ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Αρχιτεκτονικές λογισμικού για υλοποίηση κατανεμημένων αυτόνομων οργανισμών (DAO) με τεχνολογίες blockchain

Software architectures for implementing distributed autonomous organizations (DAO) with blockchain technologies

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σταραντζής Δημήτρης

Επιβλέπων : Βασίλειος Βεσκούκης

Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Αρχιτεκτονικές λογισμικού για υλοποίηση κατανεμημένων αυτόνομων οργανισμών (DAO) με τεχνολογίες blockchain

Software architectures for implementing decentralized autonomous organizations (DAO) with blockchain technologies

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σταραντζής Δημήτρης

Επιβλέπων : Βασίλειος Βεσκούκης

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11 Μαρτίου 2022.

.....	.....	.....
Βασίλειος Βεσκούκης Καθηγητής Ε.Μ.Π.	Αριστείδης Παγουρτζής Καθηγητής Ε.Μ.Π.	Φωτάκης Δημήτριος Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2022

.....

Σταραντζής Δημήτρης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη ενός αποκεντρωμένου αυτόνομου οργανισμού (Decentralized Autonomous Organization) ή απλώς DAO, με χρήση της τεχνολογίας blockchain. Για την καλύτερη κατανόηση των βασικών αρχών της τεχνολογίας του blockchain χρησιμοποιήθηκε το Bitcoin. Πέρα από τις βασικές αρχές αυτές, εξηγούμε τα έξυπνα συμβόλαια (smart contracts), το βασικό συστατικό στοιχείο ενός DAO. Παρουσιάζουμε μία νέα μορφή ψηφιακού οργανισμού, το DAO, που αποτελείται και ελέγχεται αποκλειστικά από άτομα ή επιχειρήσεις ως ομότιμα (peers) μέλη. Ως εργαλείο ανάπτυξης διαλέξαμε το Hyperledger Fabric, ένα από τα πιο επιτυχημένα εργαλεία για την ανάπτυξη ιδιωτικών αδειοδοτημένων (private permissioned) δικτύων blockchain. Καταλήγουμε πως η πρωτοπόρα αρχιτεκτονική του αλλά και το σύνολο των εργαλείων του, το καθιστούν κατάλληλο για οποιαδήποτε πιθανή υλοποίηση ενός DAO. Εξετάζουμε τα προβλήματα ενός έμπιστου τρίτου φορέα (Trusted 3rd Party) στο παράδειγμα της διαμοίρασης της παραγωγής ηλεκτρικής ενέργειας μεταξύ παραγωγών σε έναν Φορέα Σωρευτικής Εκπροσώπησης ή απλώς ΦΟΣΕ. Προτείνουμε ως λύση των προβλημάτων αυτών, τη δημιουργία ενός DAO με ομότιμα μέλη του, τους παραγωγούς ενός ΦΟΣΕ. Αναπτύσσουμε τη δομή ενός DAO, ένα ιδιωτικό αδειοδοτημένο δίκτυο blockchain αποτελούμενο από 4 παραγωγούς, με χρήση του εργαλείου Fabric. Πάνω στο δίκτυο αυτό ορίζουμε τα έξυπνα συμβόλαια που αποτελούν την επιχειρηματική λογική και τους κανόνες που διέπουν το DAO. Για την αλληλεπίδραση των παραγωγών με τις λειτουργίες του DAO κατασκευάζουμε μια διεπαφή χρήστη (user interface). Συγκρίνουμε ως προς την ευχρηστία και την καταλληλότητα της ανάπτυξης ενός DAO, το εργαλείο Fabric με το εργαλείο GoEthereum που είναι ένα από τα πιο πετυχημένα στο τομέα δημόσιων blockchain. Διαπιστώνουμε επίσης, πως το DAO είναι μια αποτελεσματική λύση των προβλημάτων που φέρει ένας έμπιστος τρίτος φορέας. Δείχνουμε ότι το Fabric είναι ώριμο και ευέλικτο εργαλείο για την ανάπτυξη ενός ιδιωτικού δικτύου blockchain αλλά απαιτεί μεγάλη εξοικείωση από τους διαχειριστές του. Τέλος από τη σύγκριση των δύο εργαλείων συμπεραίνουμε πως το Fabric είναι πιο δύσχρηστο από το GoEthereum λόγω της πολυπλοκότητάς του αλλά πιο κατάλληλο και πιο αποτελεσματικό.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Blockchain, Τεχνολογίες Λογισμικού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Blockchain, DAO, Bitcoin, Hyperledger Fabric, Έξυπνα Συμβόλαια, Chaincode, Αποκεντρωμένες Εφαρμογές.

ABSTRACT

The purpose of this diploma thesis is to develop a decentralized autonomous organization or simply DAO, using blockchain technology. Bitcoin was used to better understand the basic principles of blockchain technology. In addition to these basic principles, we explain smart contracts, the key component of a DAO. We present a new form of digital organization, the DAO, which consists of and is controlled exclusively by individuals or companies as peer members. As a development tool we chose Hyperledger Fabric, one of the most successful tools for developing private permissioned blockchain networks. We conclude that its pioneering architecture and all its tools make it suitable for any possible implementation of a DAO. We look at the problems of a trusted 3rd party in the example of sharing electricity generation between producers of an electricity aggregator. We propose as a solution to these problems, the creation of a DAO with its equal members, the producers of an aggregator. We develop the structure of a DAO, a private permissioned blockchain network consisting of 4 producers, using the Fabric tool. On this network we define the smart contracts that constitute the business logic and the rules that govern the DAO. For the interaction of producers with the functions of DAO we build a user interface. We compare in terms of usability and suitability for the development of a DAO, the Fabric tool with the GoEthereum tool which is one of the most successful in the field of public blockchains. We also find that DAO is an effective solution to the problems posed by a trusted 3rd party. We show that Fabric is a mature and flexible tool for developing a private blockchain network but requires a great deal of familiarity from its administrators. Finally, by comparing the two tools, we conclude that Fabric is more difficult to use than GoEthereum due to its complexity, but more convenient and more efficient.

SUBJECT AREA: Blockchain, Software Technologies

KEYWORDS: Blockchain, DAO, Bitcoin, Hyperledger Fabric, Smart Contracts, Chaincode, Decentralized Applications.

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα ήθελα να ευχαριστήσω τον συνάδελφο Σέργιο Έξαρχο, πτυχιούχο του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιά, για τη συνεργασία και την πολύτιμη συμβολή του στην ολοκλήρωση της.

Συμπληρωματικά θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Βασίλειο Βεσκούκη αλλά και τον Υποψήφιο Διδάκτορα και Ερευνητή Γιάννη Τζαννέτο, για την σημαντική τους συμβολή στην εποπτεία της παρούσας εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου και την αδελφή μου, που ήταν μαζί μου όλα αυτά τα χρόνια και με στηρίζουν συνεχώς σε πολλούς τομείς.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1 Εισαγωγή

Το ολοένα και αυξανόμενο ενδιαφέρον γύρω από τα θέματα ψηφιακής ασφάλειας, απασχολεί ολοένα και περισσότερο τον κόσμο, αλλά και η εξέλιξη της επιστήμης της κρυπτογραφίας που στρέφει το ενδιαφέρον των μελετητών για την διατήρηση της ασφάλειας, οδήγησαν στην δημιουργια της τεχνολογίας blockchain. Η απαίτηση της αυτοματοποίησης των συναλλαγών μεταξύ επιχειρήσεων ή/και οργανισμών Business-to-Business (B2B) κάνει πιο επιτακτική την ανάγκη για την χρήση του blockchain και την δημιουργία κατανεμημένων αυτόνομων οργανισμών.

Η βασική διάρθρωση της παρούσας πτυχιακής εργασίας εστιάζει στην μελέτη της τεχνολογίας blockchain, με στόχο την κατανόηση της σε πρακτικό επίπεδο εστιάζοντας στην αναπτυξιακή προσέγγιση. Πιο συγκεκριμένα, η παρούσα εργασία έχει ως στόχο να γίνει κατανοητή με παραστατικό τρόπο η τεχνολογία blockchain και ο τρόπος που αυτή λειτουργεί. Για αυτόν τον λόγο το Κεφάλαιο 2 περιγράφει την δομή του blockchain, εισάγει έννοιες όπως οι συναλλαγές, αλλά παραθέτει και την δομή του μπλοκ, την αλυσιδωτή σύνδεση των μπλοκ και την έννοια του ιδιωτικού blockchain. Για την καλύτερη κατανόηση των βασικών αρχών της τεχνολογίας του blockchain χρησιμοποιήθηκε το Bitcoin, παρόλο που η τεχνολογία αυτή δεν περιορίζεται μόνο στον οικονομικό τομέα.

Στο κεφάλαιο 3 αναλύουμε το λογισμικό του Hyperledger Fabric, εξηγούμε τα εργαλεία που το αποτελούν καθώς και την εμπειρία μας με την χρήση τους στην υλοποίηση ενός κατανεμημένου αυτόνομου οργανισμού.

Στο κεφαλαιο 4 περιγράφουμε την περίπτωση χρήσης ενός κατανεμημένου αυτόνομου οργανισμού στο πρόβλημα διαμοίρασης της παραγωγής ηλεκτρικής ενέργειας μεταξύ παρόχων. Επίσης περιγράφουμε με παραστατικό τρόπο την αποκεντρωμένη εφαρμογή που υλοποιήσαμε για τη λύση του παραπάνω προβλήματος.

Στο κεφαλαιο 5 εξηγούμε τη διαδικασία υλοποίησης της αποκεντρωμένης εφαρμογής. Χωρίζουμε την υλοποίηση σε 4 ενότητες, οι οποίες είναι η ενότητα διεπαφής χρήστη, η ενότητα του ενδιάμεσου λογισμικού, η ενότητα της επιχειρηματικής λογικής ενός DAO μέσω του έξυπνου συμβολαίου καθώς και την ενότητα της δομής του DAO μέσω του δικτύου.

Στο κεφάλαιο 6 παρουσιάζουμε μία προσομοίωση της χρήσης της αποκεντρωμένης εφαρμογής. Αυτό επιτυγχάνεται μέσω της χρήσης της αποκεντρωμένης εφαρμογής από τους παραγωγούς - μέλοι του DAO. Το σενάριο χρήσης εξετάζεται μέσω στιγμιοτύπων οθόνης (screenshots) τα οποία αναλύονται.

Στο κεφάλαιο 7 γίνεται μία σύγκριση του εργαλείου Fabric και του εργαλείου GoEthereum για την ανάπτυξη αποκεντρωμένης εφαρμογής με τη λειτουργία DAO. Εξετάζουμε παρατηρήσεις που διαμορφώθηκαν κατά τη διαδικασία υλοποίησης του DAO.

Στο κεφάλαιο 8 συλλέγονται τα συμπεράσματα της διπλωματικής εργασίας που αφορούν την τεχνολογία blockchain, του DAO καθώς και του εργαλείου Fabric. Ακόμα παραθέτουμε ορισμένες πιθανές μελλοντικές χρήσεις της παρούσας εργασίας.

Συνοψίζοντας, η συνεισφορά της παρούσας διπλωματικής εργασίας μαζί με την αδελφική συνεργατική της πτυχιακή του Σέργιου Έξαρχου του τμήματος Ψηφιακών Συστημάτων του πανεπιστημίου Πειραιά εντοπίζεται κατά κύριο λόγο τόσο στο επίπεδο της μελέτης και της σύγκρισης του Ethereum και του Hyperledger Fabric στην ανάπτυξη και υλοποίηση μιας αποκεντρωμένης εφαρμογής, όσο και στην έρευνα και την υλοποίηση μιας αποκεντρωμένης εφαρμογής που θα έχει ως απώτερο σκοπό να επιδείξει τη χρησιμότητα ενός DAO. Έτσι, η παρούσα μελέτη διαθέτει έρευνα με συγκριτική μέθοδο και επεκτείνεται στο επίπεδο προγραμματιστικής εφαρμογής.

2 Εισαγωγή στο Blockchain

2.1 Βασικές έννοιες του blockchain

Η τεχνολογία blockchain αποτελεί την Πέμπτη επανάσταση των ηλεκτρονικών υπολογιστών [1]. Βασικός στόχος της είναι η αντικατάσταση του έμπιστου τρίτου φορέα που υπάρχει πλέον στις περισσότερες συναλλαγές. Για παράδειγμα, για τη συναλλαγές χρημάτων τυπικά βασιζόμαστε στην ακεραιότητα της τράπεζας ως τον έμπιστο τρίτο φορέα. Πρόδρομος του blockchain είναι ένα έγγραφο του 1991 με το όνομα “How to timestamp a digital document” [2]. Το blockchain βασίζεται σε ένα αποκεντρωμένο σύστημα ομότιμης σύνδεσης μεταξύ υπολογιστών. Ένα δίκτυο blockchain συντάσσεται από διασυνδεδεμένους ανεξάρτητους χρήστες (distributed network of independent users). Οι ηλεκτρονικοί υπολογιστές που συμμετέχουν στο δίκτυο δύνανται να βρίσκονται σε περισσότερες από μία τοποθεσίες αποδίδοντας έτσι τη μέγιστη ασφάλεια στο σύστημα. Στο χαρακτηριστικό αυτό βασίζεται η επιτυχία του Bitcoin. Το Bitcoin εμφανίστηκε σε μία επιστημονική εργασία το 2008 [3] και κυκλοφόρησε ως λογισμικό το 2009. Η ανάγνωση του συνίσταται απαραίτητη για την κατανόηση της ιδέας πίσω από τη δημιουργία του blockchain. Ερωτήσεις όπως “ποιο πρόβλημα λύνει;”, “ποια λύση προσφέρει;“ καθώς και “ποια εξαρτήματα χρησιμοποιούνται;” οδηγούν τον αναγνώστη σε βαθύτερη κατανόηση. Όταν μια πληροφορία έχει εγγραφεί στην αλυσίδα blockchain, είναι αδύνατον αυτή να διαγραφεί εκτός αν η πλειοψηφία (>50%) των χρηστών του δικτύου το αποφασίσει. Έτσι, εμπορικές, εταιρικές και τραπεζικές συναλλαγές αποκτούν τη μέγιστη δυνατή αμεταβλητότητα (immutability) και ασφάλεια.

Το θεμελιώδες στοιχείο του blockchain είναι το μπλοκ (block) [1]. Τα μπλοκ είναι ομάδες καταχωρήσεων συναλλαγών που έχουν καταγραφεί στο δημόσιο μητρώο (ledger) και προς τούτο μοιάζουν με μητρώα-λίστες δεδομένων. Εν συνεχεία, η σύνδεση τους σχηματίζει μια αλυσίδα (chain) μέσω της μεθόδου κατακερματισμού (hashing) του ενός μπλοκ με το επόμενο. Όταν αλλάζουμε κάποια δεδομένα σε ένα μπλοκ, αλλάζουμε την τιμή κατακερματισμού του και ως αποτέλεσμα την τιμή κατακερματισμού όλων των μπλοκ μετά από αυτό. Η τεχνολογία κατακερματισμού υποβοηθά την ασφάλεια του συστήματος [4] καθώς κάθε μπλοκ δημιουργείται μόνο από δεδομένα-κανόνες του προηγούμενου μπλοκ, κατά τρόπο ώστε ένα μπλοκ να αναγνωρίζει μόνο το προηγούμενο του κι όχι οποιοδήποτε άλλο. Η μέθοδος κατακερματισμού χρησιμοποιείται για την παραγωγή ενός μοναδικού ψηφιακού αποτυπώματος για τα δεδομένα-κανόνες ενός μπλοκ. Μια κεφαλίδα μπλοκ περιέχει λεπτομέρειες σχετικά με τη δομή των δεδομένων μέσα στο μπλοκ (όπως ο κατακερματισμός των προηγούμενων μπλοκ, η χρονική στιγμή που δημιουργήθηκε το μπλοκ, η ρίζα merkle και το nonce). Η χρονική σήμανση μας βοηθά να αποτρέψουμε τις διπλές δαπάνες (double spending). Η ρίζα merkle είναι ένας κατακερματισμός που αντιπροσωπεύει κάθε συναλλαγή στο μπλοκ. Ολόκληρο το σύνολο των συναλλαγών μπορεί να ανακατασκευαστεί αντιστρέφοντας την μέθοδο του κατακερματισμού. Το nonce είναι ένας αυθαίρετος, μοναδικός αριθμός που χρησιμοποιείται για την αποτροπή επιθέσεων επανάληψης (replay attack). Το μέγεθος του μπλοκ επιβάλλει ένα όριο στον αριθμό των συναλλαγών που μπορεί να επεξεργαστεί ένα δίκτυο blockchain ανά δευτερόλεπτο και έτσι μπορεί να φανεί ότι εμποδίζει την ικανότητα του να κλιμακώνεται. Τέλος, ο κάθε κόμβος του δικτύου έχει ένα αντίγραφο του blockchain το οποίο ενημερώνει ο ίδιος κάθε φορά που προστίθεται ένα καινούργιο μπλοκ.

Το blockchain υποστηρίζεται από έναν τύπο ομότιμος-σε-ομότιμο κατανεμημένου δικτύου (Distributed peer-to-peer network). Οποιαδήποτε εφαρμογή συνομιλίας χρησιμοποιεί ένα ομότιμος-σε-ομότιμο δίκτυο για να επιτρέπει την ανοιχτή επικοινωνία μεταξύ των χρηστών της. Σε ένα κατανεμημένο δίκτυο όλοι λαμβάνουν ένα αντίγραφο των

πληροφοριών με τα ίδια δικαιώματα πρόσβασης και ελέγχου με οποιονδήποτε άλλο χρήστη.

Η συναίνεση (consensus) είναι μια ομάδα αλγορίθμων που χρησιμοποιούνται για την επίτευξη συμφωνίας σχετικά με το ποιες συναλλαγές είναι έγκυρες. Το πρόβλημα των βυζαντινών στρατηγών: Εννέα στρατηγοί, ο καθένας διοικεί ένα τμήμα του βυζαντινού στρατού, περικυκλώνουν μια πόλη και πρέπει να σχηματίσουν ένα σχέδιο για να την επιτεθούν. Ψηφίζουν είτε για επίθεση είτε για υποχώρηση. Και οι εννέα πρέπει να συμφωνήσουν να επιτεθούν ή και οι εννέα πρέπει να συμφωνήσουν να υποχωρήσουν, γιατί αν μέρος του στρατού υποχωρήσει έχουν υψηλό κίνδυνο να χάσουν. Ας υποθέσουμε ότι 4 στρατηγοί ψηφίζουν για υποχώρηση, 4 στρατηγοί ψηφίζουν για επίθεση και ένας στρατηγός στέλνει ψήφο «υποχώρησης» στους υποχωρητές και στέλνει ψήφο «επίθεσης» στους επιτιθέμενους.

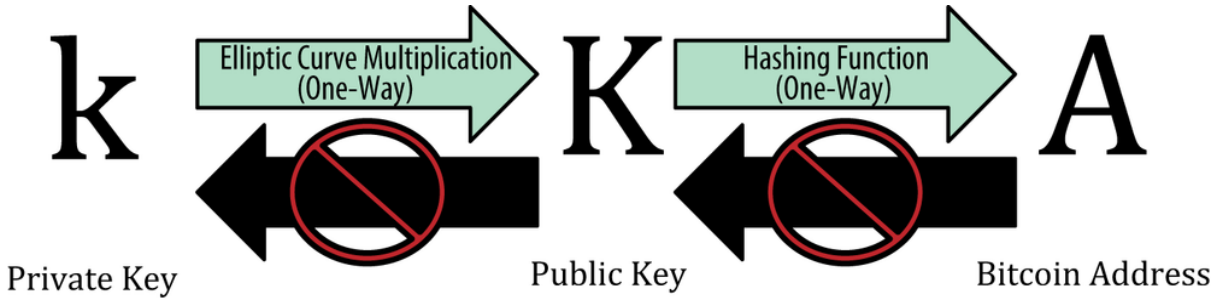
Για την επίλυση του προβλήματος των βυζαντινών στρατηγών το Bitcoin και το Ethereum χρησιμοποιούν τον αλγόριθμο proof-of-work. Άλλοι σημαντικοί αλγόριθμοι είναι το proof-of-stake, proof-of-authority ή και η μέθοδος των 3 φάσεων Execute-Order-Validate που χρησιμοποιεί το Hyperledger Fabric.

2.2 Διαχείριση Συναλλαγών

Όπως αναφέρθηκε στην προηγούμενη ενότητα ένα μπλοκ δεν είναι τίποτα άλλο από μια ομάδα συναλλαγών. Το blockchain διαχειρίζεται όλες αυτές τις συναλλαγές με τη χρήση ενός μηχανισμού ψηφιακής υπογραφής (digital signature). Χρειαζόμαστε έναν μηχανισμό ψηφιακής υπογραφής για να κάνουμε συναλλαγές, για να αποκτήσουμε μία διεύθυνση που μπορούμε να μοιραστούμε με άλλους ώστε να κάνουμε συναλλαγές μαζί τους αλλά και για να μπορούμε να αποδείξουμε ότι οι συναλλαγές έγιναν από εμάς. Για να κατανοήσουμε καλύτερα τη διαδικασία διαχείρισης συναλλαγών και το μηχανισμό ψηφιακής υπογραφής που χρησιμοποιεί ένα blockchain, θα αναλύσουμε το πορτοφόλι Bitcoin.

Ένα πορτοφόλι Bitcoin έχει ένα ιδιωτικό κλειδί (private key), ένα δημόσιο κλειδί (public key) και μια διεύθυνση πορτοφολιού (wallet address). Ένα ιδιωτικό κλειδί λειτουργεί ως ο κρυφός κωδικός του χρήστη και δεν πρέπει να μοιράζεται με κανέναν. Μια διεύθυνση πορτοφολιού λειτουργεί με παρόμοιο τρόπο με τη διεύθυνση κατοικίας, με την έννοια ότι χρησιμοποιείται ως η διεύθυνση αποστολής των συναλλαγών που μας αφορούν. Η διεύθυνση πορτοφολιού είναι η μόνη αναπαράσταση των κλειδιών που οι χρήστες θα βλέπουν τακτικά, επειδή αυτό είναι το μέρος που πρέπει να μοιραστούν με τον κόσμο. Το δημόσιο κλειδί είναι μία δυσανάγνωστη μορφή της διεύθυνσης πορτοφολιού και δεν χρειάζεται να τη γνωρίζει ο χρήστης.

Το δημόσιο κλειδί είναι το αποτέλεσμα του αλγορίθμου ECDSA [5] με είσοδο το ιδιωτικό κλειδί. Μια διεύθυνση πορτοφολιού προέρχεται από ένα δημόσιο κλειδί. Αρχικά το δημόσιο κλειδί χρησιμοποιείται ως είσοδος του αλγόριθμου SHA256 [6] και το αποτέλεσμα (ένας αριθμός από 256 bit) μεταφέρεται ως είσοδος του αλγόριθμου RIPEMD-160 [7]. Το αποτέλεσμα είναι ένας αριθμός 160 bit και δίνεται ως είσοδος στον αλγόριθμο BASE58CHECK [8] για τη δημιουργία ενός πιο αναγνώσιμου κωδικού που ονομάζουμε διεύθυνση πορτοφολιού.



Εικόνα 1: Μηχανισμός ψηφιακής υπογραφής στο Bitcoin.

Τα ιδιωτικά κλειδιά δημιουργούνται από μια κρυπτογραφικά ασφαλή γεννήτρια τυχαίων αριθμών (όπως οι βιβλιοθήκες [9] και [10]). Μια ψηφιακή υπογραφή δημιουργεί απόδειξη ιδιοκτησίας για κάθε συναλλαγή στο blockchain. **Παράδειγμα**: Ο Δημήτρης θέλει να στείλει ένα Bitcoin στο Σέργιο. Ο Δημήτρης παίρνει τη διεύθυνση πορτοφολιού του Σέργιου και δημιουργεί μια συναλλαγή με χρέωση 0,003 Bitcoin. Στη συνέχεια, επαληθεύει τα στοιχεία του και στέλνει τη συναλλαγή. Το πορτοφόλι υπογράφει τη συναλλαγή του χρησιμοποιώντας το ιδιωτικό του κλειδί και μεταδίδεται στην ουρά αναμονής συναλλαγών (mempool) εντός του δικτύου. Η συναλλαγή γίνεται κάποια στιγμή αποδεκτή από τους εξορύκτες (miners), τη βάζουν σε ένα μπλοκ, βρίσκουν το nonce και εκχωρούν στο μπλοκ μια τιμή κατακερματισμού. Το μπλοκ τοποθετείται στο blockchain, καθώς κερδίζει επιβεβαιώσεις, γίνεται αποδεκτό ως έγκυρη συναλλαγή στο δίκτυο και τελικά ο Σέργιος παίρνει το Bitcoin του.

2.3 Μία συναλλαγή

Στην προηγούμενη ενότητα αναλύσαμε τον μηχανισμό ψηφιακής υπογραφής που χρησιμοποιείται για την διαχείριση συναλλαγών στο blockchain. Στην παρούσα ενότητα θα μελετήσουμε τα συστατικά που συνθέτουν μία συναλλαγή σε ένα δίκτυο blockchain. Γνωρίζουμε πως μια συναλλαγή είναι το θεμελιώδες στοιχείο ενός μπλοκ δεδομένων. Μια συναλλαγή είναι μια δομή δεδομένων που κωδικοποιεί μια μεταφορά αξίας από μια πηγή που ονομάζεται “είσοδος” σε έναν προορισμό που ονομάζεται “έξοδος”. Σε μια συναλλαγή, οι εισροές είναι απλώς οι μη δαπανηθείσες εκροές από μια άλλη συναλλαγή (UTXO).

Συναλλαγή = “έκδοση” + “πλήθος εισόδου” + “πληροφορίες εισόδου” + “πλήθος εξόδου” + “πληροφορίες εξόδου” + “ώρα κλειδώματος”.

Πληροφορίες εισόδου = “κατακερματισμός προηγούμενης εξόδου” + “δείκτης προηγούμενης εξόδου” + “μέγεθος σεναρίου ξεκλειδώματος” + “σενάριο ξεκλειδώματος” + “αριθμός ακολουθίας”

Πληροφορίες εξόδου = “ποσό” + “μέγεθος σεναρίου κλειδώματος” + “σενάριο κλειδώματος”

Κάθε UTXO έχει ένα σενάριο κλειδώματος (locking script) που περιέχει τις προϋποθέσεις που απαιτούνται για να ξοδευτεί. Για να επικυρώσουμε μια συναλλαγή, αρπάζουμε το σενάριο ξεκλειδώματος που περιέχεται στην είσοδο και ελέγχουμε αν ξεκλειδώνει το σενάριο κλειδώματος του UTXO. Σημειώστε ότι το σενάριο ξεκλειδώματος σε μια συναλλαγή δεν ξεκλειδώνει το σενάριο κλειδώματος της ίδιας συναλλαγής, αλλά της προηγούμενης συναλλαγής.

2.4 Έξυπνα συμβόλαια

Τα έξυπνα συμβόλαια πρωτοεμφανίστηκαν στο κεντρικό δίκτυο blockchain του Ethereum. Με τη χρήση έξυπνων συμβολαίων τα blockchain μπορούν να εκτελούν αυθαίρετη,

προγραμματιζόμενη λογική συναλλαγών. Τα script στο Bitcoin ήταν η πρώιμη μορφή των έξυπνων συμβολαίων. Ένα έξυπνο συμβόλαιο λειτουργεί ως μια αξιόπιστη κατανεμημένη εφαρμογή και κερδίζει την αξιοπιστία του από τη τεχνολογία blockchain και το αντίστοιχο πρωτόκολλο συναίνεσης του εκάστοτε δικτύου.

Πολλά υπάρχοντα blockchain που χρησιμοποιούν έξυπνα συμβόλαια εφαρμόζουν ένα πρωτόκολλο ή συναίνεση ή ατομική εκπομπή (atomic broadcast) που πρώτον, ταξινομεί τις συναλλαγές και τις διαδίδει σε όλους τους ομότιμους και δεύτερον, κάθε ομότιμος εκτελεί τις συναλλαγές διαδοχικά. Την διαδικασία αυτή την ονομάζουμε αρχιτεκτονική ταξινομηση-εκτέλεση (order-execute) και απαιτεί από όλους τους ομότιμους να εκτελούν κάθε συναλλαγή και όλες οι συναλλαγές να είναι ντετερμινιστικές.

2.5 Κατανεμημένος Αυτόνομος Οργανισμός (DAO)

2.5.1 Τι είναι DAO

Ένας κατανεμημένος αυτόνομος οργανισμός είναι μια κοινότητα από μέλη χωρίς κεντρική ηγεσία. Τα μέλη μπορεί να είναι άτομα ή οργανισμοί. Λειτουργεί σαν μια πλήρως διαδικτυακή επιχείρηση που ανήκει συλλογικά στα μέλη της και διαχειρίζεται από αυτά. Κανένα μέλος του, δεν έχει την εξουσία να έχει πρόσβαση στα δεδομένα του οργανισμού χωρίς την έγκριση της ομάδας. Οι αποφάσεις διέπονται από προτάσεις και ψηφοφορία για να διασφαλιστεί ότι όλοι στον οργανισμό έχουν φωνή.

Δεν υπάρχει διευθύνων μέλος που να μπορεί να πάρει αποφάσεις με βάση τις δικές του ιδιοτροπίες και καμία πιθανότητα να χειραγωγήσει τα δεδομένα. Όλα είναι διαφανή (transparent) και οι κανόνες σχετικά με τις ενέργειες του DAO ενσωματώνονται μέσω του κώδικα του.

2.5.2 Γιατί χρειαζόμαστε ένα DAO

Για να ξεκινήσουμε έναν οργανισμό, απαιτείται να δείξουμε μεγάλη εμπιστοσύνη στα άτομα με τα οποία σκοπεύουμε να συνεργαστούμε. Αλλά είναι δύσκολο να εμπιστευτούμε κάποιον που δεν γνωρίζουμε ή που έχουμε αλληλεπιδράσει μόνο στο διαδίκτυο. Με ένα DAO δεν χρειάζεται να εμπιστευόμαστε κανέναν άλλον στην ομάδα, μόνο τον κώδικα του DAO, ο οποίος είναι 100% διαφανής και επαληθεύσιμος από οποιονδήποτε. Αυτό δημιουργεί ευκαιρίες για συνεργασία και συντονισμό αγνώστων μελών χωρίς έμπιστο τρίτο φορέα.

Παρακάτω παραδίδουμε ένα πίνακα σύγκρισης ενός DAO με ένα παραδοσιακό οργανισμό για τη καλύτερη κατανόηση των περιπτώσεων που θα επιλέγαμε ένα DAO.

Πίνακας 1: Σύγκριση ενός Κατανεμημένου Αυτόνομου Οργανισμού με έναν Παραδοσιακό Οργανισμό. [11]

Κατανεμημένος Αυτόνομος Οργανισμός	**Παραδοσιακός Οργανισμός**
Συνήθως επίπεδος και πλήρως δημοκρατικός	Συνήθως ιεραρχικός
Απαιτείται ψηφοφορία για την υλοποίηση οποιασδήποτε αλλαγής	Ανάλογα τη δομή, αλλαγές μπορεί να απαιτηθούν από ένα μέλος ή μπορεί να προσφέρεται ψηφοφορία

Οι ψήφοι καταμετρούνται και το αποτέλεσμα εφαρμόζεται αυτόματα χωρίς έμπιστο τρίτο φορέα	Εάν προσφέρεται ψηφοφορία, οι ψήφοι καταμετρούνται εσωτερικά και το αποτέλεσμα αντιμετωπίζεται χειροκίνητα
Οι λειτουργίες του οργανισμού αντιμετωπίζονται αυτόματα	Απαιτεί ανθρώπινο χειρισμό ή κεντρικά ελεγχόμενο αυτοματισμό, επιρρεπής σε χειραγώγηση
Κάθε δραστηριότητα είναι διαφανής και πλήρως δημόσια	Η δραστηριότητα είναι συνήθως ιδιωτική και περιορισμένη στο κοινό

2.5.3 Πως λειτουργεί ένα DAO

Η ραχοκοκαλιά ενός DAO είναι το έξυπνο συμβόλαιό του. Το συμβόλαιο ορίζει τους κανόνες του οργανισμού και διαχειρίζεται τις λειτουργίες του. Μόλις το συμβόλαιο τεθεί σε λειτουργία, κανείς δεν μπορεί να αλλάξει τους κανόνες παρά μόνο με ψηφοφορία. Αν κάποιος προσπαθήσει να κάνει κάποια ενέργεια που δεν καλύπτεται από τους κανόνες και τη λογική του κώδικα, θα αποτύχει. Η ομάδα λαμβάνει αποφάσεις συλλογικά και οι λειτουργίες εξουσιοδοτούνται αυτόματα όταν υπολογιστούν οι ψήφοι. Αυτό είναι δυνατό επειδή τα έξυπνα συμβόλαια είναι αμετάβλητα από τη στιγμή που ενεργοποιηθούν. Ένα μέλος, δεν μπορεί απλώς να τροποποιήσει τον κώδικα (οι κανόνες του DAO) χωρίς να το αντιληφθούν τα υπόλοιπα μέλη επειδή όλα είναι δημόσια.

Ένα μέλος ενός DAO υλοποιημένο στο Fabric είναι ένας οργανισμός. Κάθε τέτοιος οργανισμός, έχει 3 επιλογές: (1) μπορεί να εγκαταστήσει ένα έξυπνο συμβόλαιο ή μία νέα έκδοσή του, (2) μπορεί να υπογράψει - να αποδεχθεί ένα έξυπνο συμβόλαιο και (3) μπορεί να ενημερώσει τους κανόνες του οργανισμού ενεργοποιώντας το νέο συμβόλαιο μόνο εφόσον η πλειονότητα έχει αποδεχτεί το συμβόλαιο.

2.6 Σύγκριση ιδιωτικών και δημοσίων δικτύων blockchain

Τα δίκτυα blockchain μπορούν να διαχωριστούν σε δύο κατηγορίες: χωρίς άδεια (permissionless) και με άδεια (permissioned). Τα χωρίς άδεια είναι γνωστά και ως δημόσια, ενώ τα με άδεια ή αδειοδοτημένα είναι γνωστά ως ιδιωτικά.

Στα δημόσια blockchain εντάσσονται οι πλατφόρμες που επιτρέπουν τη συμμετοχή στο δίκτυο χωρίς επιλογή ή άλλη διαδικασία έγκρισης. Με λίγα λόγια, σε ένα δημόσιο blockchain ο καθένας έχει τη δυνατότητα να λάβει ένα αντίγραφο του blockchain και να συμμετάσχει στο δίκτυο, με μόνη προϋπόθεση την εφαρμογή των πρωτοκόλλων της πλατφόρμας. Τα δημόσια blockchain συνήθως περιλαμβάνουν ένα εγγενές κρυπτονόμισμα και συχνά χρησιμοποιούν συναίνεση που βασίζεται σε PoW και οικονομικά κίνητρα.

Τα ιδιωτικά ή με άδεια blockchain, από την άλλη πλευρά, δημιουργούν ένα δίκτυο μεταξύ ενός συνόλου γνωστών, αναγνωρισμένων συμμετεχόντων. Ένα μέλος του δικτύου μπορεί να προσκαλέσει ένα πιθανό νέο μέλος, που αν λάβει έγκριση από την πλειονότητα των υπολοίπων μελών, προστίθεται στο δίκτυο. Ένα ιδιωτικό blockchain παρέχει έναν τρόπο διασφάλισης των αλληλεπιδράσεων μεταξύ μιας ομάδας οντοτήτων που έχουν κοινό στόχο αλλά δεν εμπιστεύονται πλήρως η μία την άλλη, όπως επιχειρήσεις που ανταλλάσσουν κεφάλαια, αγαθά ή πληροφορίες.

Για τη σύγκριση των δύο ειδών blockchain θα εξετάσουμε τρεις παράγοντες: κλιμακωσιμότητα, ασφάλεια και αμεταβλητότητα. Με τον όρο “κλιμακωσιμότητα” εννοούμε την ικανότητα του δικτύου να διατηρεί την αποδοτικότητα του καθώς το μέγεθος του

φόρτου του δικτύου μεγαλώνει. Ο παράγοντας “ασφάλεια” σχετίζεται με την ικανότητα του δικτύου να παραμένει άτρωτο από επιθέσεις κακόβουλων χρηστών που θέλουν να παραποιήσουν το μητρώο προς όφελος τους. Τέλος, ως “αμεταβλητότητα” ορίζουμε την ικανότητα του δικτύου να διατηρεί την κατάσταση των μπλοκ απαράλλαχτη από τη στιγμή που προστεθούν στο δίκτυο.

Τα δημόσια blockchain είναι λιγότερο αποτελεσματικά σε σύγκριση με τα ιδιωτικά blockchain. Βασιζόμενη στις ταυτότητες των ομότιμων, ένα ιδιωτικό blockchain μπορεί να χρησιμοποιήσει την παραδοσιακή συναίνεση με ανοχή βυζαντινών σφαλμάτων (byzantine fault tolerance) ή απλώς BFT που είναι ταχύτερη στον ρυθμό προσθήκης νέων μπλοκ σε σχέση με τον παραδοσιακό αλγόριθμο proof-of-work που χρησιμοποιείται στα μεγαλύτερα δημόσια blockchain. Επίσης, τα δημόσια blockchain αντιμετωπίζουν ζητήματα κλιμακωσιμότητας λόγω ύπαρξης πολλών κόμβων, σε σύγκριση με τα ιδιωτικά και ως αποτέλεσμα ο ρυθμός προσθήκης νέων μπλοκ στο blockchain μειώνεται ραγδαία.

Σε μια επίθεση Sybil, οι χάκερ δημιουργούν και χρησιμοποιούν πολλές ψευδείς ταυτότητες δικτύου για να πλημμυρίσουν το δίκτυο και να καταστρέψουν το σύστημα. Εκτελώντας επιθέσεις Sybil, οι χάκερ μπορούν να αποκτήσουν δυσανάλογη επιρροή με τους ειλικρινείς κόμβους στο δίκτυο, εάν δημιουργήσουν αρκετές πλαστές ταυτότητες. Στη συνέχεια, μπορούν να αρνηθούν να λάβουν ή να μεταδώσουν μπλοκ, αποκλείοντας ουσιαστικά άλλους κόμβους από ένα δίκτυο. Εάν αυτοί οι κόμβοι φτάσουν αριθμούς >50% του δικτύου, τότε η επίθεση ονομάζεται “51% attack”. Το να έχεις περισσότερο από το 50% του ελέγχου του δικτύου σημαίνει να ελέγχεις το μητρώο και να μπορείς να το παραποιήσεις για να αντιστρέψεις συναλλαγές. Σε παγκόσμιο επίπεδο, οι επιχειρήσεις χάνουν περίπου 20 εκατομμύρια δολάρια ετησίως λόγω επιθέσεων 51% [12]. Σε ένα ιδιωτικό δίκτυο blockchain τέτοιου τύπου επιθέσεις αποφεύγονται αφού όλοι οι χρήστες είναι γνωστοί και έχουν εισέλθει στο δίκτυο με πρόσκληση ή κατά την δημιουργία του δικτύου.

Τα δημόσια blockchain είναι πλήρως αμετάβλητα, δηλαδή από τη στιγμή που ένα μπλοκ μπει στην αλυσίδα, δεν υπάρχει τρόπος να αλλαχθεί ή να διαγραφεί. Έτσι, διασφαλίζεται πως κανείς δεν μπορεί απλώς να αλλάξει ένα συγκεκριμένο μπλοκ και να ωφεληθεί εις βάρος των υπόλοιπων χρηστών. Από την άλλη πλευρά, τα ιδιωτικά blockchain είναι αμετάβλητα εκτός εξαιρετικών περιπτώσεων όπου μπορούν να διαγράψουν ένα συγκεκριμένο μπλοκ εάν το κρίνουν κατάλληλο.

Πίνακας 2: Σύγκριση δημοσίου και ιδιωτικού blockchain.

	Δημόσιο blockchain	Ιδιωτικό blockchain
Άδεια (permission)	Χωρίς άδεια	Με άδεια
Κλιμακωσιμότητα (scalability)	Δυσκολότερη	Ευκολότερη
Ασφάλεια (vulnerability)	Λιγότερο ασφαλές	Περισσότερο ασφαλές
Αμεταβλητότητα (immutability)	Ευκολότερη	Δυσκολότερη

2.7 Δημοφιλή blockchain

2.7.1 Δημόσια

Bitcoin

Η τεχνολογία blockchain πρωτοεμφανίστηκε όταν ένα επιστημονικό άρθρο με τίτλο "Bitcoin: A peer-to-peer electronic cash system" δημοσιεύτηκε το 2008 από άγνωστο συγγραφέα με το ψευδώνυμο Satoshi Nakamoto [3]. Το Bitcoin εξακολουθεί να αποτελεί σήμερα το δημοφιλέστερο κρυπτονόμισμα, όμως αντιμετωπίζει ορισμένα βασικά προβλήματα, όπως η μεγάλη σπατάλη πόρων για την επιβεβαίωση κάθε μπλοκ και ο αργός ρυθμός επικύρωσης συναλλαγών.

Ethereum

Στα τέλη του 2013, μια ομάδα προγραμματιστών αποτελούμενη από τους Vitalik Buterin, Jeffrey Wilcke και Gavin Wood εργάστηκε για τη δημιουργία μιας κατανεμημένης υπολογιστικής πλατφόρμας ανοικτού κώδικα με βάση το blockchain, το Ethereum [13]. Ο στόχος τους ήταν να δημιουργήσουν μια πλήρως αξιόπιστη πλατφόρμα έξυπνων συμβολαίων.

Το Ethereum είναι μια ανοιχτή πλατφόρμα blockchain που επιτρέπει σε οποιονδήποτε να αναπτύξει και να εκτελέσει αποκεντρωμένες εφαρμογές (Decentralized Applications ή απλώς DApps) στην τεχνολογία blockchain. Αυτή η πλατφόρμα επικεντρώνεται στη δυνατότητα της αυτόματης διαχείρισης ψηφιακών αγαθών (digital asset management) και, για να το πραγματοποιήσει αυτό, υποστηρίζει τα έξυπνα συμβόλαια. Όπως και το Bitcoin, κανείς δεν μπορεί να κατέχει ή να ελέγχει την πλατφόρμα Ethereum. Χρησιμοποιεί το κρυπτονόμισμα Ether για την τροφοδότηση ολόκληρου του οικοσυστήματος του Ethereum.

Stellar

Η πλατφόρμα Stellar είναι μια πλατφόρμα κατανεμημένου μητρώου, ειδικά κατασκευασμένη για τη διευκόλυνση των μεταβιβάσεων αξίας μεταξύ περιουσιακών στοιχείων [14]. Με το Stellar Consensus Protocol (SCP), καθίσταται δυνατή η επίτευξη συναίνεσης, χωρίς να βασίζεται σε ένα κλειστό σύστημα καταγραφής χρηματοοικονομικών συναλλαγών. Σε αντίθεση με τους αλγόριθμους συναίνεσης PoS και PoW, το SCP περιορίζει τα εμπόδια εισόδου στο δίκτυο για να συμμετάσχουν νέα μέλη. Η ασφάλειά του βασίζεται σε έναν μηχανισμό ψηφοφορίας και στην εφαρμογή ενός αλγόριθμου συναίνεσης μεταξύ κόμβων που δεν εμπιστεύονται ο ένας τον άλλον (Federated Byzantine Agreement FBA).

2.7.2 Ιδιωτικά

R3 Corda

Η πλατφόρμα R3 Corda είναι μια πρωτοποριακή πλατφόρμα blockchain για επιχειρήσεις, που έχει ως στόχο να μειώσει το κόστος των επιχειρηματικών συναλλαγών και να αυξήσει την ταχύτητά τους [15]. Αρχικά, είχε κατασκευαστεί μόνο για το χρηματοπιστωτικό τομέα, πλέον όμως μπορεί να εφαρμοστεί και σε άλλες περιπτώσεις χρήσης, όπως η υγειονομική περίθαλψη, η εφοδιαστική αλυσίδα, οι κυβερνητικές και δημόσιες υπηρεσίες και η χρηματοδότηση του εμπορίου.

Quorum

Η πλατφόρμα Quorum δημιουργήθηκε από την εταιρία JP Morgan και αποτελεί την εταιρική έκδοση του Ethereum blockchain [16]. Με την τροποποίηση του πυρήνα του Ethereum, η πλατφόρμα Quorum μπορεί να ενσωματώσει γρήγορα τις ενημερώσεις Ethereum.

Πρόκειται για μια πλατφόρμα blockchain ανοιχτού κώδικα που χρησιμοποιεί αλγόριθμους με βάση την ψήφο (vote based algorithms) για την εκτέλεση εκατοντάδων συναλλαγών ανά δευτερόλεπτο. Δεδομένου ότι είναι μια ιδιωτική πλατφόρμα blockchain, επιτρέπει μόνο στους εξουσιοδοτημένους συμμετέχοντες να λαμβάνουν μέρος στις συναλλαγές.

Ripple

Η πλατφόρμα Ripple στοχεύει στη σύνδεση των φορέων που πραγματοποιούν οικονομικές συναλλαγές, εμπορικών εταιρειών, τραπεζών και παρόχων υπηρεσιών πληρωμών [17]. Το Ripple επιτρέπει διεθνείς πληρωμές μέσω ενός ψηφιακού περιουσιακού στοιχείου που ονομάζεται Ripple ή XRP.

Χρησιμοποιώντας μια πιθανολογική μέθοδο ψηφοφορίας (probabilistic voting), η πλατφόρμα Ripple επιτυγχάνει τη συναίνεση μεταξύ των κόμβων στο δίκτυο. Αρκετές μεγάλες εταιρείες όπως η American Express, η SBI Holdings και η Deloitte πειραματίζονται με τις δυνατότητες του blockchain της Ripple για να μετασχηματίσουν τις διαδικασίες πληρωμής.

Ίδρυμα Hyperledger

Σύμφωνα με τον Brian Behlendorf, Εκτελεστικό Διευθυντή (Executive Director) στο ίδρυμα Hyperledger: “Καθώς νέα τεχνολογία αναπτύσσεται, υπάρχει ανάγκη για πρότυπα. Οι συμμετέχοντες θέλουν να επικεντρωθούν στο χρόνο και την προσπάθεια και τις επενδύσεις για να δημιουργήσουν λύσεις αντί να ανησυχούν για τη πλατφόρμα. Αυτό είναι το σκεπτικό για τα ανοιχτά πρότυπα…συγκεντρώνουμε το πιο συναρπαστικό χαρτοφυλάκιο με μια πολυμερή κοινότητα προγραμματιστών και προμηθευτών. Είναι παρόμοιο με τα οφέλη που έφερε το Linux στον κόσμο των λειτουργικών συστημάτων.” (μεταφρασμένο) [18]

Το ίδρυμα Hyperledger παρέχει ένα πορτοφόλιο συστημάτων blockchain ανοικτού κώδικα [19]. Κάποια από αυτά είναι τεχνολογίες κατανεμημένων μητρώων (distributed ledger technologies), άλλα είναι βιβλιοθήκες (libraries), άλλα στοχεύουν σε ειδικούς τομείς (domain specific), ενώ κάποια είναι εργαλεία (tools).

Οι τεχνολογίες κατανεμημένων μητρώων είναι το Fabric, το Sawtooth, το Iroha, το Besu και το Indy. Αυτές οι πέντε τεχνολογίες blockchain μπορούν να αξιοποιηθούν για την ανάπτυξη εφαρμογών σε διαφορετικούς τομείς, αλλά απαιτείται η ακριβής εκτίμηση των πλεονεκτημάτων και των αδυναμιών τους, ώστε να γίνει η επιλογή του καταλληλότερου για να αναπτυχθεί μια εφαρμογή. Οι βιβλιοθήκες είναι το Aries, το Transact και το Ursa. Το Grid στοχεύει στον ειδικό τομέα της εφοδιαστικής αλυσίδας. Τα εργαλεία είναι το Avalon, το Bevel, το Cactus, το Caliper, το Cello, το Explorer και το Firefly.

Το Hyperledger ξεκίνησε τον Δεκέμβριο του 2015 από τον οργανισμό Linux Foundation. Ο κύριος στόχος της χρήσης της πλατφόρμας Hyperledger είναι η ενίσχυση της αξιοπιστίας και της απόδοσης των τεχνολογιών blockchain.

Hyperledger Fabric

Η πλατφόρμα Hyperledger Fabric αποτελεί ένα από τα πλέον δημοφιλή Hyperledger Projects που έχει σχεδιαστεί για τη δημιουργία εφαρμογών blockchain χρησιμοποιώντας μια ευέλικτη αρχιτεκτονική. Με αυτό τον τρόπο προσφέρει στους σχεδιαστές των δικτύων την απαραίτητη ελευθερία να υλοποιήσουν ένα blockchain δίκτυο όπως αυτοί το θέλουν. Αυτή η ευέλικτη αρχιτεκτονική διαφοροποιεί την Hyperledger Fabric από τις άλλες πλατφόρμες blockchain.

Το Hyperledger Fabric έχει σχεδιαστεί για αδειοδοτημένα δίκτυα και επιτρέπει μόνο τη συμμετοχή οντοτήτων με γνωστή ταυτότητα στο σύστημα. Μόνο οι εξουσιοδοτημένοι συμμετέχοντες μπορούν να συμμετέχουν στις συναλλαγές που πραγματοποιούνται στην πλατφόρμα Hyperledger Fabric.

Hyperledger Sawtooth

Η πλατφόρμα Hyperledger Sawtooth αρχικά δημιουργήθηκε χάρη στη συνεισφορά της εταιρείας Intel. Πρόκειται για μια πλατφόρμα με δυνατότητα δυναμικής συναίνεσης που επιτρέπει τους αλγόριθμους συναίνεσης να εναλλάσσονται όσο το δίκτυο είναι σε λειτουργία. Χρησιμοποιώντας το μηχανισμό συναίνεσης Proof of Elapsed Time (PoET), η Hyperledger Sawtooth μπορεί να ενσωματωθεί σε λύσεις ασφαλείας υπολογιστικών συστημάτων που ονομάζονται αξιόπιστα περιβάλλοντα εκτέλεσης (trusted execution environments).

Hyperledger Iroha

Η πλατφόρμα Hyperledger Iroha ιδρύθηκε από την Linux Foundation με σκοπό τη δημιουργία αξιόπιστων, γρήγορων και ασφαλών αποκεντρωμένων εφαρμογών στην τεχνολογία blockchain. Βασίζεται σε έναν πολύ γρήγορο και ασφαλή αλγόριθμο συναίνεσης, τον Yet Another Consensus (YAC), που προστατεύει τους κόμβους από αστοχίες ή από άλλους κακόβουλους κόμβους. Το Hyperledger Iroha χρησιμοποιείται στην Καμπότζη για τη δημιουργία ενός νέου συστήματος πληρωμών παράλληλα με την Εθνική Τράπεζα της Καμπότζης [20], και σε διάφορα άλλα έργα στον τομέα της υγειονομικής περίθαλψης, της χρηματοδότησης και της διαχείρισης ταυτότητας.

Hyperledger Indy

Το Hyperledger Indy παρέχει εργαλεία, βιβλιοθήκες και επαναχρησιμοποιήσιμα στοιχεία για την παροχή ψηφιακών ταυτοτήτων που βασίζονται σε blockchains ή άλλα κατανεμημένα μητρώα, έτσι ώστε να είναι διαλειτουργικά σε διαχειριστικούς τομείς και εφαρμογές. Το Indy είναι διαλειτουργικό με άλλα blockchain ή μπορεί να χρησιμοποιηθεί αυτόνομο τροφοδοτώντας την αποκέντρωση της ταυτότητας.

Hyperledger Besu

Το Hyperledger Besu είναι μία παραλλαγή του Ethereum που έχει σχεδιαστεί για να είναι φιλικό προς τις επιχειρήσεις τόσο για περιπτώσεις δημοσίων όσο και ιδιωτικών δικτύων. Μπορεί επίσης να εκτελεστεί στα δίκτυα δοκιμής του Ethereum όπως το Rinkeby, το Ropsten και το Görli. Το Hyperledger Besu περιλαμβάνει αρκετούς συναινετικούς αλγόριθμους, συμπεριλαμβανομένων των PoW και PoA (IBFT, IBFT 2.0, Etherhash και Clique).

3 Hyperledger fabric

3.1 Γιατί Hyperledger Fabric;

Το Hyperledger Fabric είναι η πρώτη πλατφόρμα κατανεμημένου μητρώου που υποστηρίζει έξυπνα συμβόλαια που συντάσσονται σε γλώσσες προγραμματισμού γενικής χρήσης, όπως η Java, η Go και η Node.js, αντί για περιορισμένες γλώσσες για συγκεκριμένους τομείς (Domain Specific Languages - DSL), όπως η Solidity του Ethereum. Αυτό σημαίνει ότι οι περισσότερες επιχειρήσεις διαθέτουν ήδη το σύνολο των δεξιοτήτων που απαιτούνται για την ανάπτυξη έξυπνων συμβολαίων και δεν απαιτείται πρόσθετη εκπαίδευση για να μάθουν μια νέα γλώσσα.

Το Fabric μπορεί να αξιοποιήσει πρωτόκολλα συναίνεσης που δεν απαιτούν κάποιο κρυπτονόμισμα για την εξόρυξη (mining) ή την τροφοδότηση της εκτέλεσης των έξυπνων συμβολαίων. Η αποφυγή του κρυπτονομίσματος μειώνει ορισμένες σημαντικές πηγές κινδύνου και η απουσία κρυπτογραφικών εξορυκτικών λειτουργιών σημαίνει ότι η πλατφόρμα μπορεί να αναπτυχθεί με περίπου το ίδιο λειτουργικό κόστος με οποιοδήποτε άλλο κατανεμημένο σύστημα.

Το Fabric εισάγει μία νέα ευέλικτη και αποδοτική αρχιτεκτονική. Αυτή η νέα αρχιτεκτονική επιτυγχάνει να βελτιώσει σημαντικά την απόδοση του, δηλαδή την ταχύτητα εισαγωγής νέων μπλοκ στο κατανεμημένο μητρώο. Η αρχιτεκτονική του Fabric ήταν εξαρχής επικεντρωμένη στην παροχή ευελιξίας στους διαχειριστές του. Με αυτόν τον τρόπο δίνει τη δυνατότητα προσαρμογής για οποιαδήποτε σχεδιαστική επιλογή σε κάθε πιθανή περίπτωση χρήσης.

Τα παραπάνω χαρακτηριστικά σε συνδυασμό με την πλούσια τεκμηρίωση (documentation) και την διαδεδομένη χρήση του, καθιστούν το Fabric μία από τις πιο ελκυστικές πλατφόρμες για ιδιωτικά αδειοδοτημένα δίκτυα blockchain που διατίθενται σήμερα.

3.2 Βασικές έννοιες

Το Hyperledger Fabric ή απλώς Fabric είναι ένα παραμετροποιήσιμο και επεκτάσιμο σύστημα ανοιχτού κώδικα για την ανάπτυξη και λειτουργία ιδιωτικών δικτύων blockchain και αποτελεί ένα από τα έργα Hyperledger που φιλοξενείται από το Linux Foundation [19]. Το Fabric είναι ένα κατανεμημένο λειτουργικό σύστημα που εκτελεί κατανεμημένες εφαρμογές γραμμένες σε γλώσσες προγραμματισμού γενικής χρήσης (π.χ. Go, Java, Node.js). Παρακολουθεί με ασφάλεια το ιστορικό εκτέλεσής του σε ένα πανομοιότυπο μητρώο (replicated ledger) όπου μπορεί να κάνει μόνο προσάρτηση και δεν έχει ενσωματωμένο κρυπτονόμισμα.

Κάθε συναλλαγή εκτελείται (υποστηρίζεται) μόνο από ένα υποσύνολο των ομοτίμων, το οποίο επιτρέπει την παράλληλη εκτέλεση και αντιμετωπίζει τον πιθανό κίνδυνο μη-ντετερμινιστικής εκτέλεσης, βασιζόμενη στον μηχανισμό “εκτέλεση-επαλήθευση” του BFT replication [21]. Χρησιμοποιεί μια ευέλικτη πολιτική υποστήριξης (endorsement policy) η οποία καθορίζει ποιοι ομότιμοι, ή πόσοι από αυτούς, πρέπει να εγγυηθούν για τη σωστή εκτέλεση ενός έξυπνου συμβολαίου. Το Fabric επιτυγχάνει ντετερμινιστική εκτέλεση αφού τα αποτελέσματα της συναλλαγής εγγράφονται στην κατάσταση του μητρώου μόνο εφόσον επιτευχθεί συναίνεση για την απόλυτη ταξινόμηση μεταξύ των συναλλαγών, στο βήμα επικύρωσης που εκτελείται από κάθε ομότιμο ξεχωριστά. Επιπλέον, το Fabric δίνει τη δυνατότητα στην εκάστοτε εφαρμογή να

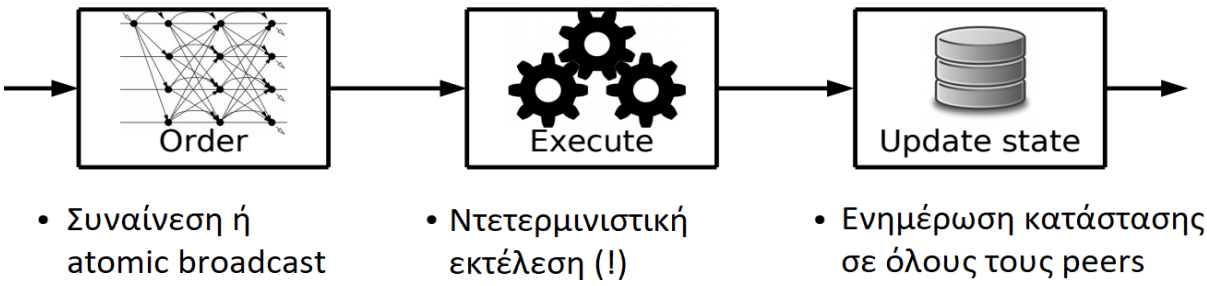
επιλέγει την πολιτική υποστήριξης της. Η υπηρεσία ταξινόμησης του Fabric (release-2.2) ακολουθεί τον αλγόριθμο συναίνεσης CFT (crash fault-tolerant) και πιο συγκεκριμένα δίνει την επιλογή ανάμεσα σε RAFT και KAFKA.

Στην επόμενη ενότητα θα περιγράψουμε την πρωτοποριακή αρχιτεκτονική του Fabric εκτέλεση - ταξινόμηση - επικύρωση (execute-order-validate) η οποία βασίζεται στα ακόλουθα θεμελιώδη συστατικά στοιχεία:

- Μια υπηρεσία ταξινόμησης υλοποιεί ατομική εκπομπή (ενότητα 3.4.2) για ενημερώσεις κατάστασης σε ομότιμους και επιτυγχάνει συναίνεση ως προς την ταξινόμηση των συναλλαγών.
- Μια αρχή έκδοσης πιστοποιητικών (certificate authority) είναι υπεύθυνη για τη συσχέτιση ομότιμων με ηλεκτρονικές κρυπτογραφικές ταυτότητες. Διατηρεί την αδειοδοτημένη φύση του Fabric.
- Μια προαιρετική ομότιμος-σε-ομότιμο υπηρεσία gossip διαδίδει τα αποτελέσματα των μπλοκ από την υπηρεσία ταξινόμησης προς όλους τους ομότιμους.
- Τα έξυπνα συμβόλαια στο Fabric εκτελούνται σε περιβάλλον container για απομόνωση. Μπορούν να γραφτούν σε τυπικές γλώσσες προγραμματισμού, αλλά δεν έχουν άμεση πρόσβαση στην κατάσταση του μητρώου.
- Κάθε ομότιμος διατηρεί τοπικά το μητρώο με τη μορφή ενός blockchain και ένα στιγμιότυπο της πιο πρόσφατης κατάστασης σε μορφή κλειδί-τιμή (key-value store).

3.3 Αρχιτεκτονική

Όλα τα προηγούμενα συστήματα blockchain, με άδεια ή όχι, ακολουθούν την αρχιτεκτονική ταξινόμηση-εκτέλεση (Εικόνα 2). Αυτό σημαίνει ότι το δίκτυο blockchain ταξινομεί τις συναλλαγές πρώτα, χρησιμοποιώντας ένα πρωτόκολλο συναίνεσης, και στη συνέχεια τις εκτελεί με την ίδια σειρά σε όλους τους ομότιμους διαδοχικά. Η αρχιτεκτονική ταξινόμηση-εκτέλεση είναι εννοιολογικά απλή και επομένως χρησιμοποιείται ευρέως. Ωστόσο, έχει πολλά μειονεκτήματα όταν χρησιμοποιείται σε ένα γενικής χρήσης ιδιωτικό blockchain με άδεια. Τα τρία πιο σημαντικά μειονεκτήματα είναι η διαδοχική εκτέλεση (sequential execution), ο μη-ντετερμινιστικός κώδικας και η εμπιστευτικότητα της εκτέλεσης. Η διαδοχική εκτέλεση των συναλλαγών σε όλους τους ομότιμους περιορίζει την αποτελεσματική απόδοση που μπορεί να επιτευχθεί από το blockchain. Η εκτέλεση μη-ντετερμινιστικών συναλλαγών οδηγούν το κατανεμημένο μητρώο σε “fork”, πράγμα που παραβιάζει τη βασική προϋπόθεση ενός blockchain, δηλαδή πως όλοι οι ομότιμοι διατηρούν την ίδια κατάσταση. Για την επίτευξη της εκτέλεσης έξυπνων συμβολαίων με εμπιστευτικότητα τα τυπικά δίκτυα blockchain χρησιμοποιούν κρυπτογραφικές τεχνικές το οποίο με συνδυασμό με την εκτέλεση όλων των έξυπνων συμβολαίων σε όλους τους ομότιμους οδηγεί σε σημαντική επιβάρυνση της απόδοσης και δεν είναι βιώσιμο στην πράξη.

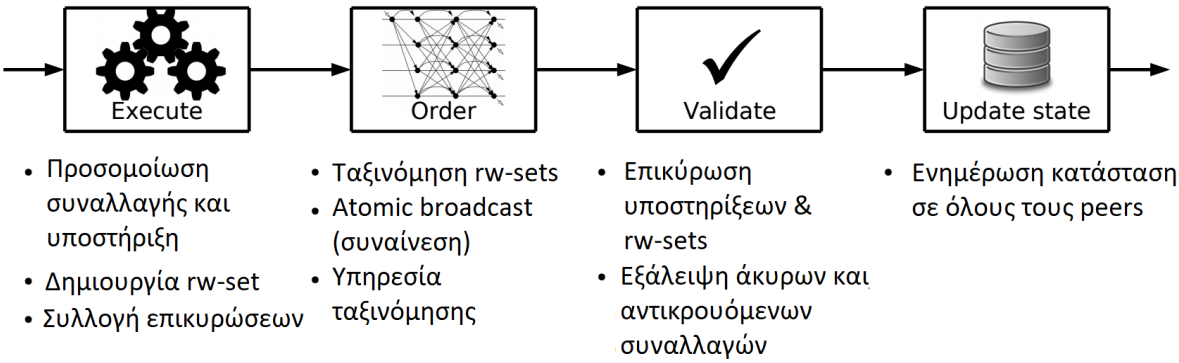


Εικόνα 2: Αρχιτεκτονική ταξινόμηση-εκτέλεση των τυπικών δικτύων blockchain. [22]

Το Fabric εισάγει μια νέα αρχιτεκτονική blockchain με στόχο την ανθεκτικότητα, την ευελιξία, την επεκτασιμότητα και την εμπιστευτικότητα. Η αρχιτεκτονική Fabric εκτέλεση - ταξινόμηση - επικύρωση (Εικόνα 3) σχεδιάστηκε για κατανεμημένη εκτέλεση μη αξιόπιστου κώδικα σε μη αξιόπιστο περιβάλλον. Διαχωρίζει τη ροή συναλλαγών σε τρία βήματα, τα οποία μπορούν να εκτελεστούν σε διαφορετικές οντότητες του συστήματος: (1) εκτέλεση μιας συναλλαγής και έλεγχος της ορθότητάς της, με αποτέλεσμα την έγκρισή της (που αντιστοιχεί στην "επικύρωση συναλλαγής" σε άλλα blockchain). (2) ταξινόμηση μέσω ενός πρωτοκόλλου συναίνεσης, ανεξάρτητα από τη σημασιολογία των συναλλαγών και (3) επικύρωση συναλλαγής με βάση την πολιτική υποστήριξης κάθε chaincode, η οποία αποτρέπει περιπτώσεις “race condition” λόγω ταυτόχρονης εκτέλεσης.

Μια κατανεμημένη εφαρμογή για Fabric αποτελείται από δύο μέρη:

- Ένα έξυπνο συμβόλαιο, που ονομάζεται chaincode, είναι προγραμματιστικός κώδικας που υλοποιεί τη λογική της εφαρμογής και εκτελείται κατά τη φάση της εκτέλεσης. Το chaincode είναι το κεντρικό μέρος μιας κατανεμημένης εφαρμογής στο Fabric και μπορεί να γραφτεί από έναν μη αξιόπιστο προγραμματιστή. Υπάρχουν ειδικά chaincode υπεύθυνα για τη διαχείριση του συστήματος blockchain και τη διατήρηση παραμέτρων, που συλλογικά ονομάζονται chaincode συστήματος (Ενότητα 3.4.6).
- Μια πολιτική υποστήριξης, που αξιολογείται στη φάση επικύρωσης. Οι πολιτικές υποστήριξης δεν μπορούν να επιλεγούν ή να τροποποιηθούν από μη αξιόπιστους προγραμματιστές εφαρμογών. Μια πολιτική υποστήριξης λειτουργεί ως στατική βιβλιοθήκη για την επικύρωση συναλλαγών στο Fabric, η οποία μπορεί απλώς να παραμετροποιηθεί από το chaincode. Μόνο οι καθορισμένοι διαχειριστές μπορούν να έχουν άδεια να τροποποιούν τις πολιτικές υποστήριξης μέσω λειτουργιών διαχείρισης συστήματος. Μια τυπική πολιτική υποστήριξης επιτρέπει στο chaincode να προσδιορίζει τους υποστηρικτές για μια συναλλαγή με τη μορφή ενός συνόλου ομοτίμων που είναι απαραίτητοι για την υποστήριξη. Χρησιμοποιεί μια λογική έκφραση (logical expression) σε σύνολα, όπως "τρεις στους πέντε" ή "(A ∧ B) ∨ C."



Εικόνα 3: Αρχιτεκτονική εκτέλεση-ταξινόμηση-επικύρωση του Fabric [22].

Ένας πελάτης (client) στέλνει συναλλαγές στους ομότιμους που έχουν καθοριστεί από την πολιτική υποστήριξης. Στη συνέχεια, κάθε συναλλαγή εκτελείται από συγκεκριμένους ομότιμους και η έξοδος της καταγράφεται, αυτό το βήμα ονομάζεται υποστήριξη (endorsement). Μετά την εκτέλεση, οι συναλλαγές εισέρχονται στη φάση ταξινόμησης, η οποία χρησιμοποιεί ένα πρωτόκολλο συναίνεσης για να παράγει μια πλήρως ταξινομημένη ακολουθία εκτέλεσης υποστηριγμένων συναλλαγών, ομαδοποιημένων σε μπλοκ τα οποία μεταδίδονται σε όλους τους ομότιμους. Το Fabric ταξινομεί εξόδους συναλλαγών σε συνδυασμό με τις εξαρτήσεις της κατάστασης (state dependencies), όπως υπολογίζονται κατά τη φάση εκτέλεσης. Έπειτα, κάθε ομότιμος επικυρώνει τις αλλαγές κατάστασης (state changes) από τις υποστηριγμένες συναλλαγές σύμφωνα με την πολιτική υποστήριξης και τη συνέπεια της εκτέλεσης στη φάση επικύρωσης. Όλοι οι ομότιμοι επικυρώνουν τις συναλλαγές με την ίδια σειρά, η επικύρωση είναι ντετερμινιστική. Υπό αυτή την έννοια, το Fabric εισάγει ένα νέο υβριδικό πρότυπο replication στο βυζαντινό μοντέλο, το οποίο συνδυάζει το παθητικό replication (τον προ συναινετικό υπολογισμό των ενημερώσεων κατάστασης) και το ενεργό replication (την επικύρωση, μετά τη συναίνεση, των αποτελεσμάτων εκτέλεσης και των αλλαγών κατάστασης).

Ένα Fabric blockchain αποτελείται από ένα σύνολο κόμβων που σχηματίζουν ένα δίκτυο. Καθώς το Fabric είναι ένα ιδιωτικό αδειοδοτημένο blockchain, όλοι οι κόμβοι που συμμετέχουν στο δίκτυο έχουν μια ταυτότητα, όπως παρέχεται από μια αρχή έκδοσης πιστοποιητικών.

Οι κόμβοι σε ένα δίκτυο Fabric αναλαμβάνουν έναν από τρεις ρόλους:

- Οι πελάτες κόμβοι υποβάλλουν προτάσεις συναλλαγών για εκτέλεση, βοηθούν στην ενορχήστρωση της φάσης εκτέλεσης και, τέλος, μεταδίδουν συναλλαγές για ταξινόμηση.
- Οι ομότιμοι κόμβοι εκτελούν προτάσεις συναλλαγών και επικυρώνουν συναλλαγές. Όλοι οι ομότιμοι διατηρούν το μητρώο του blockchain καθώς και την κατάσταση, που είναι μια συνοπτική αναπαράσταση της τελευταίας κατάστασης του μητρώου. Οι ομότιμοι που εκτελούν τις προτάσεις συναλλαγών ονομάζονται ομότιμοι υποστήριξης (ή απλώς υποστηρικτές), όπως καθορίζεται από την πολιτική του έξυπνου συμβολαίου στο οποίο ανήκει η συναλλαγή. Ένας οργανισμός εκπροσωπείται στο δίκτυο από έναν ή περισσότερους ομότιμους κόμβους. Κάθε ομότιμος κόμβος διατηρεί ένα αντίγραφο του μητρώου κάθε καναλιού στα οποία συμμετέχει καθώς και μία βάση δεδομένων με τη τελευταία κατάσταση αυτών. Κάθε ομότιμος κόμβος έχει έναν διαχειριστή που καταχωρείται στη βάση δεδομένων της αρχής έκδοσης πιστοποιητικών του οργανισμού. Σε έναν ομότιμο κόμβο μπορούμε

να εγκαταστήσουμε όσα chaincode επιθυμούμε. Κάθε ένα από αυτά λειτουργεί σε ένα δικό του container και είναι προσβάσιμα μόνο από τον ομότιμο. Κάθε κλήση συνάρτησης ενός chaincode πρέπει να εκτελεστεί από κάθε ομότιμο που το έχει εγκαταστήσει.

- Οι κόμβοι ταξινομητές (orderers) είναι οι κόμβοι που σχηματίζουν συλλογικά την υπηρεσία ταξινόμησης. Εν ολίγοις, η υπηρεσία ταξινόμησης καθορίζει τη σειρά εκτέλεσης όλων των συναλλαγών στο Fabric, όπου κάθε συναλλαγή περιέχει ενημερώσεις και εξαρτήσεις κατάστασης που υπολογίζονται κατά τη φάση εκτέλεσης, μαζί με κρυπτογραφικές υπογραφές των υποστηρικτών της. Οι ταξινομητές δεν διατηρούν το μητρώο του blockchain και δεν συμμετέχουν ούτε στην εκτέλεση ούτε στην επικύρωση των συναλλαγών.

Ένα δίκτυο Fabric υποστηρίζει πολλαπλά blockchain που συνδέονται στην ίδια υπηρεσία ταξινόμησης ταυτόχρονα. Κάθε τέτοιο blockchain ονομάζεται κανάλι και μπορεί να έχει διαφορετικούς ομότιμους ως μέλη. Τα κανάλια μπορούν να χρησιμοποιηθούν για τον διαχωρισμό της κατάστασης του δικτύου blockchain, αλλά η συναίνεση μεταξύ των καναλιών δεν συντονίζεται και η σειρά εκτέλεσης των συναλλαγών σε κάθε κανάλι είναι ξεχωριστή από τις άλλες. Σε ορισμένες περιπτώσεις όπου θεωρούνται όλοι οι ταξινομητές ως αξιόπιστοι μπορεί επίσης να εφαρμοστεί έλεγχος πρόσβασης μεταξύ καναλιών για ομότιμους.

3.4 Βασικά συστατικά στοιχεία του Fabric

Το Fabric είναι γραμμένο στη γλώσσα προγραμματισμού Go και χρησιμοποιεί το πρωτόκολλο gRPC [23] για επικοινωνία μεταξύ πελατών, ομότιμων και ταξινομητών. Στη συνέχεια περιγράφουμε ορισμένα σημαντικά στοιχεία με περισσότερες λεπτομέρειες.

3.4.1 Αρχή Έκδοσης Πιστοποιητικών

Η αρχή έκδοσης πιστοποιητικών είναι υπεύθυνη για την έκδοση πιστοποιητικών κόμβων (πελατών, διαχειριστών, ομότιμων και ταξινομητών) που χρησιμοποιούνται για αυθεντικοποίηση και προσβασιμότητα. Εφόσον το Fabric είναι αδειοδοτημένο, όλες οι αλληλεπιδράσεις μεταξύ των κόμβων πραγματοποιούνται μέσω μηνυμάτων που αυθεντικοποιούνται, συνήθως με ψηφιακές υπογραφές.

Η αρχή έκδοσης πιστοποιητικών επιτρέπει τη λειτουργία μιας ομοσπονδίας διαχείρισης ταυτότητας, για παράδειγμα, όταν πολλοί οργανισμοί λειτουργούν ένα δίκτυο blockchain. Κάθε οργανισμός εκδίδει ταυτότητες στα δικά του μέλη και κάθε ομότιμος αναγνωρίζει μέλη όλων των οργανισμών. Αυτό μπορεί να επιτευχθεί με πολλαπλές αρχές έκδοσης πιστοποιητικών, για παράδειγμα, δημιουργώντας μία αρχή έκδοσης πιστοποιητικών για κάθε οργανισμό. Το σύνολο των αρχών έκδοσης πιστοποιητικών πρέπει να διασφαλίζει ότι όλοι οι κόμβοι, ειδικά όλοι οι ομότιμοι, αναγνωρίζουν τις ίδιες ταυτότητες και ελέγχους ταυτότητας ως έγκυρες.

3.4.2 Υπηρεσία Ταξινόμησης

Η υπηρεσία ταξινόμησης διαχειρίζεται πολλά κανάλια. Σε κάθε κανάλι παρέχει τις 2 ακόλουθες βασικές υπηρεσίες:

1. Ατομική εκπομπή για την ταξινόμηση συναλλαγών.
2. Αναδιαμόρφωση ενός καναλιού, όταν τα μέλη του τροποποιούν το κανάλι εκπέμποντας μια συναλλαγή ενημέρωσης της διαμόρφωσης καναλιού.

Ένας ταξινομητής μαζεύει συναλλαγές και με τη χρήση της ατομικής εκπομπής τις ταξινομεί και σχηματίζει μπλοκ. Ένα μπλοκ συμπληρώνεται μόλις πληρούνται μία από τις τρεις προϋποθέσεις: (1) το μπλοκ περιέχει τον καθορισμένο μέγιστο αριθμό συναλλαγών, (2) το μπλοκ έχει φτάσει στο μέγιστο μέγεθος (σε byte) ή (3) έχει παρέλθει ένας συγκεκριμένος χρόνος από τη λήψη της πρώτης συναλλαγής ενός νέου μπλοκ.

3.4.3 Peer Gossip

Ένα πλεονέκτημα του διαχωρισμού των φάσεων εκτέλεσης, ταξινόμησης και επικύρωσης είναι ότι μπορούν να κλιμακωθούν ανεξάρτητα. Η συναίνεση δεν μπορεί να κλιμακωθεί με την προσθήκη περισσότερων κόμβων [24][25], αντίθετα, η απόδοση θα μειωθεί. Ωστόσο, δεδομένου ότι η ταξινόμηση και η επικύρωση είναι ανεξάρτητες, μας ενδιαφέρει η αποτελεσματική μετάδοση των αποτελεσμάτων εκτέλεσης σε όλους τους ομότιμους για επικύρωση, μετά τη φάση ταξινόμησης. Αυτό, μαζί με τη μεταφορά κατάστασης σε νεοεισερχόμενους ομότιμους και ομότιμους που αποσυνδέθηκαν για μεγάλο χρονικό διάστημα, είναι ακριβώς ο στόχος του μηχανισμού gossip. Το Fabric gossip χρησιμοποιεί την epidemic multicast [26] για αυτόν το σκοπό. Τα μπλοκ υπογράφονται από την υπηρεσία ταξινόμησης. Αυτό σημαίνει ότι ένας ομότιμος μπορεί, μόλις λάβει όλα τα μπλοκ, να συναρμολογήσει ανεξάρτητα το blockchain και να επαληθεύσει την ακεραιότητά του.

Το gossip βασίζεται στο gRPC και χρησιμοποιεί το TLS με αμοιβαίο έλεγχο ταυτότητας, το οποίο επιτρέπει σε κάθε πλευρά να συνδέει τα διαπιστευτήρια TLS με την ταυτότητα του απομακρυσμένου ομότιμου. Ο μηχανισμός gossip ενός ομότιμου διατηρεί μια ενημερωμένη λίστα των ενεργών ομότιμων του συστήματος. Επιπλέον, ένας ομότιμος μπορεί να επανασυνδεθεί μετά από συντριβή ή διακοπή δικτύου.

Προκειμένου να μειωθεί το φορτίο αποστολής μπλοκ από τους κόμβους ταξινόμησης προς το δίκτυο, το πρωτόκολλο gossip εκλέγει επίσης έναν ομότιμο ηγέτη που συλλέγει μπλοκ από την υπηρεσία ταξινόμησης για λογαριασμό τους και τα διανέμει σε αυτούς. Αυτός ο μηχανισμός είναι ανθεκτικός σε αποτυχίες ηγετών.

3.4.4 Μητρώο

Κάθε ομότιμος διατηρεί το μητρώο και την κατάσταση αυτού και επιτρέπει τις φάσεις προσομοίωσης, επικύρωσης και ενημέρωσης του μητρώου. Σε γενικές γραμμές, αποτελείται από μία αποθήκη από μπλοκ και έναν μηχανισμό διαχείρισης συναλλαγών.

Ο χώρος αποθήκευσης μπλοκ του μητρώου διατηρεί μπλοκ συναλλαγών και υλοποιείται ως ένα σύνολο αρχείων μόνο προσάρτησης. Δεδομένου ότι τα μπλοκ είναι αμετάβλητα και φτάνουν με καθορισμένη σειρά, μια δομή μόνο με προσάρτηση δίνει τη μέγιστη απόδοση. Επιπλέον, η αποθήκη μπλοκ διατηρεί μερικούς δείκτες για τυχαία προσπέλαση σε ένα μπλοκ ή σε μια συναλλαγή σε ένα μπλοκ.

Ο μηχανισμός διαχείρισης συναλλαγών διατηρεί την πιο πρόσφατη κατάσταση σε μια αποθήκη κλειδιού-τιμής-έκδοσης. Αποθηκεύει μία πλειάδα της μορφής (κλειδί, τιμή, έκδοση) για κάθε μοναδικό κλειδί που είναι αποθηκευμένο από οποιονδήποτε chaincode, και περιέχει την πιο πρόσφατα αποθηκευμένη τιμή και την τελευταία έκδοση του. Η αποθήκη κλειδιού-τιμής-έκδοσης υλοποιείται με χρήση LevelDB (σε Go) [27] ή με χρήση Apache CouchDB [28].

Στη φάση προσομοίωσης ο μηχανισμός διαχείρισης συναλλαγών παρέχει ένα σταθερό στιγμιότυπο της τελευταίας κατάστασης στη συναλλαγή. Στη φάση επικύρωσης συναλλαγής, ο μηχανισμός διαχείρισης συναλλαγών επικυρώνει όλες τις συναλλαγές ενός μπλοκ διαδοχικά. Αυτό ελέγχει εάν μια συναλλαγή έρχεται σε διένεξη με οποιαδήποτε προηγούμενη συναλλαγή (εντός του μπλοκ ή παλαιότερη). Στη φάση ενημέρωσης το

μητρώο έχει ανοχή σε πιθανές καταστροφές ομότιμων με χρήση ενός μηχανισμού “savepoint” που χρησιμοποιείται για ανάκτηση των δεικτών και της τελευταίας κατάστασης που χάθηκαν.

3.4.5 Εκτέλεση Chaincode

Το chaincode εκτελείται σε ένα περιβάλλον χαλαρά συνδεδεμένο με έναν ομότιμο, το οποίο υποστηρίζει προς το παρόν τις γλώσσες προγραμματισμού Go, Java και Node.js.

Κάθε chaincode εφαρμογής εκτελείται μέσα σε ένα περιβάλλον container Docker, το οποίο απομονώνει τα chaincode μεταξύ τους και από τον ομότιμο. Αυτό απλοποιεί τη διαχείριση του κύκλου ζωής των chaincode (δηλαδή εκκίνηση, διακοπή ή ματαίωση). Το chaincode και ο ομότιμος επικοινωνούν χρησιμοποιώντας μηνύματα gRPC. Μέσω αυτής της χαλαρής σύνδεσης, ο ομότιμος είναι αγνωστικιστής της πραγματικής γλώσσας στην οποία υλοποιείται το chaincode.

Σε αντίθεση με το chaincode εφαρμογής, το chaincode του συστήματος εκτελείται απευθείας στον ομότιμο. Το chaincode συστήματος μπορεί να εφαρμόσει συγκεκριμένες λειτουργίες που απαιτούνται από το Fabric και μπορεί να χρησιμοποιηθεί σε περιπτώσεις όπου η απομόνωση μεταξύ των chaincode εφαρμογής είναι υπερβολικά περιοριστική.

3.4.6 Διαμόρφωση και Chaincode Συστήματος

Το Fabric προσαρμόζεται μέσω της διαμόρφωσης καναλιών και μέσω ειδικών chaincode, γνωστών ως chaincode συστήματος. Θυμηθείτε ότι κάθε κανάλι στο Fabric σχηματίζει ένα blockchain. Η διαμόρφωση ενός καναλιού διατηρείται σε μεταδεδομένα που παραμένουν σε ειδικά μπλοκ διαμόρφωσης. Κάθε μπλοκ διαμόρφωσης περιέχει τη διαμόρφωση ολόκληρου καναλιού και δεν περιέχει άλλες συναλλαγές. Κάθε δίκτυο blockchain ξεκινά με ένα μπλοκ διαμόρφωσης γνωστό ως “genesis block”.

Η διαμόρφωση ενός καναλιού μπορεί να ενημερωθεί χρησιμοποιώντας μια συναλλαγή ενημέρωσης διαμόρφωσης καναλιού. Αυτή η συναλλαγή περιέχει μια αναπαράσταση των αλλαγών που πρέπει να γίνουν στη διαμόρφωση, καθώς και ένα σύνολο υπογραφών. Οι κόμβοι ταξινόμησης αξιολογούν εάν η ενημέρωση είναι έγκυρη χρησιμοποιώντας την τρέχουσα διαμόρφωση για να επαληθεύσουν ότι οι τροποποιήσεις είναι εξουσιοδοτημένες με χρήση των υπογραφών. Στη συνέχεια, οι ταξινομητές δημιουργούν ένα νέο μπλοκ διαμόρφωσης, το οποίο ενσωματώνει τη νέα διαμόρφωση και τη συναλλαγή ενημέρωσης διαμόρφωσης. Οι ομότιμοι που λαμβάνουν αυτό το μπλοκ επικυρώνουν εάν η ενημέρωση διαμόρφωσης είναι εξουσιοδοτημένη με βάση την τρέχουσα διαμόρφωση, εάν επικυρωθεί, ενημερώνουν την τρέχουσα διαμόρφωσή τους.

4 Περίπτωση Χρήσης: Φορείς Σωρευτικής Εκπροσώπησης (ΦΟΣΕ) και η ανάγκη αντικατάστασης του έμπιστου τρίτου φορέα

4.1 Το ενεργειακό σύστημα υπό μετάβαση

Οι επιπτώσεις της κλιματικής αλλαγής γίνονται πλέον αντιληπτές στην καθημερινή ζωή των ανθρώπων σε όλο τον πλανήτη. Οι στόχοι που έχουν τεθεί σε ευρωπαϊκό επίπεδο για το 2030 περιγράφουν ένα ενεργειακό σύστημα με σημαντικά χαμηλότερες εκπομπές διοξειδίου του άνθρακα. Ο ηλεκτρικός τομέας συνεισφέρει σημαντικά στη διαδικασία απανθρακοποίησης, με την παραγωγή ηλεκτρικής ενέργειας από Ανανεώσιμων Πηγών Ενέργειας (ΑΠΕ) να πρωταγωνιστεί. Είμαστε πλέον στην εποχή όπου το κόστος παραγωγής ηλεκτρικής ενέργειας από ΑΠΕ είναι ευθέως συγκρίσιμο και ανταγωνιστικό ως προς το κόστος παραγωγής των συμβατικών σταθμών ηλεκτρικής ενέργειας. Οι ΑΠΕ διεκδικούν ισότιμο ρόλο στο ηλεκτρικό σύστημα και ισότιμους όρους συμμετοχής στην αγορά ηλεκτρικής ενέργειας και το θεσμικό πλαίσιο αναπροσαρμόζεται στη συγκεκριμένη προοπτική. Οι Φορείς Σωρευτικής Εκπροσώπησης (ΦΟΣΕ) είναι οι νέες οντότητες της αγοράς ηλεκτρικής ενέργειας μέσω των οποίων θα πραγματοποιηθεί η πλήρης ενσωμάτωση των μονάδων ΑΠΕ στην αγορά ηλεκτρικής ενέργειας. Οι ΦΟΣΕ αποτελούν εξειδικευμένες καινοτόμες εταιρείες που χρησιμοποιούν σύγχρονα εργαλεία και καλούνται να βελτιστοποιήσουν την εμπορική διαχείριση των μονάδων πράσινης ενέργειας ελαχιστοποιώντας το κόστος εξισορρόπησης με την ακριβή πρόβλεψη της παραγωγής τους.

Καθώς λοιπόν τα επόμενα χρόνια, η συμμετοχή των ανανεώσιμων πηγών ενέργειας στην αγορά θα γίνεται με ανταγωνιστικούς όρους αγοράς, οι ΦΟΣΕ θα καθίστανται όλο και πιο αναγκαίοι για τη διαχείριση πολλών έργων ΑΠΕ, ώστε να αυξάνεται το όφελος για τον παραγωγό.

Στο θεσμικό επίπεδο η πρώτη μεγάλη αλλαγή έγινε με το νόμο 4414/2016 [29] και γίνεται το πρώτο μεγάλο βήμα καθώς τα έργα ΑΠΕ που κατασκευάζονται έχοντας κερδίσει στους διαγωνισμούς της Ρυθμιστικής Αρχής Ενέργειας (ΡΑΕ) εφόσον έχουν ισχύ πάνω από 3 MW για τα αιολικά και πάνω από 500 kw για τα φωτοβολταϊκά, αναλαμβάνουν την υποχρέωση να συμμετάσχουν στη χονδρεμπορική αγορά και ταυτόχρονα αναλαμβάνουν υποχρεώσεις εξισορρόπησης. Αυτό σημαίνει πως εάν δηλώσουν παραπάνω ή λιγότερη ισχύ διαθέσιμη από την πραγματική θα επωμιστούν το κόστος που προκύπτει από την απόκλιση.

Στο εξωτερικό και σε πιο ώριμες αγορές, όπως η ιταλική, αρκετές εταιρείες ΑΠΕ προτιμούν να εντάσσουν τα έργα τους με ανταγωνιστικούς όρους στην αγορά και όχι μέσω διαγωνισμών, καθώς διαπιστώνουν ότι αφενός οι τιμές στις δημοπρασίες κινούνται πτωτικά και δεν προσφέρουν ικανοποιητικές αποδόσεις και αφετέρου ότι η συμμετοχή στη χονδρεμπορική αγορά για μεγάλα χαρτοφυλάκια ΑΠΕ μπορεί να αποφέρει μεγαλύτερα έσοδα και αποδόσεις. Αυτές οι μεγαλύτερες αποδόσεις, επιτυγχάνονται μέσω των ΦΟΣΕ οι οποίοι διαχειρίζονται μεγάλο αριθμό μονάδων ΑΠΕ, με γεωγραφική διασπορά, και περιορίζουν την αβεβαιότητα και τη μεταβλητότητα της παραγωγής των ΑΠΕ, μπορούν να υποβάλουν ανταγωνιστικές προσφορές που εξασφαλίζουν την καθημερινή συμμετοχή του χαρτοφυλακίου στο ενεργειακό χρηματιστήριο και να αποφέρουν μεγαλύτερα έσοδα. Το ίδιο προβλέπεται να γίνει και στην Ελλάδα με την εφαρμογή του μοντέλου στόχος το οποίο θα μεταφέρει την ευθύνη της εξισορρόπησης της αγοράς από τον Ανεξάρτητο Διαχειριστή

Μεταφοράς Ηλεκτρικής Ενέργειας (ΑΔΜΗΕ) στους παραγωγούς ΑΕΠ, ενώ η εμπορική διαχείριση των πράσινων μονάδων θα γίνεται από τους ΦΟΣΕ. Οι ΦΟΣΕ θα περιορίζουν την απόκλιση μεταξύ πρόβλεψης και πραγματικής παραγωγής και συνακόλουθα τη μεταβλητότητα της παραγωγής των σταθμών ΑΠΕ. Αυτή η μεταβλητότητα και η αβεβαιότητα θα περιοριστεί ακόμη περισσότερο στο μέλλον όταν προωθηθούν και εφαρμοστούν στην αγορά των ΑΠΕ συστήματα αποθήκευσης ενέργειας. [30]

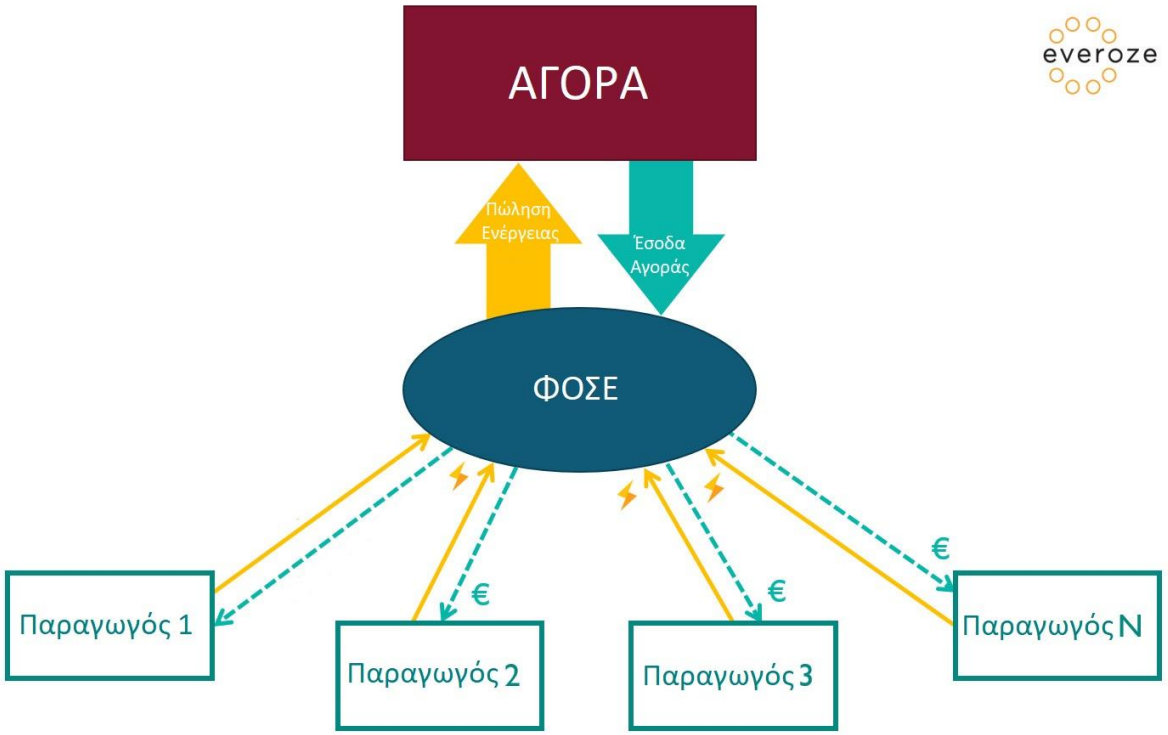
Ακόμα ας σημειωθεί πως η πρώτη εταιρία στον τομέα αυτό ήταν η Optimus Energy, στην οποία συμμετέχει η ΤΕΡΝΑ με άδεια εκπροσώπησης 350 MW. Αδειες κατέχουν επίσης η Mytilineos [31] για 500 MW, η Eunice για 300 MW, τα ΕΛΠΕ Ανανεώσιμες για 300 MW, η NRG, του ομίλου της Motor Oil για 200 MW, η Forena Energy [32] για 200 MW, η Sentrade για 200 MW καθώς και άλλες εταιρίες.

4.2 Τι είναι και πως λειτουργεί ένας ΦΟΣΕ

Ένας Φορέας Σωρευτικής Εκπροσώπησης είναι ένας οργανισμός που εκπροσωπεί παραγωγούς Ανανεώσιμων Πηγών Ενέργειας στην ελεύθερη αγορά ηλεκτρικής ενέργειας. Πιο συγκεκριμένα, θα ασχοληθούμε με ΦΟΣΕ που εκπροσωπούν παραγωγούς ηλεκτρικής ενέργειας μέσω αιολικών πάρκων.

Αιολικό πάρκο ή Αιολικός Σταθμός Παραγωγής Ηλεκτρικής Ενέργειας (ΑΣΠΗΕ) ονομάζεται η χερσαία ή θαλάσσια έκταση στην οποία έχει τοποθετηθεί ένας αριθμός ανεμογεννητριών με σκοπό τη μετατροπή της κινητικής ενέργειας του ανέμου σε ηλεκτρική.

Ένας ΦΟΣΕ (όχι απαραίτητα παραγωγός ο ίδιος) συνάπτει συμφωνίες με παραγωγούς να συμμετέχουν σε αυτόν. Οι αποφάσεις που αφορούν τη λειτουργία του ΦΟΣΕ μπορούν να λαμβάνονται με διαδικασίες εταιρικής διακυβέρνησης. Ενα κρίσιμο σημείο είναι αυτό που αφορά την αναπροσαρμογή των ποσοστών συμμετοχής και γενικά τη διαδικασία αποζημιώσεων των συμμετεχόντων παραγωγών στον ΦΟΣΕ. Η αγορά ενός ΦΟΣΕ είναι υποσύνολο της συνολικής αγοράς παροχής ηλεκτρικής ενέργειας. Με βάση τις δηλώσεις των άλλων παραγωγών που ανήκουν στον ίδιο ΦΟΣΕ υπολογίζεται το μερίδιο αγοράς σε ποσοστό που καλύπτει ο καθένας, τα οποία αθροίζουν στο 100%.



Εικόνα 4: Μερίδιο αγοράς του ΦΟΣΕ και του παραγωγού. [33]

Ας μελετήσουμε λίγο πιο αναλυτικά τη διαδικασία αυτή. Ο ΦΟΣΕ στην αρχή του μήνα πρέπει να κάνει μια δήλωση της παροχής ενέργειας σε MWh που θα παρέχει στη συνολική αγορά. Η δήλωση αυτή πρέπει να είναι ακριβής για να μην έχει κυρώσεις. Το ποσό αυτό υπολογίζεται από το άθροισμα των δηλώσεων της πρόβλεψης της παραγωγής ενέργειας του κάθε παραγωγού του. Επομένως, στην αρχή του κάθε μήνα ένας παραγωγός δηλώνει τη πρόβλεψή του για το ποσό ενέργειας σε MWh που θα παράξει για όλο το μήνα. Ο ΦΟΣΕ πληρώνει τον κάθε παραγωγό με βάση τη δήλωσή του. Έπειτα κάθε παραγωγός, κάθε μέρα καταγράφει τη παραγωγή ενέργειας σε MWh κάθε διαστήματος 15 λεπτών.

Παράδειγμα: Έστω ένας ΦΟΣΕ που αντιπροσωπεύει 4 παραγωγούς (Org1, Org2, Org3, Org4). Οι δηλώσεις των παραγωγών στην αρχή του μήνα και η παραγωγή στο τέλος του μήνα είναι οι εξής:

Πίνακας 3: Περίπτωση χρήσης: Δήλωση και παραγωγή ηλεκτρικής ενέργειας των παραγωγών ενός ΦΟΣΕ.

	Org1	Org2	Org3	Org4
Δήλωση (MWh)	114	120	180	186
Δήλωση (%)	19%	20%	30%	31%
Παραγωγή (MWh)	85	110	140	165

Παραγωγή (%)	17%	22%	28%	33%
Ισχυρισμός (MWh)	114	120	180	186

Παρατηρούμε ότι κάθε παραγωγός παρήγαγε τελικά λιγότερο από αυτό που είχε δηλώσει. Επομένως πρέπει κάθε ένας τους να επιστρέψει τη διαφορά πίσω στο ΦΟΣΕ. Όπως φαίνεται στη τελευταία γραμμή του Πίνακα 3 ο παραγωγός Org1 ισχυρίζεται ότι παρήγαγε το ποσό που είχε δηλώσει και πως δεν χρωστάει χρήματα στο ΦΟΣΕ. Για την λύση αυτού του προβλήματος υπάρχει ανάγκη για έναν έμπιστο τρίτο φορέα.

4.3 Το πρόβλημα του έμπιστου τρίτου φορέα

Ένας έμπιστος τρίτος φορέας συλλέγει τα δεδομένα των εμπλεκόμενων παραγωγών. Έτσι εκκινεί τη διαδικασία επανυπολογισμού του ποσού παραγωγής του κάθε ενός. Με το αποτέλεσμα της διαδικασίας αυτής κρίνει τα οφειλόμενα χρήματα από και προς το ΦΟΣΕ. Εύκολα παρατηρεί κανείς τα προβλήματα της χρήσης ενός έμπιστου τρίτου φορέα. Το βασικότερο πρόβλημα είναι ότι ο έμπιστος τρίτος φορέας αποτελείται από ανθρώπους που θα κινήσουν την όλη διαδικασία επανυπολογισμού των ποσοστών και σαν αποτέλεσμα δεν είναι αδιάφθορος. Επιπρόσθετα τα δεδομένα του κάθε παραγωγού είναι ιδιωτικά και πιθανώς ευαίσθητα για το κάθε παραγωγό. Τέλος, ο έμπιστος τρίτος φορέας δεν μπορεί να καλεστεί αν δεν είναι και οι δύο πλευρές πρόθυμες για διαπραγμάτευση. Από το παράδειγμα συμπεραίνουμε πως αν ο παραγωγός Org1 δεν θέλει να προβεί σε διαπραγμάτευση, ενώ ο ΦΟΣΕ επιζητά διαπραγμάτευση, ο έμπιστος τρίτος φορέας δεν μπορεί να καλεστεί. Παρατηρούμε ότι παρόμοια διαδικασία ακολουθείται στη περίπτωση που ο ΦΟΣΕ ισχυρίζεται λανθασμένα ποσά παραγωγής.

4.4 Η λύση του προβλήματος

Η πρόταση για τη λύση του προβλήματος είναι η εξής: Κατασκευή DAO με μέλη τους παραγωγούς Org1, Org2, Org3, Org4 για την αντικατάσταση του έμπιστου τρίτου φορέα. Ως μέλη του DAO κάθε παραγωγός έχει μία θέση στο διοικητικό συμβούλιο και δικαίωμα ψήφου σε αυτό. Σε περίπτωση ανάγκης επανυπολογισμού των ποσοστών παραγωγής κάποιος παραγωγός εκκινεί μία νέα ψηφοφορία. Κάθε παραγωγός ψηφίζει στο δικό του χρόνο έως ότου τουλάχιστον 3 ψήφοι να είναι υπέρ ή κατά του επανυπολογισμού. Στη περίπτωση του επανυπολογισμού το σύστημα διαβάζει τα δεδομένα όλων των παραγωγών και υπολογίζει τα νέα ποσοστά.

Στην δική μας περίπτωση ο οργανισμός αυτός βρίσκεται και ζει πάνω στο blockchain και οι αποφάσεις - λειτουργίες του είναι μοντελοποιημένα πάνω σε έξυπνα συμβόλαια. Οπότε κάθε παραγωγός θα μπορεί να ψηφίσει υπέρ η κατά στον επανυπολογισμό των ποσοστών, βλέποντας μόνο το δικό του ιστορικό και γνωρίζοντας πληροφορίες για το δικό του σύστημα χωρίς να έχει πρόσβαση στον άλλων, εξού και η ανάγκη χρήσης ιδιωτικών αδειοδοτημένων blockchain.

Η περίπτωση χρήσης που θα μελετήσουμε θα διερευνήσει μια υλοποίηση της διαδικασίας λήψης αποφάσεων στον ΦΟΣΕ με χρήση blockchain ως Αυτόνομο Αποκεντρωμένο Οργανισμό.

5 Υλοποίηση: Δημιουργία DAO για τη διαμοίραση της παραγωγής ηλεκτρικής ενέργειας μεταξύ παρόχων

Οι λειτουργικές απαιτήσεις της αποκεντρωμένης εφαρμογής είναι οι παρακάτω. Το δίκτυο blockchain αποτελείται από 4 οργανισμούς. Κάθε οργανισμός εισάγει στο blockchain τη πρόβλεψη του για το ποσό παραγωγής ενέργειας σε MWh, που θα παράξει σε ένα χρονικό διάστημα. Έπειτα κάθε 15 λεπτά της ώρας, κάθε οργανισμός καταγράφει στο blockchain το ποσό ενέργειας σε MWh που παράχθηκε στο διάστημα αυτό. Τα δεδομένα του κάθε οργανισμού είναι ιδιωτικά, και επομένως ορατά μόνο σε αυτόν. Οποιοσδήποτε οργανισμός έχει τη δυνατότητα να εκκινήσει ψηφοφορία για τον επανυπολογισμό των ποσοστών της παραγωγής. Εάν η πλειονότητα των οργανισμών ψηφίσει θετικά, τότε αυτόματα επανυπολογίζονται τα νέα ποσοστά και γίνονται ορατά στους ίδιους.

Για την ικανοποίηση των παραπάνω λειτουργικών απαιτήσεων υλοποιήσαμε μια αποκεντρωμένη εφαρμογή που αποτελείται από 4 συστατικά στοιχεία, τα οποία αναλύουμε παρακάτω.

5.1 Δημιουργία διεπαφής χρήστη

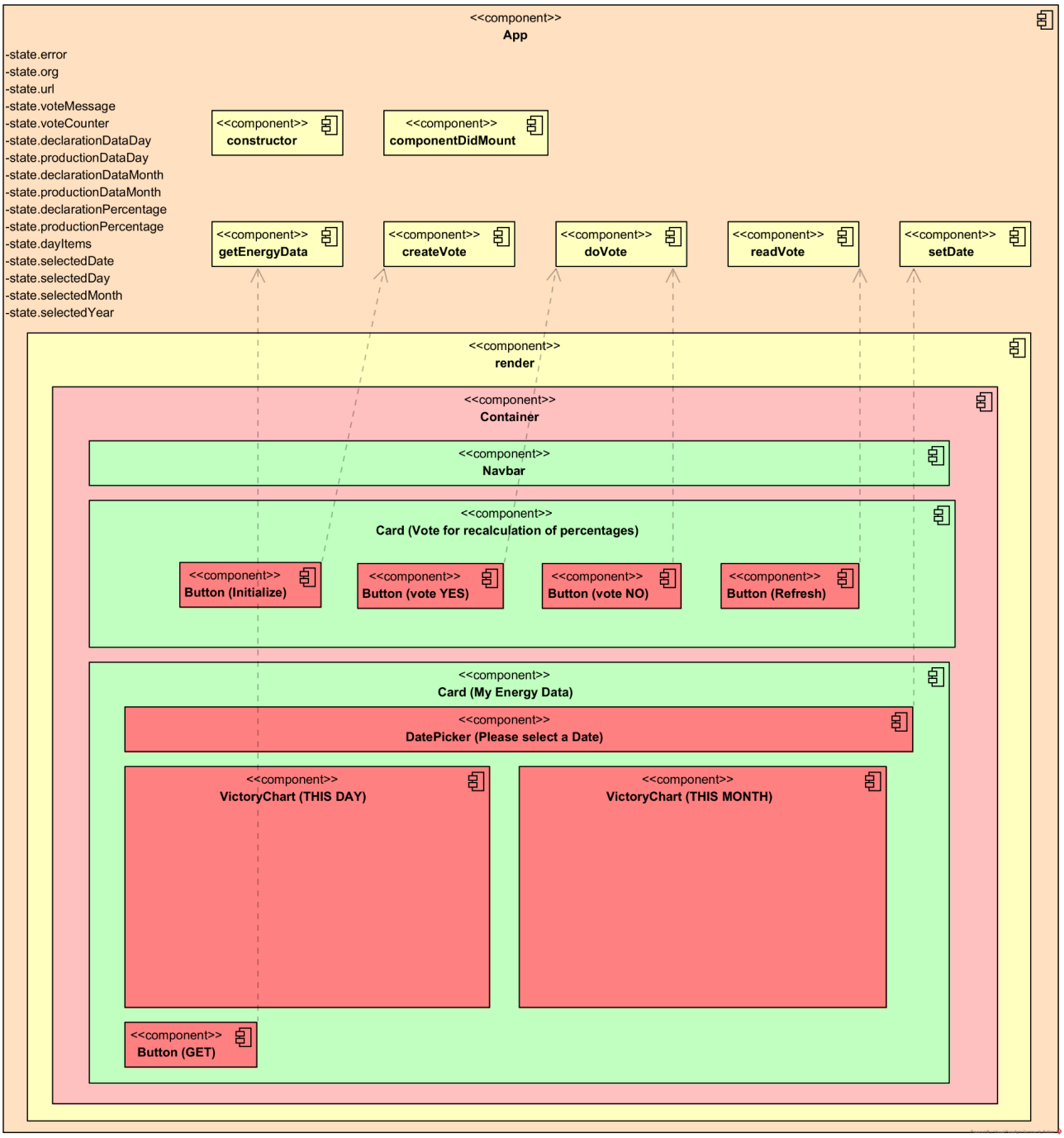
Η διεπαφή χρήστη είναι η εικόνα της αποκεντρωμένης εφαρμογής που βλέπει κάθε παραγωγός. Στόχος της διεπαφής είναι να διευκολύνει κάθε παραγωγό να συμμετάσχει στη διαδικασία της ψηφοφορίας και να οπτικοποιήσει τα δεδομένα του μέσω διαγραμμάτων ανά ημέρα και ανά μήνα. Συμπληρωματικά η διεπαφή προσφέρει κάποιες επιπλέον πληροφορίες όπως το ποσοστό παραγωγής του.

Για τη καλύτερη κατανόηση της διεπαφής χρήστη δημιουργήσαμε ένα διάγραμμα συνιστωσών (Component diagram) με χρήση της γλώσσας μοντελοποίησης Unified Modeling Language (UML). Όπως φαίνεται στο παρακάτω διάγραμμα η διεπαφή χωρίζεται σε 3 βασικά κομμάτια: μία μπάρα πλοήγησης (navigation bar), μία κάρτα ψηφοφορίας και μία κάρτα ενεργειακών δεδομένων.

Η μπάρα πλοήγησης περιέχει πληροφορίες όπως το όνομα του οργανισμού που είναι συνδεδεμένος, το ποσοστό πρόβλεψης παραγωγής και το ποσοστό παραγωγής.

Στη κάρτα ψηφοφορίας διαδραματίζεται όλη η διαδικασία της ψηφοφορίας. Η κάρτα ψηφοφορίας περιέχει ένα μήνυμα που περιγράφει τη κατάσταση της ψηφοφορίας καθώς και 4 κουμπιά. Το κουμπί εκκίνησης ψηφοφορίας “Initialize” αρχικοποιεί την ψηφοφορία επανυπολογισμού των ποσοστών. Το κουμπί θετικής ψήφου “vote YES” και το κουμπί αρνητικής ψήφου “vote NO” καταχωρούν θετική ή αρνητική ψήφο για τον οργανισμό που είναι συνδεδεμένος. Το κουμπί ανανέωσης “Refresh” ενημερώνει το μήνυμα κατάστασης της ψηφοφορίας και πιθανώς το ποσοστό παραγωγής εάν αυτό έχει αλλάξει λόγω επανυπολογισμού.

Στη κάρτα ενεργειακών δεδομένων οπτικοποιούνται τα δεδομένα του συνδεδεμένου οργανισμού με χρήση διαγραμμάτων ανά ημέρα και ανά μήνα. Η κάρτα ενεργειακών δεδομένων περιέχει τη δυνατότητα επιλογής ημερομηνίας, 2 διαγράμματα και ένα κουμπί. Ο συνδεδεμένος παραγωγός επιλέγει την ημερομηνία που επιθυμεί να οπτικοποιήσει και χρησιμοποιεί το κουμπί “GET” ώστε να φέρει τα δεδομένα από το blockchain και να κατασκευαστούν τα διαγράμματα.



Εικόνα 5: Διάγραμμα συνιστωσών UML διεπαφής χρήστη.

Ακολουθεί μία αναλυτική περιγραφή κάθε συνάρτησης του προγράμματος διεπαφής χρήστη:

- Η συνάρτηση “constructor”. Είναι η πρώτη συνάρτηση που καλείται πριν την αρχική απόδοση (render) της σελίδας της εφαρμογής. Σκοπός της είναι η αρχικοποίηση των μεταβλητών κατάστασης (state variables) που είναι ορατά στο αριστερό-πάνω μέρος του διαγράμματος.
- Η συνάρτηση “componentDidMount”. Είναι η πρώτη συνάρτηση που καλείται αμέσως μετά την αρχική απόδοση της σελίδας της εφαρμογής. Χρησιμοποιείται για την ενημέρωση ορισμένων μεταβλητών κατάστασης.
- Η συνάρτηση “createVote”. Καλείται κάθε φορά που χρησιμοποιείται το κουμπί “Initialize”. Η συνάρτηση καλεί τον πόρο (resource) “/vote/init” του ενδιάμεσου

λογισμικού με σκοπό την αρχικοποίηση της ψηφοφορίας επανυπολογισμού των ποσοστών.

- Η συνάρτηση “doVote”. Καλείται κάθε φορά που χρησιμοποιείται το κουμπί “vote YES” ή το κουμπί “vote NO”. Η συνάρτηση καλεί τον πόρο “/vote/yes” ή τον πόρο “/vote/no” με σκοπό τη καταχώρηση της ψήφου του συνδεδεμένου οργανισμού.
- Η συνάρτηση “readVote”. Καλείται κάθε φορά που χρησιμοποιείται το κουμπί “Refresh”. Η συνάρτηση καλεί τον πόρο “/vote” με σκοπό να ενημερώσει το μήνυμα της κατάστασης της ψηφοφορίας. Συμπληρωματικά, εάν η ψηφοφορία έχει ολοκληρωθεί καλεί τον πόρο “/energyData/percentage” για την ενημέρωση του ποσοστού παραγωγής.
- Η συνάρτηση “setDate”. Καλείται κάθε φορά που χρησιμοποιείται ο επιλογέας ημερομηνίας. Η συνάρτηση ενημερώνει τις μεταβλητές κατάστασης που αφορούν την ημερομηνία.
- Η συνάρτηση “getEnergyData”. Καλείται κάθε φορά που χρησιμοποιείται το κουμπί “GET”. Με βάση την ημερομηνία που επιλέχθηκε καλεί τον πόρο “/energyData/<μήνας>/<ημέρα>” με σκοπό να φέρει τα δεδομένα από το blockchain και να δημιουργήσει τους πίνακες ημέρας και μήνα που είναι απαραίτητοι για την κατασκευή των διαγραμμάτων.

Η διεπαφή χρήστη υλοποιήθηκε με χρήση του προγραμματιστικού πλαισίου “React-Bootstrap” [34] και είναι διαθέσιμο στο προσωπικό αποθετήριο Github [35].

5.2 Δημιουργία ενδιάμεσου λογισμικού

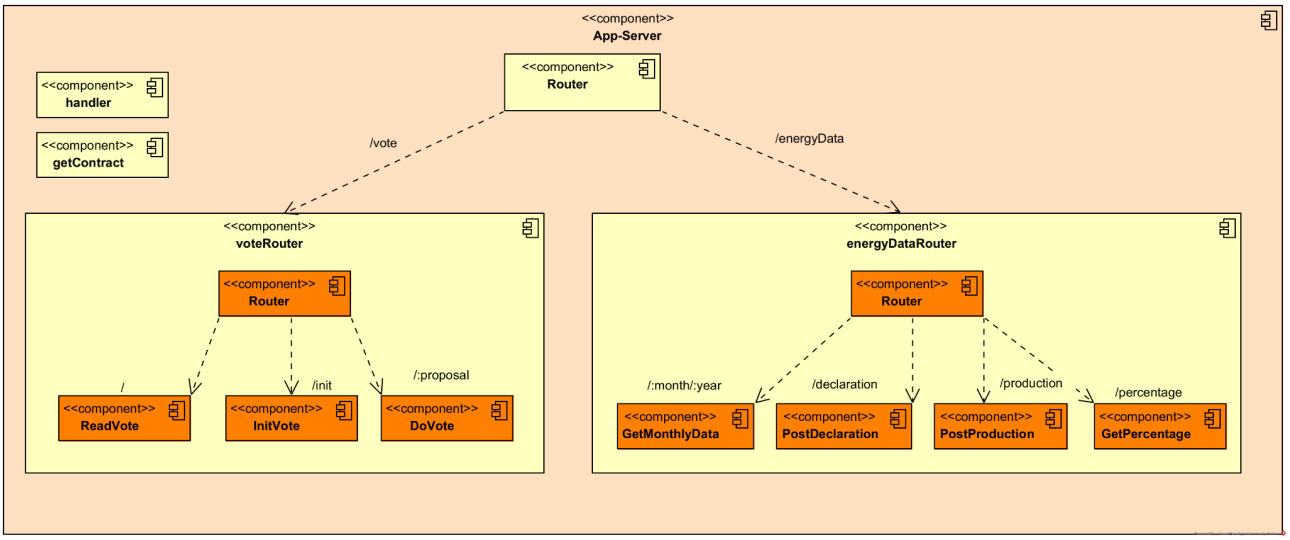
Το ενδιάμεσο λογισμικό είναι ο συνδετικός κρίκος ανάμεσα στη διεπαφή χρήστη και στα έξυπνα συμβόλαια που είναι εγκατεστημένα στον κόμβο δικτύου του συνδεδεμένου οργανισμού. Στόχος του ενδιάμεσου λογισμικού είναι να προσφέρει συντηρησιμότητα και προσαρμοστικότητα στο συνολικό σύστημα απομονώνοντας τη διαδικασία σύνδεσης στο δίκτυο και της καταχώρησης νέων συναλλαγών ή την επικύρωση ήδη υπάρχοντων. Επιπρόσθετα, απομονώνοντας τη διεπαφή χρήστη από τα έξυπνα συμβόλαια δίνουμε τη δυνατότητα χρήσης έξυπνων συσκευών Internet of Things (IoT), που πιθανώς να είναι συνδεδεμένες σε μία ανεμογεννήτρια, να συνδεθούν και να καταχωρούν δεδομένα παραγωγής κάθε 15 λεπτά της ώρας.

Για την καλύτερη κατανόηση του ενδιάμεσου λογισμικού δημιουργήσαμε ένα διάγραμμα συνιστωσών με χρήση της γλώσσας μοντελοποίησης UML. Όπως φαίνεται στο παρακάτω διάγραμμα το ενδιάμεσο λογισμικό χωρίζεται σε 2 βασικά κομμάτια: ένα δρομολογητή ψηφοφορίας και ένα δρομολογητή ενεργειακών δεδομένων. Συμπληρωματικά, το ενδιάμεσο λογισμικό περιέχει τη συνάρτηση “getContract” που είναι υπεύθυνη για τη σύνδεση με το έξυπνο συμβόλαιο και τη συνάρτηση “handler” που είναι υπεύθυνη για την επιστροφή κατάλληλου μηνύματος στη διεπαφή χρήστη.

Ο δρομολογητής ψηφοφορίας εξυπηρετεί όλες τις κλήσεις σε πόρους που ξεκινούν με “/vote”. Κλήσεις στον πόρο “/vote/init” οδηγούνται στη συνάρτηση “InitVote”, κλήσεις στον πόρο “/vote” οδηγούνται στη συνάρτηση “ReadVote” και κλήσεις στον πόρο “/vote/:proposal” οδηγούνται στη συνάρτηση “DoVote”.

Ο δρομολογητής ενεργειακών δεδομένων εξυπηρετεί όλες τις κλήσεις σε πόρους που ξεκινούν με “/energyData”. Κλήσεις στον πόρο “/energyData/declaration” οδηγούνται στη συνάρτηση “PostDeclaration”, κλήσεις στον πόρο “/energyData/production” οδηγούνται στη συνάρτηση “PostProduction”, κλήσεις στον πόρο

"/energyData/:month/:year" οδηγούνται στη συνάρτηση "GetMonthlyData" και κλήσεις στον πόρο "/energyData/percentage" οδηγούνται στη συνάρτηση "GetPercentage".



Εικόνα 6: Διάγραμμα συνιστωσών UML ενδιάμεσου λογισμικού.

Ακολουθεί μία αναλυτική περιγραφή κάθε συνάρτησης του προγράμματος ενδιάμεσου λογισμικού:

- Η συνάρτηση "getContract". Καλείται μία φορά στην ενεργοποίηση του ενδιάμεσου λογισμικού. Σκοπός της είναι η σύνδεση με τον κόμβο του δικτύου που αντιστοιχεί στο συνδεδεμένο οργανισμό ώστε να έχουμε πρόσβαση στο εγκατεστημένο έξυπνο συμβόλαιο.
- Η συνάρτηση "handler". Καλείται κάθε φορά που θέλουμε να κάνουμε κλήση συνάρτησης του έξυπνου συμβολαίου. Σκοπός της είναι η ενεργοποίηση μίας συνάρτησης του έξυπνου συμβολαίου και μετά η επιστροφή του κατάλληλου μηνύματος στην διεπαφή χρήστη.
- Η συνάρτηση "ReadVote". Καλείται κάθε φορά που ζητείται ο πόρος "/vote". Ενεργοποιεί της συνάρτηση "ReadVote" του έξυπνου συμβολαίου και επιστρέφει στη διεπαφή χρήστη τη κατάσταση της ψηφοφορίας επανυπολογισμού των ποσοστών.
- Η συνάρτηση "InitVote". Καλείται κάθε φορά που ζητείται ο πόρος "/vote/init". Ενεργοποιεί τη συνάρτηση "InitVote" του έξυπνου συμβολαίου με σκοπό την αρχικοποίηση της ψηφοφορίας επανυπολογισμού των ποσοστών.
- Η συνάρτηση "DoVote". Καλείται κάθε φορά που ζητείται ο πόρος "/vote/:proposal". Ενεργοποιεί τη συνάρτηση "DoVote" του έξυπνου συμβολαίου με σκοπό την καταχώρηση της ψήφου του συνδεδεμένου οργανισμού.
- Η συνάρτηση "GetMonthlyData". Καλείται κάθε φορά που ζητείται ο πόρος "/energyData/:month/:year". Ενεργοποιεί τη συνάρτηση "GetMonthlyData" του έξυπνου συμβολαίου με σκοπό την επιστροφή στη διεπαφή χρήστη των δεδομένων παραγωγής ενέργειας για τον μήνα που επιλέχθηκε.
- Η συνάρτηση "PostDeclaration". Καλείται κάθε φορά που ζητείται ο πόρος "/energyData/declaration". Ενεργοποιεί τη συνάρτηση "PostDeclaration" του έξυπνου συμβολαίου με σκοπό την καταχώρηση της πρόβλεψης της παραγωγής ενέργειας του συνδεδεμένου οργανισμού.

- Η συνάρτηση “PostProduction”. Καλείται κάθε φορά που ζητείται ο πόρος “/energyData/production”. Ενεργοποιεί τη συνάρτηση “PostProduction” του έξυπνου συμβολαίου με σκοπό την καταχώρηση της παραγωγής ενέργειας του συνδεδεμένου οργανισμού για ένα συγκεκριμένο χρονικό διάστημα 15 λεπτών της ώρας.
- Η συνάρτηση “GetPercentage”. Καλείται κάθε φορά που ζητείται ο πόρος “/energyData/percentage”. Ενεργοποιεί τη συνάρτηση “GetPercentage” του έξυπνου συμβολαίου με σκοπό την επιστροφή στην διεπαφή χρήστη του καταχωρημένου ποσοστού παραγωγής του συνδεδεμένου παραγωγού.

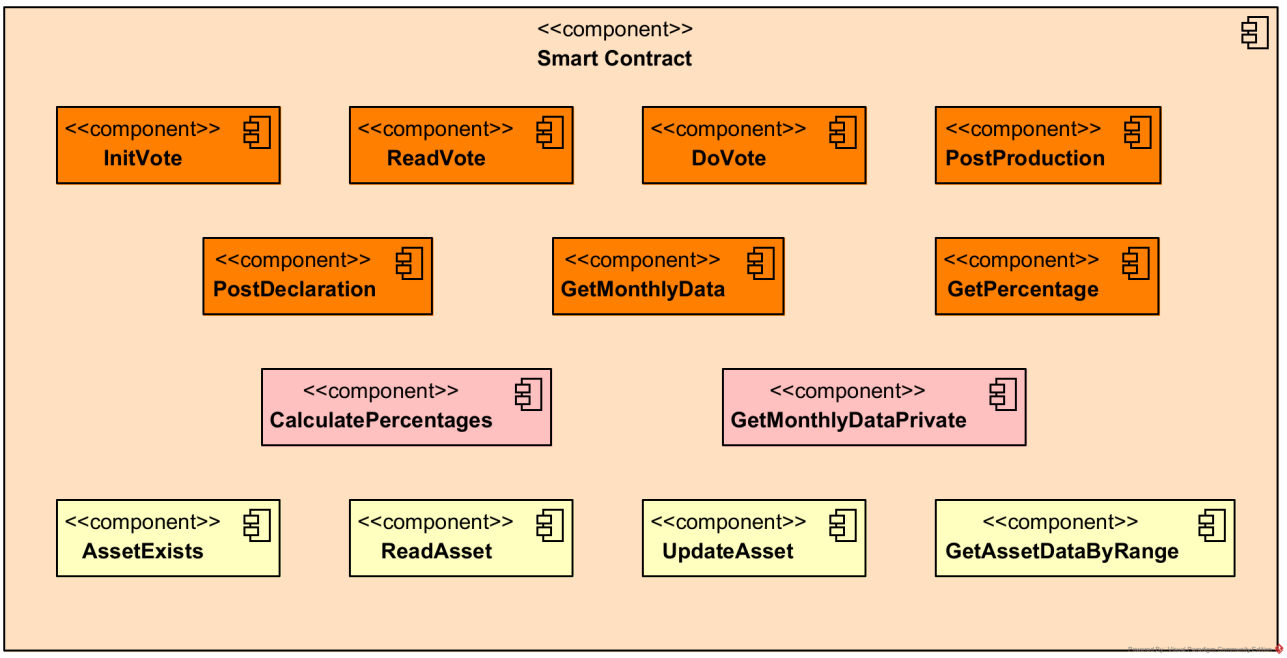
Το ενδιάμεσο λογισμικό υλοποιήθηκε με χρήση του προγραμματιστικού πλαισίου “Express.js” [36] και του “Hyperledger Fabric SDK for Node.js” [37] και είναι διαθέσιμο στο προσωπικό αποθετήριο Github [38].

5.3 Δημιουργία επιχειρηματικής λογικής του DAO

Η επιχειρηματική λογική του DAO υλοποιείται με την συγγραφή των έξυπνων συμβολαίων. Στην περίπτωση χρήσης που αναλύουμε το έξυπνο συμβόλαιο είναι ένα και περιλαμβάνει την λογική τόσο της διαδικασίας την ψηφοφορίας όσο και της διαδικασίας των ενεργειακών δεδομένων. Το συμβόλαιο αυτό γίνεται εγκατάσταση από όλους τους παραγωγούς, γίνεται αποδοχή ή αλλιώς υπογράφεται από την πλειονότητα των παραγωγών και τελικά δηλώνεται από έναν παραγωγό και ενεργοποιείται.

Οι συναρτήσεις του έξυπνου συμβολαίου χωρίζονται σε δύο βασικές κατηγορίες: ιδιωτικές και δημόσιες. Ιδιωτικές ονομάζουμε τις συναρτήσεις που είναι ορατές μέσα στο έξυπνο συμβόλαιο αλλά δεν είναι ορατές στους χρήστες του δηλαδή, στην συγκεκριμένη περίπτωση, τους παραγωγούς. Δημόσιες ονομάζονται οι συναρτήσεις που είναι προσβάσιμες από όσους οργανισμούς έχουν εγκαταστήσει το παρόν έξυπνο συμβόλαιο στον δικό τους κόμβο του δικτύου. Με λίγα λόγια, οι δημόσιες συναρτήσεις ενεργοποιούνται από τους οργανισμούς ενώ οι ιδιωτικές ενεργοποιούνται από άλλες δημόσιες.

Για την καλύτερη κατανόηση του έξυπνου συμβολαίου δημιουργήσαμε ένα διάγραμμα συνιστωσών με χρήση της γλώσσας μοντελοποίησης UML. Όπως φαίνεται στο παρακάτω διάγραμμα το έξυπνο συμβόλαιο περιλαμβάνει 7 δημόσιες και 2 ιδιωτικές συναρτήσεις. Επίσης περιλαμβάνει και 4 βοηθητικές ιδιωτικές συναρτήσεις. Από τις δημόσιες συναρτήσεις, οι πρώτες 3: “InitVote”, “ReadVote” και “DoVote” καλύπτουν την διαδικασία της ψηφοφορίας ενώ οι υπόλοιπες καλύπτουν την διαδικασία των ενεργειακών δεδομένων.



Εικόνα 7: Διάγραμμα συνιστωσών UML έξυπνου συμβολαίου.

Ακολουθεί μία αναλυτική περιγραφή κάθε συνάρτησης του έξυπνου συμβολαίου:

Δημόσιες συναρτήσεις

- Η συνάρτηση “InitVote”. Σκοπός της είναι η αρχικοποίηση της ψηφοφορίας του επανυπολογισμού των ποσοστών παραγωγής. Πιο συγκεκριμένα, αρχικοποιεί ένα αντικείμενο Javascript με 4 πεδία: ένα μήνυμα που περιγράφει την κατάσταση της ψηφοφορίας, ένα μετρητή των θετικών ή αρνητικών ψήφων, ένα πίνακα καταγραφής της ψήφου κάθε οργανισμού και μία μεταβλητή που δηλώνει αν η ψηφοφορία είναι ολοκληρωμένη ή όχι. Μετά την αρχικοποίηση του παραπάνω αντικειμένου καλεί την βοηθητική συνάρτηση “UpdateAsset” και καταχωρεί ή ενημερώνει στο blockchain στη θέση με μοναδικό κλειδί “vote” το αντικείμενο αυτό.
- Η συνάρτηση “ReadVote”. Σκοπός της είναι η επιστροφή της κατάστασης της ψηφοφορίας στον οργανισμό που την κάλεσε. Διαβάζει και επιστρέφει το αντικείμενο που αντιστοιχεί στο μοναδικό κλειδί “vote” με χρήση της βοηθητικής συνάρτησης “ReadAsset”.
- Η συνάρτηση “DoVote”. Σκοπός της είναι η καταγραφή της ψήφου ενός οργανισμού στο blockchain. Αρχικά διαβάζει την υπάρχουσα κατάσταση της ψηφοφορίας που αντιστοιχεί στο μοναδικό κλειδί “vote” με χρήση της βοηθητικής συνάρτησης “ReadVote”. Έπειτα, με βάση την θετική ή αρνητική ψήφο που διαβάζει ως είσοδο, αν το σύνολο των θετικών ψήφων είναι η πλειονότητα τότε καλεί την ιδιωτική συνάρτηση “CalculatePercentages” με σκοπό τον επανυπολογισμό των ποσοστών παραγωγής. Τέλος, ενημερώνει την κατάσταση της ψηφοφορίας με χρήση της βοηθητικής συνάρτησης “UpdateAsset”.
- Η συνάρτηση “PostDeclaration”. Σκοπός της είναι η καταγραφή της πρόβλεψης παραγωγής ενέργειας στα ιδιωτικά δεδομένα του συνδεδεμένου οργανισμού. Αρχικά ελέγχεται εάν η πρόβλεψη αυτή έχει ήδη καταχωρηθεί με χρήση της βοηθητικής συνάρτησης “AssetExists”. Εάν έχει ήδη καταχωρηθεί τότε επιστρέφει με μήνυμα σφάλματος. Διαφορετικά, η πρόβλεψη αυτή καταχωρείται στα ιδιωτικά δεδομένα αυτού του οργανισμού με χρήση της βοηθητικής συνάρτησης “UpdateAsset”.

- Η συνάρτηση “PostProduction”. Σκοπός της είναι η καταγραφή της παραγωγής ενέργειας στα ιδιωτικά δεδομένα του συνδεδεμένου οργανισμού. Αρχικά ελέγχεται εάν η πρόβλεψη της παραγωγής για αυτό το χρονικό διάστημα έχει καταχωρηθεί. Εάν δεν έχει καταχωρηθεί τότε επιστρέφει με μήνυμα σφάλματος, αφού η καταχώρηση της παραγωγής πρέπει να ακολουθεί την καταχώρηση της πρόβλεψης. Διαφορετικά, η παραγωγή καταχωρείται στα ιδιωτικά δεδομένα αυτού του οργανισμού με χρήση της βοηθητικής συνάρτησης “UpdateAsset”.
- Η συνάρτηση “GetMonthlyData”. Σκοπός της είναι η επιστροφή των μηνιαίων δεδομένων παραγωγής στον συνδεδεμένο οργανισμό. Η συνάρτηση διαβάζει τα ιδιωτικά δεδομένα του συνδεδεμένου οργανισμού για τον ζητούμενο μήνα με χρήση της βοηθητικής συνάρτησης “GetAssetDataByRange” και τα επιστρέφει σε αυτόν.
- Η συνάρτηση “GetPercentage”. Σκοπός της είναι η επιστροφή του ποσοστού παραγωγής στον συνδεδεμένο οργανισμό. Η συνάρτηση διαβάζει το ιδιωτικό αυτό δεδομένο του συνδεδεμένου οργανισμού με χρήση της βοηθητικής συνάρτησης “ReadAsset” και το επιστρέφει σε αυτόν.

Ιδιωτικές συναρτήσεις

- Η συνάρτηση “GetMonthlyDataPrivate” είναι ακριβώς παρόμοια με την δημόσια συνάρτηση “GetMonthlyData” που περιγράψαμε παραπάνω. Η διαφορά τους βρίσκεται στο γεγονός ότι η ιδιωτική έκδοση δέχεται ως όρισμα τον οργανισμό που θέλουμε να διαβάσουμε τα δεδομένα του. Η συνάρτηση αυτή καλείται μόνο από την δημόσια συνάρτηση “CalculatePercentages” με σκοπό την συλλογή όλων των δεδομένων όλων των οργανισμών για τον υπολογισμό των ποσοστών τους. Παρατηρούμε ότι η συνάρτηση αυτή τηρεί την λειτουργική απαίτηση “Τα δεδομένα του κάθε οργανισμού είναι ιδιωτικά, και επομένως ορατά μόνο σε αυτόν” αφού είναι ιδιωτική και επομένως ορατή μόνο στο έξυπνο συμβόλαιο και όχι σε κάποιο οργανισμό.
- Η συνάρτηση “CalculatePercentages”. Η συνάρτηση αυτή καλείται μόνο από τη δημόσια συνάρτηση “DoVote”. Σκοπός της είναι ο υπολογισμός των ποσοστών παραγωγής ενέργειας όλων των οργανισμών. Αρχικά υπολογίζει τον προηγούμενο μήνα από την τρέχουσα ημερομηνία. Έπειτα, για τον μήνα που υπολογίστηκε διαβάζει τα δεδομένα όλων των οργανισμών με χρήση της συνάρτησης “GetMonthlyDataPrivate” και υπολογίζει τόσο το άθροισμα της παραγωγής ενέργειας για κάθε οργανισμό όσο και το συνολικό άθροισμα όλων των οργανισμών. Για κάθε οργανισμό υπολογίζει το ποσοστό παραγωγής του με βάση το άθροισμα του προς το συνολικό άθροισμα. Τέλος, ενημερώνει τα ποσοστά αυτά στο blockchain με χρήση της βοηθητικής συνάρτησης “UpdateAsset” και τα επιστρέφει στη συνάρτηση “DoVote” που την κάλεσε.

Βοηθητικές συναρτήσεις

- Η συνάρτηση “AssetExists”. Σκοπός της είναι να ελέγξει αν ένα ψηφιακό περιουσιακό στοιχείο (asset) είναι καταχωρημένο στο blockchain. Δέχεται ως όρισμα το μοναδικό κλειδί και επιστρέφει “true” ή “false”.
- Η συνάρτηση “ReadAsset”. Σκοπός της είναι να διαβάσει ένα ψηφιακό περιουσιακό στοιχείο που είναι καταχωρημένο στο blockchain. Δέχεται ως όρισμα το μοναδικό κλειδί και επιστρέφει το περιουσιακό στοιχείο αν υπάρχει ή μήνυμα σφάλματος διαφορετικά.

- Η συνάρτηση “UpdateAsset”. Σκοπός της είναι η καταχώρηση ενός νέου ψηφιακού περιουσιακού στοιχείου ή την ενημέρωση ενός υπάρχοντος. Δέχεται ως όρισμα το περιουσιακό στοιχείο και το μοναδικό κλειδί στο οποίο θα γίνει η ενημέρωση.
- Η συνάρτηση “GetAssetDataByRange”. Σκοπός της είναι η επιστροφή όλων των περιουσιακών στοιχείων που ανήκουν στο ζητούμενο διάστημα μοναδικών κλειδιών. Δέχεται ως όρισμα το διάστημα μοναδικών κλειδιών και επιστρέφει όλα τα περιουσιακά στοιχεία ανάμεσα σε αυτό το διάστημα.

Αξίζει να σημειωθεί ότι η συγγραφή του έξυπνου συμβολαίου βασίστηκε στο “asset-transfer-basic” που βρίσκεται στα παραδείγματα του Fabric [39]. Οι συναρτήσεις που χρησιμοποιήθηκαν από το παράδειγμα αυτό τοποθετήθηκαν σε έναν φάκελο “utils” και εισάγονται (import) στην αρχή του συμβολαίου.

Το έξυπνο συμβόλαιο υλοποιήθηκε με χρήση του προγραμματιστικού πλαισίου “Node.js” [40] και του “Hyperledger Fabric Contract API for node.js” [41] και είναι διαθέσιμο στο προσωπικό αποθετήριο Github [42].

5.4 Δημιουργία δομής του DAO με χρήση του test-network

Η δομή του DAO υλοποιείται από τη δημιουργία του οργανισμού ταξινόμησης, τη δημιουργία των συμμετεχόντων οργανισμών καθώς και τη δημιουργία ενός καναλιού επικοινωνίας. Ως δομή του οργανισμού ταξινόμησης ορίζουμε έναν ταξινομητή, μία αρχή έκδοσης πιστοποιητικών καθώς και τους διάφορους διαχειριστές του οργανισμού. Αντίστοιχα ως δομή ενός ομότιμου οργανισμού ορίζουμε έναν ομότιμο κόμβο, μία αρχή έκδοσης πιστοποιητικών καθώς και τους διάφορους διαχειριστές του οργανισμού.

Ακολουθεί μία σύντομη περιγραφή του “test-network” που χρησιμοποιείται ως το προεπιλεγμένο δίκτυο τοπικών δοκιμών ανάπτυξης έξυπνων συμβολαίων στα παραδείγματα της πλατφόρμας Fabric [39].

Η διαδικασία της δημιουργίας του δικτύου ξεκινά με τη δημιουργία της δομής του οργανισμού ταξινόμησης καθώς και τη δημιουργία της δομής των ομότιμων οργανισμών. Το πρώτο βήμα για τη δημιουργία του ιδιωτικού δικτύου είναι η έκδοση πιστοποιητικών τόσο για τον οργανισμό ταξινόμησης όσο και για τους συμμετέχοντες οργανισμούς. Παρακάτω γίνεται η ανάλυση δημιουργίας ενός οργανισμού ταξινόμησης ή ομότιμου. Αρχικά δημιουργούμε σε ένα container μία αρχή έκδοσης πιστοποιητικών “fabric-ca” μαζί με μία βάση δεδομένων αυτού. Με τη χρήση ενός διαχειριστή και το μυστικό κωδικό αυτού εκκινούμε την αρχή έκδοσης πιστοποιητικών. Συνδεόμαστε στην αρχή έκδοσης πιστοποιητικών με χρήση του προγράμματος “fabric-ca-client” χρησιμοποιώντας το όνομα και το κωδικό του διαχειριστή ώστε να παράξουμε τα πιστοποιητικά και να κάνουμε εγγραφή στη βάση δεδομένων τις οντότητες:

- Στη περίπτωση οργανισμού ταξινόμησης: τον διαχειριστή του οργανισμού και έναν κόμβο ταξινομητή
- Στη περίπτωση ομότιμου οργανισμού: τον διαχειριστή του οργανισμού, έναν ομότιμο κόμβο και έναν χρήστη οργανισμού

Το δεύτερο βήμα για τη δημιουργία του δικτύου είναι η δημιουργία του αρχικού μπλοκ (genesis block) που περιέχει τη περιγραφή όλων των οργανισμών καθώς και μία κοινοπραξία (consortium) από ομότιμους οργανισμούς που έχουν την ικανότητα να δημιουργήσουν κανάλια επικοινωνίας. Το βήμα αυτό εκτελείται από τον οργανισμό ταξινόμησης.

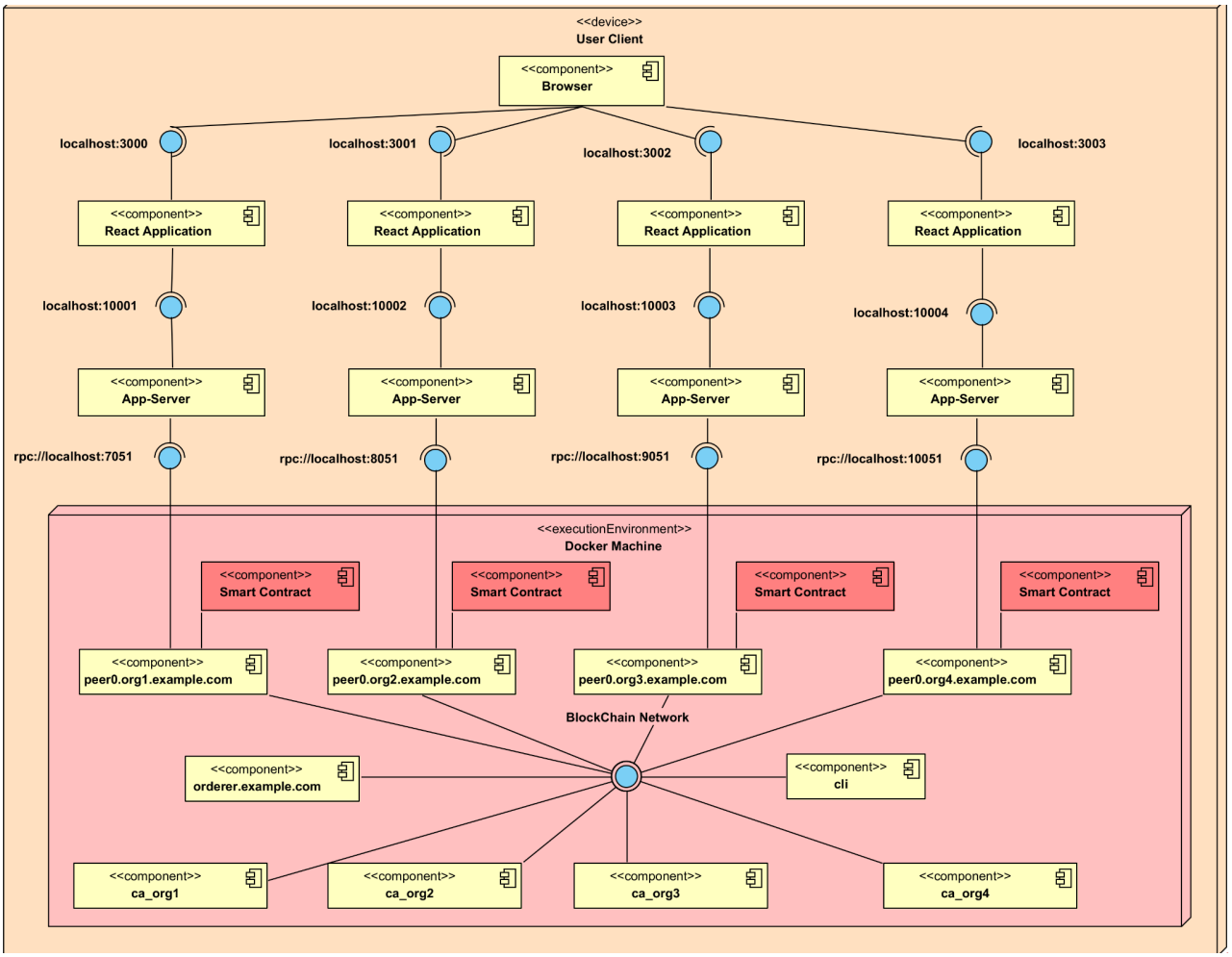
Το τρίτο βήμα αποτελείται από τη δημιουργία τουλάχιστον ενός ταξινομητή που ανήκει στον οργανισμό ταξινόμησης καθώς και τη δημιουργία τουλάχιστον ενός ομότιμου κόμβου για κάθε συμμετέχον οργανισμό.

Το τέταρτο βήμα είναι η δημιουργία του καναλιού επικοινωνίας. Ένας οργανισμός που ανήκει στη κοινοπραξία δημιουργεί το κανάλι και όλοι οι υπόλοιποι οργανισμοί συνδέουν τον κόμβο τους σε αυτό.

Το πέμπτο και τελευταίο βήμα είναι η εγκατάσταση και η ενεργοποίηση του έξυπνου συμβολαίου. Κάθε ομότιμος οργανισμός εγκαθιστά το έξυπνο συμβόλαιο στον κόμβο του, το αποδέχεται και αφού μαζέψει τη πλειονότητα των υπογραφών, αυτό ενεργοποιείται από έναν οργανισμό.

Για την προσαρμογή του test-network στη δική μας περίπτωση χρήσης έγιναν οι ακόλουθες αλλαγές. Αρχικά δημιουργήσαμε τα πιστοποιητικά για 4 οργανισμούς που αντιστοιχούν στους 4 παραγωγούς της περίπτωσης χρήσης μας. Επιπλέον το αρχικό μπλοκ περιέχει τη περιγραφή των τεσσάρων αυτών οργανισμών. Δημιουργήσαμε για κάθε οργανισμό έναν ομότιμο κόμβο, δηλαδή σύνολο 4 ομότιμους κόμβους. Συνδέσαμε τους κόμβους αυτούς στο κανάλι επικοινωνίας. Τέλος, αποδεχθήκαμε το έξυπνο συμβόλαιο και το ενεργοποιήσαμε.

Για την καλύτερη κατανόηση όλων των συστατικών στοιχείων που αποτελούν το δίκτυο καθώς και την τοπολογία του παρατίθεται παρακάτω ένα διάγραμμα συνιστωσών με χρήση της γλώσσας μοντελοποίησης UML.



Εικόνα 8: Διάγραμμα συνιστωσών UML δομής και τοπολογίας δικτύου.

Το παραπάνω σχήμα αναπαριστά την εκτέλεση ενός δικτύου DAO σε “Docker Machine” καθώς και το ενδιάμεσο λογισμικό “App Server” και την διεπαφή χρήστη “React Application” για 4 οργανισμούς πάνω σε μία συσκευή “User Client”. Παρατηρήστε πως για κάθε οργανισμό εκτελούμε μία διεπαφή χρήστη, ένα ενδιάμεσο λογισμικό, έναν ομότιμο κόμβο με όνομα “peer0.org<NUM>.example.com” στο DAO μαζί με μία αρχή έκδοσης πιστοποιητικών με όνομα “ca_org<NUM>” και το έξυπνο συμβόλαιο “Smart Contract”.

Στις προηγούμενες υποενότητες του κεφαλαίου αναλύσαμε τη διεπαφή χρήστη, το ενδιάμεσο λογισμικό και το έξυπνο συμβόλαιο. Με τη χρήση του “test-network” όπως το αναλύσαμε παραπάνω, δημιουργήσαμε όλα τα Docker containers που περικλείονται από το “Docker Machine”. Τα containers των ομότιμων κόμβων, του κόμβου ταξινόμησης, των αρχών έκδοσης πιστοποιητικών, των έξυπνων συμβολαίων καθώς και του “cli” παρέχονται ως ανοιχτό λογισμικό από το ίδρυμα Hyperledger [43][44][45][46][47].

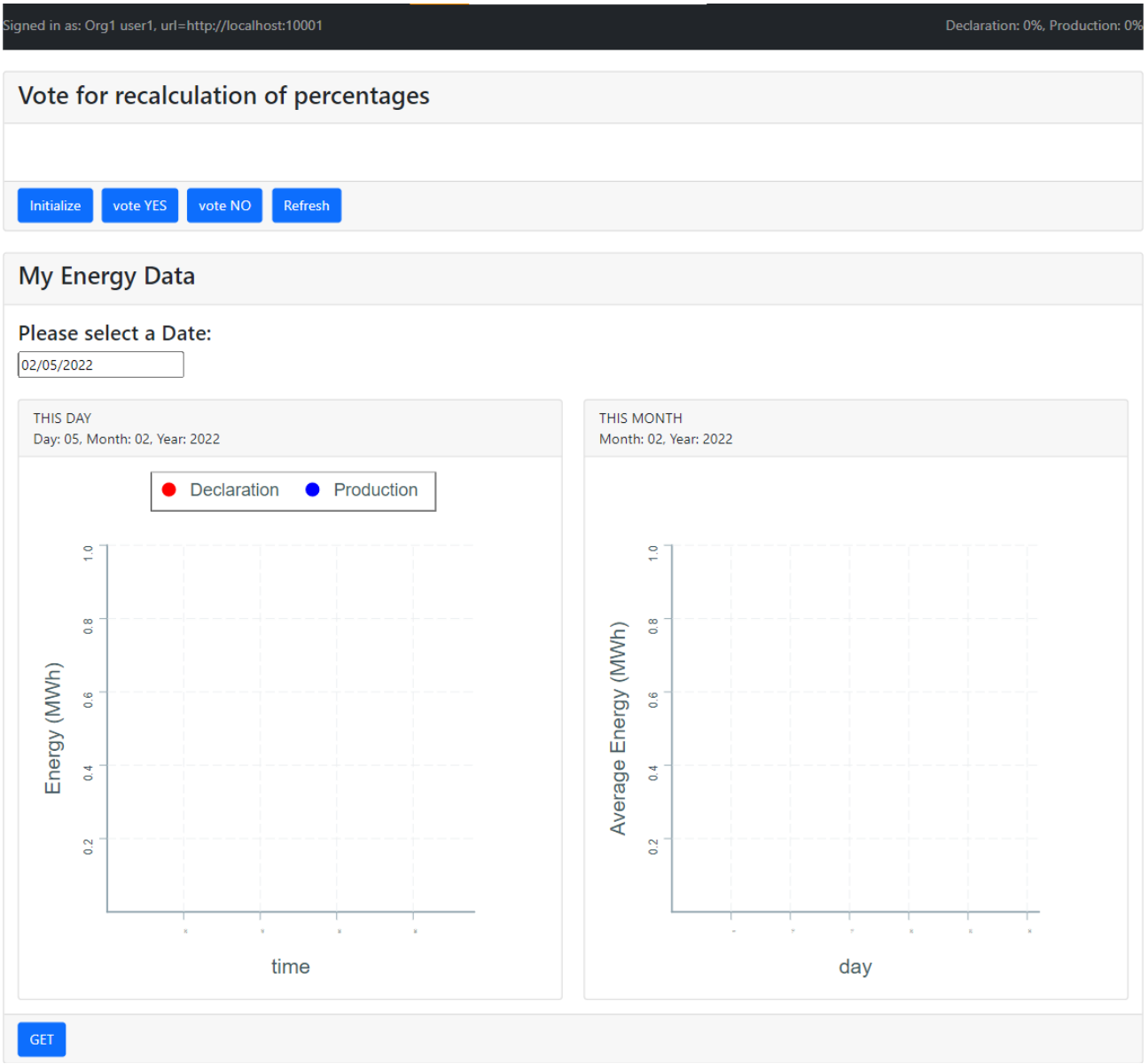
6 Προσομοίωση της αποκεντρωμένης εφαρμογής

Στο κεφάλαιο αυτό θα παρουσιάσουμε μία προσομοίωση της λειτουργικότητας του DAO που περιγράψαμε στο κεφάλαιο 5.

Για την προσομοίωση της αποκεντρωμένης εφαρμογής ακολουθήσαμε τα βήματα που περιγράφονται στο προσωπικό μας δημόσιο αποθετήριο [48]. Τα βήματα αυτά είναι:

1. Πρώτον, ξεκινάμε ένα νέο δίκτυο και ενεργοποιούμε τα έξυπνα συμβόλαια.
2. Δεύτερον, ανοίγουμε 1 ενδιάμεσο λογισμικό για κάθε οργανισμό.
3. Τρίτον, στέλνουμε τα δεδομένα προσομοίωσης στο blockchain για κάθε οργανισμό διαδοχικά.
4. Τέλος, ανοίγουμε τη διεπαφή χρήστη για κάθε οργανισμό.

Η αρχική εικόνα της διεπαφής χρήστη του παραγωγού “Org1” φαίνεται στη παρακάτω εικόνα:



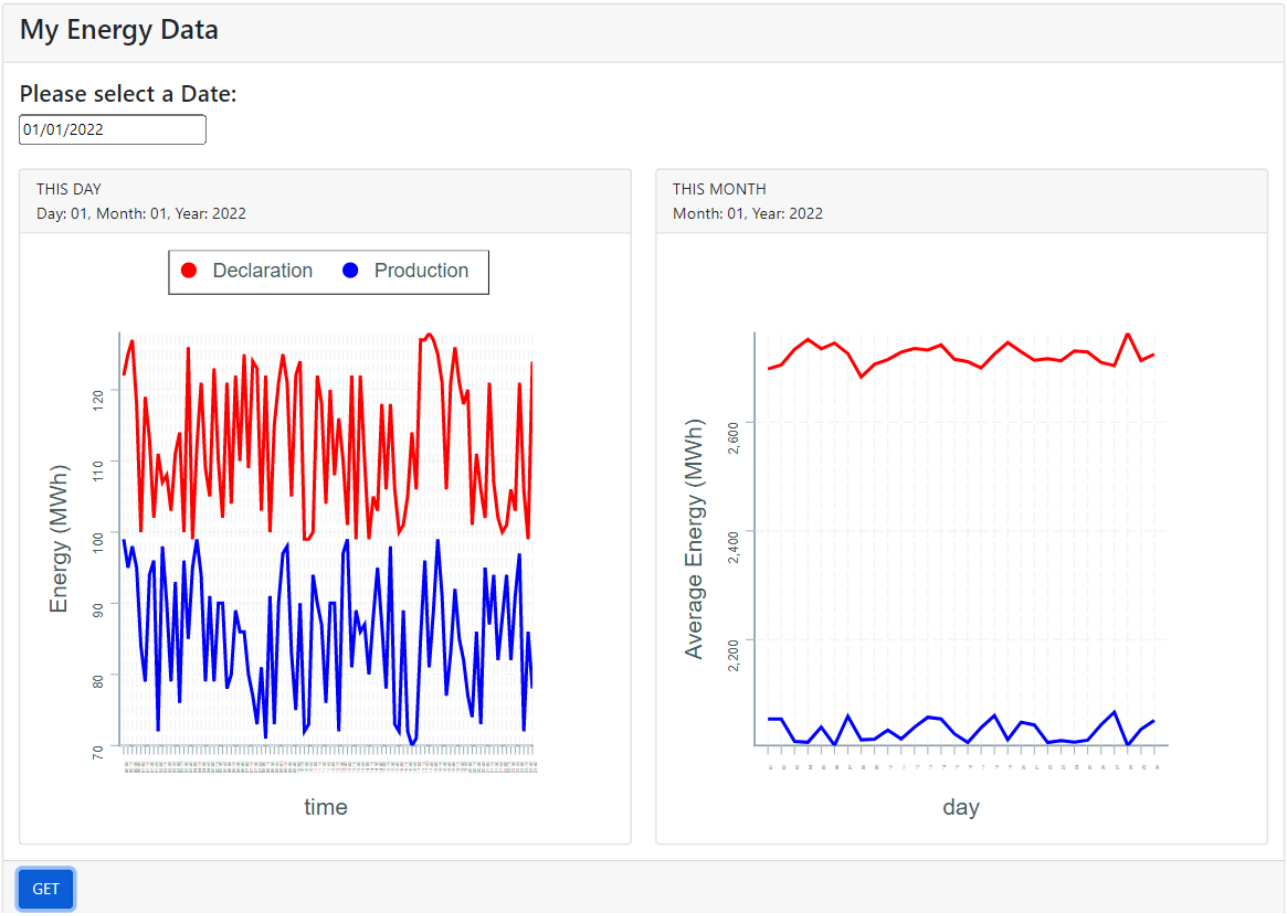
Εικόνα 9: Αρχική σελίδα διεπαφής χρήστη του παραγωγού Org1.

Παρατηρούμε πως η διεπαφή χωρίζεται σε 3 κομμάτια: μία μπάρα πλοήγησης, μία κάρτα ψηφοφορίας και μία κάρτα ενεργειακών δεδομένων. Η μπάρα πλοήγησης δείχνει τον

οργανισμό που είναι συνδεδεμένος “org1”, το “url” του ενδιάμεσου λογισμικού που είναι συνδεδεμένη η διεπαφή καθώς το ποσοστό πρόβλεψης παραγωγής “Declaration” και το ποσοστό παραγωγής “Production”. Η κάρτα ψηφοφορίας για τον επανυπολογισμό των ποσοστών περιέχει το μήνυμα κατάστασης της ψηφοφορίας καθώς και 4 κουμπιά που αφορούν τη ψηφοφορία. Η κάρτα ενεργειακών δεδομένων απεικονίζει τα δεδομένα του οργανισμού για την επιλεγμένη ημερομηνία.

6.1 Απεικόνιση ενεργειακών δεδομένων

Για την προσομοίωση των ενεργειακών δεδομένων δημιουργήσαμε δεδομένα προσομοίωσης για ένα μήνα για κάθε παραγωγό του DAO. Αναλυτικότερα δημιουργήσαμε για το μήνα του Ιανουαρίου του 2022 ένα δεδομένο για κάθε 15 λεπτά της ώρας που αποτελείται από 3 τιμές: μία χρονοσφραγίδα (timestamp), μία τιμή για την πρόβλεψη παραγωγής και μία τιμή για την παραγωγή. Μετά τη παραγωγή των δεδομένων έπεται η καταχώρησή τους στο blockchain. Αρχικά γίνεται καταχώρηση της δήλωσης πρόβλεψης της παραγωγής και έπειτα η καταχώρηση της παραγωγής. Τόσο το πρόγραμμα παραγωγής των δεδομένων αυτών όσο και η διαδικασία καταχώρησής τους στο blockchain βρίσκεται στο προσωπικό μας αποθετήριο [49].

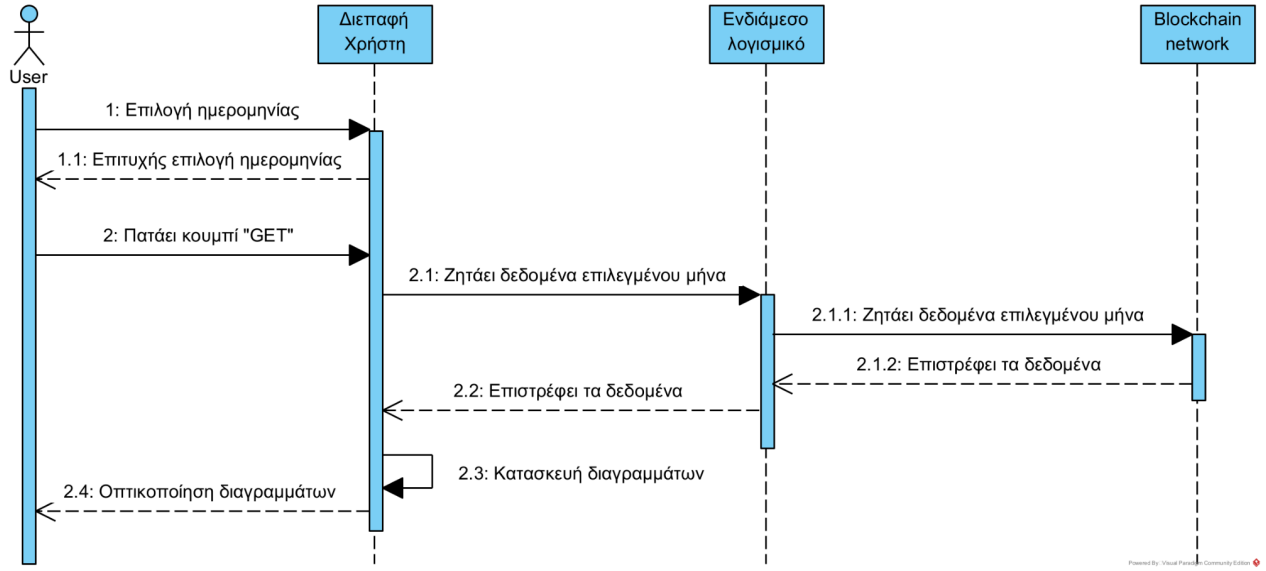


Εικόνα 10: Ο χρήστης επιλέγει την ημερομηνία “01/01/2022” και πατάει το κουμπί “GET”.

Ο χρήστης επιλέγει την ημερομηνία “01/01/2022” και πατάει το κουμπί “GET” για την οπτικοποίηση των δεδομένων του όπως παρουσιάζεται στην εικόνα. Με κόκκινο χρώμα εμφανίζονται τα δεδομένα δήλωσης της πρόβλεψης της παραγωγής και με μπλε τα δεδομένα παραγωγής. Το αριστερό διάγραμμα είναι το διάγραμμα ημέρας με τιμές κάθε 15

λεπτά της ώρας ενώ το δεξί διάγραμμα είναι το διάγραμμα μήνα με τιμές το άθροισμα των τιμών της κάθε μέρας.

Για τη καλύτερη κατανόηση του πληροφοριακού συστήματος στη διαδικασία απεικόνισης των ενεργειακών δεδομένων δημιουργήσαμε ένα διάγραμμα ακολουθίας (Sequence diagram) με χρήση της γλώσσας μοντελοποίησης UML. Παρατηρήστε πως τα αντικείμενα αντιστοιχούν στα συστατικά στοιχεία της αποκεντρωμένης εφαρμογής που αναπτύξαμε στο κεφάλαιο 5.



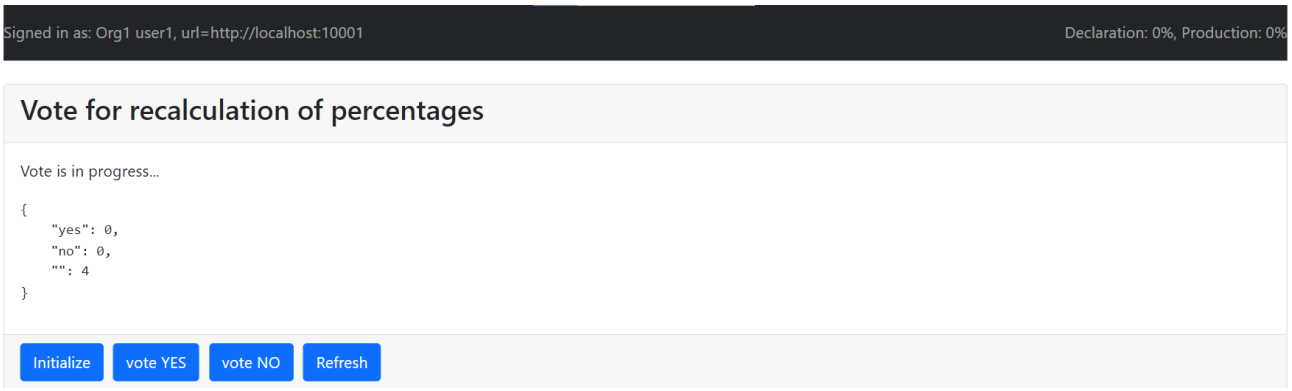
Εικόνα 11:Διάγραμμα ακολουθίας UML απεικόνισης ενεργειακών δεδομένων.

6.2 Διαδικασία ψηφοφορίας

Σε αυτή την υποενότητα θα παρουσιάσουμε μία διαδικασία ψηφοφορίας με θετικό αποτέλεσμα. Ο οργανισμός 1 εκκινεί τη διαδικασία της ψηφοφορίας. Έπειτα κάθε οργανισμός ψηφίζει θετικά σε διαφορετικό χρόνο μέχρι να συγκεντρωθεί πλειονότητα. Το σύστημα αυτόματα επανυπολογίζει τα ποσοστά παραγωγής του κάθε οργανισμού με βάση τα δεδομένα που έχουν καταχωρήσει σε αυτό. Πλέον κάθε παραγωγός βλέπει το αποτέλεσμα της ψηφοφορίας καθώς και τα ανανεωμένα ποσοστά της πρόβλεψης και της παραγωγής του στη μπάρα πλοήγησης.

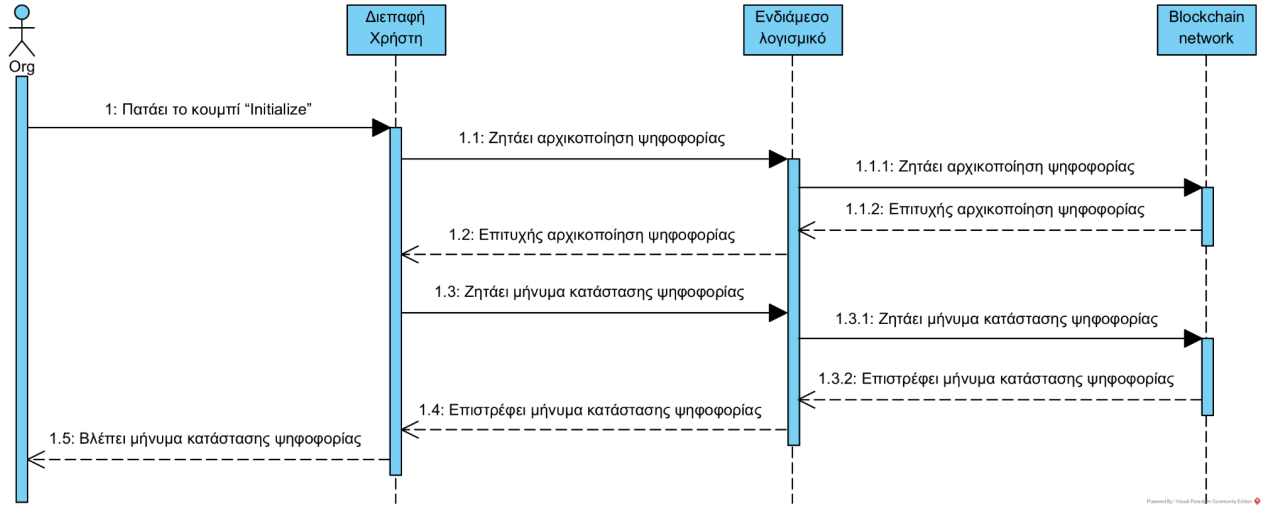
Βήμα 1: Ο παραγωγός Org1 πατάει το κουμπί “Initialize” και η ψηφοφορία αρχικοποιείται. Εμφανίζεται το μήνυμα με την κατάσταση της ψηφοφορίας. Παρατηρείστε πως απεικονίζεται το πλήθος των ψήφων που:

- είναι θετικοί “yes”
- είναι αρνητικοί “no”
- δεν έχουν καταχωρηθεί



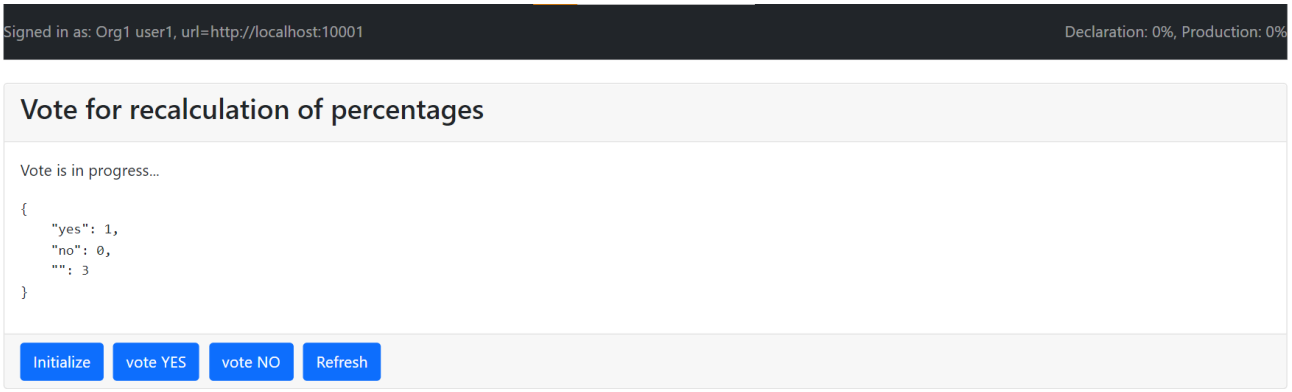
Εικόνα 12: Ο παραγωγός Org1 πατάει το κουμπί "Initialize" και η ψηφοφορία αρχικοποιείται.

Για τη καλύτερη κατανόηση του πληροφοριακού συστήματος στη διαδικασία αρχικοποίησης της ψηφοφορίας δημιουργήσαμε ένα διάγραμμα ακολουθίας με χρήση της γλώσσας μοντελοποίησης UML. Το διάγραμμα αυτό αναπαριστά την λειτουργικότητα του συστήματος κάθε φορά που κάποιος παραγωγός πατάει το κουμπί "Initialize" .



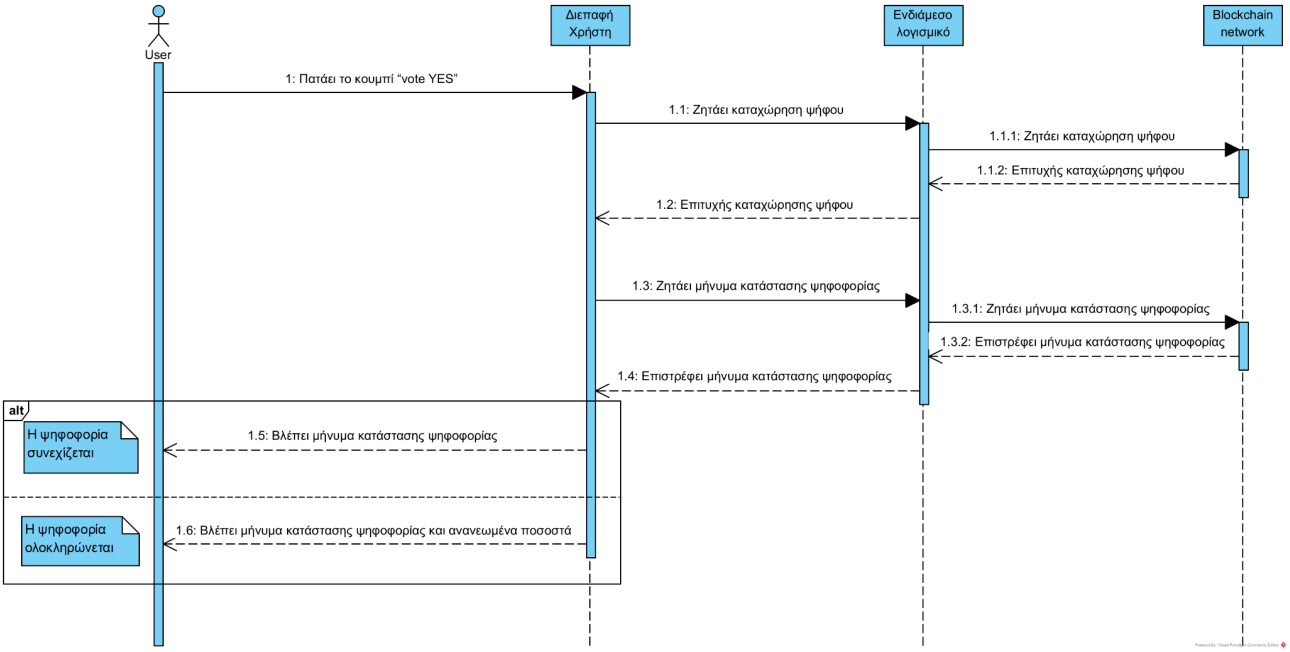
Εικόνα 13: Διάγραμμα ακολουθίας UML αρχικοποίησης ψηφοφορίας

Βήμα 2: Ο παραγωγός Org1 πατάει το κουμπί "vote YES" και η ψηφοφορία συνεχίζεται. Η κατάσταση της ψηφοφορίας ενημερώνεται.



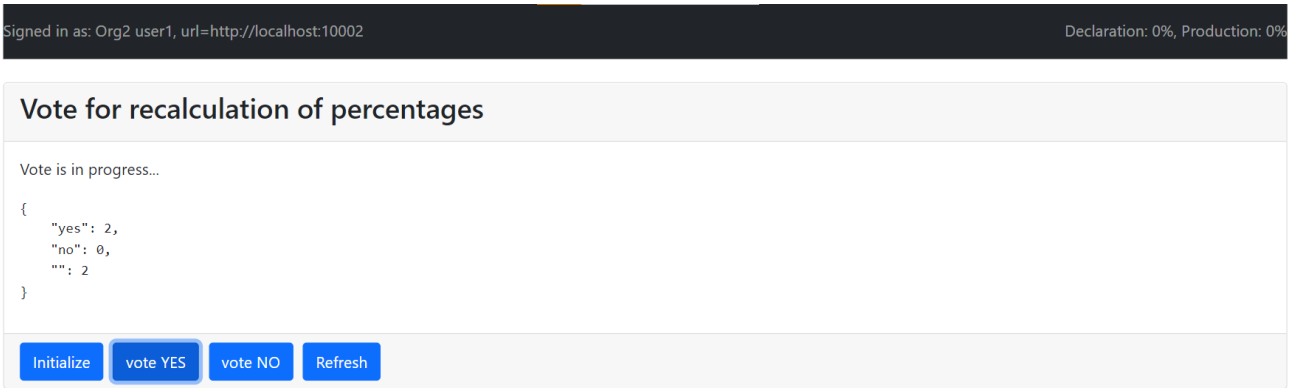
Εικόνα 14: Ο παραγωγός Org1 πατάει το κουμπί "vote YES" και η ψηφοφορία συνεχίζεται.

Για τη καλύτερη κατανόηση του πληροφοριακού συστήματος στη διαδικασία καταχώρησης ψήφου δημιουργήσαμε ένα διάγραμμα ακολουθίας με χρήση της γλώσσας μοντελοποίησης UML. Το διάγραμμα αυτό αναπαριστά τη λειτουργικότητα του συστήματος κάθε φορά που κάποιος παραγωγός πατάει το κουμπί “Vote YES”.



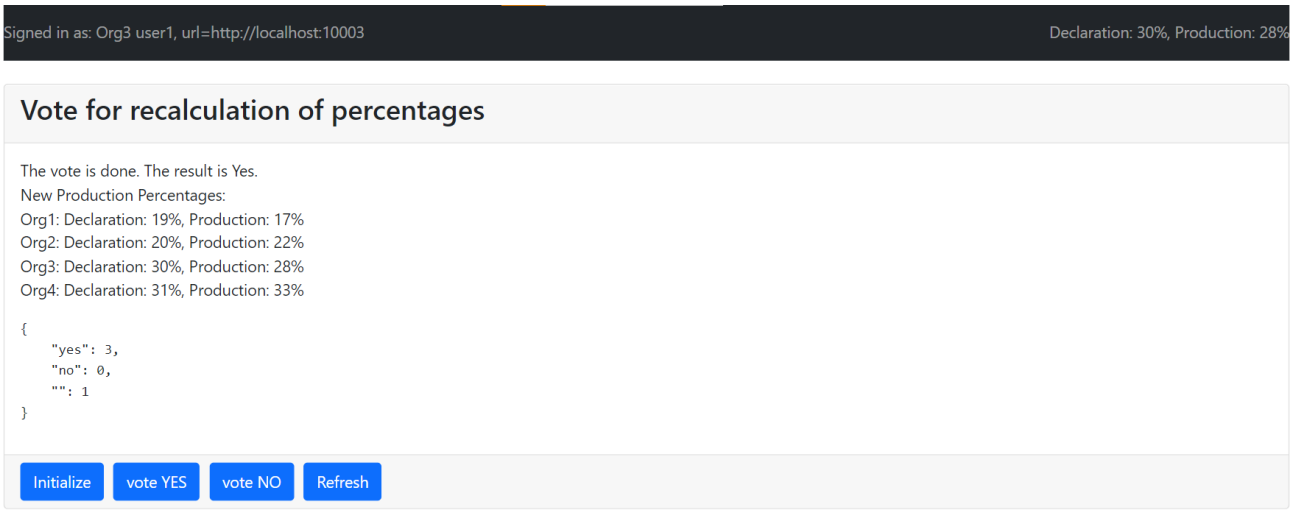
Εικόνα 15: Διάγραμμα ακολουθίας UML καταχώρησης ψήφου.

Βήμα 3: Ο παραγωγός Org2 πατάει το κουμπί “vote YES” και η ψηφοφορία συνεχιζεται. Η κατάσταση της ψηφοφορίας ενημερώνεται.



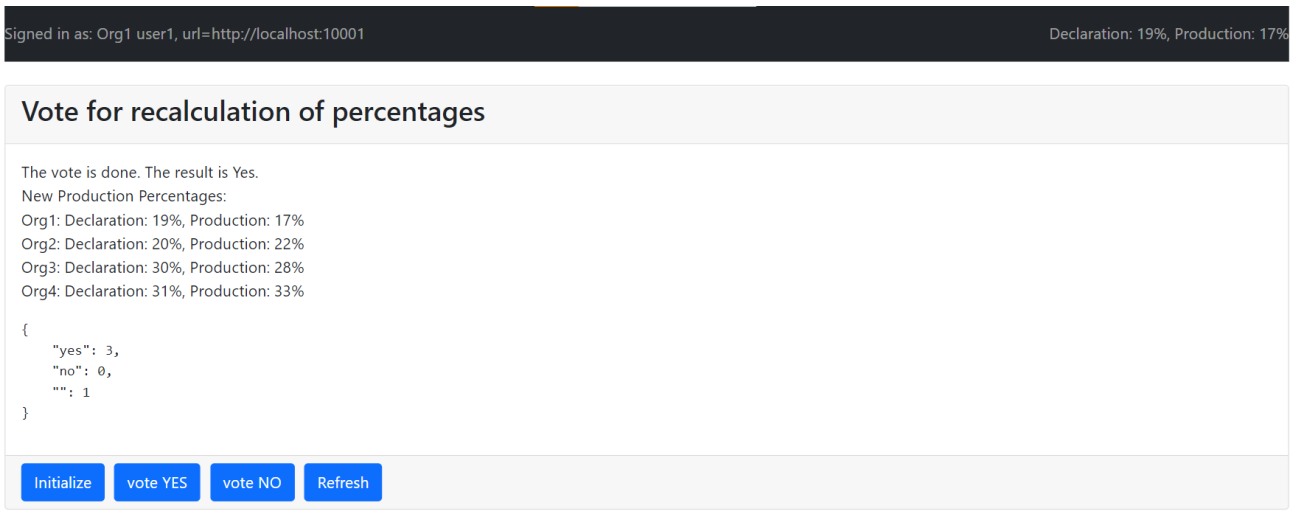
Εικόνα 16: Ο παραγωγός Org2 πατάει το κουμπί “vote YES” και η ψηφοφορία συνεχιζεται.

Βήμα 4: Ο παραγωγός Org3 πατάει το κουμπί “vote YES” και η ψηφοφορία ολοκληρώνεται. Τα ποσοστά πρόβλεψης και παραγωγής ανανεώνονται. Στη μπάρα πλοήγησης ο συνδεδεμένος οργανισμός βλέπει τα ανανεωμένα ποσοστά του. Στο μήνυμα κατάστασης ψηφοφορίας παρουσιάζονται τα ανανεωμένα ποσοστά όλων των οργανισμών.



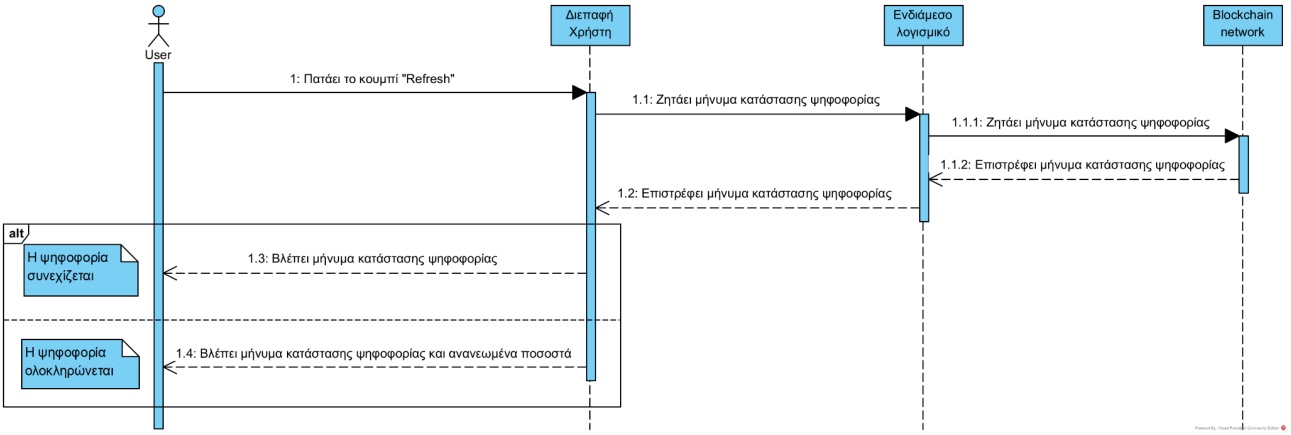
Εικόνα 17: Ο παραγωγός Org3 πατάει το κουμπί "vote YES" και η ψηφοφορία ολοκληρώνεται. Τα ποσοστά πρόβλεψης και παραγωγής ανανεώνονται.

Βήμα 5: Ο παραγωγός Org1 πατάει το κουμπί "Refresh" και βλέπει το αποτέλεσμα της ψηφοφορίας. Τα ποσοστά πρόβλεψης και παραγωγής ανανεώνονται.



Εικόνα 18: Ο παραγωγός Org1 πατάει το κουμπί "Refresh" και βλέπει το αποτέλεσμα της ψηφοφορίας. Τα ποσοστά πρόβλεψης και παραγωγής ανανεώνονται.

Για τη καλύτερη κατανόηση του πληροφοριακού συστήματος στη διαδικασία ανανέωσης της ψηφοφορίας δημιουργήσαμε ένα διάγραμμα ακολουθίας με χρήση της γλώσσας μοντελοποίησης UML. Το διάγραμμα αυτό αναπαριστά την λειτουργικότητα του συστήματος κάθε φορά που κάποιος παραγωγός πατάει το κουμπί "Refresh".



Εικόνα 19: Διάγραμμα ακολουθίας UML ανανέωσης ψηφοφορίας.

7 Σύγκριση Fabric με το GoEthereum

Στο κεφάλαιο 5 παρουσιάσαμε την υλοποίηση ενός DAO με τη χρήση του εργαλείου Fabric στη περίπτωση διαμοιρασμού παραγωγής ηλεκτρικής ενέργειας μεταξύ παρόχων. Η παρούσα διπλωματική εργασία έχει και μια αδελφική συνεργατική πτυχιακή εργασία, αυτή του Σέργιου Έξαρχου του τμήματος Ψηφιακών Συστημάτων του πανεπιστημίου Πειραιά [50]. Οι δύο εργασίες χρησιμοποιούν την ίδια περίπτωση χρήσης για τον ίδιο σκοπό. Η συνεργατική αυτή εργασία χρησιμοποίησε ως εργαλείο υλοποίησης το GoEthereum γνωστό και ως Geth [51]. Τώρα θα χρησιμοποιήσουμε την εργασία αυτή του Σέργιου Έξαρχου για να συγκρίνουμε το εργαλείο Fabric με το εργαλείο GoEthereum με βάση τις παρατηρήσεις που διαμορφώθηκαν κατά τη διάρκεια της υλοποίησης. Η σύγκριση θα χωριστεί σε 3 ενότητες: παρατηρήσεις δικτύου, παρατηρήσεις για τα έξυπνα συμβόλαια και παρατηρήσεις για τη διεπαφή χρήστη.

Πίνακας 4: Σύγκριση Fabric με GoEthereum για την ανάπτυξη ενός DAO σε ιδιωτικό δίκτυο.

Ενότητα	Παρατηρήσεις	Fabric	Geth
Δίκτυο	Ευκολία εκκίνησης και κατανόησης		✔
	Ευκολία δημιουργίας δικτύου		✔
	Ευκολία εγκατάστασης έξυπνων συμβολαίων στο δίκτυο		✔
	Ευελιξία και παραμετροποίηση δικτύου	✔	
	Ρυθμός καταγραφής μπλοκ (Throughput)	✔	
	Εκπαιδευτικό υλικό και τεκμηρίωση δικτύου (Tutorial και documentation)	✔	
Έξυπνα συμβόλαια	Εκπαιδευτικό υλικό και τεκμηρίωση έξυπνων συμβολαίων	✔	✔
	Ευκολία δημιουργίας έξυπνων συμβολαίων	✔	
	Δυνατότητες έξυπνων συμβολαίων	✔	
Διεπαφή χρήστη	Ευκολία σύνδεσης διεπαφής χρήστη με το δίκτυο	✔	✔

7.1 Παρατηρήσεις δικτύου

Όπως είδαμε σε προηγούμενο κεφάλαιο το εργαλείο Fabric είναι πολύπλοκο και δυσνόητο γιατί προσπαθεί να προσφέρει πολλές επιλογές στους διαχειριστές του για οποιαδήποτε πιθανή υλοποίηση ενός ιδιωτικού δικτύου blockchain. Αντίθετα το εργαλείο Geth είναι πιο απλό και εύκολο στη χρήση του επειδή στοχεύει στη δημιουργία αποκεντρωμένων εφαρμογών και την εγκατάστασή τους στο κεντρικό δημόσιο δίκτυο (mainnet) του ethereum. Με άλλα λόγια ένας χρήστης του εργαλείου Geth κατασκευάζει μία αποκεντρωμένη εφαρμογή, τη δοκιμάζει είτε σε τοπικό δίκτυο είτε σε ένα από τα παρεχόμενα δίκτυα δοκιμών (Testnets), με απώτερο σκοπό την εγκατάστασή της στο κεντρικό δημόσιο δίκτυο. Η δημιουργία ιδιωτικών δικτύων είναι διαθέσιμη με χρήση του Geth ως δευτερεύουσα λειτουργία. Η αντίθεση στη πολυπλοκότητα γίνεται ξεκάθαρη αν αναλογιστεί κανείς πως όταν σχεδιάζονταν τα δύο αυτά εργαλεία, ο τελικός χρήστης (end user) του Geth ήταν ανεξάρτητοι χρήστες ενώ του Fabric ήταν οργανισμοί - επιχειρήσεις.

Με βάση τα παραπάνω γίνεται σαφές πως το εργαλείο Geth είναι πιο περιορισμένο στις δυνατότητες που προσφέρει αλλά ταυτόχρονα πιο εύκολο να ξεκινήσει κάποιος να το χρησιμοποιεί. Ακόμα το Geth καθιστά πιο εύκολη τη δημιουργία ενός δικτύου καθώς και την εγκατάσταση των έξυπνων συμβολαίων σε αυτό.

Στο Geth από τη στιγμή που ενεργοποιηθεί στο σύστημα ένα έξυπνο συμβόλαιο δεν μπορεί να τροποποιηθεί, κάτι που παρίσταται ως πρόβλημα σε περίπτωση εσφαλμένης συγγραφής κώδικα ή της ανάγκης ενημέρωσης του. Αντίθετα στο Fabric ένα έξυπνο συμβόλαιο μπορεί να τροποποιηθεί και είναι προφανές πως ένα δίκτυο Fabric παρέχει περισσότερη ευελιξία και παραμετροποίηση των έξυπνων συμβολαίων.

Στην επίσημη ιστοσελίδα του Fabric υπάρχει πάρα πολύ τεκμηρίωση για το πως λειτουργεί και πως να χρησιμοποιηθεί ένα δίκτυο Fabric καθώς και διάφορα παραδείγματα [52]. Σε αντίθεση, στην αντίστοιχη ιστοσελίδα του Geth το υλικό που υπάρχει δεν είναι ικανοποιητικό και χρειάζεται εμπλουτισμό [51]. Αυτό είναι αρκετά σημαντικό πλεονέκτημα του Fabric σε σχέση με το Geth αφού κάθε νέος χρήστης έχει διαθέσιμη την απαραίτητη καθοδήγηση για τη δημιουργία, τη παραμετροποίηση αλλά και την συντήρηση ενός ιδιωτικού δικτύου.

Τέλος, ως απόδοση ορίζουμε το ρυθμό καταγραφής των μπλοκ στο blockchain. Χωρίς να εκτελέσουμε κάποιου είδους βαθμολόγησης επιδόσεων (benchmarking) παρατηρήσαμε πως η απόδοση των δύο εργαλείων είναι παρόμοια για μικρά δίκτυα. Όμως θεωρητικά, όσο το δίκτυο κλιμακώνεται το Fabric είναι αποδοτικότερο σε σχέση με το Geth. Αυτό επιτυγχάνεται μέσω της πρωτοπόρας αρχιτεκτονικής εκτέλεση - ταξινόμηση - επικύρωση του Fabric που εξηγήσαμε στο κεφάλαιο 3.3.

7.2 Παρατηρήσεις έξυπνων συμβολαίων

Στο Fabric τα έξυπνα συμβόλαια συντάσσονται σε γλώσσες προγραμματισμού γενικής χρήσης, όπως η Java, η Go και η Node.js, αντί για DSL, όπως η Solidity που χρησιμοποιείται στο Geth. Αυτό σημαίνει ότι οι περισσότερες επιχειρήσεις διαθέτουν ήδη το σύνολο των δεξιοτήτων που απαιτούνται για την ανάπτυξη έξυπνων συμβολαίων και δεν απαιτείται πρόσθετη εκπαίδευση για να μάθουν μια νέα γλώσσα. Επίσης το Fabric παρέχει το “Contract API” που διαθέτει όλες τις απαραίτητες συναρτήσεις για τη συγγραφή έξυπνων συμβολαίων [41]. Έτσι το Fabric παρέχει περισσότερη ευκολία στη δημιουργία έξυπνων συμβολαίων καθώς και περισσότερες δυνατότητες στα έξυπνα συμβόλαια.

Η επίσημη ιστοσελίδα της solidity παρέχει αρκετό εκπαιδευτικό υλικό και παραδείγματα όσον αφορά την συγγραφή έξυπνων συμβολαίων [53]. Συμπληρωματικά, στη πλατφόρμα του Ethereum, συχνή είναι η χρήση των εργαλείων Truffle [54] ή Remix [55] για την ανάπτυξη έξυπνων συμβολαίων τα οποία διαθέτουν το απαραίτητο υλικό υποστήριξης για προγραμματιστές. Το Fabric παρέχει ικανοποιητική τεκμηρίωση για το Contract API αλλά και πολλά παραδείγματα και εκπαιδευτικό υλικό [39] όσον αφορά τα έξυπνα συμβόλαια. Συνεπώς τόσο το Fabric όσο και το Geth παρέχουν ικανοποιητική υποστήριξη όσον αφορά τη συγγραφή οποιασδήποτε υλοποίησης έξυπνου συμβολαίου.

7.3 Παρατηρήσεις διεπαφής χρήστη

Για την ανάπτυξη μιας αποκεντρωμένης εφαρμογής είναι απαραίτητη μια διεπαφή χρήστη. Συνήθως αυτή η διεπαφή αναπτύσσεται με κάποιο προγραμματιστικό πλαίσιο όπως το React.js που χρησιμοποιήθηκε στην παρούσα εργασία. Στη περίπτωση των αποκεντρωμένων εφαρμογών χρειαζόμαστε ένα εργαλείο για την παραχώρηση πρόσβασης στις συναρτήσεις των έξυπνων συμβολαίων ενός ενεργού δικτύου blockchain. Στο Fabric το εργαλείο αυτό είναι το SDK [37] όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο ενώ στο Geth το εργαλείο αυτό είναι το Web3 [56]. Επομένως και οι 2

πλατφόρμες παρέχουν έναν εύκολο και πολύ παρόμοιο τρόπο για τη σύνδεση της διεπαφής με το δίκτυο.

8 Συμπεράσματα και μελλοντικές κατευθύνσεις

Όπως είδαμε από την ανάλυση του πρωτόκολλου Bitcoin στο δεύτερο κεφάλαιο ένα blockchain λειτουργεί ως ένα καθολικό δηλαδή ως ένα αμετάβλητο μητρώο καταγραφής συναλλαγών. Ένα δίκτυο blockchain ή μία τεχνολογία κατανεμημένου μητρώου (Distributed ledger technology) ή απλώς DLT είναι μία τεχνολογία η οποία επιτρέπει τους συμμετέχοντες να ενεργούν ως ομότιμα μέλη του δικτύου και να διατηρούν και να ενημερώνουν το δικό τους μητρώο με αποτέλεσμα η κατάσταση του μητρώου τους να αντικαθρεφτίζει τη κατάσταση των υπολοίπων μητρώων του δικτύου.

Η αμεταβλητότητα του blockchain σε συνδυασμό με την αποθήκευση του μητρώου από κάθε μέλος του δικτύου θέτουν τη βάση για την δημιουργία ενός κατανεμημένου αυτόνομου οργανισμού. Ένα DAO είναι ένας ψηφιακός οργανισμός που επιτρέπει σε πιθανώς άγνωστα και μη έμπιστα μέλη να συνεργάζονται για την επίτευξη ενός κοινού σκοπού. Είδαμε στα κεφάλαια 4, 5 και 6 μία υλοποίηση ενός τέτοιου οργανισμού με μέλη του 4 παραγωγούς ηλεκτρικής ενέργειας που έχουν ως σκοπό την διαμοίραση της παραγωγής χωρίς τη χρήση ενός έμπιστου τρίτου φορέα. Με την υλοποίηση αυτή δείξαμε πως ένα τέτοιο πληροφοριακό σύστημα μπορεί να προσφέρει εμπιστοσύνη σε αγνώστους αλλά και την απελευθέρωση από ενός πιθανώς διεφθαρμένου εξωτερικού οργανισμού.

Στην παρούσα εργασία χρησιμοποιήθηκε η τεχνολογία κατανεμημένου μητρώου Hyperledger Fabric που αποτελεί αυτή τη χρονική στιγμή ένα από τα πιο διαδεδομένα και σταθερά λογισμικά συστήματα για τη κατασκευή ιδιωτικών αδειοδοτημένων δικτύων blockchain. Από τη σύγκριση του Fabric με το εργαλείο GoEthereum στη κατασκευή ενός DAO σε ιδιωτικό δίκτυο blockchain συμπεράναμε πως το Fabric είναι καταλληλότερο λόγω της ευελιξίας, συντηρησιμότητας και κλιμακωσιμότητας που προσφέρει. Αξίζει να σημειωθεί πως το αδύναμο σημείο του Fabric είναι η απότομη καμπύλη εκμάθησης (steep learning curve) που απαιτεί από τους διαχειριστές του και αποτελεί τον κύριο λόγο πιθανής επιλογής του GoEthereum την παρούσα χρονική στιγμή.

Η μελλοντική χρήση της παρούσας εργασίας αποτελεί κυρίως την ανάπτυξη ενός DAO που παρέχει τη δυνατότητα ψηφιακού διοικητικού συμβουλίου με τη παροχή ισότιμου μηχανισμού ψηφοφορίας. Συμπληρωματικά το εύρος της χρήσης της αυξάνεται αφού μπορεί να χρησιμοποιηθεί σαν προσχέδιο για την ανάπτυξη οποιονδήποτε πιθανού DAO.

Η παρούσα εργασία μπορεί να αποτελέσει εγχειρίδιο μελέτης και έρευνας, καθώς η ελληνική βιβλιογραφία είναι ακόμα σε αρχικό στάδιο, για την κατανόηση των εννοιών blockchain, Hyperledger Fabric και DAO καθώς και την απάντηση των ερωτημάτων “γιατί blockchain, τι είναι και πως λειτουργεί;”, “γιατί να επιλέξουμε το εργαλείο Fabric;” και “τι είναι, πως υλοποιείται και ποια η χρησιμότητα ενός DAO;”.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός όρος**
Decentralized Autonomous Organization	Αποκεντρωμένος Αυτόνομος Οργανισμός
Smart contract	Έξυπνο συμβόλαιο
Peers	Ομότιμοι
Private permissioned	Ιδιωτικά αδειοδοτημένα
Trusted 3rd party	Έμπιστος τρίτος φορέας
User interface	Διεπαφή χρήστη
Screenshots	Στιγμιότυπα
Distributed network of independent users	Δίκτυο από διασυνδεδεμένους ανεξάρτητους χρήστες
Immutability	Αμεταβλητότητα
Block	Μπλοκ
Ledger	Μητρώο
Chain	Αλυσίδα
Hashing	Κατακερματισμός
Double spending	Διπλή δαπάνη
Replay attack	Επίθεση επανάληψης
Distributed peer-to-peer network	Ομότιμος-σε-ομότιμο κατανεμημένο δίκτυο
Peer-to-peer	Ομότιμος-σε-ομότιμο
Consensus	Συναίνεση
Digital signature	Ψηφιακή υπογραφή
Private key	Ιδιωτικό κλειδί
Public key	Δημόσιο κλειδί
Wallet address	Διεύθυνση πορτοφολιού
Mempool	Ουρά αναμονής συναλλαγών
Miners	Εξορύκτες
Locking script	Σενάριο κλειδώματος
Atomic broadcast	Ατομική εκπομπή

Order-execute	Ταξινομηση-εκτέλεση
Transparent	Διαφανής
Permissionless	Χωρίς άδεια
Permissioned	Με άδεια
Byzantine Fault Tolerance	Ανοχή βυζαντινών σφαλμάτων
51% attack	Επίθεση κατά 51%
Permission	Άδεια
Scalability	Κλιμακωσιμότητα
Vulnerability	Ασφάλεια
Decentralized Applications	Αποκεντρωμένες εφαρμογές
Digital asset management	Διαχείριση ψηφιακών αγαθών
Vote based algorithms	Αλγόριθμους με βάση την ψήφο
Probabilistic voting	Πιθανολογική μέθοδος ψηφοφορίας
Executive Director	Εκτελεστικός διευθυντής
Distributed ledger technologies	Τεχνολογίες κατανεμημένων μητρώων
Libraries	Βιβλιοθήκες
Domain specific	Ειδικού τομέα
Tools	Εργαλεία
Trusted execution environments	Αξιόπιστα περιβάλλοντα εκτέλεσης
Domain Specific Languages	Γλώσσες για συγκεκριμένους τομείς
Mining	Εξόρυξη
Documentation	Τεκμηρίωση
Replicated ledger	Πανομοιότυπο μητρώο
Endorsement policy	Πολιτική υποστήριξης
Execute-order-validate	εκτέλεση - ταξινόμηση - επικύρωση
Certificate authority	Αρχή έκδοσης πιστοποιητικών
Key-value store	Αποθήκευση σε μορφή κλειδί-τιμή
Sequential execution	Διαδοχική εκτέλεση
Logical expression	Λογική έκφραση

Client	Πελάτης
Endorsement	Υποστήριξη
State dependencies	Εξαρτήσεις της κατάστασης
State changes	Αλλαγές κατάστασης
Chaincode	Έξυπνο συμβόλαιο στο Fabric
Orderers	Ταξινομητές
Target model	Μοντέλο στόχος
Component diagram	Διάγραμμα συνιστωσών
Sequence diagram	Διάγραμμα ακολουθίας
Navigation bar	Μπάρα πλοήγησης
Render	Απόδοση της σελίδας της εφαρμογής
State variables	Μεταβλητές κατάστασης
Resource	Πόρος
Router	Δρομολογητής
Asset	Ψηφιακό περιουσιακό στοιχείο
Import	Εισαγωγή
Genesis block	Αρχικό μπλοκ
Consortium	Κοινοπραξία
Timestamp	Χρονοσφραγίδα
Throughput	Ρυθμός καταγραφής μπλοκ
Tutorial	Εκπαιδευτικό υλικό
Mainnet	Κεντρικό δημόσιο δίκτυο του Ethereum
Testnets	Δίκτυα δοκιμών του Ethereum
End user	Τελικός χρήστης
Benchmarking	Βαθμολόγηση επιδόσεων
Steep learning curve	Απότομη καμπύλη εκμάθησης

ΣΥΝΤΜΗΣΕΙΣ - ΑΡΚΤΙΚΟΛΕΞΑ - ΑΚΡΩΝΥΜΙΑ

DAO	Decentralized Autonomous Organization
ΦΟΣΕ	Φορείς Σωρευτικής Εκπροσώπησης
B2B	Business to Business
ECDSA	Elliptic Curve Digital Signature Algorithm
SHA256	Secure Hash Algorithm
RIPEMD-160	RIPE Message Digest
UTXO	Unspent Transaction Output
BFT	Byzantine Fault Tolerance
PoW	Proof of Work
PoA	Proof of Authority
PoS	Proof of Stake
SCP	Stellar Consensus Protocol
FBA	Federated Byzantine Agreement
PoET	Proof of Elapsed Time
DApps	Decentralized Applications
YAC	Yet Another Consensus
IBFT	Istanbul Byzantine Fault Tolerance
DSL	Domain Specific Languages
CFT	Crash Fault Tolerance
RAFT	Reliable, Replicated, Redundant, And Fault-Tolerant
gRPC	Google Remote Procedure Call
ΑΠΕ	Ανανεώσιμων Πηγών Ενέργειας
ΑΣΠΗΕ	Αιολικός Σταθμός Παραγωγής Ηλεκτρικής Ενέργειας
UML	Unified Modeling Language
IoT	Internet of Things
DLT	Distributed Ledger Technology
ΡΑΕ	Ρυθμιστική Αρχή Ενέργειας
ΑΔΜΗΕ	Ανεξάρτητος Διαχειριστής Μεταφοράς Ηλεκτρικής Ενέργειας

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

[1] Laurence, T. (2019). Blockchain for dummies. John Wiley & Sons.
[2] Haber, S., & Stornetta, W. S. (1990, August). How to time-stamp a digital document. In Conference on the Theory and Application of Cryptography (pp. 437-455). Springer, Berlin, Heidelberg.
[3] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 21260.
[4] Morabito, V. (2017). Business innovation through blockchain. Cham: Springer International Publishing.
[5] Johnson, D., Menezes, A., & Vanstone, S. (2001). The elliptic curve digital signature algorithm (ECDSA). International journal of information security, 1(1), 36-63.
[6] Naik, R. P., & Courtois, N. T. (2013). Optimising the sha256 hashing algorithm for faster and more efficient bitcoin mining. MSc Information Security Department of Computer Science UCL, 1-65.
[7] Dobbertin, H., Bosselaers, A., & Preneel, B. (1996, February). RIPEMD-160: A strengthened version of RIPEMD. In International Workshop on Fast Software Encryption (pp. 71-82). Springer, Berlin, Heidelberg.
[8] Daulay, R. S. A., Nasution, S. M., & Paryasto, M. W. (2017, November). Realization and addressing analysis in blockchain bitcoin. In IOP Conference Series: Materials Science and Engineering (Vol. 260, No. 1, p. 012002). IOP Publishing.
[9] Python Software Foundation (2022) random.py [Source code].
https://github.com/python/cpython/blob/3.10/Lib/random.py.
[10] Oracle (2022) SecureRandom.java [online].
https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html.
[11] Ethereum, “Decentralized autonomous organizations”, www.ethereum.org [Online]. Available: https://ethereum.org/en/dao . [Accessed Feb. 10, 2022]
[12] D. Geroni, “Top 5 Blockchain Security Issues In 2021”, 101blockchains.com, June 13, 2021. [Online]. Available: https://101blockchains.com/blockchain-security-issues. [Accessed Feb. 08, 2022].
[13] Ethereum Foundation (2022) Ethereum [Online]. https://ethereum.org.
[14] Stellar Development Foundation (2022) Stellar [Online]. https://www.stellar.org.
[15] R3 (2022) R3 Corda [Online] https://www.r3.com/corda-platform.
[16] CONSENSYS (2022) Quorum [Online] https://consensys.net/quorum.
[17] Ripple (2022) Ripple [Online] https://ripple.com.
[18] S. Gupta “Hyping the hyperledger with blockchain boffin Brian Behlendorf”, www.hfsresearch.com [Online]. Available: https://www.hfsresearch.com/outsourcing-heros/brian-behlendorf-interview_082417/ . [Accessed Feb. 09, 2022]
[19] The Linux Foundation (2022) Hyperledger Foundation [Online]. https://www.hyperledger.org .
[20] K. Kotoski, "NBC signs blockchain agreement", Phnom Penh Post, April 24, 2017.[Online], Available:https://www.phnompenhpost.com/business/nbc-signs-blockchain-agreement .[Accessed Feb 9, 2022].
[21] Kapritsos, M., Wang, Y., Quema, V., Clement, A., Alvisi, L., & Dahlin, M. (2012). All about Eve:{Execute-Verify} Replication for {Multi-Core} Servers. In 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12) (pp. 237-250).
[22] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Yellick, J. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the thirteenth EuroSys conference (pp. 1-15).
[23] gRPC Authors (2022) gRPC [Online]. https://grpc.io/.
[24] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, et al. On scaling decentralized blockchains. In International Conference on Financial Cryptography and Data Security (FC), pages 106–125. Springer, 2016.
[25] M. Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In International Workshop on Open Problems in Network Security (iNetSec), pages 112–125, 2015.
[26] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In ACM Symposium on Principles of Distributed Computing (PODC), pages 1–12. ACM, 1987.
[27] Suryandaru Triandana (2012) LevelDB in Go [Source Code]. https://github.com/syndtr/goleveldb/
[28] The Apache Software Foundation (2021) Apache CouchDB. http://couchdb.apache.org

[29] Νόμος 4414/2016, "Νέο καθεστώς στήριξης των σταθμών παραγωγής ηλεκτρικής ενέργειας από Ανανεώσιμες Πηγές Ενέργειας και Συμπαραγωγή Ηλεκτρισμού και Θερμότητας Υψηλής Απόδοσης - Διατάξεις για το νομικό και λειτουργικό διαχωρισμό των κλάδων προμήθειας και διανομής στην αγορά του φυσικού αερίου και άλλες διατάξεις.", ΦΕΚ 149/Α/9-8-2016, [Online], Available: https://www.e-nomothesia.gr/energeia/nomos-4414-2016.html , [Accessed Feb. 11, 2022]
[30] Χ. Φλουδόπουλος, "Τι είναι οι ΦΟΣΕ στους οποίους επενδύουν Τερνα και Μυτιληναίος", www.energypress.gr, March 18, 2019 [Online], Available: https://energypress.gr/news/h-floydopoylos-ti-einai-oi-fose-stoys-opoioys-ependyoyn-terna-kai-mytilinaios , [Accessed Feb. 11, 2022]
[31] Protergia, "Φορέας Σωρευτικής Εκπροσώπησης (ΦΟΣΕ)", www.protergia.gr , [Online], Available: https://www.protergia.gr/el/foreas-swreytikhs-ekproswphshs , [Accessed Feb. 11, 2022]
[32] Forena, "Γιατί είναι κρίσιμη για τους παραγωγούς ΑΠΕ η επιλογή ΦΟΣΕ που θα τους εκπροσωπήσει στο target model", www.forenaenergy.gr, [Online], Available: https://www.forenaenergy.gr/el/2319-2/ , [Accessed Feb. 11, 2022]
[33] Everoze "AGGREGATORS: WHO ARE YOU?", www.everoze.com, February 2018, [Online], Available: https://everoze.com/aggregators-who-are-you/ , [Accessed Feb. 11, 2022]
[34] J. Collings, M. Honnibal, P. Vanderwerff (2022) react-bootstrap [Online]. https://react-bootstrap.github.io/
[35] Δ. Σταραντζής (2022) react-app [Source Code]. https://github.com/jimwheaty/blockchainDAO/tree/main/react-app
[36] OpenJS Foundation (2017) Express [Online]. https://expressjs.com
[37] Hyperledger Foundation (2022) Hyperledger Fabric SDK for Node.js [Online]. https://hyperledger.github.io/fabric-sdk-node/release-2.2/
[38] Δ. Σταραντζής (2022) app-server-javascript [Source Code]. https://github.com/jimwheaty/blockchainDAO/tree/main/app-server-javascript
[39] Hyperledger Foundation (2022) fabric-samples [Source Code]. https://github.com/hyperledger/fabric-samples/tree/release-2.2
[40] OpenJS Foundation (2022) Node.js [Online]. https://nodejs.org/en/
[41] Hyperledger Foundation (2022) fabric-contract-api [Online]. https://hyperledger.github.io/fabric-chaincode-node/release-2.2/api
[42] Δ. Σταραντζής (2022) chaincode-energyDAO [Source Code]. https://github.com/jimwheaty/blockchainDAO/tree/main/chaincode-energyDAO
[43] Hyperledger Foundation (2022) fabric-peer [Online]. https://hub.docker.com/r/hyperledger/fabric-peer
[44] Hyperledger Foundation (2022) fabric-orderer [Online]. https://hub.docker.com/r/hyperledger/fabric-orderer
[45] Hyperledger Foundation (2022) fabric-ca [Online]. https://hub.docker.com/r/hyperledger/fabric-ca
[46] Hyperledger Foundation (2022) fabric-nodeenv [Online]. https://hub.docker.com/r/hyperledger/fabric-nodeenv
[47] Hyperledger Foundation (2022) fabric-tools [Online]. https://hub.docker.com/r/hyperledger/fabric-tools
[48] Δ. Σταραντζής (2022) blockchainDAO [Source Code]. https://github.com/jimwheaty/blockchainDAO
[49] Δ. Σταραντζής (2022) simulation [Source Code]. https://github.com/jimwheaty/blockchainDAO/tree/main/simulation
[50] Σ. Έξαρχος, "Αρχιτεκτονικές λογισμικού για υλοποίηση κατανεμημένων αυτόνομων οργανισμών (DAO) με τεχνολογίες blockchain.", Πτυχιακή εργασία, Σχολή Τεχνολογιών Πληροφορικής και Τηλεπικοινωνιών, Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Πειραιά, 2022.
[51] go-ethereum authors (2022) GoEthereum [Online]. https://geth.ethereum.org
[52] Hyperledger Foundation, "A Blockchain Platform for the Enterprise," hyperledger-fabric.readthedocs.io, 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2 . [Accessed Feb. 10, 2022].
[53] Ethereum (2022) Solidity [Online]. https://docs.soliditylang.org/en/v0.8.11/
[54] ConsenSys Software Inc (2022) truffle suite [Online]. https://trufflesuite.com/docs/index.html
[55] Remix (2021) Remix [Online] https://remix.ethereum.org/
[56] Web3 (2022) web3.js [Source Code] https://github.com/ChainSafe/web3.js

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING

FIELD OF INFORMATION TECHNOLOGY

Software architectures for implementing distributed autonomous organizations (DAO) with blockchain technologies

DIPLOMA THESIS

Starantzis Dimitris

Supervisor : Vassilios Vescoukis

Professor of NTUA

Athens, March 2022

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING

FIELD OF INFORMATION TECHNOLOGY

Software architectures for implementing decentralized autonomous organizations (DAO) with blockchain technologies

DIPLOMA THESIS

Starantzis Dimitris

Supervisor : Vassilios Vescoukis

Professor of NTUA

Approved by the three-member examination committee in 11 March 2022.

.....	.....	.....
Vassilios Vescoukis Professor of NTUA	Aris Pagourtzis Professor of NTUA	Fotakis Dimitrios Associate Professor of NTUA

Athens, March 2022

.....

Starantzis Dimitris

Graduate Electrical Engineer and Computer Engineer NTUA

ABSTRACT

The purpose of this diploma thesis is to develop a decentralized autonomous organization or simply DAO, using blockchain technology. Bitcoin was used to better understand the basic principles of blockchain technology. In addition to these basic principles, we explain smart contracts, the key component of a DAO. We present a new form of digital organization, the DAO, which consists of and is controlled exclusively by individuals or companies as peer members. As a development tool we chose Hyperledger Fabric, one of the most successful tools for developing private permissioned blockchain networks. We conclude that its pioneering architecture and all its tools make it suitable for any possible implementation of a DAO. We look at the problems of a trusted 3rd party in the example of sharing electricity generation between producers of an electricity aggregator. We propose as a solution to these problems, the creation of a DAO with its equal members, the producers of an aggregator. We develop the structure of a DAO, a private permissioned blockchain network consisting of 4 producers, using the Fabric tool. On this network we define the smart contracts that constitute the business logic and the rules that govern the DAO. For the interaction of producers with the functions of DAO we build a user interface. We compare in terms of usability and suitability for the development of a DAO, the Fabric tool with the GoEthereum tool which is one of the most successful in the field of public blockchains. We also find that DAO is an effective solution to the problems posed by a trusted 3rd party. We show that Fabric is a mature and flexible tool for developing a private blockchain network but requires a great deal of familiarity from its administrators. Finally, by comparing the two tools, we conclude that Fabric is more difficult to use than GoEthereum due to its complexity, but more convenient and more efficient.

SUBJECT AREA: Blockchain, Software Technologies

KEYWORDS: Blockchain, DAO, Bitcoin, Hyperledger Fabric, Smart Contracts, Chaincode, Decentralized Applications.

ACKNOWLEDGMENTS

For the completion of this diploma thesis, I would like to thank my colleague Sergios Exarchos, a graduate of the Department of Digital Systems of the University of Piraeus, for his cooperation and valuable contribution to its completion.

In addition, I would like to thank my professor Mr. Vassilios Vescoukis and the PhD Candidate and Researcher Giannis Tzannetos, for their important contribution to the supervision of this work.

Finally, I would like to thank my parents and my sister, who have been with me all these years and support me constantly in many areas.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

The growing interest in digital security issues is of increasing concern to the world, but also the evolution of the science of cryptography, which focuses the interest of scholars in maintaining security, led to the creation of blockchain technology. The requirement to automate business-to-business (B2B) transactions makes the need to use blockchain and the creation of distributed autonomous organizations even more pressing.

The basic structure of this thesis focuses on the study of blockchain technology, with the aim of understanding it on a practical level, focusing on the development approach. More specifically, the present work aims to demonstrate blockchain technology and how it works. For this reason Chapter 2 describes the structure of the blockchain, introduces concepts such as transactions, but also describes the structure of the block, the chain linking of blocks and the concept of the private blockchain. Bitcoin was used to better understand the basic principles of blockchain technology, although this technology is not limited to the financial sector.

In Chapter 3 we analyze the Hyperledger Fabric software, we explain the tools that make it up and our experience with their use in the implementation of a distributed autonomous organization.

In Chapter 4 we describe the use of a distributed autonomous organization in the problem of sharing electricity generation between providers. We also describe what is a decentralized application that we implemented for the solution of the above problem.

In Chapter 5 we explain the process of implementing the decentralized application. We divide the implementation into 4 sections, which are the user interface section, the middleware section, the business logic section of a DAO through the use of smart contracts and the DAO structure section through the explanation of the network.

In Chapter 6 we present a simulation of the use of the decentralized application. This is achieved through the use of the decentralized application by the DAO member producers. The usage scenario is examined through screenshots which are analyzed.

Chapter 7 compares the Fabric tool and the GoEthereum tool for decentralized application development that operates as a DAO. We consider observations made during the DAO implementation process.

Chapter 8 summarizes the conclusions of the thesis on blockchain technology, DAO and the Fabric tool. We also list some possible future uses of this work.

In summary, the contribution of the present thesis together with the fraternal collaboration of Sergios Exarchos of the Department of Digital Systems development of Piraeus University is mainly located at the level of study and comparison of Ethereum and Hyperledger Fabric in the development of a decentralized application, as well as in the research and implementation of a decentralized application that will have as its ultimate purpose to demonstrate the usefulness of a DAO. Thus, the present study has a research with a comparative method and extends to the level of programming application.

2 Introduction to Blockchain

2.1 Basic concepts of blockchain

Blockchain technology is the fifth computer revolution [1]. Its main goal is to replace the trusted third party that now exists in most transactions. For example, for any money transaction we typically rely on the integrity of the bank as the trusted third party. The predecessor of blockchain is a 1991 document entitled "How to timestamp a digital document" [2]. The blockchain is based on a decentralized peer-to-peer connection system. A blockchain network is made up of a distributed network of independent users. The computers participating in the network can be located in more than one location, thus providing maximum security to the system. The success of Bitcoin is based on this feature. Bitcoin appeared in a research paper in 2008 [3] and was released as software in 2009. Reading it is essential to understanding the idea behind the creation of blockchain technology. Questions such as "what problem does it solve?", "what solution does it offer?" and "what components are used?" lead the reader to a deeper understanding. Once information has been registered in a blockchain, it is impossible to delete it unless the majority (> 50%) of network users decide to do so. Thus, commercial, corporate and banking transactions acquire the maximum possible immutability and security.

The basic element of blockchain is a block [1]. Blocks are groups of transaction entries that are registered in a public register (ledger) and therefore look like key-value lists. Then, their connection forms a chain through the method of hashing one block with the next. When we change some data in a block, we change its hash value and as a result the hash value of all blocks after that. Hashing technology aids the security of the system [4] as each block is created only from data-rules of the previous block, so that a block recognizes only its predecessor and not any other. The hash method is used to generate a unique digital fingerprint for the data of a block. A block header contains details about the data structure within the block (such as the hash of previous blocks, the time the block was created, the merkle root, and the nonce). Time marking helps us to avoid double spending. The merkle root is a hash that represents every transaction on the block. The whole set of transactions can be reconstructed by reversing the method of hashing. Nonce is an arbitrary, unique number used to prevent replay attacks. The size of the block imposes a limit on the number of transactions that a blockchain network can process per second and thus may appear to impede its ability to escalate. Finally, each node in the network has a copy of the blockchain which he updates each time a new block is added.

The blockchain is supported by a type of peer-to-peer network. Any chat application uses a peer-to-peer network to allow open communication between its users. In a distributed network, everyone receives a copy of the information with the same access and control as any other user.

Consensus is a set of algorithms used to agree on which transactions are valid. The problem of the Byzantine generals: Nine generals, each commanding a section of the Byzantine army, encircle a city and must form a plan to attack it. They vote either for attack or for retreat. All nine must agree to attack or all nine must agree to retreat, because if part of the army retreats they have a high risk of losing. Suppose 4 generals vote for retreat, 4 generals vote for attack and a general sends a "retreat" vote to the retreaters and sends an "attack" vote to the attackers.

Bitcoin and Ethereum use the proof-of-work algorithm to solve the problem of the Byzantine generals. Other important algorithms are the proof-of-stake, proof-of-authority or the 3-phase execute-order-validate method used by Hyperledger Fabric.

2.2 Transaction Management

As mentioned in the previous section, a block is nothing more than a group of transactions. The blockchain manages all these transactions using a digital signature mechanism. We need a digital signature mechanism to make transactions, to get an address that we can share with others to make transactions with them but also to be able to prove that the transactions were made by us. To better understand the transaction management process and the digital signature mechanism used by a blockchain, we will analyze the Bitcoin wallet.

A Bitcoin wallet has a private key, a public key and a wallet address. A private key acts as the user's secret code and should not be shared with anyone. A wallet address works in a similar way to a home address, in the sense that it is used as the shipping address of our transactions. Wallet address is the only representation of the keys that users will see regularly, because this is the address to share with the world. The public key is an illegible form of wallet address and the user does not need to know it.

The public key is the result of the ECDSA algorithm [5] with private key as input. A wallet address comes from a public key. Initially the public key is used as the input of the SHA256 algorithm [6] and the result (a number of 256 bits) is transferred as the input of the RIPEMD-160 algorithm [7]. The result is a 160 bit number and is given as input to the BASE58CHECK algorithm [8] to create a more readable code called a wallet address.

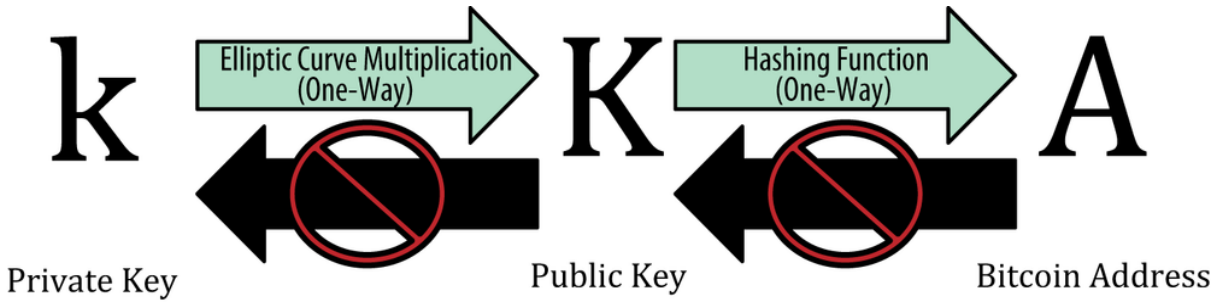


Figure 1: Digital signature mechanism in Bitcoin

Private keys are generated by a cryptographically secure random number generator (such as libraries [9] and [10]). A digital signature creates proof of ownership for each blockchain transaction. **Example**: Dimitris wants to send a Bitcoin to Sergios. Dimitris takes the address of Sergio's wallet and creates a transaction with a charge of 0.003 Bitcoin. He then verifies its details and sends the transaction. The wallet signs its transaction using its private key and is transmitted to the transaction waiting queue (mempool) within the network. The transaction is at some point accepted by the miners, they put it in a block, the miners find the nonce and assign a hash value to the block. The block is placed on the blockchain and as it gains confirmations, is accepted as a valid transaction on the network and finally Sergios gets his Bitcoin.

2.3 A transaction

In the previous section we analyzed the digital signature mechanism used to manage transactions in the blockchain. In this section we will study the components that make up a transaction in a blockchain network. We know that a transaction is the fundamental element of a data block. A transaction is a data structure that encodes a transfer of value from a source called "input" to a destination called "output". In a transaction, the inputs are simply the unspent outputs from another transaction (UTXO).

Transaction = “version” + “input count” + “input info” + “output count” + “output info” + “locktime”.

Input info = “previous output hash” + “previous output index” + “unlocking script size” + “unlocking script” + “sequence number”

Output info = “amount” + “locking script size” + “locking script”

Each UTXO has a locking script that contains the conditions required to spend it. To validate a transaction, we grab the unlock script contained in the input and check if it unlocks the UTXO lock script. Note that the unlock scenario in a transaction does not unlock the lock scenario of the same transaction, but of the previous transaction.

2.4 Smart Contracts

Smart contracts debuted on Ethereum's main public blockchain network. By using smart contracts, blockchains can execute arbitrary, programmable transaction logic. Bitcoin scripts were the earliest form of smart contracts. A smart contract operates as a reliable distributed application and gains its reliability from blockchain technology and the corresponding consensus protocol of each network.

Many existing blockchains that use smart contracts implement a protocol or consensus or atomic broadcast that first sorts the transactions and distributes them to all peers and second, each peer executes the transactions sequentially. This process is called order-execute architecture and it requires all peers to execute each transaction and all transactions to be deterministic.

2.5 Distributed Autonomous Organization

2.5.1 What is a DAO

A distributed autonomous organization is a community of members without central leadership. Members can be individuals or organizations. It operates as a fully online business that is collectively owned and managed by its members. None of its members has the authority to access the organization's data without the group's approval. Decisions are governed by proposals and voting to ensure that everyone in the organization has a voice.

There is no executive member who can make decisions based on his own interests and no chance of manipulating the data. Everything is transparent and the rules regarding the actions of the DAO are incorporated through its code.

2.5.2 Why we need a DAO

To start an organization, we need to show great confidence in the people we intend to work with. But it is difficult to trust someone we do not know or who have only interacted online. With a DAO we do not need to trust anyone else in the team, only the DAO code, which is 100% transparent and verifiable by anyone. This creates opportunities for collaboration and coordination of unknown members without a trusted third party.

Below is a comparison table of a DAO with a traditional organization for a better understanding of the cases in which we would choose a DAO.

Table 1: Comparison of a Distributed Autonomous Organization with a Traditional Organization [11]

Distributed Autonomous Organization	**Traditional Organization**
Usually flat and fully democratic	Usually hierarchical
A vote is required to implement any change	Depending on the structure, changes may be required by one member or a vote may be offered
The votes are counted and the result is applied automatically without a trusted third party	If a vote is offered, the votes are counted internally and the result is handled manually
The functions of the organization are addressed automatically	Requires human manipulation or centrally controlled automation, prone to manipulation
Every activity is transparent and fully public	The activity is usually private and limited to the public

2.5.3 How a DAO works

The backbone of a DAO is its smart contract. The contract defines the rules of the organization and manages its functions. Once the contract is in place, no one can change the rules except by voting. If someone tries to do something that is not covered by the rules and logic of the code, they will fail. The team makes decisions collectively and functions are automatically delegated when votes are counted. This is possible because smart contracts are unchanged from the moment they are activated. A member can not just modify the code (DAO rules) without the other members realizing it because everything is public.

A member of a DAO implemented in Fabric is an organization. Each such organization has 3 options: (1) it can install a smart contract or a new version of it, (2) it can sign (accept) a smart contract and (3) it can update the rules of the organization by enabling the new smart contract only if the majority has accepted the contract.

2.6 Comparison of private and public blockchain networks

Blockchain networks can be divided into two categories: permissionless and permissioned. Permissionless networks are also known as public, while permissioned networks are also known as private.

Public blockchains include platforms that allow network participation without selection or other approval process. In short, in a public blockchain everyone has the opportunity to receive a copy of the blockchain and join the network, provided that the platform protocols are implemented. Public blockchains typically include an inherent cryptocurrency and often use PoW-based consensus algorithm and financial incentives.

Private or permissioned blockchains, on the other hand, form a network between a set of known and recognized participants. A member of the network may invite a potential new member, who, if approved by the majority of the other members, is added to the network. A private blockchain provides a way to secure interactions between a group of

entities that share a common goal but do not fully trust each other, such as companies that exchange capital, goods or information.

To compare the two types of blockchain networks we will consider three factors: scalability, security and immutability. By "scalability" we mean the ability of the network to maintain its efficiency as the size of the network load increases. The "security" factor is related to the ability of the network to remain invulnerable to attacks by malicious users who want to falsify the registry to their advantage. Finally, "immutability" is defined as the ability of the network to maintain the state of the blocks unchanged from the moment they are added to the network.

Public blockchains are less effective than private blockchains. Based on peer-to-peer identities, a private blockchain can use traditional byzantine fault tolerance or simply BFT, which is faster at adding new blocks than the traditional proof-of-work algorithm used on common public blockchain networks. Also, public blockchains face scalability issues due to the presence of multiple nodes, compared to private ones and as a result the rate of adding new blocks to the blockchain is rapidly decreasing.

In a Sybil attack, hackers create and use many fake network identities to flood the network and destroy the system. By carrying out Sybil attacks, hackers can gain disproportionate influence over honest network nodes if they create several fake identities. They can then refuse to receive or transmit blocks, effectively blocking other nodes from a network. If these nodes reach numbers >50% of the network, then the attack is called "51% attack". Having more than 50% control of the network means controlling the registry and being able to tamper with it to reverse transactions. Globally, businesses lose about $20 million a year due to 51% attacks [12]. In a private blockchain network such attacks are avoided since all users are known and have entered the network by invitation or during the creation of the network.

Public blockchains are completely immutable, meaning that once a block enters the chain, there is no way to change or delete it. This ensures that no one can just change a particular block and benefit to the detriment of other users. Private blockchains, on the other hand, are immutable except in exceptional cases where they can delete a particular block if they see fit.

Table 2: Comparison of public and private blockchain

	Public blockchain	Private blockchain
Permission	With permissions	Without permissions
Scalability	Harder	Easier
Vulnerability	Less safe	More safe
Immutability	Easier	Harder

2.7 Popular blockchains

2.7.1 Public

Bitcoin

Blockchain technology first appeared when a scientific paper entitled "Bitcoin: A peer-to-peer electronic cash system" was published in 2008 by an unknown author under the pseudonym Satoshi Nakamoto [3]. Bitcoin is still the most popular cryptocurrency today, but it faces some key issues, such as the high waste of resources to verify each block and the slow pace of transaction validation.

Ethereum

In late 2013, a team of developers consisting of Vitalik Buterin, Jeffrey Wilcke and Gavin Wood worked to create a distributed open source computing platform based on blockchain, Ethereum [13]. Their goal was to create a fully trusted smart contract platform.

Ethereum is an open blockchain platform that allows anyone to develop and run decentralized applications (or simply DApps) on blockchain technology. This platform focuses on the possibility of digital asset management and, to do so, supports smart contracts. Like Bitcoin, no one can own or control the Ethereum platform. Uses Ether cryptocurrency to power the entire Ethereum ecosystem.

Stellar

The Stellar platform is a distributed ledger platform, specifically designed to facilitate value transfers between assets [14]. With the Stellar Consensus Protocol (SCP), it is possible to reach a consensus without relying on a closed financial transaction recording system. Unlike the PoS and PoW consent algorithms, SCP reduces barriers to entry for new members. Its security is based on a voting mechanism and the implementation of a trusted algorithm between nodes that do not trust each other (Federated Byzantine Agreement FBA).

2.7.2 Private

R3 Corda

The R3 Corda platform is an innovative blockchain platform for businesses, which aims to reduce the cost of business transactions and increase their speed [15]. Originally designed for the financial sector only, it can now be applied to other uses, such as healthcare, supply chain, government and public services, and trade finance.

Quorum

The Quorum platform was created by the company JP Morgan and is the corporate version of the Ethereum blockchain [16]. By modifying the Ethereum kernel, the Quorum platform can quickly integrate Ethereum updates.

It is an open source blockchain platform that uses vote based algorithms to execute hundreds of trades per second. As it is a private blockchain platform, it only allows authorized participants to take part in transactions.

Ripple

The Ripple platform aims to connect financial operators, trading companies, banks and payment service providers [17]. Ripple allows international payments through a digital asset called Ripple or XRP.

Using a probabilistic voting method, the Ripple platform achieves consensus between nodes on the network. Several large companies such as American Express, SBI Holdings and Deloitte are experimenting with Ripple's blockchain capabilities to transform payment processes.

Hyperledger Foundation

According to Brian Behlendorf, Executive Director at the Hyperledger Foundation: “As new technology develops, there is a call for standards. Participants want to focus on time and effort and investment to build solutions versus worrying about the framework. This is the rationale for open standards... we are pulling together the most exciting portfolio with a multilateral developer and vendor community. It’s similar to the benefits that Linux brought to the world of operating systems.” [18]

The Hyperledger Foundation provides an open source blockchain system portfolio [19]. Some of these are distributed ledger technologies, others are libraries, others are domain specific, and some are tools.

Distributed ledger technologies are Fabric, Sawtooth, Iroha, Besu and Indy. These five blockchain technologies can be used to develop applications in different areas, but a careful assessment of their strengths and weaknesses is required to select the most appropriate one to develop an application. The libraries are Aries, Transact and Ursa. The Grid targets the specific sector of the supply chain. The tools are Avalon, Bevel, Cactus, Caliper, Cello, Explorer and Firefly.

Hyperledger was launched in December 2015 by the Linux Foundation. The main purpose of using the Hyperledger platform is to enhance the reliability and performance of blockchain technologies.

Hyperledger Fabric

The Hyperledger Fabric platform is one of the most popular Hyperledger Projects designed to build blockchain applications using a flexible architecture. In this way it offers network designers the necessary freedom to implement a blockchain network as they wish. This flexible architecture sets Hyperledger Fabric apart from other blockchain platforms.

Hyperledger Fabric is designed for permissioned networks and allows only entities with known identities to participate in the system. Only authorized participants can participate in the transactions made on the Hyperledger Fabric platform.

Hyperledger Sawtooth

The Hyperledger Sawtooth platform was originally created thanks to the contribution of Intel. It offers a dynamic consensus platform that allows consensus algorithms to be switched on while the network is in operation. Using the Proof of Elapsed Time (PoET) consensus mechanism, Hyperledger Sawtooth can be integrated into computer systems security solutions called trusted execution environments.

Hyperledger Iroha

The Hyperledger Iroha platform was founded by the Linux Foundation to create reliable, fast and secure decentralized applications in blockchain technology. It is based on a very

fast and secure consensus algorithm, the Yet Another Consensus (YAC), which protects nodes from failures or other malicious nodes. Hyperledger Iroha is being used in Cambodia to set up a new payment system alongside the National Bank of Cambodia [20], and various other projects in the fields of healthcare, finance and identity management.

Hyperledger Indy

Hyperledger Indy provides tools, libraries, and reusable data to provide digital identities based on blockchains or other distributed registers, so that they are interoperable in administrative domains and applications. Indy is interoperable with other blockchains or can be used standalone to power identity decentralization.

Hyperledger Besu

Hyperledger Besu is a variant of Ethereum designed to be business friendly for both public and private networks. It can also be run on Ethereum test networks such as Rinkeby, Ropsten and Görli. Hyperledger Besu includes several consensus algorithms, including PoW and PoA (IBFT, IBFT 2.0, Etherhash and Clique).

3 Hyperledger fabric

3.1 Why Hyperledger Fabric;

Hyperledger Fabric is the first distributed ledger platform to support smart contracts written in generic programming languages such as Java, Go, and Node.js, instead of Domain Specific Languages (DSL), such as Ethereum's Solidity. This means that most companies already have all the skills needed to develop smart contracts and no additional training is required to learn a new language.

Fabric can utilize consensus protocols that do not require any cryptocurrency to mine or power the execution of smart contracts. Avoiding cryptocurrency reduces some significant sources of risk, and the absence of cryptographic mining functions means that the platform can be deployed at about the same operating cost as any other distributed system.

Fabric introduces a new flexible and efficient architecture. This new architecture manages to significantly improve its performance, ie. the speed of inserting new blocks into the distributed ledger. Fabric's architecture has always been focused on providing flexibility to its administrators. In this way it gives the possibility to adapt for any design choice in every possible case of use.

The above features in combination with the rich documentation and its widespread use, make Fabric one of the most attractive platforms for private permissioned blockchain networks available today.

3.2 Basic concepts

Hyperledger Fabric or simply Fabric is a configurable and scalable open source system for the development and operation of private blockchain networks and is one of the Hyperledger projects hosted by the Linux Foundation [19]. Fabric is a distributed operating system that runs distributed applications written in general purpose programming languages (e.g. Go, Java, Node.js). It securely tracks its execution history in a replicated ledger where it can only append data and has no built-in cryptocurrency.

Each transaction is executed (endorsed) only by a subset of peers, which allows parallel execution and addresses the potential risk of non-deterministic execution, based on the "execute-verify" mechanism of BFT replication [21]. It uses a flexible endorsement policy that determines which peers, or how many of them, must guarantee the proper execution of a smart contract. Fabric achieves a deterministic execution since the results of the transaction are recorded in the ledger only if a consensus is reached for the absolute ordering between the transactions, in the validation step performed by each peer separately. In addition, Fabric enables the application to choose its support policy. The Fabric ordering service (release-2.2) follows the CFT (crash fault-tolerant) consensus algorithm and more specifically gives the choice between RAFT and KAFKA.

In the next section we will describe the pioneering Fabric execute-order-validate architecture based on the following essential components:

- An ordering service implements atomic broadcast (section 3.4.2) for peer-to-peer status updates and obtains consensus on the ordering of transactions.
- A certificate authority is responsible for associating peers with electronic cryptographic identities. Maintains the licensed nature of Fabric.

- An optional peer-to-peer gossip service spreads the results of the blocks from the ordering service to all peers.
- The smart contracts in Fabric are executed in a container environment for isolation. They can be written in standard programming languages, but do not have direct access to the ledger status.
- Each peer maintains the ledger locally in the form of a blockchain and a snapshot of the latest status in the form of a key-value store.

3.3 Architecture

All previous blockchain systems, permissioned or not, follow the order-execute architecture (Figure 2). This means that the blockchain network orders the transactions first, using a consensus protocol, and then executes them in the same order across all peers in sequence. The order-execute architecture is conceptually simple and therefore widely used. However, it has many disadvantages when used in a public domain permissioned blockchain. The three most important disadvantages are the sequential execution, the non-deterministic code and the confidentiality of the execution. Sequential execution of transactions on all peers limits the effective performance that can be achieved by the blockchain. Executing non-deterministic transactions leads the distributed ledger to a fork, which violates the basic premise of a blockchain that all peers maintain the same status. To achieve the execution of smart contracts with confidentiality, standard blockchain networks use cryptographic techniques which in combination with the execution of all smart contracts on all peers leads to a significant burden of performance and is not viable in practice.

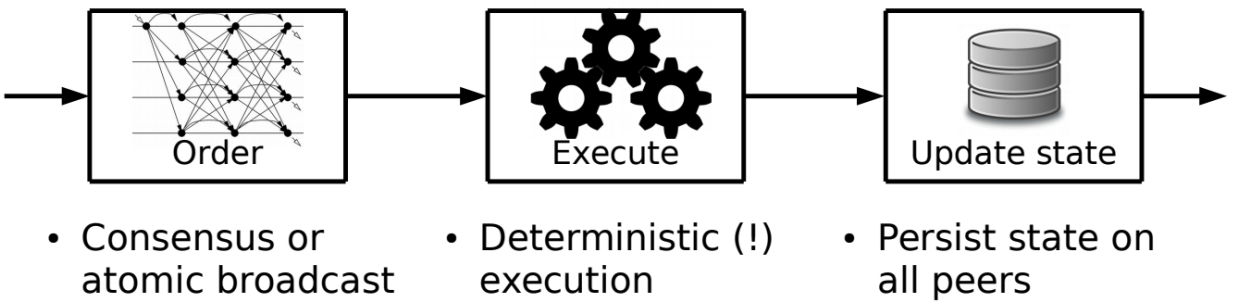


Figure 2: Order-execute architecture of typical blockchain networks. [22]

Fabric introduces a new blockchain architecture aimed at durability, flexibility, scalability and confidentiality. The Fabric's execute-order-validate architecture (Figure 3) was designed for distributed execution of unreliable code in an unreliable environment. It divides the transaction flow into three steps, which can be executed in different entities of the system: (1) execute a transaction and check its correctness, resulting in its approval (corresponding to "transaction validation" in other blockchains). (2) ordering through a consensus protocol, regardless of the semantics of the transaction, and (3) transaction validation based on the endorsement policy of each chaincode, which prevents instances of "race condition" due to parallel execution.

A distributed application for Fabric consists of two parts:

- A smart contract, called a chaincode, is programming code that implements the logic of the application and is executed during the execution phase. The chaincode is the central part of a distributed application in Fabric and can be written by an unreliable developer. There are special chaincodes responsible for configuring the blockchain system and maintaining parameters, collectively called the system chaincode (Section 3.4.6).
- An endorsement policy, evaluated in the validation phase. Endorsement policies cannot be selected or modified by unreliable application developers. An endorsement policy acts as a static library for the validation of transactions in Fabric, which can simply be configured by the chaincode. Only designated administrators may be allowed to modify support policies through system administration features. A standard endorsement policy allows the chaincode to identify sponsors for a transaction in the form of a set of peers needed for endorsement. Uses a logical expression in sets, such as "three in five" or "(A ∧ B) ∨ C."

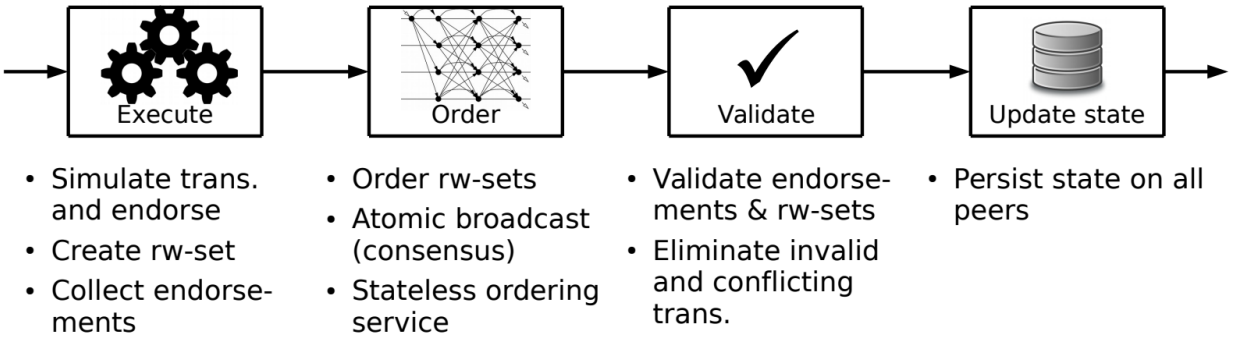


Figure 3: Execute-order-validate architecture of Fabric [22].

A client sends transactions to peers specified by the endorsement policy. Then, each transaction is executed by specific peers and its output is recorded, this step is called endorsement. After execution, trades enter the ordering phase, which uses a consensus protocol to generate a fully sorted execution sequence of endorsed transactions, grouped into blocks that are transmitted to all peers. Fabric classifies transaction outputs in conjunction with state dependencies, as calculated during the execution phase. Each peer then validates the state changes from the endorsed transactions according to the endorsement policy and the consistency of execution in the validation phase. All peers endorse transactions in the same order, validation is deterministic. In this sense, Fabric introduces a new hybrid replication model in the Byzantine model, which combines passive replication (pre-consensus calculation of status updates) and active replication (validation, after consensus, of execution results and status changes).

A Fabric blockchain consists of a set of nodes that form a network. As Fabric is a privately licensed blockchain, all nodes participating in the network have an identity, as provided by a certificate authority.

Nodes in a Fabric network take on one of three roles:

- Client nodes submit execution proposals for execution, help orchestrate the execution phase, and finally transmit transactions for ordering.
- Peer nodes execute transaction proposals and validate transactions. All peers maintain the blockchain ledger as well as the status, which is a summary

representation of the latest ledger status. Peers who execute transaction proposals are called endorsing peers (or simply endorsers), as defined by the smart contract policy to which the transaction belongs. An organization is represented in the network by one or more peer nodes. Each peer node maintains a copy of the ledger of each channel in which it participates as well as a database with their latest status. Each peer node has an administrator who is registered in the database of the certificate authority of the organization. In a peer node we can install as many chaincodes as we want. Each of them operates in its own container and is accessible only by the peer. Each function call of a chaincode must be executed by each peer that has installed it.

- Orderers are the nodes that collectively form the ordering service. In short, the ordering service determines the execution order of all transactions in Fabric, where each transaction contains status updates and dependencies that are calculated during the execution phase, along with cryptographic signatures of its endorsers. Orderers do not maintain the blockchain ledger and do not participate in the execution or validation of transactions.

A Fabric network supports multiple blockchains connected to the same ordering service at the same time. Each such blockchain is called a channel and may have different peers as members. The channels can be used to separate the status of the blockchain network, but the consensus between the channels is not coordinated and the order of execution of the transactions in each channel is different from the others. In some cases where all orderers are considered trustworthy, peer-to-peer access control may also be applied.

3.4 Basic components of Fabric

Fabric is written in the Go programming language and uses the gRPC protocol [23] for communication between clients, peers and orderers. We will know describe some key concepts in detail.

3.4.1 Certificate Authority

The certificate authority is responsible for issuing certificate nodes (clients, administrators, peers, and orderers) that are used for authentication and accessibility. Once Fabric is permissioned, all interactions between nodes take place via authenticated messages, usually digital signatures.

The certificate authority allows an identity management federation to operate, for example, when multiple organizations operate a blockchain network. Each organization issues IDs to its own members and each peer recognizes members of all organizations. This can be achieved with multiple certificate authorities, for example, by creating one certificate authority for each organization. The total sum of the certificate authorities must ensure that all nodes, especially all peers, recognize the same identities and authentications as valid.

3.4.2 Ordering Service

The ordering service manages many channels. In each channel provides the following 2 basic services:

1. Atomic broadcast for the ordering of transactions.
2. Configuration of a channel when its members modify the channel by transmitting a channel configuration update transaction.

An orderer collects transactions and, using the atomic broadcast, orders them and forms blocks. A block is completed as soon as one of the three conditions is met: (1) the block contains the specified maximum number of transactions, (2) the block has reached the maximum size (in bytes) or (3) a certain time has elapsed since the receipt of the first transaction of a new block.

3.4.3 Peer Gossip

One advantage of separating the execution, ordering, and validation phases is that they can be scaled independently. Consensus can not be scaled by adding more nodes [24] [25], in contrast performance will be reduced. However, since the ordering and validation are independent, we are interested in the efficient transmission of the execution results to all peers for validation, after the ordering phase. This, together with the transfer of status to new peers and peers who have been disconnected for a long time, is precisely the goal of the gossip mechanism. Fabric gossip uses epidemic multicast [26] for this purpose. The blocks are signed by the ordering service. This means that a peer can, once he has received all the blocks, assemble the blockchain independently and verify its integrity.

Gossip is based on gRPC and uses TLS with mutual authentication, which allows each side to link TLS credentials to the remote peer ID. The peer gossip mechanism maintains an up-to-date list of active system peers. Additionally, a peer can be reconnected after a crash or network outage.

In order to reduce the load of sending blocks from the ordering nodes to the network, the gossip protocol also elects a peer leader who collects blocks from the ordering service on their behalf and distributes them to them. This mechanism is resistant to failures of leaders.

3.4.4 Ledger

Each peer maintains the ledger and its status and allows the phases of simulation, validation and updating of the ledger. In general, it consists of a block storage and a transaction management mechanism.

The ledger’s block storage maintains transaction blocks and is implemented as a set of append-only files. Since the blocks are immutable and arrive in a defined order, a structure with append only gives maximum performance. Additionally, the block storage retains some indexes for random access to a block or to a transaction on a block.

The transaction management mechanism maintains the latest status in a key-value-version storage. Stores a tuple of format (key, value, version) for each unique key stored by any chaincode, and contains the most recently stored value and its latest version. The key-value-version storage is implemented using LevelDB (in Go) [27] or using Apache CouchDB [28].

In the simulation phase the transaction management mechanism provides a still image of the latest transaction status. In the transaction validation phase, the transaction management mechanism validates all the transactions of a block in sequence. This controls whether a transaction conflicts with any previous transaction (within the block or older). In the update phase the ledger is tolerant of possible peer damage using a "savepoint" mechanism used to retrieve indexes and the last lost status.

3.4.5 Chaincode Execution

A chaincode runs in an environment loosely connected with a peer, which currently supports the Go, Java, and Node.js programming languages.

Each application chaincode is executed in a container Docker environment, which isolates the chaincodes from each other and from the peer. This simplifies the management of the chaincode life cycle (ie start, stop or cancel). The chaincode and peer communicate using gRPC messages. Through this loose connection, the peer is agnostic of the programming language in which the chaincode is implemented.

Unlike application chaincode, the system chaincode runs directly on the peer. The system chaincode can implement specific functions required by Fabric and can be used in cases where the isolation between application chaincodes is too restrictive.

3.4.6 Configuration και System Chaincode

Fabric is configured through channel configuration and through special chaincodes, known as system chaincodes. Remember that each channel in Fabric forms a blockchain. The configuration of a channel is maintained in metadata that remain in special configuration blocks. Each configuration block contains the entire channel configuration and contains no other transactions. Each blockchain network starts with a configuration block known as the "genesis block".

A channel configuration can be updated using a channel configuration update transaction. This transaction contains a representation of the changes that need to be made to the configuration, as well as a set of signatures. Ordering nodes evaluate whether the update is valid using the current configuration to verify that modifications are authorized using signatures. The orderers then create a new configuration block, which integrates the new configuration and configuration update transaction. Peers receiving this block validate if configuration update is authorized based on the current configuration; if validated, they update their current configuration.

4 Use Case: Electricity aggregators and the need to replace the trusted 3rd party

4.1 The energy system in transition

The effects of climate change are now being felt in the daily lives of people around the globe. The targets set at European level for 2030 describe an energy system with significantly lower carbon dioxide emissions. The electricity sector contributes significantly to the decarbonization process, with the production of electricity from Renewable Energy Sources (RES) leading the way. We are now in an era where the cost of electricity production from RES is directly comparable and competitive in terms of production costs of conventional power stations. RES claim an equal role in the electricity system and equal conditions for participation in the electricity market and the institutional framework is being adjusted in this perspective. Electricity aggregators are the new entities in the electricity market through which the full integration of RES units in the electricity market will take place. Aggregators are specialized innovative companies that use modern tools and are called to optimize the commercial management of green energy units by minimizing the cost of balancing with the accurate forecast of their production.

As the participation of renewable energy sources in the market in the coming years will be done in competitive market conditions, aggregators will become more and more necessary for the management of many RES projects, in order to increase the benefit for the producer.

At the institutional level the first big change in Greece was made with the law 4414/2016 [29] and it becomes the first big step as the RES projects that are constructed having won in the tenders of the Energy Regulatory Authority if they have a capacity of more than 3 MW for wind and over 500 kw for photovoltaics, undertake the obligation to participate in the wholesale market and at the same time undertake balancing obligations. This means that if they declare more or less power available than the real one they will bear the cost resulting from the deviation.

Abroad and in more mature markets, such as Italy, many RES companies prefer to place their projects on competitive terms in the market rather than through tenders, as they find that on the one hand auction prices are declining and do not offer satisfactory returns and on the other hand participating in the wholesale market for large RES portfolios can generate higher revenue and returns. These higher yields, achieved through aggregators which manage a large number of RES units, with geographical dispersion, and reduce the uncertainty and variability of RES production, can submit competitive bids that ensure the daily participation of the portfolio in the energy exchange and generate higher revenue. The same is foreseen to be done in Greece with the implementation of the target model which will transfer the responsibility of balancing the market from the Independent Electricity Transmission Operator to the RES producers, while the commercial management of the green units will be done by the aggregators. Aggregators will reduce the discrepancy between forecast and actual production and consequently the variability of the production of RES stations. This volatility and uncertainty will be further reduced in the future when energy storage systems are promoted and implemented in the RES market. [30].

It should also be noted that the first company in this field was Optimus Energy, in which TERNA participates with a 350 MW representation license. Licenses are also held by Mytilineos [31] for 500 MW, Eunice for 300 MW, ELPE Renewables for 300 MW, NRG, the Motor Oil group for 200 MW, Forena Energy [32] for 200 MW, Sentrade for 200 MW as well as other companies.

4.2 What is an electricity aggregator and how does it work

An electricity aggregator is an organization that represents Renewable Energy Producers in the free electricity market. More specifically, we will deal with aggregators that represent producers of electricity through wind farms.

Wind farm or Wind Power Plant is the land or sea area in which a number of wind turbines have been placed in order to convert the kinetic energy of the wind into electricity.

An aggregator (not necessarily a producer itself) enters into agreements with producers to be represented by him. Decisions regarding the operation of an aggregator can be taken through corporate governance procedures. A crucial point is the one that concerns the adjustment of the participation percentages and in general the process of compensations of the participating producers in an aggregator. The market of an aggregator is a subset of the total electricity supply market. Based on the statements of the other producers belonging to the same aggregator, the market share is calculated at a percentage covered by each, which add up to 100%.

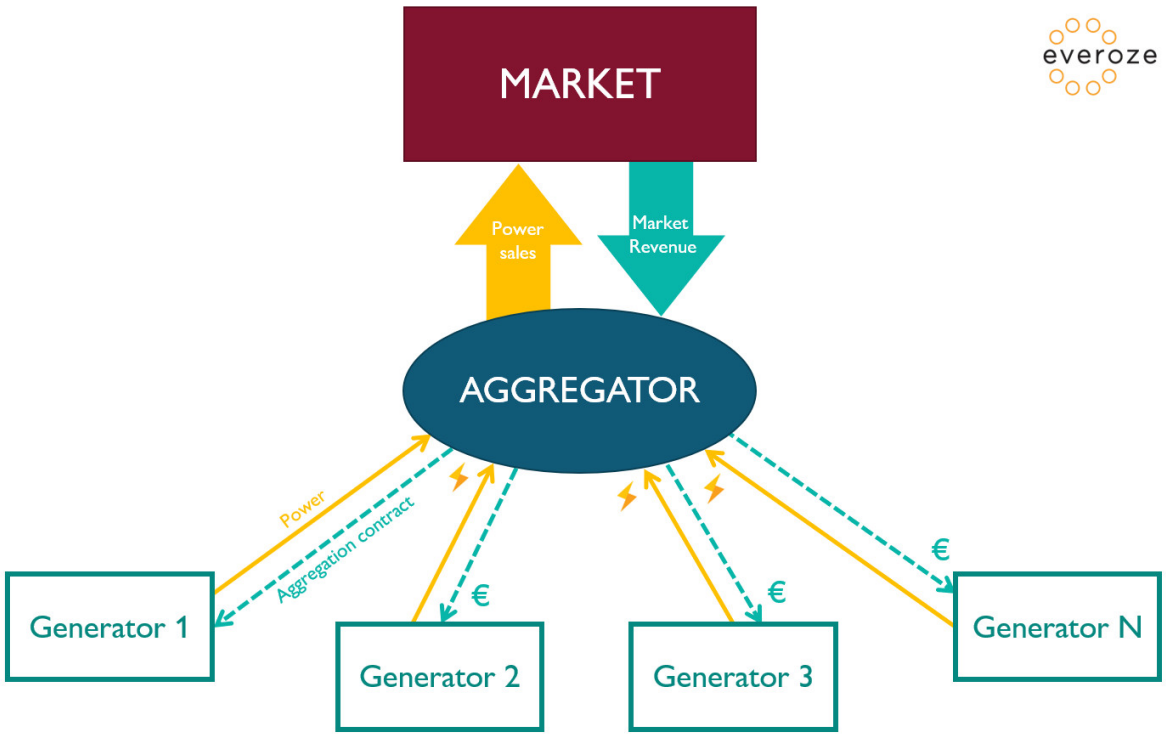


Figure 4: Market share of electricity aggregators and the producers. [33]

Let us study this process in a little more detail. At the beginning of the month, electricity aggregators must make a statement of the energy supply in MWh that it will provide to the total market. This statement must be accurate so that there are no penalties. This amount is calculated from the sum of the energy production forecast declarations of each of its producers. Therefore, at the beginning of each month a producer declares his forecast for the amount of energy in MWh he will produce for the whole month. Aggregators pay each producer based on his declaration. Then each producer, every day records the energy production in MWh every 15 minutes.

Example: Suppose an aggregator representing 4 producers (Org1, Org2, Org3, Org4). The declarations of the producers at the beginning of the month and the production at the end of the month are the following:

Table 3: Use Case: Declaration and production of electricity of the producers of an aggregator.

	Org1	Org2	Org3	Org4
Declaration (MWh)	114	120	180	186
Declaration (%)	19%	20%	30%	31%
Production (MWh)	85	110	140	165
Production (%)	17%	22%	28%	33%
Claim (MWh)	114	120	180	186

We notice that each producer eventually produced less than what he had declared. Therefore, each of them must return the difference back to the aggregator. As shown in the last line of Table 3, the producer Org1 claims that he produced the amount he had declared and that he does not owe money to the aggregator. To solve this problem there is a need for a trusted third party.

4.3 The problem of the trusted third party

A trusted third party collects the data of the producers involved. Thus begins the process of recalculating the production amount of each. As a result of this process, it judges the money owed to and from the aggregator. The problems of using a trusted third party are easily observed. The main problem is that the trusted third party is made up of people who will initiate the whole process of recalculating the percentages and as a result is not incorruptible. In addition, the data of each producer is private and possibly sensitive for each producer. Finally, the trusted third party cannot be invited unless both parties are willing to negotiate. From the example we conclude that if the producer Org1 does not want to negotiate, while the aggregator seeks negotiation, the trusted third party can not be invited. We observe that a similar procedure is followed in the case that the aggregator claims wrong production amounts.

4.4 The solution to the problem

The proposal for the solution of the problem is the following: Construction of DAO with members the producers Org1, Org2, Org3, Org4 to replace the trusted third party. As members of the DAO each producer has a seat on the board and a voting right on it. In case of need to recalculate the production percentages, a producer starts a new voting. Each producer votes at its own time until at least 3 votes are in favor or against the recount. In the case of recalculation the system reads the data of all producers and calculates the new percentages.

In our case, this organization is located and lives on the blockchain and its decisions - operations are modeled on smart contracts. So virtually every producer will be able to vote for or against the recalculation of percentages, seeing only their own history and knowing information about their own system without having access to others, hence the need to use a private permissioned blockchain.

The use case that we will study will investigate an implementation of the decision-making process in an electricity aggregator using blockchain as an Autonomous Decentralized Organization.

5 Implementation: Creation of DAO for the sharing of electricity generation between providers

The functional requirements of the decentralized application are as follows. The blockchain network consists of 4 organizations. Each organization enters in the blockchain its forecast for the amount of energy production in MWh, which it will produce in a period of time. Then every 15 minutes, each organization records in the blockchain the amount of energy in MWh produced during that time. The data of each organization is private, and therefore visible only to him. Any organization has the opportunity to initiate a vote to recalculate production rates. If the majority of organizations vote in favor, then the new percentages are automatically recalculated and become visible to them.

To meet the above operational requirements we implemented a decentralized application consisting of 4 components, which we analyze below.

5.1 Creating a user interface

The user interface is the image of the decentralized application that each manufacturer sees. The aim of the interface is to facilitate each producer to participate in the voting process and to visualize his data through diagrams per day and per month. Additionally the interface offers some additional information such as its production percentage.

To better understand the user interface we created a Component diagram using the Unified Modeling Language (UML). As shown in the diagram below, the interface is divided into 3 main parts: a navigation bar, a voting card and an energy data card.

The navigation bar contains information such as the name of the connected organization, the production forecast percentage and the production percentage.

The entire voting process takes place on the vote card. The vote card contains a message describing the voting status as well as 4 buttons. The "Initialize" vote button initializes the percentage recalculation vote. The positive vote button "vote YES" and the negative vote button "vote NO" register a positive or negative vote for the connected organization. The "Refresh" button updates the vote status message and possibly the production percentage if this has changed due to recalculation.

The energy data card visualizes the data of the connected organization using diagrams per day and per month. The energy data card contains a date option, 2 charts and a button. The connected producer selects the date he wants to visualize and uses the "GET" button to bring the data from the blockchain and construct the charts.

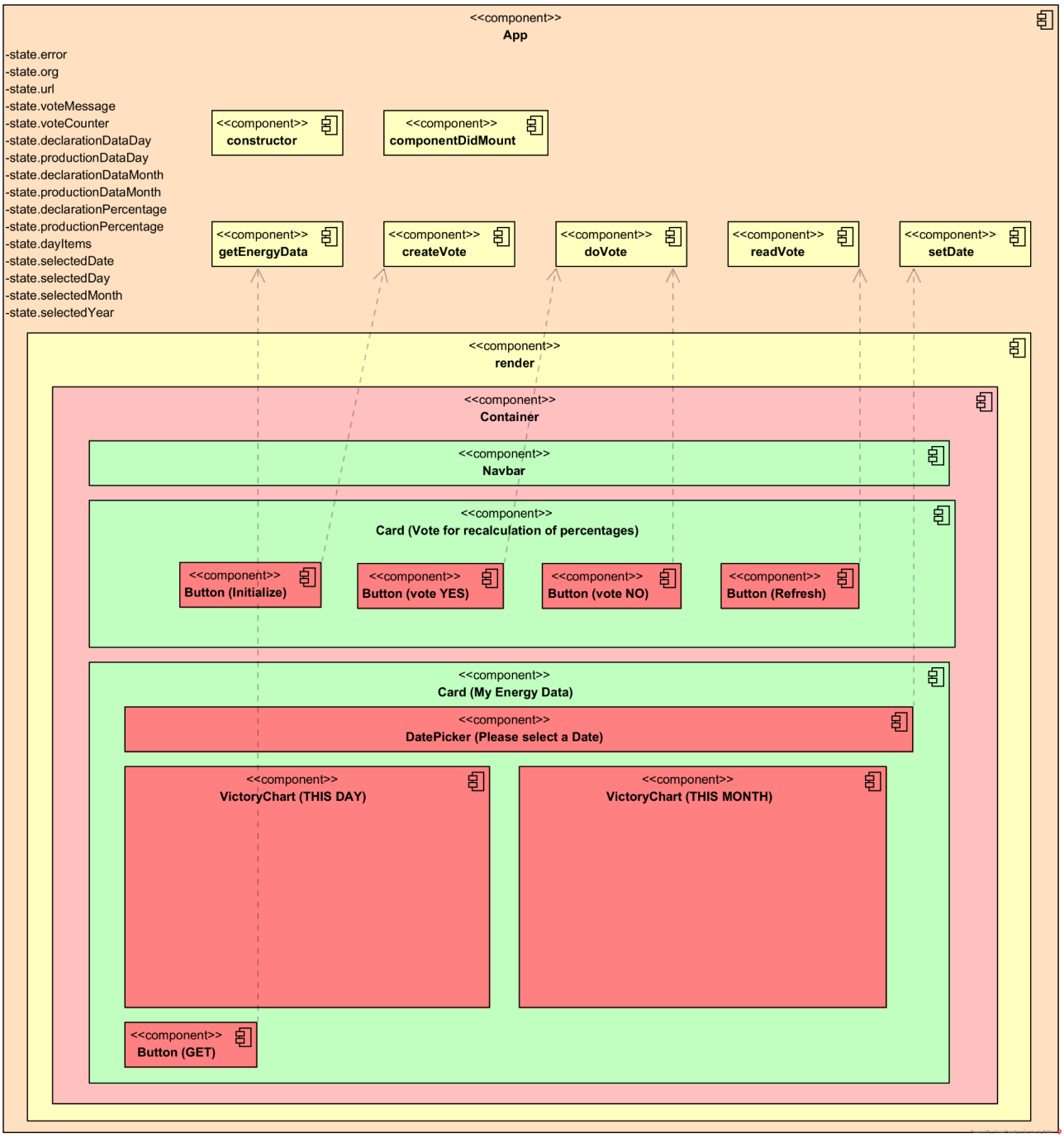


Figure 5: User interface UML component diagram.

The following is a detailed description of each function of the user interface program:

- The "constructor" function. It is the first function called before the initial rendering of the application page. Its purpose is to initialize the state variables that are visible at the top left of the chart.
- The "componentDidMount" function. It is the first function called immediately after the initial rendering of the application page. Used to update some status variables.
- The "createVote" function. Called each time the "Initialize" button is used. The function calls the "/vote/init" resource of the middleware in order to initialize the recalculation vote of the percentages.

- The "doVote" function. Called each time the "vote YES" button or the "vote NO" button is used. The function calls the resource "/vote/yes" or the resource "/vote/no" in order to register the vote of the connected organization.
- The "readVote" function. Called each time the "Refresh" button is used. The function calls the "/vote" resource in order to update the message of the voting status. Additionally, if the vote is completed, call the resource "/energyData/percentage" to update the production percentage.
- The "setDate" function. Called each time the date selector is used. The function updates the status variables for the date.
- The "getEnergyData" function. Called each time the "GET" button is used. Based on the selected date, it calls the resource "/energyData/<month>/<day>" in order to bring the data from the blockchain and create the day and month tables that are necessary for the construction of the charts.

The user interface was implemented using the "React-Bootstrap" software framework [34] and is available in the personal Github repository [35].

5.2 Creating the middleware

The middleware software is the link between the user interface and the smart contracts installed on the network node of the connected organization. The purpose of the middleware is to provide maintainability and adaptability to the overall system by isolating the process of connecting to the network and registering new transactions or validating existing ones. In addition, by isolating the user interface from smart contracts, we enable the use of Internet of Things (IoT) smart devices, which may be connected to a wind turbine, to connect and record production data every 15 minutes.

For a better understanding of the middleware we created a component diagram using the UML modeling language. As shown in the diagram below, the middleware is divided into 2 main parts: a voting router and an energy data router. Additionally, the middleware contains the "getContract" function which is responsible for connecting to the smart contract and the "handler" function which is responsible for returning a suitable message to the user interface.

The voting router handles all resource calls starting with "/vote". Calls to the "/vote/init" resource are routed to the "InitVote" function, calls to the "/vote" resource are routed to the "ReadVote" function, and calls to the "/vote /:proposal" resource are routed to the "DoVote" function.

The energy data router handles all calls to resources starting with "/energyData". Calls to “/energyData/declaration” resource are routed to “PostDeclaration” function, calls to resource “/energyData/production” are routed to “PostProduction” function, calls to resource “/ energyData/:month/:year” are routed to GetMonthlyData function and calls to the resource “/ energyData/percentage” are routed to the “GetPercentage” function.

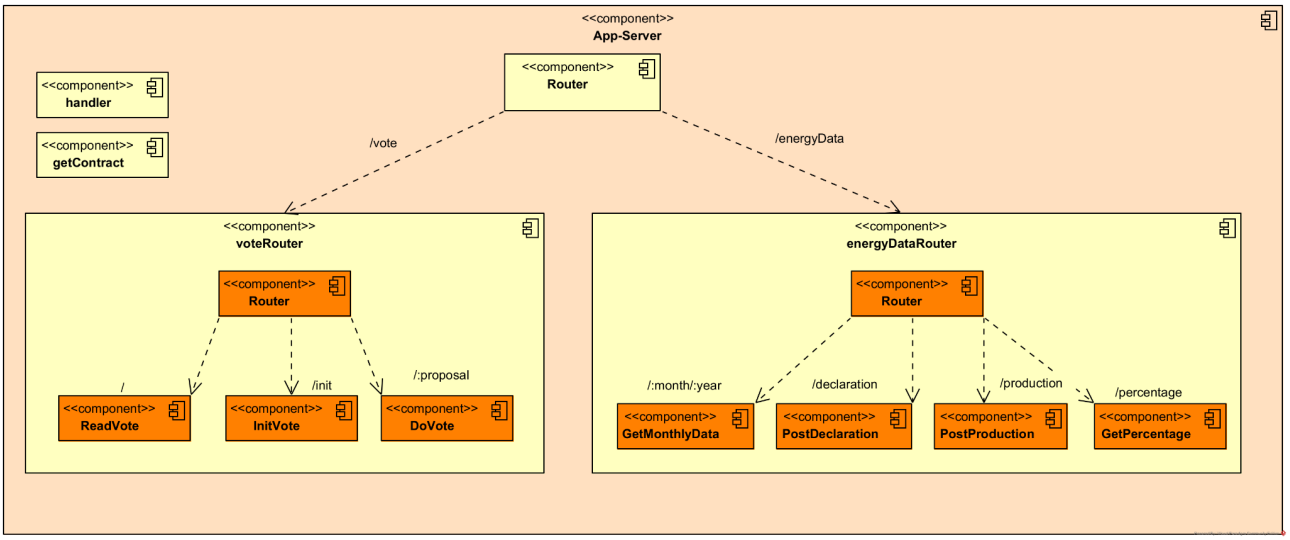


Figure 6: Middleware UML component diagram.

The following is a detailed description of each function of the middleware:

- The "getContract" function. It is called once in the activation of the middleware. Its purpose is to connect to the network node corresponding to the connected organization so that we have access to the installed smart contract.
- The "handler" function. It is called every time we want to make a function call of the smart contract. Its purpose is to activate a function of the smart contract and then return the appropriate message to the user interface.
- The "ReadVote" function. It is called every time the resource "/vote" is requested. Enables the "ReadVote" function of the smart contract to return to the user interface the percentage recalculation voting status.
- The "InitVote" function. It is called whenever the resource is requested "/vote/init". Activates the "InitVote" function of the smart contract in order to initialize the percentage recalculation vote.
- The "DoVote" function. It is called whenever the resource is requested "/vote/:proposal". Activates the "DoVote" function of the smart contract in order to register the vote of the connected organization.
- The "GetMonthlyData" function. It is called whenever the resource "/energyData/:month/:year" is requested. Activates the "GetMonthlyData" function of the smart contract in order to return to the user interface the electricity production data for the selected month.
- The "PostDeclaration" function. Called each time the "/energyData/declaration" resource is requested. Activates the "PostDeclaration" function of the smart contract in order to record the forecast of energy production of the connected organization.
- The “PostProduction” function. It is called whenever the resource "/energyData/production" is requested. Activates the "PostProduction" function of the smart contract in order to record the electricity production of the connected organization for a specific period of 15 minutes.
- The "GetPercentage" function. Called each time the "/energyData/percentage" resource is requested. Activates the "GetPercentage" function of the smart contract

in order to return to the user interface of the registered production percentage of the connected producer.

The middleware was implemented using the "Express.js" software framework [36] and the "Hyperledger Fabric SDK for Node.js" [37] and is available in the personal Github repository [38].

5.3 Creating DAO's business logic

The business logic of DAO is implemented by writing smart contracts. In our use case the smart contract is one and includes the logic of both the voting process and the energy data process. This contract is installed by all producers, accepted (signed) by the majority of producers and finally declared by one producer and activated.

The functions of the smart contract are divided into two main categories: private and public. We name as private the functions that are visible in the smart contract but are not visible to its users, that is, in this case, the producers. The functions that are accessible by those organizations that have installed this smart contract on their own network node are named public. In short, public functions are activated by organizations while private ones are activated by other public ones.

To better understand the smart contract we created a component diagram using the UML modeling language. As shown in the diagram below the smart contract includes 7 public and 2 private functions. It also includes 4 auxiliary private functions. Of the public functions, the first 3: "InitVote", "ReadVote" and "DoVote" cover the voting process while the rest cover the energy data process.

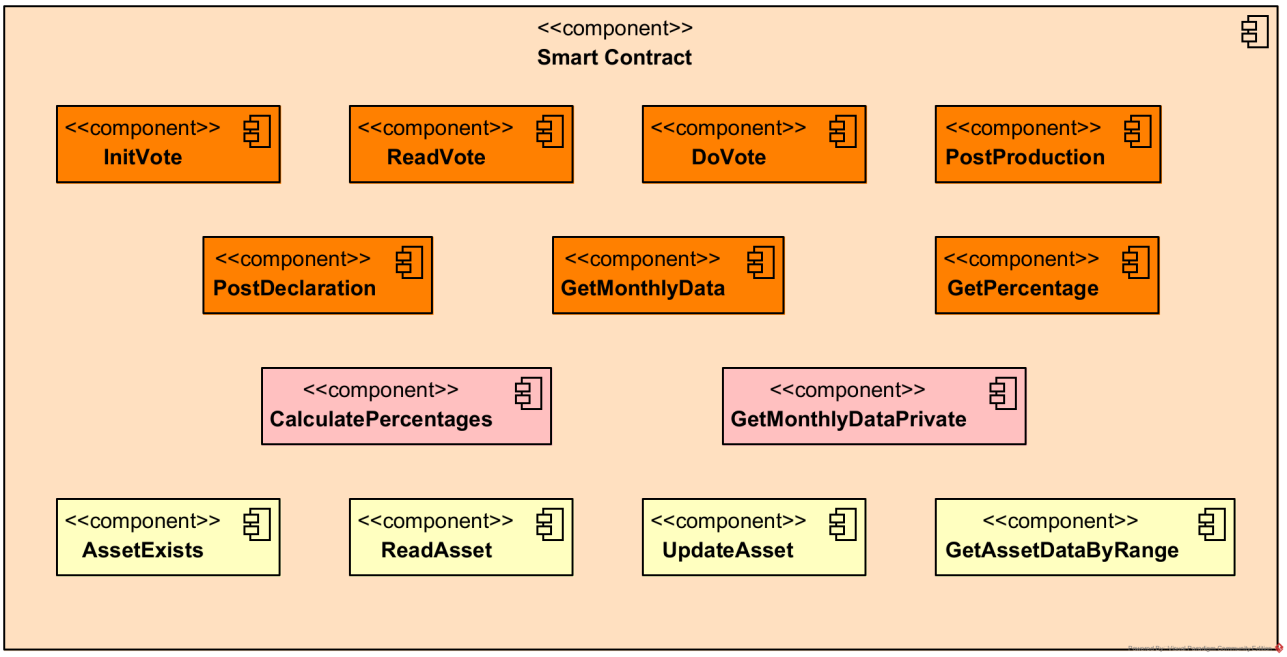


Figure 7: Smart contract UML component diagram.

The following is a detailed description of each function of the smart contract:

Public functions

- The "InitVote" function. Its purpose is to initialize the vote on the recalculation of production percentages. More specifically, it initializes a Javascript object with 4 fields: a message that describes the voting status, a counter of positive or negative votes, the vote of each organization and a variable that indicates whether the voting is complete or not. After initializing the above object, it calls the "UpdateAsset" auxiliary function and registers or updates the blockchain in the position with a unique "vote" key.
- The "ReadVote" function. Its purpose is to return the voting status to the organization that called it. Reads and returns the object corresponding to the unique "vote" key using the "ReadAsset" utility.
- The "DoVote" function. Its purpose is to record the vote of an organization in the blockchain. It first reads the current voting status corresponding to the unique "vote" key using the "ReadVote" auxiliary function. Then, based on the positive or negative vote it reads as input, if all the positive votes are the majority then it calls the private function "CalculatePercentages" in order to recalculate the production percentages. Finally, it updates the voting status using the "UpdateAsset" utility function.
- The "PostDeclaration" function. Its purpose is to record the forecast of electricity production in the private data of the connected organization. It is first checked if this prediction has already been registered using the "AssetExists" utility. If it is already registered then it returns with an error message. Otherwise, this forecast is stored in the private data of this organization using the "UpdateAsset" utility.
- The "PostProduction" function. Its purpose is to record electricity production in the private data of the associated organization. It is initially checked whether the production forecast for this period has been recorded. If it is not registered then it returns with an error message, since the production entry must follow the forecast entry. Otherwise, the output is stored in the private data of this organization using the "UpdateAsset" auxiliary function.
- The "GetMonthlyData" function. Its purpose is to return the monthly production data to the connected organization. The function reads the private data of the connected organization for the requested month using the "GetAssetDataByRange" auxiliary function and returns it to him.
- The "GetPercentage" function. Its purpose is to return the percentage of production to the associated organization. The function reads this private data of the connected organization using the "ReadAsset" auxiliary function and returns it to him.

Private functions

- The "GetMonthlyDataPrivate" function is exactly the same as the "GetMonthlyData" public function we described above. The difference is that the private version accepts as an argument the organization whose data we want to read. This function is called only by the public function "CalculatePercentages" in order to collect all the data of all organizations to calculate their percentages. We observe that this function meets the functional requirement "The data of each organization is private, and therefore visible only to him" since it is private and therefore visible only in the smart contract and not in an organization.
- The "CalculatePercentages" function. This function is called only by the public function "DoVote". Its purpose is to calculate the energy production rates of all organizations. Initially calculates the previous month from the current date. Then,

for the calculated month, it reads the data of all the organizations using the function "GetMonthlyDataPrivate" and calculates both the sum of the energy production for each organization and the total sum of all the organizations. Each organization calculates its production percentage based on its sum to the total sum. Finally, it updates these percentages in the blockchain using the "UpdateAsset" auxiliary function and returns them to the "DoVote" function that called it.

Auxiliary functions

- The "AssetExists" function. Its purpose is to check if a digital asset is registered in the blockchain. Accepts the unique key as an argument and returns "true" or "false".
- The "ReadAsset" function. Its purpose is to read a digital asset registered in the blockchain. Accepts the unique key as an argument and returns the asset if it exists or an error message otherwise.
- The "UpdateAsset" function. Its purpose is to register a new digital asset or to update an existing one. It accepts as an argument the asset and the unique key in which the information will be updated.
- The "GetAssetDataByRange" function. Its purpose is the return of all assets belonging to the requested period of unique keys. Accepts the unique key interval as an argument and returns all assets between that interval.

It is worth noting that the writing of the smart contract was based on the "asset-transfer-basic" found in Fabric's examples [39]. The functions used in this example were placed in a "utils" folder and imported at the beginning of the contract.

The smart contract was implemented using the "Node.js" software framework [40] and the "Hyperledger Fabric Contract API for node.js" [41] and is available in the personal Github personal repository [42].

5.4 Creating a structure of the DAO using the test-network

The structure of the DAO is implemented by the creation of the ordering organization, the creation of the participating organizations as well as the creation of a communication channel. The structure of the ordering organization is defined as one orderer, one certificate authority as well as the various administrators of the organization. Respectively, as a structure of a peer organization, we define one peer node, one certificate authority as well as the various administrators of the organization.

The following is a brief description of the "test-network" used as the default smart contract development network in the examples of the Fabric platform [39].

The process of creating the network begins with the creation of the structure of the ordering organization as well as the creation of the structure of the peer organizations. The first step in setting up a private network is to issue certificates to both the ordering and the participating organizations. Below is the analysis of creating an ordering or a peer organization. First we create in a container a "fabric-ca" certificate authority together with a database for it. Using an administrator and his / her secret code, we start the certificate authority. We log in to the certificate authority using the “fabric-ca-client” program using the administrator's name and password to generate the certificates and register the entities in the database:

- In the case of an ordering organization: the organization administrator and a orderer node

- In the case of a peer organization: the organization administrator, a peer node and an organization user

The second step in creating the network is to create the genesis block that contains the description of all the organizations as well as a consortium of peer organizations that have the ability to create communication channels. This step is performed by the ordering organization.

The third step consists of creating at least one orderer node that belongs to the ordering organization as well as creating at least one peer node for each participating organization.

The fourth step is to create the communication channel. An organization that belongs to the consortium creates the channel and all the other organizations connect their node to it.

The fifth and final step is to install and activate the smart contract. Each peer organization installs the smart contract on its node, accepts it and after collecting the majority of signatures, it is activated by an organization.

The following changes were made to adapt the test-network to our use case. Initially we created the certificates for 4 organizations that correspond to the 4 producers of our use case. In addition the original block contains the description of these four organizations. We created for each organization a peer node, i.e. a total of 4 peer nodes. We connected these nodes to the communication channel. Finally, we accepted the smart contract and activated it.

For a better understanding of all the components that make up the network as well as its topology, a component diagram using the UML modeling language is provided below.

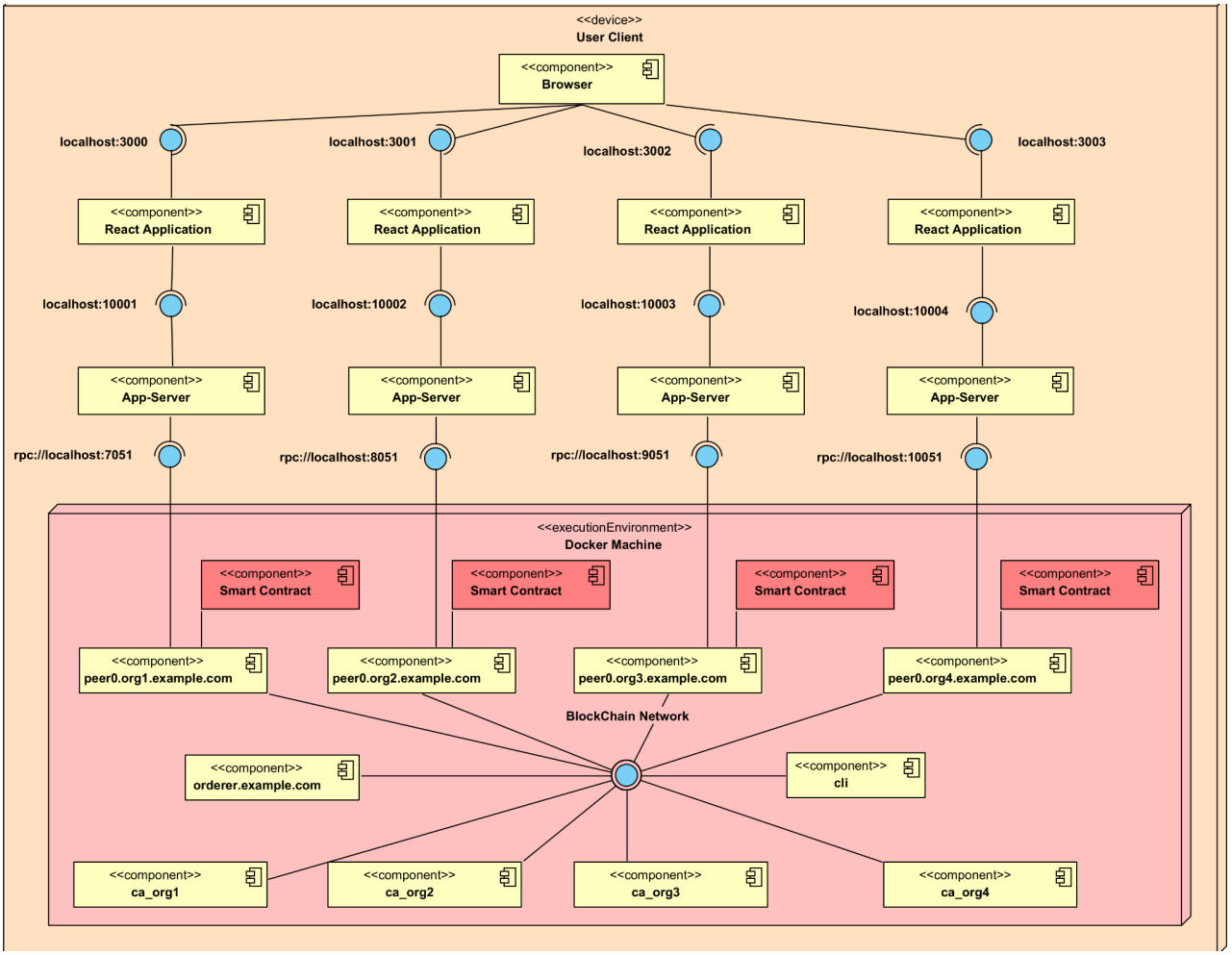


Figure 8: Diagram of UML components of network structure and topology.

The above figure represents the execution of a DAO network in "Docker Machine" as well as the middleware "App Server" and the user interface "React Application" for 4 organizations on one device "User Client". Notice that for each organization we run a user interface, a middleware, a peer node named "peer0.org <NUM> .example.com" in the DAO along with a certificate authority named "ca_org <NUM>" and the Smart Contract.

In the previous subsections of the chapter we analyzed the user interface, the middleware and the smart contract. Using the "test-network" as we analyzed above, we created all the Docker containers that are enclosed by the "Docker Machine". The containers of peer nodes, the ordering nodes, the certification authorities, the smart contracts as well as the "cli" are provided as open source software by the Hyperledger Foundation [43][44][45][46][47].

6 Simulation of the decentralized application

In this chapter we will present a simulation of the DAO functionality described in Chapter 5.

To simulate the decentralized application we followed the steps described in our personal public repository [48]. These steps are:

1. First, we start a new network and activate smart contracts.
2. Second, we open 1 middleware for each organization.
3. Third, we send the simulation data to the blockchain for each organization sequentially.
4. Finally, we open the user interface for each organization.

The original image of the user interface of the producer "Org1" is shown in the following screenshot:

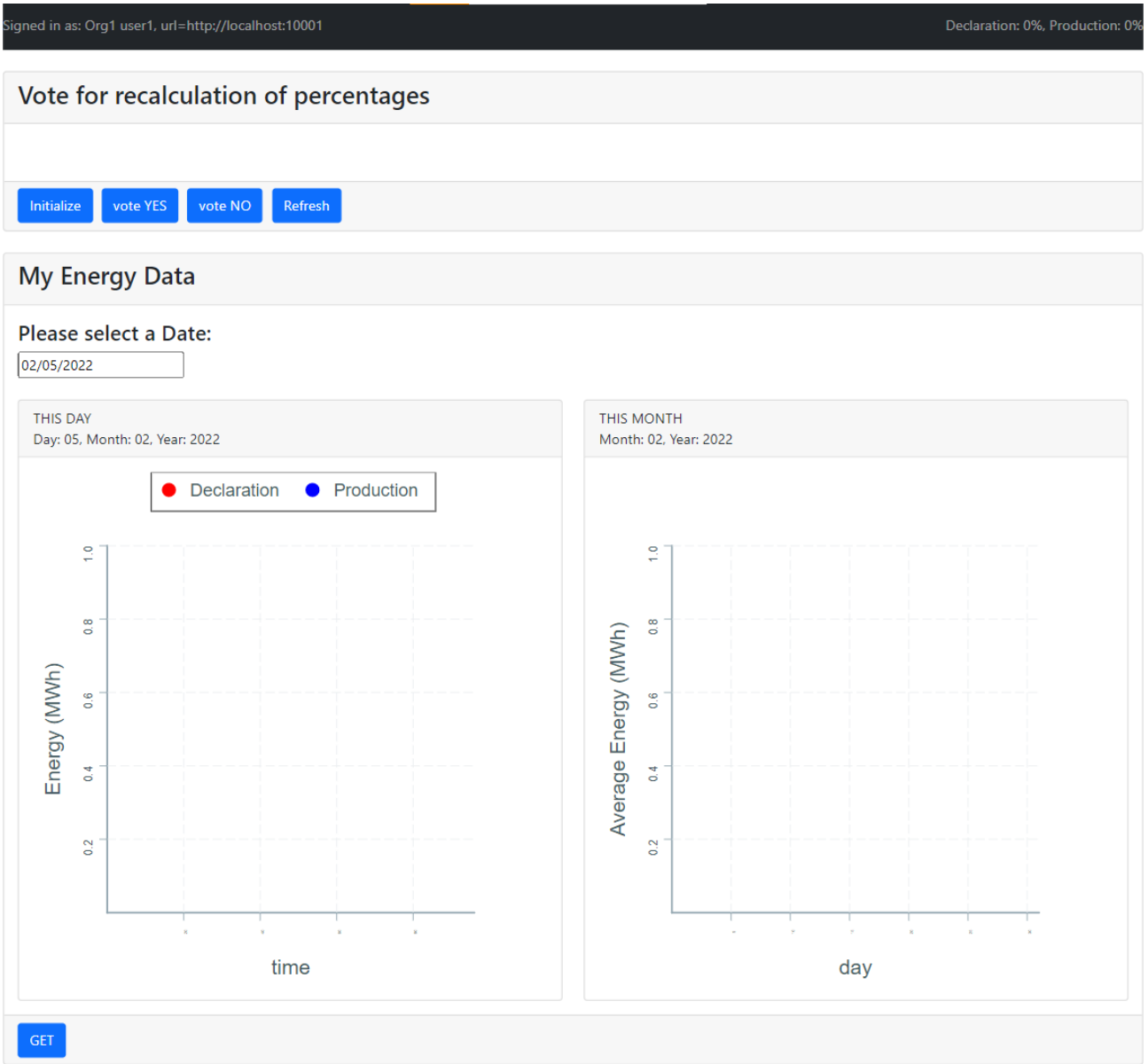


Figure 9: Org1 Producer User Interface Home.

We notice that the interface is divided into 3 parts: a navigation bar, a voting card and an energy data card. The navigation bar shows the organization that is connected "org1", the "url" of the middleware that is connected to the interface as well as the production forecast percentage "Declaration" and the production percentage "Production". The voting card for the recalculation of the percentages contains the message of the voting status as well as 4 buttons related to the voting. The energy data card displays the organization's data for the selected date.

6.1 Display of energy data

To simulate energy data, we created one-month simulation data for each DAO producer. In more detail, we created for the month of January 2022 a data for every 15 minutes of the hour which consists of 3 values: a time stamp, a value for the production forecast and a price for the production. After the data is generated, it is registered in the blockchain. First the production forecast declaration is registered and then the production is registered. Both the production program of this data and the process of their registration in the blockchain are in our personal repository [49].

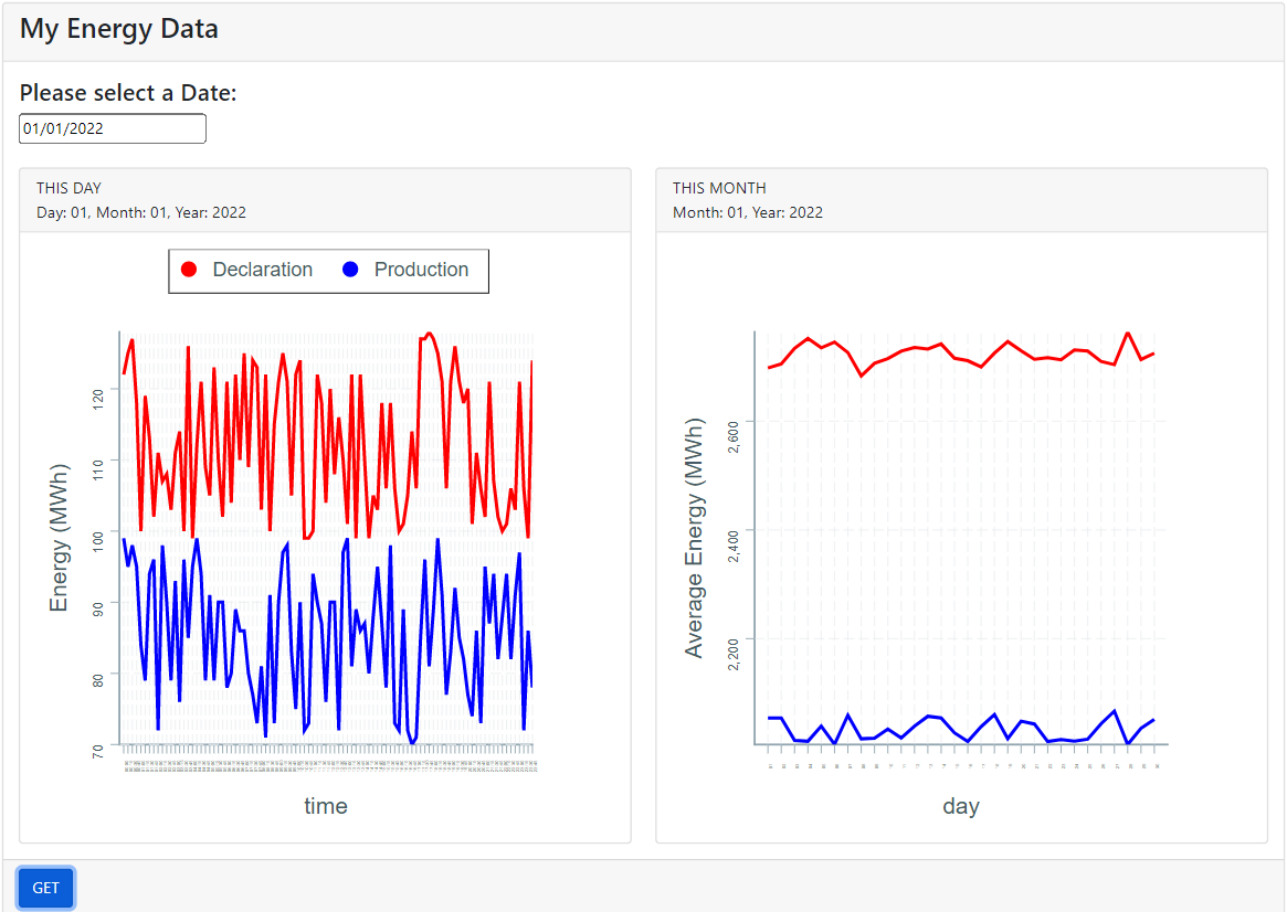


Figure 10: The user selects the date "01/01/2022" and presses the "GET" button.

The user selects the date "01/01/2022" and presses the "GET" button to visualize his data as shown in the image. The production forecast declaration data are displayed in red and the production data in blue. The left chart is the day chart with values every 15 minutes while the right chart is the month chart with values the sum of the values of each day.

To better understand the information system in the process of displaying energy data we created a sequence diagram using the UML modeling language. Notice how the objects correspond to the components of the decentralized application we developed in Chapter 5.

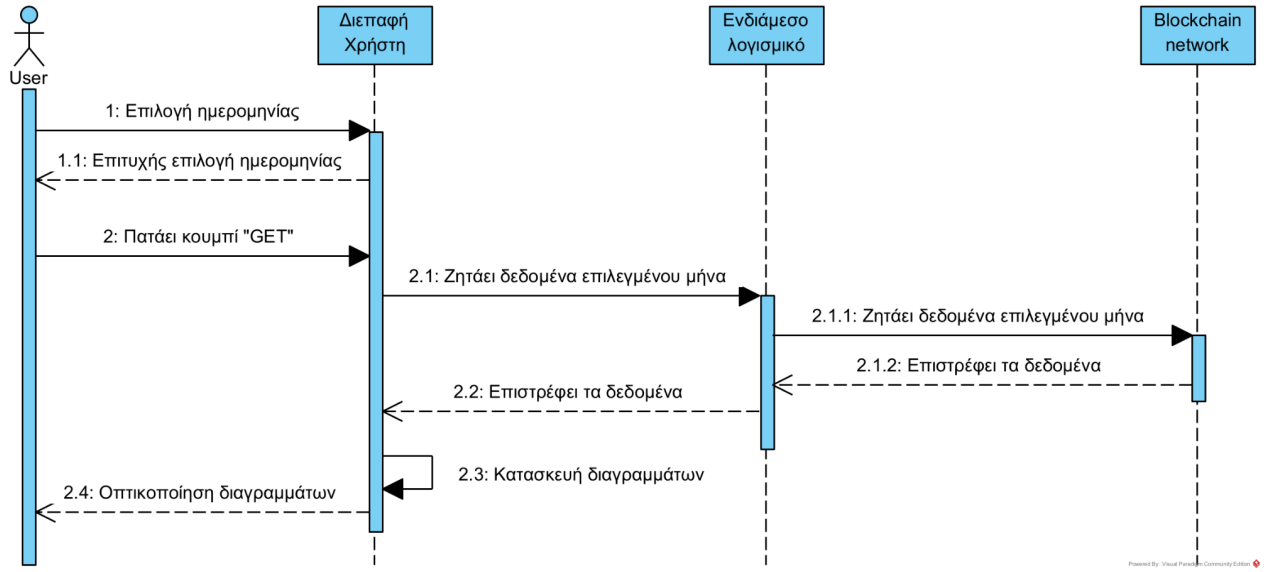


Figure 11: Energy data visualization UML sequence diagram.

6.2 Voting procedure

In this subsection we will present a voting process with a positive result. Producer Org1 starts the voting process. Then each organization votes in favor at a different time until a majority is gathered. The system automatically recalculates the production percentages of each organization based on the data entered in it. Now every producer sees the result of the vote as well as the updated percentages of its forecast and production in the navigation bar.

Step 1: The Org1 producer presses the "Initialize" button and the vote is initialized. The message with the voting status is displayed. Notice how the number of votes which are displayed:

- are positive "yes"
- are negative "no"
- have not been registered

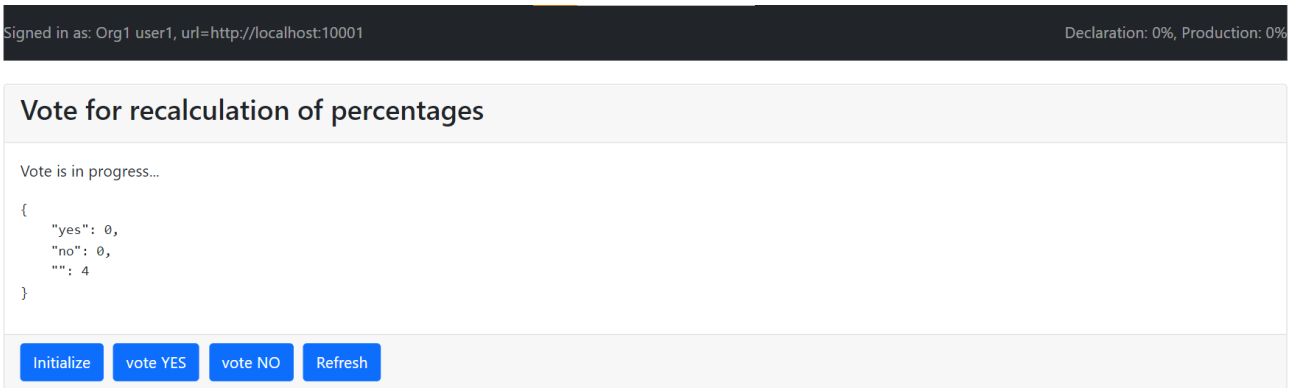


Figure 12: The Org1 producer presses the "Initialize" button and the vote is initialized.

To better understand the information system in the voting initialization process we created a sequence diagram using the UML modeling language. This diagram represents the functionality of the system each time a producer presses the "Initialize" button.

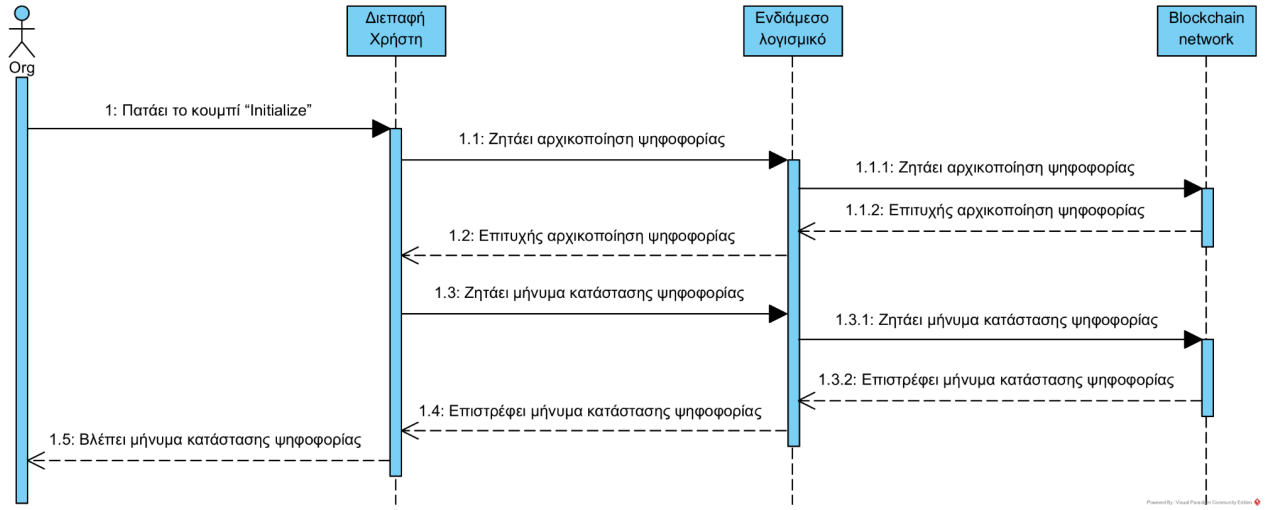


Figure 13: Voting initialization UML sequence diagram

Step 2: The Org1 producer presses the "vote YES" button and the voting continues. The voting status is updated.

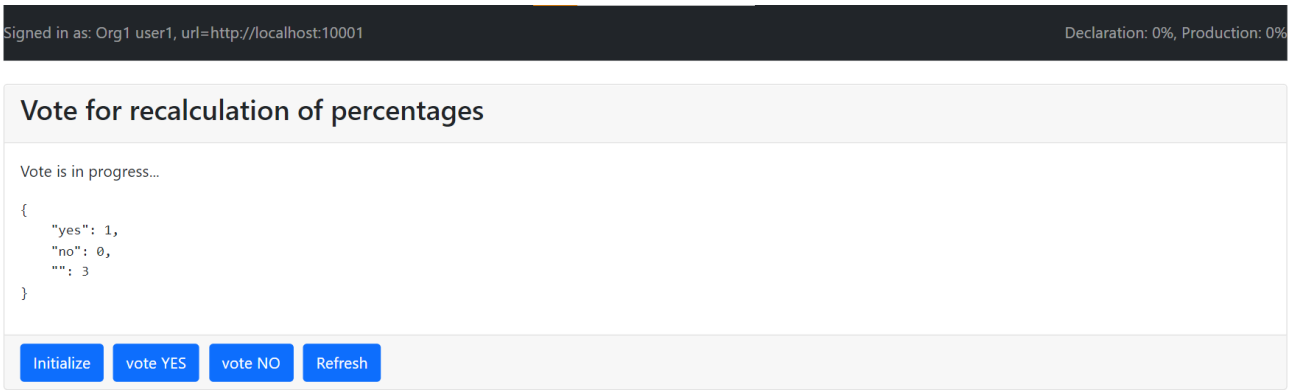


Figure 14: The producer Org1 presses the "vote YES" button and the voting continues.

To better understand the information system in the vote registration process we created a sequence diagram using the UML modeling language. This diagram represents the functionality of the system each time a producer presses the "Vote YES" button.

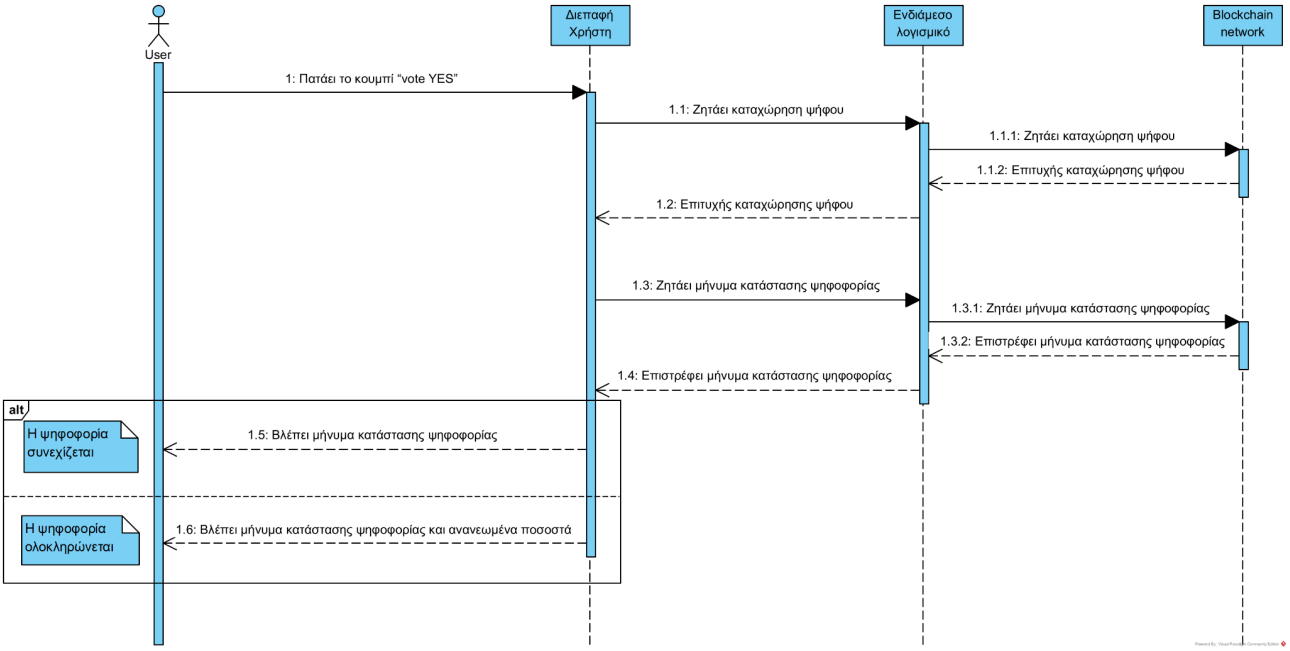


Figure 15: Vote registration UML sequence diagram.

Step 3: The Org2 producer presses the "vote YES" button and the voting continues. The voting status is updated.

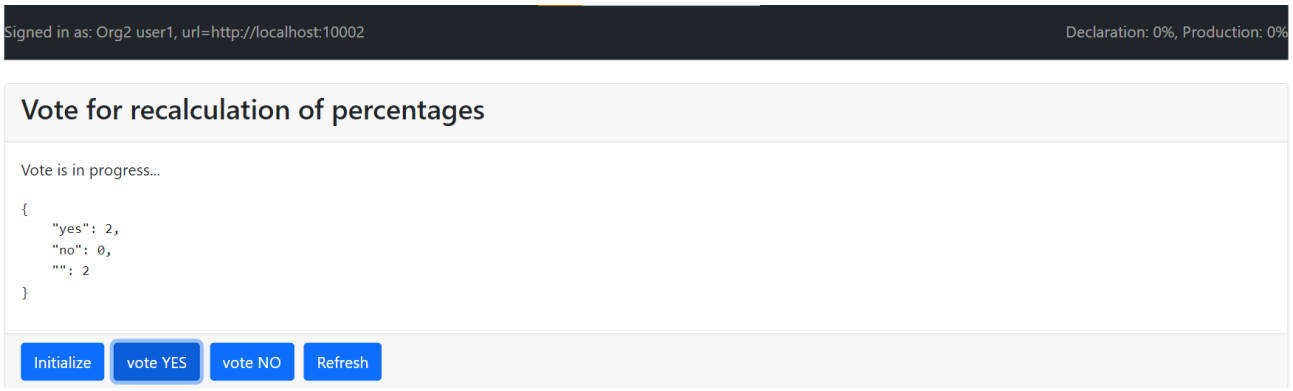


Figure 16: The producer Org2 presses the "vote YES" button and the voting continues.

Step 4: The Org3 producer presses the "vote YES" button and the voting is completed. Forecast and production percentages are updated. In the navigation bar the connected organization sees its updated percentages. The voting status message shows the updated percentages of all organizations.

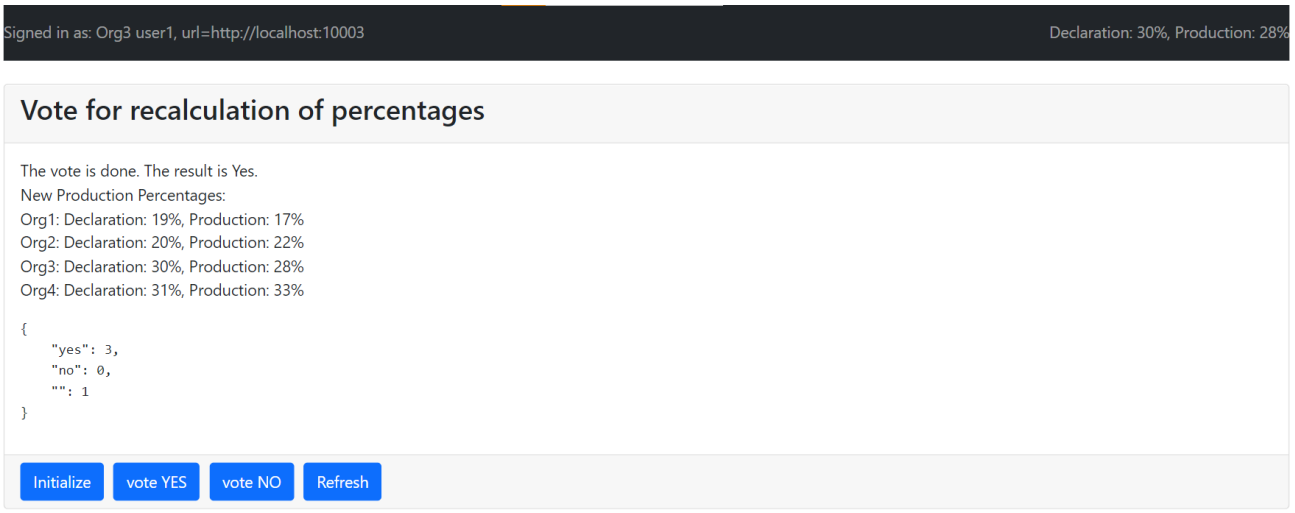


Figure 17: The producer Org3 presses the "vote YES" button and the voting is completed. Forecast and production rates are updated.

Step 5: The Org1 producer presses the "Refresh" button and sees the result of the vote. Forecast and production percentages are updated.

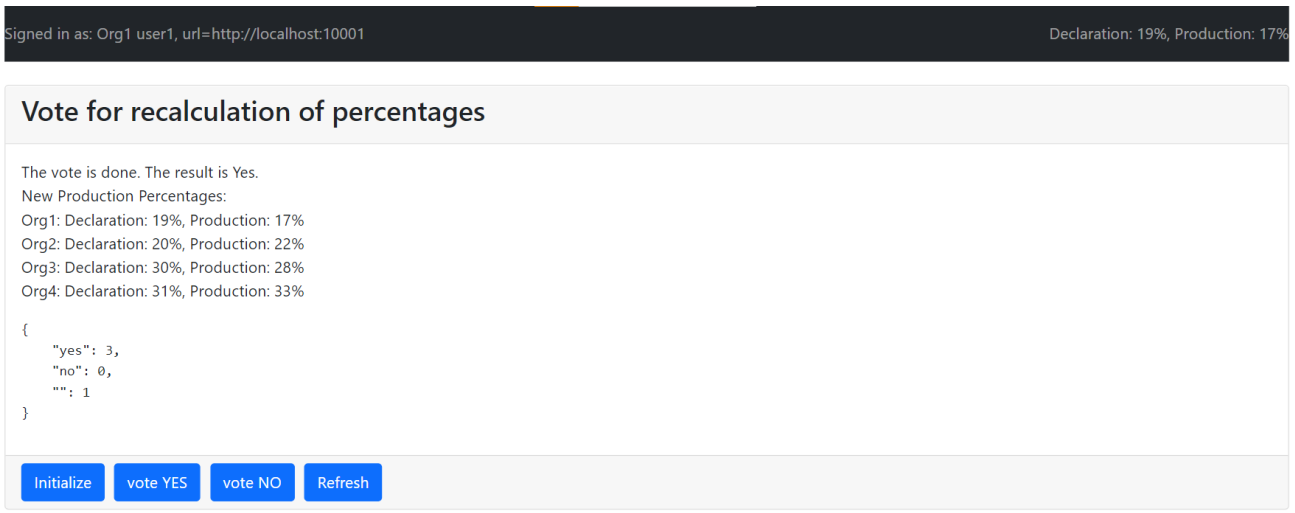


Figure 18: The producer Org1 presses the "Refresh" button and sees the result of the vote. Forecast and production rates are updated.

To better understand the information system in the voting renewal process we created a sequence diagram using the UML modeling language. This diagram represents the functionality of the system each time a manufacturer presses the "Refresh" button.

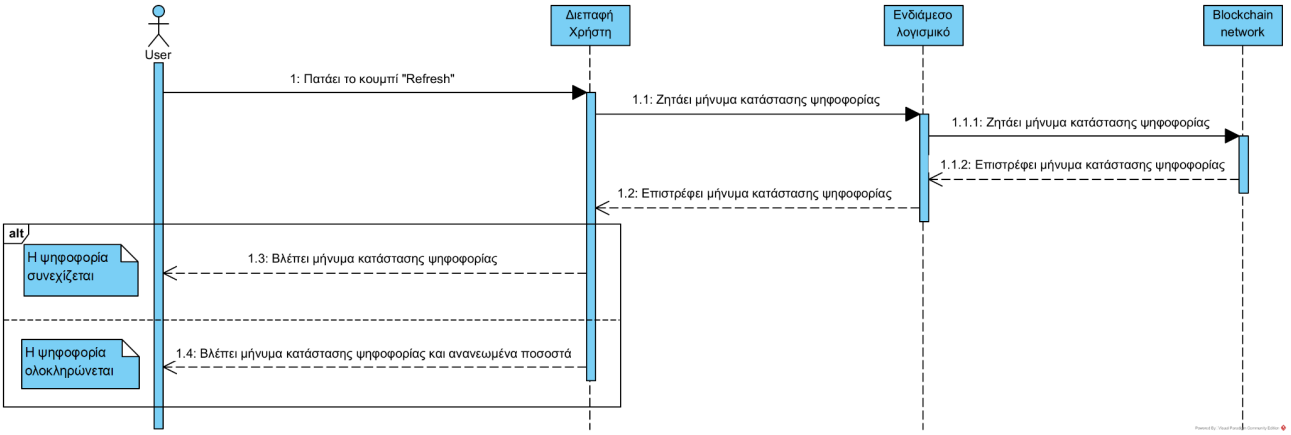


Figure 19: Vote renewal UML sequence diagram.

7 Comparison of Fabric with GoEthereum

In Chapter 5 we presented the implementation of a DAO using the Fabric tool in the case of electricity generation sharing between providers. The present thesis also has a fraternal collaborative dissertation, that of Sergios Exarchos of the Department of Digital Systems of the University of Piraeus [50]. Both theses use the same use case for the same purpose. This collaborative project used GoEthereum, also known as Geth, as an implementation tool [51]. We will now use this work by Sergios Exarchos to compare the Fabric tool with the GoEthereum tool based on the observations made during the implementation. The comparison will be divided into 3 sections: network observations, smart contract observations, and user interface observations.

Table 4: Comparison of Fabric with GoEthereum for the development of a DAO on a private network.

Sections	Παρατηρήσεις	Fabric	Geth
Network	Ease of starting and understanding		✔
	Ease of network creation		✔
	Ease of installing smart contracts on the network		✔
	Network flexibility and configuration	✔	
	Block recording rate (Throughput)	✔	
	Tutorials and documentation	✔	
Smart Contracts	Tutorials and documentation	✔	✔
	Ease of creation	✔	
	Smart contract capabilities	✔	
User Interface	Ease of connecting user interface to the network	✔	✔

7.1 Network observations

As we saw in a previous chapter the Fabric tool is complex and difficult to understand because it tries to offer many options to its administrators for any possible implementation of a private blockchain network. On the contrary, the Geth tool is simpler and easier to use because it aims to create decentralized applications and install them on the ethereum mainnet. In other words, a user of the Geth tool builds a decentralized application, tests it either on a local network or on one of the provided testnets, with the ultimate goal of installing it on the central public network. Creating private networks is available using Geth as a secondary function. The complexity difference becomes clear if one considers that when these two tools were designed, Geth's end user was an independent user while Fabric's were organizations. Based on the above, it becomes clear that the Geth tool is more limited in the possibilities it offers but at the same time it is easier for someone to start using it. Geth also makes it easier to set up a network and install smart contracts on it.

In Geth, once a smart contract is activated on the system, it cannot be modified, which is a problem in case of incorrect code writing or the need to update it. In Fabric, on the other hand, a smart contract can be modified and it is obvious that a Fabric network provides more flexibility and customization of smart contracts.

There is a lot of documentation on the official Fabric website about how a Fabric network works and how to use it, as well as various examples [52]. In contrast, in the corresponding Geth website the material available is unsatisfactory and needs enrichment [51]. This is a very important advantage of Fabric in relation to Geth since every new user has the necessary guidance available for the creation, configuration and maintenance of a private network.

Finally, we define throughput as the block register rate in the blockchain. Without performing any kind of benchmarking we noticed that the performance of the two tools is similar for small networks. But in theory, as the network escalates, Fabric is more efficient than Geth. This is achieved through the pioneering architectural execute-order-validate of Fabric that we explained in chapter 3.3.

7.2 Smart contracts observations

In Fabric, smart contracts are written in general-purpose programming languages such as Java, Go and Node.js, instead of DSL, such as Solidity used in Geth. This means that most companies already have all the skills needed to develop smart contracts and no additional training is required to learn a new language. Fabric also provides the "Contract API" which has all the necessary functions for writing smart contracts [41]. So Fabric provides more convenience in creating smart contracts as well as more features in smart contracts.

The official website of solidity provides a lot of educational material and examples regarding the writing of smart contracts [53]. Additionally, on the Ethereum platform, Truffle [54] or Remix [55] tools are often used to develop smart contracts and they also have the necessary support for developers. Fabric provides satisfactory Contract API documentation as well as many examples and training materials [39] regarding smart contracts. Therefore both Fabric and Geth provide satisfactory support when writing any smart contract implementation.

7.3 User interface observations

A user interface is required to develop a decentralized application. This interface is usually developed with a programming framework such as React.js used in this thesis. In the case of decentralized applications, we need a tool to give access to the smart contract functions of an active blockchain network. In Fabric this tool is the SDK [37] as mentioned in the previous chapter while in Geth this tool is Web3 [56]. Therefore both platforms provide an easy and very similar way to connect the interface to the network.

8 Conclusions and future directions

As we saw from the analysis of the Bitcoin protocol in the second chapter, a blockchain functions as a ledger, that is, as an unchanged register of transactions. A blockchain network or Distributed ledger technology (DLT) is a technology that allows participants to act as network members and maintain and update their own ledger so that their ledger status reflects the status of the other network ledgers.

The immutability of the blockchain in combination with the storage of the ledger by each member of the network lay the foundation for the creation of a distributed autonomous organization. A DAO is a digital organization that allows potentially unknown and untrustworthy members to work together to achieve a common goal. We saw in Chapters 4, 5 and 6 an implementation of such an organization with members of its 4 electricity producers aiming to share production without the use of a trusted third party. With this implementation we have shown that such an information system can offer confidence in strangers but also liberation from a potentially corrupt external organization.

In the present work, the Hyperledger Fabric distributed ledger technology is used, which is currently one of the most widespread and stable software systems for the construction of private permissioned blockchain networks. From the comparison of Fabric with the GoEthereum tool in the construction of a DAO in a private blockchain network, we concluded that Fabric is more suitable due to its flexibility, maintainability and scalability. It is worth noting that the weak point of Fabric is the steep learning curve that it requires from its managers and that is the main reason for the possible choice of GoEthereum at the moment.

The future use of this work is mainly the development of a DAO that provides the possibility of a digital board by providing an equal voting mechanism. In addition, the scope of its use increases as it can be used as a blueprint for the development of any potential DAO.

This paper can be a study and research manual to understand the concepts of blockchain, Hyperledger Fabric and DAO and to answer the questions "why blockchain, what is it and how does it work?", "Why choose the Fabric tool?" and "what is it, how is it implemented and what is the utility of a DAO?".

ABBREVIATIONS

DAO	Decentralized Autonomous Organization
B2B	Business to Business
ECDSA	Elliptic Curve Digital Signature Algorithm
SHA256	Secure Hash Algorithm
RIPEMD-160	RIPE Message Digest
UTXO	Unspent Transaction Output
BFT	Byzantine Fault Tolerance
PoW	Proof of Work
PoA	Proof of Authority
PoS	Proof of Stake
SCP	Stellar Consensus Protocol
FBA	Federated Byzantine Agreement
PoET	Proof of Elapsed Time
DApps	Decentralized Applications
YAC	Yet Another Consensus
IBFT	Istanbul Byzantine Fault Tolerance
DSL	Domain Specific Languages
CFT	Crash Fault Tolerance
RAFT	Reliable, Replicated, Redundant, And Fault-Tolerant
gRPC	Google Remote Procedure Call
UML	Unified Modeling Language
IoT	Internet of Things
DLT	Distributed Ledger Technology

REFERENCES

[1] Laurence, T. (2019). Blockchain for dummies. John Wiley & Sons.
[2] Haber, S., & Stornetta, WS (1990, August). How to time-stamp a digital document. In Conference on the Theory and Application of Cryptography (pp. 437-455). Springer, Berlin, Heidelberg.
[3] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 21260.
[4] Morabito, V. (2017). Business innovation through blockchain. Cham: Springer International Publishing.
[5] Johnson, D., Menezes, A., & Vanstone, S. (2001). The elliptic curve digital signature algorithm (ECDSA). International journal of information security, 1(1), 36-63.
[6] Naik, RP, & Courtois, NT (2013). Optimising the sha256 hashing algorithm for faster and more efficient bitcoin mining. MSc Information Security Department of Computer Science UCL, 1-65.
[7] Dobbertin, H., Bosselaers, A., & Preneel, B. (1996, February). RIPEMD-160: A strengthened version of RIPEMD. In International Workshop on Fast Software Encryption (pp. 71-82). Springer, Berlin, Heidelberg.
[8] Daulay, RSA, Nasution, SM, & Paryasto, MW (2017, November). Realization and addressing analysis in blockchain bitcoin. In IOP Conference Series: Materials Science and Engineering (Vol. 260, No. 1, p. 012002). IOP Publishing.
[9] Python Software Foundation (2022) random.py [Source code].
https://github.com/python/cpython/blob/3.10/Lib/random.py.
[10] Oracle (2022) SecureRandom.java [online].
https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html.
[11] Ethereum, “Decentralized autonomous organizations”, www.ethereum.org [Online]. Available: https://ethereum.org/en/dao . [Accessed Feb. 10, 2022]
[12] D. Geroni, “Top 5 Blockchain Security Issues In 2021”, 101blockchains.com, June 13, 2021. [Online]. Available: https://101blockchains.com/blockchain-security-issues. [Accessed Feb. 08, 2022].
[13] Ethereum Foundation (2022) Ethereum [Online]. https://ethereum.org.
[14] Stellar Development Foundation (2022) Stellar [Online]. https://www.stellar.org.
[15] R3 (2022) R3 Corda [Online] https://www.r3.com/corda-platform.
[16] CONSENSYS (2022) Quorum [Online] https://consensys.net/quorum.
[17] Ripple (2022) Ripple [Online] https://ripple.com.
[18] S. Gupta “Hyping the hyperledger with blockchain boffin Brian Behlendorf”, www.hfsresearch.com [Online]. Available: https://www.hfsresearch.com/outsourcing-heros/brian-behlendorf-interview_082417/ . [Accessed Feb. 09, 2022]
[19] The Linux Foundation (2022) Hyperledger Foundation [Online]. https://www.hyperledger.org .
[20] K. Kotoski, "NBC signs blockchain agreement", Phnom Penh Post, April 24, 2017.[Online], Available:https://www.phnompenhpost.com/business/nbc-signs-blockchain-agreement .[Accessed Feb 9, 2022].
[21] Kapritsos, M., Wang, Y., Quema, V., Clement, A., Alvisi, L., & Dahlin, M. (2012). All about Eve:{Execute-Verify} Replication for {Multi-Core} Servers. In 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12) (pp. 237-250).
[22] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Yellick, J. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the thirteenth EuroSys conference (pp. 1-15).
[23] gRPC Authors (2022) gRPC [Online]. https://grpc.io/.
[24] K. Croman, C. Decker, I. Eyal, AE Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, EG Sirer, et al. On scaling decentralized blockchains. In International Conference on Financial Cryptography and Data Security (FC), pages 106–125. Springer, 2016.
[25] M. Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In International Workshop on Open Problems in Network Security (iNetSec), pages 112–125, 2015.
[26] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In ACM Symposium on Principles of Distributed Computing (PODC), pages 1–12. ACM, 1987.
[27] Suryandaru Triandana (2012) LevelDB in Go [Source Code]. https://github.com/syndtr/goleveldb/
[28] The Apache Software Foundation (2021) Apache CouchDB. http://couchdb.apache.org

[29] Law 4414/2016, “New scheme for support of electricity stations from Renewable Energy Sources and Cogeneration of High Efficiency Electricity and Heat - Provisions for the legal and operational separation of the supply and distribution sectors in the gas market and other provisions.(translated)”, Goverment Newspaper 149/A/9-8-2016, [Online], Available: https://www.e-nomothesia.gr/energeia/nomos-4414-2016.html , [Accessed Feb. 11, 2022]
[30] H. Floydopoylos, “What are the electricity aggregators in which Terna and Mytilineos invest (translated)”, www.energypress.gr, March 18, 2019 [Online], Available: https://energypress.gr/news/h-floydopoylos-ti-einai-oi-fose-stoys-opoioys-ependyoyn-terna-kai-mytilinaios , [Accessed Feb. 11, 2022]
[31] Protergia, “Electricity aggregators (translated)” www.protergia.gr , [Online], Available: https://www.protergia.gr/el/foreas-swreytikhs-ekproswphshs , [Accessed Feb. 11, 2022]
[32] Forena, “Why is the choice of electricity aggregators that will represent them in the target model crucial for RES producers (translated)?”, www.forenaenergy.gr, [Online], Available: https://www.forenaenergy.gr/el/2319-2/ , [Accessed Feb. 11, 2022]
[33] Everoze “AGGREGATORS: WHO ARE YOU?”, www.everoze.com, February 2018, [Online], Available: https://everoze.com/aggregators-who-are-you/ , [Accessed Feb. 11, 2022]
[34] J. Collings, M. Honnibal, P. Vanderwerff (2022) react-bootstrap [Online]. https://react-bootstrap.github.io/
[35] D. Starantzis (2022) react-app [Source Code]. https://github.com/jimwheaty/blockchainDAO/tree/main/react-app
[36] OpenJS Foundation (2017) Express [Online]. https://expressjs.com
[37] Hyperledger Foundation (2022) Hyperledger Fabric SDK for Node.js [Online]. https://hyperledger.github.io/fabric-sdk-node/release-2.2/
[38] D. Starantzis (2022) app-server-javascript [Source Code]. https://github.com/jimwheaty/blockchainDAO/tree/main/app-server-javascript
[39] Hyperledger Foundation (2022) fabric-samples [Source Code]. https://github.com/hyperledger/fabric-samples/tree/release-2.2
[40] OpenJS Foundation (2022) Node.js [Online]. https://nodejs.org/en/
[41] Hyperledger Foundation (2022) fabric-contract-api [Online]. https://hyperledger.github.io/fabric-chaincode-node/release-2.2/api
[42] D. Starantzis (2022) chaincode-energyDAO [Source Code]. https://github.com/jimwheaty/blockchainDAO/tree/main/chaincode-energyDAO
[43] Hyperledger Foundation (2022) fabric-peer [Online]. https://hub.docker.com/r/hyperledger/fabric-peer
[44] Hyperledger Foundation (2022) fabric-orderer [Online]. https://hub.docker.com/r/hyperledger/fabric-orderer
[45] Hyperledger Foundation (2022) fabric-ca [Online]. https://hub.docker.com/r/hyperledger/fabric-ca
[46] Hyperledger Foundation (2022) fabric-nodeenv [Online]. https://hub.docker.com/r/hyperledger/fabric-nodeenv
[47] Hyperledger Foundation (2022) fabric-tools [Online]. https://hub.docker.com/r/hyperledger/fabric-tools
[48] D. Starantzis (2022) blockchainDAO [Source Code]. https://github.com/jimwheaty/blockchainDAO
[49] D. Starantzis (2022) simulation [Source Code]. https://github.com/jimwheaty/blockchainDAO/tree/main/simulation
[50] S. Exarchos, "Software architectures for the implementation of distributed autonomous organizations (DAO) with blockchain technologies.", Thesis, School of Information Technology and Telecommunications, Department of Digital Systems, University of Piraeus, 2022.
[51] go-ethereum authors (2022) GoEthereum [Online]. https://geth.ethereum.org
[52] Hyperledger Foundation, “A Blockchain Platform for the Enterprise,” hyperledger-fabric.readthedocs.io, 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2 . [Accessed Feb. 10, 2022].
[53] Ethereum (2022) Solidity [Online]. https://docs.soliditylang.org/en/v0.8.11/
[54] ConsenSys Software Inc (2022) truffle suite [Online]. https://trufflesuite.com/docs/index.html
[55] Remix (2021) Remix [Online] https://remix.ethereum.org/
[56] Web3 (2022) web3.js [Source Code] https://github.com/ChainSafe/web3.js