



**ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ  
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ  
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ  
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

## **ΕΦΑΡΜΟΓΗ ΣΕ ΥΠΟΔΟΜΗ BLOCKCHAIN**

**Μια Υπηρεσία Επικύρωσης (Notarization) για την Ανάκτηση Ιατρικής  
Πληροφορίας**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

**ΚΛΕΙΝΑΚΗ ΑΘΗΝΑ ΣΤΥΛΙΑΝΗ**

**ΕΠΙΒΛΕΠΩΝ: ΕΦΡΑΙΜΙΔΗΣ ΠΑΥΛΟΣ  
ΑΝΑΠΛ. ΚΑΘΗΓΗΤΗΣ Δ.Π.Θ**

**ΞΑΝΘΗ, ΙΟΥΛΙΟΣ 2018**





**ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ  
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ  
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ  
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

**ΕΦΑΡΜΟΓΗ ΣΕ ΥΠΟΔΟΜΗ BLOCKCHAIN**

**Μια Υπηρεσία Επικύρωσης (Notarization) για την Ανάκτηση Ιατρικής  
Πληροφορίας**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ  
της  
ΚΛΕΙΝΑΚΗ ΑΘΗΝΑΣ ΣΤΥΛΙΑΝΗΣ**

**ΕΠΙΒΛΕΠΩΝ: ΕΦΡΑΙΜΙΔΗΣ ΠΑΥΛΟΣ  
ΑΝΑΠΛ. ΚΑΘΗΓΗΤΗΣ Δ.Π.Θ**

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5<sup>η</sup> Ιουλίου 2018

(Υπογραφή)

.....  
Εφραιμίδης Π.  
Αναπλ. Καθηγητής Δ.Π.Θ

(Υπογραφή)

.....  
Συρακούλης Γ.  
Καθηγητής Δ.Π.Θ

(Υπογραφή)

.....  
Τσαουσίδης Β.  
Καθηγητής Δ.Π.Θ.

Ξάνθη, Ιούλιος 2018



## **ΕΥΧΑΡΙΣΤΙΕΣ**

Με το τέλος της παρούσας διπλωματικής εργασίας αισθάνομαι την υποχρέωση να ευχαριστήσω θερμά τον επιβλέποντά μου, κ. Εφραιμίδη Παύλο, αναπληρωτή καθηγητή, για την εμπιστοσύνη που μου έδειξε για την εκπόνηση αυτής της εργασίας, για την πολύτιμη καθοδήγησή του, για τις συμβουλές και παρατηρήσεις του στη δομή και το περιεχόμενο της εργασίας μου, για το χρόνο που διέθεσε και για την άψογη και ουσιαστική συνεργασία μας.

Επίσης θα ήθελα να ευχαριστήσω τους κ. Δροσάτο Γεώργιο, μεταδιδάκτορα, και κ. Καλδούδη Ελένη, αναπληρώτρια καθηγήτρια για τις υποδείξεις και την πολύτιμη συνεισφορά τους και τα μέλη της εξεταστικής επιτροπής κ. Συρακούλη Γεώργιο και Τσαουσίδη Βασίλειο, καθηγητές του Δ.Π.Θ, για το χρόνο που μου αφιέρωσαν.

Επιπλέον ευχαριστίες εκφράζω και στους καθηγητές του ΔΠΘ, οι οποίοι κατά τη διάρκεια της φοίτησής μου με στήριξαν και προσπάθησαν να μου μεταδώσουν τις απαραίτητες γνώσεις.

Ιδιαίτερα θερμές ευχαριστίες απευθύνω και στην οικογένειά μου για τη συνεχή συμπαράστασή τους, υλική και ηθική.

Ξάνθη, Ιούλιος 2018

Κλεινάκη Αθηνά Στυλιανή

## Σύνοψη

Μια τεχνολογία που αναπτύσσεται με ραγδαίους ρυθμούς τα τελευταία χρόνια είναι η τεχνολογία του blockchain, η οποία εφευρέθηκε το 2008 από ένα άτομο ή μια ομάδα ατόμων, γνωστών με το ψευδώνυμο Satoshi Nakamoto. Αρχικά χρησιμοποιήθηκε για τη λειτουργία του κρυπτονομίσματος Bitcoin και από τότε μέχρι σήμερα συνεχίζει να εξελίσσεται και χρησιμοποιείται σε ένα ευρύ πεδίο εφαρμογών. Ο λόγος που η τεχνολογία αυτή είναι ιδιαίτερα δημοφιλής είναι διότι προσφέρει την αναλλοίωτη διανομή της ψηφιακής πληροφορίας που υπάρχει στο δίκτυο της, σε όλους τους συμμετέχοντες σε αυτό. Είναι μια ψηφιακή πλατφόρμα χωρίς κεντρική διαχείριση, με μια κατανεμημένη βάση δεδομένων η οποία συνεχώς αυξάνεται και θα μπορούσε να παρομοιαστεί με μια αλυσίδα χρονικά ταξινομημένων αρχείων που ονομάζονται μπλοκ. Κάθε μπλοκ συνδέεται με το προηγούμενό του και έτσι δομείται μια αλυσίδα. Εκτός από τον τομέα που αφορά τη λειτουργία των κρυπτονομισμάτων, η τεχνολογία blockchain χρησιμοποιείται και για την δημιουργία και λειτουργία αποκεντρωμένων εφαρμογών (decentralized application or DApps), οι οποίες βασίζονται σε ένα peer-to-peer δίκτυο κόμβων χωρίς κεντρική αρχή διαχείρισης. Μια από τις πιο γνωστές πλατφόρμες που χρησιμοποιούν την τεχνολογία blockchain είναι το Ethereum. Η πλατφόρμα αυτή παρέχει στους συμμετέχοντες του δικτύου της τόσο τη δυνατότητα χρήσης του δικού της ψηφιακού κρυπτονομίσματος, όσο και τη δυνατότητα υλοποίησης και λειτουργίας αποκεντρωμένων εφαρμογών.

Η τεχνολογία blockchain μπορεί να βρει εφαρμογή και στον τομέα της υγείας ώστε να διασφαλιστεί η ακεραιότητα και η μη αποποίηση των δεδομένων που παρέχονται από τις βιοϊατρικές βάσεις, δεδομένου ότι οι πληροφορίες στις βάσεις αυτές μεταβάλλονται με την πρόοδο της επιστήμης και έχουν υψηλή αξία καθώς βρίσκουν εφαρμογή στον ευαίσθητο τομέα της ιατρικής επιστήμης. Για τους λόγους αυτούς σκοπός της παρούσας διπλωματικής εργασίας είναι η χρήση της τεχνολογίας blockchain ως συμβολαιογραφικό μέσο, ώστε να δεσμεύεται ο πάροχος ιατρικής γνώσης για την πληροφορία που παρέχει. Για την επίτευξη του στόχου αυτού υλοποιήθηκε μια αποκεντρωμένη εφαρμογή η οποία βασίζεται στην πλατφόρμα του Ethereum και των εργαλείων που το υποστηρίζουν. Η εφαρμογή θα μπορούσε να προσαρμοστεί για να υποστηρίξει οποιαδήποτε βάση. Στην παρούσα διπλωματική χρησιμοποιήθηκαν η πλατφόρμα του CARRE και του PubMed. Πιο συγκεκριμένα η εφαρμογή που αναπτύχθηκε δίνει τη δυνατότητα στον χρήστη να θέσει ένα ερώτημα σε μια από τις δύο βάσεις και να πάρει την πληροφορία που επιθυμεί καθώς και μια απόδειξη ότι τα δεδομένα που έλαβε ο χρήστης δεν μπορούν να τροποποιηθούν και να αποποιηθεί η βιοϊατρική βάση ότι τα παρείχε. Παράγεται δηλαδή μια κρυπτογραφικά ασφαλής hash τιμή που σχετίζεται τόσο με το ερώτημα όσο και με την αντιστοιχουμένη σε αυτό απάντηση, η οποία στη συνέχεια αποθηκεύεται αναλλοίωτα στο δίκτυο του Ethereum blockchain μαζί με μια χρονοσφραγίδα. Έτσι ο χρήστης της εφαρμογής μπορεί να αποδείξει ότι μια συγκεκριμένη χρονική στιγμή, μετά την υποβολή ενός συγκεκριμένου ερωτήματος, έλαβε την αντίστοιχη απάντηση.

**Λέξεις κλειδιά:** Blockchain, Ethereum, Έξυπνα συμβόλαια, Αποκεντρωμένες εφαρμογές, Βιοϊατρικές βάσεις δεδομένων, Κρυπτογραφικές τεχνικές, Ακεραιότητα, Μη-αποποίηση

## Abstract

One technology that is rapidly advancing in recent years is the blockchain technology, which was invented in 2008 by one person or a group of people, known by the name Satoshi Nakamoto. At the beginning it was used for the operation of the Bitcoin cryptocurrency and from that moment until now it keeps evolving and to date it is used in a wide scope of applications. The main reason why this technology is especially popular is because it offers the unaltered distribution of digital information that exists in its network, to all the participants that are part of it. It is a digital platform operating without the need of a central authority, with a distributed base of data that is constantly increasing and may be likened with a chain of data that are in chronological order, called blocks. Every block is linked with its previous structuring a chain. Besides the sector regarding the operation of cryptocurrencies, blockchain technology is used for the development and operation of decentralized applications (DApps), which are based on a peer-to-peer network of nodes without the need of a central authority. One of the most known platforms that use the blockchain technology is Ethereum. This platform provides to the participants of its network both the capability to use the platform's cryptocurrency and the possibility to develop and operate decentralized applications.

Blockchain technology can be applied and in healthcare in order to ensure the integrity and non-repudiation of the data provided by a biomedical repository, given that the information stored in such repositories change as the science progresses and they are highly important as they are applied in the sensitive domain of biomedical science. For those reasons the purpose of this diploma thesis is the use of blockchain technology as a notary service, so the provider of medical knowledge is bounded to the information he provides. In order to achieve this goal, a decentralized application was developed that is running on the Ethereum platform and uses the tools supporting it. The application can be adapted to support any biomedical database. In this diploma thesis the platforms that were tested are these of CARRE and PubMed. More specifically the application that is developed provides to the users the possibility to submit a query, in one of the two biomedical databases, and subsequently to receive the information that they seek along with a receipt that the data send to the user cannot be altered and that the biomedical database cannot deny providing them. In other words a cryptographically secure hash value is calculated, which is relative both to the query and its corresponding answer, and which subsequently will be stored unaltered in to Ethereum blockchain along with a timestamp. So the user of this decentralized application can prove that at a specific time in the past, after he deployed a specific question, he received the specific corresponding answer.

**Keywords:** Blockchain, Ethereum, Smart Contracts, Decentralized applications, Biomedical repositories, Cryptographic techniques, Integrity, Non-repudiation

## ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

### ΚΕΦΑΛΑΙΟ 1

<b>Εισαγωγή</b>	<b>8</b>
1.1 Σκοπός της διπλωματικής εργασίας	9
1.2 Διάρθρωση της διπλωματικής εργασίας	12

### ΚΕΦΑΛΑΙΟ 2

<b>Υπόβαθρο (βασικές έννοιες)</b>	<b>13</b>
<b>2.1 Η τεχνολογία Blockchain</b>	<b>13</b>
2.1.1 Γενική δομή της τεχνολογίας Blockchain	13
2.1.2 Ασφάλεια – Έλεγχος που παρέχεται στο Blockchain	13
2.1.3 Πλεονεκτήματα της blockchain τεχνολογίας	14
<b>2.2 Bitcoin: Πρώτη εφαρμογή της τεχνολογίας Blockchain</b>	<b>14</b>
2.2.1 Η ανάγκη για τη δημιουργία του	14
2.2.2 Συναλλαγές στο Bitcoin	14
2.2.3 Χρονοσήμανση των συναλλαγών	15
2.2.4 Απόδειξη εργασίας	15
2.2.5 Μορφή του δικτύου	15
2.2.6 Κίνητρα – Ανταμοιβή	16
2.2.7 Απλοποίηση- Επαλήθευση	16
2.2.8 Τα είδη των συναλλαγών	17
2.2.9 Μυστικότητα στο δίκτυό του	17
2.2.10 Οι υπολογισμοί που κατοχυρώνουν την αξιοπιστία του συστήματος	17
2.2.11 Περιληπτική περιγραφή του δικτύου	17
<b>2.3 Εφαρμογή του Blockchain</b>	<b>18</b>
2.3.1 Ακεραιότητα και μη αποποίηση	18
2.3.2 Τομείς αξιοποίησης της τεχνολογίας Blockchain	21
2.3.3 Η τεχνολογία Blockchain σήμερα	22
2.3.4 Αποτελέσματα από την χρήση της τεχνολογίας Blockchain	23
<b>2.4 Ethereum Blockchain</b>	<b>23</b>
2.4.1 Δημιουργία	24
2.4.2 Δομή	24
2.4.3 Λογαριασμοί	24



2.4.4	Τα DAO	24
2.4.5	Προμήθεια	25
2.4.6	Το σχίσμα (Hard Fork)	25
2.4.7	Δυνατότητες από τη χρήση του Ethereum	26
2.4.8	Smart contracts	26
2.4.9	Επεξήγηση των εννοιών ABI-API	28
2.4.10	Κλιμάκωση (Scalability)	28
<b>2.5</b>	<b>Proof of Work ή Proof of Stake</b>	<b>29</b>
2.5.1	Proof of Work (POW)	29
2.5.2	Proof of Stake (POS)	30
2.5.3	Σύγκριση των δύο τεχνικών	30
<b>2.6</b>	<b>Hyperledger Fabric: ένα permissioned blockchain</b>	<b>31</b>

## **ΚΕΦΑΛΑΙΟ 3**

<b>Περιγραφή της εφαρμογής</b>	<b>35</b>
3.1 Καθορισμός του προβλήματος που διευθετείται και των απαιτήσεων που εξασφαλίζονται	35
3.2 Περιγραφή της αρχιτεκτονικής της εφαρμογής	36
3.3 Περιγραφή της λογικής των έξυπνων συμβολαίων που χρησιμοποιεί η εφαρμογή	37
3.4 Περιγραφή της ροής εργασίας	38
3.5 Αναλυτική παρουσίαση του κώδικα των συμβολαίων	43
3.6 Εισαγωγή των συμβολαίων στο Ethereum Ropsten Testnet	50
3.7 Οικονομικά στοιχεία από την αλληλεπίδραση με το Ethereum Ropsten Testnet	54
3.8 Παρουσίαση της λειτουργίας της εφαρμογής	58
3.8.1 Η αρχική σελίδα της εφαρμογής	58
3.8.2 Εισαγωγή ενός ερωτήματος στη σελίδα της εφαρμογής	59
3.8.3 Επικύρωση δεδομένων	63
3.8.3.1 Χρήση της εφαρμογής για επικύρωση δεδομένων	63
3.8.3.2 Επαλήθευση δεδομένων από το χρήστη χωρίς τη χρήση της εφαρμογής	65

## **ΚΕΦΑΛΑΙΟ 4**

<b>Εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν στη διπλωματική</b>	<b>67</b>
4.1 Ethereum blockchain	67

4.1.1	Geth	67
4.1.2	Truffle Framework	67
4.1.3	Ganache CLI	69
4.1.4	Solidity	69
4.1.5	Web3	71
4.1.6	Truffle-contract	71
<b>4.2</b>	<b>Κατασκευή της σελίδας που αλληλεπιδρά ο χρήστης</b>	<b>71</b>
4.2.1	jQuery	72
<b>4.3</b>	<b>ExpressJS</b>	<b>72</b>
<b>4.4</b>	<b>Javascript</b>	<b>72</b>
<b>4.5</b>	<b>Επιπλέον βιβλιοθήκες</b>	<b>73</b>
<b>4.6</b>	<b>MongoDB</b>	<b>73</b>
 <b>ΚΕΦΑΛΑΙΟ 5</b>		
	<b>Επίλογος</b>	<b>77</b>
5.1	Συμπεράσματα	77
5.2	Πιθανές επεκτάσεις της διπλωματικής	78
 <b>ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ</b>		<b>80</b>
 <b>ΠΑΡΑΡΤΗΜΑ Ι : Υπολογισμός μηδενικών bytes</b>		<b>84</b>
 <b>ΠΑΡΑΡΤΗΜΑ ΙΙ : Αρχεία κώδικα, εκτός από αυτά των έξυπνων συμβολαίων και τρόπος εγκατάστασης της εφαρμογής</b>		<b>85</b>

## ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

<b>Εικόνα 2.1.</b>	Σχηματική αναπαράσταση του δικτύου Blockchain.....	16
<b>Εικόνα 2.2.</b>	Απλοποιημένο σχήμα του Blockchain δικτύου του Bitcoin.....	18
<b>Εικόνα 2.3.</b>	Fabric Composer Positioning .....	32
<b>Εικόνα 2.4</b>	Αρχιτεκτονική του Hyperledger Composer.....	33
<b>Εικόνα 3.1.</b>	Αρχιτεκτονική της εφαρμογής.....	37
<b>Εικόνα 3.2</b>	Η ροή εργασίας για την υποβολή ενός ερωτήματος όπου επιλέχθηκε το βασικό συμβόλαιο .....	42
<b>Εικόνα 3.3.</b>	Η ροή εργασίας για την υποβολή ενός ερωτήματος όπου επιλέχθηκε το ενισχυμένο συμβόλαιο .....	43
<b>Εικόνα 3.4</b>	Αποτελέσματα από την εκτέλεση της εντολής "truffle migrate – network ropsten .....	51
<b>Εικόνα 3.5.</b>	Η απόδειξη που παράχθηκε από την τοποθέτηση του συμβολαίου Migrations.sol .....	52
<b>Εικόνα 3.6.</b>	Η απόδειξη που παράχθηκε από την τοποθέτηση του συμβολαίου Project.sol .....	53
<b>Εικόνα 3.7.</b>	Η απόδειξη που παράχθηκε από την τοποθέτηση του συμβολαίου Factoryproject.sol.....	53
<b>Εικόνα 3.8</b>	Αλληλεπίδραση σχετικά με το βασικό συμβόλαιο .....	57
<b>Εικόνα 3.9.</b>	Αλληλεπίδραση σχετικά με το ενισχυμένο συμβόλαιο .....	58
<b>Εικόνα 3.10.</b>	Η αρχική σελίδα της εφαρμογής .....	58
<b>Εικόνα 3.11</b>	Πριν ο χρήστης πατήσει Submit .....	60
<b>Εικόνα 3.12</b>	Η μορφή της σελίδας μετά την υποβολή του ερωτήματος στη βάση PubMed και ό,τι εμφανίζεται στην consola και χρησιμοποιείται για τον έλεγχο της ορθής λειτουργίας της εφαρμογής .....	60
<b>Εικόνα 3.13</b>	Η μορφή της σελίδας μετά τη δεύτερη υποβολή του ίδιου ερωτήματος στη βάση PubMed και ό,τι εμφανίζεται στην consola..	61
<b>Εικόνα 3.14</b>	Πριν ο χρήστης πατήσει Submit .....	61
<b>Εικόνα 3.15</b>	Η μορφή της σελίδας μετά την υποβολή του ερωτήματος στη βάση CARRE και ό,τι εμφανίζεται στην consola και χρησιμοποιείται για τον έλεγχο της ορθής λειτουργίας της εφαρμογής .....	62
<b>Εικόνα 3.16</b>	Πριν ο χρήστης πατήσει το Submit .....	62
<b>Εικόνα 3.17</b>	Η μορφή της σελίδας μετά την υποβολή του ερωτήματος στη βάση CARRE και ό,τι εμφανίζεται στην consola και χρησιμοποιείται για τον έλεγχο της ορθής λειτουργίας της εφαρμογής .....	63
<b>Εικόνα 3.18</b>	Η απάντηση μετά την υποβολή του ερωτήματος από τον χρήστη για να πάρει την πιο πρόσφατη απάντηση .....	64
<b>Εικόνα 3.19</b>	Μετά την υποβολή των στοιχείων, ώστε ο χρήστης να μάθει το	

<b>Εικόνα 3.20</b>	χρονικό διάστημα ισχύος μιας συγκεκριμένης απάντησης ..... Μετά την υποβολή των στοιχείων, ώστε ο χρήστης να λάβει τις τιμές των hash όλων των απαντήσεων που αφορούν ένα συγκεκριμένο ερώτημα .....	64 65
<b>Εικόνα 3.21</b>	Μετά την υποβολή των στοιχείων, ώστε ο χρήστης να λάβει την τιμή του hash που είναι αποθηκευμένη στο Personal contract που δημιουργήθηκε για το ερώτημά του.....	65
<b>Εικόνα 3.22</b>	Αλληλεπίδραση του χρήστη με το συμβόλαιο μέσα από ένα terminal που επικοινωνεί με ένα κόμβο .....	66

## ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

<b>Πίνακας 3.1</b>	Κόστος σε gas τοποθέτησης συμβολαίων στο blockchain	53
<b>Πίνακας 3.2</b>	Κόστος εισαγωγής σε σχέση με την τιμή hash	54
<b>Πίνακας 3.3</b>	Συγκεντρωτικός πίνακας κόστους συμβολαίων	55
<b>Πίνακας 4.1</b>	Καταχώρηση συλλογής “demo”	75
<b>Πίνακας 4.2</b>	Καταχώρηση συλλογής “demoPrivate	76

# ΚΕΦΑΛΑΙΟ 1

## Εισαγωγή

### 1.1 Σκοπός της διπλωματικής εργασίας

Η βιοϊατρική έρευνα και οι κλινικές δοκιμές βασίζονται σε μεγάλο βαθμό στα επίσημα δεδομένα που συγκεντρώνονται στις ιατρικές βάσεις δεδομένων [1]. Κάποια παραδείγματα τέτοιων βάσεων αποτελούν οι ιατρικές βάσεις που κατέχουν κλινικά δεδομένα για ομάδες ασθενών [2] οι βιοϊατρικές βάσεις [3] με δεδομένα φαρμακευτικά [4], σχετικά με την έρευνα των μεταβολιτών [5] και της κληρονομικότητας. Μερικά ακόμα είναι τα αποθετήρια με δημοσιεύσεις άρθρων και ιατρικών αποδείξεων [6] τα οποία μπορεί να είναι είτε γενικού σκοπού, με κυριότερο το παράδειγμα της υπηρεσίας PubMed που παρέχεται από το National Library of Medicine που βρίσκεται στην Αμερική, είτε πιο υψηλής ποιότητας αποδείξεις που έχουν περάσει από έλεγχο, όπως αυτές που παρέχονται από τις αναφορές στο Cochrane Library.

Στις βιοϊατρικές αυτές βάσεις τα δεδομένα συνεχώς ενημερώνονται και προστίθενται καινούριες πληροφορίες και ομάδες δεδομένων. Οποιαδήποτε χρονική στιγμή κάποιος άνθρωπος (για παράδειγμα ένας γιατρός, ένας ερευνητής), ή κάποιο πρόγραμμα μπορεί να απευθυνθεί σε αυτές για να αποκτήσει ιατρική γνώση. Η γνώση αυτή είναι σημαντική καθώς μπορεί να βρει εφαρμογή στην ανάπτυξη μιας επιστημονικής έρευνας, όπως είναι οι μελέτες για τη κατασκευή ενός καινούριου φαρμάκου όπου είναι απαραίτητα δεδομένα από την επίδρασή του στους ανθρώπους. Επίσης μπορεί να είναι χρήσιμη σε γιατρούς ώστε να μπορούν να προσδιορίσουν κατάλληλα τον τρόπο θεραπείας ενός ασθενή αλλά και σε πολλούς άλλους τομείς όπου η ιατρική πληροφορία θα μπορούσε να φανεί χρήσιμη. Για τον λόγο αυτό είναι σημαντικό να δεσμεύεται ο πάροχος ιατρικής γνώσης για τα δεδομένα που παρέχει και να μπορεί ο παραλήπτης τους να αποδείξει ότι παρέλαβε μια δεδομένη χρονική στιγμή τα συγκεκριμένα στοιχεία. Με τον τρόπο αυτό εξασφαλίζεται ότι οι πληροφορίες στις βάσεις δεν θα μεταβάλλονται αυτεξούσια καθώς σε οποιοδήποτε χρόνο θα πρέπει να είναι υπεύθυνες για το περιεχόμενο και την ακεραιότητα των δεδομένων που έχουν αποθηκευμένα και παρέχουν σε όσους έχουν πρόσβαση σε αυτές.

Μια αξιόπιστη υπηρεσία που παρέχει τέτοιου είδους υψηλής σημασίας γνώση θα πρέπει να ικανοποιεί τουλάχιστον τις δύο πολύ σημαντικές απαιτήσεις που είναι η ακεραιότητα των δεδομένων και η δυνατότητα μη αποποίησης παροχής της πληροφορίας. Με τον όρο ακεραιότητα εννοείται ότι το ερώτημα που τέθηκε και τα αντίστοιχα δεδομένα που παραλήφθηκαν σχετικά με αυτό δεν μπορούν να μεταβληθούν, είτε κατά λάθος είτε οικειοθελώς, μετά την διαδικασία παράδοσής τους. Επίσης με τον όρο μη αποποίηση εννοείται ότι για οποιαδήποτε διαδικασία ανάκτησης πληροφορίας η υπηρεσία δεν θα μπορεί να αρνηθεί ότι την παρείχε, μετά την υποβολή σε αυτήν ενός συγκεκριμένου ερωτήματος μια συγκεκριμένη χρονική στιγμή.

Επειδή τα παραπάνω ζητήματα είναι μείζονος σημασίας, στόχος της διπλωματικής εργασίας ήταν η μελέτη της τεχνολογίας blockchain που πετυχαίνει ανάμεσα σε άλλα τη δυνατότητα της ακεραιότητας της πληροφορίας που βρίσκεται στο

δίκτυό της και πώς μπορεί αυτή να συνδυαστεί με τις ψηφιακές υπογραφές ώστε να δημιουργηθεί μια εφαρμογή που θα ικανοποιεί τις δύο απαιτήσεις που περιγράφηκαν προηγουμένως, αυτών της ακεραιότητας και της μη αποποίησης.

Αρχικά επιλέχθηκε ο τύπος του δικτύου blockchain που κρίθηκε καταλληλότερος να υποστηρίξει την εφαρμογή.

Υπάρχουν τρεις τύποι δικτύων blockchain[7] :

► *Τα ιδιωτικά (private)*, στα οποία παρέχονται άδειες συμμετοχής. Έτσι κάποιος μπορεί να συμμετέχει μόνο αν προσκληθεί από τους διαχειριστές του δικτύου

► *Τα consortium*, στα οποία παρέχονται επίσης άδειες συμμετοχής, αλλά η διαχείριση του δικτύου δε γίνεται από μια μόνο εταιρεία. Το δίκτυο ελέγχεται από πολλές εταιρίες, καθεμία από τις οποίες ίσως να λειτουργεί έναν κόμβο του δικτύου. Σε αυτό επίσης οι διαχειριστές καθορίζουν τα δικαιώματα ανάγνωσης των συμμετεχόντων και προσδιορίζουν έναν περιορισμένο αριθμό κόμβων για να εφαρμόσει την διαδικασία της συναίνεσης. Το consortium blockchain συχνά θεωρείται ότι είναι ημι-αποκεντρωμένο.

► *Τα δημόσια ( public ) blockchain*, αποκαλούμενα επίσης permissionless, στα οποία δεν απαιτούνται άδειες και όλοι μπορούν να διαβάσουν να γράψουν και να συμμετέχουν στη διαδικασία της συναίνεσης για την επιβεβαίωση των έγκυρων συναλλαγών.

Ο τύπος του δικτύου blockchain που επιλέχθηκε για την εφαρμογή είναι ένα public blockchain, καθώς στόχος ήταν ο καθένας που χρησιμοποιεί την εφαρμογή να έχει πρόσβαση σε όλες τις συναλλαγές που έχουν γίνει και να μπορεί να διαβάσει λεπτομέρειες γι αυτές, όπως τότε εκτελέστηκαν και ποιους αφορά.

Μια από τις πλατφόρμες που εξετάστηκε είναι αυτή του Ethereum[8] που υποστηρίζει στο blockchain δίκτυό του την ύπαρξη αμετάβλητου, εκτελέσιμου κώδικα που περιέχεται στα έξυπνα συμβόλαια και έτσι αυτή παρέχει στους προγραμματιστές την ικανότητα ανάπτυξης αποκεντρωμένων εφαρμογών. Η πλατφόρμα αυτή είναι πολύ γνωστή και την εμπιστεύονται αρκετοί χρήστες γι' αυτό και επιλέχθηκε ώστε να σχεδιασθεί και να υποστηριχθεί από αυτό η εφαρμογή που υλοποιήθηκε σε αυτή τη διπλωματική.

Στη συνέχεια σχεδιάστηκε ο τρόπος λειτουργίας της εφαρμογής, η οποία μπορεί να περιγραφεί συνοπτικά ως εξής: Η εφαρμογή δίνει τη δυνατότητα στον χρήστη να θέσει ένα ερώτημα σε μια βάση, να πάρει την πληροφορία που επιθυμεί και μαζί μια απόδειξη ότι τα δεδομένα που έλαβε προήλθαν από τη συγκεκριμένη βάση, δεσμεύοντας αυτήν για τα στοιχεία που παρείχε. Πιο συγκεκριμένα τόσο από το ερώτημα που τέθηκε όσο και από την αντίστοιχη σε αυτό απάντηση, υπολογίζεται μέσω μιας συνάρτησης κατακερματισμού μια hash τιμή. Η τιμή αυτή είναι μοναδική για κάθε είσοδο στην συνάρτηση και έτσι διαφορετικές εισοδοί παράγουν διαφορετική τιμή hash, οπότε μπορεί να διατυπωθεί ότι μια hash τιμή αφορά μια συγκεκριμένη τιμή εισόδου. Επίσης η τιμή αυτή είναι κρυπτογραφικά ασφαλής, δηλαδή αν κάποιος γνωρίζει την έξοδο που υπολογίστηκε από τη συνάρτηση κατακερματισμού, δεν μπορεί να υπολογίσει την είσοδο, μόνο κατά τύχη δοκιμάζοντας διαφορετικές εισόδους έως ότου βρει την ίδια τιμή εξόδου. Κατά τη διάρκεια ανάπτυξης του κειμένου αυτής της διπλωματικής όπου συναντάτε ο όρος hash τιμή θα θεωρείται μια κρυπτογραφικά ασφαλής hash τιμή. Στη συνέχεια η τιμή που υπολογίστηκε αποθηκεύεται αναλλοίωτα μέσω των έξυπνων συμβολαίων που χρησιμοποιεί η εφαρμογή, στο δίκτυο του Ethereum blockchain μαζί με μια χρονοσφραγίδα. Για αυτή την αποθήκευση είναι απαραίτητο κάποιος να δημιουργήσει έναν λογαριασμό στο δίκτυο του Ethereum, δηλαδή να αποκτήσει ένα

ζευγάρι δημόσιου και ιδιωτικού κλειδιού ώστε να εκτελεί συναλλαγές με το δίκτυο. Το δημόσιο κλειδί συνδέεται μοναδικά με μια συγκεκριμένη αναπαράσταση 160 bit που αποτελεί μια διεύθυνση και η οποία είναι αυτή που φαίνεται ως αποστολέας μιας συναλλαγής. Συνεπώς μια έμπιστη αρχή πιστοποίησης θα μπορούσε να χρησιμοποιηθεί για να πιστοποιηθεί ότι το συγκεκριμένο δημόσιο κλειδί που χρησιμοποιείται στο δίκτυο του blockchain, και άρα και η διεύθυνση και το αντίστοιχο ιδιωτικό κλειδί που συνδέονται με αυτό, ανήκουν στον συγκεκριμένο ιδιοκτήτη της εφαρμογής και έτσι να συνδεθεί η ταυτότητα αυτού του ιδιοκτήτη με τις συναλλαγές που εκτελούνται από τη συγκεκριμένη διεύθυνση. Τέλος το πακέτο δεδομένων με τις απαραίτητες πληροφορίες επιστρέφεται στον χρήστη και αυτό μπορεί να είναι ψηφιακά υπογεγραμμένο από τον ιδιοκτήτη της εφαρμογής ώστε να εξασφαλιστεί ή μη αποποίηση προέλευσής τους.

Η λειτουργικότητα που παρέχει η εφαρμογή είναι αρκετά σημαντική, καθώς κάποιος χρήστης της μπορεί να αποδείξει ότι μια συγκεκριμένη χρονική στιγμή, μετά την υποβολή ενός συγκεκριμένου ερωτήματος, έλαβε την αντίστοιχη απάντηση και έτσι να ικανοποιηθούν οι απαιτήσεις της ακεραιότητας και της μη αποποίησης. Επιπλέον ένα τμήμα της εφαρμογής παρέχει την δυνατότητα ελέγχου των διαφορετικών απαντήσεων που έχουν δοθεί για μια συγκεκριμένη ερώτηση.

Για το στάδιο ανάπτυξης της εφαρμογής χρησιμοποιήθηκε ένα τοπικό blockchain που δημιουργήθηκε με το εργαλείο Ganache Cli. Σε αυτό το τοπικό δίκτυο οι χρόνοι αναμονής για την επιβεβαίωση των συναλλαγών καθορίζονται να είναι πολύ μικροί έως και μηδενικοί και δίνονται αρκετά δωρεάν ether για την αλληλεπίδραση με το δίκτυο. Οι ιδιότητες αυτές το κατέστησαν το καλύτερο περιβάλλον για να δοκιμαστεί η εφαρμογή μέχρι αυτή να πάρει την τελική της μορφή.

Στη συνέχεια η εφαρμογή έπρεπε να ελεγχθεί σε ένα δημόσιο Ethereum blockchain. Από τα υπάρχοντα δύο είδη, τα Ethereum testnets, τα οποία είναι blockchain δίκτυα όπου δεν χρησιμοποιούνται πραγματικά ether άρα δεν υπάρχει πραγματικό χρηματικό κόστος και το Ethereum Mainnet, όπου για τις συναλλαγές χρησιμοποιούνται πραγματικά ether, επιλέχθηκε η εφαρμογή να ελεγχθεί σε ένα Ethereum testnet. Ο λόγος που δεν επιλέχθηκε το Ethereum Mainnet είναι διότι σε αυτό θα υπήρχαν πραγματικές χρεώσεις, μεγάλοι χρόνοι αναμονής και επίσης θα τοποθετούνταν στο δίκτυο του blockchain δεδομένα που δεν είναι χρήσιμα, καθώς η εφαρμογή δεν χρησιμοποιείται από κάποια υπηρεσία και σκοπός αυτής της διπλωματικής ήταν η επίδειξη της λειτουργικότητάς της. Υπάρχουν τρία είδη Ethereum testnet [9]: Ropsten, Kovan και Rinkeby. Τα δύο τελευταία στηρίζονται στον τρόπο συναίνεσης Proof of Authority (PoA)[10] κάτι που κάνει τα δίκτυα αυτά πιο σταθερά σε σχέση με το Ropsten δίκτυο που χρησιμοποιεί Proof of Work (PoW). Επειδή όμως προς το παρόν το Ethereum Mainnet χρησιμοποιεί PoW για την επιβεβαίωση των έγκυρων συναλλαγών για το λόγο αυτό επιλέχθηκε το Ethereum Ropsten testnet και σε αυτό τοποθετήθηκαν τα συμβόλαια που χρησιμοποιεί η εφαρμογή και επιβεβαιώθηκε η ομαλή λειτουργία της. Εφόσον η εφαρμογή λειτούργησε με τον επιθυμητό τρόπο στο testnet με τον ίδιο τρόπο θα πρέπει να μπορεί να λειτουργήσει και στο Ethereum Mainnet.

Οι δύο ιατρικές βάσεις που χρησιμοποιήθηκαν για την παροχή της ιατρικής γνώσης είναι αυτή του PubMed Medline Database [11] και του CARRE Risk Factor Repository [12]. Οι βάσεις αυτές είναι ενδεικτικές και η εφαρμογή θα μπορούσε με εύκολα να επεκταθεί για να υποστηρίξει και άλλες βάσεις δεδομένων.



Τα αποτελέσματα από την τοποθέτηση των έξυπνων συμβολαίων στο Ethereum Ropsten testnet επιβεβαίωσαν την ομαλή λειτουργία της εφαρμογής. Επειδή οι κύριες δυνατότητες που παρέχει η εφαρμογή για ακεραιότητα και μη αποποίηση των δεδομένων είναι πολύ σημαντικές, αργότερα αυτή θα μπορούσε να προσαρμοστεί στην λειτουργία μιας οποιασδήποτε βάσης δεδομένων που παρέχει κρίσιμες πληροφορίες σε χρήστες και να τροποποιηθεί ώστε να επικοινωνεί με το Ethereum Mainnet.

## 1.2 Διάρθρωση της διπλωματικής εργασίας

Το κείμενο της διπλωματικής εργασίας αποτελείται από 5 κεφάλαια, τη Βιβλιογραφία και δύο Παραρτήματα.

Το πρώτο κεφάλαιο αποτελεί την εισαγωγή όπου προσδιορίζεται το αντικείμενο της διπλωματικής εργασίας.

Στο κεφάλαιο 2 αναλύονται οι βασικές έννοιες που θα χρησιμοποιηθούν κατά την ανάπτυξη του κειμένου και πιο συγκεκριμένα η τεχνολογία blockchain, η δομή της στην πρώτη εφαρμογή της στο Bitcoin, το Ethereum blockchain στο οποίο στηρίζεται η εφαρμογή που δημιουργήθηκε σε αυτή τη διπλωματική, η διαφορά στις έννοιες Proof of Work και Proof of Stake και τέλος δίνεται ένα παράδειγμα ενός permissioned blockchain.

Στο κεφάλαιο 3 περιγράφεται λεπτομερώς η εφαρμογή που υλοποιήθηκε σε αυτή τη διπλωματική. Για το σκοπό αυτό δίνεται η ροή εργασίας της, τα έξυπνα συμβόλαια που αυτή χρησιμοποιεί και η τοποθέτησή τους στο Ethereum Ropsten Testnet. Επίσης δίνονται οικονομικά στοιχεία από την αλληλεπίδραση με τα συμβόλαια και παρουσιάζεται ένα παράδειγμα από τη λειτουργία της εφαρμογής.

Στο κεφάλαιο 4 αναφέρονται τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής και αφορούν όλα τα μέρη που την απαρτίζουν, δηλαδή την επικοινωνία με το δίκτυο blockchain, το front-end που εκτελείται στο περιβάλλον του χρήστη, το backend για την ανάπτυξη της του server και την αρμονική λειτουργία με την τοπική βάση MongoDB.

Στο κεφάλαιο 5 δίνονται συνοπτικά τα συμπεράσματα από τη χρήση της εφαρμογής και οι πιθανές επεκτάσεις της.

Στο τέλος παρατίθενται οι βιβλιογραφικές αναφορές και ακολουθούν δύο παραρτήματα. Το πρώτο σχετίζεται με τον τρόπο υπολογισμού των μηδενικών byte και στο δεύτερο παρουσιάζονται τα υπόλοιπα τμήματα κώδικα, πλην των έξυπνων συμβολαίων, τα οποία χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής και περιγράφεται ο τρόπος εγκατάστασής της.

## ΚΕΦΑΛΑΙΟ 2

### Υπόβαθρο (βασικές έννοιες)

#### 2.1 Η τεχνολογία Blockchain

Η πρώτη εργασία [7] στην οποία περιγράφηκε μια κρυπτογραφικά ασφαλή αλυσίδα ήταν αυτή των Stuart Haber και W. Scott Stornetta το 1991[1]. Σκοπός τους ήταν να δημιουργήσουν ένα σύστημα όπου η χρονική σήμανση των αρχείων δεν θα μπορούσε να αλλοιωθεί ή να αλλάξει σε μια παλιότερη ημερομηνία. Αργότερα το 1992 συνεργάστηκε μαζί τους ο Dave Bayer και στο αρχικό σχέδιο ενσωμάτωσαν τα Merkle trees, η χρήση των οποίων αύξησε την απόδοση καθώς επέτρεπε να συγκεντρωθούν πολλά αρχεία σε ένα block [13]. Όμως η πρώτη σύλληψη της ιδέας του blockchain περιγράφηκε στο άρθρο με τίτλο "*Bitcoin: A Peer-to-Peer Electronic Cash System*" [13], που δημοσιεύθηκε στο newsletter «The Cryptography Mailing List» στις 30 Οκτωβρίου 2008, υπογεγραμμένο με το ψευδώνυμο Satoshi Nakamoto, που ανακοίνωνε και εξηγούσε την τεχνολογία blockchain. Ένα χρόνο αργότερα, η τεχνολογία αυτή, εφαρμόστηκε ως εργαλείο οικονομικής ανταλλαγής για το Bitcoin.

##### 2.1.1 Γενική δομή της τεχνολογίας Blockchain

Το Blockchain μπορεί να οριστεί ως μία σειρά καταχωρίσεων που αφορούν συναλλαγές, σε ένα δημόσιο κατάστιχο (ledger), όπου κάθε νέα ομάδα καταχωρίσεων (block) συνδέεται μονοσήμαντα με τις προηγούμενες, δημιουργώντας μία αλυσίδα (chain), με τρόπο που εξασφαλίζει την ασφάλεια και διαφάνεια των συναλλαγών, ενώ ταυτόχρονα προστατεύει την ανωνυμία των συμμετεχόντων (συναλλασσόμενων) [15].

Η βάση δεδομένων blockchain δεν αποθηκεύεται σε καμία θέση, πράγμα που σημαίνει ότι τα αρχεία που διατηρεί είναι πραγματικά δημόσια και εύκολα επαληθεύσιμα. Δεν υπάρχει κεντρική έκδοση των πληροφοριών αυτών που μπορούν να εκμεταλλευτούν οι χάκερ υπολογιστών. Φιλοξενείται από εκατομμύρια υπολογιστές ταυτόχρονα και τα δεδομένα του είναι προσβάσιμα σε οποιονδήποτε στο Διαδίκτυο.

##### 2.1.2 Ασφάλεια – Έλεγχος που παρέχεται στο Blockchain

Οι μέθοδοι ασφαλείας στο Blockchain χρησιμοποιούν την τεχνολογία κρυπτογράφησης. Η βάση για αυτό είναι τα λεγόμενα δημόσια και ιδιωτικά "κλειδιά". Από ένα δημόσιο κλειδί (μια μακρά, τυχαία δημιουργημένη σειρά αριθμών) μπορεί να παραχθεί μια μοναδική και συγκεκριμένη διεύθυνση χρηστών στο blockchain και το ιδιωτικό κλειδί είναι σαν ένας κωδικός πρόσβασης που δίνει στον ιδιοκτήτη του πρόσβαση σε ψηφιακά στοιχεία.

Τα blockchains επιτρέπουν πολλά διαφορετικά επίπεδα ελέγχου. Μπορεί να έχουν αυστηρούς κανόνες πρόσβασης, που ελέγχονται από μια κεντρική αρχή (permissioned) ή να είναι εντελώς ανοιχτά σε οποιονδήποτε (permissionless). Για παράδειγμα, το Bitcoin χρησιμοποιεί ένα permissionless blockchain [16].

### 2.1.3 Πλεονεκτήματα της blockchain τεχνολογίας

Η τεχνολογία blockchain προσφέρει τα παρακάτω πλεονεκτήματα:

- *Μείωση κόστους και επιτάχυνση διαδικασιών* με τον περιορισμό ή κατάργηση των μεσαζόντων και την αυτοματοποίηση πολλών σχετικών διαδικασιών.
- *Ασφάλεια* με την πολύπλοκη κρυπτογράφηση που χρησιμοποιείται και την τήρηση του μητρώου σε πολλούς διαφορετικούς υπολογιστές.
- *Ιχνηλασιμότητα* με την τήρηση στο μητρώο όλου του ιστορικού ιδιοκτησίας οποιουδήποτε περιουσιακού στοιχείου (bitcoin, επιταγές, χρυσός, ακίνητη περιουσία ή οτιδήποτε άλλο)

## 2.2 Bitcoin: Πρώτη εφαρμογή της τεχνολογίας Blockchain

Η τεχνολογία blockchain σχεδιάστηκε αρχικά ως εργαλείο οικονομικής ανταλλαγής για το Bitcoin, όπως παρουσιάστηκε από τον Satoshi Nakamoto στο άρθρο που έφερε την υπογραφή του, με τίτλο " *Bitcoin: A Peer-to-Peer Electronic Cash System*" [13]. Πρόκειται για ένα ηλεκτρονικό σύστημα μετρητών που αποτελείται από ομότιμους χρήστες και σύμφωνα με το άρθρο παράγονται οι εξής πληροφορίες για αυτό οι οποίες περιγράφονται στα επόμενα υποκεφάλαια.

### 2.2.1 Η ανάγκη για τη δημιουργία του

Η ύπαρξη μεσάζοντος στις ηλεκτρονικές συναλλαγές, το κόστος διαμεσολάβησης που αυξάνει την αξία του προϊόντος και κόβει τη δυνατότητα για μικρές περιστασιακές συναλλαγές, η έλλειψη εμπιστοσύνης που δημιουργεί αβεβαιότητα, η απώλεια της ικανότητας να γίνονται μη αντιστρέψιμες πληρωμές για μη αντιστρέψιμες υπηρεσίες οδήγησαν στη δημιουργία μιας peer-to-peer αποκεντρωμένης ηλεκτρονικής μορφής χρήματος που βασίζεται πάνω στις αρχές της κρυπτογραφίας για την διασφάλιση του δικτύου και την επαλήθευση των συναλλαγών. Είναι ένα σύστημα ηλεκτρονικών πληρωμών με βάση την κρυπτογραφική απόδειξη αντί της εμπιστοσύνης, που επιτρέπει οποιαδήποτε δύο μέρη να συναλλάσσονται απευθείας μεταξύ τους, χωρίς την ανάγκη ύπαρξης ενός αξιόπιστου τρίτου. Οι συναλλαγές που είναι υπολογιστικά ανέφικτες να αντιστραφούν προστατεύουν τους πωλητές από την απάτη και οι μηχανισμοί μεσεγγύησης μπορούν εύκολα να εφαρμοστούν για την προστασία των αγοραστών. Τα Bitcoins είναι ψηφιακά νομίσματα τα οποία δεν έχουν εκδοθεί από κάποια κυβέρνηση ή τράπεζα ή από κάποια οργάνωση, βασίζονται εξ' ολοκλήρου σε πρωτόκολλα κρυπτογράφησης και σε ένα ειδικά καταμεμημένο δίκτυο από συγκεκριμένους χρήστες, στο οποίο υπάρχει η δυνατότητα αποθήκευσης και μεταφοράς χρημάτων. Επιπλέον η διασφάλιση μη ύπαρξης διπλής δαπάνης στηρίζεται σε χρονοσφραγίδες που υποδεικνύουν ποια συναλλαγή προηγείται.

### 2.2.2 Συναλλαγές στο Bitcoin

Το ηλεκτρονικό νόμισμα είναι μια αλυσίδα ψηφιακών υπογραφών. Η μεταβίβασή του γίνεται από τον ιδιοκτήτη με την προσθήκη στο τέλος του νομίσματος της ψηφιακής υπογραφής, του hash της προηγούμενης συναλλαγής, του δημόσιου κλειδιού του παραλήπτη και τη χρήση ψηφίων τυχαιοποίησης (*nonce*). Για να αποφευχθεί ότι ο αποστολέας στέλνει το ίδιο νόμισμα σε δυο διαφορετικούς παραλήπτες, κάθε συναλλαγή γίνεται δημόσια γνωστή και αποδεκτή γίνεται εκείνη που πραγματοποιήθηκε πρώτη,

σύμφωνα με τη χρονική σειρά που συναίνεσαν και αποδέχτηκαν οι συμμετέχοντες (κόμβοι).

### 2.2.3 Χρονοσήμανση των συναλλαγών

Η χρονική ακολουθία των συναλλαγών αποδεικνύεται με τη βοήθεια ενός σέρβερ χρονοσφραγίδων. Τα δεδομένα τα οποία θέλουμε να πιστοποιήσουμε τότε παράχθηκαν οργανώνονται σε ένα μπλοκ, το οποίο θα συμμετέχει στη δημιουργία ενός hash, όπου και τίθεται η χρονοσφραγίδα. Στη συνέχεια το hash αυτό γίνεται ευρέως γνωστό. Για την χρονική σήμανση των επόμενων δεδομένων, που οργανώνονται επίσης σε μπλοκ, χρησιμοποιούμε το hash από την προηγούμενη χρονοσφραγίδα και το νέο μπλοκ δεδομένων και παράγουμε ένα νέο hash, το οποίο γίνεται γνωστό και η διαδικασία επαναλαμβάνεται. Με αυτόν τον τρόπο δημιουργείται μια αλυσίδα που λειτουργεί σαν ένας δημόσιος, συλλογικός μηχανισμός χρονοσήμανσης χωρίς κεντρική αρχή.

### 2.2.4 Απόδειξη εργασίας

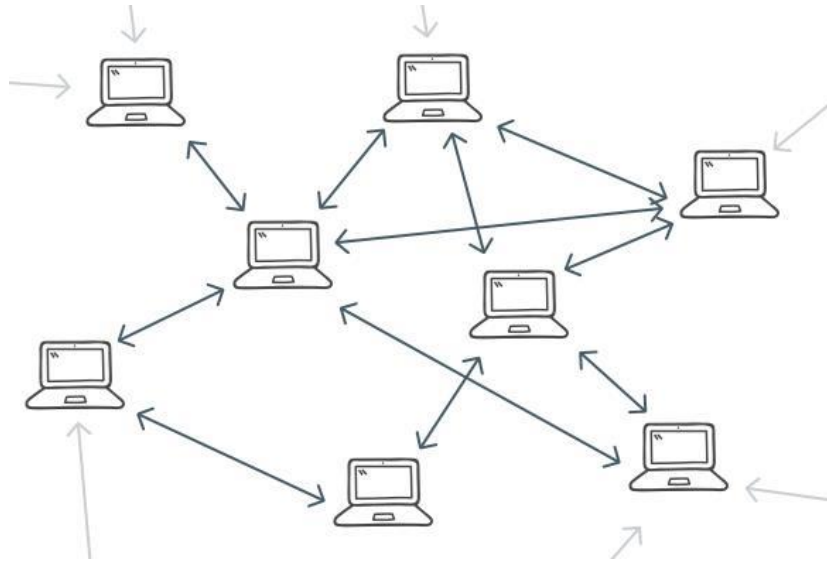
Κομβικό ρόλο στη διαδικασία παίζει η απόδειξη της εργασίας: για να είναι έγκυρο το παραγόμενο hash θα πρέπει να έχει συγκεκριμένη μορφή, με ένα συγκεκριμένο πλήθος αρχικών μηδενικών. Η αλλαγή ενός μπλοκ απαιτεί την αναγέννηση όλων των διαδόχων και την επαναφορά του έργου που περιέχουν. Αυτό προστατεύει την αλυσίδα μπλοκ από παραβίαση.

Η απόδειξη εργασίας είναι καθοριστική για την επιλογή της κοινά αποδεκτής αλυσίδας, αφού πάνω σ' αυτήν βασίζεται η κοινή συναίνεση των κόμβων για τη χρονολογική σειρά των συναλλαγών. Η απόδειξη της εργασίας είναι ουσιαστικά μία CPU ψήφος.

### 2.2.5 Μορφή του δικτύου

Για την υλοποίηση του δικτύου:

- Νέες συναλλαγές μεταδίδονται σε όλους τους κόμβους.
- Κάθε κόμβος συλλέγει νέες συναλλαγές σε ένα μπλοκ και προσπαθεί να βρει μια δύσκολη απόδειξη εργασίας γι' αυτό.
- Όταν ένας κόμβος βρει μια απόδειξη εργασίας, μεταδίδει το μπλοκ σε όλους τους κόμβους, οι οποίοι το δέχονται μόνο εάν όλες οι συναλλαγές σε αυτό είναι έγκυρες και δεν έχουν ήδη δαπανηθεί.
- Οι κόμβοι εκφράζουν την αποδοχή του μπλοκ δουλεύοντας στη δημιουργία του επόμενου μπλοκ στην αλυσίδα, χρησιμοποιώντας το hash του αποδεκτού μπλοκ.
- Οι κόμβοι θεωρούν πάντα ότι η μεγαλύτερη αλυσίδα είναι η σωστή.
- Εάν κόμβος λάβει διαφορετικές εκδόσεις του επόμενου τμήματος, θεωρεί σωστή την πρώτη χρονικά, ωστόσο αποθηκεύει και τη δεύτερη μέχρι η διαδικασία φανερώσει τη μακρύτερη αλυσίδα.
- Οι κόμβοι μπορούν να εγκαταλείπουν και να επανεντάσσονται στο δίκτυο κατά βούληση, αποδεχόμενοι την αλυσίδα απόδειξης εργασίας ως απόδειξη του τι συνέβη όταν είχαν φύγει, και πραγματοποιούν τη λήψη του τελευταίου μπλοκ πριν από οποιαδήποτε συναλλαγή.



**Εικόνα 2.1.** Σχηματική αναπαράσταση του δικτύου Blockchain.

Πηγή: <https://jeiwan.cc/posts/building-blockchain-in-go-part-7/>

### 2.2.6 Κίνητρα – Ανταμοιβή

Κάθε κόμβος ανταμείβεται με νέα δημιουργημένα ψηφιακά νομίσματα για κάθε νέο μπλοκ που παράγει και με προμήθεια από όλες τις περιλαμβανόμενες συναλλαγές στο μπλοκ. Για να κερδίσει αυτήν την ανταμοιβή, ανταγωνίζεται στη λύση ενός δύσκολου μαθηματικού προβλήματος βασισμένο σε έναν αλγόριθμο κρυπτογραφικού κατακερματισμού. Η λύση σε αυτό το πρόβλημα, που ονομάζεται απόδειξη εργασίας (proof-of-work), περιλαμβάνεται στο νέο μπλοκ και λειτουργεί ως απόδειξη ότι έχει ξοδέψει σημαντική υπολογιστική προσπάθεια.

Με την πάροδο του χρόνου οι μαθηματικές εξισώσεις που πρέπει να επιλυθούν προκειμένου να λειτουργήσει το σύστημα στην ανταλλαγή ψηφιακού χρήματος γίνονται όλο και πιο δύσκολες, αφού η χρήση του Bitcoin αυξάνεται συνεχώς σε παγκόσμιο επίπεδο, γι' αυτό και η δημιουργία νέων νομισμάτων βασίζεται εξ' ολοκλήρου στο σύστημα του μοντέλου εξόρυξης χρυσού.

### 2.2.7 Απλοποίηση- Επαλήθευση

Για εξοικονόμηση χώρου στο σκληρό δίσκο του κόμβου, σε κάθε μπλοκ της αλυσίδας περιέχεται μία συνάθροιση όλων των συναλλαγών με τη χρήση ενός δέντρου Merkle.

Για να επαληθευτεί ότι μία συναλλαγή περιέχεται σε ένα μπλοκ, χωρίς να χρειαστεί να γίνει λήψη όλων των συναλλαγών σ' αυτό, χρησιμοποιείται μία διαδρομή πιστοποίησης ή αλλιώς διαδρομή Merkle. Έτσι τοποθετώντας τη συναλλαγή σε μια θέση στην αλυσίδα μπορεί να γνωρίζει, αν έχει γίνει αποδεκτή από το δίκτυο. Η επαλήθευση είναι αξιόπιστη μόνο εάν το δίκτυο δεν έχει υπερφορτωθεί από έναν εισβολέα. Μια στρατηγική για προστασία από αυτό θα ήταν να δεχτούν οι χρήστες ειδοποιήσεις από κόμβους δικτύου, όταν εντοπίζουν ένα μη έγκυρο μπλοκ, ζητώντας από το λογισμικό του χρήστη να πραγματοποιήσει λήψη του πλήρους μπλοκ και επιβεβαίωση των συναλλαγών.

### **2.2.8 Τα είδη των συναλλαγών**

Μια συναλλαγή μπορεί να έχει πολλαπλές εισόδους και εξόδους. Κανονικά υπάρχει μία μόνο είσοδος από μια μεγαλύτερη προηγούμενη συναλλαγή ή πολλαπλές εισοδοί που συνδυάζουν μικρότερες ποσότητες και το πολύ δύο εξοδοί : μία για την πληρωμή και μία επιστροφή, εάν υπάρχει, στον αποστολέα . Αξίζει να σημειωθεί ότι το fan-out, δεν αποτελεί πρόβλημα, αφού δεν υπάρχει ποτέ η ανάγκη ενός αντίγραφου του ιστορικού μιας συναλλαγής.

### **2.2.9 Μυστικότητα στο δίκτυό του**

Αν και κάθε συναλλαγή δημοσιοποιείται, ωστόσο δεν είναι εφικτή η πιστοποίηση του ατόμου που την εκτελεί εξαιτίας της ανωνυμίας των δημόσιων κλειδιών. Σε κάθε συναλλαγή χρησιμοποιείται ένα καινούριο ζευγάρι κλειδιών, γεγονός που λειτουργεί σαν τείχος προστασίας για τον ιδιοκτήτη. Πρόβλημα πιθανόν να υπάρξει σε συναλλαγές με πολλαπλές εισόδους, οι οποίες αναγκαστικά αποκαλύπτουν ότι οι εισροές τους ανήκαν στον ίδιο ιδιοκτήτη. Σε μία τέτοια περίπτωση η αποκάλυψη του κατόχου του κλειδιού μπορεί να οδηγήσει στην αποκάλυψη και των άλλων συναλλαγών που υλοποιήθηκαν από τον ίδιο.

### **2.2.10 Οι υπολογισμοί που κατοχυρώνουν την αξιοπιστία του συστήματος**

Ένας εισβολέας, για να ακυρώσει μια συναλλαγή, πρέπει να κατασκευάσει μια αλυσίδα με μπλοκ μεγαλύτερη από αυτή που περιέχει τη συναλλαγή, αφού γίνεται πάντα δεκτή από τους κόμβους η μακρύτερη αλυσίδα. Ο ρόλος του περιορίζεται στο να προσπαθήσει να αλλάξει μία από τις δικές του συναλλαγές, για να πάρει πίσω χρήματα που έδωσε πρόσφατα.

Η σχέση μεταξύ της κανονικής αλυσίδας και μιας αλυσίδας εισβολέα μπορεί να χαρακτηριστεί ως διωνυμικός περίπατος Random Walk. Επιτυχία είναι η κανονική αλυσίδα να επεκταθεί κατά ένα μπλοκ και το συμβάν αποτυχίας είναι η επέκταση της αλυσίδας του εισβολέα κατά ένα μπλοκ. Η πιθανότητα επιτυχίας ενός επιτιθέμενου είναι ανάλογη με εκείνη στο πρόβλημα καταστροφής παίκτη: μειώνεται εκθετικά, καθώς ο επιτιθέμενος οφείλει να καλύψει μεγαλύτερο αριθμό μπλοκ.

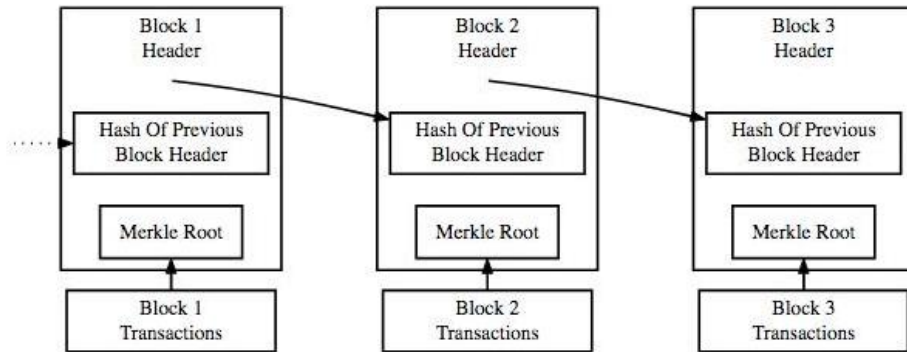
Όσον αφορά το χρονικό διάστημα που ο παραλήπτης μιας νέας συναλλαγής πρέπει να περιμένει για να είναι σίγουρος ότι ο αποστολέας δεν μπορεί να την αλλάξει, αποδεικνύεται εύκολα με τη βοήθεια της κατανομής Poisson ότι όσο μεγαλώνει ο αριθμός των μπλοκ που προηγείται η αλυσίδα, τόσο μικραίνει η πιθανότητα ο επιτιθέμενος να μπορέσει να δημιουργήσει μια μεγαλύτερη αλυσίδα μπλοκ, ακυρώνοντας έτσι τη συναλλαγή. Αυτό συμβαίνει γιατί, για λόγους ασφαλείας, ο αποδέκτης παράγει ένα ζεύγος κλειδιών και δίνει το δημόσιο κλειδί του στον αποστολέα λίγο πριν την υπογραφή, γεγονός που αποτρέπει τον αποστολέα από την προετοιμασία μιας αλυσίδας, ο οποίος και αρχίζει αμέσως μετά την αποστολή της συναλλαγής να εργάζεται μυστικά σε μια παράλληλη αλυσίδα που περιέχει μια εναλλακτική έκδοση της συναλλαγής του.

### **2.2.11 Περιληπτική περιγραφή του δικτύου**

Ένα peer to peer (P2P) δίκτυο με την απλή δομή του και την ανθεκτικότητά του εξασφαλίζει στις συναλλαγές την ανωνυμία και αποτρέπει τις διπλοπληρωμές.



Οι κόμβοι εργάζονται ταυτόχρονα με ελάχιστο συντονισμό. Δεν χρειάζεται να ταυτοποιηθούν και μπορούν να εγκαταλείπουν και να επανεντάσσονται στο δίκτυο κατά βούληση. Ψηφίζουν με την εξουσία της CPU, εκφράζουν την αποδοχή τους στα έγκυρα μπλοκ, δουλεύοντας στην επέκτασή τους και απορρίπτουν τα άκυρα. Όλοι οι απαραίτητοι κανόνες και τα κίνητρα μπορούν να ενισχυθούν με αυτόν τον μηχανισμό συναίνεσης.



Εικόνα 2.2. Απλοποιημένο σχήμα του Blockchain δικτύου του Bitcoin.

Πηγή: <https://bitcoin.org/en/developer-guide#block-chain-overview>

## 2.3 Εφαρμογή του Blockchain

### 2.3.1 Ακεραιότητα και μη αποποίηση

Ένα από τα κύρια ζητήματα που εξετάζεται στη χρήση μιας εφαρμογής είναι η ασφάλεια που αυτή παρέχει. Η ασφάλεια μεταξύ άλλων πρέπει να καλύπτει απαιτήσεις όπως η ακεραιότητα (integrity) και η μη αποποίηση (non-repudiation) και η τεχνολογία Blockchain μπορεί ενσωματωθεί στην λειτουργικότητα μιας εφαρμογής ώστε να παρέχονται αυτές οι ιδιότητες.

“Η ακεραιότητα αναφέρεται στη διατήρηση των δεδομένων ενός πληροφοριακού συστήματος σε μια γνωστή κατάσταση, χωρίς ανεπιθύμητες τροποποιήσεις, αφαιρέσεις ή προσθήκες από μη εξουσιοδοτημένα άτομα, καθώς και στην αποτροπή της πρόσβασης ή/και χρήσης των υπολογιστών και δικτύων του συστήματος από άτομα χωρίς άδεια. Συνεπώς, η ακρίβεια και η πληρότητα των πληροφοριών και των μεθόδων επεξεργασίας τους, διασφαλίζεται μόνο από εξουσιοδοτημένα προγράμματα, κάτω από ένα πλήρως ελεγχόμενο περιβάλλον.” [17]

Στα πληροφορικά συστήματα είναι σημαντικό να διατηρείται η συνεκτικότητα των δεδομένων και να εξασφαλίζεται η ακεραιότητά τους. Για το λόγο αυτό οι ιδιοκτήτες τέτοιων συστημάτων χρησιμοποιούν μεθόδους όπως η κρυπτογραφία και μηχανισμούς ψηφιακών υπογραφών.

Μια από τις μεθόδους κρυπτογραφίας που χρησιμοποιείται είναι η σύγκριση των hash τιμών. Πιο συγκεκριμένα μια hash τιμή μπορεί να παραχθεί από μια κρυπτογραφικά ασφαλή συνάρτηση κατακερματισμού που δέχεται ως είσοδο ένα δεδομένο τυχαίου μεγέθους και δίνει ως έξοδο μια τιμή σταθερού μεγέθους [18]. Διαφορετικοί είσοδοι δίνουν διαφορετική έξοδο και συνεπώς ακόμα και αν ένα bit αλλάξει στα δεδομένα εισόδου η συνάρτηση θα παράγει διαφορετική έξοδο. Η ιδιότητα αυτή είναι πολύ χρήσιμη για την ακεραιότητα της πληροφορίας καθώς αν από ένα σετ δεδομένων

παραχθεί μια hash τιμή οποιαδήποτε άλλη στιγμή αργότερα θα πρέπει το ίδιο σετ δεδομένων να παράγει την ίδια hash τιμή, αν τα δεδομένα δεν έχουν τροποποιηθεί. Αυτή η τεχνική της σύγκρισης hash τιμών για την εξασφάλιση της ακεραιότητας χρησιμοποιείται και στη τεχνολογία blockchain. Για παράδειγμα το Bitcoin χρησιμοποιεί τη συνάρτηση κατακερματισμού sha-256 ενώ το Ethereum την keccak-256. Έτσι το blockchain παρέχει ακεραιότητα των συναλλαγών που υπάρχουν στο δίκτυο του, καθώς οποιαδήποτε συναλλαγή καταγράφεται δεν μπορεί να διαγραφεί ή να τροποποιηθεί. Επίσης αυτό θα μπορούσε να χρησιμοποιηθεί από μια εφαρμογή για την αθέμιτη αποθήκευση μιας hash τιμής η οποία οποιαδήποτε στιγμή θα μπορούσε να συγκριθεί με έναν μετέπειτα υπολογισμό της hash τιμή των ίδιων δεδομένων ώστε να διαπιστωθεί αν αυτά έχουν αλλοιωθεί.

Εξίσου σημαντικό εκτός από την ακεραιότητα είναι η μη αποποίηση και με τον όρο μη αποποίηση εννοείται “ότι, κανένας από τους συναλλασσόμενους δεν έχει τη δυνατότητα να αρνηθεί τη συμμετοχή του εκ των υστέρων σε μια ψηφιακή συναλλαγή. Αντιθέτως, η πράξη που έγινε είναι νομικά δεσμευτική και μπορεί να αποδειχθεί στο δικαστήριο. Οι υπηρεσίες μη αποποίησης ευθύνης πρέπει, αν χρειαστεί, να μπορούν να αποδείξουν την προέλευση, τη μεταφορά/μετάδοση και την παράδοση των δεδομένων.” [17].

Υπάρχουν τέσσερα είδη [19] μη αποποίησης όπως έχουν καθοριστεί από το ISO [20]

1. *Non-repudiation of origin (μη αποποίηση προέλευσης)*: Ο αποστολέας δεν μπορεί να αρνηθεί ότι έστειλε ένα μήνυμα. Για παράδειγμα όταν παρέχεται μη αποποίηση προέλευσης ο ιδιοκτήτης ενός ηλεκτρονικού καταστήματος να μπορεί να αποδείξει σε κάποιον τρίτο, πχ σε ένα δικαστή, ότι ένας συγκεκριμένος πελάτης έκανε αίτηση για να αγοράσει κάποια είδη από το κατάστημα, ώστε ο πελάτης να μην μπορεί να αρνηθεί ότι τα ζήτησε και να μη δεχτεί να τα πληρώσει. Στον μη ψηφιακό κόσμο αυτό μπορεί επιτευχθεί με μια ιδιόχειρη υπογραφή από τον πελάτη.
2. *Non-repudiation of submission (μη αποποίηση υποβολής)*: Παρέχεται στον αποστολέα ενός μηνύματος μια απόδειξη ότι το μήνυμά του έχει υποβληθεί. Στον ψηφιακό κόσμο, αυτήν θα μπορούσε να την παρέχει ο Πάροχος Υπηρεσιών Διαδικτύου (Internet Service Provider).
3. *Non-repudiation of delivery (μη αποποίηση παράδοσης)*: Ο παραλήπτης δεν μπορεί να αρνηθεί ότι παρέλαβε ένα μήνυμα. Αυτό μπορεί να επιτευχθεί όταν για παράδειγμα η ταχυδρομική υπηρεσία ενημερώνει τον αποστολέα ότι το μήνυμα το παρέλαβε ο παραλήπτης. Η μη αποποίηση διανομής δεν αναφέρεται στο περιεχόμενο του μηνύματος και σχετίζεται μόνο με την παραλαβή του.
4. *Non-repudiation of receipt (μη αποποίηση απόδειξης παραλαβής)*: Παρέχεται στον αποστολέα ενός μηνύματος η απόδειξη ότι το μήνυμα παραλήφθηκε χωρίς λάθη από τον επιθυμητό παραλήπτη. Για παράδειγμα σε μια περίπτωση ηλεκτρονικού εμπορίου η μη αποποίηση απόδειξης παραλαβής εξασφαλίζει ότι ο έμπορος δεν μπορεί να αργότερα να αρνηθεί ότι έλαβε μια παραγγελία από κάποιον πελάτη ή να ισχυριστεί ότι έλαβε την παραγγελία άλλα σε διαφορετική χρονική στιγμή από την πραγματική.

Μια από τις συνήθεις τεχνικές που χρησιμοποιούνται για την επίτευξη της απαίτησης για μη αποποίηση είναι οι ψηφιακές υπογραφές [21] που είναι το ανάλογο της



ιδιόχειρης υπογραφής. Ο αποστολέας υπογράφει το μήνυμα ή τη σύνοψη του μηνύματος η οποία μπορεί να υπολογιστεί από μια κρυπτογραφικά ασφαλή συνάρτηση hash. Οι ψηφιακές υπογραφές χρησιμοποιούν ασύμμετρη κρυπτογραφία για την κρυπτογράφηση και αποκρυπτογράφηση του μηνύματος. Στην ασύμμετρη κρυπτογραφία [22] χρησιμοποιούνται δύο κλειδιά ένα δημόσιο, που είναι γνωστό σε όλους, και ένα ιδιωτικό γνωστό μόνο στον ιδιοκτήτη του κλειδιού. Σύμφωνα με τις αρχές της ασύμμετρης κρυπτογράφησης είναι υπολογιστικά ανέφικτο κάποιος που γνωρίζει το δημόσιο κλειδί να υπολογίσει το ιδιωτικό κλειδί και επίσης κάποιος μπορεί να χρησιμοποιήσει το δημόσιο κλειδί για να αποκρυπτογραφήσει ένα κείμενο που έχει κρυπτογραφηθεί με το αντίστοιχο ιδιωτικό κλειδί. Συνεπώς η διαδικασία που ακολουθείται είναι ο αποστολέας να υπογράφει το μήνυμα με το ιδιωτικό κλειδί του και ο παραλήπτης να χρησιμοποιεί το δημόσιο κλειδί του αποστολέα για να αποκρυπτογραφήσει το μήνυμα. Ένα πρόβλημα που προκύπτει είναι ότι κατά την αποκρυπτογράφηση ενός κειμένου με το δημόσιο κλειδί δεν αποδεικνύεται ότι το κλειδί αυτό ανήκει στον συγκεκριμένο αποστολέα. Για το σκοπό αυτό χρησιμοποιείται ο Πάροχος Υπηρεσιών Πιστοποίησης ο οποίος είναι ένας οργανισμός-οντότητα που πιστοποιεί ότι το συγκεκριμένο δημόσιο κλειδί ανήκει στον συγκεκριμένο άνθρωπο. Συνεπώς με τις ψηφιακές υπογραφές μπορεί να εξασφαλιστεί ότι το συγκεκριμένο μήνυμα στάλθηκε από τον συγκεκριμένο αποστολέα.

Μια εναλλακτική μέθοδος που χρησιμοποιείται είναι το μήνυμα αντί να υπογράφεται από τον αποστολέα, να αποστέλλεται στον Πάροχο Υπηρεσίας Πιστοποίησης ο οποίος το υπογράφει χρησιμοποιώντας το ιδιωτικό κλειδί του. Είναι απαραίτητο ο Πάροχος Υπηρεσίας Πιστοποίησης να λαμβάνει τα απαραίτητα μέτρα σχετικά με την ταυτότητα του αποστολέα ώστε να μην υπογράφει παράνομα έγγραφα. Αυτή η εναλλακτική μειώνει την ανάγκη για έκδοση πολλών πιστοποιητικών που πιστοποιούν τη σχέση ενός ανθρώπου με το δημόσιο κλειδί του.

Επιπλέον τα τελευταία χρόνια με την ανάπτυξη της τεχνολογίας blockchain εξετάζεται η χρήση αυτής για την επίτευξη μη αποποίησης. Ένα δημόσιο blockchain δίκτυο παρέχει μη αποποίηση καθώς όλοι έχουν το δικαίωμα να διαβάσουν την αλυσίδα των block που περιέχουν όλες τις συναλλαγές που έχουν γίνει σε αυτό. Ένα από τα πλεονεκτήματα αυτής της τεχνικής είναι ότι μπορούν να βρεθούν όλες οι συναλλαγές που έχουν γίνει από μια συγκεκριμένη διεύθυνση που ανήκει σε κάποιον συμμετέχοντα του δικτύου. Επίσης κάθε συναλλαγή συνοδεύεται από μια απόδειξη στην οποία αναγράφονται μεταξύ άλλων ο αποστολέας, ο παραλήπτης και η χρονική σήμανση που δείχνει πότε αυτή εκτελέστηκε. Όμως ο οποιοσδήποτε που συμμετέχει σε ένα δίκτυο blockchain μπορεί να αποκτήσει να δημιουργήσει έναν λογαριασμό, να αποκτήσει ένα ζευγάρι δημόσιου και ιδιωτικού κλειδιού για να εκτελεί συναλλαγές χωρίς να χρειάζεται να παρέχει κάποια πιστοποιητικά της πραγματικής του ταυτότητας. Το δημόσιο κλειδί που χρησιμοποιείται για την εκτέλεση μιας συναλλαγής στο δίκτυο του blockchain συνδέεται με μια συγκεκριμένη αναπαράσταση 160 bit η οποία ονομάζεται διεύθυνση και είναι αυτή που είναι ορατή στις αποδείξεις των συναλλαγών. Για περισσότερες πληροφορίες πώς από ένα δημόσιο κλειδί παράγεται μια συγκεκριμένη διεύθυνση μπορεί κάποιος να ανατρέξει για την πλατφόρμα του Bitcoin στη σελίδα που παρέχεται από την κοινότητα του Bitcoin [23] και για την πλατφόρμα του Ethereum στο Appendix F του yellow paper του Ethereum [24]. Άρα οι διευθύνσεις δεν συνδέονται με κάποιον χρήστη για αυτό χρειάζεται να υπάρχει ένας τρόπος που να πιστοποιεί τη σχέση ενός ανθρώπου με τη συγκεκριμένη διεύθυνση. Με τον τρόπο αυτό ο ιδιοκτήτης της διεύθυνσης δεν

μπορεί να αρνηθεί οποιαδήποτε συναλλαγή στην οποία συμμετέχει αυτή η διεύθυνση. Συνοψίζοντας η τεχνολογία blockchain μπορεί να εφαρμοστεί για να επιτευχθεί η ακεραιότητα και η μη αποποίηση κάποιων δεδομένων. Για παράδειγμα έστω ότι ένας αποστολέας επιθυμεί να στείλει δεδομένα σε ένα συγκεκριμένο παραλήπτη και να εγγυηθεί για την ακεραιότητά τους. Τότε θα μπορούσε να αποθηκεύσει αναλλοίωτα και παντοτινά τη hash τιμή των δεδομένων αυτών μέσα σε ένα έξυπνο συμβόλαιο στο δίκτυο του blockchain. Ο παραλήπτης αφού λάβει τα δεδομένα μπορεί να διαπιστώσει ότι τα παρέλαβε ακέραια ελέγχοντας την τιμή που υπάρχει μέσα στο έξυπνο συμβόλαιο. Παράλληλα ο αποστολέας μπορεί να υπογράψει ψηφιακά τα δεδομένα που επιθυμεί να στείλει ώστε να εξασφαλιστεί η μη αποποίηση προέλευσης αυτών. Επίσης λόγω της τοποθέτησης δεδομένων στο blockchain δίκτυο θα εκτελεστεί μια συναλλαγή από μια διεύθυνση και θα παραχθεί μια απόδειξη συναλλαγής. Αυτή μπορεί να την υπογράψει ψηφιακά ο αποστολέας και να τη στείλει στον παραλήπτη ώστε να δεσμευτεί ο αποστολέας ότι γνωρίζει για την συναλλαγή που εκτελέστηκε στο δίκτυο του blockchain ή το δημόσιο κλειδί με το οποίο συνδέεται η διεύθυνση που χρησιμοποιήθηκε για την εκτέλεση της συναλλαγής θα μπορούσε να πιστοποιηθεί από μια έμπιστη αρχή πιστοποίησης ότι ανήκει στον συγκεκριμένο αποστολέα και έτσι αυτός να δεσμευτεί ότι η συγκεκριμένη συναλλαγή με το δίκτυο του blockchain εκτελέστηκε από αυτόν.

### 2.3.2 Τομείς αξιοποίησης της τεχνολογίας Blockchain

Η τεχνολογία Blockchain συνεχίζει να εξελίσσεται και θα μπορούσε να βρει εφαρμογή σε πολλούς διαφορετικούς τομείς. Σήμερα χρησιμοποιείται σε πολλά διαφορετικά συστήματα και έχουν προκύψει αρκετά οφέλη από την εφαρμογή της.

Η τεχνολογία Blockchain βρίσκει εφαρμογή σε:

- *Χρηματοπιστωτικές υπηρεσίες:* υπηρεσίες που αφορούν την ανταλλαγή των χρημάτων. Τέτοιες υπηρεσίες περιλαμβάνουν μεταφορές χρημάτων, εμβάσματα και πληρωμές.
- *Έξυπνα συμβόλαια:* software που αποθηκεύει τους όρους της συμφωνίας, όρους που κανείς δεν μπορεί να παραποιήσει, πιστοποιεί πως οι όροι του συμβολαίου τηρήθηκαν βάση των συμφωνηθέντων και αποδεσμεύει χρήματα, μόνο όταν οι όροι έχουν εκπληρωθεί.
- *Πιστοποίηση ιδιοκτησίας:* όπως αυτοκίνητα, ακίνητα, αποθέματα και άλλα περιουσιακά στοιχεία.
- *Εγγραφή τίτλου γης.* Τα blockchains, ως βιβλία που είναι προσβάσιμα από το κοινό, μπορούν να χρησιμοποιηθούν για όλα τα είδη καταγραφής τίτλων ιδιοκτησίας.
- *Υπηρεσίες ελέγχου ταυτότητας,* που περιλαμβάνουν επαλήθευση ταυτότητας και απόδειξη ιδιοκτησίας.
- *Διαχείριση αρχείων:* Περιλαμβάνει τη διαχείριση των φακέλων υγείας, τα κυβερνητικά αρχεία και ψηφοφορία.
- *Αποθήκευση αρχείων.* Η αποκεντρωμένη αποθήκευση αρχείων στο διαδίκτυο αποφέρει σαφή οφέλη. Η διανομή δεδομένων σε όλο το δίκτυο προστατεύει τα αρχεία από το να χάνονται.
- *Διαχείριση δικτύου και συσκευών:* επιτρέπει συνδεδεμένα αντικείμενα να επικοινωνούν μεταξύ τους.

- *Λήψη οργανωτικών αποφάσεων.* Στην πράξη, αυτό σημαίνει ότι η εταιρική διακυβέρνηση γίνεται πλήρως διαφανής και επαληθεύσιμη κατά τη διαχείριση των ψηφιακών περιουσιακών στοιχείων, της ισότητας ή της πληροφόρησης.
- *Έλεγχος της εφοδιαστικής αλυσίδας* για μια σειρά καταναλωτικών αγαθών. Οι κατανεμημένες βιβλιοθήκες παρέχουν έναν εύκολο τρόπο πιστοποίησης. Η διαφάνεια έρχεται με τη χρονική σφράγιση της ημερομηνίας και της τοποθεσίας, που αντιστοιχούν σε έναν αριθμό προϊόντος.
- *Προστασία της πνευματικής ιδιοκτησίας.* Οι έξυπνες συμβάσεις μπορούν να προστατεύσουν τα πνευματικά δικαιώματα και να αυτοματοποιήσουν την πώληση των δημιουργικών έργων στο διαδίκτυο, εξαλείφοντας τον κίνδυνο αντιγραφής και αναδιανομής αρχείων.
- *Διαχείριση ταυτότητας,* δυνατότητα ψηφιοποίησης προσωπικών εγγράφων. Η κατοχύρωση μιας ασφαλούς ταυτότητας.
- *Χρηματιστηριακές συναλλαγές* Όταν εκτελούνται peer-to-peer, οι επιβεβαιώσεις συναλλαγών γίνονται σχεδόν στιγμιαίες.  
*Ένα ιδιωτικό blockchain* διανέμει τα δεδομένα με τον ίδιο τρόπο όπως το δημόσιο blockchain, αλλά συνδέει τους υπολογιστές μέσω ενός πρωτοκόλλου που επιτρέπει μόνο σε αυτούς να ανταλλάσσουν συναλλαγές και να διατηρούν ένα κοινό βιβλίο, χωρίς επιθεώρηση από τον έξω κόσμο.

### 2.3.3 Η τεχνολογία Blockchain σήμερα

Η τεχνολογία Blockchain έχει τραβήξει την ευρύτερη προσοχή μόλις τα τελευταία δύο χρόνια. Αυτό οφείλεται στην ανάπτυξη πολλών νέων και ιδιαίτερα χρήσιμων εφαρμογών της τεχνολογίας αυτής. Μεγάλες εταιρείες και κρατικοί φορείς όπως η Citibank, η JPMorgan Chase, η Εσθονική κυβέρνηση, η Australia Post ερευνούν ή πειραματίζονται με την blockchain τεχνολογία, προκειμένου να διαφυλάττουν καλύτερα τα αρχεία και να παρέχουν νέες και πιο αποτελεσματικές υπηρεσίες. Περισσότερες από πενήντα τράπεζες στον κόσμο «τρέχουν» σε πειραματικό στάδιο εφαρμογές blockchain, όπως και μεγάλες επιχειρήσεις τεχνολογίας, η Microsoft, η IBM και η Google. Η Deutsche Bank, η HSBC, η KBC, η Natixis, η Rabobank, η Société Générale και η UniCredit ανέθεσαν στην IBM τη δημιουργία μιας πλατφόρμας βασισμένης στο blockchain, το Hyperledger Fabric, και προτίθενται να αξιοποιήσουν αυτήν την τεχνολογία στο πεδίο των διασυνοριακών συναλλαγών μεταξύ μικρομεσαίων επιχειρήσεων, αυτοματοποιώντας την.

Ο Faisal Husain, Διευθύνων Σύμβουλος της Synechron, αναφέρει ότι η εταιρεία του έχει αναπτύξει blockchain εφαρμογές που καλύπτουν τις παγκόσμιες πληρωμές, τη χρηματοδότηση του εμπορίου, τις έξυπνες συμβάσεις, την επεξεργασία ασφαλιστικών απαιτήσεων, την ταυτοποίηση του πελάτη (KYC), τη χρηματοδότηση και την επεξεργασία των ενυπόθηκων δανείων.

Με βάση την τεχνολογία του blockchain αναπτύχθηκε και η πλατφόρμα του Ethereum που παρέχει στους συμμετέχοντες του δικτύου της τόσο τη δυνατότητα χρήσης του δικού της ψηφιακού κρυπτονομίσματος για την εκτέλεση συναλλαγών, όσο και τη δυνατότητα υλοποίησης και λειτουργίας αποκεντρωμένων εφαρμογών που στηρίζονται σε προγραμματιζόμενα κομμάτια κώδικα που παραμένει αμετάβλητος και ονομάζονται έξυπνα συμβόλαια.

Η Smartcontracts.com παρέχει προγραμματιζόμενες συμβάσεις που εκτελούν τις πληρωμές μεταξύ δύο μερών, από την στιγμή που πληρούνται ορισμένα κριτήρια, χωρίς

να παρεμβάλλεται μεσάζων. Οι συμβάσεις αυτές είναι ασφαλισμένες στο blockchain ως «αυτοεκτελούμενες συμβατικές καταστάσεις», γεγονός που εξαλείφει τον κίνδυνο της εξάρτησης από άλλους για την εκπλήρωση των δεσμεύσεών τους.

Η τεχνολογία Blockchain επιτρέπει την αγορά και πώληση των ανανεώσιμων πηγών ενέργειας που παράγονται από μικροσυστήματα γειτονιάς. Όταν τα ηλιακά πάνελ παράγουν υπερβολική ενέργεια, τα έξυπνα συμβόλαια που βασίζονται στην τεχνολογία Ethereum την αναδιανέμουν αυτόματα. Παρόμοιοι τύποι έξυπνης αυτοματοποίησης συμβολαίων θα έχουν πολλές άλλες εφαρμογές, καθώς το IoT γίνεται πραγματικότητα.

Η Startup venture LO3 Energy ανέπτυξε μια πλατφόρμα υποστήριξης παραγωγής ηλεκτρισμού η οποία βασίζεται στο Blockchain και επιτρέπει την εύκολη διαχείριση της παραγωγής, της διανομής και της πώλησης ενέργειας σε τοπικό επίπεδο.

### **2.3.4 Αποτελέσματα από την χρήση της τεχνολογίας Blockchain**

Η τεχνολογία blockchain μπορεί να αξιοποιηθεί ευρύτερα. Καθιστώντας τα αποτελέσματα πλήρως διαφανή και προσβάσιμα στο κοινό, θα μπορούσε να εξασφαλίσει πλήρη διαφάνεια στις εκλογές ή σε οποιοδήποτε άλλο είδος δημοσκοπήσης. Οι έξυπνες συμβάσεις που βασίζονται στο Ethereum μπορούν να συμβάλλουν στην αυτοματοποίηση της διαδικασίας.

Η χρήση της τεχνολογίας blockchain θα μπορούσε να αποδειχθεί επωφελής για συγκεκριμένες εφαρμογές που διασχίζουν τα εθνικά σύνορα ή απαιτούν την αλληλεπίδραση και συμφωνία πολλών μη αξιόπιστων μερών. Επιπλέον, τα οφέλη θα μπορούσαν να αυξηθούν από την ικανότητα της τεχνολογίας να βοηθήσει στο χαμηλότερο κόστος, την επιτάχυνση των συναλλαγών και την εισαγωγή ενός επιπέδου αυτοματοποίησης στις διαδικασίες.

Η καταπολέμηση της νομιμοποίησης εσόδων από παράνομες δραστηριότητες (AML) και η γνώση των πρακτικών των πελατών (KYC) έχουν μεγάλες δυνατότητες να προσαρμοστούν στο blockchain. Σήμερα η τεχνολογία blockchain χρησιμοποιείται κατά κύριο λόγο για το Bitcoin, για λόγους ιχνηλασιμότητας και για την τήρηση μητρώου ακίνητης περιουσίας.

Δυστυχώς εφαρμογές όπως τα έξυπνα συμβόλαια δεν μπορούν ακόμα να λειτουργήσουν σωστά, εκτός εάν έχει προηγηθεί σωστή μελέτη και υλοποίηση, ενώ εφαρμογές που αφορούν την τήρηση ιατρικών φακέλων, ασφαλιστικά συμβόλαια, διαχείριση και αυτόματη εκτέλεση συμβολαίων προμηθευτών, συστήματα διεξαγωγής εκλογών, ψηφιακή ταυτοποίηση αναμένεται να τελειοποιηθούν.

## **2.4 Ethereum Blockchain**

Το Ethereum [7] είναι μια ανοικτού λογισμικού πλατφόρμα, η οποία βασίζεται στην τεχνολογία του διανεμημένου δημόσιου δικτύου, blockchain. Το δημόσιο blockchain είναι μια αλυσίδα χρονολογικά συνδεδεμένων δεδομένων, τα οποία είναι γνωστά σε όλους τους συμμετέχοντες του δικτύου και τα οποία από την στιγμή που καταχωρηθούν στην αλυσίδα παραμένουν αμετάβλητα. Το Ethereum δίνει τη δυνατότητα στους κόμβους που συμμετέχουν στο δίκτυο να εκτελούν peer to peer συναλλαγές χωρίς την ύπαρξη κάποιου μεσάζοντα, καθώς και να εκτελούν προγραμματιζόμενα σενάρια, που ονομάζονται smart contracts. Τόσο οι συναλλαγές όσο και τα smart contracts αποθηκεύονται στο blockchain. Το Ethereum χρησιμοποιεί το νόμισμα ether για τις συναλλαγές μεταξύ

λογαριασμών των χρηστών, αλλά και για να ανταμείψει τους κόμβους που βοήθησαν στους υπολογισμούς και στην παραγωγή των block που θα αποθηκευτούν στην αλυσίδα.

#### **2.4.1 Δημιουργία**

Η δημιουργία του Ethereum προτάθηκε για πρώτη φορά το 2013 από τον προγραμματιστή Vitalik Buterin ο οποίος συμμετείχε στο project του bitcoin, που είναι η πρώτη πλατφόρμα που χρησιμοποίησε την τεχνολογία blockchain για χρηματοοικονομικές συναλλαγές. Ο Vitalik υποστήριζε ότι το blockchain θα μπορούσε να χρησιμοποιηθεί και για κατασκευή διανεμημένων εφαρμογών που λειτουργούν χωρίς την ανάγκη ύπαρξης τρίτης έμπιστης αρχής. Όμως η πρότασή του δεν έγινε αρεστή και έτσι ξεκίνησε ένα project για τη δημιουργία μιας νέας πλατφόρμας, που ονομάστηκε Ethereum και είναι βασισμένη στο Bitcoin, το οποίο επιδοτήθηκε και τελικά υλοποιήθηκε στις 30 Ιουλίου του 2015.

#### **2.4.2 Δομή**

Βασική έννοια που χρησιμοποιείται στο Ethereum είναι το EVM (Ethereum Virtual Machine), που είναι μια εικονική μηχανή στην οποία μπορεί να εκτελείται οποιοδήποτε πρόγραμμα γραμμένο σε φιλικές γλώσσες προγραμματισμού, που βασίζονται σε ήδη υπάρχουσες γλώσσες, όπως η Javascript και η Python. Το blockchain του Ethereum διατηρείται και αναβαθμίζεται από πολλούς κόμβους που συμμετέχουν στο δίκτυό του. Κάθε κόμβος που συμμετέχει στο δίκτυο τρέχει το EVM και εκτελεί τις ίδιες εντολές, γι' αυτό πολλές φορές το Ethereum περιγράφεται ως παγκόσμιος υπολογιστής. Οι υπολογισμοί στο EVM είναι πιο αργοί από ότι θα ήταν αν γίνονταν σε έναν κανονικό υπολογιστή, όμως αυτή η καθυστέρηση είναι απαραίτητη για την επίτευξη μιας κοινής συναίνεσης μεταξύ των χρηστών στην οποία στηρίζεται η δημιουργία του blockchain του Ethereum.

#### **2.4.3 Λογαριασμοί**

Στο Ethereum υπάρχουν δύο είδη λογαριασμών :

- Externally Owned Accounts (EOAs), οι οποίες ελέγχονται από τα ιδιωτικά κλειδιά των χρηστών
- Smart contract Accounts, οι οποίες ελέγχονται από τον κώδικα που είναι γραμμένος σε αυτά και υποβάλλονται στο δίκτυο από κάποιον χρήστη ο οποίος έχει λογαριασμό EOA.

#### **2.4.4 Τα DAO**

Μια άλλη έννοια στο Ethereum είναι τα DAO (Decentralized autonomous organization), που είναι οργανώσεις βασισμένες σε ένα σύνολο από smart contracts και σκοπός τους είναι να μαζέψουν χρήματα, τα οποία θα δώσουν ως επένδυση σε κάποιες start-up επιχειρήσεις, οι οποίες αν πετύχουν θα δώσουν ένα μέρος των κερδών σε όσους επένδυσαν σε αυτές. Μια DAO δημιουργεί ένα νόμισμα ICO το οποίο πουλάει σε νόμισμα ether και όποιος κατέχει μερίδιο μπορεί να ψηφίσει σε ποιες επιχειρήσεις θα ήθελε να επενδύσει. Έτσι δίνεται η δυνατότητα σε όσους κατέχουν νόμισμα της dao να αποφασίσουν πού θα επενδυθούν τα χρήματα. Βέβαια ανάλογα με τους κανονισμούς του



DAO ο αριθμός των ψήφων που μπορεί να δώσει κάποιος εξαρτάται από τα νομίσματα DAO ICO που έχει στην κατοχή του.

#### 2.4.5 Προμήθεια

Όπως και στο Bitcoin έτσι και στο Ethereum, το blockchain αποτελείται από μπλοκ τα οποία περιέχουν δεδομένα, δημιουργήθηκαν από κόμβους που συμμετέχουν στο δίκτυο και επιθυμούν να κάνουν εξόρυξη (mine) για να αποκτήσουν ether. Όλοι οι miner συναγωνίζονται να δημιουργήσουν το επόμενο μπλοκ που θα μπει στην αλυσίδα και το οποίο πρέπει να έχει μια προκαθορισμένη μορφή, κάτι που είναι γνωστό ως proof of work και χρησιμοποιείται και στο Bitcoin. Έτσι επιτυγχάνεται η κοινή συναίνεση των χρηστών και διασφαλίζεται ότι το επόμενο μπλοκ που θα προστεθεί στην αλυσίδα θα είναι η μοναδική έκδοση της αλήθειας και δεν πρόκειται κάποιος επιτιθέμενος να την αλλάξει. Στο κοντινό μέλλον η πλατφόρμα του Ethereum σκέφτεται να αλλάξει τον αλγόριθμο συναίνεσης από proof-of-work σε proof-of-stake, όπου ο αριθμός των νομισμάτων στο δίκτυο είναι καθορισμένος και η πιθανότητα να παράγει κάποιος miner το επόμενο μπλοκ δεν βασίζεται στην υπολογιστή ισχύ της cpu του αλλά στον αριθμό των ether που κατέχει. Δηλαδή αν κάποιος έχει 300 ether, έχει τριπλάσια πιθανότητα να επιλεγεί σε σχέση με κάποιον που έχει 100 ether. Ο πρώτος miner που θα καταφέρει να δημιουργήσει επιτυχώς το block, αμείβεται με κάποια ether. Επιπλέον αμείβεται από τους αποστολείς των συναλλαγών που περιλαμβάνονται στο block με μια προμήθεια σε ether. Τέλος εάν χρειάστηκε να κάνει και κάποιους υπολογισμούς (πχ από την εκτέλεση κώδικα σε κάποιο smart contract) τότε αμείβεται με κάποια ακόμα ether τα οποία ισούνται με το γινόμενο  $\text{gas limit} \times \text{gas price}$ . Κάθε εντολή για να εκτελεστεί καθορίζεται από μια σταθερά gas. Δηλαδή η εντολή if ( $a > 2$ ) κοστίζει πάντα το ίδιο gas και δεν εξαρτάται από συναλλαγματικές αξίες όπως το ether. Το σύνολο των υπολογισμών που χρειάζεται να γίνουν ονομάζεται total gas. Για αποφευχθεί η περίπτωση που κάποιος υπολογισμός χρειάζεται να τρέξει πολλές εντολές, δηλαδή αν πέσει λόγω κάποιου λάθους στον κώδικα σε ατελείωτες επαναλήψεις, ο αποστολέας που επιθυμεί να τρέξει τον κώδικα καθορίζει κάποιο gas limit που είναι η ποσότητα του gas που επιθυμεί να ξοδέψει για την εκτέλεση των υπολογισμών. Επίσης καθορίζει και το gas price που είναι η τιμή σε ether για κάθε gas και το οποίο αν είναι πολύ μικρό θα αποθαρρύνει τον miner να το εκτελέσει για να το συμπεριλάβει στο block.

#### 2.4.6 Το σχίσμα (Hard Fork)

Το 2016 μια DAO με όνομα The DAO κατάφερε να μαζέψει από χρηματοδοτήσεις \$150 εκατομμύρια δολάρια, τα οποία θα επενδύονταν σε επιχειρήσεις. Όμως κάποιος χάκερ κατάφερε να επιτεθεί και να αποσπάσει \$50 εκατομμύρια δολάρια. Αν και η επίθεση σχετιζόταν με ένα ελάττωμα στον κώδικα του smart contract της DAO και όχι στην πλατφόρμα του Ethereum, οι δημιουργοί του Ethereum κλήθηκαν να λύσουν το πρόβλημα. Έτσι κατάφεραν να βρουν ποιος ήταν ο υπαίτιος να τον χακάρουν και να πάρουν τα χρήματα πίσω. Στη συνέχεια, για να διανεμηθούν τα κλοπιμαία πίσω στους κατόχους, έπρεπε να γίνει μια επαναφορά στο σύστημα του Ethereum, το λεγόμενο fork [25], δηλαδή οι συναλλαγές στο blockchain έπρεπε να μεταφερθούν στο σημείο που ήταν πριν την επίθεση. Όμως μια από τις βασικές αρχές που στηρίζεται το blockchain είναι ότι απαγορεύεται να γίνουν μεταγενέστερα αλλαγές σε αυτό και μια τέτοια επαναφορά θα έθετε σε κίνδυνο την αξιοπιστία του Ethereum. Για το λόγο αυτό οι δημιουργοί της

πλατφόρμας έπρεπε να έχουν τη σύμφωνη γνώμη όσων συμμετείχαν στο Ethereum. Μια καθολική συναίνεση ήταν πολύ δύσκολη κι έτσι δημιουργήθηκαν δύο πλατφόρμες μια του Ethereum, όπου τα κλοπιμαία έχουν επιστραφεί στους κατόχους και του Ethereum classic, όπου δεν έχουν διανεμηθεί.

#### **2.4.7 Δυνατότητες από τη χρήση του Ethereum**

Το Ethereum εκτός από peer to peer χρηματοοικονομικές συναλλαγές, δίνει τη δυνατότητα δημιουργίας εφαρμογών που χρησιμοποιούν τα smart contracts και ονομάζονται DApps. Με αυτόν τον τρόπο καθιστά εύκολη τη ανάπτυξη εφαρμογών, οι οποίες, αντί να χρειάζεται να δημιουργήσουν η καθεμία το δικό της blockchain, μπορούν όλες να τρέχουν στην ίδια πλατφόρμα, αυτή του Ethereum, και να αλληλεπιδρούν με το blockchain του. Όπως μια web εφαρμογή έχει ένα μέρος με το οποίο αλληλεπιδρά ο χρήστης (front-end) και χρησιμοποιεί ένα API, που είναι ένα σύνολο κανόνων και υπηρεσιών, για να αλληλεπιδράσει με μία βάση δεδομένων έτσι και μια DApp εφαρμογή web έχει επίσης ένα front-end και χρησιμοποιεί τα smart contract για να αλληλεπιδράσει με το blockchain. Μια DApp εφαρμογή δεν χρειάζεται μια έμπιστη τρίτη αρχή για να λειτουργήσει, καθώς ο κώδικάς της τρέχει στο δίκτυο. Μπορούν να δημιουργηθούν DApp που σχετίζονται με τον τομέα της υγείας, για την καταγραφή και αποθήκευση στοιχείων ασθενών ή και για την καταγραφή των κλινικών μελετών που γίνονται στα φάρμακα, τα οποία θα γίνονται γνωστά σε όλους, στον τομέα των κτηματοοικονομικών για την αγορά ακινήτων χωρίς μεσάζοντα, στον τομέα της αυτοβιομηχανίας, όπου όταν γίνει κάποιο ατύχημα άμεσα θα μεταφέρονται τα χρήματα από την ασφάλεια στους δικαιούχους, στο internet of things, όπου συσκευές θα μπορούν να αλληλεπιδρούν με το blockchain και να αποθηκεύουν ή να παίρνουν πληροφορίες και σε άλλους τομείς. Για τον έλεγχο μιας DApp εφαρμογής και την εύρεση προβλημάτων στον κώδικά της, το Ethereum δίνει τη δυνατότητα στους προγραμματιστές να χρησιμοποιήσουν προγράμματα όπως το testrpc που δημιουργεί ένα blockchain στον κόμβο που τρέχει ή να συνδεθούν σε ένα test network (ropsten, rinkeby, kovan) και να ελέγξουν αν η εφαρμογή τους λειτουργεί με τον επιθυμητό τρόπο, χρησιμοποιώντας ether τα οποία όμως δεν έχουν αξία στον πραγματικό κόσμο.

Μια άλλη δυνατότητα που παρέχει το Ethereum είναι η χρηματοδότηση project. Εάν για παράδειγμα κάποιος θέλει να υλοποιήσει μια εφαρμογή, αλλά δεν έχει τα χρήματα, μπορεί να δημιουργήσει ένα νόμισμα ico να το πουλήσει για ether και έτσι, μέσω χρηματοδότησης, να συγκεντρώσει χρήματα. Όποιος κατέχει ico που δημιουργήθηκε από τη συγκεκριμένη επιχείρηση σε περίπτωση που η εφαρμογή επιφέρει κέρδη θα έχει και αυτός μερίδιο σε αυτά.

#### **2.4.8 Smart contracts**

Το Ethereum δίνει τη δυνατότητα στους χρήστες, εκτός από χρηματοοικονομικές συναλλαγές, να εκτελούν στο δίκτυό του προγραμματιζόμενα σενάρια, που ονομάζονται smart contracts [26] [27]. Τα έξυπνα συμβόλαια είναι τμήματα κώδικα, γραμμένα σε μια γλώσσα προγραμματισμού, τα οποία όταν ενεργοποιηθούν, εκτελούν πιστά ό,τι αναφέρει ο κώδικάς τους. Συντάσσονται σε γλώσσες κατανοητές από τους ανθρώπους και ύστερα περνάνε από το μεταγλωττιστή και ο κώδικάς τους μετατρέπεται σε bytecode, μια γλώσσα μηχανής κατανοητή από την εικονική μηχανή του Ethereum (EVM). Παράλληλα με την μεταγλώττιση παράγεται το ABI (application binary interface), που αφορά το

συγκεκριμένο συμβόλαιο και το οποίο μπορεί να το χρησιμοποιήσει ένας χρήστης που επιθυμεί να κάνει χρήση κάποιας συνάρτησης μιας έξυπνης σύμβασης, δηλαδή χρησιμοποιεί το ABI για να κάνει hash την επιθυμητή συνάρτηση έτσι ώστε να μπορεί να δημιουργήσει το EVM bytecode που απαιτείται για να την καλέσει αργότερα. Με αυτόν τον τρόπο οι εντολές στο συμβόλαιο γίνονται κατανοητές από το EVM ώστε να μπορεί να τις εκτελέσει. Οι γλώσσες προγραμματισμού που μπορούν να χρησιμοποιηθούν για την σύνταξη των συμβολαίων είναι:

- *Solidity*, που χρησιμοποιείται περισσότερο και μοιάζει με την γλώσσα Javascript και C++
- *Serpent*, η οποία μοιάζει με την Python και
- *LLL*, που βασίζεται στην γλώσσα Lisp.

Για να μπορεί να αποθηκευτεί κάποιο contract σε ένα block στο blockchain, χρειάζεται ο δημιουργός του να πληρώσει κάποια ether, τα οποία πηγαίνουν στον miner ως προμήθεια, επειδή συμπεριέλαβε το contract στο block. Με τον τρόπο αυτό το συμβόλαιο αποκτά μια διεύθυνση, την οποία χρησιμοποιούν οι χρήστες για να αλληλεπιδράσουν με αυτό.

Κάθε συμβόλαιο μπορεί να αποθηκεύσει στο blockchain  $2^{256}$  πιθανά κλειδιά και  $2^{256}$  τιμές που είναι αρκετός χώρος για την αποθήκευση μιας βάσης δεδομένων. Βέβαια κάθε φορά που κάποιος αποθηκεύει δεδομένα σε ένα συμβόλαιο ή του ζητά να εκτελέσει υπολογισμούς και να του επιστρέψει ένα αποτέλεσμα, χρειάζεται να πληρώνει κάποιο αντίτιμο σε ether, το οποίο πηγαίνει στους miner ως ανταμοιβή.

Ένα συμβόλαιο μπορεί να αλληλεπιδράσει με ένα άλλο συμβόλαιο καθώς και να ενεργοποιηθεί μόνο του, όταν εκπληρωθούν κάποιες προϋποθέσεις (πχ όταν γίνει κάποιο ατύχημα να μεταφερθούν τα χρήματα από την ασφαλιστική στον δικαιούχο). Τα στοιχεία που προέρχονται από τον εξωτερικό κόσμο και τα οποία ενεργοποιούν το συμβόλαιο, όπως η ύπαρξη ατυχήματος και το ποιος είναι ο υπαίτιος στο παραπάνω παράδειγμα, θα δίνονται από οντότητες που θα ονομάζονται oracle. Αυτές θα ενημερώνουν κάποιο έξυπνο συμβόλαιό τους, με το οποίο θα μπορούν να αλληλεπιδρούν άλλα συμβόλαια για να μαθαίνουν πληροφορίες για τον έξω κόσμο. Υπάρχουν εταιρίες που εξειδικεύονται στην εγγραφή συμβολαίων για oracle και δρουν σαν έμπιστο μέσο.

Με τα smart contract μπορούν:

1. Να κατασκευαστούν DApp (decentralized applications), εφαρμογές που χρησιμοποιούνται από τα έξυπνα συμβόλαια για να αλληλεπιδρούν με το blockchain. Η κατασκευή τέτοιων εφαρμογών μπορεί να είναι χρήσιμη σε πολλούς τομείς όπως στην ιατρική για καταχώρηση στοιχείων των ασθενών στο blockchain στα οποία θα έχει άμεση πρόσβαση ο ασθενής, στη φαρμακοβιομηχανία για καταχώρηση των δοκιμών που γίνονται στα φάρμακα στο blockchain ώστε να είναι γνωστό σε όλους, στην αυτοβιομηχανία πχ για πληρωμή των δικαιούχων από τις ασφαλιστικές, στην άμεση πληρωμή πάροχων υπηρεσιών από όσους τις χρησιμοποιούν, όπως για παράδειγμα στην πληρωμή κάποιας συνδρομητικής τηλεόρασης και σε πολλούς άλλους τομείς.
2. Να δημιουργηθούν αποκεντρωμένες οργανώσεις που δεν έχουν κάποια κεντρική αρχή, τα DAO (Decentralized Autonomous Organizations)
3. Να αναζητηθούν χρηματοδότες. Για παράδειγμα αν κάποιος ψάχνει για ανθρώπους να επενδύσουν σε κάποια ιδέα του, μπορεί να εκδώσει ένα νόμισμα ICO και να το πουλήσει σε άλλους χρήστες με τη χρήση κάποιου smart contract



παίρνοντας ether. Μετά την υλοποίηση της ιδέας, ένα μέρος των κερδών πηγαίνει σε όσους είχαν αγοράσει τα συγκεκριμένα ICO.

#### 2.4.9 Επεξήγηση των εννοιών ABI-API

Ο όρος API χρησιμοποιείται ευρύτατα, ενώ όρος ABI έγινε πιο δημοφιλής τα τελευταία χρόνια και είναι πολύ διαδεδομένος στο δίκτυο του blockchain αφού σχετίζεται με κάθε έξυπνο συμβόλαιο. Πιο αναλυτικά οι όροι αυτοί:

► *Application Binary Interface (ABI)* [28]: Ο όρος Application Binary Interface - Διεπαφή Δυναμικών Εφαρμογών περιγράφει μια χαμηλού επιπέδου διεπαφή ώστε να μπορούν να επικοινωνούν δύο ή περισσότερες δυαδικές μονάδες [29]. Κάθε smart contract στο Ethereum περιγράφεται από ένα ABI το οποίο είναι μια περιγραφή σε μορφή JSON η οποία περιλαμβάνει τα ονόματα όλων των συναρτήσεων που χρησιμοποιεί το συμβόλαιο, τις εισόδους και τις εξόδους αυτών των συναρτήσεων καθώς και το είδος τους, τα ονόματα των event που χρησιμοποιεί το συμβόλαιο καθώς και οι παράμετροι που ενεργοποιούνται με κάθε event. Τα smart contracts αποθηκεύονται ως bytecode σε δυαδική μορφή μέσα στο blockchain σε συγκεκριμένη διεύθυνση, επομένως για να προσπελαστούν αυτά τα δυαδικά δεδομένα χρησιμοποιείται το ABI. Με λίγα λόγια το ABI περιγράφει τον τρόπο με τον οποίο μπορούν να καλεστούν οι συναρτήσεις ενός συμβολαίου ώστε να επιστρέψουν δεδομένα. Το ABI ενός smart contract δημιουργείται αφού αυτό γίνει compile. Ένα παράδειγμα ABI που προέκυψε για ένα από τα συμβόλαια που υλοποιήθηκαν σε αυτή τη διπλωματική είναι το εξής:

Παράδειγμα ABI:

```
[{"inputs": [{"name": "_hashValue", "type": "bytes32"}], "payable": false, "stateMutability": "nonpayable", "type": "constructor"}, {"constant": true, "inputs": [], "name": "getData", "outputs": [{"name": "_data", "type": "bytes32"}], "payable": false, "stateMutability": "view", "type": "function"}]
```

► *Application Program Interface (API)* [30]: Ο όρος Application Program Interface - Προγραμματιστική Διεπαφή Εφαρμογών περιγράφει ένα σύνολο κανόνων και προδιαγραφών ώστε τα προγράμματα λογισμικού να μπορούν εύκολα να αλληλεπιδρούν μεταξύ τους. Στο Ethereum για να αλληλεπιδράσει κάποιος με ένα smart contract θα χρειαστεί μια συλλογή από βιβλιοθήκες που ονομάζεται web3.js και η οποία αποτελεί το επίσημο Ethereum Javascript API. Το web3.js επιτρέπει την αλληλεπίδραση ενός τοπικού ή απομακρυσμένου Ethereum κόμβου χρησιμοποιώντας HTTP ή IPC σύνδεση.

#### 2.4.10 Κλιμάκωση (Scalability)

Το blockchain δίκτυο του Ethereum συνεχώς μεγαλώνει, προστίθενται νέοι κόμβοι και αυξάνεται ο αριθμός των συναλλαγών που γίνονται σε αυτό. Όμως τίθενται κάποια προβλήματα από την επέκταση αυτή καθώς είναι απαραίτητο κάθε κόμβος στο δίκτυο να επεξεργάζεται κάθε συναλλαγή και έτσι περιορίζεται η ικανότητα επεξεργασίας των συναλλαγών του δικτύου στην ικανότητα επεξεργασίας που παρέχει ένας κόμβος. Αυτό έχει ως αποτέλεσμα ο χρόνος για την επικύρωση και την εκτέλεση των συναλλαγών να αυξάνεται αρκετά.

Σύμφωνα με το επίσημο blog του Ethereum [31], σκέφτονται να αντιμετωπίσουν το πρόβλημα της επεκτασιμότητας με δύο τεχνικές.

Η πρώτη ονομάζεται “*sharding*” και περιγράφει ένα μηχανισμό όπου μόνο ένα μέρος των κόμβων χρειάζεται να επικυρώσει τις συναλλαγές. Το σύστημα θα επιτρέπει την παράλληλη επεξεργασία των συναλλαγών και θα είναι ασφαλές, όσο υπάρχει αρκετός αριθμός κόμβων για την επικύρωση κάθε συναλλαγής. Με τον τρόπο η κατάσταση όλων των λογαριασμών, *externally owned accounts* και *smart contract accounts*, θα χωριστεί σε μικρότερα κομμάτια που ονομάζονται “*shard*” και κάθε κομμάτι θα έχει το δικό του ιστορικό υπολογισμών και θα μπορεί με ένα μηχανισμό να επικοινωνεί με ασφάλεια με τα υπόλοιπα κομμάτια.

Η δεύτερη περιλαμβάνει την δημιουργία “*layer 2*” πρωτοκόλλων τα οποία θα στέλνουν τις περισσότερες συναλλαγές εκτός της αλυσίδας του blockchain και θα αλληλεπιδρούν με το βασικό blockchain δίκτυο μόνο για την είσοδο και έξοδο από το σύστημα *layer-2* καθώς και σε περίπτωση που κάποιος επιτίθεται στο σύστημα. Κάποια παραδείγματα συστημάτων που χρησιμοποιούν το σύστημα *layer-2* είναι τα εξής: Plasma [32], State channels [33], Raiden [34].

Σύμφωνα με το Ethereum στοχεύουν στην υποστήριξη μιας πολυσύνθετης στρατηγικής για την επίλυση του προβλήματος της επεκτασιμότητας που θα περιλαμβάνει και τις δύο τεχνικές γι’ αυτό και αυτές αντιμετωπίζονται ισάξια και συνεχίζεται η έρευνα για την υλοποίησή τους.

## 2.5 Proof of Work ή Proof of Stake

Για την επίτευξη της κοινής συναίνεσης σε ένα blockchain δίκτυο χρησιμοποιούνται μεταξύ άλλων δύο τύποι αλγορίθμων. Ο ένας ονομάζεται Proof of Work και ο άλλος Proof of Stake [35]. Στο κοντινό μέλλον η πλατφόρμα του Ethereum σκέφτεται να αλλάξει τον τρόπο επίτευξης κοινής συναίνεσης από Proof of Work σε Proof of Stake.

### 2.5.1 Proof of Work (POW)

Πρόκειται για τον περισσότερο χρησιμοποιούμενο αλγόριθμο για την επίτευξη κοινής συναίνεσης για την χρονική σειρά των συναλλαγών μεταξύ των συμμετεχόντων σε μια πλατφόρμα blockchain. Στο Proof of Work τα μπλοκ που περιέχουν τις συναλλαγές δημιουργούνται από όσους επιθυμούν να τα φτιάξουν και συμμετέχουν στο δίκτυο μέσα από μια διαδικασία που ονομάζεται *mining* και όσοι την εφαρμόζουν ονομάζονται *miner*. Για να θεωρεί ένα μπλοκ που δημιούργησε κάποιος *miner* έγκυρο και για να γίνει αποδεκτό από όλους χρειάζεται ο κατακερματισμός αυτού του block, δηλαδή το *hash* του, να έχει μια συγκεκριμένη μορφή η οποία έχει προκαθοριστεί. Σε περίπτωση που δεν είναι συμβατή ο *miner* ξαναπροσπαθεί μέχρι να βρεθεί κάποιος που δημιούργησε μπλοκ με τη συγκεκριμένη μορφή. Όλη αυτή η διαδικασία που κάνει ο *miner* βασίζεται σε κάποια μονάδα του υπολογιστή του, το οποίο εκτελεί συνεχώς πράξεις μέχρι να πετύχει την προκαθορισμένη μορφή του *hash*. Η μονάδα αυτή μπορεί να είναι η CPU ή η GPU ή κάποιο άλλο μηχάνημα ειδικό για αυτήν την εργασία. Με λίγα λόγια στο Proof of Work όλοι οι *miner* συναγωνίζονται ταυτόχρονα για την παραγωγή ενός block με συναλλαγές και νικητής είναι αυτός που κατάφερε να το δημιουργήσει πρώτος. Ο νικητής αμείβεται για τον κόπο του με μια ποσότητα από νομίσματα της πλατφόρμας, πχ το Ethereum χρησιμοποιεί το κρυπτονόμισμα *ether* και με αυτό αμείβει τους *miner*. Τα χρήματα αυτά δεν τα παίρνει από κάποιον χρήστη του δικτύου, αλλά δημιουργούνται εκείνη τη στιγμή. Ο κερδισμένος *miner* αμείβεται και με επιπλέον χρήματα για κάθε συναλλαγή που περιέλαβε στο block του. Με άλλα λόγια ο αποστολέας κάθε συναλλαγής, η οποία

περιλαμβάνεται στο μπλοκ που θεωρήθηκε έγκυρο, πληρώνει ένα αντίτιμο στον δημιουργό (miner) του μπλοκ.

Ο αλγόριθμος συναίνεσης Proof of Work χρησιμοποιείται στη πλατφόρμα του Bitcoin και του Ethereum, δίκτυα τα οποία κάνουν χρήση της τεχνολογίας blockchain.

### 2.5.2 Proof of Stake (POS)

Ένας άλλος αλγόριθμος για την επίτευξη κοινής συναίνεσης είναι το Proof of Stake. Σε αυτό ο δημιουργός του επόμενου block, που ονομάζεται validator, επιλέγεται με μια πιθανότητα μέσα από το σύνολο όσων επιθυμούν να συμμετέχουν στη δημιουργία block. Όσο περισσότερα χρήματα της πλατφόρμας έχει κάποιος τόσο μεγαλύτερη πιθανότητα έχει να είναι αυτός που θα επιλεγεί. Στο Proof of Stake ο αριθμός των χρημάτων που θα χρησιμοποιεί η πλατφόρμα είναι συγκεκριμένος και έτσι ο δημιουργός του block δεν παίρνει κάποια νέα νομίσματα όπως συμβαίνει στο POW, αλλά αμείβεται μόνο με τους φόρους που πληρώνουν οι αποστολείς των συναλλαγών που περιέχονται στο block. Υπάρχουν πολλές μορφές υλοποίησης του αλγορίθμου POS οι δύο πιο γνωστές είναι [36]:

- *chain-based proof of stake* και
- *BFT-style proof of stake* που βασίζεται στο Byzantine Fault Tolerance (BFT) [37]. Στο σύστημα αυτό εφαρμόζεται το πρωτόκολλο Byzantine Consensus και είναι ανθεκτικό σε μια κατηγορία αποτυχιών που είναι γνωστή ως προβλήματα Byzantine Generals και ως εκ τούτου θεωρείται ανθεκτικό στην οικογένεια προβλημάτων Byzantine.

Στο *chain-based proof of stake* ο αλγόριθμος τυχαία επιλέγει ένα validator ανά κάποιο χρονικό διάστημα και του αναθέτει να δημιουργήσει το επόμενο μπλοκ και να το μεταδώσει σε όλους στο δίκτυο.

Στο *BFT-style proof of stake* οι validators επιλέγονται τυχαία από τον αλγόριθμο και τους δίνεται το δικαίωμα δημιουργίας block. Θεωρείται έγκυρο εκείνο που προκύπτει ύστερα από μια σειρά γύρων όπου ψηφίζουν για την εγκυρότητα οι υπόλοιποι validator που δεν επιλέχθηκαν για τη δημιουργία block ή ένα μέρος αυτών, ανάλογα με το πρωτόκολλο που χρησιμοποιείται. Έτσι μετά από μια σειρά πολλών γύρων, τελικά όλοι οι validator συναινούν σε ένα μπλοκ και αυτό είναι που θεωρείται έγκυρο.

Μερικά κρυπτονομίσματα που εφαρμόζουν τον αλγόριθμο συναίνεσης Proof of Stake είναι:

- Peercoin, or PPCoin (peercoin.net)
- Nxt (nxt.org)
- BlackCoin (blackcoin.co)
- Novacoin (novacoin.org)

Στο Ethereum σκέφτονται να μεταβούν από έναν αλγόριθμο POW σε έναν τύπου POS, ο οποίος έχει προταθεί και ονομάζεται Casper. Ένα άλλο blockchain που θα χρησιμοποιεί έναν αλγόριθμο συναίνεσης τύπου POS και θα βασίζεται στο Byzantine Agreement είναι το Algorand [38].

### 2.5.3 Σύγκριση των δύο τεχνικών

Τα αποτελέσματα από τη σύγκριση των δύο τεχνικών είναι τα εξής:

- Η εφαρμογή του Proof of Stake θεωρείται πιο οικολογική διότι ο επόμενος δημιουργός του block επιλέγεται τυχαία. Σε αντίθεση στο POW χρειάζεται αυξημένη κατανάλωση ηλεκτρικού ρεύματος από τα μηχανήματα ώστε αυτά να παρέχουν την απαραίτητη υπολογιστική ισχύ για την εύρεση του block με την συγκεκριμένη μορφή.
- Η εφαρμογή του POS θα μπορούσε να κάνει το δίκτυο πιο ασφαλές καθώς το πρωτόκολλο POW είναι ευάλωτο σε επιθέσεις όπου κάποιος κατέχει το 51% της υπολογιστικής ισχύος και έτσι μπορεί να δημιουργεί αυτός τα επόμενα έγκυρα μπλοκ συμπεριλαμβάνοντας τις συναλλαγές που επιθυμεί. Αυτό βέβαια απαιτεί την αγορά ισχυρών μηχανημάτων. Στο POS για να έχει κάποιος τον έλεγχο των συναλλαγών χρειάζεται να κατέχει το 51% των χρημάτων στο σύστημα κάτι πολύ δαπανηρό. Επίσης στο POS για να γίνει κάποιος validator δεσμεύονται τα χρήματα του και σε περίπτωση απόκλισης από τους κανόνες κατάσχονται, επομένως δεν είναι ωφέλιμο για κάποιον χρήστη να μην συμπεριφερθεί σύμφωνα με τους κανόνες και να χάσει τα χρήματά του. Επιπλέον δεν θα συνέφερε κάποιον που κατέχει το 51% των νομισμάτων να κάνει επίθεση και να υπονομεύσει την αξιοπιστία της πλατφόρμας, ρίχνοντας έτσι την συναλλαγματική ισχύ του νομίσματος που θα οδηγούσε και στην πτώση της οικονομικής αξίας των νομισμάτων του.
- Στο Proof of Work μπορεί να έχουμε συσσώρευση ισχύος σε μια περιοχή. Επειδή το να δημιουργήσει πρώτος κάποιος το block με την επιθυμητή μορφή, είναι δύσκολο να επιτευχθεί από έναν απλό χρήστη και απαιτεί ισχυρά μηχανήματα, οι χρήστες συμμετέχουν σε ομάδες pool όπου οι miners μοιράζονται την επεξεργαστική ισχύ των μηχανημάτων τους με άλλα μέλη του pool και εάν δημιουργηθεί από την δική τους ομάδα το μπλοκ, μοιράζονται ανάλογα με την ισχύ που παρείχαν τα έσοδα. Με τον τρόπο αυτό όμως συσσωρεύεται μεγάλη ισχύς σε ένα μέρος κάτι που θέτει σε αμφισβήτηση την αξιοπιστία του πρωτοκόλλου σε επιθέσεις όπου ένας κακεντρεχής χρήστης κατέχει το 51% της ισχύος. Ενώ στο POS ο δημιουργός επιλέγεται τυχαία και έτσι δεν υπάρχει φόβος για συσσώρευση ισχύος σε ορισμένους χρήστες.
- Στο POW μπορεί να δημιουργηθούν δυο αλυσίδες που να θεωρούνται έγκυρες έως ότου επικρατήσει η μεγαλύτερη, κάτι που ονομάζεται fork. Το fork μπορεί να οδηγήσει σε καθυστέρηση της επιβεβαίωσης μιας συναλλαγής καθώς αυτή ίσως να ήταν μέρος ενός block που ανήκε στη μικρότερη αλυσίδα η οποία τελικά δεν έγινε αποδεκτή και θα χρειαστεί η συναλλαγή να συμπεριληφθεί σε άλλο μπλοκ που θα ανήκει στην έγκυρη αλυσίδα. Για τον λόγο αυτό χρειάζεται στο POW να περάσει κάποιος χρόνος μέχρι να είναι σίγουρος ο αποδέκτης μιας συναλλαγής ότι δεν έχει συμβεί doublespent και είναι ο μοναδικός αποδέκτης της συναλλαγής. Σε αντίθεση στο POS ένα μόνο μπλοκ θεωρείται έγκυρο από όλους τους χρήστες κάθε φορά και έτσι δεν έχουμε τη δημιουργία δυο αλυσίδων.

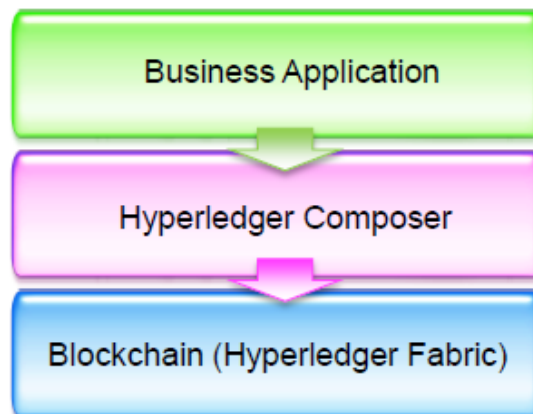
## 2.6 Hyperledger Fabric: ένα permissioned blockchain

Μια κατηγορία ενός blockchain δικτύου, είναι το permissioned blockchain. Σε αυτή την κατηγορία ανήκουν όσα δίκτυα επιτρέπουν σε ορισμένους μόνο κόμβους να συμμετέχουν στην διαδικασία της επικύρωσης των συναλλαγών των block. Παράλληλα στα δίκτυα αυτά δεν έχουν όλοι οι κόμβοι το δικαίωμα να διαβάσουν και να γράψουν

στην αλυσίδα του blockchain. Για παράδειγμα σε ένα permissioned blockchain που συμμετέχουν 10 εταιρείες, στις οποίες η εταιρεία 1 συναλλάσσεται μόνο με την εταιρεία 2, δεν υπάρχει λόγος να δοθεί δικαίωμα στις υπόλοιπες εταιρίες να έχουν πρόσβαση στις συναλλαγές που γίνονται μεταξύ των εταιριών 1 και 2. Επιπλέον, σε αντίθεση με τα δημόσια blockchain, οι συναλλαγές γίνονται γρηγορότερα, η αμοιβή των κόμβων που συμμετέχουν στην επικύρωση των συναλλαγών είναι μικρή και οι κόμβοι που συμμετέχουν στην διαδικασία της επικύρωσης δεν είναι άγνωστοι. Ένα τέτοιο είδος permissioned blockchain είναι το Hyperledger Fabric.

Το Hyperledger [39] [40] δημιουργήθηκε από οργανισμό “THE LINUX FOUNDATION” τον Δεκέμβριο του 2015 και πρόκειται για μια παγκόσμια συνεργασία μεταξύ ηγετών που ασχολούνται σε τομείς της οικονομίας, των τραπεζικών εργασιών, του Internet of Things, της αλυσίδας εφοδιασμού, των κατασκευών και της τεχνολογίας. Είναι μια ανοιχτού κώδικα συνεργατική προσπάθεια που έχει ως στόχο της να διευκολύνει την επικοινωνία μεταξύ των επιχειρήσεων που είναι μέλη του. Σήμερα αριθμεί 130+ μέλη και 8 project που βρίσκονται σε εξέλιξη, μεταξύ των οποίων είναι τα Hyperledger Fabric και Hyperledger Composer.

Το Hyperledger Fabric είναι ένα permissioned blockchain, δηλαδή ένα blockchain στο οποίο χρειάζεσαι άδεια για να διαβάσεις και να γράψεις, που δημιουργήθηκε από την IBM και την Digital Asset. Το Hyperledger Composer είναι μια εφαρμογή που χρησιμοποιείται, ώστε ένα επιχειρηματικό δίκτυο να μπορεί να μοντελοποιηθεί στο blockchain και στη συνέχεια να χρησιμοποιηθεί αυτό το μοντέλο για να δημιουργηθεί ένα δίκτυο που κάνει χρήση της τεχνολογίας blockchain όπως φαίνεται στην εικόνα 2.3.



**Εικόνα 2.3.** Fabric Composer Positioning

Πηγή: <https://www.slideshare.net/dselman/hyperledger-composer-update-2017-0405>

Χρησιμοποιώντας το Hyperledger Composer μπορεί κάποιος εύκολα να ορίσει [41]:

- ποια θα είναι τα μέλη του δικτύου,
- ποιος είναι ο κώδικας των έξυπνων συμβολαίων που θα χρησιμοποιεί (π.χ ένα έξυπνο συμβόλαιο θα μπορούσε να περιέχει τον κώδικα για την πραγματοποίηση μιας αγοράς),
- ποιες άδειες θα έχει ο κάθε συμμετέχων στο blockchain δίκτυο ως προς την ανάγνωση των συναλλαγών που έχουν αποθηκευτεί στο blockchain αλλά και ως προς τη δημιουργία των συναλλαγών. Για παράδειγμα έστω ένα επιχειρηματικό

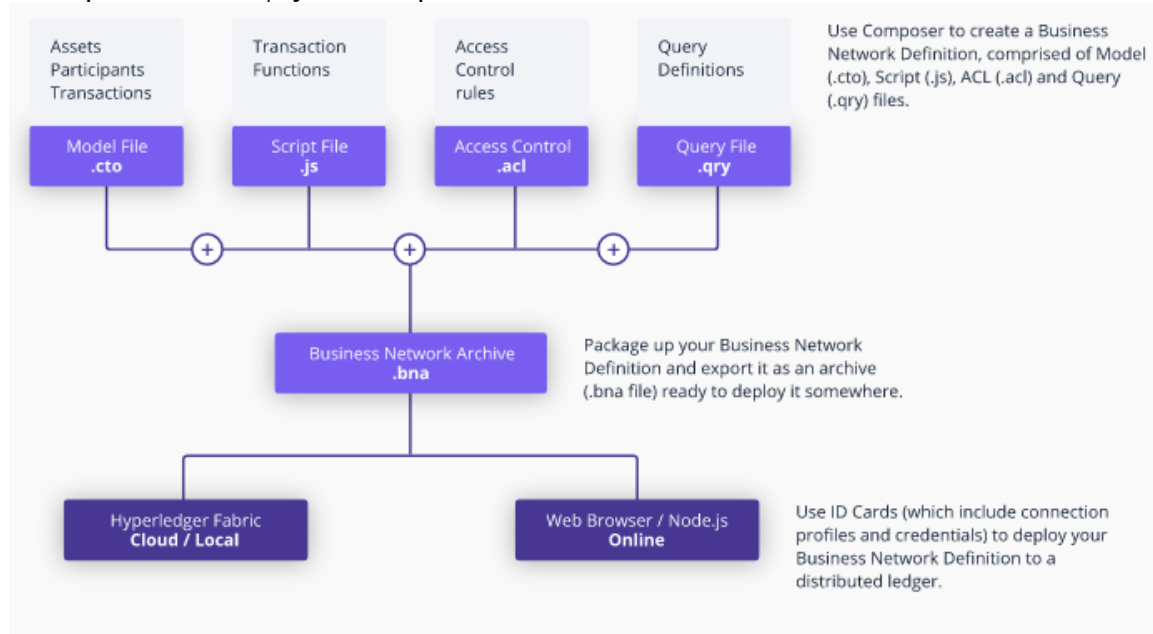


δίκτυο αγοροπωλησέων που περιλαμβάνει ως μέλη μεταξύ άλλων πωλητές και αγοραστές. Δεν θα ήταν θεμιτό όλοι να ξέρουν λεπτομέρειες από μια συναλλαγή που έγινε μεταξύ ενός πωλητή και ενός αγοραστή διότι μπορεί κάποιος να είναι ανταγωνιστής και να κάνει καλύτερη προσφορά στον αγοραστή.

- τα κριτήρια με τα οποία επιλέγονται τα αντικείμενα που μπορούν να μεταφερθούν στο δίκτυο καθώς και οι συμμετέχοντες στο δίκτυο.

Για να χρησιμοποιήσει κάποιος το Hyperledger Composer μπορεί να επισκεφτεί την ιστοσελίδα [42] και να μοντελοποιήσει ένα επιχειρηματικό δίκτυο. Στη συνέχεια, μπορεί να εξάγει αυτό το μοντέλο από τον composer σε ένα αρχείο(.bna file) που θα περιέχει πακεταρισμένα τα αρχεία που δημιουργήθηκαν με το εργαλείο του Hyperledger Composer. Για να διαπιστώσει κάποιος την ορθή λειτουργία του μοντέλου μπορεί να φορτώσει το (.bna) αρχείο είτε στο Hyperledger Composer που βρίσκεται στην ιστοσελίδα που δόθηκε προηγουμένως είτε σε ένα Hyperledger Component που τρέχει τοπικά στον υπολογιστή, αφού πρώτα κατεβάσει τα απαραίτητα αρχεία.

Τα παραπάνω συνοψίζονται στην εικόνα 2.4 [43]:



**Εικόνα 2.4.** Αρχιτεκτονική του Hyperledger Composer.

Πηγή: <https://hyperledger.github.io/composer/v0.16/introduction/introduction.html>

Σε αυτό το private blockchain μπορεί για τη συναίνεση των κόμβων να επιλεγεί, ανάλογα με τις ανάγκες, κάποιος από τους εξής αλγόριθμους συναίνεσης : Proof of work, Proof of stake, Solo, Proof of Elapsed Time, PBFT-based, Kafka/ Zookeeper.

Οι κόμβοι που συμμετέχουν στο blockchain δίκτυο δεν έχουν όλοι τους ίδιους ρόλους. Πιο συγκεκριμένα υπάρχουν οι κόμβοι:

- *committing peer* οι οποίοι κρατάνε ένα αντίγραφο των συναλλαγών που δημιουργούνται, το ledger όπως ονομάζεται, και οι οποίοι μπορούν να δημιουργήσουν μια συναλλαγή.
- *endorsing peer* που δέχονται τις συναλλαγές και αναλόγως είτε τις εγκρίνουν είτε όχι. Για να μπορούν να επιτελέσουν την λειτουργία τους πρέπει να έχουν στην κατοχή τους τον κώδικα των smart contract που χρησιμοποιεί το δίκτυο.

- *ordering nodes(service)*: λαμβάνει τις συναλλαγές που έχουν εγκριθεί και χρησιμοποιώντας κάποιον αλγόριθμο συναίνεσης τις ομαδοποιεί σε block τα οποία στέλνει στους κόμβους committing peer ώστε να ενημερώσουν το ledger που είναι το αρχείο με τις συναλλαγές. Οι κόμβοι αυτοί δεν έχουν κάποιο αντίγραφο του ledger ούτε των smart contract.

# ΚΕΦΑΛΑΙΟ 3

## Περιγραφή της εφαρμογής

### 3.1 Καθορισμός του προβλήματος που διευθετείται και των απαιτήσεων που εξασφαλίζονται

Η εφαρμογή που αναπτύχθηκε σε αυτή την διπλωματική έχει ως σκοπό να αντιμετωπίσει το πρόβλημα το οποίο μπορεί να καθοριστεί ως εξής:

Κάποιος χρήστης που επιθυμεί να λάβει ιατρική γνώση εκτελεί ένα ερώτημα σε μια βιοϊατρική βάση δεδομένων χρησιμοποιώντας τη σελίδα της εφαρμογής. Δηλαδή, η εφαρμογή δρα σαν μεσάζοντας ανάμεσα στον χρήστη και την ιατρική βάση. Στη συνέχεια ο χρήστης λαμβάνει τα δεδομένα της απάντησης που υπέβαλε, όπως επιστρέφονται από την βιοϊατρική βάση, και μαζί με αυτά λαμβάνει και μια εγγύηση ότι έχουν εκπληρωθεί κάποιες απαιτήσεις. Αυτές τις απαιτήσεις τις διασφαλίζει η εφαρμογή χρησιμοποιώντας το blockchain, τα έξυπνα συμβόλαια και υποδομή δημόσιου κλειδιού.

Έτσι όταν ένας χρήστης υποβάλλει ένα ερώτημα μέσω της εφαρμογής είναι σίγουρος:

- για την ακεραιότητα των δεδομένων που λαμβάνει για το συγκεκριμένο ερώτημα που έθεσε στη βιοϊατρική βάση.
- ότι τα δεδομένα είναι ενημερωμένα, καθώς με το που υποβάλλεται το ερώτημα στη σελίδα της εφαρμογής αυτό προωθείται στην βιοϊατρική βάση για ανάκτηση της απάντησης.
- ότι η βιοϊατρική βάση δεν μπορεί να αποποιηθεί ότι παρείχε αυτά τα δεδομένα.

Η εφαρμογή εξασφαλίζει τις παραπάνω απαιτήσεις αποθηκεύοντας αναλλοίωτα για πάντα τη hash τιμή της απάντησης που επιστρέφεται από την ιατρική βάση, στα έξυπνα συμβόλαια στο δίκτυο του blockchain. Ο κώδικας των συμβολαίων απαγορεύει την οποιαδήποτε μεταβολή αυτής της hash τιμής και επειδή ο κώδικας αυτός των συμβολαίων που βρίσκονται στο blockchain δεν μπορεί να αλλάξει, διασφαλίζεται η αναλλοίωτη και παντοτινή αποθήκευση αυτής της hash τιμής. Ο λόγος που αποθηκεύεται η hash τιμή της απάντησης και όχι η απάντηση αυτούσια είναι για μείωση του κόστους, διότι όσο μεγαλύτερο είναι το μέγεθος των δεδομένων που αποθηκεύονται στα έξυπνα συμβόλαια τόσο μεγαλύτερο είναι το κόστος. Για τον υπολογισμό της αυτής της hash τιμής χρησιμοποιείται μια κρυπτογραφικά ασφαλής συνάρτηση κατακερματισμού της οποίας η έξοδος είναι μοναδική για κάθε τιμή εισόδου. Έτσι αν για κάποιο λόγο η τιμή της εισόδου, της απάντησης δηλαδή που επιστράφηκε, αλλάξει έστω και κατά ένα γράμμα η hash τιμή που θα παραχθεί θα είναι διαφορετική. Συνεπώς τα δεδομένα που βρίσκονται στα συμβόλαια είναι σύμφωνα με την απάντηση που δόθηκε από την βιοϊατρική βάση και έτσι αυτή δεν μπορεί αργότερα να τα μεταβάλλει και να αποποιηθεί ότι τα παρείχε. Για την αποθήκευση των δεδομένων στα συμβόλαια ο ιδιοκτήτης της εφαρμογής δημιούργησε αρχικά έναν λογαριασμό στο δίκτυο του Ethereum, δηλαδή

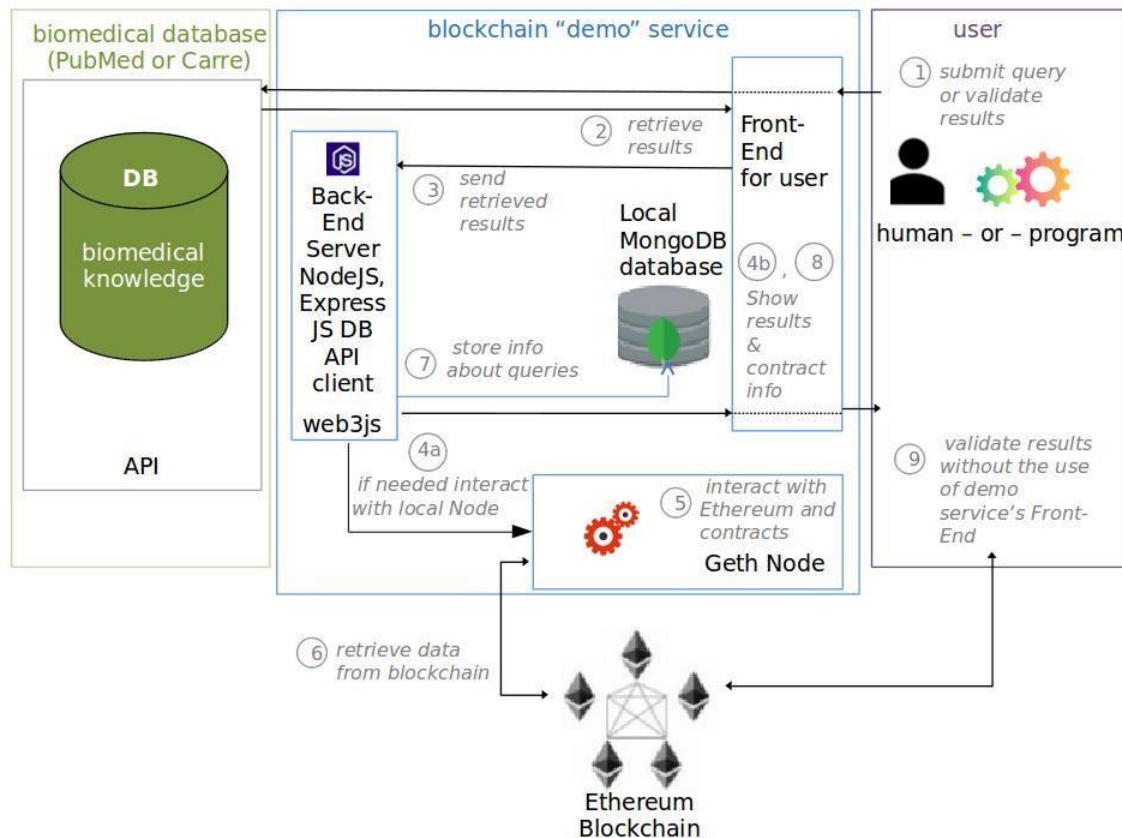


απέκτησε ένα ζευγάρι δημόσιου και ιδιωτικού κλειδιού ώστε να εκτελεί συναλλαγές με το δίκτυο. Το δημόσιο κλειδί συνδέεται μοναδικά με μια συγκεκριμένη αναπαράσταση 160 bit [Appendix F[24]] που αποτελεί μια διεύθυνση και η οποία είναι αυτή που φαίνεται ως αποστολέας μιας συναλλαγής. Θα μπορούσε να χρησιμοποιηθεί μια έμπιστη αρχή πιστοποίησης για να πιστοποιηθεί ότι το συγκεκριμένο δημόσιο κλειδί, που χρησιμοποιείται στο δίκτυο του blockchain, και άρα και η διεύθυνση που του αντιστοιχεί, ανήκει στον ιδιοκτήτη της εφαρμογής και έτσι να συνδεθεί η ταυτότητα αυτού του ιδιοκτήτη με τις συναλλαγές που εκτελούνται από τη συγκεκριμένη διεύθυνση και έτσι να μην μπορεί να αποποιηθεί ότι η συναλλαγή προήλθε από αυτόν. Τέλος τα δεδομένα που επιστρέφονται από την εφαρμογή στο χρήστη και πρέπει να είναι ψηφιακά υπογεγραμμένα ώστε να εξασφαλιστεί η μη αποποίηση προέλευσης των δεδομένων που παραλαμβάνει ο χρήστης.

## 3.2 Περιγραφή της αρχιτεκτονικής της εφαρμογής

Η εφαρμογή που υλοποιήθηκε αποτελείται από πέντε μέρη τα οποία είναι:

1. *το front-end μέρος*: είναι η σελίδα που βλέπει ο χρήστης όταν επισκέπτεται την εφαρμογή και με αυτό αλληλεπιδρά ο χρήστης. Εκεί εκτελείται το ερώτημα στη βιοϊατρική βάση και τα δεδομένα της απάντησης προωθούνται στον server. Ο λόγος που το ερώτημα εκτελείται στο μέρος του πελάτη και όχι στο back-end είναι για να μειωθούν οι διεργασίες που εκτελεί ο server και έτσι το σύστημα να είναι πιο γρήγορο. Σε αυτό το μέρος προβάλλονται στον χρήστη τα δεδομένα που επιστρέφονται από το server μετά την υποβολή ενός ερωτήματος ή την αίτηση για μια διαδικασία επικύρωσης. Ο χρήστης μπορεί να είναι είτε κάποιος άνθρωπος είτε κάποιο πρόγραμμα που επικοινωνεί με το API της εφαρμογής.
2. *Το back-end μέρος του server*. Διαχειρίζεται τα αιτήματα που έρχονται από το front-end και στέλνει σε αυτό τα απαραίτητα δεδομένα. Επικοινωνεί με τον τοπικό κόμβο Geth και προσδιορίζει ποιες συναλλαγές χρειάζεται να υπάρξουν με το blockchain και τέλος αποθηκεύει κάποιες πληροφορίες σχετικά με τα ερωτήματα στη τοπική MongoDB βάση.
3. *Η τοπική βάση MongoDB* στην οποία αποθηκεύονται δεδομένα σχετικά με τα ερωτήματα και πληροφορίες που σχετίζονται με αυτά.
4. *Ο κόμβος Geth* ο οποίος τρέχει τοπικά στο ίδιο μηχάνημα με τον server και παρέχει τη δυνατότητα επικοινωνίας της εφαρμογής με το Ethereum blockchain και με τα έξυπνα συμβόλαια που χρησιμοποιεί η εφαρμογή.
5. *Τα έξυπνα συμβόλαια*. Η εφαρμογή χρησιμοποιεί τέσσερα είδη έξυπνων συμβολαίων (smart contracts). Το ένα παρέχεται από το εργαλείο Truffle και τα υπόλοιπα τρία αναπτύχθηκαν σε αυτή την διπλωματική και σε αυτά αποθηκεύονται οι hash τιμές των δεδομένων σχετικών με τα ερωτήματα που εκτελούνται μέσω της σελίδας της εφαρμογής στη βάση PubMed ή Carre.



Εικόνα 3.1. Αρχιτεκτονική της εφαρμογής

Όπως φαίνεται και την εικόνα 3.1 με την αρχιτεκτονική της εφαρμογής ο ίδιος ο χρήστης μπορεί να επικυρώσει τα δεδομένα που έλαβε είτε μέσω της εφαρμογής είτε χωρίς τη διαμεσολάβηση της. Το δεύτερο μπορεί αν το επιτύχει καθώς η εφαρμογή δίνει το ABI των συμβολαίων και έτσι το μόνο που χρειάζεται ο χρήστης είναι να έχει στη διάθεση του έναν κόμβο για να επικοινωνεί με το blockchain ώστε να εκτελέσει τα απαραίτητα ερωτήματα.

### 3.3 Περιγραφή της λογικής των έξυπνων συμβολαίων που χρησιμοποιεί η εφαρμογή

Η εφαρμογή που αναπτύχθηκε σε αυτή τη διπλωματική χρησιμοποιεί τέσσερα διαφορετικά συμβόλαια. Το πρώτο είναι το *Migration.sol* και παρέχεται από το εργαλείο Truffle το οποίο χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής. Τα υπόλοιπα τρία δημιουργήθηκαν για να υποστηρίξουν τη λειτουργικότητα της εφαρμογής και είναι τα εξής:

- *Personal.sol*: Όταν τίθεται ένα ερώτημα στη βιοϊατρική βάση αυτή επιστρέφει μια απάντηση. Η απάντηση αυτή παραλαμβάνεται από το server της εφαρμογής και υπολογίζεται η hash τιμή της. Η τιμή αυτή αποθηκεύεται σε ένα συμβόλαιο τύπου *Personal.sol* και έτσι όταν κάποιος διαβάσει το συμβόλαιο αυτό, μέσω της

αντίστοιχης συνάρτησης που αυτό υποστηρίζει θα μάθει τη hash τιμή που είναι αποθηκευμένη σε αυτό.

- *Factoryproject.sol*: Το συμβόλαιο αυτό χρησιμοποιείται για να δημιουργηθούν και να τοποθετηθούν στο blockchain συμβόλαια τύπου *Personal.sol*. Είναι ένα μηχανισμός ώστε η τοποθέτηση καινούριων συμβολαίων στο Ethereum blockchain να γίνεται αυτόματα.
- *Project.sol*: Στη διάρκεια ανάπτυξης αυτής της διπλωματικής μελετήθηκε αν θα μπορούσε να υπάρχει ένας τρόπος μείωσης του κόστους διότι η τοποθέτηση κάθε φορά ενός καινούριου συμβολαίου ώστε να αποθηκεύεται αναλλοίωτα η hash τιμή της απάντησης έχει υψηλό χρηματικό κόστος. Έτσι προέκυψε το ενισχυμένο συμβόλαιο “Project.sol” στο οποίο αποθηκεύονται η hash τιμή όλων των ερωτημάτων που έχουν τεθεί μέσω της εφαρμογής και για κάθε ερώτηση αποθηκεύεται ένας μονοδιάστατος πίνακας που περιέχει το hash της τιμής της απάντησης που επιστράφηκε από τη βιοϊατρική βάση. Με άλλα λόγια τα περιεχόμενα του συμβολαίου μπορούν να παρομοιαστούν με ένα πίνακα που στη πρώτη στήλη του και σε κάθε γραμμή έχει το hash των ερωτημάτων των ερωτήσεων και στη δεύτερη στήλη σε κάθε γραμμή έχει ένα πίνακα με τις hash τιμές των απαντήσεων που αντιστοιχούν σε κάθε ερώτημα. Επιπλέον για κάθε hash τιμή της απάντησης αποθηκεύονται και δύο χρονικές στιγμές, μια που δείχνει πότε έγινε η εισαγωγή της απάντησης και μια που δείχνει μέχρι πότε ίσχυε ή αν ισχύει ακόμα η απάντηση αυτή. Με τον τρόπο αυτό δεν χρειάζεται να δημιουργείται και να τοποθετείται κάθε φορά ένα καινούριο συμβόλαιο στο blockchain και έτσι πρώτον μειώνεται αρκετά το κόστος, δεύτερον εξασφαλίζεται επίσης όπως και με τη περίπτωση του *Personal.sol*, η ακεραιότητα των δεδομένων και τρίτον δίνεται η δυνατότητα ελέγχου των διαφορετικών απαντήσεων που έχουν επιστραφεί για ένα ερώτημα καθώς και το χρονικό διάστημα για το οποίο ισχύει ή ισχύει η κάθε απάντηση. Συνεπώς το συμβόλαιο *Project.sol* είναι η βελτιωμένη έκδοση του απλού *Personal* συμβολαίου και ουσιαστικά είναι σαν να περιέχει πολλά συμβόλαια τύπου *Personal* με επιπλέον δυνατότητες όπως ο έλεγχος των διαφορετικών απαντήσεων σε κάθε απάντηση. Όμως στο σημείο αυτό αξίζει να σημειωθεί ότι εφαρμογή υποστηρίζει και την ύπαρξη του απλού συμβολαίου καθώς αυτό θα μπορούσε αντί για το hash της τιμής της απάντησης, να περιέχει το hash της απάντησης μαζί με κάποιο διαπιστευτήριο του χρήστη που έκανε την ερώτηση και έτσι να υπάρχει προσωπική σύνδεση ότι ο τάδε χρήστης έκανε την ερώτηση και έλαβε την συγκεκριμένη απάντηση κάτι που δεν θα μπορούσε να υποστηριχθεί από το συμβόλαιο τύπου *Project*.

### 3.4 Περιγραφή της ροής εργασίας

Η ροή εργασίας της εφαρμογής περιγράφεται ως εξής. Αρχικά ο χρήστης εισέρχεται στην σελίδα της εφαρμογής όπου έχει τη δυνατότητα είτε να υποβάλλει ένα ερώτημα είτε να κάνει επικύρωση δεδομένων που έλαβε για κάποιο ερώτημα που είχε θέσει μέσω της εφαρμογής παλιότερα. Έστω ότι αυτός επιθυμεί να θέσει ένα ερώτημα. Τότε επιλέγει τη βάση στην οποία θέλει να το θέσει (*PubMed* ή *CARRE*) και στη συνέχεια ένα από τα δύο είδη συμβολαίων που υποστηρίζει η εφαρμογή (*Απλό* ή *Ενισχυμένο*). Έπειτα το ερώτημα προωθείται στην βιοϊατρική βάση και τα αποτελέσματα μεταβιβάζονται στον server.

Ανάλογα με το είδος του συμβολαίου που επέλεξε ο χρήστης καθορίζεται και η διαδικασία που θα ακολουθηθεί από τον server. Τα σενάρια είναι τα εξής:

- Ο χρήστης επέλεξε ένα απλό συμβόλαιο. Τότε στον server υπολογίζεται η hash τιμή του κειμένου που περιλαμβάνει την απάντηση που επιστράφηκε από την ιατρική βάση και το ερώτημα που τέθηκε. Στη συνέχεια ο server επικοινωνεί μέσω του κόμβου Geth με το έξυπνο συμβόλαιο “Factoryproject” και πιο συγκεκριμένα με τη συνάρτηση “createContract” που αυτό υποστηρίζει, ώστε να δημιουργηθούν συμβόλαια για το απλό είδος που επέλεξε ο χρήστης. Το όρισμα εισόδου στη συνάρτηση αυτή είναι η hash τιμή της απάντησης που υπολογίσθηκε. Έτσι μετά την ολοκλήρωση της εκτέλεσης της συνάρτησης έχει δημιουργηθεί ένα συμβόλαιο που περιέχει μέσα του γραμμένη μόνο τη hash τιμή της απάντησης. Ταυτόχρονα ο server επικοινωνεί και με τη τοπική βάση MongoDB για την αποθήκευση των απαραίτητων στοιχείων. Επειδή τοποθετείται στο Ethereum blockchain ένα καινούριο συμβόλαιο, αυτό αποκτά μια διεύθυνση στο blockchain δίκτυο ώστε να μπορεί κάποιος να το βρει και να διαβάσει τα περιεχόμενά του. Επιπλέον επειδή με αυτή την τοποθέτηση αλλάζει η κατάσταση των δεδομένων του blockchain, θεωρείται ότι γίνεται μια συναλλαγή με αυτό και για το λόγο αυτό παράγεται από το Ethereum και παρέχεται στον τελικό χρήστη μέσω της εφαρμογής, ο αριθμός της απόδειξης της συναλλαγής. Στην απόδειξη αναγράφεται τόσο η ημερομηνία δημιουργίας της όσο και η διεύθυνση του συμβολαίου που παράχθηκε. Έτσι κάποιος μπορεί να αποδείξει αργότερα, με βάση τον αριθμό αυτόν, ότι εκτέλεσε την συγκεκριμένη χρονική στιγμή το ερώτημα και πήρε τη συγκεκριμένη απάντηση. Για να αποδείξει ότι πήρε τη συγκεκριμένη απάντηση χρειάζεται να χρησιμοποιήσει την διεύθυνση του συμβολαίου που δημιουργήθηκε και είτε μέσω της εφαρμογής είτε μόνος του ο χρήστης, να ζητήσει να διαβάσει τα περιεχόμενα του συμβολαίου που βρίσκεται στη συγκεκριμένη διεύθυνση. Τότε θα του επιστραφεί η hash τιμή, την οποία μπορεί να συγκρίνει με τα δεδομένα της απάντησης που του επιστράφηκε από την εφαρμογή όταν έθεσε το ερώτημα. Για να κάνει αυτή τη σύγκριση χρειάζεται να υπολογίσει τη hash τιμή του κειμένου που προκύπτει από το ερώτημα που έθεσε και από την απάντηση που του επιστράφηκε και στη συνέχεια να συγκρίνει τις δύο hash τιμές. Αξίζει να σημειωθεί ξανά ότι η έξοδος από τον υπολογισμό μιας hash τιμής είναι μοναδικός δηλαδή μόνο για ίδια δεδομένα εισόδου παράγεται ίδια έξοδος.

- Ο χρήστης επέλεξε ένα ενισχυμένο συμβόλαιο. Στο συμβόλαιο αυτό δεν αποθηκεύεται μόνο η hash τιμή της απάντησης που επιστράφηκε από την ιατρική βάση, αλλά ως hash τιμή της απάντησης αποθηκεύεται αυτή που προκύπτει από το κείμενο που περιέχει την απάντηση και σε αυτή έχει προστεθεί ένα χρονικό κλειδί το οποίο είναι η τρέχουσα ώρα και παρέχεται από τον server. Η χρονική αυτή ένδειξη προστίθεται ώστε να μπορούν να διαφοροποιηθούν οι απαντήσεις σε περίπτωση που μια απάντηση αλλάζει στην ιατρική βάση, πχ λόγω της επιστημονικής εξέλιξης, και έπειτα αλλάζει ξανά στην αρχική μορφή της. Ο server επικοινωνεί μέσω του κόμβου Geth με το έξυπνο συμβόλαιο “Project.sol” και ελέγχει αν το ερώτημα έχει ξαναγίνει. Αυτό το κάνει μέσω της συνάρτησης returnLatest είτε μέσω της συνάρτησης “getlength” οι οποίες επιστρέφουν 0 αν το ερώτημα δεν υπάρχει αποθηκευμένο στο συμβόλαιο.

Αν δεν έχει ξαναγίνει τότε υπολογίζεται το χρονικό κλειδί, προστίθεται στην απάντηση και τελικά υπολογίζεται η hash τιμή που θα αποθηκευτεί στο συμβόλαιο μαζί με τη hash τιμή της ερώτησης. Στη συνέχεια τα δεδομένα αυτά εισέρχονται στο συμβόλαιο και ο server επιστρέφει μέσω του front-end στο χρήστη την απάντηση όπως

αυτή επιστράφηκε από τη βιοϊατρική βάση, τη hash τιμή της απάντησης που αποθηκεύεται στο συμβόλαιο καθώς και το χρονικό κλειδί με το οποίο σχετίζεται και τη hash τιμή της ερώτησης, ώστε να μπορεί να εξακριβωθεί ότι τα στοιχεία που του επιστράφηκαν αφορούν το ερώτημα που έθεσε. Ταυτόχρονα ο server επικοινωνεί και με τη τοπική βάση MongoDB για την αποθήκευση των απαραίτητων στοιχείων.

Αν έχει ξαναγίνει, τότε ζητάει ξανά πληροφορίες από το ίδιο συμβόλαιο ώστε να μάθει το χρονικό κλειδί που είχε χρησιμοποιηθεί για τον υπολογισμό της hash τιμής της τελευταίας απάντησης που υπάρχει αποθηκευμένη για το συγκεκριμένο ερώτημα. Αφού το λάβει το προσθέτει στην απάντηση που επιστράφηκε από την βιοϊατρική βάση και υπολογίζει τη νέα hash τιμή τους. Έπειτα θα ζητήσει από το συμβόλαιο Project.sol η τελευταία hash τιμή της απάντησης που έχει αποθηκευμένη, ώστε να συγκρίνει τις δύο hash τιμές.

Αν είναι ίδιες σημαίνει ότι το hash της απάντησης περιέχει την πιο πρόσφατη έκδοση της απάντησης και συνεπώς δεν χρειάζεται να αποθηκευτεί ξανά το ίδιο hash στο συμβόλαιο, γλιτώνοντας το χρηματικό κόστος της προσθήκης ιδίων δεδομένων στο Ethereum. Στη συνέχεια ο server επιστρέφει στο front end την απάντηση όπως επιστράφηκε από τη βιοϊατρική βάση, το χρονικό κλειδί της τελευταίας απάντησης, τη hash τιμή της απάντησης που αποθηκεύτηκε στο συμβόλαιο και τη hash τιμή του ερωτήματος ώστε να μπορεί να εξακριβώσει ότι τα στοιχεία αφορούν το ερώτημα που έθεσε.

Αν δεν είναι ίδιες τότε η τιμή του hash της απάντησης δεν είναι η πιο πρόσφατη και γι' αυτό χρειάζεται η καινούρια τιμή να προστεθεί στο συμβόλαιο. Έτσι ο server αλληλεπιδρά με τη συνάρτηση "addValue" του συμβολαίου και της παρέχει το hash του ερωτήματος και την τελευταία έκδοση της hash τιμής της απάντησης, η οποία είναι η νέα hash τιμή που υπολογίστηκε. Έτσι με βάση τη hash τιμή του ερωτήματος το συμβόλαιο βρίσκει τα δεδομένα που έχει αποθηκεύσει για αυτή την ερώτηση και προσθέτει την νέα απάντηση. Έπειτα ο server επιστρέφει στο χρήστη την απάντηση όπως αυτή επιστράφηκε από τη βιοϊατρική βάση, τη hash τιμή της απάντησης που αποθηκεύεται στο συμβόλαιο καθώς και το χρονικό κλειδί με το οποίο σχετίζεται και τη hash τιμή της ερώτησης ώστε να μπορεί να εξακριβωθεί ότι τα στοιχεία που του επιστράφηκαν αφορούν το ερώτημα που έθεσε. Ταυτόχρονα ο server επικοινωνεί και με τη τοπική βάση MongoDB για την αποθήκευση των απαραίτητων στοιχείων.

Έστω ότι ο χρήστης έχει κάνει μια ερώτηση στο παρελθόν και εισέρχεται στην σελίδα της εφαρμογής για να κάνει επικύρωση των δεδομένων που έχει λάβει. Η επικύρωση μπορεί να γίνει:

1. Για το βασικό συμβόλαιο. Ο χρήστης χρειάζεται να παρέχει στο αντίστοιχο πεδίο στη σελίδα της εφαρμογής την διεύθυνση του συμβολαίου που του είχε επιστραφεί όταν έθεσε το ερώτημα. Στη συνέχεια ο server θα λάβει την αίτηση, θα επικοινωνήσει μέσω του Geth με το συμβόλαιο που βρίσκεται στη συγκεκριμένη θέση και θα επιστρέψει τα περιεχόμενα που βρίσκονται αποθηκευμένα σε αυτό, δηλαδή την hash τιμή της απάντησης. Έτσι ο χρήστης μπορεί να συγκρίνει το hash αυτό της απάντησης με τα δεδομένα που είχε λάβει για να διαπιστώσει αν είναι ίδια. Συνεπώς έχοντας στην κατοχή του ο χρήστης τον αριθμό της απόδειξης που του είχε επιστραφεί και η οποία αναγράφει και τη διεύθυνση του συμβολαίου μπορεί να αποδείξει ότι τη συγκεκριμένη χρονική στιγμή που αναγράφεται και στην απόδειξη έλαβε τα συγκεκριμένα δεδομένα από τη βάση.

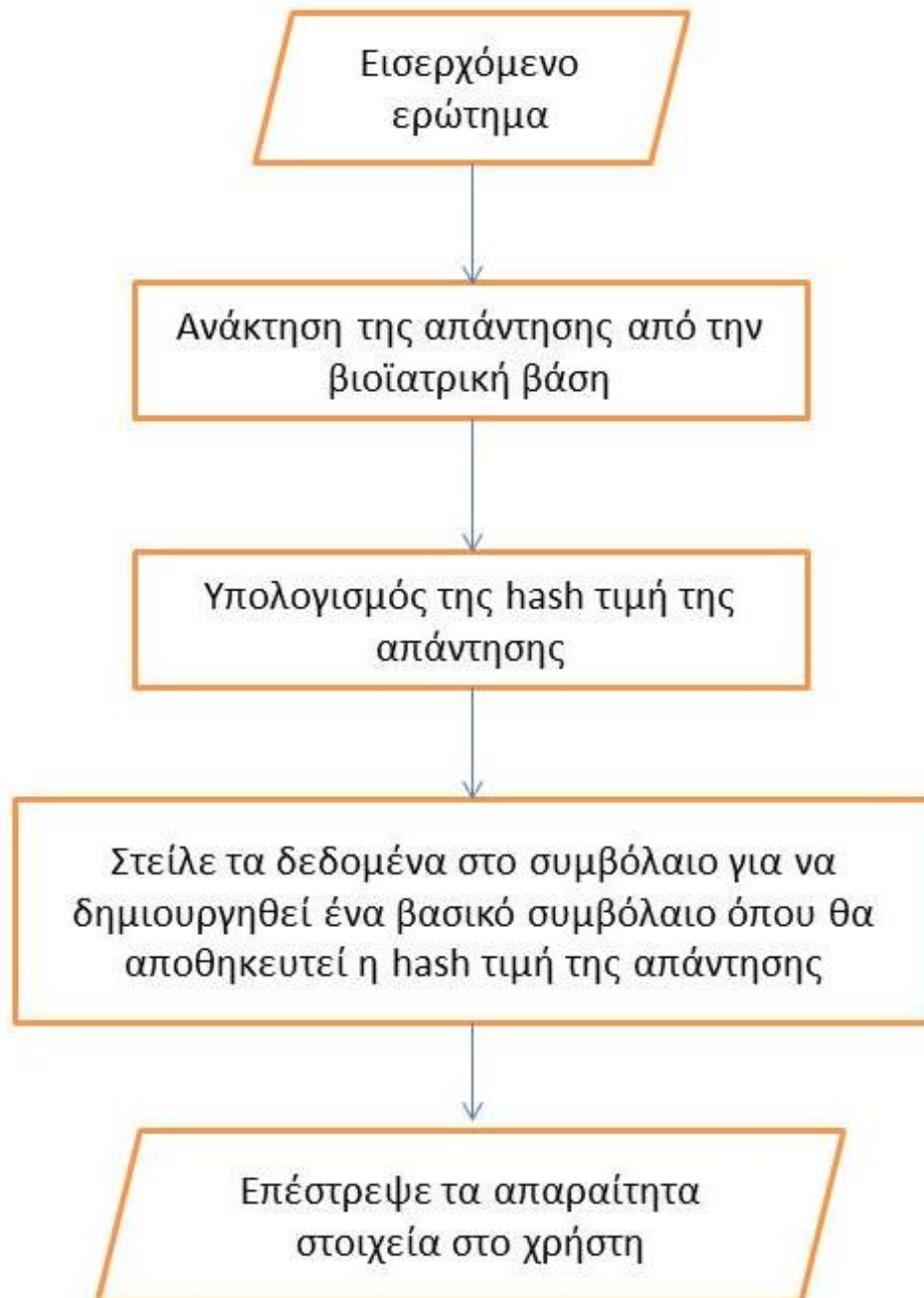


2. Για το ενισχυμένο συμβόλαιο. Γι' αυτό παρέχονται τρεις συναρτήσεις επικύρωσης και για κάθε μια υπάρχει το αντίστοιχο πεδίο στη σελίδα της εφαρμογής, το οποίο πρέπει να συμπληρώσει ο χρήστης ανάλογα με το είδος της επικύρωσης που επιθυμεί να κάνει. Αν επιθυμεί να μάθει την τελευταία τιμή της απάντησης που υπάρχει αποθηκευμένη για ένα ερώτημα στο συμβόλαιο, τότε θα συμπληρώσει το αντίστοιχο πεδίο παρέχοντας το ερώτημα που τον ενδιαφέρει και τη βάση στην οποία το υπέβαλε. Το αίτημα θα το λάβει ο server και θα επικοινωνήσει μέσω του Geth με το συμβόλαιο και τη συνάρτηση `returnLatest` και έπειτα θα επιστρέψει τα ανάλογα αποτελέσματα στο front-end. Έτσι ο χρήστης μπορεί να μάθει αν έχει την τελευταία έκδοση. Αν επιθυμεί να μάθει για μια συγκεκριμένη hash τιμή απάντησης που αφορά ένα ερώτημα τότε έγινε η εισαγωγή της στο συμβόλαιο και μέχρι τότε ίσχυε ή αν ακόμα ισχύει, δηλαδή αν ακόμα είναι η πιο πρόσφατη απάντηση τότε θα πρέπει να συμπληρώσει το αντίστοιχο πεδίο παρέχοντας το ερώτημα που έκανε, τη βάση που το έκανε και τη hash τιμή της απάντησης που τον ενδιαφέρει. Στη συνέχεια ο server θα επικοινωνήσει μέσω του Geth με το συμβόλαιο και τη συνάρτηση `returntime` και θα επιστρέψει τα αποτελέσματα στο front end. Έτσι ο χρήστης μπορεί να αποδείξει ότι έλαβε την συγκεκριμένη απάντηση που σχετίζεται μονοσήμαντα με αυτό το hash, όταν έθεσε το συγκεκριμένο ερώτημα μέσα στο στο χρονικό διάστημα που υποδεικνύει το συμβόλαιο ότι βρισκόταν σε ισχύ η απάντηση. Τέλος αν επιθυμεί να δει όλα τα hash των απαντήσεων που υπάρχουν αποθηκευμένα για ένα συγκεκριμένο ερώτημα αρκεί να συμπληρώσει το αντίστοιχο πεδίο παρέχοντας το ερώτημα που τον ενδιαφέρει και τη βάση στην οποία είχε εκτελεστεί. Έπειτα το client side μέρος της εφαρμογής θα ζητήσει αρχικά να μάθει το μέγεθος του πίνακα με τις απαντήσεις, το αίτημα θα το λάβει ο server, θα επικοινωνήσει μέσω του Geth με το συμβόλαιο και τη συνάρτηση `getLength` και θα μεταφέρει τα αποτελέσματα στο front end το οποίο θα ορίσει κατάλληλα τους δείκτες και τις επαναλήψεις, ώστε να επιστραφούν όλα τα στοιχεία του πίνακα. Για κάθε επανάληψη θα δίνονται οι κατάλληλοι δείκτες και μέσω του Geth ο server θα παίρνει και θα επιστρέφει τα αποτελέσματα στο χρήστη. Έτσι αυτός θα μπορεί να διαπιστώσει αν η απάντηση που έχει λάβει αντιστοιχεί στο συγκεκριμένο ερώτημα και επίσης ένας ερευνητής θα μπορούσε να μελετήσει τη συχνότητα μεταβολής των δεδομένων που υπάρχουν στη βιοϊατρική βάση και σχετίζονται με ένα συγκεκριμένο ερώτημα.

Σε όλες τις περιπτώσεις που περιγράφηκαν τα δεδομένα που επιστρέφονται στον χρήστη θα πρέπει να είναι ψηφιακά υπογεγραμμένα από τον ιδιοκτήτη της εφαρμογής. Στην συγκεκριμένη περίπτωση θα πρέπει ο ιδιοκτήτης να έχει δύο κλειδιά ένα για το CARRE και ένα για το PubMed, τα οποία θα χρησιμοποιεί αντίστοιχα για να υπογράψει δεδομένα που σχετίζονται με την αντίστοιχη βάση. Τα κλειδιά αυτά θα έχουν εκδοθεί από μια ψηφιακή αρχή πιστοποίησης ύστερα από σχετικό αίτημα των βάσεων, του CARRE και του PubMed αντίστοιχα. Συνεπώς τα κλειδιά που χρησιμοποιεί η εφαρμογή θα είναι πιστοποιημένα ότι ανήκουν αντίστοιχα στις δύο αυτές βάσεις και έτσι θα επιτυγχάνεται η αδυναμία αποποίησης της προέλευσης των δεδομένων. Επίσης το δημόσιο κλειδί που χρησιμοποιείται από την εφαρμογή για την εκτέλεση των συναλλαγών στο δίκτυο του blockchain θα πρέπει επίσης να είναι πιστοποιημένο ότι ανήκει στον ιδιοκτήτη της εφαρμογής.

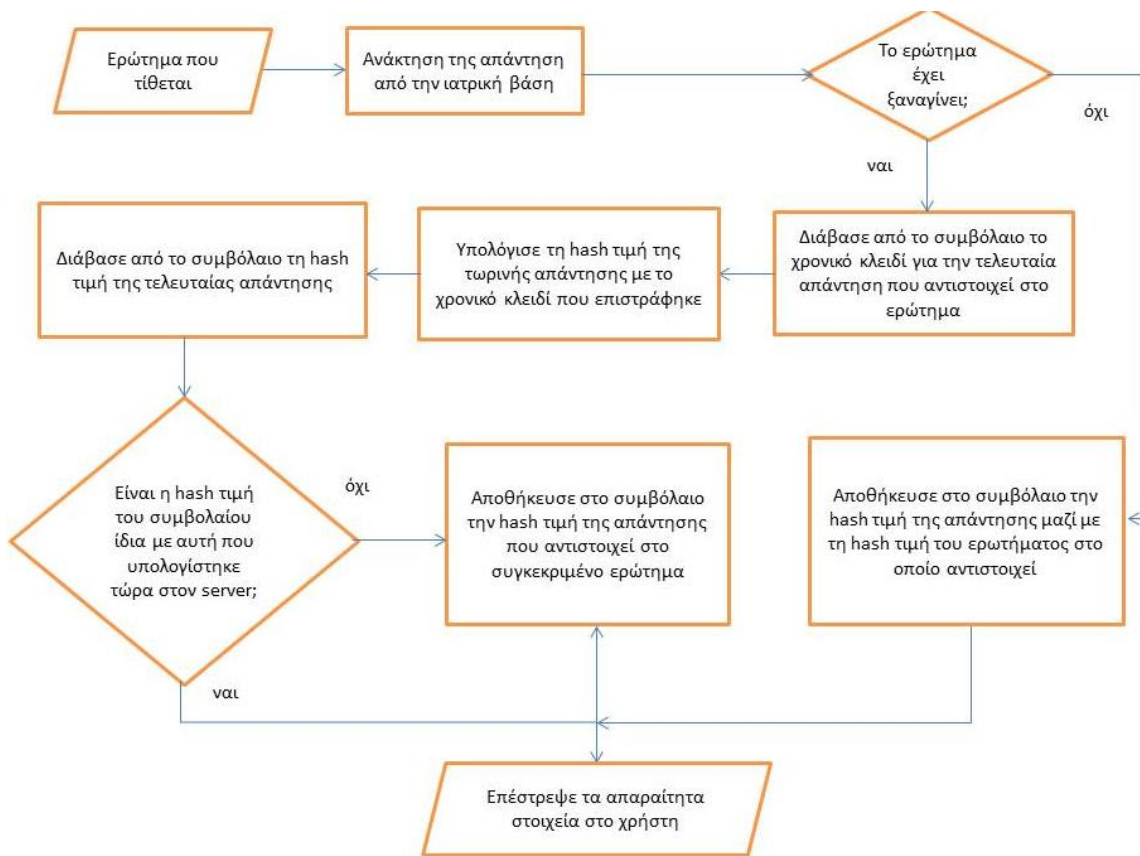
Ένας εναλλακτικός τρόπος θα ήταν ο ιδιοκτήτης της εφαρμογής να χρησιμοποιούσε τα ίδια ζευγάρια δημόσιων και ιδιωτικών κλειδιών και για τις συναλλαγές στο δίκτυο του Ethereum blockchain και για να υπογράψει ψηφιακά τα

δεδομένα που παρέχει στον χρήστη. Αυτός ο τρόπος θα απαιτούσε ο ιδιοκτήτης της εφαρμογής να είχε δημιουργήσει δύο δημόσια κλειδιά στη πλατφόρμα του Ethereum των οποίων τις αντίστοιχες διευθύνσεις θα όριζε ως διαχειριστές, για να μπορεί να εκτελεί αντίστοιχα συναλλαγές που αφορούν το PubMed και το CARRE, οι οποίοι θα είχαν πιστοποιήσει μέσω μιας αρχής πιστοποίησης τη σχέση τους με αυτά τα κλειδιά.



**Εικόνα 3. 2.** Η ροή εργασίας για την υποβολή ενός ερωτήματος όπου επιλέχθηκε το βασικό συμβόλαιο





**Εικόνα 3.3.** Η ροή εργασίας για την υποβολή ενός ερωτήματος όπου επιλέχθηκε το ενισχυμένο συμβόλαιο

### 3.5 Αναλυτική παρουσίαση του κώδικα των συμβολαίων

Όπως έχει ήδη αναφερθεί η εφαρμογή χρησιμοποιεί κάποια έξυπνα συμβόλαια (smart contracts) όπου αποθηκεύονται δεδομένα σχετικά με τα ερωτήματα που εκτελούνται μέσω της σελίδας της στη βάση PubMed ή Carre. Τα συμβόλαια έχουν συνταχθεί σε γλώσσα προγραμματισμού solidity και είναι τα εξής:

- *Migrations.sol* [44]. Είναι ένα contract που χρησιμοποιείται για να μπορεί να είναι διαθέσιμη στον developer η λειτουργία των αρχείων “migration” που παρέχει ένα truffle project. Τα αρχεία στο φάκελο migration ενός truffle project είναι αρχεία σε γλώσσα Javascript τα οποία εκτελούνται με την εντολή “truffle migrate”. Τα αρχεία αυτά σε συνδυασμό με το συμβόλαιο “Migration.sol” κρατάνε ένα μέτρημα ώστε εάν χρειασθεί να χρησιμοποιηθούν αργότερα και άλλα συμβόλαια σε ένα project, να διευκολύνεται η τοποθέτησή τους στο blockchain δίκτυο χωρίς να είναι απαραίτητη η τοποθέτηση ξανά στο δίκτυο και των συμβολαίων που ήδη έχουν τοποθετηθεί και χρησιμοποιούνται από την εφαρμογή του project.
- *Project.sol*: Είναι το συμβόλαιο που χρησιμοποιείται όταν ο χρήστης επιλέγει στο front-end της εφαρμογής, ένα ενισχυμένο συμβόλαιο για να υποστηρίξει το ερώτημα που θέτει στην βάση PubMed ή Carre. Ο κώδικας του συμβολαίου είναι ο εξής:

<pre> pragma solidity ^0.4.23;  contract Project{     mapping (bytes32=&gt;HashStruct) query;     //the address of the administrator     address private owner=0xA97467bB87D1512dff5c3B69AF9A3E0274fcF20;      struct HashStruct{         bytes32[] arrayofhash;         mapping (bytes32=&gt;uint256) timestamps;         mapping (bytes32=&gt;uint256) timestampsEnd;         mapping (bytes32=&gt;uint256) timeKey;     }      function addValue(bytes32 _queryhash,bytes32 _answerhash,uint256 _timekey)public{         //only the administrator can add new values         require(msg.sender==owner);         require(query[_queryhash].timestamps[_answerhash]==0);         if(query[_queryhash].arrayofhash.length&gt;0){             bytes32 last_answer = query[_queryhash].arrayofhash[query[_queryhash].arrayofhash.length-1];             query[_queryhash].timestampsEnd[last_answer]=now;         }         query[_queryhash].arrayofhash.push(_answerhash);         query[_queryhash].timestamps[_answerhash]=now;         query[_queryhash].timeKey[_answerhash]=_timekey;     }      function returnLatest(bytes32 _queryhash)view public returns(bytes32){         if(query[_queryhash].arrayofhash.length==0)         {             return 0;         }         return query[_queryhash].arrayofhash[query[_queryhash].arrayofhash.length- 1];     }      function returntime(bytes32 _queryhash,bytes32 _answerhash)view public returns(uint256 _from, uint256 _since){         return (query[_queryhash].timestamps[_answerhash],query[_queryhash].timestampsEnd[_an swerhash]);     } </pre>
---

```

function getAll(bytes32 _queryhash,uint256 _cursor,uint256 _amount)view public
returns(bytes32[] values){
    uint256 length = _amount;
    if (length > query[_queryhash].arrayofhash.length - _cursor) {
        length = query[_queryhash].arrayofhash.length - _cursor;
    }
    values = new bytes32[](length);
    for (uint256 i = 0; i < length; i++) {
        values[i] = query[_queryhash].arrayofhash[_cursor + i];
    }
    return values;
}

function getLength(bytes32 _queryhash)view public returns(uint256){
    return query[_queryhash].arrayofhash.length;
}

function getTimekey(bytes32 _queryhash)view public returns(uint256){
    require(query[_queryhash].arrayofhash.length>0);
    return
    query[_queryhash].timeKey[query[_queryhash].arrayofhash[query[_queryhash].arrayo
fhash.length-1]];
}
}

```

#### Σχόλια για τον κώδικα που χρησιμοποιεί το συμβόλαιο

Το συμβόλαιο χρησιμοποιεί την μεταβλητή owner που είναι τύπου address και ορίζει ποιος θα είναι ο διαχειριστής του συμβολαίου και ο οποίος είναι ο μόνος που μπορεί να κάνει χρήση της συνάρτησης addValue του συμβολαίου ώστε να εισαχθούν νέα στοιχεία στο συμβόλαιο. Αυτό επιτυγχάνεται με την εντολή “require(msg.sender==owner);” Επίσης στο συμβόλαιο ορίζεται και η μεταβλητή “query” η οποία μπορεί να παρομοιασθεί με ένα πίνακα που στην πρώτη στήλη βρίσκονται τιμές hash 256-bits και οι οποίες σχετίζονται με τα ερωτήματα που θέτονται προς την βάση. Για κάθε τιμή της στήλης αντιστοιχεί μια γραμμή που περιέχει τις τιμές των hash 256-bit των απαντήσεων που λήφθηκαν από την βάση όπου τέθηκε το ερώτημα. Κάθε φορά που τίθεται το ίδιο ερώτημα στην ίδια βάση (PubMed ή Carre) και λαμβάνεται καινούρια απάντηση, η hash τιμή της απάντησης αυτής εισάγεται στον πίνακα “query” στο τέλος της γραμμής η οποία αφορά το συγκεκριμένο ερώτημα. Επιπλέον κάθε απάντηση που αποθηκεύεται στο συμβόλαιο έχει τρεις δείκτες. Ο πρώτος δείχνει σε μια μεταβλητή “timestamps” η οποία αποθηκεύει την ημερομηνία που έγινε η εισαγωγή της hash τιμής της απάντησης στο συμβόλαιο, ο δεύτερος δείχνει σε μια μεταβλητή “timestampsEnd” που αποθηκεύει μέχρι πότε ίσχυε αυτή η απάντηση και ο τρίτος δείχνει σε μια μεταβλητή “timeKey ” που είναι το χρονικό κλειδί που χρησιμοποιεί ο server για να αποθηκεύσει μια απάντηση. Και οι τρεις αυτές μεταβλητές αρχικά έχουν την τιμή 0 και η τιμή τους αρχικοποιείται είτε με την εισαγωγή της απάντησης με την οποία σχετίζονται, αυτό αφορά τις μεταβλητές

“timestamps” και “timeKey” είτε όταν εισάγεται η πιο καινούρια απάντηση σε σχέση με την απάντηση που σχετίζονται, αυτό αφορά την μεταβλητή “timestampsEnd”.

Το συμβόλαιο έχει μια δομή (struct) και τις εξής έξι συναρτήσεις:

- Τη δομή *HashStruct* η οποία αφορά κάθε ερώτημα. Έτσι σε αυτή περιγράφεται ότι για κάθε ερώτημα αποθηκεύεται ένας πίνακας, με όνομα μεταβλητής *arrayofhash*, με τα hash των απαντήσεων, και για κάθε τιμή hash μιας απάντησης αντιστοιχούν τρεις μεταβλητές: *timestamps*, *timestampsEnd* και *timeKey*.
- Τη συνάρτηση *addValue* η οποία δέχεται ως ορίσματα εισόδου δύο τιμές τύπου *bytes32* που αντιστοιχούν σε δύο hash τιμές, μια που σχετίζεται με την ερώτηση που τέθηκε στη βάση και μια που σχετίζεται με την απάντηση που λήφθηκε από την βάση, και μια αριθμητική τιμή που αφορά το timestamp από τον server και είναι τύπου *uint256*. Η συνάρτηση αυτή μπορεί να εκτελεσθεί μόνο από τον διαχειριστή και εισάγει νέα στοιχεία στη μεταβλητή “query”.
- Τη συνάρτηση *returnLatest* που έχει ως όρισμα εισόδου μια τιμή τύπου *bytes32* που είναι η hash τιμή του ερωτήματος και η οποία συνάρτηση όταν εκτελεσθεί επιστρέφει είτε την τιμή 0, αν το ερώτημα που δόθηκε ως όρισμα εισόδου δεν υπάρχει αποθηκευμένο στο συμβόλαιο, είτε το hash της τελευταίας απάντησης που είναι αποθηκευμένη στο συμβόλαιο και αφορά το ερώτημα που δόθηκε ως όρισμα εισόδου. Στη θέση του τμήματος του κώδικα που ελέγχει αν το ερώτημα υπάρχει ώστε να επιστρέψει 0 θα μπορούσαμε να είχαμε τοποθετήσει μια εντολή `require(query[_queryhash].arrayofhash.length>0)` ώστε μόνο αν υπάρχει το ερώτημα να επιστραφεί η απάντηση. Επιλέχθηκε ο πρώτος τρόπος διότι αν η συνθήκη της εντολής `require` δεν ικανοποιούνταν τότε δεν θα επιστρεφόταν απάντηση και η συνάρτηση θα επέστρεφε ότι απέτυχε να εκτελεστεί χωρίς να είμαστε σίγουροι ότι ο λόγος που απέτυχε να εκτελεστεί είναι επειδή το ερώτημα δεν υπάρχει.
- Τη συνάρτηση *returntime* που έχει ως ορίσματα εισόδου δύο hash τιμές, μια που σχετίζεται με την ερώτηση που τέθηκε στη βάση και μια που σχετίζεται με την απάντηση που λήφθηκε από την βάση σε συνδυασμό με το χρονικό κλειδί από τον server. Η συνάρτηση αυτή όταν καλεστεί επιστρέφει δύο τιμές τύπου *uint256* τις *\_from* και *\_since*. Οι τιμές αυτές σχετίζονται με τη hash τιμή της απάντησης που δόθηκε ως όρισμα εισόδου και είναι η απάντηση στο ερώτημα που δόθηκε ως όρισμα εισόδου. Η πρώτη αφορά την ημερομηνία που εισήχθη η hash τιμή της απάντησης αυτής και η δεύτερη δείχνει αν αυτή ισχύει ακόμα ή μέχρι πότε ίσχυε. Αν η μεταβλητή *timestampsEnd* είναι ίση με το μηδέν τότε, όπως επισημάνθηκε και προηγουμένως, η hash τιμή της απάντησης ισχύει ακόμα και δεν έχει εισαχθεί πιο καινούρια απάντηση για το συγκεκριμένο ερώτημα.
- Τη συνάρτηση *getAll* που έχει ως όρισμα εισόδου τρεις μεταβλητές. Η πρώτη είναι τύπου *bytes32* και αντιστοιχεί στην hash τιμή του ερωτήματος, η δεύτερη και η τρίτη είναι τύπου *uint256* και αφορούν αντίστοιχα τη τιμή ενός κέρσους και την ποσότητα των στοιχείων που επιθυμούμε να μας επιστραφούν. Η συνάρτηση *getAll* όταν εκτελεσθεί επιστρέφει ένα πίνακα με τις hash τιμές των απαντήσεων που έχουν εισαχθεί στο συμβόλαιο και αφορούν το συγκεκριμένο ερώτημα. Επειδή για κάθε απάντηση μπορεί να αποθηκευτούν

πολλές απαντήσεις δεν θα ήταν σκόπιμο να επιστραφούν όλες οι απαντήσεις μαζί, καθώς μια τέτοια εκτέλεση λόγω του μεγέθους του πίνακα θα μπορούσε να οδηγήσει στο μπλοκάρισμα της εφαρμογής. Για το λόγο αυτό επιλέχθηκε να επιστρέφονται λίγα στοιχεία κάθε φορά με σημείο εκκίνησης το στοιχείο που δείχνει η μεταβλητή εισόδου που αφορά τον κέρσορα. Ο αριθμός των στοιχείων που επιστρέφονται δίνεται από τη μεταβλητή εισόδου που αφορά την ποσότητα. Αν η τιμή των στοιχείων που επιθυμούμε να επιστραφούν είναι μεγαλύτερο από το πλήθος των στοιχείων τότε η συνάρτηση επιστρέφει λιγότερα στοιχεία από αυτά που δείχνει η μεταβλητή της ποσότητας και είναι όσα στοιχεία υπάρχουν. Επειδή στο κομμάτι της εφαρμογής έχει επιλεγεί να επιστρέφονται στον χρήστη όλα τα στοιχεία το front-end κομμάτι της εφαρμογής αναλαμβάνει να δώσει κατάλληλες τιμές στις μεταβλητές εισόδου που αφορούν τον κέρσορα και την ποσότητα ώστε να φέρνονται λίγα λίγα τα στοιχεία και τελικά ο χρήστης να μπορεί να δει όλα τα στοιχεία του πίνακα, δηλαδή όλες τις απαντήσεις που αφορούν το συγκεκριμένο ερώτημα.

- Τη συνάρτηση *getLength* που έχει ως όρισμα εισόδου μια μεταβλητή τύπου `bytes32` η οποία αφορά τη hash τιμή του ερωτήματος. Η συνάρτηση αυτή όταν καλεστεί επιστρέφει μήκος του πίνακα με τις hash τιμές των απαντήσεων που αντιστοιχούν στην ερώτηση που δόθηκε ως όρισμα εισόδου, δηλαδή επιστρέφει το πλήθος των διαφορετικών απαντήσεων που έχουν εισαχθεί για το συγκεκριμένο ερώτημα.
- Τη συνάρτηση *getTimekey* που έχει ως όρισμα εισόδου μια μεταβλητή τύπου `bytes32` η οποία αφορά τη hash τιμή του ερωτήματος. Η συνάρτηση αυτή όταν εκτελεσθεί επιστρέφει τη χρονική στιγμή που είχε δοθεί από το server μαζί με την καταχώρηση μιας απάντησης και η οποία αντιστοιχεί στην τιμή που αναφέρθηκε νωρίτερα ως `timeKey`. Τη συνάρτηση αυτή την χρησιμοποιεί το back-end τμήμα της εφαρμογής για να κάνει τους κατάλληλους ελέγχους ώστε να μην εισαχθούν στο συμβόλαιο δυο συνεχόμενα ίδιες απαντήσεις. Με άλλα λόγια πριν εισαχθεί μια απάντηση ελέγχεται πρώτα αν η τελευταία απάντηση που επιστράφηκε από την ιατρική βάση για το συγκεκριμένο ερώτημα είναι ίδια με την τελευταία απάντηση που υπάρχει αποθηκευμένη στο συμβόλαιο. Για να μπορεί να γίνει η σύγκριση των δύο απαντήσεων χρειάζεται το χρονικό κλειδί που χρησιμοποιήθηκε για την αποθήκευση του ερωτήματος. Έτσι με το χρονικό κλειδί και την απάντηση από την ιατρική βάση μπορεί να παραχθεί η hash τιμή της απάντησης και να συγκριθεί με αυτή που βρίσκεται στο συμβόλαιο, ως πιο πρόσφατη απάντηση σε ένα συγκεκριμένο ερώτημα.

*Factoryproject.sol*: Είναι το συμβόλαιο που χρησιμοποιείται όταν ο χρήστης επιλέγει στο front-end της εφαρμογής, ένα βασικό συμβόλαιο για να υποστηρίξει το ερώτημα που θέτει στην βάση PubMed ή Carre. Όταν επιλέγεται ένα συμβόλαιο βασικού τύπου πρέπει να τοποθετηθεί στο blockchain ένα νέο συμβόλαιο που έχει τη μορφή του “Personal.sol” που περιγράφεται παρακάτω. Για να γίνει αυτόματα η τοποθέτηση του καινούριου συμβολαίου στο blockchain χρησιμοποιείται το συμβόλαιο *Factoryproject* το οποίο παρέχει τη συνάρτηση “*createContract*” η οποία τοποθετεί στο blockchain δίκτυο το νέο συμβόλαιο με αποθηκευμένα τα απαραίτητα στοιχεία που σχετίζονται με το ερώτημα. Ο κώδικας του συμβολαίου είναι ο εξής:

```

pragma solidity ^0.4.23;
import "./Personal.sol";

contract Factoryproject {
    address private owner=0xA97467bB87D1512dff5c3B69AF9A3E0274fcF20;
    address[] public addresses_contracts;
    mapping (address => bool) addressExist;

    event LogNewContract(address newContract);

    function createContract(bytes32 _data) public{
        require(msg.sender==owner);
        address c = new Personal(_data);
        addresses_contracts.push(c);
        addressExist[c] = true;
        emit LogNewContract(c);
    }

    function getData(address _contract) view public returns(bytes32 specificdata){
        if(addressExist[_contract]==false){
            return 0;
        }else{
            return (Personal(_contract).getdata());
        }
    }
}

```

#### Σχόλια για τον κώδικα που χρησιμοποιεί το συμβόλαιο

Το συμβόλαιο χρησιμοποιεί την μεταβλητή owner που είναι τύπου address και ορίζει ποιος θα είναι ο διαχειριστής του συμβολαίου και ο οποίος είναι ο μόνος που μπορεί να κάνει χρήση της συνάρτησης createContract του συμβολαίου ώστε να δημιουργηθεί ένα νέο συμβόλαιο τύπου Personal.sol. Αυτό επιτυγχάνεται με την εντολή “require(msg.sender==owner);”. Επίσης στο συμβόλαιο ορίζεται και η μεταβλητή “addresses\_contracts” που είναι ένας πίνακας με όλες τις διευθύνσεις των συμβολαίων που έχουν δημιουργηθεί μέσω της συνάρτησης createContract. Η μεταβλητή αυτή ορίστηκε σε public ώστε να μπορεί κάποιος να αλληλεπιδράσει με το συμβόλαιο και να μάθει τις τιμές που είναι αποθηκευμένες σε αυτόν τον πίνακα, δηλαδή ποιες διευθύνσεις συμβολαίων έχουν δημιουργηθεί κάτι που ενδεχομένως χρειαστεί για κάποιον έλεγχο. Ακόμα ορίζεται η μεταβλητή “addressExist” που αντιστοιχεί σε ένα “εγχειρίδιο” όπου για κάθε διεύθυνση που ρωτάμε μας δείχνει μια bool τιμή η οποία είναι true αν η διεύθυνση που μας ενδιαφέρει αντιστοιχεί σε διεύθυνση συμβολαίου που δημιουργήθηκε από τη συνάρτηση createContract αλλιώς είναι false. Επιπλέον το συμβόλαιο εκπέμπει ένα “event” που μας δείχνει την διεύθυνση που απέκτησε το συμβόλαιο που τοποθετήθηκε στο blockchain όταν εκτελέστηκε η συνάρτηση createContract από τον διαχειριστή. Επειδή η συνάρτηση createContract προκαλεί αλλαγές στο blockchain και στις μεταβλητές του ίδιου του συμβολαίου, γι’ αυτό, όταν εκτελείται, απαιτούνται



κάποια “ether” και παράγεται ένα transaction receipt, που ουσιαστικά αποτελεί την απόδειξη αυτής της συναλλαγής. Σε αυτό το transaction receipt αναγράφεται και το event με τα δεδομένα που αυτό εξέπεμψε, δηλαδή τη διεύθυνση του καινούριου συμβολαίου που τοποθετήθηκε στο blockchain.

Το συμβόλαιο χρησιμοποιεί ένα event και τις εξής δύο συναρτήσεις:

- Το event *LogNewContract* που όταν ενεργοποιηθεί με την εκτέλεση της συνάρτησης *createContract* εκπέμπει μια μεταβλητή τύπου *address* η οποία είναι η διεύθυνση του συμβολαίου που δημιουργήθηκε.
- Τη συνάρτηση *createContract* η οποία δέχεται ως όρισμα εισόδου μια τιμή τύπου *bytes32* που αντιστοιχεί στην hash τιμή της απάντησης του ερωτήματος που έλαβε η εφαρμογή όταν έθεσε την ερώτηση του χρήστη στη βάση PubMed ή Carre. Η συνάρτηση αυτή μπορεί να εκτελεσθεί μόνο από τον διαχειριστή και τοποθετεί ένα συμβόλαιο τύπου *Personal.sol* με αποθηκευμένη μέσα σε αυτό το συμβόλαιο την τιμή του hash που δόθηκε ως όρισμα εισόδου, προσθέτει στο τέλος του πίνακα *addresses\_contracts* την τιμή της διεύθυνσης του συμβολαίου που μόλις τοποθετήθηκε στο blockchain, θέτει την *bool* τιμή που αντιστοιχεί σε αυτή τη διεύθυνση σε *true* και εκπέμπει το event το οποίο όπως περιγράφηκε παραπάνω είναι ορατό στο transaction receipt της συναλλαγής.
- Τη συνάρτηση *getData* που έχει ως όρισμα εισόδου μια μεταβλητή τύπου *address* που αφορά μια διεύθυνση. Η συνάρτηση αυτή όταν εκτελεσθεί ελέγχει αν η διεύθυνση αυτή αντιστοιχεί σε κάποιο συμβόλαιο που δημιουργήθηκε από την εκτέλεση της συνάρτησης *createContract* και επιστρέφει την τιμή 0 αν δεν έχει δημιουργηθεί αλλιώς επικοινωνεί με το συμβόλαιο που βρίσκεται σε αυτή τη διεύθυνση, παίρνει τη hash τιμή της απάντησης που είχε αποθηκευτεί σε αυτό κατά την τοποθέτηση του συμβολαίου στο blockchain και την επιστρέφει ως μεταβλητή εξόδου της συνάρτησης. Όπως και με τη συνάρτηση *getAll* του συμβολαίου *Project.sol* έτσι και εδώ χρησιμοποιήθηκε έλεγχος με *if* και όχι κάποια εντολή *require* για να υπάρχει καλύτερος έλεγχος αν η συνάρτηση αποτύχει να εκτελεστεί.

*Personal.sol*: Είναι το συμβόλαιο που δημιουργείται από την εκτέλεση της συνάρτησης *createContract* του συμβολαίου *Factoryproject.sol*. Ο κώδικας του συμβολαίου είναι ο εξής:

```
pragma solidity ^0.4.23;
contract Personal{
    bytes32 hashValue;

    constructor(bytes32 _hashValue) public{
        hashValue=_hashValue;
    }

    function getdata()view public returns(bytes32 _data){
        return hashValue;
    }
}
```

### Σχόλια για τον κώδικα που χρησιμοποιεί το συμβόλαιο

Το συμβόλαιο χρησιμοποιεί την μεταβλητή `hashValue` που αποθηκεύει την hash τιμή της απάντησης του ερωτήματος που έλαβε η εφαρμογή όταν έθεσε την ερώτηση του χρήστη, οποίος είχε επιλέξει ένα Personal contract, στη βάση PubMed ή Carre.

Το συμβόλαιο έχει τις εξής δύο συναρτήσεις:

- *Τη συνάρτηση `constructor` που εκτελείται όταν τοποθετείται το συμβόλαιο στο blockchain. Όταν η συνάρτηση `createContract` του συμβολαίου `Factoryproject.sol` εκτελεσθεί, δημιουργεί ένα συμβόλαιο `Personal.sol` με όρισμα εισόδου μια τιμή τύπου `bytes32` που αφορά το hash της απάντησης από τον server. Αυτό το όρισμα εισόδου εισέρχεται ως ορίσματα εισόδου στην συνάρτηση `constructor` του συμβολαίου και αποθηκεύεται στην μεταβλητή `hashValue`. Έτσι μετά την εκτέλεση της εντολής `createContract` του συμβολαίου `Factoryproject.sol` τοποθετείται στο blockchain ένα συμβόλαιο `Personal.sol` με αποθηκευμένη στη τιμή της μεταβλητής του `hashValue` την hash τιμή της απάντησης που δόθηκε ως όρισμα εισόδου στη συνάρτηση `constructor`.*
- *Τη συνάρτηση `getData` που δεν έχει ορίσματα εισόδου και επιστρέφει την τιμή της μεταβλητής `hashValue` που είναι αποθηκευμένη στο συμβόλαιο. Η συνάρτηση `getData` είναι αυτή που καλείται όταν εκτελείται από τη συνάρτηση `getData` με όρισμα εισόδου μια διεύθυνση που δημιουργήθηκε μέσω της συνάρτησης `createContract`. Τόσο η συνάρτηση `getData` όσο και η `createContract` αναφέρονται στις συναρτήσεις του συμβολαίου `Factoryproject.sol` που περιγράφηκε παραπάνω.*

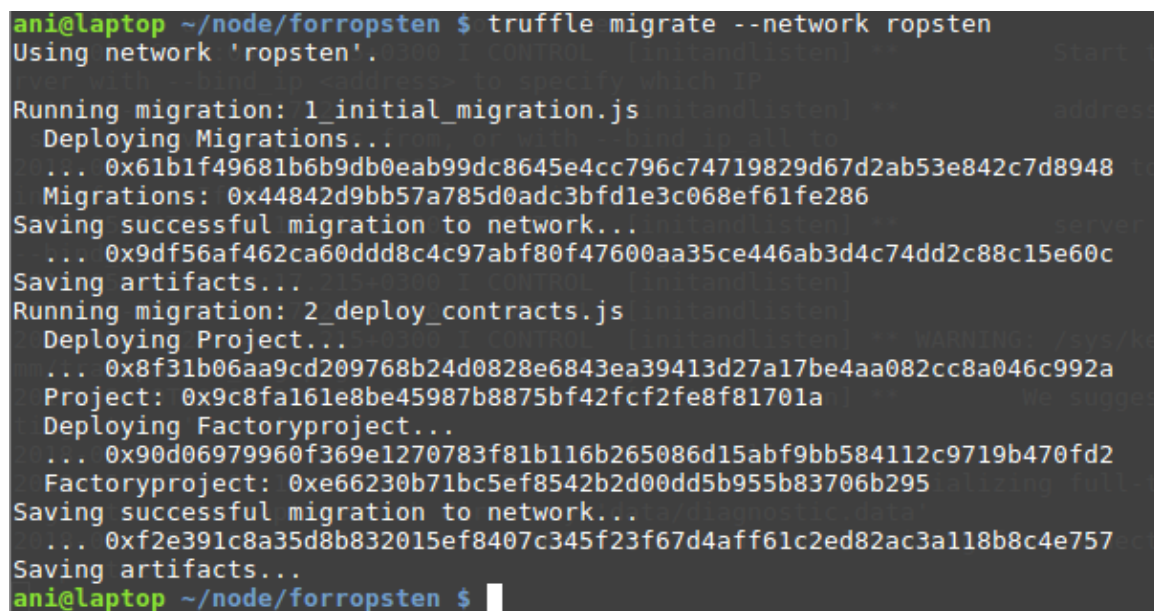
Οι hash τιμές που αποθηκεύονται στα συμβόλαια αντιστοιχούν σε τιμές hash που παράχθηκαν με SHA-256 και αναπαριστούνται σε δεκαεξαδική μορφή (hex format) που αποτελείται μόνο από μικρά γράμματα. Για μείωση του κόστους επιλέχθηκε ο τύπος των μεταβλητών που χρησιμοποιούν τα συμβόλαια για την αποθήκευση των hash τιμών να είναι `bytes32` αντί για `string`. Για τον λόγο αυτό όταν εισάγονται οι hash τιμές σε hex format αναπαράσταση ως είσοδοι στα συμβόλαια, είναι απαραίτητο να τοποθετηθεί μπροστά το “0x” ώστε να αποθηκευτεί σωστά η τιμή στην αντίστοιχη μεταβλητή του συμβολαίου.

### **3.6 Εισαγωγή των συμβολαίων στο Ethereum Ropsten Testnet**

Η λειτουργία της εφαρμογής δοκιμάστηκε κατά τη διάρκεια ανάπτυξής της σε ένα τοπικό testnet και στη συνέχεια, αφού οριστικοποιήθηκε η λειτουργία της και τα συμβόλαια που θα χρησιμοποιεί, δοκιμάστηκε και στο Ethereum Ropsten Testnet [45]. Επιλέχθηκε η δοκιμή της στο δημόσιο Ropsten testnet και όχι στο κύριο (Mainnet) Ethereum blockchain διότι οι συναλλαγές στο κύριο δίκτυο απαιτούν τη χρήση πραγματικών ether, τα οποία για να ανακτηθούν χρειάζεται είτε κάποιος να κάνει mine κάτι που απαιτεί υψηλή υπολογιστική ισχύ ώστε να επιφέρει κέρδος ή χρειάζεται να πληρώσει πραγματικά χρήματα για να αγοράσει ether. Επίσης εκτός από το υψηλό χρηματικό κόστος, θα υπήρχαν μεγαλύτεροι χρόνοι αναμονής για την έγκριση κάθε συναλλαγής καθώς ο χρόνος για να γίνει mine ένα block που περιέχει συναλλαγές είναι πολύ μεγαλύτερος. Σε αντίθεση με το Ethereum Mainnet, στο Ethereum Ropsten Testnet ο χρόνος για την έγκριση ενός block είναι πολύ μικρότερος και τα ether που χρησιμοποιούνται σε αυτό δεν έχουν πραγματική αξία και μπορούν εύκολα να αποκτηθούν. Ο τρόπος απόκτησής τους είναι είτε μέσω της διαδικασίας mining όμως

τώρα η απαραίτητη υπολογιστική ισχύ είναι μικρότερη είτε μέσω ενός faucet. Το faucet είναι μια ιστοσελίδα η οποία χαρίζει δωρεάν test ether και στην οποία αρκεί να εισαχθεί η διεύθυνση που αντιστοιχεί σε κάποιο πορτοφόλι και στη συνέχεια να γίνει υποβολή του αιτήματος για παραλαβή των ether. Ουσιαστικά το faucet έχει μια διεύθυνση πορτοφολιού η οποία έχει test ether, τα οποία ίσως έχουν αποκτηθεί με τη διαδικασία του mining, και έτσι όταν εισάγουμε τη διεύθυνση που αντιστοιχεί σε κάποιο πορτοφόλι τότε γίνεται μεταφορά των ether από το πορτοφόλι που χρησιμοποιεί το facet στο δικό μας.

Για την τοποθέτηση των συμβολαίων στο Ethereum Ropsen Testnet δημιουργήθηκε μέσω του εργαλείου Geth μια διεύθυνση πορτοφολιού και είναι η "0xaA97467bB87D1512dff5c3B69AF9A3E0274fcF20". Στη συνέχεια ζητήθηκαν από το faucet [46] κάποια test ether. Έπειτα μέσω του εργαλείου Truffle τοποθετήθηκαν τα συμβόλαια στο blockchain, με την εντολή "truffle migrate --network ropsten". Στην εικόνα 3.4 μπορούμε να δούμε τα αποτελέσματα από την εκτέλεση της εντολής.



```
ani@laptop ~/node/forropsten $ truffle migrate --network ropsten
Using network 'ropsten'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
    ... 0x61b1f49681b6b9db0eab99dc8645e4cc796c74719829d67d2ab53e842c7d8948
  Migrations: 0x44842d9bb57a785d0adc3bfd1e3c068ef61fe286
Saving successful migration to network...
    ... 0x9df56af462ca60ddd8c4c97abf80f47600aa35ce446ab3d4c74dd2c88c15e60c
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying Project...
    ... 0x8f31b06aa9cd209768b24d0828e6843ea39413d27a17be4aa082cc8a046c992a
  Project: 0x9c8fa161e8be45987b8875bf42fcf2fe8f81701a
  Deploying Factoryproject...
    ... 0x90d06979960f369e1270783f81b116b265086d15abf9bb584112c9719b470fd2
  Factoryproject: 0xe66230b71bc5ef8542b2d00dd5b955b83706b295
Saving successful migration to network...
    ... 0xf2e391c8a35d8b832015ef8407c345f23f67d4aff61c2ed82ac3a118b8c4e757
Saving artifacts...
ani@laptop ~/node/forropsten $
```

**Εικόνα 3.4.** Αποτελέσματα από την εκτέλεση της εντολής "truffle migrate --network ropsten"

Από την εικόνα 3.4, μπορούμε να δούμε ότι οι το συμβόλαιο Migration.sol βρίσκεται στην διεύθυνση 0x44842d9Bb57A785d0adc3Bfd1e3C068Ef61fe286, το ενισχυμένο συμβόλαιο που σχετίζεται με το όνομα του συμβολαίου Project.sol βρίσκεται στην διεύθυνση 0x9c8Fa161E8Be45987B8875bf42Fcf2FE8f81701a ενώ το συμβόλαιο Factoryproject.sol που είναι απαραίτητο για τη δημιουργία ενός βασικού συμβολαίου βρίσκεται στη διεύθυνση 0xe66230B71bc5EF8542b2D00Dd5B955b83706b295. Η τοποθέτηση ενός συμβολαίου τύπου Personal.sol δεν είναι ακόμα απαραίτητη. Η τοποθέτηση αυτού στο blockchain δίκτυο θα γίνει μέσω του συμβολαίου Factoryproject.sol και μόνο όταν ένας χρήστης το ζητήσει.

Το Ethereum Ropsten Testnet είναι ένα δημόσιο testnet στο οποίο ο καθένας μπορεί να έχει πρόσβαση στις συναλλαγές που γίνονται σε αυτό. Έτσι μπορεί κάποιος να επισκεφτεί τη σελίδα που αφορά το Ethereum Ropsten Testnet [45] και να εισάγει τη

διεύθυνση “0xaA97467bB87D1512dff5c3B69AF9A3E0274fcF20” και να διαπιστώσει τη δημιουργία των συμβολαίων.

The screenshot shows a web browser window with the URL <https://ropsten.etherscan.io/tx/0x61b1f49681b6b9db0eab99dc8645e4cc796c74719829d67d2ab53e842c7d8948>. The page title is "Transaction 0x61b1f49681b6b9db0eab99dc8645e4cc796c74719829d67d2ab53e842c7d8948". Below the title, there is a tab labeled "Overview". The main content area is titled "Transaction Information" and contains the following details:

TxHash:	0x61b1f49681b6b9db0eab99dc8645e4cc796c74719829d67d2ab53e842c7d8948
TxReceipt Status:	Success
Block Height:	3334860 (35585 block confirmations)
TimeStamp:	5 days 15 hrs ago (May-29-2018 05:14:29 PM +UTC)
From:	0xaa97467bb87d1512dff5c3b69af9a3e0274fcf20
To:	[Contract 0x44842d9bb57a785d0adc3bfd1e3c068ef61fe286 Created] ✓
Value:	0 Ether (\$0.00)
Gas Limit:	4700000
Gas Used By Txn:	277462
Gas Price:	0.0000001 Ether (100 Gwei)
Actual Tx Cost/Fee:	0.0277462 Ether (\$0.000000)
Nonce & {Position}:	9   {0}

**Εικόνα 3.5.** Η απόδειξη που παράχθηκε από την τοποθέτηση του συμβολαίου Migrations.sol

Πηγή: <https://ropsten.etherscan.io/>

Transaction Information	
TxHash:	0x8f31b06aa9cd209768b24d0828e6843ea39413d27a17be4aa082cc8a046c992a
TxReceipt Status:	Success
Block Height:	3334864 (35581 block confirmations)
TimeStamp:	5 days 15 hrs ago (May-29-2018 05:15:40 PM +UTC)
From:	0xaa97467bb87d1512dff5c3b69af9a3e0274fcf20
To:	[Contract 0x9c8fa161e8be45987b8875bf42fcf2fe8f81701a Created] ✓
Value:	0 Ether (\$0.00)
Gas Limit:	4700000
Gas Used By Txn:	613834
Gas Price:	0.0000001 Ether (100 Gwei)
Actual Tx Cost/Fee:	0.0613834 Ether (\$0.000000)
Nonce & {Position}:	11   {0}

**Εικόνα 3.6.** Η απόδειξη που παράχθηκε από την τοποθέτηση του συμβολαίου Project.sol  
 Πηγή: <https://ropsten.etherscan.io/>

Transaction Information	
TxHash:	0x90d06979960f369e1270783f81b116b265086d15abf9bb584112c9719b470fd2
TxReceipt Status:	Success
Block Height:	3334866 (35579 block confirmations)
TimeStamp:	5 days 15 hrs ago (May-29-2018 05:15:57 PM +UTC)
From:	0xaa97467bb87d1512dff5c3b69af9a3e0274fcf20
To:	[Contract 0xe66230b71bc5ef8542b2d00dd5b955b83706b295 Created] ✓
Value:	0 Ether (\$0.00)
Gas Limit:	4700000
Gas Used By Txn:	453075
Gas Price:	0.0000001 Ether (100 Gwei)
Actual Tx Cost/Fee:	0.0453075 Ether (\$0.000000)
Nonce & {Position}:	12   {0}

**Εικόνα 3.7.** Η απόδειξη που παράχθηκε από την τοποθέτηση του συμβολαίου Factoryproject.sol  
 Πηγή: <https://ropsten.etherscan.io/>

Όπως φαίνεται στο πεδίο “From” από τις εικόνες 3.5, 3.6, 3.7, που δείχνουν την απόδειξη της συναλλαγής που παράχθηκε από την τοποθέτηση κάθε συμβολαίου στο Ehtereum Ropsten testnet, τα απαραίτητα ether για την τοποθέτηση των συμβολαίων προήλθαν από τη διεύθυνση πορτοφολιού που χρησιμοποιήσαμε και στάλθηκαν για τη δημιουργία του συμβολαίου το οποίο απέκτησε μια διεύθυνση, όπως δείχνει στο πεδίο “To”.

### 3.7 Οικονομικά στοιχεία από την αλληλεπίδραση με το Ethereum Ropsten Testnet

Όταν τοποθετούνται νέα συμβόλαια στο blockchain δίκτυο ή αλλάζουν οι μεταβλητές που είναι αποθηκευμένες σε ένα έξυπνο συμβόλαιο τότε χρειάζεται να πληρωθούν κάποια ether ως αντίτιμο για τις υπολογιστές πράξεις που απαιτούνται για να γίνουν αυτές οι αλλαγές στο δίκτυο του blockchain. Έτσι κάθε φορά που μέσω της εφαρμογής προστίθενται οι hash τιμές στο συμβόλαιο Project.sol ή δημιουργείται ένα καινούριο βασικό συμβόλαιο ξοδεύονταν test ether, τα οποία παρέχονται από τη διεύθυνση “0xaA97467bB87D1512dff5c3B69AF9A3E0274fcF20”. Το κόστος σε ether που απαιτούνται δίνεται από το γινόμενο  $gas * gas\ price$ . Το gas price καθορίζεται από αυτόν που επιθυμεί να κάνει μια αλλαγή στο blockchain δίκτυο ενώ το gas σχετίζεται με τους υπολογισμούς που απαιτούνται κάθε φορά και είναι σταθερό για την εκτέλεση ίδιων υπολογισμών. Για πιο αναλυτική περιγραφή του τρόπου υπολογισμού του gas μπορεί κάποιος να ανατρέξει στο Appendix H του Ethereum yellow paper [24]. Στο Ethereum Ropsten Testnet ο καθορισμός του gas price μπορεί να γίνει αυθαίρετα και να τεθεί αρκετά μεγάλος καθώς τα ether που χρησιμοποιούνται σε αυτό δεν είναι πραγματικά. Όμως στο Ethereum Mainnet θα πρέπει να καθοριστεί μια κατάλληλη τιμή και κάποιος για να την προσδιορίσει μπορεί να συμβουλευτεί είτε το διάγραμμα στη σελίδα του Ethereum Mainnet [46] είτε μια σελίδα που προτείνεται από πολλούς χρήστες του Ethereum Mainnet [47].

Το κόστος σε gas για την τοποθέτηση των συμβολαίων μπορεί κανένας να το δει από τις εικόνες 3.5, 3.6, 3.7 και από το πεδίο Gas Used By Txn. Συνεπώς τα κόστη σε gas φαίνονται στον πίνακα 3.1:

Όνομα συμβολαίου	Κόστος τοποθέτησης στο blockchain
Migrations.sol	277462 gas
Project.sol	613834 gas
Factoryproject.sol	453075 gas

**Πίνακας 3.1.** Κόστος σε gas τοποθέτησης συμβολαίων στο blockchain

Για να γίνει αναλυτικός έλεγχος για τα κόστη έπρεπε να γίνουν αρκετές αλληλεπιδράσεις με τα συμβόλαια. Για το λόγο αυτό χρησιμοποιήθηκε ένα τοπικό δίκτυο blockchain με τη χρήση του εργαλείου Ganache cli ώστε να μην υπάρχουν χρόνοι αναμονής και έξοδα σε ether. Από τα πειράματα που έγιναν προέκυψαν τα εξής:

1. Για την υποβολή ενός ερωτήματος που σχετίζεται με το βασικό συμβόλαιο, τα κόστη διαμορφώνονται ως εξής:



- 172169 πρώτη δημιουργία ενός βασικού συμβολαίου.
- 157169 οποιαδήποτε άλλη εισαγωγή εκτός της πρώτης

#### Παρατηρήσεις:

Η πρώτη εισαγωγή κοστίζει παραπάνω διότι αρχικοποιείται ο πίνακας `addresses_contracts` και οποιαδήποτε εισαγωγή μετά της πρώτης έχει σταθερό κόστος διότι εκτελούνται οι ίδιες πράξεις. Επιπλέον οι παραπάνω τιμές προέκυψαν για τιμές hash όπου δεν υπήρχε κάποιο byte που να είναι μηδέν [παράρτημα Ι]. Αν κάποιο byte είναι μηδέν τότε το κόστος μειώνεται κατά 64 gas και αν η τιμή του hash έχει μόνο μηδενικά byte τότε το κόστος μειώνεται κατά 17.048 και άρα για κάθε κατηγορία αντίστοιχα διαμορφώνεται στα 170185 και 155185. Τα παραπάνω στοιχεία δίνονται στον πίνακα 3.2:

Είδος τιμής hash	Κόστος 1ης εισαγωγής	Κόστος οποιαδήποτε άλλης εισαγωγής
Τιμή hash με κανένα μηδενικό byte	172169 gas	157169 gas
Τιμή hash μόνο με μηδενικά byte	170185 gas	155185 gas

**Πίνακας 3.2.** Κόστος εισαγωγής σε σχέση με την τιμή hash

2. Για την υποβολή ενός ερωτήματος που σχετίζεται με το ενισχυμένο συμβόλαιο τα κόστη διαμορφώνονται ως εξής:

- 108362 αν η ερώτηση δεν έχει ξαναγίνει
- 114485 αν το ερώτημα υπάρχει ήδη και απλά προστίθεται η καινούρια απάντηση.

#### Παρατηρήσεις:

Η πρώτη εισαγωγή κοστίζει παραπάνω διότι αρχικοποιείται ο πίνακας που περιέχει τις απαντήσεις για το συγκεκριμένο ερώτημα και οποιαδήποτε εισαγωγή μετά της πρώτης έχει σταθερό κόστος διότι εκτελούνται οι ίδιες πράξεις. Επιπλέον οι παραπάνω τιμές προέκυψαν για τιμές hash όπου δεν υπήρχε κάποιο byte που να είναι μηδέν [Παράρτημα Ι]. Αν κάποιο byte είναι μηδέν τότε το κόστος μειώνεται κατά 64 gas εκτός αν η τιμή του hash έχει μόνο μηδενικά bytes τότε οι τιμές διαμορφώνονται διαφορετικά. Το ίδιο ισχύει και για το χρονικό κλειδί, δηλαδή ανάλογα με τον αριθμό των byte που είναι μηδενικά το κόστος διαμορφώνεται αναλόγως κατά 64 gas. Για το λόγο αυτό για κάθε εισαγωγή στη διάρκεια των πειραμάτων χρησιμοποιήθηκε το ίδιο χρονικό κλειδί 1527614740032 το οποίο έχει 26 bytes που είναι μηδενικά [Παράρτημα Ι].

Επιπλέον μελετήθηκαν οι περιπτώσεις όπου η hash τιμή είχε μόνο μηδενικά byte και ως κλειδί εισαγωγής χρησιμοποιήθηκε το 1527614740032. Αρχικά χρησιμοποιήθηκε η τιμή hash με μόνο μηδενικά byte ως η hash τιμή του ερωτήματος και ως hash απάντησης χρησιμοποιήθηκαν τιμές με μη μηδενικά byte. Τα αποτελέσματα έδειξαν ότι για την εισαγωγή της πρώτης απάντησης, δηλαδή αν το ερώτημα δεν έχει ξαναγίνει το κόστος είναι 106314 gas ενώ το κόστος για την εισαγωγή οποιαδήποτε επόμενης

απάντησης είναι 112437. Έπειτα μελετήθηκε η περίπτωση της εισαγωγής μιας τιμής hash με μη μηδενικά byte ως ερώτηση και μιας hash τιμή απάντησης που να έχει μόνο μηδενικά byte. Αν η απάντηση αυτή είναι η πρώτη που εισέρχεται το κόστος από 108362 μειώνεται στα 91314 ενώ αν αφορά οποιαδήποτε άλλη εισαγωγή πριν της πρώτης τότε από 114485 διαμορφώνεται στα 97437. Τέλος η περίπτωση να είναι και το hash της απάντησης και της ερώτησης ίδιο και να αποτελείται μόνο από μηδενικά byte δεν εξετάστηκε καθώς στο κείμενο που αφορά την ερώτηση ο server προσθέτει το όνομα της βάσης που γίνεται το ερώτημα και ύστερα υπολογίζει το hash του κειμένου της συνολικής ερώτησης που προκύπτει. Άρα η περίπτωση να είναι ίδια τα κείμενα της συνολικής ερώτησης και της απάντησης θεωρήθηκε αδύνατο γι' αυτό και δεν μελετήθηκε.

Τα παραπάνω μπορούν να συνοψιστούν στον πίνακα 3.3:

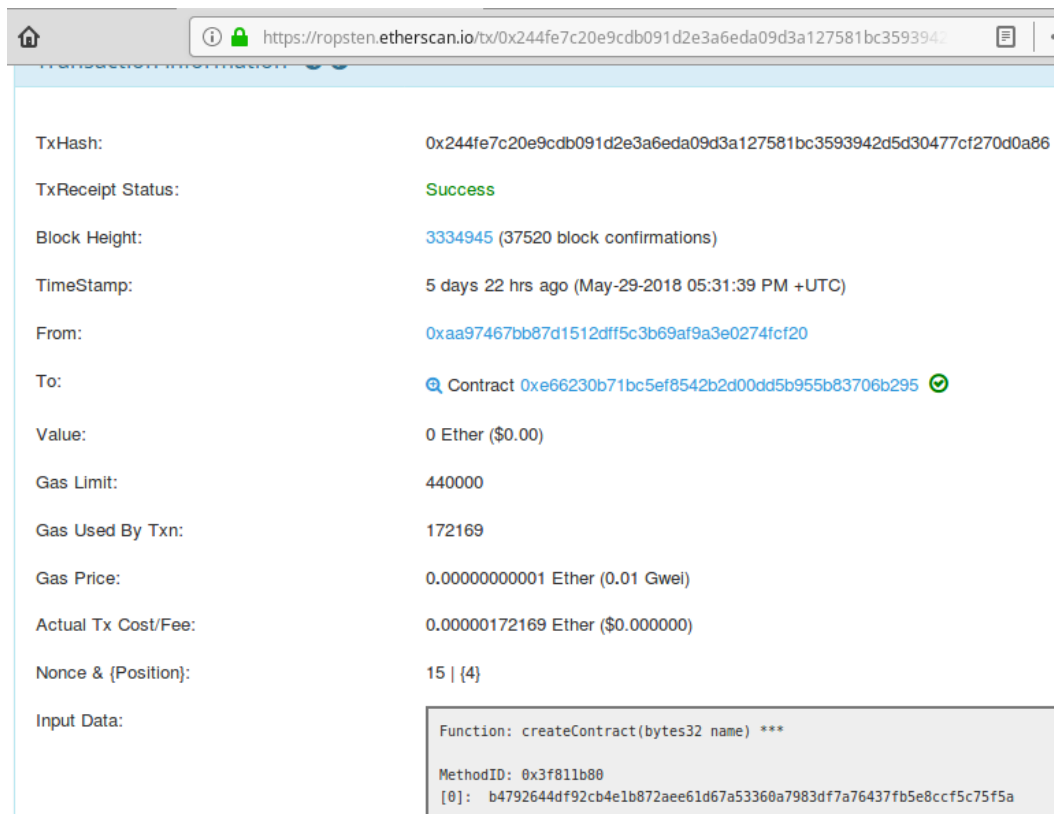
Κόστη	Κατηγορίες αλληλεπίδρασης					
	Ερώτημα με τιμή hash χωρίς μηδενικό byte			Ερώτημα με τιμή hash μόνο με μηδενικά byte		
	1η εισαγωγή απάντησης με τιμή hash χωρίς μηδενικά byte		1η εισαγωγή απάντησης με τιμή hash μόνο με μηδενικά byte		1η εισαγωγή απάντησης με τιμή hash χωρίς μηδενικά byte	1η εισαγωγή απάντησης με τιμή hash μόνο με μηδενικά byte
Κόστος (gas)	108362		91314		106314	
	Οποιαδήποτε άλλη εισαγωγή απάντησης με τιμή hash χωρίς μηδενικά byte	Εισαγωγή απάντησης με τιμή hash μόνο με μηδενικά byte	Οποιαδήποτε άλλη εισαγωγή απάντησης με τιμή hash χωρίς μηδενικά byte	Εισαγωγή απάντησης με τιμή hash μόνο με μηδενικά byte	Οποιαδήποτε άλλη εισαγωγή απάντησης με τιμή hash χωρίς μηδενικά byte	Εισαγωγή απάντησης με τιμή hash μόνο με μηδενικά byte
Κόστος (gas)	114485	97437	114485	Αδύνατο	112437	Αδύνατο

**Πίνακας 3.3.** Συγκεντρωτικός πίνακας κόστους συμβολαίων

Χρειάζεται να επισημανθεί ότι οι τιμές του για τα κόστη ίσως δεν είναι ακριβής διότι στο συμβόλαιο αποθηκεύονται μεταβλητές που αφορούν χρονικές στιγμές σε unix χρόνο και είναι οι timestamps και timestampsEnd. Οι δύο αυτές μεταβλητές δεν εισέρχονται στην συνάρτηση που τις χρησιμοποιεί αλλά παίρνουν τιμή από την εντολή “now” η οποία τους αναθέτει μια unix τιμή χρόνου που σχετίζεται με τη χρονική στιγμή που εκτελείται η συνάρτηση. Επειδή δεν βρέθηκε σχετική βιβλιογραφία που να αναφέρει το κόστος σε gas για τη χρήση της εντολής now, θεωρήθηκε ότι το κόστος για τη διαμόρφωση της τιμής των δύο αυτών μεταβλητών υπολογίζεται με παρόμοιο τρόπο με αυτό για την αποθήκευση της τιμής timeKey η οποία επίσης δίνεται σε χρόνο unix, όμως ο ακριβής προσδιορισμός τους θα απαιτούσε λεπτομερή υπολογισμό και ακριβή

συντονισμό ώστε οι εκτέλεση της συνάρτησης να γίνεται σε συγκεκριμένες χρονικές στιγμές για τις οποίες θα ξέραμε τον αριθμό των ομάδων byte που είναι μηδενικά.

Συνεπώς αν κάποιος θέλει ένα πιο ακριβή υπολογισμό του κόστους σε gas θα πρέπει να λάβει υπόψιν του τον αριθμό των μηδενικών byte τόσο στις hash τιμές των ερωτήσεων και των απαντήσεων αλλά και των μεταβλητών που σχετίζονται με τις χρονικές στιγμές. Για να το κάνει αυτό πρέπει να λάβει υπόψιν του τη χρονική περίοδο που γίνονται οι εισαγωγές των τιμών στα συμβόλαια, ώστε να προσδιορίσει ανάλογα και το timekey και την χρονική τιμή που αποθηκεύεται στις μεταβλητές timestamp και timestampEnd κατά την εκτέλεση της συνάρτησης addValue, και έτσι να υπολογίσει τον αριθμό των μηδενικών byte και συνεπώς το gas.



The screenshot shows a web browser window with the URL <https://ropsten.etherscan.io/tx/0x244fe7c20e9cdb091d2e3a6eda09d3a127581bc3593942d5d30477cf270d0a86>. The page displays transaction details for a transaction on the Ropsten testnet.

TxHash:	0x244fe7c20e9cdb091d2e3a6eda09d3a127581bc3593942d5d30477cf270d0a86
TxReceipt Status:	Success
Block Height:	3334945 (37520 block confirmations)
TimeStamp:	5 days 22 hrs ago (May-29-2018 05:31:39 PM +UTC)
From:	0xaa97467bb87d1512dff5c3b69af9a3e0274fcf20
To:	Contract 0xe66230b71bc5ef8542b2d00dd5b955b83706b295
Value:	0 Ether (\$0.00)
Gas Limit:	440000
Gas Used By Txn:	172169
Gas Price:	0.0000000001 Ether (0.01 Gwei)
Actual Tx Cost/Fee:	0.00000172169 Ether (\$0.000000)
Nonce & {Position}:	15   {4}
Input Data:	<pre>Function: createContract(bytes32 name) *** MethodID: 0x3f811b80 [0]: b4792644df92cb4e1b872aee61d67a53360a7983df7a76437fb5e8ccf5c75f5a</pre>

**Εικόνα 3.8.** Αλληλεπίδραση σχετικά με το βασικό συμβόλαιο  
Πηγή: <https://ropsten.etherscan.io/>

Transaction details from etherscan.io:

- TxHash:** 0x16b50f6c17e37f309e618752558867ada186883965a40b8717b678006adcf6f7
- TxReceipt Status:** Success
- Block Height:** 3342219 (30349 block confirmations)
- TimeStamp:** 4 days 21 hrs ago (May-30-2018 07:27:46 PM +UTC)
- From:** 0xaa97467bb87d1512dff5c3b69af9a3e0274fcf20
- To:** Contract 0x9c8fa161e8be45987b8875bf42fcf2fe8f81701a
- Value:** 0 Ether (\$0.00)
- Gas Limit:** 140000
- Gas Used By Txn:** 108362
- Gas Price:** 0.000000004 Ether (4 Gwei)
- Actual Tx Cost/Fee:** 0.000433448 Ether (\$0.000000)
- Nonce & {Position}:** 18 | {2}

**Εικόνα 3.9.** Αλληλεπίδραση σχετικά με το ενισχυμένο συμβόλαιο  
 Πηγή: <https://ropsten.etherscan.io/>

## 3.8 Παρουσίαση της λειτουργίας της εφαρμογής

### 3.8.1 Η αρχική σελίδα της εφαρμογής

**Blockchain Query Notary Service**

Select Repository  
 PubMed MEDLINE Database

**Contract for the Enhanced Scheme**  
 For more information see [this](#) paper

put query here [Submit](#)

**Contract for the Basic Scheme**  
 If you want a Personal Contract for your query insert your question in the box below. For more information see [this](#) paper

put query here [Submit](#)

**Validation based on smart contract's data**

**Provides validation for a query supported by an Enhanced Scheme contract**  
 Return the hash value of the latest answer to a specific query  
 put query here [Submit](#)

**Return the time values for a specific answer to a query**  
 put query here [Submit](#)  
 put the hash of the query here [Submit](#)

**Return the hash values of every answer to a specific query**  
 put query here [Submit](#)

**Provides validation for a query supported by a Basic Scheme contract**  
 Return the hash value stored in a contract  
 put address here [Submit](#)

Contract Type	Abi. It can be used for personal validation by the user
Personal	[{"inputs": [{"name": "_hashValue", "type": "bytes32"}], "payable": false, "stateMutability": "nonpayable", "type": "constructor"}, {"constant": true, "inputs": [], "name": "getData", "outputs": [{"name": "_data", "type": "bytes32"}], "payable": false, "stateMutability": "view", "type": "function"}]
	[{"constant": false, "inputs": [{"name": "_queryhash", "type": "bytes32"}, {"name": "_answerhash", "type": "bytes32"}, {"name": "_timekey", "type": "uint256"}], "name": "addValue", "outputs": [{"name": "payable": true}], "payable": true}], [{"constant": true, "inputs": [{"name": "_queryhash", "type": "bytes32"}, {"name": "_answerhash", "type": "bytes32"}, {"name": "_timekey", "type": "uint256"}], "name": "getValue", "outputs": [{"name": "data", "type": "bytes32"}], "payable": false}], [{"constant": true, "inputs": [{"name": "_queryhash", "type": "bytes32"}, {"name": "_answerhash", "type": "bytes32"}, {"name": "_timekey", "type": "uint256"}], "name": "isAnswerValid", "outputs": [{"name": "valid", "type": "bool"}], "payable": false}]]

The address of the general contract is: AFTER DEPLOYMENT

**Εικόνα 3.10.** Η αρχική σελίδα της εφαρμογής

Η αρχική σελίδα της εφαρμογής φαίνεται στη εικόνα 3.10. Στα αριστερά υπάρχει το κομμάτι της εφαρμογής στο οποίο ο χρήστης εισάγει ένα ερώτημα και παίρνει την απάντηση από το server μαζί με τα απαραίτητα στοιχεία για την επαλήθευση. Έτσι υπάρχει ένα κουτί ώστε ο χρήστης να επιλέξει την βάση στην οποία θέλει να κάνει το

ερώτημα (PubMed ή Carre) και ένα input box κάτω από το “Contract for the Enhanced Scheme” όπου ο χρήστης μπορεί να εισάγει το ερώτημα που θέλει να απευθύνει στη βάση. Επιπλέον αν ο χρήστης επιθυμεί ένα Personal contract για το ερώτημα που θέλει να κάνει μπορεί να εισάγει την ερώτησή του στο κουτί κάτω από το “Contract for the Basic scheme”.

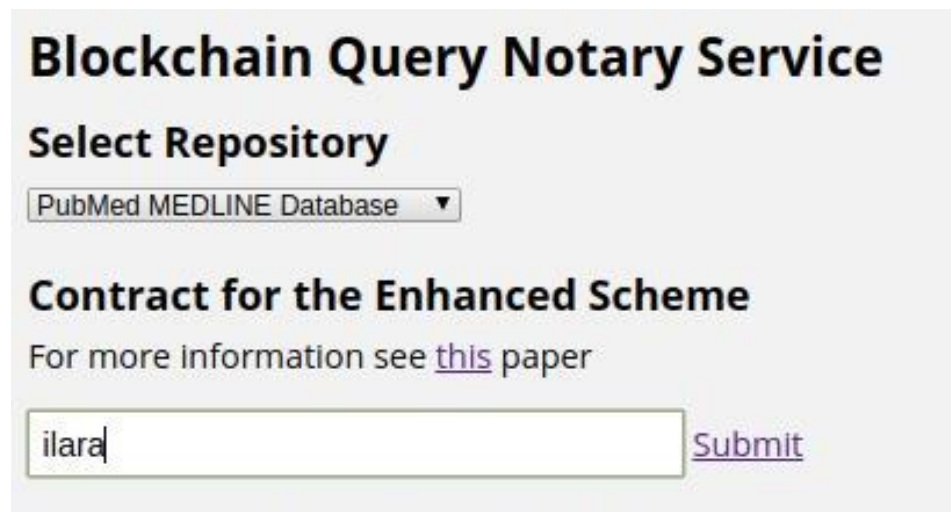
Στα δεξιά υπάρχει το κομμάτι της εφαρμογής που αφορά την επαλήθευση και την ακεραιότητα των δεδομένων που έλαβε ο χρήστης μετά από κάποιο ερώτημα προς τη βάση. Έτσι ο χρήστης αρχικά επιλέγει την βάση που έκανε το ερώτημα (PubMed ή Carre). Έπειτα αν ο χρήστης είχε επιλέξει για το ερώτημά του ένα Personal contract τότε θα εισάγει τα απαραίτητα στοιχεία στο τελευταίο input box που βρίσκεται στο δεξιό μέρος και το οποίο αφορά ένα βασικό συμβόλαιο. Αλλιώς ο χρήστης μπορεί να χρησιμοποιήσει μια από τις συναρτήσεις για το ενισχυμένο εισάγοντας τα απαραίτητα στοιχεία σε ένα από τα τρία πρώτα κουτιά που παρέχονται στο δεξιό τμήμα της σελίδας.

Τέλος στο κάτω μέρος της σελίδας δίνεται ένα πίνακας με το Application Binary Interface (ABI) που αφορά το βασικό και το ενισχυμένο συμβόλαιο. Έτσι κάποιος χρήστης ο οποίος έχει κάποιον κόμβο όπως ο Geth εγκατεστημένο στον υπολογιστή του, που επικοινωνεί με το Ethereum Ropsten Testnet, μπορεί να αλληλεπιδράσει μόνος του με το συμβόλαιο μέσω των συναρτήσεων που αυτό υποστηρίζει. Για να μπορεί να αλληλεπιδράσει μόνος του με το συμβόλαιο θα χρειαστεί το ABI του συμβολαίου και τη διεύθυνση αυτού στο blockchain. Η διεύθυνση του για το ενισχυμένο συμβόλαιο δίνεται στο κάτω μέρος της σελίδας και παραμένει ίδια καθώς ένα είναι το συμβόλαιο στο οποίο εισάγονται οι hash τιμές των ερωτημάτων και των απαντήσεων, ενώ στην περίπτωση του βασικού συμβολαίου δημιουργείται ένα καινούριο προσωπικό συμβόλαιο για κάθε ερώτημα και έτσι η διεύθυνση του παράγεται κατά την τοποθέτηση του συμβολαίου στο blockchain. Την διεύθυνση αυτή την παραλαμβάνει ο χρήστης μαζί με της απάντηση από το server όταν επιλέγει ένα βασικό συμβόλαιο για το ερώτημά του. Έτσι με το ABI και τη διεύθυνση μπορεί να μιλήσει με το συμβόλαιο και για παράδειγμα στην περίπτωση του ενισχυμένου συμβολαίου να χρησιμοποιήσει τη συνάρτηση returnLatest για να πάρει την τελευταία απάντηση που αφορά ένα συγκεκριμένο ερώτημα. Για να αλληλεπιδράσει με τη συνάρτηση returnLatest πρέπει να δώσει σαν input το hash που του επιστράφηκε από την εφαρμογή όταν υπέβαλε το ερώτημα και το οποίο λειτουργεί σαν index για την εύρεση των απαντήσεων σε ένα συγκεκριμένο ερώτημα.

### 3.8.2 Εισαγωγή ενός ερωτήματος στη σελίδα της εφαρμογής

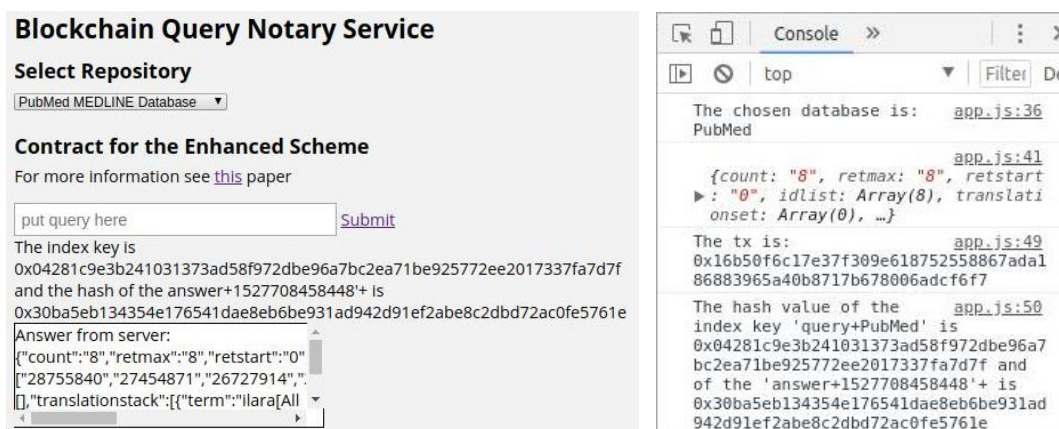
Παρακάτω δίνεται ένα παράδειγμα από τη λειτουργία της εφαρμογής. Το blockchain που χρησιμοποιήθηκε είναι το Ethereum Ropsten Testnet και η διεύθυνση που χρησιμοποιήθηκε ως administrator είναι: “0xaA97467bB87D1512dff5c3B69AF9A3E0274fcF20”. Αρχικά, όπως αναφέρθηκε στο υποκεφάλαιο 3.6, έγινε τοποθέτηση των συμβολαίων στο blockchain έπειτα παίρνουμε την διεύθυνση που αφορά το ενισχυμένο συμβόλαιο και την εισάγουμε στο κάτω μέρος της σελίδας κάτω από το πινάκι και στο αντίστοιχο κείμενο “The address of the contract is:” ώστε να μπορεί ο χρήστης να αλληλεπιδράσει και μόνος του με το συμβόλαιο χρησιμοποιώντας το ABI και ένα Geth κόμβο όπως περιγράφηκε. Μετά την τοποθέτηση των συμβολαίων, η διεύθυνση του Project.sol που αφορά το ενισχυμένο συμβόλαιο είναι: “0x9c8fa161e8be45987b8875bf42fcf2fe8f81701a”.

Έστω ότι ο χρήστης επιλέγει το ενισχυμένο συμβόλαιο και θέλει να κάνει το ερώτημα “ilara” στη βάση PubMed. Εισάγει τα κατάλληλα στοιχεία και πατάει Submit.



The screenshot shows the 'Blockchain Query Notary Service' interface. At the top, it says 'Select Repository' with a dropdown menu set to 'PubMed MEDLINE Database'. Below this is a section titled 'Contract for the Enhanced Scheme' with a link to 'this paper'. There is a text input field containing the word 'ilara' and a 'Submit' button to its right.

Εικόνα 3.11. Πριν ο χρήστης πατήσει Submit



The screenshot shows the same interface as before, but now the 'Submit' button has been clicked. The text input field now contains 'put query here'. Below the input field, the 'Answer from server:' section displays a JSON object: 

```
{ "count": "8", "retmax": "8", "retstart": "0", "idlist": ["28755840", "27454871", "26727914"], "translationstack": [{"term": "ilara"}] }
```

. To the right of the interface, a console window is open, showing the following output: 

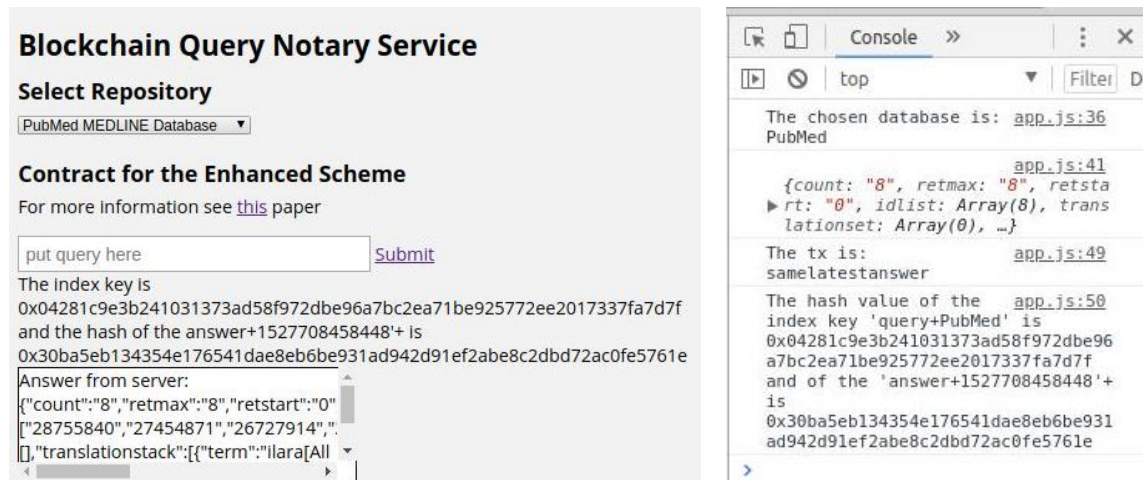
```
The chosen database is: app.js:36 PubMed
app.js:41 {count: "8", retmax: "8", retstart: "0", idlist: Array(8), translationstack: Array(0), ...}
app.js:49 The tx is: 0x16b50f6c17e37f309e618752558867ada186883965a40b8717b678006adcf6f7
app.js:50 The hash value of the index key 'query+PubMed' is 0x04281c9e3b241031373ad58f972dbe96a7bc2ea71be925772ee2017337fa7d7f and of the 'answer+1527708458448' is 0x30ba5eb134354e176541dae8eb6be931ad942d91ef2abe8c2dbd72ac0fe5761e
```

Εικόνα 3.12. Η μορφή της σελίδας μετά την υποβολή του ερωτήματος στη βάση PubMed (αριστερά) και ό,τι εμφανίζεται στην consola και χρησιμοποιείται για τον έλεγχο της ορθής λειτουργίας της εφαρμογής (δεξιά)

Από την εικόνα 3.12 βλέπουμε ότι, αφού ο χρήστης υποβάλει την ερώτησή του, λαμβάνει από την εφαρμογή ένα index key, που όπως φαίνεται και από τα δεδομένα στην consola στην δεξιά εικόνα, προέρχεται από το hash του κειμένου “ερώτημα+βάση που υποβλήθηκε το ερώτημα”. Αυτό το index key μπορεί να το χρησιμοποιήσει στη συνέχεια ο χρήστης για την επαλήθευση που αν θέλει μπορεί να κάνει μόνος του, χωρίς τη χρήση της εφαρμογής. Επίσης ο χρήστης λαμβάνει και το hash της απάντησης από τη βάση, το οποίο αποθηκεύεται στο έξυπνο συμβόλαιο, καθώς και ένα πλαίσιο με την απάντηση από το server. Στην δεξιά εικόνα βλέπουμε ότι υπάρχει κάποιο transactionHash(tx) το οποίο δημιουργήθηκε επειδή έγινε εισαγωγή νέων δεδομένων στο συμβόλαιο. Αυτή η εισαγωγή μπορεί να οφείλεται είτε στο ότι έγινε ένα καινούριο ερώτημα είτε επειδή χρειάστηκε να περαστεί μια καινούρια απάντηση για ένα ήδη υπάρχον ερώτημα.



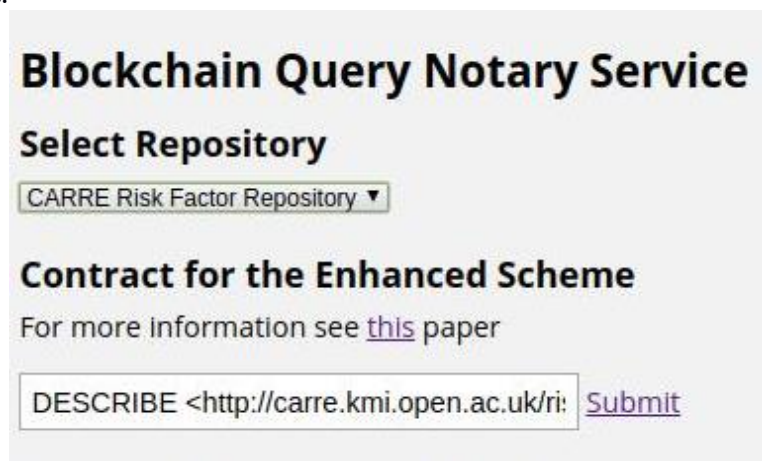
Αν αμέσως μετά ξανακάνουμε το ίδιο ερώτημα στην βάση PubMed τα αποτελέσματα θα είναι τα εξής:



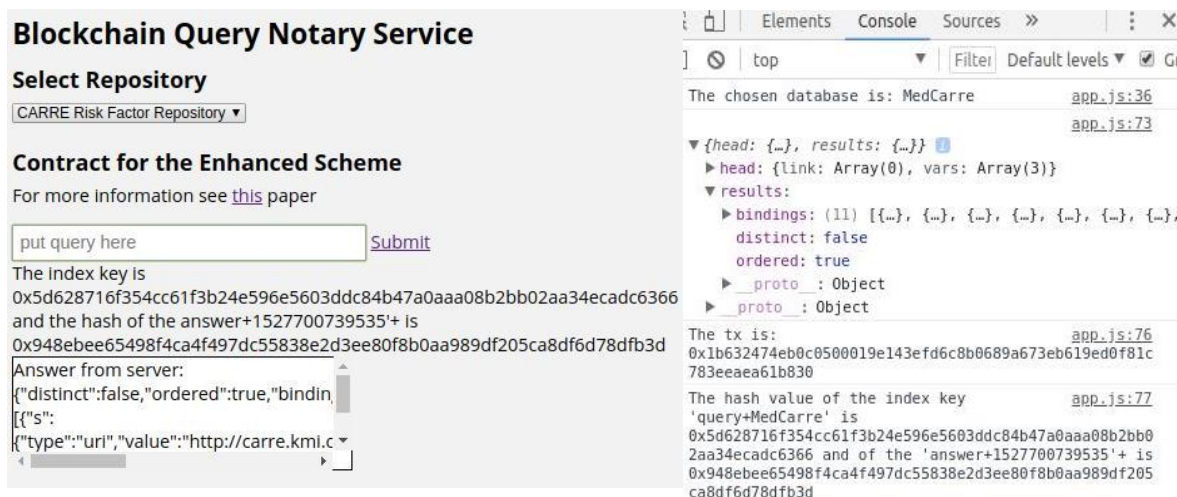
Εικόνα 3.13. Η μορφή της σελίδας μετά την δεύτερη υποβολή του ίδιου ερωτήματος στη βάση PubMed (αριστερά) και ό,τι εμφανίζεται στην console (δεξιά)

Από την εικόνα 3.13 στα δεξιά φαίνεται ότι αυτή την φορά δεν παράχθηκε κάποιο transaction hash καθώς για το ερώτημα "ilara" η απάντηση που λήφθηκε από τη βάση PubMed είναι ίδια και επομένως δεν χρειάστηκε να ανανεωθούν τα περιεχόμενα του smart contract. Είναι λογικό η απάντηση να είναι ίδια καθώς το ερώτημα εκτελέστηκε ξανά, χωρίς να μεσολαβήσει αρκετό χρονικό διάστημα ώστε αυτή να έχει τροποποιηθεί. Επίσης από την εικόνα 3.11 στα αριστερά βλέπουμε ότι το περιβάλλον του χρήστη μένει ίδιο καθώς δεν τον αφορά αν άλλαξε κάτι στο συμβόλαιο και τα μόνα στοιχεία που χρειάζεται είναι η απάντηση από τη βάση, η τιμή του hash της και το index key ώστε να μπορεί να κάνει επαλήθευση.

Έστω ότι ο χρήστης επιλέγει το ενισχυμένο συμβόλαιο για να υποβάλει ένα ερώτημα στην βάση CARRE. Τα ερωτήματα σε αυτή τη βάση γίνονται σε "sparql language" και ένα τέτοιο ερώτημα είναι το "DESCRIBE <http://carre.kmi.open.ac.uk/risk\_factors/RF\_1>". Εισάγει τα κατάλληλα στοιχεία και πατάει Submit.



Εικόνα 3.14. Πριν ο χρήστης πατήσει Submit



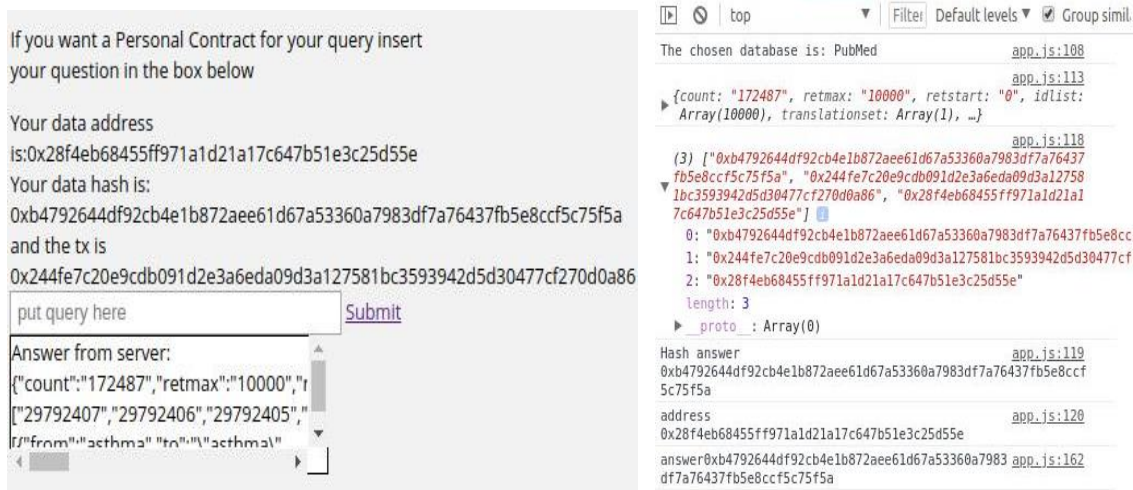
**Εικόνα 3.15.** Η μορφή της σελίδας μετά την υποβολή του ερωτήματος στη βάση CARRE (αριστερά) και ό,τι εμφανίζεται στην console και χρησιμοποιείται για τον έλεγχο της ορθής λειτουργίας της εφαρμογής (δεξιά)

Τα αποτελέσματα των εικόνων 3.15 έχουν την ίδια λογική με αυτά της εικόνας 3.12 όπως ήταν αναμενόμενο, καθώς το μόνο που άλλαξε είναι η βάση στην οποία υποβάλλεται το ερώτημα.

Έστω ότι ο χρήστης επιλέγει ένα βασικό συμβόλαιο και θέλει να κάνει το ερώτημα “asthma” στη βάση PubMed. Εισάγει τα κατάλληλα στοιχεία και πατάει Submit.



**Εικόνα 3.16.** Πριν ο χρήστης πατήσει το Submit



**Εικόνα 3.17.** Η μορφή της σελίδας μετά την υποβολή του ερωτήματος στη βάση CARRE( αριστερά) και ό,τι εμφανίζεται στην console και χρησιμοποιείται για τον έλεγχο της ορθής λειτουργίας της εφαρμογής (δεξιά)

Στον χρήστη επιστρέφονται τα δεδομένα από την βιοϊτρική βάση, η τιμή του hash αυτών και η διεύθυνση του προσωπικού αυτού συμβολαίου που δημιουργήθηκε και τοποθετήθηκε στο blockchain και αφορά προσωπικά τον συγκεκριμένο χρήστη και το ερώτημά του. Επίσης επειδή έγινε εισαγωγή συμβολαίου στο blockchain δίκτυο παράγεται ένα transaction hash το οποίο επιστρέφεται στο χρήστη ως αποδεικτικό στοιχείο για την δημιουργία του συμβολαίου.

### 3.8.3 Επικύρωση δεδομένων

Η διαδικασία της επικύρωσης των δεδομένων απαιτεί μόνο την ανάγνωση των τιμών που περιέχονται στα συμβόλαια χωρίς να γίνονται αλλαγές στις μεταβλητές που βρίσκονται αποθηκευμένες σε αυτά. Επομένως για τη διαδικασία αυτή δεν υπάρχει κάποιο χρηματικό κόστος. Η επικύρωση των δεδομένων μπορεί να γίνει είτε μέσω της εφαρμογής είτε από το χρήστη χωρίς τη μεσολάβηση της εφαρμογής.

#### 3.8.3.1 Χρήση της εφαρμογής για επικύρωση δεδομένων

Για την διαδικασία της επικύρωσης ο χρήστης θα χρησιμοποιήσει το δεξιό τμήμα της εφαρμογής που αφορά την επαλήθευση και την ακεραιότητα των δεδομένων που ελήφθησαν μετά την υποβολή κάποιου ερωτήματος.

Έστω ότι ο χρήστης θέλει να εξακριβώσει αν έχει αλλάξει η απάντηση που έχει για ένα συγκεκριμένο ερώτημα ή επιθυμεί να ελέγξει αν ακριβώς μετά την υποβολή του ερωτήματος του στη βάση η τελευταία απάντηση που έλαβε είναι όντως η πιο πρόσφατη. Τότε θα κάνει χρήση του input box κάτω από το κείμενο “Return the hash value of the latest answer to a specific query”. Εκεί θα εισάγει το ερώτημα, θα επιλέξει τη βάση στην οποία έγινε το ερώτημα και θα πατήσει Submit. Έστω ότι επιλέγει το ερώτημα “ilara” που έγινε στη βάση PubMed.

**Validation based on smart contract's data**

---

**Provides validation for a query supported by an Enhanced Scheme contract**

**Return the hash value of the latest answer to a specific query**

[Submit](#)

The latest answer is  
0x30ba5eb134354e176541dae8eb6be931ad942d91ef2abe8c2dbd72ac0fe5761e

**Εικόνα 3.18.** Η απάντηση μετά την υποβολή του ερωτήματος από τον χρήστη για να πάρει την πιο πρόσφατη απάντηση

Από την εικόνα 3.18 φαίνεται ότι η hash τιμή που επιστράφηκε είναι η αναμενόμενη και είναι αυτή που εισήχθη νωρίτερα και φαίνεται στην εικόνα 3.12 .

Έστω ότι ο χρήστης επιθυμεί να μάθει το χρονικό διάστημα για το οποίο ισχύει μια συγκεκριμένη απάντηση που έλαβε για ένα ερώτημα. Τότε θα κάνει χρήση των input box κάτω από το κείμενο “Return the time values for a specific answer to a query”. Εκεί θα εισάγει στο πάνω κουτί το ερώτημα, στο κάτω την απάντηση όπως την έλαβε από τη βάση, θα επιλέξει την βάση στην οποία είχε υποβληθεί το ερώτημα και θα πατήσει Submit. Έστω ότι επιλέγει το ερώτημα “ilara” που έγινε στη βάση PubMed και εισάγει και το hash της απάντησης που είχε λάβει και το οποίο φαίνεται στις εικόνες 3.12, 3.13.

**Return the time values for a specific answer to a query**

[Submit](#)

The answer was inserted in Wed, 30 May 2018 19:27:46 GMT  
The answer still applies

**Εικόνα 3.19.** Μετά την υποβολή των στοιχείων ώστε ο χρήστης να μάθει το χρονικό διάστημα ισχύος μιας συγκεκριμένης απάντησης

Έστω ότι ο χρήστης επιθυμεί να μάθει όλες τις απαντήσεις που υπάρχουν στο συμβόλαιο για ένα συγκεκριμένο ερώτημα. Τότε θα κάνει χρήση του input box κάτω από το κείμενο “Return the hash values of every answer to a specific query”. Εκεί θα εισάγει το ερώτημα, έστω “asthma” και θα επιλέξει τη βάση στην οποία έγινε το ερώτημα έστω PubMed και θα πατήσει Submit.



**Return the hash values of every answer to a specific query**

ilara [Submit](#)

Values:

0x30ba5eb134354e176541dae8eb6be

**Εικόνα 3.20.** Μετά την υποβολή των στοιχείων ώστε ο χρήστης να λάβει τις τιμές των hash όλων των απαντήσεων που αφορούν ένα συγκεκριμένο ερώτημα

Έστω ότι ο χρήστης είχε επιλέξει ένα βασικό συμβόλαιο για το ερώτημά του και θέλει να επαληθεύσει αν η διεύθυνση του συμβολαίου που του δόθηκε όντως περιέχει το hash της απάντησης που του δόθηκε. Τότε θα κάνει χρήση του input box κάτω από το κείμενο “Return the hash values of every answer to a specific query”. Εκεί θα εισάγει τη διεύθυνση του συμβολαίου, έστω η διεύθυνση που επιστράφηκε στην εικόνα 3.17 που είναι η “0x28f4eb68455ff971a1d21a17c647b51e3c25d55e” και θα επιλέξει τη βάση στην οποία έγινε το ερώτημα, που στη συγκεκριμένη περίπτωση ήταν το PubMed και θα πατήσει Submit.

**Provides validation for a query supported by a Basic Scheme contract**

**Return the hash value stored in a contract**

0x28f4eb68455ff971a1d21a17c647b51e3c [Submit](#)

Your data

are:0xb4792644df92cb4e1b872aee61d67a53360a7983df7a76437fb5e8ccf5c75f5a

**Εικόνα 3.21.** Μετά την υποβολή των στοιχείων, ώστε ο χρήστης να λάβει την τιμή του hash που είναι αποθηκευμένη στο Personal contract που δημιουργήθηκε για το ερώτημά του

Από την εικόνα 3.21 φαίνεται ότι η τιμή του hash που περιέχει το συμβόλαιο που βρίσκεται στη διεύθυνση “0x28f4eb68455ff971a1d21a17c647b51e3c25d55e” είναι ίδια με αυτή που δόθηκε στο χρήστη όταν υπέβαλε το ερώτημα και η οποία φαίνεται στην εικόνα 3.17.

### 3.8.3.2 Επαλήθευση δεδομένων από το χρήστη χωρίς τη χρήση της εφαρμογής

Έστω ότι κάποιος χρήστης θέλει να κάνει μόνος του επαλήθευση των στοιχείων που πήρε, χωρίς την βοήθεια της εφαρμογής. Τότε χρειάζεται να έχει ένα κόμβο όπως ο Geth για να μπορεί να επικοινωνήσει με το δίκτυο. Για το πραγματικό δίκτυο κάποιος που χρησιμοποιεί ένα Geth κόμβο πρέπει να συνδεθεί με αυτόν με την εντολή “geth attach”. Με τον τρόπο αυτό ο χρήστης μπορεί να επικοινωνήσει με το blockchain δίκτυο. Έπειτα στην κονσόλα που εμφανίζεται ο χρήστης πρέπει να δημιουργήσει ένα στιγμιότυπο του συμβολαίου χρησιμοποιώντας το ABI αυτού και την διεύθυνση του στο blockchain. Αφού ορισθεί το στιγμιότυπο ο χρήστης μπορεί να αλληλεπιδράσει με το συμβόλαιο μέσω των συναρτήσεων που αυτό υποστηρίζει.

Παρακάτω δίνεται ένα παράδειγμα πώς ο χρήστης θα μπορούσε μόνος του να πάρει τις πληροφορίες για ένα βασικό συμβόλαιο. Έστω το βασικό συμβόλαιο που δημιουργήθηκε στην εικόνα 3.17 και έχει την διεύθυνση “0x28f4eb68455ff971a1d21a17c647b51e3c25d55e”.

```
ani@laptop ~ $ geth attach http://localhost:8545
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.7-stable-66432f38/linux-amd64/go1.10
coinbase: 0xaa97467bb87d1512dff5c3b69af9a3e0274fcf20
at block: 3493581 (Sat, 23 Jun 2018 11:52:25 EEST)
modules: eth:1.0 net:1.0 personal:1.0 rpc:1.0 web3:1.0

> var abi=[{"inputs": [{"name": "_hashValue", "type": "bytes32"}], "payable": false, "stateMutability": "nonpayable", "type": "constructor"}, {"constant": true, "inputs": [], "name": "getData", "outputs": [{"name": "data", "type": "bytes32"}], "payable": false, "stateMutability": "view", "type": "function"}];
undefined
> var instance=eth.contract(abi).at("0x28f4eb68455ff971a1d21a17c647b51e3c25d55e");

undefined
> instance.getData.call()

"0xb4792644df92cb4e1b872aee61d67a53360a7983df7a76437fb5e8ccf5c75f5a"
```

**Εικόνα 3.22.** Αλληλεπίδραση του χρήστη με το συμβόλαιο μέσα από ένα terminal που επικοινωνεί με ένα κόμβο.

Στην εικόνα 3.22 φαίνεται ότι αρχικά συνδεόμαστε με τον κόμβο και στη συνέχεια ορίζουμε τη μεταβλητή ABI η οποία ισούται με την τιμή του Personal ABI που δίνεται στον πίνακα στο κάτω μέρος της σελίδας της εφαρμογής. Έπειτα δημιουργείται το στιγμιότυπο του συμβολαίου και τέλος καλείται η συνάρτηση αυτού που μας δείχνει τα δεδομένα που αυτό περιέχει ώστε ο χρήστης να τα συγκρίνει για να ελέγξει αν είναι ίδια με αυτά που του δόθηκαν. Όπως φαίνεται από την εικόνα τα δεδομένα που περιέχει το συμβόλαιο που βρίσκεται στην διεύθυνση “0x28f4eb68455ff971a1d21a17c647b51e3c25d55e” είναι ίδια με αυτά που είχαν δοθεί στον χρήστη και είναι ορατά στην εικόνα 3.17.



# ΚΕΦΑΛΑΙΟ 4

## Εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν στη διπλωματική

### 4.1 Ethereum blockchain

Η εφαρμογή αναπτύχθηκε με τέτοιο τρόπο ώστε να μπορεί να επικοινωνεί με το Ethereum blockchain. Επειδή στο Ethereum mainnet απαιτούνται κρυπτονομίσματα ether με πραγματική αξία, επιλέχθηκε να τοποθετηθούν τα έξυπνα συμβόλαια σε ένα Ethereum Testnet και πιο συγκεκριμένα με το Ethereum Ropsten testnet. Έτσι η εφαρμογή που περιγράφεται σε αυτή τη διπλωματική αλληλεπιδρά με το Ethereum Ropsten testnet. Ο λόγος που επιλέχθηκε ένα testnet είναι διότι σε αυτό αν και γίνεται πάλι χρήση των κρυπτονομισμάτων ether, αυτά δεν έχουν πραγματική αξία, και επιπλέον οι χρόνοι έγκρισης των συναλλαγών είναι μικρότεροι. Για το λόγο αυτό κρίθηκε πιο κατάλληλο δίκτυο το Ethereum Ropsten testnet για την λειτουργία της εφαρμογής, η οποία βρίσκεται ακόμα σε πιλοτικό στάδιο. Επίσης κατά την ανάπτυξη της εφαρμογής για να γίνει πιο εύκολος, γρήγορος και ασφαλής ο έλεγχος της λειτουργίας της χρησιμοποιήθηκε ένα τοπικό προσωπικό Ethereum blockchain δίκτυο.

#### 4.1.1 Geth

Μια από τις τρεις εφαρμογές του πρωτοκόλλου του Ethereum είναι το Go Ethereum [48], το οποίο είναι γραμμένο σε γλώσσα προγραμματισμού Go, είναι ανοιχτού κώδικα και έχει άδεια τύπου GNU LGPL v3. Το Go Ethereum είναι διαθέσιμο και σαν standalone client που ονομάζεται Geth και μπορεί να εγκατασταθεί σε σχεδόν οποιοδήποτε λειτουργικό σύστημα. Ουσιαστικά το Geth είναι μια command line διεπαφή για την εκτέλεση ενός πλήρους Ethereum κόμβου. Μετά την εγκατάσταση και την εκτέλεση του είναι δυνατή η επικοινωνία με το Ethereum blockchain και έτσι μερικές από τις δυνατότητες που προσφέρει σε κάποιον είναι :

- να γίνει miner
- να μεταφέρει ether μεταξύ διαφόρων διευθύνσεων
- να δημιουργήσει έξυπνα συμβόλαια και κάνει συναλλαγές με αυτά
- να εξερευνήσει τη σειρά και τα block της αλυσίδας του blockchain

Η εφαρμογή που αναπτύχθηκε σε αυτή τη διπλωματική χρησιμοποιεί την έκδοση Geth 1.8.7-stable και ο Geth κόμβος χρησιμοποιήθηκε για τον συγχρονισμό με το Ethereum Ropsten testnet ώστε να γίνει δυνατή η τοποθέτηση των συμβολαίων σε αυτό.

#### 4.1.2 Truffle Framework

Το Truffle framework [49] παρέχει ένα περιβάλλον για την ανάπτυξη, τον έλεγχο και την οργάνωση των έξυπνων συμβολαίων. Είναι ένα εργαλείο χρήσιμο σε κάθε προγραμματιστή που θέλει να δημιουργήσει έξυπνα συμβόλαια σε γλώσσα

προγραμματισμού solidity και να αλληλεπιδρά εύκολα με αυτά. Συνεπώς το Truffle παρέχει τις εξής δυνατότητες:

- αφού συνταχθεί ο κώδικας του συμβολαίου σε γλώσσα προγραμματισμού Solidity, να γίνει το compilation, το linking, η υποβολή του συμβολαίου στο blockchain και το binary management το οποίο αφορά τη διαχείριση των artifacts που παράγονται από την επεξεργασία των έξυπνων συμβολαίων
- εκτέλεση αυτοματοποιημένων τεστ αρχείων ώστε ο προγραμματιστής να μπορεί να ελέγξει αν τα αποτελέσματα από την αλληλεπίδραση με τα συμβόλαια είναι τα επιθυμητά και έτσι να διαπιστώσει αν η λογική του κώδικα των συμβολαίων είναι σωστή. Αυτά τα αρχεία μπορούν να γραφούν είτε σε γλώσσα Javascript όπου το Truffle χρησιμοποιεί το Mocha testing framework [50] και το Chai Assertion Library [51], είτε σε γλώσσα Solidity.
- να διευκολυνθεί η τοποθέτηση των συμβολαίων στο blockchain με τη σύνταξη απλών script αρχείων. Με αυτόν τον τρόπο ακόμα και αν χρειαστεί να επεκταθεί η ανάπτυξη ενός project, για το οποίο έχουμε ήδη υλοποιήσει και τοποθετήσει στο blockchain έξυπνα συμβόλαια, με την τοποθέτηση επιπλέον συμβολαίων στο blockchain αυτό μπορεί να γίνει εύκολα χωρίς να απαιτείται τα συμβόλαια που υπάρχουν ήδη στο blockchain να τοποθετηθούν ξανά.
- διαχείριση των δικτύων blockchain ώστε να καθοριστεί το blockchain δίκτυο στο οποίο θέλουμε να τοποθετήσουμε το έξυπνο συμβόλαιο. Τα δίκτυα αυτά μπορεί να είναι το Ethereum mainnet είτε κάποιο Ethereum testnet είτε κάποιο private δίκτυο.
- διαχείριση πακέτων με τη χρήση EthPM και NPM ώστε να γίνεται δυνατός ο διαμοιρασμός μεταξύ προγραμματιστών, αρχείων με κώδικα που αφορούν έξυπνα συμβόλαια, βιβλιοθήκες, ανάπτυξη αποκεντρωμένων εφαρμογών και τα λοιπά.
- μια διαδραστική κονσόλα για γρήγορη και εύκολη επικοινωνία και αλληλεπίδραση με τα έξυπνα συμβόλαια.
- εύκολος τρόπος συγγραφής external script που επικοινωνούν με τα έξυπνα συμβόλαια.
- βελτιστοποίησης τους κώδικα των συμβολαίων και γρήγορη εκτέλεση των test αρχείων. Επιπλέον το truffle παρέχει και το εργαλείο Ganache, λεπτομέρειες για το οποίο δίνονται παρακάτω, το οποίο όταν συνδυαστεί με το Truffle καθιστά γρήγορη και απλή τη δημιουργία αποκεντρωμένων εφαρμογών.

Ακόμα το Truffle βοηθάει επιπλέον τους προγραμματιστές και παρέχει κάποια Truffle-boxes που είναι βοηθητικά boilerplates, δηλαδή στερεότυπος κώδικας που χρησιμοποιείται για την δημιουργία αποκεντρωμένων εφαρμογών. Τα Truffle-boxes μπορεί να προέρχονται είτε από προγραμματιστές που εργάζονται στο Truffle, αυτά είναι τα Official Boxes, είτε από μέλη της κοινότητας του Truffle, αυτά είναι τα Community Boxes, και εκτός από τη χρήση του Truffle μπορεί να χρησιμοποιούν και άλλες χρήσιμες δομές όπως έξυπνα συμβόλαια γραμμένα σε γλώσσα Solidity, όψεις που χρησιμοποιεί στο front-end μια αποκεντρωμένη εφαρμογή και πολλά άλλα. Έτσι ένας προγραμματιστής μπορεί να βασιστεί στα truffle-boxes και στη συνέχεια να εμπλουτίσει επιπλέον την εφαρμογή του, προσαρμόζοντας κατάλληλα των κώδικα και προσθέτοντας επιπλέον εργαλεία.

Η εφαρμογή που αναπτύχθηκε σε αυτή τη διπλωματική χρησιμοποιεί την 4.18 έκδοση του Truffle. Επίσης για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το Community Box με όνομα: “arvindkalra/express-box” [52]. Αυτό το box παρέχει ένα παράδειγμα μιας αποκεντρωμένης εφαρμογής που κάνει χρήση του NodeJS (ExpressJS)

και παρέχει API endpoints για την επικοινωνία με το Ethereum Blockchain και τα έξυπνα συμβόλαια.

#### 4.1.3 Ganache CLI

Το Ganache CLI [53] είναι ένα από τα εργαλεία που χρησιμοποιούνται μαζί με το Truffle, ανήκει δηλαδή στο Truffle suite, για την ανάπτυξη εφαρμογών στο Ethereum blockchain και αποτελεί την command line έκδοση του εργαλείου Ganache [54] το οποίο παρέχει ένα προσωπικό Ethereum blockchain τοπικά στον υπολογιστή του χρήστη. Το Ganache CLI είναι γραμμένο σε γλώσσα προγραμματισμού Javascript και είναι διαθέσιμο σαν ένα Node πακέτο μέσω του npm. Χρησιμοποιεί τη συλλογή βιβλιοθηκών ethereumjs, για να προσομοιώσει τη συμπεριφορά ενός πλήρους κόμβου και έτσι να κάνει την ανάπτυξη εφαρμογών για το Ethereum πιο εύκολη γρήγορη και ασφαλή. Επίσης περιλαμβάνει τη χρήση γνωστών Remote Procedure Call (RPC) συναρτήσεων και δυνατοτήτων όπως τα events. Για τους λόγους αυτούς το Ganache CLI είναι ένα ιδανικό εργαλείο που χρησιμοποιείται για τον έλεγχο της ορθής λειτουργίας μιας εφαρμογής σχεδιασμένης για το Ethereum.

Κατά την ανάπτυξη της εφαρμογής που υλοποιήθηκε σε αυτή τη διπλωματική, και πριν αυτή πάρει την τελική της μορφή, χρησιμοποιήθηκε το εργαλείο Ganache CLI για τον έλεγχο της ομαλής λειτουργίας τόσο των συμβολαίων όσο και της εφαρμογής καθώς παρέχει ένα προσωπικό Ethereum blockchain τοπικά στον υπολογιστή του χρήστη μαζί με κάποιες διευθύνσεις (addresses) που αντιστοιχούν σε πορτοφόλια με 100 ether, τα 100 ether είναι η προκαθορισμένη ρύθμιση η οποία όμως μπορεί να οριστεί από το χρήστη. Αυτά τα ether είναι απαραίτητα τόσο για την τοποθέτηση των συμβολαίων στο Ethereum όσο και για την αλληλεπίδραση με τα συμβόλαια και παρέχονται κάθε φορά με την ενεργοποίηση της εφαρμογής Ganache, χωρίς να χρειάζεται ο χρήστης να κάνει mine ή να ζητήσει κάποια test ether όπως συμβαίνει στο Ethereum Ropsten testnet. Επίσης τα συμβόλαια τοποθετούνται μόνο στο προσωπικό αυτό blockchain και οποιαδήποτε αλληλεπίδραση με αυτά δεν είναι δημόσια ορατή. Ένα ακόμα πλεονέκτημα από τη χρήση του Ganache CLI είναι ότι οι χρόνος για την δημιουργία κάθε block της αλυσίδας του blockchain μπορεί να ρυθμιστεί από το χρήστη ώστε να είναι μικρός ακόμα και μηδενικός. Έτσι αποφεύγονται διαστήματα αναμονής και επιταχύνεται η διαδικασία ελέγχου της ορθής λειτουργίας της εφαρμογής. Συνεπώς για αυτούς τους λόγους το Ganache CLI κρίθηκε απαραίτητο για το στάδιο ανάπτυξης της εφαρμογής.

#### 4.1.4 Solidity

Η γλώσσα Solidity [55] είναι μια από τις γλώσσες προγραμματισμού που χρησιμοποιούνται για να γραφτεί ο κώδικας ενός έξυπνου συμβολαίου και βασίζεται στις γλώσσες προγραμματισμού Javascript και C++. Δημιουργήθηκε από τους Gavin Wood, Christian Reitwiessner, Alex Beregszaszi, Yoichi Hirai και κάποιους άλλους που συμμετείχαν στην ομάδα του Ethereum ώστε να γίνει δυνατή η εγγραφή των συμβολαίων στο blockchain. Η γλώσσα solidity εκτός από την πλατφόρμα blockchain του Ethereum υποστηρίζεται και από άλλες όπως είναι το Tendermint, το Zeppelin και το Counterparty.

Ο μεταγλωττιστής (compiler) που χρησιμοποιείται για την μετατροπή από την υψηλού επιπέδου γλώσσα προγραμματισμού σε γλώσσα μηχανής κατανοητή από τον EVM, μπορεί να είναι εγκατεστημένος στον υπολογιστή και ονομάζεται solc ή να είναι

κάποιος compiler σε μια ιστοσελίδα ο οποίος μπορεί να εγκατασταθεί και σε κάποιον υπολογιστή για να τρέχει τοπικά και ονομάζεται browser-solidity(remix).

Η γλώσσα solidity είναι μια στατικού τύπου γλώσσα, δηλαδή είναι απαραίτητο σε χρόνο εκτέλεσης (compile time) να είναι γνωστοί οι τύποι των μεταβλητών σε αντίθεση με τη γλώσσα javascript που επιτρέπει μεγαλύτερη ευελιξία στον τύπο των μεταβλητών. Οι τύποι των μεταβλητών που υποστηρίζει είναι :

- Boolean που παίρνουν την τιμή αληθής(true) ή ψευδής(false)
- Int/uint προσημασμένοι και μη προσημασμένοι ακέραιοι των 256 bit. Είναι δυνατόν να χρησιμοποιηθούν και λιγότερα bit αν προσδιοριστούν κατάλληλα οι τύποι, πχ uint8 για 8 bit.
- Address που υποστηρίζει 20 byte όπως είναι και οι διευθύνσεις στο ethereum. Η μεταβλητή αυτή μπορεί να χρησιμοποιεί τις μεθόδους “.balance”, “.transfer” που αντίστοιχα επιστρέφουν το υπόλοιπο του λογαριασμού ή μεταφέρουν ether σε έναν λογαριασμό
- Strings
- Arrays (πίνακες)
- Enums για το καθορισμό μεταβλητών από τον χρήστη
- Operators (χειριστές)
- Structs (δομές)
- Mappings που χρησιμοποιείται για τη δημιουργία πινάκων (table) που λειτουργούν σαν ευρετήρια. Η χρήση τους θα γίνει κατανοητή με το παρακάτω παράδειγμα:

```
mapping(address => uint) public balances;
function update(uint newBalance) {
    balances[msg.sender] = newBalance;
}
```

Όπου όπως προσδιορίστηκε το κλειδί msg.sender είναι τύπου address και το newBalance τύπου uint.

Εκτός από την δήλωση μεταβλητών μπορούν να οριστούν και συναρτήσεις (functions), event που περικλύουν εντολές που θα εκτελεστούν όταν συμβεί κάτι που θα ενεργοποιήσει τα event. Επίσης είναι δυνατόν τα συμβόλαια να κληρονομούν άλλα συμβόλαια (inheritance).

Μια τυπική μορφή ενός συμβολαίου στη γλώσσα solidity έχει την παρακάτω μορφή:

```
pragma solidity ^0.4.0;
contract OwnedToken {
    //καθορισμός μεταβλητών

    constructor(bytes32 _name) public {
        ....
    }
    function changeName(bytes32 newName) {
        ...
    }
}
```

Από την παραπάνω μορφή παρατηρούμε ότι αρχικά πρέπει να οριστεί η έκδοση της γλώσσας solidity που χρησιμοποιείται. Έπειτα ορίζουμε το συμβόλαιο δίνοντάς του ένα όνομα contract OwnedToken. Στη συνέχεια μπορούμε, εάν θέλουμε, να δημιουργήσουμε έναν constructor που ενεργοποιείται όταν γίνεται υποβολή του συμβολαίου στο blockchain, καθώς και να κατασκευάσουμε και άλλες συναρτήσεις.

Στην παρούσα διπλωματική χρησιμοποιήθηκε ο μεταγλωττιστής solc και η έκδοση v0.4.23. Ο μεταγλωττιστής αυτός παρέχεται με την εγκατάσταση του Truffle και είναι αυτός που χρησιμοποιεί το Truffle όταν γίνεται compile των έξυπνων συμβολαίων

#### **4.1.5 Web3**

Το Web3 [56] χρησιμοποιείται για την επικοινωνία της εφαρμογής με το Ethereum και είναι διαθέσιμο, από τη βιβλιοθήκη web3.js [57]. Η βιβλιοθήκη αυτή είναι διαθέσιμη μέσω npm και επιτρέπει την επικοινωνία με έναν τοπικό ή απομακρυσμένο κόμβο του Ethereum, χρησιμοποιώντας HyperText Transfer Protocol (HTTP) ή Inter Process Communication (IPC) συνδέσεις.

Η εφαρμογή που παρουσιάζεται σε αυτή τη διπλωματική χρησιμοποιεί την έκδοση 0.20.0 του web3. Με τη χρήση του καθίσταται δυνατή η επικοινωνία του server side της εφαρμογής με τον τοπικό κόμβο Geth και γενικότερα με το Ethereum και τα έξυπνα συμβόλαια.

#### **4.1.6 Truffle-contract**

Το truffle-contract [58] είναι ένα πακέτο που είναι διαθέσιμο μέσω npm και το οποίο παρέχει διάφορες συναρτήσεις για καλύτερη επικοινωνία με το Ethereum και τα έξυπνα συμβόλαια και παρέχεται για χρήση στο Node ή στον φυλλομετρητή. Μερικά από τα χαρακτηριστικά που προσφέρει η χρήση του είναι:

4. ο εύκολος έλεγχος του πότε μια συναλλαγή έχει γίνει mine ώστε στη συνέχεια να εκτελεστεί μια άλλη συνάρτηση.
5. η υποστήριξη της χρήσης υποσχέσεων (promises) που λύνει το πρόβλημα του “callback hell” που προκύπτει από τον ασύγχρονο χαρακτήρα της Javascript.
6. η δυνατότητα καθορισμού προκαθορισμένων τιμών στις μεταβλητές from και gas που ανάλογα με την περίπτωση ίσως διευκολύνουν τον προγραμματιστή στον κώδικα που συντάσσει.
7. Πρόσβαση στην απόδειξη συναλλαγής (transaction receipt), στη hash τιμή που την συνοδεύει (transaction hash) και στα αρχεία καταγραφής (logs) που περιέχονται σε αυτή.

Η εφαρμογή που αναπτύσσεται σε αυτή τη διπλωματική χρησιμοποιεί την έκδοση 1.1.11 του truffle-contract. Χρησιμοποιείται στη μεριά του server της εφαρμογής ώστε να μπορούν να βρεθούν οι απαραίτητες πληροφορίες για τα έξυπνα συμβόλαια που έχουμε κάνει compile και έχουμε τοποθετήσει στο blockchain μέσω του εργαλείου truffle, και για την αλληλεπίδραση με αυτά.

## **4.2 Κατασκευή της σελίδας που αλληλεπιδρά ο χρήστης**

Η σελίδα που βλέπει ο χρήστης είναι γραμμένη σε γλώσσα προγραμματισμού HTML (Hypertext Markup Language) και για την μορφοποίησή της συντάχθηκε ένα CSS (Cascading Style Sheets) αρχείο. Για την επικοινωνία της σελίδας με τον server, την



εκτέλεση σεναρίων (scripts) στη σελίδα του χρήστη και την εμφάνιση των αποτελεσμάτων σε αυτή, συντάχθηκε ένα αρχείο γραμμένο σε γλώσσα Javascript που κάνει χρήση της βιβλιοθήκης jQuery.

#### 4.2.1 jQuery

Το jQuery [59] είναι μια βιβλιοθήκη της Javascript που χρησιμοποιείται για να κάνει πιο εύκολη την υλοποίηση σεναρίων στη σελίδα που εκτελείται στη μεριά του πελάτη (client side) που χρησιμοποιεί την εφαρμογή. Κυκλοφόρησε τον Ιανουάριο του 2006 από τον John Resig, χρησιμοποιείται στις περισσότερες ιστοσελίδες και έχει άδεια ελεύθερου λογισμικού MIT (Massachusetts Institute of Technology).

Στην παρούσα διπλωματική χρησιμοποιήθηκε η έκδοση jQuery 3.1.1.

### 4.3 ExpressJS

Το ExpressJS [60] είναι ένα web application framework που είναι βασισμένο στο Node.js platform και παρέχει διάφορα εργαλεία για την δημιουργία εφαρμογών διαδικτύου ή κινητών. Όσον αφορά το Node.js είναι μια ανοιχτού κώδικα πλατφόρμα με ένα περιβάλλον εκτέλεσης, που δίνει τη δυνατότητα στους προγραμματιστές να δημιουργήσουν server-side εργαλεία και εφαρμογές σε γλώσσα προγραμματισμού Javascript. Μερικά από τα πλεονεκτήματα που παρέχει είναι [61]:

- εξαιρετική απόδοση και ταχύτητα καθώς έχει σχεδιασθεί ώστε να επιτυγχάνεται βελτιστοποίηση των κλιμακωτών εφαρμογών και παρέχει μια αρχιτεκτονική event-driven.
- παρέχει το node package manager (npm) το οποίο δίνει πρόσβαση σε ένα πλήθος βιβλιοθηκών και κάνει την εγκατάσταση τους σε ένα project απλή.

Με το ExpressJS διευκολύνεται η υλοποίηση διαδικτυακών εφαρμογών που χρησιμοποιούν το Nodejs, δηλαδή μερικά από τα χαρακτηριστικά που προσφέρει είναι ότι διευκολύνει το διαχωρισμό των αιτήσεων που δέχεται ο server σε διαφορετικά μονοπάτια (routes) και κάνει εύκολο τον καθορισμό των διαδικτυακών ρυθμίσεων όπως είναι για παράδειγμα τα port που χρησιμοποιείται για τη σύνδεση του server.

Η έκδοση του ExpressJS που χρησιμοποιήθηκε είναι η v4.16.2 και ο σκοπός χρήσης του εργαλείου αυτού είναι για τη δημιουργία του server-side.

### 4.4 Javascript

Η Javascript [62] είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού και ανήκει μαζί με την HTML και το CSS στις κύριες τεχνολογίες που χρησιμοποιούνται στο διαδίκτυο. Δημιουργήθηκε από τον Brendan Eich και κυκλοφόρησε το 1995 αρχικά με το όνομα LiveScript και στη συνέχεια μετονομάστηκε σε Javascript. Η γλώσσα αυτή χρησιμοποιείται έχει ευρεία εφαρμογή και σήμερα χρησιμοποιείται:

- στη μεριά του πελάτη για να δημιουργηθούν σενάρια και οι ιστοσελίδες να γίνουν διαδραστικές ώστε να μπορεί να αλληλεπιδρά με αυτές ο χρήστης.
- στη μεριά του server τόσο για κατασκευή διαδικτυακών server, για παράδειγμα η τεχνολογία NodeJS χρησιμοποιεί Javascript, όσο και από βάσεις δεδομένων με τις οποίες επικοινωνεί ο server.
- από προγράμματα επεξεργασίας κειμένου και λογισμικά για το άνοιγμα αρχείων με PDF διαμόρφωση.



- και από περιβάλλοντα χρόνου εκτέλεσης (runtime environments) κάτι που κάνει τη Javascript δυνατόν να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών για κινητά ή υπολογιστές.

Στην παρούσα διπλωματική εργασία, η Javascript χρησιμοποιήθηκε τόσο στο front-end ώστε να γίνει αλληλεπιδραστική η σελίδα του χρήστη όσο και στο back-end για την σύνταξη του κώδικα που χρησιμοποιεί ο server.

## 4.5 Επιπλέον βιβλιοθήκες

Για την ανάπτυξη της εφαρμογής που περιγράφεται σε αυτή τη διπλωματική χρησιμοποιούνται κάποιες επιπλέον βιβλιοθήκες που είναι διαθέσιμες μέσω του διαχειριστή πακέτων NPM του Node. Αυτές είναι οι:

- mongodb [63] και assert [64] που χρησιμοποιούνται για τη σύνδεση της μεριάς του server με την τοπική βάση και την επιβεβαίωσή της.
- body-parser [65] που λειτουργεί ως διαμεσολαβητής και κάνει πιο εύκολη την επεξεργασία των αιτήσεων που δέχεται ο server.
- nodemon [66] το οποίο διευκόλυνε την ανάπτυξη και τον έλεγχο της λειτουργίας της εφαρμογής καθώς όταν γινόταν μια διόρθωση στον κώδικα γινόταν αυτόματη επανακίνηση του server.
- js-sha256 [67] η οποία χρησιμοποιήθηκε για τον υπολογισμό των hash τιμών που αποθηκεύονται στα συμβόλαια.

## 4.6 MongoDB

Η MongoDB [68] είναι μια ανοικτού κώδικα διαπλατφορμική εγγραφοκεντρική βάση δεδομένων. Η βάση αυτή διαφέρει από τις σχεσιακές βάσεις δεδομένων καθώς η δομή της στηρίζεται σε έγγραφα και συλλογές και όχι σε πίνακες γι' αυτό και κατηγοριοποιείται στις NoSQL τύπου βάσεις δεδομένων. Το ανάλογο μιας εγγραφής (record) ενός πίνακα σε μια σχεσιακή βάση δεδομένων είναι ένα έγγραφο (document) που είναι αποθηκευμένο στη βάση MongoDB. Τα έγγραφα της MongoDB είναι παρόμοια με τα αντικείμενα της μορφής JSON, ονομάζονται BSON, και περιέχουν πεδία που απαρτίζονται από ζεύγη κλειδιών-τιμών (key-value pairs). Τα πεδία αυτά μπορούν να περιέχουν άλλα έγγραφα, πίνακες καθώς και πίνακες από έγγραφα. Στη MongoDB τα έγγραφα μπορούν να οργανωθούν σε group τα οποία ονομάζονται συλλογές (collection). Το αντίστοιχο μιας συλλογής που χρησιμοποιείται σε μια βάση MongoDB είναι ένας πίνακας (table) μιας σχεσιακής βάσης δεδομένων.

Τα κύρια πλεονεκτήματα μιας MongoDB βάσης είναι [69]:

- Υψηλή απόδοση: Το MongoDB υποστηρίζει ενσωματωμένα μοντέλα δεδομένων που μειώνουν τις λειτουργίες εισόδου/εξόδου ( I/O ) σε ένα σύστημα βάσης δεδομένων, καθώς και δείκτες (indexes) οι οποίοι μπορούν να δεικτοδοτήσουν κλειδιά και σε ενσωματωμένα έγγραφα (documents) για ταχύτερη εκτέλεση ερωτημάτων.
- Υψηλή διαθεσιμότητα: Για να παρέχει υψηλή διαθεσιμότητα το MongoDB έχει μια υπηρεσία λειτουργίας αντιγράφων της βάσης (replication facility) που είναι γνωστή και ως “replica sets” και η οποία παρέχει τόσο αυτόματη ανάκαμψη από βλάβες όσο και πλεονασμό δεδομένων. Το replica set είναι μια ομάδα από

MongoDB servers οι οποίοι διατηρούν το ίδιο σετ δεδομένων και παρέχουν τόσο πλεονασμό όσο και αυξημένη διαθεσιμότητα δεδομένων.

- Αυτόματη κλιμάκωση: Το MongoDB παρέχει οριζόντια κλιμάκωση σαν ένα μέρος από τις κύριες λειτουργίες του. Αυτή η λειτουργία επιτυγχάνεται με την αυτόματη δυνατότητα κατακερματισμού των δεδομένων (sharding) που επιτρέπει την αποθήκευση των δεδομένων σε πολλά διαφορετικά μηχανήματα. Με αυτόν τον τρόπο μπορούν να υποστηριχθούν εφαρμογές που διαχειρίζονται μεγάλο όγκο δεδομένων.

Το MongoDB δημιουργήθηκε στην πρώτη δεκαετία του 2000 και ο λόγος για τη δημιουργία αυτής της τεχνολογίας ήταν εξαιτίας των προβλημάτων κλιμάκωσης που παρουσίαζαν οι παραδοσιακές σχεσιακές βάσεις δεδομένων όταν οι ιδρυτές της MongoDB κατασκεύαζαν web εφαρμογές. Το MongoDB κυκλοφόρησε το 2009 και εκδόθηκε κάτω από το συνδυασμό αδειών GNU Affero General Public License [70] και Apache License [71]. Σήμερα πολλές γνωστές εταιρίες χρησιμοποιούν το MongoDB μερικές από τις οποίες είναι οι Google, Facebook, Nokia, ebay κ.α. [72].

Στην εφαρμογή που παρουσιάζεται στην παρούσα διπλωματική η απαίτηση για ύπαρξη μιας τοπικής βάσης δεδομένων δεν είναι απαραίτητη, καθώς όλα τα στοιχεία υπάρχουν στα έξυπνα συμβόλαια που βρίσκονται στο blockchain. Όμως επιλέχθηκε η χρήση της, ώστε ναδειχθεί η αρμονική λειτουργία αυτής με την εφαρμογή και επιπλέον διότι ίσως κάποια στοιχεία θα μπορούσαν να διαβαστούν από την τοπική βάση, για παράδειγμα αν ένα ερώτημα έχει ξαναγίνει, ώστε να μην χρειάζεται συνεχής αλληλεπίδραση με το blockchain και έτσι να μειωθεί η ταχύτητα απόκρισης της εφαρμογής. Για τη λειτουργία της εφαρμογής δημιουργήθηκε μέσω της εφαρμογής MongoDB μια βάση με όνομα “Diplomatiki” στην οποία δημιουργήθηκαν δύο συλλογές (collection). Οι συλλογές είναι το αντίστοιχο ενός πίνακα σε μια σχεσιακή βάση δεδομένων. Η μια συλλογή ονομάστηκε “demo” και περιέχει πληροφορίες για τα ερωτήματα στα οποία ο χρήστης επέλεξε ένα General contract και η άλλη ονομάστηκε “demoPrivate” και περιέχει πληροφορίες για τα ερωτήματα στα οποία ο χρήστης επέλεξε ένα Personal contract.

Κάθε έγγραφο που αποθηκεύεται στη συλλογή “demo” αποτελείται από τα εξής πεδία:

- “\_id”: είναι το ObjectId το οποίο δημιουργείται αυτόματα από τη βάση κατά την εισαγωγή ενός νέου εγγράφου και είναι μοναδικό για κάθε έγγραφο.
- “medicalBase”: συμπληρώνεται με το όνομα της βάσης (PubMed ή Carre) στην οποία επέλεξε ο χρήστης να κάνει το ερώτημα.
- “query” : συμπληρώνεται με το ερώτημα που υπέβαλε ο χρήστης στη σελίδα της εφαρμογής.
- “answerstring”: είναι ένας πίνακας γραμμή ο οποίος συμπληρώνεται με την hash τιμή της απάντησης που επέστρεψε η βάση (PubMed ή Carre) μετά την υποβολή σε αυτή, του ερωτήματος του χρήστη. Κάθε φορά που υποβάλλεται το ίδιο ερώτημα στην ίδια βάση και επιστρέφεται μια νέα απάντηση, η hash τιμή της νέας αυτής απάντησης αποθηκεύεται στο τέλος του πίνακα. Αυτή η τιμή είναι και αυτή που αποθηκεύεται και στο συμβόλαιο. Στην περίπτωση του General contract εκτός από αυτή την τιμή μπορεί να χρειαστεί να αποθηκευτεί και η hash τιμή του

κειμένου “ερώτημα + όνομα της βάση που υποβλήθηκε το ερώτημα” αν το ερώτημα είναι καινούριο και συνεπώς δεν υπάρχει αποθηκευμένο στο συμβόλαιο.

- “transactionHash”: είναι ένας πίνακας γραμμή ο οποίος συμπληρώνεται με το transactionHash που παράγεται όταν εισάγεται μια καινούρια απάντηση στο General συμβόλαιο. Το transactionHash είναι ένα αναγνωριστικό μοναδικό για κάθε συναλλαγή στο Ethereum blockchain και το οποίο μπορεί να χρησιμοποιηθεί για να βρεθούν πληροφορίες για μια συγκεκριμένη συναλλαγή.

Κάθε έγγραφο που αποθηκεύεται στη συλλογή “demoPrivate” αποτελείται από τα εξής πεδία:

- “\_id”: είναι το ObjectId το οποίο δημιουργείται αυτόματα από τη βάση κατά την εισαγωγή ενός νέου εγγράφου και είναι μοναδικό για κάθε έγγραφο
- “medicalBase”: συμπληρώνεται με το όνομα της βάσης (PubMed ή Carre) στην οποία επέλεξε ο χρήστης να κάνει το ερώτημα.
- “query” : συμπληρώνεται με το ερώτημα που υπέβαλε ο χρήστης στη σελίδα της εφαρμογής.
- “answerstring”: συμπληρώνεται με την hash τιμή της απάντησης που επέστρεψε η βάση (PubMed ή Carre) μετά την υποβολή σε αυτή, του ερωτήματος του χρήστη. Αυτή η τιμή είναι και αυτή που αποθηκεύεται και στο συμβόλαιο.
- “transactionHash” : συμπληρώνεται με το transactionHash που παράγεται όταν αλληλεπιδρούμε με το συμβόλαιο Factoryproject το οποίο δημιουργεί συμβόλαια τύπου Personal τα οποία τοποθετεί στο blockchain. Το συμβόλαιο Factoryproject το έχουμε κάνει deploy όπως και το συμβόλαιο General, στο δίκτυο του blockchain και συνεπώς αυτό έχει μια αποκτήσει μια συγκεκριμένη διεύθυνση στο δίκτυο. Το transactionHash είναι ένα αναγνωριστικό μοναδικό για κάθε συναλλαγή στο Ethereum blockchain και το οποίο μπορεί να χρησιμοποιηθεί για να βρεθούν πληροφορίες για μια συγκεκριμένη συναλλαγή.
- “contract\_address” : συμπληρώνεται με τη διεύθυνση του Personal contract που δημιουργείται όταν αλληλεπιδρούμε με την συνάρτηση του συμβολαίου Factoryproject ορίζοντας τις ανάλογες εισόδους για τη δημιουργία ενός νέου συμβολαίου τύπου Personal, η συνάρτηση δημιουργεί ένα συμβόλαιο τύπου Personal και το τοποθετεί στο blockchain. Επειδή το συμβόλαιο τοποθετείται στο blockchain αυτό αποκτά μια διεύθυνση στο δίκτυο. Αυτή την διεύθυνση μπορεί να τη χρησιμοποιήσει αργότερα ο χρήστης για να αλληλεπιδράσει με το Personal συμβόλαιο που δημιουργήθηκε.

Συνεπώς επειδή οι συλλογές είναι το αντίστοιχο ενός πίνακα σε μια σχεσιακή βάση δεδομένων, θα μπορούσαμε να αντιστοιχίσουμε τις δύο συλλογές σε πίνακες ως εξής, όπου σε κάθε ένα φαίνεται και ένα παράδειγμα μιας καταχώρησης:

Συλλογή “demo”				
_id	medicalBase	Query	answerstring	transactionHash
5b0034e44ba80d1158e50216d	PubMed	asthma	0x6e8df5dd93f9852d1520986715cc900e1ec06369ecec79ee4b7ad4efed2882b	0x527e4d896084df62cbc54940aff517f0af1e5a05509455e8e9dfaccc1c12e56

**Πίνακας 4.1.** Καταχώρηση συλλογής “demo”

Συλλογή “demoPrivate”					
_id	medica lBase	Q ue ry	answerstring	transcactionHash	Contract_address
5b00374 dba80d1 158e502 16f	Pub Med	as th m a	0x6e8df5dd93f9852d152 0986715cc900e1ec06369 eacea79ee4b7ad4efed288 2b	0x678b3a91727b85f805 8812c8f42972138d8357 7ea34de18a3be641a74ae 273f5	0x65ca0097d3d0 78526f8fcb0ec69 72aac4937b019

**Πίνακας 4.2.** Καταχώρηση συλλογής “demoPrivate”

# ΚΕΦΑΛΑΙΟ 5

## Επίλογος

### 5.1 Συμπεράσματα

Στην παρούσα διπλωματική εργασία έγινε λόγος για μια καινούρια τεχνολογία που συνεχώς αναπτύσσεται τα τελευταία χρόνια, αυτή του blockchain και παρουσιάστηκε ο τρόπος λειτουργίας του. Εκτενής αναφορά έγινε στο Ethereum, μια πλατφόρμα που χρησιμοποιεί την blockchain τεχνολογία και υποστηρίζει τη σύνταξη προγραμματιζόμενων αμετάβλητων τμημάτων κώδικα που ονομάζονται έξυπνα συμβόλαια. Για το λόγο αυτό η πλατφόρμα αυτή θεωρήθηκε κατάλληλη και χρησιμοποιήθηκε για την ανάπτυξη μιας αποκεντρωμένης εφαρμογής, η οποία χρησιμοποιεί ένα front-end για την αλληλεπίδραση με το χρήστη και ένα backend server που προωθεί ερωτήματα σε μια βιοϊατρική βάση δεδομένων. Ο server έχει σχεδιαστεί ώστε να επικοινωνεί με τα έξυπνα συμβόλαια που έχουν τοποθετηθεί στο blockchain δίκτυο και να ανακτά από αυτά ή να αποθηκεύει σε αυτά τα απαραίτητα δεδομένα καθώς και επίσης να στέλνει τις απαραίτητες πληροφορίες για αποθήκευση σε μια τοπική βάση, που έχει κατασκευαστεί με το εργαλείο MongoDB. Η εφαρμογή παρέχει δύο είδη συμβολαίων για κάθε ερώτημα που υποβάλλει ο χρήστης μέσω αυτής, το βασικό και το ενισχυμένο. Τα δεδομένα που εισάγονται στον αποθηκευτικό χώρο των έξυπνων συμβολαίων είναι hash τιμές που σχετίζονται με ερωτήματα που έχουν τεθεί σε ιατρικές βάσεις, συγκεκριμένα σε αυτή τη διπλωματική χρησιμοποιήθηκαν οι βάσεις του PubMed και του CARRE, και τις απαντήσεις που έχουν επιστραφεί για αυτά. Κάθε hash τιμή μπορεί να παραχθεί από μια συγκεκριμένη είσοδο και οποιαδήποτε μικρή μεταβολή στο κείμενο της εισόδου μπορεί να ανιχνευτεί καθώς θα οδηγούσε σε διαφορετική hash τιμή εξόδου. Κάθε hash τιμή που εισέρχεται στα συμβόλαια συνοδεύεται από μια χρονική ένδειξη που υποδεικνύει τη χρονική στιγμή που έγινε η εισαγωγή της σ'αυτά. Στην περίπτωση του βασικού συμβολαίου αυτή δίνεται έμμεσα μέσω του αριθμού της απόδειξης συναλλαγής (transactionHash) που παράγεται και παρέχεται στο χρήστη, ενώ στην περίπτωση του ενισχυμένου συμβολαίου μπορεί να δοθεί άμεσα από τη σελίδα της εφαρμογής μέσω των συναρτήσεων που αυτό υποστηρίζει.

Επίσης τα δεδομένα που επιστρέφονται στον χρήστη θα πρέπει να είναι ψηφιακά υπογεγραμμένα από τον ιδιοκτήτη της εφαρμογής και μέσω ενός παρόχου ψηφιακής πιστοποίησης να πιστοποιείται η σχέση του ιδιοκτήτη της εφαρμογής με τα δημόσια κλειδιά που αυτός χρησιμοποιεί τόσο για την ψηφιακή υπογραφή των δεδομένων που παρέχονται μέσω της εφαρμογής όσο και για τις συναλλαγές που εκτελούνται από την εφαρμογή με το δίκτυο του blockchain.

Με τον τρόπο αυτό η εφαρμογή λειτουργεί σαν μεσάζοντας ανάμεσα στον χρήστη που θέτει ένα ερώτημα και την ιατρική βάση στην οποία προωθούνται τα ερωτήματα και εξασφαλίζει δύο προϋποθέσεις αυτό της ακεραιότητας και της μη αποποίησης παροχής των δεδομένων από την ιατρική βάση. Επιπλέον η λειτουργικότητα

του ενισχυμένου συμβολαίου δίνει τη δυνατότητα στο χρήστη να διαπιστώσει μέσω της hash τιμή της απάντησης αν έχει την πιο πρόσφατη απάντηση για ένα ερώτημα καθώς και το πλήθος και το είδος των hash τιμών των διαφορετικών απαντήσεων που έχουν δοθεί για ένα συγκεκριμένο ερώτημα. Αυτή η δυνατότητα εξακρίβωσης του αριθμού και των hash τιμών των απαντήσεων θα μπορούσε να χρησιμοποιηθεί από κάποιον ερευνητή ώστε αυτός να παρακολουθήσει τη συχνότητα μεταβολής της απάντησης σε ένα ερώτημα και έτσι πιθανώς να εξάγει χρήσιμες πληροφορίες για τα ερωτήματα αυτά.

Θα πρέπει να επισημανθεί ότι τα δεδομένα στα περιεχόμενα του συμβολαίου για κάθε ερώτημα ανανεώνονται ύστερα από τη σχετική υποβολή του ερωτήματος στη βάση από τον χρήστη. Για τον λόγο αυτό καλό είναι να μη στηρίζεται κάποιος στην τελευταία hash τιμή της απάντησης που επιστρέφεται για ένα ερώτημα από το συμβόλαιο για να εξακριβώσει αν έχει την πιο πρόσφατη απάντηση αλλά θα πρέπει να υποβάλλει κάθε φορά το ερώτημα που τον ενδιαφέρει στη βάση. Αυτό μπορεί να αποφευχθεί αν η εφαρμογή χρησιμοποιεί ένα πρόγραμμα που θα αναλαμβάνει να ελέγχει τα ερωτήματα που υπάρχουν στο συμβόλαιο και ανά τακτά χρονικά διαστήματα να υποβάλλει τα ερωτήματα αυτά ξανά στις βάσεις, ώστε το συμβόλαιο να περιέχει τη hash τιμή της πιο πρόσφατης απάντησης η οποία υπάρχει και στη βιοϊατρική βάση.

Τέλος χρήσιμο είναι να αναφερθούν κάποια συμπεράσματα που προέκυψαν από τη διαδικασία ανάπτυξης και υλοποίησης της εφαρμογής. Επιβεβαιώθηκε ότι το Ethereum Ropsten testnet είναι κατάλληλο για την ανάπτυξη μιας δοκιμαστικής εφαρμογής που να επικοινωνεί με το Ethereum blockchain. Αυτό διότι οι χρόνοι αναμονής σε αυτό είναι σχετικά μικροί σε σχέση με το Ethereum Mainnet, η διαδικασία απόκτησης test ether είναι γρήγορη και απλή, κυρίως μέσω της χρήσης ενός faucet, και επίσης δεν είναι δύσκολο κάποιος να εγκαταστήσει τα απαραίτητα εργαλεία για να επικοινωνήσει με αυτό. Όμως ένα μειονέκτημά του είναι ότι είναι ευάλωτο σε επιθέσεις κακόβουλων χρηστών, οι οποίοι μπλοκάρουν το δίκτυο με την εκτέλεση μη χρήσιμων συναλλαγών, ανεβάζοντας το κόστος για την εκτέλεση μιας συναλλαγής με αποτέλεσμα πολλές συναλλαγές των χρηστών να μην μπορούν να εκτελεστούν. Παράλληλα έγινε φανερό ότι τα έξυπνα συμβόλαια αν προγραμματιστούν σωστά και λόγω της μη δυνατότητας τροποποίησης του κώδικα τους, είναι κατάλληλα να αποθηκεύσουν αμετάβλητα και για πάντα τιμές hash, οπότε αποτελούν μια καλή λύση για να χρησιμοποιηθούν ως εργαλείο για την επίτευξη της ακεραιότητας.

## 5.2 Πιθανές επεκτάσεις της διπλωματικής

Η εφαρμογή που αναπτύχθηκε σε αυτή τη διπλωματική εξασφαλίζει κυρίως την ακεραιότητα και την μη αποποίηση των δεδομένων που παρέχονται μέσω αυτής στους χρήστες της. Επίσης μέσα από το ενισχυμένο συμβόλαιο προσφέρει και έλεγχο των εκδόσεων των hash τιμών των απαντήσεων που δόθηκαν για ένα ερώτημα. Τα δεδομένα που χρησιμοποιήθηκαν προέρχονταν από βιοϊατρικές βάσεις δεδομένων και πιο συγκεκριμένα από τη πλατφόρμα του CARRE και του PubMed. Όμως η λειτουργικότητα της εφαρμογής θα μπορούσε να διευρυνθεί και σε άλλους τομείς εκτός του ιατρικού, δηλαδή και σε άλλα είδη βάσεων που παρέχουν πληροφορίες και γενικότερα οπουδήποτε χρειάζεται να εξασφαλιστεί η αμεταβλητότητα των δεδομένων μετά την αποστολή τους και ότι ο πάροχος δεν μπορεί να αρνηθεί ότι μια συγκεκριμένη χρονική στιγμή επέστρεψε τα συγκεκριμένα δεδομένα. Για παράδειγμα η hash τιμή που αποθηκεύεται στα συμβόλαια θα μπορούσε να προέρχεται από στρατιωτικές ή κυβερνητικές βάσεις,



όπου είναι σημαντικό να εξασφαλίζεται η εγκυρότητα και αμεταβλητότητα των περιεχομένων τους. Επίσης η hash τιμή θα μπορούσε να προκύπτει από ένα συμφωνητικό μεταξύ δύο μερών, όπως για παράδειγμα μεταξύ εργοδότη και εργαζομένου και, αν το hash αποθηκευόταν στο ενισχυμένο συμβόλαιο, θα μπορούσε να αποθηκεύει για μια συμφωνία μεταξύ δύο συγκεκριμένων ατόμων τις διαφορετικές τιμές των hash που προέκυψαν από την αλλαγή στα συμφωνητικά που έκαναν τα δύο μέρη κατά τη διάρκεια ενός χρονικού διαστήματος.

Επιπλέον όσον αφορά το βασικό συμβόλαιο που χρησιμοποιείται στην εφαρμογή αντί να αποθηκεύεται μόνο η hash τιμή που προκύπτει από το κείμενο της ερώτησης που τέθηκε και της απάντησης που επιστρέφεται για αυτό από την βιοϊατρική βάση θα μπορούσε να αποθηκεύεται η τιμή του hash που θα παράγεται από ένα συνδυασμό της ερώτησης, της αντίστοιχης απάντησης και ενός διαπιστευτηρίου του χρήστη. Με τον τρόπο αυτό η πληροφορία που θα υπάρχει στο συμβόλαιο που θα δημιουργηθεί θα συνδέεται με ένα συγκεκριμένο χρήστη.

Ακόμα θα μπορούσε να εισαχθεί ένα πιστωτικό σύστημα ώστε τα χρήματα για την αλληλεπίδραση των συμβολαίων να μην παρέχονται από τον ιδιοκτήτη της εφαρμογής αλλά αυτή να αποκτήσει μια μορφή υπηρεσίας όπου χρειάζεται να πληρώσει κάποιος ένα αντίτιμο για να την χρησιμοποιήσει. Αν και η εφαρμογή έχει σχεδιαστεί ώστε να μην απαιτεί υψηλό κόστος για τη λειτουργία της, μια καλή πρόταση θα ήταν η δημιουργία ενός ιδιωτικού blockchain δικτύου μεταξύ ρυθμιστών του συστήματος υγείας, μερικά παραδείγματα των ρυθμιστών αυτών είναι τα ιδρύματα υγειονομικής περίθαλψης και οι οργανισμοί ιατρικής έρευνας, ώστε να μειωθεί το κόστος.

## ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

- [1] P. Mytis-Gkometh, G. Drosatos, P. S. Efraimidis, E. Kaldoudi, Notarization of Knowledge Retrieval from Biomedical Repositories Using Blockchain Technology, Vol. 66 of IFMBE Proceedings, Springer Singapore, 2018, pp.69–73. doi:10.1007/978-981-10-7419-6\_12
- [2] W. G. Williams, Uses and limitations of registry and academic databases, Seminars in Thoracic and Cardiovascular Surgery: Pediatric Cardiac Surgery Annual 13 (1) (2010) 66-70. doi:10.1053/j.pcsu.2010.02.007.
- [3] G. Duck, G. Nenadic, M. Filannino, A. Brass, D. L. Robertson, R. Stevens, A survey of bioinformatics database and software usage through mining the literature, PLOS ONE 11 (6) (2016) 1-25. doi:10.1371/journal.pone.0157989.
- [4] K. Vaughan, K. L. Sclaro, H. N. Anksorus, M. W. Roederer, An evaluation of pharmacogenomic information provided by five common drug information resources, Journal of the Medical Library Association: JMLA 102 (1) (2014) 47. doi:10.3163/1536-5050.102.1.009.
- [5] E. P. Go, Database resources in metabolomics: An overview, Journal of Neuroimmune Pharmacology 5 (1) (2010) 18-30. doi:10.1007/360 s11481-009-9157-3.
- [6] M. E. Falagas, E. I. Pitsouni, G. A. Malietzis, G. Pappas, Comparison of pubmed, scopus, web of science, and google scholar: strengths and weaknesses, The FASEB Journal 22 (2) (2008) 338-342, PMID: 17884971. doi:10.1096/fj.07-9492LSF.
- [7] Blockchain, (last edited on 13 June 2018), <https://en.wikipedia.org/wiki/Blockchain>
- [8] V. Buterin, (2014), *A next-generation smart contract and decentralized application platform*, <https://github.com/ethereum/wiki/wiki/White-Paper> (τελευταία πρόσβαση 25 /4/ 2018)
- [9] <https://testnet.etherscan.io/>
- [10] Proof-of-authority Improving the Efficiency and Reliability of Digital Time-Stamping( last edited on 23 May 2018), <https://en.wikipedia.org/wiki/Proof-of-authority>
- [11] PubMed Medline Database, <https://www.ncbi.nlm.nih.gov/pubmed/> (τελευταία πρόσβαση 25 /5/ 2018)
- [12] CARRE Project, *CARRE risk factor reference repository*, <https://www.carre-project.eu/innovation/carre-risk-factor-entry-system>, FP7 EU project (FP7-ICT-611140) (2016). (τελευταία πρόσβαση 25 /5/ 2018)
- [13] Bayer D., Haber S., Stornetta W. S.,( 1992), Improving the Efficiency and Reliability of Digital Time-Stamping, [https://link.springer.com/chapter/10.1007/978-1-4613-9323-8\\_24](https://link.springer.com/chapter/10.1007/978-1-4613-9323-8_24)
- [14] S. Nakamoto, (2008), *Bitcoin: A peer-to-peer electronic cash system*, <https://bitcoin.org/bitcoin.pdf> (τελευταία πρόσβαση 20/5/2018)
- [15] *Θέματα Διπλωματικών Εργασιών* <http://academics.epu.ntua.gr/LinkClick.aspx?fileticket=PPZxteQyC24%3D&tabid=386&mid=1430> (τελευταία πρόσβαση 20 /4/ 2018)
- [16] Βλασσόπουλος Α, (2017), Η τεχνολογία του bitcoin φέρνει επανάσταση στις συναλλαγές, Accountancy Greece, Τεύχος 24,

- [https://issuu.com/accountancygreece/docs/ag\\_024\\_web](https://issuu.com/accountancygreece/docs/ag_024_web) (τελευταία πρόσβαση 10/ 5/ 2018)
- [17] Υπουργείο Οικονομικών, Βασικές Οδηγγίες για την ασφάλεια πληροφοριών H/Y,εγχειρίδιο αρ.1,  
[http://www.mof.gov.cy/mof/dits/dits.nsf/All/A1E16919C0781BECC2257D6A002FA8A0/\\$file/%CE%95%CE%B3%CF%87%CE%B5%CE%B9%CF%81%CE%AF%CE%B4%CE%B9%CE%BF%20%CE%91%CF%83%CF%86%CE%AC%CE%BB%CE%B5%CE%B9%CE%B1%20%CE%A0%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%B9%CF%8E%CE%BD.pdf](http://www.mof.gov.cy/mof/dits/dits.nsf/All/A1E16919C0781BECC2257D6A002FA8A0/$file/%CE%95%CE%B3%CF%87%CE%B5%CE%B9%CF%81%CE%AF%CE%B4%CE%B9%CE%BF%20%CE%91%CF%83%CF%86%CE%AC%CE%BB%CE%B5%CE%B9%CE%B1%20%CE%A0%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%B9%CF%8E%CE%BD.pdf)
  - [18] Συνάρτηση κατατεμαχισμού, (Τελευταία τροποποίηση 2017)  
[el.wikipedia.org/wiki/Συνάρτηση\\_κατατεμαχισμού](http://el.wikipedia.org/wiki/Συνάρτηση_κατατεμαχισμού)
  - [19] Ashley P. & Vandenwauver M., (1999), *Practical Intranet Security – Overview of the State of the Art and Available Technologies*, Boston: Kluwer Academic Publishers
  - [20] ISO 7498-2. Information processing - Open systems interconnection Basic reference model - Part 2: Security architecture. International Organization for Standardization, Geneva, Switzerland, 1989 (first edition).
  - [21] Ψηφιακή υπογραφή,  
[https://el.wikipedia.org/wiki/%CE%A8%CE%B7%CF%86%CE%B9%CE%B1%CE%BA%CE%AE\\_%CF%85%CF%80%CE%BF%CE%B3%CF%81%CE%B1%CF%86%CE%AE](https://el.wikipedia.org/wiki/%CE%A8%CE%B7%CF%86%CE%B9%CE%B1%CE%BA%CE%AE_%CF%85%CF%80%CE%BF%CE%B3%CF%81%CE%B1%CF%86%CE%AE), (Τελευταία τροποποίηση 17 Απριλίου 2018)
  - [22] Κρυπτογράφηση Δημόσιου Κλειδιού,  
[https://el.wikipedia.org/wiki/%CE%9A%CF%81%CF%85%CF%80%CF%84%CE%BF%CE%B3%CF%81%CE%AC%CF%86%CE%B7%CF%83%CE%B7\\_%CE%94%CE%B7%CE%BC%CF%8C%CF%83%CE%B9%CE%BF%CF%85\\_%CE%9A%CE%BB%CE%B5%CE%B9%CE%B4%CE%B9%CE%BF%CF%8D](https://el.wikipedia.org/wiki/%CE%9A%CF%81%CF%85%CF%80%CF%84%CE%BF%CE%B3%CF%81%CE%AC%CF%86%CE%B7%CF%83%CE%B7_%CE%94%CE%B7%CE%BC%CF%8C%CF%83%CE%B9%CE%BF%CF%85_%CE%9A%CE%BB%CE%B5%CE%B9%CE%B4%CE%B9%CE%BF%CF%8D), (Τελευταία τροποποίηση 12 Οκτωβρίου 2016)
  - [23] Technical background of version 1 Bitcoin addresses  
[https://en.bitcoin.it/wiki/Technical\\_background\\_of\\_version\\_1\\_Bitcoin\\_addresses#How\\_to\\_create\\_Bitcoin\\_Address](https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses#How_to_create_Bitcoin_Address)(last edited on 6 June 2018)
  - [24] Ethereum yellow paper, <https://ethereum.github.io/yellowpaper/paper.pdf>
  - [25] Madeira A, (2018), *The DAO, The Hack, The Soft Fork and The Hard Fork*  
<https://www.cryptocompare.com/coins/guides/the-dao-the-hack-the-soft-fork-and-the-hard-fork/> (τελευταία πρόσβαση 25 /5/ 2018)
  - [26] Smart Contracts: The Blockchain Technology That Will Replace Lawyers (2017)  
<https://blockgeeks.com/guides/smart-contracts/> (τελευταία πρόσβαση 20 /4/ 2018)
  - [27] Solidity [2018], <https://en.wikipedia.org/wiki/Solidity> (τελευταία πρόσβαση 20/5/2018)
  - [28] Application Binary Interface Specification Solidity, (2018),  
<http://solidity.readthedocs.io/en/v0.4.24/abi-spec.html> (τελευταία πρόσβαση 20/5/2018)
  - [29] Application binary interface,(2018),  
[https://en.wikipedia.org/wiki/Application\\_binary\\_interface](https://en.wikipedia.org/wiki/Application_binary_interface) (τελευταία πρόσβαση 20/5/2018)
  - [30] JavaScript API, (2018), <https://github.com/ethereum/wiki/wiki/JavaScript-API>

- [31] Buterin V., (2018), Ethereum scalability research and development subsidy programs, <https://blog.ethereum.org/2018/01/02/ethereum-scalability-research-development-subsidy-programs/>
- [32] Poon J., Buterin V., (2017), Plasma: Scalable Autonomous Smart Contracts, <http://plasma.io/>
- [33] Coleman J., (2015), State Channels, <http://www.jeffcoleman.ca/state-channels/>
- [34] What is the Raiden Network? <https://raiden.network/101.html>
- [35] BitFury Group, 2015 (Version 1.0). *Proof of Stake versus Proof of Work*, white paper, <http://bitfury.com/content/5-white-papers-research/pos-vs-pow-1.0.2.pdf>
- [36] Proof of Stake FAQ, (2018), <https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ> (τελευταία πρόσβαση 30/5/2018)
- [37] Byzantine fault tolerance (2018) [https://en.wikipedia.org/wiki/Byzantine\\_fault\\_tolerance](https://en.wikipedia.org/wiki/Byzantine_fault_tolerance) (τελευταία πρόσβαση 10/6/2018)
- [38] Chen, J. και Micali, S. Algorand (2015), arXiv:1607.01341v9 [cs.CR] 26 May 2017 <https://arxiv.org/pdf/1607.01341.pdf> (τελευταία πρόσβαση 28/5/2018)
- [39] Hyperledger Fabric, <https://www.hyperledger.org/projects/fabric>
- [40] Hyperledger, (2018), <https://en.wikipedia.org/wiki/Hyperledger> (τελευταία πρόσβαση 10/6/2018)
- [41] Architecture Explained (2017), <http://hyperledger-fabric.readthedocs.io/en/release-1.1/arch-deep-dive.html> (τελευταία πρόσβαση 15/4/2018)
- [42] Hyperledger Composer Playground, <https://composer-playground.mybluemix.net/>
- [43] Hyperledger Composer <https://hyperledger.github.io/composer/v0.16/introduction/introduction.html>
- [44] Running migrations, [http://truffleframework.com/docs/getting\\_started/migrations](http://truffleframework.com/docs/getting_started/migrations)
- [45] Ethereum Ropsten Testnet, <https://ropsten.etherscan.io/>
- [46] Ethereum Ropsten Faucet, <http://faucet.ropsten.be:3001/>
- [47] Ethereum Mainnet average gasprice chart, <https://etherscan.io/chart/gasprice>  
<https://ethgasstation.info/>
- [48] Go Ethereum, <https://geth.ethereum.org/>
- [49] Truffle framework, <http://truffleframework.com/>
- [50] Mocha testing framework, <https://mochajs.org/>
- [51] Chai Assertion Library, <http://www.chaijs.com/>
- [52] Truffle Community express-box, <http://truffleframework.com/boxes/express-box>
- [53] Ganache CLI, <https://github.com/trufflesuite/ganache-cli>
- [54] Ganache, <http://truffleframework.com/ganache/>
- [55] Solidity, <http://solidity.readthedocs.io/en/v0.4.23/index.html>
- [56] Web3, <https://github.com/ethereum/wiki/wiki/JavaScript-API>
- [57] web3.js, <https://github.com/ethereum/web3.js/>
- [58] Truffle-contract, <https://github.com/trufflesuite/truffle-contract>
- [59] Βιβλιοθήκη jQuery, <https://jquery.com/>
- [60] ExpressJS framework, <https://expressjs.com/>
- [61] [https://developer.mozilla.org/enUS/docs/Learn/Serverside/Express\\_Nodejs/Introduction](https://developer.mozilla.org/enUS/docs/Learn/Serverside/Express_Nodejs/Introduction)
- [62] Javascript, <https://en.wikipedia.org/wiki/JavaScript>
- [63] Mongodb npm package, <https://www.npmjs.com/package/mongodb>

- [64] Assert npm package, <https://www.npmjs.com/package/assert>
- [65] Body-parser npm package, <https://www.npmjs.com/package/body-parser> (τελευταία πρόσβαση 20/5/2018)
- [66] Nodemon npm package, <https://nodemon.io/> (τελευταία πρόσβαση 20/5/2018)
- [67] js-sha256 npm package, <https://www.npmjs.com/package/js-sha256>
- [68] MongoDB Architecture, (2018), <https://www.mongodb.com/mongodb-architecture>, (τελευταία πρόσβαση 10/5/2018)
- [69] MongoDB and MySQL Compared, (2018) <https://www.mongodb.com/compare/mongodb-mysql> (τελευταία πρόσβαση 20/5/2018)
- [70] GNU Affero General Public License, [https://en.wikipedia.org/wiki/GNU\\_Affero\\_General\\_Public\\_License](https://en.wikipedia.org/wiki/GNU_Affero_General_Public_License) (τελευταία πρόσβαση 30/5/2018)
- [71] Apache License, [https://en.wikipedia.org/wiki/Apache\\_License](https://en.wikipedia.org/wiki/Apache_License) (τελευταία πρόσβαση 11/6/2018)
- [72] <https://www.mongodb.com/who-uses-mongodb> (τελευταία πρόσβαση 10/6/2018)

## ΠΑΡΑΡΤΗΜΑ Ι : Υπολογισμός μηδενικών bytes

Στο παράρτημα αυτό θα δοθεί ο τρόπος υπολογισμού των μηδενικών byte στις τιμές hash και στις τιμές που αφορούν το χρόνο.

► *Για τις τιμές hash.* Οι τιμές των hash που χρησιμοποιήθηκαν για την εισαγωγή τιμών στα συμβόλαια παράχθηκαν με sha256 και συνεπώς είναι μία αναπαράσταση που αποτελείται από 256bits. Για να αποθηκευτεί στα συμβόλαια αυτή η τιμή μετατράπηκε σε δεκαεξαδική αναπαράσταση που αποτελείται μόνο από μικρά γράμματα. Ένα παράδειγμα αυτής της τιμής είναι το "0x948ebec65498f4ca4f497dc55838e2d3ee80f8b0aa989df205ca8df6d78dfb3d". Όπως φαίνεται το κείμενο αυτό αποτελείται από 64 χαρακτήρες όπου κάθε χαρακτήρας θα πρέπει να αντιστοιχεί σε 4 bits ώστε  $4 \cdot 64 = 256\text{bits}$ . Ένα byte αποτελείται από 8bits συνεπώς αν χωρίσουμε το κείμενο της hash τιμής σε διαδοχικές δυάδες και βρούμε μια δυάδα να αποτελείται από δύο μηδενικά τότε 8 bits θα είναι 0 και άρα 1 byte θα είναι μηδενικό. Ένα παράδειγμα μιας τιμής hash με ένα μηδενικό byte είναι το "0x948ebec65498f4ca4f497dc55838e2d3ee80f8b0aa989df205ca8df6d78dfb00" που είναι η τιμή του hash που παρατέθηκε πριν με αλλαγμένους σε μηδέν τους δύο τελικούς χαρακτήρες.

► *Για τις τιμές που αφορούν τον unix χρόνο.* Μια τιμή unix χρόνου, η οποία χρησιμοποιήθηκε στα πειράματα για το κόστος που έγιναν σε αυτή τη διπλωματική, είναι η 1527614740032 η οποία αντιστοιχεί στην δυαδική αναπαράσταση "10110001110101100111011110001011001000000", όπου κάθε χαρακτήρας αντιστοιχεί σε ένα bit. Οι μεταβλητές που αποθηκεύουν χρονικές στιγμές στα συμβόλαια είναι τύπου uint256 και άρα αφορούν ένα μη προσημασμένο ακέραιο που αποτελείται από 256bits. Η αναπαράσταση του αριθμού 1527614740032 απαιτεί 41 bits για να αναπαρασταθεί σε ακέραιο αριθμό από byte θα απαιτούνταν 6 byte διότι  $6 \cdot 8 = 48$  και  $5 \cdot 8 = 40$ . Αν χωρίσουμε τη δυαδική αναπαράσταση σε ομάδες των 8 bit ξεκινώντας από το τέλος θα διαπιστώσουμε ότι καμία από τις 5 ομάδες που προκύπτουν δεν έχει μόνο μηδενικά bit και επίσης το τελευταίο bit με τιμή 1 ανήκει στην 6η ομάδα και επειδή έχει την τιμή 1 ανήκει και αυτό σε μια ομάδα από 8 bit που δεν είναι όλα μηδενικά. Συνεπώς η αναπαράσταση του αριθμού απαιτεί 6 byte που κανένα δεν είναι μηδενικό. Τα 256 bit είναι 32 byte και επομένως η για την αποθήκευση του αριθμού 1527614740032 σε μια μεταβλητή τύπου uint256 απαιτούνται 6 μη μηδενικά byte και 16 μηδενικά byte. Με αυτόν τον τρόπο εάν κάποιος θέλει να είναι ακριβής στους υπολογισμούς του κόστους πρέπει να υπολογίσει ακριβώς τις χρονικές στιγμές για να βρει τον ακριβή αριθμό των byte που είναι μηδενικά.



## ΠΑΡΑΡΤΗΜΑ II : Αρχεία κώδικα, εκτός από αυτά των έξυπνων συμβολαίων και τρόπος εγκατάστασης της εφαρμογής

Επιπλέον του κώδικα των έξυπνων συμβολαίων, η εφαρμογή χρησιμοποιεί για την υλοποίηση της σελίδας που αλληλεπιδρά ο χρήστης τα αρχεία *index.html*, *app.css*, που χρησιμοποιείται για τη μορφοποίηση της σελίδας και *app.js* που χρησιμοποιείται για την εκτέλεση σεναρίων. Επίσης χρησιμοποιεί ένα αρχείο *server.js* για τον καθορισμό της λειτουργίας του server. Τέλος χρησιμοποιεί ένα ακόμα *app.js* αρχείο με το οποίο επιτυγχάνεται η επικοινωνία του server με το blockchain δίκτυο.

Η εφαρμογή εγκαταστάθηκε σε έναν υπολογιστή με λειτουργικό σύστημα Linux Mint 18.2 και τα βήματα που μπορεί να ακολουθήσει κάποιος για την εγκατάστασή της είναι:

1. Εγκατάσταση του node σύμφωνα με της οδηγίες στο site . Οι εντολές που εκτελέστηκαν ήταν

- `curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -`
- `sudo apt-get install -y nodejs`
- `sudo apt-get install -y build-essential`

Για την εκτέλεση αυτών απαιτήθηκε και η εγκατάσταση του curl με την εντολή `sudo apt install curl`.

2. Εγκατάσταση του Truffle σύμφωνα με τις οδηγίες στο site

- `sudo install -g truffle`

3. Εγκατάσταση του git σύμφωνα με τις οδηγίες στο site . Οι εντολές που εκτελέστηκαν είναι

- `sudo apt-get update`
- `sudo apt-get install git`

4. Εγκατάσταση της βάσης MongoDB Community Edition σύμφωνα με τις οδηγίες στο site

- `sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5`
- `echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.6 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.6.list`
- `sudo apt-get update`
- `sudo apt-get install -y mongodb-org`

5. Μέσα από ένα terminal δημιουργούμε κάπου ένα φάκελο έστω με το όνομα “mongodb” και μέσα σε αυτόν δημιουργούμε ένα φάκελο όπου θα τοποθετηθούν τα δεδομένα που θα αποθηκεύει ο server και έστω το όνομα αυτού “data”

6. Από ένα terminal μπαίνουμε στον φάκελο “mongodb” και εκτελούμε την εντολή
  - `mongod --dbpath=data`

Με την εντολή αυτή ανοίγει η βάση για να μπορεί να μπορούμε να αλληλεπιδράσουμε με αυτή.

7. Ανοίγουμε ένα άλλο terminal και εκτελούμε την εντολή “mongo” και σε αυτή δημιουργούμε τη βάση και τα collection που θέλουμε να αποθηκεύσουμε τα δεδομένα. Οδηγίες για τον τρόπο διαχείρισης της βάσης δίνονται στο site

8. Δημιουργία του φακέλου της εφαρμογής. Το όνομα του φακέλου που δόθηκε σε αυτή την εφαρμογή είναι “diplomatikiapp”
9. Από ένα terminal μπαίνουμε μέσα στον φάκελο “diplomatikiapp” και εκτελούμε την εντολή
  - truffle init
10. Στη συνέχεια την εντολή “npm init” και ακολουθούμε τα βήματα στην οθόνη
11. Στον φάκελο “diplomatikiapp” έχει πλέον δημιουργηθεί ένα αρχείο package.json. Το ανοίγουμε και διαμορφώνουμε το πεδίο script ως εξής:

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "start": "nodemon server.js"  
}
```

Αυτό γίνεται για την εκκίνηση αργότερα του server.

12. Στο ίδιο αρχείο package.json τοποθετούμε στο τέλος του πεδίου licence ένα κόμμα και τα απαραίτητα devDependencies και dependencies . Η μορφή του αρχείου κάτω από το licence πεδίο είναι η εξής:

```
"license": "ISC",  
"devDependencies": {  
  "truffle-contract": "^1.1.11",  
  "web3": "^0.20.0"  
},  
"dependencies": {  
  "assert": "^1.4.1",  
  "body-parser": "^1.18.2",  
  "express": "^4.16.2",  
  "js-sha256": "^0.9.0",  
  "mongodb": "^3.1.0-beta4",  
  "nodemon": "^1.14.12"  
}
```

13. Από ένα terminal μπαίνουμε στον φάκελο “diplomatikiapp” και εκτελούμε τις εντολές:
  - npm install
  - npm install –dev
14. Ανοίγουμε τον φάκελο “diplomatikiapp” και τον φάκελο contracts δημιουργούμε τρία αρχεία με τον κώδικα των έξυπνων συμβολαίων και τα οποία είναι Project.sol, Factoryproject.sol και Personal.sol.
15. Μέσα στον φάκελο “diplomatikiapp” δημιουργούμε ένα φάκελο “connection” και εκεί τοποθετούμε το app.js που χρησιμοποιείται για την επικοινωνία του server με το Ethereum blockchain και τα έξυπνα συμβόλαια
16. Μέσα στον φάκελο “migrations” που βρίσκεται στον φάκελο “diplomatikiapp” δημιουργούμε το αρχείο με όνομα “2\_deploy\_contracts” στο οποίο εισάγουμε το εξής κείμενο

```
var Project = artifacts.require("./Project.sol");  
var Factoryproject = artifacts.require("./Factoryproject.sol");
```

```
var Personal=artifacts.require("./Personal.sol");
```

```
module.exports = function(deployer) {  
  deployer.deploy(Project);  
  deployer.deploy(Factoryproject);  
};
```

Το αρχείο αυτό χρησιμοποιείται για την τοποθέτηση των συμβολαίων στο Ethereum blockchain

17. Μέσα στον φάκελο “diplomatikiapp” δημιουργούμε ένα φάκελο “public\_static” και μέσα σε αυτόν τοποθετούμε το αρχείο “index.html” και δημιουργούμε δύο φακέλους. Ο πρώτος έχει το όνομα “javascript” και σε αυτόν τοποθετούμε το αρχείο “app.js” που σχετίζεται με την ενεργοποίηση script στην σελίδα με την οποία αλληλεπιδρά ο χρήστης. Ο δεύτερος φάκελος έχει όνομα “stylesheets” και σε αυτόν τοποθετούμε το αρχείο “app.css” ώστε να μορφοποιηθεί κατάλληλα η σελίδα του front-end μέρους της εφαρμογής.

18. Μέσα στον φάκελο “diplomatikiapp” τοποθετούμε το αρχείο server.js

19. Ανοίγουμε το αρχείο truffle.js και εισάγουμε το εξής κείμενο:

```
module.exports = {  
  networks: {  
    ropsten: {  
      host: "127.0.0.1",  
      port: 8545,  
      network_id: 3,  
      gas: 4700000  
    }  
  }  
}
```

Με αυτόν τον τρόπο μπορεί να επικοινωνεί η εφαρμογή με το δίκτυο του Ethereum Ropsten testnet.

20. Εγκαθιστούμε τον κόμβο Geth σύμφωνα με τις εντολές στο site

- sudo apt-get install software-properties-common
- sudo add-apt-repository -y ppa:ethereum/ethereum
- sudo apt-get update
- sudo apt-get install ethereum

21. Ανοίγουμε ένα terminal και εκκινούμε το κόμβο Geth με την εντολή:

- geth --testnet --syncmode "fast" --rpc --rpcapi="db,eth,net,web3,personal,web3" --rpcaddr="localhost" --rpcport="8545" --rpccorsdomain=""

Οδηγίες γιατί επιλέχθηκαν τα συγκεκριμένα flag μπορεί κάποιος να βρει στη σελίδα <https://github.com/ethereum/wiki/wiki/JSON-RPC#json-rpc-endpoint>

22. Για να μπορούμε να κάνουμε συναλλαγές με το δίκτυο είναι απαραίτητο να συγχρονίσουμε πρώτα με το δίκτυο. Για να διαπιστώσουμε πότε το δίκτυο έχει συγχρονίσει ανοίγουμε ένα άλλο terminal και σε αυτό εκτελούμε την εντολή

- geth attach
- Έπειτα εκτελούμε την εντολή “eth.syncing”. Όταν η εντολή επιστρέψει “false” τότε έχει γίνει συγχρονισμός, αλλιώς επιστρέφει ένα άλλο κείμενο και χρειάζεται

- αργότερα να εκτελέσουμε ξανά την εντολή μέχρι αυτή να επιστρέψει “false” ώστε να ξέρουμε ότι έχει συγχρονίσει
23. Την πρώτη φορά που χρησιμοποιούμε το Geth κόμβο χρειάζεται να δημιουργήσουμε μια διεύθυνση που θα λειτουργεί σαν πορτοφόλι και αυτή θα είναι που θα τοποθετήσουμε σαν διαχειριστή για την εφαρμογή. Έτσι στο ίδιο terminal που εκτελούμε την εντολή στο βήμα 23, εκτελούμε την εντολή “geth account new” και καθορίζουμε έναν κωδικό πρόσβασης. Τη διεύθυνση που επιστράφηκε στην οθόνη την αντιγράφουμε και την τοποθετούμε στα κατάλληλα πεδία μέσα στα συμβόλαια όπου καθορίζεται η διεύθυνση του διαχειριστή. Επίσης χρειάζεται να αλλάξουμε τα κατάλληλα πεδία και στο αρχείο “app.js” που βρίσκεται στον φάκελο “connections” και πιο συγκεκριμένα χρειάζεται να τροποποιήσουμε τον αντίστοιχο κώδικα στις συναρτήσεις privateConAdd και addVal.
  24. Πηγαίνουμε σε μια σελίδα ενός faucet [46] και τοποθετούμε τη διεύθυνση που μας επιστράφηκε ώστε να αποκτήσουμε κάποια test ether για την τοποθέτηση των συμβολαίων στο Ethereum Ropsten testnet και για την αλληλεπίδραση αργότερα με τα συμβόλαια.
  25. Αφού έχουμε συγχρονίσει με το δίκτυο και αφήνουμε το κόμβο Geth να συνεχίσει να εκτελείται ανοίγουμε ένα άλλο terminal πηγαίνουμε στον φάκελο “diplomatikiapp” και εκτελούμε την εντολή “truffle compile” και στη συνέχεια την εντολή “truffle migrate –network ropsten” ώστε να γίνει η τοποθέτηση των συμβολαίων στο blockchain. Μετά την εκτέλεση της εντολής μπορούμε να δούμε τις διευθύνσεις που απέκτησαν τα συμβόλαια στο Ethereum Ropsten testnet
  26. Από ένα terminal μπαίνουμε στον φάκελο “diplomatikiapp” και εκτελούμε την εντολή “npm start” για να εκκινήσει η εφαρμογή
  27. Από ένα φυλλομετρητή μπαίνουμε στην σελίδα "" όπου μπορούμε να αλληλεπιδράσουμε με την εφαρμογή.

Αν κάποιος επιθυμεί να δοκιμάσει την εφαρμογή σε ένα τοπικό blockchain χρειάζεται να εγκαταστήσει το Ganache CLI σύμφωνα με τις οδηγίες στο site . Για να το ενεργοποιήσει χρειάζεται να εκτελέσει την εντολή “ganache-cli” σε ένα terminal και να δώσει αν επιθυμεί κάποια flag τα οποία περιγράφονται στην σελίδα . Χρειάζεται να τροποποιηθούν τα κατάλληλα πεδία στους κώδικες, όπως προηγουμένως, ώστε η πρώτη διεύθυνση που δημιουργείται από το εργαλείο Ganache CLI να καθοριστεί ως η διεύθυνση του διαχειριστή. Επίσης το αρχείο truffle.js πρέπει να τροποποιηθεί ως εξής:

```
module.exports = {
  networks: {
    ganache: {
      host: '127.0.0.1',
      port: 7545,
      network_id: 5777
    }
  }
}
```

Στο αρχείο server.js χρειάζεται να αλλάξει το port από 8545 σε 7545. Τέλος πάλι γίνεται compile με την εντολή “truffle compile” και για την τοποθέτηση των συμβολαίων χρησιμοποιείται τώρα η εντολή “truffle migrate –network ganache”. Αν αργότερα

εκκινήσουμε ξανά το Ganache CLI χρειάζεται να τρέξουμε την εντολή “truffle migrate – reset --network ganache”.