the wallets of voters that are allowed to vote in an election.

5.2 Smart contracts module

The smart contract is written in Vyper, a pythonic programming language that is not Turing-complete, so it's less powerful than Solidity which is the most widely used language that compiles for the Ethereum Virtual Machine.

Vyper was chosen because it is a lot more readable, which goes in line with the transparency and audit-ability proposed by the system. More on this decision is presented in the Subsection 5.2.6.

Votes are stored directly in the blockchain, which is used as a database. This decision is further explained in subsection 5.2.3.

5.2.1 The smart contract

The figure 5.2 shows the files of the module, their functions and important variables.

The functions are presented in more detail below:

5.2.1.1 create_election [Use Case 2]

Result: Adds a new election

Parameters:

- String with the name of the election
- String with the description of the election
- Timestamp with the starting time of the election
- Timestamp with the ending time of the election

Constraints:

- Can only be called by the chairperson

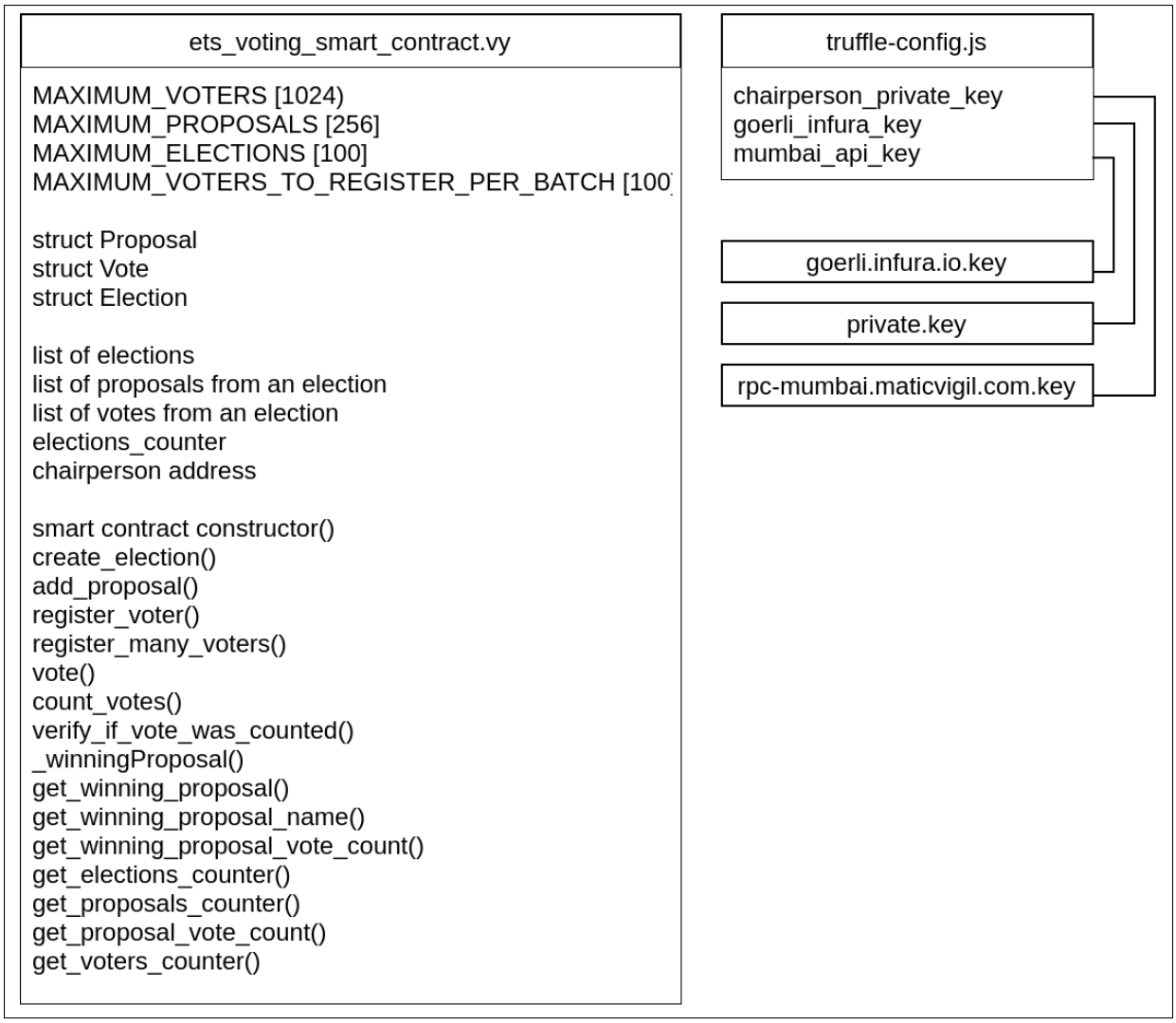

Figure 5.2 Diagram showing the files of the smart contracts module, their functions and main variables

Notes: The chairperson can register elections with repeated names. This constraint has not been enforced to save gas, because the responsibility can be delegated to the chairperson to make sure that they will not try to register an election with the same name twice.

5.2.1.2 add_proposal [Use Case 3]

Result: Adds a new proposal to an existing election

Parameters:

- Integer with the ID of an election
- String with the name of the proposal
- String with the description of the proposal

Constraints:

- Can only be called by the chairperson
- The target election must be valid and due to start

Notes: The chairperson can register proposals with repeated names. This constraint has not been enforced to save gas, because the responsibility can be delegated to the chairperson to make sure that they will not try to register an election with the same name twice.

5.2.1.3 register_voter [Use Case 4]

Result: Enables a voter to vote in an election

Parameters:

- Integer with the ID of an election
- Address from a voter account

Constraints:

- Can only be called by the chairperson
- The target election must be valid and due to start
- The maximum number of voters of the election has not been reached
- Voter is not already registered to vote in the target election

5.2.1.4 register_many_voters [Use Case 4]

Result: Enables many voters to vote in an election

Parameters:

- Integer with the ID of an election
- List of Addresses from a voter account

Constraints:

- Can only be called by the chairperson
- The target election must be valid and due to start
- The maximum number of voters of the election would not be reached after adding all voters in the list
- All voters from the list are not already registered to vote in the target election
- The list of voters has a maximum number of elements, defined in the smart contract and currently at 100. The list must be padded with null addresses if there is less than the limit.

5.2.1.5 vote [Use Case 5]

Result: Stores the vote of a voter, yet to be counted, but verifiable that they voted.

Parameters:

- Integer with the ID of an election
- Integer with the ID of a proposal

Constraints:

- The target election must be valid and started but not finished (voting period is active)
- The target proposal must be valid
- The voter has the right to vote in the election
- The user has not already voted in the election

5.2.1.6 count_votes [Use Case 6]

Result: Counts the votes received by all voters to each proposal of an election

Parameters:

- Integer with the ID of an election

Constraints:

- Can only be called by the chairperson
- The target election must be valid and finished (voting period is over)
- Votes of the election have not been counted yet
- No vote of any voter has already been counted for the target election

5.2.1.7 verify_if_vote_was_counted [Use Case 8]

Result: Returns True if the vote has been counted and returns False if it has not been counted

Parameters:

- Integer with the ID of an election
- Address from a voter account

Constraints:

- The target election must be valid and finished (voting period is over)

5.2.1.8 _winningProposal [Use Case 7]

Result: Computes and returns the ID of the winning proposal taking into account all votes that have been counted

Parameters:

- Integer with the ID of an election

Constraints:

- It's an internal function that can only be called by other functions of the smart contract
- The target election must be valid and finished (voting period is over)
- The votes of the target election have already been counted

5.2.1.9 get_winning_proposal [Use Case 7]

Result: Returns the ID of the winning proposal of an election

Parameters:

- Integer with the ID of an election

Constraints:

- The target election must be valid and finished (voting period is over)

Notes:

- Calls the internal function _winningProposal()

5.2.1.10 get_winning_proposal_name [Use Case 7]

Result: Returns the name of the winning proposal of an election

Parameters:

- Integer with the ID of an election

Constraints:

- The target election must be valid and finished (voting period is over)

Notes:

- Calls the internal function _winningProposal()

5.2.1.11 get_winning_proposal_vote_count [Use Case 7]

Result: Returns the vote count of the winning proposal of an election

Parameters:

- Integer with the ID of an election

Constraints:

- The target election must be valid and finished (voting period is over)

Notes:

- Calls the internal function _winningProposal()

5.2.2 Code optimizations

The mail goal of optimizations is to save on gas costs.

One way of doing so it using memory instead of storage to keep intermediate results of calculations. This was done whenever possible.

Another attempt was to use data structures to keep the data in a more concise way.

Structs allocates a certain amount of memory that will be padded to fill its memory space. So the solution consisted in using the bits of an integer array to keep boolean values. This can be seen in Figure 5.3.

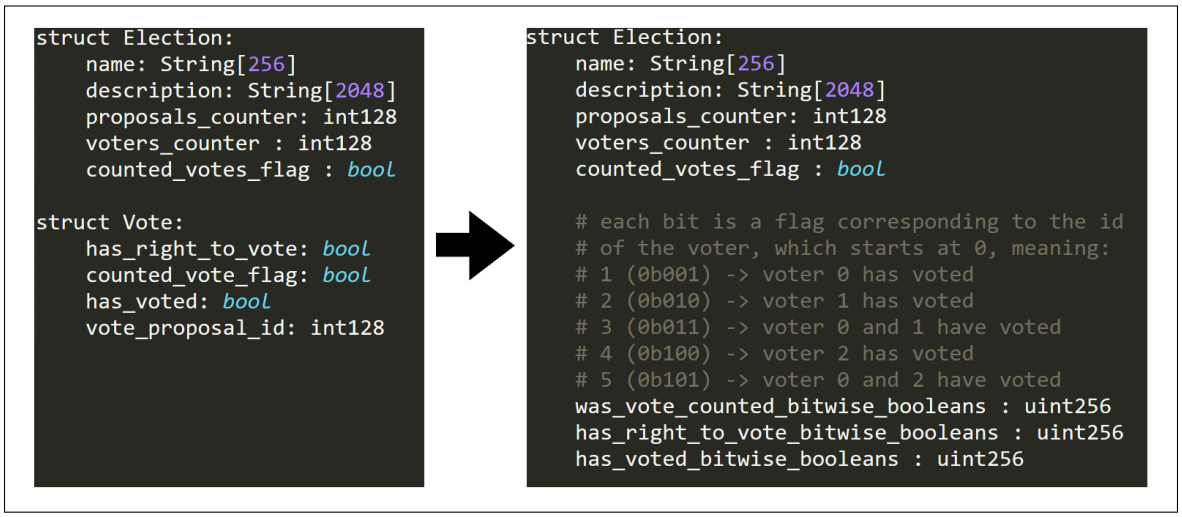

Figure 5.3 On the left is the original snippet of code and on the right is the equivalent code with optimizations

This solution showed some good results, around 33% reduction in the gas costs, but only for certain cases, in which the number of voters participating was optimal. In other words, the 33% improvement was not seen across all simulation configurations.

Another drawback of such solution is that it limits the quantity of voters, unless arrays of integers are used, which would greatly increase the complexity of the code.

The complexity of the code is a very important point, because trust in the system is one of the principles of a successful voting system, and as such, the code should be readable and simple, so that it can be audited by even a voter that is not technically skilled in programming.

Because of these reasons, the bitwise optimizations branch of the code was not merged into the main branch. Later, it was found out that a way better option to tackle the costs was deploying the smart contract in blockchain platforms other than Ethereum.

5.2.3 The decision of where to store the votes

The voting system persists the votes in the blockchain blocks.

This decision was made after a case study was done for the master's course "INF7210 - New perspectives in databases" at the Université du Québec à Montréal (UQAM).

The conclusion was that a blockchain database such as BigchainDB could be used to store votes in an electronic voting system deployed on a blockchain platform, but it's not necessary unless such system meets a scaleability ceiling.

So, for simplicity, and given the context of this system being for the elections or referendums of a university, it was decided to store the votes directly on the blockchain.

5.2.4 The choice of Ethereum as the blockchain to host the system

Ethereum was chosen as the blockchain platform to host this project because of its principles, such as being open source and having a shared governance system.

However it's not the only blockchain platform. Initially, another alternative considered was EOS, which was built with scaleability in mind since the beginning, which is actually a problem with Ethereum. This means that currently, EOS has a bigger throughput than Ethereum, meaning that it does many more transactions per second.

EOS is a blockchain owned by a company which centralizes governance for the project, which is the main problem with it to host a project that has deep political and social roots.

Although EOS governance is centralized, it does avoid problems that are present in decentralized blockchains platforms. The platform is also open source, so it was considered as a strong contender to host the voting system.

The issue can be summarized as "So far, one of the biggest criticisms of EOS is its centralization — an integral part of the platform's design. The delegated proof-of-stake consensus only ever allows a fixed number of 21 block producers, with token holders entitled to cast their votes for who gets to participate as one of the groups." as said by Kuznetsov (2020).

So, after investigation, it was decided not to go with EOS for this kind of system.

Like it was mentioned before, Ethereum has scaling issues. It's openly documented that the upcoming Ethereum 2.0 is going to improve its throughput from the current 10tx/s to at least 1000 times that on Tech (2020), Georgiev (2021).

However, as of 2021-07-29, the release date of this stage is still uncertain. The openness of the project helps to increase the trust that it will succeed.

For this reason, the Binance Smart Chain and the Matic platforms were considered and used. Both of them are able to run the same code from Ethereum projects, because they use the EVM.

Their security and decentralization aspects still need analysis to conclude if they are enough for big elections or referendums, but they are a good indicator of what we can expect from Ethereum 2.0 once it's released and the Matic network uses the underlying security of the Ethereum blockchain.

5.2.5 The choice of one deployment for many elections or referendums

One design option would be to have each voting run by a smart contract. In other words, starting an election would mean deploying a smart contract.

It was decided not to go this way, because it can be more expensive, as there is a cost to deploy smart contracts.

Another reason is that the system would have decreased security, as the user interface and the centralized API would have to keep track of which contracts are out there and are valid, and it could be a weak point of attack.

Something else that could be included is the list of voters with the deployment of the contract, but this would also reduce the flexibility for the organizers of an election.

But doing so would be limited for having not enough gas to run the transaction: the gas limit per transaction could quickly be depleted by the list of voters, because deploying a contract is basically sending a transaction to the blockchain, which has a limit of how much gas can be used.

So what justifies separating executions in the contract is a must. In other words, Smart Contracts make justice to their names, and it seems to be good design to have well-defined functions.

5.2.6 The choice of Vyper to code the Smart Contract

Once it was decided which blockchain to host the system, it was time to decide which programming language to code the Smart Contract on.

The two main languages that compile to the Ethereum Virtual Machine (EVM) are Solidity and Vyper. The article Kaleem, Mavridou & Laszka (2020), among others, was very important to make the decision to proceed with Vyper.

A possible programing language considered was the functional programming language Plutus used by the Cardano blockchain, which has qualities that make it a good candidate for coding a voting system.

Using Plutus would also mean having to deploy the project on the Cardano platform, that still doesn't have support for smart contracts.

So, the selected language to deploy it on the Ethereum network is Vyper, because it was designed with simplicity, audit ability and security in mind, contrasting with the numerous security vulnerabilities and attacks witnessed on contracts written in Solidity.

However, Solidity does have the audited code of the framework Open Zepelin, which could be used to make the system more secure, so Vyper was mostly selected for being much easier to read the code, which helps the system to gain trust from the users.

Vyper is not a Turing Complete language, but its features are enough to implement smart contracts that run elections or referendums.

5.3 Restul API module

The Restles API is a gateway for the centralized infrastructure of the system. It's meant to be used by the mobile and web apps.

The endpoints of this API are the following:

- Hello World GET: Endpoint meant to test the connectivity to the API. Returns a hello world message.
- Hello World Protected GET: Endpoint meant to test the authentication to the API. Returns a hello world message with the logged user's email.
- User login POST: Authenticates a user and returns a pair of Bearer tokens (access and refresh). The access token is flagged as FRESH, meaning it can be used to access fresh-token protected endpoints.
- User login DELETE (logout): De-authenticates a user by revoking the current access token.

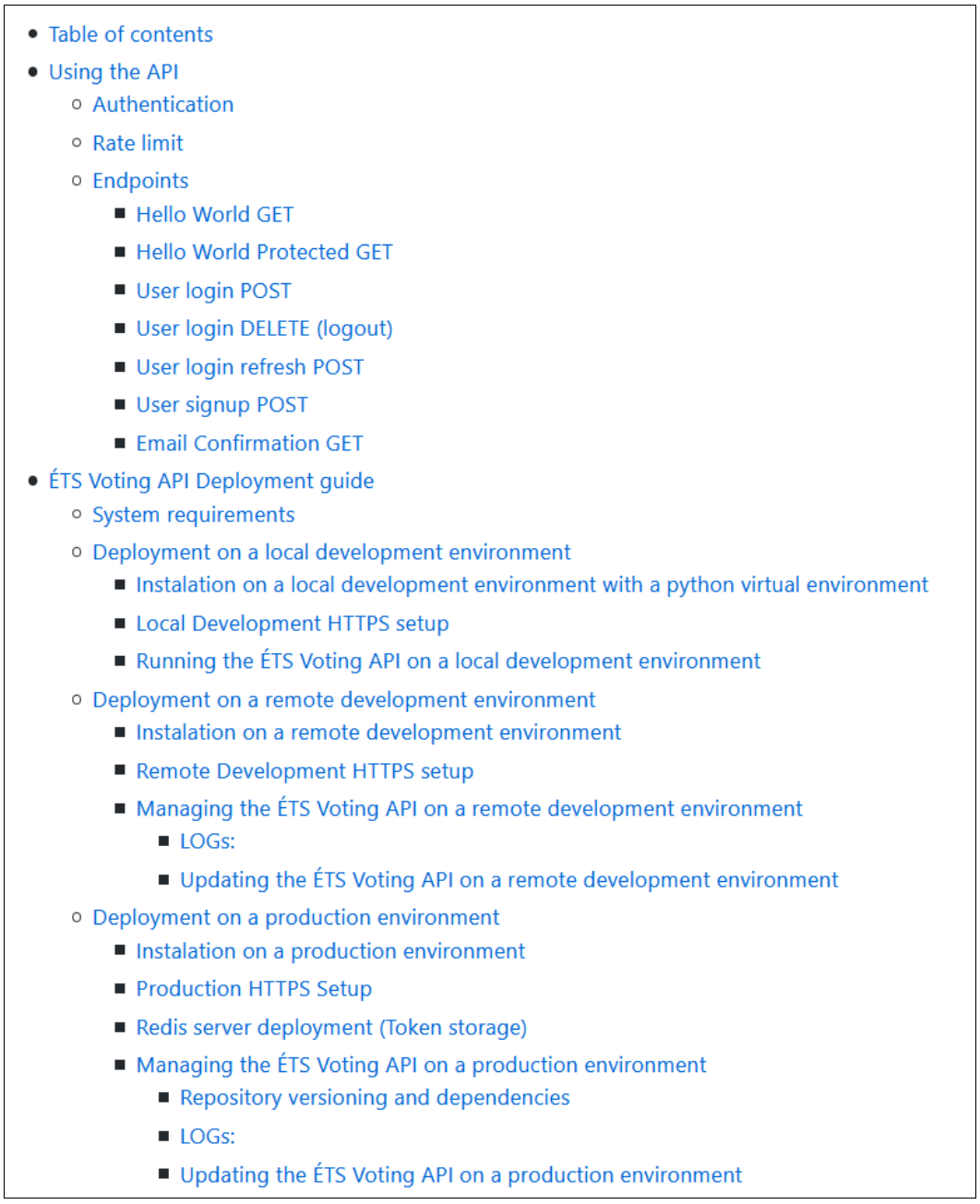

Figure 5.4 The table of contents of the documentation of the relational database module

- User login refresh POST: Re-authenticates a user and returns a pair of Bearer tokens (access and refresh). The input is a refresh token, which is valid for a little longer than the regular access token. This endpoint is used to the keep-alive functionality. The inputted refresh

token is invalidated after being consumed by the endpoint. The access token is flagged as NOT FRESH, meaning it can't be used to access fresh-token protected endpoints.

- User signup POST: Registers a user and sends an email for the user to confirm they own it.
- Email Confirmation GET: Confirms the email of a registered user.

Figure 5.4, shows a screenshot of the table of contents of the documentation for this module available in its repository (https://github.com/eduardoxfurtado/ets_voting_api).

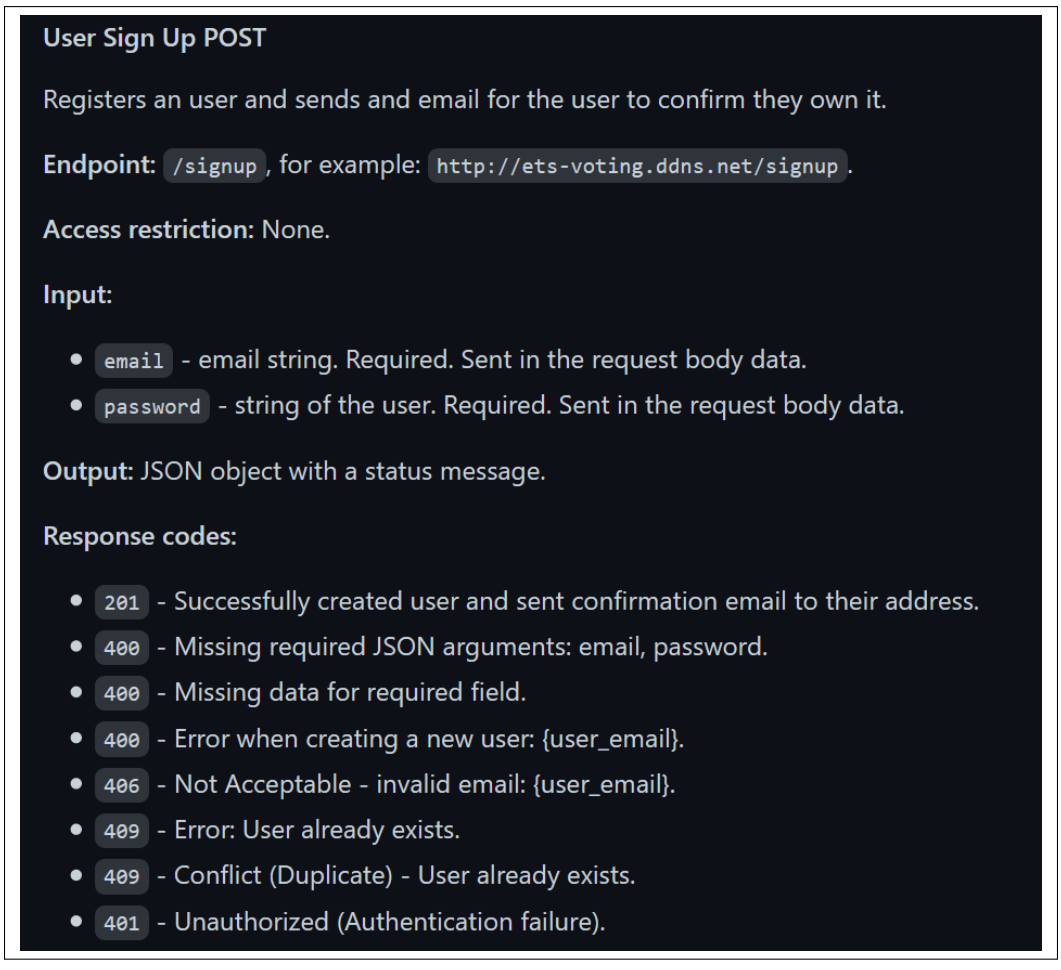

Figure 5.5 Screenshot of the documentation of the User Sign Up endpoint

An example of the documentation for one of the endpoints can be seen in the screenshot shown in Figure 5.5.

5.4 Relational Database

The relational database is simple, as it can be seen in Figure 5.6, it only has two tables to store the data for the API users and the voters.

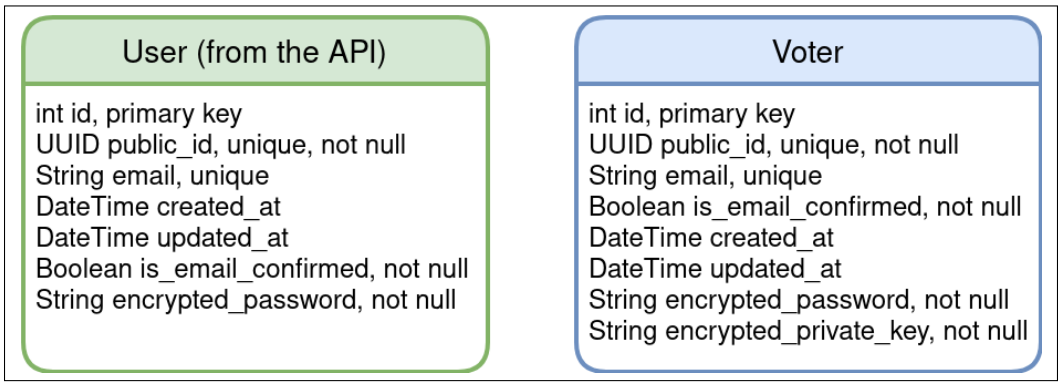

Figure 5.6 The tables of the relational database

There is a Python module that wraps the database and it contains the software architecture that defines the database. It should contain both the schemas and the controllers. It has tools for manipulating database resources as well.

Figure 5.7, shows a screenshot of the table of contents of the documentation for this module available in its repository (https://github.com/eduardoxfurtado/ets_voting_relationaldb).

5.5 Voting app module

This module was not developed because it needs technical expertise in the development of mobile applications. A complete design is presented in this section.

Fluter is the technology chosen for this module because it allows the module to be built with a single code base that can be built into the two major mobile platforms (iOS and Android) and also as a web app.

Android 10 (Pie) was selected for the tests as it seems to be the version of the OS with the biggest market share - https://www.appbrain.com/stats/top-android-sdk-versions

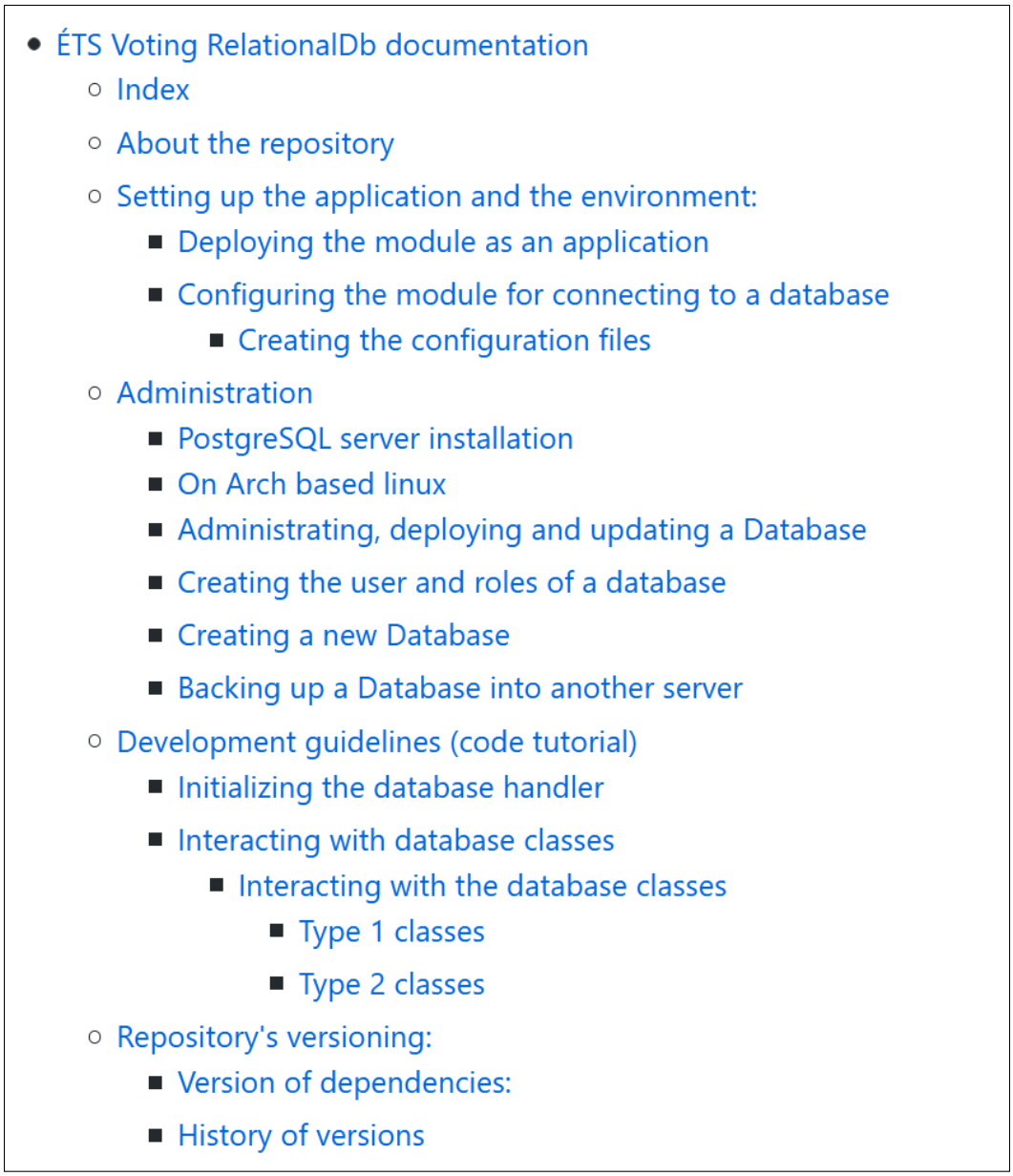

Figure 5.7 The table of contents of the documentation of the relational database module

The figure 5.8 shows the simplified design of the app, because the full design is a big diagram that is hard to be included in a report document.

The figure 5.9 shows the full mobile design of the app. Its size makes it not ideal to be visualized in this document, but it's available as a bigger image at https://raw.githubusercontent.com/

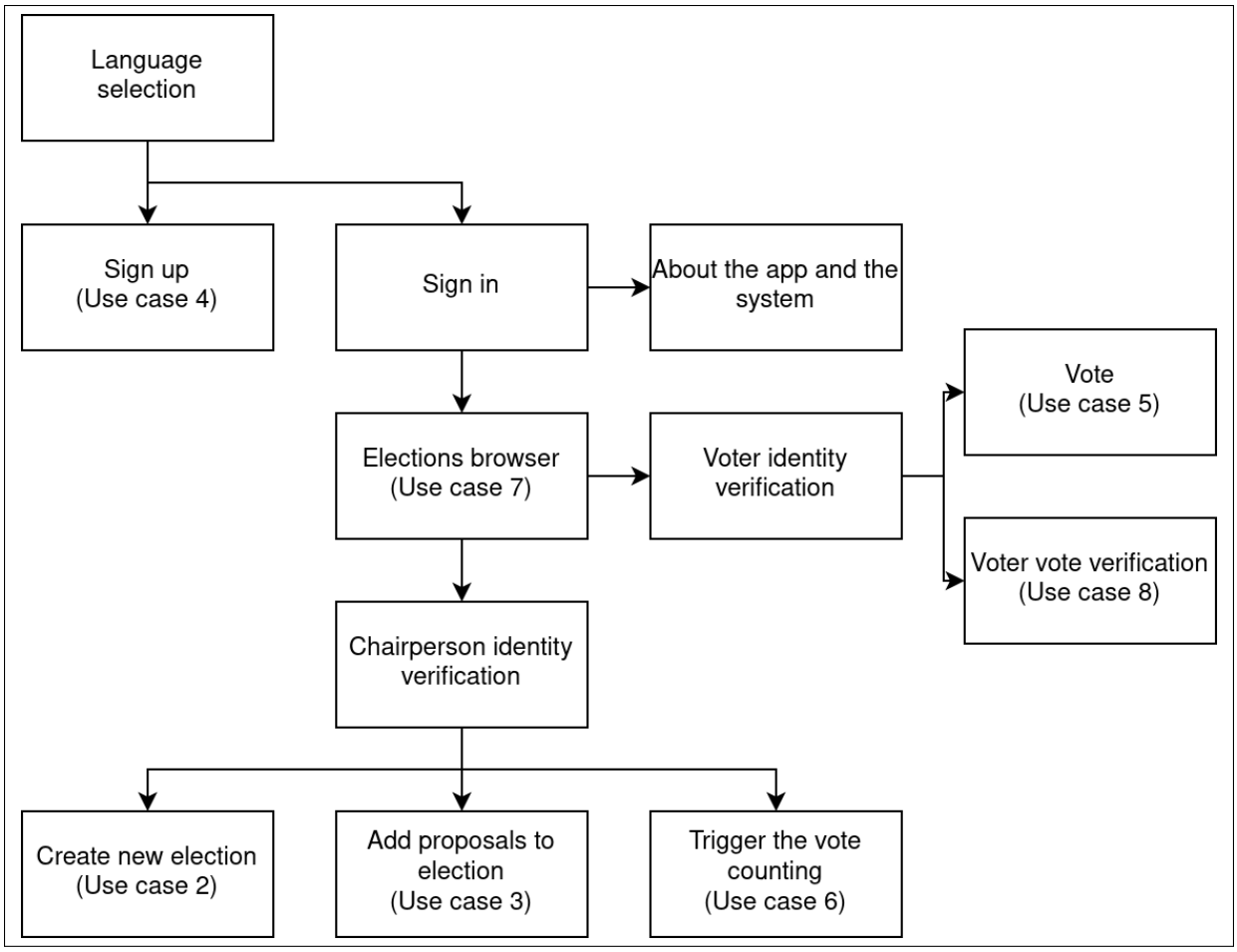

Figure 5.8 Simplified design of the voting app

eduardoxfurtado/ets_voting_report/master/project-latex/Diagrams/ets_voting_app.png. The original file with the "drawio" extension that can be opened with the open-source tool "Diagrams.net" and it's available at the open source repository of this module: https://github.com/eduardoxfurtado/ets_voting_app.

5.5.1 Authentication

It's crucial to validate the identity of a voter or chairperson before any action that results in transactions with the blockchain. The validation process is supposed to be automated, which

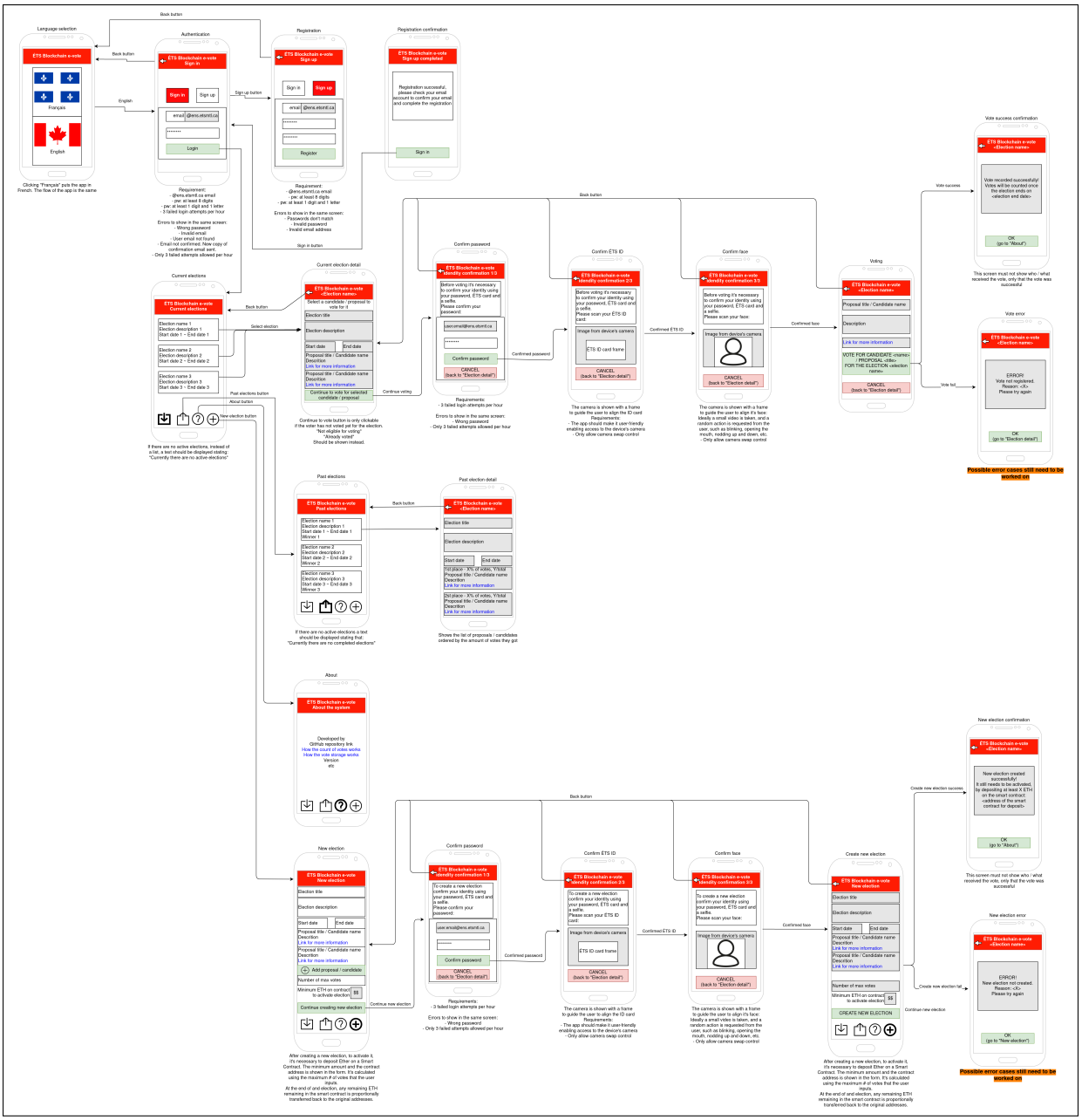

Figure 5.9 Full mobile design of the voting app. A bigger resolution file is available at https://raw.githubusercontent.com/eduardoxfurtado/ets_voting_report/master/project-latex/Diagrams/ets_voting_app.png

means utilizing computer vision to validate a selfie of the voter and a piece of identity from the voter, such as the school's ID card.

It's extremely important to identify each voter that is using the system.

To do so in an ideal scenario:

- The voter's school ID card should be scanned and validated;
- The voter's face should be validated according to their student ID and the image should also be validated as being taken in the moment of voting;
- The voter should create an account with their email;
- Another ID, besides the school ID, should be scanned, such as a driver's license, passport or citizen ID card.

The school's ID cards are a property of the school, and one must return them once they cut ties with the ÉTS. However, people might lose their cards, allowing an attacker to use them to vote. To solve this reason, maybe the centralized database might be included to assist the process of validation of documents and faces with access to school's data. This should take care of cases such as someone trying to use an expired student card to vote.

Access to a centralized resource owned by the school could leave traces in the logs, such as time of authentication, to allow an attacker to infer who the voter voted for by using the time of the vote, so the solution should not threaten the separation of a voter's vote and its identity.

5.5.2 Flutter template

Different templates were considered for being a base for the implementation.

Using a template allows the coding to be done faster, as there is no need to code things that are usually common among projects, such as the code to route one screen of an application to another screen.

The following apps were discarded because they either had poor documentation or they had a default UI that was too complex compared to what is wanted for this project:

- Open source Flutter-based template for a business assistant application - https://flutterawesome.com/open-source-flutter-based-template-for-a-business-assistant-application/

- mitesh77 / Best-Flutter-UI-Templates - https://github.com/mitesh77/Best-Flutter-UI-Templates/tree/master/best_flutter_ui_templates/lib
- Norbert515 / BookSearch - https://github.com/Norbert515/BookSearch
- zubairehman / flutter-boilerplate-project - https://github.com/zubairehman/flutter-boilerplate-project
- trentpiercy / trace - https://github.com/trentpiercy/trace

The project X-Wei / flutter_catalog - https://github.com/X-Wei/flutter_catalog was greatly used as reference for different Flutter UI elements that can be seen in the live demo: https://x-wei.github.io/flutter_catalog/.

It's recommended that future developments also use this template, or find a better one, considering the others that were tested and listed above.

5.6 Election simulator module

The module was implemented in python and it contains many unit tests that will test both the centralized API and database as well as the decentralized Smart Contracts module using the Web3 library.

Figure 5.10 shows the execution flow of the election simulator module. The 24 tests are classified by color as shown in the legend.

An example of a simulation run can be seen in Figure 5.11.

If all the tests pass, it means that it's possible to run an election using the system deployed on a Blockchain and also that the backend for applications with a graphical user interface can run in front of the system.

It is a non-exhaustive list of tests, meaning that there is no guarantee that the system is secure, so more tests could be implemented. Adding more tests would increase the quality of the system, or contemplate more than the use cases presented in this report.

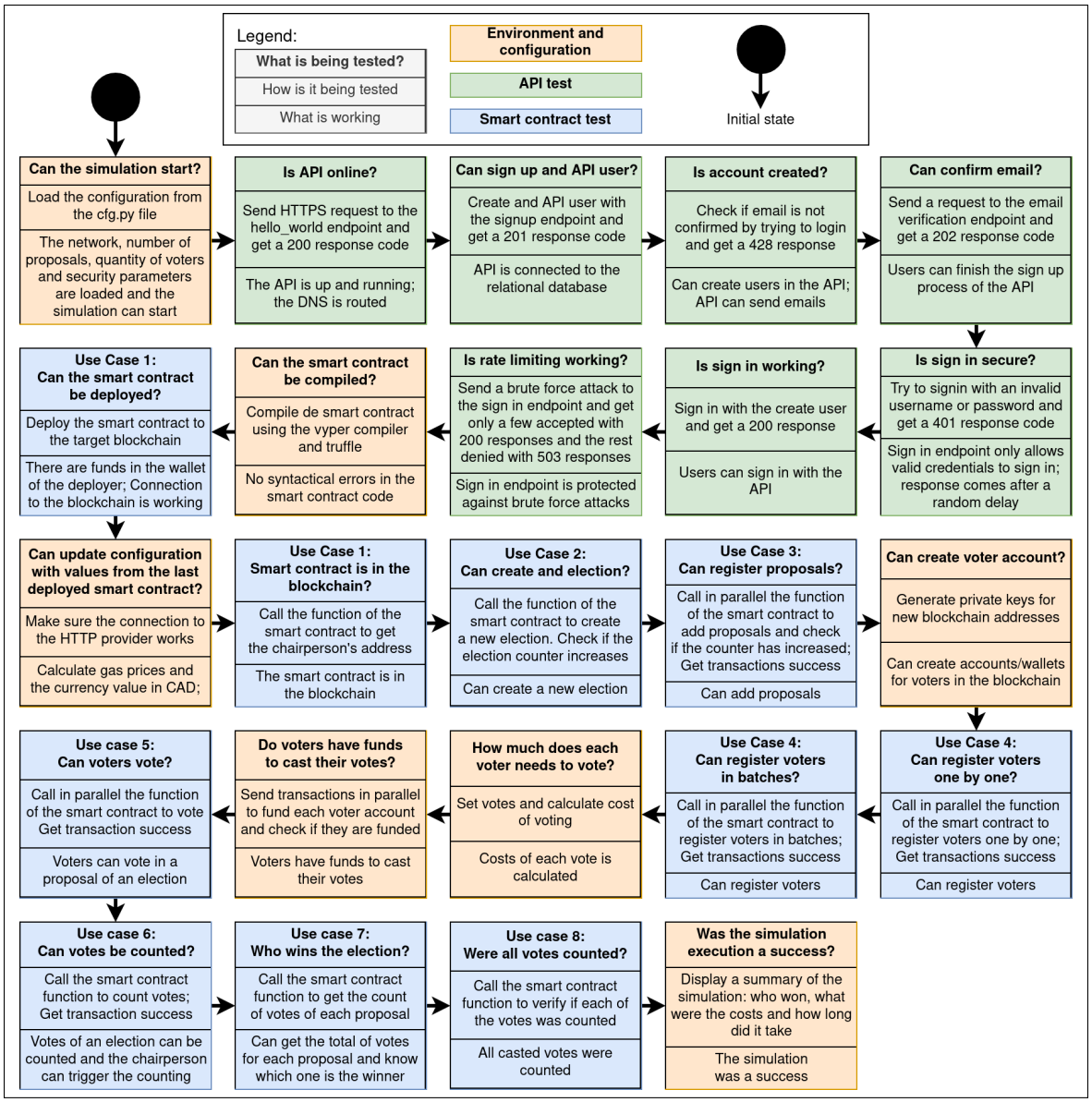

Figure 5.10 Steps taken during the simulation of an election

After all the tests have run with success, a summary of the simulation is presented, like the screenshot shown in Figure 5.12.

Log files are also created, as it can be seen in figure 5.13 that shows the files of the module, their functions and their main variables.

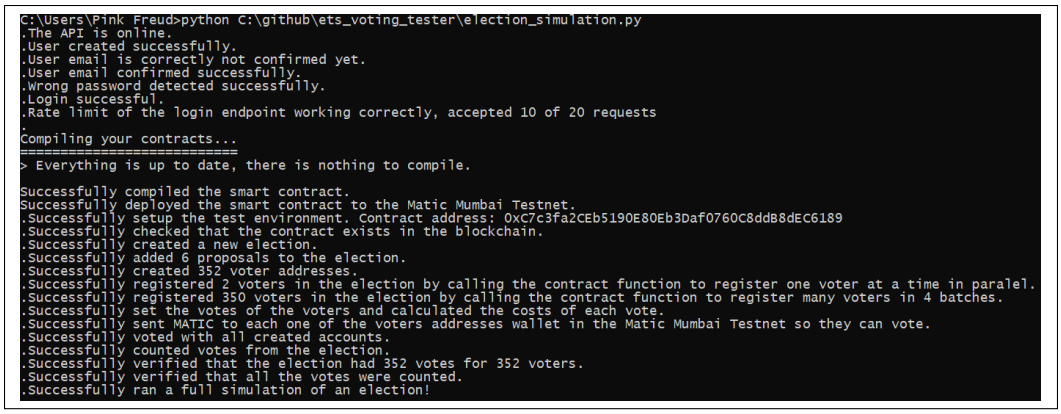

Figure 5.11 Screenshot of a simulation run

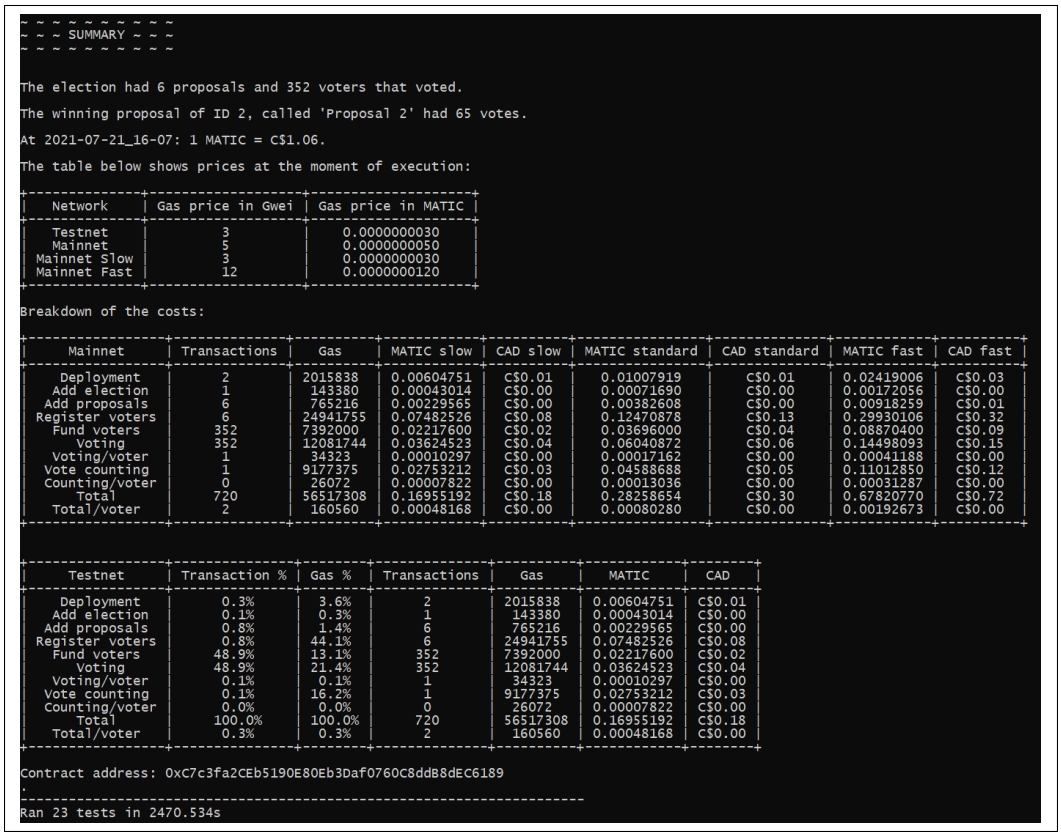

Figure 5.12 Screenshot of the summary displayed at the end of a simulation

Figure 5.14 is a screenshot of the transactions log file, of which the transactions can be checked using the transaction hash in a block explorer.

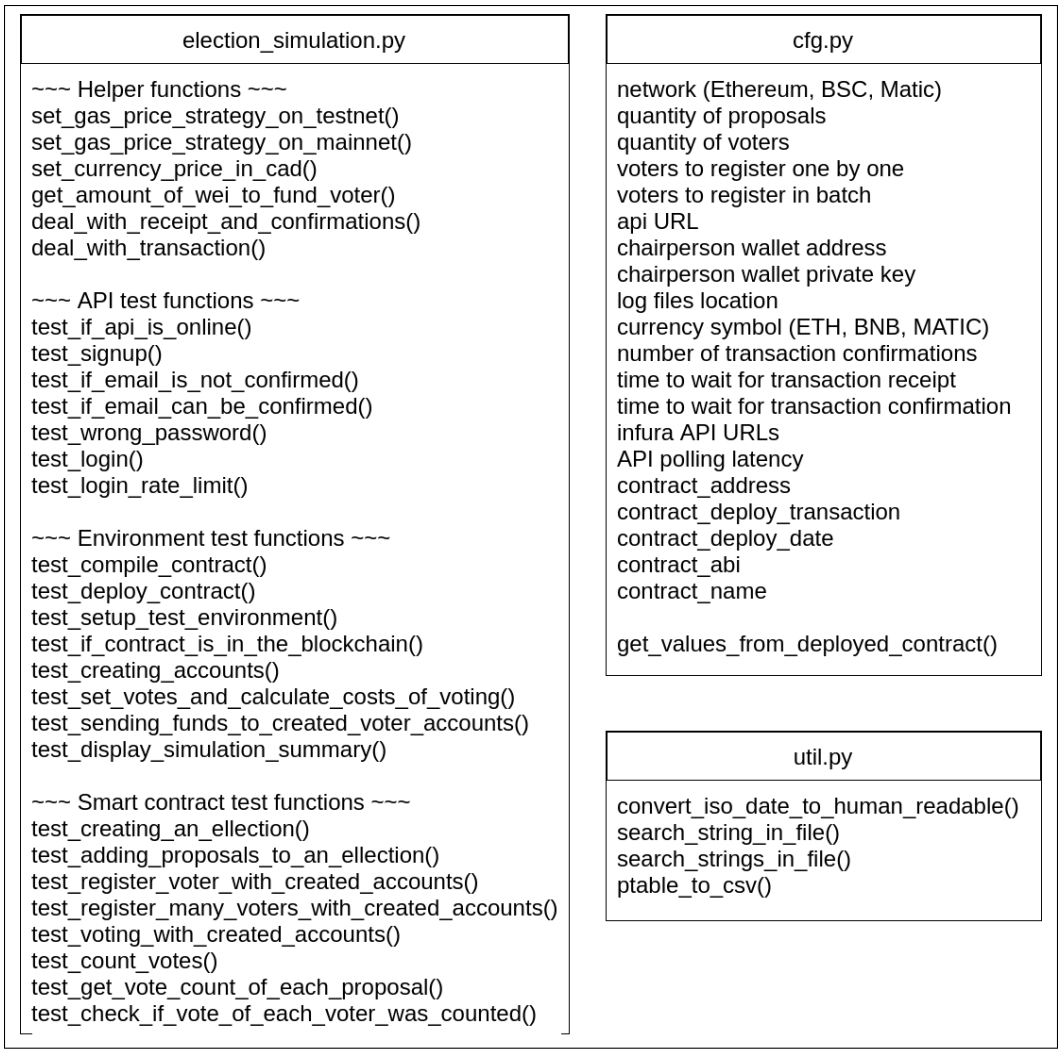

Figure 5.13 File structure of the simulator module with their functions and most important variables

Figure 5.15 is a screenshot of one of the log files showing the receipts with details of two transactions.

Figure 5.16 is a screenshot of the Comma-Separated Values (CSV) file displaying the costs of how much it would cost to run an election with the set parameters. These files were combined in the data presented in chapter 6.

0xf99a3e60332d96abc31e1d37e05d7829d4fe9a5ab01554c6fa2a9ee6c63fe2c0, nonce 1905, type: new_election
0xfb548a32ce768a0e5f2c9a912531f671c3544a273d28fcba8ec516615f4d4fef, nonce 1906, type: new_proposal
0x451f8e5b6c039990ef454b94c67d7cbcc5cd4ddad7b0d4cd58b5f420a2b5d929, nonce 1907, type: new_proposal
0x69e99069e0e8d008b17e13b68950941266daae62e60490359c1d70fde0790df8, nonce 1908, type: new_proposal
0xf2a072cc1a11ad7645282cf2c4c02e16fa6af86255dcecf6e165bf18db11f654, nonce 1909, type: new_proposal
0xaba1e97cc38be1eab6e06300514e39c7042fb51a7fe7ee8d084fbc165a65349e, nonce 1910, type: new_proposal
0xe4dfe560e12df449c25a18ee519d1e362ffdd29a5ae4950e78feb0595e830cb4, nonce 1911, type: new_proposal
0x4a932274156410969c34c95f1d6ab79c606d53ccbb0ff3e73f687422bbf2ccf9, nonce 1912, type: new_proposal
0x369d928c21b1a503547a3124a56bd3c7c776fea60fec97e465cbc8456f880dfc, nonce 1913, type: new_proposal
0x5a255cdd2dea8e75cdf91151b353b47bfb3cd40dc04655a73ec9a1c32a686139, nonce 1914, type: new_proposal
0x6fd11912d62327b87c790f209ad8d2a852668742dba1ee0c61a5e38f49cb7b5e, nonce 1915, type: register_voters
0xf0d06ce36cdb834c23cd37e90f0b2474a61683bb0416f8f76e2d7be8131a0df1, nonce 1916, type: register_voters
0x575cacb90d3e9cb6e35e5b44050c36a89fd7bc36739e8dc18ab45716ab9d133b, nonce 1917, type: register_voters
0x6469208714ab6b713bcd28482f69d37fa0bf32d10f3aae0307c3c4e1087939d3, nonce 0, type: voting
0xea720ac831dd303a29b2fd7fd35c1ddf0bc3a38480ddef1d63c12c7241ff99e7, nonce 0, type: voting
0x6ccdc3a7f68d35209faf7333cfbb968c434dcfa81b500208c97c963d125605ce, nonce 0, type: voting
0xec112ae80c6186a0acc92d832d04aed3b05d15501117fe499a71c9f2cd023df4, nonce 0, type: voting
0xcf81395f1b7fb2820111520658ff04349e17b344ebac81908829b1a71402452d, nonce 0, type: voting
0xe27d3ffc6fac8455315a865323543131d98d4dd7db35153f7c45c2cf091e71b9, nonce 0, type: voting
0x82988a1ccab362cbf97318d2fcc3a25c8b6aedf3aeea35a94e61c8bbaba64f47, nonce 0, type: voting
0x77f6333550eb9f90e8a06e4e0e10f83f69c44ba23f681093ac0ff1bebc5c586a, nonce 0, type: voting
0xc8bf5ed2b8e22c51c4aff2b67fe9efc8691edfc9f740416112a4e8bce5182476, nonce 0, type: voting
0x29a570882e1b7602b2c8e19c1ce28f503289ce59195056386d88ec65549990cf, nonce 0, type: voting
0xa9aec312d619f1204c1aa6a4de5cea3fce60f3dc52bd984b03aec8998a96f144, nonce 0, type: voting
0x45a11a26558839cfde1233290c0670883f9e2eefba1f84ae79039625e44b007e, nonce 0, type: voting
0x75037ac3681d4e3afbd74f35392480bdf6a761225b3ba277cdee28d11603547e, nonce 0, type: voting
0x8538327ebd5b606d45e7a7df67ddc7fb5508848ba50ae424788a293b4b2c5306, nonce 0, type: voting
0xe2ac0631cffbbd4fa978b82caf23706b558f20bf1b31ae9ebddfe45c4a7f000f, nonce 0, type: voting
0x9e59943fc8f590cb5d7fc8c71fed28975f6d0a81593d6ea7a39b4de4b81e381c, nonce 0, type: voting
0xd0536fb471d4d282f3140a6da2d4a9fbdd965eb6a0f00548824ec415d91929f0, nonce 1918, type: vote_counting

Figure 5.14 Screenshot of the log files showing the transactions of a simulation

Figure 5.15 Screenshot of the log files showing the receipts of transactions of a simulation

	A	B	C	D	E	F	G	H	I
1	At 2021-07-21_16-07: 1 MATIC = C$1.06.								
2									
3	Network	Gas price in Gwei	Gas price in MATIC						
4	Testnet	3	0.000000003						
5	Mainnet	5	0.000000005						
6	Mainnet Slow	3	0.000000003						
7	Mainnet Fast	12	0.000000012						
8									
9	Mainnet	Transactions	Gas	MATIC slow	CAD slow	MATIC standard	CAD standard	MATIC fast	CAD fast
10	Deployment	2	2015838	0.00604751	C$0.01	0.01007919	C$0.01	0.02419006	C$0.03
11	Add election	1	143380	0.00043014	C$0.00	0.0007169	C$0.00	0.00172056	C$0.00
12	Add proposals	6	765216	0.00229565	C$0.00	0.00382608	C$0.00	0.00918259	C$0.01
13	Register voters	6	24941755	0.07482526	C$0.08	0.12470878	C$0.13	0.29930106	C$0.32
14	Fund voters	352	7392000	0.022176	C$0.02	0.03696	C$0.04	0.088704	C$0.09
15	Voting	352	12081744	0.03624523	C$0.04	0.06040872	C$0.06	0.14498093	C$0.15
16	Voting/voter	1	34323	0.00010297	C$0.00	0.00017162	C$0.00	0.00041188	C$0.00
17	Vote counting	1	9177375	0.02753212	C$0.03	0.04588688	C$0.05	0.1101285	C$0.12
18	Counting/voter	0	26072	7.822E-05	C$0.00	0.00013036	C$0.00	0.00031287	C$0.00
19	Total	720	56517308	0.16955192	C$0.18	0.28258654	C$0.30	0.6782077	C$0.72
20	Total/voter	2	160560	0.00048168	C$0.00	0.0008028	C$0.00	0.00192673	C$0.00
21									
22	Testnet	Transaction %	Gas %	Transactions	Gas	MATIC	CAD		
23	Deployment	0.3%	3.6%	2	2015838	0.00604751	C$0.01		
24	Add election	0.1%	0.3%	1	143380	0.00043014	C$0.00		
25	Add proposals	0.8%	1.4%	6	765216	0.00229565	C$0.00		
26	Register voters	0.8%	44.1%	6	24941755	0.07482526	C$0.08		
27	Fund voters	48.9%	13.1%	352	7392000	0.022176	C$0.02		
28	Voting	48.9%	21.4%	352	12081744	0.03624523	C$0.04		
29	Voting/voter	0.1%	0.1%	1	34323	0.00010297	C$0.00		
30	Vote counting	0.1%	16.2%	1	9177375	0.02753212	C$0.03		
31	Counting/voter	0.0%	0.0%	0	26072	7.822E-05	C$0.00		
32	Total	100.0%	100.0%	720	56517308	0.16955192	C$0.18		
33	Total/voter	0.3%	0.3%	2	160560	0.00048168	C$0.00		

Figure 5.16 Screenshot of a CSV log file showing the costs of running a simulation and how much it would cost in the main blockchain network

Many transactions can be sent at the same time to the blockchain, because they all end up in the pool of transactions. There is also a nonce (number only used once), to differentiate transactions and also know what their order is.

This fact was taken advantage in the simulator module, as it can be seen in the Python code snippet shown in Figure 5.16. It's a good representation of the real world usage of the system, in which voters could end up casting their votes at the same time.

5.6.1 Funding of voter accounts

In a perfect scenario, each voter would have their own wallet and keep their own private keys safe. This means they would be the ones paying to cast their votes.

This is not ideally from a point of view of convenience, which is not a surprise, given that security is inversely proportional to convenience.

```python
# auxiliary function that will run in a thread
def send_proposal_thread(proposal_index, base_nonce):
    pre_transaction = contract.functions.add_proposal(
        _election_id = g_election_id,
        _name = "Proposal {}".format(proposal_index),
        _description = "Description of the proposal {}".format(proposal_index)
    )
    threads_receipt_status[proposal_index] = deal_with_transaction(pre_transaction,
                                                                   'new_proposal',
                                                                   cfg.CHAIRPERSON_WALLET_PRIVATE_KEY,
                                                                   base_nonce+proposal_index)

# create and start threads with the transaction
for i in range(0,cfg.quantity_of_proposals):
    # initialize the receipt status of the thread as if it failed
    threads_receipt_status.append(0)
    # add the thread to the array of threads to start
    threads.append(threading.Thread(target=send_proposal_thread,kwargs=dict(proposal_index=i,base_nonce=nonce)))

# start the threads
for i in range(0,cfg.quantity_of_proposals):
    threads[i].start()
    time.sleep(0.5)

# wait for all threads to finish running
for i in range(0,cfg.quantity_of_proposals):
    threads[i].join()
```

Figure 5.17 Screenshot of the code showing a snippet of code to make transactions in parallel

For this project, it was decided to go with the convenience way, so the private keys of the voters are kept by the system and the funding of their accounts is done by the system or, more specifically, it's the chairperson that funds their wallets so they have enough to cast their votes.

CHAPTER 6

RESULTS AND COST ANALYSIS

This chapter presents how much it would cost to use the system in the real world for elections such as the ones held by the AÉÉTS, based on the results from the many simulations that were run. A screenshot of the simulation can be seen in figure 5.12 and a video-demo is available at: https://youtu.be/NORpjwKhPkc.

The smart contract that runs the elections was deployed in the testnet, a test network that doesn't require real money to run transactions on, of three blockchains:

1. Ethereum (Goerli testnet)
2. Binanance Smart Chain (Binance Smart Chain Testnet)
3. Matic (Mumbai testnet)

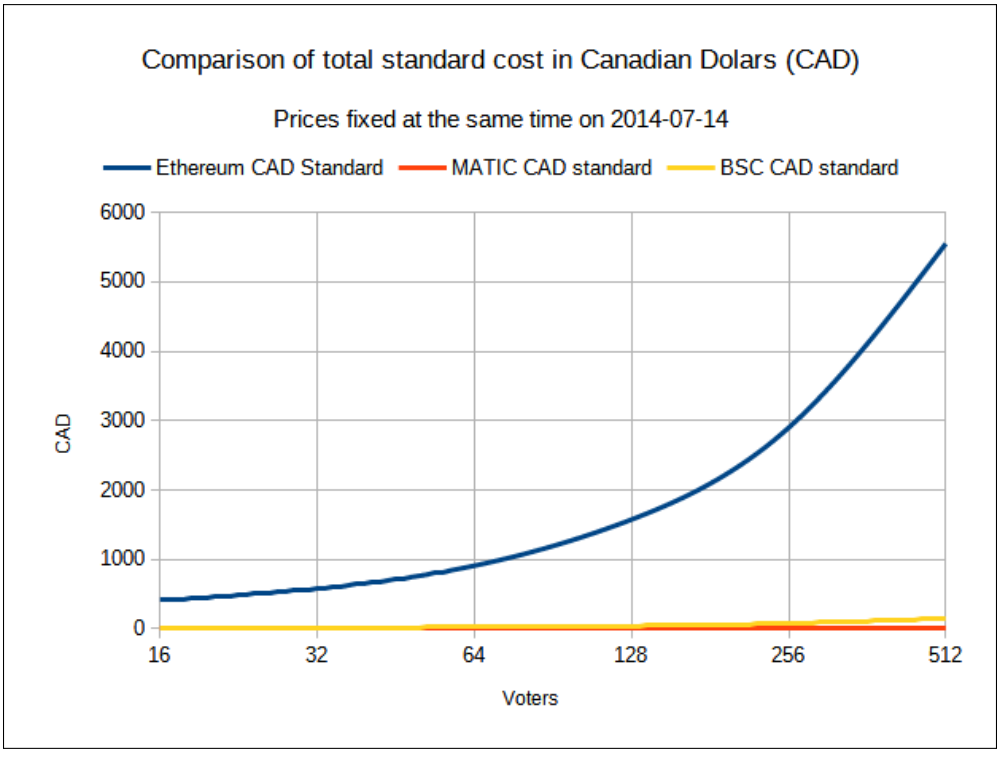

Figure 6.1 Cost of running the elections in the blockchains compared

Figure 6.1 shows how much it would cost to run an election with the system if it was paid with Canadian Dollars. The cheapest network would be Matic, less than 6 CAD to run elections for 256 voters, and the most expensive would be Ethereum, more than 2000 CAD to run elections for 256 voters.

But cost is not the only thing that differs between the blockchains used, and this discussion can be seen in more detail in chapter 5.

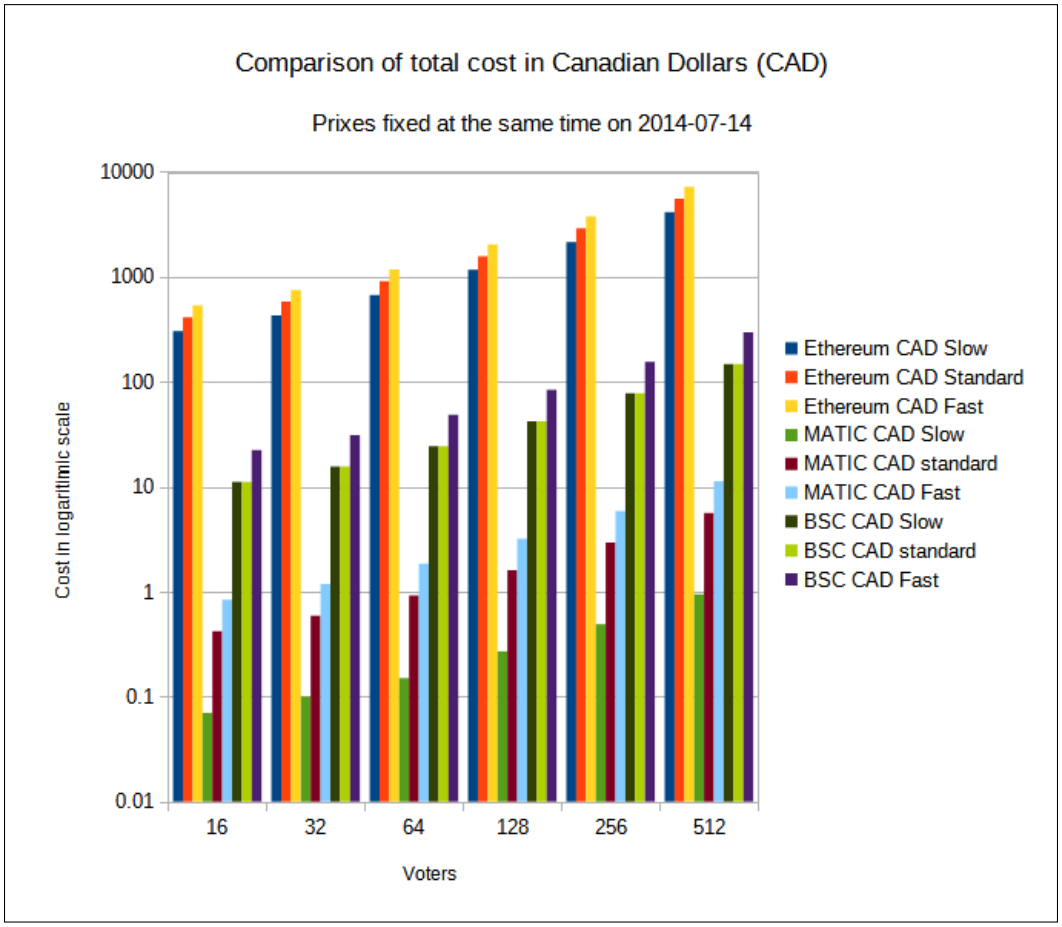

Figure 6.2 Cost of running the elections in logarithmic scale compared

The magnitude in the difference of cost can be observed in Figure 6.2, that shows a logarithmic scale for how much it would cost to run the election for different quantities of voters on the three blockchains.

The total cost of running an election depends on many factors:

- The quantity of proposals. Each new proposal adds one transaction.
- The quantity of voters. Each new voter adds one voting transaction but also increases the amount of data being written when they are registered and the cost of counting the votes.
- The current cost of the gas necessary to pay for the execute code or store data on the blockchain which fluctuates according to the current load of a network. In periods of intense traffic the gas cost increases, and in periods in traffic is low, the gas cost decreases. Three different strategies were used:
  - Fast: a high price is paid for gas, which increases the chance that the transaction will be included in the next block. In other words, the transaction will probably be confirmed sooner.
  - Standard: a standard gas price is paid for gas, and the transaction is included in a block with normal priority.
  - Slow: a low price is paid for gas, which means the transaction can take a long time to be included in a block, and there is also a probability that it will expire before being included in a block.
- The cost in gas for executing different instructions, which can vary from blockchain to blockchain, but also, in rare cases, it can change in new versions of a blockchain, such as Ethereum's EIP-2929, which is about increasing gas cost state access opcodes.
- The current value of the crypto-asset in fiat money, which can change drastically from an hour to the other, as the crypto-markets are known to be volatile compared to the ancient stock markets. For this reason, all tests were run with the value of the crypto-assets fixed at the same time.
- The amount of data being written to the blockchain with a transaction. Writing to the blockchain is the most expensive operation.

Figure 6.3 shows how the quantity of transactions scales when increasing the quantity of voters. The quantity is the same in each blockchain.

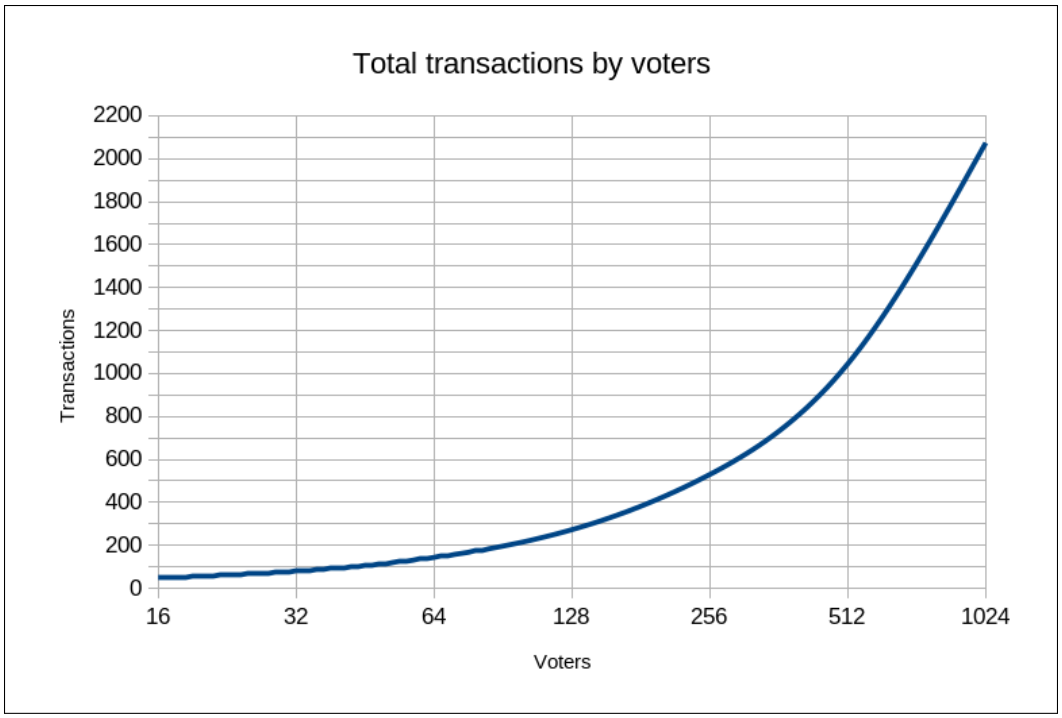

Figure 6.3 Quantity of transactions with the blockchain per voter

Each extra voter adds at least one more transaction: to cast a vote. The registering of voters can be done in batches, so the quantity of transactions doesn't grow so much, but there is still extra data being written to the blockchain for each new voter.

Counting the votes is also dependant on the quantity of voters, because more votes will have to be counted, and for each there is also a write operation of data to the blockchain to flag a vote as counted.

Figure 6.4 shows the distribution of gas cost on the Ethereum network, and how it changes according to how many voters are participating in an election. Each voter registration includes writing data to be persisted in the blockchain, which is a more expensive operation than just executing code.

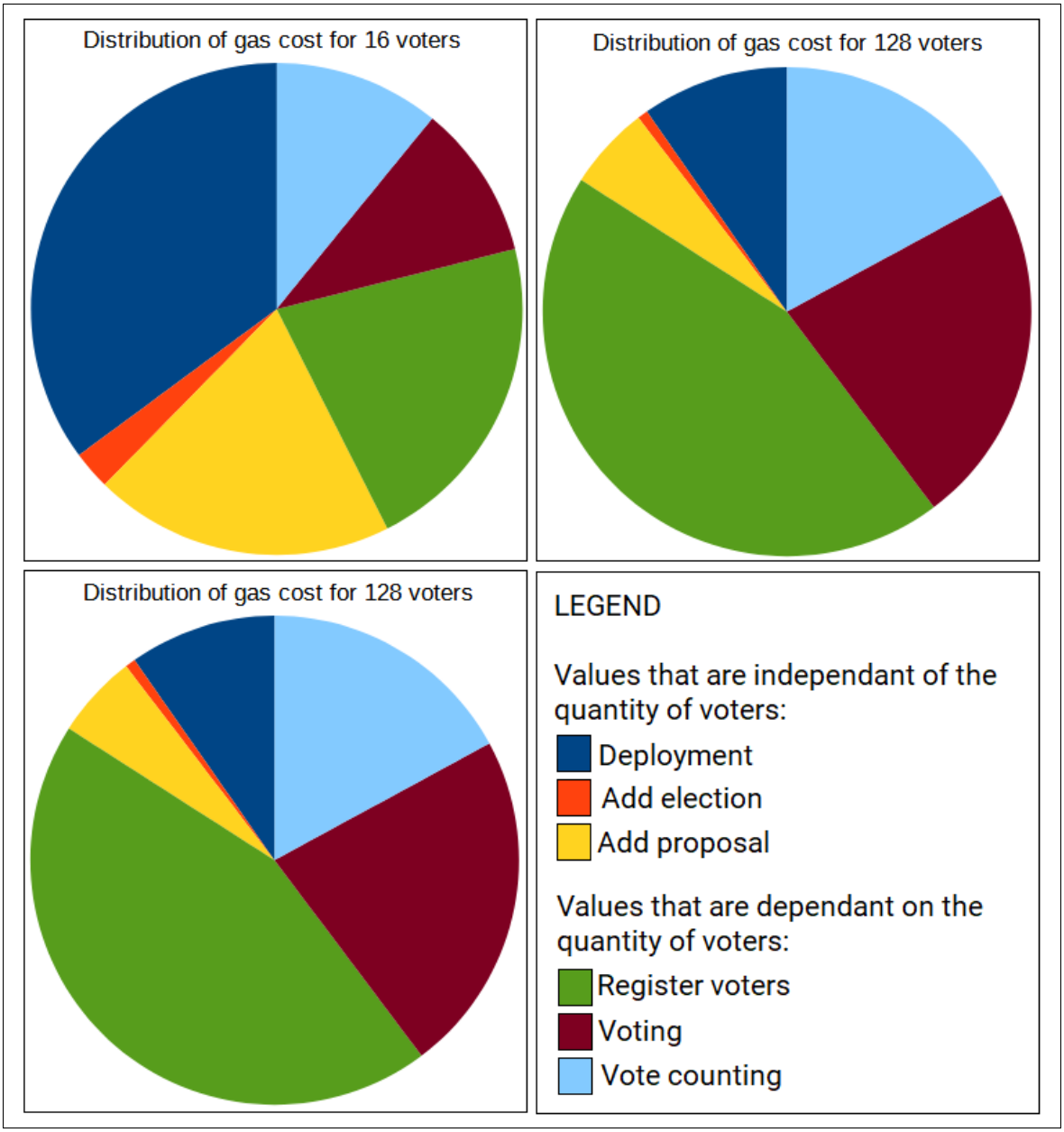

Figure 6.4 Distribution of the gas cost for different election parameters on the Ethereum blockchain

The distribution is important for optimizing the costs where it matters the most. The registration of voters has the greatest impact on the costs, so it would be the first target for receiving optimizations.

6.1 Ethereum

Ethereum was by far the most expensive blockchain to execute the code on, but it was also the only one to support 1024 voters with the current implementation, as it can be seen in Figure 6.5.

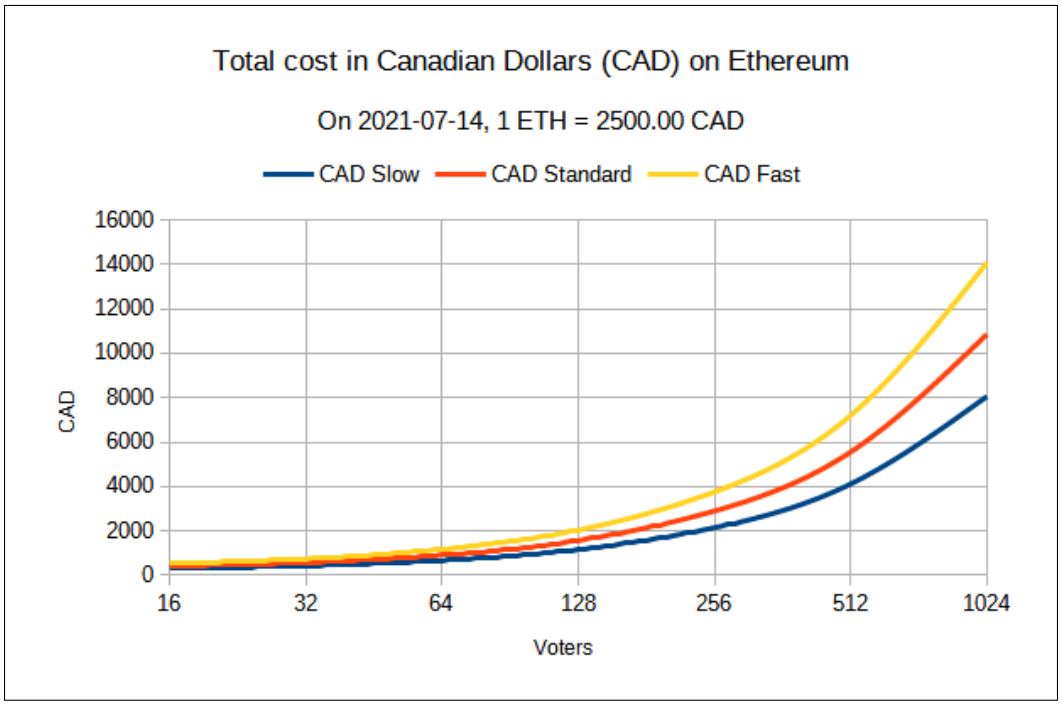

Figure 6.5 The total cost in CAD for running elections on Ethereum

Ethereum is also the most decentralized of all the blockchain options that were used to host the smart contract that runs the elections. Future upgrades to the network promise to drastically reduce the gas cost in Ethers, which is actually what currently makes it expensive, as it can be seen in Figure 6.6. After Ethereum 2.0 is released, the data presented in Figure 6.6 is expected to drastically change.

In contrast, after Ethereum 2.0, the gas cost of running the simulation, as seen in Figure 6.7 should remain stable, unless updates make minor changes to the cost of some instructions or the data persisted on the blockchain.

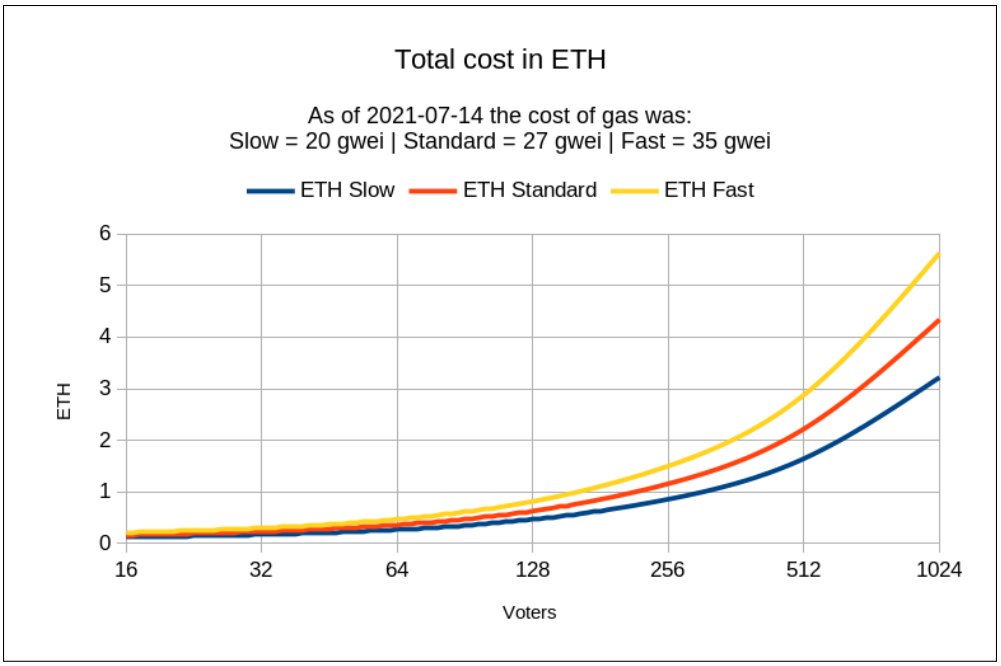

Figure 6.6 The total cost in Ethers for running elections on Ethereum

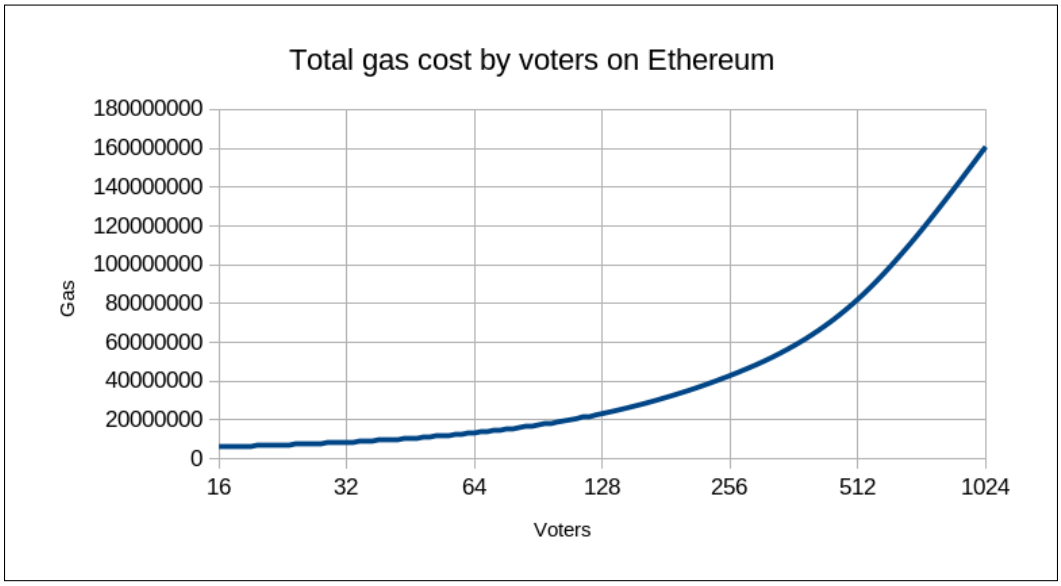

Figure 6.7 The total cost in gas for running elections on Ethereum

6.2 Binance Smart Chain

The Binance Smart Chain is a fork of the Ethereum blockchain and also a cheaper alternative, but it's also a lot less decentralized. The costs of running an election on the BSC are shown in Figure 6.8.

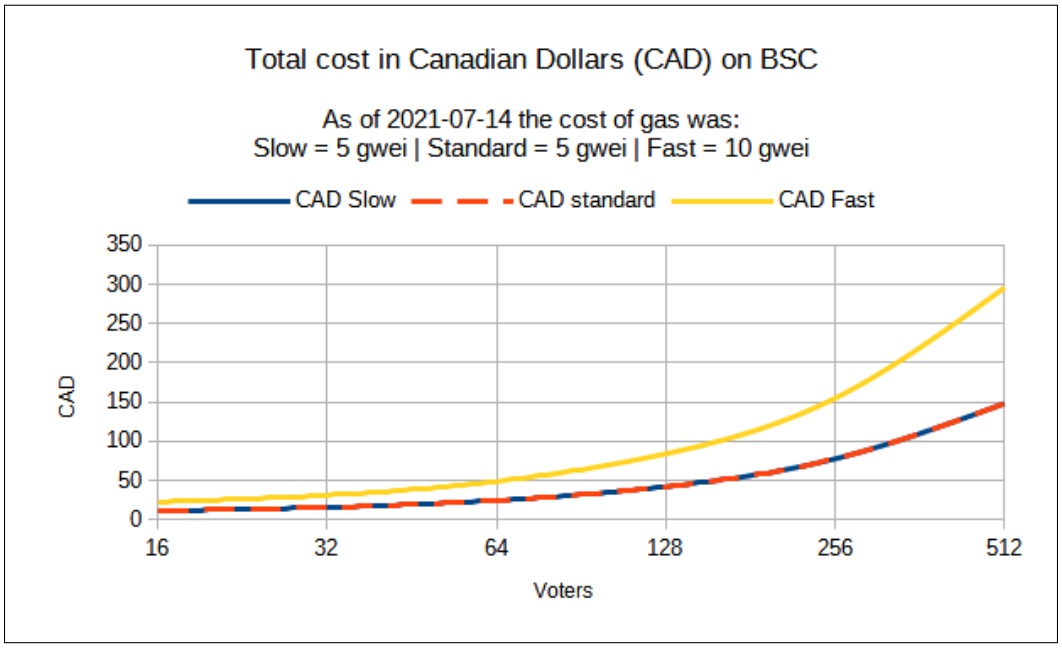

Figure 6.8 The total cost in CAD for running elections on the Binance Smart Chain network

Prices are a lot better than those seen in Ethereum, but they are still elevated to the reality of a voting system for a school.

According to bscgas (2021) "Most of the time, prices are 5 GWei for all speeds. If you came from the Ethereum network, you might find this a little odd. But rest assured, it is just the way BSC works". So, for the simulation configuration, the cost for slow and standard transaction speeds are the same, and the cost for fast is higher because it was observed in the data that sometimes it does go a bit higher.

Figure 6.9 show the cost of running elections in the Binance Smart Chain in its native currency, BNB.

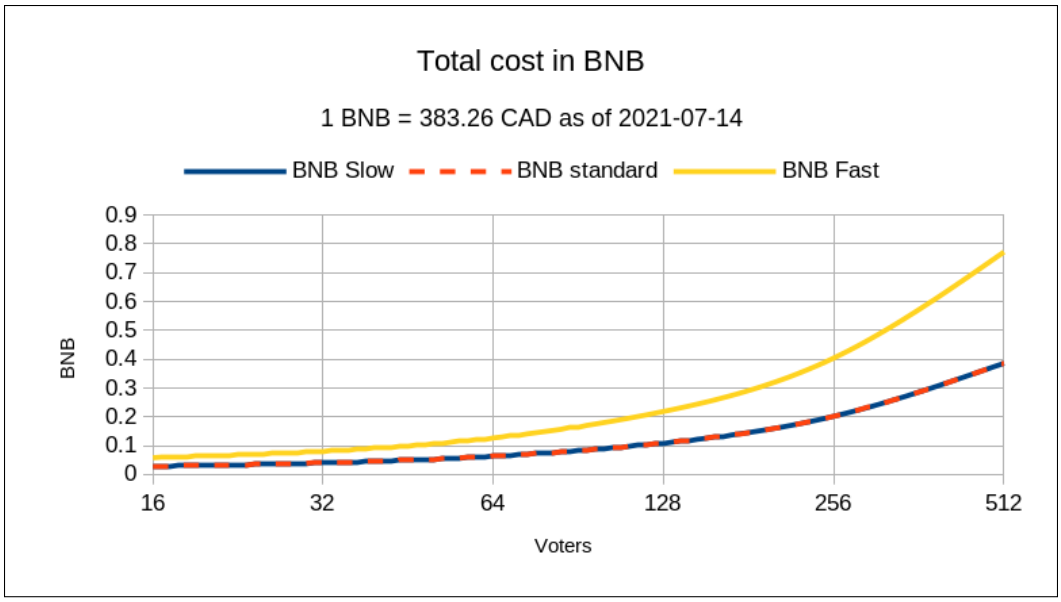

Figure 6.9 The total cost in BNB for running elections on the Binance Smart Chain network

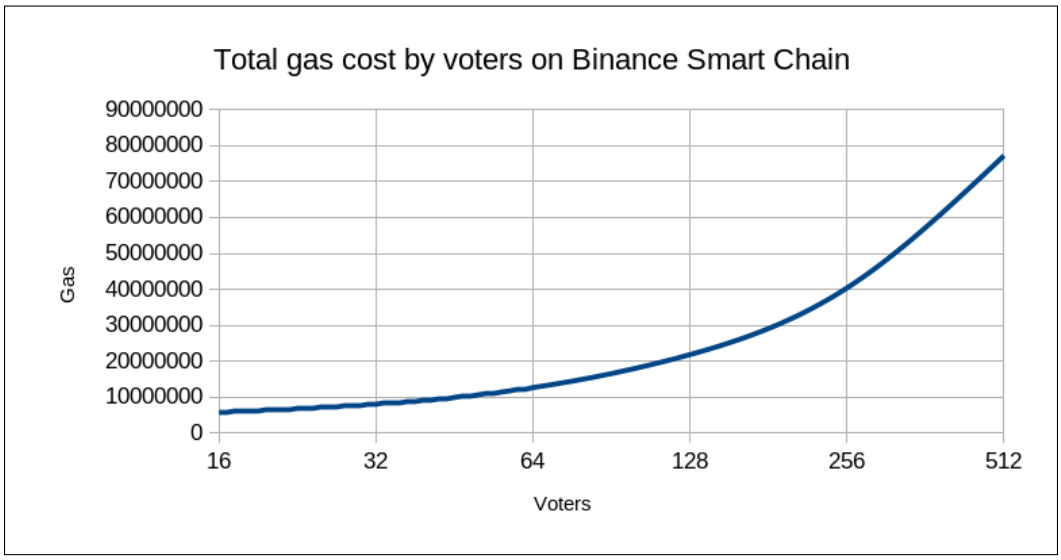

Figure 6.10 The total cost in gas for running elections on the Binance Smart Chain network

Figure 6.10 show the cost in gas of running elections in the Binance Smart Chain, roughly the same gas cost as in the Ethereum network.

6.3 Matic

Matic is a layer two solution, meaning that it's built on top of Ethereum and uses the underlying Ethereum blockchain to provide security, while it adds on top of it speed and low costs. And indeed the costs are lower, Matic was the cheapest of all the blockchains, as it can be seen in detail in Figure 6.11.

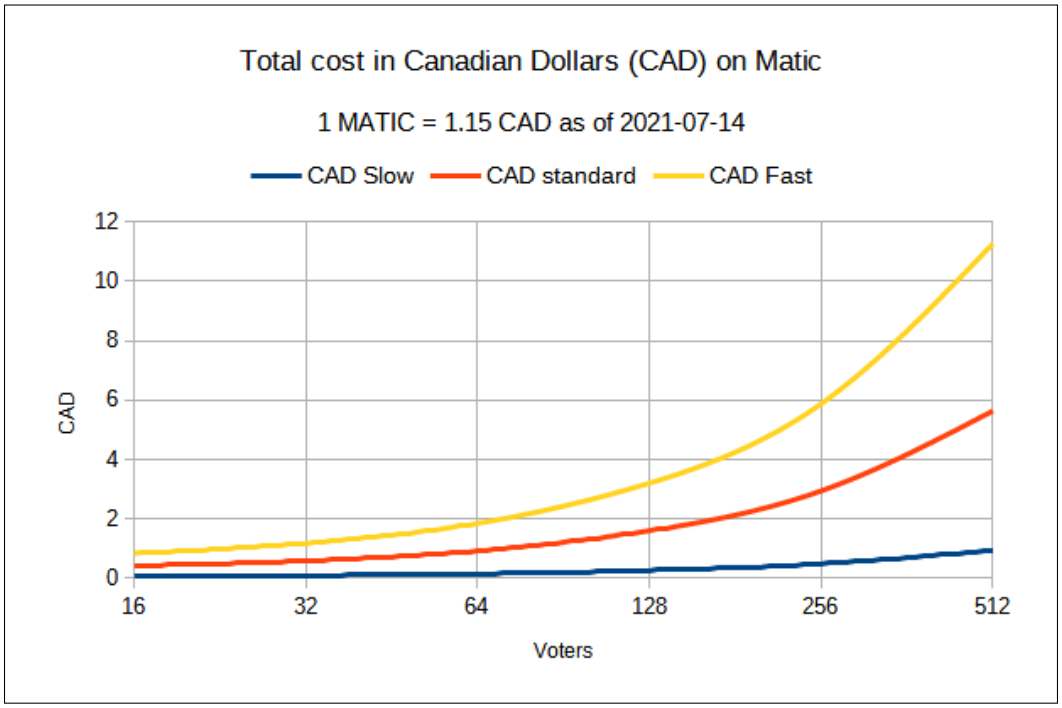

Figure 6.11 The total cost in CAD for running elections on Matic

Prices on the Matic network seem impossibly low when compared to the those on Ethereum or Binance Smart Chain: less than 12 CAD to run them for 512 voters. If this voting system is to be adopted by the AÉÉTS or other kinds of non-official elections, the Matic blockchain would definitely be the one where the system should be deployed.

Official elections of a country, however, would require further analysis to define, as the stakes are a lot higher and the security would need to be deeply studied.

The detailed cost in MATIC can be seen in figure 6.12 and the cost in gas, roughly the same as in the other two blockchains, can be seen in Figure 6.13.

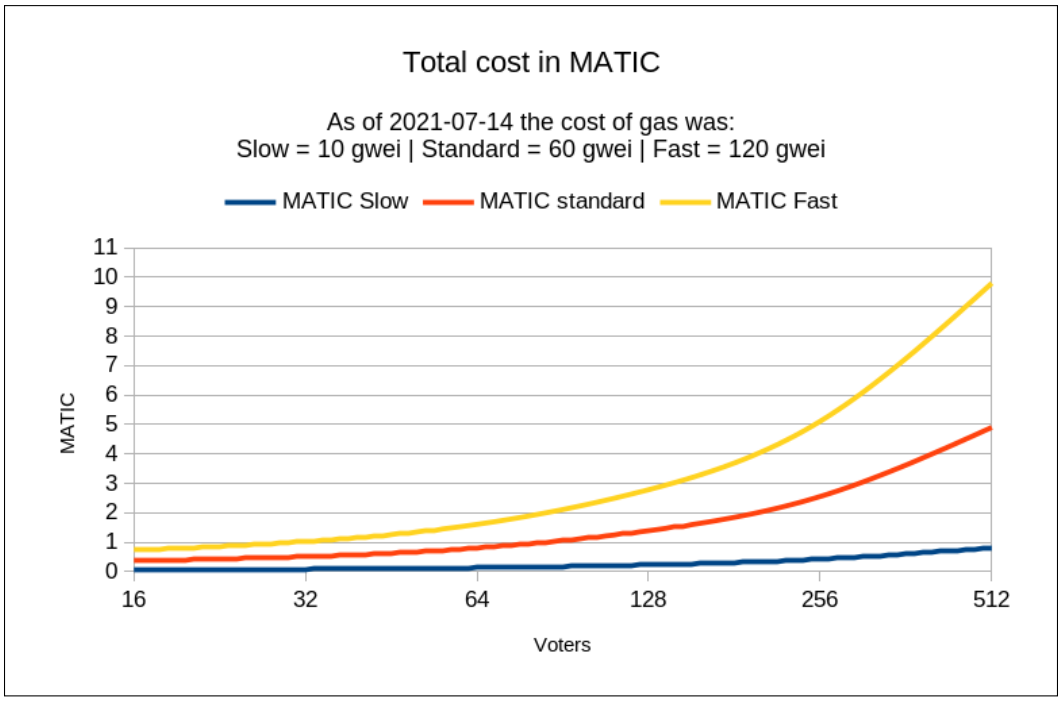

Figure 6.12 The total cost in MATIC for running elections on Matic

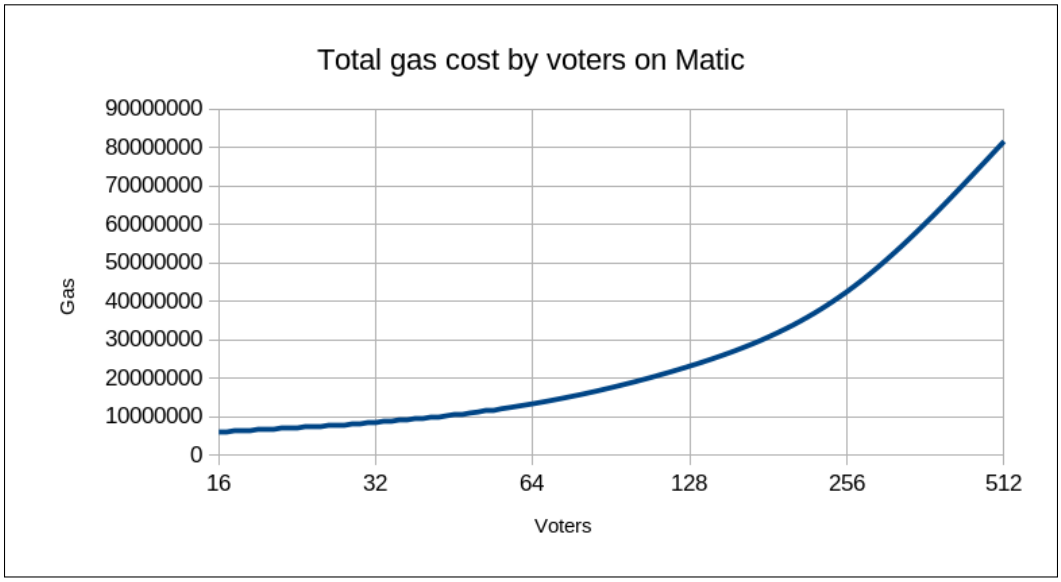

Figure 6.13 The total cost in gas for running elections on Matic

CHAPTER 7

WEAKNESSES OF THE SYSTEM AND POSSIBLE IMPROVEMENTS

The world is always changing, new aspects of life generate new demands, which means that software development is a never-ending process. One can always improve or fix an aspect of a system, and it's pretentious to believe that a system is flawless.

No design is perfect, as software design is wicked problem, meaning that any solution can be improved indefinitely.

This chapter indicates ways in which the present system could be improved.

7.1 Vote obfuscation

Votes must be transparent and separated from the identity of a voter. The main goals to be achieved by this challenge are:

1. Possible to verify that the voter has voted.
2. Possible to verify that the vote of a voter was counted.
3. Impossible to verify which proposal received the voter's vote.

An attempt at developing this was made, however it's a state-of-the-art problem and solving it could be a project as big as this one.

7.1.1 Zero Knowledge Proofs

The key to solving this problem might be Zero Knowledge Proofs(ZPK):

A zero-knowledge proof (ZKP) is a form of cryptography in which an individual can prove to another party that a piece of information is correct without having to disclose the specific piece of information. For example, zero-knowledge proofs can be used to prove to Charlie that Alice sent Bob x amount of money without having to tell Charlie how much money was actually sent.

The implications for ZKP cryptography is vast and it is being used in multiple iterations for public blockchains and cryptocurrencies.

An implementation that could possibly be used is found at https://github.com/meilof/pysnark.

7.1.2 Secure Multi-party Computation

Another possibility could be the Secure Multi-party Computation (SMC) technology, in which a vote is encrypted by multisig (multiple signatures) of all the voters using a TEE trusted execution environment. Some implementations are:

- SGX from intel;
- Memory encryption technology from AMD;
- TrustZone from ARM.

7.1.3 Code obfuscation

Having private keys inside the smart-contract codes would completely eliminate the human element necessary to count votes. This is the ideal case.

However, for this to work, a code obfuscating mechanism would have to be used. It's possible to think of such obfuscator as a way to "encrypt" a program, so that the encrypted program would still run the same as it's non-obfuscated original.

According to Buterin (2019), this is still an open problem in which slow progress has been made from 2014 to 2019.

As Buterin (2019) puts it: "The holy grail is to create an obfuscator O, such that given any program P the obfuscator can produce a second program O(P) = Q such that P and Q return the same output if given the same input and, importantly, Q reveals no information whatsoever about the internals of P. One can hide inside of Q a password, a secret encryption key, or one can simply use Q to hide the proprietary workings of the algorithm itself."

In other words, the challenge is to find a way to "encrypt" a program, so that the encrypted program would give the same outputs for the same inputs as a non-encrypted program. The difference being, it would not be possible to see the internals of such program, as they would be hidden by the encryption, and it could be used to encrypt private keys to sign transactions, for example.

Sadly, this is indeed a hard problem, and all the advancements are still far from being able to create something secure and that can be used, but if progress is made, it could be used in a voting system to greatly reduce the centralization aspects that it still needs to, for example, store the voter's data in a centralized database to make usage of the system more convenient.

7.2 Election period

The time constraints parameters of an election was not included during the simulation runs, with the intention to make each simulation execute faster. It takes around 20 minutes without this constraint.

The code related to this constraint is present but commented out, meaning that before a deployment for production, they could simply be activated.

7.3 Graphical user interface and user identity verification

The main weakness of the system is that it has no graphical user interface. However, it's only a matter of implementation, since a complete design is presented in section 5.5.

An attempt of implementation was made by the author, however, mobile development with Flutter is a world of its own, and it was concluded that such implementation could be a project of its own.

A voluntary bachelor's student interested in Flutter development tried to help with the project, but it was proven that they simply didn't have the time to dedicate for such a big task.

The hardest part of this module would be the user identity verification, which is a huge topic in computer vision and machine learning, and presently it seems that there are only theoretical papers about it, but no open source implementation done in Flutter's Dart.

7.4 Email notification module

A notification module that would send emails to the voters registered in the system with status updates about elections, such as when a new election is going to be held or the outcomes of finished elections.

Such module doesn't exist in the current system, but the API is already capable of sending emails, so it would be a matter of developing the specifics of the module and integrating it with the rest of the system.

7.5 Email verification

The email verification process is done using an external API that runs on a centralized non-distributed server.

Ideally, all "server-side" code should run on the Blockchain, but this is not possible because smart contracts are not capable of receiving or sending emails.

This is because smart contracts run in an isolated environment (the Ethereum Virtual Machine) which has no connection with external networks. Therefore, it's not possible to implement the standard email protocols, such as SMTP or pop3, with smart contracts.

The closest thing possible would be reading from transactions as one was receiving emails, and publishing data to the Blockchain so that an observer service could actually send an email.

So, this part of the authentication process could be improved by being deployed on a decentralized and distributed platform.

7.6 Limits of the smart contract

The maximum number of voters of an election is currently capped by the number of votes that can be counted in a single transaction.

This is done by the function count_votes, presented in subsection 5.2.1.6.

Each blockchain has a limit of gas for each new block:

The Matic mumbai testnet gas limit of a block was exceeded when counting the votes of an election with 1024 voters. The error message says:

```
{'code': -32000, 'message': 'gas required exceeds allowance
   (20000000)'}
```

For the Binance Smart Chain testnet the gas limit was also reached when trying to count the votes of 1024 voters:

```
{'code': -32000, 'message': 'gas required exceeds allowance
   (25000000)'}
```

For the Ethereum Goerli testnet the limit was reached when trying to count the votes of 2048 voters:

```
{'code': -32000, 'message': 'gas required exceeds allowance
   (29999972)'}
```

It's possible to solve this problem by making the count of votes work in batches. So one call would count, for example, 256 votes, the next call would count more 256 votes and so forth.

Also, it's not a good idea to use the whole block gas limit, because that would mean that only a single transaction would occupy a whole block.

On the busier main networks, this would mean having to pay an extra for gas fees, to make sure the transaction gets added to a block with priority over all other transactions waiting to be added to a block.

7.7 Independent audit of the system

The results of elections or referendums can have great impact in the lives of people. Therefore, the system must be robust and secure. For this reason, an independent audit of the code and the infrastructure is welcomed to make sure the quality is high enough for it to be used in the real world.

7.8 Run own nodes of the target blockchains

To communicate with the blockchain platforms where the system was deployed, API services such as Infura (https://infura.io/), which provides the ability to interact with a blockchain without running a full node of such blockchain.

This solution is OK for running simulations, but it adds a middle entity that requires trust, increasing the centralization of the system.

There are also limitations to the number of requests that can be made to the API per day, which can be increased if the product is purchased or subscribed to. There is also a risk that the limit of requests for the free plan decreases to a level that is not acceptable for a real world deployment.

For these reasons, the system should have its own node deployed, so it can talk directly to the blockchain platform of choice for a real world deployment.

7.9 Is a perfect electronic voting system possible?

This section is the author's free speech in a discussion about the possibility that a perfect electronic voting solution could exist. This is done by an attempt at conceptualizing such perfect system and then trying to evaluate whether the requirements are possible or not possible.

A perfect solution for electronic voting should satisfy the two principles of an election:

1. Anonymity of the voters;
2. Trust in the system.

If these two principles are mutually excluding, it would be impossible to implement a perfect solution.

Blockchain systems are extremely decentralized. However a platform like Ethereum 1.0 that uses proof of work and Ethereum 2.0 uses Proof of Stake can give room to the argument that they are not decentralized enough, because power can be given to certain groups of people, even though the idea is to be decentralized.

For example, someone who has a lot of money can buy a lot of hardware and pay electricity to be used by proof of work mechanisms, or who has a lot of money can simply stake a lot to participate in proof of stake mechanisms.

So, an ideal system should ask the question "how much decentralization is enough?" and "Is there alternative consensus mechanisms that are more democratic?".

One of the ways of tampering with elections is to cast votes from fake people, making the vote from someone that doesn't exist be counted.

To counter this, a perfect voting system should consider how to validate the identity of voters so that there is absolutely no doubt that they are real.

The questions "is someone real?", "is someone alive? or "is someone a person?" can be very philosophical, but the more science put into answering it, the better.

So, when talking about user identification by biometric data, it's important to keep in mind that a finger can be cut and then used on a fingertip sensor, an eye can be removed and its iris shown to a camera, a face can be shown from a picture or even generated by some powerful AI, etc.

The image of the person could be compared to a picture database to determine if it's someone that can vote. This process could also be used to make sure that the same person would not authenticate and cast votes twice by authenticating as someone else. But all of this would raise ethical concerns related to people's privacy and personal data.

There is also the case of identical twins, which should be studied with care.

Therefore, a perfect voting system should think of all possible ways in which user validation could be tampered with. Everything that is available should probably be used, such as not only using visual data, but also voice signature data.

Another big problem is the lacking aspect of the ritual of voting when voters can simply cast their votes from a smartphone. So, making a person say a certain phrase and use it to prove its identity, would also tackle the problem of lack of ritualization of an electronic voting system.

Maybe it's a good idea to have the voters, that are capable of speech, say out loud: "I understand that voting can have a direct impact on my life and in the life of the others around me. I am in a sane state of mind and I am not being forced to do this by anyone. I am not involved in any kind of activity trying to tamper with the voting to decide who will be president of the imaginary barrier that I was taught to call my nation".

It would also have to be considered the case of people that are mute.

Also, a perfect voting system should be completely automated, to totally remove any individual authority from it, so it's a challenge to do automated analysis to determine whether the person is real, alive, well and not being forced to vote.

So the authentication process should be ritualistic and formal, as once someone is authenticated, they can cast a vote, that can't be changed and that the decision of the vote is not visible to anyone, not even the voter.

7.9.1 Idea for decentralization: Proof of identity consensus mechanism

This idea is intended to maximize the principle of trust in the system.

Starting from the fact that it's arguable that Proof of Work or Proof of Stake can benefit the richest, a mechanism called "Proof of identity" is proposed, and its main idea is to allow each voter that participates in an election to contribute to the consensus mechanism with their device.

In other words, the idea is to have someone who votes using their phone, being able to use that same phone to count the votes. The power of each voter would be the same in such system.

Still, it would leave room to argue that voters could still gather together to try to change the results, but that's exactly what they do with their votes already, so it seems that it would not matter.

In "proof of identity", the counting would be done by everyone that participates in the voting. All votes would be counted by everyone as long as they can be identified as a voter in the voting.

Another way of seeing it, is that instead of deploying a voting system on a blockchain, the system would be a blockchain of its own, formed by the devices of the voters.

It could be argued that this mechanism is the one that provides the highest degree of decentralization, because no voter would have a higher influence than any other with their vote and with their contribution to the consensus mechanism that would be used for counting the votes.

7.9.2 Idea for anonymity: encrypted blocks

This idea is intended to maximize the anonymity of the voters.

The blockchain makes a secret question that only a specific voter can read. This would only be visible on the voter's device and during the moment of voting.

If there are three candidates the blockchain would assign them to different encrypted data blocks, nobody would be able to know which block corresponds do which candidates.

The voter then, through a ZKP (Zero Knowledge Proof) proves that they agree with one of these data blocks, meaning it has votes on that block.

So, the information of what proposal received the vote is never revealed because what is stored in the blockchain, the vote, is actually a ZKP that chooses a block of data containing the target of the vote.

However, like many problems in security, the snake ends up eating its own tail. This idea seems to hit a dead end when considering the fact that a private key would be needed to decrypt the encrypted data block.

7.9.3 A simpler idea for anonymity: Enough anonymity

This idea is another intended to maximize the anonymity of the voters.

The idea consists in making voters participate in the counting process by receiving votes from other voters. It can be seen as a simplified version of proof of identity.

It's probably not ideal for official elections, such as those held by nations, but it could be used for small elections.

It's a protocol to conduct the voting and counting of voting, in which some voters will know the votes of some other voters.

The motivation for this idea is the fact that in the encrypted blocks idea we would meet a dead end when needing to decrypt some private key to decrypt the encrypted data blocks containing votes.

In this case, each voter would keep their private key, that would be used to decrypt the votes from other randomly assigned voters.

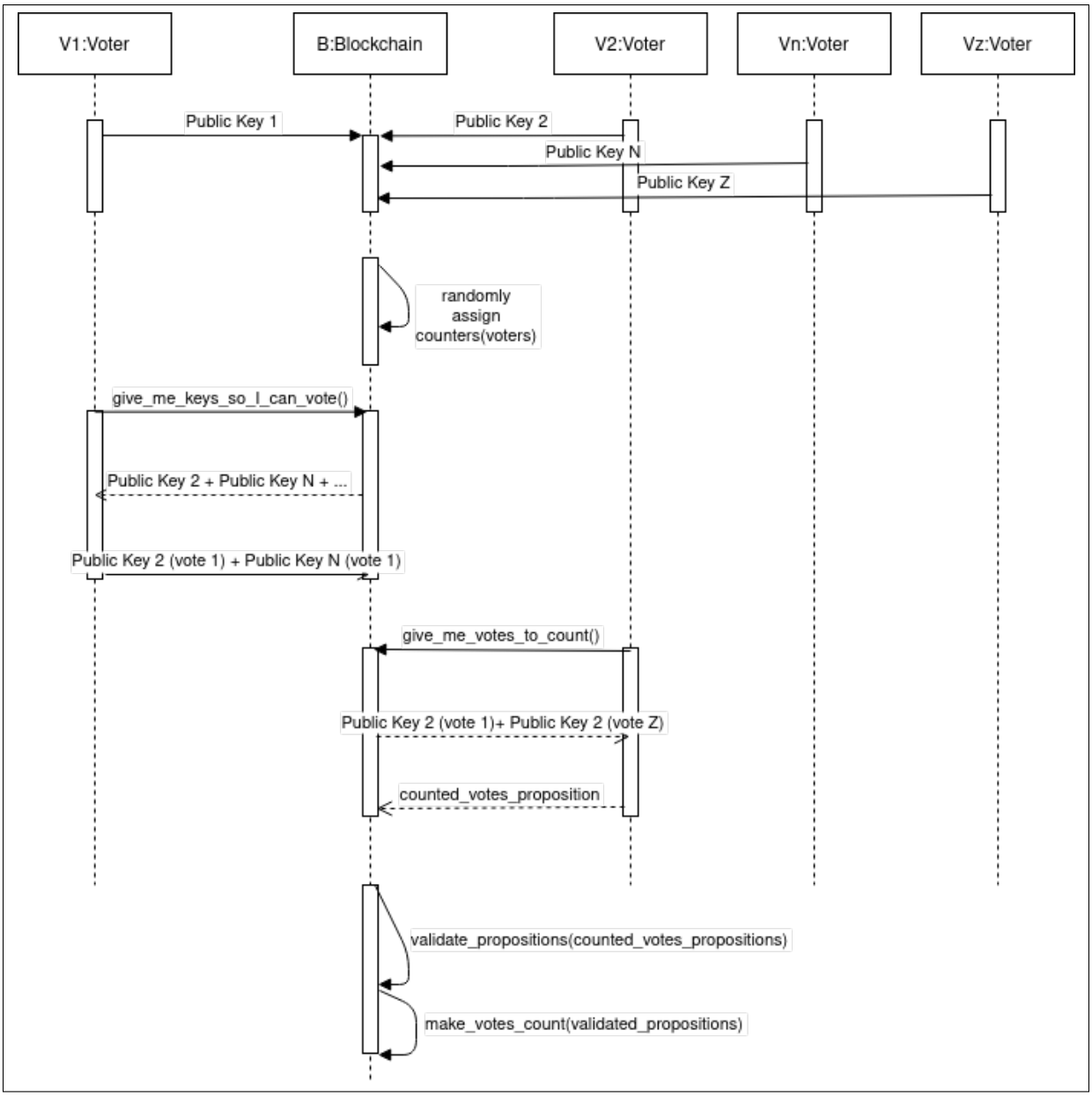

Figure 7.1 Sequence diagram for the proposed protocol "Enough anonymity"

As it can be seen in figure 7.1, a voter V1 could receive a ’Public Key 2’ and a ’Public Key N’ from voters V2 and Vn, respectfully, and use those keys to encrypt it’s vote ’vote 1’ and send it to the blockchain.

Voters V2 and Vn would receive ’vote 1’ encrypted with their public keys as long as other votes to count. All the votes would be counted together, and the result count is a proposition sent to the blockchain.

The smart contract in the blockchain validates all the propositions, a consensus mechanism, and finally makes the votes count from the validated propositions.

Each private key necessary to decrypt a vote will be shared with X random vote counters. This value should be big enough as to provide consensus, and not too big as too void the anonymity of a voter. In other words, a vote counter will receive many votes to count at the same time, and then send to the blockchain a count proposition, with a list of who those votes belong too.

This is to be used by a smart contract to count the votes without breaking anonymity, and also validate the consensus of the voters. A cleaver mechanism will be used, raid 5 style, to make sure that the counters were honest, and also make the system safe against some certain number of entities.

Currently in some official elections, a box with votes is only to be opened in the presence of a certain number of people, that includes people of all the political parties involved. Electronic voting will allow for many more people to "be there when the box is going to be open", and they would also all be randomly selected, eliminating the trust factor is so many humans that are necessary to make these paper voting systems work.

There could be a private key associated with each voter, stored in the blockchain encrypted by the public key of the voters that will cast it.

It’s also at this moment that the decryptors will be randomly selected. Also, it’s possible that at this moment the voter will be randomly assigned some votes to decrypt. A clever mechanism should be put in place to balance the number of assignments for the decryptors, making the early voters receive more assignations. It’s easy to do, as this assignment will be kept in the blockchain with all the public keys of voters.

Voting could be done in a day, or in a certain period, and counting would happen after that period is over. The counting performance should be well analyzed, but it should also run for a certain period. After counting by the participants is done, then it's time for the blockchain to run the smart contract code that will validate all the votes and effectively make them count in the blockchain.

CHAPTER 8

ABOUT THE DEVELOPMENT OF THE PROJECT

An Agile development methodology was followed. Since only one person worked on the project, it was done in sprints.

Test driven development was also used, as the simulator module was implemented before the smart contracts.

8.1 Tasks

According to the adopted Agile approach. The steps to conclude this project were the following:

- General research on elections and blockchain
- Project idea document
- More general research on elections and blockchain
- Project development plan document (specification)
- Initial design
- State-of-the-art research (scientific documentation and existing systems)
- Review of the document project plan document
- Refined design
- Implementation
  - Centralized infrastructure (API and relational database)
  - Simulator module
  - Smart contracts
  - Attempt at developing the mobile and web app
- Deployment of the smart contract on Ethereum
- Testing and collection of resulting data
- Review of the implementation and optimizations
- Deployment of the smart contract on Binance Smart Chain
- More testing and collection of resulting data

- Another review of the implementation and improvements to the log data
- Deployment of the smart contract on Matic
- Work on the report
  - First version of the text
  - Elaboration of a presentation
  - Production of the demo videos in English
  - Review of the deployment instructions for each module
  - Final version of the text
- Publication of the repositories
- Present the project to the AÉÉTS, ask if they are interested in finishing development with the collaboration of the author

8.2 Stack, tools and technologies used

Blockchains:

- Ethereum;
- Binance Smart Chain;
- Matic.

Editors:

- Sublime Text for everything but the report;
- Overleaf for the report;
- Deepl.com to assist with translations;
- Antidote to assist with the quality of the English;
- Diagrams.net for the diagrams;
- LibreOffice Calc for the cost analysis;
- Remix for quickly testing smart contract code.

Programing languages:

- Vyper for the smart contracts.

- Python for the API with Flask framework and Werkzeug.security library.
- Python for the Relational Database with Alembic and SQLAlchemy.
- Python for the simulator module with web3.
- LaTeX for the report with the ÉTS template for synthesis projects.
- SQL for the Relational Database.
- Bash for the deployment of the centralized modules and also to write a bot that would get Matic from the Matic Mumbai faucet.
- Dart for the Flutter module.
- Markdown for the documentation of the repositories of the modules.

Server infrastructure:

- A home server running Arch Linux where the centralized infrastructure was deployed.
- Nginx and uWSGI for the API.
- Let's Encrypt as Certificate Authority for enabling HTTPS on the API.
- Noip.com for the DNS of the API.
- PostgreSQL for the Relational Database.
- SSH for interfacing with the server machine.

Workstation: Running Manjaro Linux and Windows 10

Version control: Git, coupled with the open-source tool GitAhead.

Deployment of smart contracts: Truffle.

Rerefences: Ieeexplore.

8.3 Challenges overcame and lessons learned

8.3.1 Learning curve of blockchain systems

At first, checking the different blockchain implementations available was overwhelming, because it's hard to keep track of an industry that is advancing so fast and is already very abstract for the great majority of people.

Many blockchains were considered before the decision of doing it for Ethereum. Just Ethereum has many "competitors" such as Cardano, Polkadot, EOS, DFINITY or the NEAR Protocol.

They don't differ only in the technological implementation of the blockchain implementation, but also on governance models or ethical and social differences that are important for a voting system that has an impact on society.

Another factor that contributed to this challenge is the knowledge that many intelligent people, including big organizations like governments, have been trying to build the perfect election system using blockchain. This made the project seem too ambitious, but the current project has value because voting is just complex and has many different contexts, such as the one chosen by this project, which is for the elections of the school's students council.

Nowadays, the author is very happy to realize that their perception of blockchain technologies is that they are easy and pleasant to work with. A lot of critical thinking has been developed about the subject.

8.3.2 Impact of the Covid-19 pandemic

The context of being a full-time master student, working part time, in a foreign country in which you arrived shortly before the pandemic started was hard. Especially during the second wave of Covid, in which isolation was too extreme, and the grandfather of the author passing away from Covid posed the most serious challenge faced during the development of the project.

To deal with mental health, pauses on the development were taken to focus on self-care. Overcoming this challenge makes the author feel more emotionally mature.

8.3.3 Maturity of the technologies used

Development often suffered from delays because one can find themselves trying to deal with a problem that is actually not a design flaw or a bug, but rather a bug within something of the ecosystem, for example Remix would often crash for no apparent reason.

Vyper 0.2.8 was used for the development of this project. It's not even in an official version 1. The documentation was at times lacking.

One problem faced was that there was absolutely no mention in the documentation that multidimensional lists declaration is inverted, compared to how it's implemented in Python.

Since it's an open source project, an issue was opened, and the problem was identified: it's not really a bug in Vyper, but rather a lack of documentation to explain that multidimensional lists declaration follows how it's in Solidity. Then, a pull request was made to have the documentation updated with clear instructions: https://github.com/vyperlang/vyper/pull/2368.

Another problem faced when working with Vyper is that array maps are very difficult to work with: When sending transactions in parallel, it's not possible to know beforehand how much a transaction using one will cost, because it changes based on the input or the output.

The Cardano blockchain is suggested as a better way of dealing with costs, because it promises determinism regarding the cost of transactions.

However, the problem was worked around by simply not using array maps when it was possible.

Another problem is that for the moment Ethereum doesn't have a framework for developing mobile apps, like the ones listed in available at https://ethereum.org/en/developers/local-environment/.

Yet another problem related to the maturity of the technologies was faced with Matic. The URL for the testnet was changed from https://rpc-mumbai.matic.today to https://rpc-mumbai.maticvigil.com but the documentation was not updated. Luckly, someone made a post on a public forum to document that they had the same issue and found the new URL by chance.

8.3.4 Reduce costs and scale

A big issue found was the fact that after running a simulation with the first version of the smart contract it indicated that a simple election for a few dozens of people would costs hundreds of dollars to run in the Ethereum platform.

Scaling was terrible, as running an election or referendum for a few hundreds of people would increase costs to the thousands.

Upgrades to Ethereum, such as the upcoming EIP1559 or the move to Ethereum 2.0, were seen as a hopeful solution to the cost problem.

However, after further research, it was clear that one should not depend on these upgrades, as they are not a total guarantee that the costs will decrease, by what factor, and also what is the date in which they will be released.

Then, the next solution started being researched, which was optimizing the smart contract, like it was described in subsection 5.2.2, and it was found out that the improvements in cost did not justify the decrease in quality of the code.

So, as a final possible solution for the issue, other blockchain systems started being looked up to host the system.

Some of these platforms can be way cheaper, but also way less decentralized.

A really good candidate for a voting system is the Cardano blockchain. It's supposed to be cheaper to use, compared to Ethereum, and it also has a functional programing language, Plutus, to write smart contracts in, that also promises determinism.

Non-determinism was another problem faced when working with Vyper, because some transactions to a same smart contract function would have different costs based on numerous factors. However, this problem could be solved by using different programing techniques.

Still, Cardano was a really good candidate, but the contract ended up not being ported to the platform because at the time of writing Cardano still had not released the capacity to run smart contracts on the platform, and also Plutus has a steep learning curve.

Cardano is still seen as an ideal candidate for future works.

Finally, the best solution found was to deploy the smart contract on the Binance Smart Chain and later on the Matic platforms. Both of these run code with the same interpreter as Ethereum, meaning that the code of the smart contract would not need to be rewritten to be deployable on these blockchains.

And that's how the journey of reducing costs reached the excellent results shown in Chapter 6.

CHAPTER 9

CONCLUSION

In this project a voting system using blockchain was developed.

Voting and blockchain were conceptualized and contextualized.

A simulation was developed and ran many times in different environments to produce the results shown in this project.

The results obtained with the Matic blockchain make it clear that this system could be used in the real world to conduct elections or referendums for the Student Association of the École de Technologie Supérieure (AÉÉTS), because it works, it's cheap and definitely better than the current solution being employed.

However, for this to happen, the development of the system should continue, so it can receive a graphical user interface.

A graphical user interface would need to be its own project, as the analysis concludes that it would be the most vulnerable part of the system, from a security point of view.

Another challenge for the graphical user interface would be the identity verification of voters in the system.

It was a huge pleasure to work with the blockchain technology because it requires a broad skill set and is such a disruptive technology, capable of quickly changing society.

All the objectives were achieved:

- A functional voting system using blockchain was developed;
- A complete software engineering was realized;
- The whole technological stack used was open source and all of the source code for the project will be published once it's approved.

BIBLIOGRAPHY

Al-madani, A. M., Gaikwad, A. T., Mahale, V. & Ahmed, Z. A. T. (2020). Decentralized E-voting system based on Smart Contract by using Blockchain Technology. *2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)*, pp. 176-180. doi: 10.1109/ICSIDEMPC49020.2020.9299581.

Anadiotis, G. (2017). How to use blockchain to build a database solution. Consulted on 2020-12-07 at https://www.zdnet.com/article/blockchains-in-the-database-world-what-for-and-how.

B, S., V, R. T., Krishna M P, N., J, B. R., Arvindh M, S. & Alagappan, D. M. (2019). Secured Electronic Voting System Using the Concepts of Blockchain. *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 0675-0681. doi: 10.1109/IEMCON.2019.8936310.

Bartolucci, S., Bernat, P. & Joseph, D. (2018). SHARVOT: Secret SHARe-Based VOTing on the Blockchain. *2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pp. 30-34.

Behlendorf, B. (2018). Can We Digitize the Voting System? Blockchain, Corruption, and Hacking. Consulted on 2020-11-14 at https://www.youtube.com/watch?v=KpvVqa-O_Tg.

Bellini, E., Ceravolo, P. & Damiani, E. (2019). Blockchain-Based E-Vote-as-a-Service. *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pp. 484-486. doi: 10.1109/CLOUD.2019.00085.

BigchainDB. (2020). Features and Use Cases - Blockchains Meet Big Data. Consulted on 2020-12-05 at https://www.bigchaindb.com/features/,acessed.

Boucher, P. (2017). How blockchain technology could change our lives. *European Parliamentary Research Service*, 1–28. Consulted at https://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_IDA(2017)581948.

Boucher, P. (2018, 08). 3 key questions on Blockchain voting. Consulted on 2020-11-17 at https://www.youtube.com/watch?v=2rgbOv_ab4c.

Brew, T. (2019). Blockchain in Voting. Consulted on 2020-08-30 at https://www.youtube.com/watch?v=d0iLN8LDJ8g.

Brock, T. (2017, 11). Blockchain, Voting and Elections: What's the Problem? Consulted on 2021-07-29 at https://www.youtube.com/watch?v=L4zR6mTUPlk.

bscgas. (2021). BSC Gas Price Forecast. Consulted on 2021-07-22 at https://bscgas.info/.

Buterin, V. (2019, 11). Hard Problems in Cryptocurrency: Five Years Later. Consulted on 2020-12-07 at https://vitalik.ca/general/2019/11/22/progress.html.

ConsenSys. (2021). gnark "gnark has not been audited and is provided as-is, use at your own risk. In particular, gnark makes no security guarantees such as constant time implementation or side-channel attack resistance.". Consulted on 2021-04-01 at https://github.com/ConsenSys/gnark.

DB-Engines. (2020, 12). DB-Engines Ranking. Consulted on 2020-12-04 at https://db-engines.com/en/ranking.

DevTeam.Space. (2020). How to Use Blockchain to Build a Scalable Database. Consulted on 2020-12-07 at https://www.devteam.space/blog/how-to-use-blockchain-to-build-a-scalable-database/.

Dhameja, G. (2017, 09). Role Based Access Control for BigchainDB Assets. Consulted on 2020-12-14 at https://blog.bigchaindb.com/role-based-access-control-for-bigchaindb-assets-b7cada491997.

Dhameja, G. (2018, 02). Role Based Access Control for BigchainDB Assets. Consulted on 2020-12-14 at https://www.youtube.com/watch?v=QOYRVgz4qLw.

Divante. (2020). Vue Storefront PWA. [Online; accessed 10-August-2020].

Documentation, A. C. (2020). Guarantees. Consulted on 2020-12-04 at https://cassandra.apache.org/doc/latest/architecture/guarantees.html.

documentation, B. (2020a). Queries in BigchainDB. Consulted on 2020-12-05 at https://docs.bigchaindb.com/en/latest/query.html.

documentation, B. (2020b). Tutorial: Role-based access control in BigchainDB. Consulted on 2020-12-14 at https://www.bigchaindb.com/developers/guide/tutorial-rbac.

Doost, M., Kavousi, A., Mohajeri, J. & Salmasizadeh, M. (2020). Analysis and Improvement of an E-voting System Based on Blockchain. *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, pp. 1-4. doi: 10.1109/ICEE50131.2020.9260875.

Durden, T. (2020, 08). The USPS Just Filed A Patent For A Blockchain-Based Secure-Voting System. Consulted on 2021-07-29 at https://www.zerohedge.com/crypto/usps-just-filed-patent-blockchain-based-secure-voting-system.

Garg, K., Saraswat, P., Bisht, S., Aggarwal, S. K., Kothuri, S. K. & Gupta, S. (2019). A Comparitive Analysis on E-Voting System Using Blockchain. *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pp. 1-4. doi: 10.1109/IoT-SIU.2019.8777471.

Georgiev, G. (2021, 03). Ethereum Could Scale 100x in a Few Months, Says Vitalik Buterin. Consulted on 2021-07-29 at https://cryptopotato.com/ethereum-could-scale-100x-in-a-few-months-says-vitalik-buterin/.

Goswami, Dhananjay, Lagneaux, M., A., Venkataraman, Swaminathan, M., Henry, Wendy, Shrestha, Aashish, Dearing & M., S. (2020, 08). SECURE VOTING SYSTEM. Consulted on 2020-11-14 at http://appft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=/netahtml/PTO/srchnum.html&r=1&f=G&l=50&s1=20200258338.PGNR.&OS=&RS=#top.

HarryR. (2019). EthSnarks "doesn't seem to be maintained anymore" - "WARNING: EthSnarks is beta quality software, improvements and fixes are made frequently, and documentation doesn't yet exist". Consulted on 2021-04-01 at https://pypi.org/project/ethsnarks/.

Jacobsen, H.-A., Sadoghi, M., Tabatabaei, M. H., Vitenberg, R. & Zhang, K. (2018, 12). Blockchain Landscape and AI Renaissance: The Bright Path Forward. *Proceedings of the 19th International Middleware Conference Tutorials*, (Middleware '18), 1. doi: 10.1145/3279945.3279947.

Kaleem, M., Mavridou, A. & Laszka, A. (2020). Vyper: A Security Comparison with Solidity Based on Common Vulnerabilities. 1–7. Consulted at https://www.researchgate.net/publication/339997773_Vyper_A_Security_Comparison_with_Solidity_Based_on_Common_Vulnerabilities.

Khandelwal, A. (2019). Blockchain implimentation on E-voting System. *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 385-388. doi: 10.1109/ISS1.2019.8907951.

Korth, H., Sudarshan, S. & Abraham Silberschatz, P. (2010). *Database System Concepts*. McGraw-Hill Education. Consulted at https://books.google.ca/books?id=re4YQAAACAAJ.

Kshetri, N. & Voas, J. (2018). Blockchain-Enabled E-Voting. *IEEE Software*, 35(4), 95-99. doi: 10.1109/MS.2018.2801546.

Kuznetsov, N. (2020, 08). Ethereum 2.0 and EOS Crossing Swords Over Scalability Supremacy. Consulted on 2021-07-22 at https://cointelegraph.com/news/ethereum-20-and-eos-crossing-swords-over-scalability-supremacy.

Lai, W., Hsieh, Y., Hsueh, C. & Wu, J. (2018). DATE: A Decentralized, Anonymous, and Transparent E-voting System. *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, pp. 24-29. doi: 10.1109/HOTICN.2018.8605994.

Legislature, T. A. S. (2021). 04/15/2021 03:30 PM STATE AFFAIRS. Consulted on 2021-07-22 at https://www.akleg.gov/basis/Meeting/Detail?Meeting=SSTA%202021-04-15%2015:30:00#tab2_4.

Major, J. (2021). Alaska is the first US state on a course to use blockchain in voting statewide. Consulted on 2021-07-22 at https://finbold.com/alaska-is-the-first-us-state-on-a-course-to-use-blockchain-in-voting-statewide/.

Martens, D. & Maalej, W. (2018). ReviewChain: Untampered Product Reviews on the Blockchain. *arXiv e-prints*, arXiv:1803.01661.

McCorry, P., Shahandashti, S. F. & Hao, F. (2017). A Smart Contract for Boardroom Voting with Maximum Voter Privacy. *Financial Cryptography and Data Security*, pp. 357–375.

meilof. (2021). PySNARK "PySNARK is experimental and not fit for production environment.". Consulted on 2021-04-01 at https://github.com/meilof/pysnark.

Miles, C. (2017). Blockchain security: What keeps your transaction data safe? Consulted on 2020-12-07 at https://www.ibm.com/blogs/blockchain/2017/12/blockchain-security-what-keeps-your-transaction-data-safe/.

Murtaza, M. H., Alizai, Z. A. & Iqbal, Z. (2019). Blockchain Based Anonymous Voting System Using zkSNARKs. *2019 International Conference on Applied and Engineering Mathematics (ICAEM)*, pp. 209-214. doi: 10.1109/ICAEM.2019.8853836.

NDTV. (2020). TikTok App Listings Flooded With 1-Star Reviews Amid Faizal Siddiqui Controversy. [Online; accessed 21-May-2020].

on Tech, I. (2020, 06). Breaking Down ETH 2.0 – Sharding Explained. Consulted on 2021-07-29 at https://academy.ivanontech.com/blog/breaking-down-eth-2-0-sharding-explained.

p. Hjálmarsson, F., Hreioarsson, G. K., Hamdaqa, M. & Hjálmtýsson, G. (2018). Blockchain-Based E-Voting System. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 983-986. doi: 10.1109/CLOUD.2018.00151.

Panja, S., Bag, S., Hao, F. & Roy, B. (2020). A Smart Contract System for Decentralized Borda Count Voting. *IEEE Transactions on Engineering Management*, 67(4), 1323-1339. doi: 10.1109/TEM.2020.2986371.

Patidar, K. & Jain, S. (2019). Decentralized E-Voting Portal Using Blockchain. *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1-4. doi: 10.1109/ICCCNT45670.2019.8944820.

Patrick, J. R. (2016). *Election Attitude: How Internet Voting Leads to a Stronger Democracy*. Attitude LLC.

Pramulia, D. & Anggorojati, B. (2020). Implementation and evaluation of blockchain based e-voting system with Ethereum and Metamask. *2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, pp. 18-23. doi: 10.1109/ICIMCIS51567.2020.9354310.

Raikwar, M., Gligoroski, D. & Velinov, G. (2020, 04). Trends in Development of Databases and Blockchain. *2020 Seventh International Conference on Software Defined Systems (SDS)*, pp. 177-182. doi: 10.1109/SDS49854.2020.9143893.

Rezende, P. (2014, 10). As urnas e o sistema fraudável sem riscos para o fraudador. Consulted on 2020-11-17 at https://jornalggn.com.br/eleicoes/pedro-rezende-as-urnas-e-o-sistema-fraudavel-sem-riscos-para-o-fraudador/.

Sam Richards,Ryan Cordell, P. W. (2020). Consensus mechanisms. Consulted on 2020-12-18 at https://www.bigchaindb.com/developers/guide/tutorial-rbac.

Sanjaya, M. (2019). EthzkSNARK-Voting "A Blockchain Based Approach For Secure E-Voting system". Consulted on 2021-04-01 at https://github.com/Blockchain-E-Voting/EthzkSNARK-Voting.

Santana, M. & Rezende, D. P. (2002, 09). Sobre a Urna Eletrônica. Consulted on 2020-11-17 at https://cic.unb.br/~rezende/trabs/urna_sd1_02.htm.

Scott, T. (2014, 12). Why Electronic Voting is a BAD Idea - Computerphile. Consulted on 2020-11-17 at https://www.youtube.com/watch?v=w3_0x6oaDmI.

Scott, T. & Elliott, S. (2019, 12). Why Electronic Voting Is Still A Bad Idea. Consulted on 2020-11-17 at https://www.youtube.com/watch?v=LkH2r-sNjQs.

Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M. & Halderman, J. A. (2014a). Independent Report on E-voting in Estonia. Consulted on 2020-11-17 at https://estoniaevoting.org/.

Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M. & Halderman, J. A. (2014b, 11). Security Analysis of the Estonian Internet Voting System. *Proceedings of the 21st ACM Conference on Computer and Communications Security*.

Stoll, C., Klaaßen, L. & Gallersdörfer, U. (2019). The Carbon Footprint of Bitcoin. *Joule*, 3(7), 1647-1661. doi: https://doi.org/10.1016/j.joule.2019.05.012.

V., V. & S., V. (2019). A Novel P2P based System with Blockchain for Secured Voting Scheme. *2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, 1, 153-156. doi: 10.1109/ICONSTEM.2019.8918895.

Vivek, S. K., Yashank, R. S., Prashanth, Y., Yashas, N. & Namratha, M. (2020). E-Voting Systems using Blockchain: An Exploratory Literature Survey. *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 890-895. doi: 10.1109/ICIRCA48905.2020.9183185.

Blockchain-based database - List of Blockchain-based databases. (2020). In *Wikipedia*. Consulted on 2020-12-04 at https://en.wikipedia.org/wiki/Blockchain-based_database.

Yang, X., Yi, X., Nepal, S. & Han, F. (2018). Decentralized Voting: A Self-tallying Voting System Using a Smart Contract on the Ethereum Blockchain. *Web Information Systems Engineering – WISE 2018*, pp. 18–35.

Yavuz, E., Koç, A. K., Çabuk, U. C. & Dalkılıç, G. (2018). Towards secure e-voting using ethereum blockchain. *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pp. 1-7. doi: 10.1109/ISDFS.2018.8355340.

Zhang, W., Yuan, Y., Hu, Y., Huang, S., Cao, S., Chopra, A. & Huang, S. (2018). A Privacy-Preserving Voting Protocol on Blockchain. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 401-408. doi: 10.1109/CLOUD.2018.00057.

ZoKrates. (2021). ZoKrates "ZoKrates is a toolbox for zkSNARKs on Ethereum. This is a proof-of-concept implementation. It has not been tested for production.". Consulted on 2021-04-01 at https://zokrates.github.io/introduction.html.