Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων
Πληροφορικής

Ανάπτυξη έξυπνων συμβολαίων σε permissioned Blockchain πλατφόρμες

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεράσιμος Τσεκούρας

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Νοέμβριος 2025

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων
Πληροφορικής

Ανάπτυξη έξυπνων συμβολαίων σε permissioned Blockchain πλατφόρμες

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεράσιμος Τσεκούρας

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Νοεμβρίου 2025.

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Συμεών Παπαβασιλείου	Εμμανουήλ Βαρβαρίγος
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2025

.....

Γεράσιμος Τσεκούρας

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

Περίληψη

Το διαδίκτυο αποτελεί μια από τις σπουδαιότερες εφευρέσεις του περασμένου αιώνα. Ακολουθεί την αρχιτεκτονική του πελάτη-εξυπηρετητή, μια αρχιτεκτονική που απαιτεί εμπιστοσύνη από την πλευρά του χρήστη προς τον εξυπηρετητή. Αυτό θα αλλάξει σε μεγάλο βαθμό εξαιτίας των νέων τεχνολογιών που έχουν κάνει την εμφάνιση τους, μεταξύ των οποίων βρίσκονται το blockchain και το Διαδίκτυο των Πραγμάτων, τα οποία ενδέχεται να δώσουν στο διαδίκτυο έναν πιο αποκεντρωμένο χαρακτήρα.

Η τεχνολογία του blockchain δημιουργήθηκε το 2009, όμως δε σταματά να εξελίσσεται και έχει αποκτήσει πλέον ευρύ πεδίο εφαρμογών. Το blockchain είναι ένα ψηφιακό, κατανεμημένο, δημόσιο λογιστικό βιβλίο (ledger) στο οποίο καταγράφονται συναλλαγές με τρόπο αδιάβλητο και το οποίο υποστηρίζεται από ένα δίκτυο ομότιμων κόμβων (P2P), έχοντας εξ ορισμού αποκεντρωμένη λειτουργία. Εκτός από μέσο για τη λειτουργία κρυπτονομισμάτων, το Blockchain, μπορεί να αποτελέσει πυλώνα για τη δημιουργία και λειτουργία αποκεντρωμένων εφαρμογών, εφαρμογών δηλαδή που βασίζονται σε ένα κατανεμημένο δίκτυο ομότιμων κόμβων και όχι στους εξυπηρετητές κάποιου οργανισμού. Το Ethereum Blockchain, προσφέροντας πληθώρα δυνατοτήτων, χρησιμοποιείται ως πλατφόρμα αποκεντρωμένων εφαρμογων (Decentralised Apps), οι οποίες δε βασίζονται πλέον στο κλασικό μοντέλο πελάτη-εξυπηρετητή.

Το Quorum αποτελεί μια επέκταση του Ethereum, η οποία απευθύνεται περισσότερο σε επιχειρήσεις και προσφέρει δυνατότητες ιδιωτικότητας συναλλαγών και πιο δομημένη οργάνωση ενός ιδιωτικού Blockchain. Είναι βασισμένο στο Ethereum, χρησιμοποιεί το Ethereum Virtual Machine για την εκτέλεση κώδικα που βρίσκεται στα έξυπνα συμβόλαια (smart contracts) στο Blockchain. Επομένως, καλύπτει έτσι μια απο τις βασικές ελλείψεις άλλων blockchains, υποστηρίζοντας βέβαια και τη λειτουργία αποκεντρωμένων εφαρμογών.

Μια τέτοια εφαρμογή αποτελεί το αντικείμενο της παρούσας διπλωματικής. Η εφαρμογή, που τρέχει στο Quorum Blockchain μέσω των έξυπνων συμβολαίων, συνδέεται με μια βάση δεδομένων και δίνει πρόσβαση στους χρήστες μέσα από μια ιστοσελίδα. Πιο συγκεκριμένα, αποτελεί μια πλατφόρμα αγοραπωλησιών δεδομένων, η οποία λειτουργεί πάνω στο Quorum blockchain, παρέχοντας ταυτόχρονα ένα σύστημα αξιολόγησης στους χρήστες της.

Λέξεις κλειδιά

Ethereum, Blockchain, Quorum, Ιδιωτικότητα, Trust and Reputation System, Έξυπνα Συμβόλαια, Αποκεντρωμένες εφαρμογές - DApps, Web3, Διαδίκτυο των Πραγμάτων

Abstract

The internet is one of the greatest inventions of the last century. It follows a client-server architecture, an architecture that requires trust from the user to the server. But this is expected to change to a large extent due to new technologies that have made their appearance, among which are blockchain and the Internet of Things that may give the internet a more decentralised character.

The blockchain technology was created in 2009, but it has not stopped evolving and has now gained a wide scope of applications. The blockchain is a digital, distributed, public ledger in which transactions are recorded in a seamless manner and which is supported by a peer-to-peer (P2P) network, being decentralised by definition. Apart from being a medium for the operation of cryptocurrencies, Blockchain can be a pillar for the creation and operation of decentralised applications, i.e. applications based on a distributed network of peer nodes and not on the servers of an organisation. Ethereum Blockchain, offering a wealth of possibilities, is used as a platform for decentralised apps, which are no longer based on the classic client-server model.

Quorum is an extension of Ethereum, which is more targeted at enterprises and offers transaction privacy features and more structured organization of a private Blockchain. It is based on Ethereum, it uses the Ethereum Virtual Machine to execute code found in smart contracts on the Blockchain. It therefore fills one of the key shortcomings of other blockchains, supporting of course the operation of decentralised applications.

Such an application is the subject of this thesis. The application, which runs on the Quorum Blockchain through smart contracts, connects to a database and gives users access through a website. More specifically, it is a data buying and selling platform,which runs on the Quorum blockchain, while providing a rating system to its users.

Keywords

Ethereum, Blockchain, Quorum, Privacy, Trust and Reputation System, Smart Contracts, Decentralised Applications - DApps, Web3, Internet of Things

Ευχαριστίες

Για την έως τώρα σταδιοδρομία μου, θα ήθελα να ευχαριστήσω την οικογένεια μου, που είναι πάντα δίπλα μου και με στηρίζει διαρκώς. Επίσης, ευχαριστώ τους φίλους και συμφοιτητές μου για την για την αλληλοϋποστήριξη και την αλληλοβοήθεια.

Ευχαριστώ τους καθηγητές μου και ιδιαίτερα την κυρία Θεοδώρα Βαρβαρίγου για την πολύτιμη καθοδήγηση της, καθώς και για την ανάθεση μιας πολύ ενδιαφέρουσας διπλωματικής εργασίας. Επίσης θέλω να ευχαριστήσω τον διδάκτορα Γιώργο Παλαιοκρασσά, τον διδάκτορα Ευθύμιο Χονδρογιάννη και τον διδάκτορα ερευνητή Αντώνη Λίτκε για τη βοήθεια και την καθοδήγηση που μου πρόσφεραν.

Γεράσιμος Τσεκούρας
Αθήνα, Νοέμβριος 2025

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1 Εισαγωγή

Η σημερινή εποχή χαρακτηρίζεται ως η εποχή της πληροφορίας, καθώς η ανάπτυξη του διαδικτύου έχει αλλάξει ριζικά την καθημερινή ζωή των ανθρώπων, αλλά και την οικονομία. Το διαδίκτυο, προσφέροντας τη δυνατότητα της εύκολης και άμεσης μεταφοράς πληροφοριών, έχει γίνει ένα απαραίτητο εργαλείο για κάθε άνθρωπο, καθώς το χρησιμοποιεί, μεταξύ άλλων, για επικοινωνία, εκπαίδευση, ψυχαγωγία, αγορές και οικονομικές συναλλαγές. Η πάροδος του χρόνου και η έλευση νέων τεχνολογιών έχει αλλάξει σημαντικά το τρόπο λειτουργίας του χρηματοπιστωτικού συστήματος, καθώς το χρήμα έχει γίνει ψηφιακό. Αν και η πρόοδος που έχει γίνει είναι σημαντική, το μοντέλο αυτό απαιτεί εμπιστοσύνη σε μια κεντρική οντότητα, το τραπεζικό σύστημα.

Η δημιουργία του Bitcoin, το οποίο αποτελεί το πρώτο κρυπτονόμισμα, προσέφερε μια εναλλακτική επιλογή στο υπάρχον σύστημα. Το Bitcoin, όπως και κρυπτονομίσματα που ακολούθησαν, δεν ελέγχονται από κάποια κυβέρνηση, τράπεζα ή άλλο μεσάζοντα, αλλά η λειτουργία τους βασίζεται στην τεχνολογία του Blockchain και ελέγχεται από ένα σύνολο κρυπτογραφικών πρωτοκόλλων. Η έλευση του Ethereum, έδωσε τη δυνατότητα για τη δημιουργία αποκεντρωμένων εφαρμογών (DApp), των οποίων η λειτουργία είναι βασισμένη στη τεχνολογία του Blockchain. Επιπλέον, τα τελευταία χρόνια έχει παρουσιάσει μεγάλη ανάπτυξη ο τομέας του Διαδικτύου των Πραγμάτων (Internet of Things – IoT), το οποίο φέρνει επανάσταση στο διαδίκτυο, καθώς το επεκτείνει διασυνδέοντας φυσικές ή εικονικές συσκευές, οι οποίες έχουν τη δυνατότητα να ταυτοποιήσουν τον εαυτό τους στο δίκτυο. Η ύπαρξη της ανάγκης ενός Blockchain, που θα προσφέρει δυνατότητες ιδιωτικότητας οδήγησε στη δημιουργία του Quorum Blockchain, το οποίο αποτελεί επέκταση του Ethereum Blockchain.

Η εκρηκτική ανάπτυξη του Διαδικτύου των Πραγμάτων (IoT) έχει οδηγήσει σε μια χιονοστιβάδα δεδομένων που παράγονται από αισθητήρες ενσωματωμένους σε όλα τα είδη, από έξυπνα ρολόγια που παρακολουθούν την υγεία μας έως περιβαλλοντικούς αισθητήρες που παρακολουθούν τα επίπεδα ρύπανσης στις πόλεις μας. Αυτός ο κατακλυσμός δεδομένων παρουσιάζει σημαντικές προκλήσεις, ιδίως όσον αφορά την ασφάλεια των δεδομένων και την προστασία της ιδιωτικής ζωής. Ευαίσθητες πληροφορίες όπως βιομετρικά δεδομένα από wearables ή παράμετροι βιομηχανικών συστημάτων ελέγχου απαιτούν ισχυρή προστασία. Η τεχνολογία blockchain, με την αποκεντρωμένη, απαραβίαστη φύση της, αναδεικνύεται ως μια πιθανή λύση, προσφέροντας μια ασφαλή και διαφανή πλατφόρμα για τη διαχείριση των δεδομένων αισθητήρων IoT. Ωστόσο, η ενσωμάτωση του blockchain σε υπάρχοντα συστήματα IoT παρουσιάζει εμπόδια όπως η επεκτασιμότητα, η καθυστέρηση και η διαλειτουργικότητα.

Επιπλέον, η εγγύηση της ιδιωτικότητας και της ασφάλειας των δεδομένων παραμένει υψίστης σημασίας λόγω της συχνά ευαίσθητης φύσης των δεδομένων IoT.

Η παρούσα διπλωματική αντιμετωπίζει την κρίσιμη ανάγκη για ένα ολοκληρωμένο και αποτελεσματικό σύστημα για την ασφαλή και ιδιωτική διαχείριση των δεδομένων αισθητήρων IoT στο blockchain. Αυτό περιλαμβάνει την αντιμετώπιση προκλήσεων όπως η προστασία της ιδιωτικής ζωής, τη διασφάλιση της επεκτασιμότητας για τον συνεχώς αυξανόμενο όγκο δεδομένων και την επίτευξη απρόσκοπτης διαλειτουργικότητας μεταξύ blockchain και υφιστάμενων συστημάτων IoT. Ενώ οι υπάρχουσες λύσεις βασίζονται συχνά στην κεντρική αποθήκευση δεδομένων, αυτές συχνά δεν διαθέτουν την ευρωστία για να χειριστούν τις απαιτήσεις ασφάλειας και ιδιωτικότητας ενός ταχέως αναπτυσσόμενου τοπίου IoT.

Η παρούσα διπλωματική εργασία στοχεύει στο να καλύψει την απουσία ενός ολοκληρωμένου και αποτελεσματικού συστήματος για την ασφαλή και ιδιωτική διαχείριση δεδομένων αισθητήρων IoT σε blockchain. Αυτό περιλαμβάνει την αντιμετώπιση προκλήσεων όπως η διασφάλιση της ιδιωτικότητας των δεδομένων, η επεκτασιμότητα και η διαλειτουργικότητα μεταξύ blockchain και υφιστάμενων συστημάτων IoT. Ο διαρκώς αυξανόμενος αριθμός συσκευών IoT και ο όγκος των δεδομένων που παράγουν παρουσιάζουν προκλήσεις κλιμάκωσης για τα παραδοσιακά συγκεντρωτικά συστήματα. Η τεχνολογία του blockchain θα μπορούσε να προσφέρει μια κλιμακούμενη και κατανεμημένη αρχιτεκτονική ικανή να διαχειριστεί τον αυξανόμενο όγκο συναλλαγών δεδομένων IoT. Πολλά συστήματα IoT δεν διαθέτουν ισχυρά μέτρα προστασίας της ιδιωτικής ζωής και ασφάλειας των δεδομένων, γεγονός που τα καθιστά ευάλωτα σε παραβιάσεις δεδομένων και μη εξουσιοδοτημένη πρόσβαση. Μια λύση βασισμένη στο blockchain θα μπορούσε να παρέχει μια ασφαλή και ανθεκτική στην παραποίηση πλατφόρμα για την αποθήκευση και τη διαχείριση ευαίσθητων δεδομένων αισθητήρων IoT. Η εφαρμογή που αναπτύχθηκε στοχεύει να ξεπεράσει αυτούς τους περιορισμούς και να αναπτύξει ένα πιο στιβαρό, ανθεκτικό στο μέλλον σύστημα.

Η τεχνολογία του blockchain έχει τη δυνατότητα να βοηθήσει στην αντιμετώπιση διαφόρων προβλημάτων της καθημερινότητας. Για παράδειγμα, πολλά υφιστάμενα συστήματα IoT δεν διαθέτουν ισχυρά μέτρα ασφαλείας, με αποτέλεσμα να είναι ευάλωτα σε παραβιάσεις δεδομένων. Ένα υπονομευμένο wearable που συλλέγει δεδομένα υγείας θα μπορούσε να εκθέσει ευαίσθητες πληροφορίες χρήστη, οδηγώντας ενδεχομένως σε κλοπή ταυτότητας ή κατάχρηση ιατρικών αρχείων. Σε βιομηχανικές εγκαταστάσεις, τα παραβιασμένα δεδομένα αισθητήρων θα μπορούσαν να διαταράξουν κρίσιμες διαδικασίες ή ακόμη και να θέσουν κινδύνους για την ασφάλεια. Επιπλέον ο τεράστιος αριθμός των συσκευών IoT και τα τεράστια δεδομένα που παράγουν υπερκαλύπτουν τα παραδοσιακά συγκεντρωτικά συστήματα. Σε μια έξυπνη πόλη με

εκατομμύρια διασυνδεδεμένους αισθητήρες που παρακολουθούν την κυκλοφορία, τη χρήση ενέργειας και τις περιβαλλοντικές συνθήκες. Ένα συγκεντρωτικό σύστημα θα δυσκολευόταν να διαχειριστεί αποτελεσματικά ένα τέτοιο τεράστιο σύνολο δεδομένων, οδηγώντας σε πιθανές καθυστερήσεις και προβλήματα επιδόσεων. Η ενσωμάτωση του blockchain με υφιστάμενα συστήματα IoT που έχουν διαφορετικές αρχιτεκτονικές και πρωτόκολλα επικοινωνίας είναι πρόκληση. Ένα εργοστάσιο με διάφορους τύπους αισθητήρων και πρότυπα επικοινωνίας χρειάζεται μια λύση που να μπορεί να ενσωματωθεί απρόσκοπτα στην υπάρχουσα υποδομή χωρίς να διακυβεύεται η ασφάλεια ή να εισάγεται καθυστέρηση. Ζητήματα ασυμβατότητας θα μπορούσαν να εμποδίσουν την ανταλλαγή δεδομένων και να περιορίσουν την αποτελεσματικότητα του συνολικού συστήματος.

Η αντιμετώπιση των προαναφερθέντων προκλήσεων είναι ζωτικής σημασίας για την πλήρη αξιοποίηση των δυνατοτήτων των εφαρμογών του IoT. Ένα ασφαλές και ιδιωτικό σύστημα βασισμένο στο blockchain για τη διαχείριση δεδομένων αισθητήρων IoT μπορεί χρησιμοποιώντας χαρακτηριστικά όπως η αμεταβλητότητα και ο επαληθεύσιμος έλεγχος πρόσβασης, το σύστημα δύναται να διασφαλίσει την ακεραιότητα των δεδομένων και να συμμορφωθεί με κανονισμούς όπως ο GDPR και ο HIPAA. Επιπλέον η κατανεμημένη αρχιτεκτονική του blockchain μπορεί να διαχειριστεί τον συνεχώς αυξανόμενο όγκο συναλλαγών δεδομένων IoT, επιτρέποντας την αποτελεσματική διαχείριση δεδομένων ακόμη και σε σενάρια με εκατομμύρια διασυνδεδεμένες συσκευές.

Αξίζει να σημειωθεί ότι διευκολύνετε η διαλειτουργικότητα καθώς, το σύστημα επιτρέπει την ομαλή ανταλλαγή δεδομένων και την επικοινωνία σε διαφορετικές πλατφόρμες.

1.1 Αντικείμενο της Διπλωματικής Εργασίας

Αντικείμενο της διπλωματικής αυτής αποτελεί η δημιουργία μιας εφαρμογής που θα τρέχει στο Quorum Blockchain. Σκοπός είναι αξιοποίηση της τεχνολογίας του Quorum Blockchain για την ανάπτυξη μίας αποκεντρωμένης εφαρμογής (DApp–Decentralised Application), η οποία θα αποτελέσει μία πλατφόρμα διαμοιρασμού δεδομένων IoT, σε συνδυασμό με ένα σύστημα αξιολόγησης (Trust and Reputation).

Η εφαρμογή αυτή βρίσκεται εξ ολοκλήρου στο blockchain και δεν θα βασίζεται καθόλου στη καθιερωμένη λογική του πελάτη-εξυπηρετητή. Η εργασία αποτελείται από δύο διακριτά κομμάτια.Το πρώτο αφορά το blockchain και το δευτερο την ιστοσελίδα. Ο κώδικας που αφορά το blockchain θα αποθηκεύεται σε έξυπνα συμβόλαια (smart contracts) και είναι αυτός που παρέχει όλη τη λειτουργικότητα της εφαρμογής. Οι

χρήστες έχουν πρόσβαση στην εφαρμογή μέσα από τη γραφική διεπαφή χρήστη(UI) της ιστοσελίδας, η οποία θα παρέχει ένα εύχρηστο περιβάλλον αλληλεπίδρασης με το blockchain.

Οι ιδιοκτήτες IoT αισθητήρων θα καταχωρούν τα δεδομένα και οι υποψήφιοι αγοραστές θα μπορούν να τα αγοράζουν. Οι υποψήφιοι αγοραστές μπορούν να βλέπουν όλα τα διαθέσιμα δεδομένα και να διαλέγουν αυτό που επιθυμούν. Οι αγοραστές έχουν την επιλογή να πληρώσουν, είτε με Ethers, είτε με την αντίστοιχη τιμή σε Ntua Tokens. Επιπλέον, προσφέρεται η δυνατότητα αξιολόγησης πωλητών και αγοραστών.

Αξίζει να σημειωθεί ότι το κρυπτονόμισμα Ntua Token δημιουργήθηκε για να προσφέρεται ως μια εναλλακτική στα υπάρχοντα κρυπτονομίσματα που μπορεί ο χρήστης να αγοράσει στα διάφορα ανταλλακτήρια, όπως για παράδειγμα το Ether. Αναπτύχθηκε αποκλειστικά για ερευνητικούς και πειραματικούς σκοπούς και δε διατίθεται προς πώληση. Η ονομασία του προκύπτει από τα αρχικά National Technical University of Athens.

Στα πλαίσια της διπλωματικής εργασίας, λοιπόν,θα διερευνηθούν και οι τρόποι με τους οποίους το Quorum blockchain μπορεί να συνδυαστεί με την υπάρχουσα τεχνολογία για την δημιουργία μίας νέας γενιάς, αποκεντρωμένου τύπου, εφαρμογής.

1.2 Οργάνωση κειμένου

Η παρούσα διπλωματική εργασία αποτελείται από 8 Κεφάλαια και 2 Παραρτήματα.

Το πρώτο κεφάλαιο, το οποίο είναι το παρόν, αποτελεί την εισαγωγή.

Στο δεύτερο κεφάλαιο αναλύεται το θεωρητικό υπόβαθρο της εργασίας. Αρχικά παρουσιάζεται το Web3, το οποίο αποτελώντας τη σύγχρονη μορφή του διαδικτύου έδωσε τη δυνατότητα για την ανάπτυξη αποκεντρωμένων εφαρμογών. Στη συνέχεια παρουσιάζονται τα δίκτυα ομότιμων κόμβων και η κρυπτογραφία ελλειπτικών καμπυλών, που αποτελούν θεμελιώδη στοιχεία των Blockchains. Έπειτα αναλύονται το Ethereum Blockchain και το Quorum Blockchain, το οποίο αποτελεί βασικό στοιχείο της παρούσας εργασίας. Τέλος, γίνεται μια σύντομη αναφορά σε εργασίες σχετικής θεματολογίας.

Στο τρίτο κεφάλαιο παρουσιάζονται τα σημαντικότερα εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν, δηλαδή το τεχνικό υπόβαθρο της διπλωματικής. Αρχικά αναλύονται οι τεχνολογίες που αφορούν το Ethereum και στη συνέχεια παρουσιάζεται το σύστημα διαχείρισης πακέτων NPM και το περιβάλλον εκτέλεσης JavaScript Node.js. Έπειτα γίνεται αναφορά στις τεχνολογίες ανάπτυξής ιστοσελίδων, που χρησιμοποιήθηκαν για τη δημιουργία του Front-end της αναπτυχθείσας εφαρμογής.

Τέλος, παρουσιάζεται το σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων MySQL, που χρησιμοποιήθηκε για την αποθήκευση των δεδομένων, αλλά και το λογισμικό Vagrant, που διευκόλυνε τη διαδικασία ανάπτυξης της εφαρμογής.

Στο τέταρτο κεφάλαιο αναλύονται οι κατηγορίες των χρηστών, οι λειτουργικές και μη λειτουργικές απαιτήσεις του συστήματος, όπως και οι παραδοχές που έγιναν κατά τη δημιουργία της εφαρμογής.

Στο πέμπτο κεφάλαιο παρουσιάζεται αναλυτικά ο σχεδιασμός και η υλοποίηση της εφαρμογής. Πιο συγκεκριμένα, αναλύονται τα έξυπνα συμβόλαια που αναπτύχθηκαν στα πλαίσια της εφαρμογής και εξηγούνται οι μέθοδοι του κώδικά τους. Στη συνέχεια περιγράφονται οι διαδικασίες που έχουν υλοποιηθεί, ώστε να είναι λειτουργική η εφαρμογή και εξηγείται ο τρόπος με τον οποίο τα διάφορα στοιχεία επικοινωνούν μεταξύ τους σε κάθε διαδικασία. Έπειτα γίνεται αναφορά στη βάση δεδομένων, αναλύονται οι λόγοι που οδήγησαν στη χρησιμοποίησή της και περιγράφεται η δομή της.

Στο έκτο κεφάλαιο γίνεται επίδειξη της λειτουργικότητας της εφαρμογής. Χρησιμοποιούνται οι αντίστοιχες εικόνες ώστε να παρουσιαστούν οι ιστοσελίδες της εφαρμογής και επίσης περιγράφεται η λειτουργία της απο την οπτική του χρήστη.

Το έβδομο Κεφάλαιο αποτελεί τον επίλογο. Εκεί παρατίθενται οι παρατηρήσεις και τα συμπεράσματα που προέκυψαν κατά την ανάπτυξη της εφαρμογής. Επίσης προτείνονται μελλοντικές επεκτάσεις του συστήματος, οι οποίες μαζί με την εξέλιξη των σχετικών τεχνολογιών θα το βελτίωναν και θα βελτίωναν τη λειτουργικότητα του.

Στο όγδοο κεφάλαιο παρατίθεται η βιβλιογραφία που χρησιμοποιήθηκε για την συγγραφή της παρούσας διπλωματικής εργασίας.

Στο Παράρτημα Ι περιέχονται αναλυτικές οδηγίες για την εγκατάσταση και τη χρήση της εφαρμογής.

Τέλος, στο Παράρτημα ΙΙ παρατίθεται ο κώδικας της εφαρμογής.

2 Θεωρητικό υπόβαθρο και σχετικές εργασίες

Στο κεφάλαιο αυτό διατυπώνεται το θεωρητικό υπόβαθρο των τεχνολογιών που αφορούν την παρούσα εργασία. Αρχικά παρουσιάζεται η εξέλιξη του παγκόσμιου ιστού, εστιάζοντας στην μορφή στην οποία έχει φτάσει, που φανερώνει την τάση για αποκεντροποίηση, στην οποία συμβάλλει η τεχνολογία των Blockchains και των αποκεντρωμένων εφαρμογών. Στη συνέχεια αναλύονται οι βασικές αρχές των ομότιμων δικτύων και της κρυπτογραφίας ελλειπτικών καμπυλών, που αποτελούν βασικά συστατικά των Blockchains. Έπειτα παρουσιάζονται τα θεωρητικά στοιχεία του

Ethereum Blockchain και του Quorum Blockchain, το οποίο βασίζεται πάνω στο Ethereum και χρησιμοποιήθηκε στην εργασία αυτή. Τέλος γίνεται αναφορά σε εργασίες σχετικές με την παρούσα.

2.1 Web3 – Ο αποκεντρωμένος ιστός

Ο παγκόσμιος ιστός (World Wide Web ή www) είναι ένα ανοιχτό σύστημα διασυνδεδεμένων πληροφοριών και περιεχομένου. Ο αρχικός Παγκόσμιος Ιστός (Web1) έκανε την εμφάνισή του στις αρχές του 1990, φέρνοντας την επανάσταση στην μετάδοση της πληροφορίας. Έδινε πλέον τη δυνατότητα στους χρήστες να έχουν πρόσβαση σε ιστοσελίδες, οι οποίες ήταν δομημένες σε Hypertext Markup Language (HTML), μέσω του web browser.

Ενώ με τον Web1 [1] ο χρήστης του διαδικτύου ήταν παθητικός δέκτης των πληροφοριών, αυτό άλλαξε με την έλευση του Web2, του διαδραστικού ιστού, καθώς οι χρήστες μπορούν πλέον να αλληλεπιδρούν μεταξύ τους, να ανταλλάσουν πληροφορίες, να διαμορφώνουν το περιεχόμενο μιας ιστοσελίδας. Τα πρώτα Web2 εργαλεία αποτέλεσαν τα ιστολόγια (blogs), τα wikis, οι ιστοσελίδες κοινωνικής δικτύωσης, το ηλεκτρονικό εμπόριο, ενώ έκαναν την εμφάνιση τους και ορισμένες εφαρμογές P2P. Το βασικό χαρακτηριστικό του Web2 είναι πως σχεδόν σε κάθε εφαρμογή του υπάρχει μία κεντρική οντότητα, η οποία συνήθως έχει τον πλήρη έλεγχο της λειτουργίας της εφαρμογής αυτής, αλλά ακόμα και στις εφαρμογές P2P, έχοντας το ρόλο του ενδιάμεσου. Αυτός ο τρόπος λειτουργίας προϋποθέτει την ύπαρξη εμπιστοσύνης από τους χρήστες προς την κεντρική οντότητα.

Μια λύση στο πρόβλημα αυτό προσφέρει το Blockchain, το οποίο φαίνεται να είναι η κινητήρια δύναμη του διαδικτύου επόμενης γενιάς, του αποκεντρωμένου ιστού «Decentralised Web» ή αλλιώς Web3. Το Blockchain επαναπροσδιορίζει τον τρόπο αποθήκευσης και διαχείρισης των δεδομένων. Επιτρέπει την άμεση αποστολή αρχείων, δίνοντας τη δυνατότητα για πραγματικές συναλλαγές P2P, χωρίς μεσάζοντες. Το Bitcoin αποτελεί μία από τις πρώτες εφαρμογές που χρησιμοποιεί την τεχνολογία αυτή.

Στο Web3 [2] τα δεδομένα αποθηκεύονται σε πολλά αντίγραφα μέσα σε ένα δίκτυο P2P. Το Blockchain, ως η ραχοκοκαλιά του Web3, επαναπροσδιορίζει τις δομές δεδομένων στο backend. Επομένως, ενώ το Web2 ήταν μια επανάσταση frontend, το Web3 είναι μια επανάσταση backend. Αξίζει να σημειωθεί ότι για το χρήστη φαινομενικά δεν υπάρχει κάποια αλλαγή και η εμπειρία χρήσης παραμένει η ίδια. Έτσι οδηγούμαστε στη μείωση της χρήσης κεντρικών εξυπηρετητών και την υιοθέτηση ενός αποκεντρωμένου μοντέλου. Το γεγονός αυτό σε συνδυασμό με την μείωση των ενδιαμέσων δίνει νέες δυνατότητες στη δημιουργία εφαρμογών και επιτρέπει την επαναπροσέγγιση του τρόπου δημιουργίας τους, με επιπλέον εναλλακτικές στο μοντέλο

του πελάτη-εξυπηρετητή. Η εξέλιξη αυτή προσφέρει αρκετά πλεονεκτήματα, αντιμετωπίζοντας τα προβλήματα και τους περιορισμούς του προηγούμενου μοντέλου και παράλληλα αυξάνοντας την ασφάλεια των αποθηκευμένων δεδομένων, καθώς δε θα υπάρχει ένα μοναδικό σημείο αποτυχίας. Επιπλέον, προστατεύει τα προσωπικά δεδομένα από υποκλοπή και κατάχρηση μέσω του εξυπηρετητή κάθε οργανισμού.

Καθώς περισσότερα Blockchains κάνουν την εμφάνιση τους, είναι πλέον δυνατή η ανάπτυξη και λειτουργία μιας αποκεντρωμένης εφαρμογής (Decentralised Application ή DApp) με μικρές υπολογιστικές και αποθηκευτικές δυνατότητες, καθώς και δυνατότητες πληρωμών, ξεπερνώντας έτσι κάποια από τα προβλήματα που δημιουργούσαν οι περιορισμοί των αρχικών προσπαθειών όσον αφορά την ταχύτητα, την επεκτασιμότητα και την ιδιωτικότητα. Θεωρείται ότι η εξέλιξη αυτης της τεχνολογίας θα οδηγήσει τα τρία στοιχεία της υπολογιστικής, τα οποία είναι η επικοινωνία, η επεξεργασία και η αποθήκευση, στο να γίνουν αποκεντρωμένα. [3]

2.2 Κρυπτογραφία ελλειπτικών καμπυλών

Η κρυπτογραφία ελλειπτικών καμπυλών [4] χρησιμοποιείται για την υπογραφή των συναλλαγών, τόσο στο Bitcoin, όσο και στο Ethereum, παρέχοντας ασφάλεια στις συναλλαγών των χρηστών.

Η θεωρία της κρυπτογραφίας ελλειπτικών καμπυλών (Elliptic Curve Cryptography – ECC) προτάθηκε πρώτη φορά το 1985 από τους Victor Miller (IBM) and Neil Koblitz (University of Washington) ως ένας εναλλακτικός μηχανισμός για την υλοποίηση της κρυπτογραφίας δημόσιου κλειδιού. Το μεγάλο πλεονέκτημα της ECC είναι το γεγονός ότι βασίζεται σε διακριτούς λογαρίθμους και συνεπώς είναι πολύ δυσκολότερο να παραβιαστεί σε σχέση με άλλους γνωστούς αλγορίθμους δημόσιων κλειδιών όπως ο RSA. [5]

Ο αλγόριθμος ψηφιακής υπογραφής ελλειπτικών καμπυλών, μέχρι και σήμερα, θεωρείται απαραβίαστος, καθώς δεν υπάρχει καμία γνωστή και αποδοτική μέθοδος επίθεσης. Επομένως αποτελεί ένα πολύ χρήσιμο εργαλείο με πολλές εφαρμογές στον τομέα της κρυπτογραφίας. [6]

Παρότι τώρα καθιστά τα Blockchains που τον χρησιμοποιούν θωρακισμένα απέναντι σε επιθέσεις, σε περίπτωση που υπάρχει μεγάλη ανάπτυξη στην κβαντική υπολογιστική (Quantum Computing), ο αλγόριθμος αυτός θα σταματήσει να προσφέρει ασφάλεια, οδηγώντας έτσι σε μια προσπάθεια εύρεσης πιθανών βελτιώσεων για την αντιμετώπιση μιας τέτοιας προοπτικής. [7]

2.3 Δίκτυα ομότιμων κόμβων

Η τεχνολογία του Blockchain είναι βασισμένη στην αρχιτεκτονική ομότιμων κόμβων. Υπάρχουν δύο μοντέλα δικτυακών εφαρμογών, τα οποία είναι η αρχιτεκτονική πελάτη-εξυπηρετητή (client-server) και η αρχιτεκτονική ομότιμων κόμβων (peer-to-peer ή P2P). [8]

Στην αρχιτεκτονική πελάτη-εξυπηρετητή υπάρχει ένας πάντα ενεργός υπολογιστής, ο εξυπηρετητής, ο οποίος εξυπηρετεί αιτήσεις για υπηρεσίες από άλλους υπολογιστές, τους πελάτες. Οι υπολογιστές-πελάτες δεν έχουν καμία επικοινωνία μεταξύ τους, αλλά όλοι εξαρτώνται από τον κεντρικό εξυπηρετητή, γεγονός που καθιστά το ρόλο του καθοριστικής σημασίας για την λειτουργία του δικτύου. Το μοντέλο αυτό έχει τα πλεονεκτήματα της εύκολης υλοποίησης και διαχείρισης, αλλά και αρκετά μειονεκτήματα όπως μοναδικό σημείο αποτυχίας (Single point of failure) και δύσκολη κλιμάκωση.

Η εναλλακτική του ειναι η αρχιτεκτονική ομότιμων κόμβων, στην οποία η στήριξη σε αποκλειστικούς εξυπηρετητές σε κέντρα δεδομένων είναι ελάχιστη ή ακόμη και μηδενική. Σε ένα δίκτυο ομότιμων κόμβων ο κάθε κόμβος ο οποίος συμμετέχει (peer) είναι ισότιμος με κάθε άλλο, έχοντας τα ίδια προνόμια, τις ίδιες δυνατότητες και τον ίδιο ρόλο στο δίκτυο. Οι κόμβοι αυτοί του δικτύου έχουν διπλό ρόλο, καθώς ο καθένας μπορεί να ενεργήσει είτε σαν πελάτης (client) είτε σαν εξυπηρετητής (server). Η επικοινωνία μεταξύ τους γίνεται μέσω προσωρινών ζευγών υπολογιστών, οι οποίοι συνδέονται απευθείας. Οι κόμβοι κατα κύριο λόγο είναι υπολογιστές που ανήκουν σε χρήστες και δεν ελέγχονται από κάποιο οργανισμό ή κεντρική οντότητα. Το δίκτυο έχει την ικανότητα να προσαρμόζεται και να αυτοδιοργανώνεται κάθε φορά που εισέρχονται ή εξέρχονται κόμβοι. Με αυτό τον τρόπο επιτρέπει τη σύνδεση μεγάλου αριθμού χρηστών, δίνει τη δυνατότητα της κλιμάκωσης αλλά και προσφέρει ανοχή σε αποτυχίες, αποφεύγοντας το Single Point Of Failure. Οι κόμβοι, προσφέροντας ένα μέρος των υπολογιστικών τους πόρων για την λειτουργία του δικτύου, εξαλείφουν την ανάγκη ύπαρξης μιας κεντρικής αρχής υπεύθυνης για τον συντονισμό και τη λειτουργία του δικτύου. Ωστόσο, αξίζει να σημειωθεί ότι υπάρχουν κάποιες εφαρμογές που βασίζονται σε μία υβριδική αρχιτεκτονική, συνδυάζοντας αυτήν του πελάτη-εξυπηρετητή και των ομότιμων.

Η αρχιτεκτονική ομότιμων κόμβων έχει εφαρμογή σε διάφορα πεδία, όπως για παράδειγμα η διανομή αρχείων (BitTorrent), η τηλεφωνία διαδικτύου (Skype) και η IPTV. Το Blockchain, στο οποίο βασίζεται και η εφαρμογή της παρούσας διπλωματικής εργασίας, αποτελεί και αυτό μια τεχνολογία η οποία βασίζεται στην αρχιτεκτονική ομότιμων κόμβων.

Τα δίκτυα ομότιμων χωρίζονται σε δύο μεγάλες κατηγορίες, στα αδόμητα (unstructured) και στα δομημένα (structured) δίκτυα. [9]

2.3.1 Αδόμητα δίκτυα

Τα αδόμητα P2P δίκτυα δεν επιβάλλουν κάποια συγκεκριμένη δομή στο ανώτερο επίπεδο του δικτύου, αλλά σχηματίζονται μέσω των τυχαίων συνδέσεων που πραγματοποιούν οι κόμβοι τυχαία μεταξύ τους (π.χ. Gnutella). Καθώς δεν υπάρχει κανένας συσχετισμός μεταξύ δεδομένων και των κόμβων που τα προσφερουν, προκύπτουν αρκετά πλεονεκτήματά, όπως η εύκολη δημιουργία και η δυνατότητα τοπικών βελτιστοποιήσεων σε διαφορετικές περιοχές της τοπολογίας. Όλοι οι κόμβοι έχουν τον ίδιο ρόλο στο δίκτυο, επομένως είναι ανθεκτικά σε συνθήκες υψηλού ρυθμού προσθήκης και αναχώρησης κόμβων (churn).

Ένα από τα βασικά μειονεκτήματά είναι ο πλημμυρισμός του δικτύου σε κάθε ερώτημα αναζήτησης δεδομένων, που οδηγεί σε αυξημένη κίνηση και χρησιμοποίηση των πόρων του δικτύου. Καθώς ορισμένα δεδομένα βρίσκονται σε λιγοτερους κόμβους από κάποια και επιπλέον δεν υπάρχει συσχέτιση ανάμεσα στον κάθε κόμβο και τα δεδομένα που διατηρεί, δεν εξασφαλίζεται ότι ένα ερώτημα που αφορά σχετικά “σπάνια” δεδομένα θα φτάσει στον κατάλληλο κόμβο, ώστε να δοθεί απάντηση στο ερώτημα της αναζήτησης.

2.3.2 Δομημένα δίκτυα

Στα δομημένα P2P δίκτυα το ανώτερο επίπεδο του δικτύου οργανώνεται σχηματίζοντας μία ορισμένη τοπολογία, ενώ το πρωτόκολλο εξασφαλίζει ότι κάθε κόμβος μπορεί να κάνει αποδοτική αναζήτηση οποιωνδήποτε δεδομένων, ακόμα και αν αυτά υπάρχουν σε πολύ μικρό αριθμό κόμβων. Ο πιο διαδεδομένος τύπος δομημένου P2P δικτύου υλοποιεί έναν κατανεμημένο πίνακα κατακερματισμού (Distributed Hash Table – DHT), στον οποίο χρησιμοποιείται μία παραλλαγή της συνάρτησης κατακερματισμού (consistent hashing), ώστε να ανατεθεί η κατοχή κάθε αρχείου σε έναν συγκεκριμένο κόμβο. Με τον τρόπο αυτό, η αναζήτηση αρχείων (δεδομένων) γίνεται αποτελεσματικά από κάθε κόμβο, με την χρησιμοποίηση του DHT. Αυτό αποτελεί και το μεγαλύτερο πλεονέκτημα των δομημένων P2P δικτύων έναντι των αδόμητων. Λόγω της παραπάνω οργάνωσης του δικτύου, για την ομαλή δρομολόγηση της κίνησης απαιτείται από κάθε κόμβο η τήρηση λιστών γειτονικών κόμβων που πληρούν συγκεκριμένα κριτήρια. Η ανάγκη αυτή κάνει το δίκτυο περισσότερο ευάλωτο σε συνθήκες υψηλού ρυθμού προσθήκης και αναχώρησης κόμβων (churn).

2.3.3 Πλεονεκτήματα των δικτύων ομότιμων κόμβων

Τα βασικά πλεονεκτήματα που παρουσιάζει η αρχιτεκτονική των ομότιμων κόμβων είναι η αυτοκλιμακωσιμότητα (self-scalability) και η μη ύπαρξη μοναδικού σημείου

αστοχίας του δικτύου (No SPOF). Ταυτόχρονα μειώνεται το κοστος, καθώς συνήθως δεν απαιτείται σημαντική υποδομή και εύρος ζώνης εξυπηρετητή. Η αυτοκλιμακωσιμότητα, είναι ένα εγγενές χαρακτηριστικό, αφού όσο περισσότεροι κόμβοι προστίθενται στο δίκτυο, τόσο αυξάνεται ο φόρτος εργασίας, δηλαδή η απαίτηση σε πόρους, όμως τόσο αυξάνονται και οι διαθέσιμοι πόροι, λόγω της διττής φύσης του κάθε ομότιμου κόμβου. Τέλος, η βλάβη σε έναν κόμβο δεν επηρεάζει την λειτουργία του υπόλοιπου δικτύου, όπως γίνεται στην αρχιτεκτονική πελάτη-εξυπηρετητή, όπου αστοχία του εξυπηρετητή συνεπάγεται τη μη διαθεσιμότητα της εφαρμογής.

2.4 Ethereum Blockchain

Το Ethereum [10] είναι μία δημόσια, ανοιχτού κώδικα, Blockchain-based κατανεμημένη υπολογιστική πλατφόρμα. Υποστηρίζει μια τροποποιημένη εκδοχή του Nakamoto consensus, χρησιμοποιώντας transaction-based μεταβάσεις καταστάσεων. Το κρυπτονόμισμα, το οποίο παράγεται και χρησιμοποιείται στο Ethereum, είναι το Ether. Με σκοπό τον περιορισμό της άσκοπης σπατάλης των πόρων του δικτύου χρησιμοποιείται ένας εσωτερικός μηχανισμός που ορίζει ένα transaction fee, ως προμήθεια για κάθε transaction, που ονομάζεται Gas.

Το Ethereum έχει αρκετά κοινά στοιχεία με άλλα Blockchains, όπως τη χρήση ενός peer-to-peer δικτύου, που συνδέει όλους τους συμμετέχοντες, τον Byzantine fault–tolerant consensus αλγόριθμο για τον συγχρονισμό των states updates, την χρησιμοποίηση κρυπτογραφικών εργαλείων (digital signatures και hashes) και φυσικά τα ψηφιακά νομίσματα (Ether). Παρόλα αυτά έχει σημαντικές διαφορές από τα προϋπάρχοντα Blockchains, τόσο στη λειτουργία του, όσο και στον σκοπό της δημιουργίας του. Δημιουργήθηκε, όχι για να αποτελέσει άλλο ένα δίκτυο ενός ψηφιακού νομίσματος, αλλά για να χρησιμοποιείται ως ένας “παγκόσμιος υπολογιστής”, στον οποίο το ψηφιακό νόμισμα, λειτουργώντας ως τρόπος πληρωμής, θα ήταν ζωτικής σημασίας για τη χρήση του.

Το Bitcoin [11] , το οποίο είναι φυσικό να συγκρίνεται με το Ethereum, διαθέτει μια πολύ περιορισμένη scripting γλώσσα, καθώς από την αρχή υπήρχε η πρόθεση να μπορούν να πραγματοποιηθούν μόνο απλοί υπολογισμοί που αφορούν την επιβεβαίωση των συνθηκών των συναλλαγών. Παρέχει κυρίως την δυνατότητα (οικονομικών) συναλλαγών του κρυπτονομίσματος. Η κατανεμημένη βάση δεδομένων του Bitcoin γίνεται αντιληπτή ως ένας πίνακας από λογαριασμούς με Bitcoin και οι συναλλαγές είναι μεταφορές των Bitcoin μεταξύ λογαριασμών. [12]

Όσο γινόταν πιο δημοφιλές, ολοένα και περισσότεροι χρήστες άρχισαν να χρησιμοποιούν το δίκτυο του Bitcoin για σκοπούς που δεν αφορούσαν τη μεταφορά κρυπτονομισμάτων. Όμως, η χρησιμοποίησή του για εφαρμογές, πέρα από αυτήν για

την οποία δημιουργήθηκε, σήμαινε ότι οι προγραμματιστές έπρεπε να βρίσκουν τρόπους να παρακάμπτουν τους εγγενείς περιορισμούς του. Σε πολλές περιπτώσεις οι περιορισμοί αυτοί έκαναν την υλοποίηση πολλών εφαρμογών αρκετά δυσκολότερη, αφού απαιτούσαν δυνατότητες και ελευθερίες που το Bitcoin δεν προσέφερε.

Το 2013 ο Vitalik Buterin πρότεινε την ιδέα ενός Blockchain, το οποίο θα είναι επαναπρογραμματιζόμενο, ώστε να μπορεί να εκτελεί πολλές και περίπλοκες εργασίες. Λίγο αργότερα δημιουργήθηκε η πλατφόρμα Ethereum, η οποία μέσω των έξυπνων συμβόλαιων (smart contracts), επιτρέπει στο χρήστη να υλοποιήσει τις εφαρμογές που επιθυμεί, χωρίς περιορισμούς στην πολυπλοκότητα, προσφέροντας περισσότερες προγραμματιστικές δυνατότητες. [13]

Το Ethereum, αποτελώντας μία πιο ευέλικτη και προσαρμόσιμη πλατφόρμα, δίνει στους developers τη δυνατότητα να δημιουργήσουν αποκεντρωμένες εφαρμογές (Decentralised Applications - DApps) με ενσωματωμένες λειτουργίες για οικονομικές συναλλαγές, παρέχοντας διαφάνεια και αξιοπιστία. [14]

Το Ethereum είναι, κατά μία έννοια, μία σουίτα πρωτοκόλλων που καθορίζουν μία πλατφόρμα για την ανάπτυξη και λειτουργία αποκεντρωμένων εφαρμογών.

Εν αντιθέσει με το Bitcoin, το Ethereum είναι Turing complete, αφού έχει σχεδιαστεί ως ένα γενικού σκοπού προγραμματίσιμο Blockchain. Κάθε κόμβος του δικτύου διαθέτει την Ethereum Virtual Machine (EVM), η οποία έχει τη δυνατότητα να εκτελέσει τα έξυπνα συμβόλαια ή smart contracts (scripts), χωρίς περιορισμούς στην πολυπλοκότητα του κώδικά τους. Τα smart contracts γράφονται σε μια προγραμματιστική γλώσσα υψηλού επιπέδου (Solidity) και στη συνέχεια μεταγλωτίζονται σε EVM bytecode. [15]

Η τεράστια παραλληλοποίηση των υπολογισμών στο Ethereum δίκτυο δεν αποσκοπεί σε μεγαλύτερη αποτελεσματικότητα στους υπολογισμούς. Αυτό που προσφέρει ο παραλληλισμός, μέσω των μηχανισμών του Blockchain consensus, είναι η μεγαλύτερη ανεκτικότητα σε σφάλματα, η δυνατότητα να λειτουργεί ασταμάτητα αλλά και η εξασφάλιση ότι τα δεδομένα που είναι αποθηκευμένα είναι αμετάβλητα. Η βάση δεδομένων του διατηρείται και συντονίζεται από τους κόμβους που είναι συνδεδεμένοι στο δίκτυο. Κάθε κόμβος του δικτύου εκτελεί και καταγράφει τις ίδιες συναλλαγές, οι οποίες οργανώνονται σε block και προστίθενται στο Blockchain.

Κάθε φορά, μπορεί να προστεθεί στο Blockchain μόνο ένα block, το οποίο περιέχει την απόδειξη εργασίας (Proof of Work), ώστε οι άλλοι κόμβοι να μπορούν να ελέγξουν την ορθότητά του. Λόγω διαφόρων προβλημάτων που προκύπτουν, όπως η μεγάλη κατανάλωση ισχύος, εξετάζεται και η μετάβαση στο Proof of Stake, που αν και χρησιμοποιείται με επιτυχία σε άλλα Blockchains, έχει και αυτό τα αρνητικά του. Οι

κόμβοι που συντηρούν το δίκτυο, δηλαδή αυτοί που δημιουργούν τα blocks ονομάζονται miners.

Ο τρόπος λειτουργίας του Ethereum

Λογαριασμοί

Ο λογαριασμός (Ethereum account) είναι η βασική μονάδα στο Ethereum και ορίζει την κατάσταση (state), διαφέροντας απο το Bitcoin Blockchain που βασίζεται στη συναλλαγή (transaction). Κάθε λογαριασμός είναι μια διεύθυνση μήκους 20 bytes.

Κατά την μετάβαση κατάστασης (state transition) του Ethereum μεταφέρονται άμεσα αξία, αλλά και πληροφορίες μεταξύ λογαριασμών. Κάθε λογαριασμός περιλαμβάνει ένα μετρητή που χρησιμοποιείται για να επαληθευτεί ότι κάθε συναλλαγή μπορεί να επεξεργαστεί μόνο μία φορά (nonce), το τρέχον ισοζύγιο του λογαριασμού (σε ethers), τον κώδικα του έξυπνου συμβολαίου του λογαριασμού, εφόσον υπάρχει, και τον αποθηκευτικό χώρο του λογαριασμού,ο οποίος αρχικά είναι κενός.

Υπάρχουν δύο είδη λογαριασμών:

- Οι Externally Owned Accounts (EOAs), οι οποίοι ελέγχονται από ιδιωτικά κλειδιά.
- Οι Contract Accounts (CAs), οι οποίοι ελέγχονται από τον κώδικα του έξυπνου συμβολαίου.

Οι χρήστες που κατέχουν EOA, μπορούν να δημιουργήσουν νέα συμβόλαια, δημοσιεύοντάς τα στο Blockchain. Το έξυπνο συμβόλαιο (smart contract) είναι επί της ουσίας ο κώδικας του λογαριασμού συμβολαίου. Σε ένα λογαριασμό συμβολαίου, κάθε φορά που ο λογαριασμός του συμβολαίου λαμβάνει ένα μήνυμα, ενεργοποιεί και εκτελεί τον κώδικα του, επιτρέποντάς του να διαβάζει και να γράφει στον αποθηκευτικό χώρο του και να στέλνει άλλα μηνύματα ή να δημιουργεί έξυπνα συμβόλαια.

Οι χρήστες στέλνουν συναλλαγές στο δίκτυο του Ethereum, χρησιμοποιώντας την κρυπτογραφία ελλειπτικών καμπυλών (ECDSA), ώστε να υπογράψουν τα δεδομένα της συναλλαγής με το ιδιωτικό τους κλειδί. Έτσι εξασφαλίζεται η εγκυρότητα της διαδικασίας, αφού μπορεί να επαληθευτει, με αυτόν τον τρόπο, η ταυτότητα του αποστολέα. [16]

Το Ether αποτελεί το κρυπτονόμισμα του Ethereum αλλά ταυτόχρονα χρησιμοποιείται και για την πληρωμή των τελών των συναλλαγών. Κάθε λειτουργία που εκτελείται στο EVM, εφόσον καταναλώνει υπολογιστική ισχύ, απαιτεί Ether για να εκτελεστεί.

Το Ether που καταναλώνεται για να γίνει μια λειτουργία, καλείται gas, αφού μπορεί να θεωρηθεί και καύσιμο του EVM. Η πληρωμή γίνεται για τους υπολογισμούς και την

μνήμη που χρησιμοποιείται για την πραγματοποίηση της συναλλαγής και είναι ανάλογη των υπολογιστικών πόρων του EVM που θα χρησιμοποιηθούν. Έτσι προστατεύεται το δίκτυο, καθώς αποθαρρύνει τις επιπόλαιες εντολές υπολογισμού, κακόβουλες επιθέσεις και επιθέσεις άρνησης υπηρεσίας (DDoS) κάνοντας τες ασύμφορες. Επιπλέον περιορίζεται η σπατάλη πόρων από προγραμματιστικά λάθη, όπως η δημιουργία άπειρων βρόχων. Η μεγάλης αξίας της μονάδας του νομίσματος Ether, καθιστά απαραίτητη τη χρήση των υποδιαιρεσεων του για το σκοπό του υπολογισμού του gas, όπως το wei (1 ETH = 1018 wei) και το Gwei (Gigawei, 1 gwei = 109 wei). [17]

Τα ανωτέρω τέλη εισπράττονται από τους κόμβους που επικυρώνουν το δίκτυο, τους miners. Οι miners αποτελούν τους κόμβους του δικτύου που λαμβάνουν, διαδίδουν, επικυρώνουν και εκτελούν συναλλαγές, καταναλώνοντας υπολογιστική ισχύ και λαμβάνοντας ως αντίτιμο το αντίστοιχο gas. Οι miners, αφού κάθε φορά συγκεντρώσουν ορισμένες συναλλαγές δημιουργώντας ένα block, ανταγωνίζονται μεταξύ τους, για το ποιός θα είναι αυτός που θα εισάγει στο Blockchain το νέο block. Μετά από κάθε εισαγωγή ενός νέου block, ο miner λαμβάνει τα ether που αντιστοιχούν στις συναλλαγές που αυτό περιέχει, κάτι που αποτελεί κίνητρο για την κατανάλωση των απαιτούμενων υπολογιστικών πόρων.

Η επικύρωση των συναλλαγών γίνεται μέσω της λύσης ενός δύσκολου μαθηματικού προβλήματος που έγκειται στον υπολογισμό του nonce κάθε συναλλαγής.

Ο consensus μηχανισμός του Ethereum είναι το Proof of Work (PoW), αλλά διαφέρει σε σχέση με το Bitcoin, στο γεγονός ότι το μαθηματικό πρόβλημα έχει μεγάλες απαιτήσεις μνήμης. Το αποτελεσμα αυτης της διαφοροποίησης είναι ένα πιο ασφαλές και αποκεντρωμένο δίκτυο, δηλαδή περισσότεροι “μικροί” miners αντί για λίγους “ισχυρούς”. Έτσι περιορίζεται η χρήση ειδικού hardware κατασκευασμένου αποκλειστικά για mining (π.χ. ASICs).

Τα έξυπνα συμβόλαια έχουν τη δυνατότητα να αλληλεπιδρουν με άλλα συμβόλαια μέσω των μηνυμάτων. Κάθε μήνυμα περιέχει τον αποστολέα, τον παραλήπτη, την ποσότητα ether που μεταφέρεται μαζί με το μήνυμα, ένα πεδίο δεδομένων (προαιρετικά) και μια τιμή STARTGAS. Όταν ο κώδικας του συμβολαίου εκτελέσει τον CALL κωδικό εκτέλεσης (opcode), παράγει και εκτελεί ένα μήνυμα, το οποίο στη συνέχεια εκτελεί τον κώδικα του λογαριασμού παραλήπτη, όπως θα γινόταν με μία συναλλαγή, με τη διαφορά ότι τα μηνύματα μπορούν να δημιουργηθούν μόνο από λογαριασμούς συμβολαίων και όχι από EOA. Επομένως γίνεται κατανοητό ότι τα συμβόλαια μπορούν να κάνουν συναλλαγές με άλλα συμβόλαια, όπως μπορούν και οι εξωτερικοί λογαριασμοί.

Ο μηχανισμός παραγωγής των block

Τα blocks στο Ethereum Blockchain δεν έχουν σταθερό μέγεθος, ενώ το χρονικό διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών blocks εξαρτάται από το επίπεδο της δυσκολίας του δικτύου και δεν είναι προκαθορισμένο. Κατά την δημιουργία των blocks οι miners πρέπει να λάβουν υπόψη τους το gas limit, το οποίο περιορίζει το μέγεθος του block αλλά και την υπολογιστική ισχύ που απαιτείται για την δημιουργία του κάθε block. Το gas χρησιμοποιείται ανάλογα με τους υπολογιστικούς πόρους που απαιτούν οι διαφορες λειτουργίες. Οι miners έχουν τη δυνατότητα, αν κριθεί ότι είναι αναγκαίο, να μεταβάλουν μέσω ψηφοφορίας το gas limit, επιρεάζοντας παράλληλα και το μεγεθος του block. Το χρονικό διάστημα ανάμεσα σε δυο blocks υπολογίζεται από μια συγκεκριμένη σχέση η οποία το κρατά στο διαστημα 10 - 19 δευτερολέπτων.

2.5 Quorum

Το Quorum [18] είναι είναι ένα permissioned Blockchain, κάτι το οποίο σημαίνει ότι μόνο ελεγμένοι/προεγκεκριμένοι κόμβοι μπορούν να συμμετάσχουν σε αυτό. Αποτελεί ένα fork του Ethereum Blockchain και κύρια προτεραιότητα του αποτελεί η παροχή ιδιωτικότητας στα smart contracts που την χρειάζονται. Οι συνηθισμένες public συναλλαγές υποστηρίζονται όπως και στο Ethereum, αλλά στο Quorum υπάρχουν επιπλέον και οι private συναλλαγές, οι οποίες λειτουργούν συνεργατικά με off-chain επικοινωνία μεταξύ των κόμβων. Όλες οι συναλλαγές είτε public είτε private καταγράφονται στο public Blockchain, χωρίς βέβαια να μπορεί να δει ο καθένας τα δεδομένα που είναι επιθυμητό να παραμείνουν ιδιωτικά. Οι συμμετέχοντες, οι οποίοι έχουν πρόσβαση στα προκαθορισμένα δεδομένα, μπορούν να τα αποκρυπτογραφήσουν και να αλληλεπιδράσουν με τα αντιστοιχα smart contracts. Κάθε κόμβος εκτός από την καταγραφή της public state του Blockchain, καταγράφει ξεχωριστά και την private state σε ένα διαφορετικό merkle tree, διασφαλίζοντας έτσι την ακεραιότητα και την ορθότητα των κρυπτογραφημένων δεδομένων.

Τα smart contracts στο Ethereum βασίζονται σε ένα shared ledger και δεν παρέχουν ιδιωτικότητα, καθώς σχεδιάστηκε με σκοπό να βελτιώνει την απόδοση και μειώνει το κόστος λειτουργίας. Στην τρέχουσα μορφή τους τα smart contracts αφήνουν τα δεδομένα εκτεθειμένα στο shared ledger σε οποιονδήποτε τα αναζητήσει. Το Quorum παρέχει μια πολύ καλή λύση σε αυτό το πρόβλημα, καθώς ενώ εξασφαλίζει την ασφάλεια τόσο των συμμετεχόντων, όσο και ολόκληρου του δικτύου, επιτρέπει την πρόσβαση στα ιδιωτικά δεδομένα μόνο σε όσους την δικαιούνται. Αξίζει να σημειωθεί ότι άλλες επιλογές προς αυτή την κατεύθυνση είναι οι εξής: zero-knowledge proofs,

homomorphic encryption, secure multi-party computation, ledger segmentation, cryptographic protocols.

Το Quorum προσπαθεί να εκμεταλλευτεί όσα περισσότερα από τα εγγενή χαρακτηριστικά της επίσημης έκδοσης Go Ethereum είναι δυνατόν, ελαχιστοποιώντας έτσι τις τροποποιήσεις που θα απαιτούνται μελλοντικά ώστε να παραμένει συμβατό με αλλαγές του public Ethereum. Αυτό καθίσταται δυνατό από το γεγονός ότι τα χαρακτηριστικά που προσφέρουν την δυνατότητα private δεδομένων, βρίσκονται πάνω από το layer του standard Ethereum protocol. Το Quorum Blockchain χαρακτηρίζεται ως private/permission Blockchain και χρησιμοποιεί τους consensus αλγόριθμους Raft και Istanbul BFT (Byzantine Fault Tolerance). Επίσης, εισάγει ένα νέο τύπο transaction, τις private transactions, οι οποίες προσδίδουν χαρακτηριστικά ιδιωτικότητας.

Οι τροποποιήσεις σε σχέση με το Go-Ethereum αφορούν στις διαδικασίες του block proposal και validation, αφού πλέον κάθε κόμβος μπορεί να επικυρώσει τις public transactions και τις private στις οποίες συμμετέχει, εκτελώντας το αντίστοιχο smart contract, αλλά θα παραλείπει τις υπόλοιπες private transactions. Επομένως έτσι σχηματίζονται δύο διαφορετικά state databases, ένα public database, για το οποίο όλοι οι κόμβοι συμφωνούν, αλλά και ένα private state database που θα διαφέρει ανάλογα με τα δεδομένα που είναι προσβάσιμα από κάθε κόμβο. Η διαφορά με άλλες προσεγγίσεις data segmentation, είναι ότι, αν και κάθε κόμβος δεν έχει όλη την πληροφορία, εξασφαλίζεται κρυπτογραφικά η ορθότητα τών δεδομένων στο Blockchain, αφού δεν μπορεί κάποιος να τα παραποιήσει (immutability).

Το Quorum αποτελείται από το Quorum node (δηλαδή έναν τροποποιημένο public Geth client) και τον Privacy Manager, ο οποίος συγκροτείται απο τον Transaction Manager και το Enclave. Το Enclave λειτουργεί απομονωμένο από τα υπόλοιπα στοιχεία, διαφυλάσοντας έτσι τα private keys με ασφάλεια. Με τη βοήθεια του Transaction Manager διαχειρίζεται την κωδικοποίηση/αποκωδικοποίηση των συναλλαγών. [19]

2.5.1 Consensus αλγόριθμοι

Οι δύο βασικοί αλγόριθμοι που είναι διαθέσιμοι στο Quorum είναι ο Raft και ο Istanbul BFT (Byzantine Fault Tolerance).

Ο raft consensus mechanism χρησιμοποιείται κυρίως σε περιπτώσεις closed-membership consortium που δεν απαιτείται να υπάρχει Byzantine fault tolerance. Δεν επιτρέπεται να γίνει fork to Blockchain, επομένως το περιεχόμενο κάθε block είναι τελικό, αφού επιβεβαιωθεί πρώτα από την πλειοψηφία. Σε αυτό το μοντέλο υπάρχει ένας leader, ο οποίος εκλέγεται από την πλειοψηφία των κόμβων. Κατά την εκλογή όλοι οι κόμβοι είναι υποψήφιοι, αλλα μετά από αυτή αποτελούν τους followers.

Όλα τα transactions φτάνουν στον leader, ο οποίος είναι ο μόνος που έχει τη δυνατότητα να κάνει mint ένα νέο block, που θα περιέχει τα bundled transactions, χωρίς να υπάρχει απαίτηση Proof of Work.

Ο Istanbul-BFT consensus αποτελείται από τρεις φάσεις: PRE-PREPARE, PREPARE και COMMIT. Οι κόμβοι οι οποίοι εκλέγονται ως block validators επιλέγουν έναν κόμβο για να προτείνει το νέο block, ο οποίος το κάνει broadcast μαζί με το μήνυμα “PRE-PREPARE”, που όταν το λαμβάνουν οι validators μπαίνουν στην πρώτη φάση (PRE-PREPARED) και απαντούν με το μήνυμα “PREPARE”, εξασφαλίζοντας έτσι ότι όλοι συμβαδίζουν στη διαδικασία. Στη συνέχεια αφού οι validators λάβουν το μήνυμα “PREPARE” από τουλάχιστον τα δύο τρίτα των κόμβων, μεταβαίνουν στη δεύτερη φάση (PREPARED) και εκπέμπουν το μήνυμα “COMMIT”, που ενημερώνει ότι το νέο block είναι αποδεκτό και θα προστεθεί στο Blockchain. Ακολουθεί η τρίτη φάση (COMMITED), όταν πλέον τα δύο τρίτα των κόμβων απαντήσουν με το μήνυμα “COMMIT” και το block προστίθεται. Με αυτό τον τρόπο, ο μηχανισμός αυτός διασφαλίζει ότι ακόμα και αν το ένα τρίτο των κόμβων του δικτύου είναι προβληματικό ή κακόβουλο, το σύστημα θα αντέξει. Τα blocks είναι αμετάβλητα, καθώς ούτε εδώ είναι δυνατόν να γίνει fork. Το χαρακτηριστικό αυτό σε συνδυασμό με την απαίτηση κάθε validator να προσθέτει την υπογραφή από το μήνυμα “COMMIT” που δέχτηκε στο πεδίο extradata, το οποίο βρίσκεται στην επικεφαλίδα κάθε block, σημαίνει ότι ακόμα και ένας προβληματικός κόμβος δεν μπορει να παρεκκλίνει από το κύριο Blockchain.

2.5.2 Ιδιωτικότητα

Data Privacy

Η υλοποίηση του data privacy επιτυγχάνεται μέσα από την κρυπτογράφηση των δεδομένων και των καταμερισμό των state databases (Segmentation). Όπως αναφέρθηκε και προηγουμένως, κάθε κόμβος έχει πρόσβαση μόνο στα δεδομένα των private συναλλαγών στις οποίες λαμβάνει μέρος και με βάση αυτές ενημερώνει το private μέρος της local state database.

Private Transactions

Οι DApps μέσω ενός API αποκτούν τη δυνατότητα να κάνουν χρήση των private transactions. Κάθε private transaction μεταφέρει ενα 256-bit hash στο πεδίο δεδομένων. Η αναγνώριση κάθε private transaction γίνεται από το πεδίο “v”, το οποίο θα έχει μια απο τις τιμες 37 ή 38, ώστε να ξεχωρίζει από τις συνηθισμένες που έχουν 27 ή 28. Ένα επιπλέον πεδίο (PrivateFor) περιέχει μία λίστα των public keys των κόμβων που συμμετέχουν στη συναλλαγή. Σε όλους τους κόμβους αποστέλλεται ένα κανονικό

transaction που που περιέχει μόνο το hash των δεδομένων/payload, στα οποία έχουν πρόσβαση οι προκαθορισμένοι κόμβοι. Επομένως ένα Quorum transaction έχει εκτός από τα κανονικά πεδία (παραλήπτης, υπογραφή αποστολέα και ποσό σε ether) και άλλα δύο προαιρετικά (privateFor και dataField).

Private Contracts

Τα private contacts δημιουργούνται αποκλειστικά από private transactions, αφού θα πρέπει να καταγράφεται ξεχωριστά το state τους, σε άλλο ξεχωριστό Merkle tree, σε σχέση με τις αντίστοιχες public transactions που χρησιμοποιούν το public Merkle tree.

Block Validation και State Consensus

Στο Ethereum η διαδικασία του block validation συμπεριλαμβάνει την επιβεβαίωση της global state όλων των smart contracts των οποίων το hash που τους αντιστοιχεί περιλαμβάνεται στην block header, αποδεικνύοντας κρυπτογραφικά ότι όλοι στο δίκτυο έχουν το ίδιο state database. Στο Quorum, καθώς υπάρχουν 2 ξεχωριστά state databases, το δίκτυο συμφώνει στην κατάσταση του public, ενώ το private διαφέρει.

Όταν χρειάζεται να επιβεβαιωθεί κρυπτογραφικά το state ενός private contract, η εφαρμογή, μέσα από το αντίστοιχο API, ανακτά το state hash για το επιλεγμένο block και το μοιράζει στους δικαιούχους είτε off-chain είτε με on-chain transaction, ανάλογα με το σκοπό της εφαρμογής.

Transaction management

Το Quorum, για την επίτευξη της ιδιωτικότητας δίνει την δυνατότητα της επιλογής ανάμεσα σε δύο, κατά κύριο λόγο, διαθέσιμα εργαλεία: το Tessera και το Constellation.

Το Tessera χρησιμοποιείται για την κωδικοποίηση/αποκωδικοποίηση και το διαμοιρασμό όλων των συναλλαγών για το Quorum. Αποτελεί ένα stateless Java σύστημα, κάθε κόμβος του οποίου δημιουργεί και διατηρεί ζευγάρια public/private κλειδιών, ενώ ανακαλύπτει όλους τους κόμβους του δικτύου, συνδεόμενος απλά με έναν άλλο Tessera κόμβο. Διαθέτει δύο διαφορετικά API, ένα private για επικοινωνία με το Quorum και ενα public για επικοινωνία μεταξύ των υπολοίπων Tessera κόμβων. Επιπλέον προσφέρει δυνατότητες αμφίδρομης SSL ασφάλειας (TLS certificates), σύνδεσης σε οποιαδήποτε SQL βάση δεδομένων που υποστηρίζει JDBC client, καθώς και IP whitelisting.

Το Constellation αποτελεί μια εναλλακτική επιλογή transaction management του Quorum, σε σχέση με το Tessera. Διαχειρίζεται τα ζεύγη δημόσιου/ιδιωτικού κλειδιού NaCl (Curve25519).

Όπως και το Tessera, αφού συγχρονιστεί με έναν άλλο κόμβο, ανιχνεύει αυτόματα τους άλλους κόμβους του δικτύου. Συγχρονίζει έναν κατάλογο δημόσιων κλειδιών που αντιστοιχίζεται στους κόμβους του δικτύου. Διαθέτει ένα public API, μέσω του οποίου οι κόμβοι μπορούν να στέλνουν κρυπτογραφημένα bytestrings μεταξύ τους και να συγχρονίζονται μοιράζοντας τις πληροφορίες που γνωριζει ο καθε κόμβος για τους υπόλοιπους. Αντίστοιχα, διαθέτει και ένα private API που επιτρέπει την αποστολή ενός bytestring σε ένα ή περισσότερα δημόσια κλειδιά, επιστρέφοντας ένα αναγνωριστικό (content-addressable). Αυτό το κρυπτογραφημένο bytestring μεταδίδεται μονο στους κόμβους που έχουν προκαθοριστεί. Το αναγνωριστικό είναι ένα hash digest του κρυπτογραφημένου payload που λαμβάνει κάθε κόμβος παραλήπτη, μαζί με ένα μικρό blob κρυπτογραφημένο για το δημόσιο κλειδί του το οποίο περιέχει το κύριο κλειδί για το κρυπτογραφημένο payload. Έτσι επιτρέπει την παραλαβή ενός αποκρυπτογραφημένου bytestring βάσει ενός αναγνωριστικού. Τα payloads που έχει αποστείλει ή λάβει ο εκάστοτε κόμβος μπορούν να αποκρυπτογραφηθούν και να ανακτηθούν με αυτόν τον τρόπο. Επίσης εκθέτει μεθόδους για διαγραφή, επανασυγχρονισμό και άλλες λειτουργίες διαχείρισης. Χρησιμοποιεί αμφίδρομα πιστοποιημένο TLS με σύγχρονες ρυθμίσεις και διάφορα μοντέλα εμπιστοσύνης, συμπεριλαμβανομένων των υβριδικών CA / tofu (προεπιλογή), tofu (OpenSSH) και whitelist (ώστε να επιτρέπεται η σύνδεση σε ένα σύνολο από δημόσια κλειδιά). Υποστηρίζει έναν αριθμό αποθηκευτικών backends (LevelDB, BerkeleyDB, SQLite, Directory / Maildir), αλλά και IP whitelisting.

Το Tessera κρίθηκε καταλληλοτερο για την παρούσα διπλωματική εργασία, καθώς παρουσιάζει αυξημένη σταθερότητα και ταχύτητα. [20]

2.6 Σχετικές Εργασίες

Στην ενότητα αυτή θα παρουσιαστούν εργασίες σχετικές με το αντικείμενο της παρούσας διπλωματικής. Έχει προηγηθεί η παρουσίαση του Ethereum blockchain, έτσι λοιπόν εδώ δεν θα παρουσιάζονται εργασίες που αφορούν τον τρόπο λειτουργίας του. Αντιθέτως, παρουσιάζονται εργασίες που ερευνούν τα πιθανά πεδία εφαρμογής του blockchain και πιο συγκεκριμένα τις δυνατότητες συνδυασμού του blockchain με Reputation Management συστήματα, αλλά και με το Internet of Things.

#	**Μοντέλο / Σύστημα**	**Αποκέντρωση**	**Ιδιωτικότητα**	**Smart Contracts**	**Νέο Blockchain**	**Κίνητρα & Ποινές**	**Τομέας**	**Περιορισμοί**
1	MAS	Ναι	Μεσαία	Ναι	Όχι	Ναι	MAS, eHealth	Υποδομή
2	NDN Vehicular	Ναι	Μεσαία	Όχι	Όχι	Ναι	Vehicular	Εστίαση σε caches
3	Proof of Reputation (PoR)	Ναι	Μεσαία	Όχι	Όχι	Ναι	P2P	Απόδοση
4	Trustless Privacy-Preserving	Ναι	Υψηλή	Όχι	Όχι	Ναι	E-commerce	Μη αξιολόγηση reviewers
5	Bitcoin Feedback System	Ναι	Χαμηλή	Ναι	Όχι	Ναι	E-commerce	Κόστος συναλλαγών
6	Rep on the Block	Ναι	Μεσαία	Ναι	Ναι	Ναι	Πολλαπλές	Κόστος, πολυπλοκότητα
7	Trust Arch. for IoT	Ναι	Μεσαία	Όχι	Ναι	Ναι	IoT	Εμπιστοσύνη δεδομένων
8	BARS (Anonymous Reput. Sys.)	Ναι	Υψηλή	Όχι	Ναι	Ναι	VANETs	Διαχείριση, πολυπλοκότητα

Πίνακας 2-1 Σύγκριση εργασιών

Σημειώσεις:

- Αποκέντρωση:
  Όλα τα συστήματα βασίζονται σε αποκεντρωμένα ή κατανεμημένα μοντέλα.
- Ιδιωτικότητα:
  Υψηλή όπου διατηρείται ανωνυμία/ψευδωνυμία, Μεσαία όταν υπάρχουν κάποια δεδομένα ορατά, Χαμηλή σε πλήρως ανοιχτές υλοποιήσεις.
- Smart Contracts:
  Αν γίνεται χρήση ή όχι για αυτοματοποίηση reputation/συναλλαγών.
- Νέο Blockchain:
  Αν δημιουργήθηκε νέο blockchain αποκλειστικά για το σύστημα.
- Κίνητρα & Ποινές:
  Αφορά ύπαρξη μηχανισμών αποτροπής κακόβουλης συμπεριφοράς.
- Τομέας:
  Ενδεικτικός τομέας εφαρμογής κάθε μοντέλου.
- Περιορισμοί:
  Κυριότερος τεχνικός ή λειτουργικός περιορισμός κάθε προσέγγισης.

2.6.1 Reputation Management in Multi-Agent Systems using Permissioned Blockchain Technology

Multi-agent systems (MAS) χρησιμοποιούνται όλο και περισσότερο σε τομείς στους οποίους η ασφάλεια και η αξιοπιστία είναι μεγίστης σημασίας, όπως eHealth, cyber-physical systems, financial services και energy market. Με σκοπό τη δημιουργία εμπιστοσύνης μεταξύ των συμμετεχόντων και τη διαχείριση του reputation του καθενός, δημιουργήθηκαν μηχανισμοί οι οποίοι συνδυάζουν τα MAS με το Blockchain. Το σύστημα που προκύπτει, εξαλείφει την ύπαρξη ενός “μεμονωμένου σημείου αποτυχίας”, ενώ ταυτόχρονα επιτρέπει την αλληλεπίδραση μεταξύ των χρηστών, καταγράφοντας τα transactions στο distributed ledger με τρόπο που διασφαλίζει ότι κάθε transaction ειναι αναλλοίωτο και ότι ο καθένας μπορεί να το επιβεβαιώσει. Τα transactions τα οποία τροποποιούν το reputation ενός χρήστη είναι προσβάσιμα σε κάθε άλλο χρήστη, επιτρέποντάς του να υπολογίσει οποιαδήποτε στιγμή όποιο reputation επιθυμεί, αλλά και να παρατηρήσει τη μεταβολή του μετά από κάθε συναλλαγή. Για τον υπολογισμό αλλά και την τροποποίηση του reputation χρησιμοποιούνται smart contracts, ενώ η πληροφορία αποθηκεύεται σε immutable distributed ledger. [21]

2.6.2 Reputation-based Blockchain for Secure NDN Caching in Vehicular Networks

Εδώ παρουσιάζεται ενας reputation-based Blockchain mechanism, με στόχο την αύξηση της ασφάλειας και της εμπιστοσύνης μεταξύ των caches και των καταναλωτών στο vehicular environment. Το σύστημα, που βασίζεται στο Blockchain, αναθέτει σε κάθε cache store ένα reputation value το οποίο μεταβάλλεται αναλογα με την ικανοποιηση των καταναλωτων για τα δεδομένα που προσφέρει. [22]

2.6.3 Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to- Peer Network

Τα ήδη υπάρχοντα reputation systems οποία σχετίζονται με Blockchain έχουν δημιουργηθεί με βάση Bitcoin-like Blockchains, επομένως έχουν εγγενή προβλήματα χαμηλής απόδοσης και υψηλής κατανάλωσης. Για να ξεπεραστεί αυτός ο περιορισμός, δημιουργήθηκε ένα reputation-based consensus protocol που ονομάζεται Proof of Reputation (PoR), το οποίο χρησιμοποιεί το ίδιο το reputation ως κίνητρο για τους συμμετέχοντες για να εξασφαλίσει την έντιμη συμπεριφορά τους και να πετύχει ταυτόχρονα την δημιουργία των blocks χωρίς mining ή χρηματική επιβράβευση. Οι συμμετέχοντες στο σύστημα έχουν πρόσβαση σε ένα distributed ledger of reputation από το οποίο μπορούν να αναζητήσουν το reputation κάθε χρήστη, διατηρώντας έτσι αποτελεσματικά την ακεραιότητα και την αξιοπιστία του συστήματος. [23]

2.6.4 A trustless privacy-preserving reputation system

Σε εφαρμογές, όπως για παράδειγμα το ηλεκτρονικό εμπόριο, στις οποίες η συμπεριφορά των χρηστών καταγράφεται και είναι σημαντική για τη εύρυθμη λειτουργία της εφαρμογής, η χρησιμοποίηση Reputation systems κρίνεται απαραίτητη.

Απο τα υπάρχοντα συστήματα που έχουν προταθεί, κανένα δεν καταφέρνει να διατηρεί την ιδιωτικότητα των χρηστών διατηρώντας το σύστημα αποκεντρωμένο, αλλά και ταυτόχρονα αξιόπιστο. Το σύστημα που προτείνεται χρησιμοποιεί το Blockchain, ωστε να διασφαλίσει την ιδιωτικότητα των χρηστών, επιτρέποντάς τους να αξιολογούν ειλικρινά, χωρίς να φοβούνται αντίποινα, λόγω της αρνητικής βαθμολόγησης, αφού δεν θα μπορεί κανείς να γνωρίζει την πραγματική τους ταυτότητα. Αξίζει όμως να αναφερθεί ότι δεν υπάρχει δυνατότητα αξιολόγησης των βαθμολογητών (μόνο οι χρήστες-πελάτες μπορούν να βαθμολογήσουν). Παράλληλα θα δίνονται κίνητρα στους χρήστες να συμπεριφέρονται έντιμα, ενώ έχουν ελαχιστοποιηθεί τα οφέλη των χρηστών με επανειλημμένη μη έντιμη συμπεριφορά. [24]

2.6.5 Feedback based Reputation on top of the Bitcoin Blockchain

Η δημιουργία ενός reputation system το οποίο βασίζεται πάνω στο Bitcoin Blockchain, είναι εξαιρετικά χρήσιμο σε web communities, αλλά και στο ηλεκτρονικό εμπόριο, αφού επιτρέπει την διαχείριση του feedback της κάθε αλληλεπίδρασης. Τα συστήματα που υπήρχαν έως τώρα ήταν μεν αξιόπιστα, αλλά centralised, ενώ το σύστημα στο οποίο αναφερόμαστε ειναι decentralised αλλα και κατανεμημένο, χωρίς δηλαδή να υπάρχει ένα κεντρικό σημείο ελέγχου. Το σύστημα δίνει κίνητρα στους συμμετέχοντες ώστε να συμπεριφέρονται έντιμα, καθώς μετά από κάθε συναλλαγή στέλνεται ένα voucher που δίνει τη δυνατότητα αξιολόγησης της συγκεκριμένης συναλλαγής με αντάλλαγμα ένα μικρό ποσοστό της, ανεξάρτητα με το περιεχόμενο της. Το feedback που προκύπτει από κάθε συναλλαγή που αξιολογείται αποθηκεύεται στο Blockchain με ασφάλεια, ενώ εξασφαλίζεται έτσι ότι δεν μπορεί κάποιος να τροποιήσει τα αποθηκευμένα δεδομένα. Το reputation κάθε χρήστη δεν υπολογίζεται μετά από κάθε αξιολόγηση, αλλά δίνεται η δυνατότητα στον καθένα να το υπολογίσει εκτός Blockchain, φιλτράροντας τις συναλλαγές τις οποίες επιθυμεί να συμπεριλάβει. Οι κακόβουλοι λογαριασμοί, που πιθανόν να χρησιμοποιηθούν σε επιθέσεις αλλοίωσης του reputation, περιορίζονται λόγω του υψηλού κόστους δημιουργίας πολλαπλών λογαριασμών αλλά και των fee των συναλλαγών. [25]

2.6.6 Rep on the block : A next generation reputation system based on the Blockchain

Το Reputation system το οποίο παρουσιάζεται εδώ, βασιζόμενο στη τεχνολογία του Blockchain, μπορεί να χρησιμοποιηθεί σε πολλά δίκτυα, στα οποία τα μέχρι τώρα συστήματα δεν μπορούν να καλύψουν με επιτυχία όλες τις απαιτήσεις. Η δημιουργία ενός νέου Blockchain, του οποίου ο αποκλειστικός σκοπός θα είναι η αποθήκευση των τιμών του Reputation, αλλά και των συναλλαγών που το μεταβάλλουν, αυξάνει την ασφάλεια του συστήματος, καθώς πλέον οποιαδήποτε επίθεση θα πρέπει να είναι επιτυχημένη και στα 2 Blockchains, πράγμα αρκετά απίθανο. Το σύστημα απαιτεί από τους χρήστες να συνδέσουν τον λογαριασμό τους με μια διεύθυνση IP, ώστε να αποφευχθούν οι επιθέσεις πολλαπλών λογαριασμών. Πριν από κάθε συναλλαγή οι συμμετέχοντες δεσμεύουν ένα ποσό, το οποίο λειτουργεί ως κίνητρο για να συμπεριφέρονται έντιμα, καθώς σε διαφορετική περίπτωση δίνεται ως fee στους miners.

Για να επιβεβαιωθεί οποιαδήποτε συναλλαγή από τους miners, δηλαδή τους χρήστες τους οποίους αφορά, πρέπει να παραμείνουν online, κάτι που ίσως αποτελεί μειονέκτημα για ορισμένες εφαρμογές. [26]

2.6.7 A Trust Architecture for Blockchain in IoT

Το μοντέλο εμπιστοσύνης που προτείνεται μπορεί να εφαρμοστεί σε ένα ευρύ φάσμα εφαρμογών IoT που βασίζονται στο Blockchain, στο οποίο, για παράδειγμα θα αποθηκεύονται οι παρατηρήσεις συστημάτων healthcare ή social media analysis. Η υλοποίηση του μοντέλου έχει γίνει σε custom private Blockchain, αλλά δίνεται η δυνατότητα και για ανάλογες υλοποιήσεις σε major public (Ethereum) και private Blockchain platforms. Πετυχαίνει τη δημιουργία εμπιστοσύνης μεταξύ των χρηστών, χρησιμοποιώντας distributed consensus mechanisms αλλά και αποθηκεύοντας τα δεδομένα σε Blockchain, εξασφαλίζοντας ότι δεν θα αλλοιωθούν ή θα διαγραφούν. Το σύστημα όμως δεν εγγυάται για την αξιοπιστια των δεδομένων που παράγει κάθε ξεχωριστός sensor, για αυτό και δίνει την δυνατότητα να αναφέρεται και η τιμή του confidence για κάθε παρατήρηση, ώστε να λαμβάνεται υπόψη στον υπολογισμό του reputation και της ποινής/επιβράβευσης αναλογικά. Καθένα από τα gateways μπορεί να διασταυρώνει τα δεδομένα γειτονικών sensors που βρίσκονται κάτω από την επίβλεψή του, τροποποιώντας το reputation τους ανάλογα με την ποιότητα των παρατηρήσεών τους. [27]

2.6.8 BARS: a Blockchain-based Anonymous Reputation System for Trust Management in VANETs

Τα ζητήματα της εμπιστοσύνης και της ιδιωτικότητας στα VANETS μπορούν να αντιμετωπιστούν με τη χρήση ενός Blockchain-based anonymous reputation system (BARS). Το σύστημα αυτό θα εμποδίζει την μετάδοση κακόβουλων μηνυμάτων μεταξύ των χρηστών, προστατεύοντας ταυτόχρονα την ταυτότητα τους. Η Certificate Authority θα αντιστοιχίζει την ταυτότητα κάθε νέου χρήστη με ένα public key, που θα χρησιμοποιείται ως ψευδώνυμο, προστατεύοντας, όμως, τα προσωπικά του δεδομένα. Η πιστοποίηση των χρηστών θα γίνεται μέσω 2 διαφορετικών Blockchains. Το reputation θα μπορεί να υπολογιστεί από τις προηγούμενες αλληλεπιδράσεις που έχουν αποθηκευτεί στο σύστημα και είναι προσβάσιμες σε όλους, ενώ οι χρήστες θα έχουν κίνητρα να φέρονται έντιμα. [28]

2.7 Ανάλυση State of the Art

Οι παραπάνω εργασίες διαφοροποιούνται ως προς το βαθμό αποκέντρωσης, την ιδιωτικότητα, τη χρήση έξυπνων συμβολαίων και την ύπαρξη νέου blockchain. Οι περισσότερες προσεγγίσεις ακολουθούν αποκεντρωμένο μοντέλο [1-7], λίγες από τις υλοποιήσεις υποστηρίζουν ιδιωτικά blockchains[4,8], ενώ μία βασίζεται σε permissioned blockchain[1]. Η χρήση έξυπνων συμβολαίων δεν είναι καθολική, αλλά αξιοποιούνται σε

αρκετές περιπτώσεις[1,5,6], ενώ όπου η υπάρχουσα υποδομή κρίνεται ανεπαρκής, προτιμήθηκε η δημιουργία ενός νέου blockchain[3,6,7,8]. Οι περισσότερες προτάσεις ενσωματώνουν μηχανισμούς ποινών και κινήτρων[1,3-8] για την ενίσχυση της αξιοπιστίας, ενώ η αξιολόγηση των αξιολογητών εμφανίζεται σε περιορισμένες λύσεις[1,8], βελτιώνοντας τη διαφάνεια και την ποιότητα του συστήματος.

Καινοτομίες

Είναι εμφανής η μεγάλη ποικιλία προσεγγίσεων και εφαρμογών που προσφέρει το Blockchain στα Reputation Management συστήματα. Κάθε εργασία εστιάζει σε διαφορετικούς τομείς, από τα Multi-Agent Systems (MAS) και τα vehicular networks έως το e-commerce και το IoT. Ενδιαφέρον προκαλούν οι καινοτομίες που εισάγονται, όπως η ιδιωτικότητα των χρηστών στο "Trustless Privacy-Preserving Reputation System" και το "BARS", η δυνατότητα χρήσης νέου blockchain για αυξημένη ασφάλεια στο "Rep on the Block", και η ευελιξία εφαρμογής του "Trust Architecture for Blockchain in IoT" τόσο σε public όσο και σε private Blockchains. Παράλληλα, στο "Proof of Reputation" και στο "Feedback-based Reputation on top of the Bitcoin Blockchain" προτείνονται μέθοδοι που μειώνουν τα ενεργειακά κόστη και ενισχύουν τη διαφάνεια.

Ανωνυμία και Προστασία Ιδιωτικότητας

Η ανωνυμία και η προστασία της ιδιωτικότητας των χρηστών, αποτελούν προτεραιότητα ορισμενων συστημάτων[4,8], προσφέροντας ψευδωνυμία μέσω μηχανισμών όπως τα public keys και διασφαλίζοντας ότι οι χρήστες μπορούν να αξιολογούν χωρίς φόβο αντιποίνων. Αντίθετα, σε άλλες περιπτώσεις [6] θυσιάζεται η απόλυτη ανωνυμία για την ενίσχυση της ασφάλειας μέσω σύνδεσης με IP διευθύνσεις. Παράλληλα, τα συστήματα που βασίζονται σε περισσότερα από ένα Blockchains [6,8], προσφέρουν αυξημένη αντοχή σε επιθέσεις, καθιστώντας εξαιρετικά δύσκολη την αλλοίωση δεδομένων ή τη διείσδυση από κακόβουλους χρήστες.

Περιορισμοί

Η αξιοποίηση του Blockchain σε Reputation συστήματα αποδεικνύεται εξαιρετικά αποτελεσματική, προσφέροντας αυξημένη ασφάλεια, διαφάνεια και ανθεκτικότητα σε επιθέσεις. Ωστόσο, κάθε προσέγγιση συνοδεύεται από περιορισμούς που εξαρτώνται από την εφαρμογή. Για παράδειγμα, το "Trustless Privacy-Preserving Reputation System" περιορίζει τη δυνατότητα αξιολόγησης των αξιολογητών, ενώ το "Rep on the Block" απαιτεί την παρουσία των χρηστών online για την επικύρωση συναλλαγών, που μπορεί να είναι μη πρακτικό σε ορισμένα σενάρια. Παράλληλα, οι προτάσεις για ειδικά blockchain (όπως στο "Rep on the Block") αυξάνουν την πολυπλοκότητα του

συστήματος αλλά και την ασφάλεια, καθώς επιβάλλουν επιπλέον επίπεδα προστασίας. Τέλος, σε περιβάλλοντα όπως το IoT, η προσθήκη τιμών confidence για τους sensors είναι ένα πολλά υποσχόμενο χαρακτηριστικό, αν και η αξιοπιστία εξαρτάται από τη σωστή διασταύρωση των δεδομένων.

Μελλοντικές Προκλήσεις

Οι εργασίες αυτές δείχνουν ότι το Blockchain προσφέρει μοναδικές δυνατότητες για τη βελτίωση των Reputation συστημάτων, αλλά ταυτόχρονα απαιτούνται περαιτέρω έρευνες για την επίλυση ζητημάτων όπως η ιδιωτικότητα, η επεκτασιμότητα και η πολυπλοκότητα των υποδομών. Η ενσωμάτωση τεχνολογιών όπως τα Distributed Consensus Mechanisms, η αξιοποίηση hybrid blockchain αρχιτεκτονικών, και η δημιουργία πιο δυναμικών incentive systems μπορούν να βελτιώσουν τη λειτουργικότητα και την αποδοχή αυτών των συστημάτων.

Επεκτασιμότητα και Αξιοπιστία

Εξετάζοντας την επεκτασιμότητα, συστήματα όπως το Feedback-based Reputation on top of the Bitcoin Blockchain αξιοποιούν υπάρχουσες υποδομές, μειώνοντας το κόστος ανάπτυξης αλλά ενσωματώνοντας περιορισμούς όπως τα transaction fees. Από την άλλη, λύσεις με custom ή private Blockchains, όπως το A Trust Architecture for Blockchain in IoT, παρέχουν μεγαλύτερη ευελιξία αλλά μπορεί να απαιτούν σημαντική υποδομή και συντήρηση. Ενδιαφέρον έχει και η προσέγγιση του Proof of Reputation (PoR), που αποφεύγει τις παραδοσιακές ενεργοβόρες διαδικασίες mining, ενισχύοντας την απόδοση και την αποδοτικότητα. Τέλος, το ζήτημα της αξιοπιστίας των δεδομένων παραμένει ανοιχτό σε εφαρμογές όπως το IoT, όπου η ποιότητα των δεδομένων εξαρτάται από εξωτερικούς παράγοντες, όπως οι αισθητήρες. Η εισαγωγή μηχανισμών confidence metrics στο σύστημα IoT δείχνει έναν χρήσιμο δρόμο για την ενίσχυση της εμπιστοσύνης στις παρατηρήσεις.

Θέση της Παρούσας Εργασίας

Η εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας αξιοποιεί το Quorum Blockchain για να προσφέρει μια αποκεντρωμένη πλατφόρμα αγοραπωλησίας δεδομένων με σύστημα αξιολόγησης χρηστών. Σε σύγκριση με τις υπάρχουσες λύσεις, ενσωματώνει τις δυνατότητες ιδιωτικότητας του Quorum, αντιμετωπίζοντας ένα βασικό περιορισμό των δημόσιων blockchains. Παράλληλα, παρέχει μια ευέλικτη πλατφόρμα που συνδυάζει δεδομένα blockchain και συμβατικές βάσεις δεδομένων, επιτυγχάνοντας αποδοτική διαχείριση της πληροφορίας. Ενισχύει την ασφάλεια και την αξιοπιστία των συναλλαγών, δίνοντας στους χρήστες τη δυνατότητα αξιολόγησης με διαφάνεια.

Συμπεράσματα

Η ανάλυση καταδεικνύει πως η τεχνολογία Blockchain προσφέρει σημαντικές δυνατότητες για αλλαγές στη διαχείριση δεδομένων και συστημάτων αξιολόγησης. Η προτεινόμενη λύση βασίζεται σε αυτά τα πλεονεκτήματα, αντιμετωπίζοντας βασικά ζητήματα, όπως η κεντρικοποίηση, ενώ είναι κατάλληλη για επέκταση σε διάφορους τομείς.

3 Εργαλεία και τεχνολογίες

Στο κεφάλαιο αυτό παρουσιάζονται τα κυριότερα εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής της παρούσας διπλωματικής εργασίας.

3.1 Ethereum

Το οικοσύστημα του Ethereum παρέχει πολλά χρήσιμα εργαλεία για την ανάπτυξη αποκεντρωμένων εφαρμογών (DApps).

3.1.1 Geth

Το Geth [29] ή Go Ethereum είναι μία διεπαφή γραμμής εντολών ανοιχτού κώδικα, υλοποιημένη στην γλώσσα προγραμματισμού Go, για την εκτέλεση του πλήρους Ethereum κόμβου. Το Geth [30] δίνει τη δυνατότητα στο χρήστη να γίνει μέλος του Ethereum δικτύου και πραγματοποιήσει, μεταξύ άλλων, λειτουργίες όπως:

- Εξόρυξη ether.
- Μεταφορά ether μεταξύ διευθύνσεων.
- Δημιουργία συμβολαίων (Smart Contracts).
- Εξερεύνηση του Ethereum Blockchain.
- Δημιουργία ενός ιδιωτικού Ethereum Blockchain.

3.1.2 Solidity

Η Solidity [31] είναι μία αντικειμενοστρεφής, υψηλού επιπέδου γλώσσα προγραμματισμού και χρησιμοποιείται για την υλοποίηση smart contracts που εκτελούνται στο Ethereum Virtual Machine (EVM). Δημιουργήθηκε με στόχο την αύξηση των δυνατοτήτων της EVM, ενώ ο ίδιος ο πηγαίος κώδικας της γλώσσας Ethereum είναι γραμμένος σε Solidity. Η γλώσσα είναι επηρεασμένη από τη C++, την Python και τη JavaScript και καθώς έχει παρόμοιο συντακτικό με αυτό της JavaScript, είναι εύκολα κατανοήσιμη από τους περισσότερους προγραμματιστές.

Είναι μια γλώσσα στατικού τύπου και μερικά από τα χαρακτηριστικά της είναι ότι υποστηρίζει κληρονομικότητα, βιβλιοθήκες και περίπλοκους τύπους ορισμένους από τον προγραμματιστή.

Στην παρούσα διπλωματική εργασία χρησιμοποιήθηκε η έκδοση: 0.5.7

3.1.3 Web3

Το web3.js [32] είναι ένα από τα πιο σημαντικά npm πακέτα που χρησιμοποιούνται για την δημιουργία μιας DApp. Αποτελεί μία συλλογή βιβλιοθηκών που επιτρέπουν την αλληλεπίδραση με έναν τοπικό ή απομακρυσμένο κόμβο του Ethereum Blockchain, χρησιμοποιώντας HTTP ή IPC σύνδεση [33]. Στην ουσία είναι μία διεπαφή επικοινωνίας (API) μεταξύ της JavaScript και του Ethereum Blockchain, δηλαδή ένας τρόπος για να αναφέρεται η JavaScript, είτε από την πλευρά του Server είτε από την πλευρά του Front-end, στα διάφορα “αντικείμενα” που υπάρχουν στο Blockchain, όπως για παράδειγμα τα έξυπνα συμβόλαια (Smart Contracts), τα δεδομένα τους, τις συναρτήσεις τους, τις διευθύνσεις και τα υπόλοιπα λογαριασμών. [34]

Στην παρούσα διπλωματική εργασία χρησιμοποιήθηκε η έκδοση: 1.2.2

3.1.4 Truffle

Το Truffle [35] είναι μια κορυφαία πλατφόρμα, η οποία προσφέρει πληθώρα εργαλείων για ανάπτυξη, δοκιμές και deployment σε Blockchains που χρησιμοποιούν την Ethereum Virtual Machine (EVM). Οι προγραμματιστές χρησιμοποιώντας το npm πακέτο του Truffle κάνουν τη διαδικασία της ανάπτυξης πιο εύκολη και γρήγορη [36] . Μερικά από τα σημαντικότερα χαρακτηριστικά του είναι:

- Δυνατότητα για compilation, linking, deployment και binary management των έξυπνων συμβολαίων.
- Αυτοματοποιημένο testing των έξυπνων συμβολαίων για επιτάχυνση της ανάπτυξης.
- Δυνατότητα αυτοματοποίησης του deployment και του migration των έξυπνων συμβολαίων μέσω scripting.
- Δυνατότητα διαχείρισης δικτύου για deployment σε πλήθος δημόσιων ή ιδιωτικών δικτύων.
- Διαχείριση πακέτων μέσω EthPM και NPM, χρησιμοποιώντας το πρότυπο ERC190.
- Διαδραστική console διεπαφή για άμεση επικοινωνία με τα έξυπνα συμβόλαια.
- Παραμετροποιήσιμο build pipeline με υποστήριξη για προσαρμοσμένα build processes.
- Δυνατότητα εκτέλεσης script στο περιβάλλον του Truffle απο εξωτερικό script runner.

3.1.5 Ganache

Το Ganache [37] είναι ένα npm πακέτο και επομένως μπορεί να εγκατασταθεί εύκολα με τη χρήση του Node Package Manager. Αποτελεί ένα εργαλείο που τυπικά περιλαμβάνεται στη σουίτα του Truffle. Είναι διαθέσιμο τόσο ως CLI (Command Line Interface), δηλαδή ως διεπαφή γραμμής εντολών, όσο και σε GUI (Graphical User Interface), δηλαδή γραφική διεπαφή χρήστη.

Το Ganache, έχοντας βασιστεί στο Node.js Ethereum client για δοκιμή και ανάπτυξη, προσομοιώνει έναν πλήρη Ethereum κόμβο χρησιμοποιώντας τις ethereumjs βιβλιοθήκες. Παρέχει στον προγραμματιστή πολλές δυνατότητες και τον διευκολύνει, καθιστώντας την ανάπτυξη έξυπνων συμβολαίων (Smart Contracts) και αποκεντρωμένων εφαρμογών (DApps) πολύ πιο απλή, γρήγορη και ασφαλής, καθώς τον απαλλάσσει από την επιβάρυνση της εκτέλεσης ενός πραγματικού Ethereum κόμβου, δημιουργώντας ένα ιδιωτικό Ethereum Blockchain. Περιλαμβάνει όλες τις δημοφιλείς RPC (Remote Procedure Call) συναρτήσεις και δυνατότητες (π.χ. events) και μπορεί να εκτελεστεί αιτιοκρατικά, επιταχύνοντας έτσι την διαδικασία ανάπτυξης. Επομένως γίνεται κατανοητό ότι το Ganache είναι ένα απαραίτητο εργαλείο για τον προγραμματιστή, καθώς διαφορετικά θα έπρεπε η ανάπτυξη και οι δοκιμές να πραγματοποιηθούν στο πραγματικό Ethereum δίκτυο είτε σε κάποιο test net. Σε μία τέτοια περίπτωση θα προέκυπταν διάφορα προβλήματα, όπως η αποθήκευση περίσσειας πληροφορίας στο δημόσιο Ethereum δίκτυο, η επιβάρυνση του με μη παραγωγική κίνηση, μεγάλα διαστήματα αναμονής αλλά και αυξημένο κόστος ανάπτυξης.

Όταν εκκινείται το Ganache [38] δημιουργεί ένα τοπικό Ethereum Blockchain με 10 λογαριασμούς, οι οποίοι χρησιμοποιούνται για τις δοκιμές που γίνονται κατα την ανάπτυξη της DApp.

3.1.6 EthereumJS

Το EthereumJS [39] είναι το JavaScript project του Ethereum Foundation, στην ανάπτυξη και τη συντήρηση του οποίου συμβάλει σημαντικά η κοινότητά του. Αποτελεί μια μεγάλη συλλογή βιβλιοθηκών και εργαλείων για την αλληλεπίδραση με το Ethereum δίκτυο, μέσω της γλώσσας JavaScript. Περιλαμβάνει πολλά χρήσιμα modules για την ανάπτυξη εφαρμογών για το Ethereum.

Το npm πακέτο ethereumjs-util [40] έχει συμπληρωματικό ρόλο, αποτελώντας μια βιβλιοθήκη βοηθητικών συναρτήσεων για το Ethereum . Περιλαμβάνει συναρτήσεις που υλοποιούν τον αλγόριθμο ψηφιακής υπογραφής ελλειπτικής καμπύλης (elliptic curve digital signature algorithm), ώστε να είναι δυνατή η πιστοποίηση των διευθύνσεων και χρηστών στο Blockchain. Επίσης χρησιμοποιήθηκε το πακέτο keythereum [41] το οποίο

περιέχει συναρτήσεις που επιτρέπουν την εύκολη δημιουργία, εισαγωγή και εξαγωγή Ethereum κλειδιών. Τα πακέτα όπως τα προαναφερθέντα προορίζονται για χρήση στο server-side, αλλά όμως χρησιμοποιώντας το npm πακέτο browserify [42] είναι δυνατή η δημιουργία κατάλληλων αρχείων που επιτρέπουν τη χρήση τους και στο front-end.

3.2 NPM

Το npm (Node Package Manager) [43] είναι ένα πρόγραμμα διαχείρισης πακέτων της γλώσσας JavaScript. Αποτελεί ένα online repository για τη δημοσίευση Node.js projects ανοιχτού κώδικα. Χαρακτηρίζεται ως το μεγαλύτερο μητρώο προγραμμάτων στον κόσμο και καθώς αποτελεί το προκαθορισμένο πρόγραμμα διαχείρισης πακέτων του προγραμματιστικού περιβάλλοντος Node.js, καθημερινά δημοσιεύονται σε αυτό πληθώρα βιβλιοθηκών και εφαρμογών του Node.js. Προσφέρει τη δυνατότητα αλληλεπίδρασης, μέσω γραμμής εντολών, με το repository, συνεισφέροντας στην εγκατάσταση πακέτων, στη διαχείριση εκδόσεων και στη διαχείριση των εξαρτήσεων (dependencies) ενός project.

Η χρήση του απλουστεύει σημαντικά την ανάπτυξη εφαρμογών [44] , καθώς επιτρέπει στον προγραμματιστή να αναζητήσει αρχικά το πακέτο ή την εφαρμογή που επιθυμεί στην ιστοσελίδα http://npmjs.org/, έχοντας τη δυνατότητα να ολοκληρώσει την εγκατάσταση με μία μόνο εντολή, χρησιμοποιώντας τη διεπαφή γραμμής εντολών (Command Line Interface). Το πακέτο (package) είναι ένα αρχείο ή ένα directory το οποίο περιγράφεται από ένα package.json, η ύπαρξη του οποίου είναι απαραίτητη προϋπόθεση για να δημοσιευτεί το πακέτο στο μητρώο (registry) του npm. Κάθε πακέτο εγκαθίσταται ως ένα module και φορτώνεται από το Node.js μέσω της εντολής require() του JavaScript κώδικα.

Η χρησιμότητα του για την κοινότητα ανάπτυξης λογισμικού είναι τεράστια, αφού επιτρέπει την ανταλλαγή πακέτων, δηλαδή κώδικα που αποτελεί πρακτικά ένα “εργαλείο” με συγκεκριμένο σκοπό, ενώ παράλληλα καθορίζει συγκεκριμένες προδιαγραφές για κάθε πακέτο που δημοσιεύεται στο αρχείο, έτσι ώστε να κάνει τη χρήση τους πιο εύκολη, ελαχιστοποιώντας τα προβλήματα.

Στη παρούσα εργασία χρησιμοποιήθηκαν αρκέτα πακέτα του npm, τα βασικότερα εκ των οποίων αναφέρονται και ξεχωριστά στο κεφάλαιο αυτό.

Χρησιμοποιήθηκε η έκδοση 6.12.0 του NPM.

3.3 Nodejs

Το Node.js [45] είναι μία server-side πλατφόρμα, η οποία έχει αναπτυχθεί πάνω στο Google Chrome JavaScript Engine (V8 Engine) . Είναι ένα ανοιχτού κώδικα (open source), cross-platform runtime περιβάλλον της γλώσσας προγραμματισμού JavaScript, κατάλληλο για την εύκολη ανάπτυξη γρήγορων και κλιμακώσιμων (scalable) διαδικτυακών εφαρμογών. Η αποτελεσματικότητά του στο να πετυχαίνει μεγάλη απόδοση χρησιμοποιώντας λίγους φυσικούς πόρους το καθιστά ιδανικό για υπολογιστικά έντονες εφαρμογές πραγματικού χρόνου που τρέχουν σε κατανεμημένα περιβάλλοντα. Το Node.js απλοποιεί σημαντικά την διαδικασία ανάπτυξης διαδικτυακών εφαρμογών, καθώς εκτός από τη χρησιμότητα του ως περιβάλλον εκτέλεσης, παρέχει και μία μεγάλη βιβλιοθήκη από JavaScript modules. [46]

Μερικά από τα πιο σημαντικά χαρακτηριστικά του Node.js είναι τα εξής:

- Asynchronous και Event-Driven
  Όλες οι προγραμματιστικές διεπαφές (APIs) της Node.js βιβλιοθήκης είναι ασύγχρονες, δηλαδή ο κώδικας που εκτελείται σε έναν Node.js server δεν σταματά για να περιμένει μία κληθείσα διεργασία να επιστρέψει δεδομένα, αλλά συνεχίζει την εκτέλεση και όταν η διεργασία αυτή ολοκληρωθεί, ο server λαμβάνει την απάντηση, μέσω ενός μηχανισμού ενημέρωσης συμβάντων (notification mechanism of Events) .

- Ταχύτητα εκτέλεσης
  Η βιβλιοθήκη Node.js είναι πολύ γρήγορη στην εκτέλεση κώδικα, καθώς έχει βασιστεί στην Google Chrome V8 JavaScript Engine.

- Single Threaded, αλλά κλιμακώσιμο (Scalable)
  Το Node.js χρησιμοποιεί μοντέλο μονού νήματος εκτέλεσης με event looping. Ο μηχανισμός συμβάντων που χρησιμοποιείται βοηθάει τον server να απαντάει με non-blocking τρόπο, κάτι που του δίνει την ιδιότητα της κλιμακωσιμότητας, σε αντίθεση με την εναλλακτική επιλογή της χρησιμοποίησης περισσότερων του ενός νημάτων για την εξυπηρέτηση πολλαπλών αιτήσεων. Επομένως το Node.js είναι τόσο βελτιστοποιημένο που ενώ χρησιμοποιεί μονονηματική αρχιτεκτονική, δεν υπολείπεται καθόλου σε απόδοση και έχει τη δυνατότητα να εξυπηρετήσει μεγαλύτερο αριθμό αιτήσεων από παραδοσιακούς Servers, όπως είναι ο ApacheHTTP.

- No Buffering

Οι εφαρμογές του Node.js δεν κάνουν προσωρινή αποθήκευση δεδομένων (buffering), αλλά στέλνουν τα δεδομένα σε μικρά κομμάτια.

Τα πλεονεκτήματα που αναφέρθηκαν καθιστούν το Node.js ένα πολύ χρήσιμο εργαλείο για πολλούς προγραμματιστές. Η ανάπτυξη, η υποστήριξη και η διάθεσή του γίνονται από την open-source community, στην οποία μπορεί να συμμετάσχει ο καθένας που επιθυμεί να συμβάλλει. [47]

Χρησιμοποιήθηκε η έκδοση 12.1.0.

3.3.1 Google V8 engine

Η ιδεατή μηχανή V8 της Google [48] είναι μία ανοικτού κώδικα μηχανή JavaScript που δημιουργήθηκε από το The Chromium Project. Εκτελεί κώδικα που γραμμένος στην γλώσσα JavaScript και χρησιμοποιείται τόσο από τον ανοικτού κώδικα browser Google Chrome, όσο και από το Node.js και πολλές άλλες εφαρμογές. Η V8 μπορεί να εκτελεστεί είτε ξεχωριστά, είτε ενσωματωμένη σε κάποια C++ εφαρμογή. Υλοποιεί το ECMAScript και το WebAssembly, ενώ τρέχει σε Windows 7 ή μεταγενέστερα, macOS 10.12+, και Linux συστήματα που χρησιμοποιούν x64, IA-32, ARM ή MIPS επεξεργαστές.

3.4 Ανάπτυξη ιστοσελίδων

Σε αυτήν την ενότητα παρουσιάζονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη των ιστοσελίδων της εφαρμογής. Όπως και στη δημιουργία κάθε ιστοσελίδας παγκοσμίως, χρησιμοποιήθηκε η γλώσσα HTML (Hypertext Markup Language), η CSS (Cascading Style Sheets) για το styling της σελίδας και η γλώσσα JavaScript τόσο για το server-side κομμάτι όσο και για το front-end. Οι τεχνολογίες αυτές είναι οι σημαντικότερες τεχνολογίες του World Wide Web.

3.4.1 JavaScript

Η JavaScript [49] είναι μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού, καθώς όλοι οι σύγχρονοι browsers την υποστηρίζουν και χρησιμοποιείται εκτενώς από τις περισσότερες διαδικτυακές εφαρμογές. Αποτελεί τον καλύτερο τρόπο να δώσουμε δυναμικό περιεχόμενο σε μία ιστοσελίδα κάνοντάς την να επιτελεί σύνθετες λειτουργίες. Αν και αρχικά χρησιμοποιούνταν μόνο σε web browsers, οι νεότερες εικονικές μηχανές και περιβάλλοντα ανάπτυξης JavaScript (όπως το Node.js) την καθιστούν μια δημοφιλή

επιλογή για την ανάπτυξη εφαρμογών [50] στην πλευρά του server (server-side) όπως και για άλλες εφαρμογές, οι οποίοι διαθέτουν μία JavaScript engine. Η JavaScript χρησιμοποιείται επίσης σε εφαρμογές εκτός ιστοσελίδων, όπως για παράδειγμα τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι εφαρμογές της επιφάνειας εργασίας (widgets).

Για την εκτέλεσή της είναι απαραίτητη κάποια μηχανή JavaScript (JavaScript engine), όπως η Google V8. Η JavaScript engine, που είναι απαραίτητη σε κάθε web browser, είναι πρόγραμμα (διερμηνέας) που εκτελεί κώδικα JavaScript.

Πρόκειται για μία υψηλού επιπέδου, δυναμική, prototype-based, interpreted, multi-paradigm γλώσσα προγραμματισμού. [51] Ως multi-paradigm γλώσσα, υποστηρίζει τα event-driven, functional και imperative προγραμματιστικά στυλ. Η JavaScript έχει διαθέσιμες προγραμματιστικές διεπαφές (application programming interfaces - APIs) που διευκολύνουν τη διαχείριση κειμένου, ημερομηνιών, δομών δεδομένων αλλά και του Document Object Model (DOM). Η JavaScript δεν επιτρέπει, για λόγους ασφαλείας, το γράψιμο και το διάβασμα αρχείων στην πλευρά του χρήστη (client-side). Επιπλέον δε μπορεί να χρησιμοποιηθεί για εφαρμογές που απαιτούν πολυνηματικές διεργασίες ή που χρειάζεται να εκμεταλλευτούν περισσότερους του ενός επεξεργαστές.

Τα βασικότερα πλεονεκτήματα της χρήσης JavaScript στις διαδικτυακές εφαρμογές είναι τα εξής:

- Λιγότερη αλληλεπίδραση με τον εξυπηρετητή.
  Κάνοντας προεπεξεργασία (preprocessing) των δεδομένων στον browser του χρήστη (client), πριν υποβληθούν στον server, επιτυγχάνει τη μείωση του φόρτου εργασίας του server.

- Άμεση ανατροφοδότηση στο χρήστη.
  Μετά από κάθε αλληλεπίδραση με το χρήστη, ανταποκρίνεται άμεσα είτε εμφανίζοντας κάποιο ενημερωτικό μήνυμα (notification) είτε αλλάζοντας το περιεχόμενο που παρουσιάζει, τροποποιώντας τη μορφοποίηση της, προσθέτοντας νέα κείμενα/πεδία κτλ.

- Αυξημένη διαδραστικότητα ιστοσελίδας.
  Επιτρέπει τη δημιουργία διεπαφών που δίνουν τη δυνατότητα για δυναμική τροποποίησή της ιστοσελίδας, ανάλογα με τις κινήσεις του χρήστη, χωρίς να χρειάζεται να γίνει ανανέωσή της.

- Πλουσιότερες διεπαφές.
  Εμπλουτίζει την HTML και τα CSS, κάνοντας διαθέσιμα στον χρήστη πολλά νέα δυναμικά στοιχεία, βελτιώνοντας έτσι την εμφάνιση της ιστοσελίδας. [52]

3.4.2 HTML

Η γλώσσα HTML (Hypertext Markup Language) [53] είναι η βασική γλώσσα σήμανσης για τις ιστοσελίδες και τα στοιχεία της (HTML elements) αποτελούν τα δομικά στοιχεία όλων των ιστοσελίδων. Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags) οι οποίες περικλείονται μέσα στα σύμβολα “<” και ”>”. Οι ετικέτες λειτουργούν ανα ζεύγη (π.χ <h1> και </h1>) και χρησιμοποιούνται για επικεφαλίδες, παραγράφους, εικόνες, υπερσυνδέσμους κτλ. Ο φυλλομετρητής αξιοποιεί τις ετικέτες, χωρίς βέβαια να τις εμφανίζει, ώστε δομήσει σωστά τη σελίδα και να παρουσιάσει το περιεχόμενό της με τον κατάλληλο τρόπο. Η HTML χρησιμοποιείται συνήθως συνδυαστικά με την CSS και την JavaScript. [54]

3.4.3 CSS

Η CSS (Cascading Style Sheets) [55] είναι μια γλώσσα ύφους φύλλων (style sheet language) και χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης (HTML). Ο ρόλος της CSS, που την κάνει και απαραίτητη για την δημιουργία μιας όμορφης και καλοσχεδιασμένης ιστοσελίδας, είναι ότι δίνει τη δυνατότητα της στυλιστικής διαμόρφωσης μιας ιστοσελίδας, προσφέροντας επιλογές χρωμάτων, στοίχισεως και άλλων χαρακτηριστικών. Συμβάλλει στην δημιουργία προσαρμοστικών ιστοσελίδων, οι οποίες αλλάζουν δυναμικά τον τρόπο με τον οποίο παρουσιάζεται το περιεχόμενό τους, ανάλογα το μέγεθος της οθόνης στην οποία προβάλλονται. Η δυνατότητα προσαρμογής είναι πλέον αναγκαία αφού όλο και περισσότεροι χρήστες χρησιμοποιούν, εκτός από υπολογιστές, κινητά και tablet για την περιήγησή τους στο διαδίκτυο. Στην παρούσα εργασία χρησιμοποιήθηκε το Bootstrap, το οποίο αποτελεί ένα έτοιμο πλαίσιο ανάπτυξης ιστοσελίδων.

3.4.4 Bootstrap

Το Bootstrap [56] είναι ένα ελεύθερο, ανοιχτού κώδικα πλαίσιο ανάπτυξης ιστοσελίδων και διαδικτυακών εφαρμογών (front-end web framework). Αποτελεί ένα συνδυασμό από HTML, CSS, και JavaScript κώδικα, σχεδιασμένο να συμβάλλει στην ταχύτερη και αποτελεσματικότερη ανάπτυξη ιστοσελίδων.

Κάνει την διαδικασία σχεδίασης και ανάπτυξης ιστοσελίδων πιο εύκολη και πιο γρήγορη, ενώ παράλληλα τις καθιστά περισσότερο λειτουργικές, με πιο όμορφες και αποτελεσματικές διεπαφές χρήστη. Δίνει τη δυνατότητα στον προγραμματιστή,

χρησιμοποιώντας τα CSS που παρέχονται από το Bootstrap, να δημιουργεί προσαρμοστικές ιστοσελίδες, δηλαδή ιστοσελίδες οι οποίες αλλάζουν δυναμικά τον τρόπο παρουσίασης του περιεχομένου τους ανάλογα το μέγεθος και τις διαστάσεις της οθόνης στην οποία προβάλλονται.

Το Bootstrap είναι ένα από τα πιο διαδεδομένα front-end frameworks, καθώς είναι εύκολο στη χρήση, αφού απαιτούνται μόνο στοιχειώδεις γνώσεις HTML και CSS για να χρησιμοποιηθεί. Επιπλέον, ένα σημαντικό πλεονέκτημα είναι η συμβατότητα του με όλους τους δημοφιλείς φυλλομετρητές (browsers), όπως Google Chrome, Firefox, Microsoft Edge, Internet Explorer, Opera και Safari. Διαθέτει πολύ καλό σύστημα πλέγματος, δηλαλή τα στοιχεία της σελίδας μπορούν εύκολα να οργανωθούν σε στήλες και επίσης περιέχει πρότυπα για πολλά βασικά HTML στοιχεία (Typography, Code, Tables, Forms, Buttons, Images, Icons) όπως και JavaScript plugins, τα οποία προσθέτουν επιπλέον λειτουργικότητα στις διεπαφές χρήστη (Dialog Boxes, Tooltips, Image Carousels). [57]

Το Bootstrap προσφέρεται ως npm πακέτο και στην εργασία αυτή χρησιμοποιήθηκε η έκδοση 4.4.1.

3.4.5 jQuery

Η jQuery [58] είναι μία γρήγορη, συμπαγής JavaScript βιβλιοθήκη, η οποία σχεδιάστηκε για να απλοποιήσει την συγγραφή JavaScript κώδικα για τη δημιουργία ιστοσελίδων. Είναι δωρεάν open-source λογισμικό και αποτελεί την πιο ευρέως χρησιμοποιούμενη JavaScript βιβλιοθήκη στο διαδίκτυο, καθώς η χρήση της απαιτεί στοιχειώδη γνώση των HTML, CSS και JavaScript, ενώ ταυτόχρονα είναι συμβατή με τους περισσότερους φυλλομετρητές (Browsers).

Αποτελεί ιδανική επιλογή για την δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών, αφού απλοποιεί τον χειρισμό HTML/DOM, CSS, HTML events όπως και τη χρήση εφέ/animations. Επίσης διευκολύνει τη χρήση των AJAX (Asynchronous JavaScript And XML) κλήσεων, αλλά και προφέρει, μέσω των jQuery plugins, ένα πολύ μεγάλο φάσμα λειτουργιών. Η συνεισφορά της κοινότητας ελεύθερου λογισμικού είναι πολύ σημαντική ώστε να αποκτά η jQuery συνεχώς νέες δυνατότητες, αφού οποιοσδήποτε χρήστης, εκτός από τη χρήση των εργαλείων της βιβλιοθήκης, μπορεί να δημιουργεί και να μοιράζεται νέα πακέτα. [59]

3.5 MySQL

Η MySQL [60] είναι μία open-source σχεσιακή βάση δεδομένων (relational database management system ή RDBMS). Μία σχεσιακή βάση δεδομένων οργανώνει

τα δεδομένα σε πίνακες, οι οποίοι συσχετίζονται μεταξύ τους και αυτές οι σχέσεις δίνουν την απαραίτητη δομή στα δεδομένα. Η SQL είναι η γλώσσα που χρησιμοποιούν οι προγραμματιστές για να δημιουργήσουν, να επεξεργαστούν και να εξάγουν δεδομένα από τη βάση δεδομένων, αλλα και για τη διαχείρισή της. Η MySQL, που έχει αποκτηθεί από την Oracle Corporation, διαθέτει έκδοση που είναι διαθέσιμη δωρεάν, ενώ παράλληλα υπάρχει και εμπορική έκδοση, η οποία προσφέρει επιπλέον δυνατότητες. Η αποδεδειγμένη απόδοση και αξιοπιστία σε συνδυασμό με την ευκολία χρήσης, την καθιστούν ιδανική για την ανάπτυξη διαδικτυακών εφαρμογών, ενώ αποτελεί μια εξαιρετικά δημοφιλή επιλογή τόσο από μικρά projects όσο και από εταιρίες, καθώς χρησιμοποιείται από ιστοσελίδες όπως το Facebook, το Twitter, το Flickr και το YouTube. [61]

Έχει υλοποιηθεί στις γλώσσες C και C++ και υποστηρίζεται από πολλά δημοφιλή λειτουργικά συστήματα, όπως Linux, Windows, Solaris, OS X και FreeBSD. Το πρωτεύον μοντέλο της είναι το σχεσιακό DBMS, ενώ ως επιπρόσθετα μοντέλα αναφέρονται τα Document store και Spatial DBMS. Υποστηρίζει πληθώρα γλωσσών προγραμματισμού και πιο συγκεκριμένα τις Ada, C, C#, C++, D, Delphi, Eiffel, Erlang, Haskell, Java, JavaScript (Node.js), Objective-C, OCaml, Perl, PHP, Python, Ruby, Scheme, Tcl. [62] Μερικά επιπλέον χαρακτηριστικά της είναι η συμβατότητα με όλα τα σύνολα χαρακτήρων, η υποστήριξη πολυνηματικής λειτουργίας, XML, Server-side scripting, αλλά και των προγραμματιστικών διεπαφών (APIs) ADO.NET, JDBC, ODBC, Proprietary native API. [63]

3.6 Vagrant

Το Vagrant [64] είναι ένα λογισμικό ανοιχτού κώδικα που χρησιμοποιείται για τη δημιουργία και τη διαχείριση φορητών εικονικών περιβάλλοντων ανάπτυξης λογισμικού. Προσπαθεί να απλοποιήσει τη διαχείριση των παραμέτρων του εικονικού περιβάλλοντος, προκειμένου να αυξήσει την παραγωγικότητα της ανάπτυξης. Εστιάζοντας στον αυτοματισμό, το Vagrant μειώνει το απαιτούμενο χρόνο εγκατάστασης και επιτυγχάνει τη λειτουργία του συστήματος, ανεξάρτητα από τον υπολογιστή που χρησιμοποιείται σε κάθε περίπτωση. Το Vagrant είναι γραμμένο στη γλώσσα Ruby, αλλά το υποστηρίζει την ανάπτυξη και σε άλλες γλώσσες, όπως PHP, Python, Java, C# και JavaScript. Στοχεύει στην παροχή κοινών περιβάλλοντων παραγωγής, διατηρώντας το ίδιο λειτουργικό σύστημα, πακέτα και ρυθμίσεις, δίνοντας ταυτόχρονα στους χρήστες την ευελιξία να χρησιμοποιούν τον IDE και το πρόγραμμα περιήγησης της επιλογής τους. Εξασφαλίζει ένα κοινό περιβάλλον εκτέλεσης για κάθε χρήστη ανεξάρτητα απο το αν χρησιμοποιεί Linux, Mac OS X ή Windows.

Το Vagrant χρησιμοποιεί "Provisioners" και "Providers" ως δομικά στοιχεία για τη διαχείριση του περιβάλλοντος ανάπτυξης. Οι Provisioners είναι εργαλεία που επιτρέπουν στους χρήστες να προσαρμόζουν τη διαμόρφωση των εικονικών περιβαλλόντων. Το Puppet και το Chef είναι οι πιο ευρέως χρησιμοποιούμενοι Provisioners. Οι Providers είναι οι υπηρεσίες που χρησιμοποιεί η Vagrant για να δημιουργήσει και να δημιουργήσει εικονικά περιβάλλοντα. Οι σημαντικότεροι υποστηριζόμενοι Providers είναι VirtualBox, Hyper-V, Docker, VMware και AWS. Το Vagrant βρίσκεται πάνω από το λογισμικό εικονικοποίησης και βοηθά τον χρήστη να αλληλεπιδρά εύκολα με τους Providers. Αυτοματοποιεί τη διαμόρφωση εικονικών περιβάλλοντων χρησιμοποιώντας τους Provisioners και μειώνοντας τις ενέργειες που πρέπει να εκτελέσει ο χρήστης. Οι απαιτήσεις όσον αφορά τον υπολογιστή αλλά και το λογισμικο είναι καταχωρημένες στο αρχείο που ονομάζεται "Vagrantfile", ώστε να εκτελεστεί η διαδικασία που θα δημιουργήσει ένα “Box” έτοιμο να χρησιμοποιηθεί. Το Box είναι ένα format αρχείου για περιβάλλοντα Vagrant το οποίο μπορεί να αντιγραφεί σε άλλο υπολογιστή προκειμένου να αναπαραχθεί το ίδιο περιβάλλον. [65]

Χρησιμοποιείται η έκδοση 2.2.18

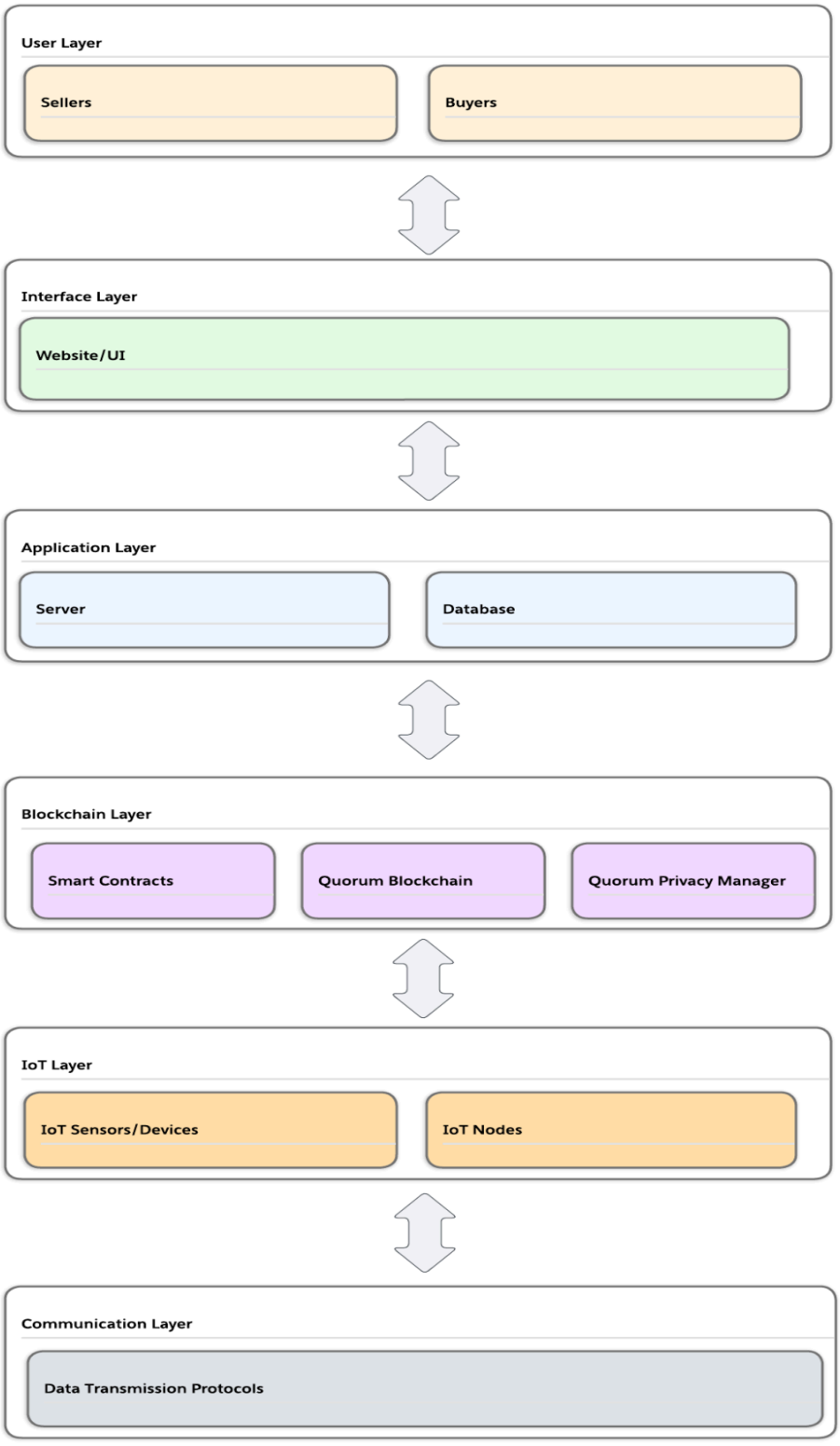
4 Ανάλυση Απαιτήσεων Συστήματος

Σε αυτό το κεφάλαιο θα αναλυθούν οι λεπτομέρειες που αφορούν στον σκοπό του συστήματος, στις κατηγορίες των χρηστών, καθώς και στις λειτουργικές και μη λειτουργικές απαιτήσεις του συστήματος. Επίσης θα παρουσιαστούν και οι παραδοχές που έγιναν κατα την ανάπτυξη της εφαρμογής. Επομένως θα διασαφηνιστούν οι λόγοι που οδήγησαν στην υλοποίηση αυτής της εφαρμογής, αλλά και οι πιθανοί πελάτες στους οποίους θα μπορούσε να απευθύνεται.

4.1 Σκοπός του συστήματος

Σκοπός του συστήματος είναι η δημιουργία μίας αποκεντρωμένης εφαρμογής (DApp), η οποία θα λειτουργεί πάνω στο Quorum Blockchain και μέσω της οποίας θα πραγματοποιούνται αγοραπωλησίες δεδομένων έξυπνων αισθητήρων (IoT), ενώ θα προσφέρει τη δυνατότητα αξιολόγησης χρηστών και αισθητήρων (Trust and Reputation). Η παρούσα εφαρμογή επιδιώκει να καλύψει την ανάγκη για την ύπαρξη ενός marketplace δεδομένων, βασισμένο στην τεχνολογία των Blockchains, το οποίο δεν θα μειονεκτεί σε σχέση με τις κλασικές εφαρμογές πελάτη-εξυπηρετητή όσον αφορά την ευχρηστία και την εμπειρία που προσφέρει στο χρήστη.

Αναλυτικότερα ο σκοπός είναι δοθεί η δυνατότητα στους χρήστες που έχουν στην κατοχή τους δεδομένα μετρήσεων αισθητήρων να τους καταχωρήσουν στο Blockchain, αλλά και στους αγοραστές να αγοράσουν τα δεδομένα αυτά μέσω ενός συστήματος που προσφέρει ασφάλεια και αξιοπιστία. Η χρησιμοποίηση του Quorum Blockchain, το οποίο αποτελώντας ένα permissioned Blockchain, προσφέρει χαρακτηριστικά ασφάλειας και ιδιωτικότητας που δεν είναι διαθέσιμα σε άλλα Blockchains. Επιπλέον η εφαρμογή επιδιώκει να δημιουργήσει ένα περιβάλλον εμπιστοσύνης μεταξύ των χρηστών, δίνοντάς τους την επιλογή να αξιολογούν τόσο άλλους χρήστες όσο και αισθητήρες, με βάση την κάθε συναλλαγή που έχει πραγματοποιηθεί.



Εικόνα 4-1 Διάγραμμα επιπέδων

4.2 Κατηγορίες χρηστών

4.2.1 Ανώνυμοι χρήστες

Ανώνυμοι χρήστες θεωρούνται όσοι δεν έχουν λογαριασμό Quorum, αλλά χρησιμοποιώντας τον φυλλομετρητή τους μπορούν να επισκεφτούν την ιστοσελίδα της εφαρμογης. Έχουν τη δυνατότητα να δουν όλους τους διαθέσιμους αισθητήρες με τα στοιχεία τους, όπως και τις συναλλαγές του Blockchain που αφορούν στην εφαρμογή.

4.2.2 Αγοραστές

Διαθέτουν λογαριασμό Quorum και αφού αποκτήσουν πρόσβαση στην ιστοσελίδα μέσω του φυλλομετρητή τους, μπορούν να εισάγουν τα απαραίτητα στοιχεία ώστε να συνδεθεί ο λογαριασμός τους με τον Quorum κόμβο της επιλογής τους και να δημιουργήσουν ένα νέο User έξυπνο συμβόλαιο, ώστε να μπορούν να χρησιμοποιήσουν την εφαρμογή.

Οι αγοραστές έχουν πρόσβαση σε όλους τους διαθέσιμους αισθητήρες και τις συναλλαγές της εφαρμογής, όπως και οι ανωνυμοι χρήστες, ενώ επιπλέον:

- Έχουν τη δυνατότητα να αγοράσουν δεδομένα αισθητήρων πληρώνοντας με Ether.
- Έχουν τη δυνατότητα να αγοράσουν δεδομένα αισθητήρων πληρώνοντας με NTUA Token.
- Έχουν πρόσβαση στα δεδομένα που έχουν αγοράσει.
- Έχουν τη δυνατότητα να αξιολογήσουν θετικά η αρνητικά κάθε κάθε αγορά που πραγματοποίησαν.

4.2.3 Πωλητές/Διαχειριστές

Οι πωλητές, όπως και οι αγοραστές, διαθέτουν λογαριασμό Quorum και αφού αποκτήσουν πρόσβαση στην ιστοσελίδα μέσω του φυλλομετρητή τους, μπορούν να εισάγουν τα απαραίτητα στοιχεία ώστε να συνδεθεί ο λογαριασμός τους με τον Quorum κόμβο της επιλογής τους και να δημιουργήσουν ένα νέο User έξυπνο συμβόλαιο, ώστε να μπορούν να χρησιμοποιήσουν την εφαρμογή.

Έχουν πρόσβαση σε όλους τους διαθέσιμους αισθητήρες και τις συναλλαγές της εφαρμογής, όπως και οι ανωνυμοι χρήστες, ενώ επιπλέον:

- Έχουν τη δυνατότητα να καταχωρούν νέους αισθητήρες στο σύστημα.
- Έχουν τη δυνατότητα να επεξεργάζονται τα στοιχεία που έχουν αφορούν κάθε καταχωρημένο αισθητήρα, όπως τιμή, δικαιώματα ιδιοκτησίας και passphrase.

- Έχουν τη δυνατότητα να ολοκληρώνουν τις εκκρεμείς πωλήσεις ή να τις απορρίπτουν, αν το επιθυμούν.
- Έχουν τη δυνατότητα να αξιολογήσουν κάθε αγοραστή μετά από κάθε ολοκληρωμένη πώληση.

4.3 Παραδοχές

Η εφαρμογή έχει υλοποιηθεί στα πλαίσια της παρούσας διπλωματικής εργασίας και όπως είναι λογικό έχει εκπαιδευτικό χαρακτήρα και δεν προορίζεται για εμπορική χρήση. Το γεγονός αυτό, σε συνδυασμό με τους περιορισμούς που προέκυψαν από τα εργαλεία και τις τεχνολογίες που χρησιμοποιήθηκαν, οδήγησαν σε κάποιες παραδοχές και απλοποιήσεις.

- Η εφαρμογή δεν ελέγχει κάθε χρήστη ώστε να εξασφαλίσει την τιμιότητά του, καθώς δεν εστιάζει εξαρχής σε αυτό το ζήτημα, δεδομένου ότι στη συνέχεια, μέσω των λειτουργιών που αφορούν την αξιολόγηση χρηστών, μπορεί να μειώσει την ύπαρξη κακόβουλων χρηστών. Ένας κακόβουλος χρήστης θα μπορούσε να προσπαθήσει να πουλήσει κενά δεδομένα ή δεδομένα που δεν του ανήκουν.
- Έχει καθοριστεί εκ των προτέρων ότι η ισοτιμία NTUA Token:Ether είναι 1:1. Φυσικά υπάρχει η δυνατότητα να οριστεί διαφορετικά, όμως δεν καθορίζεται δυναμικά από την προσφορά και την ζήτηση, καθώς το Token χρησιμοποιουνται με απλουστευμένη λογική.
- Θεωρήθηκε ότι η επικοινωνία τόσο ανάμεσα στο Blockchain και την εφαρμογή, όσο και προς τη βάση δεδομένων γίνεται με ασφάλεια, ωστόσο λόγω της πρώιμης κατάστασης των διαφόρων εργαλείων που χρησιμοποιήθηκαν, ιδανικά θα ήταν επιθυμητά περισσότερα μέτρα προστασίας, ώστε οι συναλλές του χρήστη να εξασφαλίζονται στον μέγιστο δυνατό βαθμό.
- Το Quorum Blockchain διαχειρίζεται τις ιδιωτικές συναλλαγές σε επίπεδο κόμβου και όχι σε επίπεδο λογαριασμού. Επομένως ο χρήστης πρέπει να δημιουργεί διαφορετικό λογαριασμο User για να χρησιμοποιήσει την εφαρμογή σε κάθε διαφορετικό κόμβο που συνδέεται και όπως είναι αναμενόμενο με βάση τον τρόπο λειτουργίας του Quorum, δε θα μπορεί να έχει πρόσβαση στα ίδια δεδομένα αν συνδεθεί από διαφορετικό κόμβο. Άρα είναι ευθύνη του χρήστη να συνδέεται από ένα συγκεκριμένο κόμβο, ώστε να έχει πρόσβαση σε όλη την ιδιωτική πληροφορία που είναι διαθέσιμη στον λογαριασμο του.

4.4 Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις εξασφαλίζουν ότι η εφαρμογή μπορεί να προσφέρει στους χρήστες τις απαιτούμενες λειτουργίες και ότι διαθέτει τα ζητούμενα χαρακτηριστικά.

Οι λειτουργικές απαιτήσεις της εφαρμογής είναι οι παρακάτω:

- Να επικοινωνεί με το Quorum Blockchain.
- Να γίνεται η σύνδεση/διαπίστευση του χρήστη αποκλειστικά με τις δυνατότητες που προσφέρει το Quorum Blockchain, δηλαδή να μην χρησιμοποιείται συμβατικό login.
- Να δίνει την δυνατότητα σε κάθε χρήστη με Quorum λογαριασμό να μπορεί να δημιουργήσει ένα νέο έξυπνο συμβόλαιο User, που θα χρησιμεύει σαν λογαριασμός χρήστη της αποκεντρωμένης εφαρμογής.
- Να δίνεται στους πωλητές η δυνατότητα να καταχωρήσουν ένα νέο αισθητήρα.
- Να δίνεται στους πωλητές η δυνατότητα να επεξεργαστούν τα στοιχεία ενός δικού τους καταχωρημένου αισθητήρα.
- Να έχουν οι πωλητές τη δυνατότητα διαγραφής ενός δικού τους αισθητήρα.
- Να μπορεί ο καθένας να δει όλους τους διαθέσιμους αισθητήρες, οι οποίοι θα παρουσιάζονται μαζί με τα στοιχεία τους και έναν μοναδικό αναγνωριστικό αριθμό (ID).
- Να δίνεται η δυνατότητα στον χρήστη να βρει τον αισθητήρα που επιθυμεί, με βάση τα στοιχεία ή τον αναγνωριστικό αριθμό του και να έχει την επιλογή να τον αγοράσει με Ether ή NTUA Token.
- Οι αγοραστές να μπορούν να δουν τους αισθητήρες που έχουν ήδη αγοράσει.
- Οι αγοραστές να έχουν τη δυνατότητα να παραλάβουν όποτε το επιθυμούν τα δεδομένα οποιουδήποτε από τους αγορασμένους αισθητήρες.
- Να δίνεται οι δυνατότητα στους αγοραστές να αξιολογήσουν τις ολοκληρωμένες αγορές τους.
- Να μπορούν οι πωλητές να ολοκληρώσουν ή να ακυρώσουν κάποια πιθανή πώληση.
- Να μπορεί να δει ο πωλητής όλες τις πωλήσεις που έχει ολοκληρώσει.
- Να έχουν τη δυνατότητα οι πωλητές να αξιολογήσουν τον αγοραστή για καθεμία ολοκληρωμένη μεταξύ τους συναλλαγή.
- Να είναι προσβάσιμες σε όλους οι συναλλαγες του Blockchain που αφορούν στην εφαρμογή.

Στη συνέχεια παρουσιάζονται οι λειτουργικές απαιτήσεις των έξυπνων συμβολαίων, τα οποία αποτελούν τα πιο βασικά στοιχεία της αποκεντρωμένης εφαρμογής.

Οι λειτουργικές απαιτήσεις του έξυπνου συμβολαίου Management, το οποίο αφορά στη διαχείριση και στην αγοραπωλησία των δεδομένων των έξυπνων αισθητήρων:

- Να δίνει τη δυνατότητα στους πωλητές/διαχειριστές των έξυπνων αισθητήρων να τους καταχωρήσουν, αποθηκεύοντας τα στοιχεία τους στο Blockchain.
- Να μπορεί ο κάθε πωλητής (και μόνο αυτός) να επεξεργαστεί τα στοιχεία του κάθε αισθητήρα του, σε περίπτωση που επιθυμεί να τροποποιήσει κάποιο από αυτά, όπως η τιμή ή το passphrase.
- Να υπάρχει η δυνατότητα διαγραφής ενός αισθητήρα.
- Να μπορούν οι πωλητές να τροποποιήσουν τα δικαιώματα ιδιοκτησίας κάθε αισθητήρα παραχωρώντας ένα ποσοστό αυτών σε κάποιον άλλο λογαριασμό.
- Να δίνει τη δυνατότητα στους χρήστες να αγοράσουν τα δεδομένα οποιουδήποτε αισθητήρα επιθυμούν.
- Να μπορούν οι πωλητές να ακυρώσουν μία εκκρεμή πώληση ή να την ολοκληρώσουν.
- Να φροντίζει για το σωστό διαμοιρασμό της αμοιβής που αντιστοιχεί σε κάθε αισθητήρα στους ιδιοκτήτες του, ανάλογα με το ποσοστό κυριότητας του καθενός.
- Να ελέγχει ότι μία πώληση έχει ολοκληρωθεί και να φροντίζει ότι ο αγοραστής θα έχει πλέον δικαίωμα πρόσβασης στα δεδομένα.

Οι λειτουργικές απαιτήσεις του έξυπνου συμβολαίου NTUA Token, το οποίο θα υλοποιεί τις λειτουργίες του αφορούν το κρυπτονόμισμα NTUA Token:

- Να δίνει τη δυνατότητα μεταφοράς NTUA Tokens από μία διεύθυνση σε μία άλλη.
- Να μπορεί να ενημερώσει για το διαθέσιμο υπόλοιπο κάθε διεύθυνσης σε NTUA Tokens.
- Να δίνει την δυνατότητα στους χρήστες να προμηθευτούν NTUA Tokens, πληρώνοντας με Ether.
- Να δίνει τη δυνατότητα στους κατόχους NTUA Token, να το πουλήσουν και να λάβουν Ether.

Οι λειτουργικές απαιτήσεις του έξυπνου συμβολαίου User, το οποίο θα υλοποιεί τις λειτουργίες που αφορούν τον λογαριασμό που διατηρεί ο χρήστης για τις ανάγκες της εφαρμογής:

- Να δίνει την δυνατότητα σε κάθε χρήστη που διαθέτει διεύθυνση Quorum να μπορεί να δημιουργήσει ένα νέο User έξυπνο συμβόλαιο που χρησιμεύει ως λογαριασμός χρήστη της εφαρμογής.
- Να μπορεί να αποθηκεύσει το κρυφό passphrase που δίνει πρόσβαση στα δεδομένα, τόσο για τους πωλητές, όσο και για τους αγοραστές.
- Να επιστρέφει την αποθηκευμένη πληροφορία μόνο στο χρήστη που είναι ο ιδιοκτήτης του έξυπνου συμβολαίου/λογαριασμού, κρατώντας την κρυφή από οποιοδήποτε άλλον.

Οι λειτουργικές απαιτήσεις του έξυπνου συμβολαίου Reputation, το οποίο θα υλοποιεί τις λειτουργίες που αφορούν την αξιολόγηση πωλητών, αγοραστών και αισθητήρων.

- Να δίνει τη δυνατότητα στους αγοραστές μετα την ολοκλήρωση μίας αγοραπωλησίας να αξιολογήσουν την συναλλαγή, συνυπολογίζοντας τη θετική ή αρνητική βαθμολογία που έδωσαν στη συνολική βαθμολογία του χρήστη/πωλητή και του αισθητήρα αντίστοιχα.
- Να δίνει τη δυνατότητα στον πωλητή, μετα την ολοκλήρωση μίας αγοραπωλησίας, να δίνει θετική ή αρνητική αξιολόγηση στον αγοραστή, η οποία θα μεταβάλει τη συνολική βαθμολογία του.
- Να ελέγχει αν η συναλλαγή που αφορά η κάθε αξιολόγηση είναι υπαρκτή, δηλαδή αν η αξιολόγηση είναι έγκυρη.
- Να μην επιτρέπει την πολλαπλή αξιολόγηση.
- Να επιστρέφει τη βαθμολογία για οποιοδήποτε χρήστη ή αισθητήρα αν ζητηθεί.

4.5 Μη λειτουργικές απαιτήσεις

Στη συνέχεια παρουσιάζονται οι μη λειτουργικές απαιτήσεις, οι οποίες δεν αφορούν στα ποιοτικά χαρακτηριστικά της εφαρμογής.

Ασφάλεια

- Η πραγματοποίηση οικονομικών συναλλαγών επιβάλλει την ύπαρξη υψηλού επιπέδου ασφαλείας.
- Καθώς γίνονται μεταφορές κρυπτονομισμάτων, πρέπει να προστατεύονται οι χρήστες από κακόβουλες επιθέσεις.
- Οι συναλλαγές που περιέχουν κρίσιμη πληροφορία για τη λειτουργία της εφαρμογής (passphrases) πρέπει να διασφαλίζεται ότι παραμείνουν ιδιωτικές, κάτι που καθίσταται δυνατό από τα χαρακτηριστικά του Quorum.

- Τα έξυπνα συμβόλαια δεν μπορούν να τροποποιηθούν, οπότε θα πρέπει να έχει εξασφαλιστεί εκ των προτέρων ότι δεν υπάρχει κάποιο κενό ασφαλείας ή γενικότερα κάποιο σφάλμα που θα μπορούσε να έχει καταστροφικές για το σύστημα συνέπειες.
- Το σύστημα χρησιμοποιεί βάση δεδομένων, επομένως πρέπει να υπάρχουν μέτρα ασφαλείας ώστε να είναι προστατευμένο απεναντι σε μια σχετική επίθεση (π.χ. SQL injection). [66]

Επεκτασιμότητα

- Η εφαρμογή είναι απαραίτητο να μπορεί να επεκταθεί εύκολα, ειδικότερα αφού η παρούσα υλοποίηση έχει εκπαιδευτικό χαρακτήρα και όχι εμπορικό.
- Μια πιθανή επέκταση θα πρέπει να είναι εφικτή χωρίς να απαιτούνται αδιαιολόγητα πολλοί επιπλέον υπολογιστικοί πόροι, δηλαδη η εφαρμογή να είναι κλιμακώσιμη (scalability).
- Η εφαρμογή θα πρέπει να είναι ευέλικτη, δηλαδή χωρίς να απαιτείται εκτενής αναδιαμόρφωση, να είναι εφικτή η μετατροπή της από marketplace δεδομένων έξυπνων αισθητήρων σε κάποιο άλλο παρεμφερές “προϊόν”.

Απόδοση

- Τα έξυπνα συμβόλαια θα πρέπει να έχουν βελτιστοποιημένο κώδικα, ώστε να ελαχιστοποιηθεί το υπολογιστικός όγκος που επιβαρύνει το Quorum Blockchain, παρά το γεγονός ότι δεν τίθεται θέμα κόστους gas, καθώς είναι μηδενικό, εν αντιθέσει, για παράδειγμα με το Ethereum Blockchain, στην περίπτωση του οποίου το gas price αποτελεί ένα σημαντικό παράγοντα απόδοσης.
- Η μεταφορά δεδομένων πρέπει να διατηρείται στο ελάχιστο δυνατό, δηλαδή να ελέγχεται αν έχει γίνει κάποια αλλαγή που αφορά το τρέχον περιεχόμενο της ιστοσελίδας και στη συνέχεια να μεταφέρονται από το Blockchain μόνο τα δεδομένα που απαιτούνται ώστε να επικαιροποιηθεί το περιεχόμενο.

5 Σχεδιασμός και υλοποίηση συστήματος

Στο κεφάλαιο αυτό θα παρουσιαστούν λεπτομέρειες που αφορούν τον σχεδιασμό και την υλοποίηση της αποκεντρωμένης εφαρμογής που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας.

Η αποκεντρωμένη εφαρμογή που δημιουργήθηκε είναι στην ουσία ένα σύστημα που χρησιμοποιεί το Quorum Blockchain για τα έξυπνα συμβόλαια της, μια βάση δεδομένων (MySQL) για την αποθήκευση των δεδομένων, και ένα Server (Node.js), ο οποίος κάνει τα αρχεία της ιστοσελίδας διαθέσιμα όταν ζητηθούν και φροντίζει για την ορθή επικοινωνία της ιστοσελίδας και της βάσης δεδομένων.

Τη δεδομένη χρονική στιγμή η ανάπτυξη αποκεντρωμένων εφαρμογών βρίσκεται σε ένα σχετικά πρώιμο στάδιο. Η προγραμματιστική κοινότητα γύρω από τα Blockchains, αν και μεγαλώνει με γρήγορο ρυθμό, απέχει αρκετά από τις αντίστοιχες κοινότητες του “παραδοσιακού” προγραμματισμού, ενώ δεν υπάρχουν καθιερωμένα πρότυπα ανάπτυξης. Το Quorum, παρόλα τα σημαντικά πλεονεκτήματα του, χρειάζεται αρκετό καιρό ακόμα ώστε γίνει ευρέως διαδεδομένο, ειδικότερα αν σκεφτεί κανείς, ότι το Ethereum, που αποτελεί ήδη μία από τις σημαντικότερες τεχνολογίες και χρησιμοποιείται σε πολύ μεγαλύτερο βαθμό, συνεχώς βελτιώνεται και έχει αρκετό “δρόμο” να διανύσει ακόμα.

5.1 Βασικές Οντότητες Συστήματος

Στο αποκεντρωμένο σύστημα αγοραπωλησίας δεδομένων αισθητήρων που υλοποιήθηκε, αναγνωρίζονται οι εξής βασικές οντότητες:

- Χρήστες:

  Διακρίνονται σε πωλητές/διαχειριστές, αγοραστές και ανώνυμους. Οι πωλητές εισάγουν αισθητήρες, ανεβάζουν δεδομένα και διαχειρίζονται τις πωλήσεις. Οι αγοραστές αναζητούν, αγοράζουν και αξιολογούν δεδομένα. Οι ανώνυμοι χρήστες έχουν μόνο δυνατότητα προβολής διαθέσιμων αισθητήρων και συναλλαγών, χωρίς να συμμετέχουν σε αγορές ή αξιολογήσεις.
- Αισθητήρες (IoT Sensors):

  Παράγουν δεδομένα που καταχωρούνται και διαχειρίζονται μέσω της πλατφόρμας.
- Έξυπνα Συμβόλαια (Smart Contracts):

  Διακρίνονται σε User, Management (αγοραπωλησία/διαχείριση αισθητήρων), Reputation (αξιολόγηση), και NTUA Token (διαχείριση συναλλαγών).

- Βάση Δεδομένων:

  Αποθηκεύει τα δεδομένα των αισθητήρων και σχετικές πληροφορίες για αποδοτική αναζήτηση και διαχείριση μεγάλης κλίμακας.
- Server (Node.js/Web Server):

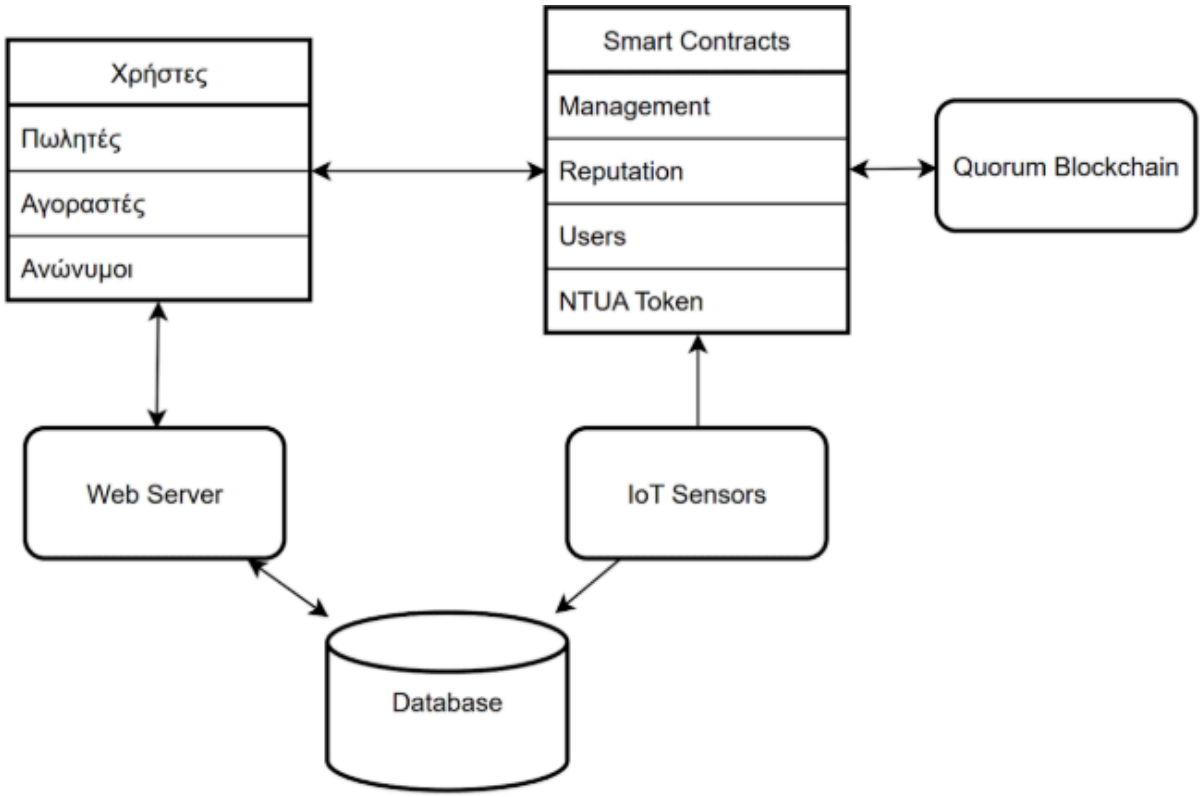
  Υποστηρίζει τις λειτουργίες της ιστοσελίδας και τη διασύνδεση blockchain, smart contracts, βάσης δεδομένων και χρηστών.

Αυτές οι οντότητες συνεργάζονται για την υλοποίηση όλης της ροής μίας ηλεκτρονικής, ασφαλούς και αποκεντρωμένης αγοράς/αξιολόγησης δεδομένων.

5.1.1 Διάγραμμα βασικών οντοτήτων

Παρακάτω παρουσιάζεται ένα διάγραμμα που συνοψίζει τις βασικές οντότητες του συστήματος και τις βασικές αλληλεπιδράσεις τους (εγγραφή, καταχώρηση, αγορά, αξιολόγηση, αποθήκευση δεδομένων):

Διάγραμμα βασικών οντοτήτων και αλληλεπιδράσεων:



Εικόνα 5-1 Διάγραμμα βασικών οντοτήτων και αλληλεπιδράσεων

5.2 Ακολουθία / Sequence Ροής Λειτουργίας

Η λειτουργική ροή του συστήματος αγοραπωλησίας δεδομένων IoT μέσω blockchain περιγράφεται στα παρακάτω βήματα. Αυτή η ακολουθία σχηματίζει το βασικό "σενάριο χρήστη" για το σύστημα:

1. Εγγραφή και Σύνδεση Χρήστη:

   Ο πωλητής ή αγοραστής δημιουργεί λογαριασμό μέσω του έξυπνου συμβολαίου User, με τη διαμεσολάβηση του Web Server. Η αυθεντικοποίηση γίνεται απευθείας στο blockchain και όχι μέσω συμβατικού login.

2. Καταχώρηση Αισθητήρα (Πωλητής):

   Ο πωλητής μέσω της πλατφόρμας εισάγει ένα νέο αισθητήρα, συμπληρώνοντας τα απαραίτητα στοιχεία και ανεβάζοντας αρχικά δεδομένα. Τα στοιχεία και τα δεδομένα διαβιβάζονται στη Βάση Δεδομένων για αποθήκευση, ενώ η ύπαρξη και τα μεταδεδομένα του αισθητήρα καταγράφονται στο blockchain μέσω του Management Smart Contract.

3. Προβολή Διαθέσιμων Αισθητήρων (Όλοι οι χρήστες):

   Κάθε χρήστης (ακόμη και ανώνυμος) έχει πρόσβαση σε λίστα διαθέσιμων αισθητήρων, τιμές, αναγνωριστικά, ιδιοκτήτες κτλ, μέσω web interface. Τα δεδομένα προέρχονται από την Database και τα Smart Contracts για επίσημες πληροφορίες/συναλλαγές.

4. Αγορά Δεδομένων (Αγοραστής):

   Ο αγοραστής επιλέγει αισθητήρα και προχωράει σε αγορά, πληρώνοντας με Ether ή NTUA Token μέσω των αντίστοιχων Smart Contracts (Management & Token). Η αγορά καταγράφεται στο blockchain και ενεργοποιείται διαδικασία επιβεβαίωσης από τον πωλητή.

5. Επιβεβαίωση Συναλλαγής (Πωλητής):

   Ο πωλητής εγκρίνει (ή απορρίπτει) την πώληση, μέσω του Management Smart Contract. Αν εγκριθεί, ο αγοραστής αποκτά δικαίωμα λήψης των αντίστοιχων δεδομένων.

6. Παράδοση Δεδομένων:

   Ο αγοραστής, έχοντας πλέον το δικαίωμα πρόσβασης, λαμβάνει τα δεδομένα και το passphrase μέσω της βάσης και του User Smart Contract.

7. Αξιολόγηση Συναλλαγής:

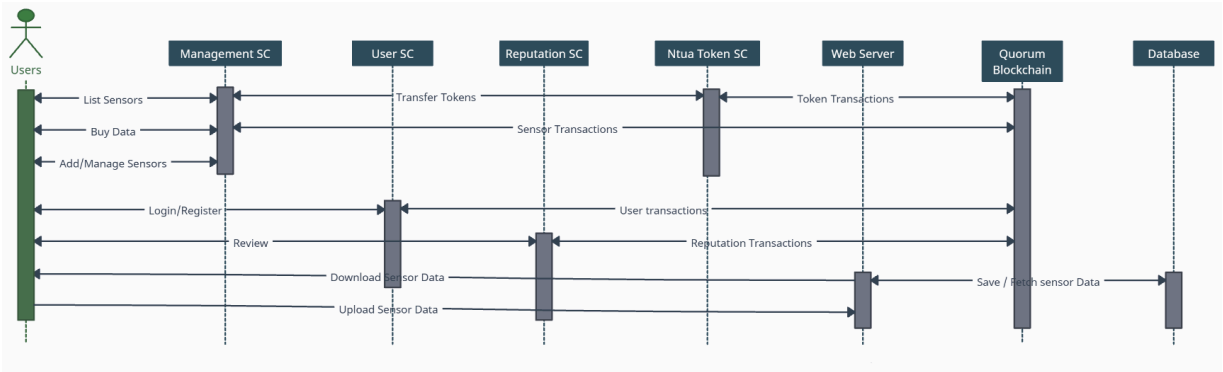
   Και οι δύο πλευρές (αγοραστής, πωλητής) αξιολογούν ο ένας τον άλλον και το ίδιο το dataset. Η αξιολόγηση επιστρέφει στο Reputation Smart Contract και επηρεάζει την συνολική φήμη και των δύο.

8. Ενημέρωση Συστήματος/Blockchain:

   Όλες οι παραπάνω ενέργειες (upload, αγορά, αξιολόγηση) αποτυπώνονται διαρκώς στο Quorum blockchain, διασφαλίζοντας αμεταβλητότητα και διαφάνεια συναλλαγών, ενώ τα δεδομένα ταυτοποιούνται και διαχειρίζονται επαρκώς από τη βάση και το web server.

5.2.1 Διάγραμμα ροής λειτουργίας

Στην συνέχεια παρουσιάζεται το διάγραμμα ροής λειτουργίας:



Εικόνα 5-2 διάγραμμα ροής λειτουργίας

5.3 Smart Contracts

Το σημαντικότερο στοιχείο κάθε αποκεντρωμένης εφαρμογής είναι τα έξυπνα συμβόλαια (Smart Contracts), τα οποία περιέχουν τον κώδικα που εκτελείται στο Ethereum VM και αποτελεί τον βασικό κορμό της.

Η ανάπτυξή των έξυπνων συμβολαίων, αποτελώντας το κρισιμότερο στάδιο στην δημιουργία της εφαρμογής, παρουσιάζει κάποιες δυσκολίες λόγω των χαρακτηριστικών του Blockchain. Καθώς τα περιεχόμενα του Blockchain είναι αμετάβλητα, απο τη στιγμή

που θα υποβληθεί σε αυτό κάποιο έξυπνο συμβόλαιο, δεν είναι δυνατό να γίνουν αλλαγές ή να διορθωθούν σφάλματα που προέκυψαν στην πορεία, παρά μόνο αν δημιουργηθεί ένα νέο έξυπνο συμβόλαιο.

Η λειτουργία της εφαρμογής είναι βασισμένη στα τέσσερα έξυπνα συμβόλαια Management, User, Reputation και NtuaToken.

Το έξυπνο συμβόλαιο Management υλοποιεί τις λειτουργίες που αφορούν την διαχείριση, την πώληση και την αγορά των αισθητήρων.

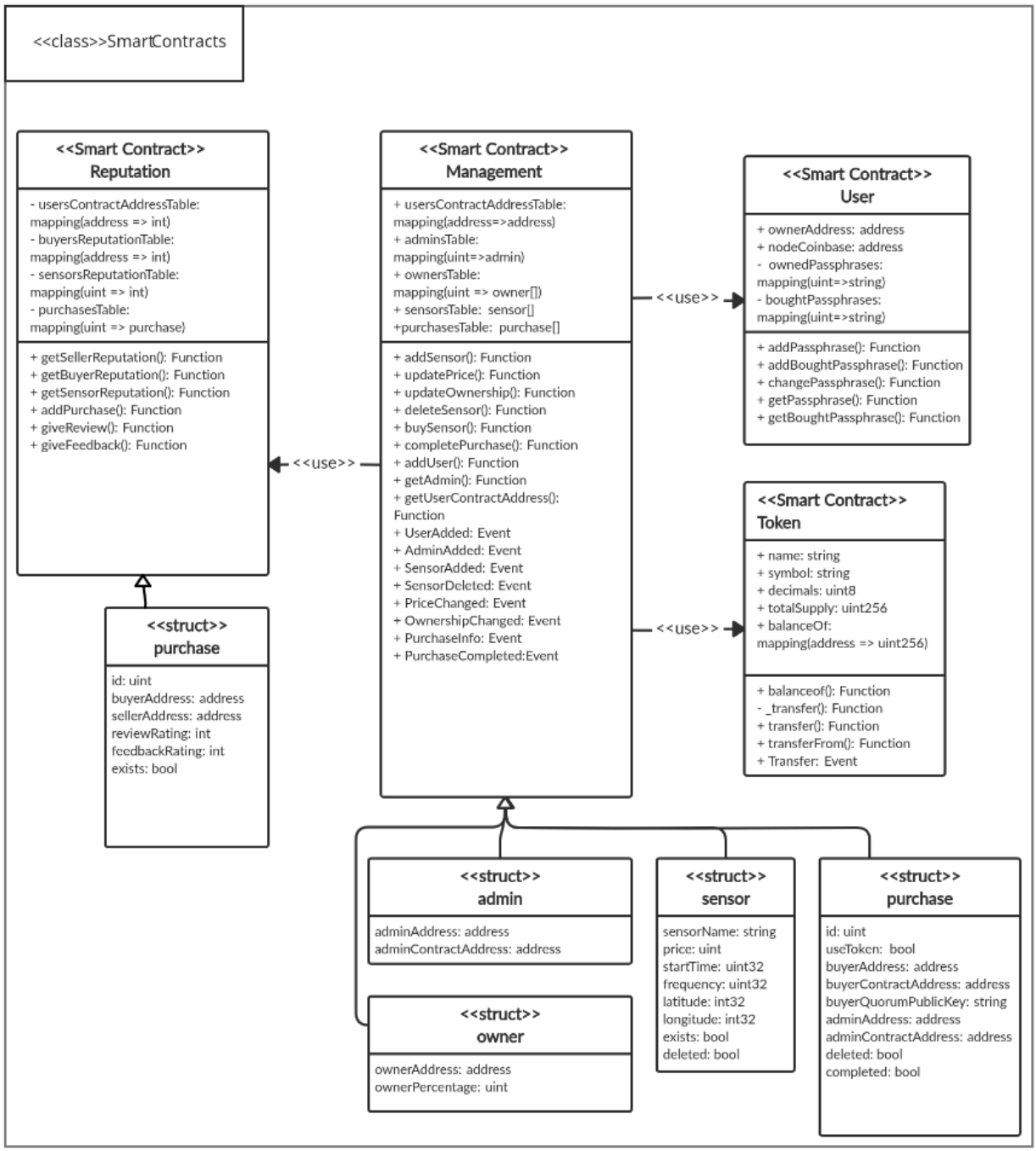
Το έξυπνο συμβόλαιο User δημιουργεί ένα λογαριασμό στον οποίο κάθε χρήστης μπορεί να αποθηκεύσει με ασφάλεια τα passphrases που αντιστοιχούν στους αισθητήρες που τον αφορούν, χωρίς να υπάρχει κίνδυνος υποκλοπής.

Το έξυπνο συμβόλαιο Reputation υλοποιεί τις λειτουργίες που αφορούν στις αξιολογήσεις των χρηστών τόσο για άλλους χρήστες, όσο και για αισθητήρες, δημιουργώντας ένα σύστημα εμπιστοσύνης μεταξύ τους.

Το έξυπνο συμβόλαιο NtuaToken δημιουργεί το κρυπτονόμισμα NTUA token (National Technical University of Athens Token) και είναι είναι υπεύθυνο για την πραγματοποίηση των μεταφορών του συγκεκριμένου κρυπτονομίσματος όταν αυτό χρησιμοποιείται από τους χρήστες της εφαρμογής για την ολοκλήρωση κάποιας οικονομικής συναλλαγής.

5.3.1 Διαγράμματα UML

Στην συνέχεια παρουσιάζεται το UML διάγραμμα κλάσεων για τα έξυπνα συμβόλαια της εφαρμογής:



Εικόνα 5-3 Διάγραμμα κλάσεων

Τα έξυπνα συμβόλαια που ακολουθούν στη συνέχεια έχουν δημιουργηθεί στη γλώσσα Solidity. Ο πηγαίος κώδικας των συμβολαίων βρίσκεται στο Παράρτημα.

5.3.2 Management Smart Contract

Στην συνέχεια γίνεται αναλυτική παρουσίαση του κώδικα του έξυπνου συμβολαίου Management.

```
pragma solidity ^0.5.7;
contract Management {
uint public oneEther = 1 ether;
```

Η έκδοση της Solidity που χρησιμοποιείται.

```
struct sensor {
    string sensorName;
    uint price;
    uint32 startTime;
    uint32 frequency;
    int32 latitude;
    int32 longitude;
    bool exists;
    bool deleted;
  }
```

Η δομή δεδομένων **sensor** χρησιμοποιείται για την αποθήκευση των στοιχείων κάθε αισθητήρα.

sensorName: το αναγνωριστικό του αισθητήρα.

price: η τιμή των δεδομένων των μετρήσεων.

startTime: η χρονική στιγμή (32 bit timestamp) της πρώτης μέτρησης του αισθητήρα.

frequency: το πλήθος μετρήσεων που πραγματοποιούνται ανά ώρα.

latitude: το γεωγραφικό πλάτος της θέσης του.

longitude: το γεωγραφικό μήκος της θέσης του.

exists: χρησιμοποιείται για να ελέγχεται εύκολα η ύπαρξη του αισθητήρα.

deleted: δηλώνει ότι ο αισθητήρας έχει πλέον διαγραφεί και δεν είναι διαθέσιμος προς πώληση.

```
struct purchase{
    uint id;
    bool useToken;
    address payable buyerAddress;
    address buyerContractAddress;
    string buyerQuorumPublicKey;
    address adminAddress;
    address adminContractAddress;
    bool exists;
    bool completed;
  }
```

Η δομή δεδομένων **purchase** χρησιμοποιείται για την αποθήκευση των στοιχείων κάθε αγοράς. Με την προσθήκη στον αντίστοιχο πίνακα γίνεται πιο απλή η δημιουργία ενος “ιστορικού” που περιέχει όλα τα απαραίτητα στοιχεία για τις μέχρι τώρα συναλλαγές.

id: το μοναδικό αναγνωριστικό του αισθητήρα που αγοράζεται, το οποίο προκύπτει από την θέση του στον πίνακα που αποθηκεύονται οι αισθητήρες με τη σειρά που προστίθενται.

useToken: δηλώνει αν η αγορά πραγματοποιήθηκε με Ntua Token, αντί για Ether.

buyerAddress: η διεύθυνση του αγοραστή του συμβολαίου.

buyerContractAddress: η διεύθυνση του User smart contract του αγοραστή.

buyerQuorumPublicKey: το κλειδί που αντιστοιχεί στον κόμβο του Quorum από τον οποίο ο αγοραστής έχει πρόσβαση στο δίκτυο, το οποίο μαζί με τις δύο προηγούμενες διευθύνσεις θα εξασφαλίσουν την αποκλειστική πρόσβαση του αγοραστή στα δεδομένα που θα του γίνουν διαθέσιμα μετα την ολοκλήρωση της διαδικασίας της αγοραπωλησίας.

adminAddress: η διεύθυνση του διαχειριστή του αισθητήρα.

adminContractAddress: η διεύθυνση του User smart contract του διαχειριστή.

exists: χρησιμοποιείται για να ελέγχεται εύκολα ότι τα δεδομένα της πώλησης έχουν καταχωρηθεί.

completed: δηλώνει ότι η πώληση έχει πλέον ολοκληρωθεί, είτε θετικά, επομένως τα δεδομένα έχουν γίνει διαθέσιμα στον αγοραστή, είτε αρνητικά, δηλαδή ο πωλητής/διαχειριστής επέλεξε να την ακυρώσει.

```
struct admin{
    address payable adminAddress;
    address adminContractAddress;
  }
```

Η δομή δεδομένων **admin** χρησιμοποιείται για την αποθήκευση των στοιχείων κάθε διαχειριστή, καταχωρώντας την διεύθυνσή του αλλα και τη διεύθυνση του User Smart Contract που του αντιστοιχεί .

```
struct owner{
    address payable ownerAddress;
    uint ownerPercentage;
  }
```

Τα στοιχεία για κάθε ιδιοκτήτη αισθητήρα αποθηκεύονται στη δομή δεδομένων **owner**, στην οποία καταχωρείται η διεύθυνσή του αλλά και το ποσοστό ιδιοκτησίας του για τον συγκεκριμένο αισθητήρα.

```
mapping(address=>address) usersContractAddressTable;
mapping(uint=>admin) adminsTable;
mapping(uint => owner[]) ownersTable;
sensor[] sensorsTable;
purchase[] sensorsTable;
```

Μεταξύ των μεταβλητών του συμβολαίου που χρησιμοποιούνται για την αποθήκευση των πληροφοριών του, ενδιαφέρον παρουσιάζουν τα τρια mappings. Το mapping αποτελεί μια δομή δεδομένων στη Solidity που έχει αντίστοιχη λειτουργία με το HashMap (JAVA).

Το mapping **usersContractAddressTable** για την δοθείσα διεύθυνση επιστρέφει την αντίστοιχη διεύθυνση του User Smart Contract. Τα mappings **adminsTable** και **ownersTable**, για τον δοθέντα ακέραιο αριθμό που είναι στην ουσία το μοναδικό αναγνωριστικό κάποιου αισθητηρα, επιστρέφουν αντίστοιχα τον διαχειριστή (Admin) και έναν πίνακα με τους ιδιοκτήτες (Owners).

Υπάρχουν επίσης οι δυο πίνακες **sensorsTable** και **purchasesTable**, στους οποίους αποθηκεύονται όλοι οι αισθητήρες και οι συναλλαγες, αποτελώντας πρακτικά το αρχείο του Smart Contract.

```
event UserAdded(address userAddress, address userNodeCoinbase,address userContractAddress, string userQuorumPublicKey);
event AdminAdded(uint id, address adminAddress,address adminContractAddress);

event SensorAdded(uint id,string sensorName, uint price, uint32 startTime, uint32 frequency, int32 latitude, int32 longitude);
event SensorDeleted(uint id);
event PriceChanged(uint id, uint price);
event OwnershipChanged(uint id, address ownerAddress, uint ownerPercentage);

event PurchaseInfo(uint purchaseId,uint id, bool useToken, address buyerAddress,
                address buyerContractAddress, string buyerQuorumPublicKey,address adminAddress, address adminContractAddress);
event PurchaseCompleted(uint purchaseId,uint id, bool purchaseAccepted);
```

Τα events του συμβολαίου στέλνονται από τις διάφορες μεθόδους του και ο ρόλος τους είναι καταστήσουν δυνατή την παρακολούθηση της εφαρμογής μέσω της JavaScript και με τη βοήθεια του web3, αλλά και να διευκολύνουν το debugging του κωδικα του Smart Contract. Καθώς η αποστολή των events απαιτεί gas, δεν μπορούν να σταλούν απο μεθόδους που είναι ορισμένες ως view, δηλαδή δεν προκαλούν αλλαγές στο Smart Contract.

UserAdded

Το event **UserAdded** ενημερώνει το δίκτυο για τη δημιουργία ενος νέου User, αναφέροντας τις απαραίτητες διευθύνσεις ώστε να είναι δυνατή η πραγματοποίηση συναλλαγής με αυτόν.

AdminAdded

Το event **AdminAdded** ενημερώνει οτι ο λογαριασμος που αναφέρεται είναι ο διαχειριστής του αισθητήρα που έχει το συγκεκριμενο id.

SensorAdded

Το event **SensorAdded** χρησιμοποιείται για να ενημερώνονται οι κόμβοι οτι προστέθηκε ένας νέος αισθητήρας, αναφέροντας τον αύξοντα αναγνωριστικό αριθμό, το όνομα, τη τιμή, το χρόνο που ξεκίνησε η καταγραφή των δεδομένων, τη συχνότητα με την οποία γίνονται και τέλος, τις συντεταγμένες του.

SensorDeleted

Το event περιέχει τον αριθμό του αισθητήρα που δεν είναι πλέον διαθέσιμος.

PriceChanged

Ενημερώνει για την αλλαγή της τιμης του αισθητήρα.

OwnershipChanged

Χρησιμοποιείται για να κάνει γνωστό ότι έχει γίνει αλλαγή στην ιδιοκτησία του αισθητήρα.

PurchaseInfo

Όταν κάποιος χρήστης επιχειρήσει να αγοράσει τα δεδομένα ενός αισθητήρα, μέσα από το event **PurchaseInfo**, γνωστοποιείται στο δίκτυο (επομένως και στον διαχειριστή/πωλητή) ώστε να ολοκληρώσει την συναλλαγή. Το event αναφέρει το αναγνωριστικό id του αισθητήρα, το αναγνωριστικό purchaseId της πώλησης, αλλά και τα απαραίτητα στοιχεία για τον αγοραστή (buyerAddress, buyerContractAddress, buyerQuorumPublicKey) και τον πωλητή (adminAddress, adminContractAddress). Επίσης αν η αγορά πραγματοποιήθηκε με χρήση Ntua Token, αυτό δηλώνεται απο τη μεταβλητή usedToken.

PurchaseCompleted

Μετά την ολοκληρωση της συναλλαγης, το event **PurchaseCompleted** ενημερώνει το δίκτυο και κατ' επέκταση τον αγοραστή, αν έγινε αποδεκτή η αγορά (δηλώνεται απο το purchaseAccepted) ή αν ο πωλητής την ακύρωσε. Στην περίπτωση που έχει γίνει αποδεκτή το συνθηματικό και τα δεδομένα είναι πλέον στη διάθεσή του αγοραστή.

```
    modifier hasAccess(uint _id) {

require(msg.sender==adminsTable[_id].adminAddress);
        _;
      }
```

Το modifier **hasAccess** ελέγχει αν ο χρήστης που καλεί μια συνάρτηση που τροποποιεί κάποιον αισθητήρα έχει το δικαίωμα να το κάνει, δηλαδή αν η διεύθυνσή του είναι καταχωρημένη σαν διαχειριστής του αισθητήρα αυτού, ώστε να μην μπορεί οποιοσδήποτε να κάνει αθέμιτες αλλαγές. Το παρόν modifier μπορεί να χρησιμοποιηθεί από όλες τις συναρτήσεις που αφορούν τους αισθητήρες, αποφεύγοντας έτσι την άσκοπη επανάληψη κώδικα. Χρησιμοποιείται το require για να ελέγξουμε αν η

διεύθυνση του αποστολέα (msg.sender) είναι δηλωμένη ως διαχειριστής. Το require είναι ένας τρόπος που προσφέρει η Solidity να ελέγξουμε την ροή εκτέλεσης των εντολών. Εάν η συνθήκη μέσα στο require ισχύει, η εκτέλεση συνεχίζεται, ενώ εάν αυτή δεν ισχύει η εκτέλεση της συνάρτησης σταματάει σε εκείνο το σημείο και η συναλλαγή αποτυγχάνει, επιστρέφοντας παράλληλα το κατάλληλο μήνυμα σφάλματος. Στην προκειμένη περίπτωση, καθώς πρόκειται για modifier, η εκτελεση συνεχίζεται με τον κώδικα της συνάρτησης που το χρησιμοποιεί.

```
modifier exists(uint _id) {
    require(sensorsTable[_id].exists);
    require(!sensorsTable[_id].deleted);
    _;
  }
```

Το modifier **exists** ελέγχει αν υπάρχει ο αισθητήρας με το id που δηλώνεται, ώστε να είναι δυνατή στη συνέχεια η τροποποίηση ή η πώλησή του. Η λειτουργία είναι αντίστοιχη με το προηγούμενο modifier, με τη διαφορά ότι εδώ γίνονται δύο έλεγχοι. Ο πρώτος ελεγχει αν ο αισθητήρας με το συγκεκριμένο id έχει καταχωρηθεί και ο δεύτερος ελέγχει ότι δεν έχει διαγραφεί.

```
function addSensor(string calldata _sensorName, uint _price, uint32 _startTime, uint32 _frequency, int32 _latitude, int32 _longitude, address _adminContractAddress) external returns (uint) {
        address payable _adminAddress=msg.sender;
        uint _id=sensorsTable.length;

         sensorsTable.push(sensor(_sensorName, _price, _startTime, _frequency, _latitude, _longitude,true, false));

          emit SensorAdded(_id, _sensorName, _price, _startTime, _frequency, _latitude, _longitude);

        adminsTable[_id] = admin(_adminAddress, _adminContractAddress);
        emit AdminAdded(_id, _adminAddress, _adminContractAddress);
        return _id;
      }
```

Η **addSensor** είναι η εξωτερική συνάρτηση που δημιουργεί μια καταχώρηση για κάθε νέο αισθητήρα.

Λαμβάνει ως παραμέτρους το όνομα, το δημιουργό, την τιμή, τη χρονική στιγμή έναρξης των μετρήσεων, τη συχνότητά τους, τις συντεταγμένες του αισθητήρα και τη διεύθυνση του διαχειριστή. Αρχικά δίνει και ένα αύξοντα αριθμό (id) ως μοναδικό αναγνωριστικό του νέου αισθητήρα και τον προσθέτει στον πίνακα sensorsTable που κρατά όλους τους διαθέσιμους αισθητήρες και αντίστοιχα τα στοιχεία του διαχειριστή στον πίνακα adminsTable, ώστε να είναι ο μοναδικός που μπορεί να κάνει τροποποιήσεις. Στη συνέχεια κάνει emit 2 events τα οποία γνωστοποιούν στο δίκτυο ότι προστέθηκε ένας νέος αισθητήρας, αλλα και τα στοιχεία του διαχειριστή που θα είναι απαραίτητα στους μελλοντικούς αγοραστές.

```
    function updatePrice(uint _id, uint _price) external hasAccess(_id) exists(_id){
        sensorsTable[_id].price = _price;
        emit PriceChanged(_id, _price);
    }
```

Η εξωτερική συνάρτηση **updatePrice** χρησιμοποιείται για να αλλάξει η τιμή ενός αισθητήρα, ενημερόνοντας στη συνέχεια το δικτυο με το αντίστοιχο event. Όπως και οι επόμενες συναρτήσεις που τροποποιούν τα δεδομένα, χρησιμοποιεί τα δύο modifier που αναφέρθηκαν προηγουμένως ώστε να εξασφαλιστεί ότι ο χρήστης έχει δικαίωμα πρόσβασης (hasAccess) αλλα και ότι το αρχείο υπάρχει και είναι διαθέσιμο (exists).

```
    function updateOwnership(uint _id, address payable _ownerAddress, uint
_ownerPercentage) external hasAccess(_id) exists(_id){
            bool foundAddress=false;
            uint sum=0;
            for(uint i=0;i<ownersTable[_id].length;i++){
              sum+=ownersTable[_id][i].ownerPercentage;
                if((sum-ownersTable[_id][i].ownerPercentage+_ownerPercentage)<=100
&& ownersTable[_id][i].ownerAddress==_ownerAddress){
                ownersTable[_id][i].ownerPercentage=_ownerPercentage;
                foundAddress=true;
                emit OwnershipChanged(_id, _ownerAddress, _ownerPercentage);
              }
            }
            if(sum+_ownerPercentage<=100 && !foundAddress){
              ownersTable[_id].push(owner(_ownerAddress, _ownerPercentage));
              emit OwnershipChanged(_id, _ownerAddress, _ownerPercentage);
            }
          }
```

Η εξωτερική συνάρτηση **updateOwnership** δίνει την δυνατότητα τροποποίησης του ποσοστού ιδιοκτησίας, αλλά και την προσθήκη ενός νέου ιδιοκτήτη. Αν η διευθυνση ήδη υπάρχει, αφού εντοπίσει την αντίστοιχη καταχώρηση αλλάζει μόνο το ποσοστό ιδιοκτησιας, ενώ αν ειναι περίπτωση προσθήκης, προστίθεται η διευθυνση του νέου ιδιοκτητη μαζι με το ποσοστό του. Πριν κάνει οποιαδήποτε αλλαγή, ελέγχει αν η αλλαγή είναι εντός ορίων, δηλαδή η συνολική ιδιοκτησία να μην ξεπερνά το 100%. Να σημειωθεί ότι για κάθε νέο αισθητήρα ο λογαριασμός που τον προσθέτει και δηλώνεται ως διαχειριστής του είναι και ο αρχικός ιδιοκτήτης του, έχοντας το 100% τον δικαιωμάτων, εκτός αν μεταφέρει την ολική ή μερική ιδιοκτησια του σε άλλους. Για να διευκολυνθεί η διαδικασία της πώλησης, ο διαχειριστής διαθέτει το ποσοστό που προκύπτει απο το υπολοιπο που απομένει αφαιρώντας το άθροισμα των ποσοστών των ιδιοκτητών που έχει δηλωθεί στην καταχώρηση του πίνακα ownersTable. Τέλος κάνει την αλλαγή γνωστή μέσω του event OwnershipChanged που περιέχει το αναγνωριστικό αριθμό του αισθητήρα και την διεύθυνση του ιδιοκτήτη με το ποσοστό του.

```
function deleteSensor(uint _id) external hasAccess(_id) exists(_id){
        sensorsTable[_id].deleted=true;
        emit SensorDeleted(_id);
      }
```

Η εξωτερική συνάρτηση **deleteSensor** χρησιμοποιείται για να δηλώσει ότι ένας αισθητήρας έχει διαγραφεί και επομένως δεν είναι πλέον διαθέσιμος προς πώληση, ενημερώνοντας στη συνέχεια το δίκτυο με το αντίστοιχο event.

```
function buySensor(uint _id, bool _useToken, address _buyerContractAddress, string calldata _buyerQuorumPublicKey) external payable exists(_id){
        require(msg.sender!=adminsTable[_id].adminAddress);
        address payable _buyerAddress = msg.sender;
        uint _price = sensorsTable[_id].price*oneEther;
        address payable _adminAddress = adminsTable[_id].adminAddress;
    address _adminContractAddress = adminsTable[_id].adminContractAddress;

        if(_price==msg.value || _useToken)
        {
          if(!_useToken)
          _adminAddress.transfer(msg.value);

                purchasesTable.push(purchase(_id, _useToken, _buyerAddress, _buyerContractAddress, _buyerQuorumPublicKey,_adminAddress, _adminContractAddress,true, false));
                emit PurchaseInfo(purchasesTable.length-1, _id, _useToken, _buyerAddress, _buyerContractAddress, _buyerQuorumPublicKey, _adminAddress, _adminContractAddress);
        }
      }
```

Η συνάρτηση **buySensor** χρησιμοποιείται για να γίνει η αγορά ενός αισθητήρα είτε με Ether είτε με Ntua Token. Εκτός από external, είναι επιπλέον και payable, που σημαίνει ότι μπορεί να δεχθεί με κάθε συναλλαγή αξία σε ether (msg.value), καθώς σε διαφορετική περίπτωση οι συναλλαγές που περιέχουν ether θα απορρίπτονταν. Δέχεται

ως ορίσματα το id του αισθητήρα, μια boolean μεταβλητή που δηλώνει αν η πληρωμή γίνεται με ether η Ntua Token (useToken), την διεύθυνση του User smart contact του αγοραστή και το δημόσιο κλειδί του Quorum που αντιστοιχεί στον κόμβο που χρησιμοποιεί.

Αρχικά χρησιμοποιείται ένα require που ελέγχει ότι ο αγοραστής δεν είναι ο διαχειριστής, ώστε να αποφευχθεί η άσκοπη αγορά. Αφού ανακτηθούν τα στοιχεία του διαχειριστή από τον πίνακα που είχαν καταχωρηθεί, ελέγχεται ότι έχει αποσταλεί το σωστό ποσό, αν η πληρωμή γίνεται με Ether. Στη συνέχεια με τη χρήση της _adminAddress.transfer (msg.value) γίνεται η μεταφορά του ποσού στον διαχειριστή και προστίθεται μια νέα καταχώρηση στον πίνακα που κρατάει το ιστορικό όλων των πληρωμών μαζί με τα απαραίτητα στοιχεία. Καθώς η ολοκλήρωση της συναλλαγής απαιτεί την επιβεβαίωση του διαχειριστή, στην κατοχή του οποίου βρίσκεται και το κρυφό συνθηματικό που επιτρέπει την πρόσβαση στα δεδομένα, υπάρχει μια boolean μεταβλητή (completed) που δηλώνει ότι ακόμα η συναλλαγή δεν έχει ολοκληρωθεί, η οποία αρχικοποιείται ως false. Στο τέλος μέσω τον events γίνονται γνωστά τα στοιχεία της συναλλαγής στο δίκτυο.

```
    function completePurchase(uint _purchaseId, uint _id, bool _useToken, bool
_acceptPurchase) public payable hasAccess(_id) {
        require(!purchasesTable[_purchaseId].completed);
        require(purchasesTable[_purchaseId].id==_id);
    if(sensorsTable[_id].deleted || !sensorsTable[_id].exists)_acceptPurchase=false;
        uint _price = sensorsTable[_id].price*oneEther;
        uint _amountSent;
        if(!_useToken){
          require(_price==msg.value);
          if(!_acceptPurchase) {
            (purchasesTable[_purchaseId].buyerAddress).transfer(_price);
            } else{
              for(uint i=0; i<ownersTable[_id].length;i++){
                address payable _ownerAddress=ownersTable[_id][i].ownerAddress;
                uint _toSend=(ownersTable[_id][i].ownerPercentage*_price)/100;
                _amountSent+=_toSend;
                (_ownerAddress).transfer(_toSend);
              }
              (msg.sender).transfer(_price-_amountSent);
```

```
            }
          }
          purchasesTable[_purchaseId].completed=true;
          emit PurchaseCompleted(_purchaseId, _id, _acceptPurchase);
        }
```

Η εξωτερική συναρτηση **completeTransaction** χρησιμοποιείται από τους διαχειριστές των αισθητήρων για να ολοκληρώσουν την πώληση, ώστε ο αγοραστής να αποκτήσει πλέον πρόσβαση στα δεδομένα. Αρχικά με δύο require ελεγχεται αν η συναλλαγή ακόμα εκκρεμεί και αν τα στοιχεία purchaseId και Id ταιριάζουν με αυτά πού είναι αποθηκευμένα στον πίνακα purchasesTable, ενώ σε διαφορετική περίπτωση η εκτέλεση σταματά εκεί και στέλνει μήνυμα σφάλματος. Αν η πληρωμή έχει γίνει με το Ntua Token το μόνο που κάνει αυτή η συνάρτηση είναι να αλλαζει την μεταβλητή που δηλώνει ότι η πληρωμή ολοκληρώθηκε σε true και να στέλνει το αντίστοιχο event που ενημερώνει για την ολοκλήρωση, είτε αν έγινε αποδεκτή είτε αν ακυρώθηκε μέσω του _acceptPurchase. Αν τυχόν έχει γίνει διαγραφή ενώ η συναλλαγή εκκρεμεί γίνεται αυτόματα ακύρωσή της. Η ακύρωση, είτε αυτόματα είτε με επιλογή του πωλητή, συνεπάγεται και επιστροφή του ποσού πίσω στον αγοραστή. Αν ο πωλητής επιλέξει να αποδεχτεί τη πωληση, υπολογίζεται το ποσό που αντιστοιχεί σε κάθε ιδιοκτήτη και του μεταφέρεται το αντίστοιχο ποσό από το λογαριασμό του διαχειριστή που είχε λάβει προηγουμένως το συνολικό ποσό. Το event ενημερώνει με βάση τις παραμέτρους για το τρόπο που ολοκληρώθηκε η συναλλαγή.

```
    function addUser(address _userNodeCoinbase,address
_userContractAddress, string calldata _usersQuorumPublicKey) external{
            address _userAddress = msg.sender;
            usersContractAddressTable[_userAddress]=_userContractAddress;
        emit UserAdded(_userAddress, _userNodeCoinbase, _userContractAddress,
_usersQuorumPublicKey);
        }
```

Η εξωτερική συναρτηση **addUser**, που χρησιμοποιείται κάθε φορά που προστίθεται ένας νέος χρήστης, καταχωρεί τη διεύθυνση του νέου User Smart Contract στον αντίστοιχο πίνακα και στέλνει το event UserAdded για να κανει τα στοιχεία του νέου χρήστη διαθέσιμα σε όλους.

```
    function getAdmin(uint _id) public view returns (address[2] memory) {
                                    return    [adminsTable[_id].adminContractAddress,
adminsTable[_id].adminContractAddress];
            }
```

```
    function   getUserContractAddress(address   _userAddress)   public
view returns (address) {
            return usersContractAddressTable[_userAddress];
            }
```

Οι συναρτήσεις **getAdmin** και **getAdminAddress** κάνουν ότι αναφέρει και το όνομα τους, δηλαδή δίνουν εύκολη πρόσβαση στα στοιχεία των διαχειριστών. Εφόσον είναι συναρτήσεις view δεν μπορούν να κάνουν καμία αλλαγή στο Smart Contract, παρά μόνο να λάβουν ήδη υπάρχοντα δεδομένα.

5.3.3 Reputation Smart Contract

Στην συνέχεια γίνεται αναλυτική παρουσίαση του κώδικα του συμβολαίου Reputation.

```
    struct purchase{
        uint id;
        address buyerAddress;
        address sellerAddress;
        int reviewRating;
        int feedbackRating;
        bool exists;
      }
```

Η δομή δεδομένων **purchase** χρησιμοποιείται για την αποθήκευση των στοιχείων κάθε πώλησης.

id: το αναγνωριστικό του αισθητήρα.
buyerAddress: η διεύθυνση του αγοραστή.
sellerAddress: η διεύθυνση του πωλητή.
reviewRating: η βαθμολογία του review από τον αγοραστή.
feedbackRating: η βαθμολογία του feedback από τον πωλητή.

exists:χρησιμοποιείται για να ελέγχεται εύκολα αν υπάρχει καταχώριση για την πώληση.

```
    mapping(address => int) private sellersReputationTable;
    mapping(address => int) private buyersReputationTable;
    mapping(uint => int) private sensorsReputationTable;
    mapping(uint => purchase) private purchasesTable;
```

Χρησιμοποιούνται τέσσερα mappings για την αποθήκευση των δεδομένων του Smart Contract.

Τα mappings **sellersReputationTable** και **buyersReputationTable** για την δοθείσα διεύθυνση επιστρέφουν το rating του πωλητή ή του αγοραστή αντίστοιχα. Το mapping **sensorReputationTable** δέχεται το αναγνωριστικό Id ενός αισθητήρα και επιστρέφει το rating του, ενώ το **purchasesTable** δέχεται το purchaseId και επιστρέφει την καταχώρηση που υπάρχει για την συγκεκριμένη πώληση.

```
    event ReviewAdded(address sellerAddress, address buyerAddress, uint
id, uint purchaseId, int rating);
    event FeedbackAdded(address sellerAddress, address buyerAddress,
uint id, uint purchaseId, int rating);
```

Το event **ReviewAdded** χρησιμοποιείται για να γίνει γνωστο στο δίκτυο το rating του review και να μπορούν οι υπόλοιποι χρήστες να το λάβουν υπόψη πριν τις μελλοντικές τους αγορές. Περιέχει τις διευθύνσεις του πωλητή, του αγοραστή, τους αυξοντες αριθμους του αισθητήρα (Id) και της πώλησης (purchaseId) και τη βαθμολογία.

Το event **FeedbackAdded** εκτελεί ακριβώς την ίδια λειτουργία στην περίπτωση ενός feedback.

```
    modifier purchaseExists(uint _purchaseId) {
      require(purchasesTable[_purchaseId].exists);
      _;
    }
```

Το modifier **purchaseExists** ελέγχει αν έχει καταχωρηθεί πώληση με το purchaseId που δηλώνεται, ώστε να είναι δυνατόν να βαθμολογούνται οι συμμετέχοντες, μόνο για υπαρκτές πωλήσεις και να αποφεύγεται κατάχρηση του συστήματος με ψευδείς βαθμολογίες για ανύπαρκτες αγορές.

```
        function getSellerReputation(address _sellerAddress) public view returns
(int){
        return sellersReputationTable[_sellerAddress];
        }

        function getBuyerReputation(address _buyerAddress) public view returns
(int){
        return buyersReputationTable[_buyerAddress];
        }

        function getSensorReputation(uint _id) public view returns (int){
        return sensorsReputationTable[_id];
        }
```

Οι συναρτήσεις **getSellerReputation**, **getBuyerReputation** και **getSensorReputation** έχουν δημιουργηθεί ώστε να είναι άμεσα διαθέσιμα τα ratings πωλητών και αγοραστών και αισθητήρων.

```
        function addPurchase(uint _purchaseId, uint _id, address _buyerAddress)
external{
        address _sellerAddress=msg.sender;
                purchasesTable[_purchaseId] = purchase(_id, _buyerAddress,
_sellerAddress, 0, 0, true) ;
        }
```

Η εξωτερική συναρτηση **addPurchase**, που χρησιμοποιείται κάθε φορά που προστίθεται μία νέα πώληση, καταχωρεί τα στοιχεία της, δηλαδή το αναγνωριστικό της (purchaseId), το αναγνωριστικό του αισθητήρα που αφορά (id) και τη διεύθυνση του αγοραστή στον αντίστοιχο πίνακα. Με αυτό τον τρόπο εξασφαλίζεται ότι τα reviews και

τα feedbacks που θα δοθούν θα πρέπει να αφορούν κάποια καταχωρημένη πώληση, επομένως δε θα μπορεί κάποιος κακόβουλος χρήστης να καταχραστεί το συστημα.

```
            function giveReview(uint _purchaseId, int _rating) external
purchaseExists(_purchaseId){
        address _buyerAddress=msg.sender;

        if(purchasesTable[_purchaseId].buyerAddress==_buyerAddress
          && purchasesTable[_purchaseId].reviewRating==0
          && (_rating==1 || _rating==-1)){
            purchasesTable[_purchaseId].reviewRating=_rating;

sellersReputationTable[purchasesTable[_purchaseId].sellerAddress]+=_rating;
            sensorsReputationTable[purchasesTable[_purchaseId].id]+=_rating;
                    emit ReviewAdded(purchasesTable[_purchaseId].sellerAddress,
_buyerAddress,
              purchasesTable[_purchaseId].id,_purchaseId, _rating);
          }
        }
```

Η εξωτερική συναρτηση **giveReview**, χρησιμοποιείται για δίνουν review οι αγοραστές, δεχόμενη ως ορίσματα το αναγνωριστικό της πώλησης και την δοσμένη βαθμολογία (_rating). Αφού εξασφαλιστεί ότι η πώληση υπάρχει μέσω του αντίστοιχου modifier, ελέγχεται επιπλέον αν ο πωλητής είναι αυτός που είχε δηλωθεί αρχικά, σε διαφορετική περίπτωση η βαθμολογία δεν καταχωρείται. Στη συνέχεια ελέγχεται ότι δεν έχει δοθεί ήδη βαθμολογία, αλλά και ότι η τιμή της είναι 1 αν πρόκειται για θετική είτε -1 αν πρόκειται για αρνητική, ενώ κάθε άλλη τιμή δε γίνεται αποδεκτή. Τέλος στέλνεται το event ReviewAdded που ενημερώνει το δίκτυο για την προσθήκη του review.

```
    function giveFeedback(uint _purchaseId, int _rating) external
purchaseExists(_purchaseId){
        address _sellerAddress=msg.sender;

        if(purchasesTable[_purchaseId].sellerAddress==_sellerAddress
          && purchasesTable[_purchaseId].feedbackRating==0
          && (_rating==1 || _rating==-1)){
           purchasesTable[_purchaseId].feedbackRating=_rating;

buyersReputationTable[purchasesTable[_purchaseId].buyerAddress]+=_rating;
                                        emit FeedbackAdded(_sellerAddress,
purchasesTable[_purchaseId].buyerAddress,
             purchasesTable[_purchaseId].id,_purchaseId, _rating);
          }
```

Η εξωτερική συναρτηση **giveFeedback**, χρησιμοποιείται για να δίνουν feedback οι πωλητές και λειτουργεί με τρόπο αντίστοιχο με εκείνον της giveReview.

5.3.4 User Smart Contract

Στην συνέχεια γίνεται αναλυτική παρουσίαση του κώδικα του συμβολαίου User.

```
    address ownerAddress;
    address nodeCoinbase;

    mapping(uint=>string) private ownedPassphrases;
    mapping(uint=>string) private boughtPassphrases;
```

Το Smart Contract User έχει 2 μεταβλητές που περιέχουν τη διευθυνση του χρήστη αλλά και το coinbase του κόμβου του χρησιμοποιεί. Υπάρχουν 2 mappings, **ownedPassphrases** και **boughtPassphrases**, που περιέχουν το σύνολο των passphrases των αισθητήρων που έχει διαθέσιμα προς πώληση κάποιος πωλητής/διαχειριστής ή αντίστοιχα που έχει ήδη αγοράσει κάποιος. Να σημειωθεί ότι ο ίδιος λογαριασμός μπορεί να κάνει παράλληλα αγορές και πωλήσεις.

```
constructor(address _nodeCoinbase) public {
  ownerAddress=msg.sender;
  nodeCoinbase = _nodeCoinbase;
}
```

Ο κατασκευαστής (constructor) του συμβολαίου εκτελείται αυτόματα μόλις το συμβόλαιο ενεργοποιηθεί στο Blockchain. Αρχικοποιεί τις μεταβλητές **ownerAddress** και **nodeCoinbase** του συμβολαίου. Με τη χρήση του msg.sender αρχικοποιείται με ασφάλεια η μεταβλητή που ορίζει τον λογαριασμό που είναι ο ιδιοκτήτης του Smart Contract και είναι επομένως ο αποστολέας της συναλλαγής.

```
modifier hasAccess() {
  require(msg.sender==ownerAddress);
  _;
}
```

Το modifier **hasAccess** ελέγχει αν ο χρήστης που καλεί μια συνάρτηση έχει το δικαίωμα να το κάνει, δηλαδή στη συγκεκριμένη περίπτωση ελέγχει αν πρόκειται για τον ιδιοκτήτη του Smart Contract, μέσω του require.

```
function addPassphrase(uint _id, string calldata _passphrase) external hasAccess(){
  ownedPassphrases[_id]=_passphrase;
}

function addBoughtPassphrase(uint _id, string calldata _passphrase) external {
  boughtPassphrases[_id]=_passphrase;
}

function changePassphrase(uint _id, string calldata _passphrase) external hasAccess(){
  ownedPassphrases[_id]= _passphrase;
}
```

Η εξωτερική συνάρτηση **addPassphrase** δέχεται ως ορίσματα τον μοναδικό αύξοντα αριθμό του αισθητήρα (id) μαζί με το κρυφό passphrase και τα αποθηκεύει στον αντίστοιχο πίνακα.

Η εξωτερική συνάρτηση **addBoughtPassphrase** εκτελεί την ίδια λειτουργία με την **addPassphrase** με την διαφορα ότι καλείται απο το frontend όταν ολοκληρώνεται κάποια πώληση, λαμβάνοντας τις πληροφορίες αυτες απο τον πωλητή. Ο σκοπός της είναι η αποθήκευση του αγορασμένου passphrase για μελλοντική χρήση.

Μέσω της εξωτερικής συνάρτησης **changePassphrase** μπορεί ο διαχειριστής να αλλάξει το passphrase.

```
        function getPassphrase(uint _id) external view  hasAccess() returns (string
memory){
          return ownedPassphrases[_id];
        }

        function getBoughtPassphrase(uint _id) external view hasAccess() returns
(string memory){
          return boughtPassphrases[_id];
        }
      }
```

Οι συναρτησεις **getPassphrase** και **getBoughtPassphrase** χρησιμοποιούνται για να αποκτήσει ο χρήστης πρόσβαση στα passphrases είτε των αισθητήρων που διαθέτει προς πώληση είτε αυτών που έχει αγοράσει.

5.3.4 NTUA Token Smart Contract

Στην συνέχεια γίνεται αναλυτική παρουσίαση του κώδικα του συμβολαίου NTUA Token.

```
      pragma solidity ^0.5.7;
contract Token {
  event Transfer(address indexed _from, address indexed _to, uint256 _value);
```

Το event Transfer χρησιμοποιείται για να ενημερώσει τους κόμβους του δικτύου ότι έχει γίνει μεταφορά NTUA Tokens απο την διεύθυνση from στη διεύθυνση to, ενώ η αξία τους δηλώνεται μέσω του value.

```
        mapping(address => uint256) public balanceOf;

    string public name;
    string public symbol;
    uint8 public decimals;
    uint256 public totalSupply;
```

Οι μεταβλητές του έξυπνου συμβολαίου χρησιμοποιούνται για την αποθήκευση της πληροφορίας που δηλώνει και το όνομά τους, ενώ υπάρχει και το mapping balanceOf στο οποίο αποθηκεύονται τα υπόλοιπα όλων των λογαριασμών σε NTUA Tokens.

```
        constructor() public{
      balanceOf[0xed9d02e382b34818e88B88a309c7fe71E65f419d] = 1000 * 1 ether;
      balanceOf[0xcA843569e3427144cEad5e4d5999a3D0cCF92B8e] = 1000 * 1 ether;
      balanceOf[0x0fBDc686b912d7722dc86510934589E0AAf3b55A] = 1000 * 1 ether;
      balanceOf[0x9186eb3d20Cbd1F5f992a950d808C4495153ABd5] = 1000 * 1 ether;
      balanceOf[0x0638E1574728b6D862dd5d3A3E0942c3be47D996] = 1000 * 1 ether;
      balanceOf[0xAE9bc6cD5145e67FbD1887A5145271fd182F0eE7] = 1000 * 1 ether;
      balanceOf[0xCC71C7546429a13796cf1BF9228bFf213e7Ae9cc] = 1000 * 1 ether;
      balanceOf[0xa9e871F88CBeb870d32D88E4221dcfBD36Dd635a] = 1000 * 1 ether;
      name = "NtuaToken";
      symbol = "NtuaTok";
      decimals = 18;
            totalSupply = 100000000 * 1 ether;
    }
```

Όταν το έξυπνο συμβόλαιο NTUA Token ενεργοποιηθεί στο Blockchain, εκτελείται αρχικά ο κατασκευαστής (constructor). Στη συγκεκριμένη περίπτωση, καθώς χρησιμοποιούνται 10 δοκιμαστικοί λογαριασμοί, μεταφέρεται στον καθέναν από αυτούς ένα προκαθορισμένο ποσό, ώστε να χρησιμοποιηθεί για αγορές εντός της εφαρμογής. Επιπλέον αρχικοποιεί τις μεταβλητές του συμβολαίου.

```
    function balanceof(address _owner) external view returns (uint256 balance){
  return balanceOf[_owner];
 }
```

Η external (εξωτερική) συνάρτηση balanceOf μας επιτρέπει να βρούμε εύκολα το υπόλοιπο σε NTUA Tokens οποιουδήποτε λογαριασμού.

```
    function _transfer(address _from, address _to, uint _value) internal {
      require(balanceOf[_from] >= _value && balanceOf[_to] + _value >= balanceOf[_to]);
   balanceOf[_from] -= _value;
   balanceOf[_to] += _value;
   emit Transfer(_from,_to,_value);
 }
```

Η internal (εσωτερική) συνάρτηση _transfer, η οποία μπορεί να κληθεί μόνο από κάποια άλλη συνάρτηση του συμβολαίου, χρησιμοποιείται για να μεταφερθούν NTUA Tokens ανάμεσα σε δύο λογαριασμούς, ελέγχοντας προφανώς αν το υπόλοιπο του αποστολέα επαρκεί. Η συνάρτηση αυτή ενημερώνει το δίκτυο για τη συναλλαγή μέσω του event Transfer.

```
    function transfer(address _to, uint256 _value) external{
  _transfer(msg.sender,_to,_value);
 }
```

Η external (εξωτερική) συνάρτηση συνάρτηση transfer για να μεταφέρει όσα NTUA Tokens καθορίζονται από την τιμή value, από έναν χρήστη (msg.sender) σε έναν άλλον (_to). Η συνάρτηση αυτή καλείται μόνο εξωτερικά και για να γίνει η μεταφορά καλεί την εσωτερική συνάρτηση _transfer.

5.4 Ανάλυση λειτουργιών ιστοσελίδας

Στη συνέχεια θα περιγράφουν οι βασικές λειτουργίες της εφαρμογής και η ακολουθία των κλήσεων των συναρτήσεων που χρησιμοποιούνται για υλοποιηθούν οι διαδικασίες της.

5.4.1 Add Sensor

Όταν ο χρήστης θέλει να προσθέσει ένα νέο αισθητήρα και να ανεβάσει τα δεδομένα του στη βάση δεδομένων, θα πρέπει να εισαγάγει τις πληροφορίες του στα αντίστοιχα πεδία της φόρμας Add sensor της ιστοσελίδας management αλλά και να επιλέξει το αρχείο που περιέχει τα δεδομένα. Οι πληροφορίες οι οποίες συμπληρώνει ο χρήστης είναι το όνομα του αισθητήρα, η τιμή του, η χρονική στιγμή έναρξη των μετρήσεων, η συχνότητα των μετρήσεων, οι γεωγραφικές συντεταγμένες του καθώς και ένα passphrase το οποίο είναι ο μοναδικός τρόπος να αποκτήσει κάποιος πρόσβαση στα καταχωρημένα δεδομένα. Για να σταλούν τα δεδομένα στη βάση δεδομένων ο χρήστης επιλέγει μέσα από το πεδίο file input το αρχείο στο οποίο είναι αποθηκευμένα τα δεδομένα και αν προηγουμένως έχει καταχωρήσει σωστά τις υπόλοιπες πληροφορίες, μπορεί να προσθέσει το νέο αισθητήρα πατώντας το κουμπί Add Sensor.

Μετά το πάτημα του αυτού του κουμπιού η ιστοσελίδα θα αρχίσει τη διαδικασία εκτελώντας αρχικά τη JavaScript συνάρτηση loadFile η οποία είναι υπεύθυνη για την εύρεση του επιλεγμένου αρχείου στον υπολογιστή του χρήστη αλλά και για τη μετατροπή του στην κατάλληλη μορφή, ώστε να είναι δυνατή η αποστολή του στο server και στη βάση δεδομένων στη συνέχεια.

Όταν η συνάρτηση loadFile ολοκληρώσει την φόρτωση του αρχείου καλεί τη συνάρτηση addSensor, δίνοντάς της ως όρισμα τα δεδομένα στην κατάλληλη μορφή. Αρχικά παίρνει τα στοιχεία που ο χρήστης πληκτρολόγησε στην φόρμα, όπως αναφέρθηκε προηγουμένως, τα φέρνει στη κατάλληλη μορφή και αφού ελέγξει ότι δεν έχει παραληφθεί κάποιο, ενημερώνει τον χρήστη με το αντίστοιχο μήνυμα.

Στη συνέχεια καλεί τη συνάρτηση addSensor του Management smart contract, η οποία, όπως αναλύθηκε στο προηγούμενο μέρος, καταχωρεί τα δεδομένα του αισθητήρα στις αντίστοιχες δομές δεδομένων και μέσω των event SensorAdded και adminAdded ενημερώνει το δίκτυο για την νέα καταχώρηση και για το λογαριασμό που δηλώθηκε ως διαχειριστής.

Ακολουθεί η συνάρτηση addPassphrase του smart contract User ώστε να αποθηκευτεί με ασφάλεια η μυστική passphrase,κάτι που γίνεται δυνατό από τη χρήση του Quorum, καθώς δεν επιτρέπει σε οποιοδήποτε άλλο χρήστη να έχει πρόσβαση στη πληροφορία αυτή. Για να συσχετιστούν οι δύο καταχωρήσεις στα διαφορετικά smart

contracts χρησιμοποιείται ο μοναδικός αριθμός ID πού αντιστοιχεί στον συγκεκριμένο αισθητήρα.

Με σκοπό τη συνεχή ενημέρωση των χρηστών/αγοραστών για τους αισθητήρες που είναι διαθέσιμοι υπάρχει η συνάρτηση getEventsSensorAdded, η οποία ανά τακτά χρονικά διαστήματα “σκανάρει” το Blockchain για τα προαναφερθέντα events και δημιουργεί δυναμικά τη λίστα Available Sensors με όλους τους διαθέσιμους αισθητήρες και τα χαρακτηριστικά τους και τα παρουσιάζει στον αγοραστή στο προκαθορισμένο σημείο στην ιστοσελίδα μαζί με 2 κουμπιά “Buy with Ether” και “Buy with Ntua Token”, όπως θα δούμε στη συνέχεια.

```
        function addSensor(data){
  if(!userExists){
    alert("You must create a User account first!")
  } else {
  var sensorName = document.getElementById("sensorName").value;
  var price = parseInt(document.getElementById("price").value, 10);
  var startTime = parseInt(document.getElementById("startTime").value, 10);
  var frequency = parseInt(document.getElementById("frequency").value, 10);
  var latitude = parseInt(document.getElementById("latitude").value, 10);
  var longitude = parseInt(document.getElementById("longitude").value, 10);
  var passphrase = document.getElementById("passphrase").value;

  if(sensorName=="" || latitude==""|| longitude==""|| passphrase==""||
  isNaN(price) || isNaN(startTime) || isNaN(frequency)){
    alert("Empty field");
  }else{

MANAGEMENTCONTRACT.methods.addSensor.apply(MANAGEMENTCONTRACT,
                [sensorName, price, startTime, frequency, latitude, longitude,
currentUserContractAddress])
      .send({from:currentUserAddress ,gas: gasGlobal})
      .then(function(result){
        USERCONTRACT.methods.addPassphrase.apply(USERCONTRACT,
          [result.events.SensorAdded.returnValues.id, passphrase])
          .send({from: currentUserAddress, gas:gasGlobal,
```

```
privateFor:[currentUserQuorumPublicKey]});
    });

//Dynamically create a form in order to send data to Server and then to Database
      var formSentData = document.createElement("form");
      formSentData.setAttribute("id", "sentData");
      formSentData.setAttribute("method", "post");
      formSentData.setAttribute("action", "/sentData");

      var hiddenFieldSensorName = document.createElement("input");
      hiddenFieldSensorName.setAttribute("type", "hidden");
      hiddenFieldSensorName.setAttribute("name", "inputSensorName");
      hiddenFieldSensorName.setAttribute("value", sensorName);

      var hiddenFieldPrice = document.createElement("input");
      hiddenFieldPrice.setAttribute("type", "hidden");
      hiddenFieldPrice.setAttribute("name", "inputPrice");
      hiddenFieldPrice.setAttribute("value", price);

      var hiddenFieldStartTime = document.createElement("input");
      hiddenFieldStartTime.setAttribute("type", "hidden");
      hiddenFieldStartTime.setAttribute("name", "inputStartTime");
      hiddenFieldStartTime.setAttribute("value", startTime);

      var hiddenFieldFrequency = document.createElement("input");
      hiddenFieldFrequency.setAttribute("type", "hidden");
      hiddenFieldFrequency.setAttribute("name", "inputFrequency");
      hiddenFieldFrequency.setAttribute("value", frequency);

      var hiddenFieldLatitude = document.createElement("input");
      hiddenFieldLatitude.setAttribute("type", "hidden");
      hiddenFieldLatitude.setAttribute("name", "inputLatitude");
      hiddenFieldLatitude.setAttribute("value", latitude);

      var hiddenFieldLongitude = document.createElement("input");
      hiddenFieldLongitude.setAttribute("type", "hidden");
```

```
        hiddenFieldLongitude.setAttribute("name", "inputLongitude");
        hiddenFieldLongitude.setAttribute("value", longitude);

        var hiddenFieldPassphrase = document.createElement("input");
        hiddenFieldPassphrase.setAttribute("type", "hidden");
        hiddenFieldPassphrase.setAttribute("name", "inputPassphrase");
        hiddenFieldPassphrase.setAttribute("value", passphrase);

        var hiddenFieldAdmin = document.createElement("input");
        hiddenFieldAdmin.setAttribute("type", "hidden");
        hiddenFieldAdmin.setAttribute("name", "inputAdmin");
        hiddenFieldAdmin.setAttribute("value", currentUserAddress);

        var hiddenFieldJson = document.createElement("input");
        hiddenFieldJson.setAttribute("type", "hidden");
        hiddenFieldJson.setAttribute("name", "inputJson");
        hiddenFieldJson.setAttribute("value", JSON.stringify(data));

        formSentData.appendChild(hiddenFieldSensorName);
        formSentData.appendChild(hiddenFieldPrice);
        formSentData.appendChild(hiddenFieldStartTime);
        formSentData.appendChild(hiddenFieldFrequency);
        formSentData.appendChild(hiddenFieldLatitude);
        formSentData.appendChild(hiddenFieldLongitude);
        formSentData.appendChild(hiddenFieldAdmin);
        formSentData.appendChild(hiddenFieldPassphrase);
        formSentData.appendChild(hiddenFieldJson);
        document.getElementById("formArea").appendChild(formSentData);
        sentForm("#sentData");
        clearElement("formArea");
      }
    }
  }
```

5.4.2 Edit Sensor

Η ιστοσελίδα μέσω των συναρτήσεων GetEventsSensorAdded και GetEventsAdminAdded, οι οποίες χρησιμοποιούν τα events που στάλθηκαν κατα τη

προηγούμενη διαδικασία, δημιουργεί και διατηρεί ενημερωμένους τους πίνακες sensors (όλοι οι διαθέσιμοι αισθητήρες), mySensors (οι αισθητήρες που διαχειρίζεται ο τρέχων χρήστης) και admins (αντιστοιχίζει κάθε αισθητήρα με την διεύθυνση του διαχειριστή του). Με αυτό το τρόπο δημιουργείται δυναμικά στο προκαθορισμένο σημείο η λιστα My Sensors, που περιέχει τους αισθητήρες που έχουν καταχωρηθεί απο τον τρέχων χρήστη/πωλητή, ώστε να είναι εύκολη η διαχείρισή τους. Η λίστα My Sensors προκύπτει απο το φιλτράρισμα του συνόλου των αισθητήρων, επιλέγοντας μόνο εκείνους που έχουν αντιστοιχηθεί με την διεύθυνση του τρέχοντος χρήστη και τη διεύθυνση του User Smart Contract του. Επομένως ο χρήστης, μέσα από την ενότητα My Sensors που εμφανίζεται στην ιστοσελίδα, μπορεί να δει τους αισθητήρες που του ανήκουν αλλά και να τους τροποποιήσει χρησιμοποιώντας το κουμπί edit ή να τους διαγράψει μέσω του κουμπιού delete, όπως θα δούμε στη συνέχεια.

Επιλέγοντας το κουμπί edit εμφανίζεται μία νέα φόρμα Edit Sensor, η οποία του δίνει τη δυνατότητα να θέσει μία νέα τιμή (updatePrice), να αλλάξει το passphrase (updatePassphrase) αλλά και να αλλάξει το ποσοστό ιδιοκτησίας ενός ιδιοκτήτη ή να προσθέσει ένα καινούργιο ιδιοκτήτη (updateOwnership). Η καθεμία από αυτές τις διαδικασίες θα περιγραφεί αναλυτικά στη συνέχεια.

Αφού ο χρήστης κάνει μία ή περισσότερες τροποποιήσεις με το με το πάτημα του κουμπιού save καλείται η JavaScript συνάρτηση editSensor, η οποία ελέγχει αρχικά αν τα νέα δεδομένα είναι σε σωστή μορφή, αλλά και αν ο χρήστης έχει δικαίωμα να τροποποιήσει το συγκεκριμένο αισθητήρα, που αν και έχει ήδη ελεγχθεί προηγουμένως, ο δεύτερος ελεγχος εξασφαλίζει ότι δεν έχει αλλάξει κάτι στο χρόνο που μεσολάβησε από τον πρώτο, όπως για παράδειγμα να έχει διαγραφεί.

Όταν η συνάρτηση editSensor, η οποία έχει το ρόλο του ενδιάμεσου, κάνει τους απαραίτητους ελέγχους, καλεί μία ή περισσότερες από τις συναρτήσεις updatePrice, updatepassphrase και updateOwnership, ανάλογα με το ποιες αλλαγές έχει επιλέξει ο χρήστης να πραγματοποιήσει. Σε περίπτωση που ο χρήστης επιλέξει να μην πραγματοποιήσει καμία μεταβολή υπάρχει και η επιλογή του κουμπιού cancel που ακυρώνει την επεξεργασία και κλείνει το πεδίο Edit Sensor.

```
    function editSensor(idEdit){
var idTemp=idEdit;
var inputNewPrice = document.createElement("input");
```

```
  inputNewPrice.id = "newPrice"+idTemp;
  inputNewPrice.type = "text";
  inputNewPrice.placeholder = "New Price";

  var inputNewPassphrase = document.createElement("input");
  inputNewPassphrase.id = "newPassphrase"+idTemp;
  inputNewPassphrase.type = "text";
  inputNewPassphrase.placeholder = "New Passphrase";

  var inputNewOwnerAddress = document.createElement("input");
  inputNewOwnerAddress.id = "newOwnerAddress"+idTemp;
  inputNewOwnerAddress.type = "text";
  inputNewOwnerAddress.placeholder = "New Owner Address";

  var inputNewOwnerPercentage = document.createElement("input");
  inputNewOwnerPercentage.id = "newOwnerPercentage"+idTemp;
  inputNewOwnerPercentage.type = "text";
  inputNewOwnerPercentage.placeholder = "New Owner Percentage";

  var buttonSaveChanges = document.createElement("button");
  buttonSaveChanges.innerText = "Save Changes";
  buttonSaveChanges.className = "button1";
  buttonSaveChanges.type = "button";
  buttonSaveChanges.value = idTemp;
  buttonSaveChanges.onclick = function () {

    var newPrice = parseInt(document.getElementById("newPrice"+idTemp).value, 10);
    var newPassphrase = document.getElementById("newPassphrase"+idTemp).value;
                                          var          newOwnerAddress          =
checkAddress(document.getElementById("newOwnerAddress"+idTemp).value);
                                        var          newOwnerPercentage          =
parseInt(document.getElementById("newOwnerPercentage"+idTemp).value, 10);

        if(isNaN(newPrice) && newPassphrase=="" && (newOwnerAddress=="" ||
isNaN(newOwnerPercentage))){
     alert("No changes!");
```

```
   } else{


if(admins[idTemp].adminContractAddress.localeCompare(currentUserContractAddres
s)!==0){
       alert("You are not admin.")
     }else{

       if(!isNaN(newPrice))
       updatePrice(idTemp, newPrice);

       if(newPassphrase!="")
       updatePassphrase(idTemp, newPassphrase);

       if(newOwnerAddress!="" && !isNaN(newOwnerPercentage))
       if(newOwnerPercentage<=0 || newOwnerPercentage>100){
         alert("Percentage is invalid.");
       } else{
         var percentageSum=0;
         if(typeof ownerships[idTemp] !== "undefined")
         ownerships[idTemp].owners.forEach(function(each) {
           if(each.ownerAddress.localeCompare(newOwnerAddress)!=0)
           percentageSum+=parseInt(each.ownerPercentage, 10);
         });
         if(percentageSum+newOwnerPercentage>100){
                 alert("Total ownership cant exceed 100%, current limit for this user is
"+(100-percentageSum)+"%")
         }else{
           updateOwnership(idTemp, newOwnerAddress, newOwnerPercentage);
           }
         }
       document.getElementById("editMySensor").classList.remove("well");
       clearElement("editMySensor");
     }
   }
 }
```

5.4.3 Update Price

Όταν ο χρήστης επιλέξει να αποθηκεύσει τις αλλαγές που έχει υποβάλλει στο πεδίο Edit Sensor της ιστοσελίδας, αν μία από αυτές είναι η τιμή του αισθητήρα, ξεκινά η διαδικασία εκτελώντας τη JavaScript συνάρτηση updatePrice. Ο σκοπός της είναι να αλλάξει την τιμή τόσο στον αντίστοιχο πίνακα που είναι αποθηκευμένη στο smart contract όσο και στη βάση δεδομένων.

Αρχικά καλείται η συνάρτηση updatePrice του Management Smart Contract, η οποία αλλάζει την τιμή και στέλνει το αντίστοιχο event για να ενημερώσει το δίκτυο. Για να αποκτήσει πρόσβαση ο χρήστης στη βάση δεδομένων ώστε να ενημερώσει και εκεί την καταχώρηση πρέπει να κληθεί πρώτα η συνάρτηση getPassphrase του Smart Contract User, ώστε να ανακτηθεί το passphrase και να επιτραπεί η πρόσβαση στη βάση δεδομένων. Όλη αυτή η διαδικασία γίνεται αυτοματοποιημένα, αφού πρώτα είχε εξασφαλιστεί ότι ο χρήστης είναι και ο διαχειριστής του συγκεκριμένου αισθητήρα. Η ιστοσελίδα ενημερώνεται για την αλλαγή σε οποιαδήποτε τιμή, αφου η συνάρτηση getEventsPriceUpdated πιάνει τα event PriceUpdated και αποθηκεύει τις νέες τιμές στον αντίστοιχο πίνακα.

```
      function updatePrice(idUpdatePrice, newPrice) {

MANAGEMENTCONTRACT.methods.updatePrice.apply(MANAGEMENTCONTRACT
, [idUpdatePrice, newPrice])
  .send({from:currentUserAddress ,gas: gasGlobal});


USERCONTRACT.methods.getPassphrase.apply(USERCONTRACT,[idUpdatePrice])
                    .call({from: currentUserAddress, gas:gasGlobal,
privateFor:[currentUserQuorumPublicKey]})
  .then(function(result){

    var formUpdatePrice = document.createElement("form");
    formUpdatePrice.setAttribute("id", "updatePrice");
    formUpdatePrice.setAttribute("method", "post");
    formUpdatePrice.setAttribute("action", "/updatePrice");

    var hiddenFieldSensorName = document.createElement("input");
```

```
    hiddenFieldSensorName.setAttribute("type", "hidden");
    hiddenFieldSensorName.setAttribute("name", "sensorName");
                                            hiddenFieldSensorName.setAttribute("value",
sensors[idUpdatePrice].sensorName);

    var hiddenFieldPassphrase = document.createElement("input");
    hiddenFieldPassphrase.setAttribute("type", "hidden");
    hiddenFieldPassphrase.setAttribute("name", "passphrase");
    hiddenFieldPassphrase.setAttribute("value", result);

    var hiddenFieldPrice = document.createElement("input");
    hiddenFieldPrice.setAttribute("type", "hidden");
    hiddenFieldPrice.setAttribute("name", "newPrice");
    hiddenFieldPrice.setAttribute("value", newPrice);

    formUpdatePrice.appendChild(hiddenFieldSensorName);
    formUpdatePrice.appendChild(hiddenFieldPrice);
    formUpdatePrice.appendChild(hiddenFieldPassphrase);
    document.getElementById("formArea").appendChild(formUpdatePrice);
    sentForm("#updatePrice");
    clearElement("formArea");
  });
}
```

5.4.4 Update Ownership

Όταν ο χρήστης πατήσει το κουμπί Save στο πεδίο Edit Sensor της ιστοσελίδας, αν έχει καταχωρήσει μία διεύθυνση και ένα ποσοστό ιδιοκτησίας στα πεδία της φόρμας ξεκινά η διαδικασία εκτελώντας τη JavaScript συνάρτηση updatePrice, που δέχεται ως ορίσματα το αναγνωριστικό ID του αισθητήρα, τη διεύθυνση και το ποσοστό του νέου ιδιοκτήτη. Ο χρήστης/διαχειριστής μέσω της διαδικασίας αυτής μπορεί είτε να μεταβίβασει ένα ποσοστό της ιδιοκτησίας του αισθητήρα σε έναν άλλο, νέο λογαριασμό είτε να τροποποιήσει το ποσοστό ενός ήδη υπάρχοντος ιδιοκτήτη. Εδώ να σημειωθεί ότι αρχικά ο χρήστης που καταχωρεί τον αισθητήρα είναι ο διαχειριστής του και εξ ορισμού είναι ο ιδιοκτήτης του 100% των δικαιωμάτων του. Αν ο διαχειριστής στη συνέχεια επιλέξει να προσθέσει νέους ιδιοκτήτες το ποσοστό που τους δίνει αφαιρείται από το αρχικό δικό του.

Αφού αρχικά έχει ελεγχθεί ότι ο τρέχων χρήστης είναι διαχειριστής του αισθητήρα και επομένως έχει δικαίωμα να κάνει αλλαγές, καλείται η αντίστοιχη συνάρτηση updatePrice του Management Smart Contract. Αυτή εξασφαλίζει ότι το δοθέν ποσοστό είναι εντός ορίων, δηλαδή το νέο συνολικό ποσοστό δεν ξεπερνά το 100% και στη συνέχεια αναζητά στον πίνακα με τους ιδιοκτήτες την διεύθυνση που έχει δοθεί ως όρισμα. Αν η αναζήτηση έχει αποτέλεσμα, ενημερώνει το ποσοστό του υπάρχον ιδιοκτήτη, ενώ αν πρόκειται για νέο ιδιοκτήτη προσθέτει στον αντίστοιχο πίνακα που αποθηκεύονται οι ιδιοκτήτες του συγκεκριμένου αισθητήρα μία νέα καταχώρηση με τη διεύθυνση και το ποσοστό. Τέλος γίνεται emit το event OwnershipChanged, με το οποίο ενημερώνει το δίκτυο για την αλλαγή. Όπως και σε προηγούμενες περιπτώσεις, γίνεται χρήση των events για να ενημερώσουμε την ιστοσελίδα, σκανάροντας τα νέα events για να εντοπίσουμε τις αλλαγές χρησιμοποιώντας τη συνάρτηση GetEventsOwnershipChanged, η οποία δημιουργεί ένα πίνακα όπου είναι αποθηκευμένοι οι ιδιοκτήτες των αισθητήρων τους οποίους διαχειρίζεται ο τρέχων χρήστης και τους παρουσιάζει στην αντίστοιχη ενότητα, ώστε να είναι εύκολα προσβάσιμες οι πληροφορίες στον διαχειριστή.

```
function updateOwnership(idUpdateOwnership, newOwnerAddress, newOwnerPercentage){

MANAGEMENTCONTRACT.methods.updateOwnership.apply(MANAGEMENTCONTRACT,
   [idUpdateOwnership, newOwnerAddress, newOwnerPercentage])
   .send({from:currentUserAddress ,gas: gasGlobal});
   alert("Sensor "+ idUpdateOwnership+ " ownership has been updated.");
 }
```

5.4.5 Update Passphrase

Η διαδικασία της αλλαγής του passphrase ξεκινά όταν ο χρήστης πληκτρολόγησει ένα νέο passphrase στο αντίστοιχο πεδίο της φόρμας Edit Sensor και επιλέξει το κουμπί Save. Στη συνέχεια αφού ελεγχθεί ότι ο χρήστης είναι ο διαχειριστής του συγκεκριμένου αισθητήρα, όπως και στις άλλες περιπτώσεις, αλλά και ότι το νέο passphrase δεν είναι κενό, καλείται η JavaScript συνάρτηση updatePassphrase η οποία δέχεται ως ορίσματα το αναγνωριστικό ID του αισθητήρα και το νέο passphrase.

Η συνάρτηση αυτή εκτελεί δύο λειτουργίες: αρχικά μέσω της κλήσης της συνάρτησης getPassphrase του Smart Contract User βρίσκει το τρέχον passphrase ώστε να αποκτήσει πρόσβαση στη βάση δεδομένων για να την ενημερώσει για την αλλαγή, και στη συνέχεια καλώντας τη συνάρτηση changePassphrase του ίδιου Smart Contract, ενημερώνει και αυτό για την αλλαγή.

Όσον αφορά οποιαδήποτε αγορά που έχει γίνει μέχρι τη στιγμή της αλλαγής του passphrase, δεν υπάρχει ενημέρωση και δεν μπορεί ο αγοραστής να έχει πλέον πρόσβαση στα δεδομένα. Επομένως το νέο passphrase θα το λάβουν όσες από τις εκκρεμείς αγορές ολοκληρωθούν, αλλά και οποιαδήποτε μελλοντική.

```
function updatePassphrase(idUpdatePassphrase, newPassphrase) {

USERCONTRACT.methods.getPassphrase.apply(USERCONTRACT,[idUpdatePassphrase])
                                .call({from: currentUserAddress, gas:gasGlobal, privateFor:[currentUserQuorumPublicKey]})
    .then(function(result){

      var formUpdatePassphrase = document.createElement("form");
      formUpdatePassphrase.setAttribute("id", "updatePassphrase");
      formUpdatePassphrase.setAttribute("method", "post");
      formUpdatePassphrase.setAttribute("action", "/updatePassphrase");

      var hiddenFieldSensorName = document.createElement("input");
      hiddenFieldSensorName.setAttribute("type", "hidden");
      hiddenFieldSensorName.setAttribute("name", "sensorName");
                                        hiddenFieldSensorName.setAttribute("value", sensors[idUpdatePassphrase].sensorName);

      var hiddenFieldPassphrase = document.createElement("input");
      hiddenFieldPassphrase.setAttribute("type", "hidden");
      hiddenFieldPassphrase.setAttribute("name", "passphrase");
      hiddenFieldPassphrase.setAttribute("value", result);

      var hiddenFieldNewPassphrase = document.createElement("input");
```

```
    hiddenFieldNewPassphrase.setAttribute("type", "hidden");
    hiddenFieldNewPassphrase.setAttribute("name", "newPassphrase");
    hiddenFieldNewPassphrase.setAttribute("value", newPassphrase);

    formUpdatePassphrase.appendChild(hiddenFieldSensorName);
    formUpdatePassphrase.appendChild(hiddenFieldPassphrase);
    formUpdatePassphrase.appendChild(hiddenFieldNewPassphrase);
    document.getElementById("formArea").appendChild(formUpdatePassphrase);
    sentForm("#updatePassphrase");
    clearElement("formArea");
   });

   USERCONTRACT.methods.changePassphrase.apply(USERCONTRACT,
    [idUpdatePassphrase, newPassphrase])
                                .send({from: currentUserAddress, gas:gasGlobal,
privateFor:[currentUserQuorumPublicKey]});
  }
```

5.4.6 Delete Sensor

Όπως αναφέρθηκε, ο χρήστης μπορεί να δει τους αισθητήρες που διαχειρίζεται στην ενότητα της ιστοσελίδας My sensors. Εκεί σε κάθε ξεχωριστή καταχώρηση υπάρχει το κουμπί Delete. Οταν πατηθεί, ξεκινά η διαδικασία της διαγραφής του συγκεκριμένου αισθητήρα.

Αρχικά, όπως και στις περιπτώσεις που αναφέρθηκαν ήδη, ελέγχεται αν ο τρέχων χρήστης είναι όντως ο διαχειριστής και όταν ολοκληρωθεί ο έλεγχος εκτελείται η JavaScript συνάρτηση deleteSensor. Αυτή με τη σειρά της καλεί τη συνάρτηση του Management Smart Contract deleteSensor, η οποία ενημερώνει τον αντίστοιχο πίνακα ότι ο αισθητήρας έχει διαγραφεί και κάνει emit το event SensorDeleted μέσα από το οποίο η ιστοσελίδα θα ενημερώσει τους πίνακές της και θα σταματήσει να τον προβάλλει ως διαθέσιμο.

Στη συνέχεια, αφού πρώτα ανακτηθεί το passhrase με τη χρήση της getPassphrase του User Smart Contract, ενημερώνεται και η βάση δεδομένων, μέσω του server, για τη διαγραφή.

```
        function deleteSensor(idDelete){

MANAGEMENTCONTRACT.methods.deleteSensor.apply(MANAGEMENTCONTRACT, [idDelete])
  .send({from:currentUserAddress ,gas: gasGlobal});

  USERCONTRACT.methods.getPassphrase.apply(USERCONTRACT,[idDelete])
                        .call({from: currentUserAddress, gas:gasGlobal, privateFor:[currentUserQuorumPublicKey]})
  .then(function(result){

    var formDeleteSensor = document.createElement("form");
    formDeleteSensor.setAttribute("id", "deleteSensor");
    formDeleteSensor.setAttribute("method", "post");
    formDeleteSensor.setAttribute("action", "/deleteSensor");

    var hiddenFieldSensorName = document.createElement("input");
    hiddenFieldSensorName.setAttribute("type", "hidden");
    hiddenFieldSensorName.setAttribute("name", "sensorName");
    hiddenFieldSensorName.setAttribute("value", sensors[idDelete].sensorName);

    var hiddenFieldPassphrase = document.createElement("input");
    hiddenFieldPassphrase.setAttribute("type", "hidden");
    hiddenFieldPassphrase.setAttribute("name", "passphrase");
    hiddenFieldPassphrase.setAttribute("value", result);

    formDeleteSensor.appendChild(hiddenFieldSensorName);
    formDeleteSensor.appendChild(hiddenFieldPassphrase);
    document.getElementById("formArea").appendChild(formDeleteSensor);
    sentForm("#deleteSensor");
    clearElement("formArea");
  });
}
```

5.4.7 Buy

Χρησιμοποιώντας τα event SensorAdded η ιστοσελίδα δημιουργεί τη λίστα με τους διαθέσιμους αισθητήρες. Ο χρήστης/αγοραστής μπορεί να επιλέξει τον αισθητήρα

που επιθυμεί από τη λίστα Available Sensors και πατώντας το κουμπί Buy with Ether ή Buy with Token καθορίζει αν θα πληρώσει με Ether ή με Token.

Στην περίπτωση που επιλέξει πληρωμή με Ether, η διαδικασία αρχίζει με την εκτέλεση της JavaScript συνάρτησης BuyWithEther, η οποία ελέγχει αν ο χρήστης διαθέτει ενεργό λογαριασμό, αν το αρχείο είναι ακόμα διαθέσιμο, αν τον έχει αγοράσει ήδη ο τρέχων λογαριασμός και αν διαθέτει το απαιτούμενο πόσο ώστε να πραγματοποιηθεί η αγορά. Διαφορετικά εμφανίζει το αντίστοιχο μήνυμα ανάλογα με το πρόβλημα που εντόπισε. Αν όλοι οι έλεγχοι περαστούν με επιτυχία, καλείται η συνάρτηση buySensor του Management Smart Contract με ορίσματα τα στοιχεία του αγοραστή, το ID και μια boolean μεταβλητή (false) που δηλώνει ότι δεν έχει γίνει χρήση του Token, η οποία αφού ελέγξει και αυτή με τη σειρά της ότι το ποσό που στάλθηκε είναι ίσο με την προκαθορισμένη τιμή, το μεταφέρει στο λογαριασμό του πωλητή/διαχειριστή, καθώς αυτός είναι υπεύθυνος για το διαμοιρασμό του στους υπόλοιπους ιδιοκτήτες, που πιθανόν να υπάρχουν, όταν ολοκληρωθεί η συναλλαγή. Επιπλέον προστίθενται τα στοιχεία της πώλησης στον πίνακα purchasesTable και γίνεται emit το event PurchaseInfo, που περιέχει τις πληροφορίες που αφορούν την συγκεκριμένη πώληση, όπως οι διευθύνσεις πωλητή, αγοραστή και τα αναγνωριστικά ID αισθητήρα αλλά και πώλησης.

Τα event αυτά χρησιμοποιούνται, με τη χρήση της συνάρτησης GetEventsPurchaseInfo, για να δημιουργηθούν δυναμικά οι λίστες My purchases και My sales, που θα παρουσιάσουν στον πωλητή και στον αγοραστή τα στοιχεία των αγοραπωλησιών που τους αφορούν. Τα events αυτά “σκανάρονται” από δύο συναρτήσεις, την purchasesInfo, η οποία δημιουργεί τη λίστα My Purchases, που παρουσιάζει στον αγοραστή τις ολοκληρωμένες και τις εκκρεμείς αγορές του, αλλά και την salesInfo, για τον πωλητή.

```
    async function buyWithEther(idBuy){
if(!userExists){
  alert("You must create a User account first!")
} else {
if(typeof sensorsDeleted[idBuy]!=="undefined"){
  alert("Not longer available!");
}else {
  if(admins[idBuy].adminAddress.localeCompare(currentUserAddress)==0){
```

```
      alert("You own this!");
    } else{
      var price = sensors[idBuy].price;
      if(typeof newPrices[idBuy]!=="undefined")
      price=newPrices[idBuy];

      var amountToSendEther = web3.utils.toWei(price, "ether");
      var balanceEther = await web3.eth.getBalance(currentUserAddress);

      if(balanceEther - amountToSendEther < 0) {
        alert("Account does not have enough balance!");
      } else {


MANAGEMENTCONTRACT.methods.buySensor.apply(MANAGEMENTCONTRACT,
          [idBuy, false, currentUserContractAddress, currentUserQuorumPublicKey])
          .send({from:currentUserAddress ,gas: gasGlobal, value:amountToSendEther});
          alert("You have bought sensor: " + idBuy);
        }
      }
    }
  }
}
```

Σαν εναλλακτική επιλογή είναι διαθέσιμη και η πληρωμή με τη χρήση Token. Αρχικά γίνονται κατάλληλοι έλεγχοι, όπως και στην πρώτη περίπτωση. Η διαδικασία είναι σχεδόν η ίδια με τη βασική διαφορά ότι η JavaScript συνάρτηση της ιστοσελίδας BuyWithToken καλεί τη συνάρτηση transfer του Token Smart Contract, ενώ στην προηγούμενη περίπτωση μεταφορά του Ether γινόταν από την buySensor του Smart Contract. Ωστόσο και σε αυτή την περίπτωση η συνάρτηση buySensor καλείται, εδώ με τιμή true στη boolean μεταβλητη useToken, με σκοπό την αποθήκευση της πώλησης στον αντίστοιχο πίνακα, αλλά και για να σταλεί το event purchaseInfo από το οποίο αντλεί η ιστοσελίδα τις πληροφορίες που χρειάζεται για να ενημερώνει τον πωλητή και τον αγοραστή για την πορεία των συναλλαγών τους.

```
      async function buyWithToken(idBuy){
```

```
 if(!userExists){
  alert("You must create a User account first!")
 } else {
 if(typeof sensorsDeleted[idBuy]!=="undefined"){
  alert("Not longer available!");
 }else {
  if(admins[idBuy].adminAddress.localeCompare(currentUserAddress)==0){
   alert("You own this!");
  } else{
   var price = sensors[idBuy].price;
   if(typeof newPrices[idBuy]!=="undefined")
   price=newPrices[idBuy];

   var amountToSendToken= web3.utils.toWei(price, "ether");
   var balanceNtuaToken;

   TOKENCONTRACT.methods.balanceof.apply(TOKENCONTRACT,
    [currentUserAddress]).call({from:currentUserAddress ,gas: gasGlobal})
    .then(function(result){
     balanceNtuaToken =result;

     if(balanceNtuaToken - amountToSendToken < 0){
      alert("Account does not have enough balance!");
     } else {
      TOKENCONTRACT.methods.transfer.apply(TOKENCONTRACT,
                                           [admins[idBuy].adminAddress,
amountToSendToken]).send({from:currentUserAddress ,gas: gasGlobal});


MANAGEMENTCONTRACT.methods.buySensor.apply(MANAGEMENTCONTRACT,
          [idBuy, true, currentUserContractAddress, currentUserQuorumPublicKey])
          .send({from:currentUserAddress ,gas: gasGlobal});
          alert("You have bought sensor: " + idBuy);
        }
       });
     }
```

```
    }
   }
  }
```

Η παραπάνω διαδικασία ήταν το πρώτο στάδιο που πρέπει να ολοκληρώσει ο αγοραστής ώστε να πραγματοποιήσει μία αγορά, ωστόσο για να ολοκληρωθεί η αγορά απαιτούνται στη συνέχεια και επιπλέον ενέργειες από τον πωλητή. Μετά από αυτό το στάδιο θα εμφανίζεται στην ιστοσελίδα του πωλητή στην λίστα My sales μία νέα καταχώρηση πώλησης η οποία θα αφορά στην προαναφερθείσα συναλλαγή. Σε αυτό το σημείο ο πωλητής θα έχει δύο επιλογές, είτε να ακυρώσει τη συναλλαγή πατώντας το κουμπί Reject και να επιστρέψει το ποσό στον αγοραστή, αν για κάποιο λόγο δεν επιθυμεί να την ολοκληρώσει, είτε μέσω του κουμπιού Complete Purchase να επιβεβαιώσει την ολοκλήρωση της συναλλαγής.

Αν επιλέξει να ολοκληρώσει τη πώληση, αρχικά η ιστοσελίδα καλεί, ανάλογα με τον τρόπο πληρωμής που έχει επιλεχθεί στο προηγούμενο βήμα, μία από τις συναρτήσεις completePurchaseMadeWithEther ή completePurchaseMadeWithToken οι οποίες μαζί με την αντίστοιχη συνάρτηση completePurchase του Smart Contract εκτελούν τις λειτουργίες που απαιτούνται για την ολοκλήρωση της συναλλαγής.

Το ποσό που είχε στείλει προηγουμένως ο αγοραστής στον διαχειριστή του αισθητήρα, μοιράζεται στους ιδιοκτήτες με βάση το ποσοστό που τους αντιστοιχεί. Αφού ανακτηθεί το passphrase από την getPassphrase του User Smart Contract του διαχειριστή, στέλνεται στο User Smart Contract του αγοραστή, ώστε να του δώσει πρόσβαση στα δεδομένα που αγόρασε. Παράλληλα ενημερώνεται το δίκτυο ότι η πώληση έχει ολοκληρωθεί μέσω του event purchaseCompleted και η πώληση στο αντίστοιχο πίνακα του smart contract ορίζεται ως completed.

Τέλος, προστίθενται τα δεδομένα για τη συγκεκριμένη πώληση στο Reputation Smart Contract με τη συνάρτηση addPurchase, ώστε να μπορεί στη συνέχεια τόσο ο αγοραστής όσο και ο πωλητής να αξιολογήσουν τη μεταξύ τους συναλλαγή. Η ιστοσελίδα όταν εντοπίσει το event PurchaseCompleted που δηλώνει ότι κάποια πώληση έχει ολοκληρωθεί, προσθέτει στην αντίστοιχη καταχώρηση στη λίστα My purchases, τα κουμπιά Get Data και Show Passphrase.

Ο χρήστης πατώντας το Show Passphrase μπορεί να δει τον passphrase του αισθητήρα που αγόρασε, σε περίπτωση που το χρειαστεί. Όταν επιλέξει το κουμπί Get

Data εκτελείται η JavaScript συνάρτηση getData, η οποία αφού αρχικά πάρει το passphrase απο το User Smart Contract καλώντας την getBoughtPassphrase, το χρησιμοποιεί ώστε με τη βοήθεια του Server να κατεβάσει το αρχείο τύπου json που περιέχει τα αποθηκευμένα δεδομένα.

```
      async function completePurchaseMadeWithEther(accept, purchase){
  var USERCONTRACTBUYER =
  new web3.eth.Contract(userContractAbi, purchase.buyerContractAddress,
  {from: currentUserAddress, gas:gasGlobal,
   privateFor:[purchase.buyerQuorumPublicKey]});

  var price = sensors[purchase.id].price;
  if(typeof newPrices[purchase.id]!=="undefined")
  price=newPrices[purchase.id];

  var amountToSendEther =web3.utils.toWei(price, "ether");
  var balanceEther = await web3.eth.getBalance(currentUserAddress);

  if(balanceEther - amountToSendEther < 0){
   alert("Account does not have enough balance.");
  } else {
   if(!accept){
    alert("Rejected Purchase");
   } else{

USERCONTRACT.methods.getPassphrase.apply(USERCONTRACT,[purchase.id])
      .call({from: currentUserAddress, gas:gasGlobal,
       privateFor:[currentUserQuorumPublicKey]})
      .then(function(result){

USERCONTRACTBUYER.methods.addBoughtPassphrase.apply(USERCONTRACT
BUYER,[purchase.id, result])
       .send({from: currentUserAddress, gas:gasGlobal,
        privateFor:[purchase.buyerQuorumPublicKey]})
       alert("Accepted and completed purchase");
      });
```

```
    }

MANAGEMENTCONTRACT.methods.completePurchase.apply(MANAGEMENTCON
TRACT, [purchase.purchaseId, purchase.id, purchase.useToken, accept])
      .send({from:currentUserAddress ,gas: gasGlobal, value:amountToSendEther});


REPUTATIONCONTRACT.methods.addPurchase.apply(REPUTATIONCONTRACT,
       [purchase.purchaseId, purchase.id, purchase.buyerAddress])
       .send({from: currentUserAddress, gas:gasGlobal})
   }
}
```

Όπως και στο πρώτο στάδιο, στο οποίο ο αγοραστής ξεκίνησε τη διαδικασία της πώλησης, έτσι και στο δεύτερο που ο πωλητής την ολοκληρώνει, υπάρχει η εναλλακτική συνάρτηση completePurchaseMadeWithToken που εκτελεί τις ίδιες λειτουργίες με την περιπτωση που χρησιμοποιήθηκε Ether, με κύρια διαφορά ότι η μεταφορά του ποσού δεν γίνεται στο Smart Contract, αλλά στον JavaScript κώδικα της ιστοσελίδας με κλήσεις της transfer του Token Smart Contract.

```
      async function completePurchaseMadeWithToken(accept, purchase){
      var USERCONTRACTBUYER = new web3.eth.Contract(userContractAbi,
purchase.buyerContractAddress,
                                {from: currentUserAddress, gas:gasGlobal,
privateFor:[purchase.buyerQuorumPublicKey]});

   var price = sensors[purchase.id].price;
   if(typeof newPrices[purchase.id]!=="undefined")
   price=newPrices[purchase.id];

   var amountToSendToken= web3.utils.toWei(price, "ether");
   var balanceNtuaToken;

   TOKENCONTRACT.methods.balanceof.apply(TOKENCONTRACT,
    [currentUserAddress]).call({from:currentUserAddress ,gas: gasGlobal})
    .then(function(result){
```

```
      balanceNtuaToken =result;

      if(balanceNtuaToken - amountToSendToken < 0){
        alert("Account does not have enough balance!");
      } else {
        if(!accept){
          alert("Rejected purchase");
          TOKENCONTRACT.methods.transfer.apply(TOKENCONTRACT,
                                                              [purchase.buyerAddress,
amountToSendToken]).send({from:currentUserAddress ,gas: gasGlobal});
          } else{

USERCONTRACT.methods.getPassphrase.apply(USERCONTRACT,[purchase.id])
                                        .call({from:  currentUserAddress,  gas:gasGlobal,
privateFor:[currentUserQuorumPublicKey]})
            .then(function(result){

USERCONTRACTBUYER.methods.addBoughtPassphrase.apply(USERCONTRACT
BUYER,[purchase.id, result])
                                       .send({from:  currentUserAddress,  gas:gasGlobal,
privateFor:[purchase.buyerQuorumPublicKey]})
              alert("Accepted and completed purchase");

              if(typeof ownerships[purchase.id]!== "undefined")
              ownerships[purchase.id].owners.forEach(function (each){
                var mod= parseInt(each.ownerPercentage, 10)/100;
                var amount = ""+parseInt(amountToSendToken*mod, 10);
                TOKENCONTRACT.methods.transfer.apply(TOKENCONTRACT,
                      [each.ownerAddress, amount]).send({from:currentUserAddress ,gas:
gasGlobal});
                });
              })
            }

MANAGEMENTCONTRACT.methods.completePurchase.apply(MANAGEMENTCON
TRACT,
```

```
            [purchase.purchaseId, purchase.id, purchase.useToken, accept])
            .send({from:currentUserAddress ,gas: gasGlobal});

REPUTATIONCONTRACT.methods.addPurchase.apply(REPUTATIONCONTRACT,
              [purchase.purchaseId, purchase.id, purchase.buyerAddress])
              .send({from: currentUserAddress, gas:gasGlobal})
            }
          });
        }
```

5.5 Βάση δεδομένων

Ένα βασικό χαρακτηριστικό των αποκεντρωμένων εφαρμογών (DApps) είναι ότι δεν ελέγχονται από μία κεντρική οντότητα, όπως για παράδειγμα μία κυβέρνηση ή μία εταιρεία. Στην ιδανική περίπτωση, όλα τα δεδομένα της εφαρμογής θα ήταν αποθηκευμένα αποκλειστικά στο Blockchain και δε θα χρησιμοποιούνταν βάση δεδομένων. Θεωρητικά μια τέτοια υλοποίηση θα ήταν εφικτή, καθώς τα δεδομένα θα μπορούσαν να συμπιεστούν και στη συνέχεια να σταλούν στο Blockchain, αλλά στην πράξη μια τέτοια επιλογή θα μείωνε σημαντικά τη χρηστικότητα και την αποτελεσματικότητα της εφαρμογής. Προφανώς ο αυξημένος όγκος δεδομένων στο Blockchain θα επηρέαζε αρνητικά τη διεκπεραιωτική της ικανότητα και θα δημιουργούσε αυξημένη χρονοκαθυστέρηση.

Κάθε φορά που κάποιος χρήστης επισκέπτεται την ιστοσελίδα θα πρέπει να είναι άμεσα διαθέσιμα τα στοιχεία των αισθητήρων προς πώληση, ενώ οι πωλητές θα πρέπει να μπορούν να διαχειριστούν εκείνους που τους ανήκουν και οι αγοραστές να έχουν πρόσβαση στα δεδομένα των αισθητήρων που έχουν αγοράσει. Όλη η πληροφορία που απαιτείται για να διαμορφωθεί η ιστοσελίδα, δεν υπάρχει αποθηκευμένη κάπου, αλλά μπορεί να αποκτηθεί από τα events. Ο φυλλομετρητής (browser) κάθε χρήστη σκανάρει ανα τακτά διαστήματα το Blockchain και βρίσκει σε κάθε νέο block τα events των έξυπνων συμβολαίων της εφαρμογής.

Μια εναλλακτική προσέγγιση θα ήταν η αποθήκευση της πληροφορίας που περιέχουν τα events στον εξυπηρετητή ο οποίος στη συνέχεια θα έστελνε τα δεδομένα που έχει συλλέξει κάθε φορά που κάποιος χρήστης συνδεόταν στην ιστοσελίδα. Όμως η επιλογή αυτή θα είχε ως αποτέλεσμα την αλλοίωση του αποκεντρωτικού χαρακτηρα της εφαρμογής καθώς ο σκοπός είναι να διατηρηθούν οι λειτουργίες που επιτελεί ο εξυπηρετητής στο ελάχιστο δυνατό, δηλαδή να παρέχει τα HTML αρχεία στον

φυλλομετρητή και να φροντίσει για την ορθή επικοινωνία με τη βάση δεδομένων. Επιπλέον θα αυξανόταν το υπολογιστικό βάρος στον εξυπηρετητή και επομένως θα υπήρχαν σημαντικές καθυστερήσεις, ενώ η απόδοση θα μειωνόταν σημαντικά.

Επομένως δημιουργήθηκε μια σχεσιακή βάση δεδομένων, για την υλοποίηση της οποίας χρησιμοποιήθηκε το σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDBMS) MySQL. Η βάση περιέχει τον πίνακα sensors στον οποίο είναι καταχωρημένα τα στοιχεία και τα δεδομένα όλων των διαθέσιμων αισθητήρων.

Στην συνέχεια παρουσιάζεται ο πινακας sensors και αναλύονται τα πεδία που περιλαμβάνει.

Όνομα	Τύπος	Περιγραφή
sensorName	VARCHAR (255)	Το αναγνωριστικό του αισθητήρα
price	INT	Η τιμή των δεδομένων του αισθητήρα
startTime	INT	Η χρονική στιγμή της πρώτης μέτρησης του αισθητήρα.
frequency	INT	Η συχνότητα με την οποία πραγματοποιεί μετρήσεις ο αισθητήρας (την ώρα).
latitude	INT	Το γεωγραφικό πλάτος της θέσης του αισθητήρα.
longitude	INT	Το γεωγραφικό μήκος της θέσης του IoT αισθητήρα.
admin	VARCHAR (255)	Η διεύθυνση του διαχειριστή/πωλητή του αισθητήρα.
passphrase	VARCHAR (255)	Η κρυφή φράση που απαιτείται για να δοθεί πρόσβαση στα δεδομένα.
data	JSON	Τα δεδομένα του αισθητήρα σε αρχείο JSON.

Πίνακας 5-1 Πεδία βάσης δεδομένων

Ο εξυπηρετητής κατά την πρώτη εκτέλεση της εφαρμογής δημιουργεί τον πίνακα sensors, στον οποίο στη συνέχεια θα καταχωρηθεί κάθε αισθητήρας, χρησιμοποιώντας τη JavaScript συνάρτηση createSensorsTable. Επίσης, κάθε φορά που ο εξυπηρετητής εκκινείται πρέπει να εκτελέσει τον JavaScript κωδικα που φροντίζει για τη σύνδεση του με τη βάση δεδομένων.

```
        var con = mysql.createConnection({
  host: "127.0.0.1",
  user: "padmin",
  password: "super-strong-password",
  database: "mydb"
});

function createSensorsTable(){
  var queryCreateTable = "CREATE TABLE Sensors (DBid INT AUTO_INCREMENT
PRIMARY KEY,"+
  " sensorName VARCHAR(255),"+
  " price INT ,"+
  " startTime INT ,"+
  " frequency INT ,"+
  " latitude INT,"+
  " longitude INT,"+
  " admin VARCHAR(255),"+
  " passphrase VARCHAR(255),"+
  " data JSON)";
  con.query(queryCreateTable, function (err, result) {
    if (err) throw err;
    console.log("Sensors Table created");
  });
}
```

Οι συναρτήσεις sentData και getData χρησιμοποιούνται για την αποστολή και παραλαβή των δεδομένων του αισθητήρα μεταξύ του εξυπηρετητή, που εκτελεί ρόλο μεσάζωντα και τη βάση. Η JavaScript συνάρτηση sentData εκτελείται όταν κάποιος

χρήστης προσθέσει ένα νέο αισθητήρα και η getData όταν, μετά την ολοκλήρωση της αγοράς, κάποιος χρήστης επιθυμεί να παραλάβει τα δεδομένα που αγόρασε.

```
        app.post('/sentData', urlencodedParser,function (req, res) {
    var sensorNameTemp = req.body.inputSensorName;
    var priceTemp = req.body.inputPrice;
    var startTimeTemp = req.body.inputStartTime;
    var frequencyTemp = req.body.inputFrequency;
    var latitudeTemp = req.body.inputLatitude;
    var longitudeTemp = req.body.inputLongitude;
    var adminTemp = req.body.inputAdmin;
    var passphraseTemp = req.body.inputPassphrase;
    var jsonTemp = req.body.inputJson;
    var jsonTempParsed = JSON.parse(jsonTemp);
    var msg;
                        var      mysqlSentData      =      "INSERT      INTO      Sensors
(sensorName,price,startTime,frequency"
    +",latitude,longitude,admin,passphrase,data) VALUES ('"
    +sensorNameTemp+"','"+priceTemp+"','"+startTimeTemp+"','"+frequencyTemp+"','"

+latitudeTemp+"','"+longitudeTemp+"','"+adminTemp+"','"+passphraseTemp+"','"+json
Temp+"')";

    con.query(mysqlSentData, function (err, result) {
      if (err) throw err;
                        msg    =    "Added    sensor    "+sensorNameTemp+"    with    DBid
"+JSON.stringify(result.insertId);
      res.status(201).json({success: msg});
    });
  });

  app.post('/getData', urlencodedParser,function (req, res) {
    var sensorNameTemp=req.body.sensorName;
    var passphraseTemp=req.body.passphrase;
    var msg;
        var  queryGetData  =  "SELECT  *  FROM  Sensors  WHERE  sensorName  =
```

```
'"+sensorNameTemp
   +"' AND passphrase ='"+passphraseTemp+"'";
   con.query(queryGetData, function (err, result) {
     if (err) throw err;
     msg = "Sent data of sensor "+sensorNameTemp;
     res.status(201).json({success: msg , sentData: result[0].data});
   });
 });
```

Ο πίνακας sensors μεταβάλλεται όταν ο χρήστης διαγράψει ή επεξεργαστεί τα στοιχεία κάποιου αισθητήρα και πιο συγκεκριμένα αλλάξει την τιμή ή το passphrase του (οι υπόλοιπες αλλαγές που μπορεί να πραγματοποιήσει δεν επηρεάζουν τη βάση δεδομένων). Η αλλαγή του passphrase γίνεται με την JavaScript συνάρτηση changePassphrase, ενώ η αλλαγή της τιμής με την updatePrice. Για την διαγραφή μιας καταχώρησης για κάποιον αισθητήρα χρησιμοποιείται η deleteSensor. Μετά από την κάθε επιτυχή αλληλεπίδραση με τη βάση, ο εξυπηρετητής προωθεί το αντίστοιχο ενημερωτικό μήνυμα στο φυλλομετρητή του χρήστη, οποίος με τη σειρά του το εμφανίζει στον χρήστη.

```
        app.post('/updatePrice', urlencodedParser,function (req, res) {
   var sensorNameTemp = req.body.sensorName;
   var passphraseTemp = req.body.passphrase;
   var newPriceTemp = req.body.newPrice;
   var msg;
   var queryUpdatePrice = "UPDATE Sensors SET price = '"+newPriceTemp
         +"' WHERE sensorName = '"+sensorNameTemp+"' AND passphrase
='"+passphraseTemp+"'";

   con.query(queryUpdatePrice, function (err, result) {
     if (err) throw err;
     msg = "Updated price of "+sensorNameTemp;
     res.status(201).json({success: msg});
   });
 });

 app.post('/updatePassphrase', urlencodedParser,function (req, res) {
```

```
  var sensorNameTemp = req.body.sensorName;
  var passphraseTemp = req.body.passphrase;
  var newPassphraseTemp = req.body.newPassphrase;
  var msg;
        var queryUpdatePassphrase = "UPDATE Sensors SET passphrase =
'"+newPassphraseTemp
        +"' WHERE sensorName = '"+sensorNameTemp+"' AND passphrase
='"+passphraseTemp+"'";

  con.query(queryUpdatePassphrase, function (err, result) {
    if (err) throw err;
    msg = "Updated passphrase of "+sensorNameTemp;
    res.status(201).json({success: msg});
  });
});

app.post('/deleteSensor', urlencodedParser,function (req, res) {
  var sensorNameTemp = req.body.sensorName;
  var passphraseTemp = req.body.passphrase;
  var msg;
      var queryDeleteSensor = "DELETE FROM Sensors WHERE sensorName =
'"+sensorNameTemp
  +"' AND passphrase ='"+passphraseTemp+"'";

  con.query(queryDeleteSensor, function (err, result) {
    if (err) throw err;
    msg = "Deleted sensor "+sensorNameTemp;
    res.status(201).json({success: msg});
  });
});
```

6 Επίδειξη λειτουργικότητας εφαρμογής

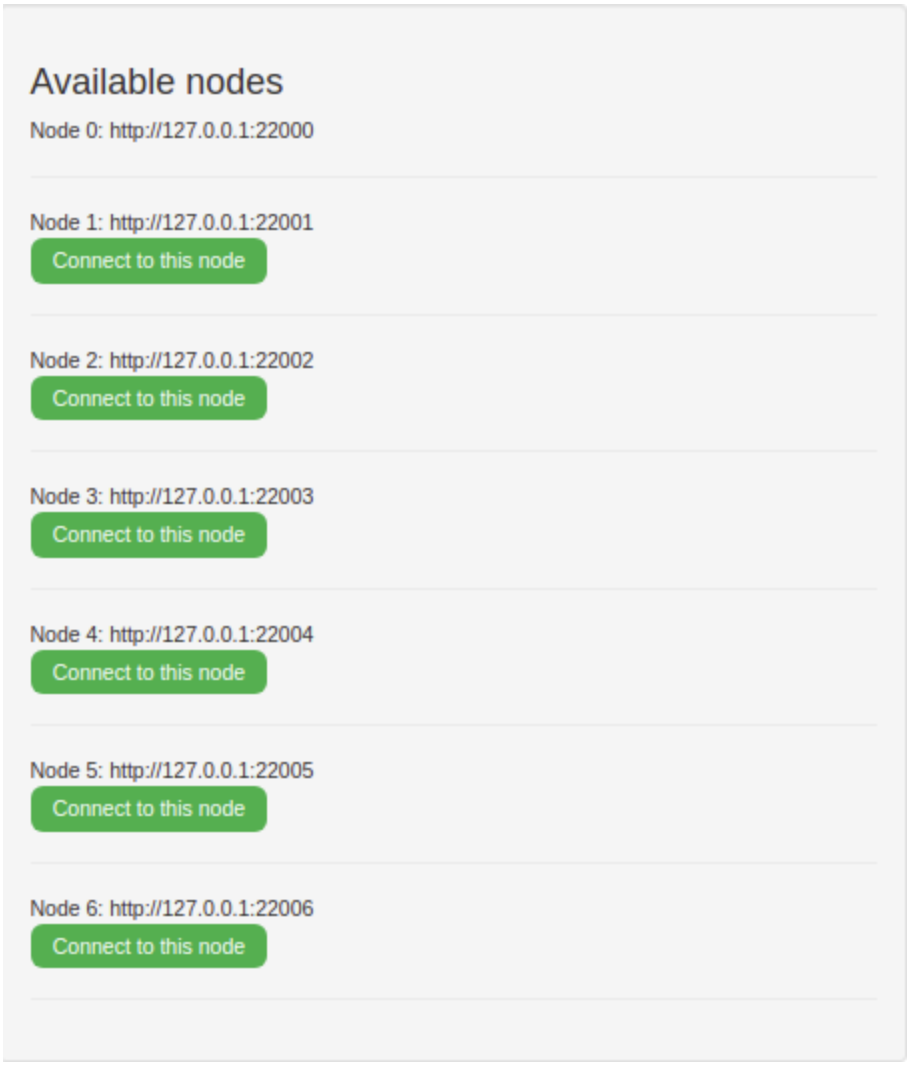
Στο κεφάλαιο αυτό θα γίνει παρουσίαση της εφαρμογής και του τρόπου λειτουργίας της. Έχοντας προηγουμένως αναλύσει την υλοποίηση της εφαρμογής, στη συνέχεια θα παρουσιαστεί η εφαρμογή από την οπτική του χρήστη.

Δεδομένου ότι η εφαρμογή βρίσκεται σε λειτουργία, ο χρήστης μεταβαίνοντας στη διεύθυνση **http://localhost:8080** μπορεί να αποκτήσει πρόσβαση στη κεντρική σελίδα της εφαρμογής στην οποία παρουσιάζονται όλοι οι διαθέσιμοι αισθητήρες.

Η εφαρμογή διαθέτει δύο βασικές σελίδες, μία για τους αγοραστές και μία για τους πωλητές, οι οποίες είναι διαθεσιμες από τις αντίστοιχες επιλόγές Buy και Sell που βρίσκονται στο κεντρικό menu και συμπεριλαμβάνουν όλες τις λειτουργίες που τους αφορούν. Επίσης είναι διαθέσιμη μια βοηθητική σελίδα Blockchain Info, η οποία περιέχει πληροφορίες σχετικά με τα υπόλοιπα των δοκιμαστικών λογαριασμών της εφαρμογής, τοσο σε Ether όσο και σε NTUA Token, αλλά και όλες τις συναλλαγές του Blockchain που αφορούν στην εφαρμογή.

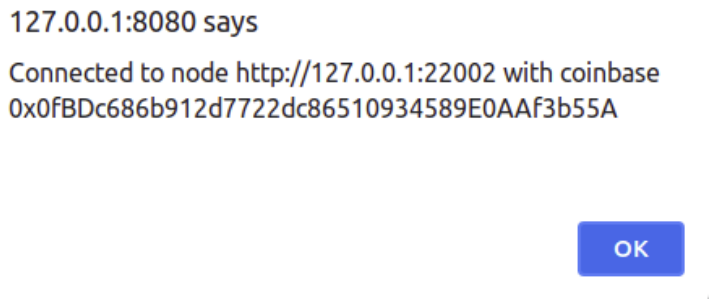
Η εφαρμογή χρησιμοποιεί ένα δοκιμαστικό τοπικό Quorum Blockchain και επομένως όταν ο χρήστης χρησιμοποιεί την ιστοσελίδα της εφαρμογής στο φυλλομετρητή του, αυτή συνδέεται με έναν από τους δοκιμαστικούς κόμβους που έχουν στηθεί τοπικά.

Με σκοπό την διευκόλυνση του χρήστη σε αυτή τη δοκιμαστική εκδοχή της εφαρμογής έχει δημιουργηθεί ένας πίνακας που αναφέρει όλους τους διαθέσιμους κόμβους του δοκιμαστικού Blockchain μαζί με ένα κουμπί μέσω του οποίου ο χρήστης μπορεί να συνδεθεί μέσω RPC (για παράδειγμα στο **http://localhost:22000** για τον πρώτο κομβο) με οποιονδήποτε δοκιμαστικό κόμβο. Προφανώς αν δοθούν τα στοιχεία ενός επιπλέον κόμβου η εφαρμογή μπορεί εύκολα να συνδεθεί και με αυτόν.



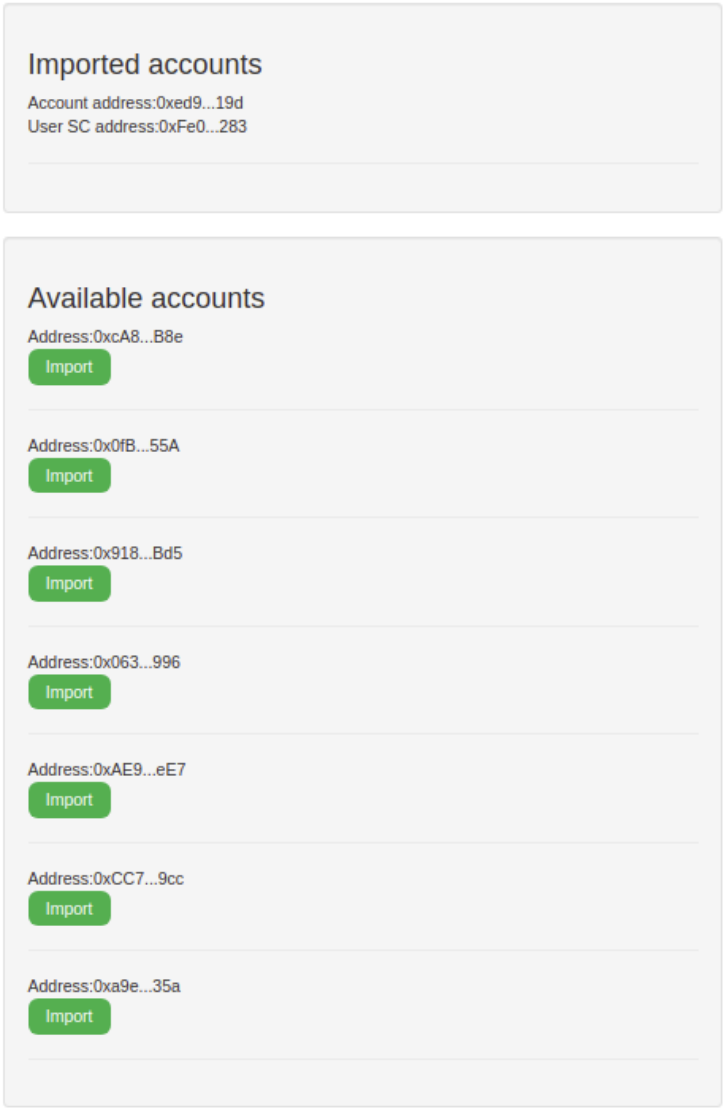
Εικόνα 6-1 Σύνδεση σε κόμβο

Αφού γίνει η συνδεση με τον κόμβο, θα εμφανιστεί ένα ενημερωτικό μήνυμα που θα αναφέρει τη διεύθυνση του κόμβου, όπως και το coinbase address του και στη συνέχεια ο χρήστης θα συνδέσει τον Quorum λογαριασμό του με την εφαρμογή.



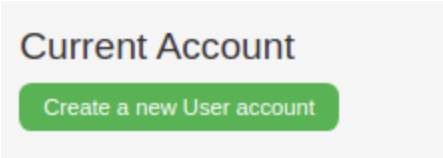
Εικόνα 6-2 Μήνυμα επιβεβαίωσης σύνδεσης σε κόμβο

Ο χρήστης έχει τη δυνατότητα να εισάγει τα στοιχεία του λογαριασμού του, ώστε να μπορεί να τον χρησιμοποιήσει για τις ανάγκες της εφαρμογής. Για να απλοποιηθεί η διαδικασία ανάπτυξης, αλλά και οι δοκιμές στη συνέχεια, έγιναν διαθέσιμοι κάποιοι δοκιμαστικοί λογαριασμοί. Ο χρήστης μπορεί να τους κάνει import κάθε φορά που συνδέεται σε κάποιο Quorum κόμβο, ενώ υπάρχει η δυνατότητα εναλλαγής αν έχει επιλέξει περισσότερους από έναν λογαριασμούς, επιλέγοντας τα αντίστοιχα κουμπιά.



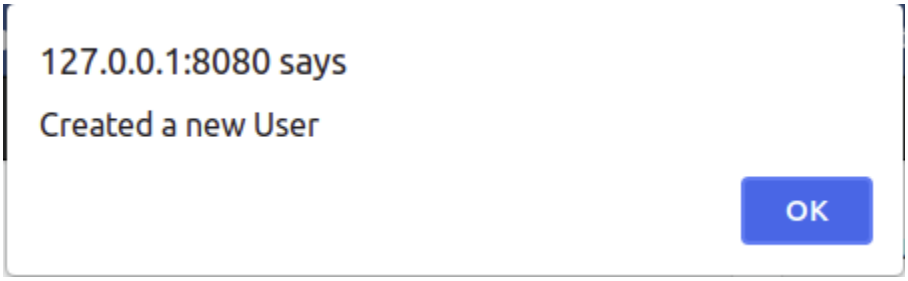
Εικόνα 6-3 Σύνδεση λογαριασμού

‘Επειτα πρέπει να δημιουργήσει ένα νέο User έξυπνο συμβόλαιο το οποίο λειτουργεί ως λογαριασμός χρήστη για τις ανάγκες της εφαρμογής. Η ενέργεια αυτή μπορεί να πραγματοποιηθεί εύκολα μέσω του κουμπιού Create a new User Account που βρίσκεται στο πεδίο Current Account της ιστοσελιδας.

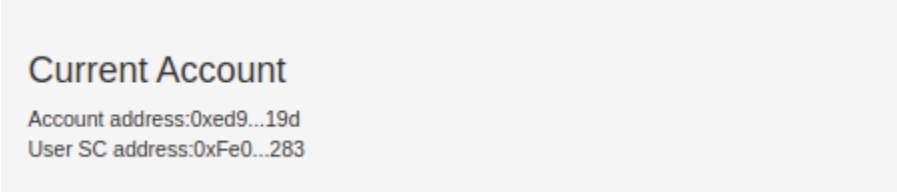


Εικόνα 6-4 Δημιουργία λογαριασμού

Στη συνέχεια ο χρήστης ενημερώνεται με το κατάλληλο μήνυμα, αν η δημιουργία του νέου έξυπνου συμβολαίου ολοκληρώθηκε με επιτυχία και μπορεί να δει τη διεύθυνση του νέου συμβολαίου μαζί με τη διευθυνση του Quorum λογαριασμού του στο πεδίο Current Account, που ανανεώθηκε αυτόματα. Να σημειωθεί ότι αν ο χρήστης έχει κάνει αυτή τη διαδικασία ήδη μια φορά και είναι συνδεδεμένος στον ίδιο κόμβο το σύστημα το αναγνωρίζει και εμφανίζει τα στοιχεία, χωρίς κάποια ενέργεια από το χρήστη.



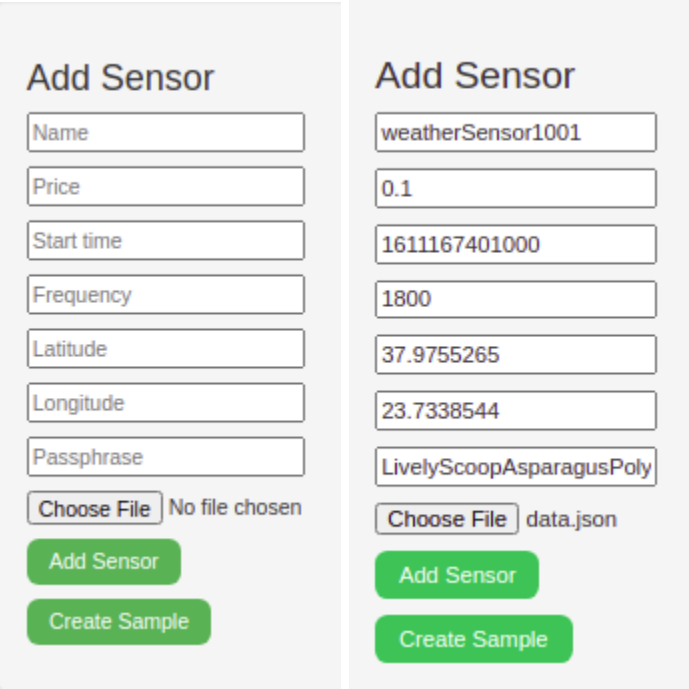
Εικόνα 6-5 Μήνυμα επιβεβαίωσης δημιουργίας λογαριασμού



Εικόνα 6-6 Τρέχων χρήστης

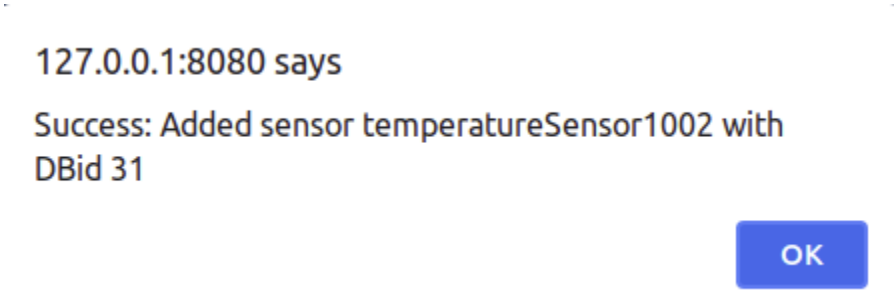
6.1 Ιστοσελίδα Sell

Η ιστοσελίδα διαθέτει τη φόρμα Add Sensors μέσα από την οποία ένας χρήστης που έχει στη κατοχή του ή διαχειρίζεται έναν αισθητήρα, μπορεί να τον καταχωρήσει στο Blockchain. Αναλυτικότερα αφού ο χρήστης καταχωρήσει τα στοιχεία του αισθητήρα στα πεδία της φόρμας και επιλέξει το αρχείο που περιέχονται τα δεδομένα μέσω του κουμπιού Choose File, καταχωρεί τον αισθητήρα στο έξυπνο συμβόλαιο Management και το passphrase στο έξυπνο συμβόλαιο User που του αντιστοιχεί, ενώ το αρχείο με τα δεδομένα ανεβαίνει στη βάση δεδομένων.



Εικόνα 6-7 Προσθήκη αισθητήρα

Όταν η διαδικασία αυτή ολοκληρωθεί με επιτυχία ο χρήστης λαμβάνει ένα ενημερωτικό μήνυμα που αναφέρει ότι ο αισθητήρας προστέθηκε και τα δεδομένα με του ανέβηκαν στη βάση δεδομένων.

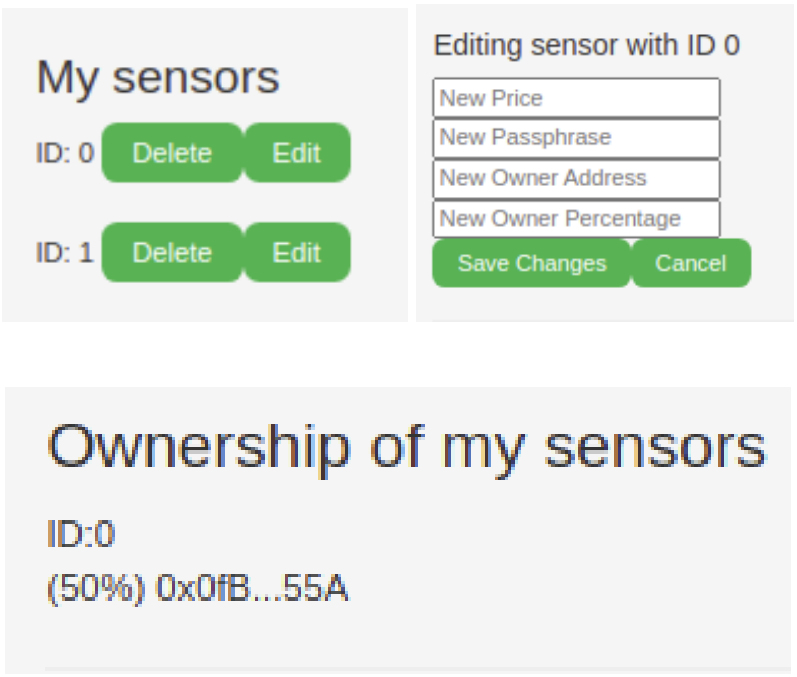


Εικόνα 6-8 Μήνυμα επιβεβαίωσης προσθήκης αισθητήρα

Οι καταχωρημένοι αισθητήρες εμφανίζονται στο πεδίο MySensors στην σελίδα του πωλητή, δίνοντας του τη δυνατότητα μεσω τον 2 κουμπιών Delete και Edit, να διαγράψει ή να τροποποιήσει τα δεδομένα του αισθητήρα αντίστοιχα.

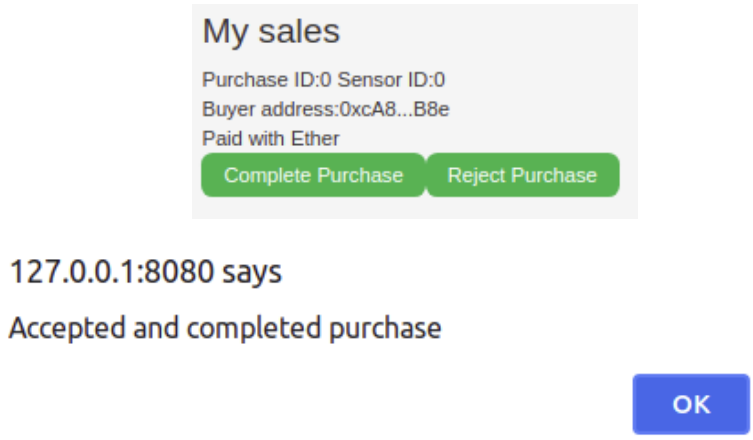
Διαγράφοντας τον αισθητήρα ενημερώνεται η καταχώρηση στο έξυπνο συμβόλαιο και κατ' επέκταση και στην ιστοσελίδα των αγοραστών ότι ο αισθητήρας έχει διαγραφεί και δεν είναι πλέον διαθέσιμος.

Η επιλογή Edit δημιουργεί δυναμικά μία νέα φόρμα που αφορά μόνο στον συγκεκριμένο αισθητήρα και δίνει τη δυνατότητα να τροποποιηθεί η τιμή, το Passphrase ή ιδιοκτησία του. Όπως έχει ήδη αναφερθεί, μπορεί είτε να προστεθεί ένας νέος ιδιοκτήτης είτε να αλλάξει το ποσοστό ενός από τους ήδη υπάρχοντες. Μετά από μία αλλαγή στα δικαιώματα ιδιοκτησίας ο χρήστης μπορεί να δει τα ποσοστά που έχει παραχωρήσει από τους αισθητήρες που διαχειρίζεται στο πεδίο Ownership of my sensors.



Εικόνα 6-9 Επεξεργασία αισθητήρα

Κάθε χρήστης μπορεί να δει τις πωλήσεις των έξυπνων αισθητήρων που διαχειρίζεται στην ενότητα My sales. Εκεί αναφέρεται το αναγνωριστικό κάθε αισθητήρα (ID), το αναγνωριστικό της συναλλαγής (Purchase ID), η διεύθυνση του αγοραστή και ο τρόπος πληρωμής. Ο πωλητής/διαχειριστής έχει τη δυνατότητα είτε να ακυρώσει τη πώληση μέσω του κουμπιού Reject Purchase, αν για οποιοδήποτε λόγο το επιθυμεί, είτε να την ολοκληρώσει επιλέγοντας το κουμπί Complete Purchase. Αν η συναλλαγή ακυρωθεί, το ποσό επιστρέφεται στον αγοραστή, ενώ αν ολοκληρωθεί, το έξυπνο συμβόλαιο φροντίζει να μοιραστεί η αμοιβή στους ιδιοκτήτες ανάλογα με το ποσοστό του καθενός. Αν ο χρήστης επιλέξει να αποδεχτεί τη συναλλαγή, τότε ενημερώνεται με το αντίστοιχο μήνυμα και στην καταχώρηση που αφορά στη συγκεκριμένη πώληση εμφανίζεται η δυνατότητα αξιολόγησης του αγοραστή, με κριτήριο τη μεταξύ τους συναλλαγή. Υπάρχει η επιλογή θετικής ή αρνητικής αξιολόγησης, ενώ μετά την επιλογή ενός από τα δύο κουμπιά, δεν μπορεί να αλλάξει η επιλογή ή να αξιολογηθεί ξανά η ίδια συναλλαγή και τα κουμπιά αντικαθίστανται από το αντίστοιχο ενημερωτικό μήνυμα.



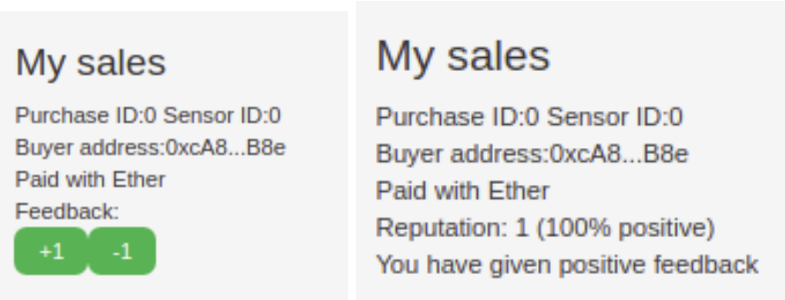
Εικόνα 6-10 Επιβεβαίωση πώλησης

Η εφαρμογή δίνει τη δυνατότητα Review από τον αγοσταστή. Μετά την ολοκλήρωση μιας πώλησης, η ιστοσελίδα, μέσω του αντίστοιχου event, ενημερώνεται και τροποποιεί δυναμικά την καταχώρηση που αφορά τη συγκεκριμένη πώληση στο πεδίο My Purchases. Πλέον, εμφανίζει στον αγοραστή την επιλογή να αφήσει review (θετικό ή αρνητικό).

Αφού επιλέξει το ανάλογο αποτέλεσμα, μέσω της συνάρτησης GiveReview, η οποία δέχεται ως ορίσματα το ID που αντιστοιχεί στην πώληση και την αξιολόγηση (ως +1 ή -1), οι πληροφορίες μεταφέρονται στο smart contract. Η συνάρτηση GiveReview, αφού ελέγξει ότι τα στοιχεία που αφορούν την πώληση είναι ορθά και δεν έχει γίνει ήδη αξιολόγηση, ώστε να αποκλειστούν οι κακόβουλες προσπάθειες, ανανεώνει τη βαθμολογία τόσο για τον συγκεκριμένο αισθητήρα, τόσο και για τον πωλητή. Επιπλέον, στέλνει το event το οποίο ενημερώνει το δίκτυο για την προσθήκη της αξιολόγησης.

Η ιστοσελίδα, μέσω των events, ανανεώνει τη βαθμολογία των αισθητήρων και των πωλητών. Τα δεδομένα αυτά, όταν είναι διαθέσιμα παρουσιάζονται σε εμφανές σημείο στην κάθε καταχώρηση, ώστε να μπορεί ο αγοραστής να τα λάβει υπόψιν.

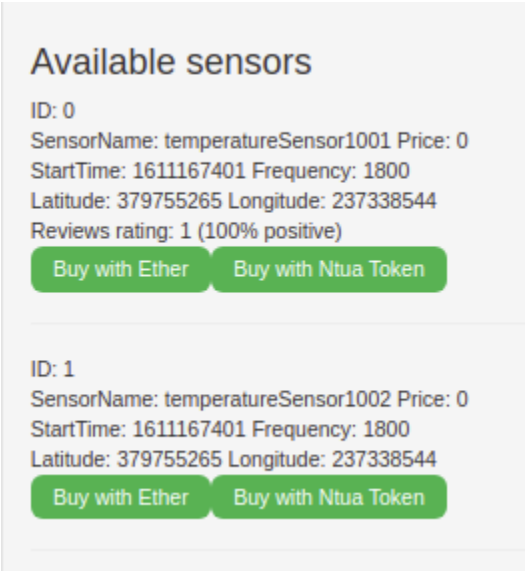
Ο πωλητής, με ανάλογο τρόπο, μπορεί να δώσει θετικό ή αρνητικό feedback, για κάθε αγοραπωλησία. Μέσω της συνάρτησης GiveFeedback το smart contract, διαμορφώνει το reputation του αγοραστή, με βαση το feedback του πωλητή. Με αυτό τον τρόπο, ο πωλητής μπορεί να μην ολοκληρώσει μία πώληση, αν ο αγοραστής δεν έχει καλό reputation.



Εικόνα 6-11 Τρέχουσες πωλήσεις

6.2 Ιστοσελίδα Buy

Ο χρήστης μέσω της σελίδας Buy μπορεί να δει όλους τους διαθέσιμους αισθητήρες και επιλέξει αυτόν που επιθυμεί να αγοράσει είτε με βάση το όνομα του αισθητήρα είτε τα υπόλοιπα στοιχεία της καταχώρησης και στη συνέχεια να πληρώσει με Ether ή NTUA Token.



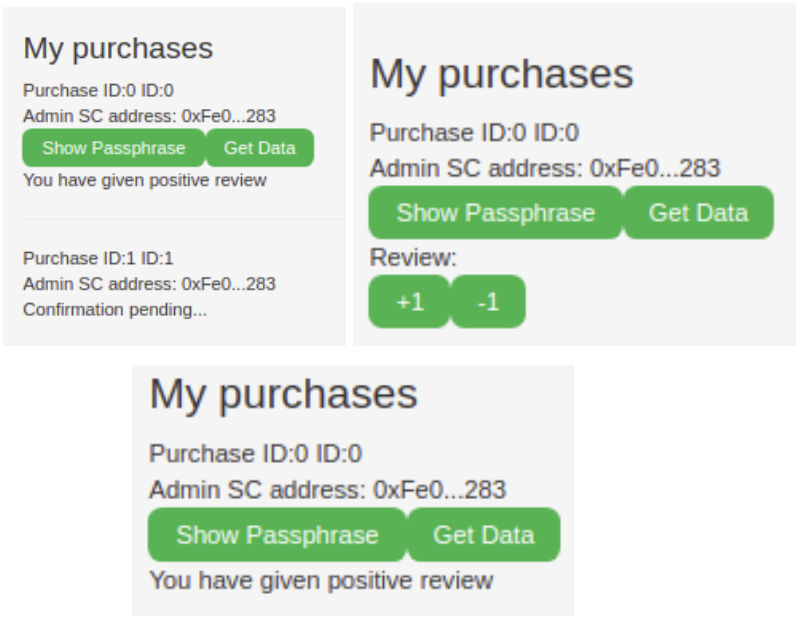
Εικόνα 6-12 Διαθέσιμοι αισθητήρες

Αφού ο χρήστης επιλέξει το αντίστοιχο κουμπί ώστε να πληρώσει με τον τρόπο που επιθυμεί και ελεγχθεί ότι το υπόλοιπό του επαρκεί για την αγορά, η σελίδα εμφανίζει ένα ενημερωτικό μήνυμα επιβεβαίωσης.

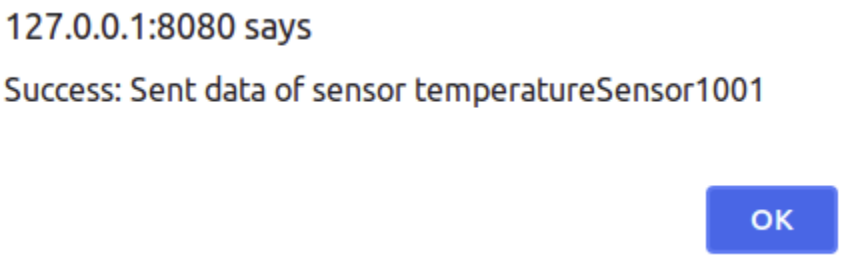
Εικόνα 6-13 Μήνυμα επιβεβαίωσης αγοράς

Στην ενότητα MyPurchases ο αγοραστής μπορεί να δει τις λεπτομέρειες που αφορούν στις αγορές του. Αρχικά κάθε αγορά θεωρείται εκκρεμής μέχρι να επιβεβαιωθεί από τον πωλητή που μπορεί είτε να την αποδεχτεί/ολοκληρώσει είτε να την απορρίψει. Επομένως η καταχώρηση που αφορά στη συγκεκριμένη αγορά αρχικά αναφέρει το μήνυμα “Confirmation Pending”, κάτω από τα βασικά στοιχεία της που είναι ο αναγνωριστικός αριθμός του αισθητήρα (ID), ο αναγνωριστικός αριθμός της αγοράς (Purchase ID) και η διεύθυνση του έξυπνου συμβολαίου User του πωλητή. Όπως φαίνεται στην εικόνα που ακολουθεί, όταν ο πωλητής ολοκληρώσει τη συναλλαγή, η καταχώρηση που βλέπει ο αγοραστής για την συγκεκριμένη αγορά ανανεώνεται αυτόματα και πλέον εμφανίζεται το κουμπί Get Data με το οποίο μπορούν να ληφθούν τα δεδομένα που έχουν αγοραστεί. Μετά το πάτημα του κουμπιού GetData ο χρήστης λαμβάνει το json αρχείο με τα δεδομένα του αισθητήρα και στέλνεται το ανάλογο ενημερωτικό μήνυμα. Ταυτόχρονα εμφανίζεται στον χρήστη και η επιλογή να αξιολογήσει την αγορά του θετικά ή αρνητικά μέσω του ανάλογου κουμπιού. Να σημειωθεί ότι μπορεί να κάνει την παραπάνω ενέργεια μόνο μια φορά, ενώ στη συνέχεια τα κουμπιά αυτά αντικαθίστανται από ένα μήνυμα που αναφέρει ότι έχει γίνει θετική ή αρνητική αξιολόγηση. Επιπλέον η αξιολόγηση του χρήστη επηρεάζει τη

βαθμολογία που εμφανίζεται κάτω από κάθε αισθητήρα στην ενότητα Available Sensors, ώστε να μπορούν οι μελλοντικοί αγοραστές να την λάβουν υπόψιν.



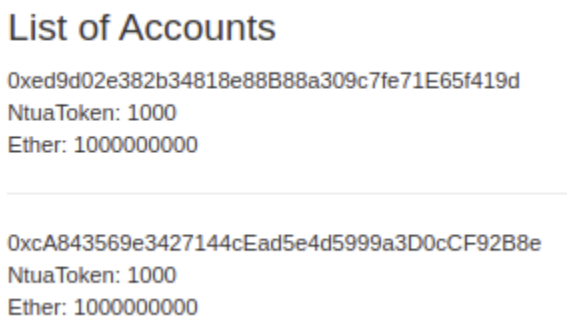
Εικόνα 6-14 Ολοκληρωμένες αγορές



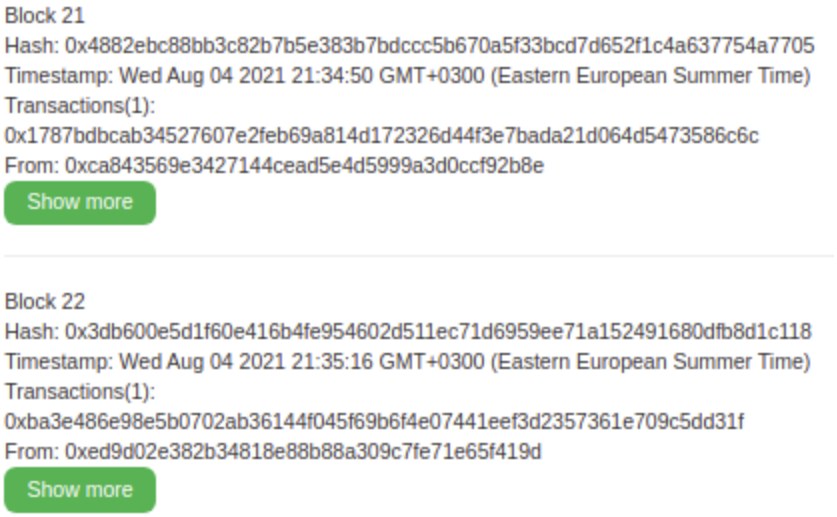
Εικόνα 6-15 Επιτυχημένη λήψη δεδομένων

6.3 Ιστοσελίδα Blockchain Info

Η ιστοσελίδας Blockchain Info δημιουργήθηκε ώστε να μπορούμε εύκολα να ελέγξουμε το υπόλοιπο κάθε λογαριασμού σε Ether και NTUA Token, ενώ παράλληλα παρουσιάζονται όλες οι συναλλαγές του Blockchain που αφορούν στην εφαρμογή. Η συγκεκριμένη σελίδα έχει βοηθητικό ρόλο και συνειδητά επιλέχθηκε να έχει περιορισμένες λειτουργίες. Για καθεμία από τις συναλλαγές μπορούμε να δούμε το block του Blockchain στο οποίο ανήκει, τη χρονική στιγμή που προστέθηκε στο Blockchain, την τιμή hash, αλλά και τη διεύθυνση του λογαριασμού από τον οποίο προέρχεται η συναλλαγή. Υπάρχει η δυνατότητα εμφάνισης όλων των διαθέσιμων πληροφοριών μέσω του κουμπιού Show more.



Εικόνα 6-16 Λογαριασμοί



Εικόνα 6-17 Καταχωρημένα blocks

7 Επίλογος

7.1 Σύνοψη και συμπεράσματα

Κατά την ανάπτυξη της παρούσας διπλωματικής εργασίας, εξετάστηκαν οι τεχνολογίες αποκέντρωσης και πιο συγκεκριμένα μια από σημαντικότερες εξ αυτών, το blockchain. Το Ethereum blockchain, αποτελώντας ένα από τα πιο δημοφιλή και διαδεδομένα blockchains, μελετήθηκε εκτενώς καθώς έχει αποδεδειγμένα μεγάλη χρησιμότητα και αρκετές εφαρμογές σε διάφορους τομείς. Το οικοσύστημα του Ethereum παρέχει τη δυνατότητα ανάπτυξης έξυπνων συμβολαίων και κατ' επέκταση αποκεντρωμένων εφαρμογών, ξεχωρίζοντας από άλλα blockchains με αντίστοιχα χαρακτηριστικά. Χωρίς αμφιβολία έχει αποτελέσει ένα πολύ σημαντικό παράγοντα στην ταχύτατη ανάπτυξη των τεχνολογιών του αποκεντρωμένου ιστού αλλά και της συνεχώς αυξανόμενης αποδοχής των blockchains ως μία κυρίαρχη τεχνολογία για το μέλλον του παγκόσμιου ιστού.

Παρά τα οφέλη των σύγχρονων πλατφορμών blockchain και την εξάπλωση έξυπνων συμβολαίων, τα περισσότερα υπάρχοντα συστήματα υστερούν σε ζητήματα ιδιωτικότητας, επεκτασιμότητας και διαλειτουργικότητας μεταξύ ετερογενών συσκευών ή υποδομών. Επιπλέον, συχνά παρατηρείται η απουσία πραγματικά αποκεντρωμένης διαχείρισης της αξιοπιστίας χρηστών και δεδομένων, ενώ ορισμένα δεν παρέχουν ισχυρούς μηχανισμούς για την αντιμετώπιση κακόβουλων ενεργειών ή λαθών αξιολόγησης. Τέλος, η επικέντρωση στην εμπορική ετοιμότητα παραμένει περιορισμένη, με πολλά συστήματα να παραμένουν σε πειραματικό επίπεδο.

Η εφαρμογή που αναπτύχθηκε αντιμετωπίζει αποτελεσματικά ζητήματα, τα οποία μεταξύ άλλων αφορούν την ιδιωτικότητα, την επεκτασιμότητα και τη διαλειτουργικότητα. Αξιοποιώντας την εγγενή ασφάλεια, τη διαφάνεια και την αποκέντρωση της τεχνολογίας blockchain, το σύστημα επέδειξε σημαντικές βελτιώσεις αυτούς τους τομείς. Σε συνδυασμό με την αμετάβλητη φύση του blockchain έχει ενισχύσει την προστασία των δεδομένων από μη εξουσιοδοτημένη πρόσβαση και παραποίηση. Η ικανότητα του συστήματος να διαχειρίζεται τον συνεχώς αυξανόμενο όγκο δεδομένων αισθητήρων IoT, εξασφαλίζει αποτελεσματική και έγκαιρη επεξεργασία δεδομένων. Εχει την δυνατότητα να πετύχει απρόσκοπτη ενσωμάτωση με ποικίλες υποδομές IoT, διευκολύνοντας την ανταλλαγή δεδομένων και την επικοινωνία σε διαφορετικές πλατφόρμες. Ενώ η διπλωματική αυτή παρουσιάζει στοιχεία τα οποία προσφέρουν λύσεις σε υπάρχοντα προβλήματα, υπάρχουν ακόμη δυνατότητες για περαιτέρω διερεύνηση και βελτίωση.

Προσφέροντας μια αποκεντρωμένη, ασφαλή και κλιμακούμενη πλατφόρμα διαμοιρασμού και αγοραπωλησίας δεδομένων αισθητήρων, ενσωματώνοντας εξελιγμένο trust & reputation system, η εφαρμογή που υλοποιήθηκε έρχεται να καλύψει αυτά τα ελλείμματα. Αυτό επιτρέπει τόσο τη διαφανή αξιολόγηση των χρηστών όσο και την αντικειμενική εκτίμηση της ποιότητας των ίδιων των δεδομένων, ενισχύοντας την εμπιστοσύνη και την αξιοπιστία σε πολυσύνθετα περιβάλλοντα, όπως τα δίκτυα IoT. Η σημασία της υιοθέτησης τέτοιων μηχανισμών είναι ιδιαίτερα κρίσιμη, αφού επιτρέπει όχι μόνο ασφαλείς συναλλαγές αλλά και τη δυνατότητα ενίσχυσης της συνεργασίας μεταξύ οργανισμών και χρηστών με αντικρουόμενα συμφέροντα, προωθώντας έτσι το οικοσύστημα των έξυπνων υποδομών σε πραγματικό επιχειρησιακό επίπεδο.

Όπως έχει αναλυθεί σε προηγούμενο κεφάλαιο, το Ethereum, παρά το γεγονός ότι τα χαρακτηριστικά του το καθιστούν πολύ δημοφιλές για την ανάπτυξη έξυπνων συμβολαίων, παρουσιάζει ορισμένες αδυναμίες σε θέματα ιδιωτικότητας και ασφάλειας. Το Quorum blockchain, βασισμένο στο Ethereum, καλύπτει αρκετά από τα χαρακτηριστικά που απουσιάζουν από το Ethereum, κάνοντάς το μία πολύ καλή επιλογή σε περιπτώσεις που η ιδιωτικότητα και η ασφάλεια αποτελούν προτεραιότητες.

Σκοπός της διπλωματικής εργασίας ήταν η εξέταση του Quorum blockchain και η χρήση του για τη δημιουργία μίας αποκεντρωμένης εφαρμογής (DApp – Decentralised Application), η οποία θα προσφέρει δυνατότητες αγοραπωλησίας δεδομενων έξυπνων αισθητήρων σε συνδυασμό με ένα σύστημα αξιολόγησης χρηστών και δεδομένων (Trust and Reputation System). Αρχικά εξετάστηκαν τα χαρακτηριστικά του Quorum και διαπιστώθηκε ότι ικανοποιούν τις ανάγκες της εφαρμογής στον τομέα της ασφάλειας. Το επόμενο βήμα αποτέλεσε η δημιουργία έξυπνων συμβολαίων, μέσω των οποίων κατέστη εφικτή η διασύνδεση της πλατφόρμας αγοραπωλησίας των δεδομένων των έξυπνων αισθητήρων, με το Quorum Blockchain.

Η εφαρμογή που δημιουργήθηκε είναι ένα παράδειγμα παράλληλης αξιοποίησης των τεχνολογιών του Blockchain και του IoT, σε συνδυασμό με ένα σύστημα Trust and Reputation. Παρόλο που έχουν υλοποιηθεί λειτουργίες που παρουσιάζουν μεγάλο ενδιαφέρον και είναι χρήσιμες για την ανάπτυξη μελλοντικών εφαρμογών, οι διάφορες ελλείψεις και περιορισμοί που παρουσίασε η χρήση των ανωτέρω τεχνολογιών, καθιστούν την εφαρμογή μία πρώιμη μη-εμπορική εκδοχή της αρχικής ιδέας, καθώς απαιτούνται αρκετά ακόμα βήματα για να αποτελέσει μια έκδοση έτοιμη για εμπορική χρήση. Η παρούσα εργασία δείχνει στην πράξη πως οι νέες αρχιτεκτονικές μπορούν να

καλύψουν υπάρχοντα τεχνολογικά και λειτουργικά κενά, ενώ η χρηστικότητά τους είναι άμεσα επεκτάσιμη σε μελλοντικές εξελιγμένες υποδομές.

7.2 Μελλοντικές επεκτάσεις

Είναι γεγονός ότι η εφαρμογή έχει βασιστεί πάνω σε νέες σχετικά τεχνολογίες, οι οποίες έχουν ταχύτατη εξέλιξη μέχρι τώρα. Όπως όλα δείχνουν, η εξέλιξη αυτή θα συνεχιστεί και μελλοντικά με τον ίδιο ρυθμό, καθώς ολοένα και περισσότεροι ερευνητές ασχολούνται με αυτές, ενώ παράλληλα το ενδιαφέρον των διαφόρων εταιρειών συνεχώς αυξάνεται. Τα περιθώρια εξέλιξης των τεχνολογιών αυτών είναι τεράστια, ενώ οι πιθανές αλλαγές και βελτιώσεις είναι δύσκολο να καθοριστούν με ακρίβεια, καθώς είναι συνεχείς και πολλές σε αριθμό. Επομένως είναι αρκετά δύσκολο να γίνουν συγκεκριμένες προβλέψεις για το μέλλον των τεχνολογιών αυτών, αλλά είναι γεγονός ότι η υιοθέτηση τους απο το ευρύ κοινό ολοένα και μεγαλώνει.

Στο κεφάλαιο αυτό προτείνονται κάποιες μελλοντικές επεκτάσεις για την παρούσα εφαρμογή, δεδομένου ότι οι τεχνολογίες που αναφέρονται είναι υπό ανάπτυξη και η εξέλιξή τους είναι απρόβλεπτη όπως εξηγήθηκε προηγουμένως.

Η χρήση ενός αποκεντρωμένου συστήματος, όπως το το IPFS (Interplanetary file system), με σκοπό την αντικατάσταση της βάσης δεδομένων που έχει επιλεγεί, θα ήταν μία προφανής βελτίωση και θα ήταν πιο κοντά στον αποκεντρωμένο χαρακτήρα της εφαρμογής. Με τη χρήση του αυτού του συστήματος θα γίνει εφικτή η αποθήκευση των αρχείων των έξυπνων αισθητήρων με αποκεντρωμένο τρόπο.

Η δημιουργία ενός περιβάλλοντος διαχείρισης του Quorum blockchain θα ήταν απαραίτητη προϋπόθεση για να δημιουργηθεί σε αυτό μια εφαρμογή που θα μπορεί να είναι λειτουργική σε πραγματικές συνθήκες παραγωγής. Καθώς το Quorum βρίσκεται σε στάδιο ανάπτυξης δεν είναι διαθέσιμη κάποια διεπαφή γραφικού περιβάλλοντος που θα έκανε την όλη διαδικασία ανάπτυξης και συντήρησης του συστήματος αρκετά ευκολότερη. Θα πρέπει να υποστηρίζονται από το διαχειριστικό περιβάλλον λειτουργίες όπως είναι η προσθήκη νέων κόμβων και η σύνδεσή τους με λογαριασμούς χρηστών, χωρίς να απαιτείται η χρήση εργαλείων που έχουν δημιουργηθεί από τρίτους.

Το frontend της εφαρμογής δημιουργήθηκε με δεδομένο ότι αφορά ένα test blockchain που θα χρησιμοποιηθεί στο πλαίσιο μίας διπλωματικής εργασίας και επομένως δεν θα ανταποκρίνεται με επιτυχία σε blockchains με μεγάλο όγκο

transactions. Άρα μια πιθανή επέκταση της εφαρμογής θα ήταν η προσαρμογή του σε συνθήκες πραγματικού blockchain, ώστε να αποφευχθούν πιθανά προβλήματα που θα αφορούν στην ταχύτητα απόκρισης αλλά και για διευκολύνει τον χρήστη με εργαλεία διαχείρισης του μεγαλύτερου όγκου πληροφορίας.

Στην εφαρμογή υπάρχει η δυνατότητα χρησιμοποίησης του NtuaToken ως μέσου συναλλαγών αντί του Ether. Στο έξυπνο συμβόλαιο έχουν υλοποιηθεί κάποιες βασικές λειτουργίες ώστε να είναι εφικτή η χρήση του στην εφαρμογή, αλλά απέχει αρκετά από το να είναι σε θέση να αποτελέσει αξιόπιστη εναλλακτική σε πραγματικές συνθήκες. Επομένως μια λογική επέκταση θα ήταν να υλοποιηθούν οι λειτουργίες που θα το καθιστούσαν ένα ολοκληρωμένο token με τις κατάλληλες προδιαγραφές ασφαλείας. Επιπλέον θα ήταν ήταν χρήσιμη μια ξεχωριστή dApp (τύπου decentralised exchange) ώστε να είναι εύκολο για τους χρήστες να ανταλλάξουν Ether με NtuaTokens με ασφάλεια, κάτι που φυσικά θα ήταν χρήσιμο και πέρα απο τα πλαίσια της χρήσης του για της εφαρμογή της εργασίας και θα είχε μεγάλο ενδιαφέρον σαν ξεχωριστό project.

Η εφαρμογή δημιουργήθηκε με σκοπό να είναι, στο βαθμό που ήταν εφικτό, ουδέτερη ως προς το αντικείμενο για το οποίο χρησιμοποιείται. Επομένως μια πιθανή επέκταση της θα ήταν η τροποποίηση της ώστε να μπορεί με κάποια παραμετροποίηση να γίνεται πλατφόρμα αγοραπωλησίας διαφόρων “προϊόντων”, είτε αυτά είναι δεδομένα έξυπνων αισθητήρων, είτε αρχεία εικόνας και ήχου, είτε ακομα και υλικών αγαθών.

8 Βιβλιογραφία

[1] 'The Journey to Web 3'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://medium.com/web3foundation/the-journey-to-web-3-3ead84261000. [Ημερομηνία πρόσβασης: 2-Οκτωβρίου-2025].

[2] 'Blockchain will usher in the era of decentralised computing'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://blog.bigchaindb.com/blockchain-will-usher-in-the-era-of-decentralised-computing-7f35e94af0b6. [Ημερομηνία πρόσβασης: 2-Οκτωβρίου-2025].

[3] 'Web3, the Stateful Web'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://blockchainhub.net/web3-decentralized-web/. [Ημερομηνία πρόσβασης: 2-Οκτωβρίου-2025].

[4] 'Elliptic Curve Cryptography (ECC)'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.certicom.com/content/certicom/en/ecc.html. [Ημερομηνία πρόσβασης: 2-Οκτωβρίου-2025].

[5] 'Elliptic-curve cryptography'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/wiki/Elliptic-curve_cryptography. [Ημερομηνία πρόσβασης: 2-Οκτωβρίου-2025].

[6] 'Elliptic Curve Pairings'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://medium.com/@VitalikButerin/exploring-elliptic-curve-pairings-c73c1864e627. [Ημερομηνία πρόσβασης: 2-Οκτωβρίου-2025].

[7] 'Elliptic-Curve Cryptography'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://medium.com/coinmonks/elliptic-curve-cryptography-6de8fc748b8b. [Ημερομηνία πρόσβασης: 2-Οκτωβρίου-2025].

[8] 'Peer-to-peer'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/wiki/Peer-to-peer. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[9] 'Peer-to-Peer Network'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://networkencyclopedia.com/peer-to-peer-network-p2p/. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[10] 'Ethereum documentation'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://ethereum.org/en/developers/docs/. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[11] Andreas M. Antonopoulos, 'Mastering Bitcoin'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/bitcoinbook/bitcoinbook/blob/develop/preface.asciidoc. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[12] Satoshi Nakamoto, 'Bitcoin Whitepaper'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://bitcoin.org/bitcoin.pdf. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[13] 'Ethereum Whitepaper'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://ethereum.org/en/whitepaper/. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[14] Gavin Wood, 'Ethereum Yellow Paper'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://ethereum.github.io/yellowpaper/paper.pdf. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[15] 'Ethereum Wire Protocol', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/ethereum/devp2p/blob/master/caps/eth.md. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[16] Andreas M. Antonopoulos, Gavin Wood, 'Mastering Ethereum'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/ethereumbook/ethereumbook/blob/develop/book.asciidoc. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[17] 'Gwei'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.investopedia.com/terms/g/gwei-ethereum.asp. [Ημερομηνία πρόσβασης: 3-Οκτωβρίου-2025].

[18] 'Quorum Whitepaper', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/ConsenSys/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf. [Ημερομηνία πρόσβασης: 5-Οκτωβρίου-2025].

[19] 'Quorum'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://consensys.net/quorum/developers/. [Ημερομηνία πρόσβασης: 5-Οκτωβρίου-2025].

[20] 'Quorum', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/ConsenSys/quorum-examples. [Ημερομηνία πρόσβασης: 5-Οκτωβρίου-2025].

[21] Davide Calvaresi, 'Reputation Management in Multi-Agent Systems using Permissioned Blockchain Technology', IEEE, 2018. Διαθέσιμο στο: https://ieeexplore.ieee.org/abstract/document/8609678. [Ημερομηνία πρόσβασης: 6-Οκτωβρίου-2025].

[22] Hakima Khelifi, 'Reputation-Based Blockchain for Secure NDN Caching in Vehicular Networks', IEEE, 2018. Διαθέσιμο στο: https://ieeexplore.ieee.org/abstract/document/8581849. [Ημερομηνία πρόσβασης: 6-Οκτωβρίου-2025].

[23] Fangyu Gai, 'Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network', 2018. Διαθέσιμο στο:

https://link.springer.com/chapter/10.1007/978-3-319-91458-9_41. [Ημερομηνία πρόσβασης: 6-Οκτωβρίου-2025].

[24] Alexander Schaub, 'A Trustless Privacy-Preserving Reputation System', IFIP Advances in Information and Communication Technology book series (IFIPAICT,volume 471). Διαθέσιμο στο: https://link.springer.com/chapter/10.1007/978-3-319-33630-5_27. [Ημερομηνία πρόσβασης: 7-Οκτωβρίου-2025].

[25] Davide Carboni, 'Feedback based Reputation on top of the Bitcoin Blockchain'. Διαθέσιμο στο: https://arxiv.org/abs/1502.01504. [Ημερομηνία πρόσβασης: 7-Οκτωβρίου-2025].

[26] Richard Dennis, 'Rep on the block: A next generation reputation system based on the blockchain', IEEE, 2015. Διαθέσιμο στο: https://ieeexplore.ieee.org/abstract/document/7412073. [Ημερομηνία πρόσβασης: 8-Οκτωβρίου-2025].

[27] Volkan Dedeoglu, 'A trust architecture for blockchain in IoT', 2019. Διαθέσιμο στο: https://dl.acm.org/doi/abs/10.1145/3360774.3360822. [Ημερομηνία πρόσβασης: 8-Οκτωβρίου-2025].

[28] Zhaojun Lu, 'BARS: A Blockchain-Based Anonymous Reputation System for Trust Management in VANETs', IEEE, 2018. Διαθέσιμο στο: https://ieeexplore.ieee.org/abstract/document/8455893. [Ημερομηνία πρόσβασης: 9-Οκτωβρίου-2025].

[29] 'Go-ethereum'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://geth.ethereum.org/. [Ημερομηνία πρόσβασης: 10-Οκτωβρίου-2025].

[30] 'Go-ethereum', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/ethereum/go-ethereum. [Ημερομηνία πρόσβασης: 10-Οκτωβρίου-2025].

[31] 'Solidity'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://docs.soliditylang.org/. [Ημερομηνία πρόσβασης: 10-Οκτωβρίου-2025].

[32] 'What is Web3.js'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://web3.hashnode.com/what-is-web3js-an-introduction-into-the-web3js-libraries. [Ημερομηνία πρόσβασης: 10-Οκτωβρίου-2025].

[33] 'web3.js - Ethereum JavaScript API', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/web3/web3.js. [Ημερομηνία πρόσβασης: 10-Οκτωβρίου-2025].

[34] 'web3.js - Documentation'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://web3js.readthedocs.io/en/v1.2.2/. [Ημερομηνία πρόσβασης: 10-Οκτωβρίου-2025].

[35] 'Truffle'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://trufflesuite.com/. [Ημερομηνία πρόσβασης: 11-Οκτωβρίου-2025].

[36] 'Truffle', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/trufflesuite/truffle. [Ημερομηνία πρόσβασης: 11-Οκτωβρίου-2025].

[37] 'What is Ganache?'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://trufflesuite.com/docs/ganache/. [Ημερομηνία πρόσβασης: 11-Οκτωβρίου-2025].

[38] 'Ganache', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/trufflesuite/ganache. [Ημερομηνία πρόσβασης: 11-Οκτωβρίου-2025].

[39] 'Ethereum js', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://ethereumjs.github.io/. [Ημερομηνία πρόσβασης: 11-Οκτωβρίου-2025].

[40] 'Ethereum utilities', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/ethereumjs/ethereumjs-util. [Ημερομηνία πρόσβασης: 11-Οκτωβρίου-2025].

[41] 'Keythereum', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/ethereumjs/keythereum. [Ημερομηνία πρόσβασης: 11-Οκτωβρίου-2025].

[42] 'Browserify'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://browserify.org/. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[43] 'About npm'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://docs.npmjs.com/about-npm. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[44] 'What is npm?'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[45] 'Node.js - documentation', Node.js. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://nodejs.org/en/docs/. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[46] 'Node.js - documentation', Node.js. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://nodejs.dev/en/api/v19/documentation/. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[47] 'Node.js - Introduction', tutorialspoint. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[48] 'V8'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://v8.dev/. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[49] 'JavaScript'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.javascript.com/. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[50] 'JavaScript - Overview', tutorialspoint. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.tutorialspoint.com/javascript/javascript_overview.htm. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[51] 'JavaScript'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://javascript.info/intro. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[52] 'JavaScript', w3schools. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.w3schools.com/js/. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[53] 'Learn HTML'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://html.com/. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[54] 'HTML Tutorial', tutorialspoint. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.tutorialspoint.com/html/index.htm. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[55] 'CSS: Cascading Style Sheets'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/CSS. [Ημερομηνία πρόσβασης: 15-Οκτωβρίου-2025].

[56] 'Bootstrap introduction'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://getbootstrap.com/docs/4.0/getting-started/introduction/. [Ημερομηνία πρόσβασης: 20-Οκτωβρίου-2025].

[57] 'Bootstrap tutorial', w3schools. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp. [Ημερομηνία πρόσβασης: 20-Οκτωβρίου-2025].

[58] 'jQuery API'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://api.jquery.com/. [Ημερομηνία πρόσβασης: 20-Οκτωβρίου-2025].

[59] 'jQuery Introduction', w3schools. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.w3schools.com/Jquery/jquery_intro.asp. [Ημερομηνία πρόσβασης: 20-Οκτωβρίου-2025].

[60] 'MySQL'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://dev.mysql.com/. [Ημερομηνία πρόσβασης: 20-Οκτωβρίου-2025].

[61] 'MySQL Introduction', w3schools. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.w3schools.com/php/php_mysql_intro.asp. [Ημερομηνία πρόσβασης: 20-Οκτωβρίου-2025].

[62] 'Database Comparison'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://db-engines.com/en/system/MySQL. [Ημερομηνία πρόσβασης: 20-Οκτωβρίου-2025].

[63] 'MySQL installation', Npm js. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.npmjs.com/package/mysql. [Ημερομηνία πρόσβασης: 20-Οκτωβρίου-2025].

[64] 'Vagrant Documentation'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://developer.hashicorp.com/vagrant/docs. [Ημερομηνία πρόσβασης: 23-Οκτωβρίου-2025].

[65] 'Vagrant', GitHub. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/hashicorp/vagrant. [Ημερομηνία πρόσβασης: 23-Οκτωβρίου-2025].

[66] 'SQL injection', Wikipedia. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/wiki/SQL_injection. [Ημερομηνία πρόσβασης: 23-Οκτωβρίου-2025].

[67] 'Cross-Origin Resource Sharing (CORS)'. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. [Ημερομηνία πρόσβασης: 24-Οκτωβρίου-2025].

Παράρτημα Ι

Στο παράρτημα αυτό παρατίθενται οδηγίες για την εγκατάσταση και τη χρήση της εφαρμογής.

Α Εγκατάσταση

1. Clone quorum-examples repository:
    - Ανοίξτε ένα CMD.
    - Μεταβείτε στον κατάλογο όπου θέλετε να κλωνοποιήσετε το αποθετήριο.
    - Εκτελέστε την παρακάτω εντολή για να κλωνοποιήσετε το αποθετήριο: “git clone https://github.com/Consensys/quorum-examples.git”

2. Ενεργοποίηση Virtualization:
    - Προσπελάστε τις ρυθμίσεις του BIOS ή του UEFI του υπολογιστή σας. Η διαδικασία μπορεί να διαφέρει ανάλογα με τον κατασκευαστή του υπολογιστή, επομένως ανατρέξτε στην τεκμηρίωση ή αναζητήστε στο διαδίκτυο οδηγίες που είναι συγκεκριμένες για τη συσκευή σας.
    - Εντοπίστε μια επιλογή που σχετίζεται με την εικονικοποίηση, όπως "Virtualization Technology (VT-x)" ή "AMD-V" και ενεργοποιήστε την.
    - Αποθηκεύστε τις αλλαγές και εξέλθετε από τις ρυθμίσεις του BIOS/UEFI.
    - Κάντε επανεκκίνηση του υπολογιστή σας.

3. Εγκατάσταση του VirtualBox:
    - Επισκεφθείτε τον ιστότοπο του VirtualBox (https://www.virtualbox.org/wiki/Downloads) και κατεβάστε την κατάλληλη έκδοση για το λειτουργικό σας σύστημα.
    - Εκτελέστε το πρόγραμμα εγκατάστασης και ακολουθήστε τις οδηγίες που εμφανίζονται στην οθόνη για να ολοκληρώσετε την εγκατάσταση.

4. Εγκατάσταση του Vagrant:
    - Ανοίξτε ένα CMD.
    - Εκτελέστε την παρακάτω εντολή για να κατεβάσετε το Vagrant:

“Wget https://releases.hashicorp.com/vagrant/2.2.18/vagrant_2.2.18_x86_64.deb”

    - Εγκαταστήστε το Vagrant χρησιμοποιώντας την παρακάτω εντολή:

“sudo dpkg -i vagrant_2.2.18_x86_64.deb”

5. Εγκατάσταση του NVM (Node Version Manager):

- Ανοίξτε ένα CMD.
- Εκτελέστε την παρακάτω εντολή για να εγκαταστήσετε το NVM:

“curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash”

- Κλείστε το CMD και ανοίξτε το ξανά για να φορτωθούν οι μεταβλητές περιβάλλοντος του NVM.

6. Εγκατάσταση του Node.js:
   - Ανοίξτε ένα CMD.
   - Εκτελέστε την παρακάτω εντολή για να εγκαταστήσετε το Node.js χρησιμοποιώντας το NVM:

     “nvm install node”
   - Επιβεβαιώστε την εγκατάσταση ελέγχοντας την έκδοση του Node.js:

     “node -v”

7. Εγκατάσταση του phpMyAdmin:
   - Ανάλογα με τη διανομή Linux που χρησιμοποιείτε, μπορείτε να εγκαταστήσετε το phpMyAdmin χρησιμοποιώντας τον διαχειριστή πακέτων της διανομής (π.χ., apt, yum, pacman) που είναι συγκεκριμένος για τη διανομή σας.
   - Για παράδειγμα, μπορείτε να εκτελέσετε την παρακάτω εντολή για να εγκαταστήσετε το phpMyAdmin:

     “sudo apt install phpmyadmin”
   - Ακολουθήστε τις οδηγίες που εμφανίζονται για να ολοκληρώσετε την εγκατάσταση.

8. Εγκατάσταση του MySQL:
   - Ανάλογα με τη διανομή Linux που χρησιμοποιείτε, μπορείτε να εγκαταστήσετε το MySQL χρησιμοποιώντας τον διαχειριστή πακέτων της διανομής (π.χ., apt, yum, pacman) που είναι συγκεκριμένος για τη διανομή σας.
   - Για παράδειγμα, μπορείτε να εκτελέσετε την παρακάτω εντολή για να εγκαταστήσετε το MySQL:

     “sudo apt install mysql-server”
   - Ακολουθήστε τις οδηγίες που εμφανίζονται για να ολοκληρώσετε την εγκατάσταση.
   - Εκκινήστε και ενεργοποιήστε την υπηρεσία του MySQL:

     “sudo systemctl start mysql”

“sudo systemctl enable mysql”

9. Εγκατάσταση του Truffle:
    - Ανοίξτε ένα CMD.
    - Εκτελέστε την παρακάτω εντολή για να εγκαταστήσετε το Truffle globally:
        “npm install -g truffle”

10. Εγκατάσταση του Ganache CLI:
    - Ανοίξτε ένα CMD.
    - Εκτελέστε την παρακάτω εντολή για να εγκαταστήσετε το Ganache CLI globally:
        “npm install -g ganache-cli”
11. Προσθήκη των επιλογων για port forward στο αρχείο Vagrantfile:
    - Εντοπίστε το αρχείο Vagrantfile στον φάκελο του έργου σας.
    - Ανοίξτε το αρχείο Vagrantfile με έναν επεξεργαστή κειμένου.
    - Βρείτε το τμήμα όπου πραγματοποιείται port forward (συνήθως βρίσκεται κάτω από το Vagrant.configure).
    - Προσθέστε μια γραμμή για την προώθηση της επιθυμητής θύρας.

12. Αντιγραφή των αρχείων της εφαρμογής, δηλαδή τόσο των έξυπνων συμβολαίων, οσο και των ιστοσελίδων.
    - Μεταφορά στους αντίστοιχους φακέλους των αρχείων των έξυπνων συμβολαίων αλλά και των αρχείων που αφορούν το migration.
    - Μεταφορά των html και javascript αρχείων.

13. Εγκατάσταση ενός CORS [67] plugin στο φυλλομετρητή μας. Ένα πρόσθετο CORS, το οποίο συνήθως υλοποιείται στην πλευρά του διακομιστή, βοηθά στην ενεργοποίηση και τη διαχείριση των cross-origin αιτημάτων θέτοντας τις κατάλληλες επικεφαλίδες HTTP στις απαντήσεις του διακομιστή. Ένα τέτοιου τύπου plugin είναι το ακόλουθο:
    “https://chrome.google.com/webstore/detail/moesif-origin-cors-change/digfbfaphojjndkpccljibejjbppifbc”

Β Χρήση εφαρμογής

1. Πλοηγηθείτε στο φάκελο που βρίσκονται τα αρχεία της εφαρμογής.

2. Εκκινήστε το 7nodes example και αποκτήστε πρόσβαση σε αυτό.

- Ανοίξτε ένα CMD.
- Εκτελέστε τις παρακάτω εντολές:
    cd quorum-examples
    vagrant up
    vagrant ssh

3. Πλοηγηθείτε στο φάκελο 7nodes example
    - Εκτελέστε την παρακάτω εντολή:
        cd path/to/7nodes

4. Αρχικοποιήστε accounts & keystores
    - Εκτελέστε την παρακάτω εντολή:
        ./{consensus}-init.sh

5. Εκκινήστε το Quorum και τους privacy manager nodes (Constellation or Tessera):
    - Εκτελέστε την παρακάτω εντολή:
        ./{consensus}-start.sh

6. Πλοηγηθείτε στο φάκελο που περιέχει τα αρχεία της solidity.
    - Εκτελέστε την παρακάτω εντολή:
        Truffle migrate

7. Πλοηγηθείτε στο φάκελο που περιέχει τα αρχεία της εφαρμογής.
    - Εκτελέστε την παρακάτω εντολή:
        Start nodejs app.js

8. Στο πρόγραμμα περιήγησής σας επισκεφτείτε την παρακάτω σελίδα για να αποκτήσετε πρόσβαση στην εφαρμογή:
    http://localhost:8080

9. Σταματήστε το Quorum:
    - Πλοηγηθείτε στο φάκελο 7nodes example
    - Εκτελέστε την παρακάτω εντολή:
        ./stop.sh

10. Τερματίσετε την λειτουργία του Vagrant.
    - Εκτελέστε την παρακάτω εντολή:
        vagrant suspend
    - Σε περίπτωση που θέλετε να διαγράψετε το vagrant instance:
        vagrant destroy

11. Για να ξεκινήσετε από το μηδέν, μετά την διαγραφή του προηγούμενου instance, εκτελέστε ξανά το vagrant up και ακολουθείστε την υπόλοιπη διαδικασία.

Παράρτημα ΙΙ

Κώδικας έξυπνων συμβολαίων

Στο παράρτημα αυτό βρίσκεται ο κώδικας των έξυπνων συμβολαίων που αναπτύχθηκαν, γραμμένα στη γλώσσα Solidity.

Ο πηγαίος κώδικας του έξυπνου συμβολαίου **main**:

```
pragma solidity ^0.5.7;

contract Main {

uint public oneEther = 1 ether;

  struct sensor {
    string sensorName;
    uint price;
    uint32 startTime;
    uint32 frequency;
    int32 latitude;
    int32 longitude;
    bool exists;
    bool deleted;
  }

  struct purchase{
    uint id;
    bool useToken;
    address payable buyerAddress;
    address buyerContractAddress;
    string buyerQuorumPublicKey;
    address adminAddress;
    address adminContractAddress;
    bool exists;
    bool completed;
  }
```

```
  struct admin{
    address payable adminAddress;
    address adminContractAddress;
  }

  struct owner{
    address payable ownerAddress;
    uint ownerPercentage;
  }

  mapping(address=>address) usersContractAddressTable;
  mapping(uint=>admin) adminsTable;
  mapping(uint => owner[]) ownersTable;
  sensor[] sensorsTable;
  purchase[] purchasesTable;

      event UserAdded(address userAddress, address userNodeCoinbase,address
userContractAddress, string userQuorumPublicKey);
  event AdminAdded(uint id, address adminAddress,address adminContractAddress);

    event SensorAdded(uint id,string sensorName, uint price, uint32 startTime, uint32
frequency, int32 latitude, int32 longitude);
  event SensorDeleted(uint id);
  event PriceChanged(uint id, uint price);
  event OwnershipChanged(uint id, address ownerAddress, uint ownerPercentage);

  event PurchaseInfo(uint purchaseId,uint id, bool useToken, address buyerAddress,
              address buyerContractAddress, string buyerQuorumPublicKey,address
adminAddress, address adminContractAddress);
  event PurchaseCompleted(uint purchaseId,uint id, bool purchaseAccepted);

  modifier hasAccess(uint _id) {
    require(msg.sender==adminsTable[_id].adminAddress);
    _;
  }

  modifier exists(uint _id) {
```

```
    require(sensorsTable[_id].exists);
    require(!sensorsTable[_id].deleted);
    _;
  }

  function addSensor(string calldata _sensorName, uint _price, uint32 _startTime,
                     uint32 _frequency, int32 _latitude, int32 _longitude, address
_adminContractAddress)
    external returns (uint) {
      address payable _adminAddress=msg.sender;
      uint _id=sensorsTable.length;
       sensorsTable.push(sensor(_sensorName, _price, _startTime, _frequency, _latitude,
_longitude,true, false));
          emit SensorAdded(_id, _sensorName, _price, _startTime, _frequency, _latitude,
_longitude);

      adminsTable[_id] = admin(_adminAddress, _adminContractAddress);
      emit AdminAdded(_id, _adminAddress, _adminContractAddress);
      return _id;
    }

    function updatePrice(uint _id, uint _price) external hasAccess(_id) exists(_id){
      sensorsTable[_id].price = _price;
      emit PriceChanged(_id, _price);
    }

          function updateOwnership(uint _id, address payable _ownerAddress, uint
_ownerPercentage) external hasAccess(_id) exists(_id){
      bool foundAddress=false;
      uint sum=0;
      for(uint i=0;i<ownersTable[_id].length;i++){
        sum+=ownersTable[_id][i].ownerPercentage;
                if((sum-ownersTable[_id][i].ownerPercentage+_ownerPercentage)<=100 &&
ownersTable[_id][i].ownerAddress==_ownerAddress){
          ownersTable[_id][i].ownerPercentage=_ownerPercentage;
          foundAddress=true;
           emit OwnershipChanged(_id, _ownerAddress, _ownerPercentage);
```

```
      }
    }
    if(sum+_ownerPercentage<=100 && !foundAddress){
      ownersTable[_id].push(owner(_ownerAddress, _ownerPercentage));
        emit OwnershipChanged(_id, _ownerAddress, _ownerPercentage);
    }
  }

  function deleteSensor(uint _id) external hasAccess(_id) exists(_id){
    sensorsTable[_id].deleted=true;
    emit SensorDeleted(_id);
  }

   function buySensor(uint _id, bool _useToken, address _buyerContractAddress, string
calldata _buyerQuorumPublicKey) external payable exists(_id){
  require(msg.sender!=adminsTable[_id].adminAddress);
    address payable _buyerAddress = msg.sender;
    uint _price = sensorsTable[_id].price*oneEther;
    address payable _adminAddress = adminsTable[_id].adminAddress;
    address _adminContractAddress = adminsTable[_id].adminContractAddress;

    if(_price==msg.value || _useToken)
    {
      if(!_useToken)
      _adminAddress.transfer(msg.value);

                        purchasesTable.push(purchase(_id, _useToken, _buyerAddress,
_buyerContractAddress, _buyerQuorumPublicKey,_adminAddress,
_adminContractAddress,true, false));
            emit PurchaseInfo(purchasesTable.length-1, _id, _useToken, _buyerAddress,
_buyerContractAddress, _buyerQuorumPublicKey, _adminAddress,
_adminContractAddress);
    }
  }
```

```
        function completePurchase(uint _purchaseId, uint _id, bool _useToken, bool
_acceptPurchase) public payable hasAccess(_id) {
    require(!purchasesTable[_purchaseId].completed);
    require(purchasesTable[_purchaseId].id==_id);
    if(sensorsTable[_id].deleted || !sensorsTable[_id].exists)_acceptPurchase=false;
    uint _price = sensorsTable[_id].price*oneEther;
    uint _amountSent;
    if(!_useToken){
    require(_price==msg.value);
    if(!_acceptPurchase) {
     (purchasesTable[_purchaseId].buyerAddress).transfer(_price);
     } else{
       for(uint i=0; i<ownersTable[_id].length;i++){
         address payable _ownerAddress=ownersTable[_id][i].ownerAddress;
         uint _toSend=(ownersTable[_id][i].ownerPercentage*_price)/100;
         _amountSent+=_toSend;
         (_ownerAddress).transfer(_toSend);
       }
       (msg.sender).transfer(_price-_amountSent);
     }
    }
    purchasesTable[_purchaseId].completed=true;
     emit PurchaseCompleted(_purchaseId, _id, _acceptPurchase);
    }


         function addUser(address _userNodeCoinbase,address _userContractAddress,
string calldata _usersQuorumPublicKey) external{
      address _userAddress = msg.sender;
      usersContractAddressTable[_userAddress]=_userContractAddress;
           emit UserAdded(_userAddress, _userNodeCoinbase, _userContractAddress,
_usersQuorumPublicKey);
    }

    function getAdmin(uint _id) public view returns (address[2] memory) {
                                        return      [adminsTable[_id].adminContractAddress,
adminsTable[_id].adminContractAddress];
```

```
    }

        function getUserContractAddress(address _userAddress) public view returns
(address) {
      return usersContractAddressTable[_userAddress];
    }

  }
```

Ο πηγαίος κώδικας του έξυπνου συμβολαίου **Reputation**:

```
pragma solidity ^0.5.7;

contract Reputation {
    event ReviewAdded(address sellerAddress, address buyerAddress, uint id, uint purchaseId, int rating);
    event FeedbackAdded(address sellerAddress, address buyerAddress, uint id, uint purchaseId, int rating);

  struct purchase{
    uint id;
    address buyerAddress;
    address sellerAddress;
    int reviewRating;
    int feedbackRating;
    bool exists;
  }

  mapping(address => int) private sellersReputationTable;
  mapping(address => int) private buyersReputationTable;
  mapping(uint => int) private sensorsReputationTable;
  mapping(uint => purchase) private purchasesTable;

  modifier purchaseExists(uint _purchaseId) {
    require(purchasesTable[_purchaseId].exists);
    _;
  }

  function getSellerReputation(address _sellerAddress) public view returns (int){
    return sellersReputationTable[_sellerAddress];
  }

  function getBuyerReputation(address _buyerAddress) public view returns (int){
    return buyersReputationTable[_buyerAddress];
  }
```

```
  function getSensorReputation(uint _id) public view returns (int){
    return sensorsReputationTable[_id];
  }

  function addPurchase(uint _purchaseId, uint _id, address _buyerAddress) external{
    address _sellerAddress=msg.sender;
     purchasesTable[_purchaseId] = purchase(_id, _buyerAddress, _sellerAddress, 0, 0,
true) ;
  }

                function giveReview(uint _purchaseId, int _rating) external
purchaseExists(_purchaseId){
    address _buyerAddress=msg.sender;

    if(purchasesTable[_purchaseId].buyerAddress==_buyerAddress
      && purchasesTable[_purchaseId].reviewRating==0
      && (_rating==1 || _rating==-1)){
        purchasesTable[_purchaseId].reviewRating=_rating;
        sellersReputationTable[purchasesTable[_purchaseId].sellerAddress]+=_rating;
        sensorsReputationTable[purchasesTable[_purchaseId].id]+=_rating;
        emit ReviewAdded(purchasesTable[_purchaseId].sellerAddress, _buyerAddress,
          purchasesTable[_purchaseId].id,_purchaseId, _rating);
        }
      }

                function giveFeedback(uint _purchaseId, int _rating) external
purchaseExists(_purchaseId){
    address _sellerAddress=msg.sender;

    if(purchasesTable[_purchaseId].sellerAddress==_sellerAddress
      && purchasesTable[_purchaseId].feedbackRating==0
      && (_rating==1 || _rating==-1)){
        purchasesTable[_purchaseId].feedbackRating=_rating;
        buyersReputationTable[purchasesTable[_purchaseId].buyerAddress]+=_rating;
                                              emit      FeedbackAdded(_sellerAddress,
purchasesTable[_purchaseId].buyerAddress,
          purchasesTable[_purchaseId].id,_purchaseId, _rating);
```

```
    }
  }
}
```

Ο πηγαίος κώδικας του έξυπνου συμβολαίου **User**:

```
pragma solidity ^0.5.7;

contract User{
  address ownerAddress;
  address nodeCoinbase;

  mapping(uint=>string) private ownedPassphrases;
  mapping(uint=>string) private boughtPassphrases;

  modifier hasAccess() {
    require(msg.sender==ownerAddress);
    _;
  }

  constructor(address _nodeCoinbase) public {
    ownerAddress=msg.sender;
    nodeCoinbase = _nodeCoinbase;
  }


  function addPassphrase(uint _id, string calldata _passphrase) external hasAccess(){
    ownedPassphrases[_id]=_passphrase;
  }

  function addBoughtPassphrase(uint _id, string calldata _passphrase) external {
    boughtPassphrases[_id]=_passphrase;
  }

      function changePassphrase(uint _id, string calldata _passphrase) external
hasAccess(){
    ownedPassphrases[_id]= _passphrase;
  }

  function getPassphrase(uint _id) external view  hasAccess() returns (string memory){
    return ownedPassphrases[_id];
```

```
  }

    function getBoughtPassphrase(uint _id) external view hasAccess() returns (string memory){
    return boughtPassphrases[_id];
  }
}
```

Ο πηγαίος κώδικας του έξυπνου συμβολαίου **Token**:

```
pragma solidity ^0.5.7;

contract Token {

  event Transfer(address indexed _from, address indexed _to, uint256 _value);


  mapping(address => uint256) public balanceOf;

  string public name;
  string public symbol;
  uint8 public decimals;
  uint256 public totalSupply;

  constructor() public{
    balanceOf[0xed9d02e382b34818e88B88a309c7fe71E65f419d] = 1000 * 1 ether;
    balanceOf[0xcA843569e3427144cEad5e4d5999a3D0cCF92B8e] = 1000 * 1 ether;
    balanceOf[0x0fBDc686b912d7722dc86510934589E0AAf3b55A] = 1000 * 1 ether;
    balanceOf[0x9186eb3d20Cbd1F5f992a950d808C4495153ABd5] = 1000 * 1 ether;
    balanceOf[0x0638E1574728b6D862dd5d3A3E0942c3be47D996] = 1000 * 1 ether;
    balanceOf[0xAE9bc6cD5145e67FbD1887A5145271fd182F0eE7] = 1000 * 1 ether;
    balanceOf[0xCC71C7546429a13796cf1BF9228bFf213e7Ae9cc] = 1000 * 1 ether;
    balanceOf[0xa9e871F88CBeb870d32D88E4221dcfBD36Dd635a] = 1000 * 1 ether;
    name = "NtuaToken";
    symbol = "NtuaTok";
    decimals = 18;
  }

  function balanceof(address _owner) external view returns (uint256 balance){
    return balanceOf[_owner];
  }

  function _transfer(address _from, address _to, uint _value) internal {
    //require(_to != 0x0);
    require(balanceOf[_from] >= _value && balanceOf[_to] + _value >= balanceOf[_to]);
```

```
    balanceOf[_from] -= _value;
    balanceOf[_to] += _value;
    emit Transfer(_from,_to,_value);
  }

  function transfer(address _to, uint256 _value) external{
    _transfer(msg.sender,_to,_value);
  }

   function transferFrom(address _from, address _to, uint256 _value) external returns
(bool success){
    _transfer(_from,_to,_value);
    return true;
  }
}
```