ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Έξυπνων Συμβολαίων στο Blockchain και εφαρμογή στο IoT

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Ν. Παπαδόδημας

Επιβλέπουσα: Θεοδώρα Βραρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Έξυπνων Συμβολαίων στο Blockchain και εφαρμογή στο IoT

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Ν. Παπαδόδημας

Επιβλέπουσα: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Ημερομηνία.

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Δημήτριος Ασκούνης	Συμεών Παπαβασιλείου
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2018

.....

Γεώργιος Ν. Παπαδόδημας
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Για την έως τώρα σταδιοδρομία μου θα ήθελα να ευχαριστήσω τους γονείς και τα αδέρφια μου, που είναι πάντα δίπλα μου και με στηρίζουν σε όλες τις επιλογές μου. Χωρίς αυτούς δεν θα μπορούσα να έχω φτάσει στο σημείο που βρίσκομαι σήμερα. Επίσης ευχαριστώ όλους μου τους φίλους.

Ευχαριστώ την κυρία Θεοδώρα Βαρβαρίγου για την πολύτιμη καθοδήγηση της κατά την διάρκεια των σπουδών μου στο Πολυτεχνείο καθώς και για την ανάθεση της πολύ ενδιαφέρουσας διπλωματικής εργασίας. Για την επιτυχή περάτωση της διπλωματικής εργασίας ευχαριστώ τον υποψήφιο διδάκτορα Γεώργιο Παλαιοκρασσά και τον δόκτορα Αντώνη Λίτκε για την βοήθεια τους, δίχως την οποία το έργο μου θα ήταν πολύ δυσκολότερο.

Τέλος, ευχαριστώ όλους εκείνους τους ανθρώπους που συντελούν στην λειτουργία του Εθνικού Μετσόβιου Πολυτεχνείου. Τους καθηγητές που με μεράκι διδάσκουν και εμπνέουν τους φοιτητές. Τους υπεύθυνους και τους βοηθούς των εργαστηρίων καθώς και όλο το υπόλοιπο υποστηρικτικό προσωπικό.

Γεώργιος Παπαδόδημας
Αθήνα, Φεβρουάριος 2018

Περίληψη

Το διαδίκτυο, έφερε μία επανάσταση στον τρόπο που οι άνθρωποι αντιλαμβάνονται τον κόσμο και τώρα αποτελεί αναπόσπαστο κομμάτι της καθημερινότητάς τους. Οι τεχνολογίες στις οποίες βασίζεται εξελίσσονται με ραγδαίους ρυθμούς ενώ παράλληλα δημιουργούνται νέες. Ένα χαρακτηριστικό του διαδικτύου που έχει παραμείνει σταθερό για μεγάλο χρονικό διάστημα, είναι η κυρίαρχη αρχιτεκτονική πελάτη-εξυπηρετητή πάνω στην οποία βασίζεται η πλειοψηφία των προσφερόμενων στο διαδίκτυο εφαρμογών και υπηρεσιών. Αυτό αναμένεται να αλλάξει σε μεγάλο βαθμό χάρη στις νέες τάσεις που δημιουργούνται λόγω νεοεμφανιζόμενων τεχνολογιών, μεταξύ των οποίων βρίσκονται το blockchain και το Διαδίκτυο των Πραγμάτων που ενδέχεται να δώσουν στο διαδίκτυο έναν πιο αποκεντρωμένο χαρακτήρα.

Η τεχνολογία του blockchain δημιουργήθηκε αρχικά για την λειτουργία του κρυπτονομίσματος Bitcoin, όμως, συνεχώς εξελίσσεται και τώρα το πεδίο εφαρμογής του έχει γίνει πολύ πιο ευρύ. Το blockchain είναι ένα ψηφιακό κατανεμημένο δημόσιο καθολικό (ledger) στο οποίο καταγράφονται συναλλαγές και συμφωνίες με τρόπο αδιάβλητο και υποστηρίζεται από ένα δίκτυο ομότιμων κόμβων. Εκτός από μέσο για την λειτουργία κρυπτονομισμάτων, μπορεί να αποτελέσει πυλώνα για την δημιουργία και λειτουργία αποκεντρωμένων εφαρμογών (DApps – Decentralized Applications), εφαρμογών δηλαδή που βασίζονται σε ένα κατανεμημένο δίκτυο ομότιμων κόμβων και όχι στους εξυπηρετητές κάποιου οργανισμού. Το πιο χαρακτηριστικό παράδειγμα της χρήσης αυτής του blockchain, αποτελεί το Ethereum blockchain, μία Turing complete κατανεμημένη υπολογιστική αρχιτεκτονική, πλατφόρμα ανάπτυξης αποκεντρωμένων εφαρμογών μέσω έξυπνων συμβολαίων. Οι τεχνολογίες της οικογένειας του Ethereum, χρησιμοποιήθηκαν για την ανάπτυξη μίας αποκεντρωμένης εφαρμογής στα πλαίσια της διπλωματικής εργασίας.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η διερεύνηση των «αποκεντρωτικών» τεχνολογιών του IoT και του blockchain καθώς και η ανάπτυξη μίας αποκεντρωμένης εφαρμογής που θα συνδυάζει τα δύο ανωτέρω πεδία. Η εφαρμογή αυτή, λειτουργεί μέσω έξυπνων συμβολαίων που «τρέχουν» στο Ethereum blockchain και συνδυάζει την τεχνολογία του blockchain με το IoT. Πιο συγκεκριμένα, η εφαρμογή «Crypto Weather», όπως ονομάζεται, είναι μία πλατφόρμα αγοραπωλησιών μετρήσεων IoT (έξυπνων) αισθητήρων καιρού, η οποία λειτουργεί πάνω στο Ethereum blockchain.

Στην διπλωματική εργασία λοιπόν, παρουσιάζονται τόσο θεωρητικά ζητήματα του αποκεντρωμένου ιστού και των συστατικών τεχνολογιών του, όσο και πρακτικά ζητήματα, μέσω της παρουσίασης της διαδικασίας ανάπτυξης μίας αποκεντρωμένης εφαρμογής. Στο τέλος παρουσιάζεται και η εφαρμογή αυτή καθαυτή.

Λέξεις κλειδιά

Ethereum, Blockchain, Έξυπνα συμβόλαια, Διαδίκτυο των Πραγμάτων, Node-RED, Προγραμματισμός ροών, Αποκεντρωμένες εφαρμογές

Abstract

The revolution of the Internet changed the way most people understand the world and now is part of our everyday lives. The technologies on which it is based on are rapidly evolving while new technologies are invented. A feature that remained unchanged for many years is the client-server architecture, which is being used on most applications and services available on the Internet. This is about to change, thanks to newly introduced technologies, among which are the blockchain and the Internet of Things (IoT) that may decentralize the Internet.

Blockchain technology was originally devised in order to serve the cryptocurrency Bitcoin, but it evolved and now its scope is very broad. The blockchain is a digitalized distributed ledger of transactions. Besides the cryptocurrency, blockchain can be used for the creation and operation of decentralized applications (DApps), scilicet applications that are based on a peer to peer network rather than an organization's server for their operation. An example of this feature of the blockchain, is Ethereum, a Turing complete distributed computing architecture, a platform for developing and deploying decentralized applications through smart contracts. Technologies of the Ethereum technology stack are being used in this thesis.

The purpose of this diploma thesis is the engagement with the decentralization technologies of the IoT and the blockchain as well as the development of a decentralized application (DApp) that will combine the two previous fields. This application operates through smart contracts that are executed on the Ethereum blockchain and combines blockchain technology with IoT. More specifically, "Crypto Weather", as is the name of the application, is a platform for sharing (buying and selling) measurements of IoT weather sensors that operates on the Ethereum blockchain.

This diploma thesis, includes a presentation on the theoretical matters of the decentralized web and its component technologies. A presentation of the development process of the decentralized application "Crypto Weather" is also included, as well as a presentation of the application itself.

Key words

Ethereum, Blockchain, Smart contracts, Internet of Things, Node-RED, Flow programming, Decentralized applications

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1

Εισαγωγή

Ο κόσμος έχει προ πολλού εισέλθει στην εποχή της ψηφιοποίησης και του διαδικτύου. Το διαδίκτυο έφερε μία επανάσταση σε πολλές πτυχές της καθημερινής ζωής και της οικονομίας. Στην αρχή ψηφιοποιήθηκε η πληροφορία και έτσι έγινε δυνατή η ταχύτατη διάδοσή της σε πολύ μεγάλες αποστάσεις μέσω του διαδικτύου, έτσι οδηγηθήκαμε στην παγκοσμιοποίηση της γνώσης. Έπειτα δημιουργήθηκαν τα ηλεκτρονικά καταστήματα, συμβάλλοντας σε μεγάλο βαθμό στην ανάπτυξη του διεθνούς εμπορίου. Ακολούθησαν τα κοινωνικά δίκτυα τα οποία συνέδεσαν ανθρώπους από όλο τον κόσμο. Όλον αυτόν τον καιρό όμως ο τρόπος που ο κόσμος αντιλαμβάνεται το χρήμα, και ο τρόπος με τον οποίο το χρηματοπιστωτικό σύστημα λειτουργεί, πολύ λίγο έχει αλλάξει. Τα τελευταία χρόνια, αρχής γενομένης της δημιουργίας του Bitcoin το 2009, υπάρχει η τάση της ψηφιοποίησης του χρήματος, η δημιουργία νέων νομισμάτων τα οποία ούτε εκδίδονται ούτε και ελέγχονται από κάποια κυβέρνηση ή κεντρική τράπεζα, αλλά η λειτουργία τους βασίζεται πάνω στην τεχνολογία του blockchain και ονομάζονται κρυπτονομίσματα, λόγω των κρυπτογραφικών πρωτοκόλλων που χρησιμοποιούν για να διασφαλίζουν την απρόσκοπτη λειτουργία τους. Επίσης, έχει εμφανιστεί και η τάση δημιουργίας αποκεντρωμένων εφαρμογών, που όπως και τα κρυπτονομίσματα βασίζεται στην τεχνολογία του blockchain. Τέλος, ήδη έχει ξεκινήσει η εξάπλωση του Διαδικτύου των Πραγμάτων (Internet of Things), που επίσης θα φέρει μία επανάσταση στον παγκόσμιο ιστό. Βλέπουμε λοιπόν ότι πολλές διαφορετικές τεχνολογίες αναπτύσσονται παράλληλα και υπόσχονται να βελτιώσουν και να αλλάξουν άρδην την μορφή και την ποιότητα των υπηρεσιών του διαδικτύου.

1.1 Αντικείμενο της Διπλωματικής Εργασίας

Στην παρούσα διπλωματική εργασία πρωταγωνιστικό ρόλο θα καταλάβει η, νέα, τεχνολογία του blockchain και πιο συγκεκριμένα το Ethereum. Επίσης θα χρησιμοποιηθούν τεχνολογίες του Internet of Things.

Σκοπός είναι αξιοποίηση της τεχνολογίας του Ethereum blockchain για την ανάπτυξη μίας πρωτοποριακής αποκεντρωμένης εφαρμογής (DApp – Decentralized Application) η οποία θα αποτελέσει μία πλατφόρμα διαμοιρασμού δεδομένων IoT αισθητήρων καιρού. Οι ιδιοκτήτες IoT αισθητήρων καιρού θα καταχωρούν τους αισθητήρες τους στο blockchain, ώστε οι υποψήφιοι αγοραστές να μπορούν να αγοράζουν δεδομένα μετρήσεων των αισθητήρων της επιλογής τους.

Η καινοτομία έγκειται στον γεγονός ότι για την υλοποίηση της εφαρμογής δεν θα χρησιμοποιηθούν αποκλειστικά παραδοσιακές μέθοδοι ανάπτυξης δικτυακού λογι-

σμικού της αρχιτεκτονικής πελάτη – εξυπηρετητή, αλλά θα γίνει χρήση νέων τεχνολογιών, κυρίως του οικοσυστήματος του Ethereum, οι οποίες βασίζονται στην αρχιτεκτονική ομότιμων κόμβων (P2P). Δηλαδή η εφαρμογή θα ανήκει στην νέα κατηγορία εφαρμογών που ονομάζονται αποκεντρωμένες εφαρμογές (DApp – Decentralized Application), που αναπτύσσεται εξαπλώνεται ταχύτατα και πιθανόν στο μέλλον να αλλάξει σε μεγάλο βαθμό τον τρόπο με τον οποίο οι υπηρεσίες του διαδικτύου, ακόμα και το ίδιο το διαδίκτυο είναι δομημένα.

Για τον ανωτέρω σκοπό θα γίνουν δύο διακριτές εργασίες. Η μία είναι η ανάπτυξη των έξυπνων συμβολαίων (smart contracts), τα οποία θα αποτελέσουν την καρδιά του συστήματος. Τα έξυπνα συμβόλαια θα περιέχουν τον κώδικα που θα εκτελείται στο Ethereum blockchain, στην ουσία θα υλοποιούν όλη την λειτουργικότητα και θα δώσουν στην εφαρμογή αποκεντρωμένο χαρακτήρα. Η άλλη εργασία είναι η ανάπτυξη του ιστοτόπου (website), μέσω του οποίου οι χρήστες θα μπορούν να αλληλοεπιδρούν με τα έξυπνα συμβόλαια στο blockchain. Δηλαδή η ιστοσελίδα θα παρέχει στους χρήστες ένα εύχρηστο περιβάλλον για την αλληλεπίδραση με το blockchain.

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

1.2 Οργάνωση κειμένου

Το κείμενο της διπλωματικής αποτελείται από 8 Κεφάλαια και 2 Παραρτήματα.

Το παρόν Κεφάλαιο, το οποίο αποτελεί την εισαγωγή.

Στο Κεφάλαιο 2 παρουσιάζεται το θεωρητικό υπόβαθρο της εργασίας. Πιο συγκεκριμένα γίνεται σύντομη παρουσίαση του Web3, όπου Web3 εννοείται το κομμάτι του διαδικτύου που αφορά τις αποκεντρωμένες εφαρμογές που στηρίζονται στο blockchain και στο μέλλον αναμένουμε να διαδοθούν σε μεγάλο βαθμό. Στην συνέχεια παρουσιάζονται τα δίκτυα ομότιμων κόμβων – συστατικό στοιχείο του Web3 και του blockchain. Ύστερα παρουσιάζεται η κρυπτογραφία ελλειπτικών καμπυλών, την οποία χρησιμοποιεί το Ethereum για την ασφάλεια του δικτύου. Αμέσως μετά παρουσιάζεται αναλυτικά το Ethereum blockchain, η τεχνολογία που αποτελεί τον κορμό της παρούσας διπλωματικής. Τέλος παρουσιάζεται σύντομα ο προγραμματισμός ροών, ένα μοντέλο προγραμματισμού που δεν απαντάται συχνά και χρησιμοποιήθηκε κατά κόρον κατά την ανάπτυξη της εφαρμογής.

Στο Κεφάλαιο 3 παρουσιάζονται τα σημαντικότερα εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής. Αρχικά παρουσιάζονται εργαλεία που ανήκουν στην τεχνολογική στοίβα του Ethereum. Στην συνέχεια παρουσιάζεται το σύστημα διαχείρισης πακέτων NPM. Ακολουθεί το server-side περιβάλλον εκτέλεσης JavaScript Node.js καθώς και εργαλεία ανάπτυξης ιστοσελίδων που χρησιμοποιήθηκαν. Έπειτα παρουσιάζονται οι οργανισμοί από τους οποίους λαμβάνονταν δεδομένα καιρού πραγματικού χρόνου για την προσομοίωση των IoT αισθητήρων. Τέλος γίνεται παρουσίαση του σχεσιακού συστήματος διαχείρισης βάσεων δεδομένων MariaDB και των υπηρεσιών νέφους του ΕΔΕΤ ~Okeanos.

Στο Κεφάλαιο 4 γίνεται αναφορά στο γενικότερο πλαίσιο ένταξης της εφαρμογής και αναλύονται οι λειτουργικές και μη λειτουργικές απαιτήσεις του συστήματος.

Στο Κεφάλαιο 5 γίνεται λεπτομερής παρουσίαση της διαδικασίας της σχεδίασης και της υλοποίησης των επιμέρους συστατικών της εφαρμογής. Στην αρχή γίνεται επεξήγηση της διαδικασίας σχεδίασης και ανάπτυξης των έξυπνων συμβολαίων (smart contracts) και υπογραμμίζεται η σημασία τους. Έπειτα περιγράφεται ο τύπος των IoT αισθητήρων καιρού που το σύστημα υποστηρίζει, οι αναμενόμενες προδιαγραφές τους, ο τρόπος ένταξής τους στο γενικότερο περιβάλλον του συστήματος και ο τρόπος με τον οποίο αυτοί προσομοιώθηκαν στα πλαίσια της διπλωματικής εργασίας. Επίσης εξηγείται ο λόγος ύπαρξης των βάσεων δεδομένων. Αναλύεται ο ρόλος τους, ο τρόπος χρησιμοποίησής τους και παρουσιάζονται όλοι οι πίνακες και τα πεδία τους. Έπειτα παρουσιάζονται όλες οι ροές του Node-RED που δημιουργήθηκαν, περιγράφονται ο σκοπός τους, η διαδικασία υλοποίησής τους και ο τρόπος λειτουργίας τους. Ακολουθεί περιγραφή της διαδικασίας που ακολουθήθηκε ώστε τα συμβόλαια να γίνουν deploy στο Ropsten Test Net. Τέλος παρουσιάζονται ορισμένα ποσοτικά χαρακτηριστικά που αφορούν ζητήματα απόδοσης και κόστους των συναλλαγών των έξυπνων συμβολαίων.

Στο Κεφάλαιο 6 παρουσιάζεται ο τρόπος λειτουργίας της εφαρμογής όσον αφορά τους χρήστες της. Παρουσιάζονται δηλαδή οι ιστοσελίδες που δημιουργήθηκαν, ο σκοπός τους και η λειτουργία τους.

Το Κεφάλαιο 7 αποτελεί τον επίλογο του κειμένου. Σε αυτό συνοψίζονται οι παρατηρήσεις και τα συμπεράσματα του συγγραφέα που αφορούν τις τεχνολογίες του Ethereum blockchain και του Internet of Things. Επίσης προτείνονται ορισμένες μελλοντικές επεκτάσεις του συστήματος, των οποίων η υλοποίηση και επιτυχία εξαρτάται από την εξέλιξη των τεχνολογιών του κατανεμημένου ιστού (Web3).

Το Κεφάλαιο 8 αποτελείται από την βιβλιογραφία που χρησιμοποιήθηκε για την σύνταξη του κειμένου και την ανάπτυξη της εφαρμογής.

Το Παράρτημα Ι περιέχει αναλυτικές οδηγίες τόσο για την εγκατάσταση όσο για την χρήση της εφαρμογής. Στην πρώτη ενότητα αναφέρονται με σαφήνεια τα προαπαιτούμενα εργαλεία για την εγκατάσταση και λειτουργία της εφαρμογής, που αφορούν τον διαχειριστή της εφαρμογής. Στην δεύτερη ενότητα αναφέρονται οι απαιτήσεις από τον χρήστη, προκειμένου αυτός να είναι σε θέση να χρησιμοποιήσει την εφαρμογή.

Τέλος, στο Παράρτημα ΙΙ υπάρχει ο κώδικας των έξυπνων συμβολαίων και υπερσύνδεσμοι για τον υπόλοιπο κώδικα της εφαρμογής, ο οποίος δεν θα ήταν δυνατόν να συμπεριληφθεί στο παρόν κείμενο.

2

Θεωρητικό υπόβαθρο και σχετικές εργασίες

Στο κεφάλαιο αυτό διατυπώνεται το θεωρητικό υπόβαθρο σχετικά με την τεχνολογία του Ethereum blockchain. Παρουσιάζονται επίσης τεχνολογίες οι οποίες είναι συγγενείς του blockchain (ή αποτελούν συστατικά του). Αρχικά παρουσιάζεται η εξέλιξη του παγκόσμιου ιστού έως την σύγχρονη εποχή καθώς και η τάση που αυτήν την στιγμή υπάρχει, η οποία προστάζει την «αποκεντροποίηση» του παγκόσμιου ιστού – Web3, έτσι ώστε να υπογραμμιστεί η σημασία της τεχνολογίας του blockchain και ιδιαιτέρως των αποκεντρωμένων εφαρμογών (DApp) του Ethereum. Στην συνέχεια παρουσιάζεται η τεχνολογία των δικτύων ομότιμων κόμβων, αφού αυτά αποτελούν τον θεμέλιο λίθο για την λειτουργία του blockchain. Έπειτα γίνεται μία σύντομη παρουσίαση της κρυπτογραφίας ελλειπτικών καμπυλών, αφού είναι ο κύριος αλγόριθμος κρυπτογράφησης που χρησιμοποιείται από το Ethereum για την ασφάλεια των συναλλαγών και του δικτύου γενικότερα. Τέλος γίνεται αναφορά στον προγραμματισμό ροών, διότι αυτό το μοντέλο προγραμματισμού χρησιμοποιείται στο Node-RED, το προγραμματιστικό εργαλείο που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής.

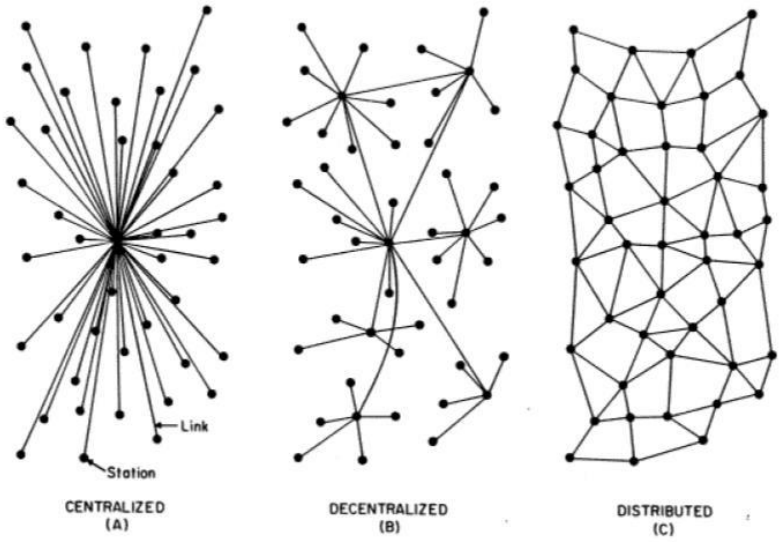
2.1 Web3 – Ο αποκεντρωμένος ιστός

Στις αρχές του 1990 ο παγκόσμιος ιστός WWW έφερε μία επανάσταση στην μετάδοση της πληροφορίας. Στην συνέχεια αναπτύχθηκε, έγινε πιο ώριμος και μετεξελίχθηκε στο λεγόμενο Web2, το οποίο έφερε μαζί του εφαρμογές όπως τα κοινωνικά δίκτυα και το ηλεκτρονικό εμπόριο. Το Web2 δεν αφορά μόνο την πληροφορία, αλλά και τις ανθρώπινες σχέσεις, οικονομικές συναλλαγές, το εμπόριο, καθώς και ορισμένες εφαρμογές P2P σε παγκόσμιο επίπεδο. Το βασικό χαρακτηριστικό του Web2 είναι πως σχεδόν σε κάθε εφαρμογή υπάρχει μία κεντρική οντότητα, αυτή μπορεί να είναι εταιρία, οργανισμός, κρατική οντότητα, η οποία συνήθως έχει τον πλήρη έλεγχο της λειτουργίας της εφαρμογής αυτής, ακόμα και στις εφαρμογές P2P συνήθως υπάρχει κάποιος ενδιάμεσος στον προαναφερθέντα ρόλο. Ο τρόπος λειτουργίας αυτός, προϋποθέτει την ύπαρξη εμπιστοσύνης από τους χρήστες προς τον οργανισμό αυτό.

Εικόνα 2.1 Χρονοδιάγραμμα εμφάνισης σημαντικών τεχνολογιών αποκέντρωσης (Decentralization Technologies)

Το παραπάνω πρόβλημα έρχεται να λύσει το Blockchain, το οποίο φαίνεται να αποτελεί τον κύριο οδηγό της νέας γενιάς του διαδικτύου, του αποκεντρωμένου ιστού «Decentralized Web» ή αλλιώς Web3 [1]. Το Blockchain παρέχει την δυνατότητα P2P συναλλαγών με πλήρη εξάλειψη του ενδιάμεσου. Πρώτη του εφαρμογή (του Blockchain), αποτελεί το Bitcoin. Βλέπουμε λοιπόν πως με την βοήθεια της τεχνολογίας αυτής καθίσταται δυνατή η δημιουργία παραδοσιακών εφαρμογών, με έναν νέο, επαναστατικό τρόπο, εξαλείφοντας τους ενδιάμεσους.

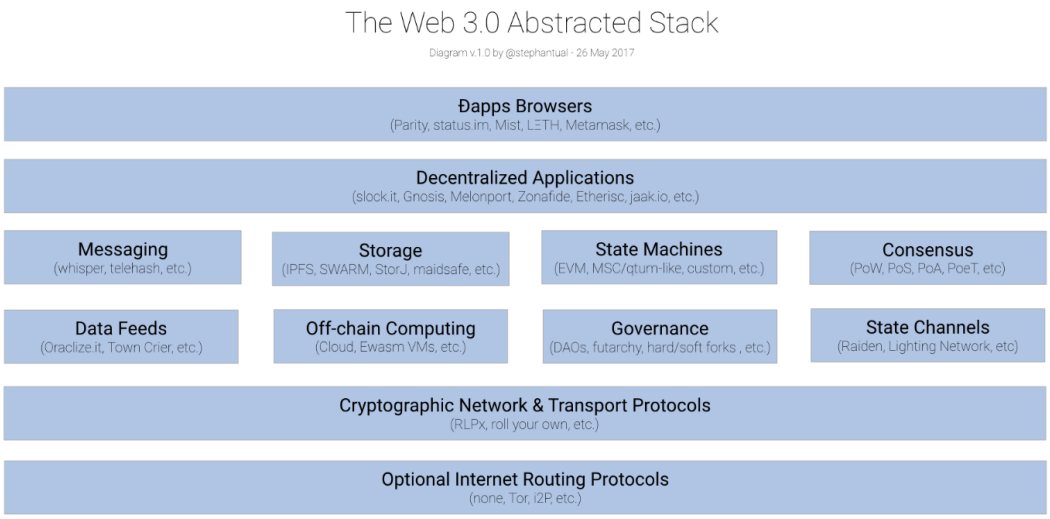
Διαπιστώνουμε πως μία νέα τάση διαμορφώνεται, η οποία επιτάσσει την μείωση της χρήσης κεντρικών εξυπηρετητών, χάριν της υιοθέτησης ενός αποκεντρωμένου μοντέλου. Με τον τρόπο αυτόν ελπίζουμε πως θα αντιμετωπίσουμε τα κακώς κείμενα του σημερινού κυρίαρχου μοντέλου πελάτη-εξυπηρετητή, που αφορούν σε μεγάλο βαθμό την ασφάλεια των δεδομένων, όπως η δυνατότητα κατάχρησής τους από τους οργανισμούς που τα έχουν αποθηκευμένα, της υποκλοπής τους, της απώλειάς τους, λόγω του μοναδικού σημείου αποτυχίας, τον εξυπηρετητή στο κέντρο δεδομένων κτλ.



Εικόνα 2.2 Web3 Centralized vs Decentralized vs Distributed

Το blockchain θα αποτελέσει θεμέλιο λίθο του Web3 [2], όμως δεν θα είναι το μοναδικό «συστατικό» του, αφού δεν ενδείκνυται για την αποθήκευση μεγάλων ποσοτήτων δεδομένων για δύο λόγους: πρώτον η επεκτασιμότητα, το blockchain είναι πολύ αργό και δύσκολα επεκτάσιμο και δεύτερον η ιδιωτικότητα, όλη η αποθηκευμένη πληροφορία στο blockchain είναι ορατή σε όλους.

Σήμερα η τεχνολογία έχει φτάσει σε τέτοιο σημείο, ώστε να είναι δυνατή η ανάπτυξη και λειτουργία μίας αποκεντρωμένης εφαρμογής «DApp» (Decentralized Application) με μικρές υπολογιστικές και αποθηκευτικές δυνατότητες, καθώς και δυνατότητες πληρωμών, παράδειγμα τέτοιας εφαρμογής αποτελεί η εφαρμογή της παρούσας διπλωματικής. Παρακάτω παρουσιάζονται ενδεικτικά μερικές τεχνολογίες του οικοσυστήματος Web3 [3].

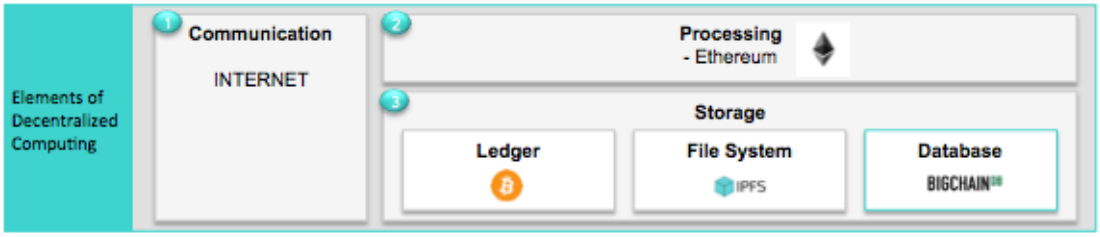


Εικόνα 2.3 Αφηρημένη στοίβα τεχνολογιών του Web3

Οι προκλήσεις για την μετάβαση από το Web2 και Web3 είναι μεγάλες, αφενός εμπορικές, οικονομικές, με τις οποίες δεν θα ασχοληθούμε στην διπλωματική αυτήν, και αφετέρου τεχνικές. Επομένως θα πρέπει να περιμένουμε μερικά χρόνια για εξάπλωση του αποκεντρωμένου ιστού, αφού πρώτα ωριμάσει αρκετά η στοίβα των τεχνολογιών που θα υποστηρίξει τις αποκεντρωμένες εφαρμογές.

Υπάρχει η πεποίθηση ότι σε βάθος χρόνου τα τρία στοιχεία της υπολογιστικής (computation) θα γίνουν όλα αποκεντρωμένα. Τα τρία αυτά στοιχεία είναι τα εξής:

- Επικοινωνία (communication)
- Επεξεργασία (processing)
- Αποθήκευση (storage)



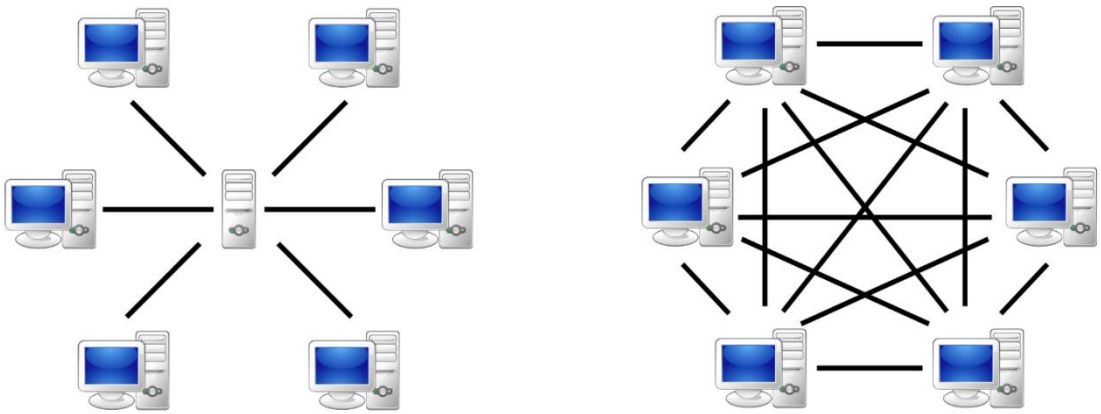
Εικόνα 2.4 Τα τρία συστατικά στοιχεία της υπολογιστικής (computation) στο αποκεντρωμένο μοντέλο

2.2 Δίκτυα ομότιμων κόμβων

Στην ενότητα αυτή, θα γίνει μία σύντομη παρουσίαση της αρχιτεκτονικής ομότιμων κόμβων, αφού αποτελεί την αρχιτεκτονική πάνω στην οποία βασίζεται η τεχνολογία του blockchain.

Στον κόσμο του διαδικτύου σήμερα δύο είναι τα κυρίαρχα μοντέλα δικτυακών εφαρμογών: η αρχιτεκτονική πελάτη-εξυπηρετητή (client-server) και η αρχιτεκτονική ομότιμων (peer-to-peer), για συντομία P2P. Στην αρχιτεκτονική πελάτη-εξυπηρετητή υπάρχει ένας πάντα ενεργός υπολογιστής, ο εξυπηρετητής, ο οποίος εξυπηρετεί αιτήσεις για υπηρεσίες από άλλους υπολογιστές, τους πελάτες. Φαίνεται λοιπόν πως ο ρόλος του εξυπηρετητή για την λειτουργία σε τέτοιου είδους δίκτυα είναι καθοριστικός.

Αντιθέτως, στην αρχιτεκτονική ομότιμων κόμβων υπάρχει μικρή ή και καμία στήριξη σε αποκλειστικούς εξυπηρετητές σε κέντρα δεδομένων [4].



Εικόνα 2.5 α. Αρχιτεκτονική πελάτη-εξυπηρετητή β. Αρχιτεκτονική ομότιμων

Η αρχή λειτουργίας των δικτύων ομότιμων κόμβων είναι η απευθείας επικοινωνία ανάμεσα σε ζεύγη κατά διαλείμματα συνδεδεμένων υπολογιστών, που καλούνται ομότιμοι (peers). Οι ομότιμοι δεν ανήκουν σε κάποιον οργανισμό, αλλά είναι κατά κανόνα υπολογιστές (ίσως και IoT συσκευές) που ελέγχονται από χρήστες. Όπως υποδηλώνει και η λέξη, οι ομότιμοι κόμβοι έχουν τα ίδια προνόμια, τις ίδιες δυνατότητες και τον ίδιο ρόλο στο δίκτυο. Οι κόμβοι λοιπόν διαθέτουν έναν μέρος των υπολογιστικών τους πόρων για την λειτουργία του δικτύου, απαλείφοντας έτσι την ανάγκη ύπαρξης μίας κεντρικής αρχής υπεύθυνης για τον συντονισμό και λειτουργία του δικτύου. Οι κόμβοι αυτοί του δικτύου βλέπουμε πως έχουν διπλό ρόλο, αυτόν του πελάτη (που προσφέρει πόρους) και αυτόν του εξυπηρετητή (που καταναλώνει πόρους). Να σημειωθεί πως πολλές φορές εφαρμογές βασίζονται σε μία υβριδική αρχιτεκτονική, συνδυάζοντας αυτήν του πελάτη-εξυπηρετητή και των ομότιμων.

Οι εφαρμογές που βασίζονται σε αρχιτεκτονικές ομότιμων περιλαμβάνουν μεταξύ άλλων, διανομή αρχείων (πχ. BitTorrent), επιτάχυνση κατεβάσματος αρχείων μέσω βοήθειας ομότιμων (πχ. Xunlei), τηλεφωνίας διαδικτύου (πχ. Skype) και IPTV (πχ. KanKan). Το blockchain, τεχνολογία πάνω στην οποία στηρίζεται μεγάλο μέρος της εφαρμογής της παρούσας διπλωματικής εργασίας, είναι τελικώς μία εφαρμογή που βασίζεται στην αρχιτεκτονική ομότιμων. Βλέπουμε πώς μία αρχιτεκτονική, η οποία έγινε κυρίως γνωστή από την εφαρμογή Napster, ενός συστήματος ανταλλαγής αρχείων (κυρίως τραγουδιών-ψηφιακών αρχείων ήχου MP3), στην συνέχεια μετεξελίχθηκε, διατηρώντας τις βασικές αρχές λειτουργίας της και πάνω της βασίστηκε μία από τις πιο πολλά υποσχόμενες τεχνολογίες του σήμερα, το blockchain.

Τα δίκτυα ομότιμων χωρίζονται σε δύο μεγάλες κατηγορίες, στα αδόμητα και στα δομημένα.

Τα αδόμητα P2P δίκτυα δεν επιβάλλουν κάποια συγκεκριμένη δομή στο ανώτερο επίπεδο του δικτύου, αλλά σχηματίζονται μέσω κόμβων οι οποίοι πραγματοποιούν συνδέσεις τυχαία μεταξύ τους (πχ. Gnutella). Δεν υπάρχει κανένας συσχετισμός μεταξύ δεδομένων και των κόμβων που τα προσφέρουν. Μερικά πλεονεκτήματά τους είναι η εύκολη δημιουργία τους, η δυνατότητα τοπικών βελτιστοποιήσεων σε διαφορετικές περιοχές της τοπολογίας και τέλος η ευρωστία τους σε συνθήκες υψηλού ρυθμού αύξησης και αναχώρησης κόμβων (churn). Ορισμένα βασικά μειονεκτήματά τους είναι ο πλημμυρισμός του δικτύου σε κάθε ερώτημα αναζήτησης δεδομένων. Το γεγονός αυτό

αφενός οδηγεί σε αυξημένη κίνηση και χρησιμοποίηση/κατασπατάληση των πόρων του δικτύου, αφετέρου δεν εξασφαλίζει την άφιξη του ερωτήματος στον κατάλληλο κόμβο και επομένως την απάντηση στο ερώτημα.

Στα δομημένα P2P δίκτυα το ανώτερο επίπεδο του δικτύου οργανώνεται σχηματίζοντας μία ορισμένη τοπολογία σύμφωνα με κάποιο πρωτόκολλο, γεγονός που εξασφαλίζει την αποδοτική αναζήτηση οποιωνδήποτε δεδομένων, από οποιονδήποτε κόμβο. Ο πιο διαδεδομένος τύπος δομημένου P2P δικτύου υλοποιεί έναν κατανεμημένο πίνακα κατακερματισμού (Distributed Hash Table – DHT), στον οποίο μέσω της χρήσης μίας παραλλαγής της «συνεπούς» συνάρτησης κατακερματισμού (consistent hashing[1]) πραγματοποιείται η ανάθεση κάθε αρχείου (δεδομένων) σε 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 η τήρηση από κάθε κόμβο λιστών γειτόνων που ικανοποιούν συγκεκριμένα κριτήρια. Η ανάγκη αυτή κάνει το δίκτυο περισσότερο ευάλωτο σε συνθήκες υψηλού ρυθμού αύξησης και αναχώρησης κόμβων (churn).

Στο blockchain του Ethereum [5] χρησιμοποιείται το πρωτόκολλο αναζήτησης γειτόνων RLPx [6], το οποίο χρησιμοποιεί Kademlia-like δρομολόγηση αναπροσαρμοσμένη ως «P2P neighbor discovery protocol». Η RLPx αναζήτηση χρησιμοποιεί 512-bit κλειδιά ως ταυτοποιητές για τους κόμβους και sha3(node-id) για xor metric.

Τα βασικότερα πλεονεκτήματα της αρχιτεκτονικής ομότιμων:

- **Αυτοκλιμακωσιμότητα** (self-scalability) **του δικτύου**, αποτελεί εγγενές χαρακτηριστικό της αρχιτεκτονικής P2P. Όσο περισσότεροι κόμβοι προστίθενται στο δίκτυο, τόσο αυξάνεται ο φόρτος εργασίας, η απαίτηση σε πόρους δηλαδή, όμως τόσο αυξάνονται και οι διαθέσιμοι πόροι, λόγω της διττής φύσης του κάθε ομότιμου κόμβου.
- **Μείωση κόστους**, καθώς συνήθως δεν απαιτείται σημαντική υποδομή και εύρος ζώνης εξυπηρετητή.
- **Μη ύπαρξη μοναδικού σημείου αστοχίας του δικτύου**. Δηλαδή η βλάβη σε έναν κόμβο δεν επηρεάζει την λειτουργία του υπόλοιπου δικτύου, όπως γίνεται στην αρχιτεκτονική πελάτη-εξυπηρετητή, όπου αστοχία του εξυπηρετητή, συνεπάγεται μη διαθεσιμότητα της εφαρμογής.

Μερικές από τις βασικές προκλήσεις που αντιμετωπίζουν οι εφαρμογές αρχιτεκτονικής ομότιμων:

- **Μη φιλικότητα προς τους ISP.** Οι περισσότεροι ISP έχουν διασταθιοποιηθεί για ασύμμετρη χρησιμοποίηση του εύρους ζώνης, δηλαδή για περισσότερη συρρευματική, παρά αντιρρευματική κίνηση, δυσκολεύοντας έτσι την παροχή πόρων προς το σύστημα από τους ομότιμους κόμβους.

[1] consistent hashing: Ένα ειδικό είδος κατακερματισμού, σύμφωνα με το οποίο όταν ένας πίνακας κατακερματισμού αλλάξει μέγεθος, μόνον K/n τα κλειδιά πρέπει να ξανά υπολογιστούν κατά μέσο όρο, όπου K είναι ο αριθμός των κλειδιών και n ο αριθμός των θέσεων. Σε αντίθεση με τους παραδοσιακούς πίνακες κατακερματισμού όπου αλλαγή του αριθμού των θέσεων συνεπάγεται επανυπολογισμό σχεδόν όλων των κλειδιών.

- **Ασφάλεια.** Λόγω της κατανεμημένης και ανοικτής φύσης τους, οι εφαρμογές P2P μπορούν να δημιουργήσουν προβλήματα ασφαλείας. Θα δούμε στην συνέχεια πώς στο blockchain το πρόβλημα αυτό λύνεται με την χρήση της κρυπτογραφίας.
- **Κίνητρα.** Ποια κίνητρα έχουν οι κόμβοι, ώστε να παρέχουν πόρους στο σύστημα. Και πάλι θα δούμε πως στα blockchain κρυπτονομισμάτων (Ethereum, Bitcoin) παρέχουν (χρηματικά) κίνητρα στους κόμβους που υποστηρίζουν την λειτουργία του δικτύου (miners).

2.3 Κρυπτογραφία ελλειπτικών καμπυλών

Το Ethereum (όπως και το Bitcoin) χρησιμοποιεί την κρυπτογραφία ελλειπτικών καμπυλών [7] για την υπογραφή των συναλλαγών, συμβάλλοντας έτσι στην ασφάλεια των 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 [8]. 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.

Δεν θα αναλυθεί ο τρόπος λειτουργίας του αλγορίθμου της κρυπτογραφίας ελλειπτικών καμπυλών, αφού αφορά ένα πολύ εξειδικευμένο πεδίο το οποίο δεν εμπίπτει στο εύρος της παρούσας διπλωματικής. Λεπτομέρειες για τον τρόπο λειτουργίας μπορούν να βρεθούν στις εξής πηγές: [9] και [10].

Θεωρείται ότι σήμερα, ο αλγόριθμος ψηφιακής υπογραφής ελλειπτικών καμπυλών (Elliptic Curve Digital Signature Algorithm – ECDSA), που χρησιμοποιεί Ethereum για κρυπτογράφηση, είναι απαραβίαστος, έτσι λοιπόν διασφαλίζεται η ασφάλεια των συναλλαγών και θεωρούμε ότι το Ethereum είναι θωρακισμένο απέναντι σε επιθέσεις. Σε περίπτωση που υπάρχει μεγάλη ανάπτυξη στην κβαντική υπολογιστική (Quantum Computing), ο αλγόριθμος αυτός θα σταματήσει να προσφέρει ασφάλεια. Για την αντιμετώπιση αυτού του ενδεχομένου, αναμένεται να παρουσιαστούν κάποιες λύσεις (Lamport signatures), στην έκδοση Constantinople [7].

2.4 Ethereum Blockchain

Ερμηνεία: Το blockchain είναι ένα ψηφιακό, κατανεμημένο, δημόσιο καθολικό (ledger), μέσω του οποίου καταγράφονται με ασφάλεια συναλλαγές, συμφωνίες, συμβόλαια και γενικώς οτιδήποτε χρειάζεται να έχει καταγραφεί και να είναι διαθέσιμο προς επαλήθευση.

Ερμηνεία: Κατανεμημένο σύστημα είναι μία συλλογή από ανεξάρτητους υπολογιστές, οι οποίοι εμφανίζονται στους χρήστες τους ως ένα ενιαίο συνεκτικό σύστημα [11].

Ερμηνεία: Κρυπτονόμισμα είναι ένα ψηφιακό νόμισμα το οποίο ασφαλίζει τις συναλλαγές με κρυπτογραφικό κώδικα, που βασίζεται στην υπολογιστική ισχύ του

hardware για την εκτέλεσή του (proof of work) ή λιγότερο ενεργειακά απαιτητικούς τρόπους, όπως το proof of stake [7].

Όπως αναφέρθηκε και προηγουμένως, το blockchain είναι μία από τις τεχνολογίες που αποτελούν τον κορμό ανάπτυξης του Web3, της νέας γενιάς του διαδικτύου. Αποτελεί μία πολύ αποτελεσματική λύση στο πρόβλημα της εμπιστοσύνης που αντιμετωπίζουν οι άνθρωποι (όπως αυτό αναφέρεται στην προηγούμενη ενότητα). Η τεχνολογία του blockchain μας δίνει την δυνατότητα να εμπιστευόμαστε τις εξόδους του δικτύου, χωρίς παράλληλα να εμπιστευόμαστε κανέναν από τους συμμετέχοντες του αυτού.

Το blockchain λειτουργεί στο διαδίκτυο, πάνω σε ένα δίκτυο ομότιμων (P2P) κόμβων που «τρέχουν» το πρωτόκολλο και διατηρούν ένα πιστό αντίγραφο του καθολικού των συναλλαγών/δεδομένων, επιτρέποντας την πραγματοποίηση συναλλαγών χωρίς την παρουσία κάποιου έμπιστου ενδιάμεσου, παρά μόνο με την βοήθεια των υπολοίπων κόμβων του δικτύου. Το blockchain καθαυτό είναι ένα αρχείο, το οποίο εμπεριέχει ένα καθολικό όλων των συναλλαγών που έχουν πραγματοποιηθεί.

Στην συνέχεια θα εξετάσουμε το Ethereum και θα αναδείξουμε τις αρετές του έναντι του Bitcoin, του κρυπτονομίσματος που «γέννησε» την τεχνολογία του blockchain [12].

Η καινοτομία του Ethereum σε σχέση με το Bitcoin, είναι πως το Ethereum αποτελεί μία πιο ευέλικτη και προσαρμόσιμη πλατφόρμα, πάνω στην οποία μπορούν να δημιουργηθούν και να λειτουργήσουν με ασφάλεια αποκεντρωμένες εφαρμογές, ενώ το Bitcoin παρέχει κυρίως την δυνατότητα (οικονομικών) συναλλαγών του κρυπτονομίσματος (Bitcoin) [13].

Το blockchain του Ethereum είναι μία Turing complete κατανεμημένη υπολογιστική αρχιτεκτονική, στην οποία κάθε κόμβος του δικτύου εκτελεί και καταγράφει τις ίδιες συναλλαγές, οι οποίες οργανώνονται σε μπλοκ και προστίθενται στο blockchain. Μόνο ένα μπλοκ μπορεί να προστεθεί κάθε φορά και κάθε μπλοκ περιέχει την απόδειξη εργασίας (Proof of Work), αν και λόγω προβλημάτων που έχουν προκύψει (μεγάλη κατανάλωση ισχύος) συζητείται η μετάβαση από το Proof of Work στο Proof of Stake. Οι κόμβοι που συντηρούν το δίκτυο, δηλαδή αυτοί που δημιουργούν τα μπλοκ ονομάζονται miners.

Το Ethereum Virtual Machine

Όπως αναφέρθηκε και πριν, το Ethereum δεν παρέχει στους χρήστες απλά ένα προκαθορισμένο σύνολο λειτουργιών, όπως κάνει το Bitcoin, αλλά αντιθέτως τους παρέχει την δυνατότητα να ορίσουν δικές τους λειτουργίες και κατ' επέκταση έξυπνα συμβόλαια και αποκεντρωμένες εφαρμογές DApps, είναι δηλαδή ένα προγραμματιζόμενο blockchain.

Το Ethereum είναι κατά μία έννοια μία σουίτα πρωτοκόλλων που καθορίζουν μία πλατφόρμα για την ανάπτυξη και λειτουργία αποκεντρωμένων εφαρμογών. Στο επίκεντρο βρίσκεται το Ethereum Virtual Machine ("EVM"), το οποίο μπορεί να εκτελεί κώδικα αυθαίρετης αλγοριθμικής πολυπλοκότητας. Το EVM είναι Turing complete.

Όπως και τα άλλα blockchain, το Ethereum περιλαμβάνει ένα πρωτόκολλο δικτύου ομότιμων κόμβων. Το πρωτόκολλο αυτό είναι υπεύθυνο για τον συντονισμό των συνδεδεμένων κόμβων, με σκοπό την απρόσκοπτη λειτουργία του δικτύου. Κάθε κόμβος «τρέχει» το EVM και εκτελεί τις ίδιες εντολές. Λόγω αυτού, το Ethereum αναφέρεται συχνά και ως «παγκόσμιος υπολογιστής».

Η μεγάλη αυτή παραλληλοποίηση των υπολογισμών δεν έχει σκοπό την απόδοση, αφού οι υπολογισμοί είναι πολύ πιο αργοί και ακριβοί απ' ότι θα γινόταν σε έναν απλό υπολογιστή, σκοπό έχει την ύπαρξη ομοφωνίας/συναίνεσης (consensus). Αυτό το χαρακτηριστικό δίνει στο Ethereum πολύ μεγάλη αντοχή σε σφάλματα, αδιάλειπτη λειτουργία και εγγυάται ότι τα δεδομένα μέσα στο blockchain θα μείνουν για πάντα (σχεδόν) αμετάβλητα.

Ο τρόπος λειτουργίας του Ethereum

Η βασική μονάδα στο Ethereum είναι ο λογαριασμός (account), σε αντίθεση με το Bitcoin blockchain όπου βασική μονάδα είναι η συναλλαγή. Στο Ethereum blockchain, παρακολουθείται η κατάσταση κάθε λογαριασμού και όλες οι μεταβάσεις κατάστασης είναι μεταβιβάσεις αξίας και πληροφορίας μεταξύ λογαριασμών. Υπάρχουν δύο είδη λογαριασμών:

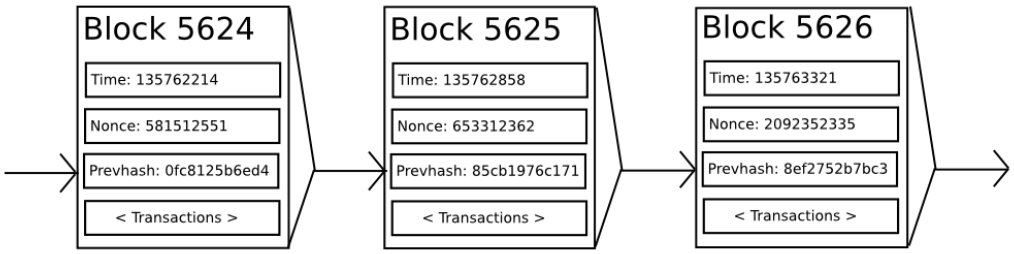
- Οι Externally Owned λογαριασμοί (EOAs), οι οποίοι ελέγχονται από ιδιωτικά κλειδιά
- Οι λογαριασμοί συμβολαίων, οι οποίοι ελέγχονται από τον κώδικα του συμβολαίου και μπορούν να ενεργοποιηθούν μόνο από έναν EOA.

Δηλαδή, οι EOA ελέγχονται από τους ανθρώπους που κατέχουν και ελέγχουν τα ιδιωτικά κλειδιά, ενώ οι λογαριασμοί συμβολαίων ελέγχονται από τον κώδικά τους. Το πολυσυζητημένο έξυπνο συμβόλαιο «smart contract» είναι πρακτικά ο κώδικας του λογαριασμού συμβολαίου, δηλαδή το πρόγραμμα που εκτελείται όταν μία συναλλαγή σταλεί στον λογαριασμό αυτόν. Οι χρήστες που κατέχουν EOA, μπορούν να δημιουργήσουν νέα συμβόλαια, δημοσιεύοντάς τα στο blockchain.

Οι χρήστες (λογαριασμοί EOA) στέλνουν συναλλαγές στο δίκτυο του Ethereum, υπογράφοντας τα δεδομένα της συναλλαγής με το ιδιωτικό τους κλειδί, χρησιμοποιώντας την κρυπτογραφία ελλειπτικών καμπυλών (ECDSA – Elliptic Curve Digital Signature Algorithm). Μία συναλλαγή είναι έγκυρη μόνο εάν είναι υπογεγραμμένη από τον αποστολέα της (από το ιδιωτικό του κλειδί). Σαν αποτέλεσμα, το δίκτυο είναι σίγουρο ότι ο αποστολέας της συναλλαγής είναι αυτός που ισχυρίζεται και όχι κάποιος κακόβουλος χρήστης.

Για κάθε συναλλαγή θα πρέπει να πληρωθεί ένα μικρό τέλος στο δίκτυο (gas). Αυτό προστατεύει το δίκτυο, αφού κάνει ασύμφορες τις επιπόλαιες εντολές υπολογισμού καθώς και κακόβουλες επιθέσεις, όπως τις επιθέσεις άρνησης υπηρεσίας (DDoS). Η πληρωμή γίνεται για τον υπολογισμό και την μνήμη που χρησιμοποιεί η πραγματοποίηση της συναλλαγής και είναι ανάλογη αυτών. Το τέλος αυτό (gas) πληρώνεται στο κρυπτονόμισμα του Ethereum, το ether.

Τα ανωτέρω τέλη εισπράττονται από τους κόμβους που επικυρώνουν το δίκτυο, τους miners. Οι miners είναι κόμβοι του δικτύου που λαμβάνουν, διαδίδουν, επικυρώνουν και εκτελούν συναλλαγές. Συγκεντρώνουν ορισμένες συναλλαγές κάθε φορά σε μπλοκ και στη συνέχεια ανταγωνίζονται τους υπόλοιπους miners ώστε αυτοί να το εισάγουν στο blockchain. Κάθε φορά που ένας miner εισαγάγει ένα νέο μπλοκ στο blockchain λαμβάνει τα ether που αντιστοιχούν στις συναλλαγές που περιέχει το μπλοκ. Αυτό το ποσόν είναι που δίνει κίνητρο στους miners να εκτελούν την εργασία αυτή.



Εικόνα 2.6 Δομή των μπλοκ [14]

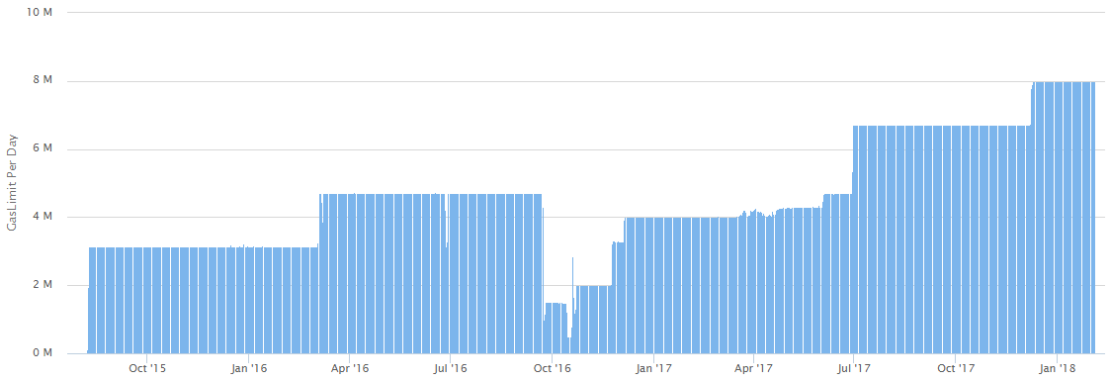
Για να εισαχθεί ένα μπλοκ στο blockchain θα πρέπει να συνοδεύεται από την απόδειξη εργασίας (nonce). Αυτό αποτελεί την λύση ενός δύσκολου μαθηματικού προβλήματος, όπως γίνεται και στο Bitcoin. Η διαφοροποίηση του Ethereum έγκειται στο γεγονός ότι το πρόβλημα αυτό έχει μεγάλες απαιτήσεις μνήμης, έτσι για το «mining» είναι απαραίτητα και η CPU και η μνήμη, ενώ στο Bitcoin αυτό δεν γίνεται (απαραίτητη μόνο κάρτα γραφικών). Η απόφαση αυτή των σχεδιαστών του Ethereum οδηγεί σε ένα πιο αποκεντρωμένο δίκτυο, αφού αποθαρρύνει την χρήση ειδικού hardware (πχ. ASICs), όπως έγινε με το Bitcoin. Το αποτέλεσμα λοιπόν, είναι ότι η απόδειξη εργασίας (PoW – Proof of Work) του Ethereum είναι πιο ανθεκτική απέναντι σε ASICs, οδηγώντας σε μεγαλύτερη αποκεντροποίηση του δικτύου, δηλαδή σε μεγαλύτερη ασφάλεια· πολλοί και μικροί miners, παρά λίγοι και ισχυροί [15].

Η επικοινωνία μεταξύ των ομότιμων κόμβων που «τρέχουν» τον Ethereum client γίνεται σύμφωνα με το πρωτόκολλο ÐΞVp2p Wire Protocol [6], [16]. Δεν θα αναλυθεί ο τρόπος λειτουργίας του πρωτοκόλλου, διότι αφορά πολύ εξειδικευμένα ζητήματα τα οποία δεν εμπίπτουν στο εύρος της διπλωματικής εργασίας.

Το κρυπτονόμισμα Ether του Ethereum έχει σαν ελάχιστη υποδιαίρεση το Wei. Ένα ether ισούται με 10^{18} wei.

Ο μηχανισμός παραγωγής των μπλοκ

Στο Ethereum blockchain τα μπλοκ δεν έχουν σταθερό μέγεθος ούτε δημιουργούνται ανά τακτά χρονικά διαστήματα. Αυτό που έχουν είναι όριο στο gas κάθε μπλοκ. Το gas limit περιορίζει και το μέγεθος του μπλοκ αλλά και την υπολογιστική ισχύ που απαιτείται για την δημιουργία του κάθε μπλοκ. Το όριο του gas για κάθε μπλοκ προκύπτει μετά από ψηφοφορία μεταξύ των miner, δηλαδή μπορεί να αλλάζει με την πάροδο του χρόνου, έτσι μπορεί αλλάζει και το μέγεθος του μπλοκ. Το gas είναι η μονάδα που χρησιμοποιεί το Ethereum για την μέτρηση της υπολογιστικής προσπάθειας, για παράδειγμα η πρόσθεση δύο αριθμών κοστίζει 3 gas, ο υπολογισμός της τιμής κατακερματισμού κοστίζει 30 gas κτλ. [17].



Εικόνα 2.7 Gas limit μπλοκ (Πηγή: etherscan.io)

Εικόνα 2.8 Μέγεθος μπλοκ (Πηγή: etherscan.io)

Το χρονικό διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών μπλοκ δεν είναι σταθερό και εξαρτάται από το επίπεδο της δυσκολίας του δικτύου. Ισχύουν οι εξής σχέσεις:

$$block\_time = current\_block\_timestamp - parent\_block\_timestamp$$

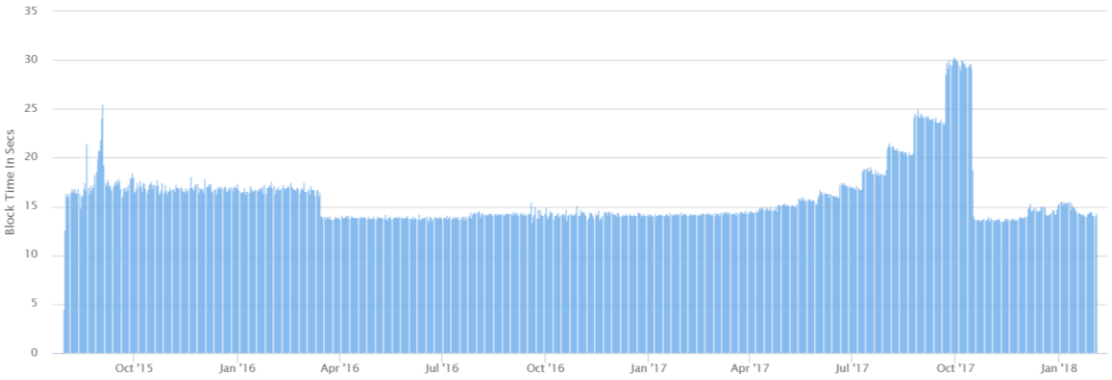
$$currentBlockDifficulty = parentBlockDifficulty + \frac{parentBlockDifficulty}{2048} * \max\left[\left(1 - \frac{blockTime}{10}\right), -99\right] + \text{floor}\left(\frac{currentBlockNumber}{100000} - 2\right)$$

Όπου floor είναι ο μεγαλύτερος ακέραιος αριθμός που είναι μικρότερος από τον περιεχόμενο αριθμό. Επίσης όλες οι διαιρέσεις είναι ακέραιες.

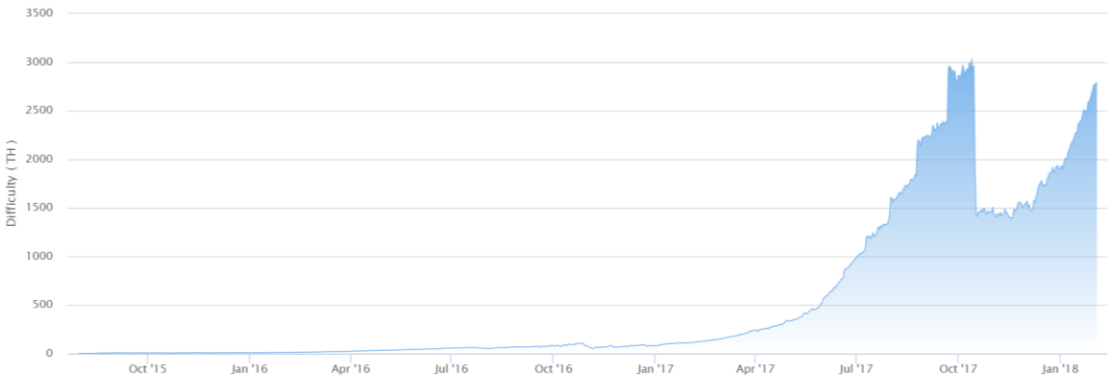
Από την παραπάνω σχέση βλέπουμε ότι εάν ο χρόνος που μεσολαβεί της δημιουργίας δύο διαδοχικών μπλοκ είναι μικρότερος από 10 δευτερόλεπτα η δυσκολία θα αυξηθεί. Εάν αυτός ο χρόνος είναι από 10 έως 19 δευτερόλεπτα η δυσκολία δεν θα μεταβληθεί (αν εξαιρέσουμε τον παράγοντα difficulty bomb). Ενώ εάν είναι μεγαλύτερος από 19 δευτερόλεπτα η δυσκολία θα μειωθεί.

Το τελευταίο κομμάτι (floor(…)) ονομάζεται difficulty bomb και η ύπαρξή του έχει σαν αποτέλεσμα την σταδιακή αύξηση της δυσκολίας παράλληλα με την αύξηση του αριθμού των μπλοκ.

Με την παραπάνω τεχνική ο χρόνος του κάθε μπλοκ σχεδόν παραμένει μέσα στο προκαθορισμένο διάστημα 10 έως 19 sec.



Εικόνα 2.9 Χρονικό διάστημα μεταξύ δύο μπλοκ (Πηγή: etherescan.io)



Εικόνα 2.10 Δυσκολία μπλοκ (Πηγή: etherscan.io)

2.5 Προγραμματισμός ροών

Το Node-RED, το προγραμματιστικό εργαλείο που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής, βασίζεται στην αρχή του προγραμματισμού βάσει ροής (Flow–based programming). Η αρχή του προγραμματισμού βάσει ροής (Flow-based programming), είναι ένα μοντέλο προγραμματισμού, σύμφωνα με το οποίο μία εφαρμογή είναι ένα δίκτυο από διεργασίες – μαύρα κουτιά – κόμβους οι οποίες ανταλλάσσουν δεδομένα μέσω προκαθορισμένων συνδέσεων. Κάθε κόμβος επιτελεί μία καλώς ορισμένη διεργασία, λαμβάνει δεδομένα στην είσοδό του, τα επεξεργάζεται και δίνει δεδομένα στην έξοδό του. Δηλαδή τα δεδομένα ρέουν από τον έναν κόμβο στον άλλον, χαρακτηριστική είναι η πασίγνωστη αρχαιοελληνική έκφραση «Πάντα ῥεῖ», που ο ίδιος ο J. Paul Rodker-Morrison παραθέτει [18]. Το δίκτυο είναι υπεύθυνο για την ανταλλαγή των δεδομένων αυτών. Το μοντέλο αυτό είναι κατάλληλο για την οπτική απεικόνιση της λειτουργικότητας της εφαρμογής και συνεπώς κάνει τις διάφορες εφαρμογές προσιτές σε μεγάλη ομάδα χρηστών. Κάποιος που δύναται να διασπάσει ένα πρόβλημα σε επιμέρους υπό προβλήματα, δύναται να κατανοήσει την αντίστοιχη ροή, χωρίς να χρειάζεται να καταλάβει τις επιμέρους γραμμές κώδικα που περιέχει ο κάθε κόμβος. Το Node-RED, επειδή ακριβώς βασίζεται σε ροές, δίνει την δυνατότητα σύνδεσης διαφορετικών συσκευών, διεπαφών (APIs) και διαδικτυακών υπηρεσιών με νέους επαναστατικούς τρόπους.

2.6 Σχετικές εργασίες

Στην ενότητα αυτήν 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 το Internet of Things και την χρήση του για αγοραπωλησίες δεδομένων αισθητήρων.

Εργασία [19]

Η ανωτέρω είναι μία εργασία που πραγματεύεται την έννοια του Sensing-as-a-Service (S^2aaS) με την χρήση του Bitcoin, ένα επιχειρηματικό μοντέλο για το IoT.Αγγίζει το θέμα της αγοραπωλησίας δεδομένων έξυπνων αισθητήρων, δίνοντας ως παράδειγμα αισθητήρες καιρού, με την χρήση της τεχνολογίας του Bitcoin blockchain ήδη από το 2014. Κατά την συγγραφή της εργασίας αυτής, το πιο διαδεδομένο blockchain ήταν αυτό του Bitcoin, το Ethereum βρισκόταν ακόμη στην φάση της δημιουργίας.

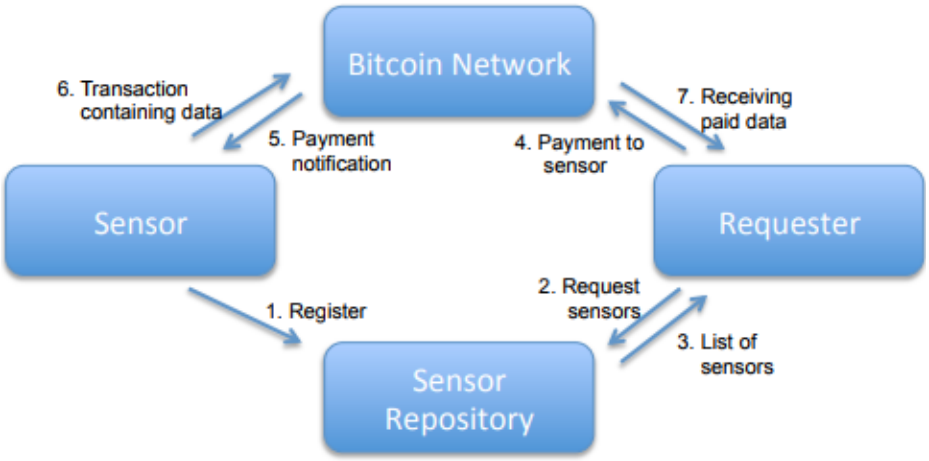
Αρχικά παρουσιάζεται το μοντέλο του S^2aaS, η θέση του στην σύγχρονη οικονομία. Στην συνέχεια παρουσιάζεται σύντομα το Bitcoin blockchain και υπογραμμίζονται χαρακτηριστικά του σχετικά με τις S^2aaS εφαρμογές. Έπειτα δίνεται σύντομα ένα παράδειγμα τέτοιας εφαρμογής και εξηγείται η βασική ιδέα. Τέλος παρουσιάζονται τα βασικά χαρακτηριστικά που προσδίδει το Bitcoin σε τέτοιου είδους εφαρμογές.

Εργασία [20]

Αποτελεί κατά κάποιον τρόπο την συνέχεια της προηγούμενης εργασίας [19], αφού σε αυτήν, περιγράφεται ένα μοντέλο του πρωτοτύπου που έχει υλοποιηθεί για την αγοραπωλησία των δεδομένων. Αποτελείται από τρία διακριτά στοιχεία:

- Sensor Client: Ο client του αισθητήρα. Θα πρέπει να είναι σε θέση να εντοπίζει τις συναλλαγές με τις οποίες ζητούνται τα δεδομένα του και στην συνέχεια να στέλνει τα ζητηθέντα δεδομένα.
- Requester Client: Ένας client που αγοράζει τα δεδομένα. Θα πρέπει να είναι σε θέση να στείλει στο Bitcoin δίκτυο συναλλαγές με τις οποίες θα αγοράζει δεδομένα από τους αισθητήρες.
- Sensor Repository: Μία «αποθήκη» αισθητήρων, στην οποία θα καταχωρούνται οι διαθέσιμοι αισθητήρες. Κάθε καταχώριση θα πρέπει να περιέχει τουλάχιστον την διεύθυνση Bitcoin του αισθητήρα, τα δεδομένα που αυτός προσφέρει, την τιμή τους και μετά-δεδομένα όπως την τοποθεσία, κτλ.

Τα τρία στοιχεία αυτά αλληλοεπιδρούν μεταξύ τους και με το Bitcoin blockchain όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 2.11 Αγορά δεδομένων IoT αισθητήρων με Bitcoin

Στην συνέχεια αναλύεται ο τρόπος υλοποίησης και λειτουργίας του πρωτοτύπου.

Εάν κάποιος διαβάσει την εργασία [20] και στην συνέχεια συγκρίνει την υλοποίηση του συστήματος με αυτήν της παρούσας διπλωματικής εργασίας, εύκολα θα παρατηρήσει πόσο πιο αποδοτικά μπορούν τέτοιου είδους εφαρμογές να αναπτυχθούν πια και να λειτουργήσουν χάρη στην ραγδαία πρόοδο της τεχνολογίας του blockchain και την εισαγωγή σε αυτό της έννοιας των έξυπνων συμβολαίων. Αυτό διότι στην παρούσα διπλωματική πολλές από τις διαφορετικές λειτουργικότητες που περιγράφονται στην εργασία [20] ενσωματώνονται μέσω των έξυπνων συμβολαίων στο Ethereum δίκτυο.

Εργασία [21]

Η εργασία αυτή εξετάζει τους τρόπους με τους οποίους το blockchain μπορεί να συνδυαστεί με το Internet of Things.

Στο πρώτο κομμάτι του κυρίως μέρους γίνεται παρουσίαση του τρόπου λειτουργίας και των συστατικών του blockchain (consensus, smart contracts, κτλ.).

Στο δεύτερο κομμάτι του κυρίως μέρους εξετάζεται ο συνδυασμός του blockchain με το IoT. Παρουσιάζονται προβλήματα που αφορούν κυρίως τα υψηλά κόστη επεκτασιμότητας και συντήρησης των δικτύων των IoT αισθητήρων και για την λύση τους προτείνεται η χρήση του blockchain. Παρουσιάζονται επίσης λοιπά πλεονεκτήματα που προκύπτουν από την σύνδεση των IoT αισθητήρων στο blockchain, όπως είναι η ύπαρξη έτοιμων μεθόδων πληρωμών, η οποία προετοιμάζει το έδαφος για μία αγορά υπηρεσιών μεταξύ συσκευών (marketplace of services between devices). Γίνεται μνεία για τα εργαλεία και τις εφαρμογές του Ethereum· κάτι που δείχνει τον ρόλο του ως ένα από τα κορυφαία blockchain για την ανάπτυξη αποκεντρωμένων εφαρμογών.

Στο τρίτο κομμάτι του κυρίως μέρους παρουσιάζονται ορισμένα θέματα που μπορούν να σταθούν εμπόδιο στον συνδυασμό του blockchain με το IoT και παράλληλα, προτείνονται λύσεις, όπου αυτό είναι δυνατό. Τέτοια θέματα αφορούν την απόδοση του δικτύου και καθυστερήσεις των συναλλαγών, την ιδιωτικότητα, την εμπιστευτικότητα, την ουδετερότητα των miners, την νομική ισχύ των έξυπνων συμβολαίων, την διακύμανση της τιμής των κρυπτονομισμάτων, την πλήρη αυτονομία λειτουργίας των έξυπνων συμβολαίων και άλλα θέματα.

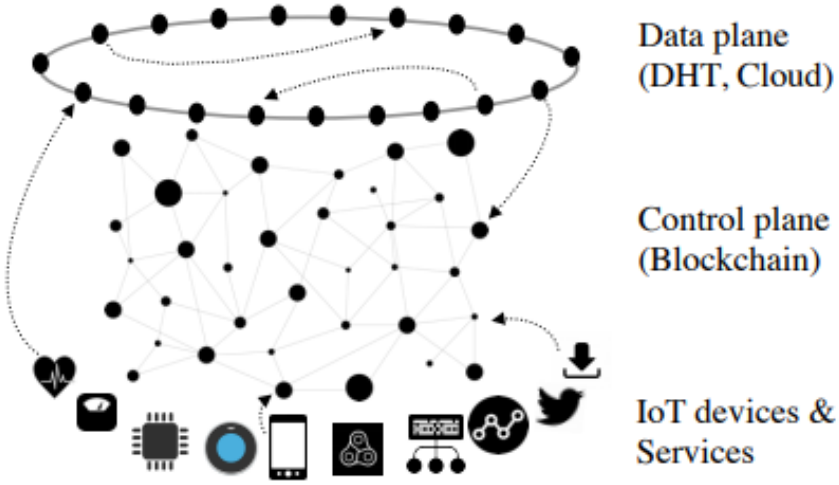
Το συμπέρασμα των συγγραφέων είναι ότι ο συνδυασμός του blockchain με το Internet of Things είναι ισχυρός και μπορεί να προκαλέσει μεγάλες αλλαγές σε διαφορετικές βιομηχανίες, οδηγώντας σε νέα επιχειρησιακά μοντέλα και δημιουργώντας κατανεμημένες εφαρμογές.

Εργασία [22]

Η εργασία αυτή εξετάζει τον συνδυασμό του blockchain με το Internet of Things υπό το πρίσμα της ασφάλειας. Αναλύονται τρόποι με τους οποίους το blockchain μπορεί να ενσωματωθεί στο IoT και η προσφορά που αυτό θα έχει σε θέματα ασφάλειας, εστιάζοντας κυρίως σε θέματα που αφορούν την εφοδιαστική αλυσίδα (supply chain). Ακολουθεί μία σύγκριση του μοντέλου νέφους, που ακολουθείται στο IoT, με το blockchain. Τέλος, συνοψίζεται η ασφάλεια που ο συνδυασμός του blockchain με τους IoT αισθητήρες μπορούν να προσφέρουν στην εφοδιαστική αλυσίδα.

Εργασία [23]

Στην εργασία αυτήν εντοπίζονται τα προβλήματα που δημιουργούνται από το μοντέλο του νέφους που σήμερα χρησιμοποιείται για το Internet of Things και προτείνεται ένα νέο μοντέλο (σχέδιο), βασισμένο στο blockchain, λειτουργίας του IoT, για την επίλυση των προβλημάτων αυτών. Το πρόβλημα με το νέφος είναι η απομόνωση των δεδομένων των IoT αισθητήρων, γεγονός που εμποδίζει την δυνατότητα πλήρους αξιοποίησής τους. Το νέο μοντέλο που προτείνεται από την εργασία αυτή, σκοπό έχει την «αποκεντροποίηση» του ελέγχου των δεδομένων. Με το μοντέλο αυτό, τα δεδομένα να διαμοιράζονται με τρόπο ασφαλή και αποδοτικό ώστε να επιτυγχάνεται η μέγιστη χρησιμότητά τους. Στην ενότητα 3 παρουσιάζεται αναλυτικά ο τρόπος λειτουργίας του συστήματος.



Εικόνα 2.12 Κρυπτογραφημένη, αποκεντρωμένη, βασισμένη στο blockchain αποθήκη δεδομένων

Εργασία [24]

Αυτή η εργασία πραγματεύεται το θέμα της διαχείρισης Internet of Things αισθητήρων μέσω του Ethereum blockchain. Αναφέρεται πως στην επιλογή του Ethereum σημαντικό ρόλο έπαιξε η δυνατότητα ανάπτυξης έξυπνων συμβολαίων που «τρέχουν» στο blockchain.

Δημιουργήθηκε ένα πρωτότυπο αποτελούμενο από ένα smartphone και τρία Raspberry Pis, τα οποία χρησιμοποιήθηκαν σαν μετρητές της κατανάλωσης ενέργειας των συσκευών. Οι IoT μετρητές τρέχουν κόμβο Ethereum blockchain και καταχωρίζουν σε αυτό τις μετρήσεις που πραγματοποιούν μέσω των έξυπνων συμβολαίων που έχουν δημιουργηθεί για τον σκοπό αυτό.

Η εργασία αυτή μας ενδιαφέρει όχι για τον τύπο των αισθητήρων, αλλά διότι δείχνει έναν τρόπο σύνδεσης IoT αισθητήρων στο Ethereum blockchain. Έτσι λοιπόν θα μπορούσαμε και εμείς στην εφαρμογή της παρούσας διπλωματικής να θεωρήσουμε μία μελλοντική επέκταση, στην οποία οι αισθητήρες δεν θα καταχωρίζονταν από τους ιδιοκτήτες τους στο Ethereum blockchain, αλλά αυτό θα γινόταν αυτόματα από τους ίδιους τους αισθητήρες, όπως και η διαχείριση των εσόδων από την πώληση των μετρήσεων, οδηγώντας σε ακόμη μεγαλύτερη αυτοματοποίηση.

Εφαρμογή [25]

Η αποκεντρωμένη αυτή εφαρμογή έχει σχεδόν πανομοιότυπο χαρακτήρα με την εφαρμογή της παρούσας διπλωματικής. Βρίσκεται ακόμη στο στάδιο της ανάπτυξης και έπεται η ICO (Initial Coin Offering). Σύμφωνα με τους δημιουργούς της, θα λειτουργήσει το δεύτερο τρίμηνο του 2018 και θα αποτελέσει μία παγκόσμια πλατφόρμα αγοραπωλησιών δεδομένων Internet of Things αισθητήρων. Όπως και στην εφαρμογή της παρούσας διπλωματικής, η πληρωμές των δεδομένων θα γίνονται μέσω του blockchain, με την χρήση κάποιου κρυπτονομίσματος. Χαρακτηριστικά, αναφέρεται ως το eBay για τα δεδομένα IoT αισθητήρων. Στόχος της είναι να μεγιστοποιήσει την χρησιμότητα των δεδομένων των IoT αισθητήρων, να προσφέρει έσοδα στους ιδιοκτήτες τους και παράλληλα να προσφέρει στους αγοραστές τα δεδομένα που χρειάζονται σε χαμηλές τιμές.

3

Εργαλεία και τεχνολογίες

Στο κεφάλαιο αυτό παρουσιάζονται τα κυριότερα εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής της παρούσας διπλωματικής εργασίας. Πιο συγκεκριμένα παρουσιάζονται ορισμένα εργαλεία που προσφέρει το Ethereum για την ανάπτυξη, λειτουργία και πρόσβαση στις αποκεντρωμένες εφαρμογές (DApps), τα Geth, Ganache CLI, Ethereumjs, Web3, Solidity, Metamask. Στην συνέχεια παρουσιάζεται το πρόγραμμα διαχείρισης πακέτων NPM. Έπειτα παρουσιάζεται το προγραμματιστικό εργαλείο Node-RED. Ακολούθως παρουσιάζεται η server-side πλατφόρμα Node.js, την οποία και χρησιμοποιεί το Node-RED. Μετά παρουσιάζονται εργαλεία για την ανάπτυξη ιστοσελίδων. Έπειτα παρουσιάζονται οι ιστοσελίδες από τις οποίες ελήφθησαν καιρικά δεδομένα, για την προσομοίωση του πωλητή των αισθητήρων. Στην συνέχεια παρουσιάζεται το σύστημα βάσεων δεδομένων MariaDB. Τέλος, παρουσιάζεται η υπηρεσία νέφους ~Okeanos, η οποία φιλοξενεί τον εξυπηρετητή της εφαρμογής.

3.1 Ethereum

Το οικοσύστημα του Ethereum παρέχει στους προγραμματιστές πολλά εργαλεία για την ανάπτυξη αποκεντρωμένων εφαρμογών (DApps), στην συνέχεια παρουσιάζονται μερικά από τα βασικότερα εργαλεία που χρησιμοποίησα.

3.1.1 Geth

Το geth [26] είναι μία command line διεπαφή, υλοποιημένη στην γλώσσα προγραμματισμού Go, για την εκτέλεση του πλήρους Ethereum κόμβου. Με το geth μπορεί κάποιος να γίνει μέλος του Ethereum δικτύου και μεταξύ άλλων να:

- Εξορύξει (mine) ether.
- Μεταφέρει ether μεταξύ διευθύνσεων.
- Να δημιουργήσει συμβόλαια και να στείλει λοιπές συναλλαγές.
- Εξερευνήσει το blockchain του Ethereum.
- Δημιουργήσει ένα ιδιωτικό Ethereum blockchain.
- Πραγματοποιήσει πολλές άλλες λειτουργίες.

3.1.2 Ganache CLI

Έκδοση που χρησιμοποίησα: 6.0.3

Το Ganache CLI [27] είναι ένα npm πακέτο. Το Ganache CLI είναι ένα βασισμένο στο Node.js Ethereum client για δοκιμή και ανάπτυξη. Χρησιμοποιεί την συλλογή βιβλιοθηκών ethereumjs για να προσομοιώσει πλήρως έναν πραγματικό Ethereum κόμβο, χωρίς όμως την επιβάρυνση που συνεπάγεται η εκτέλεση του πραγματικού Ethereum κόμβου. Δημιουργεί δηλαδή ένα ιδιωτικό Ethereum blockchain για γρήγορη δοκιμή και ανάπτυξη νέων έξυπνων συμβολαίων και κατανεμημένων εφαρμογών DApp (Decentralized application). Επίσης περιλαμβάνει όλες τις δημοφιλείς RPC (Remote Procedure Call) συναρτήσεις και δυνατότητες (πχ events) και μπορεί να εκτελεστεί αιτιοκρατικά, καθιστώντας έτσι την διαδικασία ανάπτυξης πολύ γρηγορότερη. Το Ganache CLI δηλαδή είναι ένα εργαλείο που προσφέρει μεγάλη ευκολία και πολλές δυνατότητες στον προγραμματιστή.

Από τα παραπάνω φαίνεται η αναγκαιότητα ενός τέτοιου εργαλείου, ειδάλλως θα έπρεπε όλη η ανάπτυξη και οι δοκιμές να γίνουν ή στο πραγματικό Ethereum δίκτυο ή κάποιο από τα test nets. Η διαδικασία αυτή θα είχε ως συνέπεια τα εξής:

- Αποθήκευση στο δημόσιο Ethereum δίκτυο περίσσειας πληροφορίας
- Μεγάλη επιβάρυνση του δικτύου με μη παραγωγική κίνηση
- Μεγάλα διαστήματα αναμονής
- Πολύ μεγαλύτερο κόστος ανάπτυξης
- Πολλές περισσότερες απαιτούμενες ανθρωποώρες για την επίτευξη ίδιου αποτελέσματος
- Λοιπά προβλήματα που συνδέονται με την φύση της τεχνολογίας των κατανεμημένων συστημάτων.

```
C:\Users\George>ganache-cli
Ganache CLI v6.0.3 (ganache-core: 2.0.2)

Available Accounts
==================
(0) 0x5408554034f850f01029719b353174c68260630b
(1) 0xf9f42d52832bfb6caba5f11ec005c439965af482
(2) 0xc2e10c514e9d8682f016325a1e50483079b99d1d
(3) 0xf0095a85e7e2e8e5665f580e10db00c3f78a4dd6
(4) 0xdbc04f0a800b50af3e7cee1aa4e76f3e0694e315
(5) 0xc34153edff9a134c23e4864a9423d45ed1d6fea1
(6) 0x35e76c04d8c953063c0584f9e683b80b9f94b4f6
(7) 0x3a9f5837f33ec258c48fad1aae15946f76528a21
(8) 0xaa33c23b240c24c6453ab162f2c059c31d704329
(9) 0x16bbac9f5bb0f509a488ef30dd6dea0d137d8296

Private Keys
==================
(0) e00af929ab8a442188067672fa00ad3a9a3b13adc49b575b94695c2ca395b50b
(1) 3847dd4d8a6758354a1d4a95fb1c295b364bbc79262d50bb5fc662900f46a310
(2) 76cb03fa0d2d9204cb3d0c1ea8d55204e50f2cf2e6e4df444bba4abdc1c14cc3
(3) bac4ef525fa5927eb2c7efb3bd52b7010b0cbaddcaed6ae7a2e55eb32b61201a
(4) 95008aabc68b89ebb9aadf1cc3dad5c13334ccb0a339b1727af60c93e52db40f
(5) 830342dad2a22ff9aba34ee290ada06e82624fd99b1ef6219cc4e4135b903885
(6) d92060c5a9c02a47c67817098b1168c1f0ce63a2444e994415004f9d65401cc4
(7) 75624dafecd5a3688073839fd8cef6d2dc6652bc370bb28aa0ed6da2c2d9e338
(8) 459198857a715989da505c87de4a7a4daa1671a80af9f895795a589a7e51c5d1
(9) a98f3a6a9caa3d2dc4416e3e31d734ad97e72f5c83cda80d012404fbf6798739

HD Wallet
==================
Mnemonic:      goose asset truth celery clip abandon clock true found inside silent parade
Base HD Path:  m/44'/60'/0'/0/{account_index}

Listening on localhost:8545
```

Εικόνα 3.1 Παράδειγμα εκτέλεσης του ganache-cli σε Windows 10 με τις προκαθορισμένες ρυθμίσεις

Όπως παρατηρούμε στην εικόνα, η εκτέλεση της εντολής ganache-cli δημιουργεί τοπικά ένα Ethereum blockchain, με 10 Ethereum λογαριασμούς/διευθύνσεις, που ακούει στην τοπική πόρτα 8545 (Listening on localhost:8545).

3.1.3 EthereumJS

Το EthereumJS [28] είναι η JavaScript κοινότητα για το Ethereum. Αποτελεί την μεγαλύτερη ως τώρα συλλογή βιβλιοθηκών και εργαλείων για την αλληλεπίδραση με το δίκτυο Ethereum μέσω της γλώσσας JavaScript. Περιλαμβάνει μία μεγάλη λίστα από modules χρήσιμα στην ανάπτυξη εφαρμογών για το Ethereum. Ενδεικτικά αναφέρω ένα πακέτο που χρησιμοποίησα.

ethereumjs-util [29] στην έκδοση 5.1.2 – Ένα npm πακέτο που χρησιμοποιείται σαν συμπληρωματικό του προηγούμενου, κυρίως στον server-side προγραμματισμό. Είναι μία συλλογή συναρτήσεων του Ethereum (utility functions for Ethereum). Το πακέτο αυτό χρειάζεται για την λειτουργία του και το πακέτο ethjs-util, το οποίο και κατεβαίνει αυτόματα μαζί του, και είναι ένα σύνολο βοηθητικών προγραμμάτων (Ethereum JS utilities). Το πακέτο αυτό περιλαμβάνει μεταξύ άλλων τις συναρτήσεις ecsign και ecrecover, οι οποίες υλοποιούν τον αλγόριθμο ψηφιακής υπογραφής ελλειπτικής καμπύλης (elliptic curve digital signature algorithm), με τον οποίο γίνεται η πιστοποίηση των διευθύνσεων/χρηστών στο blockchain.

3.1.4 Web3.js

Έκδοση που χρησιμοποίησα: 0.19.1

Ένα από τα πιο σημαντικά npm πακέτα που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Το πακέτο web3.js [30] είναι μία συλλογή βιβλιοθηκών που επιτρέπουν την αλληλεπίδραση με ένα τοπικό ή απομακρυσμένο κόμβο του Ethereum blockchain, χρησιμοποιώντας HTTP ή IPC σύνδεση. Με άλλα λόγια είναι μία διεπαφή επικοινωνίας μεταξύ της JavaScript και του Ethereum blockchain («Ethereum compatible JavaScript API which implements the Generic JSON RPC specification»). Είναι δηλαδή ο πιο διαδεδομένος για να αναφέρεται η JavaScript (server-side ή/και front-end) σε αντικείμενα που «ζουν» μέσα στο blockchain, όπως τα έξυπνα συμβόλαια (solidity smart contracts), τα δεδομένα τους, τις συναρτήσεις τους, τις διευθύνσεις και τα υπόλοιπα λογαριασμών, όπως και πολλά άλλα.

3.1.5 Solidity

Έκδοση που χρησιμοποιήθηκε: 0.4.16

Η Solidity [31] είναι μία «συμβολαιοστρεφής», υψηλού επιπέδου γλώσσα προγραμματισμού που εκτελείται στο Ethereum Virtual Machine (EVM) και έχει δημιουργηθεί για να μεγεθύνει τις δυνατότητες της ιδεατής αυτής μηχανής. Χρησιμοποιείται κυρίως για την δημιουργία έξυπνων συμβολαίων (smart contracts) για το Ethereum blockchain. Ο ίδιος ο πηγαίος κώδικας της γλώσσας Ethereum είναι γραμμένος σε Solidity.

Έχει παρόμοιο συντακτικό με αυτό της JavaScript, οπότε είναι εύκολα κατανοήσιμη από έναν πολύ μεγάλο αριθμό προγραμματιστών. Είναι μία στατικού τύπου

γλώσσα. Υποστηρίζει την κληρονομικότητα με παρόμοιο τρόπο με άλλες γλώσσες προγραμματισμού (πχ. C++).

Remix

Το Remix [32], [33] είναι μία σουίτα εργαλείων για την αλληλεπίδραση με το Ethereum blockchain. Το Remix IDE είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης, προσβάσιμο από φυλλομετρητή για την ανάπτυξη έξυπνων συμβολαίων (Solidity smart contracts), την μεταγλώττιση και κατόπιν την δημιουργία (deploy) και εκτέλεσή τους στο Ethereum blockchain. Πιο συγκεκριμένα μέσω του Remix πραγματοποιήθηκε το deployment των συμβολαίων στο δίκτυο του Ethereum (test network).

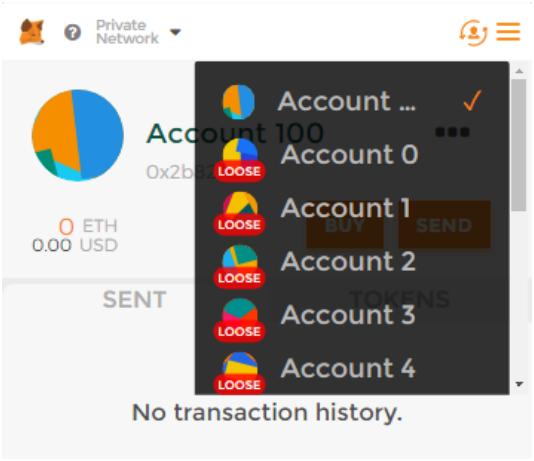
Τέλος, για να γίνει η ανάπτυξη των έξυπνων συμβολαίων γρηγορότερη χρησιμοποίησα το npm πακέτο solc (έκδοση 0.4.18). Αυτό είναι ένα npm πακέτο που περιλαμβάνει την Solidity με τον μεταγλωττιστή της, την προγραμματιστική γλώσσα στην οποία γράφονται όλα τα έξυπνα συμβόλαια (smart contracts) του Ethereum. Είναι μία ανεξάρτητη πλατφόρμας βιβλιοθήκη της JavaScript, μέσω της οποίας μπορούμε και να μεταγλωττίσουμε κώδικα Solidity. Το χρησιμοποίησα για να μπορώ να μεταγλωττίζω solidity πηγαίο κώδικα μέσα στις ροές του Node-RED και στη συνέχεια να κάνω αυτόματα deploy στο blockchain τα έξυπνα συμβόλαια.

3.1.6 MetaMask

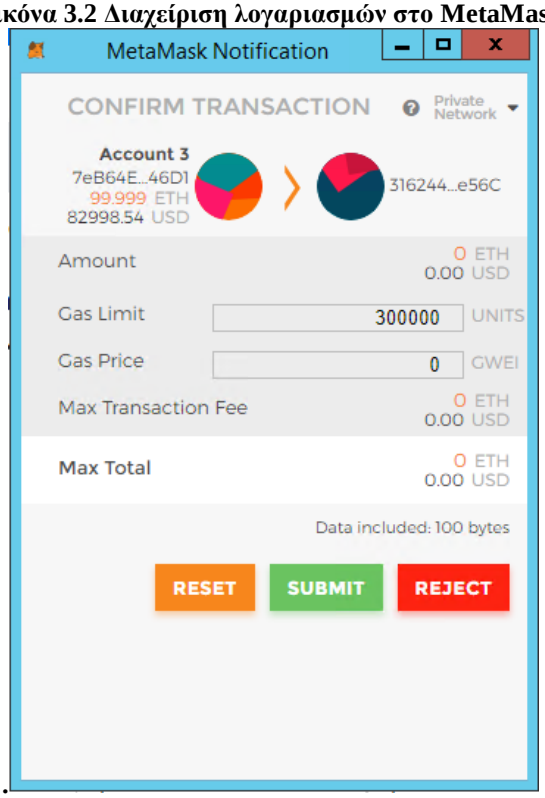
Ιστοσελίδα: https://MetaMask.io/

Το MetaMask [34] είναι ένα plugin διαθέσιμο για τους φυλλομετρητές Google Chrome, Mozilla Firefox, Brave και Opera. Το MetaMask είναι στην ουσία μία γέφυρα η οποία δίνει στον browser πρόσβαση στις κατανεμημένες εφαρμογές (DApps), που για την λειτουργία τους βασίζονται στο δίκτυο Ethereum. Το μεγάλο πλεονέκτημα που προσφέρει είναι το γεγονός ότι με την χρήση του, η πρόσβαση στις εφαρμογές αυτές, δεν απαιτεί την εκτέλεση του πλήρους Ethereum κόμβου στο μηχάνημα του χρήστη, όπως για παράδειγμα απαιτεί ο ειδικός για το Ethereum φυλλομετρητής Mist.

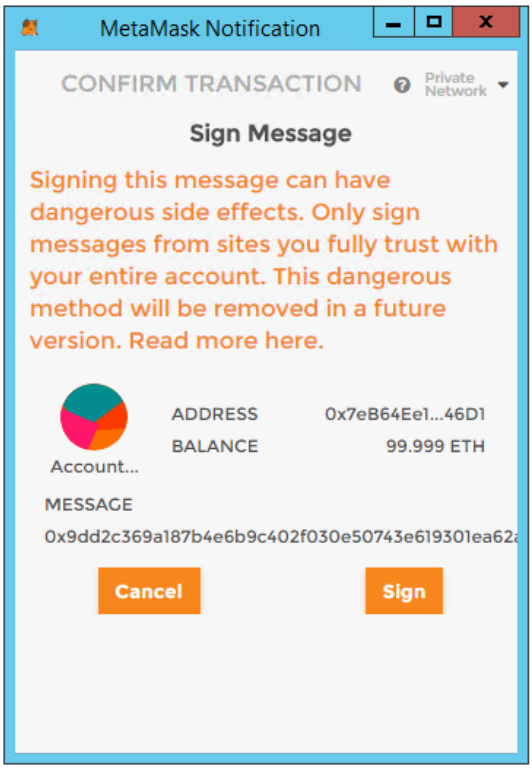
Συμπεριλαμβάνει μία ασφαλή κρύπτη και παρέχει στον χρήστη μία διεπαφή, μέσω της οποίας αυτός μπορεί να διαχειρίζεται τους Ethereum λογαριασμούς/διευθύνσεις του, ώστε να αλληλοεπιδρά με τις ιστοσελίδες, να στέλνει ή και να υπογράφει συναλλαγές και να υπογράφει δεδομένα.



Εικόνα 3.2 Διαχείριση λογαριασμών στο MetaMask



Εικόνα 3.3 Αποστολή συναλλαγής στο MetaMask



Εικόνα 3.4 Υπογραφή δεδομένων στο MetaMask

Σύμφωνα με τους δημιουργούς του, ο σκοπός του είναι να κάνει το Ethereum προσβάσιμο σε όσο το δυνατόν περισσότερο κόσμο.

Προσωπική μου γνώμη είναι πως ως ένα σημείο έχει ήδη πετύχει τον σκοπό του. Πιο συγκεκριμένα αναφέρω πως για την εγκατάσταση και την εκμάθηση των βασικών του λειτουργιών δεν απαιτήθηκε καμία τεχνική γνώση, ενώ ο συνολικός χρόνος δεν ξεπέρασε τα είκοσι λεπτά. Από την άλλη η διαδικασία εγκατάστασης και εκτέλεσης του πλήρους Ethereum κόμβου απαιτεί πολλές τεχνικές γνώσεις, πολλούς πόρους συστήματος (πόροι του επεξεργαστή και πολλή μνήμη) και πολύ χρόνο. Πράγματα τα οποία περιστασιακός, μη τεχνικός χρήστης δεν είναι συνήθως διατεθειμένος ή δεν έχει την δυνατότητα να κάνει.

3.2 NPM

Το npm – Node Package Manager [35] είναι ένα πρόγραμμα διαχείρισης πακέτων της γλώσσας JavaScript, επίσης είναι το μεγαλύτερο μητρώο προγραμμάτων στον κόσμο. Διαχειριστής του είναι η εταιρία npm, Inc. Είναι το προκαθορισμένο πρόγραμμα διαχείρισης πακέτων του προγραμματιστικού περιβάλλοντος Node.js. Εκτός από πακέτα, στο αρχείο του npm υπάρχουν και node modules τα οποία χρησιμοποιούνται στο server-side προγραμματισμό. Το πακέτο (package) είναι ένα αρχείο ή ένα directory το οποίο περιγράφεται από ένα package.json, ενώ το module είναι ένα αρχείο ή ένα directory το οποίο φορτώνεται από το Node.js μέσω του require().

Αποτελείται από τρία διακριτά κομμάτια:

- Την ιστοσελίδα (https://www.npmjs.com/).

- Ένα αρχείο (registry), στο οποία είναι αποθηκευμένα τα JavaScript πακέτα και τα Node modules.
- Τον command line client (Commend LineInterface) (που βρίσκεται στον υπολογιστή του χρήστη), μέσω του οποίου είτε λαμβάνονται τα διαθέσιμα πακέτα είτε δημοσιεύονται στο αρχείο πακέτα που ο χρήστης έχει δημιουργήσει.

Είναι πολύ χρήσιμο στην κοινότητα ανάπτυξης λογισμικού, διότι αφενός επιτρέπει την ανταλλαγή πακέτων, δηλαδή κώδικα που λύνει ένα συγκεκριμένο πρόβλημα πολύ καλά και αφετέρου έχει θεσπίσει προδιαγραφές τις οποίες ακολουθούν όλα τα πακέτα που δημοσιεύονται στο αρχείο, έτσι ώστε να υπάρχει ένας όσο γίνεται ενιαίος τρόπος χρησιμοποίησής τους από τους προγραμματιστές.

Ένα παράδειγμα npm πακέτου που χρησιμοποίησα είναι το git, στην έκδοση 0.1.5. Αυτό δεν είναι παρά το γνωστό σε όλους σύστημα διαχείρισης εκδόσεων Git. Είναι απαραίτητο για την ορθή λειτουργία του πακέτου web3.js.

3.3 Node-RED

Έκδοση που χρησιμοποιήθηκε: v0.17.5

Το Node-RED [36] είναι ένα προγραμματιστικό εργαλείο ανοιχτού κώδικα, το οποίο αρχικά αναπτύχθηκε από την ομάδα «Emerging Technology Services» της IBM το 2013, ενώ από το 2016 είναι ένα από τα χρηματοδοτούμενα έργα του JS Foundation. Το γεγονός αυτό, δείχνει την σημαντικότητα του Node-RED για την βιομηχανία ως επίσης και το μέγεθος της δραστηριότητας καθώς και το ενδιαφέρον την κοινότητας ανοικτού κώδικα γύρω από αυτό.

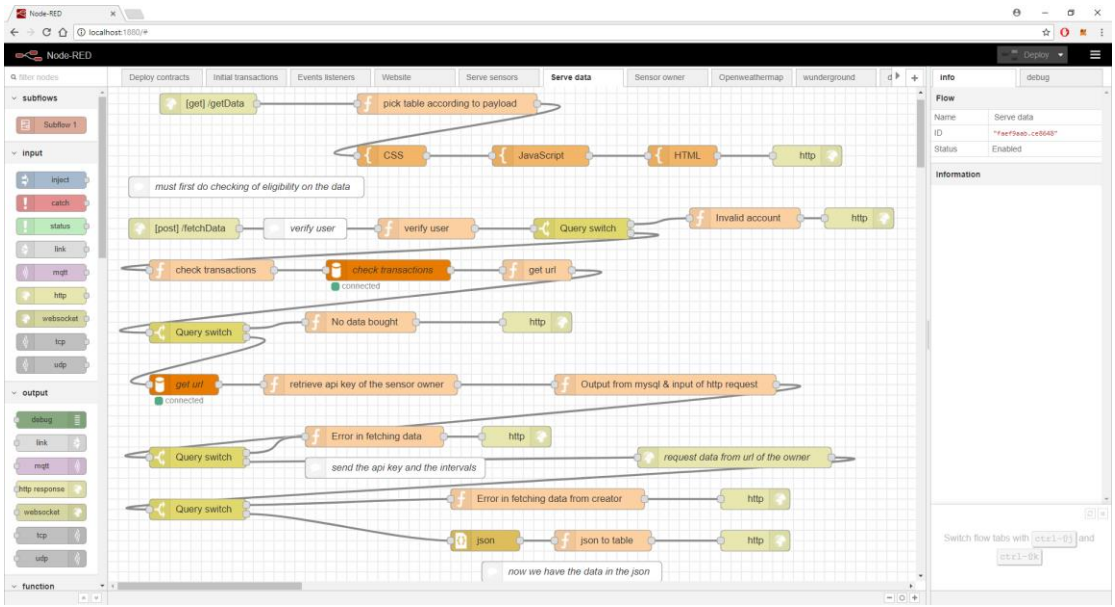
Αρχικά εθεωρείτο ιδανικό για εφαρμογές του Διαδικτύου των Πραγμάτων (Internet of Things), όμως σύντομα μετεξελίχθηκε σε ένα πολύ πιο γενικό εργαλείο που εύκολα μπορεί να επεκταθεί και προς διαφορετικές κατευθύνσεις, ώστε να χρησιμοποιηθεί και σε άλλης φύσης και τύπου εφαρμογές. Θα μπορούσε κάποιος να ισχυριστεί ότι είναι μία μηχανή επεξεργασίας γενικών συμβάντων (generic event-processing engine). Για παράδειγμα μπορεί να χρησιμοποιηθεί για την εξυπηρέτηση HTTP, Web sockets, TCP, Twitter και λοιπών συμβάντων. Ένα παράδειγμα αυτού, αποτελεί η παρούσα διπλωματική, στα πλαίσια της οποίας όχι μόνο διαπραγματεύονται θέματα του IoT, αλλά έχει γίνει και ανάπτυξη μίας διαδικτυακής εφαρμογής μέσω του Node-RED, δηλαδή χρησιμοποιήθηκε σαν full-stack web development πλατφόρμα.

Το Node-RED αποτελείται από το Node.js περιβάλλον εκτέλεσης και βασίζεται στην αρχή του προγραμματισμού βάσει ροής (Flow-based programming). Μέσω ενός φυλλομετρητή ιστοσελίδων μπορούμε να αποκτήσουμε πρόσβαση στον επεξεργαστή των ροών «flow editor». Στη συνέχεια για την δημιουργία της εφαρμογής επιλέγονται κόμβοι διαθέσιμοι στην παλέτα οι οποίοι στην συνέχεια συνδέονται μεταξύ τους σχηματίζοντας ροές. Η κοινότητα ελεύθερου λογισμικού μπορεί να δημιουργεί και να προσθέτει νέους κόμβους στην παλέτα, όπως επίσης και να δημιουργεί ολόκληρες ροές και να τις μοιράζεται εύκολα με την μορφή JSON αρχείων.

Τέλος, το Node-RED, εκμεταλλευόμενο την φύση του Node.js παρέχει πολλές δυνατότητες όσον αφορά το περιβάλλον/hardware εκτέλεσής του. Μπορεί να τρέχει είτε στο νέφος (cloud) (IBM Bluemix, SenseTechnic FRED, Amazon Web Services, Microsoft Azure), είτε σε μία IoT συσκευή, είτε τοπικά σε έναν υπολογιστή.

Τα npm πακέτα που χρησιμοποίησα και απευθύνονταν ειδικά στο Node-RED:

- node-red-node-mysql@0.0.16 – Ένα πακέτο που κάνει διαθέσιμο στο Node-RED ένα ειδικό κόμβο, μέσω του οποίου μπορούμε με αυτοματοποιημένο τρόπο, γλιτώνοντας πολύ κώδικα, να επικοινωνήσουμε με MySQL βάσεις δεδομένων στέλνοντας σε αυτές queries και λαμβάνοντας δεδομένα [37].
- node-red-node-openweathermap@0.1.20 – Ένα πακέτο που κάνει διαθέσιμους στο Node-RED ειδικούς κόμβους [38], μέσω των οποίων μπορούμε να παίρνουμε καιρικά δεδομένα από το OpenWeatherMap (https://www.openweathermap.org).
- node-red-node-weather-underground@0.1.11 – Ένα πακέτο που κάνει διαθέσιμους στο Node-RED ειδικούς κόμβους [39], μέσω των οποίων μπορούμε να παίρνουμε καιρικά δεδομένα από το The Weather Underground (https://www.wunderground.com).
- node-red-node-darksky@0.1.17 – Ένα πακέτο που κάνει διαθέσιμους στο Node-RED ειδικούς κόμβους [40], μέσω των οποίων μπορούμε να παίρνουμε καιρικά δεδομένα από το Dark Sky (www.darksky.net).



Εικόνα 3.5 Παράδειγμα από τον editor του Node-RED

Να αναφερθεί επίσης ότι ο κώδικας JavaScript που γράφεται στο Node-RED εκτελείται σε sandbox, έτσι δεν μπορεί να χρησιμοποιηθεί το require(…) της JavaScript για την φόρτωση κάποιου πακέτου. Έτσι ό,τι πακέτο χρειάζεται, πρέπει να φορτώνεται από το Node-RED στο «Global context» κατά την εκκίνησή του και γι' αυτόν τον σκοπό θα πρέπει για κάθε πακέτο που θέλουμε να χρησιμοποιήσουμε να αλλάζουμε το πεδίο ιδιοτήτων functionGlobalContext αρχείου settings.js [41], να αλλάζουμε δηλαδή τα configuration files του Node-RED.

3.4 Node.js

Έκδοση που χρησιμοποιήθηκε: v8.9.3

Το Node.js [42] είναι μία server-side πλατφόρμα, η οποία έχει αναπτυχθεί πάνω στο Google Chrome JavaScript Engine(V8 Engine) το 2009 και από τότε συνεχώς βελτιώνεται. Είναι ένα ανοιχτού κώδικα (open source), cross-platform περιβάλλον ανάπτυξης και εκτέλεσης της γλώσσας προγραμματισμού JavaScript, κατάλληλο για εύκολη και γρήγορη ανάπτυξη κλιμακώσιμων διαδικτυακών εφαρμογών. Χρησιμοποιεί ένα event-driven, non-blocking I/O μοντέλο και συνεπώς είναι αποτελεσματικό, πετυχαίνει δηλαδή μεγάλη απόδοση χρησιμοποιώντας λίγους φυσικούς πόρους. Θεωρείται ιδανικό για εφαρμογές πραγματικού χρόνου, υπολογιστικά έντονες που τρέχουν σε κατανεμημένα περιβάλλοντα. Επίσης, το Node.js παρέχει μία μεγάλη βιβλιοθήκη από JavaScript modules, γεγονός που απλοποιεί ουσιωδώς την διαδικασία ανάπτυξης διαδικτυακών εφαρμογών. Δηλαδή το Node.js είναι και περιβάλλον εκτέλεσης, αλλά και βιβλιοθήκη της JavaScript.

Χαρακτηριστικά του Node.js [43]:

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

- Πολύ γρήγορο – Επειδή βασίζεται πάνω στην ιδεατή μηχανή Google Chrome V8 JavaScript Engine η βιβλιοθήκη Node.js είναι πολύ γρήγορη στην εκτέλεση κώδικα.

- Μονού νήματος, αλλά πολύ κλιμακώσιμο – Το Node.js χρησιμοποιεί μοντέλο μονού νήματος εκτέλεσης με event looping. Ο μηχανισμός αυτός των συμβάντων βοηθάει τον εξυπηρετητή να απαντάει με non-blocking τρόπο κάτι που του δίνει την ιδιότητα της κλιμακωσιμότητας, σε αντίθεση με άλλους παραδοσιακούς εξυπηρετητές που χρησιμοποιούν περιορισμένο αριθμό νημάτων για την εξυπηρέτηση αιτήσεων. Το πρόγραμμα ενός νήματος του Node.js μπορεί να εξυπηρετήσει πολύ μεγαλύτερο αριθμό αιτήσεων από παραδοσιακούς εξυπηρετητές, όπως είναι ο ApacheHTTP Server.

- Δεν υπάρχει προσωρινή αποθήκευση δεδομένων – Οι εφαρμογές του Node.js δεν κάνουν προσωρινή αποθήκευση δεδομένων, αλλά στέλνουν τα δεδομένα σε μικρά κομμάτια.

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

Η υποστήριξη από την βιομηχανία είναι μεγάλη. Κάθε χρόνο πραγματοποιούνται πολλά συνέδρια για τους προγραμματιστές, όπως τα NodeConf, Node Interactive και Node Summit. Επίσης έχουν αναπτυχθεί πολλά frameworks από την

κοινότητα ελεύθερου λογισμικού, με πιο δημοφιλή τα Connect, Express.js, Socket.IO, Koa.js, Hapi.js, Sails.js, Meteor. Τέλος, υπάρχουν διαθέσιμα ολοκληρωμένα περιβάλλοντα ανάπτυξης (IDEs) όπως τα Atom, Brackets, JetBrains WebStorm, Microsoft Visual Studio, NetBeans, Nodeclipse Enide Studio, και Visual Studio Code.

Google V8 engine

Η ιδεατή μηχανή V8 της Google είναι μία ανοικτού κώδικα υψηλής απόδοσης μηχανή JavaScript [44]. Είναι γραμμένη σε C++ και JavaScript χρησιμοποιείται από τον ανοικτού κώδικα browser Google Chrome, το Node.js και σε πολλές άλλες εφαρμογές. Είναι υπεύθυνη για την εκτέλεση κώδικα που γραμμένος στην γλώσσα JavaScript. Υλοποιεί το ECMAScript και τρέχει σε Windows 7 ή μεταγενέστερα, macOS 10.5+, και Linux συστήματα που χρησιμοποιούν IA-32, ARM, ή MIPS επεξεργαστές.

3.5 Ανάπτυξη ιστοσελίδων

Σε αυτήν την ενότητα θα παρουσιαστούν οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη των ιστοσελίδων, μέσω των οποίων οι χρήστες χρησιμοποιούν την εφαρμογή. Φυσικά χρησιμοποιήθηκε η γλώσσα HTML (Hypertext Markup Language), η οποία είναι η γλώσσα που χρησιμοποιείται σε ολόκληρο τον κόσμο για την δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Η HTML μαζί με τα Cascading Style Sheets (CSS) και την JavaScript είναι οι ίσως οι τρεις πιο σημαντικές τεχνολογίες για τον παγκόσμιο ιστό.

3.5.1 Bootstrap

Το Bootstrap [45] είναι ένα ελεύθερο, ανοιχτού κώδικα πλαίσιο ανάπτυξης ιστοσελίδων και διαδικτυακών εφαρμογών (front-end web framework), δηλαδή μία συλλογή εργαλείων. Απαρτίζεται από μία σειρά από Sass stylesheets (Less stylesheets έως την έκδοση 3) που υλοποιούν τις διάφορες λειτουργικότητες. Παράλληλα, είναι ένας συνδυασμός από HTML, CSS, και JavaScript κώδικα, σχεδιασμένος να συμβάλλει στην αποτελεσματικότερη την δημιουργία διεπαφών χρήστη. Κάνει την διαδικασία σχεδίασης και ανάπτυξης ιστοσελίδων πιο εύκολη και πιο γρήγορη, βοηθώντας παράλληλα τον προγραμματιστή να πετύχει καλύτερο αποτέλεσμα, με πιο όμορφες και χρηστικές διεπαφές. Επίσης, με τα CSS που παρέχει, συμβάλλει στην δημιουργία προσαρμοστικών ιστοσελίδων· δηλαδή ιστοσελίδων οι οποίες δυναμικά αλλάζουν τον τρόπο το περιεχόμενό τους και τον τρόπο παρουσίασής του, ανάλογα το μέγεθος της οθόνης στην οποία προβάλλονται, πράγμα απαραίτητο, αφού όλο και περισσότεροι χρήστες χρησιμοποιούν κινητά και tablet και όχι μόνο υπολογιστές για την περιήγησή τους στο διαδίκτυο.

Το Bootstrap είναι ένα από τα πιο διαδεδομένα front-end frameworks. Ορισμένοι λόγοι γι’ αυτό είναι [46]:

- Ευκολία στην χρήση. Ο οποιοσδήποτε με στοιχειώδεις γνώσεις HTML και CSS μπορεί να το χρησιμοποιήσει.
- Προσαρμοστικότητα στην παρουσίαση. Τα CSS του Bootstrap προσαρμόζονται σε κινητά, tablets, και υπολογιστές.

- Συμβατότητα με φυλλομετρητές. Υποστηρίζει όλους τους ευρέως χρησιμοποιούμενους φυλλομετρητές, όπως Google Chrome, Firefox, Microsoft Edge, Internet Explorer, Opera, Safari και άλλους.
- Πολύ καλό σύστημα πλέγματος. Όταν χρησιμοποιείται το Bootstrap τα πάντα μπορούν να οργανωθούν σε στήλες.
- Βασικό στυλ σε πολλά στοιχεία. Παρέχεται βασικό στυλ για τα περισσότερα HTML στοιχεία (Typography, Code, Tables, Forms, Buttons, Images, Icons).
- JavaScript components με την μορφή jQuery plugins, τα οποία παρέχουν επιπλέον στοιχεία για την διεπαφή του χρήστη, όπως dialog boxes, tooltips, carousels. Επίσης προσθέτουν επιπλέον λειτουργικότητα σε ήδη υπάρχοντα στοιχεία. Ενδεικτικά μερικά JavaScript plugins που υποστηρίζονται: Dropdown, Scrollspy, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel και Typehead.

3.5.2 JavaScript

Εκτός από το back-end (κώδικας που εκτελείται στον εξυπηρετητή), JavaScript [47] χρησιμοποιήθηκε και για το front-end (κώδικας που εκτελείται στον φυλλομετρητή του πελάτη – client-side JavaScript). Η JavaScript είναι μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού, αφού χρησιμοποιείται κατά κόρον σε διαδικτυακές εφαρμογές (και όχι μόνο) και υποστηρίζεται από όλους τους μοντέρνους φυλλομετρητές. Είναι ο καλύτερος (αν όχι ο μόνος ενδεδειγμένος) τρόπος να δώσουμε δυναμικό περιεχόμενο σε μία ιστοσελίδα κάνοντάς την να επιτελεί σύνθετες λειτουργίες. Μπορεί επίσης να χρησιμοποιηθεί και για την ανάπτυξη προγραμμάτων που δεν εκτελούνται σε φυλλομετρητές, όπως για παράδειγμα σε εξυπηρετητές, βάσεις δεδομένων, επεξεργαστές PDF εγγράφων, Desktop εφαρμογές, αλλά και εφαρμογές κινητών.

Για την εκτέλεσή της είναι απαραίτητη κάποια μηχανή JavaScript (JavaScript engine), όπως είναι για παράδειγμα η Google V8. Η μηχανή JavaScript είναι στην ουσία ένα πρόγραμμα ή διερμηνέας που εκτελεί κώδικα γραμμένο στην γλώσσα JavaScript. Συναντάται κυρίως στους φυλλομετρητές, όμως χρησιμοποιείται και από άλλες πλατφόρμες, όπως είναι το Node.js.

Η JavaScript είναι μία υψηλού επιπέδου, δυναμική, weakly typed, prototype-based, multi-paradigm, interpreted γλώσσα προγραμματισμού. Ως multi-paradigm γλώσσα, η JavaScript υποστηρίζει τα event-driven, functional, and imperative (including object-oriented and prototype-based) προγραμματιστικά στυλ. Έχει έτοιμες προγραμματιστικές διεπαφές (APIs) για την επεξεργασία κειμένου, πινάκων, ημερομηνιών, καθώς και τον χειρισμό DOM (Document Object Model), όμως δεν υποστηρίζει λειτουργίες εισόδου εξόδου (I/O), όπως δικτύωση, αποθήκευση, γραφικά, παρά βασίζει την υλοποίηση των λειτουργιών αυτών στο περιβάλλον εκτέλεσής της.

Μερικά από τα πλεονεκτήματα της χρήσης JavaScript στις διαδικτυακές εφαρμογές είναι [48]:

- Λιγότερη αλληλεπίδραση μεταξύ πελάτη – εξυπηρετητή. Προσφέρεται η δυνατότητα προεπεξεργασίας των δεδομένων τοπικά, στον φυλλομετρητή του πελάτη πριν αυτά σταλούν στον εξυπηρετητή. Μειώνεται έτσι ο φόρτος του εξυπηρετητή, αλλά και ολόκληρου του δικτύου.

- Άμεση ανατροφοδότηση στον χρήστη. Υπάρχει η δυνατότητα εμφάνισης μηνύματα στον χρήστη που τον ενημερώνουν σχετικά με την κατάσταση διεργασιών που απαιτούν πολύ χρόνο κτλ.
- Αυξημένη διαδραστικότητα ιστοσελίδας. Μπορούν να δημιουργηθούν διεπαφές, οι οποίες ανταποκρίνονται στις κινήσεις του χρήστη, χωρίς να χρειάζεται να ανανεωθεί ολόκληρη η ιστοσελίδα.
- Πλουσιότερες διεπαφές. Η JavaScript εμπλουτίζει την HTML και τα CSS, κάνοντας έτσι διαθέσιμα στον χρήστη πολλά νέα στοιχεία.

3.5.3 jQuery

Η jQuery [49] είναι μία γρήγορη, συμπαγής, με πλούσια χαρακτηριστικά JavaScript βιβλιοθήκη, σχεδιασμένη να απλοποιήσει την συγγραφή JavaScript κώδικα για το front-end. Είναι δωρεάν (ελεύθερη), και ανοιχτού κώδικα και αποτελεί την πιο ευρέως χρησιμοποιούμενη JavaScript βιβλιοθήκη στο διαδίκτυο. Χαρακτηριστικό είναι το σύνθημά της «write less, do more». Η αρθρωτή φύση της καθώς και η εύκολη στην χρήση προγραμματιστική διεπαφή την καθιστούν ιδανική για την δημιουργία πλούσιων ιστοσελίδων και διαδικτυακών εφαρμογών.

Πιο συγκεκριμένα παρέχονται τα παρακάτω χαρακτηριστικά [50]:

- Εύκολος χειρισμός HTML/DOM.
- Εύκολος χειρισμός CSS.
- Εύκολος χειρισμός συμβάντων (events).
- Εφέ και κινούμενα σχέδια.
- Απλοποίηση των AJAX (Asynchronous JavaScript And XML) κλήσεων.
- Συναρτήσεις για την διευκόλυνση και τυποποίηση της υλοποίηση συνηθισμένων διεργασιών.

Επιπλέον λειτουργικότητες προσφέρονται μέσω των jQuery plugins. Καλύπτεται έτσι από την jQuery ένα πολύ μεγάλο φάσμα λειτουργιών. Τέτοιου είδους πακέτα μπορούν να βρεθούν έτοιμα, από την κοινότητα ελεύθερου λογισμικού, ή και να δημιουργηθούν από οποιονδήποτε και να μοιραστούν και σε άλλους χρήστες.

3.6 Δεδομένα καιρού

Στα πλαίσια της ανάπτυξης της εφαρμογής, δεν έγινε εγκατάσταση κάποιου αισθητήρα καιρού, αφού η εφαρμογή αφορά μία πλατφόρμα μέσω της οποίας θα μπορούν κάτοχοι αισθητήρων να πωλούν τα δεδομένα αυτών άλλους χρήστες της εφαρμογής μέσω του Ethereum blockchain. Αυτό που έγινε όμως, είναι η προσομοίωση του αισθητήρα χρησιμοποιώντας προγραμματιστικές διεπαφές (APIs) που προσφέρονται στο διαδίκτυο, από μεγάλους μετεωρολογικούς οργανισμούς. Προτιμήθηκαν διεπαφές για τις οποίες υπήρχε έτοιμος κόμβος στο Node-RED, προκειμένου να μειωθεί η πολυπλοκότητα, αλλά και να γίνει επίδειξη της δημοφιλίας του Node-RED και της σημασίας των πακέτων που ανεβάζουν χρήστες (contributors) στο αρχείο του npm.

3.6.1 Open Weather Map

Ιστοσελίδα: openweathermap.org

Προσφέρεται ένας ειδικός κόμβος, ο openweathermap, ο οποίος στέλνει περιοδικά ερωτήματα στην βάση δεδομένων καιρού του openweathermap.org, και επιστρέφει δεδομένα καιρού για την επιλεγμένη περιοχή. Από τα στοιχεία που επιστρέφει η εφαρμογή κρατάει τα εξής:

- Θερμοκρασία (temperature) σε βαθμούς κελσίου (°C).
- Σχετική υγρασία (relative humidity) σε ποσοστό επί τις 100 (%).
- Ατμοσφαιρική πίεση (pressure) σε hecto Pascal (hPa).
- Ορατότητα (visibility) σε μέτρα.
- Ταχύτητα (speed) και διεύθυνση (direction) ανέμου σε μέτρα ανά δευτερόλεπτο (m/s) και μετεωρολογικές μοίρες (°) αντίστοιχα.
- Κάλυψη του ουρανού από σύννεφα (sky cloud coverage) σε ποσοστό επί τις εκατό (%).

3.6.2 Weather underground

Ιστοσελίδα: www.wunderground.com

Προσφέρεται ένας ειδικός κόμβος, ο wunderground, ο οποίος στέλνει περιοδικά ερωτήματα στην βάση δεδομένων καιρού του wunderground.com, και επιστρέφει δεδομένα καιρού για την επιλεγμένη περιοχή. Από τα στοιχεία που επιστρέφει η εφαρμογή κρατάει τα εξής:

- Θερμοκρασία (temperature) σε βαθμούς κελσίου (°C).
- Σχετική υγρασία (relative humidity) σε ποσοστό επί τις 100 (%).
- Ατμοσφαιρική πίεση (pressure) σε hecto Pascal (hPa).
- Ορατότητα (visibility) σε χιλιόμετρα.
- Ταχύτητα (speed) και διεύθυνση (direction) ανέμου σε χιλιόμετρα ανά ώρα (km/h) και μετεωρολογικές μοίρες (°) αντίστοιχα.
- Σημείο δρόσου ή σημείο υγροποίησης ή σημείο κόρου ατμόσφαιρας (dew point) σε βαθμούς κελσίου (°C).
- Ηλιακή ακτινοβολία σε watt/m².
- Δείκτης υπέρυθρης (UV) ακτινοβολίας σε κλίμακα 0 έως 11.

3.6.3 Dark Sky

Ιστοσελίδα: darksky.net

Προσφέρεται ένας ειδικός κόμβος, ο darksky, ο οποίος στέλνει περιοδικά ερωτήματα στην βάση δεδομένων καιρού του darksky.net, και επιστρέφει δεδομένα καιρού για την επιλεγμένη περιοχή. Από τα στοιχεία που επιστρέφει η εφαρμογή κρατάει τα εξής:

- Θερμοκρασία (temperature) σε βαθμούς κελσίου (°C).

- Σχετική υγρασία (relative humidity) σε ποσοστό επί τις 100 (%) διά εκατό.
- Ατμοσφαιρική πίεση (pressure) σε hecto Pascal (hPa).
- Ορατότητα (visibility) σε χιλιόμετρα.
- Ταχύτητα (speed) και διεύθυνση (direction) ανέμου σε μέτρα ανά δευτερόλεπτο (m/s) και μετεωρολογικές μοίρες (°) αντίστοιχα.
- Κάλυψη του ουρανού από σύννεφα (sky cloud coverage) σε ποσοστό επί τις εκατό (%).
- Σημείο δρόσου ή σημείο υγροποίησης ή σημείο κόρου ατμόσφαιρας (dew point) σε βαθμούς κελσίου (°C).
- Δείκτης υπέρυθρης (UV) ακτινοβολίας σε κλίμακα 0 έως 11.
- Πυκνότητα στρώματος όζοντος της ατμόσφαιρας (columnar density of total atmospheric ozone layer) σε μονάδες Dobson.

3.7 Βάσεις Δεδομένων

Η βάση δεδομένων που χρησιμοποιήθηκε είναι η MariaDB [51]. Η MariaDB είναι μία σχεσιακή βάση δεδομένων ελεύθερη και ανοικτού κώδικα, η οποία αποτελεί fork της MySQL. Από το 2009 υποστηρίζεται από το MariaDB Foundation και την κοινότητα ανοιχτού κώδικα. Χρησιμοποιείται όχι μόνο σε μικρά έργα, αλλά και από μεγάλες εταιρίες, όπως η Wikipedia, το WordPress και η Google.

Η MariaDB είναι μία MySQL εφαρμογή που αναπτύσσεται ως ανοιχτού κώδικα λογισμικό RDBMS (Relational Database Management System). Είναι εμπλουτισμένη με δυνατότητες υψηλής διαθεσιμότητας, ασφάλειας, δια λειτουργικότητας και υψηλής απόδοσης. Χρησιμοποιεί την γλώσσα SQL (Structured Query Language), την πιο γνωστή γλώσσα για την προσθήκη, πρόσβαση και επεξεργασία των δεδομένων – εγγραφών σε Βάσεις Δεδομένων.

Μερικά από τα κυριότερα χαρακτηριστικά της είναι τα εξής [52]:

- Πρωτεύον μοντέλο: Σχεσιακό
- Επιπρόσθετα μοντέλα: Document store, Key-value store
- Άδεια λειτουργίας: Ανοιχτού κώδικα
- Γλώσσα υλοποίησης: C και C++
- Υποστηριζόμενα λειτουργικά συστήματα: FreeBSD, Linux, Solaris, Windows
- Data scheme: Ναι
- Προκαθορισμένοι τύποι δεδομένων: Ναι
- Υποστήριξη XML: Ναι
- Προγραμματιστικές διεπαφές (APIs): ADO.NET, JDBC, ODBC
- Υποστηριζόμενες γλώσσες δεδομένων: Ada, C, C#, C++, D, Eiffel, Erlang, Go, Haskell, Java, JavaScript (Node.js), Objective-C, OCaml, Perl, PHP, Python, Ruby, Scheme, Tcl
- Υποστήριξη όλων των συνόλων χαρακτήρων: Ναι
- Triggers: Ναι
- Πολυνηματική λειτουργία: Ναι

Τα μεγαλύτερα πλεονεκτήματά της σύμφωνα με τον προμηθευτή της:

- Υψηλή διαθεσιμότητα
- Κλιμακωσιμότητα
- Υψηλότερη απόδοση από την MySQL και άλλες βάσεις δεδομένων

Τυπικά σενάρια εφαρμογής της αποτελούν τα εξής:

- Διαδικτυακές εφαρμογές
- SaaS – Software as a Service
- Επιχειρησιακές/συναλλακτικές εφαρμογές στο νέφος Cloud operational/transactional applications

3.8 ~Okeanos

Ο ~Okeanos [53] είναι μία IaaS (Infrastructure as a Service) υπηρεσία νέφους που προσφέρει το Εθνικό Δίκτυο Έρευνας και Τεχνολογίας. Αποτελεί μία παροχή στην ακαδημαϊκή και ερευνητική κοινότητα. Φοιτητές, καθηγητές και ερευνητές έχουν στην διάθεσή τους δωρεάν την ιδεατή υποδομή (υπολογιστική, δικτυακή, αποθηκευτική ισχύς).

Έχει πολύ φιλική διεπαφή για τους χρήστες, έτσι ώστε να είναι προσβάσιμη η υπηρεσία από όλη την ακαδημαϊκή και ερευνητική κοινότητα, ακόμα και από τους μη εξειδικευμένους χρήστες. Οι φοιτητές μπορούν να χρησιμοποιήσουν ένα VM για την υλοποίηση μίας εργασίας τους, ή σαν πλατφόρμα για να πειραματιστούν με νέες τεχνολογίες με μηδενικό για αυτούς κόστος, αλλά και πολλά άλλα. Οι καθηγητές μπορούν να δημιουργήσουν εντελώς νέα εργαστήρια (PC labs) με σκοπό την ευκολότερη διεξαγωγή του μαθήματος, χωρίς να υπάρχει η απαίτηση για εύρεση φυσικού χώρου στέγασης του εργαστηρίου, ούτε τα κόστη της αγοράς εξοπλισμού. Οι ερευνητές μπορούν να υλοποιήσουν τα πειράματά τους με μεγάλη αποτελεσματικότητα και ευελιξία. Το αποτέλεσμα είναι ότι όλα τα μέλη της ακαδημαϊκής και ερευνητικής κοινότητας έχουν στην διάθεσή τους ένα πολύ ισχυρό εργαλείο, το οποίο τους γλιτώνει χρόνο, χρήμα, προσπάθεια, αυξάνει την παραγωγικότητα, την αποτελεσματικότητα, αλλά και την ποιότητα της εργασίας τους.

Παρέχει δύο κύρια εργαλεία, μία υπηρεσία χώρου αποθήκευσης, τον Πίθο «Pithos» και μία υπηρεσία παροχής εικονικών μηχανών, τις Κυκλάδες «Cyclades». Ο χώρος αποθήκευσης χρησιμοποιείται για τις αυξημένες ανάγκες αποθήκευσης της κοινότητας. Οι Κυκλάδες χρησιμοποιούνται για την ταχύτατη δημιουργία εικονικών μηχανών από μία λίστα έτοιμων εικόνων «images», υπάρχει όμως και η δυνατότητα εισαγωγής εικόνας από τον χρήστη.

Στην παρούσα διπλωματική χρησιμοποιήθηκε ένας Windows Server 2012, στον οποίο φιλοξενήθηκαν όλα τα ανωτέρω λογισμικά (Node-RED, MariaDB server κτλ.), λειτουργούσε δηλαδή σαν εξυπηρετητής διαδικτύου που φιλοξενούσε την εφαρμογή μας. Έτσι η εφαρμογή λειτουργεί όσο το δυνατόν σαν μία πραγματική διαδικτυακή εφαρμογή.

4

Ανάλυση Απαιτήσεων Συστήματος

Σε αυτό το κεφάλαιο θα καταγραφούν λεπτομέρειες που αφορούν τον σκοπό του συστήματος, τις κατηγορίες των χρηστών, τις παραδοχές που έγιναν, καθώς και τις λειτουργικές και μη λειτουργικές απαιτήσεις του συστήματος. Η διαδικασία αυτή αποτελεί μέρος της προεργασίας που γίνεται κάθε φορά που κάποιος οργανισμός σκοπεύει να παράξει ή να αναθέσει σε τρίτο την δημιουργία σύνθετου λογισμικού. Στο στάδιο αυτό καθορίζονται με σαφήνεια, ακρίβεια και πληρότητα οι λειτουργίες τις οποίες το λογισμικό πρέπει να υλοποιήσει ώστε να εξυπηρετήσει τους μελλοντικούς του χρήστες.

Σκοπός του συστήματος

Σκοπός του συστήματος είναι η παροχή μίας αποκεντρωμένης εφαρμογής (DApp – Decentralized Application) μέσω της οποίας θα πραγματοποιούνται αγοραπωλησίες δεδομένων (μετρήσεων) έξυπνων (IoT) αισθητήρων καιρού. Πιο συγκεκριμένα σκοπός είναι να δοθεί η δυνατότητα σε ιδιοκτήτες τέτοιων αισθητήρων η δυνατότητα καταχώρισης των αισθητήρων τους στο Ethereum blockchain, καθώς και στους αγοραστές η δυνατότητα αγοράς δεδομένων των διαθέσιμων αισθητήρων. Αφενός να προσφέρεται η ίδια ή μεγαλύτερη ευχρηστία που προσφέρουν παραδοσιακές εφαρμογές πελάτη-εξυπηρετητή και αφετέρου η εφαρμογή να είναι αποκεντρωμένη και να χρησιμοποιεί καινοτόμες τεχνολογίες, όπως τα έξυπνα συμβόλαια του Ethereum blockchain.

Κατηγορίες χρηστών

Ανώνυμοι χρήστες

Δεν έχουν λογαριασμό Ethereum. Επισκέπτονται το website από οποιονδήποτε φυλλομετρητή. Μπορούν να περιηγηθούν στο website και να δουν όλους τους διαθέσιμους αισθητήρες καθώς και όλες τις συναλλαγές που έχουν πραγματοποιηθεί στο blockchain και αφορούν τα έξυπνα συμβόλαια της εφαρμογής.

Αγοραστές

Έχουν κάποιον λογαριασμό (EOA) Ethereum. Επισκέπτονται το website με κάποιον φυλλομετρητή ο οποίος τους δίνει πρόσβαση στις κατανεμημένες εφαρμογές του Ethereum (πχ. Mist browser, Metamask plugin). Έχουν πρόσβαση σε ό,τι και οι ανώνυμοι χρήστες και επιπλέον:

- Έχουν την δυνατότητα να αγοράσουν και να πουλήσουν μονάδες του κρυπτονομίσματος NTUA Token.

- Έχουν την δυνατότητα να αγοράζουν δεδομένα αισθητήρων.
- Έχουν πρόσβαση στα δεδομένα που έχουν αγοράσει.

Πωλητές

Έχουν κάποιον λογαριασμό (EOA) Ethereum. Όπως και οι αγοραστές επισκέπτονται το website με κατάλληλο φυλλομετρητή. Έχουν πρόσβαση σε ό,τι και οι ανώνυμοι χρήστες και επιπλέον:

- Έχουν την δυνατότητα να αγοράσουν και να πουλήσουν μονάδες του κρυπτονομίσματος NTUA Token.
- Έχουν την δυνατότητα να καταχωρούν νέους αισθητήρες στο σύστημα.
- Έχουν την δυνατότητα να επεξεργάζονται χαρακτηριστικά των αισθητήρων που έχουν καταχωρίσει στο σύστημα (πχ. αλλαγή της τιμής της κάθε μέτρησης).

Ιδιοκτήτης έξυπνων συμβολαίων

Είναι ο άνθρωπος που ελέγχει τον Externally Owned λογαριασμό Ethereum ο οποίος (ο λογαριασμός) έχει ενεργοποιήσει/δημιουργήσει τα έξυπνα συμβόλαια. Έχει την δυνατότητα αλλαγής της τιμής του NTUA Token καθώς και τον έλεγχο του υπολοίπου του συμβολαίου NtuaToken σε Ethers.

Παραδοχές

Λόγω του γεγονότος ότι η εφαρμογή θα έχει εκπαιδευτικό χαρακτήρα και όχι εμπορικό, έχουν γίνει κάποιες παραδοχές/απλοποιήσεις που στόχο έχουν να μειώσουν την πολυπλοκότητα που θα εισαγόταν, ώστε να φανούν καλύτερα τα χαρακτηριστικά των τεχνολογιών που χρησιμοποιήθηκαν. Έχουν γίνει λοιπόν, οι εξής παραδοχές/απλοποιήσεις που αφορούν την σχεδίαση και την λειτουργία του συστήματος:

- Το Ethereum blockchain προσφέρει ασφάλεια στις συναλλαγές. Η παραδοχή αυτή υιοθετείται από το σύνολο της κοινότητας, αφού οι αλγόριθμοι κρυπτογράφησης που χρησιμοποιούνται θεωρούνται ασφαλείς. Έτσι χρειάζεται μέριμνα από μέρους της εφαρμογής για επιπρόσθετη ασφάλεια.
- Δεν υπάρχουν κακόβουλοι δράστες που υποδύονται τους πωλητές δεδομένων αισθητήρων. Η επίλυση αυτού του προβλήματος θα εμπεριείχε διαδικασίες πιστοποίησης των πωλητών, κάτι που ούτως ή άλλως δεν αποτελεί χαρακτηριστικό των αποκεντρωμένων εφαρμογών.

4.1 Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις εξασφαλίζουν ότι η εφαρμογή θα έχει τα ζητούμενα χαρακτηριστικά και πως θα παρέχει στους χρήστες της τις κατάλληλες λειτουργίες. Οι λειτουργικές απαιτήσεις λοιπόν της εφαρμογής είναι οι εξής:

- Η εφαρμογή να επικοινωνεί με το Ethereum blockchain.

- Να μην υπάρχει συμβατικό login, αλλά να γίνεται διαπίστευση του κάθε χρήστη μόνο μέσω των εργαλείων που προσφέρει το Ethereum blockchain.
- Να μην ζητείται ποτέ και από κανέναν χρήστη το ιδιωτικό κλειδί του Ethereum λογαριασμού του.
- Να προσφέρεται στους χρήστες η εποπτεία όλων των συναλλαγών των έξυπνων συμβολαίων της εφαρμογής στο blockchain και δυνατότητα εύκολης αναζήτησης αυτών.
- Να δίνεται στους χρήστες η δυνατότητα αγοράς και πώλησης του κρυπτονομίσματος NTUA Token έναντι του κρυπτονομίσματος Ether, καθώς και πληροφορίες για το υπόλοιπο του λογαριασμού τους.
- Να δίνεται στους πωλητές η δυνατότητα καταχώρισης νέου αισθητήρα.
- Να δίνεται στους πωλητές/ιδιοκτήτες αισθητήρων η δυνατότητα επεξεργασίας στοιχείων των αισθητήρων τους.
- Να ζητείται από τους πωλητές των αισθητήρων η ύπαρξη μίας διεπαφής (interface) για την πρόσβαση των δεδομένων των αισθητήρων τους.
- Οι χρήστες να μπορούν να αναζητήσουν κάποιον αισθητήρα.
- Όλοι οι διαθέσιμοι αισθητήρες να παρουσιάζονται στους χρήστες, με όλα τους τα χαρακτηριστικά και την θέση τους στον χάρτη.
- Οι χρήστες να μπορούν να αγοράζουν δεδομένα αισθητήρων για χρονικό διάστημα της επιλογής τους.
- Οι χρήστες να μπορούν να δουν και να αναζητήσουν τους αισθητήρες για τους οποίους έχουν αγοράσει δεδομένα.
- Οι χρήστες να έχουν πρόσβαση στα δεδομένα που έχουν αγοράσει.
- Ο ιδιοκτήτης των έξυπνων συμβολαίων να μπορεί να τα ενεργοποιήσει (deploy στο Ethereum blockchain).
- Ο ιδιοκτήτης των έξυπνων συμβολαίων να έχει την δυνατότητα να προβεί σε αλλαγή της τιμής του κρυπτονομίσματος NTUA Token.
- Ο ιδιοκτήτης των έξυπνων συμβολαίων να μπορεί να αποσύρει ή να μεταφέρει πόρους (Ethers) προς τα έξυπνα συμβόλαια.

Ακολουθούν οι λειτουργικές απαιτήσεις των έξυπνων συμβολαίων (smart contracts), τα οποία και θα αποτελέσουν τον κορμό της αποκεντρωμένης εφαρμογής.

Οι λειτουργικές απαιτήσεις του έξυπνου συμβολαίου NtuaToken, το οποίο θα υλοποιεί τις λειτουργίες του κρυπτονομίσματος NTUA Tokens:

- Να μπορεί να διαχειριστεί τα Ether που λαμβάνει, δηλαδή τα Ether σε συναλλαγές στις οποίες παραλήπτης είναι η διεύθυνσή του.
- Να δίνει στον δημιουργό του την δυνατότητα αλλαγής της τιμής σε Ether του NTUA Token.
- Να δίνει στον δημιουργό του την δυνατότητα ανάληψης των Ether τα οποία κατέχει το συμβόλαιο.
- Να είναι διαθέσιμο το υπόλοιπο της κάθε διεύθυνσης (χρήστη) σε NTUA Tokens.
- Να δίνει στους κατόχους των NTUA Tokens την δυνατότητα να τα πουλήσουν πίσω στον ιδιοκτήτη του συμβολαίου.
- Να παρέχει την δυνατότητα μεταφοράς NTUA Tokens από μία διεύθυνση (χρήστη) σε άλλη διεύθυνση (χρήστη).

Οι λειτουργικές απαιτήσεις του έξυπνου συμβολαίου Broker, το οποίο θα υλοποιεί τις λειτουργίες της διαχείρισης των αισθητήρων και της αγοραπωλησίας των δεδομένων τους:

- Να δίνει την δυνατότητα στους πωλητές να καταχωρίσουν τον/τους αισθητήρα/ες που διαθέτουν.
- Να αποθηκεύει στο Ethereum blockchain τα χαρακτηριστικά του κάθε αισθητήρα, όπως τα εισήγαγε ο εκάστοτε πωλητής.
- Να δίνει την δυνατότητα στους πωλητές/ιδιοκτήτες των αισθητήρων να αλλάζουν ορισμένα χαρακτηριστικά των αισθητήρων που κατέχουν, όπως την τιμή της κάθε μέτρησης ή το URL στο οποίο είναι διαθέσιμα τα δεδομένα ή ακόμη και να μεταβιβάσουν σε κάποιον άλλον την κυριότητα του αισθητήρα.
- Να δίνει την δυνατότητα στους χρήστες να αγοράσουν δεδομένα για συγκεκριμένη χρονική στιγμή για κάποιον αισθητήρα της επιλογής τους και να αποθηκεύει κάθε τέτοιου είδους συναλλαγή.

4.2 Μη λειτουργικές απαιτήσεις

Ακολουθούν οι μη λειτουργικές απαιτήσεις που αφορούν όχι τις λειτουργίες της εφαρμογής, αλλά τα ποιοτικά χαρακτηριστικά της.

Ασφάλεια – Ακεραιότητα

- Δεδομένου ότι στην εφαρμογή αυτή θα πραγματοποιούνται οικονομικές συναλλαγές θα πρέπει να υπάρχει υψηλό επίπεδο ασφάλειας.
- Τα χρήματα – κρυπτονομίσματα των χρηστών πρέπει να είναι ασφαλή απέναντι σε επιθέσεις.
- Οι συναλλαγές μεταξύ των χρηστών πρέπει και αυτές να είναι ασφαλείς και να διασφαλίζεται η ακεραιότητά τους.
- Τα έξυπνα συμβόλαια θα πρέπει να είναι απολύτως ασφαλή, δεν επιτρέπεται να υπάρξει κανένα κενό ασφαλείας (bug) στον κώδικά τους, αφού αυτός δεν θα μπορέσει να αλλάξει.
- Στα σημεία όπου εισάγονται δεδομένα στις βάσεις δεδομένων, να ληφθούν κατάλληλα μέτρα για την προστασία του συστήματος από κακόβουλη έγχυση κώδικα (SQL injection).

Ευελιξία

- Η εφαρμογή να μπορεί εύκολα να επεκταθεί, χωρίς να απαιτούνται πολλοί επιπλέον πόροι.
- Επίσης η υλοποίηση να είναι ευέλικτη, δηλαδή να «γενικεύσει» την έννοια του αισθητήρα, έτσι ώστε με λίγη επιπλέον εργασία να μπορούμε να μετατρέψουμε την εφαρμογή για πώληση πχ. δικαιωμάτων φωτογραφιών κτλ.
- Τα έξυπνα συμβόλαια θα πρέπει να έχουν βελτιστοποιημένο κώδικα, ώστε να πραγματοποιούνται οι συναλλαγές γρήγορα και με μικρό κόστος gas (προμήθεια).

Απόδοση – Αποκρισιμότητα

- Σε σημεία της εφαρμογής όπου χρειάζονται δεδομένα από το blockchain και παράλληλα δεν χρειάζεται να σταλεί κάποια συναλλαγή σε αυτό, τα δεδομένα αυτά να είναι αποθηκευμένα σε κάποια βάση δεδομένων του εξυπηρετητή, έτσι ώστε να επιτυγχάνουμε υψηλή απόδοση και αποκρισιμότητα του συστήματος.

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. Θα περιγραφούν επίσης τα κυριότερα προβλήματα που αντιμετωπίστηκαν κατά την ανάπτυξη της εφαρμογής, ώστε να προβληθεί ακόμη περισσότερο η εκπαιδευτική πτυχή της όλης υπόθεσης.

Αυτό που είναι σημαντικό να αναφερθεί στο σημείο αυτό είναι το γεγονός ότι η ευρύτερη τεχνολογική στοίβα ανάπτυξης αποκεντρωμένων εφαρμογών στο Ethereum βρίσκεται ακόμα σε νηπιακό στάδιο. Αυτό έχει ως συνέπεια την μη ύπαρξη προτύπων ακόμη και για συνηθισμένες λειτουργίες, την ταχύτατη εξέλιξη και αλλαγή των διάφορων τεχνολογιών που χρησιμοποιούνται, το σχετικά μικρό, προς το παρόν, μέγεθος της κοινότητας. Αυτή η τάση, δηλαδή η ραγδαία εξέλιξη και αλλαγή βασικών αρχών, αναμένεται να διαρκέσει έως ότου η τεχνολογία γίνει αρκετά ώριμη.

Η παρουσίαση των παραπάνω οργανώνεται στην συνέχεια βάσει των επιμέρους συστατικών της εφαρμογής, ώστε παρουσιάζεται κάθε φορά πλήρως ένα συστατικό προς αποφυγή σύγχυσης του αναγνώστη.

5.1 Έξυπνα συμβόλαια

Τα έξυπνα συμβόλαια – smart contracts αποτελούν το κυριότερο συστατικό στοιχείο κάθε αποκεντρωμένης εφαρμογής. Περιέχουν τον κώδικα που εκτελείται στο Ethereum VM, αποτελούν δηλαδή την ραχοκοκαλιά της εφαρμογής.

Μία ιδιαιτερότητά τους είναι πως ο κώδικάς τους είναι οριστικός και δεν μπορεί να αλλάξει. Άπαξ και ενεργοποιηθεί κάποιο συμβόλαιο στο blockchain, τότε δεν υπάρχει τρόπος να αλλάξουμε την συμπεριφορά του, ούτε να διορθώσουμε κάποιο σφάλμα όσο μικρό και αν αυτό είναι και τούτο διότι, όπως έχει ήδη αναφερθεί, τα περιεχόμενα του blockchain δεν μπορούν να μεταβληθούν. Στην περίπτωση που, κατά την διάρκεια λειτουργίας της εφαρμογής διαπιστώναμε κάποιο κρίσιμο σφάλμα στα συμβόλαια αυτής, ο μόνος τρόπος διόρθωσής του θα ήταν η δημιουργία κάποιου εντελώς νέου συμβολαίου. Τέλος, είναι εύκολο κάποιος προγραμματιστής νέος στην Solidity (η γλώσσα συγγραφής των συμβολαίων) να υποπέσει σε σφάλματα, διότι η Solidity εκ κατασκευής είναι πολύ υψηλού επιπέδου, κάθε γραμμή κώδικα μπορεί να «κρύβει» πολλές λειτουργίες και είναι μία σχετικά νέα γλώσσα και συνεπώς δεν υπάρχει ακόμα η απαιτούμενη εμπειρία.

Εξαιρετική προσοχή δόθηκε λοιπόν στο στάδιο ανάπτυξης των συμβολαίων λόγω της σπουδαιότητάς τους για την ορθή λειτουργία της εφαρμογής και την δυσκολία ορθής ανάπτυξής τους.

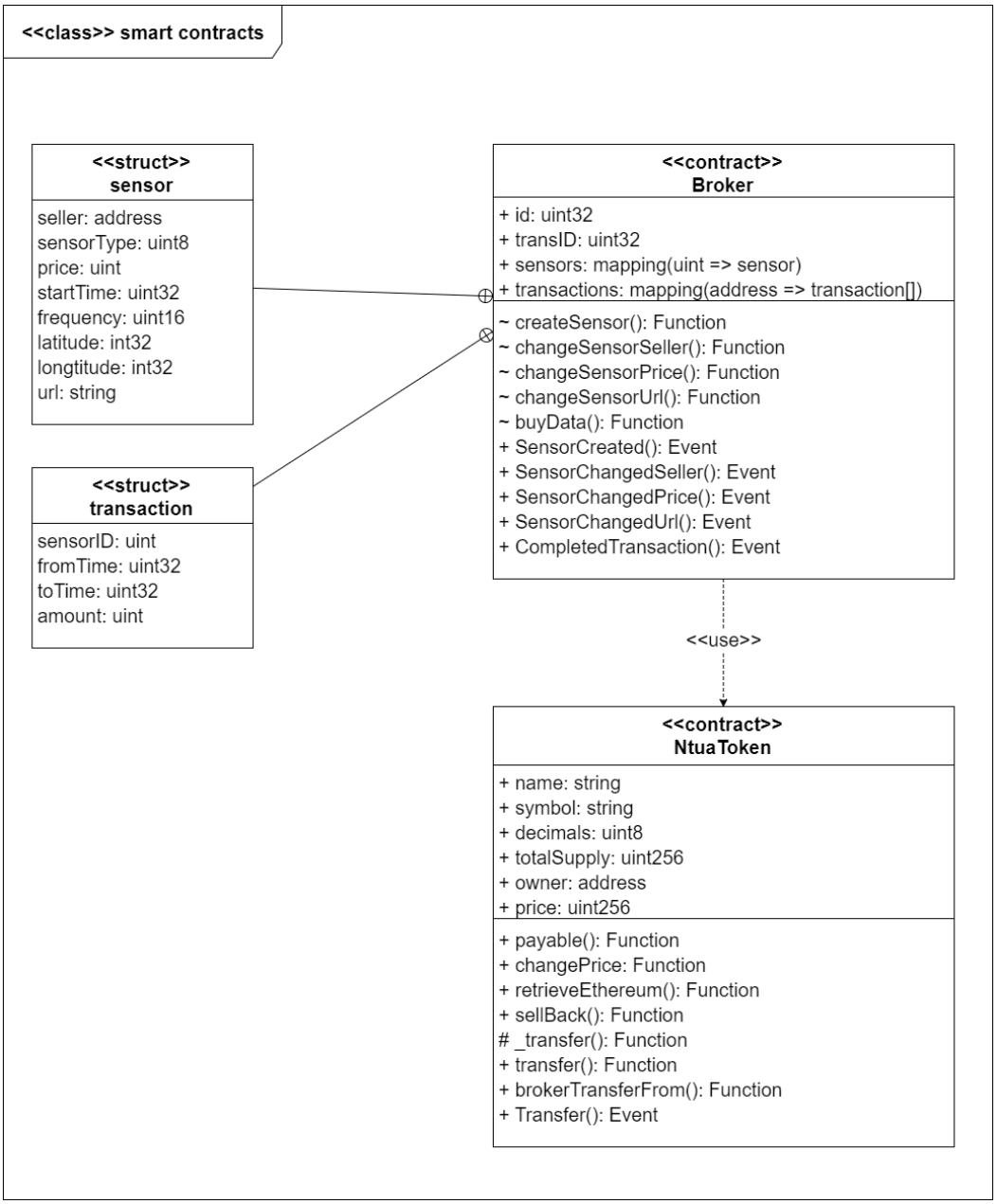
Για την εφαρμογή δημιουργήθηκαν δύο έξυπνα συμβόλαια το NtuaToken και το Broker. Αυτά υλοποιούν όλες τις λειτουργικές απαιτήσεις που τέθηκαν στο προηγούμενο κεφάλαιο.

Το έξυπνο συμβόλαιο NtuaToken δημιουργεί το κρυπτονόμισμα NTUA token (National Technical University of Athens Token). Το νόμισμα αυτό χρησιμοποιείται από την εφαρμογή για όλες τις οικονομικές συναλλαγές μεταξύ των πωλητών και αγοραστών των δεδομένων των αισθητήρων.

Το έξυπνο συμβόλαιο Broker υλοποιεί όλη την λειτουργικότητα που αφορά την διαχείριση των αισθητήρων.

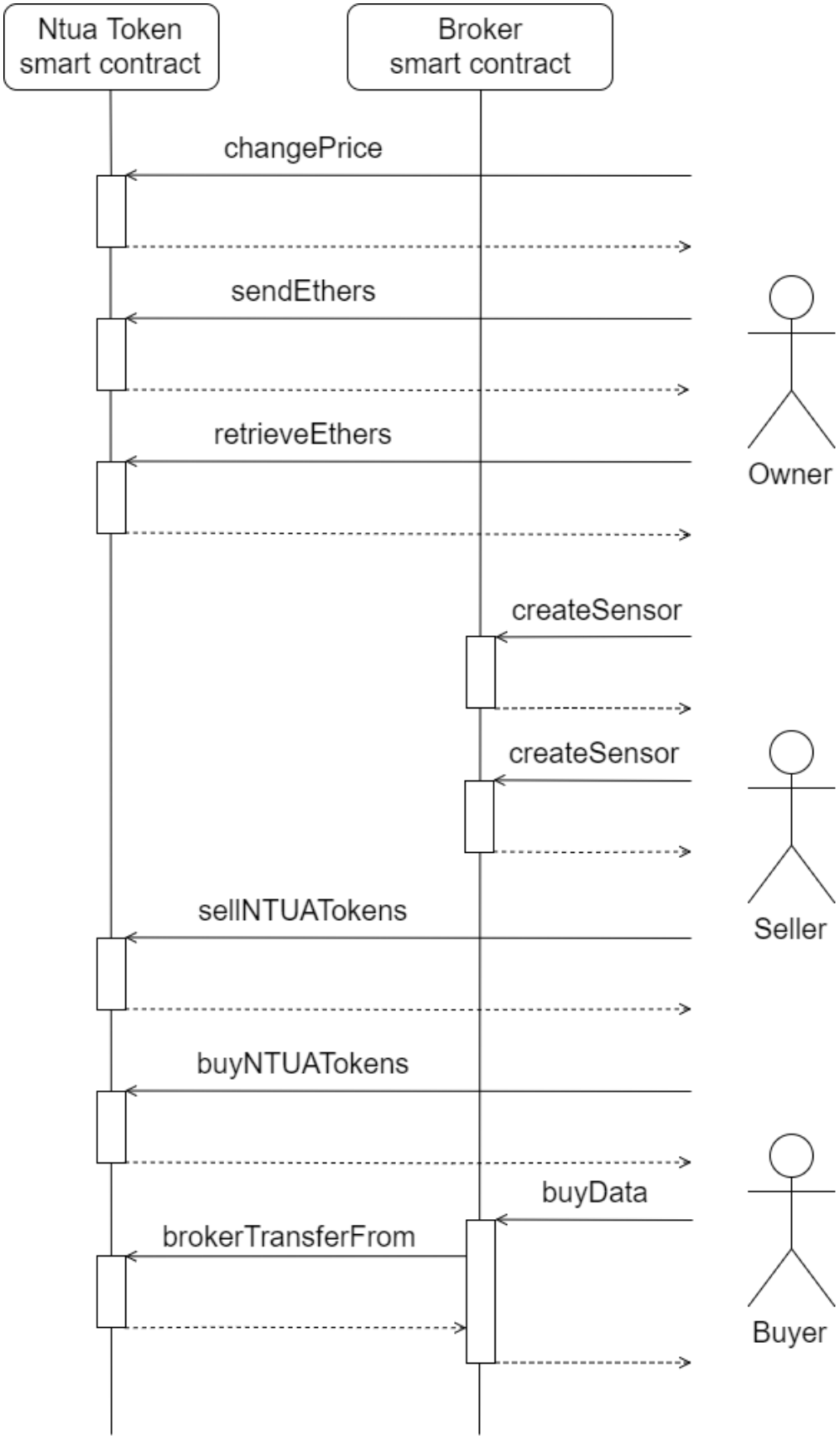
5.1.1 Διαγράμματα UML

Στην συνέχεια παρουσιάζεται το UML διάγραμμα κλάσεων για τα δύο συμβόλαια:



Εικόνα 5.1 UML διάγραμμα κλάσεων έξυπνων συμβολαίων

Ακολουθεί το UML διάγραμμα ακολουθίας για τα δύο συμβόλαια:



Εικόνα 5.2 UML διάγραμμα ακολουθίας έξυπνων συμβολαίων

Δημιούργησα λοιπόν τα δύο έξυπνα συμβόλαια σε Solidity (έκδοση 0.4.16). Ο επεξεργαστής κειμένου και μεταγλωττιστής που χρησιμοποίησα περιλαμβάνονται στο Remix IDE [32], [33].

Ο πηγαίος κώδικας των συμβολαίων NtuaToken και Broker βρίσκεται στο Παράρτημα.

5.1.2 Αναλυτική παρουσίαση συμβολαίου NtuaToken

Στην συνέχεια γίνεται **αναλυτική παρουσίαση** κάθε συνάρτησης/πεδίου του συμβολαίου **NtuaToken**.

```
pragma solidity ^0.4.16
```

Η έκδοση της Solidity που χρησιμοποιείται.

```
event Transfer(address indexed from, address indexed to, uint256
value)
```

Ένα event που ονομάζεται Transfer. Χρησιμοποιείται για την ενημέρωση των κόμβων του δικτύου για την πραγματοποίηση της μεταφοράς value NTUA Tokens από την διεύθυνση from προς την διεύθυνση to.

```
//Constructor-Initializes contract with initial supply tokens to the
creator of the contract
function NtuaToken() {
      balanceOf[msg.sender] = 100000000 * 1 ether ;
      totalSupply = 100000000 * 1 ether ;
      name = "NTUA Token" ;
      symbol= "NTUATok" ;
      decimals = 18 ;
      owner = msg.sender ;
      price = 1000000000000000000 ;          //set initial price
}
```

Ο κατασκευαστής (constructor) του συμβολαίου NtuaToken. Εκτελείται αυτόματα μόλις το συμβόλαιο ενεργοποιηθεί στο blockchain. Όπως βλέπουμε κάνει κάποιες αρχικοποιήσεις των μεταβλητών του συμβολαίου.

msg.sender είναι ο λογαριασμός/διεύθυνση ο οποίος έστειλε (και υπέγραψε) την συναλλαγή. Χρησιμοποιείται κατά κόρον από την Solidity σαν ένας ασφαλής τρόπος για την εξακρίβωση του αποστολέα της συναλλαγής (μηνύματος).

```
//contract receives ethers and gives back tokens
function () payable {
      if(msg.sender != owner) {
           uint256 amount=msg.value*1000000000000000000/price;
           _transfer(owner, msg.sender, amount) ;
      }
}
```

Η συνάρτηση αυτή είναι μία ειδικού τύπου, χωρίς όνομα, συνάρτηση της Solidity. Από το documentation της Solidity [31] έχουμε ότι:

"Fallback Function

A contract can have exactly one unnamed function. This function cannot have arguments and cannot return anything. It is executed on a call to the contract if none of the other functions match the given function identifier (or if no data was supplied at all).

Furthermore, this function is executed whenever the contract receives plain Ether (without data). Additionally, in order to receive Ether, the fallback function must be marked payable. If no such function exists, the contract cannot receive Ether through regular transactions."

Η συνάρτηση αυτή δίνει στο συμβόλαιο την δυνατότητα να δέχεται Ether και να τα κρατάει ως υπόλοιπο στην διεύθυνσή του. Κάθε φορά που στέλνονται στην διεύθυνση του συμβολαίου Ether από τον ιδιοκτήτη/δημιουργό, τότε απλώς αυξάνεται το υπόλοιπο σε Ethers. Κάθε φορά που στέλνονται στην διεύθυνση του συμβολαίου Ethers από κάποιον που δεν είναι ο ιδιοκτήτης, αυτός λαμβάνει τα αντίστοιχα NTUA Tokens.

msg.value είναι το ποσό σε wei (1 ether= 10^{18} wei) που στέλνεται με την συναλλαγή (μήνυμα).

```
//owner changes the price of the token
function changePrice(uint _price) {
      require (msg.sender == owner) ;
      price = _price ;
}
```

Με την συνάρτηση αυτήν μόνον ο ιδιοκτήτης/δημιουργός μπορεί να μεταβάλλει την τιμή, σε Ether, πώλησης του NTUA Token. Το «require (msg.sender == owner)» 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 κατάλληλο μήνυμα.

```
//owner takes ethers from the contract
function retrieveEthereum(uint256 amount) {
      require (msg.sender == owner) ;
      require(this.balance >= amount) ;
      owner.transfer(amount) ;
}
```

Με την συνάρτηση αυτήν μόνον ο ιδιοκτήτης/δημιουργός μπορεί να αποσύρει από την διεύθυνση του συμβολαίου Ethers και να τα μεταφέρει στην δική του διεύθυνση (Externally Owned λογαριασμό).

```
//token holders can sell  their tokens back to the owner of the
contract
function sellBack(uint256 amount, uint _price) {
      require(_price <= price) ;  //only sell the coins if the
asking price is lower than the price of the token
      require(balanceOf[msg.sender] >= amount) ;
      uint256 totalEth = amount*price/1000000000000000000 ;
      require(this.balance >= totalEth) ;
      _transfer(msg.sender, owner, amount) ;
      msg.sender.transfer(totalEth) ;
}
```

Η συνάρτηση αυτή δίνει την δυνατότητα στους χρήστες να «επιστρέψουν» NTUA Tokens και να λάβουν Ether στην τιμή που καθορίζουν. Απαραίτητη προϋπόθεση είναι η τιμή αγοράς να μην είναι μεγαλύτερη από την τρέχουσα τιμή πώλησης. Εάν η συνθήκη ικανοποιείται μεταφέρονται NTUA Tokens από την διεύθυνση του πωλητή προς την διεύθυνση του ιδιοκτήτη του συμβολαίου και Ethers από την διεύθυνση του συμβολαίου προς την διεύθυνση του πωλητή.

```
//Internal transfer, only can be called by this contract
function _transfer(address _from,address _to,uint _value) internal {
      require(_to != 0x0) ;  // Prevent transfer to 0x0 address. Use
burn() instead
      require(balanceOf[_from] >= _value) ; // Check if the sender
has enough
      // Check for overflows
      require(balanceOf[_to] + _value > balanceOf[_to]) ;
      balanceOf[_from] -= _value ; //Subtract from the sender
      balanceOf[_to]+=_value;      //Add the same to the recipient
      Transfer(_from, _to, _value) ;
}
```

Η συνάρτηση αυτή είναι internal, που σημαίνει ότι μπορεί να κληθεί μόνο εσωτερικά του συμβολαίου, δηλαδή μόνο από κάποια άλλη συνάρτηση του ίδιου συμβολαίου. Χρησιμοποιείται για να γίνει η μεταφορά value NTUA Token από την διεύθυνση from προς την διεύθυνση το εάν φυσικά το υπόλοιπο του αποστολέα επαρκεί. Κάθε φορά που η συναλλαγή επιτυγχάνει δημιουργείται ένα event Transfer.

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

Καλείται και από το εξωτερικό του συμβολαίου και μεταφέρει value NTUA Token από την διεύθυνση του αποστολέα της συναλλαγής προς την διεύθυνση το· για να το πετύχει αυτό καλεί την εσωτερική συνάρτηση _transfer με παραμέτρους τα msg.sender, _to, _value.

```
function brokerTransferFrom(address _from, address _to, uint256
_value) returns (bool success) {
      require(msg.sender ==
0xb1719bf761175017eeb6bb72423ece49914fef9b) ; //the broker contract
will never try to steal, we trust it
      _transfer(_from, _to, _value) ;
      return true ;
}
```

Τούτη η συνάρτηση έχει δημιουργηθεί ώστε να καλείται από το συμβόλαιο Broker. Η γραμμή require(msg.sender == 0x3162447df38985e24d38cf3b67181a82b61de56c) εξασφαλίζει ακριβώς αυτό, ότι δηλαδή εκτελείται μόνο εάν αυτός που την έχει καλέσει είναι το συμβόλαιο Broker (η διεύθυνσή του). Εδώ να σημειωθεί πως η διεύθυνση «0x3162447df38985e24d38cf3b67181a82b61de56c» καθορίζεται ντετερμινιστικά από το blockchain, το θέμα αυτό αναλύεται παρακάτω. Να σημειωθεί εδώ ότι η Solidity προσφέρει και την μεταβλητή tx.origin η οποία δίνει το αρχικό αποστολέα της συναλλαγής (full call chain), δηλαδή αυτόν που κάλεσε την συνάρτηση του Broker (συνήθως Externally Owned Account) η οποία στην συνέχεια κάλεσε την συνάρτηση brokerTransferFrom. Ο λόγος που δεν επιλέχθηκε η λύση αυτή αφορά αποκλειστικά την ασφάλεια, αφού το tx.origin έχει μεγάλο κενό ασφαλείας [54].

```
string public name ;                  //name of the token
string public symbol ;                //symbol of the token
uint8 public decimals ;               //decimals of the token
uint256 public totalSupply ;          //total supply of the token
address public owner ;                //the owner of the contract
uint256 public price;      //price of one ntua Token in ethereum
//The array with all balances
mapping (address => uint256) public balanceOf ;
```

Αυτές είναι οι μεταβλητές του συμβολαίου, σε αυτές αποθηκεύονται οι πληροφορίες που υποδηλώνουν και τα ονόματά τους. Ενδιαφέρον παρουσιάζει το mapping, που είναι μία απεικόνιση διευθύνσεων στα υπόλοιπά τους σε NTUA Tokens.

5.1.3 Αναλυτική παρουσίαση συμβολαίου Broker

Στην συνέχεια γίνεται **αναλυτική παρουσίαση** κάθε συνάρτησης/πεδίου του συμβολαίου **Broker**.

```
import "./NtuaToken.sol"
```

Εισαγωγή του αρχείου NtuaToken.sol, έτσι ώστε το συμβόλαιο Broker να μπορεί να χρησιμοποιεί συναρτήσεις του συμβολαίου NtuaToken.

```
NtuaToken Ntua=NtuaToken(0xd50972d2b1747e6277134d29db36e88dacf12844)
```

Δημιουργία αναφοράς στο συμβόλαιο NtuaToken το οποίο βρίσκεται στο blockchain στην διεύθυνση 0xd50972d2b1747e6277134d29db36e88dacf12844 (η διεύθυνση του συμβολαίου καθορίζεται ντετερμινιστικά από το blockchain).

```
struct sensor {
      address seller ;
      uint8 sensorType ;
      uint price ;
      uint32 startTime ;
      uint16 frequency ;          //measurements per hour =>
frequency*3600 measurements per second
      int32 latitude ;
      int32 longitude ;
      string url ;
}
```

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

- Seller: ο δημιουργός πωλητής/ιδιοκτήτης του IoT αισθητήρα
- sensorType: ο τύπος του αισθητήρα, παίρνει τιμές από 1 έως 10
- price: η τιμή της κάθε μέτρησης του αισθητήρα
- startTime: η χρονική στιγμή (32 bit Linux timestamp) της πρώτης μέτρησης του IoT αισθητήρα
- frequency: πόσες μετρήσεις την ώρα πραγματοποιεί ο αισθητήρας
- latitude: το γεωγραφικό πλάτος της θέσης του IoT αισθητήρα
- longitude: το γεωγραφικό μήκος της θέσης του IoT αισθητήρα
- url: η διεύθυνση στην οποία είναι διαθέσιμα τα δεδομένα του IoT αισθητήρα

```
struct transaction {
      uint sensorID ;
      uint32 fromTime ;
      uint32 toTime ;
      uint amount ;
}
```

Η δομή δεδομένων transaction χρησιμοποιείται για την αποθήκευση των στοιχείων κάθε αγοράς δεδομένων αισθητήρων.

- sensorID: το μοναδικό αναγνωριστικό του αισθητήρα του οποίου δεδομένα αγοράστηκαν
- fromTime, toTime: το διάστημα για το οποίο αγοράζονται μετρήσεις
- amount: το συνολικό ποσό σε NTUA Token που πληρώθηκε για την αγορά των δεδομένων

```
event SensorCreated(address indexed seller,uint32 indexed
sensorID,uint8 sensorType,uint price,uint32 startTime, uint16
frequency,int32 latitude,int32 longtitude,string url);
```

Το event αυτό χρησιμοποιείται για την ενημέρωση των κόμβων του δικτύου για την δημιουργία του αισθητήρα με πωλητή τον seller, μοναδικό κωδικό sensorID, τύπο sensorType, τιμή κάθε μέτρησης price, πρώτη μέτρηση την στιγμή startTime, frequency μετρήσεις ανά ώρα, γεωγραφικό πλάτος latitude, γεωγραφικό μήκος longitude, και διαθέσιμα δεδομένα στο url.

```
event SensorChangedSeller(uint32 sensorID, address seller);
```

Το event αυτό χρησιμοποιείται για την ενημέρωση των κόμβων του δικτύου για την αλλαγή του ιδιοκτήτη/πωλητή του αισθητήρα με αριθμό sensorID.

```
event SensorChangedPrice(uint32 sensorID, uint price) ;
```

Το event αυτό χρησιμοποιείται για την ενημέρωση των κόμβων του δικτύου για την αλλαγή της τιμής κάθε μέτρησης του αισθητήρα με αριθμό sensorID.

```
event SensorChangedUrl(uint32 sensorID, string url) ;
```

Το event αυτό χρησιμοποιείται για την ενημέρωση των κόμβων του δικτύου για την αλλαγή της διεύθυνσης στην οποία είναι διαθέσιμα τα δεδομένα του αισθητήρα με αριθμό sensorID.

```
event CompletedTransaction(uint32 transID, address indexed buyer,
uint32 indexed sensorID, uint32 fromTime, uint32 toTime, uint
amount) ;
```

Το event αυτό χρησιμοποιείται για την ενημέρωση των κόμβων του δικτύου για την ολοκλήρωση της αγοράς με αριθμό transID δεδομένων του αισθητήρα με αριθμό sensorID που αφορούν το χρονικό διάστημα από fromTime έως toTime από τον buyer προς amount NTUA Token.

```
//Constructor
function Broker() {
      id = 1 ;
      transID = 1 ;
}
```

Ο κατασκευαστής (constructor) του συμβολαίου Broker. Εκτελείται αυτόματα μόλις το συμβόλαιο ενεργοποιηθεί στο blockchain. Όπως βλέπουμε αρχικοποιεί τις μεταβλητές id και transID του συμβολαίου.

```
function createSensor(address seller1, uint8 type1, uint price1,
uint32 startTime1, uint16 frequency1, int32 latitude1, int32
longitude1, string url1) external {
      sensors[id].seller = seller1 ;
      sensors[id].sensorType = type1 ;
      sensors[id].price = price1 ;
      sensors[id].startTime = startTime1 ;
      sensors[id].frequency = frequency1 ;
      sensors[id].latitude = latitude1 ;
      sensors[id].longitude = longitude1 ;
      sensors[id].url = url1 ;
      SensorCreated(seller1, id, type1, price1, startTime1,
frequency1, latitude1, longitude1, url1) ;
      id++ ;                  //give unique id identifier
}
```

Η (εξωτερική) συνάρτηση που δημιουργεί έναν νέο αισθητήρα. Λαμβάνει ως παραμέτρους τα χαρακτηριστικά του αισθητήρα και δημιουργεί έναν νέο αισθητήρα με αυτά τα χαρακτηριστικά και με αριθμό sensorID=id. Επίσης δημιουργεί και ένα event SensorCreated για να ειδοποιήσει τους κόμβους για την δημιουργία του αισθητήρα.

```
function changeSensorSeller(uint32 sensorID1, address seller1)
external {
      require(msg.sender == sensors[sensorID1].seller) ;
      sensors[sensorID1].seller = seller1 ;
      SensorChangedSeller(sensorID1,seller1) ;
}
```

Η (εξωτερική) συνάρτηση που αλλάζει τον ιδιοκτήτη/πωλητή του αισθητήρα με αριθμό sensorID. Επιτυγχάνει μόνο εάν κληθεί από τον ιδιοκτήτη του αισθητήρα με αριθμό sensorID.

```
function changeSensorPrice(uint32 sensorID1, uint price1) external {
      require(msg.sender == sensors[sensorID1].seller) ;
      sensors[sensorID1].price = price1 ;
      SensorChangedPrice(sensorID1,price1) ;
}
```

Η (εξωτερική) συνάρτηση που αλλάζει την τιμή μέτρησης για τον αισθητήρα με αριθμό sensorID. Επιτυγχάνει μόνο εάν κληθεί από τον ιδιοκτήτη του αισθητήρα με αριθμό sensorID.

```
function changeSensorUrl(uint32 sensorID1, string url1) external {
      require(msg.sender == sensors[sensorID1].seller) ;
      sensors[sensorID1].url = url1 ;
      SensorChangedUrl(sensorID1,url1) ;
}
```

Η (εξωτερική) συνάρτηση που αλλάζει το url στο οποίο είναι διαθέσιμο τα δεδομένα του με αριθμό sensorID. Επιτυγχάνει μόνο εάν κληθεί από τον ιδιοκτήτη του αισθητήρα με αριθμό sensorID.

```
function buyData(uint32 sensorID, uint32 fromTime, uint32 toTime)
external {
      address seller = sensors[sensorID].seller ;
      require(fromTime >= sensors[sensorID].startTime) ;
      uint32 interval = toTime - fromTime ;
      require(interval >= 3600) ;     //no less than 1 hour
      uint amount = interval * sensors[sensorID].frequency *
sensors[sensorID].price / 3600 ;
      Ntua.brokerTransferFrom(msg.sender, seller, amount) ;
      transactions[msg.sender].push(transaction(sensorID, fromTime,
toTime, amount)) ;
      CompletedTransaction(transID, msg.sender, sensorID, fromTime,
toTime, amount) ;
      transID ++ ;
}
```

Μία (εξωτερική) συνάρτηση μέσω της οποίας γίνεται η αγορά δεδομένων στο χρονικό διάστημα από fromTime έως toTime του αισθητήρα με αριθμό sensorID από τον msg.sender. για να επιτύχει, το διάστημα που δίνεται θα πρέπει να μην είναι προγενέστερο της χρονικής στιγμής έναρξης λειτουργίας του αισθητήρα. Επίσης θα πρέπει το διάστημα αυτό να είναι μεγαλύτερο ή ίσο της μίας ώρας και τέλος ο αγοραστής να έχει αρκετά NTUA Tokens για την πραγματοποίηση της αγοράς. Η συνάρτηση αυτή

καλεί την συνάρτηση brokerTransferFrom του συμβολαίου NtuaToken και εάν αυτή επιτύχει, ολοκληρώνεται η αγορά και δημιουργείται ένα νέο event CompletedTransaction για την ειδοποίηση των κόμβων για την δημιουργία του αισθητήρα.

```
//auto generated SensorID
uint32 id ;
```

Το id είναι μία μεταβλητή η οποία αρχικοποιείται στο 1 κατά την δημιουργία του συμβολαίου και αυξάνεται κατά ένα κάθε φορά που δημιουργείται ένας νέος αισθητήρας. Η μεταβλητή αυτή δίνει έναν μοναδικό κωδικό/αριθμό σε κάθε αισθητήρα που καταχωρείται στο blockchain μέσω του συμβολαίου Broker.

```
//auto generated TransactionID
uint32 transID ;
```

Το transID είναι μία μεταβλητή η οποία αρχικοποιείται στο 1 κατά την δημιουργία του συμβολαίου και αυξάνεται κατά ένα κάθε φορά που ολοκληρώνεται μία αγορά δεδομένων/συναλλαγή. Η μεταβλητή αυτή δίνει έναν μοναδικό κωδικό/αριθμό σε κάθε αγοραπωλησία δεδομένων που πραγματοποιείται στο blockchain μέσω του συμβολαίου Broker.

```
//all the sensors in the system (sensorID => sensor)
    mapping(uint => sensor) sensors ;
```

Απεικόνιση του αριθμού/κωδικού του αισθητήρα σε δομή sensors που περιέχει στα χαρακτηριστικά του.

```
//all the transactions completed (buyer => sensorID)
    mapping(address => transaction[]) transactions ;
```

Απεικόνιση της κάθε διεύθυνσης σε πίνακα της δομής transaction που περιέχει τα στοιχεία όλων των αγορών που έχουν πραγματοποιηθεί από την διεύθυνση αυτήν.

5.1.4 Παρατηρήσεις

Παρατήρηση: Έχει γίνει μεγάλη προσπάθεια βελτιστοποίησης του κώδικα των συμβολαίων οι συναρτήσεις τους να έχουν την ελάχιστη δυνατή απαίτηση σε gas, στοιχείο πολύ σημαντικό, αφού έτσι μειώνεται σε μεγάλο βαθμό το επιπλέον κόστος των προμηθειών που δίνονται για την ολοκλήρωση των συναλλαγών. Μικρή απαίτηση σε gas μεταφράζεται σε μικρότερο κόστος για την συναλλαγή και γρηγορότερη διεκπεραίωση της συναλλαγής.
Παρατήρηση: Στο συμβόλαιο NtuaToken χρησιμοποιούνται public μέθοδοι, ενώ στο συμβόλαιο Broker αντί για public, χρησιμοποιούνται external μέθοδοι. Αυτό γίνεται για να μειωθεί το κόστος (gas) που απαιτείται για την εκτέλεση της συνάρτησης. Όταν μία συνάρτηση είναι external και καλείται έξω από το συμβόλαιο με πολλά δεδομένα τότε συνήθως απαιτεί λιγότερους πόρους (gas) για την εκτέλεση της (σε σχέση με την public). Οι external συναρτήσεις του Broker επιτυγχάνουν οικονομία στο gas σε σχέση με αν ήταν public. Στο συμβόλαιο NtuaToken οι συναρτήσεις καλούνται πάντα με μικρό όγκο δεδομένων, οπότε το να ήταν external δεν θα συνέβαλε στην οικονομία του

gas. Ειδικά οι συναρτήσεις createSensor και changeSensorUrl που λαμβάνουν στις παραμέτρους τους μεταβλητή τύπου string η οποία μπορεί να έχει πολύ μεγάλο μέγεθος, πετυχαίνουν μεγάλη οικονομία στο gas όντας external, σε σχέση με το αν ήταν public.

Καθορισμός της διεύθυνσης ενός έξυπνου συμβολαίου.

Η διεύθυνση του κάθε συμβολαίου υπολογίζεται ντετερμινιστικά στο Ethereum blockchain. Εξαρτάται αποκλειστικά από την διεύθυνση του δημιουργού του και από τον αριθμό των συναλλαγών που αυτός έχει στείλει στο blockchain (nonce). Αυτά τα δύο κρυπτογραφούνται με τον αλγόριθμο RLP [55] και το αποτέλεσμα περνάει στην συνέχεια μέσα από την συνάρτηση κατακερματισμού Keccak-256 [56]. Στο NODE-Red στην ροή Deploy contracts, στον κόμβο future contract address γίνεται ακριβώς αυτός ο υπολογισμός, έτσι ώστε να μπει στο συμβόλαιο Broker η σωστή διεύθυνση για το NtuaToken και το αντίστροφο.

Για την σωστή λειτουργία των συμβολαίων, αφού υπολογίσουμε τις διευθύνσεις τις οποίες θα λάβουν πρέπει να τις αλλάξουμε στον πηγαίο κώδικά τους πριν το deployment τους.

Έχουν υλοποιηθεί στο Node-RED τρεις διαφορετικοί τρόποι για το deployment των έξυπνων συμβολαίων, είναι τρεις αποκλειστικά για λόγους εκπαιδευτικούς και επίδειξης. Ο πρώτος χρησιμοποιεί το ABI (Application Binary Interface) των συμβολαίων, όπως αυτό προκύπτει από την μεταγλώττιση του Solidity κώδικα στο Remix. Ο δεύτερος χρησιμοποιεί το πακέτο solc (ο μεταγλωττιστής της Solidity για χρήση από την JavaScript), με την βοήθεια του οποίου διαβάζεται ο πηγαίος κώδικας, γίνεται επί τόπου η μεταγλώττιση και ακολουθεί το deployment. Ο τρίτος τρόπος, χρησιμοποιεί το ABI, όπως αυτό προκύπτει από την μεταγλώττιση του Solidity κώδικα στο Remix, όμως προσφέρει στον admin μία γραφική διεπαφή για το deployment, μέσω μίας ιστοσελίδας.

5.2 IoT αισθητήρες καιρού

Η εφαρμογή υποστηρίζει 10 διαφορετικούς τύπους αισθητήρων καιρού, με πρόβλεψη εάν κριθεί αναγκαίο την εύκολη προσθήκη έως 245 επιπλέον τύπων αισθητήρων, χωρίς να αλλάξουν βασικά στοιχεία της αρχιτεκτονικής του συστήματος.

Οι δέκα τύποι αισθητήρων καιρού που υποστηρίζονται:

Κωδικός τύπου	Τύπος αισθητήρα	Προτεινόμενη μονάδα μέτρησης
1	Θερμοκρασίας – Θερμόμετρο (Temperature)	Βαθμοί κελσίου – °C
2	Σχετικής υγρασίας – Υγρόμετρο (Relative humidity)	Ποσοστό επί τοις εκατό – %
3	Ατμοσφαιρικής πίεσης – Βαρόμετρο (Pressure)	hecto Pascal – hPa
4	Ορατότητας (Visibility)	Χιλιόμετρα – km
5	Ανέμου (ταχύτητα και διεύθυνση) – Ανεμόμετρο (Wind speed and direction)	μέτρα ανά δευτερόλεπτο – m/s και μετεωρολογικές μοίρες – °
6	Κάλυψης ουρανού από σύννεφα (Sky cloud coverage)	Ποσοστό επί τοις εκατό – %
7	Σημείου δρόσου ή σημείο υγροποίησης ή σημείο κόρου ατμόσφαιρας (Dew point)	Βαθμοί κελσίου – °C
8	Ηλιακής ακτινοβολίας – Πυρανόμετρο (Solar Radiation)	watt/m²
9	Δείκτης υπέρυθρης (UV) ακτινοβολίας (UV index)	Κλίμακα 0 έως 11
10	Πυκνότητα στρώματος όζοντος της ατμόσφαιρας (Columnar density of total atmospheric ozone layer)	Μονάδες Dobson – DU

Πίνακας 5.1 Τύποι IoT αισθητήρων καιρού

Να σημειωθεί πως εφαρμογή αναπτύχθηκε ως μία πλατφόρμα διαμοιρασμού δεδομένων αισθητήρων. Έχει δηλαδή σκοπό να συνδέσει μέσω του blockchain τους ιδιοκτήτες των αισθητήρων με τους πιθανούς αγοραστές δεδομένων. Σε κανονικές συνθήκες λειτουργίας δεν διαθέτουμε κάποιον αισθητήρα, ούτε διατηρείται κάποια βάση δεδομένων που περιέχει δεδομένα αισθητήρων, αλλά αυτά διατηρούνται από τους ιδιοκτήτες των αυτών (των αισθητήρων). Για τις ανάγκες της ανάπτυξης και της παρουσίασης της εφαρμογής όμως, δεδομένου ότι δεν βρίσκεται σε λειτουργία στον πραγματικό κόσμο ώστε να καταχωρηθούν αληθινοί αισθητήρες, έγινε προσομοίωση των αισθητήρων.

Ο εκάστοτε ιδιοκτήτης πραγματικών IoT αισθητήρων καιρού έχει κάποιον μηχανισμό με τον οποίο λαμβάνει τα δεδομένα από τους αισθητήρες του και τα αποθηκεύει ώστε να είναι διαθέσιμα όταν η εφαρμογή μας του τα ζητήσει.

Για την προσομοίωση των IoT αισθητήρων καιρού χρησιμοποιήθηκαν τρεις διαφορετικοί μετεωρολογικοί οργανισμοί, ο Open Weather Map, ο The Weather Underground και τέλος ο Dark Sky. Οι ιστοσελίδες τους είναι οι εξής: www.openweathermap.org, www.wunderground.com, www.darksky.net. Και οι τρεις προσφέρουν προγραμματιστικές διεπαφές (APIs) και προτιμήθηκαν έναντι άλλων, διότι για αυτές υπήρχε έτοιμος κόμβος στο Node-RED, κάτι που μειώνει την πολυπλοκότητα και οδηγεί σε πιο κομψή και περιεκτική υλοποίηση.

5.3 Βάσεις δεδομένων

Η εφαρμογή θεωρητικά θα μπορούσε να λειτουργήσει χωρίς την χρήση κάποιας βάσης δεδομένων, αφού όλα τα δεδομένα που την αφορούν αποθηκεύονται στο blockchain (όχι τα δεδομένα των αισθητήρων). Αυτή είναι και η ουσία της αποκεντρωμένης εφαρμογής (DApp), κανένας οργανισμός να μην μπορεί να ελέγχει μονόπλευρα την λειτουργία της.

Παρόλο που η λειτουργία χωρίς βάσεις δεδομένων είναι θεωρητικά εφικτή, στην πραγματικότητα μία τέτοια προσέγγιση θα έκανε την εφαρμογή πολύ δύσκολη στην χρήση. Ο λόγος δεν είναι άλλος από την απόδοση και την χρονοκαθυστέρηση, εγγενή χαρακτηριστικά του blockchain.

Στην συνέχεια περιγράφεται το σενάριο λειτουργίας άνευ βάσεων δεδομένων, ώστε να δειχθεί ο λόγος ύπαρξης βάσης δεδομένων.

Όταν κάποιος χρήστης επισκέπτεται την ιστοσελίδα, πρέπει οπωσδήποτε να μπορεί να δει τους διαθέσιμους αισθητήρες όπως και τα δεδομένα που έχει αγοράσει. Εφόσον ο εξυπηρετητής δεν έχει την πληροφορία αυτή κάπου αποθηκευμένη, αυτή θα πρέπει να ανακτάται από το blockchain κάθε φορά που φορτώνεται μία σελίδα που την χρειάζεται. Για την επίτευξη αυτού υπάρχουν δύο επιλογές, ανάκτηση της πληροφορίας αυτής από το blockchain στον εξυπηρετητή κάθε φορά που λαμβάνει μία σχετική αίτηση ή ανάκτησή της (της πληροφορίας) στον φυλλομετρητή του πελάτη.

Η πρώτη λύση αφενός δεν ενδείκνυται σαν καλή πρακτική από την κοινότητα του Ethereum blockchain, διότι αυξάνει το υπολογιστικό βάρος στον εξυπηρετητή, αφετέρου θα εισήγαγε πάρα πολύ μεγάλη αχρείαστη πολυπλοκότητα, θα αύξανε κατακόρυφα τις απαιτήσεις σε πόρους, την καθυστέρηση εξυπηρέτησης των αιτήσεων, δηλαδή θα μείωνε σε πολύ μεγάλο βαθμό την απόδοση. Η καλή πρακτική προστάζει το υπολογιστικό βάρος να αφορά τον πελάτη και όχι τον εξυπηρετητή, ο οποίος υπάρχει για να εκτελεί την ελάχιστη δυνατή εργασία ή ακόμη απουσιάζει πλήρως σε μερικές εφαρμογές. Επομένως, στην φύση της εφαρμογής θα ταίριαζε περισσότερο η δεύτερη λύση.

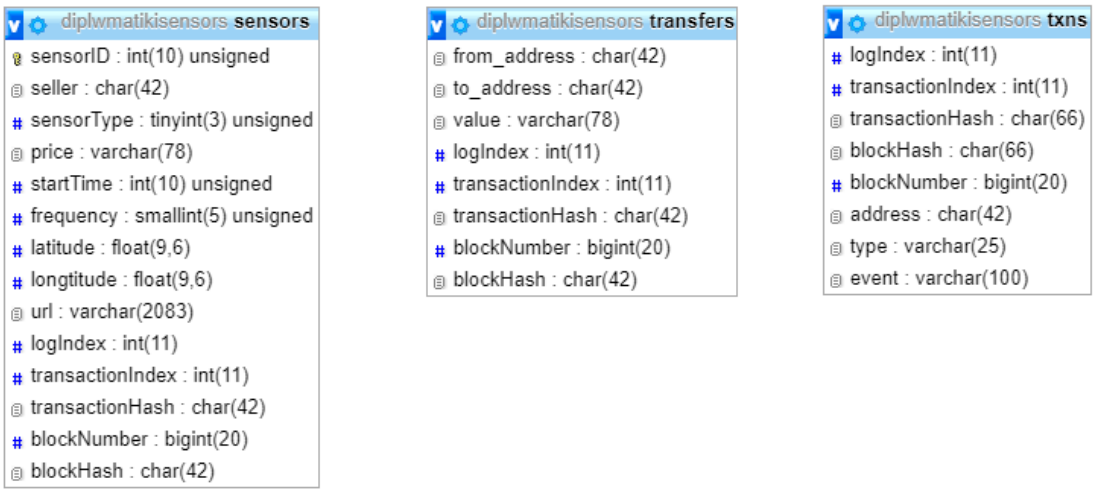
Η δεύτερη λύση αν και ικανοποιεί την απαίτηση για μεταφορά της υπολογιστικής πολυπλοκότητας από τον εξυπηρετητή στον πελάτη και προκρίνεται από την κοινότητα ανάπτυξης αποκεντρωμένων εφαρμογών, θεωρήθηκε ακατάλληλη για την εφαρμογή αυτή. Πιο συγκεκριμένα απορρίφθηκε και αυτή για λόγους απόδοσης και εισαγωγής καθυστέρησης. Για παράδειγμα κάθε φορά που κάποιος χρήστης θα ήθελε να δει τους διαθέσιμους αισθητήρες, θα έπρεπε να διαβάσει όλα τα μπλοκ[2] από την ενεργοποίηση των συμβολαίων έως το τελευταίο, ώστε να βρει όλα τα events SensorCreated που έχουν παραχθεί και σηματοδοτούν την καταχώριση αισθητήρων· έπειτα θα πρέπει να

[2] Η διαδικασία αυτή θα γινόταν με την χρησιμοποίηση client-side JavaScript.

βρει και όλα τα events SensorChangedSeller, SensorChangedPrice, SensorChangedUrl, ώστε να είναι σίγουρος ότι βλέπει τα τρέχοντα στοιχεία κάθε αισθητήρα. Συνέπεια των παραπάνω θα ήταν αυξημένη καθυστέρηση στο φόρτωμα κάθε ιστοσελίδας. Το φαινόμενο αυτό θα γινόταν όλο και χειρότερο με το πέρασμα του χρόνου, αφού το blockchain συνεχώς μεγαλώνει, για την ακρίβεια προστίθεται ένα μπλοκ κάθε 10-19 δευτερόλεπτα (ο στόχος είναι τα 12 δευτερόλεπτα), με το μέγεθός του τους τελευταίους μήνες να είναι σταθερά πάνω από 10 kBytes. Με έναν πρόχειρο υπολογισμό (θεωρώντας πως δεν θα αλλάξει δραματικά το μέγεθος του μπλοκ) συμπεραίνουμε ότι μετά από έναν μόνο χρόνο λειτουργίας της εφαρμογής ο πελάτης θα έπρεπε να εξετάσει περίπου $\frac{60\ sec*60\ min*24\ hours*365\ days}{20\frac{sec}{block}} * 10\frac{kBytes}{block} = 788400\ kBytes$ δεδομένων πριν την φόρτωση κάθε ιστοσελίδας κάτι που είναι μη αποδεκτό. Έτσι λοιπόν η λύση αυτή απορρίπτεται ως ακατάλληλη.

Επομένως η καλύτερη λύση περιλαμβάνει την χρήση βάσεων δεδομένων και είναι η μόνιμη παρακολούθηση (σε πραγματικό χρόνο) του blockchain από τον εξυπηρετητή και η αποθήκευση των πληροφοριών όλων των event που αφορούν τα δύο συμβόλαια της εφαρμογής NtuaToken και Broker. Τα οφέλη αυτής της λύσης αφορούν τόσο την εξοικονόμηση πόρων όσο και την απόδοση και βελτίωση της εμπειρίας των χρηστών. Ο εξυπηρετητής «διαβάζει» σε πραγματικό χρόνο κάθε νέο μπλοκ του Ethereum blockchain και εάν βρει κάποιο από τα ζητούμενα events το αποθηκεύει. Όταν κάποιος χρήστης κάνει αίτηση μίας ιστοσελίδας που περιλαμβάνει πληροφορία που υπάρχει στα events, παραδείγματος χάριν λίστα των διαθέσιμων αισθητήρων, τότε ο εξυπηρετητής απλά ανακτά την ζητηθείσα πληροφορία από την βάση δεδομένων που διατηρεί και την στέλνει στον πελάτη. Η καθυστέρηση και η χρησιμοποίηση υπολογιστικών πόρων είναι οι ελάχιστες δυνατές. Έτσι η εφαρμογή πετυχαίνει υψηλή απόδοση.

Δημιουργήθηκε λοιπόν μία σχεσιακή βάση δεδομένων η οποία περιέχει όλους τους αισθητήρες με τα στοιχεία τους και όλες τις συναλλαγές που αφορούν τα δύο συμβόλαια όπως ακριβώς υπάρχουν στο blockchain. Δηλαδή η βάση δεδομένων περιέχει μόνο το υποσύνολο εκείνο της πληροφορίας του blockchain που μας ενδιαφέρει. Ακολουθεί το σχεσιακό (relational) σχήμα της βάσης δεδομένων:



Εικόνα 5.3 Σχεσιακό σχήμα βάσης δεδομένων

Για την υλοποίηση της βάσης χρησιμοποιήθηκε το σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDBMS) MariaDB.

Όπως φαίνεται, διατηρούνται τρεις διαφορετικοί πίνακες. Ένας πίνακας, ο sensors, που διατηρεί στις εγγραφές του όλους τους διαθέσιμους αισθητήρες με τα χαρακτηριστικά τους. Ένας πίνακας, ο transfers, που διατηρεί στις εγγραφές του όλες τις μεταφορές NTUA Token που έχουν πραγματοποιηθεί. Ένας πίνακας, ο txns, που διατηρεί όλα τα event που σχετίζονται με τα δύο συμβόλαια της εφαρμογής. Στην συνέχεια παρουσιάζεται αναλυτικά κάθε στοιχείο κάθε πίνακα.

Ο πίνακας **sensors** μεταβάλλεται όταν «διαβάζεται» από το blockchain κάποιο από τα event SensorCreated, SensorChangedSeller, SensorChangedPrice, SensorChangedUrl. Έχει τα εξής πεδία:

Όνομα	Τύπος	Περιεχόμενο πεδίου
sensorID	int unsigned	Ο μοναδικός αριθμός/αναγνωριστικό του αισθητήρα.
seller	char(42)	Η Ethereum διεύθυνση του ιδιοκτήτη/πωλητή του αισθητήρα.
sensorType	tinyint unsigned	Ο τύπος του αισθητήρα, παίρνει τιμές από 1 έως 10.
price	varchar(78)	Η τιμή της κάθε μέτρησης του αισθητήρα.
startTime	int unsigned	Η χρονική στιγμή (Linux timestamp) της πρώτης μέτρησης του IoT αισθητήρα.
frequency	smallint(5) unsigned	Πόσες μετρήσεις την ώρα πραγματοποιεί ο αισθητήρας.
latitude	float(9,6)	Το γεωγραφικό πλάτος της θέσης του IoT αισθητήρα.
longtitude	float(9,6)	Το γεωγραφικό μήκος της θέσης του IoT αισθητήρα.
url	varchar (2083)	Η διεύθυνση στην οποία είναι διαθέσιμα τα δεδομένα του IoT αισθητήρα.
logIndex	int	Integer of the log index position in the block.
transactionIndex	int	Integer of the transactions index position log was created from.
transactionHash	char(42)	Hash of the transactions this log was created from.
blockNumber	bigint	The block number where this log was in.
blockHash	char(42)	Hash of the block where this log was in.

Πίνακας 5.2 Ο πίνακας sensors της βάσης δεδομένων

Τα events που μεταβάλλουν το περιεχόμενο του πίνακα sensors:

```
event SensorCreated(address indexed seller,uint32 indexed
sensorID,uint8 sensorType,uint price,uint32 startTime, uint16
frequency,int32 latitude,int32 longtitude,string url);
event SensorChangedSeller(uint32 sensorID,address seller);
event SensorChangedPrice(uint32 sensorID, uint price) ;
event SensorChangedUrl(uint32 sensorID, string url) ;
```

Όταν «διαβαστεί» το event SensorCreated δημιουργείται μία νέα εγγραφή στον πίνακα η οποία αντιστοιχεί στον νέο αισθητήρα που καταχωρήθηκε στο blockchain.

Όταν «διαβαστεί» το event SensorChangedSeller τότε ανανεώνεται το πεδίο seller της εγγραφής του αισθητήρα με αριθμό sensorID.

Όταν «διαβαστεί» το event SensorChangedPrice τότε ανανεώνεται το πεδίο price της εγγραφής του αισθητήρα με αριθμό sensorID.

Όταν «διαβαστεί» το event SensorChangedUrl τότε ανανεώνεται το πεδίο url της εγγραφής του αισθητήρα με αριθμό sensorID.

Ο πίνακας **transfers** μεταβάλλεται όταν «διαβάζεται» από το blockchain ένα νέο event Transfer. Έχει τα εξής πεδία:

Όνομα	Τύπος	Περιεχόμενο πεδίου
fromAddress	char(42)	Ο αποστολέας των NTUA Tokens.
toAddress	char(42)	Ο παραλήπτης των NTUA Tokens.
value	varchar(78)	Το ποσόν NTUA Tokens που στάλθηκε.
logIndex	int	Integer of the log index position in the block.
transactionIndex	int	Integer of the transactions index position log was created from.
transactionHash	char(42)	Hash of the transactions this log was created from.
blockNumber	bigint	The block number where this log was in.
blockHash	char(42)	Hash of the block where this log was in.

Πίνακας 5.3 Ο πίνακας transfers της βάσης δεδομένων

Το event που μεταβάλλει το περιεχόμενο του πίνακα transfers είναι το:

Transfer(from, to, value), το οποίο στέλνεται από το συμβόλαιο NtuaToken κάθε φορά που πραγματοποιείται μεταφορά NTUA Tokens.

Ο πίνακας **txns** μεταβάλλεται όταν «διαβάζεται» από το blockchain οποιοδήποτε event έχει σχέση με τα συμβόλαια της εφαρμογής. Έχει τα εξής πεδία:

Όνομα	Τύπος	Περιεχόμενο πεδίου
logIndex	int	Integer of the log index position in the block.
transactionIndex	int	Integer of the transactions index position log was created from.
transactionHash	char(42)	Hash of the transactions this log was created from.
blockNumber	bigint	The block number where this log was in.
blockHash	char(42)	Hash of the block where this log was in.
address	char(42)	Η διεύθυνση του συμβολαίου το οποίο δημιούργησε το event.
type	varchar(25)	Ο τύπος του block.
event	varchar(42)	Το όνομα του event.

Πίνακας 5.4 Ο πίνακας txns της βάσης δεδομένων

Η εφαρμογή αναπτύχθηκε ως μία πλατφόρμα διαμοιρασμού δεδομένων αισθητήρων. Δηλαδή σκοπό έχει να συνδέσει μέσω του blockchain τους ιδιοκτήτες των αισθητήρων με τους πιθανούς αγοραστές δεδομένων. Σε κανονικές συνθήκες λειτουργίας δεν διατηρείται κάποια βάση δεδομένων που περιέχει δεδομένα αισθητήρων, αλλά αυτά διατηρούνται από τους ιδιοκτήτες των αυτών (των αισθητήρων). Για τις ανάγκες της ανάπτυξης και της παρουσίασης της εφαρμογής, δεδομένου ότι δεν βρίσκεται σε

λειτουργία στον πραγματικό κόσμο, ώστε να υπάρχουν πραγματικοί ιδιοκτήτες αισθητήρων, δημιουργήθηκε μία βάση δεδομένων που περιέχει τα δεδομένα έξυπνων αισθητήρων καιρού. Έτσι γίνεται μία ικανοποιητική, όσον αφορά την εφαρμογή, προσομοίωση του πραγματικού ιδιοκτήτη των IoT αισθητήρων καιρού.

Η βάση δεδομένων αυτή ονομάζεται diplwmatikiData. Περιέχει 10 διαφορετικούς τύπους πινάκων, έναν τύπο πίνακα για κάθε τύπο αισθητήρα. Όλοι οι πίνακες έχουν ένα πεδίο που ονομάζεται time και αντιστοιχεί στην χρονική στιγμή (Linux timestamp) λήψης της μέτρησης. Ακολουθεί πίνακας με τον τύπο του κάθε αισθητήρα και τον αντίστοιχο τύπο πίνακα στην βάση δεδομένων.

Α/Α	Τύπος αισθητήρα	Μονάδα μέτρησης	Όνομα πεδίου	Τύπος
1	Θερμοκρασίας	Βαθμοί κελσίου – °C	temperature	FLOAT(10,2)
2	Σχετικής υγρασίας	Ποσοστό επί τοις εκατό – %	humidity	TINYINT UNSIGNED
3	Ατμοσφαιρικής πίεσης	hecto Pascal – hPa	pressure	SMALLINT UNSIGNED
4	Ορατότητας	Χιλιόμετρα – km	visibility	SMALLINT UNSIGNED
5	Ανέμου	μέτρα ανά δευτερόλεπτο – m/s και μετεωρολογικές μοίρες – °	speed direction	FLOAT(10,2) UNSIGNED SMALLINT UNSIGNED
6	Κάλυψης ουρανού από σύννεφα	Ποσοστό επί τοις εκατό – %	cloudCoverage	TINYINT UNSIGNED
7	Σημείου δρόσου	Βαθμοί κελσίου – °C	dewPoint	FLOAT(10,2)
8	Ηλιακής ακτινοβολίας	watt/m²	solarRad	SMALLINT UNSIGNED
9	Δείκτης υπέρυθρης (UV) ακτινοβολίας	Κλίμακα 0 έως 11	uv	SMALLINT UNSIGNED
10	Πυκνότητα στρώματος όζοντος της ατμόσφαιρας	Μονάδες Dobson – DU	ozone	FLOAT(16,2) UNSIGNED

Πίνακας 5.5 Μετρήσεις αισθητήρων στην βάση δεδομένων

5.4 Οι ροές του Node-RED

Όπως έχει ήδη αναφερθεί, η εφαρμογή που δημιουργήθηκε είναι αποκεντρωμένη, αυτό σημαίνει ότι μπορεί να λειτουργεί πάνω στο Ethereum blockchain και από αυτό να «παίρνει» όσους υπολογιστικούς πόρους χρειάζεται. Βέβαια οι τεχνολογίες στις οποίες βασίζεται βρίσκονται ακόμα σε νηπιακό στάδιο και συνεπώς, ενώ οι χρήστες που «τρέχουν» έναν Ethereum κόμβο μπορούν να στέλνουν συναλλαγές, να καλούν τις συναρτήσεις των έξυπνων συμβολαίων και έτσι να γίνεται η καταχώριση και η αγορά δεδομένων αισθητήρων, δεν προσφέρεται από το blockchain κάποια ικανοποιητική διεπαφή για την αλληλεπίδραση των χρηστών αυτών με τα έξυπνα συμβόλαια. Έτσι, δημιουργήθηκε ένας ιστότοπος που αποτελεί ένα φιλικό προς κάθε χρήστη περιβάλλον για την χρήση της εφαρμογής.

Ο εξυπηρετητής του website αυτού είναι επιφορτισμένος να παρέχει στους χρήστες τις απαραίτητες διεπαφές και να διατηρεί, όπως αναλύεται στην προηγούμενη ενότητα, ορισμένες βάσεις δεδομένων. Ο εξυπηρετητής επιτελεί το ελάχιστο δυνατό έργο και δεν είναι τόσο σημαντικός για την λειτουργία αυτής της εφαρμογής όσο θα ήταν εάν ακολουθείτο το παραδοσιακό μοντέλο πελάτη εξυπηρετητή. Δεν υπάρχει μεγάλη ανάγκη για διασφάλιση των δεδομένων στον εξυπηρετητή, αφού όλο το περιεχόμενο των βάσεων δεδομένων βρίσκεται και στο Ethereum blockchain. Τέλος δεν υπάρχουν ιδιαίτερες απαιτήσεις για ασφάλεια από τον εξυπηρετητή.

Δεν υπάρχει ανάγκη για ιδιαίτερη μέριμνα για την ασφάλεια των δεδομένων που είναι αποθηκευμένα στον εξυπηρετητή, διότι αυτά (τα δεδομένα) ούτε είναι ιδιωτικά (πχ. κωδικοί), ώστε να χρειάζονται προστασία έναντι υποκλοπής, ούτε είναι μοναδικά, αφού υπάρχουν και στο blockchain.

Ενδεχόμενη καταστροφή της βάσης από κάποιον κακόβουλο πράκτορα, δεν επηρεάζει την διαθεσιμότητα της εφαρμογής. Το μόνο που θα συμβεί, θα είναι η μη διαθεσιμότητα μέρους της πληροφορίας που βρίσκεται αποθηκευμένη στο blockchain για μικρό χρονικό διάστημα. Μόλις ο εξυπηρετητής αντιληφθεί ότι καταστράφηκε η βάση δεδομένων στην οποία είναι αποθηκευμένοι όλοι οι αισθητήρες, οι αγορές και γενικότερα οι συναλλαγές που έχουν σταλεί στο blockchain και αφορούν τα δύο έξυπνα συμβόλαια NtuaToken και Broker θα αρχίσει να διαβάζει το blockchain, ψάχνοντας για τα κατάλληλα events και θα ξαναδημιουργήσει τις βάση σε μικρό χρονικό διάστημα.

Ενδεχόμενη επίθεση από κακόβουλους χρήστες για την υποκλοπή ιδιωτικών δεδομένων των χρηστών δεν θα έχει καμία απολύτως επιτυχία, για τον απλό λόγο ότι δεν διατηρούνται στον server ιδιωτικά δεδομένα των χρηστών. Τέτοιου είδους επιθέσεις αφορούν παραδοσιακές εφαρμογές που ακολουθούν την αρχιτεκτονική πελάτη-εξυπηρετητή. Για την πρόσβαση σε αυτές ο χρήστης συνήθως έχει έναν λογαριασμό και ένα συνθηματικό. Παράλληλα, η εφαρμογή συνήθως διατηρεί τα στοιχεία της πιστωτικής του κάρτας (ή κάποιου άλλου μέσου πληρωμής) για την πραγματοποίησης αγορών κτλ. Μία επιτυχημένη επίθεση στους εξυπηρετητές της προαναφερθείσας εφαρμογής λοιπόν, θα μπορούσε να αποδειχθεί καταστροφική για τους χρήστες. Αυτό δεν ισχύει για την αποκεντρωμένη εφαρμογή «Crypto Weather» που αναπτύχθηκε στα πλαίσια της διπλωματικής αυτής. Ο εξυπηρετητής δεν λαμβάνει ποτέ το ιδιωτικό κλειδί του λογαριασμού κάποιου χρήστη, ούτε εκτελεί κάποια συναλλαγή αντί γι' αυτόν. Όλες οι συναλλαγές στο blockchain στέλνονται από τον πελάτη απευθείας στο blockchain χωρίς

την ανάμειξη του εξυπηρετητή. Τέλος όταν απαιτείται ταυτοποίηση του χρήστη αυτός απλά υπογράφει κάποια δεδομένα με την χρήση της κρυπτογραφίας ελλειπτικών καμπυλών (δημοσίου κλειδιού) την οποία παρέχει το Ethereum blockchain. Η ασφάλεια δηλαδή της εφαρμογής εξαρτάται από την ασφάλεια του δικτύου του Ethereum, το οποίο μέχρι στιγμής θεωρείται απαραβίαστο [17].

Βλέπουμε λοιπόν, ότι η κυριότερη απαίτηση από τον εξυπηρετητή είναι η αποστολή των ιστοσελίδων (HTML, CSS, JavaScript) και μερικών επιπλέον δεδομένων (από την βάση δεδομένων) στους πελάτες (clients) έτσι ώστε αυτοί να μπορούν να αλληλοεπιδράσουν εύκολα με τα έξυπνα συμβόλαια, χωρίς να υπάρχουν ιδιαίτερες απαιτήσεις σε ασφάλεια και σε υπολογιστική δύναμη για τον εξυπηρετητή. Χρησιμοποιήθηκε λοιπόν ο Node.js server ο οποίος συμπεριλαμβάνεται στο Node-RED. Βλέπουμε λοιπόν πως με την επιλογή του Node-RED συνδυάζουμε σε μία πλατφόρμα και την προσομοίωση των IoT αισθητήρων καιρού αλλά και την φιλοξενία της ιστοσελίδας.

Πριν την παρουσίαση των ροών επεξηγείται ο τρόπος με τον οποίο γίνονται στις συναρτήσεις του Node-RED διαθέσιμα εξωτερικά πακέτα, δεδομένου ότι ο JavaScript κώδικας εκτελείται σε sandbox και η εντολή require() της JavaScript δεν μπορεί να χρησιμοποιηθεί μέσα σε κάποιον κόμβο. Στο directory .node-red, που περιέχει τις ροές του χρήστη, τοποθετούμε τα node modules τα οποία θα χρειαστούμε. Ύστερα στο αρχείο settings.js του ίδιου directory βρίσκουμε το πεδίο functionGlobalContext και προσθέτουμε μία εγγραφή για το κάθε πακέτο που χρειαζόμαστε. Η μορφή της εγγραφής αυτής είναι η εξής: x:require('y'), όπου το y είναι το όνομα του node module, το require('y') σημαίνει να γίνει η φόρτωση του πακέτου y και x είναι το όνομα του πακέτου αυτού στο Global Context του Node-RED. Έτσι, για παράδειγμα η εντολή ethereumjs:require('ethereumjs-util') φορτώνει στο Global Context του Node-RED το node module ethereumjs-util με το όνομα ethereumjs.

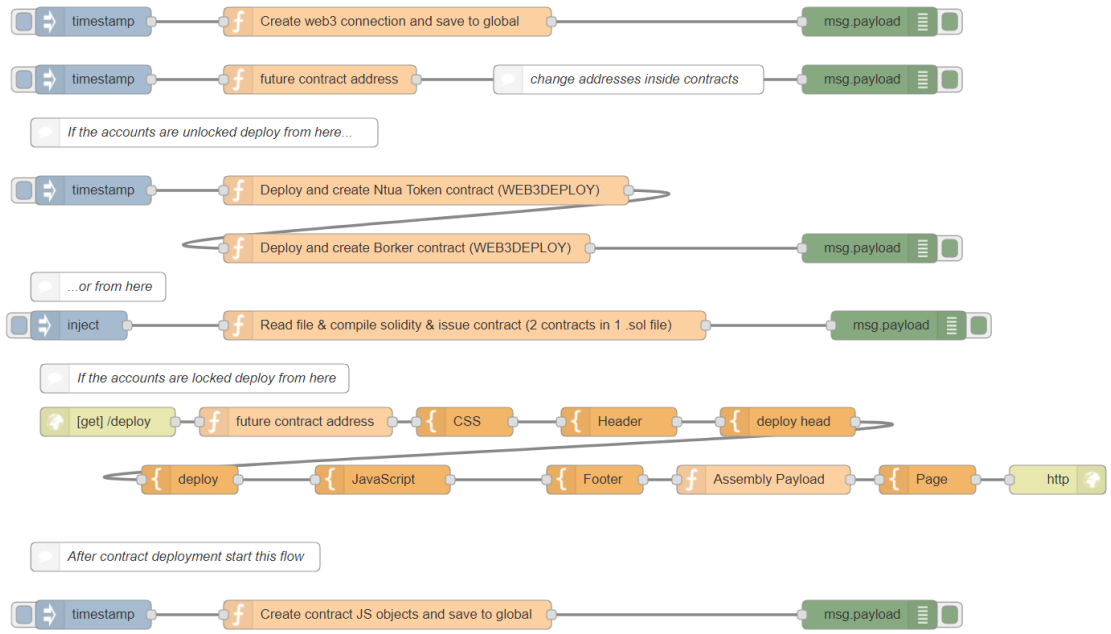
Για την λειτουργία της εφαρμογής προστέθηκαν οι εξής εντολές στο functionGlobalContext του αρχείου settings.js:

- `web3:require('web3')`
- `solc:require(solc)`
- `ethereumjs:require('ethereumjs-util')`
- `fs:require('fs')`

Στην ενότητα αυτή θα γίνει αναλυτική παρουσίαση των βασικότερων ροών του Node-RED που αφορούν το backend της εφαρμογής και θα αναφερθούν διάφορες παρατηρήσεις που αφορούν την ανάπτυξη αυτών.

5.4.1 Ροή Deploy contracts

Η ροή Deploy contracts δημιουργήθηκε κυρίως για να αυτοματοποιηθεί η διαδικασία του deployment των συμβολαίων, συμβάλλοντας έτσι στην γρηγορότερη ανάπτυξη του συστήματος. Επίσης υλοποιήθηκαν πολλοί τρόποι για το deployment των συμβολαίων για εκπαιδευτικούς λόγους, εξερεύνηση της νέας τεχνολογίας του Ethereum.



Εικόνα 5.4 Ροή Deploy contracts

Για την αλληλεπίδραση του εξυπηρετητή με το blockchain χρησιμοποιείται η βιβλιοθήκη web3.js. Μέσω αυτής της βιβλιοθήκες δημιουργείται μία HTTP σύνδεση με τον Ethereum κόμβο που τρέχει τοπικά και «ακούει» στην πόρτα 8545. Ο κόμβος μπορεί να είναι ενός τοπικού blockchain για testing (πχ. με το ganache-cli) ή κόμβος κάποιου test network ή κόμβος του κυρίως Ethereum δικτύου. Η σύνδεση αυτή γίνεται μία φορά και αποθηκεύεται στο global context του Node-RED και είναι διαθέσιμη σε όλους τους κόμβους, με τον τρόπο αυτό πετυχαίνουμε αύξηση της απόδοσης σε σχέση με το εάν δημιουργούσαμε μία νέα σύνδεση σε κάθε κόμβο που θα να χρησιμοποιούσε την web3.

Έχουμε έναν κόμβο ο οποίος υπολογίζει a priori τις διευθύνσεις που τα έξυπνα συμβόλαια θα έχουν όταν γίνουν deploy στο blockchain βάσει της Ethereum διεύθυνσης μας και των συναλλαγών που έχουμε ήδη στείλει στο blockchain (nonce). Η ροή αυτή είναι απαραίτητη, καθώς και το έξυπνο συμβόλαιο Broker γίνεται αναφορά στο NtuaToken, αλλά και το NtuaToken γίνεται αναφορά στο Broker. Έτσι βλέπουμε τις διευθύνσεις που αυτά θα πάρουν όταν γίνουν deploy στο blockchain και βάζουμε τις σωστές διευθύνσεις στον πηγαίο κώδικά τους, για την αναφορά σε ένα συμβόλαιο μέσα από το άλλο.

Αφού έχουν γίνει οι κατάλληλες αλλαγές στον πηγαίο κώδικα των συμβολαίων, έρχεται η ώρα του deployment τους. Έχουν δημιουργηθεί τρεις διαφορετικοί τρόποι για το deployment των συμβολαίων, ώστε να εξεταστούν πολλά διαφορετικά διαθέσιμα εργαλεία. Οι πρώτοι δύο τρόποι λειτουργούν μόνο εάν η διεύθυνση από την οποία θέλουμε να δημιουργήσουμε τα συμβόλαια είναι ξεκλείδωτη, ενώ ο τρίτος τρόπος λειτουργεί σε κάθε περίπτωση.

Ο πρώτος τρόπος υλοποιείται από τους κόμβους «Deploy and create Ntua Token contract (WEB3DEPLOY)» και «Deploy and create Broker contract (WEB3DEPLOY)». Τα συμβόλαια έχουν γραφτεί και μεταγλωττιστεί στο Remix και αντιγράφεται από εκεί το κομμάτι κώδικα που ονομάζεται «WEB3DEPLOY» και αφορά το deployment του μεταγλωττισμένου συμβολαίου με την χρησιμοποίηση της web3.

Ο κώδικας αυτός έχει την εξής μορφή:

```
var brokerContract = web3.eth.contract(ABI);
var broker = brokerContract.new({
        from: web3.eth.accounts[0],
        data: '0x606…029',
          gas: '4700000'
      }, function (e, contract){
      console.log(e, contract) ;
      if (typeof contract.address !== 'undefined') {
        console.log("Message for successful deployment of
contract.") ;
      }
})
```

Το ABI είναι η διεπαφή του συμβολαίου (Application Binary Interface), είναι ο τρόπος με τον οποίο μπορούμε να αλληλοεπιδράσουμε με το συμβόλαιο, να καλούμε συναρτήσεις του κτλ. [57].

Το πεδίο from καθορίζει την διεύθυνση που θα στείλει την εντολή δημιουργίας του συμβολαίου.

Το πεδίο data προκύπτει από την μεταγλώττιση του πηγαίου κώδικα.

Τέλος υπάρχει μία callback συνάρτηση που εκτελείται όταν τελειώσει η (ασύγχρονη) εκτέλεση της συναλλαγής στο blockchain.

Ο δεύτερος τρόπος υλοποιείται από τον κόμβο «Read file & compile solidity & issue contract (2 contracts in 1 .sol file)». Μόλις ενεργοποιηθεί ο κόμβος inject στέλνει στον προαναφερθέν κόμβο την θέση στο file system του αρχείου .sol, που περιέχει τον πηγαίο κώδικα και των δύο στο συμβολαίων. Έπειτα ο κόμβος «Read file & compile solidity & issue contract (2 contracts in 1 .sol file)» βρίσκει και διαβάζει το αρχείο με την βοήθεια του πακέτου fs (file system) και μεταγλωττίζει τον πηγαίο κώδικα με την βοήθεια του πακέτου solc, του compiler δηλαδή της Solidity. Έπειτα ακολουθείται η ίδια διαδικασία με αυτήν του πρώτου τρόπου. Ο τρόπος αυτός έχει τον μεγαλύτερο αυτοματισμό, απαιτεί τις ελάχιστες ενέργειες από την μεριά του χρήστη.

Ο τρίτος τρόπος υλοποιείται με την δημιουργία μίας ιστοσελίδας.

Στην οθόνη εμφανίζονται οι διευθύνσεις που θα πάρουν τα συμβόλαια. Ο χρήστης αρκεί να πατήσει το κουμπί «Deploy Ntua token Contract» και ύστερα το «Deploy Broker Contract», αυστηρά με αυτήν την σειρά, αλλιώς οι διευθύνσεις τους θα είναι αντίστροφες, για το deployment των συμβολαίων.

Υπάρχουν και άλλοι τρόποι να γίνουν deploy τα συμβόλαια. Απευθείας από το Remix. Από το wallet του Ethereum κτλ. Οι ροές στο Node-RED έγιναν για εκπαιδευτικούς σκοπούς.

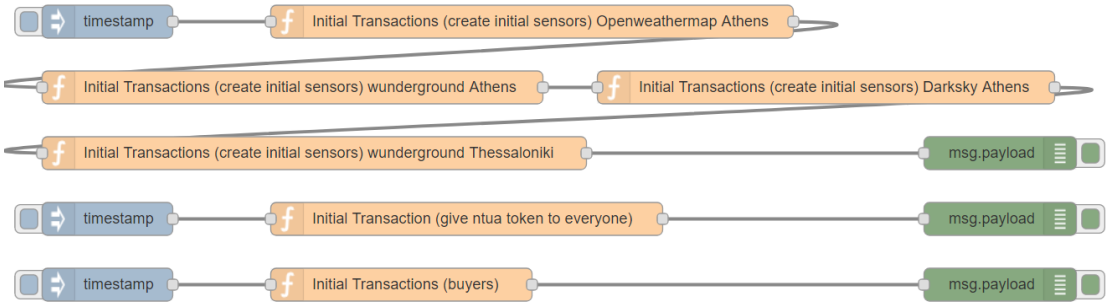
Αφού τα συμβόλαια γίνουν deploy με κάποιον από τους παραπάνω τρόπους θα πρέπει να δημιουργηθούν τα JS objects, ώστε να μπορούν οι υπόλοιποι κόμβοι να αλληλοεπιδρούν με αυτά. Αυτό γίνεται στον κόμβο «Create contract JS objects and save to global». Ενδεικτικά ο κώδικας έχει την εξής μορφή:

```
const NtuaTokenContract =
web3.eth.contract(NtuaTokenContractAbi).at(NtuaTokenContractAddress,
(err, ctr) => {}) ;
const BrokerContract =
web3.eth.contract(BrokerContractAbi).at(BrokerContractAddress, (err,
ctr) => {}) ;
```

Βλέπουμε ότι για την δημιουργία του JS αντικειμένου χρειάζονται το ABI του συμβολαίου (η διεπαφή) και η διεύθυνσή του (η τοποθεσία).

5.4.2 Ροές Initial transactions και All transactions

Η ροή Initial transactions δημιουργήθηκε για την επιτάχυνση της διαδικασίας ανάπτυξης, δοκιμής και ελέγχου του συστήματος καθώς και για την αποτελεσματικότερη παρουσίασή του. Αφορά κυρίως την λειτουργία της εφαρμογής με το βοηθητικό/ιδιωτικό blockchain που δημιουργούμε με το εργαλείο ganache-cli. Σκοπός της είναι η αποστολή στο blockchain πολλών αρχικών συναλλαγών, ώστε να προσομοιωθεί γρήγορα η περίοδος κανονικής λειτουργίας της εφαρμογής (εάν ή όταν γίνει εμπορική).



Εικόνα 5.5 Ροή Initial transactions

Οι κόμβοι «Initial Transactions (create initial sensors) …» δημιουργούν και καταχωρούν στο blockchain τους IoT αισθητήρες καιρού που προσομοιώνουμε με την βοήθεια των Open Weather Map, The Weather Underground και Dark Sky. Έτσι πολύ γρήγορα η εφαρμογή γεμίζει με 31 αισθητήρες.

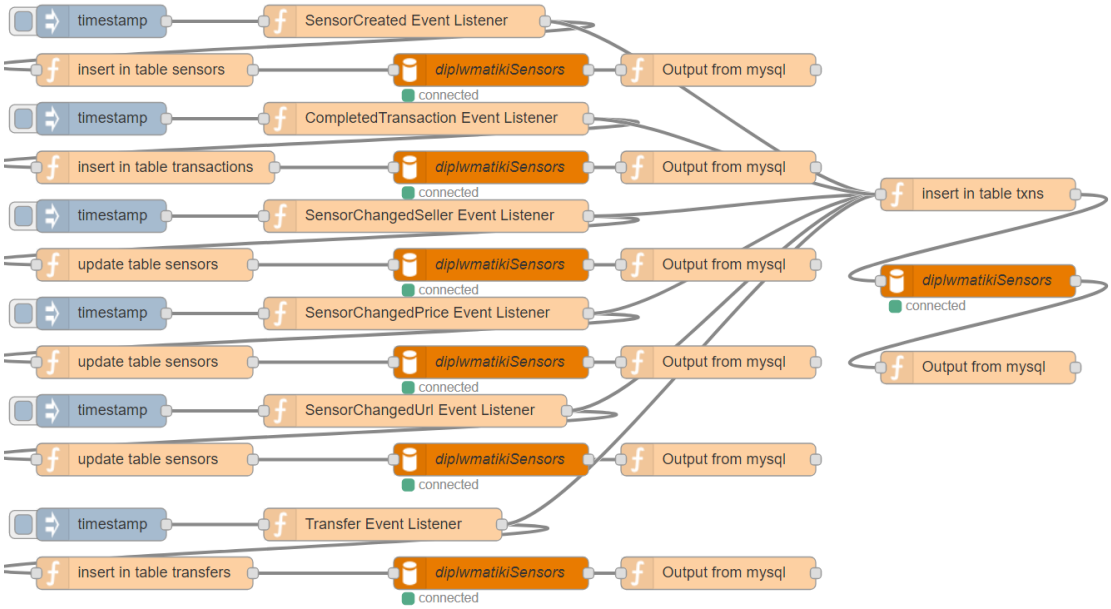
Ο κόμβος «Initial Transaction (give ntua token to everyone)» μοιράζει NTUA Tokens στους 10 λογαριασμούς του ιδιωτικού blockchain (που δημιουργούμε με το ganache-cli), έτσι ώστε να μπορέσουμε μετέπειτα αυτόματα να κάνουμε αγορές από όλες τις διευθύνσεις.

Τέλος, ο κόμβος «Initial Transactions (buyers)» καλεί την συνάρτηση «buyData» του συμβολαίου Broker και έτσι γίνονται πολλές αγορές δεδομένων αισθητήρων από διαφορετικές διευθύνσεις.

Η ροή «All transactions», αποτελείται από κόμβους οι οποίοι περιέχουν κώδικα με τον οποίο μπορούμε να καλέσουμε πολλές φορές όλες τις συναρτήσεις των έξυπνων συμβολαίων. Χρησιμοποιήθηκε κυρίως για λόγους ανάπτυξης. Επίσης αποτέλεσε ένα από τα εργαλεία για την εύρεση του gas usage κάθε συνάρτησης.

5.4.3 Ροή Events listeners

Η ροή Events listeners περιλαμβάνει τους κόμβους που διαβάζουν το blockchain και βρίσκουν τα events που αφορούν τα έξυπνα συμβόλαια της εφαρμογής μας και αποθηκεύουν την πληροφορία αυτή στη βάση δεδομένων, ώστε να είναι γρήγορα και εύκολα προσβάσιμη. Επίσης περιέχει κόμβους για την διαχείριση των πινάκων της βάσης.



Εικόνα 5.6 Ροή Events listeners A

Οι κόμβοι με το όνομα «…Event Listener» παρακολουθούν το blockchain και όπου βρουν το αντίστοιχο event στέλνουν το περιεχόμενό του στους επόμενους κόμβους, ώστε να αποθηκευτεί στην βάση δεδομένων.

Ο κόμβος «SensorCreated Event Listener» παρακολουθεί τα events SensorCreated του έξυπνου συμβολαίου Broker. Ενδεικτικά παρουσιάζεται ο κώδικας που χρησιμοποιείται:

```
BrokerContract.SensorCreated({}, { fromBlock: 0, toBlock: 'latest'
}, (err, res) => {
    if(!err) {
      node.send( {payload:res} ) ;
    } else {
      console.log("Error.") ;
    }
}) ;
```

Το BrokerContract είναι το JS αντικείμενο που έχει δημιουργηθεί από την ροή Deploy contracts και ακολουθεί το όνομα του event που θέλουμε να παρακολουθήσουμε. Στην συνέχεια ακολουθεί ένα φίλτρο, το οποίο ορίζει το διάστημα των μπλοκ στα οποία θα αναζητηθεί το συγκεκριμένο event. Το fromBlock θα πάρει την τιμή του μπλοκ στο οποίο υπάρχει η συναλλαγή με την οποία γίνεται deploy το συμβόλαιο Broker, αφού πριν από αυτό το μπλοκ το συμβόλαιο δεν υπήρχε, συνεπώς δεν υπάρχει

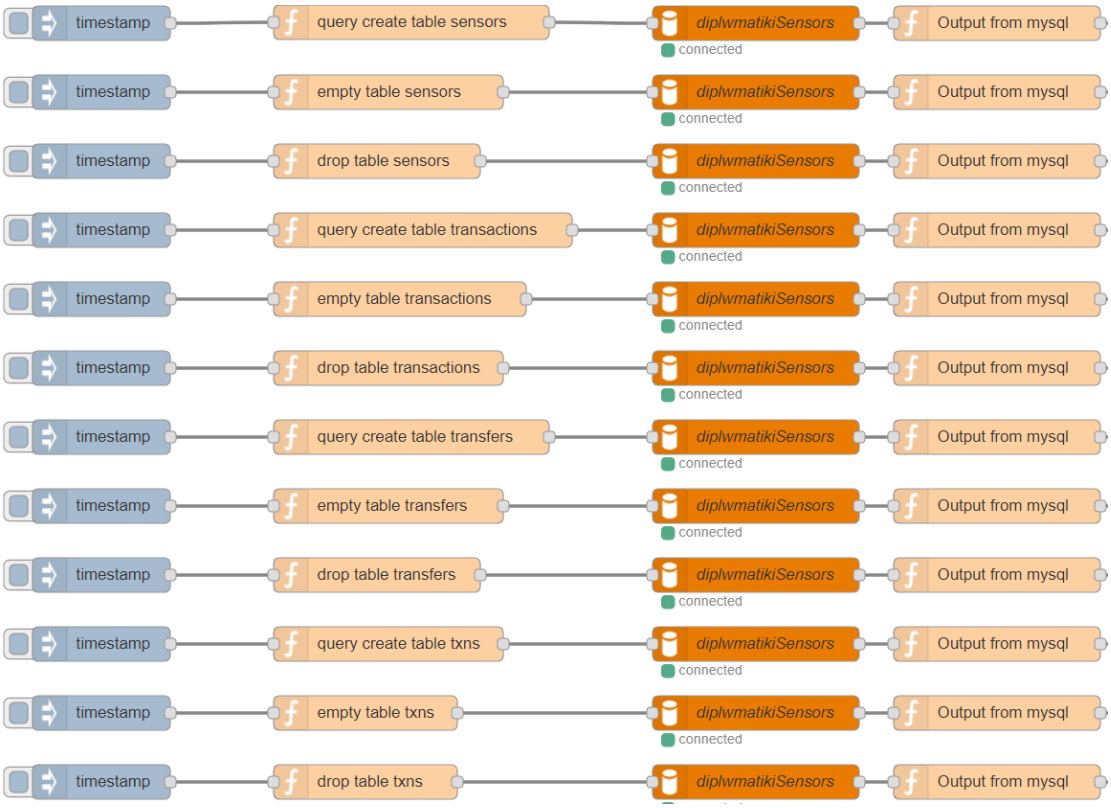
νόημα να εξεταστούν προηγούμενα μπλοκ. Το toBlock παίρνει την τιμή latest και αντιπροσωπεύει το τελευταίο μπλοκ του blockchain σε οποιαδήποτε στιγμή. Όταν βρεθεί ένα event SensorCreated καλείται η callback συνάρτηση και στέλνεται στον επόμενο κόμβο ένα μήνυμα με τις πληροφορίες που αφορούν την συγκεκριμένη εγγραφή του event που βρέθηκε.

Το μήνυμα με τις πληροφορίες που στέλνεται το λαμβάνει ο κόμβος «insert in table sensors» και αφού μετασχηματίσει τα δεδομένα, δημιουργεί ένα SQL query, ώστε να εισάγει μία νέα εγγραφή στον πίνακα sensors της βάσης δεδομένων για τον νέο αισθητήρα.

Ο κόμβος «CompletedTransaction Event Listener» παρακολουθεί τα events CompletedTransaction του έξυπνου συμβολαίου NtuaToken. Μόλις βρεθεί κάποιο event, προωθείται ένα μήνυμα στους επόμενους κόμβους και προστίθεται μία νέα εγγραφή στον πίνακα transactions της βάσης.

Ο κόμβος « SensorChangedSellerEvent Listener» παρακολουθεί τα events SensorChangedSeller του έξυπνου συμβολαίου Broker. Ο κόμβος « SensorChangedPrice Event Listener» παρακολουθεί τα events SensorChangedPrice του έξυπνου συμβολαίου Broker. Ο κόμβος « SensorChangedUrl Event Listener» παρακολουθεί τα events SensorChangedUrl του έξυπνου συμβολαίου Broker. Μόλις αυτοί βρουν αντίστοιχο event στέλνουν ένα μήνυμα στους επόμενους κόμβους, ώστε να ενημερωθεί το κατάλληλο πεδίο της κατάλληλης εγγραφής, προκειμένου τα στοιχεία των καταχωρημένων αισθητήρων να είναι πάντα ενημερωμένα.

Σε κάθε event, ενημερώνεται ένας πίνακας txns ο οποίος περιέχει όλες τις συναλλαγές που αφορούν τα έξυπνα συμβόλαια της εφαρμογής.

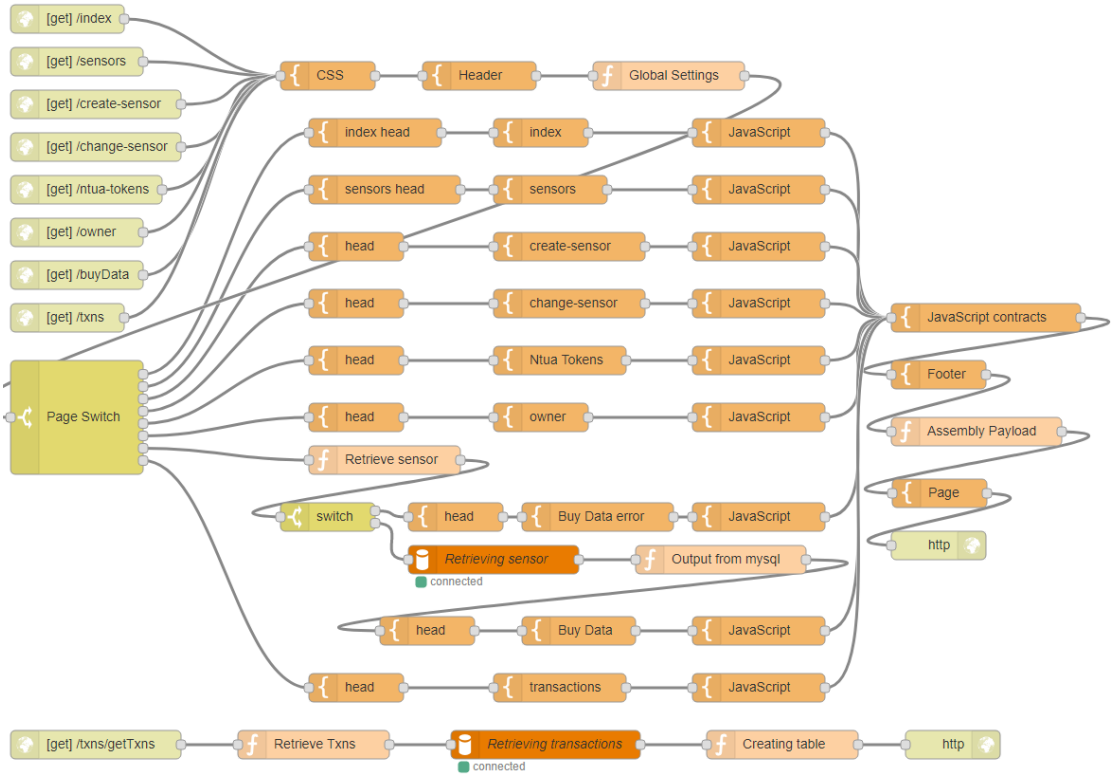


Εικόνα 5.7 Ροή Events listeners B

Αυτοί οι κόμβοι κατασκευάστηκαν για ευκολία κατά την ανάπτυξη του συστήματος. Δεν προσθέτουν κάποια επιπλέον λειτουργικότητα στην εφαρμογή σε συνθήκες ομαλής λειτουργίας. Χρησιμοποιήθηκαν κυρίως για στο στάδιο ανάπτυξης. Παραμένουν για εκπαιδευτικούς σκοπούς και επειδή θα μπορούσαν να φανούν χρήσιμοι στο ενδεχόμενο επίθεσης καταστροφής της της βάσης δεδομένων.

5.4.4 Ροή Website

Η ροή αυτή φιλοξενεί την συντριπτική πλειοψηφία των κόμβων που υλοποιούν τις λειτουργίες του website της εφαρμογής. Το website προσφέρει στους χρήστες ένα φιλικό περιβάλλον για την χρήση της εφαρμογής και την αλληλοεπίδραση με τα smart contracts στο Ethereum blockchain.



Εικόνα 5.8 Ροή Website

Οι κόμβοι «get /url» είναι HTTP end-points για την δημιουργία διαδικτυακών υπηρεσιών. Μόλις φτάνει ένα HTTP GET request στο συγκεκριμένο url, τότε ενεργοποιείται ο κόμβος αυτός. Στην συνέχεια κατασκευάζεται μία HTML σελίδα και αποστέλλεται σαν HTTP reply. Ο τρόπος με τον οποίο κατασκευάζεται η HTML ιστοσελίδα είναι ο εξής:

- Ένας «get /url» κόμβος λαμβάνει το αίτημα HTTP. Στέλνει μήνυμα στον επόμενο κόμβο.
- Ο επόμενος κόμβος είναι ο «CSS», οποίος περιέχει το CSS (Cascading Style Sheet) της ιστοσελίδας και το προσαρτά στο msg.payload.style. Τέλος να αναφερθεί ότι το CSS που χρησιμοποιήθηκε δημιουργήθηκε με το εργαλείο που προσφέρεται στην ιστοσελίδα https://pikock.github.io/bootstrap-magic/app/index.html#!/editor. Αυτό το εργαλείο χρησιμοποιείται για την γρήγορη και αποτελεσματική δημιουργία Bootstrap 4.0 θεμάτων.

- Στην συνέχεια ο κόμβος «Header» προσαρτά στο msg.payload.header το navigation bar της ιστοσελίδας. Τα δύο τελευταία στοιχεία είναι κοινά για όλες τις ιστοσελίδες.
- Ο κόμβος «Global Settings» που ακολουθεί αποθηκεύει τα msg.payload.style (CSS) και msg.payload.header (Header) στο Global context του Node-RED, ώστε αυτά να χρησιμοποιηθούν αργότερα.
- Έπειτα οδηγούμαστε στον κόμβο «Page Switch» ο οποίος αποτελεί έναν διακόπτη που κατευθύνει την ροή της εκτέλεσης στο κατάλληλο κλαδί, ώστε να φορτωθεί η κατάλληλη για το HTTP αίτημα ιστοσελίδα.
- Ο κόμβος «head» εμπεριέχει τον κώδικα για το <head> στοιχείο της HTML και τον προσαρτά στο msg.payload.head.
- Ακολουθεί το κυρίως περιεχόμενο της HTML ιστοσελίδας, το οποίο προσαρτάται στο msg.payload.section.
- Ο κόμβος «JavaScript» εμπεριέχει τον client-side JavaScript κώδικα, ο οποίος είναι διαφορετικός για κάθε ιστοσελίδα. Προσαρτάται στο msg.payload.script.
- Ο κόμβος «JavaScript contracts» περιέχει πληροφορίες για τα συμβόλαια, τις διευθύνσεις τους και τα ABI τους. Είναι κοινός για όλες τις ιστοσελίδες. Προσαρτάται στο msg.payload.script1.
- Ακολουθεί το υποσέλιδο, το οποίο προσαρτάται στο msg.payload.footer.
- Τέλος, στους κόμβους «Assembly Payload» και «Page» γίνεται η συναρμολόγηση της ιστοσελίδας (από τα αντικείμενα που υπάρχουν στο msg.payload) πριν αυτή σταλεί στον πελάτη.

Απόκλιση από το παραπάνω αποτελεί η διαδικασία δημιουργίας της ιστοσελίδας για /buyData. Λαμβάνεται ένα HTTP get request με δεδομένα id που είναι το id του αισθητήρα για τον οποίο πρέπει να αγοραστούν δεδομένα. Στην συνέχεια ανακτώνται οι πληροφορίες για τον συγκεκριμένο αισθητήρα, αν αυτός υπάρχει, και μετά κατασκευάζεται η σελίδα.

Τέλος παρατηρούμε μία υπό ροή η οποία ξεκινάει με τον κόμβο «/txns/getTxns». Αυτή η υπό ροή χρησιμοποιείται για την εξυπηρέτηση AJAX αιτημάτων, που αφορούν την αναζήτηση και εμφάνιση των συναλλαγών που έχουν πραγματοποιηθεί έως τώρα στην εφαρμογή.

Όπως έχει αναφερθεί η εφαρμογή είναι αποκεντρωμένη και σκοπός του website είναι να προσφέρει στους χρήστες ένα φιλικό περιβάλλον για την χρήση της εφαρμογής και την αλληλοεπίδραση με τα smart contracts στο Ethereum blockchain. Έτσι θα πρέπει κάπου στην ιστοσελίδα να δίνεται η δυνατότητα στο χρήστη να συνδεθεί με το blockchain ώστε να μπορεί να στείλει συναλλαγές. Αυτό επιτυγχάνεται με τον τρόπο που περιγράφεται στην συνέχεια. Ο χρήστης επισκέπτεται την ιστοσελίδα χρησιμοποιώντας τον browser Mist ή το plugin Metamask. Έτσι, αυτόματα γίνεται inject το web3 και η Web3.js βιβλιοθήκη στο JavaScript περιεχόμενο της ιστοσελίδας. Αυτό που πρέπει η ιστοσελίδα να κάνει είναι να πάρει αυτό το web3 και να ανοίξει μία σύνδεση με το blockchain (από τον browser του πελάτη). Το κομμάτι JavaScript κώδικα με το οποίο αυτό επιτυγχάνεται στην γενική περίπτωση είναι το παρακάτω [58]:

```
window.addEventListener('load', function() {
    //Checking if Web3 has been injected by the browser
(Mist/MetaMask)
    if (typeof web3 !== 'undefined') {
         window.web3 = new Web3(web3.currentProvider); //Use
Mist/MetaMask's provider
    } else {
    console.log('No web3? You should consider trying MetaMask!')
        // fallback - use your fallback strategy
    }
    startApp() ;
});
```

Η παραπάνω συνάρτηση, είναι καθοριστικής σημασίας για την λειτουργία της εφαρμογής, εκτελείται μόλις τελειώσει το φόρτωμα της ιστοσελίδας. Εάν χρησιμοποιείται το ο Mist browser ή το Metamask, τότε το αντικείμενο web3 είναι ορισμένο και έτσι μπορούμε να ανοίξουμε νέα σύνδεση με το blockchain. Στην συνέχεια καλείται η συνάρτηση startApp(), ώστε να ξεκινήσει το κυρίως μέρος της εφαρμογής. Χρησιμοποιώντας το window.web3 δίνουμε στον χρήστη την δυνατότητα να στέλνει συναλλαγές στο blockchain, πχ.:

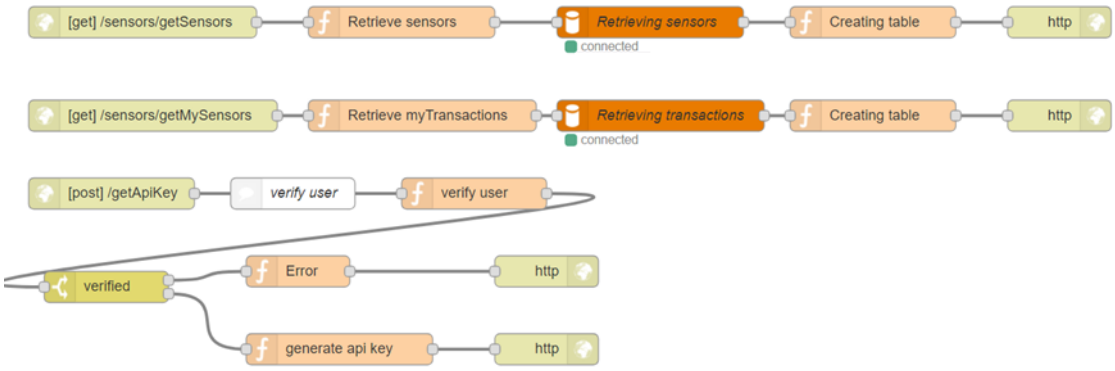
```
BrokerContract.buyData(sensorID, fromTime, toTime, {gas: 300000},
(err,res) => {}
```

Μπορούμε επίσης να χρησιμοποιήσουμε όλες τις συναρτήσεις του web3, όπως να δημιουργήσουμε αναφορά στα έξυπνα συμβόλαια:

```
const BrokerContract =
window.web3.eth.contract(BrokerContractAbi).at(BrokerContractAddress
, (err, ctr) => {})
```

5.4.5 Ροή Serve sensors

Η ροή αυτή περιλαμβάνει κόμβους που αντικείμενο έχουν την ανάκτηση των αισθητήρων από την βάση για την εξυπηρέτηση JavaScript αιτημάτων.



Εικόνα 5.9 Ροή Serve sensors

Η πρώτη υπό ροή γίνεται ενεργοποιείται όταν φτάσει ένα HTTP GET request στο url /sensors/getSensors. Στην συνέχει ο κόμβος «retrieve sensors» διαβάζει τα δε-

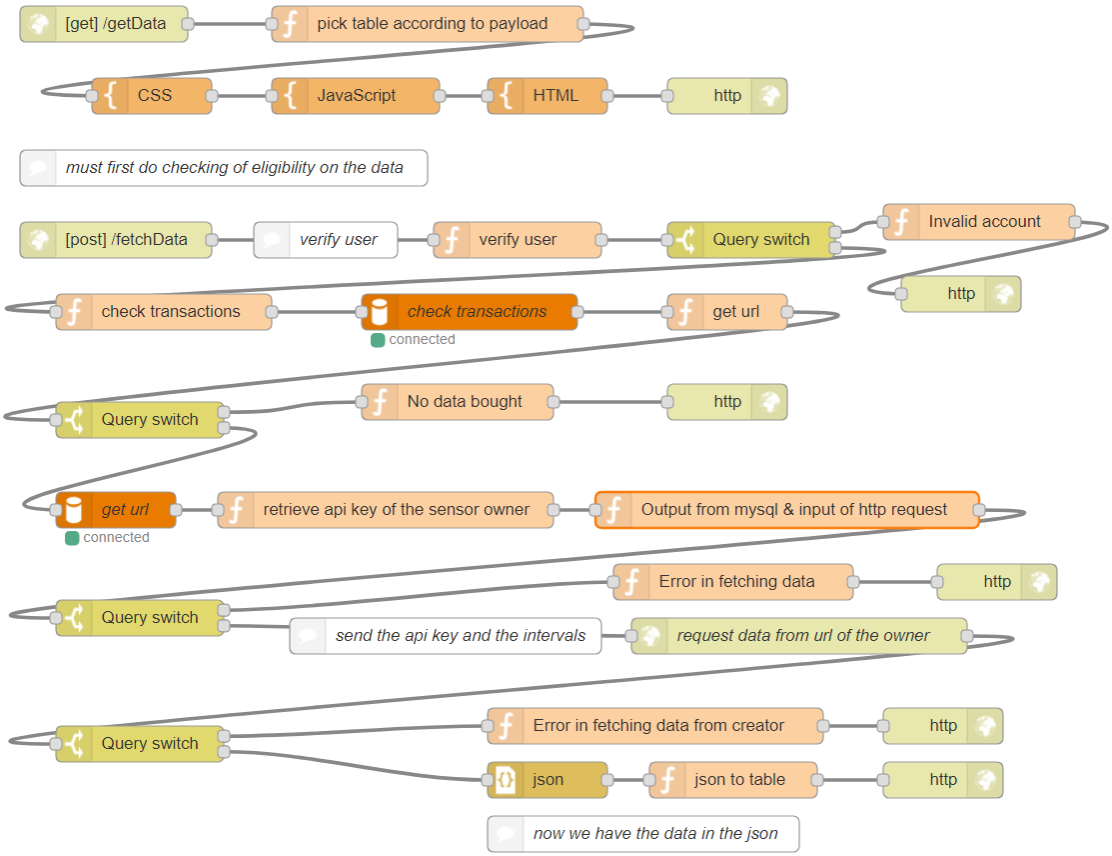
δομένα που συνοδεύουν το request. Αναλόγως τα δεδομένα που λαμβάνονται, ετοιμάζει ένα SQL query το οποίο εκτελείται από τον κόμβο «Retrieving sensors», επιλέγονται οι εγγραφές που ζητήθηκαν, από τον πίνακα sensors της βάσης και στην συνέχεια ο κόμβος «Creating table» ετοιμάζει τον HTML πίνακα και ο επόμενος κόμβος τον στέλνει σαν απάντηση (HTTP response) στο αρχικό HTTP request.

Η δεύτερη υπό ροή έχει παρόμοια λειτουργία με την πρώτη, μόνο που εδώ επιλέγονται αγορές δεδομένων αισθητήρων που έχουν γίνει από μία συγκεκριμένη Ethereum διεύθυνση από τον πίνακα transactions της βάσης.

Η τρίτη υπό ροή έχει ξεχωριστό χαρακτήρα και σκοπό έχει να δημιουργήσει ένα api key για κάθε ιδιοκτήτη κάποιου IoT αισθητήρα καιρού. Ενεργοποιείται από ένα HTTP POST request με δεδομένα address και signedData. Για την πρόσβαση στο περιεχόμενο απαιτείται ταυτοποίηση του χρήστη, έτσι αυτός υπογράφει με τον αλγόριθμο ελλειπτικών καμπυλών κάποια δεδομένα στον browser του και στην συνέχεια στέλνεται ένα request που συνοδεύεται από την διεύθυνσή του και τα υπογεγραμμένα δεδομένα. Ο εξυπηρετητής ακολουθεί την αντίθετη πορεία για την αποκρυπτογράφηση των δεδομένων που ελήφθησαν (ecrecover – elliptic curve recover) με την βοήθεια της βιβλιοθήκης ethereumjs-util. Εάν από την αποκρυπτογράφηση προκύψουν τα αρχικά δεδομένα τότε η ταυτοποίηση του χρήστη είναι επιτυχής και η εκτέλεση συνεχίζεται. Ακολουθεί ο κόμβος «generate api key» ο οποίος δημιουργεί το API key για την δοθείσα Ethereum διεύθυνση με ντετερμινιστικό τρόπο. Θέλουμε δηλαδή κάθε φορά που κάποιος Ethereum λογαριασμός ζητάει το API key που του αντιστοιχεί να λαμβάνει την ίδια απάντηση και δύο διαφορετικοί λογαριασμοί να μην έχουν το ίδιο API key. Για να το πετύχει αυτό χρησιμοποιεί και πάλι την κρυπτογραφία ελλειπτικών καμπυλών και την βιβλιοθήκη της JavaScript ethereumjs-util, αφού υπογράφει με το ιδιωτικό κλειδί (ενός βοηθητικού Ethereum λογαριασμού της εφαρμογής) την δοθείσα Ethereum διεύθυνση του χρήστη (ecsign – elliptic curve sign). Το αποτέλεσμα στέλνεται πίσω μέσω ενός HTTP response.

5.4.6 Η ροή Serve data

Η ροή αυτή περιλαμβάνει κόμβους που αντικείμενο έχουν την ανάκτηση των δεδομένων των IoT αισθητήρων καιρού από τους ιδιοκτήτες τους και την αποστολή τους στους χρήστες που τα έχουν αγοράσει.



Εικόνα 5.10 Ροή Serve data

Η πρώτη υπό ροή ενεργοποιείται όταν λαμβάνεται HTTP GET request στο url /getData. Αποστέλλεται με το HTTP response μία HTML ιστοσελίδα με το style το CSS και client-side JavaScript. Μόλις αυτή η σελίδα φορτωθεί ζητείται από τον χρήστη να υπογράψει με το ιδιωτικό του κλειδί, με τον αλγόριθμο κρυπτογραφίας ελλειπτικών καμπυλών, ορισμένα δεδομένα για να γίνει η ταυτοποίησή του. Μόλις αυτά υπογραφούν, στέλνεται ένα AJAX post request στο url /fetchData που συνοδεύεται από το sensorID του αισθητήρα του οποίου τα δεδομένα ζητάμε (sensor), την διεύθυνσή του χρήστη (address) και τα υπογεγραμμένα δεδομένα (signedData).

Να σημειωθεί ότι χρησιμοποιείται η συνάρτηση web3.eth.sign για την οποία, κατά την στιγμή της ανάπτυξης της εφαρμογής, υπήρχε η παρατήρηση ότι θα αντικατασταθεί από κάποια άλλη συνάρτηση. Το πρόβλημα είναι πως αυτή η μέθοδος είναι ευάλωτη σε κρυπτογραφικές επιθέσεις γνωστού κρυπτοκειμένου. Η μέθοδος χρησιμοποιήθηκε επειδή δεν υπήρχε άλλη εναλλακτική και εύκολα θα αντικατασταθεί όταν προταθεί η νέα λύση.

Η δεύτερη υπό ροή ενεργοποιείται όταν λαμβάνεται HTTP post request που περιέχει τα δεδομένα που προαναφέρθηκαν. Πρώτο βήμα είναι να γίνει στο κόμβο «verify user» ταυτοποίηση του χρήστη μέσω της αποκρυπτογράφησης των signedData με την μέθοδο ecrecover χρησιμοποιώντας το δημόσιο κλειδί του χρήστη. Εάν το αποκρυπτογραφημένο κείμενο ταυτίζεται με το δημόσιο κλειδί του χρήστη (την Ethereum διεύθυνσή του δηλαδή) η ταυτοποίηση επιτυγχάνει και η εκτέλεση της ροής προχωράει παρακάτω, αλλιώς η ροή οδηγείται στον κόμβο «Invalid account» που ενημερώνει για το σφάλμα ή την προσπάθεια υποκλοπής δεδομένων. Έπειτα ο κόμβος «check transactions» ετοιμάζει ένα SQL query με το οποίο ανακτώνται όλες οι αγορές του χρήστη που αφορούν τον ζητούμενο αισθητήρα. Υπενθυμίζεται εδώ ότι ο πίνακας transactions

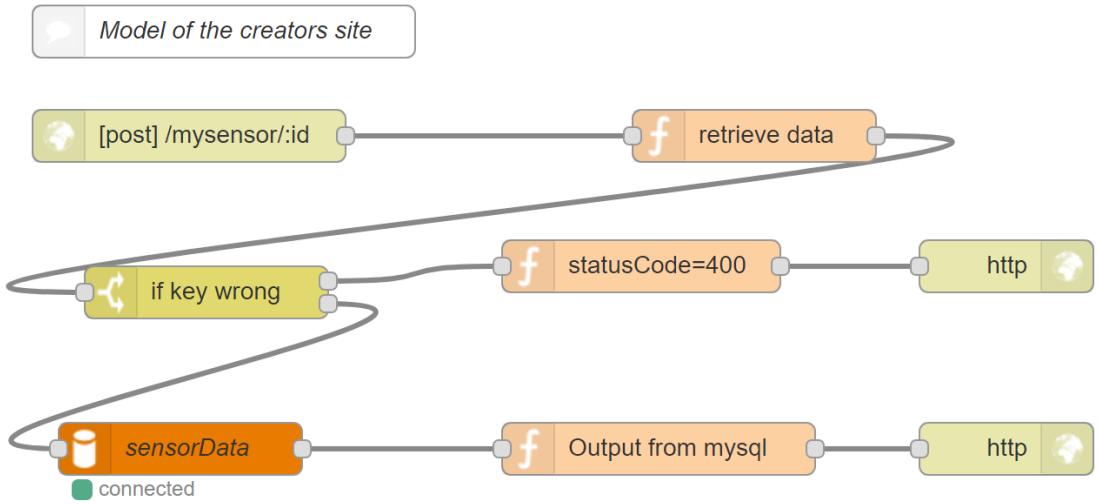
της βάσης δεδομένων διατηρεί ένα πιστό αντίγραφο των event transactions του συμβολαίου NtuaToken στο blockchain, τίποτα περισσότερο, τίποτα λιγότερο. Στην συνέχεια εάν έχει πραγματοποιηθεί κάποια αγορά η ροή προχωράει, αλλιώς η ροή οδηγείται στον κόμβο «No data bought», ο οποίος ενημερώνει τον χρήστη ότι δεν έχει πρόσβαση στα δεδομένα του αισθητήρα που ζητάει. Σε περίπτωση ύπαρξης αγοράς, ανακτάται από τον πίνακα sensors της βάσης το url που ο ιδιοκτήτης του αισθητήρα έχει δηλώσει ότι θα περιέχει τα δεδομένα. Ύστερα, στον κόμβο «retrieve api key of the sensor owner» ανακτάται το API key που έχει δοθεί στον ιδιοκτήτη του αισθητήρα (η διαδικασία είναι η ίδια με αυτήν που παρουσιάζεται για τον κόμβο «generate api key» της ροής «Serve sensors» στην προηγούμενη ενότητα). Στην συνέχεια κατασκευάζεται και στέλνεται ένα HTTP post request στο προ ανακτηθέν url, με δεδομένα το API key του ιδιοκτήτη (key), έναν πίνακα intervals ο οποίος περιέχει τα διαστήματα για τα οποία υπάρχουν αγορές δεδομένων. Εάν το σταλθέν HTTP request απαντηθεί με επιτυχία (κωδικός status Code=200) και λάβουμε τα ζητούμενα δεδομένα, αυτά παρουσιάζονται στον τελικό χρήστη, ενώ εάν υπάρξει κάποιο σφάλμα ο πελάτης ενημερώνεται με σχετικό μήνυμα.

Βλέπουμε λοιπόν πως ο εξυπηρετητής μας δεν έχει κάπου αποθηκευμένα τα δεδομένα των IoT αισθητήρων καιρού, αλλά αυτά ζητούνται κάθε φορά από τους ιδιοκτήτες τους, οι οποίοι είναι υποχρεωμένοι να τα παρέχουν στο url που έχουν δηλώσει.

5.4.7 Η ροή Sensor owner

Η ροή sensor owner δεν αποτελεί κανονικό μέρος της εφαρμογής. Έχει δημιουργηθεί για να προσομοιώσει την υπηρεσία που προσφέρουν οι ιδιοκτήτες/πωλητές των δεδομένων των αισθητήρων. Όπως έχει ήδη αναφερθεί η εφαρμογή είναι αποκεντρωμένη και σαν τέτοια δεν διατηρεί τα δεδομένα των αισθητήρων, επομένως οι ιδιοκτήτες τους πρέπει να προσφέρουν στους χρήστες έναν τρόπο πρόσβασης στα δεδομένα.

Ο ιδιοκτήτης του αισθητήρα όταν τον δημιουργεί καταχωρεί και ένα url. Τα δεδομένα θα ζητούνται από την εφαρμογή στο url αυτό με ένα HTTP POST request με τα εξής δεδομένα {key:apiKey, [fromTime:fromThisTimestamp, toTime:toThisTimestamp]}. Όταν το request αυτό φτάσει στον εξυπηρετητή του ιδιοκτήτη αυτός θα πρέπει, εάν το apiKey είναι σωστό να στείλει τα δεδομένα του αισθητήρα καιρού για τα χρονικά διαστήματα που ζητούνται.

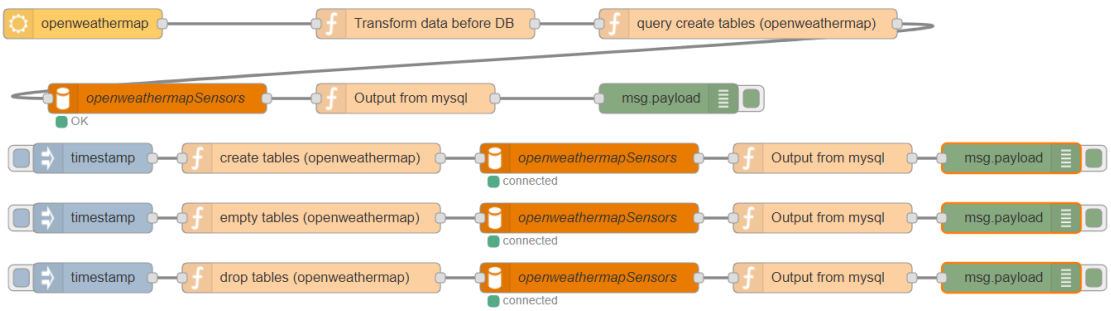


Εικόνα 5.11 Ροή Sensor owner

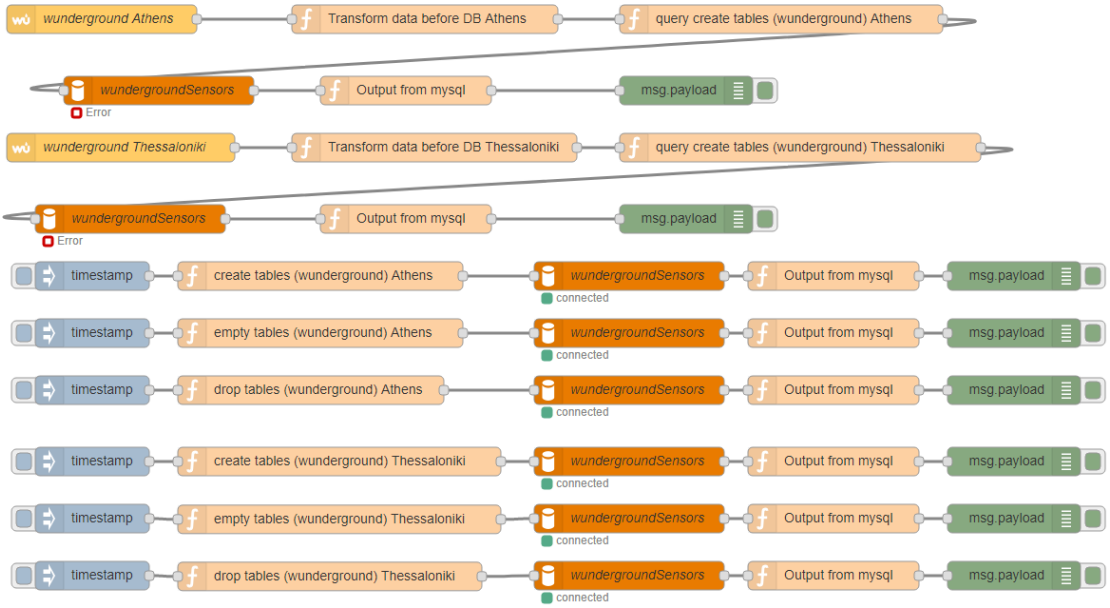
Η παραπάνω ροή υλοποιεί ακριβώς αυτήν την λειτουργικότητα. Οι IoT που έχουμε προσομοιώσει καταχωρούνται στο blockchain και σαν url δίνεται το /mysensor/Αριθμός. Όταν φτάνει το HTTP POST request σε αυτό το url, η ροή ενεργοποιείται. Στον κόμβο «retrieve data» γίνεται έλεγχος του apiKey που δόθηκε, ώστε να μην σταλούν τα δεδομένα σε κάποιον που δεν έχει πρόσβαση σε αυτά, και εάν αυτό είναι σωστό προετοιμάζεται το SQL query με το οποίο θα ανακτηθούν τα ζητούμενα δεδομένα από την βάση. Σε περίπτωση που το apiKey δεν ταιριάζει, αποστέλλεται μήνυμα σφάλματος. Στην συνέχεια δημιουργείται ένα JSON array της μορφής [{time:Timestamp, measurementType:value}], δηλαδή ζευγάρια χρόνου μέτρησης (time) και τιμής μέτρησης (measurementType = τύπος του αισθητήρα πχ. temperature). Έτσι η εφαρμογή μας λαμβάνει τις μετρήσεις και τις παρουσιάζει στον τελικό χρήστη.

5.4.8 Οι ροές Openweathermap, wunderground, darksky

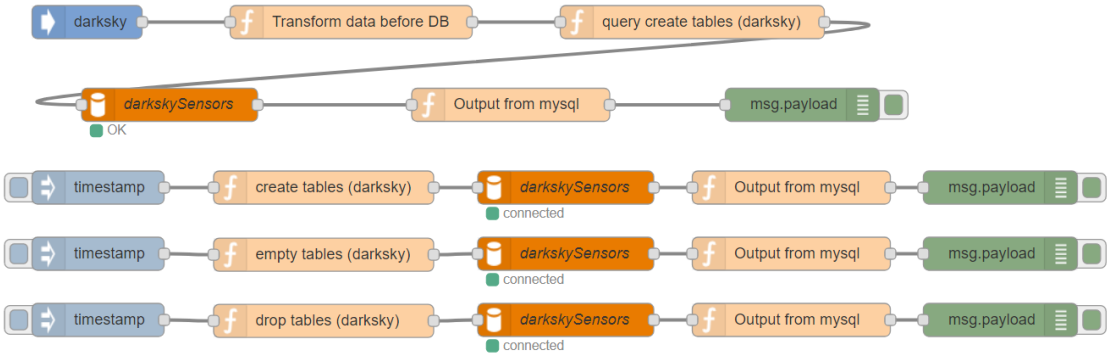
Οι ροές αυτές δημιουργήθηκαν για την προσομοίωση των IoT αισθητήρων καιρού. Χρησιμοποιούν τους έτοιμους κόμβους που προσφέρουν τα πακέτα node-red-node-openweathermap, node-red-node-weather-underground και node-red-node-darksky και οι οποίοι στέλνουν περιοδικά ερωτήματα στις βάσεις δεδομένων των αντίστοιχων οργανισμών και όταν λαμβάνουμε απάντηση την προωθούν στους επόμενους κόμβους.



Εικόνα 5.12 Ροή Openweathermap



Εικόνα 5.13 Ροή wunderground



Εικόνα 5.14 Ροή darksky

Στην συνέχεια θα παρουσιαστεί μόνον η ροή wunderground, αφού και και οι τρεις είναι παρόμοιες. Θα μιλήσουμε μόνο για τον κόμβο της μίας υπό ροής, αυτής που αφορά την Αθήνα. Ο κόμβος με το όνομα «wunderground Athens» περιοδικά στέλνει αιτήματα στην προγραμματιστική διεπαφή του The Weather Underground για τα τρέχοντα δεδομένα του καιρού και μόλις εντοπιστούν νέα η ροή προχωράει στον επόμενο κόμβο με όνομα «Transform data before DB Athens» ο οποίος φέρνει τα δεδομένα σε κατάλληλη μορφή (μετατροπή μονάδων μέτρησης, στρογγυλοποίηση κτλ.) για την αποθήκευσή τους στην βάση δεδομένων. Η ροή εκτέλεσης στην συνέχεια φτάνει στον κόμβο «query create tables (wunderground) Athens» ο οποίος φτιάχνει το SQL query για την εισαγωγή των δεδομένων στην βάση. Ο κόμβος «wundergroundSensors» στέλνει στην βάση το query και η εκτέλεση συνεχίζεται. Για την υπό ροή που αφορά την Θεσσαλονίκη ισχύουν τα ίδια, με την εξαίρεση ότι οι μετρήσεις αφορούν την Θεσσαλονίκη.

Τέλος δημιουργήθηκαν κάποιες υπό ροές για την εύκολη και γρήγορη διαχείριση των πινάκων της βάσης δεδομένων. Χρησιμοποιήθηκαν κυρίως στο στάδιο της ανάπτυξης της εφαρμογής.

Εάν είχαμε κάποιον πραγματικό IoT αισθητήρα τότε θα υπήρχε στην θέση του κόμβου «wunderground Athens» ένας άλλος κόμβος ο οποίος θα λάμβανε τα δεδομένα του αισθητήρα και με αντίστοιχο τρόπο θα τα αποθήκευε σε κάποια βάση δεδομένων.

Η παραπάνω ροή μας δίνει μία (απλοποιημένη) εικόνα για την ενδεχόμενη μορφή της Node-RED ροής που χρησιμοποιείται στην πραγματικότητα για την διασύνδεση IoT αισθητήρων και βάσεων δεδομένων με την χρήση του εργαλείου Node-RED. Με τις ροές «Openweathermap», «wunderground», «darksky» στο Node-RED, προσομοιώνονται οι IoT αισθητήρες καιρού.

Ακολουθεί αναλυτική περιγραφή του τύπου και του τρόπου επεξεργασίας των δεδομένων που λαμβάνονται από την κάθε διαφορετική πηγή.

Το OpenWeatherMap παρέχει τα εξής δεδομένα πραγματικού χρόνου:

- Θερμοκρασία (temperature) σε βαθμούς κελσίου (°C), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Σχετική υγρασία (relative humidity) σε ποσοστό επί τις 100 (%), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Ατμοσφαιρική πίεση (pressure) σε hecto Pascal (hPa), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Ορατότητα (visibility) σε μέτρα, η οποία πρώτα μετατρέπεται σε χιλιόμετρα και ύστερα αποθηκεύεται στην βάση δεδομένων.
- Ταχύτητα (speed) και διεύθυνση (direction) ανέμου σε μέτρα ανά δευτερόλεπτο (m/s) και μετεωρολογικές μοίρες (°) αντίστοιχα, οι οποίες αποθηκεύονται απευθείας στην βάση δεδομένων.
- Κάλυψη του ουρανού από σύννεφα (sky cloud coverage) σε ποσοστό επί τις εκατό (%), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.

Το Weather Underground παρέχει τα εξής δεδομένα πραγματικού χρόνου:

- Θερμοκρασία (temperature) σε βαθμούς κελσίου (°C), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Σχετική υγρασία (relative humidity) σε ποσοστό επί τις 100 (%), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Ατμοσφαιρική πίεση (pressure) σε hecto Pascal (hPa) , η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Ορατότητα (visibility) σε χιλιόμετρα, η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Ταχύτητα (speed) ανέμου σε χιλιόμετρα ανά ώρα (km/h), η οποία πρώτα μετατρέπεται σε m/s και ύστερα αποθηκεύεται στην βάση δεδομένων και διεύθυνση (direction) ανέμου σε μετεωρολογικές μοίρες (°), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Σημείο δρόσου ή σημείο υγροποίησης ή σημείο κόρου ατμόσφαιρας (dew point) σε βαθμούς κελσίου (°C) , το οποίο αποθηκεύεται απευθείας στην βάση δεδομένων.
- Ηλιακή ακτινοβολία σε watt/m², η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Δείκτης υπέρυθρης (UV) ακτινοβολίας σε κλίμακα 0 έως 11, η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.

Το Dark Sky παρέχει τα εξής δεδομένα πραγματικού χρόνου:

- Θερμοκρασία (temperature) σε βαθμούς κελσίου (°C), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.

- Σχετική υγρασία (relative humidity) σε ποσοστό επί τις 100 (%) διά εκατό, η οποία αποθηκεύεται στην βάση δεδομένων, αφού μετατραπεί σε επί τις εκατό (%).
- Ατμοσφαιρική πίεση (pressure) σε hecto Pascal (hPa), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Ορατότητα (visibility) σε χιλιόμετρα, η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Ταχύτητα (speed) ανέμου σε χιλιόμετρα ανά ώρα (km/h), η οποία πρώτα μετατρέπεται σε m/s και ύστερα αποθηκεύεται στην βάση δεδομένων και διεύθυνση (direction) ανέμου σε μετεωρολογικές μοίρες (°), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Κάλυψη του ουρανού από σύννεφα (sky cloud coverage) σε ποσοστό επί τις εκατό (%), η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Σημείο δρόσου ή σημείο υγροποίησης ή σημείο κόρου ατμόσφαιρας (dew point) σε βαθμούς κελσίου (°C), το οποίο αποθηκεύεται απευθείας στην βάση δεδομένων.
- Δείκτης υπέρυθρης (UV) ακτινοβολίας σε κλίμακα 0 έως 11, η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.
- Πυκνότητα στρώματος όζοντος της ατμόσφαιρας (columnar density of total atmospheric ozone layer) σε μονάδες Dobson, η οποία αποθηκεύεται απευθείας στην βάση δεδομένων.

5.5 Δημιουργία συμβολαίων στο Ropsten Test Net

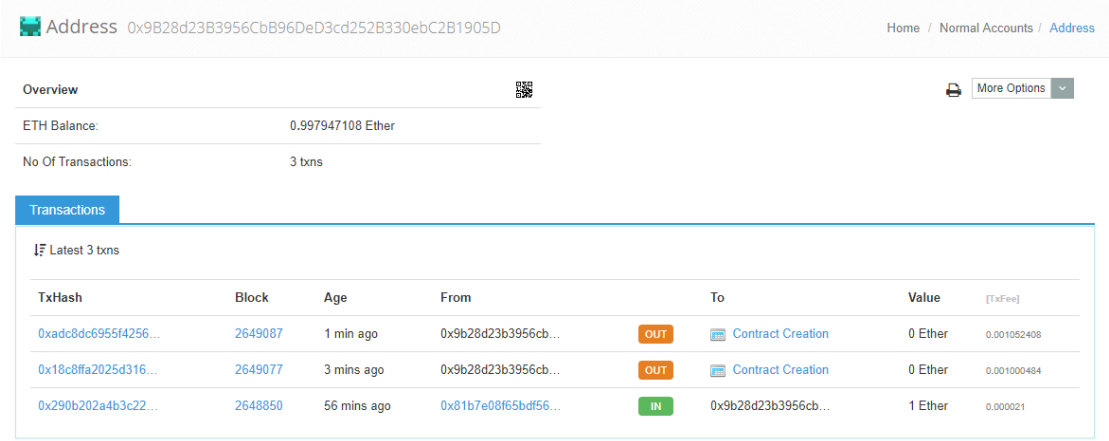
Σε αυτήν την ενότητα παρουσιάζονται αναλυτικά τα βήματα που πραγματοποιήθηκαν, ώστε να γίνουν deploy στο Ropsten Test Net τα δύο έξυπνα συμβόλαια NtuaToken και Broker.

Τα δύο συμβόλαια έγιναν deploy στο Ropsten Test Net. Για να γίνει αυτό ακολουθήθηκε η παρακάτω διαδικασία.

Αρχικά έγινε σύνδεση μέσω το Metamask στο Ropsten Test network. Στην συνέχεια έγινε δημιουργία ενός Ethereum λογαριασμού στο Ropsten Test network. Ο λογαριασμός αυτός έχει την διεύθυνση (δημόσιο κλειδί) «0x9B28d23B3956CbB96DeD3cd252B330ebC2B1905D». Έπειτα και πάλι μέσω του Metamask ζήτησα να μεταφερθούν ether στον λογαριασμό μου. Επειδή το Ropsten είναι test network τα ether δεν έχουν κάποια αξία, όπως έχουν τα ether του κυρίως δικτύου, δεν χρειάστηκε κάποια επιπλέον διαδικασία (πληρωμή ή mining), ιστοσελίδα: https://faucet.metamask.io/. Υπολογισμός των διευθύνσεων τις οποίες θα λάβουν τα έξυπνα συμβόλαια στο δίκτυο Ropsten (υπενθυμίζεται ότι αυτές εξαρτώνται αποκλειστικά από την διεύθυνση του δημιουργού και τον αριθμός των συναλλαγών τις οποίες έχει στείλει στο blockchain). Για το βήμα αυτό χρησιμοποιήθηκε ο κόμβος «future contract address». Κατά την στιγμή του υπολογισμού, δεν είχε σταλεί καμία συναλλαγή από τον λογαριασμό «0x9B28d23B3956CbB96DeD3cd252B330ebC2B1905D». Επομένως έγινε ο εξής υπολογισμός:
futureAddressNtuaRopsten = ethJsUtil.bufferToHex(ethJsUtil.generateAddress('0x9B28d23B3956CbB96DeD3cd252B330ebC2B1905D', 0)) = 0xd50972d2b1747e6277134d29db36e88dacf12844

futureAddressBrokerRopsten = ethJsUtil.bufferToHex(ethJsUtil.generate-Address('0x28ba479f162907e4915e33789dcb32a641ffe370', 1)) = 0xb1719bf761175017eeb6bb72423ece49914fef9b

Δηλαδή η πρώτη συναλλαγή του λογαριασμού «0x9B28d23B3956CbB96DeD3cd252B330ebC2B1905D» είναι η δημιουργία του NtuaToken συμβολαίου και η δεύτερη είναι η δημιουργία του Broker συμβολαίου. Ακολούθησε η αλλαγή των διευθύνσεων μέσα στον πηγαίο κώδικα των συμβολαίων. Τέλος έγινε το deployment των συμβολαίων στο Ropsten Test Net και πράγματι, τα συμβόλαια έλαβαν τις παραπάνω υπολογισμένες διευθύνσεις. Ακολουθεί screenshot από το etherscan.io που δείχνει την συναλλαγή με την οποία λαμβάνονται τα ether και τις συναλλαγές με τις οποίες δημιουργούνται τα δύο έξυπνα συμβόλαια.



Εικόνα 5.15 Συναλλαγές για την δημιουργία των συμβολαίων

Ακολουθούν ορισμένα στοιχεία των δύο συμβολαίων που αφορούν το Ropsten test network.

Όνομα συμβολαίου	NtuaToken
Διεύθυνση	0xd50972d2b1747e6277134d29db36e88dacf12844
Αριθμός μπλοκ δημιουργίας	2649077
Περισσότερες λεπτομέρειες	https://ropsten.etherscan.io/tx/0x18c8ffa2025d316b2e98f2e765913b3819acff6f903e6c814f1b43f875019f1a

Πίνακας 5.6 Λεπτομέρειες του συμβολαίου NtuaToken στο Ropsten Test Net

Όνομα συμβολαίου	Broker
Διεύθυνση	0xb1719bf761175017eeb6bb72423ece49914fef9b
Αριθμός μπλοκ δημιουργίας	2649087
Περισσότερες λεπτομέρειες	https://ropsten.etherscan.io/tx/0xadc8dc6955f42563214e640f7e04229d05f3ea3d72a439ab23c09cd7d4d5c077

Πίνακας 5.7 Λεπτομέρειες του συμβολαίου Broker στο Ropsten Test Net

5.6 Μετρικές απόδοσης

Σε αυτήν την ενότητα παρουσιάζονται ορισμένα ποσοτικά μεγέθη που αφορούν την απόδοση των έξυπνων συμβολαίων. Τα μεγέθη αυτά αφορούν την απαίτηση σε gas κάθε συνάρτησης, καθώς και την καθυστέρηση επικύρωσης της συναλλαγής.

Ακολουθούν οι απαιτήσεις της κάθε συνάρτησης σε gas για το συμβόλαιο NtuaToken. Οι αναγνώσεις από το blockchain είναι δωρεάν, ενώ οι εγγραφές και οι υπολογισμοί κοστίζουν. Ο αναλυτικός τρόπος υπολογισμού του κόστους σε gas μπορεί να βρεθεί στο Appendix H του Yellow Paper του Ethereum [17].

Ακολουθούν οι απαιτήσεις κάθε συνάρτησης σε gas του συμβολαίου NtuaToken.

Συνάρτηση	Gas usage
Δημιουργία συμβολαίου (constructor NtuaToken)	1000484
Fallback function (payable)	21320 όταν καλείται από τον ιδιοκτήτη του συμβολαίου 50294 όταν καλείται για πρώτη φορά από κάποιον μη ιδιοκτήτη
Καλείται όταν γίνεται μεταφορά ether στην διεύθυνση του συμβολαίου	35294 κάθε επόμενη φορά
changePrice	Από 27114 έως 27562 gas αναλόγως τις παραμέτρους. 27114 gas αντιστοιχεί σε αλλαγή της τιμής σε 1 wei (=10^{-18} ether) 27434 gas αντιστοιχεί σε αλλαγή της τιμής σε ether τάξης μεγέθους περίπου 1 (π.χ. 0.1 ether, 1 ether, 8 ether). 27562 gas αντιστοιχεί σε αλλαγή της τιμής σε 10^{9} ether
retrieveEthereum	Από 30184 έως 30568 gas (και λίγο περισσότερο), αναλόγως τις παραμέτρους. 30184 gas αντιστοιχεί σε ανάληψη 1 wei (=10^{-18} ether) 30568 gas αντιστοιχεί σε ανάληψη 100 ether
transfer	~ 52000 gas για μεταφορές από έναν λογαριασμό σε άλλον, ο οποίος ποτέ στο παρελθόν δεν κατείχε NTUATok. (πχ. 52098 για 1 NTUATok) ~37000 gas για μεταφορές από έναν λογαριασμό σε άλλον (πχ. 36778 για μεταφορά 10^{-18} NTUATok, 37098 για μεταφορά 1 NTUATok.
brokerTransferFrom	Δεν μπορούμε να απομονώσουμε το gas usage, διότι καλείται μόνο από το συμβόλαιο Broker.

Πίνακας 5.8 Απαιτήσεις σε gas των συναρτήσεων του συμβολαίου NtuaToken

Ακολουθούν οι απαιτήσεις της κάθε συνάρτησης σε gas για το συμβόλαιο Broker.

Συνάρτηση	Gas usage
Δημιουργία συμβολαίου (constructor Broker)	1052408
createSensor	142600-142800 όμως μπορεί να υπάρχει μεγάλη απόκλιση σε ορισμένες κλίσεις. Αυτό συμβαίνει, διότι οι παράμετροι είναι πολλές και υπάρχουν πολλοί δυνατοί διαφορετικοί συνδυασμοί
changeSensorSeller	Από 30106 έως 30166 ή και λίγο περισσότερο ή λίγο λιγότερο, αναλόγως το sensorID και την διεύθυνση στις παραμέτρους, ανεξάρτητα από τον αριθμό του μπλοκ.
changeSensorPrice	Από 28723 έως 30000 αναλόγως τις παραμέτρους, ανεξάρτητα από τον αριθμό του μπλοκ.
changeSensorUrl	Εξαρτάται αποκλειστικά από το μέγεθος του url. Όσο μεγαλύτερο είναι το url τόσο περισσότερο gas χρειάζεται.
buyData	~120000 gas

Πίνακας 5.9 Απαιτήσεις σε gas των συναρτήσεων του συμβολαίου Broker

Ο χρόνος αναμονής κάθε συναλλαγής δεν εξαρτάται μόνο από τον κώδικα που πρέπει να εκτελεστεί, αλλά και από άλλες παραμέτρους. Μερικές από αυτές τις παραμέτρους είναι η τιμή του gas που θέτει ο χρήστης, η τιμή του gas που δέχονται οι miners, τον φόρτος του δικτύου κτλ.

Στο ιδιωτικό Ethereum blockchain (που κατασκεύαζα με το εργαλείο ganache-cli) ο χρόνος αναμονής της κάθε συναλλαγής ήταν αμελητέος.

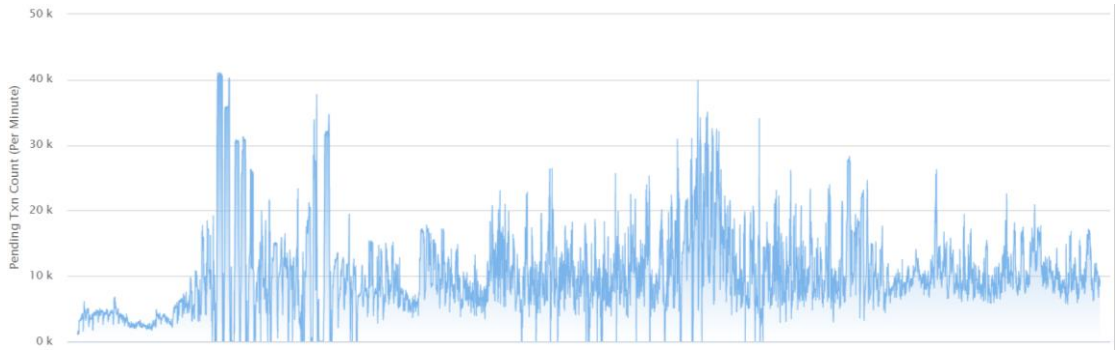
Στο Ropsten Test Net, κατά το χρονικό διάστημα στο οποίο πραγματοποίησα τα test ο χρόνος αναμονής για την κάθε συναλλαγή κυμαινόταν από μερικά δευτερόλεπτα έως λίγα λεπτά (έως 4 λεπτά).

Στο Ethereum δίκτυο (και στο κυρίως και στα Test Nets) ο χρόνος αναμονής της κάθε συναλλαγής μπορεί να ποικίλλει από μερικά δευτερόλεπτα έως μερικές ώρες, αναλόγως της απαίτησης σε gas της συναλλαγής και της τιμής του gas που προσφέρεται στους miners.

Σε κάθε περίπτωση οι συναρτήσεις πρέπει να έχουν την ελάχιστη απαίτηση σε gas, διότι έτσι οι συναλλαγές θα πραγματοποιούνται γρηγορότερα και με μικρότερη προμήθεια προς τους miners. $Προμήθεια = Gas * Gas\ Price$ βλέπουμε λοιπόν ότι με μικρότερη απαίτηση σε gas μπορούμε να προσφέρουμε μεγαλύτερο gas price και έτσι η συναλλαγή μας να πραγματοποιηθεί γρηγορότερα, αυξάνοντας την απόδοση όλης της εφαρμογής.

Ακολουθούν διαγράμματα που δείχνουν την μεταβολή της τιμής του gas και του μεγέθους της ουράς αναμονής των συναλλαγών προς επικύρωση με την πάροδο του χρόνου στο Ethereum Main Net.

Εικόνα 5.16 Τιμή του gas (Πηγή: etherscan.io)



Εικόνα 5.17 Μέγεθος ουράς αναμονής (Πηγή: etherscan.io)

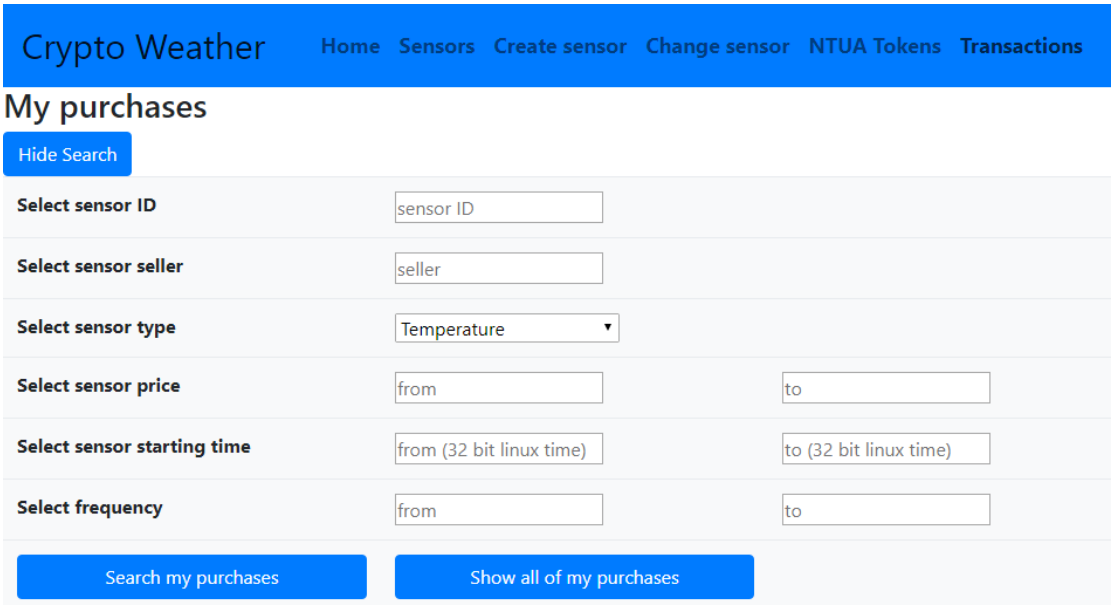
6

Επίδειξη λειτουργικότητας εφαρμογής

Στο κεφάλαιο αυτό θα γίνει παρουσίαση της εφαρμογής και του τρόπου λειτουργίας της. Η παρουσίαση του κεφαλαίου αυτού αφορά το κομμάτι της εφαρμογής που αφορά τον χρήστη, ενώ στο προηγούμενο κεφάλαιο γίνεται παρουσίαση του τρόπου υλοποίησής της.

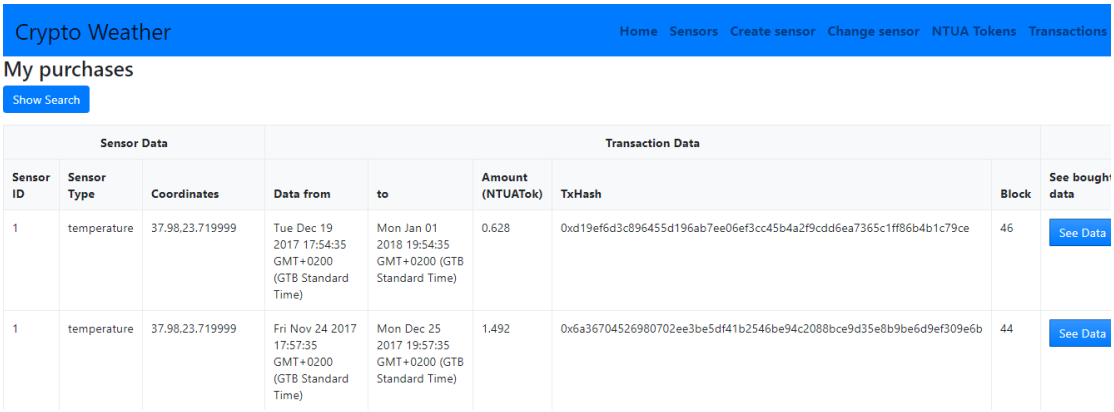
Η εφαρμογή αποτελείται από 6 βασικές ιστοσελίδες, την σελίδα Home (index), την σελίδα Sensors, την Create Sensor, την Change Sensor, την NTUA Tokens και την σελίδα Transactions.

Η σελίδα Home είναι η σελίδα στην οποία ο χρήστης μπορεί να αναζητήσει και δει όλες τις αγορές τις οποίες έχει πραγματοποιήσει.



Εικόνα 6.1 Home, αναζήτηση αγορών

Βλέπουμε ότι ο χρήστης έχει την δυνατότητα σύνθετης αναζήτησης των συναλλαγών του, όπως αυτές υπάρχουν στο Ethereum blockchain. Μπορεί να αναζητήσει κάποια αγορά του βάσει του ID του αισθητήρα, του πωλητή του, του τύπου του, της τιμής της κάθε μέτρησής του κτλ.

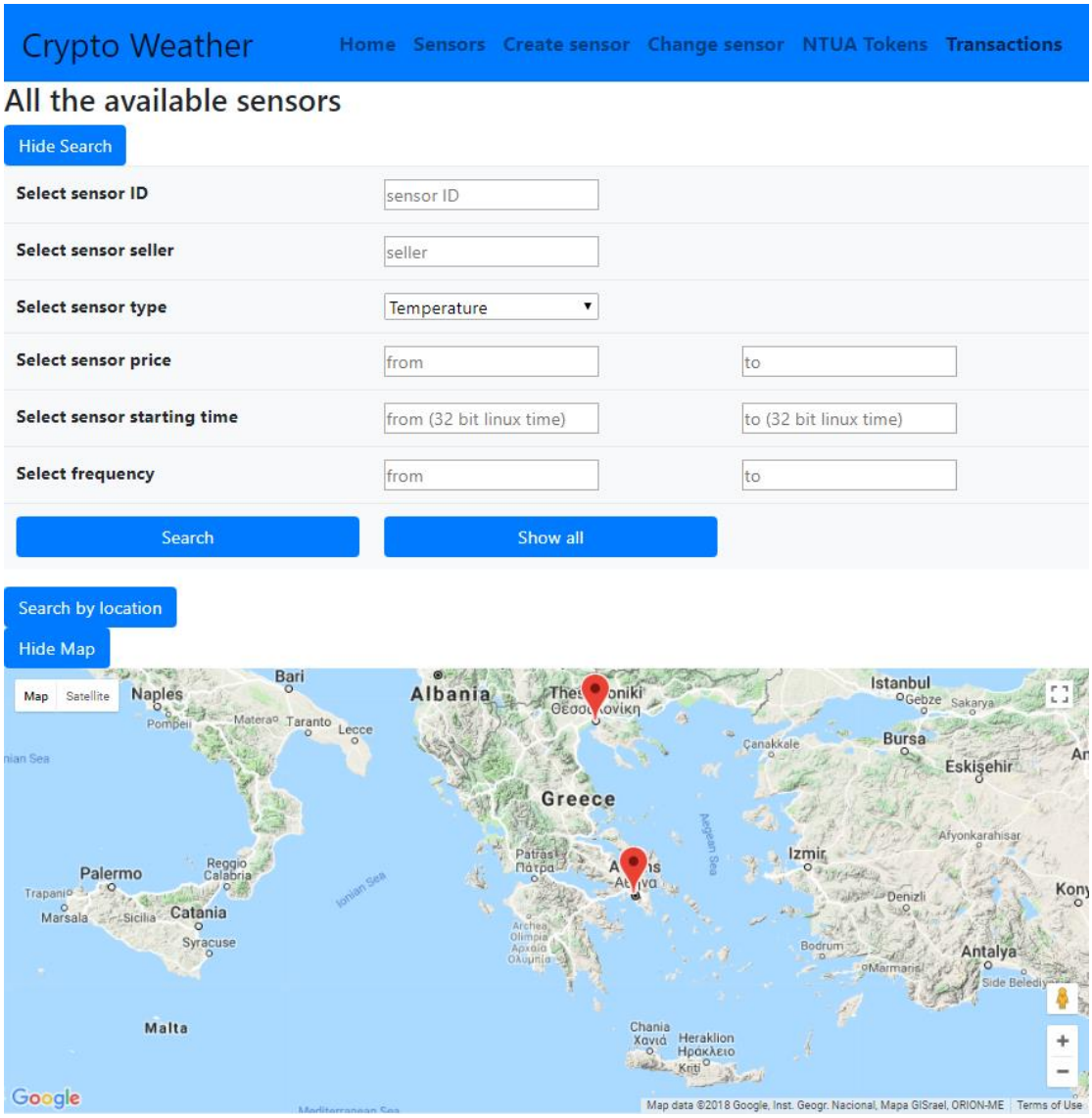


Εικόνα 6.2 Home, εμφάνιση αγορών

Αφού ο χρήστης πραγματοποιήσει την αναζήτηση, του εμφανίζονται οι αντίστοιχες αγορές. Εμφανίζονται δεδομένα που αφορούν τον αισθητήρα. Τα δεδομένα αυτά είναι, το ID, ο τύπος και οι συντεταγμένες του αισθητήρα. Επίσης εμφανίζονται δεδομένα που αφορούν την αγορά/συναλλαγή. Τα δεδομένα αυτά είναι, το διάστημα αγοράς μετρήσεων, το πληρωτέο ποσόν, το hash της συναλλαγής (TxHash – μοναδικό αναγνωριστικό κάθε συναλλαγής στο Ethereum δίκτυο) και ο αριθμός του μπλοκ στο οποίο βρίσκεται η συναλλαγή, για εύκολη εύρεσή της στο Ethereum blockchain. Τέλος εμφανίζεται το κουμπί «See Data». Όταν ο χρήστης πατήσει το κουμπί αυτό ανακατευθύνεται σε μία άλλη σελίδα, όπου του ζητείται να υπογράψει κάποια δεδομένα για να γίνει η ταυτοποίησή του, ώστε να του παρουσιαστούν οι μετρήσεις του αισθητήρα, όπως φαίνεται παρακάτω:

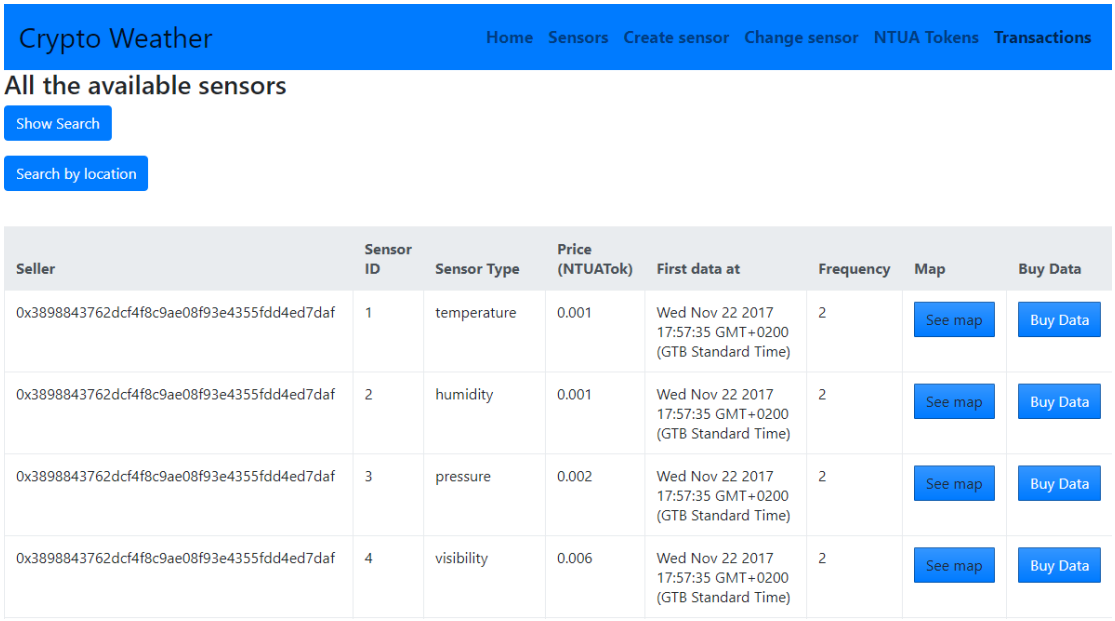
Time	Temperature (hPa)
Sat Dec 23 2017 13:20:00 GMT+0200 (GTB Standard Time)	8
Sat Dec 23 2017 13:50:00 GMT+0200 (GTB Standard Time)	8.5
Sat Dec 23 2017 14:20:00 GMT+0200 (GTB Standard Time)	8
Sat Dec 23 2017 14:50:00 GMT+0200 (GTB Standard Time)	8.5
Sat Dec 23 2017 15:20:00 GMT+0200 (GTB Standard Time)	8.5
Sat Dec 23 2017 15:50:00 GMT+0200 (GTB Standard Time)	9
Sat Dec 23 2017 16:20:00 GMT+0200 (GTB Standard Time)	8.5

Συνεχίζοντας, υπάρχει η σελίδα Sensors η οποία δίνει στον χρήστη την δυνατότητα σύνθετης αναζήτησης αισθητήρων, αλλά και η δυνατότητα αναζήτησής και εμφάνισής τους πάνω στον χάρτη (με την χρήση Google maps), όπως φαίνεται παρακάτω:



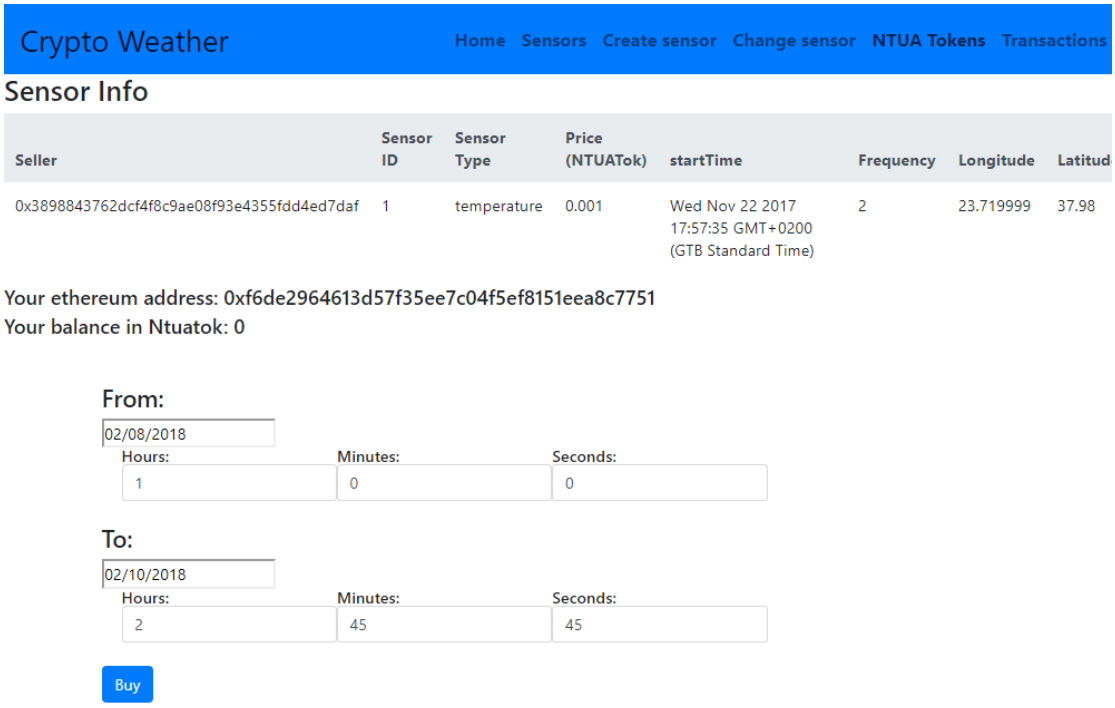
Εικόνα 6.3 Sensors, αναζήτηση αισθητήρων

Στην συνέχεια, αφού γίνει η αναζήτηση, εμφανίζονται στον χρήστη οι αισθητήρες με τα χαρακτηριστικά τους, όπως αυτοί έχουν καταχωρηθεί από τους ιδιοκτήτες τους στο Ethereum blockchain μέσω του έξυπνου συμβολαίου Broker.



Εικόνα 6.4 Sensors, εμφάνιση αισθητήρων

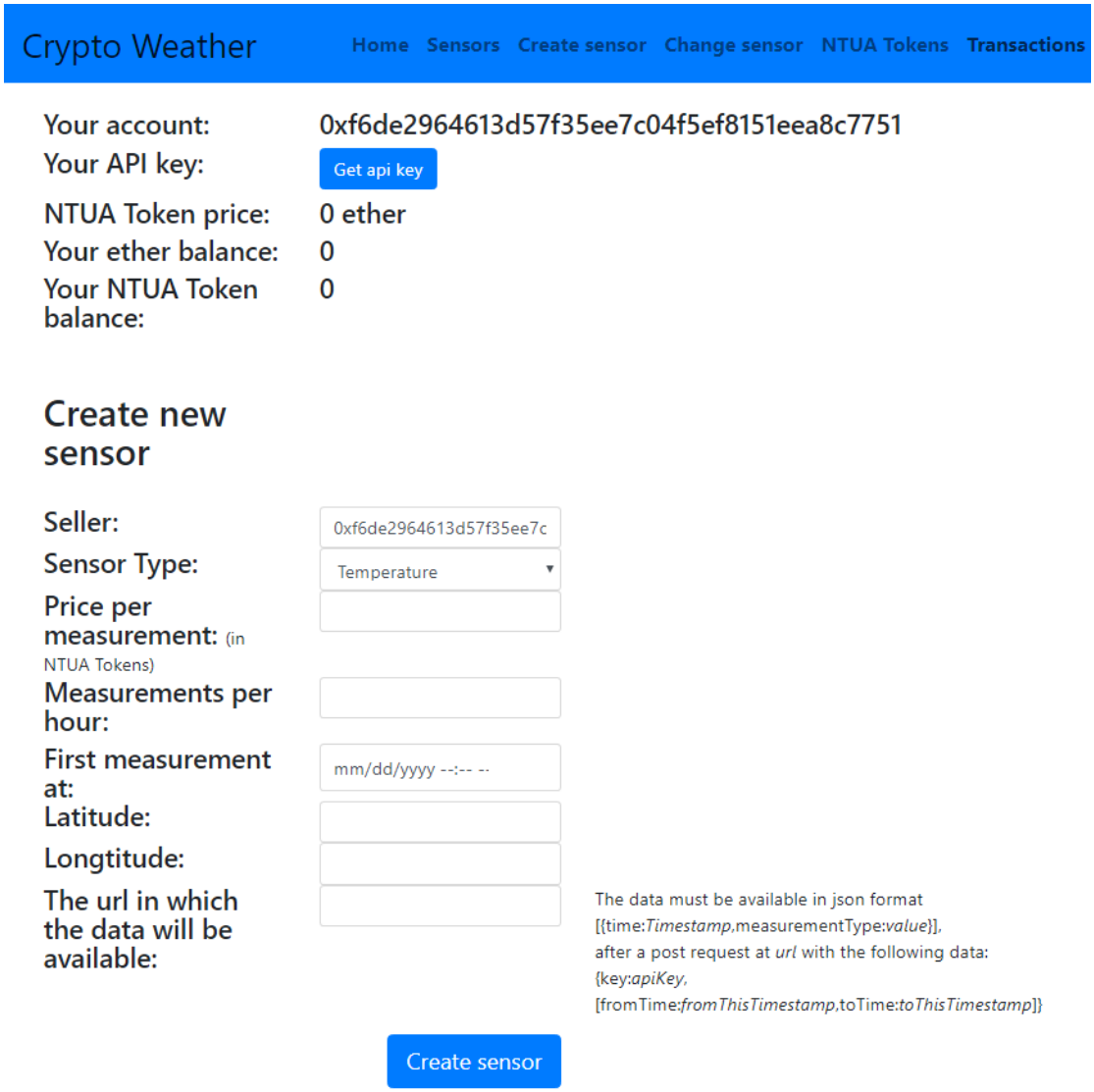
Για κάθε αισθητήρα, εκτός από τα χαρακτηριστικά του, υπάρχει η επιλογή αγοράς μετρήσεων (κουμπί «Buy Data»). Όταν πατηθεί αυτό το κουμπί ο χρήστης ανακατευθύνεται σε μία άλλη σελίδα, μέσω της οποίας δύναται να αγοράσει μετρήσεις του επιλεγμένου αισθητήρα για το χρονικό διάστημα της επιλογής του.



Εικόνα 6.5 Buy Data, *αγορά* μετρήσεων αισθητήρα

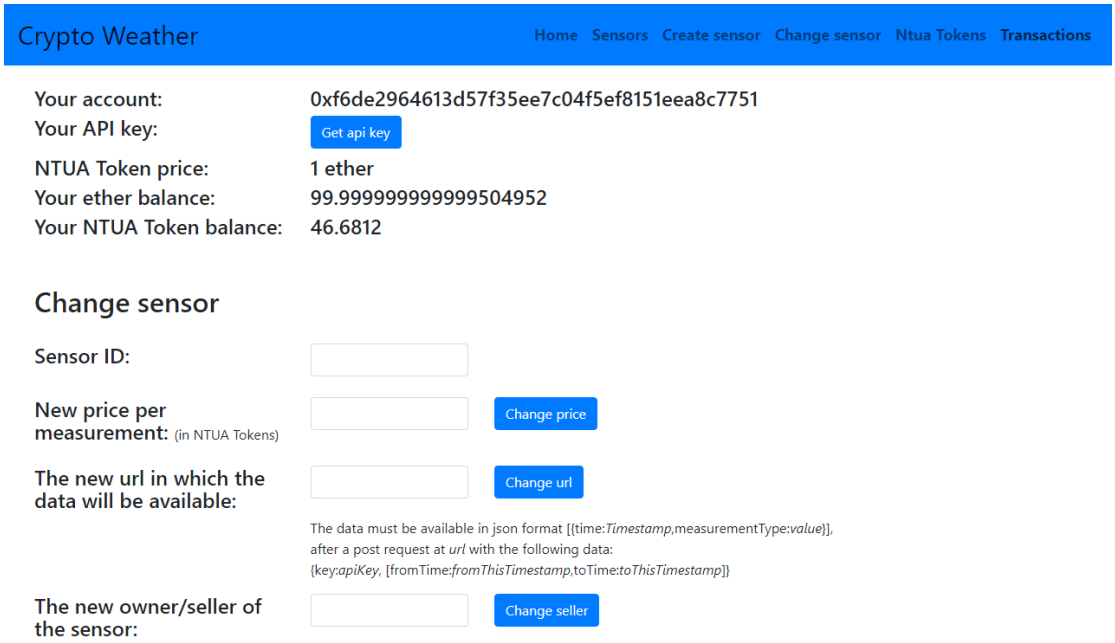
Συνεχίζοντας, η σελίδα Create Sensors παρέχει μία διεπαφή ώστε οι χρήστες να καταχωρούν αισθητήρες στο Ethereum blockchain, πιο συγκεκριμένα, αφού ο ιδιοκτήτης κάποιου αισθητήρα συμπληρώσει τα απαιτούμενα πεδία και πατήσει το κουμπί «Create sensor» θα κληθεί η συνάρτηση createSensor του συμβολαίου Broker στο

Ethereum blockchain, ώστε να καταχωρηθεί ο αισθητήρας στο blockchain. Στην σελίδα αυτή παρουσιάζονται στον χρήστη πληροφορίες όπως το υπόλοιπο του λογαριασμού του σε Ether Τέλος, πατώντας το κουμπί «Get api key» θα του εμφανιστεί ο μοναδικός κωδικός - API key που αντιστοιχεί στην Ethereum διεύθυνσή του και το οποίο χρησιμοποιείται από την εφαρμογή για την ανάκτηση των μετρήσεων των αισθητήρων του.



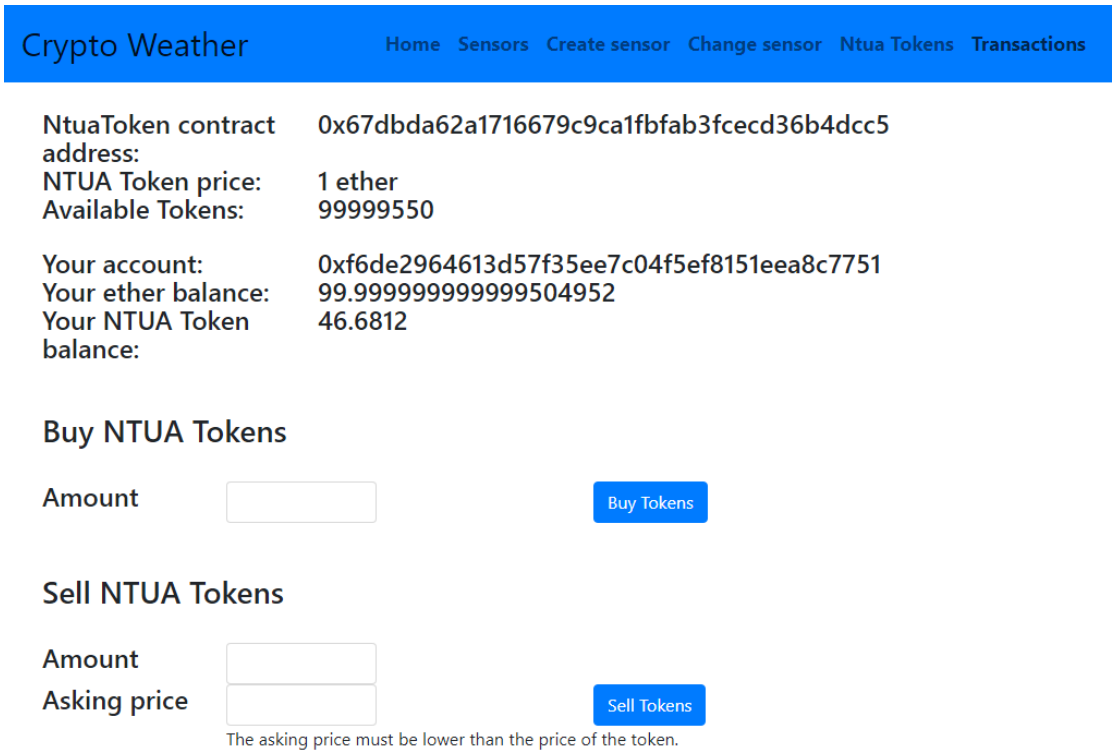
Εικόνα 6.6 Create sensor, δημιουργία αισθητήρα

Από το σύστημα προσφέρεται η δυνατότητα αλλαγής ορισμένων χαρακτηριστικών αισθητήρων που έχουν ήδη δημιουργηθεί. Η λειτουργικότητα αυτή προσφέρεται στους ιδιοκτήτες των αισθητήρων μέσω της σελίδας Change Sensor. Ο χρήστης μπορεί να αλλάξει την τιμή της κάθε μέτρησης, το url στο οποίο είναι διαθέσιμες οι μετρήσεις ή να μεταβιβάσει την κυριότητα του αισθητήρα σε άλλη διεύθυνση. Εισάγοντας το ID του αισθητήρα και την νέα τιμή του πεδίου που επιθυμεί να αλλάξει και πατώντας το κουμπί «Change …» καλείται η συνάρτηση createSensor του συμβολαίου Broker στο Ethereum blockchain, ώστε να καταχωρηθεί η αλλαγή στο blockchain. Κρίθηκε ότι αυτά τα πεδία μπορούν να αλλάξουν χωρίς να προκύψει πρόβλημα με αυτούς που ήδη έχουν αγοράσει δεδομένα για τους αισθητήρες, όπως θα υπήρχε εάν μπορούσε να αλλάξει η τοποθεσία ή ακόμη και να διαγραφεί τελείως ο αισθητήρας.



Εικόνα 6.7 Change Sensor, αλλαγή χαρακτηριστικών αισθητήρα

Η σελίδα NTUA Tokens είναι η σελίδα μέσω της οποίας οι χρήστες μπορούν να αγοράζουν και να πωλούν NTUA Tokens έναντι Ether (του κρυπτονομίσματος του Ethereum).

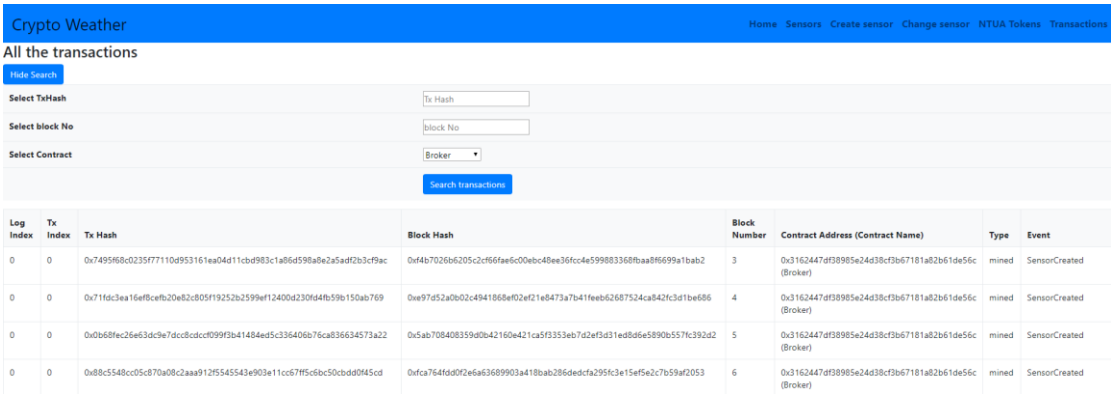


Εικόνα 6.8 NTUA Tokens, αγορά και πώληση

Τέλος οι χρήστες μπορούν να αναζητήσουν και να δουν όλα τα συμβάντα (events) που έχουν εκπεμφθεί στο blockchain από τα δύο έξυπνα συμβόλαια της εφαρμογής NtuaToken και Broker. Η σελίδα αυτή έχει δημιουργηθεί στα πρότυπα άλλων

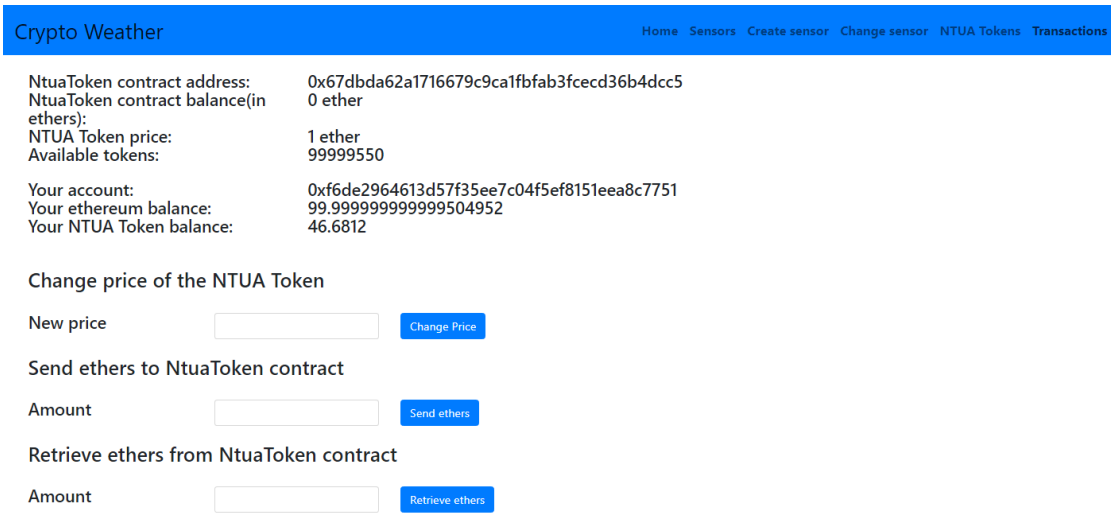
σελίδων που σκανάρουν το blockchain και εμφανίζουν στους χρήστες τους πληροφορίες για τις συναλλαγές (πχ etherscan.io, etherchain.org, ethplorer.io). Εμφανίζονται για την κάθε event οι πληροφορίες:

- Log Index – η θέση της εγγραφής στο μπλοκ.
- Tx Index – ο δείκτης της θέσης της συναλλαγής.
- Tx Hash – η hash τιμή της συναλλαγής.
- Block Hash – η hash τιμή του block στο οποίο βρίσκεται η συναλλαγή.
- Block Number – ο αριθμός του μπλοκ.
- Contract Address (Contract Name) – η διεύθυνση του συμβολαίου στο οποίο ανήκει η συναλλαγή (το όνομα του συμβολαίου).
- Type – ο τύπος του μπλοκ, πχ. mined, pending.
- Event – το όνομα του event.



Εικόνα 6.9 Transactions

Τέλος έχει δημιουργηθεί μία σελίδα η οποία προορίζεται για χρήση από τον ιδιοκτήτη των έξυπνων συμβολαίων της εφαρμογής, η σελίδα Owner. Μέσω της σελίδας αυτής ο ιδιοκτήτης των συμβολαίων μπορεί εύκολα να αλλάξει την τιμή του NTUA Token, να στείλει ether στην διεύθυνση του συμβολαίου NtuaToken καθώς και να αποσύρει ether από αυτό.



Εικόνα 6.10 Owner

7

Επίλογος

7.1 Σύνοψη και συμπεράσματα

Σκοπός της διπλωματικής εργασίας ήταν η εξέταση των νέων τεχνολογιών αποκέντρωσης και η ανάπτυξη έξυπνων συμβολαίων για την λειτουργία μίας αποκεντρωμένης εφαρμογής (DApp – Decentralized Application), η οποία θα μπορούσε να αποτελέσει κομμάτι του αποκεντρωμένου ιστού, ως επίσης και η συνεισφορά έργου στην νεότευκτη και ταχύτατα αναπτυσσόμενη κοινότητα του αποκεντρωμένου ιστού. Οι δύο κυρίαρχες τεχνολογίες που αλλάζουν τον παγκόσμιο ιστό, προσθέτοντας κατανεμημένα χαρακτηριστικά είναι το blockchain και το Internet of Things. Στην εργασία αυτή λοιπόν, έγινε προσπάθεια συνδυασμού των δύο κυρίαρχων τεχνολογιών κατανεμημένων συστημάτων, του blockchain και του Internet of Things, για την δημιουργία μίας πρωτοποριακής και χρήσιμης αποκεντρωμένης εφαρμογής. Η εφαρμογή που δημιουργήθηκε ονομάζεται «Crypto Weather» και αποτελεί μία πλατφόρμα αγοραπωλησιών μετρήσεων IoT αισθητήρων καιρού μέσω του Ethereum blockchain.

Εξετάστηκαν πολλές διαφορετικές τεχνολογίες υλοποίησης blockchain, ώστε να βρεθεί η πιο κατάλληλη και τελικά, επιλέχθηκε η αποκεντρωμένη πλατφόρμα του Ethereum. Μετά από αξιολόγηση των διαφορετικών blockchain, κρίθηκε ότι το Ethereum είναι η καταλληλότερη τεχνολογία για τον σκοπό μας. Αυτό διότι, είναι ένα από τα πιο διαδεδομένα blockchain, πιο συγκεκριμένα είναι δεύτερο μετά από αυτό του Bitcoin και είναι το μόνο οικοσύστημα που συνδυάζει την δημοφιλία με την δυνατότητα ανάπτυξης έξυπνων συμβολαίων. Επίσης στο Ethereum προσφέρονται τα περισσότερα και πιο πλήρη εργαλεία για την ανάπτυξη και λειτουργία αποκεντρωμένων εφαρμογών. Το Ethereum δηλαδή επιλέχθηκε όχι μόνον λόγω των δυνατοτήτων που προσφέρει για την λειτουργία αποκεντρωμένων εφαρμογών, αλλά και λόγω της δημοφιλίας του και της υιοθέτησης της τεχνολογίας του από τους χρήστες.

Επίσης χρησιμοποιήθηκε το Node-RED ως το κύριο εργαλείο ανάπτυξης της εφαρμογής. Ένα εργαλείο που αρχικά δημιουργήθηκε για την υλοποίηση IoT εφαρμογών (σύνδεση διαφορετικών συσκευών και διεπαφών) αλλά συνεχώς επεκτείνεται και χρησιμοποιείται σε όλο και περισσότερους τομείς. Εκτός από το κομμάτι της διασύνδεσης και προσομοίωσης των έξυπνων αισθητήρων, το Node-RED χρησιμοποιήθηκε με έναν αρκετά καινοτόμο τρόπο. Χρησιμοποιήθηκε για την ανάπτυξη ολόκληρου του ιστοτόπου της εφαρμογής και την διασύνδεση με το blockchain. Έτσι συνδυάστηκε η ανάπτυξη των περισσότερων επιμέρους κομματιών της εφαρμογής πάνω σε μία κοινή πλατφόρμα, καταλήγοντας σε μία αρκετά κομψή και απέριττη υλοποίηση.

Η παρούσα διπλωματική εργασία λοιπόν, αποτελεί μία από τις πρώτες προσπάθειες συνδυασμού της τεχνολογίας του blockchain και του Internet of Things, δύο τεχνολογιών πολλά υποσχόμενων που μπορούν να δράσουν συμπληρωματικά και στο

μέλλον να δημιουργήσουν πολλές νέες ευκαιρίες. Τελικό προϊόν της προσπάθειας αυτής, είναι η εφαρμογή «Crypto Weather».

7.2 Μελλοντικές επεκτάσεις

Οι τεχνολογίες πάνω στις οποίες βασίζεται η εφαρμογή είναι σχετικά νέες, όμως η κοινότητά τους συνεχώς μεγαλώνει και αυτές αναπτύσσονται ταχύτατα. Γίνονται μεγάλες αλλαγές σε μικρά χρονικά διαστήματα, έτσι δεν είμαστε σε θέση να προβλέψουμε με ακρίβεια την πορεία που θα ακολουθήσουν. Οι προβλέψεις είναι πολλές και διαφέρουν. Αυτό που όμως οι περισσότερες έχουν κοινό είναι η πρόβλεψη μελλοντικής επιτυχίας των τεχνολογιών του blockchain και του IoT και το γεγονός ότι θα διαδραματίσουν καθοριστικό ρόλο στο εγγύς μέλλον. Υπάρχουν επίσης ορισμένες προβλέψεις που, λόγω της συγγένειας του blockchain με το IoT, προστάζουν τον συνδυασμό των δύο και την δημιουργία κατανεμημένων συστημάτων τα οποία θα αλλάξουν άρδην το τοπίο [21], [59], [60].

Προτείνονται λοιπόν ορισμένες μελλοντικές επεκτάσεις της εφαρμογής, με την σημείωση ότι η υλοποίηση και επιτυχία μερικών εξαρτάται σε μεγάλο βαθμό από την εξέλιξη της τεχνολογίας.

- **Προσθήκη και επιπλέον τύπων αισθητήρων καιρού.** Κατά την σχεδίαση και ανάπτυξη του συστήματος, επιδιώχθηκε η δυνατότητα επεκτασιμότητας της εφαρμογής όσον αφορά τους τύπους των αισθητήρων. Πιο συγκεκριμένα έχει προβλεφθεί η δυνατότητα προσθήκης νέων τύπων αισθητήρων, έως το νούμερο των 255 διαφορετικών τύπων.
- **Αλλαγή πεδίου της εφαρμογής.** Τα έξυπνα συμβόλαια έχουν αναπτυχθεί ώστε να είναι όσο το δυνατόν πιο ουδέτερα όσον αφορά τον τύπο των αισθητήρων. Υπάρχει λοιπόν η δυνατότητα μετασχηματισμού της εφαρμογής, ώστε να αποτελέσει πλατφόρμα αγοραπωλησιών αισθητήρων διαφορετικού πεδίου, για παράδειγμα αισθητήρων σεισμικών δονήσεων κτλ.
- **Συνδυασμός των IoT αισθητήρων με το blockchain.** Εάν η τεχνολογία το επιτρέψει, στο μέλλον θα μπορούσαν οι IoT αισθητήρες καιρού να τρέχουν έναν Ethereum κόμβο. Έτσι οι ίδιοι οι αισθητήρες θα μπορούσαν να υποστηρίζουν το δίκτυο. Σε αυτήν την περίπτωση, ο ίδιος ο αισθητήρας θα μπορούσε να καταχωρείται αυτόματα στο Ethereum blockchain και έπειτα ο αγοραστής να αγοράζει μετρήσεις και να τις βλέπει, επικοινωνώντας απευθείας με αυτόν (τον αισθητήρα).
- **Πλήρης απαλοιφή της ανάγκης ιστοτόπου για την λειτουργία της εφαρμογής.** Αναλόγως την κατεύθυνση που θα πάρει η εξέλιξη της τεχνολογίας του Ethereum blockchain, μπορεί να καταστεί δυνατή η πλήρης κατάργηση του ιστοτόπου της εφαρμογής, χωρίς να υπάρξει επίπτωση στην χρηστικότητά της και την φιλικότητα προς τον χρήστη. Έτσι η εφαρμογή θα μπορούσε να αποδεσμευτεί πλήρως από το μοντέλο αρχιτεκτονικής πελάτη-εξυπηρετητή και να υιοθετήσει πλήρως το κατανεμημένο μοντέλο.

8

Βιβλιογραφία

[1] «What is the Web3? The Decentralized Web - Blockchain,» [Ηλεκτρονικό]. Available: https://blockchainhub.net/web3-decentralized-web/. [Πρόσβαση 12 1 2018].

[2] B. Pon, «Blockchain will usher the era of decentralized computing,» 15 4 2016. [Ηλεκτρονικό]. Available: https://blog.bigchaindb.com/blockchain-will-usher-in-the-era-of-decentralised-computing-7f35e94af0b6. [Πρόσβαση 12 1 2018].

[3] «Bigchain DB The scalable blockchain database,» [Ηλεκτρονικό]. Available: https://www.bigchaindb.com/. [Πρόσβαση 14 1 2018].

[4] J. Kurose και K. Ross, Δικτύωση Υπολογιστών, Αθήνα: Μ. Γκιούρδας.

[5] «devp2p/rlpx.md,» [Ηλεκτρονικό]. Available: https://github.com/ethereum/devp2p/blob/master/rlpx.md. [Πρόσβαση 10 1 2018].

[6] F. Lange, «ÐΞVp2p Wire Protocol,» [Ηλεκτρονικό]. Available: https://github.com/ethereum/wiki/wiki/%C3%90%CE%9EVp2p-Wire-Protocol. [Πρόσβαση 19 1 2018].

[7] J. Ray, «Ethereum introduction,» [Ηλεκτρονικό]. Available: https://github.com/ethereum/wiki/wiki/Ethereum-introduction. [Πρόσβαση 19 1 2018].

[8] «Elliptic Curve Cryptography (ECC),» [Ηλεκτρονικό]. Available: https://www.certicom.com/content/certicom/en/ecc.html. [Πρόσβαση 10 1 2018].

[9] L. C. WASHINGTON, Elliptic Curves Number Theory and Cryptography, Maryland: Chapman & Hall, 2008.

[10] V. Buterin, «Exploring Elliptic Curve Pairings,» [Ηλεκτρονικό]. Available: https://medium.com/@VitalikButerin/exploring-elliptic-curve-pairings-c73c1864e627. [Πρόσβαση 10 1 2018].

[11] A. S. Tanenbaum και M. v. Steen, Κατανεμημένα Συστήματα, Αθήνα: Κλειδάριθμος, 2005, p. 26.

[12] S. Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System,» 2008. [Ηλεκτρονικό]. Available: https://bitcoin.org/bitcoin.pdf.

[13] «What is Ethereum?,» [Ηλεκτρονικό]. Available: http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html. [Πρόσβαση 19 1 2018].

[14] waygie, «White Paper,» [Ηλεκτρονικό]. Available: https://github.com/ethereum/wiki/wiki/White-Paper#ethereum. [Πρόσβαση 19 1 2018].

[15] V. Buterin, «On Stake,» 5 7 2014. [Ηλεκτρονικό]. Available: https://blog.ethereum.org/2014/07/05/stake/. [Πρόσβαση 15 1 2018].

[16] N. Savers, «Ethereum Wire Protocol,» [Ηλεκτρονικό]. Available: https://github.com/ethereum/wiki/wiki/Ethereum-Wire-Protocol. [Πρόσβαση 19 1 2018].

[17] G. Wood, «Ethereum Yellow Paper: a formal specification of Ethereum, a programmable blockchain,» 7 2 2018. [Ηλεκτρονικό]. Available: https://ethereum.github.io/yellowpaper/paper.pdf. [Πρόσβαση 7 2 2018].

[18] «Flow-based Programming,» [Ηλεκτρονικό]. Available: http://www.jpaulmorrison.com/fbp/. [Πρόσβαση 9 1 2018].

[19] K. Noyen, D. Volland, D. Wörner και E. Fleisch, «When Money Learns to Fly: Towards Sensing as a Service Applications Using Bitcoin,» 20 9 2014. [Ηλεκτρονικό]. Available: https://arxiv.org/abs/1409.5841. [Πρόσβαση 1 2 2018].

[20] D. Wörner και T. v. Bomhard, «When Your Sensor Earns Money: Exchanging Data for Cash with Bitcoin,» 17 9 2014. [Ηλεκτρονικό]. Available: https://dl.acm.org/citation.cfm?id=2638786. [Πρόσβαση 5 2 2018].

[21] K. Christidis και M. Devetsikiotis, «Blockchains and Smart Contracts for the Internet of Things,» 10 5 2016. [Ηλεκτρονικό]. Available: http://ieeexplore.ieee.org/document/7467408/. [Πρόσβαση 4 1 2018].

[22] N. Kshetri, «Can Blockchain Strengthen the Internet of Things?,» 17 8 2017. [Ηλεκτρονικό]. Available: http://ieeexplore.ieee.org/abstract/document/8012302/. [Πρόσβαση 5 2 2018].

[23] H. Shafagh, L. Burkhalter, A. Hithnawi και S. Duquennoy, «Towards Blockchain-based Auditable Storage and Sharing of IoT Data,» 14 11 2017. [Ηλεκτρονικό]. Available: https://arxiv.org/pdf/1705.08230.pdf. [Πρόσβαση 4 2 2018].

[24] S. Huh, S. Cho και S. K. , «Managing IoT devices using blockchain platform,» 30 3 2017. [Ηλεκτρονικό]. Available: http://ieeexplore.ieee.org/abstract/document/7890132/. [Πρόσβαση 15 1 2018].

[25] DataBroker DAO, «DataBrokerDAO · Global market for local data,» [Ηλεκτρονικό]. Available: https://databrokerdao.com/. [Πρόσβαση 25 2 2018].

[26] F. Lange, «Geth * ethereum/go-ethereum Wiki,» [Ηλεκτρονικό]. Available: https://github.com/ethereum/go-ethereum/wiki/geth. [Πρόσβαση 4 1 2018].

[27] «Ganache CLI,» [Ηλεκτρονικό]. Available: https://github.com/trufflesuite/ganache-cli. [Πρόσβαση 3 1 2018].

[28] «ethereumjs by ethereumjs,» [Ηλεκτρονικό]. Available: http://ethereumjs.github.io/. [Πρόσβαση 2 1 2018].

[29] «ethereumjs-util,» [Ηλεκτρονικό]. Available: https://github.com/ethereumjs/ethereumjs-util. [Πρόσβαση 2 1 2018].

[30] «Web3.js,» [Ηλεκτρονικό]. Available: https://github.com/ethereum/web3.js/. [Πρόσβαση 4 1 2018].

[31] «Solidity documentation,» [Ηλεκτρονικό]. Available: https://solidity.readthedocs.io/en/develop/. [Πρόσβαση 4 1 2018].

[32] «Remix - Solidity IDE,» [Ηλεκτρονικό]. Available: http://remix.readthedocs.io/en/latest/. [Πρόσβαση 6 2 2018].

[33] «Remix -Solidity IDE,» [Ηλεκτρονικό]. Available: https://remix.ethereum.org. [Πρόσβαση 6 2 2018].

[34] «MetaMask,» [Ηλεκτρονικό]. Available: https://metamask.io/. [Πρόσβαση 5 1 2018].

[35] «What is npm?,» [Ηλεκτρονικό]. Available: https://docs.npmjs.com/getting-started/what-is-npm. [Πρόσβαση 8 1 2018].

[36] «NODE-RED : About,» [Ηλεκτρονικό]. Available: https://nodered.org/about/. [Πρόσβαση 9 1 2018].

[37] «node-red-node-mysql,» [Ηλεκτρονικό]. Available: https://www.npmjs.com/package/node-red-node-mysql. [Πρόσβαση 9 1 2018].

[38] «node-red-node-openweathermap,» [Ηλεκτρονικό]. Available: https://flows.nodered.org/node/node-red-node-openweathermap. [Πρόσβαση 9 1 2018].

[39] «node-red-node-weather-underground,» [Ηλεκτρονικό]. Available: https://flows.nodered.org/node/node-red-node-weather-underground. [Πρόσβαση 9 1 2018].

[40] «node-red-node-darksky,» [Ηλεκτρονικό]. Available: https://flows.nodered.org/node/node-red-node-darksky. [Πρόσβαση 9 1 2018].

[41] JS Foundation, «Node-RED: Writing Functions,» JS Foundation, [Ηλεκτρονικό]. Available: https://nodered.org/docs/writing-functions.html. [Πρόσβαση 6 1 2018].

[42] «Docs | Node.js,» [Ηλεκτρονικό]. Available: https://nodejs.org/en/docs/. [Πρόσβαση 10 1 2018].

[43] «Node.js Introduction,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm. [Πρόσβαση 10 1 2018].

[44] «Chrome V8 | Google Developers,» [Ηλεκτρονικό]. Available: https://developers.google.com/v8/. [Πρόσβαση 10 1 2018].

[45] «Bootstrap,» [Ηλεκτρονικό]. Available: https://getbootstrap.com/docs/4.0/getting-started/introduction/. [Πρόσβαση 6 1 2018].

[46] «Bootstrap Get Started,» [Ηλεκτρονικό]. Available: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp. [Πρόσβαση 6 1 2018].

[47] «JavaScript,» [Ηλεκτρονικό]. Available: https://www.javascript.com/. [Πρόσβαση 6 1 2018].

[48] «JavaScript Overview,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/javascript/javascript_overview.htm. [Πρόσβαση 15 1 2018].

[49] «jQuery API Documentation,» [Ηλεκτρονικό]. Available: https://api.jquery.com/. [Πρόσβαση 7 1 2018].

[50] «jQuery Introduction,» [Ηλεκτρονικό]. Available: https://www.w3schools.com/Jquery/jquery_intro.asp. [Πρόσβαση 7 1 2018].

[51] «MariaDB.org Supporting continuity and open collaboration,» [Ηλεκτρονικό]. Available: https://mariadb.org/. [Πρόσβαση 10 1 2018].

[52] «MariaDB System Properties,» [Ηλεκτρονικό]. Available: https://db-engines.com/en/system/MariaDB. [Πρόσβαση 10 1 2018].

[53] «~okeanos IAAS,» [Ηλεκτρονικό]. Available: https://okeanos.grnet.gr/about/what/. [Πρόσβαση 12 1 2018].

[54] «Security Considerations,» [Ηλεκτρονικό]. Available: http://solidity.readthedocs.io/en/develop/security-considerations.html. [Πρόσβαση 6 2 2018].

[55] A. Gou, «RLP,» 19 December 2017. [Ηλεκτρονικό]. Available: https://github.com/ethereum/wiki/wiki/RLP. [Πρόσβαση 6 2 2018].

[56] Team Keccak, «Keccak,» Team Keccak, [Ηλεκτρονικό]. Available: https://keccak.team/keccak.html. [Πρόσβαση 6 2 2018].

[57] Ethereum, «Application Binary Interface Specification,» Ethereum, 2017. [Ηλεκτρονικό]. Available: https://solidity.readthedocs.io/en/develop/abi-spec.html. [Πρόσβαση 12 2 2018].

[58] Metamask, «MetaMask Compatibility Guide,» [Ηλεκτρονικό]. Available: https://github.com/MetaMask/faq/blob/master/DEVELOPERS.md. [Πρόσβαση 27 10 2017].

[59] IBM, «The convergence of IoT and blockchain,» IBM, 29 1 2018. [Ηλεκτρονικό]. Available: https://developer.ibm.com/tv/convergence-iot-blockchain/. [Πρόσβαση 21 2 2018].

[60] S. Huckle, R. Bhattacharya, M. White και N. Beloff, «Internet of Things, Blockchain and Shared Economy Applications,» 21 9 2016. [Ηλεκτρονικό]. Available: https://www.sciencedirect.com/science/article/pii/S1877050916322190?via%3Dihub. [Πρόσβαση 21 1 2018].

Παράρτημα Ι: Εγχειρίδιο χρήσης

Στο παράρτημα αυτό θα γίνει παρουσίαση οδηγιών για την εγκατάσταση του συστήματος σε Windows Server και για την χρήση του συστήματος.

Α Εγκατάσταση

Εδώ παρουσιάζονται αναλυτικά τα προγράμματα που χρειάζονται για την εγκατάσταση και λειτουργία της εφαρμογής. Τα βήματα που παρουσιάζονται πραγματοποιήθηκαν για την επιτυχή εγκατάσταση της εφαρμογής σε Windows Server 2012 R2 Datacenter, σε Windows 10 Home και σε Windows 10 Pro χωρίς πρόβλημα.

Βήμα 1

Θα χρειαστεί το σύστημα διαχείρισης πακέτων NPM. Ο installer του npm (συμπεριλαμβάνεται ο Node.js) βρίσκεται εδώ: https://nodejs.org/en/.

Βήμα 2

Θα πρέπει να ληφθεί ο client για την εκτέλεση του κόμβου του Ethereum blockchain. Πιο συγκεκριμένα χρησιμοποιήθηκε ο geth, το command line interface υλοποιημένο στην γλώσσα προγραμματισμού Go. Ο installer του geth βρίσκεται εδώ: https://geth.ethereum.org/downloads/.

Να σημειωθεί ότι αφού ληφθεί και εγκατασταθεί ο client θα πρέπει να συγχρονιστεί με το υπόλοιπο Ethereum δίκτυο. Η διαδικασία αυτή θα καθυστερήσει, αφού πρέπει ο κόμβος να λάβει όλα τα παρελθόντα μπλοκ του blockchain (η ένα μέρος του αν επιλεχθεί γρήγορος συγχρονισμός) από το υπόλοιπο δίκτυο.

Για απλό έλεγχο λειτουργίας της εφαρμογής σε ένα ιδιωτικό Ethereum blockchain αρκεί η λήψη του ganache-cli ως npm πακέτου με την παρακάτω εντολή:
npm install ganache-cli

Βήμα 3

Θα χρειαστεί το εργαλείο Node-RED. Οδηγίες για την εγκατάσταση του Node-RED βρίσκονται εδώ: https://nodered.org/docs/getting-started/installation.

Βήμα 4

Θα χρειαστούν τα εξής npm πακέτα: web3, fs, solc, ethereumjs-util, git, node-red-node-mysql, node-red-node-openweathermap, node-red-node-weather-underground και node-red-node-darksky. Για την εγκατάστασή τους θα ανοίξουμε ένα command line, θα μεταβούμε στο directory $HOME/.node-red και θα εισάγουμε τις παρακάτω εντολές:

- npm init
- npm install git@0.1.5
- npm install web3@0.19.1 –save
- npm install file-system
- npm install solc
- npm install ethereumjs-util@5.1.2
- npm install node-red-node-mysql@0.0.16
- npm install node-red-node-openweathermap@0.1.20
- npm install node-red-node-weather-underground@0.1.11
- npm install node-red-node-darksky@0.1.17

Εάν όλα πάνε καλά (δεν υπάρξει κάποιο πρόβλημα πχ. δικτύου κτλ.) θα έχει δημιουργηθεί ένας νέος φάκελος με το όνομα node_modules που θα περιέχει όλα τα παραπάνω modules που κατεβάσαμε.

Βήμα 5

Αλλαγή του configuration file settings.js, έτσι ώστε να είναι διαθέσιμα τα πακέτα στις συναρτήσεις των κόμβων του Node-RED. Ανοίγουμε το αρχείο settings.js και βρίσκουμε το πεδίο functionGlobalContext. Εκεί προσθέτουμε μία εγγραφή για κάθε πακέτο ώστε τελικά να έχουμε:

```
functionGlobalContext: {
    web3:require('web3'),
    fs:require('fs'),
    solc:require('solc'),
    ethereumjs:require('ethereumjs-util')
}
```

Βήμα 6

Θα πρέπει να γίνει εγκατάσταση της MariaDB ή κάποιου άλλου SQL like συστήματος βάσεων δεδομένων (πχ. MySQL). Ο installer της MariaDB βρίσκεται εδώ: https://downloads.mariadb.org/.

Βήμα 7

Ανοίγουμε τον command line client της MariaDB και δημιουργούμε δύο βάσεις δεδομένων, μία για τις καταχωρίσεις του blockchain, την diplwmatikisensors και μία για τα δεδομένα για την προσομοίωση των αισθητήρων καιρού, diplwmatikidata.

```
MariaDB [(none)]> create database diplwmatikiSensors
MariaDB [(none)]> create database diplwmatikiData
```

Έπειτα επιλέγω την βάση diplwmatikiData

```
MariaDB [(none)]> use  diplwmatikiData
```

και φορτώνω τα δεδομένα από το αρχείο diplwmatikidata.sql

```
MariaDB [(diplwmatikiData)]> source diplwmatikidata.sql
```

Βήμα 8

Αφού έχουν εγκατασταθεί όλα τα ανωτέρω προγράμματα και λειτουργούν σωστά θα «τρέξουμε» το Node-RED, ανοίγοντας ένα cmd και πληκτρολογώντας την εντολή «node-red». Ύστερα θα ανοίξουμε έναν browser και θα επισκεφθούμε την διεύθυνση «http://localhost:1880» (πόρτα 1880 στο localhost). Θα εμφανιστεί το γραφικό περιβάλλον (editor) του Node-RED, πάνω δεξιά ανοίγουμε τις επιλογές, «Import», «Clipboard» και κάνουμε paste το περιεχόμενο του αρχείου «all-flows.json» και επιβεβαιώνουμε. Θα ανοίξουν στο Node-RED όλες οι ροές. Ενδέχεται μερικοί κόμβοι να απαιτούν κάποια επεξεργασία (πχ. ο darksky χρειάζεται εισαγωγή του API key). Οι κόμβοι mysql, θα απαιτούν να εισαχθεί το όνομα της βάσης καθώς και τα διαπιστευτήρια του χρήστη της βάσης. Ο κόμβος darksky να ξαναζητήσει τα διαπιστευτήρια κτλ. Αφού τελειώσουμε με αυτά, πατάμε το κουμπί Deploy που βρίσκεται πάνω δεξιά, το οποίο και θα κάνει Deploy όλες τις ροές.

Βήμα 9

Η εγκατάσταση όλων των προγραμμάτων έχει πραγματοποιηθεί και η ιστοσελίδα λειτουργεί. Για να είναι η ιστοσελίδα ορατή και έξω από το localhost πρέπει να αλλάξουμε το settings.js αρχείο.
Επιλέγουμε την πόρτα στην οποία θέλουμε το Node-RED να ακούει:

```
// the tcp port that the Node-RED web server is listening on
uiPort: process.env.PORT || 1880
```

Στην συνέχεια «κλειδώνουμε» τον editor του Node-RED, όπως παρακάτω:

```
//Securing Node-RED
// ----------------
// To password protect the Node-RED editor and admin API, the following
// property can be used. See http://nodered.org/docs/security.html for details.
adminAuth: {
 type: "credentials",
 users: [{
 username: "admin",
 password: "$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNogENOMcxWV9DN.",
       permissions: "*"
 }]
}
```

Βήμα 10

Κάνουμε deploy τα συμβόλαια προσέχοντας και αλλάζοντας εάν χρειαστεί τις διευθύνσεις τους μέσα στον Solidity κώδικα, (εάν δεν έχουν ήδη γίνει) από την ροή «Deploy contracts».

Δημιουργούμε τους πίνακες της βάσης δεδομένων diplwmatikisensors και θέτουμε όλους τους event listeners από την ροή «Events listeners».

Η εφαρμογή τώρα είναι έτοιμη.

Βήμα 11 (προαιρετικό)

Εάν θέλουμε να χρησιμοποιήσουμε τα δεδομένα καιρού από τα Open Weather Map, Weather Underground και Dark Sky, θα τα βρούμε στο αρχείο diplw-matikidata.sql. Ο πιο εύκολος τρόπος είναι να δημιουργήσουμε μία βάση δεδομένων, create diplwmatikidata και μετά να προσθέσουμε σε αυτήν τα δεδομένα: source diplwmatikidata.sql.

Β Χρήση εφαρμογής

Στην προηγούμενη ενότητα δόθηκαν οδηγίες για την εγκατάσταση της εφαρμογής σε κάποιον εξυπηρετητή Windows Server 2012 R2 Datacenter ή Windows 10 Home και Windows 10 Pro.

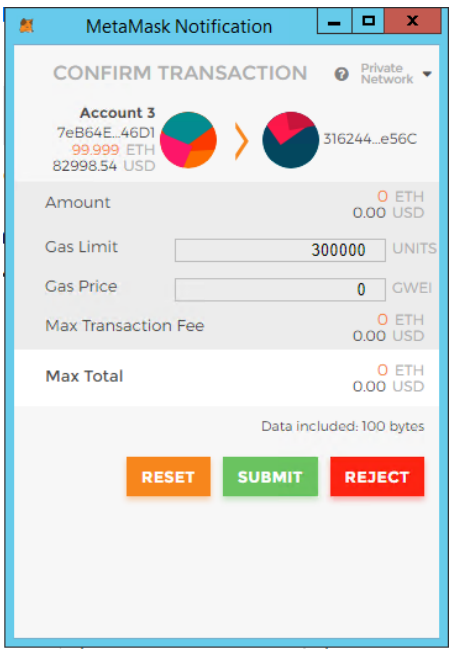
Με την προϋπόθεση ότι η ιστοσελίδα λειτουργεί κανονικά, εδώ θα δοθούν οδηγίες προς τους χρήστες της εφαρμογής.

Ο χρήστης θα πρέπει να «τρέχει» έναν Ethereum κόμβο στο μηχάνημά του εάν χρησιμοποιεί τον Mist browser. Εάν δεν θέλει να τρέχει τον πλήρη Ethereum κόμβο, τότε έχει την επιλογή να κατεβάσει το Metamask plugin για τον browser του.

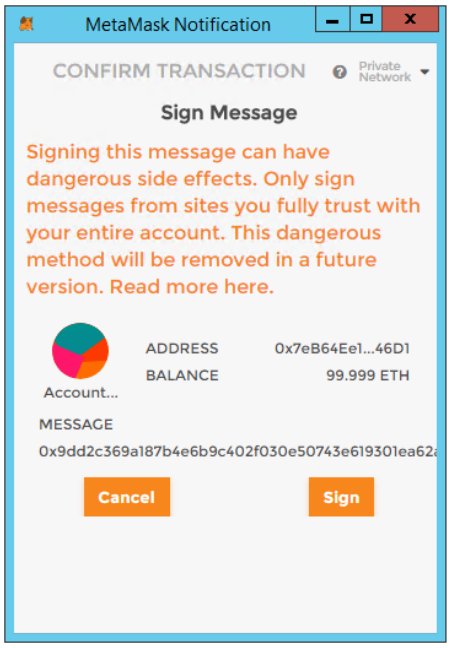
Το Metamask είναι διαθέσιμο στους φυλλομετρητές: Chrome, Firefox και Opera. Για τον Google Chrome είναι διαθέσιμο στο παρακάτω link: https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=en.

Έπειτα ο χρήστης προσθέτει τον Ethereum λογαριασμό του στο Metamask και είναι έτοιμος να χρησιμοποιήσει την εφαρμογή.

Εάν επιλεγεί η λύση του Metamask, κάθε φορά που ο χρήστης προσπαθεί να κάνει μία συναλλαγή (πχ. να αγοράσει δεδομένα για έναν αισθητήρα), θα του εμφανίζεται ένα pop-up μήνυμα από το Metamask, στο οποίο συμπληρώνει πεδία που αφορούν το όριο της προμήθειας που είναι διατεθειμένος να πληρώσει για την συναλλαγή αυτήν, καθώς και η δυνατότητα να την αποδεχτεί ή να την απορρίψει.



Επίσης κάθε φορά που θα επιθυμεί να κάνει κάποια ενέργεια η οποία απαιτεί την διαπίστευσή του, για παράδειγμα να δει τα δεδομένα που αγόρασε θα πρέπει να υπογράψει κάποια δεδομένα με το ιδιωτικό κλειδί του.



Χρησιμοποιείται η συνάρτηση web3.eth.sign για την οποία, κατά την στιγμή της ανάπτυξης της εφαρμογής, υπήρχε η παρατήρηση ότι θα αντικατασταθεί από κάποια άλλη συνάρτηση. Το πρόβλημα είναι πως αυτή η μέθοδος είναι ευάλωτή σε κρυπτογραφικές επιθέσεις γνωστού κρυπτοκειμένου. Η μέθοδος χρησιμοποιήθηκε επειδή δεν υπήρχε άλλη εναλλακτική και εύκολα θα αντικατασταθεί όταν προταθεί η νέα λύση.

Εάν εξαιρέσει κανείς την ιδιαίτερη απαίτηση της εφαρμογής για την χρήση ειδικού browser ή κάποιου plugin, η χρήση της είναι πολύ εύκολη από τον μέσο χρήστη. Το σίγουρο είναι ότι όσο περισσότεροι άνθρωποι χρησιμοποιούν τις τεχνολογίες του Ethereum, τόσο αυτό θα αναπτύσσεται και η χρήση τέτοιου είδους εφαρμογών θα γίνεται όλο και πιο εύκολη, όλο και πιο προσιτή στους απλούς χρήστες. Μία πολύ αισιόδοξη πρόβλεψη θα έλεγε πως στο εγγύς μέλλον οι browser θα έχουν ενσωματωμένο ηλεκτρονικό πορτοφόλι για τα πιο γνωστά κρυπτονομίσματα, όπως είναι το Ethereum, έτσι θα σταματήσουν να υπάρχουν ειδικές απαιτήσεις από τους χρήστες και αυτό θα συμβάλλει στην περαιτέρω εξάπλωση της χρήσης των αποκεντρωμένων εφαρμογών.

Παράρτημα II: Κώδικες

Στο παράρτημα αυτό βρίσκεται ο κώδικας των δύο έξυπνων συμβολαίων που αναπτύχθηκαν στα πλαίσια της παρούσας διπλωματικής. Ο κώδικας που αφορά τις ροές του Node-RED βρίσκεται στο παρακάτω link. Στο link αυτό βρίσκονται επίσης και τα δεδομένα των αισθητήρων καιρού.

https://github.com/george500/diplwmatiki_ergasia

A Έξυπνα συμβόλαια

Σε αυτήν την ενότητα βρίσκεται ο κώδικας των έξυπνων συμβολαίων (smart contracts) γραμμένος στην γλώσσα Solidity.

Ο πηγαίος κώδικας του συμβολαίου NtuaToken:

```
pragma solidity ^0.4.16 ;

contract NtuaToken {

    //EVENTS
    event Transfer(address indexed from, address indexed to, uint256
value) ;

    //METHODS
    //Constructor-Initializes contract with initial supply tokens to
the creator of the contract
    function NtuaToken() {
        balanceOf[msg.sender] = 100000000 * 1 ether ;
        totalSupply = 100000000 * 1 ether ;
        name = "NTUA Token" ;
        symbol= "NTUATok" ;
        decimals = 18 ;
        owner = msg.sender ;
        price = 1000000000000000000 ;          //set initial price
    }

    //contract receives ethers and gives back tokens
    function () payable {
        if(msg.sender != owner) {
           uint256 amount=msg.value*1000000000000000000/price;
            _transfer(owner, msg.sender, amount) ;
        }
    }
```

```
    //owner changes the price of the token
    function changePrice(uint _price) {
        require (msg.sender == owner) ;
        price = _price ;
    }

    //owner takes ethers from the contract
    function retrieveEthereum(uint256 amount) {
        require (msg.sender == owner) ;
        require(this.balance >= amount) ;
        owner.transfer(amount) ;
    }

    //token holders can sell  their tokens back to the owner of the
contract
    function sellBack(uint256 amount, uint _price) {
        require(_price <= price) ;  //only sell the coins if the
asking price is lower than the price of the token
        require(balanceOf[msg.sender] >= amount) ;
        uint256 totalEth = amount*price/1000000000000000000 ;
        require(this.balance >= totalEth) ;
        _transfer(msg.sender, owner, amount) ;
        msg.sender.transfer(totalEth) ;
    }

    //Internal transfer, only can be called by this contract
    function _transfer(address _from, address _to, uint _value)
internal {
        require(_to != 0x0) ;  // Prevent transfer to 0x0 address.
Use burn() instead
        require(balanceOf[_from] >= _value) ; // Check if the sender
has enough
        // Check for overflows
        require(balanceOf[_to] + _value > balanceOf[_to]) ;
        balanceOf[_from] -= _value ; //Subtract from the sender
        balanceOf[_to]+=_value;      //Add the same to the recipient
        Transfer(_from, _to, _value) ;
    }

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

    function brokerTransferFrom(address _from, address _to, uint256
_value) returns (bool success) {
        require(msg.sender ==
0xb1719bf761175017eeb6bb72423ece49914fef9b) ; //the broker contract
will never try to steal, we trust it
        _transfer(_from, _to, _value) ;
        return true ;
    }
```

```
    //FIELDS
    string public name ;            //name of the token
    string public symbol ;          //symbol of the token
    uint8 public decimals ;         //decimals of the token
    uint256 public totalSupply ;    //total supply of the token
    address public owner ;          //the owner of the contract
    uint256 public price;      //price of one ntua Token in ethereum
    //The array with all balances
    mapping (address => uint256) public balanceOf ;
}
```

Ο πηγαίος κώδικας του συμβολαίου Broker:

```
pragma solidity ^0.4.16 ;
import "./NtuaToken.sol" ;

contract Broker {

    NtuaToken Ntua =
NtuaToken(0xd50972d2b1747e6277134d29db36e88dacf12844) ;
    //TYPES
    //struct for the information of every sensor
    struct sensor {
        address seller ;
        uint8 sensorType ;
        uint price ;
        uint32 startTime ;
        uint16 frequency ;          //measurements per hour =>
frequency*3600 measurements per second
        int32 latitude ;
        int32 longitude ;
        string url ;
    }
    struct transaction {
        uint sensorID ;
        uint32 fromTime ;            //converted in unix timestamp
        uint32 toTime ;
        uint amount ;
    }

    //EVENTS
    event SensorCreated(address indexed seller,uint32 indexed
sensorID,uint8 sensorType,uint price,uint32 startTime, uint16
frequency,int32 latitude,int32 longtitude,string url);
    event SensorChangedSeller(uint32 sensorID,address seller);
    event SensorChangedPrice(uint32 sensorID, uint price) ;
    event SensorChangedUrl(uint32 sensorID, string url) ;
    event CompletedTransaction(uint32 transID, address indexed
buyer, uint32 indexed sensorID, uint32 fromTime, uint32 toTime, uint
amount) ;

    //METHODS
    //Constructor
    function Broker() {
        id = 1 ;
        transID = 1 ;
    }
    //functions for the sellers
    function createSensor(address seller1, uint8 type1, uint price1,
uint32 startTime1, uint16 frequency1, int32 latitude1, int32
longitude1, string url1) external {
```

```
        sensors[id].seller = seller1 ;
        sensors[id].sensorType = type1 ;
        sensors[id].price = price1 ;
        sensors[id].startTime = startTime1 ;
        sensors[id].frequency = frequency1 ;
        sensors[id].latitude = latitude1 ;
        sensors[id].longitude = longitude1 ;
        sensors[id].url = url1 ;
        SensorCreated(seller1, id, type1, price1, startTime1,
frequency1, latitude1, longitude1, url1) ;
        id++ ;          //give unique id identifier
    }

    function changeSensorSeller(uint32 sensorID1, address seller1)
external {
        require(msg.sender == sensors[sensorID1].seller) ;
        sensors[sensorID1].seller = seller1 ;
        SensorChangedSeller(sensorID1,seller1) ;
    }

    function changeSensorPrice(uint32 sensorID1, uint price1)
external {
        require(msg.sender == sensors[sensorID1].seller) ;
        sensors[sensorID1].price = price1 ;
        SensorChangedPrice(sensorID1,price1) ;
    }
    function changeSensorUrl(uint32 sensorID1, string url1) external
{
        require(msg.sender == sensors[sensorID1].seller) ;
        sensors[sensorID1].url = url1 ;
        SensorChangedUrl(sensorID1,url1) ;
    }

    //functions for the buyers
    function buyData(uint32 sensorID, uint32 fromTime, uint32
toTime) external {
        address seller = sensors[sensorID].seller ;
        require(fromTime >= sensors[sensorID].startTime) ;
        uint32 interval = toTime - fromTime ;
        require(interval >= 3600) ;     //no less than 1 hour
        uint amount = interval * sensors[sensorID].frequency *
sensors[sensorID].price / 3600 ;

        Ntua.brokerTransferFrom(msg.sender, seller, amount) ;

        transactions[msg.sender].push(transaction(sensorID,
fromTime, toTime, amount)) ;
        CompletedTransaction(transID, msg.sender, sensorID,
fromTime, toTime, amount) ;
        transID ++ ;
    }
```

```
    //FIELDS
    //auto generated SensorID
    uint32 id ;
    //auto generated TransactionID
    uint32 transID ;
    //all the sensors in the system (sensorID => sensor)
    mapping(uint => sensor) sensors ;
    //all the transactions completed (buyer => sensorID)
    mapping(address => transaction[]) transactions ;
}
```