Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων & Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εφαρμογή Αποκεντρωμένης Αποθήκευσης Δεδομένων σε Ethereum Blockchain

Διπλωματικη Εργασια

του

Χρήστου Ι. Αναγνώστου

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής ΕΜΠ

Αθήνα, Μάρτιος 2020

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων & Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εφαρμογή Αποκεντρωμένης Αποθήκευσης Δεδομένων σε Ethereum Blockchain

Διπλωματικη Εργασια

του

Χρήστου Ι. Αναγνώστου

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την κάτωθι τριμελή επιτροπή την 10η Μαρτίου 2020.

Νεκτάριος Κοζύρης	Γεώργιος Γκούμας	Δημήτριος Τσουμάκος
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Αν/τής Καθηγητής Ιονίου Παν/μίου

Χρήστος Ι. Αναγνώστου
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

Περίληψη

Η ανατρεπτική τεχνολογία του Blockchain, έχει αρχίσει, πλέον, να αποκολλάται από την αρχική του υλοποίηση (τα κρυπτονομίσματα), και αρχίζει να αλλάζει για τα καλά τον τρόπο με τον οποίο οι άνθρωποι αντιλαμβάνονται και χρησιμοποιούν το διαδίκτυο.

Το Blockchain είναι ένας ψηφιακός κατανεμημένος και δημόσιος κατάλογος (ledger) στον οποίο καταγράφονται συναλλαγές και συμφωνίες με τρόπο αδιάβλητο και υποστηρίζεται από ένα δίκτυο ομότιμων κόμβων. Αυτό, έχει οδηγήσει στην ανάπτυξη αποκεντρωμένων εφαρμογών (DApps), οι οποίες δίνουν δυνατότητα στους συμμετέχοντες σε ένα Blockchain να αλληλεπιδρούν, χωρίς να υπάρχει ανάγκη για εμπιστοσύνη μεταξύ τους, αφού αυτή διασφαλίζεται από την ίδια την τεχνολογία. Η τελευταία, βασίζεται σε ένα δίκτυο ομότιμων κόμβων και θωρακίζεται με τεχνικές consensus μεταξύ των κόμβων, ώστε να διασφαλίζεται η ορθότητα και εγκυρότητα των δεδομένων που αποθηκεύονται στην αλυσίδα. Μία από τις πιο διαδεδομένες και ισχυρές, όσον αφορά τις δυνατότητες της, υλοποιήσεις Blockchain, είναι το Ethereum.

Για τις ανάγκες της παρούσας διπλωματικής, αναπτύχθηκε, πάνω στο Ethereum, μία αποκεντρωμένη εφαρμογή, που δίνει στους χρήστες της τη δυνατότητα να προσφέρουν και να αγοράζουν αποθηκευτικό χώρο, χρησιμοποιώντας Ether, το κρυπτονόμισμα του Ethereum.

Στην παρούσα εργασία, θα γίνει εκτενής αναφορά στα θεωρητικά θεμέλια τα οποία χρειάζονται για την ανάπτυξη της προαναφερθείσας εφαρμογής, καθώς και μια παρουσίαση της εφαρμογής καθεαυτής, με περιγραφή του τρόπου με τον οποίο αναπτύχθηκε, αλλά και κάποιων από τα βασικότερα use-cases.

Λέξεις-κλειδιά:Blockchain, Ethereum, Web3.0, Peer to Peer Storage Sharing, Solidity, Αποκεντρωμένη Εφαρμογή, Έξυπνα Συμβόλαια

Abstract

The disruptive technology of Blockchain, has now started to widen its scope from its initial form -which was a simple ledger, to store cryptocurrency transactions- into a powerful tool, to help people change the way they understand and interact with the Internet.

Blockchain is a digitalised distributed and public ledger, based on a peer to peer network of nodes, that immutably stores data, transactions and agreements. It has led to the development of Decentralised Applications (DApps), which ensure a trustful interaction between the participants of the Network, without them having to actually trust one another. Blockchain technology is based on this peer to peer network, and uses specific consensus algorithms to ensure the validity and immutability of the data stored on the chain. One of the most widely adopted and robust implementations of Blockchain is Ethereum.

As part of this diploma thesis, a decentralised application for sharing storage has been developed, using Ether, the cryptocurrency of Ethereum. Throughout the thesis, you may find thorough explanation of all the theoretical tools needed for developing an application of this kind, a description of the way it was developed, and lastly some of its most defining use-cases.

Keywords: Blockchain, Ethereum, Web3.0, Peer to Peer Storage Sharing, Solidity, Decentralised Application, Smart Contracts

Ευχαριστίες

Θα ήθελα να ευχαριστήσω, πρωτίστως, τους γονείς μου, που έχουν σταθεί δίπλα μου και είναι εκείνοι που αξίζουν κάθε έπαινο για οτιδήποτε έχω καταφέρει στη μέχρι τώρα ακαδημαϊκή, και όχι μόνο, πορεία μου.

Θα ήθελα, επίσης, να ευχαριστήσω όλους εκείνους τους φίλους και συμφοιτητές, που έκαναν την ακαδημαϊκή μου πορεία πιο ευχάριστη και εύκολη.

Τέλος, ευχαριστώ θερμά το εκπαιδευτικό και υποστηρικτικό προσωπικό του Εθνικού Μετσοβίου Πολυτεχνείου, και ειδκότερα της σχολής Ηλεκτρολόγων, για την απρόσκοπτη συνεργασία αλλά και τις σημαντικές γνώσεις που απλόχερα μου προσέφεραν από το 2013 μέχρι και σήμερα.

Χρήστος Ι. Αναγνώστου
Ιωάννινα, Μάρτιος 2020

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο

Έχει αρχίσει να γίνεται πλέον κατανοητό, ακόμα και για κάποιον ξένο με την Επιστήμη της Πληροφορίας, ότι τα προσωπικά δεδομένα και η προάσπισή τους, αποτελούν μία από τις πλέον σημαντικές υποχρεώσεις ενός σύγχρονου ανθρώπου.

Παρ΄ όλα αυτά, γίνεται όλο και πιο σαφές ότι, με τη σημερινή του μορφή, το διαδίκτυο δεν μπορεί σε καμία περίπτωση να εγγυηθεί την απόλυτη ασφάλεια των προσωπικών δεδομένων των χρηστών του. Ακόμα και σήμερα, η πιο διαδεδομένη αρχιτεκτονική αποθήκευσης δεδομένων βασίζεται στο μοντέλο stand-alone μηχανημάτων, με τη διαχείριση και αποθήκευση των δεδομένων να γίνεται κεντρικά από έναν (ή περισσότερους) εξυπηρετητή (server) για να σταλούν σε έναν (ή περισσότερους) πελάτη (client). Κάθε φορά που αυτό συμβαίνει, ο χρήστης χάνει οποιονδήποτε έλεγχο πάνω στα δεδομένα του. Το παραπάνω είναι ακόμα πιο παράλογο, αν σκεφτεί κανείς πως πλέον αν και ζούμε σε ένα κόσμο στον οποίο σχεδόν κάθε συσκευή είναι συνδεδεμένη στο διαδίκτυο (Η/Υ, κινητό, ρολόι, αυτοκίνητο κ.ά.), συνεχίζουμε να αποθηκεύουμε τα δεδομένα μας κατά κύριο λόγο κεντρικά, είτε στα προσωπικά μας μηχανήματα είτε σε δομές cloud, αγνοώντας τους κινδύνους που αυτή μας η επιλογή επιφυλάσσει.

Τα παραπάνω σίγουρα εγείρουν ζητήματα εμπιστοσύνης μεταξύ των ιδιοκτητών των δεδομένων και εκείνων που τα διαχειρίζονται. Η πιθανότητα, εσκεμμένης ή μη, αλλοίωσης ή υποκλοπής των δεδομένων που αποθηκεύουμε αλλά και επικοινωνούμε μέσω του διαδικτύου, είναι σημαντικά μεγάλη όσο η παρούσα κεντροποιημένη server-client αρχιτεκτονική επιμένει να χρησιμοποιείται.

Η αποθήκευση δεδομένων στο cloud είναι μια εξαιρετικά διαδεδομένη τακτική τα τελευταία χρόνια με πολλούς παρόχους (Google, Amazon, Microsoft, MEGA Upload κ.ά) να προσφέρουν αποθηκευτικό χώρο σε πελάτες τους αντί κάποιας χρηματικής αποζημίωσης, η οποία είναι συγκεκριμένη και μη συζητήσιμη. Τα δεδομένα του χρήστη αποθηκεύονται κεντρικά (συνήθως με τεχνικές κατανεμημένων συστημάτων) στις φάρμες φυσικών μηχανημάτων του παρόχου (server farms) και είναι προσβάσιμα από κάθε συσκευή με σύνδεση

στο διαδίκτυο. Οι κίνδυνοι στους οποίους υπόκεινται τα συγκεκριμένα δεδομένα, λόγω ακριβώς του κεντροποιημένου τρόπου με τον οποίο αποθηκεύονται είναι προφανείς με βάση και τις προηγούμενες διαπιστώσεις. Σε αυτό το πρόβλημα, έρχεται να δώσει λύση μια νέα τεχνολογία, το Blockchain και ο επαναστατικός τρόπος με τον οποίο διαχειρίζεται τα δεδομένα και τις συναλλαγές μεταξύ των χρηστών του.

1.2 Σκοπός της διπλωματικής

Σκοπός της παρούσας διπλωματικής εργασίας, είναι η ανάπτυξη μιας αποκεντροποιημένης (decentralised) εφαρμογής για την επίλυση των προβλημάτων των οποίων αναφέρονται παραπάνω και ειδικότερα της αγοράς χώρου για την αποθήκευση δεδομένων. Για το λόγο αυτό, αναπτύχθηκε εφαρμογή, η οποία παρέχει σε χρήστες τη δυνατότητα να βρίσκουν παρόχους αποθηκευτικού χώρου σε επίπεδο peer to peer, χωρίς δηλαδή την ανάμιξη κάποιου τρίτου (μεσάζοντα), βάζοντας, ταυτόχρονα, τους παρόχους σε διαδικασία δημοπρασίας όσον αφορά την τιμή για την οποία θα προσφέρουν το χώρο αυτό.

Για τη διασφάλιση της εμπιστοσύνης μεταξύ των χρηστών (παρόχων και πελατών) της εφαρμογής, χρησιμοποιείται το Ethereum Blockchain αλλά και η τεχνική του Proof of Retrievability.

1.3 Οργάνωση του τόμου

Ο τόμος οργανώνεται σε 6 κεφάλαια.

Στα Κεφάλαια 2 και 3, αναλύονται τόσο το θεωρητικό υπόβαθρο που χρειάστηκε καθώς και όσο και οι τεχνολογίες που χρησιμοποιήθηκαν για την εκπόνηση της εργασίας. Στο Κεφάλαιο 4, αναλύονται οι οντότητες της εφαρμογής, καθώς και οι τρόποι με τους οποίους ο χρήστης μπορεί να αλληλεπιδράσει με αυτές μέσω της διεπαφής χρήστη που επίσης έχει αναπτυχθεί. Στο Κεφάλαιο 5, επεξηγούνται διάφορες περιπτώσεις χρήσης της εφαρμογής και συνοδεύονται από use case diagrams για την καθεμία. Τέλος, στο Κεφάλαιο 6, παρουσιάζονται τα συμπεράσματα που προκύπτουν από την ανάπτυξη της εφαρμογής, καθώς και μελλοντικές πιθανές επεκτάσεις της.

Κεφάλαιο 2

Blockchain, Web3 και Proof of Retrievability

2.1 Εισαγωγή

Στο κεφάλαιο αυτό, διατυπώνεται το θεωρητικό υπόβαθρο των τριών βασικών τεχνολογιών που χρησιμοποιήθηκαν για τη δημιουργία της αποκεντρωμένης εφαρμογής (decentralized app ή DApp) της εργασίας. Η υλοποίηση του Blockchain με την οποία θα ασχοληθεί αυτή η διπλωματική είναι το Ethereum, ενώ επεξηγείται η έννοια του Web3 αλλά και της τεχνικής του Proof of Retrievability.

2.2 Blockchain

Αναφορά στο Blockchain ως ιδέα (χωρίς να αναφερθεί το ακριβές λεκτικό) για πρώτη φορά, γίνεται σε ένα whitepaper[1] που δημοσιεύτηκε το 2008 από συγγραφέα (ή και ομάδα συγγραφέων) με το ψευδώνυμο Satoshi Nakatomo[2].

Το Blockchain είναι ένας αυξανόμενος κατάλογος συναλλαγών(transactions) ή εναλλακτικά ένα "ημερολόγιο" ηλεκτρονικών ενεργειών, που ονομάζονται blocks, και συνδέονται χρησιμοποιώντας τεχνικές κρυπτογραφίας. Κάθε block περιέχει μια κρυπτογραφημένη "υπογραφή" του προηγούμενου block, ένα χρονικό σήμα (timestamp) και δεδομένα συναλλαγών ή κατέπέκταση ενεργειών. Μια συναλλαγή μπορεί να είναι μια μεταφορά χρημάτων, μια κλήση σε κάποιο έξυπνο συμβόλαιο κ.ά.

Ένα από τα βασικά χαρακτηριστικά του Blockchain είναι ότι αποτρέπει την τροποποίηση των ήδη αποθηκευμένων πληροφοριών (immutability). Ουσιαστικά αποτελεί ένα δημόσιο και κατανεμημένο -στους συμμετέχοντες του δικτύου που δημιουργείται από τους χρήστες του- κατάλογο συναλλαγών, που μπορεί να καταγράφει τις συναλλαγές μεταξύ δύο ή περισσότερων μερών αποτελεσματικά και κατά τρόπο επαληθεύσιμο και μόνιμο. Όλοι οι συμμετέχοντες σε ένα blockchain δημιουργούν ένα δίκτυο peer-to-peer συλλογικά, ακολουθώντας ένα πρωτόκολλο για επικοινωνία μεταξύ κόμβων και καθολικό πρωτόκολλο

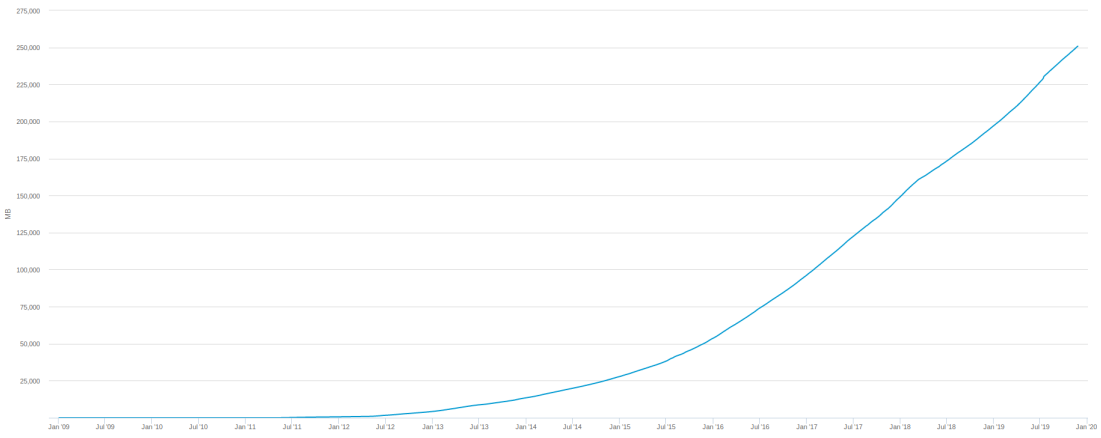
για την επικύρωση νέων blocks. Μόλις καταγραφούν, τα δεδομένα σε οποιοδήποτε block δεν μπορούν να τροποποιηθούν, χωρίς να αλλοιωθούν τα περιεχόμενα όλων των επόμενων block, πράγμα που απαιτεί συναινετική πλειοψηφία του δικτύου. Παρόλο που τα αρχεία των block της αλυσίδας δεν είναι αναλλοίωτα, το Blockcain μπορεί να θεωρηθεί ασφαλές από το σχεδιασμό του και να αποτελέσει παράδειγμα ενός κατανεμημένου συστήματος υπολογιστών με υψηλή ανοχή σε λάθη (συγκεκριμένα στο Ethereum Blockchain χρησιμοποιείται το Byzantine Fault tolerance). Επομένως, η αποκεντρωμένη συναίνεση (Decentralized consensus) απαιτείται και είναι απαραίτητη στην υλοποίηση ενός Blockcain.

Η πρώτη εφαρμογή του Blockcain και ο λόγος για τον οποίο ο Satoshi Nakatomo δημοσίευσε το whitepaper του το 2008, είναι το Bitcoin. Η υλοποίηση του Blockcain για το Bitcoin το κατέστησε το πρώτο ψηφιακό νόμισμα για την επίλυση του προβλήματος των διπλών δαπανών [3] (double-spending problem) χωρίς την ανάγκη μιας αξιόπιστης αρχής ή κεντρικού διακομιστή. Ο σχεδιασμός του Bitcoin έχει εμπνεύσει άλλες εφαρμογές, και υλοποιήσεις Blockcain που είναι προσβάσιμες από κάθε χρήστη του διαδικτύου και χρησιμοποιούνται για την δημιουργία κρυπτονομισμάτων (cryptocurrencies). Ιδιωτικά Blockchains έχουν προταθεί για εταιρική χρήση.

Το 1991 ξεκίνησε η εργασία για μια κρυπτογραφημένη, ασφαλή αλυσίδα απο μπλοκ με στόχο τη δημιουργία ενός συστήματος εγγράφων των οποίων η χρονοσήμανση (timestamp) να μην μπορεί να αλλοιωθεί[4]. Το 1993 οι Bayer, Haber και Stornetta συμπεριέλαβαν τα Merkle Trees[5] στο σχεδιασμό, το οποίο οδήγησε στην αύξηση της απόδοσης, αφού επέτρεπε την ενσωμάτωση πολλαπλών πιστοποιητικών των εγγράφων σε ένα μεμονωμένο block.

Ο Satoshi δουλεύοντας στον μέχρι τότε σχεδιασμό, και με την παραδοχή πως δεν ήταν απαραίτητο για κάθε block που προστίθεται στην αλυσίδα να "υπογράφεται" από μία αξιόπιστη, κεντρική, αρχή, κατέληξε στη δημιουργία του Bitcoin. Το Bitcoin πήρε μεγάλη δημοσιότητα και αυτό είναι εμφανές από τον όγκο του Blockchain[24], το οποίο μέχρι τον Αύγουστο του 2014 είχε φτάσει τα 20Gigabytes, ενώ τον Ιανουάριο του 2015 ήταν κατα προσέγγιση 30Gigabytes. Μέχρι το Νοέμβριο του 2019 και λόγω της δημοσιότητας (αλλά και της αύξησης της αξίας) που είχε λάβει το συγκεκριμένο κρυπτονόμισμα ο όγκος των συναλλαγών που ήταν αποθηκευμένες στο Blockchain είχε φτάσει τα 247 Gigabytes. Στο σχήμα 2.1 φαίνεται ο όγκος που καταλαμβάνει η αλυσίδα σε Megabytes από τη δημιουργία της το 2009 μέχρι και το Νοέμβριο του 2019. Το γράφημα αποτυπώνει όχι μόνο την αύξηση της χρήσης του συγκεκριμένου κρυπτονομίσματος, αλλά εκδηλώνει τη γενικότερη τάση για τη χρήση της τεχνολογίας αυτής για εμπορικούς σκοπούς. Το Bitcoin αποτέλεσε το εναρκτήριο λάκτισμα για τη διάδοση της ιδέας και δημιουργία πολλών κρυπτονομισμάτων και εφαρμογών που έχουν βάση το Blockchain.

Μία από τις πιο διαδεδομένες υλοποιήσεις της τεχνολογίας του Blockchain, είναι το Ethereum. Τα βασικά γνωρίσματα ενός Blockchain, και ο τρόπος με τον οποίο αυτά υλοποιούνται στο Ethereum, είναι τα παρακάτω[6]:



Σχήμα 2.1: Ρυθμός Αύξησης Αλυσίδας Bitcoin σε Megabytes

Block

Δομική μονάδα του Blockchain είναι το block. Το πρώτο block ενός Blockchain αποκαλείται Genesis Block.Ένα block περιέχει ομάδες από ελεγμένες συναλλαγές οι οποίες είναι κατακερματισμένες και κωδικοποιημένες με χρήση ασύμμετρης κρυπτογραφίας. Με δεδομένο ότι κάθε block συνδέεται με το χρονικά προηγούμενο, σε κάθε block υπάρχει το κρυπτογραφημένο hash του προηγούμενου. Αυτή η σύνδεση σχηματίζει την αδιάσπαστη αλληλουχία που ονομάζουμε αλυσίδα.

Consensus

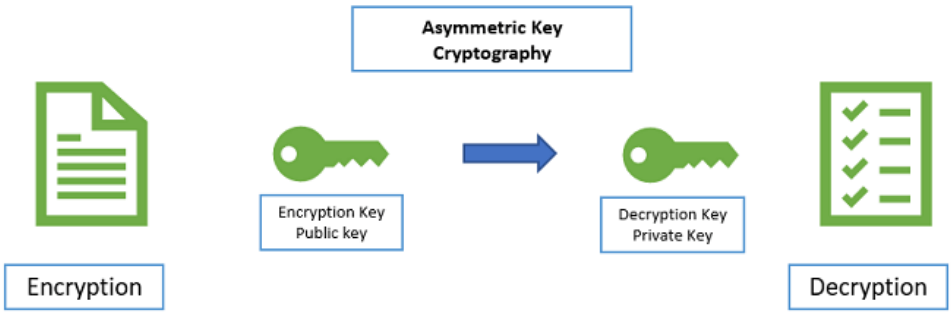
Για να μπορέσει ένα block να προστεθεί στην αλυσίδα και να θεωρηθεί αποδεκτό, θα πρέπει να γίνει αποδεκτό από το δίκτυο των χρηστών μέσω του Consensus. Το Ethereum (όπως και το Bitcoin) έχει επιλέξει ως τεχνική για το consensus, το Proof of Work, ή PoW. Για να μπορέσει, λοιπόν, ένας χρήστης να προσθέσει ένα block στην αλυσίδα, θα πρέπει να λύσει έναν κρυπτογραφικό ¨γρίφο¨, προσθέτοντας στο τέλος των δεδομένων του block τυχαίους χαρακτήρες (nonce), οι οποίοι θα κάνουν το hash του να πάρει μια συγκεκριμένη μορφή. Η επίλυση αυτού του γρίφου είναι πολύ απαιτητική όσον αφορά την υπολογιστική ισχύ. Κάθε χρήστης και συμμετέχων στο δίκτυο, δεν είναι υποχρεωμένος να συμμετέχει στη διαδικασία, αλλά αυτοί που το κάνουν, οι λεγόμενοι miners, επιβραβεύονται για κάθε ορθή λύση με λίγο Ether, που είναι το κρυπτονόμισμα του Ethereum.

Λόγω, όμως, του ότι η διαδικασία του mining απαιτεί τόσο μεγάλη υπολογιστική ισχύ, το PoW γίνεται αργό και ασύμφορο σαν consensus protocol. Για το λόγο αυτό, το Ethereum έχει ανακοινώσει ότι σύντομα θα αλλάξει το consensus του σε Proof of Stake ή PoS. Στο PoS, δε χρειάζεται η επίλυση κάποιου γρίφου για να θεωρηθεί ένα block αληθές. Σε αυτό τον τύπο consensus (στον οποίο συμμετέχουν, κυκλικά, όλοι οι χρήστες του δικτύου) κερδίζει το block εκείνου με το μεγαλύτερο stake. Το PoS, λοιπόν, κάνει

τη διαδικασία πιο γρήγορη, αλλά και πιο ασφαλή, αφού κάποιος με μεγάλο stake (δηλαδή το μεγαλύτερο ποσό από ether στην κατοχή του) δε θα έχει κανένα συμφέρον να κάνει κάποια κακόβουλη ενέργεια στην αλυσίδα, κάτι που θα έχει ως άμεση συνέπεια τη μείωση της αξίας των κρυπτονομισμάτων που έχει στην κατοχή του.

Cryptography

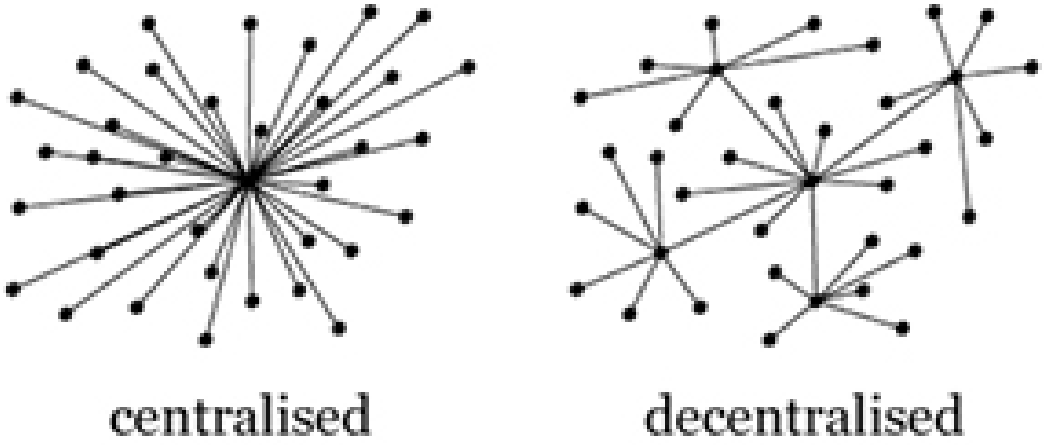
Λόγω του ότι κάθε δημόσια υλοποίηση Blockchain είναι εξ ορισμού προσβάσιμη από τον καθένα με πρόσβαση στο διαδίκτυο, χρησιμοποιείται η κρυπτογραφία σαν δικλείδα ασφάλειας των δεδομένων που βρίσκονται σε αυτή. Σε αυτές τις μεθόδους περιλαμβάνεται η μέθοδος της ασύμμετρης κρυπτογραφίας[7](Σχήμα 2.2). Συνοπτικά, κάθε χρήστης έχει δύο κλειδιά: ένα δημόσιο, το οποίο αντιπροσωπεύει ένα λογαριασμό στο δίκτυο του Blockchain, και ένα ιδιωτικό το οποίο είναι γνωστό μόνο από τον κάτοχο του λογαριασμού. Το ιδιωτικό κλειδί δίνει πρόσβαση σε όλα τα ηλεκτρονικά περιουσιακά στοιχεία του λογαριασμού αφού ουσιαστικά αποτελεί τον κωδικό πρόσβασης του λογαριασμού με αναγνωριστικό το αντίστοιχο δημόσιο κλειδί.



Σχήμα 2.2: Ασύμμετρη Κρυπτογραφία

Decentralisation

Λόγω του ότι το Blockchain αποθηκεύεται σε όλους τους κόμβους του peer-to-peer δικτύου του, συνεπάγεται ότι δεν υπάρχει κεντρική αρχή η οποία διατηρεί/διαμοιράζει τα δεδομένα, ή ελέγχει την ακεραιότητα τους. Δηλαδή ένα δίκτυο Blockchain δεν έχει κανένα single-point of failure (Σχήμα 2.3) αφού δεν διατηρείται ένα κεντρικό αντίγραφο. Αντίθετα, ο κάθε κόμβος που συμμετέχει στο δίκτυο, διατηρεί αντίγραφο της αλυσίδας και διαδίδει κάθε νέα προσθήκη αλλά και ενημερώνεται από άλλους κόμβους για αλλαγές, ενώ παράλληλα είναι σε θέση να ελέγχει τα δεδομένα τα οποία λαμβάνει.



Σχήμα 2.3: Centralised vs Decentralised

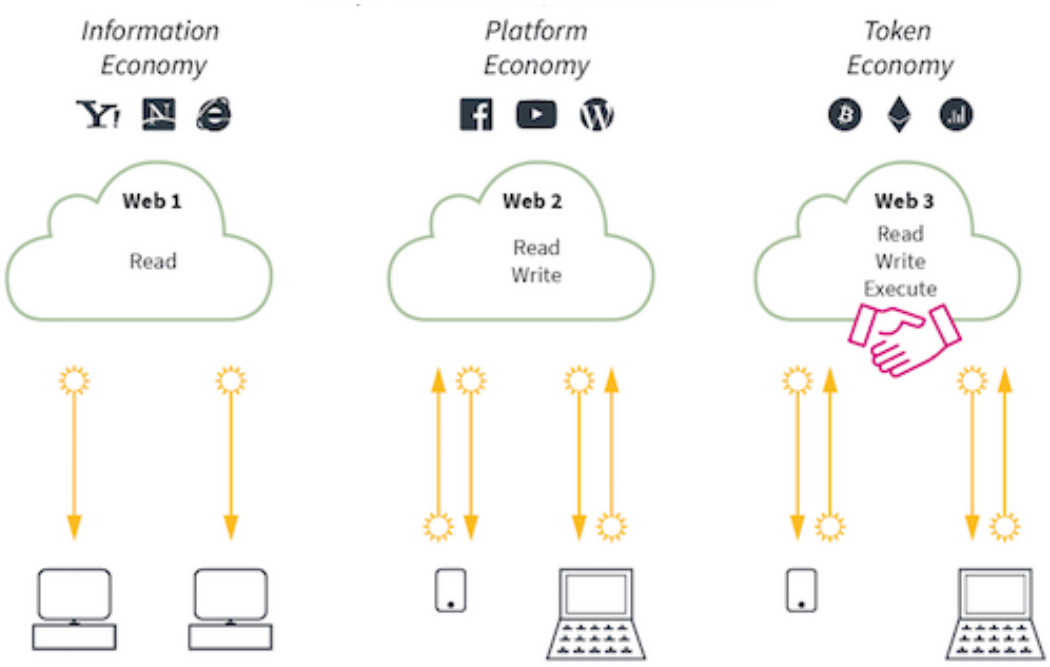
2.3 Web3

Το Web1.0 (ή WWW)[8], είναι η πρώτη μορφή του διαδικτύου που έγινε διαδεδομένη στους χρήστες των ηλεκτρονικών υπολογιστών γύρω στο 1990. Στην πρωταρχική αυτή μορφή του, το διαδίκτυο έδινε μόνο τη δυνατότητα για ανάγνωση (read) στους χρήστες.

Μετά το dot com crash[9] του 2002, μία νέα μορφή διαδικτύου αναπτύχθηκε, το Web2.0[8], το οποίο έδινε τη δυνατότητα στους χρήστες τόσο για εγγραφή όσο και για ανάγνωση (read-write), και έφερε στο φως εταιρίες κολοσσούς, που κυριαρχούν μέχρι και σήμερα, όπως το Facebook, το YouTube και η Wikipedia.

Στις μέρες μας, ζούμε πλέον στην εποχή του Web3.0[8] το οποίο αναπτύχθηκε ως ιδέα μετά την ανάπτυξη της τεχνολογίας του Blockchain αλλά ιδιαίτερα των έξυπνων συμβολαίων (smart contracts). Το Web3.0, δίνει τη δυνατότητα στους χρήστες να μπορούν να εκτελούν κώδικα μέσω του διαδικτύου, ενώ ακόμα έχουν τη δυνατότητα, φυσικά, να διαβάζουν και να γράφουν σε αυτό. (Σχήμα 2.4)

Η επανάσταση που φέρνει το Web3.0 στον κόσμο του διαδικτύου, έγγειται κυρίως στην ανάπτυξη της λογικής των έξυπνων συμβολαίων, τα οποία αποτελούν τη δομική ύλη για την ανάπτυξη αποκεντρωμένων εφαρμογών (Decentralised Applications ή DApps).



Σχήμα 2.4: Web1.0 vs Web2.0 vs Web3.0

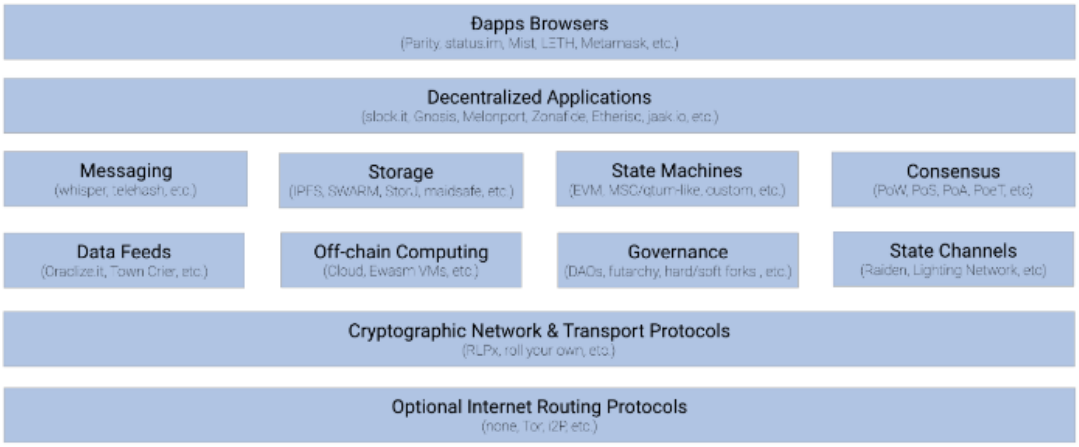
Έξυπνα Συμβόλαια

Όπως έχει αναφερθεί και προηγουμένως, το Ethereum δεν παρέχει στους χρήστες απλά ένα προκαθορισμένο σύνολο λειτουργιών, όπως κάνει το Bitcoin, αλλά αντιθέτως τους παρέχει την δυνατότητα να ορίσουν δικές τους λειτουργίες και κατ'επέκταση έξυπνα συμβόλαια και αποκεντρωμένες εφαρμογές (DApps)[10].

Στο επίκεντρο βρίσκεται το Ethereum Virtual Machine (EVM), το οποίο μπορεί να εκτελεί κώδικα αυθαίρετης αλγοριθμικής πολυπλοκότητας. Το EVM είναι Turing Complete. Όπως και τα άλλα Blockchain, το Ethereum περιλαμβάνει ένα πρωτόκολλο δικτύου ομότιμων κόμβων(peer to peer)[11]. Το πρωτόκολλο αυτό είναι υπεύθυνο για τον συντονισμό των συνδεδεμένων κόμβων, με σκοπό την απρόσκοπτη λειτουργία του δικτύου. Πάνω σε κάθε κόμβο που συμμετέχει στο δίκτυο ¨τρέχει' το EVM, και εκτελεί τις ίδιες εντολές, δίνοντας στο Ethereum Blockchain τη μορφή ενός ¨παγκόσμιου υπολογιστή¨. Ο κώδικας που ¨τρέχει' πάνω στο EVM[12], συνήθως προέρχεται από έξυπνα συμβόλαια τα οποία χρήστες έχουν αναπτύξει και κάνει deploy πάνω στην αλυσίδα. Το κάθε συμβόλαιο έχει δική του διεύθυνση στο δίκτυο και μπορεί να καλείται μέσω του ABI[13] του, το οποίο είναι ουσιαστικά η διεπαφή που δίνει το συμβόλαιο στον έξω κόσμο για να καλούνται οι συναρτήσεις του (παρόμοιο με το API[14] των Web εφαρμογών).

Τα έξυπνα συμβόλαια, δίνουν τη δυνατότητα ανάπτυξης DApps, οι οποίες είναι διαδικτυακές εφαρμογές οι οποίες ¨τρέχουν' πάνω σε κατανεμημένα υπολογιστικά συστήματα, τα οποία συνήθως υλοποιούν Distributed Ledger Technologies (DLTs)[15], όπως το Blockchain. Πρακτικά, οι αποκεντρωμένες εφαρμογές, είναι ένα σύνολο έξυπνων συμβολα-

ίων, τα οποία υλοποιούν μια συγκεκριμένη λειτουργικότητα, όπως ένα κρυπτονόμισμα, ένα ηλεκτρονικό κατάστημα ανταλλαγής βιβλίων ή μία εφαρμογής διαμοιρασμού αποθηκευτικού χώρου, όπως γίνεται στην περίπτωση της συγκεκριμένης εργασίας.



Σχήμα 2.5: Web3.0 Technology Stack

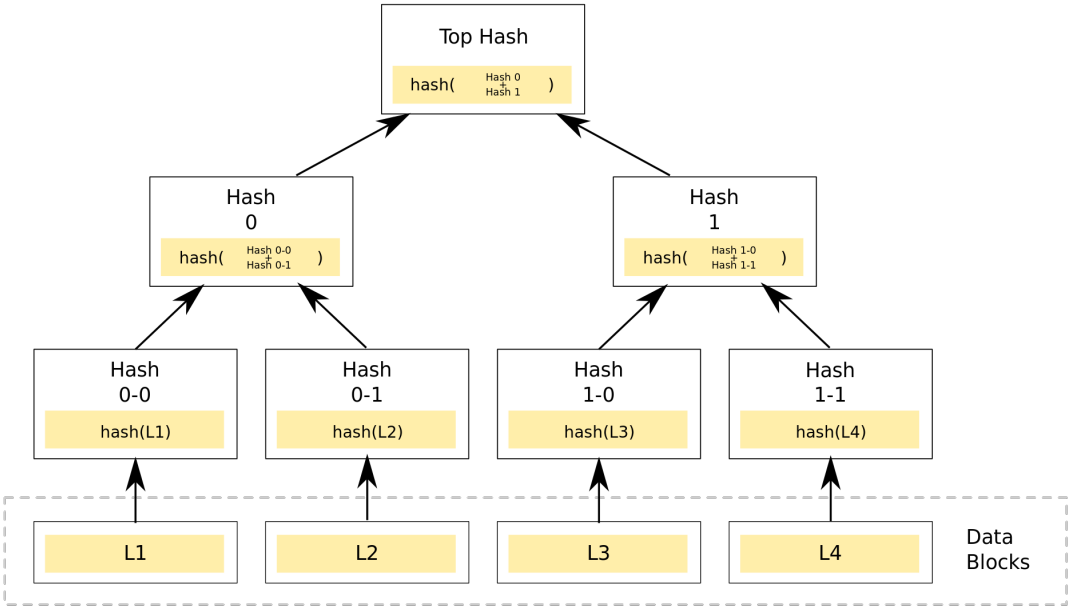
2.4 Proof of Retrievability

Το βασικότερο πρόβλημα που αντιμετωπίζει μια αποκεντρωμένη εφαρμογή είναι η έλλειψη εμπιστοσύνης που υπάρχει ανάμεσα στις δύο πλευρές, τον πάροχο και τον πελάτη. Στην περίπτωση της εφαρμογής της συγκεκριμένης εργασίας, έπρεπε να βρεθεί κάποιος τρόπος ώστε ο πάροχος, που αποθηκεύει το αρχείο, να μπορεί να αποδείξει στον πελάτη ότι έχει στην κατοχή του το αρχείο που καλείται να αποθηκεύσει (χωρίς φυσικά να πρέπει να στείλει ολόκληρο το αρχείο).

Το πρόβλημα αυτό λύνεται με την τεχνική του Proof of Retrievability[16]. Η συγκεκριμένη τεχνική, χρησιμοποιώντας το Merkle Proof[17], δίνει τη δυνατότητα σε έναν πάροχο αποθηκευτικού χώρου να αποδείξει σε έναν πελάτη ότι το αρχείο του είναι ανακτήσιμο.

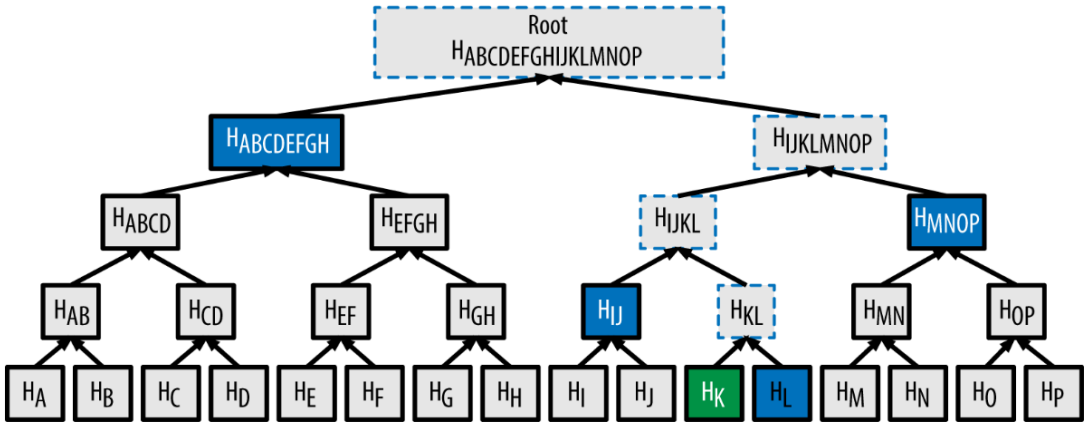
Για να μπορέσει να γίνει αυτό, το αρχείο προς αποθήκευση χωρίζεται σε κομμάτια (leaves), από τα οποία σχηματίζεται ένα δέντρο Merkle, όπως φαίνεται στο Σχήμα 2.6.

Τα φύλλα αυτά αρχικά κρυπτογραφούνται με χρήση κάποιου αλγορίθμου κρυπτογράφησης (για παράδειγμα sha256[18]). Έπειτα, αφού κρυπτογραφηθούν εκ νέου, αυτή τη φορά μαζί με το διπλανό τους φύλλο, περνάνε στο ανώτερο επίπεδο του δέντρου. Αφού δημιουργηθεί το δέντρο, ο πελάτης κρατά το root element, το οποίο καλείται root hash. Έχοντας αυτό το hash στην κατοχή του, ο πελάτης μπορεί ανά πάσα στιγμή να προκαλέσει τον πάροχο, δίνοντας του ένα τυχαίο φύλλο, και ζητώντας του να του επιστρέψει το σύνολο των κόμβων του Merkle Tree, το οποίο οδηγεί στο root. Για να μπορέσει ο πάροχος να επιστρέψει το ζητούμενο σύνολο κόμβων, θα πρέπει να έχει στην κατοχή του εξόλοκλήρου το δέντρο, άρα να έχει και τα φύλλα του, τα οποία ουσιαστικά αποτελούν το αρχείο το οποίο καλείται να αποθηκεύσει. Στο Σχήμα 2.7, οι μπλε κόμβοι, αλλά και αυτοί με δια-



Σχήμα 2.6: Merkle Tree Example

κεκομμένες μπλε γραμμές, είναι αυτοί που θα έπρεπε ένας πάροχος να επιστρέψει στον πελάτη, σε περίπτωση που ο τελευταίος τον προκαλούσε για το -πράσινο- φύλλο L.



Σχήμα 2.7: Merkle Proof Example

Κεφάλαιο 3

Εργαλεία και Τεχνολογίες

3.1 Εισαγωγή

Στο παρόν κεφάλαιο, παρουσιάζονται τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της αποκεντρωμένης εφαρμογής που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας. Επιγραμματικά, χρησιμοποιήθηκαν εργαλεία που δίνουν τη δυνατότητα για δημιουργία ενός τοπικού Ethereum Blockchain, το Ganache, το Truffle που δίνει τη δυνατότητα για πρόσβαση και κλήση σε μεθόδους έξυπνων συμβολαίων του Blockchain, η προγραμματιστική γλώσσα Solidity που χρησιμοποιήθηκε για την ανάπτυξη των έξυπνων συμβολαίων, το Web3.js extension της Javascript, καθώς και το Metamask, ένα browser extension για τη σύνδεση του UI της εφαρμογής με την αλυσίδα. Τέλος, αναφορά θα γίνει και στο Node.js, το πρόγραμμα διαχείρισης πακέτων NPM και κάποια εργαλεία για ανάπτυξη ιστοσελίδων.

3.2 Truffle

Έκδοση που χρησιμοποιήθηκε: 5.0.1

Το Truffle[19] είναι μια σουίτα, που επιτρέπει και διευκολύνει την ανάπτυξη αποκεντρωμένων εφαρμογών. Μέσω του Truffle, δίνεται στο χρήστη η δυνατότητα για πρόσβαση στο Blockchain μέσω κονσόλας, ενώ υπάρχουν πολλά εργαλεία του τα οποία βοηθούν στο testing της εφαρμογής.

Μέσω της κονσόλας του Truffle, ο χρήστης μπορεί να κάνει compile τα έξυπνα συμβόλαιά του, να τα κάνει deploy στο Blockchain και να καλεί τις μεθόδους τους. Επίσής, διευκολύνει τις δικτυακές ρυθμίσεις που πρέπει να γίνουν για να μπορέσει κάποιος να διαδράσει με το Blockchain.

canagno@canagnoLaptop: /ethChain/truffle truf console

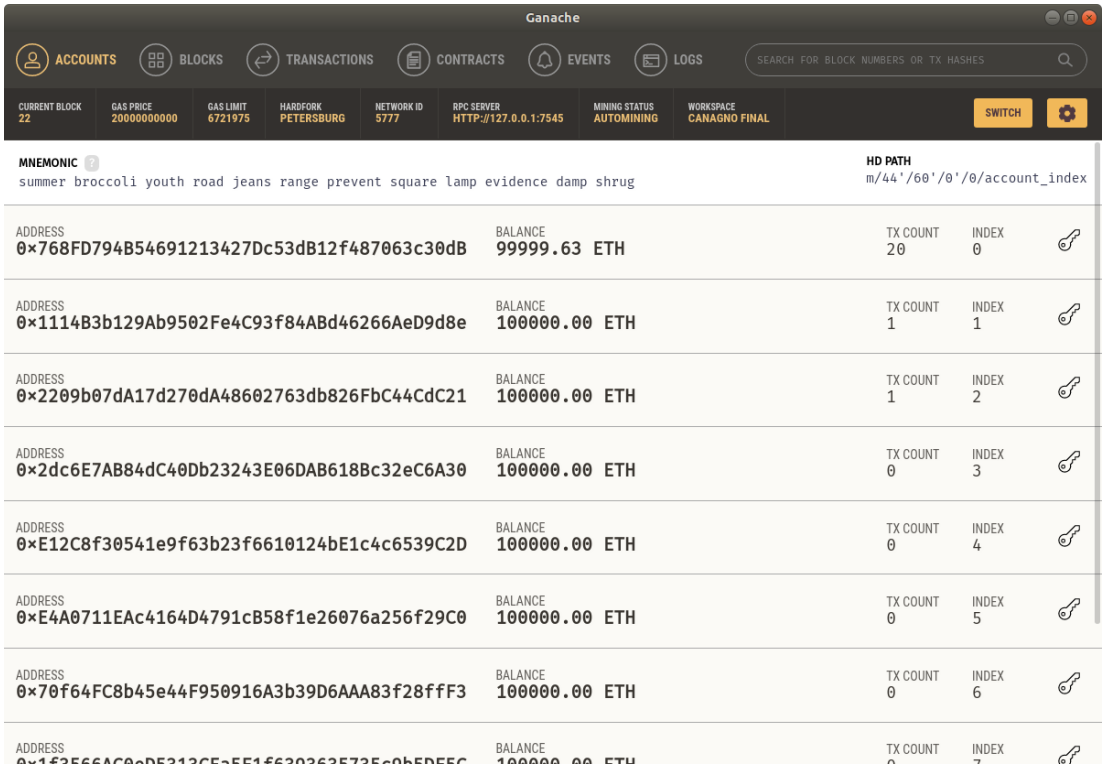
truffle(development)>

3.3 Ganache

Έκδοση που χρησιμοποιήθηκε: 2.1.2

Από τη σουίτα του Truffle, χρησιμοποιήθηκε ένα ακόμα εργαλείο, το Ganache[20], το οποίο δίνει στο χρήστη τη δυνατότητα να δημιουργήσει ένα τοπικό Ethereum Blockchain, κάνοντας όλες τις απαραίτητες δικτυακές ρυμίσεις (IP, δικτυακές θύρες κ.λ.π.) αλλά και να έχει εκ των προτέρων δημιουργήσει κάποιους λογαριασμούς χρηστών με συγκεκριμένα ποσά ether. Ουσιαστικά, το Ganache δίνει στο χρήστη ένα UI στο οποίο βάζει τις ρυθμίσεις του Genesis Block του, και με βάση αυτό δημιουργεί ένα τοπικό Blockchain για developing(Σχήμα 3.1).

Η χρήση ενός τοπικού Blockchain και όχι του Ethereum Main Net, είναι επιτακτική, καθώς εφόσον είχε χρησιμοποιηθεί για την εφαρμογή το Main Net, ο χρόνος αναμονής για την επικύρωση ενός transaction θα ήταν σημαντικά μεγαλύτερος (περίπου 10 δευτερόλεπτα, ενώ τοπικά γίνεται σε επίπεδο ms), αλλά και θα δημιουργούταν αχρείαστη κίνηση και δεδομένα στην κεντρική αλυσίδα του Ethereum.



Σχήμα 3.1: Ganache

3.4 Web3.js

Έκδοση που χρησιμοποιήθηκε: 2.0.0(alpha)

Ένα από τα πιο σημαντικά npm πακέτα που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Το πακέτο web3.js[21] είναι μία συλλογή βιβλιοθηκών που επιτρέπουν την

αλληλεπίδραση με ένα τοπικό ή απομακρυσμένο κόμβο του Ethereum Blockchain, χρησιμοποιώντας HTTP ή IPC σύνδεση. Με άλλα λόγια είναι μία διεπαφή επικοινωνίας μεταξύ της JavaScript και του Blockchain. Είναι, δηλαδή, ο πιο διαδεδομένος τρόπος για να αναφέρεται η JavaScript(server-side ή/και front-end) σε αντικείμενα που «ζουν» μέσα στο Blockchain, όπως τα έξυπνα συμβόλαια, τα δεδομένα τους, τις συναρτήσεις τους, τις διευθύνσεις, τα υπόλοιπα λογαριασμών, όπως και πολλά άλλα.

3.5 Solidity

Έκδοση που χρησιμοποιήθηκε: 0.5.0

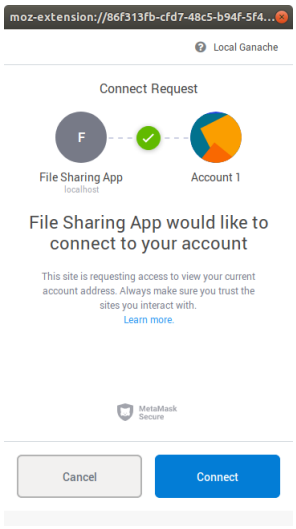
Η Solidity[22] είναι μία αντικειμενοστραφής, υψηλού επιπέδου γλώσσα προγραμματισμού που εκτελείται στο Ethereum Virtual Machine(EVM) και έχει δημιουργηθεί για να μεγεθύνει τις δυνατότητες της ιδεατής αυτής μηχανής. Αντικείμενο για τη Solidity δεν είναι το αντικείμενο όπως το έχουμε στο μυαλό μας για παράδειγμα στη Java, αλλά το ίδιο το έξυπνο συμβόλαιο. Χρησιμοποιείται κυρίως για την δημιουργία έξυπνων συμβολαίων για το Ethereum Blockchain. Ο ίδιος ο πηγαίος κώδικας της γλώσσας Ethereum είναι γραμμένος σε Solidity. Έχει παρόμοιο συντακτικό με αυτό της JavaScript, οπότε είναι εύκολα κατανοήσιμη. Είναι μία στατικού τύπου γλώσσα. Υποστηρίζει την κληρονομικότητα με παρόμοιο τρόπο με άλλες γλώσσες προγραμματισμού (πχ. Java).

3.6 Metamask

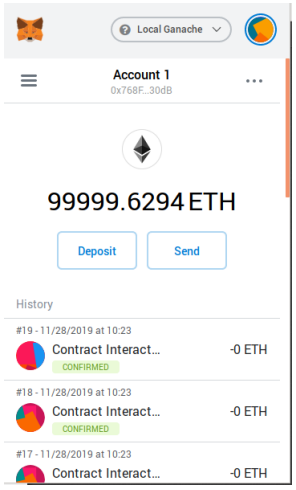
Έκδοση που χρησιμοποιήθηκε: 7.6.1

Το ΜεταΜασκ[23] είναι ένα plugin διαθέσιμο για τους φυλλομετρητές Google Chrome, Mozilla Firefox, Brave και Opera. Το MetaMask είναι στην ουσία μία γέφυρα η οποία δίνει στον browser πρόσβαση στις αποκεντρωμένες εφαρμογές (DApps), που για την λειτουργία τους βασίζονται στο δίκτυο Ethereum. Το μεγάλο πλεονέκτημα που προσφέρει είναι το γεγονός ότι με την χρήση του, η πρόσβαση στις εφαρμογές αυτές δεν απαιτεί την εκτέλεση του πλήρους Ethereum κόμβου στο μηχάνημα του χρήστη, όπως για παράδειγμα απαιτεί ο ειδικός για το Ethereum φυλλομετρητής Mist. Συμπεριλαμβάνει μία ασφαλή κρύπτη και παρέχει στον χρήστη μία διεπαφή, μέσω της οποίας αυτός μπορεί να συνδέεται στο Ethereum Blockchain(Σχήμα 3.2), να διαχειρίζεται τους Ethereum λογαριασμούς/διευθύνσεις του (Σχήμα 3.3), ώστε να αλληλοεπιδρά με τις ιστοσελίδες, να στέλνει ή και να υπογράφει συναλλαγές (Σχήμα 3.4).

Σύμφωνα με τους δημιουργούς του, ο σκοπός του είναι να κάνει το Ethereum προσβάσιμο σε όσο το δυνατόν περισσότερο κόσμο, κάτι που πετυχαίνει σε πολύ μεγάλο βαθμό, καθώς η εγκατάστασή του, η σύνδεσή του στην αλυσίδα αλλά και η αλλιλεπίδραση με αυτή γίνονται άμεσα και χωρίς την απαίτηση για πολλές τεχνικές γνώσεις.



Σχήμα 3.2: Σύνδεση στην αλυσίδα μέσω Metamask



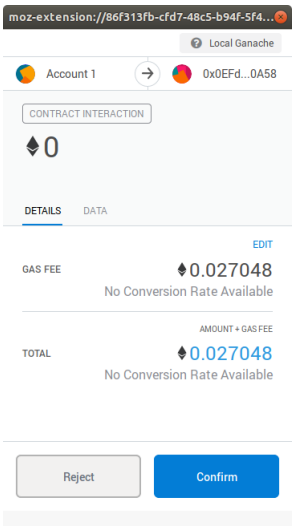
Σχήμα 3.3: Διαχείριση λογαριασμών χρήστη στο Metamask

3.7 NPM

Έκδοση που χρησιμοποιήθηκε: 6.9.0

Το NPM–Node Package Manager[24] είναι ένα πρόγραμμα διαχείρισης πακέτων της JavaScript, επίσης είναι το μεγαλύτερο μητρώο προγραμμάτων στον κόσμο. Διαχειριστής του είναι η εταιρία NPM, Inc. Είναι το προκαθορισμένο πρόγραμμα διαχείρισης πακέτων του προγραμματιστικού περιβάλλοντος Node.js. Εκτός από πακέτα, στο αρχείο του υπάρχουν και nodemodules τα οποία χρησιμοποιούνται στο server-side προγραμματισμό. Το πακέτο (package) είναι ένα αρχείο ή ένα directory το οποίο περιγράφεται από ένα package.json αρχείο, ενώ το module είναι ένα αρχείο ή ένα directory το οποίο φορτώνεται από το Node.js μέσω της εντολής require().

Είναι πολύ χρήσιμο στην κοινότητα ανάπτυξης λογισμικού, διότι αφενός επιτρέπει την ανταλλαγή πακέτων, δηλαδή κώδικα που λύνει ένα συγκεκριμένο πρόβλημα και αφετέρου



Σχήμα 3.4: Αποδοχή/απόρριψη συναλλαγής στο Metamask

έχει θεσπίσει προδιαγραφές τις οποίες ακολουθούν όλα τα πακέτα που δημοσιεύονται στο αρχείο, έτσι ώστε να υπάρχει ένας όσο γίνεται ενιαίος τρόπος χρησιμοποίησής τους από τους προγραμματιστές.

3.8 Node.js

Έκδοση που χρησιμοποιήθηκε: 10.17.0

Το Node.js[25] είναι μία server-side πλατφόρμα, η οποία έχει αναπτυχθεί πάνω στο Google Chrome JavaScript Engine(V8 Engine) το 2009. Είναι ένα ανοιχτού κώδικα (Opensource), cross-platform περιβάλλον ανάπτυξης και εκτέλεσης της γλώσσας προγραμματισμού JavaScript, κατάλληλο για εύκολη και γρήγορη ανάπτυξη κλιμακώσιμων διαδικτυακών εφαρμογών. Χρησιμοποιεί ένα event-driven, non-blocking I/O μοντέλο και πετυχαίνει μέσω αυτού μεγάλη απόδοση χρησιμοποιώντας λίγους φυσικούς πόρους. Θεωρείται ιδανικό για εφαρμογές πραγματικού χρόνου, υπολογιστικά έντονες που τρέχουν σε κατανεμημένα περιβάλλοντα. Επίσης, το Node.js παρέχει μία μεγάλη βιβλιοθήκη από JavaScript modules, γεγονός που απλοποιεί ουσιωδώς τη διαδικασία ανάπτυξης διαδικτυακών εφαρμογών. Δηλαδή, το Node.js είναι και περιβάλλον εκτέλεσης, αλλά και βιβλιοθήκη της JavaScript.

Τα βασικά του χαρακτηριστικά είναι τα εξής:

- **Ασύγχρονο και οδηγούμενο από γεγονότα (Asynchronous and Event Driven)** – Όλες οι προγραμματιστικές διεπαφές (API application programming interface) της Node.js βιβλιοθήκης είναι ασύγχρονες κάτι που σημαίνει ότι ο κώδικας δεν κολλάει ποτέ σε ένα σημείο περιμένοντας από μία σύνθετη (εξωτερική) διεργασία (API) να επιστρέψει δεδομένα, παρά προχωράει η εκτέλεση, και όταν η κληθείσα διεργασία ολοκληρωθεί, ο server, μέσω ενός ειδικού μηχανισμού ενημέρωσης συμ-

βάντων (notification mechanism of events) λαμβάνει την απάντηση/δεδομένα από αυτήν. Αυτό το χαρακτηριστικό είναι ιδιαίτερα χρήσιμο για ανάπτυξη Blockchain εφαρμογών, αφού σε κάποιες περιπτώσεις η επιβεβαίωση κάποιων transactions μπορεί να είναι μια ιδιαίτερα χρονοβόρα διαδικασία.

- **Πολύ γρήγορο** – Επειδή βασίζεται πάνω στην ιδεατή μηχανή Google Chrome V8 JavaScript Engine η βιβλιοθήκη Node.js είναι πολύ γρήγορη στην εκτέλεση κώδικα.
- **Μονού νήματος, αλλά πολύ κλιμακώσιμο** – Το Node.js χρησιμοποιεί μοντέλο μονού νήματος εκτέλεσης με event looping. Ο μηχανισμός αυτός των συμβάντων βοηθάει τον εξυπηρετητή να απαντάει με non-blocking τρόπο κάτι που του δίνει την ιδιότητα της κλιμακωσιμότητας, σε αντίθεση με άλλους παραδοσιακούς εξυπηρετητές που χρησιμοποιούν περιορισμένο αριθμό νημάτων για την εξυπηρέτηση αιτήσεων. Το πρόγραμμα ενός νήματος του Node.js μπορεί να εξυπηρετήσει πολύ μεγαλύτερο αριθμό αιτήσεων από παραδοσιακούς εξυπηρετητές, όπως για παράδειγμα ο Apache HTTP Server.
- **Δεν υπάρχει προσωρινή αποθήκευση δεδομένων (caching)** – Οι εφαρμογές του Node.js δεν κάνουν προσωρινή αποθήκευση δεδομένων, αλλά στέλνουν τα δεδομένα σε μικρά κομμάτια.

Το Node.js αναπτύσσεται από την Κοινότητα Ανοιχτού Λογισμικού και ο έλεγχος, η υποστήριξη, και η διάθεση του Node.js γίνεται από την Community Committee (διατίθεται δωρεάν). Στην επιτροπή αυτή μπορεί να συμμετάσχει ο οποιοσδήποτε επιθυμεί να συμβάλλει στην ανάπτυξη του κώδικα του προγράμματος.

3.9 Ανάπτυξη ιστοσελίδας

Για τις ανάγκες της παρουσίασης της εργασίας, έγινε ανάπτυξη μιας απλής ιστοσελίδας, ώστε να μπορούν αν γίνουν κάποια απλά use case σενάρια και να φανεί η λειτουργικότητα του back end κώδικα. Για την ανάπτυξη της ιστοσελίδας χρησιμοποιήθηκαν τα εξής εργαλεία/τεχνολογίες:

- **HTML (Hypertext Markup Language)**: Είναι γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ. Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και να τα συνθέσει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να παρουσιάσει το περιεχόμενο της σελίδας[26].

- **CSS (Cascading Style Sheets)**: Είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων ύφους και χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την HTML. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη[27].

- **JavaScript**: Εκτός από το back-end (κώδικας που εκτελείται στον εξυπηρετητή μέσω του Node.js), JavaScript[28] χρησιμοποιήθηκε και για το front-end (κώδικας που εκτελείται στον φυλλομετρητή του πελάτη – client-side JavaScript). Η JavaScript είναι μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού, αφού χρησιμοποιείται κατά κόρον σε διαδικτυακές εφαρμογές (και όχι μόνο) και υποστηρίζεται από όλους τους μοντέρνους φυλλομετρητές. Είναι ο καλύτερος (αν όχι ο μόνος ενδεδειγμένος) τρόπος να δώσουμε δυναμικό περιεχόμενο σε μία ιστοσελίδα κάνοντάς την να επιτελεί σύνθετες λειτουργίες. Μπορεί επίσης να χρησιμοποιηθεί και για την ανάπτυξη προγραμμάτων που δεν εκτελούνται σε φυλλομετρητές, όπως για παράδειγμα σε εξυπηρετητές, βάσεις δεδομένων, επεξεργαστές PDF εγγράφων, Desktop εφαρμογές, αλλά και εφαρμογές κινητών. Η JavaScript είναι μία υψηλού επιπέδου, δυναμική, weakly typed, prototype-based, multi-paradigm, interpreted γλώσσα προγραμματισμού. Ως multi-paradigm γλώσσα, η JavaScript υποστηρίζει τα event-driven, functional, and imperative (including object-oriented and prototype-based) προγραμματιστικά στυλ.Έχει έτοιμες προγραμματιστικές διεπαφές (APIs) για την επεξεργασία κειμένου, πινάκων, ημερομηνιών, καθώς και τον χειρισμό DOM (Document Object Model), όμως δεν υποστηρίζει λειτουργίες εισόδου εξόδου (I/O), όπως δικτύωση, αποθήκευση, γραφικά, παρά βασίζει την υλοποίηση των λειτουργιών αυτών στο περιβάλλον εκτέλεσής της.

Κεφάλαιο 4

Οντότητες Εφαρμογής

4.1 Εισαγωγή

Στο παρόν κεφάλαιο θα παρουσιαστούν οι οντότητες που σχηματίστηκαν για την ανάπτυξη της αποκεντρωμένης εφαρμογής.

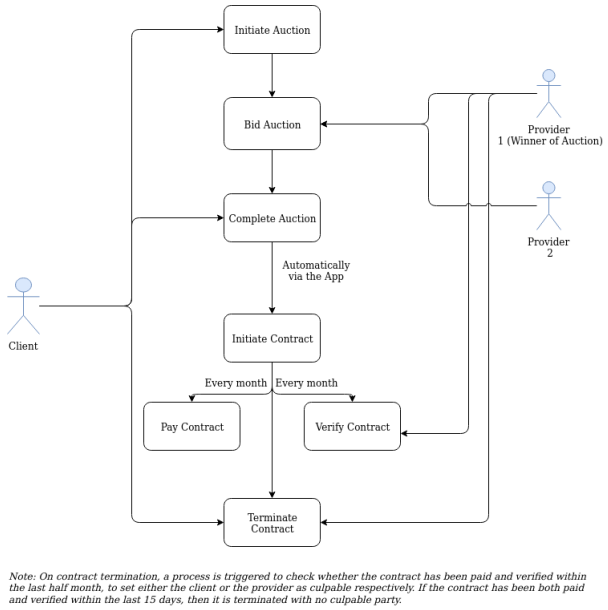
Όλη η εφαρμογή δομήθηκε με βάση τον αντικειμενοστραφή προγραμματισμό, ο οποίος, στα δεδομένα των έξυπνων συμβολαίων, περιγράφεται καλύτερα ως συμβολαιοστραφής προγραμματισμός. Πιο συγκεκριμένα, κάθε οντότητα της εφαρμογής αποτελεί ένα ξεχωριστό smart contract, στο οποίο περιέχονται όλα τα πεδία της οντότητας αλλά και βασικές μέθοδοι πρόσβασης στην οντότητα αυτή (constructor, getters, setters).

Αξίζει να σημειωθεί ότι για την πιο σημαντική οντότητα της εφαρμογής, τον ίδιο τον χρήστη, δε χρειάστηκε να δημιουργηθεί κάποια επιπλέον οντότητα, αφού ως αναγνωριστικό χρήστη (και το μόνο στοιχείο που χρειάζεται σε μια αποκεντρωμένη εφαρμογή) χρησιμοποιήθηκε η, μοναδική, δεκαεξαδικής μορφής διεύθυνση που δίνεται σε κάθε συμμετέχοντα στο Ethereum Blockchain. Οι χρήστες χωρίζονται σε δύο μεγάλες κατηγορίες, τους παρόχους, που προσφέρουν τις υπηρεσίες αποθήκευσης αρχείου, και τους πελάτες, που αγοράζουν τις υπηρεσίες αυτές με wei[29] (10^{-18} ether). Ένας πελάτης ενός συμβολαίου, μπορεί να γίνει πάροχος σε ένα **άλλο** συμβόλαιο και το αντίστροφο, αλλά ένας χρήστης δε μπορεί να έχει και τους δύο ρόλους στο ίδιο συμβόλαιο.

Οι πρωταρχικές οντότητες που δημιουργήθηκαν είναι αυτή του συμβολαίου μεταξύ ενός παρόχου και ενός πελάτη της εφαρμογής, που περιέχει όλα εκείνα τα στοιχεία που χρειάζονται για να επικυρώσουν τη συμφωνία μεταξύ τους, καθώς και η οντότητα της δημοπρασίας, που προηγείται της δημιουργίας του συμβολαίου και η λειτουργικότητα της θα επεξηγηθεί σε βάθος στο Κεφάλαιο 5. Επίσης, δημιουργήθηκαν οντότητες για τη διαχείριση των συμβολαίων και των δημοπρασιών, που είναι υπεύθυνες να ορίσουν τον κύκλο ζωής των πρωταρχικών οντοτήτων. Τέλος, δημιουργήθηκε οντότητα για την αξιολόγηση των παρόχων, καθώς και μία οντότητα για την εκτέλεση της λειτουργικότητας του Proof of Retrievability.

Στο Σχήμα 4.1 φαίνεται διάγραμμα με μια βασική ροή χρήσης της εφαρμογής. Οι

οντότητες και οι διαδικασίες θα γίνουν πιο σαφείς στα υποκεφάλαια που ακολουθούν.



Σχήμα 4.1: Ένα high level use case της εφαρμογής.

4.2 Δημοπρασία

Η οντότητα της δημοπρασίας αποτελεί την αναπαράσταση της διαδικασίας την οποία εκκινεί ένας χρήστης ώστε να βρει τον πάροχο εκείνο που του προσφέρει την καλύτερη τιμή σε wei ώστε να αναλάβει την αποθήκευση του αρχείου του για ένα μήνα.

Τα πεδία της συγκεκριμένης οντότητας είναι τα εξής:

- **providersList** – Λίστα με τις δεκαεξαδικές διευθύνσεις όλων των παρόχων που έχουν κάνει bid στη συγκεκριμένη δημοπρασία. Στην πρώτη φάση της εφαρμογής, η λίστα αυτή περιέχει το πολύ έναν πάροχο.

- **providersPriceList** – Λίστα με τις τιμές σε wei που έχει προσφέρει ο αντίστοιχος πάροχος από την providersList. Ομοίως με τη λίστα διευθύνσεων των παρόχων, στην πρώτη φάση της εφαρμογής η λίστα των τιμών περιέχει το πολύ μία τιμή.

- **client** – Η δεκαεξαδική διεύθυνση του πελάτη που εκκίνησε τη διαδικασία της δημοπρασίας.

- **fileSize** – Το μέγεθος του αρχείου προς αποθήκευση, ώστε να μπορεί ο πάροχος να προσαρμόσει την τιμή την οποία θα προσφέρει για την αποθήκευσή του.

- **rootHash** – Το hash του root κόμβου το δέντρου Merkle που δημιουργείται από το αρχείο, που αποθηκεύεται στην οντότητα της δημοπρασίας για να μπορέσει με την ολοκλήρωσή της να εκκινήσει το αντίστοιχο συμβόλαιο μεταξύ πελάτη και παρόχου.

- **collateralAmount** – Το collateralAmount αποτελεί ένα ποσό, εκφρασμένο σε wei, που θα πρέπει να αποσταλλεί στη διεύθυνση του αντίστοιχου smart contract ώστε να μπορέσει να εκκινήσει το συμβόλαιο μεταξύ πελάτη και παρόχου. Το collateralAmount χρησιμοποιείται ως δικλείδα ασφαλείας σε περίπτωση που μία από της δύο πλευρές του συμβολαίου (πελάτης ή πάροχος) αθετήσει τους όρους του και τίθεται από τον πελάτη.

- **startTime** – Η ώρα εκκίνησης της δημοπρασίας. Η ώρα στο Ethereum Blockchain μετριέται σε δευτερόλεπτα από την τελευταία Εποχή (Epoch[1]).

- **endTime** – Η ώρα ολοκλήρωσης της δημοπρασίας.

- **status** – Η κατάσταση στην οποία βρίσκεται η δημοπρασία. Παίρνει δύο τιμές, 1 και 2. Το 1 αντιστοιχεί στο ONGOING, ότι, δηλαδή, η δημοπρασία είναι σε εξέλιξη, και το 2 σε COMPLETED, ότι, δηλαδή, η δημοπρασία έχει ολοκληρωθεί.

```
pragma solidity ^0.5.0;

contract Auction{

    address payable[] private providersList;
    uint[] private providersPriceList;
    address payable private client;
    uint8 private providersNumber;
    uint8 private fileSize;
    bytes32 private rootHash;
    uint private collateralAmount;
    uint private startTime; //startTime is initiated to now in the init function of the auction. It is caclulated in seconds since the last Epoch.
    uint private endTime; //time the auction was completed.
    uint8 private status; //state of the contract
    //Status Constants
    uint8 constant public ONGOING = 1;
    uint8 constant public COMPLETED = 2;

    constructor(address payable _client, uint8 _providersNumber, uint8 _fileSize, bytes32 _rootHash, uint _collateralAmount)
    public
    {
        providersList = new address payable[](providersNumber);
        providersPriceList = new uint[](providersNumber);
        client = _client;
        fileSize = _fileSize;
        rootHash = _rootHash;
        collateralAmount = _collateralAmount;
        providersNumber = _providersNumber;
        startTime = now;
        status = ONGOING;
    }

```

Σχήμα 4.2: Κώδικας για την οντότητα της δημοπρασίας.

4.3 Αξιολόγηση Παρόχων

Η οντότητα αυτή αποθηκεύει και διαχειρίζεται την αξιολόγηση των παρόχων που έχουν συμμετάσχει στην εφαρμογή αποθηκεύοντας κάποιο αρχείο. Κάθε πάροχος ξεκινά από rating 2, το οποίο αυξάνεται ή μειώνεται ανάλογα με το πως εξελίσσονται τα συμβόλαια στα οποία συμμετέχει (θα επεξηγηθεί αναλυτικά στο Κεφάλαιο 5) και μπορεί να φτάσει

[1]With Bitcoin, Ethereum, and many other cryptocurrencies, the blockchains are secured by the process known as mining. Usually, one's ability to mine is restricted mostly by one's access to computation power – faster processor = more revenue. With Ethereum, an additional resource is pretty much necessary to mine: memory. When mining using a GPU, this means memory on your graphics card. Ethereum deliberately makes mining more memory intensive as time goes on; this happens on a fixed schedule. Every 30000 blocks, a new piece of data (a DAG) is used for mining new blocks. Each new group of 30000 blocks is known as an epoch. And epoch switch is when the next DAG is loaded.

μέχρι την αξιολόγησή του μέχρι το 10. Ο σκοπός της οντότητας είναι, αρχικά, να λύνει τις περιπτώσεις εκείνες που δύο πάροχοι κάνουν την ίδια οικονομική πρόταση για την αποθήκευση ενός αρχείου. Στην περίπτωση αυτή προτείνεται ο πάροχος με την υψηλότερη αξιολόγηση. Επίσης, αποκλείει παρόχους οι οποίοι έχουν αποδειχθεί κακόβουλοι σε δύο ή περισσότερα συμβόλαια.

Τα πεδία της οντότητας είναι τα εξής:

- **ratings** – Mapping που αποθηκεύει την αντιστοιχία της δεκαεξαδικής διεύθυνσης ενός παρόχου με το rating του. Η δομή δεδομένων mapping[2] στη Solidity είναι ανάλογη του Map της Java.
- **blackList** – Λίστα που αποθηκεύει τις διευθύνσεις των παρόχων που έχουν μηδενικό rating και δεν μπορούν πλέον να συμμετέχουν σε δημοπρασίες και συμβόλαια.

Τέλος, η οντότητα του ProviderRating περιέχει μεθόδους για να προσθέτει παρόχους στη λίστα με τις αξιολογήσεις (addProvider), για να αυξάνει (levelUpProvider) και να μειώνει (levelDownProvider) την αξιολόγηση ενός παρόχου, αλλά και για να ελέγχει αν ένας πάροχος είναι αποκλεισμένος (providerBlackListed).

```
pragma solidity ^0.5.0;

contract ProviderRating{

    mapping(address => uint8) private ratings;
    mapping(address => uint8) private blackList;

    constructor ()
    public
    {}

    function addProvider(address payable _provider)
    public
    returns (bytes32)
    {
        if (!providerExists(_provider) && !providerBlackListed(_provider))
        {
            ratings[_provider] = 2;
            return "OK";
        }
        else if (providerExists(_provider))
            return "Exists";
        else if (providerBlackListed(_provider))
            return "BlackListed";
    }

```

Σχήμα 4.3: Κώδικας για την οντότητα της αξιολόγησης των παρόχων.

4.4 Auction Executor

Η οντότητα αυτή έχει ως σκοπό να διαχειρίζεται τον κύκλο ζωής μιας δημοπρασίας. Ο auctionExecutor έχει τα εξής πεδία:

- **id** – Μοναδικό αναγνωριστικό για κάθε δημοπρασία, δημιουργείται, ανατίθεται σε κάθε δημοπρασία και αποθηκεύεται στον auctionExecutor. Αποτελεί, ουσιαστικά,

[2]Mapping types are declared as mapping(KeyType => ValueType). Here KeyType can be almost any type except for a mapping, a dynamically sized array, a contract, an enum and a struct. ValueType can actually be any type, including mappings. Mappings can be seen as hash tables which are virtually initialized such that every possible key exists and is mapped to a value whose byte-representation is all zeros: a type's default value. The similarity ends here, though; The key data is not actually stored in a mapping, only its keccak256 hash used to look up the value.

ένα sequence αριθμών που ξεκινά από το 1 και αυξάνεται κατά ένα με κάθε νέα δημοπρασία.

- **auctions** – Mapping που αποθηκεύει την αντιστοιχία του id με το instance του συμβολαίου auction στο οποίο αντιστοιχεί.
- **ratings** – Αποτελεί ένα instance της οντότητας του ProviderRating, στο οποίο αποθηκεύονται όλες οι αξιολογήσεις για τους παρόχους.

Η οντότητα του auctionExecutor χρησιμοποιείται μέσω κάποιων μεθόδων/διεπαφών που προσφέρει, οι οποίες εξυπηρετούν τη δημιουργία, προσθήκη κάποιας νέας προσφοράς και της ολοκλήρωσης της δημοπρασίας. Οι μεθόδοι αυτόι είναι οι εξής:

Εκκίνηση Δημοπρασίας (initAuction)

Η μέθοδος αυτή καλείται από κάποιον πελάτη, ώστε να δημιουργήσει μια νέα δημοπρασία. Παίρνει σαν ορίσματα τα εξής:

- **providersNumber** – Τον αριθμό των παρόχων στους οποίους θέλει να αποθηκευτεί το αρχείο του. Στην πρώτη φάση της εφαρμογής, αυτό είναι πάντα 1.
- **fileSize** – Το μέγεθος του αρχείου που θέλει να αποθηκεύσει.
- **rootHash** – Το hash του root κόμβου το δέντρου Merkle που δημιουργείται από το αρχείο του.
- **collateralAmount** – Το ποσό που θα κατατεθεί στο συμβόλαιο με την εκκίνησή του και θα χρησιμοποιηθεί ως δικλείδα ασφαλείας, όπως αναφέρθηκε παραπάνω.

Δίνοντας αυτά τα στοιχεία, η μέθοδος καλεί τον constructor της οντότητας Auction και δημιουργεί ένα νέο instance δημοπρασίας, με διεύθυνση πελάτη τη διεύθυνση του καλούντα τη μέθοδο. Παράλληλα, αποθηκεύει το instance αυτό στο mapping auctions και αυξάνει το αναγνωριστικό id κατά ένα

Κατάθεση προσφοράς (bid)

Η μέθοδος αυτή καλείται από παρόχους που ενδιαφέρονται να καταθέσουν προσφορά για κάποια συγκεκριμένη δημοπρασία. Παίρνει σαν όρισμα μόνο το μοναδικό αναγνωριστικό της δημοπρασίας (id), αλλά έχει συγκεκριμένους περιορισμούς:

- Η δημοπρασία πρέπει να είναι σε status ONGOING.
- Η συγκεκριμένη μέθοδος δεν μπορεί να κληθεί από τον ίδιο τον πελάτη.
- Η προσφορά θα πρέπει να είναι πάνω από 0 wei.

- Ο πάροχος που καλεί τη συνάρτηση δε θα πρέπει να είναι αποκλεισμένος λόγω μηδενικής αξιολόγησης.
- Ο πάροχος δεν μπορεί να ξανακάνει προσφορά αν ήδη υπάρχει στη λίστα παρόχων της συγκεκριμένης δημοπρασίας.

Η λογική της συγκεκριμένης μεθόδου μπορεί να υποστηρίξει και περισσότερους από έναν παρόχους στη λίστα των παρόχων. Πιο συγκεκριμένα, αποθηκεύει όλες τις προσφορές μέχρι να συμπληρωθεί ο αριθμός που ορίζεται από το πεδίο providersNumber της δημοπρασίας. Όταν η λίστα έχει το μήκος ίσο με αυτό τον αριθμό και γίνει κάποια νέα προσφορά, τότε, αν η νέα προσφορά είναι μικρότερη από κάποια από αυτές που ήδη υπάρχουν, ο πάροχος με τη μεγαλύτερη προσφορά διαγράφεται από τη λίστα και αντικαθίσταται από τον νέο πάροχο. Σε περίπτωση ίσης προσφοράς, γίνεται σε δεύτερο χρόνο σύγκριση της αξιολόγησης των παρόχων και μπαίνει στη λίστα εκείνος με την υψηλότερη αξιολόγηση.

Ολοκλήρωση Δημοπρασίας (completeAuction)

Η μέθοδος αυτή καλείται από τον πελάτη, όταν αυτός αποφασίσει ότι θέλει να ολοκληρώσει τη δημοπρασία. Αυτό μπορεί να γίνει ακόμα και αν δεν έχει συμπληρωθεί ο αριθμός παρόχων που ορίζεται στο πεδίο providersNumber. Η μέθοδος αυτή καλείται με μοναδικό όρισμα το μοναδικό αναγνωριστικό της δημοπρασίας (id), και έχει ως περιορισμό ότι μπορεί να κληθεί μόνο από τη διεύθυνση του πελάτη, όπως αυτή είναι αποθηκευμένη στο πεδίο client της δημοπρασίας.

Η λογική της μεθόδου είναι πολύ απλή, αφού απλά θέτει το πεδίο status σε 2 (COMPLETED) και το endTime στην τρέχουσα ώρα.

Ανάκτηση Στοιχείων Δημοπρασίας (getAuctionInfo)

Η μέθοδος αυτή, έχει ως σκοπό να επιστρέφει τα στοιχεία μιας δημοπρασίας για το id που δίνεται ως όρισμα κατά την κλήση. Τα στοιχεία που επιστρέφει είναι τα εξής:

- Η λίστα διευθύνσεων των παρόχων.
- Η διεύθυνση του πελάτη.
- Η λίστα τιμών που έχουν προσφέρει οι πάροχοι.
- Ο επιθυμητός αριθμός παρόχων.
- Η ώρα εκκίνησης της δημοπρασίας.
- Η ώρα ολοκλήρωσης της δημοπρασίας.
- Το μέγεθος του αρχείου προς αποθήκευση.

- Το collateralAmount.
- Η κατάσταση στην οποία βρίσκεται η δημοπρασία (ONGOING ή COMPLETED).

4.5 Συμβόλαιο

Με την επιτυχημένη ολοκλήρωση μιας δημοπρασίας, δημιουργείται ένα συμβόλαιο μεταξύ του πελάτη που ξεκίνησε τη δημοπρασία και του παρόχου με την καλύτερη προσφορά. Όπως και στην περίπτωση της δημοπρασίας, στην πρώτη μορφή της η εφαρμογή δέχεται μόνο ένα χρήστη σαν πάροχο, αλλά οι δομές δεδομένων των smart contracts που αναπτύχθηκαν μπορούν να υποστηρίξουν περισσότερους από έναν παρόχους.

Ο σκοπός της οντότητας του συμβολαίου είναι να αποθηκεύει όλες τις πληροφορίες που χρειάζονται για τη σχέση που δημιουργείται μεταξύ πελάτη και παρόχου. Για να μπορέσει αυτό να επιτευχθεί, η οντότητα του συμβολαίου έχει τα εξής πεδία:

- **provider** – Λίστα με τις δεκαεξαδικές διευθύνσεις όλων των παρόχων που έχουν κερδίσει τη δημοπρασία και θα αποθηκεύσουν το αρχείο. Στην πρώτη φάση της εφαρμογής, η λίστα αυτή περιέχει το πολύ έναν πάροχο.
- **client** – Η δεκαεξαδική διεύθυνση του πελάτη του οποίου το αρχείο θα αποθηκευθεί.
- **rootHash** – Το hash του root κόμβου το δέντρου Merkle που δημιουργείται από το αρχείο προς αποθήκευση. Χρησιμοποιείται για τις ανάγκες του Proof of Retrievability.
- **collateralAmount** – Το collateralAmount αποτελεί ένα ποσό, εκφρασμένο σε wei, που θα πρέπει να αποσταλλεί στη διεύθυνση του αντίστοιχου smart contract ώστε να μπορέσει να εκκινήσει το συμβόλαιο μεταξύ πελάτη και παρόχου. Το collateralAmount χρησιμοποιείται ως δικλείδα ασφαλείας σε περίπτωση που μία από της δύο πλευρές του συμβολαίου (πελάτης ή πάροχος) αθετήσει τους όρους του.
- **price** – Η τιμή την οποία θα πληρώνει ο πελάτης ανά μήνα, ώστε οι πάροχοι να αποθηκεύουν το αρχείο του. Η τιμή διαμορφώνεται ως το άθροισμα των τιμών που προσέφεραν οι πάροχοι. Το ποσό της τιμής, μοιράζεται ισομερώς στους παρόχους κατά την πληρωμή του συμβολαίου.
- **startTime** – Η ώρα εκκίνησης του συμβολαίου.
- **state** – Η κατάσταση στην οποία βρίσκεται το συμβόλαιο. Παίρνει τέσσερις τιμές, 1, 2, 3 και 4.
  - ACTIVE : Αντιστοιχεί στο 1 και δηλώνει ότι το συμβόλαιο είναι ενεργό.
  - INVALID : Αντιστοιχεί στο 2 και δηλώνει ότι το συμβόλαιο δεν έχει ολοκληρωθεί, αλλά έχει συμβεί κάποιο σφάλμα και δεν είναι πλέον έγκυρο.

– CANCELED : Αντιστοιχεί στο 3 και δηλώνει ότι κάποιο από τα δύο μέρη (πελάτης ή πάροχος) έχει ενεργήσει κακόβουλα και αυτό έχει αποδειχθεί on-chain.

– COMPLETED : Αντιστοιχεί στο 4 και δηλώνει ότι το συμβόλαιο έχει ολοκληρωθεί κοινή συνεναίσει.

Η σημασία των κωδικών του state θα φανεί καλύτερα παρακάτω.

```
pragma solidity ^0.5.0;

contract Contract{

    address payable[] private provider;
    address payable private client;
    bytes32 private rootHash;
    uint private collateralAmount; //collateralAmount is stored in wei (10^-19 eth)
    uint private price; //price is stored in wei (10^-19 eth)
    uint private startTime; //startTime is initiated to now in the init function. It is caclulated in seconds since the last Epoch.
    uint8 private state; //state of the contract
    //State Constants
    uint8 constant public ACTIVE = 1;
    uint8 constant public INVALID = 2;
    uint8 constant public CANCELED = 3;
    uint8 constant public COMPLETED = 4;

    constructor(address payable[] memory _provider, address payable _client, bytes32 _rootHash, uint _collateralAmount, uint _price)
    public
    {
        provider = _provider;
        client = _client;
        rootHash = _rootHash;
        collateralAmount = _collateralAmount;
        price = _price;
        startTime = now;
        state = ACTIVE;
    }

```

Σχήμα 4.4: Κώδικας για την οντότητα του Συμβολαίου.

4.6 Contract Executor

Ομοίως με την οντότητα του auction executor για τη δημοπρασία, έτσι και για το συμβόλαιο υπάρχει η οντότητα του contract executor με σκοπό να ορίζεται ο κύκλος ζωής του. Για να επιτευχθεί αυτό ο contract executor έχει τα εξής πεδία:

- **id** – Μοναδικό αναγνωριστικό για κάθε συμβόλαιο, δημιουργείται, ανατίθεται σε κάθε συμβόλαιο και αποθηκεύεται στον contractExecutor. Αποτελεί, ουσιαστικά, ένα sequence αριθμών που ξεκινά από το 1 και αυξάνεται κατά ένα με κάθε νέο συμβόλαιο. Ίδια λογική με το αντίστοιχο πεδίο id της δημοπρασίας.

- **contracts** – Mapping που αποθηκεύει την αντιστοιχία του id με το instance του συμβολαίου στο οποίο αντιστοιχεί. Ίδια λογική με το αντίστοιχο πεδίο auctions της δημοπρασίας.

- **paymentTime** – Mapping που αποθηκεύει την αντιστοιχία του id με το χρόνο στον οποίο πληρώθηκε τελευταία φορά το αντίστοιχο συμβόλαιο.

- **leafToVerify** – Mapping που αποθηκεύει την αντιστοιχία του id με το φύλλο του δέντρου Merkle για το οποίο ο πελάτης έχει ζητήσει Proof. Χρησιμοποιείται κατά τη διαδικασία του verification για το Proof of Retrievability.

- **verifications** – Mapping που αποθηκεύει την αντιστοιχία του id με το χρόνο στον οποίο εκτελέστηκε για τελευταία φορά η τεχνική του Proof of Retrievability για το συγκεκριμένο συμβόλαιο.
- **ratings** – Αποτελεί ένα instance της οντότητας του ProviderRating, στο οποίο αποθηκεύονται όλες οι αξιολογήσεις για τους παρόχους.

Όπως και ο auction executor, έτσι και ο contract executor χρησιμοποιείται μέσω κάποιων μεθόδων/διεπαφών. Αυτές, δίνουν στους καλούντες τη δυνατότητα να δημιουργήσουν, να πληρώσουν, να τρέξουν Proof of Retrievability και να ολοκληρώσουν κάποιο συμβόλαιο. Οι μέθοδοι αυτές είναι οι εξής:

Εκκίνηση Συμβολαίου (initContract)

Η μέθοδος αυτή καλείται από κάποιο χρήστη, ώστε να δημιουργήσει ένα νέο συμβόλαιο. Καλείται είτε μετά την ολοκλήρωση μιας δημοπρασίας είτε άμεσα από κάποιον χρήστη, εφόσον υπάρχει κάποια συνεννόηση off-chain μεταξύ χρηστών. Παίρνει σαν ορίσματα τα εξής:

- **provider** – Λίστα με τις δεκαεξαδικές μοναδικές διευθύνσεις των χρηστών οι οποίοι θα λειτουργήσουν ως πάροχοι για το συγκεκριμένο συμβόλαιο.
- **client** – Η δεκαεξαδική μοναδική διεύθυνση του πελάτη.
- **rootHash** – Το hash του root κόμβου το δέντρου Merkle που δημιουργείται από το αρχείο προς αποθήκευση.
- **collateralAmount** – Το ποσό που θα κατατεθεί στο συμβόλαιο με την εκκίνησή του και θα χρησιμοποιηθεί ως δικλείδα ασφαλείας, όπως αναφέρθηκε παραπάνω.
- **price** – Η τιμή που έχει συμφωνηθεί ότι θα πληρώνει ο χρήστης κάθε μήνα για την αποθήκευση του αρχείου του.

Δίνοντας αυτά τα στοιχεία, η μέθοδος καλεί τον constructor της οντότητας Contract και δημιουργεί ένα νέο instance συμβολαίου. Παράλληλα, εκτελεί και κάποιες επιπλέον ενέργειες:

- Ελέγχει αν κάποιος από τους παρόχους στη λίστα providers δε μπορεί να συμμετάσχει λόγω μηδενικής αξιολόγησης. Σε περίπτωση που υπάρχει, το συμβόλαιο δε δημιουργείται.
- Θέτει τον χρόνο του τελευταίου verification για το αρχείο στην τρέχουσα ώρα.
- Θέτει τον χρόνο της τελευταίας πληρωμής για το συμβόλαιο στην τρέχουσα ώρα.

- Αποθηκεύει το instance του συμβολαίου που δημιουργήθηκε στο mapping contracts με το τρέχον id. Έπειτα αυξάνει το id κατά ένα.
- Ελέγχει αν μαζί με την κλήση έχει σταλεί το συμφωνημένο και απαραίτητο collateralAmount σε wei. Σε περίπτωση που δεν έχει σταλεί, το συμβόλαιο δε δημιουργείται.

Πληρωμή Συμβολαίου (pay)

Η μέθοδος αυτή, έχει σκοπό να εκτελεί τη διαδικασία της πληρωμής του συμβολαίου με το αντίστοιχο id το οποίο δίνεται σαν όρισμα κατά την κλήση της μεθόδου. Λόγω της κρισιμότητας της μεθόδου, γίνονται ενδελεχείς έλεγχοι ώστε να ολοκληρωθεί με επιτυχία η πληρωμή. Αυτοί είναι οι εξής:

- Η μέθοδος μπορεί να κληθεί μόνο από τη δεκαεξαδική διεύθυνση του πελάτη.
- Το συμβόλαιο θα πρέπει να είναι σε state "active" ή "invalid"
- Η πληρωμή μπορεί να γίνει μόνο 1 μήνα μετά την τελευταία πληρωμή.
- Η κλήση θα πρέπει να συνοδεύεται από τη συμφωνημένη τιμή, όπως αυτή φαίνεται στο πεδίο price του συμβολαίου.

Εφόσον οι έλεγχοι που αναφέρθηκαν ολοκληρωθούν με επιτυχία, το ποσό που στάλθηκε μοιράζεται ισομερώς στους παρόχους, σύμφωνα με τις διευθύνσεις που υπάρχουν στη λίστα του πεδίου provider. Εφόσον όλες οι συναλλαγές προς τους παρόχους πετύχουν, η διαδικασία ολοκληρώνεται επιτυχώς. Σε διαφορετική περίπτωση, η διαδικασία διακόπτεται, και το state του συμβολαίου γίνεται "invalid".

Ουσιαστικά, ο πελάτης κάνει την πληρωμή στο smart contract, το οποίο στη συνέχεια διαμοιράζει το αντίστοιχο ποσό στον κάθε πάροχο. Τα smart contracts, έχουν τη δυνατότητα να δέχονται και να εκτελούν συναλλαγές ως κανονικοί χρήστες του Blockchain.

Τερματισμός Συμβολαίου (terminateMutual, terminateCulpClient, terminateCulpProvider)

Για τον τερματισμό ενός συμβολαίου υπάρχουν 3 διαφορετικές μέθοδοι. Και οι 3 μέθοδοι καλούνται με όρισμα το μοναδικό αναγνωριστικό του συμβολαίου (id). Αναλυτικότερα:

- **TerminateMutual** - Καλείται σε περίπτωση που το συμβόλαιο τερματίζεται κοινή συνεναίσει. Για να μπορέσει να ολοκληρωθεί ένα συμβόλαιο επιτυχώς με αυτή τη μέθοδο θα πρέπει:
  - Να έχει πληρωθεί το συμβόλαιο μέσα στις τελευταίες 15 μέρες (μισός μήνας), ώστε να εξασφαλιστεί ότι ο πελάτης έχει καλύψει τις υποχρεώσεις του.

– Να έχει αποδείξει ο πάροχος ότι το αρχείο είναι ανακτήσιμο μέσω του Proof of Retrievability τις τελευταίες 15 μέρες, ώστε να αποδείξει και αυτός ότι έχει καλύψει τις υποχρεώσεις του.

Στην περίπτωση που τα παραπάνω ισχύουν, το συμβόλαιο τερματίζεται επιτυχώς, και το state του αλλάζει σε completed. Φυσικά, σε όλους τους συμμετέχοντες στο συμβόλαιο (πελάτη και παρόχους) επιστρέφεται ισομερώς το collateralAmount που είχαν δώσει κατά την εκκίνηση του συμβολαίου.

- **TerminateCulpClient** - Η μέθοδος αυτή καλείται από κάποιον από τους παρόχους του συβολαίου, και το τερματίζει με υπαιτιότητα του πελάτη. Για να μπορέσει να τερματιστεί με αυτόν τον τρόπο το συμβόλαιο, θα πρέπει ο πελάτης να έχει να πληρώσει το συμβόλαιο περισσότερο από ένα μήνα πριν την τρέχουσα ημερομηνία. Εφόσον αυτό ισχύει, το συμβόλαιο ολοκληρώνεται και το state του αλλάζει σε canceled. Στη συγκεκριμένη περίπτωση ο πελάτης δεν παίρνει πίσω το ποσό που του αντιστοιχεί από το collateralAmount, σε αντίθεση με τους παρόχους, που το λαμβάνουν κανονικά.

- **TerminateCulpProvider** - Η μέθοδος αυτή καλείται είτε απευθείας από τον πελάτη, είτε μετά από κάποια αποτυχημένη εκτέλεση του Proof of Retrievability. Στην περίπτωση αυτή, και πάλι το συμβόλαιο τερματίζεται με state canceled, αλλά αυτή τη φορά το collateralAmount επιστρέφεται μόνο στον πελάτη και όχι στον πάροχο. Η συγκεκριμένη μέθοδος έχει αναπτυχθεί μόνο για την περίπτωση του μοναδικού παρόχου.

Σε αυτές τις 3 μεθόδους μεταβάλλεται ανάλογα και η αξιολόγηση του παρόχου. Στην περίπτωση που το συμβόλαιο ολοκληρώνεται συναινετικά αλλά και στην περίπτωση που είναι υπάιτιος για τη διακοπή του ο πελάτης, η αξιολόγηση του παρόχου αυξάνεται κατά ένα, αφού έχει καλύψει τις υποχρεώσεις του. Στην αντίθετη περίπτωση, που δηλαδή ο πάροχος δε μπορεί να αποδείξει την ανακτησιμότητα του αρχείου, η αξιολόγησή του μειώνεται κατά ένα.

Επαλήθευση Ανακτησιμότητας Αρχείου (getVerification, getLeaf, setLeaf)

Η τελευταία και ίσως σημαντικότερη μέθοδος της οντότητας του contractExecutor είναι αυτή που εκτελεί τη λογική του Proof of Retrievability. Πιο συγκεκριμένα, η μέθοδος getVerification καλείται από τον πάροχο με δύο ορίσματα:

- Το μοναδικό αναγνωριστικό id του συμβολαίου.

- Ένα πίνακα bytes, που αποτελεί το proof, την απόδειξη, δηλαδή, που δίνει ο πάροχος (διαδρομή στο δέντρο Merkle που οδηγεί στο root hash όπως έχει προαναφερθεί)

ότι το αρχείο είναι ανακτήσιμο.[3]

Για να μπορέσει ο πάροχος να δημιουργήσει την απόδειξη για την ανακτησιμότητα, πρέπει να ξέρει για ποιό φύλλο του δέντρου θέλει ο πελάτης να λάβει απόδειξη. Αυτό γίνεται γνωστό μέσω της δομής leafToVerify, η οποία περιέχει το συγκεκριμένο φύλλο, σε κρυπτογραφημένη μορφή. Η τιμή του φύλλου τίθεται από τον πελάτη μέσω της μεθόδου setLeaf, που καλείται με ορίσματα το id του συμβολαίου και το ίδιο το φύλλο, ενώ ο πάροχος μπορεί να λάβει το φύλλο μέσω της μεθόδου getLeaf, που καλείται με όρισμα μόνο το id. Σημειώνεται, ότι η μέθοδος setLeaf κάνει throw ένα event, το οποίο ¨πιάνει' η διεπαφή χρήστη, που αναπτύχθηκε στα πλαίσια της εργασίας, και ενημερώνει τον πάροχο ότι πρέπει να προχωρήσει σε απόδειξη ανακτησιμότητας. Κατά την κλήση της, η getVerification καλεί με τη σειρά της τη μέθοδο verify της οντότητας storage proof που αναλύεται παρακάτω. Αν αυτή η μέθοδος γυρίσει true, σημαίνει ότι η απόδειξη του παρόχου είναι δεκτή, και το αρχείο αποδεικνύεται ανακτήσιμο. Τότε τίθεται η τρέχουσα ώρα στο πεδίο verifications του συγκεκριμένου συμβολαίου.

Σε περίπτωση που η verify επιστρέψει false, αυτό σημαίνει ότι το αρχείο δεν είναι ανακτήσιμο, και το συμβόλαιο τερματίζεται με υπαίτιο τον πάροχο, αφού καλείται η μέθοδος termCulpProvider, που έχει αναλυθεί παραπάνω.

Η προκαθορισμένη τιμή στη δομή leafToVerify για κάθε συμβόλαιο είναι η κενή συμβολοσειρά. Σε περίπτωση που διενεργείται μια επιτυχημένη απόδειξη, η τιμή στο πεδίο επανέρχεται στην κενή συμβολοσειρά, εώς ότου ο πελάτης θέσει ξανά κάποια τιμή φύλλου για απόδειξη.

4.7 Storage Proof

Η τελευταία, αλλά πολύ σημαντική οντότητα που αναπτύχθηκε, είναι αυτή του storage proof. Σκοπός της συγκεκριμένης οντότητας είναι να εκτελεί τη λογική του Proof of Retrievability. Πιο συγκεκριμένα, η συγκεκριμένη οντότητα έχει μόλις μία μέθοδο, τη verify, η οποία μπορεί να κληθεί μόνο από το έτερο smart contract, τον contract executor, σε περίπτωση που κληθεί η μέθοδος getVerification του τελευταίου.

Η μέθοδος verify του storage proof έχει τα εξής ορίσματα:

- **root** - Το root hash του δέντρου Merkle που προκύπτει από το αρχείο προς αποθήκευση.
- **leaf** - Ένα string που περιέχει το φύλλο του δέντρου Merkle για το οποίο πρέπει να τρέξει η διαδικασία του proof.
- **proof** - Ένας πίνακας από bytes που περιέχει τα δεδομένα που χρειάζεται να δώσει ο πάροχος για να αποδείξει την ανακτησιμότητα του αρχείου. Όπως αναφέρθηκε και

[3]Σημειώνεται, ότι η απόδειξη που πρέπει να δώσει ο πάροχος στη μέθοδο, πρέπει να δημιουργηθεί εκτός Blockchain.

προηγουμένως, η απόδειξη αυτή πρέπει να περιέχει μία διαδρομή του δέντρου Merkle, η οποία καταλήγει στο root hash.

Η λογική της μεθόδου του verify είναι ακριβώς αυτή που περιγράφεται στο Κεφάλαιο 2.4 και φαίνεται στο Σχήμα 2.7. Με δεδομένο το φύλλο για το οποίο καλείται να καταστρώσει την απόδειξη, το root hash αλλά και την απόδειξη αυτή κάθε αυτή, η μέθοδος εκτελεί βήμα βήμα τους κατακερματισμούς (hashes) που απαιτούνται, ώστε να φτάσει στην κορυφή του δέντρου, που πρέπει να ταυτίζεται με το όρισμα root. Εφόσον όντως ταυτίζεται, η συνάρτηση επιστρέφει true, διαφορετικά, επιστρέφει false.

Κεφάλαιο 5

Περιπτώσεις Χρήσης Εφαρμογής

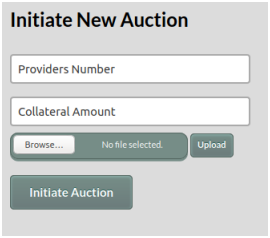
5.1 Εισαγωγή

Στο παρόν κεφάλαιο, θα παρουσιαστούν διάφορες περιπτώσεις χρήσης (use cases) της αποκεντρωμένης εφαρμογής που αναπτύχθηκε.

Πιο συγκεκριμένα, θα παρουσιαστεί, τμηματικά, με αντίστοιχα διαγράμματα (use case diagrams) όλη η ροή που δημιουργείται από την εκκίνηση της δημοπρασίας, μέχρι και τον τερματισμό του αντίστοιχου συμβολαίου.

5.2 Ροή Δημοπρασίας

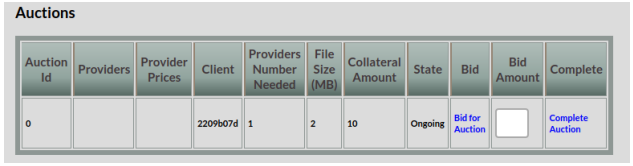
Όπως έχει προαναφερθεί, εκείνος που ορίζει τον κύκλο ζωής μίας δημοπρασίας είναι ο πελάτης. Ένας συμμετέχοντας του Ethereum δικτύου στο οποίο είναι ανεπτυγμένη η αποκεντρωμένη εφαρμογή, μπορεί να δημιουργήσει ένα νέο instance δημοπρασίας μέσω της διεπαφής χρήστη, δίνοντας ως input τον αριθμό των παρόχων που θέλει να αποθηκεύσουν το αρχείο, το collateral amount, αλλά και το ίδιο το αρχείο. Έτσι, δημιουργείται μια νέα δημοπρασία, στην οποία πελάτης, είναι ο χρήστης εκείνος που την εκκίνησε.



Σχήμα 5.1: Διεπαφή χρήστη για τη δημιουργία δημοπρασίας.

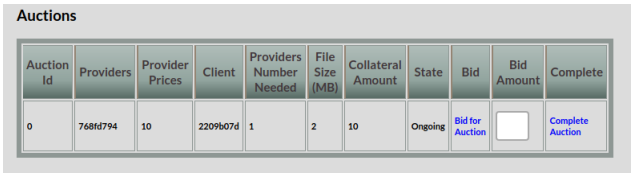
Στη συνέχεια, άλλοι συμμετέχοντες στο δίκτυο ειδοποιούνται για την εκκίνηση μιας νέας δημοπρασίας και έχουν τη δυνατότητα να καταθέσουν την προσφορά τους. Με τον τρόπο αυτό, παίρνουν το ρόλο του παρόχου για τη συγκεκριμένη δημοπρασία.

Εφόσον στην πρώτη φάση της εφαρμογής μπορεί να συμμετέχει το πολύ ένας πάροχος



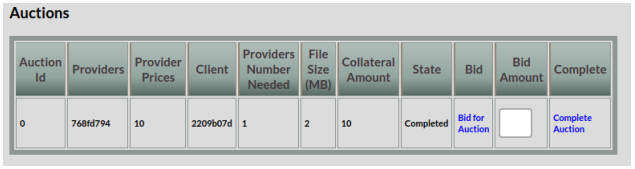
Σχήμα 5.2: Διεπαφή χρήστη με τη λίστα των δημοπρασιών χωρίς προσφορές.

σε ένα συμβόλαιο, στη δημοπρασία κερδίζει πάντα εκείνος που κάνει την προσφορά με τη μικρότερη τιμή. Όταν ο πελάτης είναι ικανοποιημένος με την προσφορά, μπορεί να ολοκληρώσει τη δημοπρασία μέσω της διεπαφής χρήστη.



Σχήμα 5.3: Διεπαφή χρήστη με τη λίστα των δημοπρασιών με προσφορά.

Με την ολοκλήρωση της δημοπρασίας από τον πελάτη της, εκκινείται από το UI αυτόματη διαδικασία για τη δημιουργία συμβολαίου για την αποθήκευση του αρχείου, μεταξύ του πελάτη και του παρόχου-νικητή της δημοπρασίας.



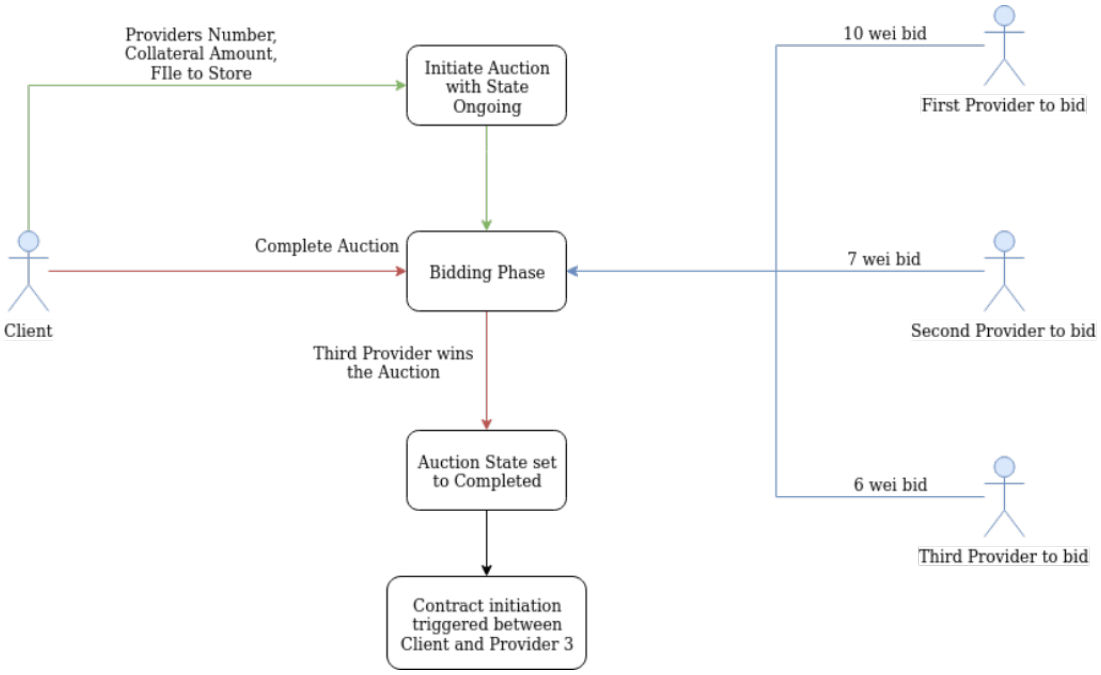
Σχήμα 5.4: Διεπαφή χρήστη με τη λίστα των δημοπρασιών μετά την ολοκλήρωση.

Η παραπάνω ροή φαίνεται και στο use case diagram του σχήματος 5.5.

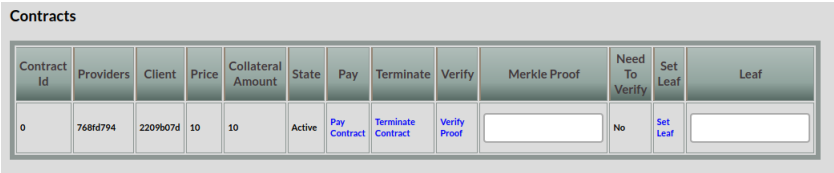
5.3 Εκκίνηση Συμβολαίου

Η εκκίνηση ενός νέου συμβολαίου είναι άμεσα συνδεδεμένη με την ολοκλήρωση της αντίστοιχης δημοπρασίας. Αφού ο πελάτης ολοκληρώσει τη δημοπρασία, ξεκινά η διαδικασία για την εκκίνηση ενός συμβολαίου για την αποθήκευση αρχείου. Το συμβόλαιο αυτό θα έχει στοιχεία για τον πελάτη, τον πάροχο, το αρχείο και το collateral amount, όπως αυτά έχουν δηλωθεί στην αντίστοιχη δημοπρασία.

Για να εκκινηθεί το συμβόλαιο, θα πρέπει να σταλεί στο smart contract που διαχειρίζεται τα συμβόλαια, το ακριβές collateral amount, καθώς και να γίνει έλεγχος για το αν ο πάροχος που έχει κερδίσει τη δημοπρασία έχει αποκλειστεί από τη διαδικασία μέσω της λογικής της αξιολόγησης στην οποία έχει γίνει αναφορά προηγουμένως. Ο χρήστης που δημιουργεί το συμβόλαιο είναι ουσιαστικά ο πελάτης, αφού εκείνος ξεκινά τη διαδικασία της δημιουργίας του, όταν ολοκληρώνει τη δημοπρασία του (Σχήμα 5.6).



Σχήμα 5.5: Διάγραμμα για τον κύκλο ζωής της διεργασίας.



Σχήμα 5.6: Διεπαφή χρήστη με τη λίστα των συμβολαίων.

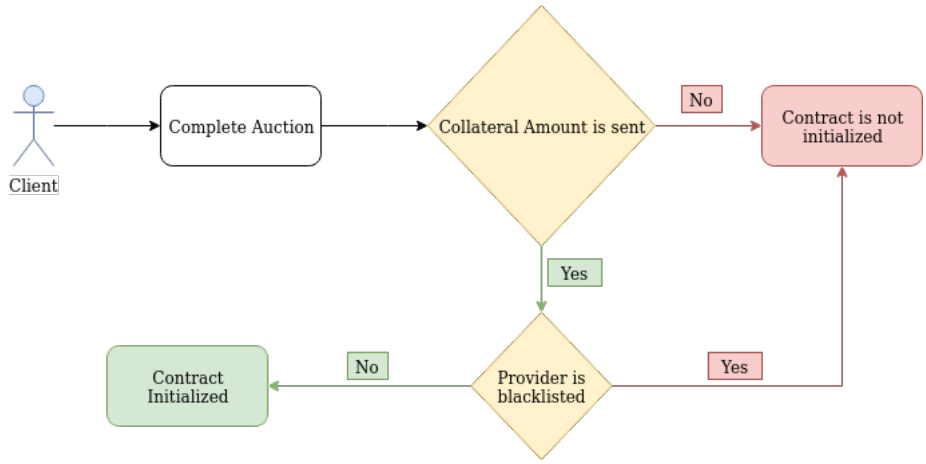
Η παραπάνω διαδικασία για την εκκίνηση ενός συμβολαίου, φαίνεται και στο use case diagram του σχήματος 5.7.

5.4 Πληρωμή Συμβολαίου

Για την πληρωμή ενός συμβολαίου, υπεύθυνος είναι ο πελάτης. Κάθε ένα μήνα, γίνεται πληρωμή από τον πελάτη προς το smart contract που διαχειρίζεται τα συμβόλαια, το οποίο με τη σειρά του προωθεί την πληρωμή στον αντίστοιχο πάροχο. Το smart contract έχει τη δυνατότητα να διαχειρίζεται και πληρωμές προς πολλαπλούς παρόχους, που, βέβαια, δε χρησιμοποιείται στην παρούσα φάση της εφαρμογής.

Όπως φαίνεται και στο σχήμα 5.6, ο πελάτης έχει τη δυνατότητα να πληρώσει το συμβόλαιό του μέσω της διεπαφής χρήστη της εφαρμογής. Ο σύνδεσμος για πληρωμή (Pay Contract) μπορεί να πατηθεί από οποιονδήποτε χρήστη της εφαρμογής, αλλά το transaction σε επίπεδο Blockchain θα επιτύχει μόνο αν ο καλών είναι ο πελάτης του συγκεκριμένου συμβολαίου.

Αφού γίνει η κλήση, για να ολοκληρωθεί με επιτυχία η πληρωμή, το smart contract



Σχήμα 5.7: Διάγραμμα ροής για την εκκίνηση νέου συμβολαίου.

ελέγχει ότι ο πελάτης έχει στείλει το σωστό ποσό σε wei (την τιμή, δηλαδή, που συμφωνήθηκε κατά τη δημοπρασία), αν έχει περάσει τουλάχιστον ένας μήνας από την τελευταία πληρωμή και αν το συμβόλαιο είναι ενεργό. Στην περίπτωση που τα παραπάνω ισχύουν, η πληρωμή ολοκληρώνεται επιτυχώς και αποθηκεύεται στο smart contract η χρονική στιγμή αυτή.

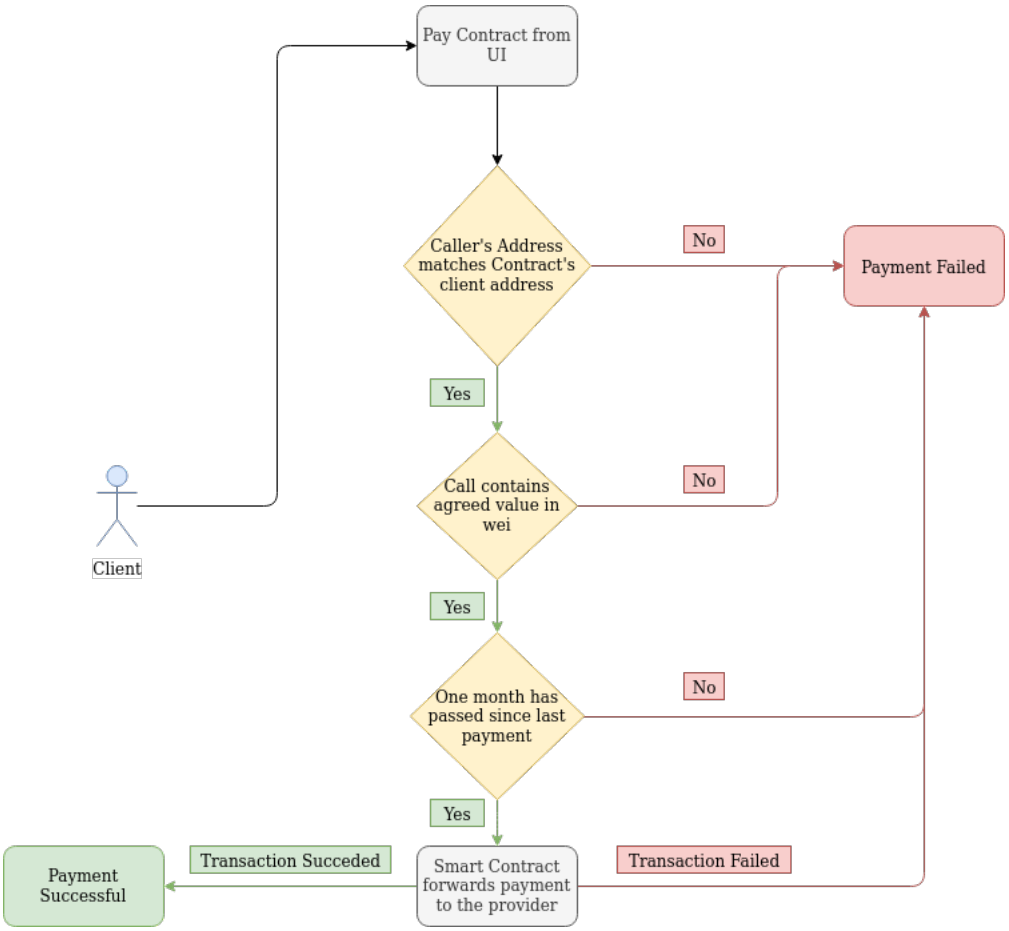
Τα παραπάνω φαίνονται και στο use case diagram του σχήματος 5.8.

5.5 Επιβεβαίωση Ανακτησιμότητας Αρχείου

Ο πελάτης, έχει τη δυνατότητα να ζητήσει από τον πάροχο να επιβεβαιώσει την ανακτησιμότητα του αρχείου που έχει δοθεί για αποθήκευση. Η διαδικασία αυτή είναι υποχρεωτική για τον πάροχο, και πρέπει να πραγματοποιείται τουλάχιστον μία φορά το μήνα. Σε περίπτωση που ένας πάροχος δεν έχει επιβεβαιώσει την ανακτησιμότητα του αρχείου για πάνω από ένα μήνα, ο πελάτης έχει το δικαίωμα να διακόψει το συμβόλαιο με υπαίτιο τον πάροχο.

Η διαδικασία επιβεβαίωσης της ανακτησιμότητας εκκινείται από τον πελάτη, μέσω της διεπαφής χρήστη που δίνει η εφαρμογή. Ο πελάτης θέτει ένα φύλλο από το δέντρο Merkle που δημιουργείται από το αρχείο (Σχήμα 5.9), και ο πάροχος, αφού ενημερώνεται από τη διεπαφή χρήστη (Σχήμα 5.10), καλείται να καταστρώσει την απαραίτητη απόδειξη (Proof of Retrievability) για το συγκεκριμένο φύλλο, και να την εισάγει στο κατάλληλο πεδίο της διεπαφής (Σχήμα 5.11). Έπειτα, το smart contract του συμβολαίου ελέγχει την απόδειξη για την ορθότητά της και ανάλογα με το αποτέλεσμα, είτε ανανεώνει το συμβόλαιο (αν η απόδειξη είναι έγκυρη), είτε το διακόπτει με υπαιτιότητα του παρόχου (αν η απόδειξη ήταν άκυρη).

Η ροή της διαδικασίας της επιβεβαίωσης της ανακτησιμότητας του αρχείου φαίνεται και στο use case diagram του σχήματος 5.12.

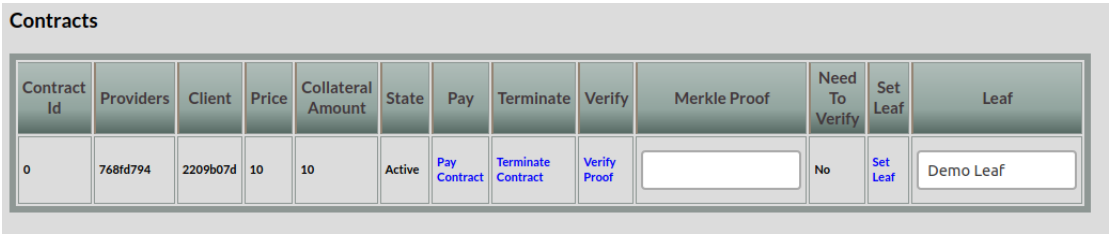


Σχήμα 5.8: Διάγραμμα ροής για την πληρωμή συμβολαίου.

5.6 Τερματισμός συμβολαίου

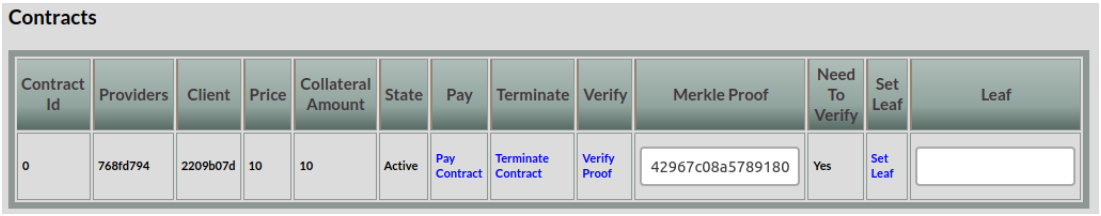
Ο τερματισμός του συμβολαίου μπορεί να εκκινηθεί και από τους δύο συμμετέχοντες στο συμβόλαιο (πελάτης ή πάροχος), μέσω της διεπαφής χρήστη, από την επιλογή Terminate Contract. Ένα συμβόλαιο μπορεί να τερματιστεί με τρεις διαφορετικούς τρόπους:

- **Τερματισμός με υπαιτιότητα του πελάτη -** Για να μπορέσει να τερματιστεί το συμβόλαιο με υπαιτιότητα του πελάτη, θα πρέπει ο πάροχος να έχει καταθέσει έγκυρη απόδειξη ανακτησιμότητας αρχείου μέσα στον τελευταίο μήνα, και ο πελάτης να **μην** έχει καταθέσει πληρωμή για το ίδιο χρονικό διάστημα. Στην περίπτωση



Σχήμα 5.9: Παράδειγμα χρήσης διεπαφής από τον πελάτη για να θέσει φύλλο.

Σχήμα 5.10: Παράδειγμα ενημέρωσης παρόχου για αίτηση επιβεβαίωσης ανάκτησης αρχείου.



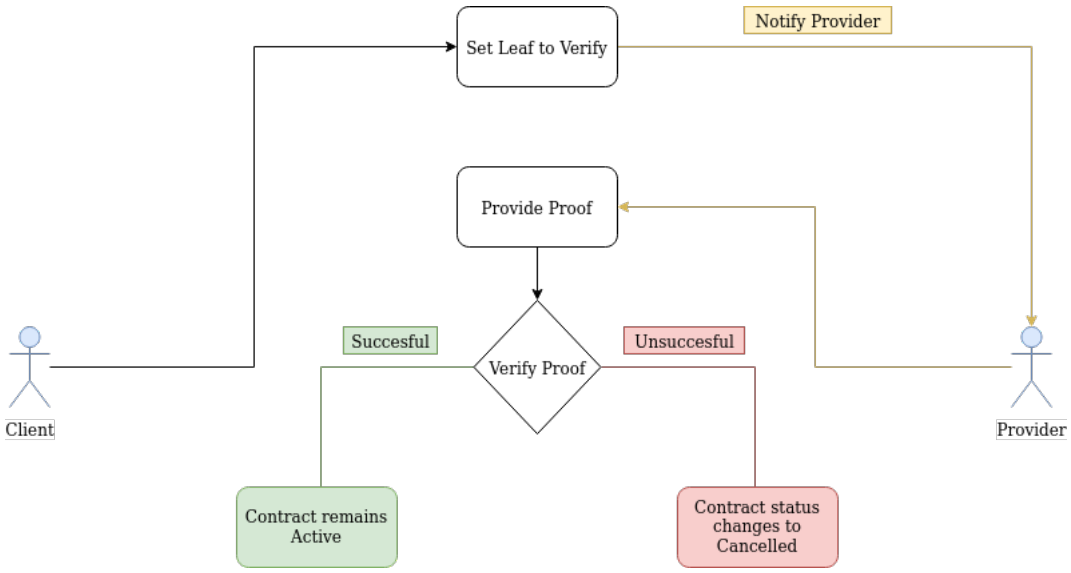
Σχήμα 5.11: Παράδειγμα εισαγωγής απόδειξης στη διεπαφή από πάροχο.

αυτή, το collateral amount επιστρέφεται στον πάροχο και το status του τίθεται σε Cancelled.

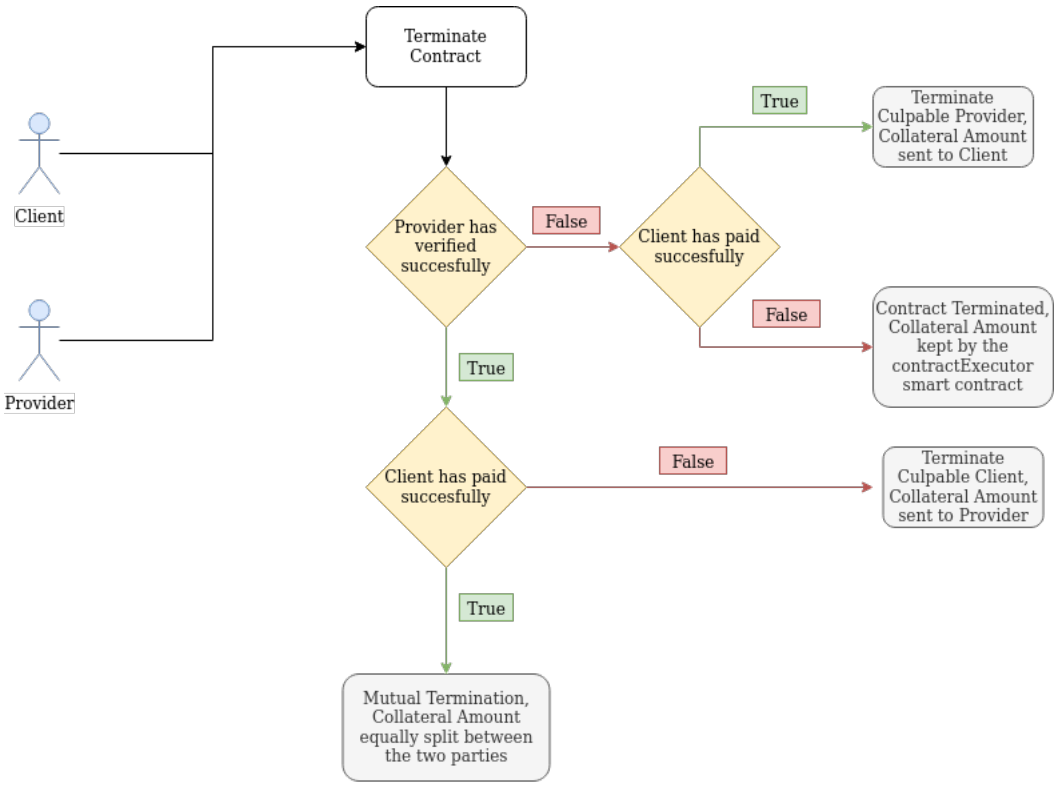
- **Τερματισμός με υπαιτιότητα του παρόχου -** Για να μπορέσει να τερματιστεί το συμβόλαιο με υπαιτιότητα του παρόχου, θα πρέπει ο πελάτης να έχει καταθέσει επιτυχημένη πληρωμή μέσα στον τελευταίο μήνα, και ο πάροχος να **μην** έχει καταθέσει έγκυρη απόδειξη ανακρησιμότητας για το ίδιο χρονικό διάστημα. Στην περίπτωση αυτή, το collateral amount επιστρέφεται στον πελάτη και το status του τίθεται σε Cancelled. Επίσης, το συμβόλαιο τερματίζεται με υπαιτιότητα του παρόχου σε περίπτωση κατάθεσης άκυρης απόδειξης ανακτησιμότητας.

- **Αμοιβαίος Τερματισμός -** Στην περίπτωση που τόσο ο πελάτης, όσο και ο πάροχος έχουν καλύψει τις υποχρεώσεις τους για τον τελευταίο μήνα το συμβόλαιο τερματίζεται αμοιβαία και το status του τίθεται σε Completed. Τότε, το collateral amount μοιράζεται ισόποσα στον πελάτη και τον πάροχο.

Αξίζει να σημειωθεί ότι η λογική για τον τρόπο με τον οποίο τερματίζεται το συμβόλαιο γίνεται μέσω του διαχειριστικού smart contract contractExecutor με τη βοήθεια του frontend κωδικα της διεπαφής χρήστη. Η ροή για τον τερματισμό του συμβολαίου φαίνεται και στο use case diagram του σχήματος 5.13.



Σχήμα 5.12: Διάγραμμα ροής για την απόδειξη ανακτησιμότητας αρχείου.



Note: Termination can be triggered either by the Client or the Provider

Σχήμα 5.13: Διάγραμμα ροής για τον τερματισμό του συμβολαίου.

Κεφάλαιο 6

Επίλογος

6.1 Συμπεράσματα

Σκοπός της διπλωματικής εργασίας ήταν η εξέταση των νέων τεχνολογιών αποκέντρωσης και η ανάπτυξη έξυπνων συμβολαίων για τη λειτουργία μίας αποκεντρωμένης εφαρμογής (DApp – Decentralized Application), η οποία θα μπορούσε να αποτελέσει κομμάτι του αποκεντρωμένου ιστού, όπως επίσης και η συνεισφορά έργου στην νεότευκτη και ταχύτατα αναπτυσσόμενη κοινότητα του αποκεντρωμένου ιστού.

Ειδικότερα, η συγκεκριμένη εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας, είχε σκοπό να δώσει μία αποκεντρωμένη λύση στους χρήστες του διαδικτύου που επιθυμούν να αποκτήσουν ή να διαθέσουν αποθηκευτικό χώρο. Η λογική της δημοπρασίας, κρατά το κόστος ανταγωνιστικό και μπορεί να διαταράξει την αγορά των μονοπολίων των μεγάλων εταιριών του κλάδου (Google, Amazon κ.λ.π.). Κάθε, ισότιμος, χρήστης του δικτύου, μπορεί να επιλέξει να συμμετάσχει είτε ως πελάτης, είτε ως πάροχος και να δημιουργήσει συμβόλαια για την αποθήκευση αρχείων, χωρίς την ανάγκη ύπαρξης κάποιας κεντρικής διαχειριστικής αρχής. Τέλος, η λογική της αξιολόγησης των χρηστών-παρόχων φιλτράρει τους συμμετέχοντες και κρατά εκτός εκείνους που λειτουργούν εις βάρος του δικτύου.

Προϊόν όλων αυτών των ιδεών, είναι η αποκεντρωμένη εφαρμογή που παρουσιάστηκε εκτενώς στα προηγούμενα κεφάλαια.

6.2 Μελλοντικές Επεκτάσεις

Η αποκεντρωμένη εφαρμογή που αναπτύχθηκε, αν και πλήρως λειτουργική, δε μπορεί να θεωρηθεί προϊόν έτοιμο για ευρεία χρήση. Μερικές επεκτάσεις τις οι οποίες θα την καταστήσουν πιο λειτουργική, αλλά και εύχρηστη, είναι οι εξής:

- **Διαμοιρασμός του αρχείου σε περισσότερους του ενός παρόχους** - Όπως έχει αναφερθεί και αρκετές φορές στην παρούσα διπλωματική, η εφαρμογή

έχει όλες εκείνες τις δομές δεδομένων, αλλά και λογική, που χρειάζονται για να υποστηρίξει περισσότερους του ενός παρόχους σε κάθε δημοπρασία και συμβόλαιο. Χρησιμοποιόντας τεχνικές κατανεμημένης αποθήκευσης, το αρχείο μπορεί να κατακερματίζεται και να μοιράζεται στους παρόχους, δημιουργώντας ταυτόχρονα πολλαπλά αντίγραφά του, ώστε να αποφευχθεί τυχόν απώλεια δεδομένων σε περίπτωση βλάβης ή εξόδου κάποιου χρήστη από το δίκτυο.

- **Διαμοιρασμός Επεξεργαστικής Ισχύος -** Ως λογικό επόμενο του διαμοιρασμού χώρου, η εφαρμογή θα μπορούσε να χρησιμοποιηθεί και ως εργαλείο για το διαμοιρασμό επεξεργαστικής ισχύος. Θα μπορούσε, λοιπόν, να δοθεί σε ένα χρήστη η δυνατότητα να ¨τρέξει' κώδικα πάνω στο δίκτυο που δημιουργείται από τους χρήστες του Ethereum Blockchain, με τη βοήθεια του Ethereum Virtual Machine.

- **Πλήρης απαλοιφή της ανάγκης ιστοτόπου για την λειτουργία της εφαρμογής -** Αναλόγως την κατεύθυνση που θα πάρει η εξέλιξη της τεχνολογίας του Ethereum Blockchain, μπορεί να καταστεί δυνατή η πλήρης κατάργηση της διεπαφής χρήστη της εφαρμογής, χωρίς να υπάρξει επίπτωση στην χρηστικότητά της και την φιλικότητα προς τον χρήστη. Έτσι η εφαρμογή θα μπορούσε να αποδεσμευτεί πλήρως από το μοντέλο αρχιτεκτονικής πελάτη-εξυπηρετητή και να υιοθετήσει πλήρως το κατανεμημένο μοντέλο, κάτι που θα πρέπει να είναι και ο στόχος της κάθε αποκεντωμένης εφαρμογής.

Βιβλιογραφία

[1] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System.* 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[2] *Bitcoin creator.* [Online]. Available: https://en.wikipedia.org/wiki/Satoshi_Nakamoto.

[3] U. W. Chohan, *The Double Spending Problem and Cryptocurrencies.* Dec. 2017. [Online]. Available: http://dx.doi.org/10.2139/ssrn.3090174.

[4] S. Haber and W. S. Stornetta, *How to Time-Stamp a Digital Document.* 1991. [Online]. Available: https://www.anf.es/pdf/Haber_Stornetta.pdf.

[5] D. Bayer, S. Haber, and W. S. Stornetta, *Improving the Efficiency and Reliability of Digital Time-Stamping.* 1993. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4613-9323-8_24.

[6] V. Buterin, *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.* Jul. 2015. [Online]. Available: http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf.

[7] *Asymmetric cryptography.* [Online]. Available: https://en.wikipedia.org/wiki/Public-key_cryptography.

[8] U. Naik and D. Shivalingaiah, *Comparative Study of Web 1.0, Web 2.0 and Web 3.0.* Feb. 2008. [Online]. Available: http://ir.inflibnet.ac.in/handle/1944/1285.

[9] *Dot com crash.* [Online]. Available: https://en.wikipedia.org/wiki/Dot-com_bubble.

[10] *Decentralised application.* [Online]. Available: https://en.wikipedia.org/wiki/Decentralized_application.

[11] *Peer to peer.* [Online]. Available: https://en.wikipedia.org/wiki/Peer-to-peer.

[12] *Ethereum.* [Online]. Available: https://en.wikipedia.org/wiki/Ethereum.

[13] *Abi.* [Online]. Available: https://solidity.readthedocs.io/en/develop/abi-spec.html.

[14] *Api*. [Online]. Available: https://en.wikipedia.org/wiki/Web_API.

[15] *Distributed ledger technology*. [Online]. Available: https://en.wikipedia.org/wiki/Distributed_ledger.

[16] K. D. Bowers, A. Juels, and A. Oprea, *Proofs of Retrievability: Theory and Implementation*. Jan. 2008. [Online]. Available: https://nds2.ccs.neu.edu/papers/PoR.pdf.

[17] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, *Proofs of ownership in remote storage systems*. Oct. 2011. [Online]. Available: https://dl.acm.org/citation.cfm?id=2046765.

[18] *Secure hash algorithms*. [Online]. Available: https://en.wikipedia.org/wiki/Secure_Hash_Algorithms.

[19] *Truffle*. [Online]. Available: https://www.trufflesuite.com/docs/truffle/overview.

[20] *Ganache*. [Online]. Available: https://www.trufflesuite.com/docs/ganache/overview.

[21] *Web3.js*. [Online]. Available: https://web3js.readthedocs.io/en/v1.2.4/.

[22] *Solidity*. [Online]. Available: https://solidity.readthedocs.io/en/v0.5.0/.

[23] *Metamask*. [Online]. Available: https://metamask.io/.

[24] *Node packet manager*. [Online]. Available: https://docs.npmjs.com/about-npm/.

[25] *Node.js*. [Online]. Available: https://nodejs.org/en/docs/.

[26] *Hypertext markup language*. [Online]. Available: https://en.wikipedia.org/wiki/HTML.

[27] *Cascading style sheets*. [Online]. Available: https://en.wikipedia.org/wiki/Cascading_Style_Sheets.

[28] *Javascript*. [Online]. Available: https://devdocs.io/javascript/.

[29] *Wei*. [Online]. Available: https://en.wikipedia.org/wiki/Ethereum#Ether.