ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

# Ανάπτυξη εφαρμογής Ψηφιακών Πιστοποιητικών με χρήση Έξυπνων Συμβολαίων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**Δημήτρης Κ. Τόλιας**

**Επιβλέπων :** Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

# Ανάπτυξη εφαρμογής Ψηφιακών Πιστοποιητικών με χρήση Έξυπνων Συμβολαίων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**Δημήτρης Κ. Τόλιας**

**Επιβλέπων :** Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 16η Μαρτίου 2023.

| | | |
|---|---|---|
| ..... | ..... | ..... |
| Νεκτάριος Κοζύρης | Δημήτριος Τσουμάκος | Ιωάννης Κωνσταντίνου |
| Καθηγητής Ε.Μ.Π. | Αναπληρωτής Καθηγητής Ε.Μ.Π. | Επίκουρος Καθηγητής Π.Θ. |

Αθήνα, Μάρτιος 2023

.....

**(Δημήτρης Κ. Τόλιας)**

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

# Περίληψη

Λόγω της συνεχούς τεχνολογικής ανάπτυξης, η κοινωνία σήμερα χρειάζεται άτομα με υψηλό επίπεδο μόρφωσης και δεξιοτήτων. Πολλές φορές οι δεξιότητες αυτές αποκτώνται στον ψηφιακό κόσμο και σε περιπτώσεις εργασίας από απόσταση είναι επιθυμητό να μπορούν να αποδεικνύονται και σε αυτόν. Με την χρήση των πιστοποιητικών στην παραδοσιακή τους μορφή είναι εύκολη η πλαστογραφία και η διαδικασία επαλήθευσης τους πολλές φορές καθίσταται δύσκολη και χρονοβόρα.

Για την απόδειξη των ικανοτήτων των ατόμων στον ψηφιακό κόσμο προτείνεται και υλοποιείται μια ολοκληρωμένη διαδικτυακή εφαρμογή αποθήκευσης, έκδοσης και επαλήθευσης ψηφιακών πιστοποιητικών με την βοήθεια της τεχνολογίας του blockchain και των έξυπνων συμβολαίων. Θα εξετασθεί η διαδικασία της ανάπτυξης σε όλο το εύρος της, από την παρουσίαση του προβλήματος, την θεωρητική μελέτη επί του γνωστικού πεδίου, την αναλυτική περιγραφή της προτεινόμενης λύσης με χρήση των σύγχρονων μεθοδολογιών, την παρουσίαση των τεχνολογιών που χρησιμοποιήθηκαν και τελικά την τεχνική υλοποίησή της.



# Λέξεις κλειδιά

Ψηφιακά πιστοποιητικά, Blockchain, Ethereum, Smart Contracts, Web3, Δικτυακή εφαρμογή, Node, React

# Abstract

Due to the continuous development of technology, society today needs people with a high level of education and skills. Many times these skills are acquired in the digital world and they may have to be proven there as well, since there is an increase in remote work. By using the certificates in their traditional form, forgery is easy. Moreover, their verification process is often difficult and time-consuming.

To prove the capabilities of individuals in the digital world, a comprehensive online application for storing, issuing and verifying digital certificates is proposed and implemented with the help of blockchain technology and smart contracts. The development process will be examined in its entirety, from the presentation of the problem, the theoretical study of the knowledge field, the detailed description of the proposed solution using modern methodologies, the presentation of the technologies used and finally its technical implementation.



# Keywords

Digital certificates, Blockchain, Ethereum, Smart Contracts, Web3, Web application, Node, React

# Ευχαριστίες

Με την ολοκλήρωση αυτής της διπλωματικής κλείνει ο κύκλος των προπτυχιακών σπουδών μου στο Πολυτεχνείο. Ένα ταξίδι μοναδικό, γεμάτο γνώσεις, νέες ιδέες, φίλους και χαρές, κόπο, ξενύχτια και επιτυχίες.

Θα ήθελα να ευχαριστήσω,

Τον καθηγητή μου και επιβλέποντα της εργασίας κ. Νεκτάριο Κοζύρη καθώς και την κα. Κατερίνα Δόκα για την βοήθεια και καθοδήγηση που μου παρείχαν.

Τους καθηγητές που με ενέπνευσαν και με συν-κίνησαν.

Τους φίλους που ήταν μαζί μου σε αυτό το ταξίδι.

Τέλος θέλω να ευχαριστήσω την οικογένειά μου και την κοπέλα μου Φανή για την στήριξη και την αγάπη τους, όλα αυτά τα χρόνια.

# Περιεχόμενα

# Κατάλογος εικόνων

# 1 Εισαγωγή στα πιστοποιητικά

## 1.1 Σύντομη ιστορική αναδρομή

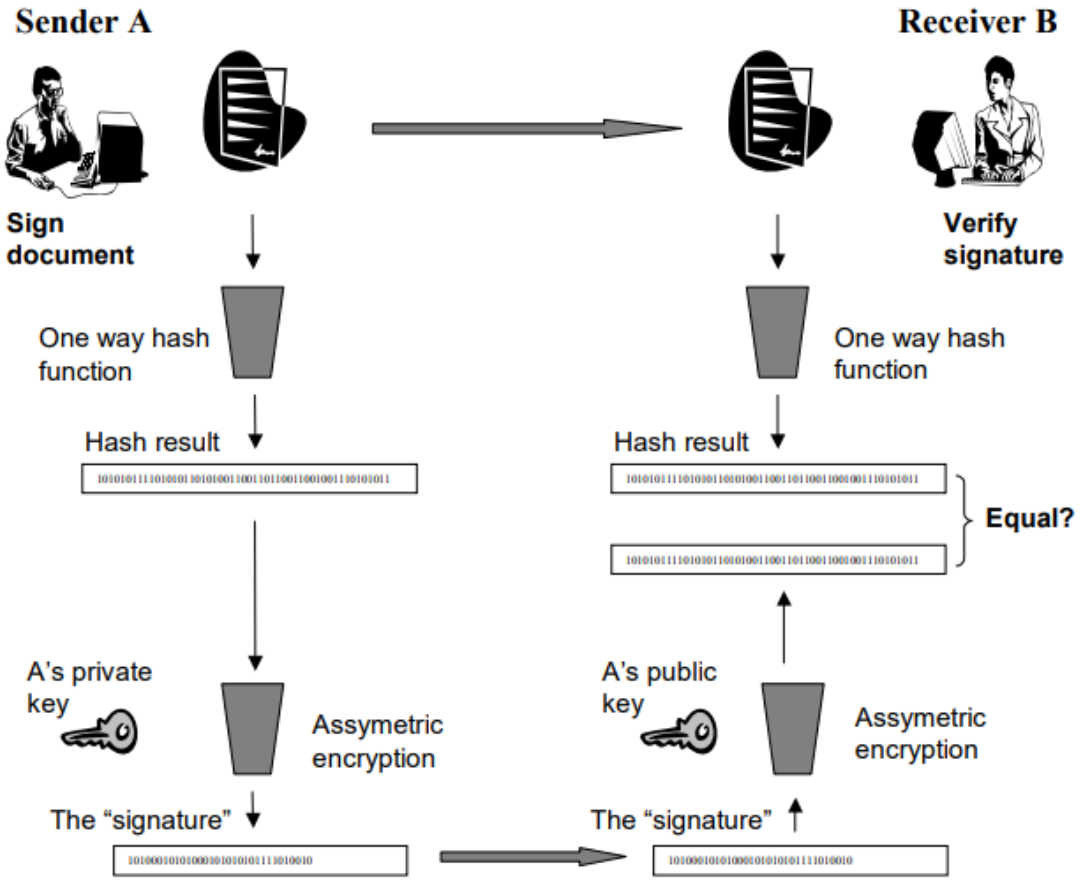
Τα πιστοποιητικά είναι αντικείμενα που πιστοποιούν κάποια ιδιότητα, γνώση, δεξιότητα κάποιου ανθρώπου, οργανισμού ή αντικειμένου. Χρησιμοποιούνται ως απτά στοιχεία που βεβαιώνουν κάποιο γεγονός και συνήθως συμπεριλαμβάνουν και την οντότητα πιστοποίησης που λειτουργεί ως ένας αξιόπιστος επικυρωτής των δεδομένων.

Οι επίσημες πιστοποιήσεις πάνε σχεδόν 500 χρόνια πίσω. Το 1518 ο Thomas Linacre ζήτησε από τον Henry VIII την άδεια να ιδρύσει ένα κολέγιο γιατρών με σκοπό τη χορήγηση αδειών άσκησης επαγγέλματος και να τιμωρήσει τους «ανειδίκευτους» ασκούμενους. Με τον καιρό αυτό έγινε το Βασιλικό Κολλέγιο Ιατρών στο Λονδίνο. Οι υποψήφιοι γιατροί έπρεπε να περάσουν εξετάσεις για να αποδείξουν ότι ήταν κλασικά μορφωμένοι και διέθεταν τις σωστές ιατρικές γνώσεις [1].

Από τα διάφορα είδη πιστοποιητικών που υπάρχουν θα μας απασχολήσουν εκείνα που αφορούν τις γνώσεις και δεξιότητες ατόμων όπως ακαδημαϊκά πτυχία, πιστοποιήσεις σεμιναρίων κ.α.

## 1.2 Ψηφιακά πιστοποιητικά

Καταλυτικό παράγοντα στην δημιουργία τον ψηφιακών πιστοποιητικών αποτέλεσε η κρυπτογράφηση δημοσίου κλειδιού ή ασύμμετρου κλειδιού [2]. Στην κρυπτογράφηση δημοσίου κλειδιού κάθε χρήστης διαθέτει δύο κλειδιά με τα οποία κρυπτογραφεί και αποκρυπτογραφεί δεδομένα και δεν μπορεί να υπολογιστεί το ένα κλειδί από την γνώση του άλλου. Το ιδιωτικό κλειδί πρέπει να διατηρείται μυστικό και το δημόσιο κλειδί δημοσιεύεται σε άλλους χρήστες και αποτελεί την ψηφιακή ταυτότητα του χρήστη στο διαδίκτυο. Τα δεδομένα που κρυπτογραφούνται με το δημόσιο κλειδί μπορούν να αποκρυπτογραφηθούν μόνο από τον κάτοχο αυτού, με το ιδιωτικό κλειδί του. Αντίστροφα δεδομένα τα οποία κρυπτογραφούνται με το ιδιωτικό κλειδί ενός χρήστη μπορούν να αποκρυπτογραφηθούν από οποιονδήποτε γνωρίζει το δημόσιο κλειδί αυτού του χρήστη. Το τελευταίο αποτελεί την βάση για τις ψηφιακές υπογραφές.




**Εικόνα 1: Διαδικασία ψηφιακής υπογραφής**

Όπως φαίνεται στην παραπάνω εικόνα ο χρήστης υπογράφει ένα ψηφιακό έγγραφο κρυπτογραφώντας με το ιδιωτικό του κλειδί το αποτέλεσμα της συνάρτησης κατακερματισμού πάνω σε αυτό το έγγραφο. Στην συνέχεια στέλνοντας το έγγραφο μαζί με την υπογραφή ο παραλήπτης μπορεί με την αντίστροφη διαδικασία να επαληθεύσει την ψηφιακή υπογραφή του αποστολέα. Εκτός από την ταυτοποίηση του αποστολέα με τις ψηφιακές υπογραφές πετυχαίνουμε και άλλο ένα πολύ σημαντικό αποτέλεσμα, την ακεραιότητα των δεδομένων του ψηφιακού εγγράφου. Αν κάποιος κακόβουλος προσπαθήσει να παραποιήσει το ψηφιακό έγγραφο τότε η υπογραφή αυτόματα ακυρώνεται και δεν μπορεί να πλαστογραφηθεί αφού για την παραγωγή της απαιτείται το ιδιωτικό κλειδί του αποστολέα το οποίο είναι κρυφό.

Έχοντας λοιπόν τα παραπάνω δεδομένα μπορούμε να συνθέσουμε τα ψηφιακά πιστοποιητικά ως ηλεκτρονικά έγγραφα που συνδέουν κάποιους ισχυρισμούς με κάποια οντότητα που χαρακτηρίζεται από το δημόσιο κλειδί της και υπογράφονται από τον εκδότη του ψηφιακού πιστοποιητικού με την ψηφιακή του υπογραφή. Το μόνο ζήτημα που δεν έχει απαντηθεί είναι πως γίνεται η αντιστοίχιση του δημόσιου κλειδιού ενός χρήστη με το φυσικό ή νομικό πρόσωπο. Αυτό γίνεται με την χρήση παρόχων υπηρεσιών εμπιστοσύνης [3] δηλαδή έμπιστων οντοτήτων που με τις δικές τους ψηφιακές υπογραφές πιστοποιούν την αναφερόμενη συσχέτιση.

Με αυτή την μετάβαση λοιπόν των πιστοποιητικών στον ψηφιακό κόσμο μπορεί ο καθένας σήμερα να αποδείξει εύκολα και άμεσα τις ικανότητες του σε οποιονδήποτε άλλο χωρίς γεωγραφικούς περιορισμούς.

## 1.3 Ψηφιακά πιστοποιητικά με χρήση Blockchain

Το blockchain είναι μία τεχνολογία που ανθίζει και βρίσκει εφαρμογές σε διάφορους τομείς. Χρησιμοποιώντας το για τα ψηφιακά πιστοποιητικά, ωφελούμαστε από τα εγγενή χαρακτηριστικά του όπως ιδιωτικότητα, ασφάλεια (ενσωματώνει κρυπτογραφία δημοσίου κλειδιού), ακεραιότητα των δεδομένων, κατανεμημένη αρχιτεκτονική κ.α.

# 2 Τεχνολογίες που χρησιμοποιήθηκαν

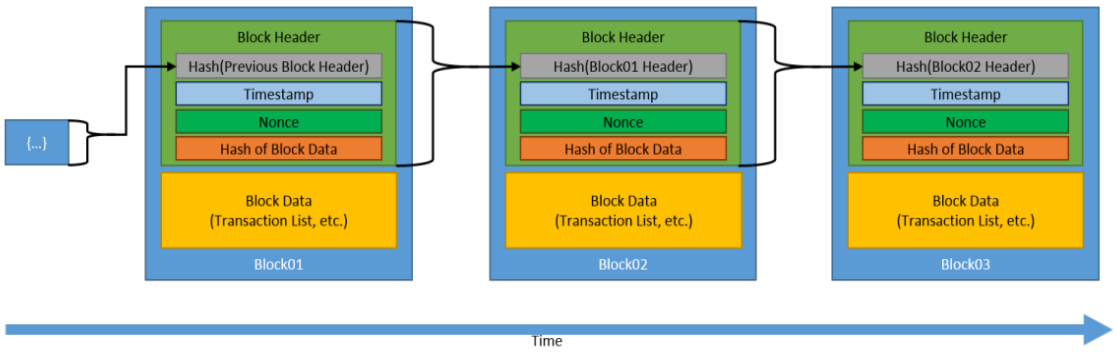
## 2.1 Blockchain

Το blockchain είναι μία κατανεμημένη βάση στην οποία τα δεδομένα αποθηκεύονται σε μπλοκ και συνδέονται μεταξύ τους σχηματίζοντας ένα μη αναστρέψιμο χρονοδιάγραμμα. Η τεχνολογία blockchain περιγράφηκε για πρώτη φορά το 1991 από τους Stuart Haber και W. Scott Stornetta [4], δύο ερευνητές που ήθελαν να εφαρμόσουν ένα σύστημα όπου οι χρονικές σημάνσεις εγγράφων δεν θα μπορούσαν να παραβιαστούν. Βασισμένος στο έργο τους, το 2008 κάποιος με το ψευδώνυμο Satoshi Nakamoto δημοσίευσε μια εργασία [5] στην οποία παρουσίασε το bitcoin, ένα νόμισμα που λύνει το πρόβλημα double-spending [6] χωρίς την ανάγκη ύπαρξης αξιόπιστου τρίτου μέρους. Για την καταγραφή των συναλλαγών αυτού του κρυπτονομίσματος περιγράφει ένα σύστημα blockchain δικτύου ομότιμων κόμβων.

### 2.1.1 Γενική λειτουργία

Οι χρήστες του συστήματος στέλνουν συναλλαγές. Αυτές στην συνέχεια επικυρώνονται ως προς την ορθότητα και την αυθεντικότητα και πολλές μαζί δημιουργούν ένα μπλοκ. Το αποτύπωμα του μπλοκ προκύπτει ως αποτέλεσμα μιας συνάρτησης κατακερματισμού πάνω στα δεδομένα του. Αυτό το αποτύπωμα αποτελεί αναγνωριστικό του μπλοκ αλλά λειτουργεί και ως ψηφιακή υπογραφή για την αναγνώριση αλλοίωσης των δεδομένων. Στην συνέχεια μόλις υπολογισθεί πλήρως το μπλοκ στέλνεται σε όλο το δίκτυο και οι χρήστες υλοποιώντας έναν κατανεμημένο αλγόριθμο ομοφωνίας καταλήγουν σε μια κοινή απόφαση για το αν θα το δεχτούν ή όχι. Αν το δεχτούν το μπλοκ αυτό συνεχίζει την αλυσίδα και συνδέεται με το τελευταίο ως τώρα μπλοκ με αναφορά στο αποτύπωμά του.

### 2.1.2 Μπλοκ




**Εικόνα 2 : Αλυσίδα από μπλοκ**

Τα περιεχόμενα του μπλοκ χωρίζονται σε δύο μέρη, την κεφαλίδα και τα δεδομένα. Η κεφαλίδα του μπλοκ περιέχει μεταδεδομένα για αυτό το μπλοκ. Τα δεδομένα περιέχουν μια λίστα επικυρωμένων και αυθεντικών συναλλαγών οι οποίες έχουν υποβληθεί στο blockchain. Τα περιεχόμενα μπορεί να διαφέρουν ανά τις υλοποιήσεις blockchain ωστόσο τα βασικά πεδία [7] τα οποία αναφέρονται παρακάτω διατηρούνται.

1) *Αποτύπωμα των δεδομένων του μπλοκ*

   Το αποτύπωμα των δεδομένων του μπλοκ παράγεται από την εφαρμογή μιας συνάρτησης κατακερματισμού στα δεδομένα του μπλοκ και στην συνέχεια

αποθηκεύεται στην κεφαλίδα του ώστε με την αλλαγή των δεδομένων να αλλάζει και το αποτύπωμα της κεφαλίδας του μπλοκ.

2) *Αποτύπωμα της κεφαλίδας του προηγούμενου μπλοκ*

Όμοια με παραπάνω το αποτύπωμα της κεφαλίδας είναι το αποτέλεσμα της εφαρμογής μιας συνάρτησης κατακερματισμού στην κεφαλίδα. Βλέπουμε πως το κάθε μπλοκ περιέχει αναφορά στο προηγούμενο μπλοκ αποθηκεύοντας το αποτύπωμα της κεφαλίδας του. Αυτή η πρακτική συνεισφέρει στην ακεραιότητα των δεδομένων. Αν κάποιος αλλάξει τα δεδομένα κάποιου μπλοκ, τότε όλα τα επόμενα δεν θα είναι πλέον έγκυρα καθώς το αποτύπωμα τους θα έχει αλλάξει αφού εξαρτάται από το αποτύπωμα των προηγούμενων μπλοκ. Έτσι για να μεταβάλει κάποιος την κατάσταση του συστήματος θα πρέπει να ανανεώσει εκτός από το μπλοκ που αλλάζει και κάθε επόμενο στην αλυσίδα και σε συνδυασμό με τον μηχανισμό ομοφωνίας η πράξη αυτή γίνεται ακόμα πιο δύσκολη.

3) *Χρονοσφραγίδα*

Οι χρονοσφραγίδες χρησιμοποιούνται στο blockchain για να παρέχουν κάποια χρονική αναφορά για τα γεγονότα, συγχρονισμό αλλά και σε ελέγχους εγκυρότητας.

4) *Nonce*

Αυτό το δεδομένο είναι ένας αριθμός. Για τα δίκτυα blockchain που χρησιμοποιούν εξόρυξη, το χρησιμοποιεί ο κόμβος που δημοσιεύει το μπλοκ για να λύσει το πάζλ κατακερματισμού. Άλλα δίκτυα μπορεί να το χρησιμοποιούν για άλλους σκοπούς.

### 2.1.3 Αλγόριθμοι ομοφωνίας

Υπάρχουν διάφοροι αλγόριθμοι ομοφωνίας αλλά οι δύο επικρατέστεροι είναι οι παρακάτω.

### Proof of work

Με αυτό τον αλγόριθμο ο κόμβος που θα δημοσιεύσει το επόμενο μπλοκ είναι αυτός που θα λύσει ένα παζλ. Το πάζλ αυτό είναι δύσκολο αλλά επιλύεται σε εύλογο χρονικό διάστημα. Είναι σχεδιασμένο έτσι ώστε να μπορεί να επαληθευτεί αρκετά εύκολα και λειτουργεί ως απόδειξη εργασίας του κόμβου για να του δοθεί η δυνατότητα να δημοσιεύσει μπλοκ στο δίκτυο. Για να συμμετέχουν οι κόμβοι σε αυτή την ανταγωνιστική διαδικασία σπαταλώντας πόρους πρέπει να έχουν κάποιο κίνητρο και συνήθως τους παρέχεται κάποιο κρυπτονόμισμα του δικτύου ως ανταμοιβή σε περίπτωση επίλυσης του παζλ. Αν αποτύχουν δεν έχουν την δυνατότητα να δημοσιεύσουν το μπλοκ αλλά ούτε ανταμείβονται για την προσπάθειά τους. Μόλις κάποιος κόμβος λύσει το πάζλ δημοσιεύει το μπλοκ στο δίκτυο, οι υπόλοιποι κόμβοι επαληθεύουν την λύση και αν είναι σωστή το αποθηκεύουν. Αυτή η υπολογιστική πολυπλοκότητα αποτελεί βασικό μηχανισμό ασφάλειας του δικτύου.

## Proof of stake

Αυτή η μέθοδος βασίζεται στην ιδέα ότι όσο περισσότερο έχει επενδύσει κάποιος στο σύστημα τόσο πιθανότερο είναι να θέλει το σύστημα να επιτύχει και να μην λειτουργήσει παραβατικά. Ως stake νοείται το μερίδιο επένδυσης του χρήστη στο σύστημα και σχετίζεται με τα κρυπτονομίσματα που αυτός έχει ξοδέψει στο σύστημα, κλειδώνοντάς τα σε ειδικούς τύπους συναλλαγών ή στέλνοντας τα σε άλλους χρήστες. Έτσι λοιπόν η πιθανότητα να επιλεγεί ένας κόμβος για να δημοσιεύσει ένα μπλοκ σχετίζεται με το ποσοστό επί του συνολικού μεριδίου που ο χρήστης έχει επενδύσει. Θα μπορούσε κανείς να παραλληλίσει αυτή την έννοια με την επιρροή ενός μετόχου σε μία εταιρία ανάλογα με το μετοχικό του κεφάλαιο. Οι μέθοδοι επιλογής κόμβων με βάση το μερίδιο μπορεί να διαφέρουν.

Ενδεικτικά κάποιες από αυτές είναι:

1. Η τυχαία επιλογή ενός κόμβου με πιθανότητα ίση με το ποσοστό των κρυπτονομισμάτων που έχει επενδύσει επί του συνόλου των κρυπτονομισμάτων του δικτύου.

2. Μια παραλλαγή του παραπάνω είναι η προσθήκη στα κρυπτονομίσματα που έχουν επενδυθεί ενός πεδίου χρόνου που σταδιακά αυξάνεται και μόνο όταν περάσει μία συγκεκριμένη τιμή το κρυπτονόμισμα προσμετράται στην μετοχική δύναμη του χρήστη για την επιλογή του και στην συνέχεια επαναφέρεται στην αρχική του τιμή. Με αυτόν τον τρόπο περιορίζεται η δύναμη των μεγαλομετόχων αφού προστίθεται κάποια χρονική περίοδος αναμονής μεταξύ της έκδοσης νέων μπλοκ από αυτούς.

3. Ένα άλλο σύστημα είναι αυτό της εκλογής εκπροσώπων όπου οι χρήστες ψηφίζουν κόμβους ως υποψήφιους για να δημοσιεύουν νέα μπλοκ. Το βάρος της κάθε ψήφου είναι ανάλογο με το μερίδιο επένδυσης του χρήστη και οι κόμβοι με τις περισσότερες ψήφους εκλέγονται και αποκτούν την δυνατότητα να δημοσιεύουν νέα μπλοκ. Οι χρήστες ψηφίζουν και για την απομάκρυνση εκλεγμένων κόμβων οπότε οι τελευταίοι αν δεν δρουν τίμια χάνουν την θέση τους. Τέλος οι κόμβοι ψηφίζουν και εκπροσώπους για την διακυβέρνηση του blockchain καθήκον των οποίων είναι να προτείνουν αλλαγές και βελτιώσεις στο σύστημα οι οποίες στην συνέχεια θέτονται προς ψηφοφορία.

Ο αλγόριθμος αυτός λύνει το πρόβλημα της αυξανόμενης απαίτησης υπολογιστικών πόρων και ενέργειας που είχε ο προηγούμενος. Σε αυτό το σύστημα καθώς οι κόμβοι δεν σπαταλούν πόρους για την παραγωγή νέων μπλοκ συνήθως δεν ανταμείβονται με την απόκτηση νέων κρυπτονομισμάτων αλλά από προμήθεια επί της πληρωμής των χρηστών για την τέλεση των συναλλαγών τους.

Οι μεγαλομέτοχοι μπορούν επενδύοντας όλο και περισσότερο να αποκτούν όλο και μεγαλύτερα μερίδια, ωστόσο για να αποκτήσουν την πλειοψηφία του συστήματος θα χρειάζονταν επενδύσεις τεραστίου κόστους.

| Name | Goals | Advantages | Disadvantages | Domains | Implementations |
|---|---|---|---|---|---|
| **Proof of work (PoW)** | To provide a barrier to publishing blocks in the form of a computationally difficult puzzle to solve to enable transactions between untrusted participants. | Difficult to perform denial of service by flooding network with bad blocks.<br><br>Open to anyone with hardware to solve the puzzle. | Computationally intensive (by design), power consumption, hardware arms race.<br><br>Potential for 51 % attack by obtaining enough computational power. | Permissionless cryptocurrencies | Bitcoin, Ethereum, many more |
| **Proof of stake (PoS)** | To enable a less computationally intensive barrier to publishing blocks, but still enable transactions between untrusted participants. | Less computationally intensive than PoW.<br><br>Open to anyone who wishes to stake cryptocurrencies.<br><br>Stakeholders control the system. | Stakeholders control the system.<br><br>Nothing to prevent formation of a pool of stakeholders to create a centralized power.<br><br>Potential for 51 % attack by obtaining enough financial power. | Permissionless cryptocurrencies | Ethereum Casper, Krypton |
| **Delegated PoS** | To enable a more efficient consensus model through a 'liquid democracy' where participants vote (using cryptographically signed messages) to elect and revoke the rights of delegates to validate and secure the blockchain. | Elected delegates are economically incentivized to remain honest<br><br>More computationally efficient than PoW | Less node diversity than PoW or pure PoS consensus implementations<br><br>Greater security risk for node compromise due to constrained set of operating nodes<br><br>As all delegates are 'known' there may an incentive for block producers to collude and accept bribes, compromising the security of the system | Permissionless cryptocurrencies<br><br>Permissioned Systems | Bitshares, Steem, Cardano, EOS |

**Εικόνα 3 : Σύγκριση αλγορίθμων ομοφωνίας**

### 2.1.4 Τύποι blockchain

## Public ή Permissionless

Τα δίκτυα αυτού του τύπου είναι ανοικτά δίκτυα όπου ο οποιοσδήποτε μπορεί να εισέλθει χωρίς να χρειάζεται την άδεια από κάποια κεντρική αρχή και να δημιουργήσει νέα μπλοκ, συναλλαγές ή να διαβάσει τα περιεχόμενα του blockchain. Αυτή η ελευθερία στον τρόπο λειτουργίας επιτρέπει την επέκτασή και χρήση τους από μεγάλο πλήθος χρηστών ανά την υφήλιο, οι οποίοι στην συνέχεια συλλογικά και χωρίς κάποια κεντρική εξουσία εξασφαλίζουν την καλή λειτουργία του δικτύου. Για την αποτροπή των κακόβουλων χρηστών, χρησιμοποιούνται οι αλγόριθμοι ομοφωνίας οι οποίοι απαιτούν την σπατάλη πόρων ή κατοχής μεριδίου. Σε αυτού του τύπου τα δίκτυα λοιπόν λόγω του μεγέθους των πόρων ή του μεριδίου που απαιτούνται, οι επιθέσεις καθίσταται πρακτικά αδύνατες. Επίσης υπάρχουν μηχανισμοί επιβράβευσης για τους χρήστες που συνεισφέρουν στην ευνοϊκή λειτουργία του δικτύου.

## Private ή permissioned

Τα δίκτυα αυτά είναι κλειστού τύπου και απαιτείται η επικύρωση οποιουδήποτε θέλει να εισέλθει στο σύστημα από κάποια αρχή. Επομένως υφίσταται έλεγχος στο ποιος μπορεί να δημιουργήσει νέα μπλοκ, συναλλαγές και να διαβάσει τα περιεχόμενα του blockchain. Συνήθως λόγω της ελεγχόμενης φύσης των δικτύων αυτών υπάρχει εμπιστοσύνη μεταξύ των χρηστών και οι αλγόριθμοι ομοφωνίας δεν απαιτούν σπατάλη

πόρων ή κατοχή μεριδίου και επομένως αυτά τα δίκτυα έχουν μεγαλύτερη ταχύτητα στις συναλλαγές. Χρησιμοποιούνται από ιδιωτικούς οργανισμούς που χρειάζονται εποπτεία και έλεγχο πάνω στο δίκτυο.

### Hybrid

Τα δίκτυα αυτά αποτελούν υβρίδια των δύο παραπάνω συνδυάζοντας βασικά στοιχεία τόσο των δημοσίων όσο και των ιδιωτικών δικτύων.

### Consortium

Και τα δίκτυα αυτού του τύπου συνδυάζουν στοιχεία δημόσιων και ιδιωτικών δικτύων. Συνήθως εφαρμόζονται από ένα σύνολο οργανισμών που συνεργάζονται προς ένα κοινό στόχο και μπορεί να υπάρχει εμπιστοσύνη (ή και όχι) μεταξύ τους. Μαζί αποφασίζουν για τα μοντέλα εμπιστοσύνης και τα πρωτόκολλα του δικτύου και δημιουργούν και επεκτείνουν το δίκτυο.

| Characterstics | Public blockchain | Private blockchain | Consortium blockchain |
|---|---|---|---|
| Determination of consensus | All miners participating in network | Group head / Lead node | Set of identified nodes |
| Accessibility | Public | Could be public / restricted | Could be public / restricted |
| Efficiency | Low | High | High |
| Immutability | Can't be tampered | Could be tampered | Could be tampered |
| Centralized | No | Yes | Partial |
| Process of consensus | Permission less | Permission required | Permission required |
| Examples | Bitcoin, Ethereum, Litcoin etc | Ripple (XRP) and Hyperledger | Quorum, Hyperledger and Corda. |

**Εικόνα 4 : Σύγκριση τύπων blockchain**

## 2.1.5 Εξέλιξη των τεχνολογιών blockchain

Το blockchain με τον καιρό εξελίσσεται και αναπτύσσεται. Ξεκίνησε ως μια ασφαλής και έμπιστη κατανεμημένη βάση δεδομένων αλλά με τον καιρό νέες ιδέες και τομείς εφαρμογής προέκυπταν αλλάζοντας την κατεύθυνση ανάπτυξης του. Τα πρωτόκολλα ξαναχτίζονταν με βάση τις νέες ανάγκες της τεχνολογίας. Η δεύτερη γενιά αυτής της τεχνολογίας μας οδηγεί στο δίκτυο Ethereum που δημιουργήθηκε ως πλατφόρμα για ένα νέο αποκεντρωμένο διαδίκτυο και με τα έξυπνα συμβόλαια έδωσε την δυνατότητα αυτοματοποιημένης εκτέλεσης προγραμμάτων στο blockchain και κατά συνέπεια δημιουργίας πάνω σε αυτό αποκεντρωμένων εφαρμογών. Τα έργα που αναδύονται σήμερα μπορούν να αναφερθούν ως η τρίτη γενιά τεχνολογίας blockchain. Καθορίζονται από την επεκτασιμότητα, την αστραπιαία επεξεργασία και τις χαμηλές χρεώσεις συναλλαγών. Αυτή η νέα γενιά blockchain χαρακτηρίζεται επίσης από τη διαλειτουργικότητα και τη χαμηλότερη κατανάλωση ενέργειας.

## 2.2 Ethereum

Το Ethereum [8] είναι μια πλατφόρμα για τη δημιουργία εφαρμογών και οργανισμών, τη διατήρηση περιουσιακών στοιχείων, τις συναλλαγές και την επικοινωνία χωρίς τον έλεγχο από κάποια κεντρική αρχή. Έχει το δικό του κρυπτονόμισμα που ονομάζεται Ether, το οποίο χρησιμοποιείται για την πληρωμή ορισμένων δραστηριοτήτων στο δίκτυο. Βασίζεται σε τεχνολογία blockchain, είναι permissionless και χρησιμοποιεί τον αλγόριθμο ομοφωνίας Gasper [9] που είναι proof of stake αλγόριθμος, ενώ παλαιότερα χρησιμοποιούσε proof of work. Σε αντίθεση με το Bitcoin δεν αποτελεί απλά ένα κατανεμημένο καθολικό αλλά μια κατανεμημένη μηχανή πεπερασμένων καταστάσεων που μπορεί να αλλάζει κατάσταση από μπλοκ σε μπλοκ με βάση κάποιους κανόνες. Τους κανόνες αυτούς ορίζει το EVM [10], η εικονική μηχανή του δικτύου πάνω στην οποία εκτελείται ο κώδικας γνωστός ως Smart contract ή έξυπνο συμβόλαιο.

## 2.3 Smart contracts

Ένα έξυπνο συμβόλαιο [11] είναι απλώς ένα πρόγραμμα που εκτελείται στο Ethereum. Είναι μια συλλογή κώδικα (οι λειτουργίες του) και δεδομένων (η κατάστασή του) που βρίσκεται σε μια συγκεκριμένη διεύθυνση στο blockchain. Τα έξυπνα συμβόλαια είναι ένας τύπος λογαριασμού Ethereum. Αυτό σημαίνει ότι έχουν υπόλοιπο και μπορούν να αποτελέσουν στόχο συναλλαγών. Ωστόσο, δεν ελέγχονται από έναν χρήστη, αντίθετα αναπτύσσονται στο δίκτυο και εκτελούνται όπως έχουν προγραμματιστεί. Οι λογαριασμοί χρηστών μπορούν στη συνέχεια να αλληλεπιδράσουν με ένα έξυπνο συμβόλαιο υποβάλλοντας συναλλαγές που εκτελούν μια λειτουργία που ορίζεται στο έξυπνο συμβόλαιο. Τα έξυπνα συμβόλαια μπορούν να ορίζουν κανόνες, όπως ένα κανονικό συμβόλαιο, και να τους επιβάλλουν αυτόματα μέσω του κώδικα. Τα έξυπνα συμβόλαια δεν μπορούν να διαγραφούν από προεπιλογή και οι αλληλεπιδράσεις μαζί τους είναι μη αναστρέψιμες.

Οποιοσδήποτε μπορεί να γράψει ένα έξυπνο συμβόλαιο και να το αποθηκεύσει στο δίκτυο πραγματοποιώντας μία συναλλαγή για την οποία θα πληρώσει κάποιο ποσό σε Ether. Τα έξυπνα συμβόλαια πριν δημοσιευτούν στο Ethereum πρέπει να μεταγλωττιστούν σε κώδικα μηχανής ώστε να μπορεί να τα ερμηνεύσει το EVM.

## 2.4 Solidity

Η Solidity [12] είναι μια αντικειμενοστραφής γλώσσα υψηλού επιπέδου για την υλοποίηση έξυπνων συμβολαίων. Έχει σχεδιαστεί για να στοχεύει την εικονική μηχανή Ethereum (EVM). Έχει επηρεαστεί από τις γλώσσες προγραμματισμού C++, Python και JavaScript. Είναι μια γλώσσα στατικών τύπων δεδομένων, υποστηρίζει κληρονομικότητα, βιβλιοθήκες και σύνθετους τύπους που καθορίζονται από το χρήστη μεταξύ άλλων χαρακτηριστικών.

## 2.5 Ganache

Το εργαλείο αυτό [13] μας παρέχει την δυνατότητα να δημιουργήσουμε εύκολα ένα blockchain τοπικά στον υπολογιστή μας, για την ανάπτυξη και τον έλεγχο αποκεντρωμένων εφαρμογών στο δίκτυο Ethereum. Παρέχει διεπαφή γραμμής εντολών αλλά και γραφική διεπαφή για εύκολη οπτικοποίηση της κατάστασης του blockchain κάθε στιγμή.

## 2.4 Metamask

Το MetaMask [14] είναι ένα ψηφιακό πορτοφόλι κρυπτονομισμάτων. Παρέχεται με την μορφή browser extension αλλά και mobile app. Συνεισφέρει στην διαχείριση και φύλαξη των λογαριασμών μας, στην δημιουργία συναλλαγών και έξυπνων συμβολαίων αλλά και ως μηχανισμός ταυτοποίησης σε αποκεντρωμένες εφαρμογές. Με την γραφική διεπαφή που μας παρέχει, διευκολύνει τον χρήστη να έχει την εποπτεία στις συναλλαγές του, βλέποντας τις λεπτομέρειές τους και επιβεβαιώνοντας τες εάν συμφωνεί ώστε να μην εξαπατηθεί. Δημιουργήθηκε για την αλληλεπίδραση με το Ethereum μέσω RPC (Remote procedure calls) και έρχεται με προεγκατεστημένες συνδέσεις με κόμβους του δικτύου ώστε να μην χρειάζεται να τρέξουμε τον δικό μας κόμβο.

## 2.6 web3.js

Το web3.js είναι ένα API της JavaScript για το Ethereum [15]. Αποτελείται από ένα σύνολο βιβλιοθηκών που μας επιτρέπουν να αλληλεπιδράμε με έναν τοπικό ή απομακρυσμένο κόμβο του Ethereum και να τελούμε συναλλαγές όπως δημιουργία έξυπνων συμβολαίων και αλληλεπίδραση με αυτά, με προγραμματιστικό τρόπο.

## 2.7 React.js

Η React.js είναι μια βιβλιοθήκη της JavaScript για την εύκολη δημιουργία γραφικών διεπαφών τόσο ιστοσελίδων όσο και εφαρμογών κινητών [16]. Καθώς είναι μία δηλωτική γλώσσα σε αντίθεση με την JavaScript, ο προγραμματιστής δεν χρειάζεται να πραγματοποιεί ο ίδιος τις αλλαγές στο δέντρο του εγγράφου αλλά αρκεί να περιγράψει τις λειτουργίες που πρέπει να πραγματοποιηθούν. Συστατικά της είναι τα components, απομονωμένα και επαναχρησιμοποιήσιμα κομμάτια κώδικα που επιτρέπουν την ένθεση τους και συνδυαζόμενα συνθέτουν την γραφική διεπαφή. Τα components είναι συναρτήσεις JavaScript που επιστρέφουν markup. Χρησιμοποιούν JSX (JavaScript Syntax Extension) [17] για τον συνδυασμό λογικής με markup που τους προσδίδει διαδραστικότητα και διευκολύνει στην ανάπτυξη και τη συντήρηση τους.

## 2.8 Node.js

Το Node.js [18] είναι ένα ασύγχρονο καθοδηγούμενο από συμβάντα περιβάλλον εκτέλεσης JavaScript που χρησιμοποιείται κυρίως για τη δημιουργία κλιμακώσιμων εφαρμογών δικτύου. Είναι λογισμικό ανοιχτού κώδικα, ανεξάρτητο πλατφόρμας και βασίζεται στην μηχανή V8 (JavaScript engine) για αυτό και έχει υψηλή απόδοση. Η εκτέλεση κώδικα γίνεται με την χρήση μιας μόνο διεργασίας και μπορεί να εξυπηρετήσει χιλιάδες συνδέσεις συγχρόνως. Αυτό επιτυγχάνεται με ασύγχρονο προγραμματισμό και την χρήση τεχνικής βρόχου εκτέλεσης, όπου εάν κάποιο αίτημα πρέπει να εκτελέσει μια αργή λειτουργία όπως πρόσβαση στην βάση ή διάβασμα από το σύστημα αρχείων τότε το αίτημα αυτό παύει να εκτελείται μέχρι να ολοκληρωθεί η λειτουργία του αλλά στην θέση του εκτελείται κάποιο άλλο αίτημα έτσι ώστε να γίνεται καλύτερη αξιοποίηση του υλικού. Μόλις ολοκληρωθεί η απαιτούμενη λειτουργία τότε το αίτημα συνεχίζει την εκτέλεσή του από εκεί που είχε μείνει. Η εκτέλεση των λειτουργιών όπως πρόσβαση στην βάση δεδομένων ή στο σύστημα αρχείων γίνεται στο παρασκήνιο με χρήση πολλών νημάτων αλλά οργανώνεται από το ίδιο το σύστημα χωρίς να χρειάζεται παρέμβαση του προγραμματιστή διευκολύνοντάς τον και αποφεύγοντας τυχόν λάθη του πολυνηματικού

προγραμματισμού. Τέλος το σύστημα αυτό περιέχει πλήθος βιβλιοθηκών που απαιτούνται για την ανάπτυξη εφαρμογών δικτύου.

## 2.9 Express.js

Το Express.js [19] είναι χτισμένο πάνω στο Node.js και μας διευκολύνει στην ανάπτυξη εφαρμογών δικτύου και APIs. Παρέχει μηχανισμούς για να:

- γράφεις κώδικα χειρισμού http αιτημάτων ανάλογα με την μέθοδο που χρησιμοποιήθηκε και την διαδρομή url.
- ενσωματώνεις μηχανές όψεων και να δημιουργείς απαντήσεις για τον χρήστη με την εισαγωγή δεδομένων σε πρότυπες όψεις.
- θέτεις σημαντικές μεταβλητές εφαρμογών δικτύου όπως τη θύρα σύνδεσης ή την τοποθεσία των προτύπων όψεων.
- δημιουργείς ενδιάμεσες συναρτήσεις επεξεργασίας των αιτημάτων (middleware).

Επίσης το Express.js παρέχει πλήθος βιβλιοθηκών που καλύπτουν ποικίλες ανάγκες των εφαρμογών δικτύου που αφορούν cookies, sessions, ταυτοποίηση χρηστών, παράμετροι των url, κεφαλίδες http κ.α.

## 2.10 MySql

Η MySql είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Τα δεδομένα αποθηκεύονται σε πίνακες ανάλογα με το είδος τους και σχετίζονται μεταξύ τους μέσω συσχετίσεων. Η γλώσσα επερωτημάτων που χρησιμοποιείται είναι η SQL (Structured Query Language). Οι λειτουργίες που επιτελεί το σύστημα αυτό αφορούν την δημιουργία της βάσης δεδομένων, την ανανέωση και ανάκτηση των δεδομένων καθώς και άλλες λειτουργίες που αφορούν την διαχείριση της.

# 3 Περιγραφή συστήματος

## 3.1 Πρόβλημα

Τα πιστοποιητικά χρησιμοποιούνται εδώ και πολλά χρόνια για την απόδειξη γνώσεων και ικανοτήτων των ατόμων. Σήμερα αποτελούν βασικό στοιχείο στην αγορά εργασίας καθώς απαιτούνται άτομα με υψηλό επίπεδο μόρφωσης και ιδιαίτερα καταρτισμένα σε νέες τεχνολογίες. Ωστόσο το πλαίσιο έκδοσης, αποστολής, φύλαξης και επαλήθευσης των πιστοποιητικών ακόμα και σήμερα σε πολλές περιπτώσεις είναι παρωχημένο.

Για την έκδοση ενός ακαδημαϊκού πτυχίου αρχικά θα πρέπει ο ενδιαφερόμενος να υποβάλλει αίτηση στην γραμματεία της σχολής του, στην συνέχεια ελέγχεται ότι ικανοποιούνται κάποιες προϋποθέσεις, επιβεβαιώνονται τα στοιχεία του πτυχίου και τελικά συμπληρώνεται με κάποια σφραγίδα του ιδρύματος ή υπογραφή κάποιου προϊσταμένου όπως για παράδειγμα του κοσμήτορα, ως απόδειξη γνησιότητας. Έπειτα το πτυχίο παραλαμβάνεται από την γραμματεία από τον ίδιο τον φοιτητή ή του αποστέλλεται μέσω ταχυδρομείου. Η παραπάνω διαδικασία είναι χρονοβόρα και για την επανέκδοση του πτυχίου, σε περίπτωση σφάλματος, φθοράς, για την μετάφραση του σε άλλη γλώσσα ή για άλλους λόγους θα πρέπει να επαναληφθεί.

Επίσης ένα άλλο σοβαρό ζήτημα που προκύπτει είναι αυτό της πλαστογραφίας. Βαθμοί φοιτητών μπορεί να αλλοιωθούν με αδιαφανείς τρόπους ή ακόμα και ολόκληρα πτυχία μπορεί να πλαστογραφηθούν. Έτσι η διάκριση μεταξύ γνήσιων και πλαστών καθίσταται μία δύσκολη διαδικασία και πολλές φορές απαιτείται ο οργανισμός που επιθυμεί να πραγματοποιήσει αυτή την ενέργεια, να έρθει σε επαφή με το ακαδημαϊκό ίδρυμα.

Είναι εύκολο να αντιληφθεί κάποιος πως οι παραπάνω διαδικασίες είναι μη αποδοτικές και επιδέχονται βελτίωση και εκσυγχρονισμό. Ένα σύστημα ψηφιακών πιστοποιητικών με χρήση της τεχνολογίας blockchain και των εγγενών ιδιοτήτων που μας παρέχει, μπορεί να αναβαθμίσει τον τρόπο με τον οποίο αυτές τελούνται. Επίσης μπορεί να συνεισφέρει στην δημιουργία μιας παγκόσμιας οντότητας πιστοποίησης με αμεσότητα, χωρίς την ανάγκη ενδιάμεσων φορέων εμπιστοσύνης.

<u>Πλεονεκτήματα ψηφιακών πιστοποιητικών με χρήση blockchain:</u>

- Επαλήθευση πιστοποιητικών και προστασία ενάντια στην πλαστογραφία με εύκολο και άμεσο τρόπο
- Διαφάνεια
- Αυτοματοποίηση διαδικασιών
- Εύκολη πρόσβαση και κοινοποίηση των πιστοποιητικών
- Προστασία των δεδομένων με χρήση κρυπτογραφικών μεθόδων

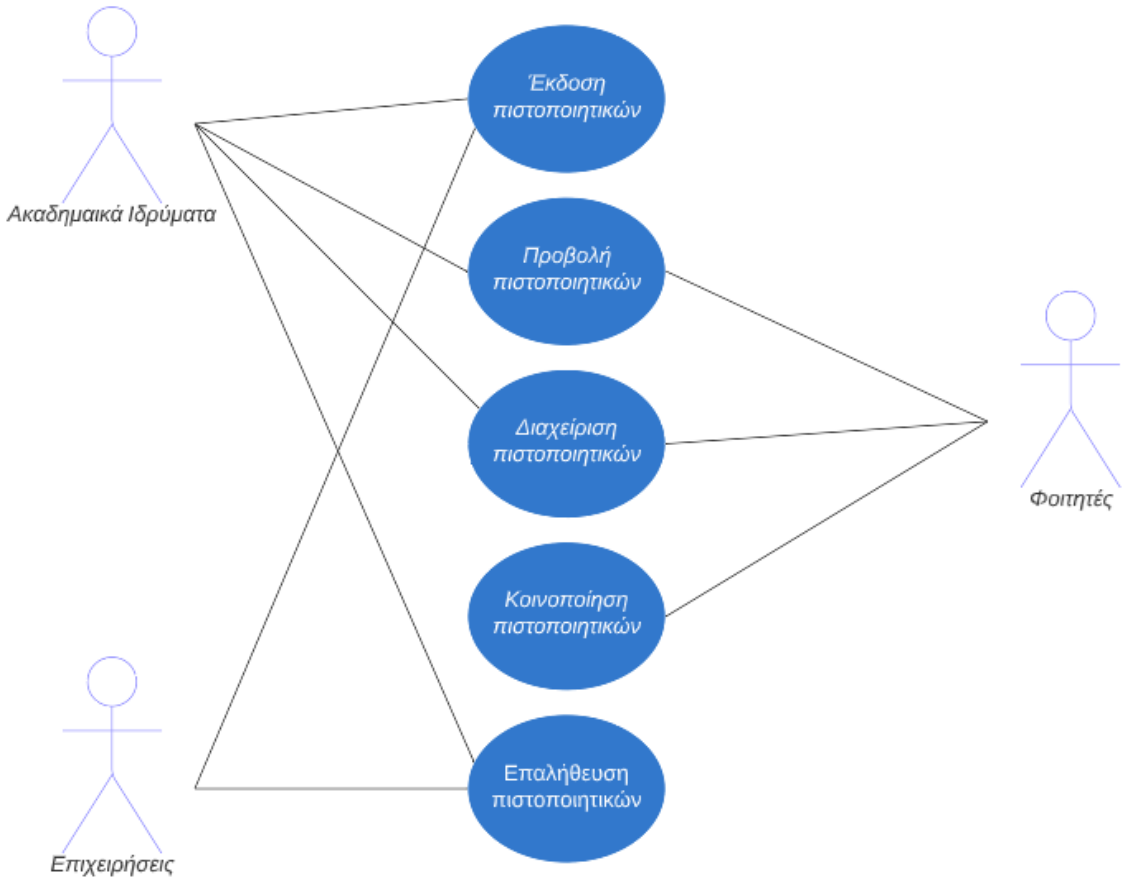
## 3.2 Προτεινόμενη λύση

Για την αντιμετώπιση των παραπάνω προβλημάτων προτείνεται ως λύση και υλοποιείται μια δικτυακή εφαρμογή ψηφιακών πιστοποιητικών με χρήση του Ethereum. Το σύστημα αυτό επιτρέπει την έκδοση, την αποθήκευση, την διαχείριση και την επαλήθευση των ψηφιακών πιστοποιητικών από τους χρήστες του.

Αρχικά μελετήθηκαν οι ομάδες χρηστών του συστήματος, οι περιπτώσεις χρήσης του από την κάθε ομάδα, οι ανάγκες και απαιτήσεις τους. Στην συνέχεια με βάση το προηγούμενο βήμα σχεδιάστηκαν οι βασικές λειτουργίες που πρέπει να υλοποιηθούν καθώς και οι μη λειτουργικές απαιτήσεις του συστήματος. Μελετήθηκαν τα βασικά δεδομένα του συστήματος δηλαδή τα ψηφιακά πιστοποιητικά και οι λογαριασμοί των χρηστών και έγινε προσπάθεια μοντελοποίησης τους. Τέλος ακολούθησε η υλοποίηση του εν λόγω συστήματος που περιγράφεται στο επόμενο κεφάλαιο.

## 3.3 Ομάδες χρηστών και περιπτώσεις χρήσης

Παρακάτω παρουσιάζονται οι βασικές κατηγορίες χρηστών που θεωρήθηκαν κατά τον αρχικό σχεδιασμό του συστήματος καθώς και οι περιπτώσεις για τις οποίες το χρησιμοποιούν.




**Εικόνα 5: Περιπτώσεις χρήσης συστήματος**

### 3.3.1 Ακαδημαϊκά Ιδρύματα

Τα ακαδημαϊκά ιδρύματα χρησιμοποιούν το σύστημα για να:

- Εκδίδουν ψηφιακά πιστοποιητικά στους φοιτητές τους
- Βλέπουν όλα τα πιστοποιητικά που έχουν εκδώσει στους φοιτητές
- Διαχειρίζονται τα πιστοποιητικά που έχουν εκδώσει
- Επαληθεύουν τη γνησιότητα ψηφιακών πιστοποιητικών που τους υποβάλλονται από φοιτητές που έρχονται από άλλα ιδρύματα, για μεταπτυχιακές σπουδές, πρόγραμμα Erasmus κ.α.

### 3.3.2 Φοιτητές

Οι φοιτητές χρησιμοποιούν το σύστημα για να:

- Βλέπουν τα πιστοποιητικά που τους έχουν εκδοθεί από διάφορους οργανισμούς
- Διαχειρίζονται τα πιστοποιητικά τους, για παράδειγμα αίτηση ανανέωσης σε περίπτωση λανθασμένων στοιχείων πιστοποιητικού
- Να κοινοποιούν τα πιστοποιητικά τους σε άλλα ακαδημαϊκά ιδρύματα ή σε επιχειρήσεις σε περίπτωση αναζήτησης εργασίας ή ακόμα και σε μέσα κοινωνικής δικτύωσης για την αναγνώριση των επιτευγμάτων τους

### 3.3.3 Επιχειρήσεις

Οι επιχειρήσεις χρησιμοποιούν το σύστημα για να:

- Επαληθεύουν με εύκολο και γρήγορο τρόπο την γνησιότητα των ψηφιακών πιστοποιητικών που τους υποβάλλουν οι υποψήφιοι εργαζόμενοι, χωρίς την ανάγκη επικοινωνίας με το ακαδημαϊκό ίδρυμα
- Εκδίδουν ψηφιακά πιστοποιητικά για την αναγνώριση εργασίας των εργαζομένων πάνω σε κάποιο αντικείμενο για παράδειγμα σε πρακτική εργασία ή σε κάποιο πρόγραμμα εξειδίκευσης

Κατά την μελέτη του συστήματος οι παραπάνω περιπτώσεις χρήσης αναγνωρίστηκαν ως βασικές και χρησιμοποιήθηκαν για την υλοποίηση του. Ωστόσο ο σχεδιασμός δεν περιορίζει το σύστημα σε αυτές τις περιπτώσεις χρήσης. Οποιοσδήποτε χρήστης του συστήματος έχει την δυνατότητα να είναι ιδιοκτήτης πιστοποιητικών, να εκδίδει νέα πιστοποιητικά σε άλλους χρήστες και να επαληθεύει κάποιο πιστοποιητικό ως προς την γνησιότητά του.

Η επιλογή αυτή έγινε για την χρήση πιστοποιητικών με μία ευρεία έννοια και την δημιουργία ενός συστήματος προσαρμόσιμου στις διαφορετικές ανάγκες των χρηστών του. Για παράδειγμα ένα ακαδημαϊκό ίδρυμα μπορεί να είναι παραλήπτης πιστοποιητικών από άλλους φορείς όπως φορείς αξιολόγησης των εκπαιδευτικών διαδικασιών. Ένας χρήστης που μπορεί να θεωρείται ειδικός σε έναν συγκεκριμένο τομέα, μπορεί να εκδώσει πιστοποιητικό σε έναν άλλο χρήστη ως επιβεβαίωση στις ικανότητές του σε αυτόν τον τομέα.

Επομένως στο σύστημα που υλοποιείται δεν γίνεται διαχωρισμός των χρηστών σε κατηγορίες αλλά όλοι έχουν όλα τα δικαιώματα χρήσης του. Κάθε χρήστης έχει την

δυνατότητα έκδοσης πιστοποιητικών. Είναι λοιπόν ευθύνη του παραλήπτη ενός πιστοποιητικού να το αποδεχτεί μόνο εφόσον συμφωνεί με αυτό και εμπιστεύεται τον αποστολέα του.

## 3.4 Δεδομένα

Τα δύο βασικά δεδομένα του συστήματος είναι οι λογαριασμοί των χρηστών και τα ψηφιακά πιστοποιητικά. Στο επόμενο κεφάλαιο θα δούμε το σύνολο των δεδομένων συμπεριλαμβανομένων των νέων που προκύπτουν από τις ανάγκες της υλοποίησης του συστήματος.

### 3.4.1 Λογαριασμοί χρηστών

Οι λογαριασμοί αποτελούν την ταυτότητα των χρηστών του συστήματος. Κατά την ανάλυση του συστήματος προκύπτει η ανάγκη της ταυτοποίησης των χρηστών ώστε να μπορεί να αποδοθεί και να αποδειχθεί η ιδιοκτησία των ψηφιακών πιστοποιητικών. Επίσης οι χρήστες πρέπει να μπορούν να αναγνωρίζονται μεταξύ τους ώστε να αποφασίζουν αν αποδέχονται ή όχι την έκδοση πιστοποιητικού από κάποιον άλλον. Η ταυτότητα αυτή πρέπει να χαρακτηρίζει μοναδικά τον κάθε χρήστη και να τον εξασφαλίζει από κακόβουλες ενέργειες πλαστοπροσωπίας, ώστε να μην προκύπτει υποκλοπή των ευαίσθητων και προσωπικών του δεδομένων. Για αυτό γίνεται χρήση της κρυπτογραφίας δημοσίου κλειδιού που περιγράφηκε στο πρώτο κεφάλαιο, με το δημόσιο κλειδί ως αναγνωριστικό του χρήστη και το ιδιωτικό κλειδί για την παραγωγή ψηφιακών υπογραφών.

Πρέπει να σημειωθεί πως η μέχρι τώρα θεώρηση συνδέει τις ψηφιακές οντότητες με τα πιστοποιητικά τους αλλά όχι με τα φυσικά ή νομικά πρόσωπα τους χρήστες δηλαδή στον πραγματικό κόσμο. Το πρόβλημα αυτό εξετάσθηκε κατά τον σχεδιασμό του συστήματος. Διάφοροι πιθανοί τρόποι αντιμετώπισης διερευνήθηκαν όπως για παράδειγμα η παροχή από τα ακαδημαϊκά ιδρύματα πιστοποιητικών από CA(Certificate Authorities) που αποδεικνύουν την εγκυρότητα της ταυτότητας τους αλλά τελικά το πρόβλημα αυτό τέθηκε εκτός της εμβέλειας της εργασίας.

### 3.4.2 Ψηφιακά πιστοποιητικά

Τα ψηφιακά πιστοποιητικά αποτελούν το βασικό δεδομένο του συστήματος. Μελετήθηκαν τα πιστοποιητικά γενικότερα ώστε να αναγνωριστούν τα βασικά στοιχεία τα οποία τα θεμελιώνουν και περιέχουν όλη την απαραίτητη πληροφορία.

Τα στοιχεία ενός πιστοποιητικού μπορούν να αναλυθούν στα παρακάτω:

- Εκδότης του πιστοποιητικού
- Παραλήπτης του πιστοποιητικού
- Ισχυρισμοί για κάποια δεξιότητα, ικανότητα, εμπειρία του παραλήπτη.
- Ημερομηνία έκδοσης πιστοποιητικού. Σημαντική πληροφορία για πιστοποιητικά με περιορισμένη διάρκεια.
- Στοιχεία που επαληθεύουν τις ταυτότητες του παραλήπτη και του εκδότη όπως σφραγίδες, υπογραφές κ.α.

Με βάση το παραπάνω μοντέλο χτίστηκαν τα πιστοποιητικά του συστήματος μας. Ως ταυτότητα του εκδότη και του παραλήπτη χρησιμοποιούμε τα δημόσια κλειδιά τους. Ως πληροφορία του πιστοποιητικού χρησιμοποιούμε κάποιο έγγραφο που θα περιγράφει όλους

τους ισχυρισμούς για τις ικανότητες του παραλήπτη. Ακόμη θα περιέχει τα στοιχεία των φυσικών οντοτήτων όπως ονοματεπώνυμο φοιτητή, όνομα ιδρύματος αλλά και τα στοιχεία των ψηφιακών οντοτήτων δηλαδή τα δημόσια κλειδιά. Τέλος τα πιστοποιητικά θα υπογράφονται στο σύστημα από τον παραλήπτη και τον εκδότη με τις ψηφιακές τους υπογραφές.

## 3.5 Λειτουργικές απαιτήσεις

Ας προχωρήσουμε τώρα στην ανάλυση των λειτουργικών απαιτήσεων των βασικών λειτουργιών δηλαδή, τις οποίες θα υλοποιεί το σύστημα και οι οποίες προκύπτουν από τις ανάγκες που μελετήθηκαν παραπάνω.

### 3.5.1 Ταυτοποίηση χρηστών

Το σύστημα πρέπει να ταυτοποιεί τους χρήστες που εισέρχονται σε αυτό.

### 3.5.2 Περιορισμός πρόσβασης χρηστών

Το σύστημα πρέπει να περιορίζει την πρόσβαση στους χρήστες του, ώστε να μην μπορούν να εκτελέσουν ενέργειες που δεν τους επιτρέπονται ή να ανακτήσουν δεδομένα που δεν τους ανήκουν.

### 3.5.3 Παρουσίαση πιστοποιητικών

Το σύστημα πρέπει να μπορεί να αποθηκεύει, να ανακτά και να παρουσιάζει οπτικά τα πιστοποιητικά (ιδιόκτητα και εκδοθέντα) των χρηστών.

### 3.5.4 Έκδοση νέων πιστοποιητικών

Το σύστημα πρέπει να δίνει την δυνατότητα στους χρήστες του να εκδίδουν νέα πιστοποιητικά προς άλλους χρήστες.

### 3.5.5 Αποδοχή ή απόρριψη πιστοποιητικού

Το σύστημα πρέπει να δίνει στους χρήστες την δυνατότητα να αποδεχθούν εάν συμφωνούν με αυτό ή αλλιώς να απορρίψουν, ένα πιστοποιητικό που έχει εκδοθεί για αυτούς.

### 3.5.6 Επαλήθευση πιστοποιητικού

Το σύστημα πρέπει να παρέχει στους χρήστες του την δυνατότητα επαλήθευσης ενός πιστοποιητικού. Πιο συγκεκριμένα πρέπει να μπορεί να επαληθεύει τα δεδομένα του πιστοποιητικού και να επιβεβαιώνει ότι δεν έχουν αλλοιωθεί. Ακόμη πρέπει να μπορεί να επαληθεύει την ταυτότητα του εκδότη και του παραλήπτη ελέγχοντας τις ψηφιακές τους υπογραφές.

### 3.5.7 Ενημέρωση πιστοποιητικού

Το σύστημα πρέπει να παρέχει την δυνατότητα ανανέωσης ενός πιστοποιητικού σε περίπτωση που κάποιο στοιχείο του είναι λανθασμένο. Σε περίπτωση ανανέωσης στοιχείων

πιστοποιητικού θα πρέπει να συμφωνήσουν και τα δύο μέλη για τα νέα στοιχεία δηλαδή και ο εκδότης και ο παραλήπτης.

### 3.5.8 Διαγραφή πιστοποιητικού

Το σύστημα πρέπει να παρέχει την δυνατότητα διαγραφής ενός πιστοποιητικού. Υπάρχουν διάφοροι λόγοι που απαιτούν αυτή την λειτουργία. Ίσως ο πιο σημαντικός λόγος είναι ότι πολλά πιστοποιητικά συνδέονται με κάποια χρονική διάρκεια ισχύος. Μετά το πέρας αυτής της διάρκειας τα πιστοποιητικά αυτά πρέπει να ακυρώνονται. Παράδειγμα τέτοιου πιστοποιητικού είναι η πιστοποίηση κάποιας ικανότητας η οποία εξασθενεί με τον χρόνο και μπορεί να απαιτεί τακτική αξιολόγηση για την ανανέωση της. Στην περίπτωση διαγραφής πιστοποιητικού ο εκδότης μπορεί να το διαγράφει άμεσα χωρίς την συγκατάθεση του παραλήπτη όταν υπάρχει κάποιος σημαντικός λόγος αλλά ο παραλήπτης πρέπει να ενημερώνεται για αυτή την ενέργεια. Ο ιδιοκτήτης ενός πιστοποιητικού μπορεί να το διαγράψει οποιαδήποτε στιγμή άμεσα, χωρίς την συγκατάθεση του εκδότη του καθώς είναι προσωπική του ιδιοκτησία.

## 3.6 Μη λειτουργικές απαιτήσεις

Οι μη λειτουργικές απαιτήσεις αφορούν κυρίως ποιοτικές απαιτήσεις του συστήματος με βάση τις παραπάνω λειτουργίες και αποτελούν κριτήρια αξιολόγησης του.

### 3.6.1 Ιδιωτικότητα των δεδομένων

Τα σύστημα περιέχει ευαίσθητα δεδομένα των χρηστών όπως προσωπικά στοιχεία, πιστοποιητικά που μπορεί να περιέχουν βαθμολογίες, αξιολογήσεις κ.α. και επομένως αυτά πρέπει να προφυλάσσονται. Στα δεδομένα κάθε χρήστη έχει πρόσβαση μόνο ο ίδιος καθώς και όσοι αυτός επιλέγει να τα δημοσιοποιήσει.

### 3.6.2 Ακεραιότητα των δεδομένων

Τα πιστοποιητικά αποτελούν ισχυρισμούς για κάποια άτομα που χρήζουν ζωτικής σημασίας για την επαγγελματική αποκατάσταση τους και για την απόδειξη των ικανοτήτων τους γενικότερα, επομένως αυτά τα δεδομένα πρέπει να προστατεύονται και να διατηρούνται αναλλοίωτα. Ο ίδιος ο χρήστης μπορεί να είχε όφελος από παραποίηση των δεδομένων του και επομένως τέτοιες ενέργειες πρέπει να αποτρέπονται.

### 3.6.3 Εύκολη και γρήγορη επαλήθευση πιστοποιητικών

Πρέπει το σύστημα να παρέχει μια απλή, άμεση και γρήγορη διαδικασία επαλήθευσης των πιστοποιητικών χωρίς ανάγκη επικοινωνίας με έμπιστο τρίτο φορέα.

### 3.6.4 Διαφάνεια

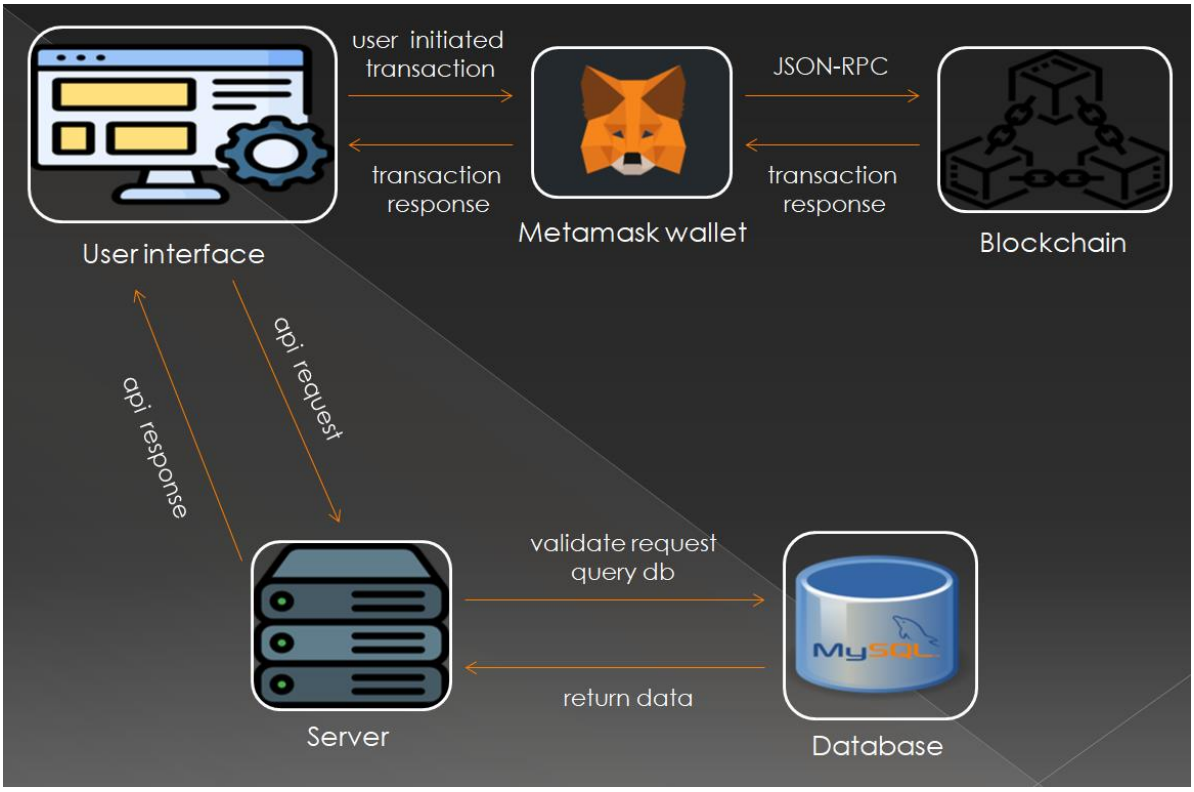
Το σύστημα πρέπει να παρέχει διαφάνεια σε όλες τις συναλλαγές που επιτελούνται ώστε οποιαδήποτε κακόβουλη ενέργεια να μπορεί να ιχνηλατηθεί και να καταπολεμώνται έτσι οι παραβάτες της.

# 4 Σχεδιασμός και υλοποίηση

Στο κεφάλαιο αυτό εξετάζουμε τον σχεδιασμό και την τεχνική υλοποίηση της εφαρμογής.

## 4.1 Αρχιτεκτονική

Αρχικά θα αναφερθούμε στην αρχιτεκτονική του συστήματος και θα δούμε τα υποσυστήματα από τα οποία αποτελείται και πως αυτά συνδέονται για να αποκτήσουμε μια συνολική εικόνα.




**Εικόνα 6: Διάγραμμα αρχιτεκτονικής**

Στην παραπάνω εικόνα βλέπουμε τα υποσυστήματα από τα οποία αποτελείται το σύστημα μας και πως αυτά αλληλεπιδρούν.

Αρχικά έχουμε την διεπαφή χρήστη, την αφετηρία οποιασδήποτε λειτουργίας. Ο χρήστης εισέρχεται στην εφαρμογή μας και βλέπει τα δεδομένα του στην γραφική διεπαφή. Στην συνέχεια μπορεί μέσα από αυτή να εκτελέσει κάποια ενέργεια όπως την έκδοση κάποιου πιστοποιητικού.

Εάν αυτή η ενέργεια δημιουργεί συναλλαγή στο blockchain, τότε αναδύεται το παράθυρο του MetaMask το οποίο προβάλλει στον χρήστη την συναλλαγή και ζητάει την επιβεβαίωση του. Αν ο χρήστης την επιβεβαιώσει, το MetaMask εκτελεί αίτημα JSON-RPC [20] στο blockchain με το οποίο είναι συνδεδεμένο, για την δεδομένη συναλλαγή. Το blockchain εκτελεί την συναλλαγή και επιστρέφει την απάντηση στο MetaMask, το οποίο με την σειρά του την επιστρέφει στην γραφική διεπαφή.

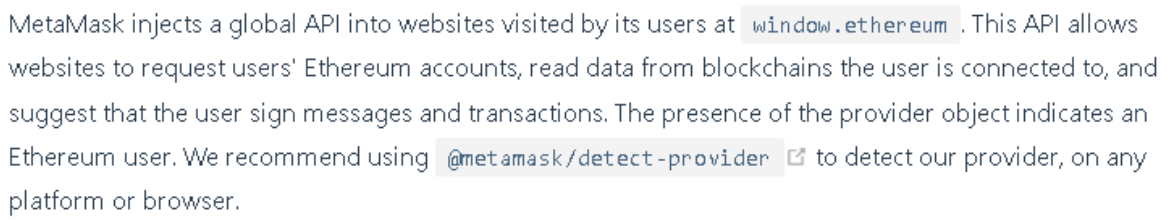
Αν απαιτείται μεταφορά δεδομένων στην γραφική διεπαφή, στέλνεται αίτημα στον server ο οποίος το ελέγχει και αν το εγκρίνει επικοινωνεί με την βάση για να ανακτήσει τα δεδομένα. Η βάση στην συνέχεια απαντάει με τα δεδομένα και ο server τα στέλνει πίσω στην γραφική διεπαφή.

## 4.2 Γραφική διεπαφή χρήστη

Η γραφική διεπαφή αποτελεί την αφετηρία του συστήματος. Ο χρήστης πλοηγείται στις διάφορες σελίδες και μπορεί να εκτελεί λειτουργίες και να βλέπει τα δεδομένα του. Μέρος της εργασίας αφορούσε τον σχεδιασμό της γραφικής διεπαφής με βάση τις λειτουργίες που πρέπει να εκτελεί, την εμπειρία χρήστη αλλά και την εμφάνιση. Για την ανάπτυξή της χρησιμοποιήθηκε η βιβλιοθήκη της JavaScript, React.js και για την εμφάνιση css.

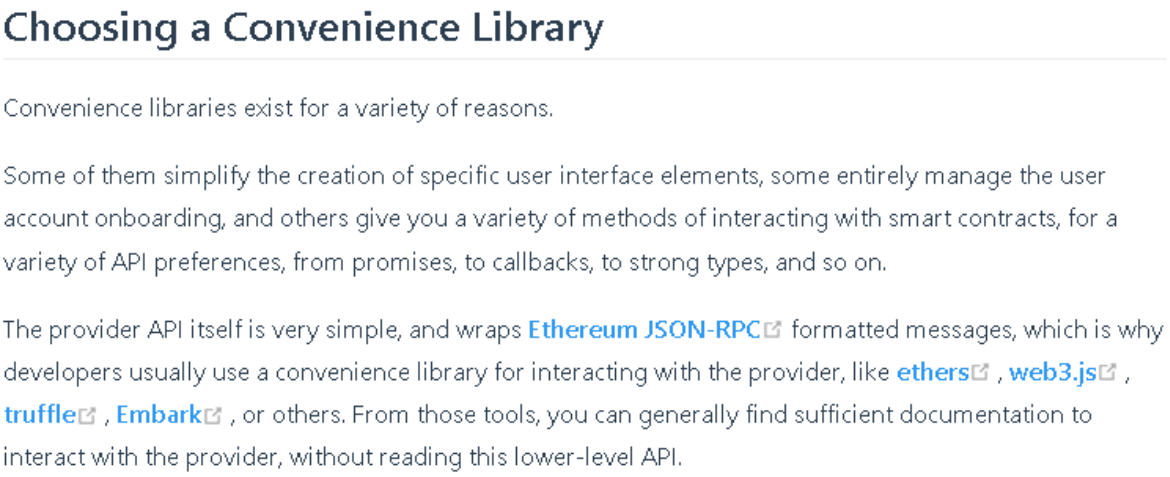
### 4.2.1 Σύνδεση με MetaMask

Για την σύνδεση με το MetaMask γίνεται χρήση του API το οποίο παρέχει στο παράθυρο του browser, στην μεταβλητή window.ethereum.




**Εικόνα 7: Απόσπασμα οδηγού MetaMask**

Συναλλαγές μπορούν να εκτελεστούν απευθείας μέσω της μεταβλητής window.ethereum. Ακόμη μπορεί να γίνει χρήση κάποιας βιβλιοθήκης, για την διευκόλυνση μας στην διενέργεια των συναλλαγών και την αλληλεπίδραση με συμβόλαια.




**Εικόνα 8: Απόσπασμα οδηγού MetaMask**

Στο παρακάτω απόσπασμα από τον κώδικα βλέπουμε πως μπορούμε να εκτελέσουμε μία ενέργεια απευθείας από την μεταβλητή window.ethereum, με κλήση της συνάρτησης

request. Αυτό που κάνει η συνάρτηση signMessage είναι να παίρνει την διεύθυνση του χρήστη και ένα μήνυμα και να εκτελεί αίτημα στο MetaMask για υπογραφή του μηνύματος με το ιδιωτικό κλειδί αυτής της διεύθυνσης. Στην συνέχεια το παράθυρο του MetaMask αναδύεται και ο χρήστης πρέπει να επιλέξει αν επιβεβαιώνει την υπογραφή του μηνύματος. Σε περίπτωση επιβεβαίωσης η συνάρτηση επιστρέφει το υπογεγραμμένο μήνυμα.

```
function signMessage(publicAddress, msg) {
    let hex_msg = `0x${Buffer.from(msg, 'utf8').toString('hex')}`;
     return (
         window.ethereum.request({
                        method : 'personal_sign',
                        params: [hex_msg, publicAddress]
                    })
     )
}
```

**Εικόνα 9: Συνάρτηση υπογραφής μηνύματος**

Εκτός από τον παραπάνω τρόπο, για την αλληλεπίδραση με τα έξυπνα συμβόλαια της εφαρμογής μας χρησιμοποιούμε την βιβλιοθήκη web3.js.

### 4.2.2 Σύνδεση με API

Η σύνδεση της γραφικής διεπαφής με τον server γίνεται με την αποστολή http μηνυμάτων με χρήση του Fetch API [21] της JavaScript. Στο παρακάτω απόσπασμα από τον κώδικα βλέπουμε την αποστολή ενός http post αιτήματος στον server.

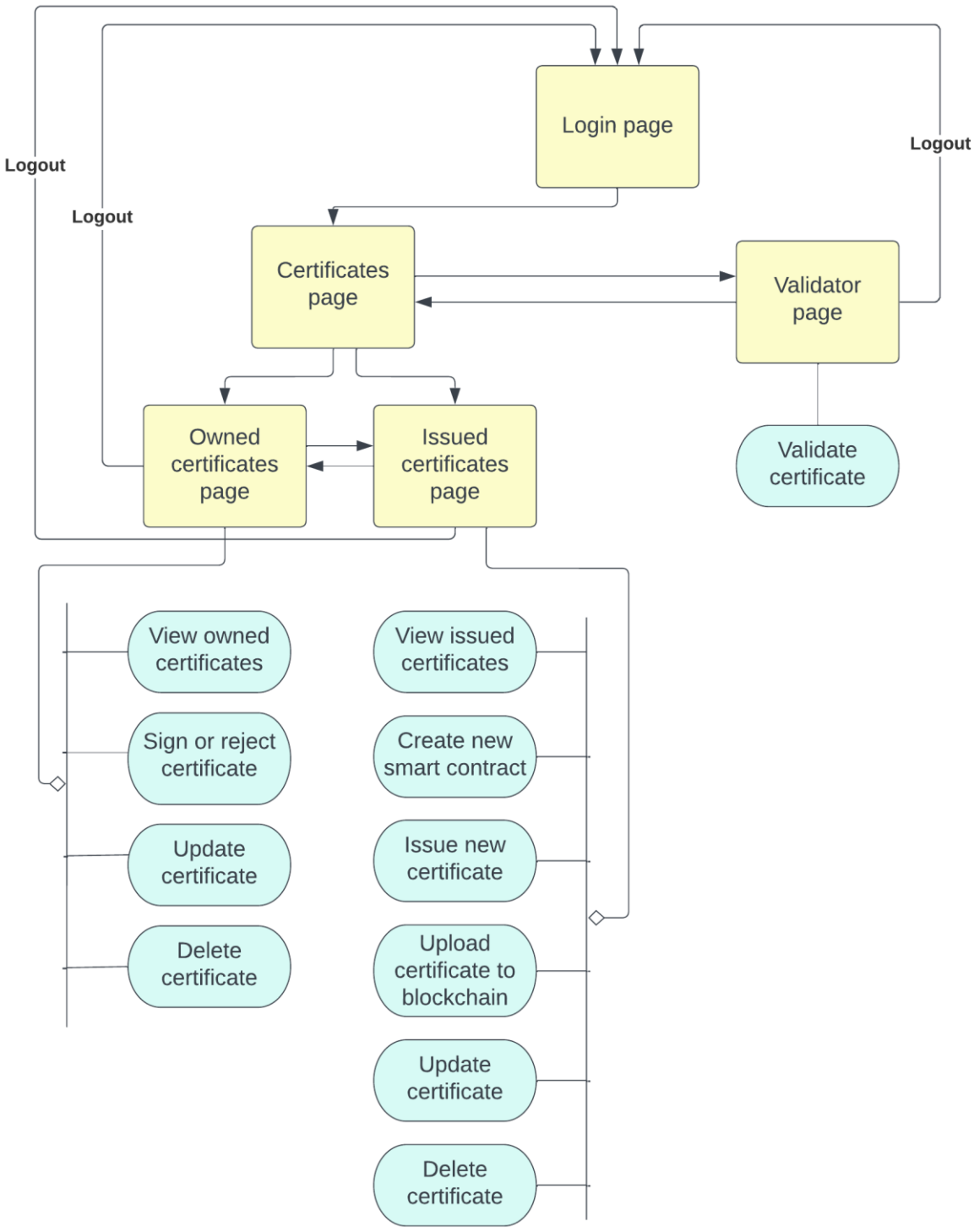
Αυτό που κάνει η συνάρτηση είναι να στέλνει στον server την υπογραφή του χρήστη και το μήνυμα το οποίο έχει υπογράψει. Ο server θα επαληθεύσει αυτή την υπογραφή και στην συνέχεια θα απαντήσει στέλνοντας πίσω το token του χρήστη το οποίο θα χρησιμοποιεί για τα επόμενα αιτήματά του.

```
async function fetchToken(publicAddress, signature, sign_msg){
    let resp = await fetch('http://localhost:3100/'+publicAddress+'/signature', {
        method : 'POST',
        headers: { 'Content-Type': 'application/json' },
        body : JSON.stringify({
            signature : signature,
            sign_msg : sign_msg
        })
    });
    if (!resp.ok) throw new Error("FETCHING JWT FAILED");
    let json = await resp.json();
    return json.jwt;
}
```

**Εικόνα 10: Συνάρτηση λήψης token**

### 4.2.3 Διάγραμμα πλοήγησης

Παρακάτω φαίνεται το διάγραμμα πλοήγησης της γραφικής διεπαφής που αποτελεί τον σκελετό της. Με μπεζ χρώμα φαίνονται οι διάφορες σελίδες στις οποίες ο χρήστης μπορεί να πλοηγηθεί και με γαλάζιο οι ενέργειες που μπορεί να εκτελέσει σε κάθε σελίδα.




**Εικόνα 11: Διάγραμμα πλοήγησης γραφικής διεπαφής**

Αρχικά ο χρήστης βρίσκεται στην σελίδα εισόδου. Μετά την ταυτοποίησή η εφαρμογή αρχικοποιείται με τα δεδομένα του και οδηγείται στην σελίδα των πιστοποιητικών. Η σελίδα των πιστοποιητικών χωρίζεται στην σελίδα των ιδιόκτητων πιστοποιητικών και στην σελίδα των εκδοθέντων πιστοποιητικών. Από την πρώτη ο χρήστης μπορεί να δει τα πιστοποιητικά που του ανήκουν, να αποδεχτεί και να υπογράψει ένα νέο πιστοποιητικό που έχει εκδοθεί για αυτόν, να ανανεώσει ή να διαγράψει κάποιο από τα πιστοποιητικά του. Από την δεύτερη ο χρήστης μπορεί να δει τα πιστοποιητικά που έχει εκδώσει ο ίδιος, να δημιουργήσει ένα νέο συμβόλαιο στο οποίο θα αποθηκεύει τα πιστοποιητικά που εκδίδει, να εκδώσει ένα νέο πιστοποιητικό, να αποθηκεύσει ένα υπογεγραμμένο από τον παραλήπτη πιστοποιητικό στο blockchain, να ανανεώσει ή να διαγράψει ένα εκδοθέν πιστοποιητικό. Τέλος, ο χρήστης μπορεί να πλοηγηθεί στην σελίδα επαλήθευσης πιστοποιητικών για να ελέγξει κάποιο πιστοποιητικό.

## 4.3 MetaMask

Για την είσοδο στο σύστημα μας η μόνη απαίτηση από τον χρήστη είναι να έχει εγκατεστημένο το MetaMask και να διαθέτει λογαριασμό. Το MetaMask διαχειρίζεται τα κλειδιά του χρήστη και παρέχει γραφική διεπαφή για την εποπτεία και επιβεβαίωση των συναλλαγών που εκτελούνται στο blockchain. Μέρος της εργασίας αφορούσε την μελέτη του εγχειριδίου του MetaMask για την κατανόηση του API που παρέχει για την τέλεση συναλλαγών και του τρόπου ενσωμάτωσής του στην εφαρμογή.

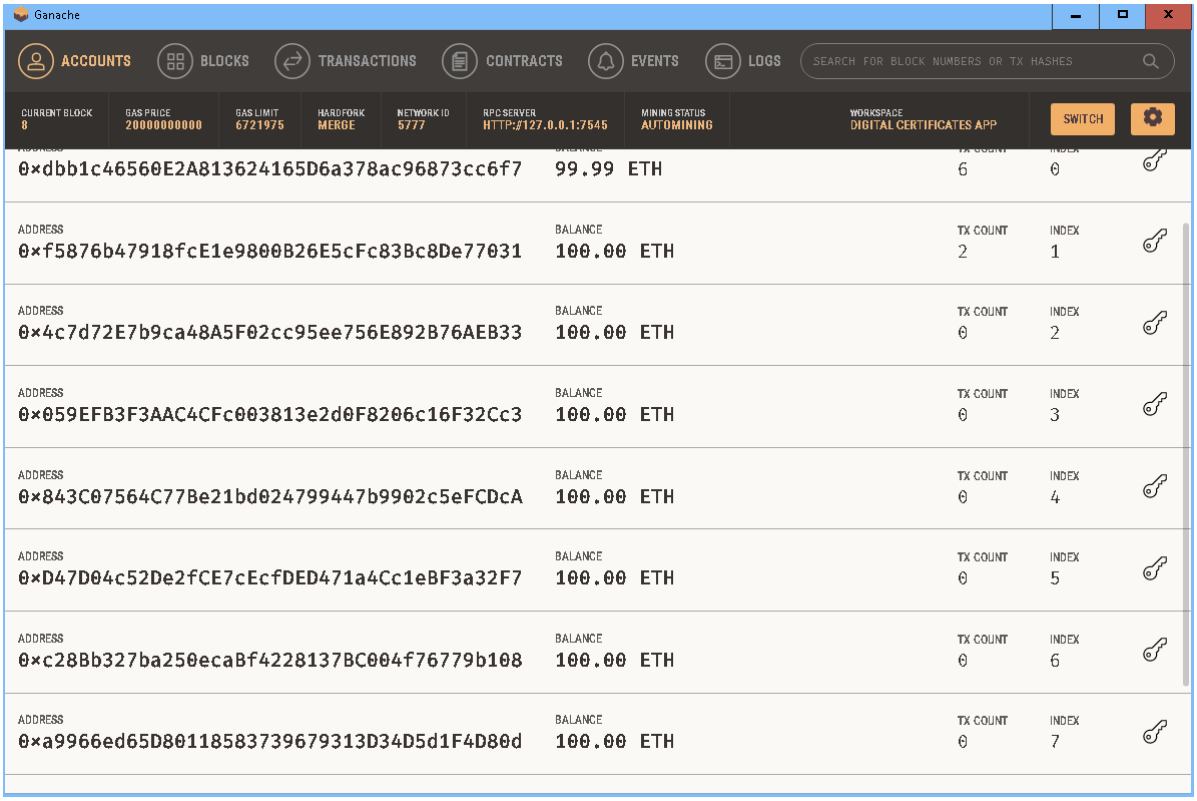
### 4.3.1 Σύνδεση με blockchain

Το MetaMask επικοινωνεί με κόμβους του Ethereum μέσω του JSON-RPC API [20] που αυτοί είναι υποχρεωμένοι να υλοποιούν. Έρχεται με προεγκατεστημένες συνδέσεις με κόμβους του Ethereum αλλά στην περίπτωση μας το συνδέουμε με το τοπικό blockchain που τρέχουμε με την εφαρμογή Ganache.


**Εικόνα 12: Απόσπασμα οδηγού MetaMask**

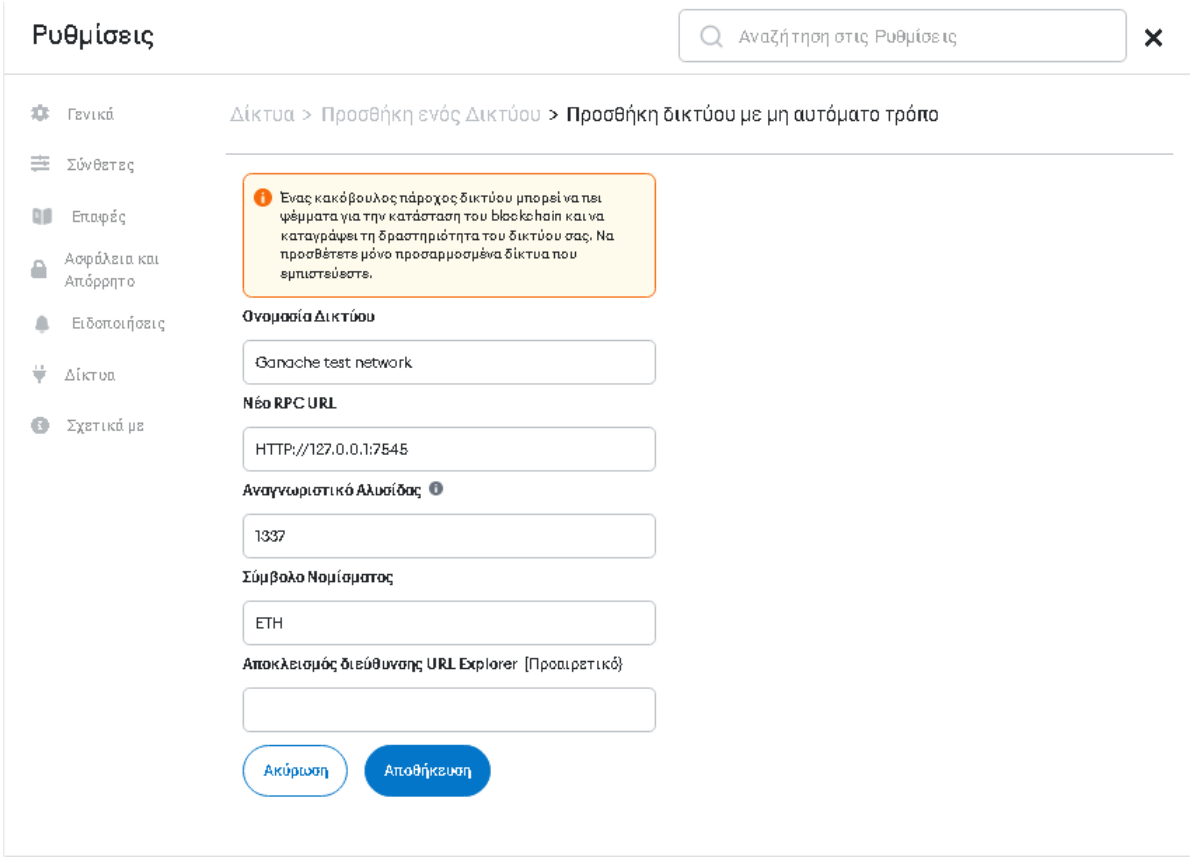
### 4.3.2 Σύνδεση με Ganache

Αρχικά τρέχουμε την εφαρμογή Ganache για να στήσουμε το τοπικό blockchain. Η εφαρμογή αυτή δημιουργεί αυτόματα κάποιους λογαριασμούς και τους αρχικοποιεί με 100 ETH.




**Εικόνα 13: Γραφική διεπαφή Ganache**

Στην συνέχεια στις ρυθμίσεις του MetaMask επιλέγουμε Δίκτυα > Προσθήκη ενός Δικτύου > Προσθήκη δικτύου με μη αυτόματο τρόπο. Έπειτα εισάγουμε τα στοιχεία του τοπικού δικτύου μας και το αποθηκεύουμε.




**Εικόνα 14: Ρυθμίσεις MetaMask**

Τώρα το MetaMask είναι συνδεδεμένο με το τοπικό blockchain στον υπολογιστή μας και μπορούμε να εκτελούμε συναλλαγές σε αυτό.

## 4.4 Back-end

Ο server συνδέει την γραφική διεπαφή του χρήστη με την βάση δεδομένων. Δέχεται αιτήματα τα επεξεργάζεται, τα επαληθεύει, συνδέεται με την βάση και επιστρέφει πίσω τα κατάλληλα δεδομένα. Στις αρμοδιότητές του εντάσσεται και η ταυτοποίηση του χρήστη και ο περιορισμός της πρόσβασής του στα δεδομένα.

Για την δημιουργία του ακολουθήθηκε η αρχιτεκτονική του Restful API [22] και για τα δεδομένα χρησιμοποιήθηκε η αναπαράσταση σε μορφή JSON. Το API απαρτίζεται από endpoints τα οποία καλούνται με αιτήματα http και με κάποια συγκεκριμένη μέθοδο και καθένα από αυτά εκτελεί μια συγκεκριμένη λειτουργία.

Για την ανάπτυξη του χρησιμοποιήθηκε το περιβάλλον Node.js σε συνδυασμό με το Express.js.

### 4.4.1 Σύνδεση με βάση

Για την σύνδεση του server με την βάση χρησιμοποιείται η βιβλιοθήκη mysql [23].

Παρακάτω φαίνεται απόσπασμα χρήσης της βιβλιοθήκης για την σύνδεση με την βάση και την εκτέλεση επερωτημάτων από τον επίσημο οδηγό.

```
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'me',
  password : 'secret',
  database : 'my_db'
});

connection.connect();

connection.query('SELECT 1 + 1 AS solution', function (error, results, fields) {
  if (error) throw error;
  console.log('The solution is: ', results[0].solution);
});

connection.end();
```

**Εικόνα 15: Απόσπασμα οδηγού mysql**

Με την συνάρτηση createConnection δημιουργούμε ένα αντικείμενο σύνδεσης ορίζοντας τις παραμέτρους της βάσης μας. Στην συνέχεια εκτελώντας την συνάρτηση connect στο αντικείμενο δημιουργούμε μια σύνδεση με την βάση και με την query εκτελούμε ένα ερώτημα και λαμβάνουμε το αποτέλεσμα. Τέλος κλείνουμε την σύνδεση με την συνάρτηση end.

### 4.4.2 Endpoints

Στην συνέχεια θα αναλύσουμε την δομή των endpoints του API και θα περιγράψουμε την λειτουργία τους.

Ο server τρέχει τοπικά στον υπολογιστή μας στην θύρα 3100 οπότε οποιοδήποτε αίτημα προς αυτόν θα ξεκινάει με το πρόθεμα http://localhost:3100/ και θα ακολουθεί ο καθορισμός του endpoint που επιθυμούμε να καλέσουμε.

***Endpoint 1***
/{user address}/nonce

***Μέθοδος***
HTTP GET

***Δεδομένα εισόδου***
Πρέπει να δοθεί η παράμετρος του endpoint {user address} που δηλώνει την διεύθυνση του χρήστη στο Ethereum.

***Δεδομένα εξόδου***
Επιστρέφεται ένα μήνυμα που περιέχει τον τυχαίο ακέραιο αριθμό nonce του χρήστη.

Παράδειγμα:
```
{
        sign_msg : ` Message to be signed by user {nonce}`
}
```

**Περιγραφή**
Επιστρέφεται ως απάντηση ένα μήνυμα που περιέχει το nonce που αντιστοιχεί στον χρήστη με την διεύθυνση που ορίζεται από την παράμετρο. Χρησιμοποιείται για την διαδικασία ταυτοποίησης του χρήστη. Το nonce είναι ένας τυχαίος αριθμός του χρήστη που χρησιμοποιείται για λόγους ασφαλείας κατά την διαδικασία ταυτοποίησης.

***Endpoint 2***
/{user address}/signature

***Μέθοδος***
HTTP POST

***Δεδομένα εισόδου***
Πρέπει να δοθεί η παράμετρος του endpoint {user address} που δηλώνει την διεύθυνση του χρήστη στο Ethereum.
Επίσης πρέπει στο σώμα του http αιτήματος να δοθούν δύο μεταβλητές, το μήνυμα που υπέγραψε ο χρήστης και η υπογραφή του σε αυτό το μήνυμα.

Παράδειγμα:
{
sign_msg : `the message that the user signed`,
signature : `user signature of the above message `
}

***Δεδομένα εξόδου***
Επιστρέφεται ένα μήνυμα που περιέχει το token του χρήστη.

Παράδειγμα:
{
jwt : `{json web token}`
}

**Περιγραφή**
Ταυτοποιεί τον χρήστη ελέγχοντας την υπογραφή του και επιστρέφει ένα token το οποίο θα χρησιμοποιεί για τα επόμενα αιτήματά του στο API.

***Endpoint 3***
/user/fetch_userdata

***Μέθοδος***
HTTP POST

***Δεδομένα εισόδου***
Η μοναδική είσοδος που απαιτείται είναι η παροχή του token που είδαμε προηγουμένως στο σώμα του αιτήματος http.

Παράδειγμα:
{
jwt : `{json web token}`
}

***Δεδομένα εξόδου***
Επιστρέφεται ένα μήνυμα που περιέχει τα παρακάτω δεδομένα του χρήστη:
- πιστοποιητικά (ιδιόκτητα και εκδοθέντα)
- αιτήματα που έχουν γίνει από και προς αυτόν
- διευθύνσεις των έξυπνων συμβολαίων του
- προσωπικά δεδομένα όπως ονοματεπώνυμο κ.α.

Παράδειγμα:
{
certificates : [{certificate1}, {certificate2}, …] ,
requests : [{request1}, {request2}, …] ,
contracts : [{contractAddress1}, {contractAddress2}, …] ,
user : {user information object}
}

**Περιγραφή**
Χρησιμοποιείται για την αρχικοποίηση όλων των δεδομένων του χρήστη όπως αναφέρθηκαν παραπάνω κατά την είσοδό του στο σύστημα.

***Endpoint 4***
/user/fetch_requests

***Μέθοδος***
HTTP POST

***Δεδομένα εισόδου***
Η μοναδική είσοδος που απαιτείται είναι η παροχή του token στο σώμα του αιτήματος http.

Παράδειγμα:
{
jwt : `{json web token}`
}

***Δεδομένα εξόδου***
Επιστρέφεται ένα μήνυμα που περιέχει τα νέα αιτήματα που έχει δεχθεί ο χρήστης.

Παράδειγμα:
{
requests : [{new_request1}, {new_request2}, …] ,
}

**Περιγραφή**
Χρησιμοποιείται για την ανανέωση της γραφικής διεπαφής του χρήστη με τα νέα αιτήματα που γίνονται προς αυτόν ενώ είναι ήδη συνδεδεμένος.

***Endpoint 5***
/user/upd_requests

***Μέθοδος***
HTTP POST

***Δεδομένα εισόδου***
Η είσοδος πρέπει να περιέχει στο σώμα του http αιτήματος τα παρακάτω δεδομένα :
- των τύπο του αιτήματος
- το token του χρήστη
- δεδομένα που αφορούν το αίτημα

Παράδειγμα:
{
type : `{request type}`,
jwt : `{json web token}`,
data : {object regarding request data}
}

***Δεδομένα εξόδου***
Επιστρέφεται ένα μήνυμα που περιέχει τα δεδομένα που αφορούν το αποτέλεσμα του χειρισμού του αιτήματος.

Παράδειγμα:
{
data : { data object, resulted from request execution }
}

**Περιγραφή**
Το endpoint αυτό αφορά τα αιτήματα μεταξύ των χρηστών. Για παράδειγμα το αίτημα έκδοσης ενός νέου πιστοποιητικού, την υπογραφή ενός αιτήματος έκδοσης πιστοποιητικού κ.α. Εκτελεί λειτουργίες ανάλογα με τον τύπο του αιτήματος, ανανεώνει το αίτημα μετά την εκτέλεσή των λειτουργιών και επιστρέφει δεδομένα σχετικά με τις αλλαγές στον χρήστη.

***Endpoint 6***
/user/insertContract

***Μέθοδος***
HTTP POST

***Δεδομένα εισόδου***
Η είσοδος πρέπει να περιέχει στο σώμα του http αιτήματος τα παρακάτω δεδομένα :
- την διεύθυνση του έξυπνου συμβολαίου
- το token του χρήστη

Παράδειγμα:
{
contractAddress : `{contract address}`,
jwt : `{json web token}`
}

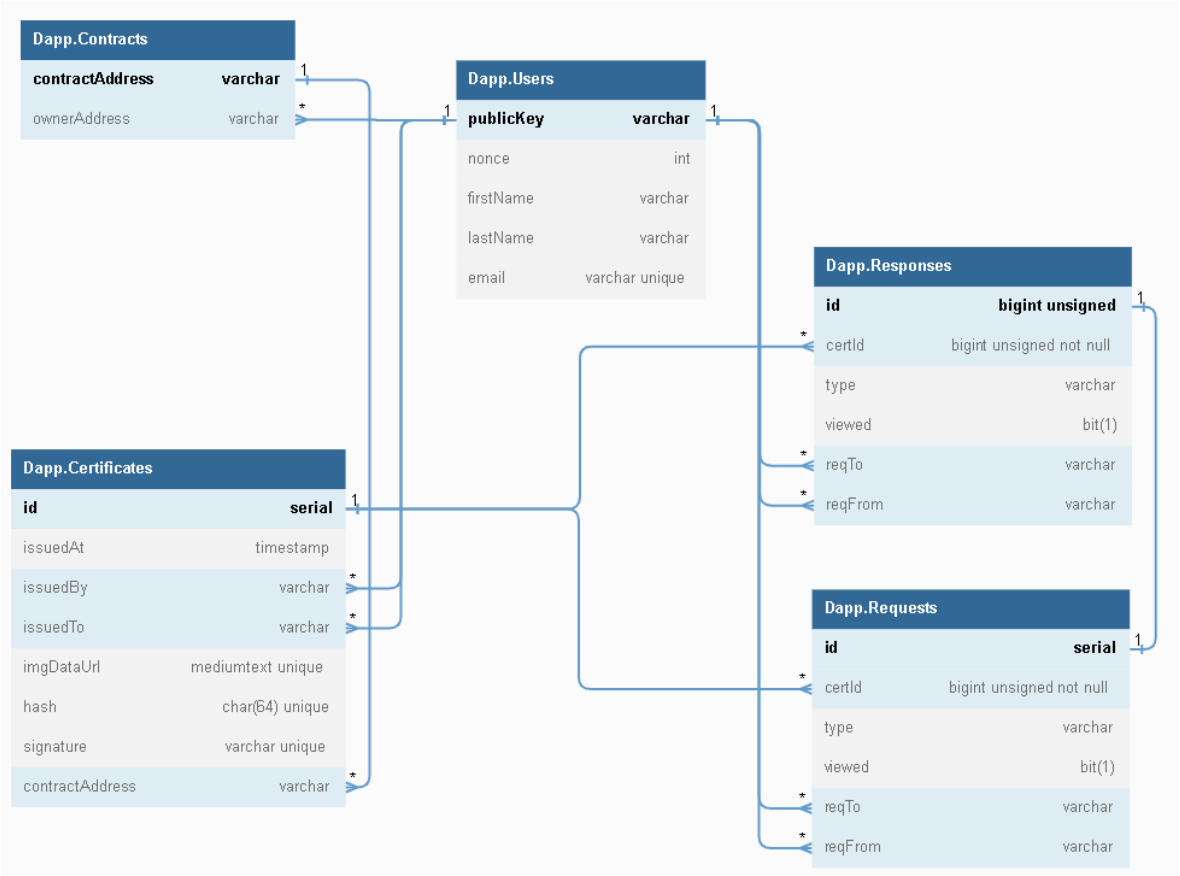
***Δεδομένα εξόδου***
Δεν επιστρέφονται δεδομένα πίσω εκτός από το τελικό αποτέλεσμα της αποθήκευσης, αν ήταν δηλαδή επιτυχής ή όχι.

**Περιγραφή**
Χρησιμοποιείται για την αποθήκευση στο σύστημα ενός νέου συμβολαίου που δημιουργήθηκε.

## 4.5 Βάση δεδομένων

Για την αποθήκευση των δεδομένων της εφαρμογής χρησιμοποιήθηκε σχεσιακή βάση δεδομένων και συγκεκριμένα το σύστημα διαχείρισης βάσεων δεδομένων MySql. Έπρεπε να μελετηθεί η συγκεκριμένη τεχνολογία για το στήσιμο και τη διαχείριση της βάσης καθώς και η γλώσσα SQL για την χρήση των επερωτημάτων. Επιπλέον μετά την ανάλυση των δεδομένων του συστήματος έπρεπε να γίνει και η μοντελοποίηση τους, δηλαδή ο σχεδιασμός του τρόπου αναπαράστασης τους καθώς και των συσχετίσεων μεταξύ τους.

### 4.5.1 Διάγραμμα οντοτήτων-συσχετίσεων




**Εικόνα 16: Διάγραμμα οντοτήτων-συσχετίσεων**

### 4.5.2 Users

Το αντικείμενο αυτό περιέχει τα στοιχεία του χρήστη. Βλέπουμε ότι ένας χρήστης χαρακτηρίζεται από το δημόσιο κλειδί του (publicKey). Το πεδίο nonce όπως έχει αναφερθεί προηγουμένως είναι ένας τυχαίος αριθμός που χρησιμοποιείται για την ταυτοποίηση του χρήστη. Κάθε χρήστης όπως βλέπουμε μπορεί να διαθέτει πολλά συμβόλαια και να σχετίζεται με πολλά πιστοποιητικά (ιδιόκτητα και εκδοθέντα).

### 4.5.3 Contracts

Τα συμβόλαια χαρακτηρίζονται από την διεύθυνση τους στο blockchain (contractAddress) ανήκουν σε έναν χρήστη και είναι ο χώρος αποθήκευσης των πιστοποιητικών. Σε ένα συμβόλαιο μπορεί ένας χρήστης να αποθηκεύσει πολλά πιστοποιητικά που έχει εκδώσει για άλλους.

### 4.5.4 Certificates

Τα πιστοποιητικά συμπεριλαμβάνουν τα παρακάτω στοιχεία:

- Ημερομηνία έκδοσης (timestamp)
- Ταυτότητα εκδότη (issuedBy), αντιστοιχεί στο δημόσιο κλειδί του εκδότη
- Ταυτότητα παραλήπτη (issuedTo), αντιστοιχεί στο δημόσιο κλειδί του παραλήπτη
- Έγγραφο (imgDataUrl)
- Αποτύπωμα του εγγράφου (hash)
- Υπογραφή του εγγράφου (signature)
- Διεύθυνση έξυπνου συμβολαίου (contractAddress)

Το έγγραφο αποθηκεύεται ως Data URL [24] της εικόνας του πιστοποιητικού δηλαδή ως συμβολοσειρά που αρχικά περιγράφει τον τύπο του αρχείου και στην συνέχεια τα δεδομένα κωδικοποιημένα σύμφωνα με την κωδικοποίηση base64.

Το αποτύπωμα του εγγράφου είναι το αποτέλεσμα μιας συνάρτησης κατακερματισμού του imgDataUrl. Το αποτύπωμα χαρακτηρίζει μοναδικά το συγκεκριμένο έγγραφο.

Αν το πιστοποιητικό έχει γίνει αποδεκτό από τον παραλήπτη του τότε θα πρέπει να έχει υπογραφεί από τον ίδιο με βάση το ιδιωτικό του κλειδί. Με τον έλεγχο της υπογραφής επιτυγχάνουμε δύο σκοπούς, την απόδειξη αποδοχής από τον παραλήπτη και την ακεραιότητα των δεδομένων που υπογράφηκαν από αυτόν.

Βλέπουμε πως ένα πιστοποιητικό μπορεί να σχετίζεται με πολλά Requests-Responses.

### 4.5.5 Requests-Responses

Οι πίνακες αυτοί αφορούν τα αιτήματα με τα οποία αλληλεπιδρούν οι χρήστες. Κάθε Request σχετίζεται με ένα συγκεκριμένο Certificate (certId), είναι ένα αίτημα δηλαδή πάνω σε ένα πιστοποιητικό που μπορεί να αφορά την έκδοση, την ανανέωση, την διαγραφή κ.α. Επίσης κάθε Request σχετίζεται με αντιστοιχία ένα προς ένα με ένα Response, δηλαδή για κάθε αίτημα υπάρχει και η απάντηση του. Το Request αφορά το αίτημα υπό την σκοπιά του παραλήπτη, ενώ το Response αφορά το αίτημα υπό την σκοπιά του αποστολέα.

Με το πεδίο type διαχωρίζουμε τα διάφορα είδη των αιτημάτων και η τιμή του ανανεώνεται καθώς ένα αίτημα μεταβαίνει από μία κατάσταση στην επόμενη. Το πεδίο viewed δείχνει εάν ο χρήστης έχει διαβάσει το συγκεκριμένο αίτημα ή όχι και έχει αντίστοιχα τις τιμές 1 και 0. Τα πεδία reqFrom, reqTo δείχνουν ποιος ξεκίνησε και σε ποιον απευθύνεται το αίτημα.

Όταν ένας χρήστης θέλει να στείλει ένα αίτημα σε κάποιον άλλο δημιουργεί δύο αντικείμενα το Request και το Response και τα αποθηκεύει στην βάση. Το Request είναι αυτό που θα διαβάσει ο παραλήπτης και θα ενημερωθεί για το αίτημα. Στην συνέχεια αν θέλει να απαντήσει θα γράψει στο Response. Από το Response θα ενημερωθεί ο αποστολέας για την απάντηση του παραλήπτη και με αυτόν τον τρόπο μπορούν οι δύο

χρήστες να συνεχίσουν να ανταλλάσσουν μηνύματα ενημερώνοντας τις καταστάσεις των αιτημάτων. Όταν ένας χρήστης απαντάει σε κάποιο αίτημα, τότε το μαρκάρει ως αδιάβαστο ώστε να το διαβάσει ο άλλος.

Για την ανανέωση των αιτημάτων που παρουσιάζονται στην γραφική διεπαφή γίνεται polling, ρωτάται δηλαδή το back-end ανά τακτά χρονικά διαστήματα και αν έχει καινούρια αιτήματα τα στέλνει πίσω.

## 4.6 Blockchain

Το blockchain αποτελεί το κατανεμημένο, ασφαλές και διαφανές δίκτυο αποθήκευσης των δεδομένων μας, για την εύκολη επαλήθευση τους από οποιονδήποτε στο μέλλον. Εκεί οι χρήστες δημιουργούν έξυπνα συμβόλαια, στα οποία αποθηκεύουν τα πιστοποιητικά που έχουν εκδώσει για άλλους χρήστες.

Μέρος της εργασίας αφορούσε την μελέτη του blockchain σε θεωρητικό επίπεδο για την κατανόηση των εννοιών, της λειτουργίας και των πλεονεκτημάτων που προσφέρει, αλλά και σε τεχνολογικό επίπεδο για την ανάπτυξη του δικτύου, των συναλλαγών και των έξυπνων συμβολαίων.

Χρησιμοποιήθηκε το δίκτυο Ethereum, η γλώσσα Solidity για την ανάπτυξη έξυπνων συμβολαίων και η βιβλιοθήκη web3.js για την αλληλεπίδραση με αυτά.

### 4.6.1 Τι είναι ένα έξυπνο συμβόλαιο;

Ένα έξυπνο συμβόλαιο είναι πολύ απλά ένα σύνολο από κώδικα και δεδομένα αποθηκευμένα στο blockchain, με το οποίο μπορεί κάποιος να επικοινωνεί, να εκτελεί εντολές και να επηρεάζει τα δεδομένα.

### 4.6.2 Δομή συναλλαγής

Μια συναλλαγή αποτελείται από τα παρακάτω βασικά στοιχεία:

- from - Η διεύθυνση του αποστολέα
- recipient - Η διεύθυνση του παραλήπτη ( λογαριασμού ή έξυπνου συμβολαίου )
- signature - Η υπογραφή της συναλλαγής από το ιδιωτικό κλειδί του αποστολέα
- nonce - Ένας μετρητής που δείχνει τον αύξων αριθμό της συναλλαγής από όλες τις συναλλαγές του αποστολέα
- value - Το ποσό των κρυπτονομισμάτων που μεταφέρεται στον παραλήπτη
- data - προαιρετικό πεδίο που περιλαμβάνει δεδομένα
- gasLimit - Το μέγιστο ποσό των μονάδων καυσίμου που μπορεί να καταναλώσει η συναλλαγή
- maxPriorityFeePerGas - Η μέγιστη τιμή ανά μονάδα καυσίμου που ο χρήστης είναι διατεθειμένος να πληρώσει ως φιλοδώρημα για την επαλήθευση της συναλλαγής
- maxFeePerGas - Η μέγιστη τιμή ανά μονάδα καυσίμου που ο χρήστης είναι διατεθειμένος να πληρώσει για την εκπλήρωση της συναλλαγής ( συμπεριλαμβάνει την παραπάνω τιμή )

Οι τύποι των συναλλαγών είναι οι παρακάτω:

- Συναλλαγές μεταφοράς κρυπτονομισμάτων από έναν λογαριασμό σε έναν άλλο
- Συναλλαγές δημιουργίας έξυπνων συμβολαίων

- Συναλλαγές εκτέλεσης λειτουργιών έξυπνων συμβολαίων

Στην εφαρμογή μας ενδιαφέρουν κυρίως οι δύο τελευταίες

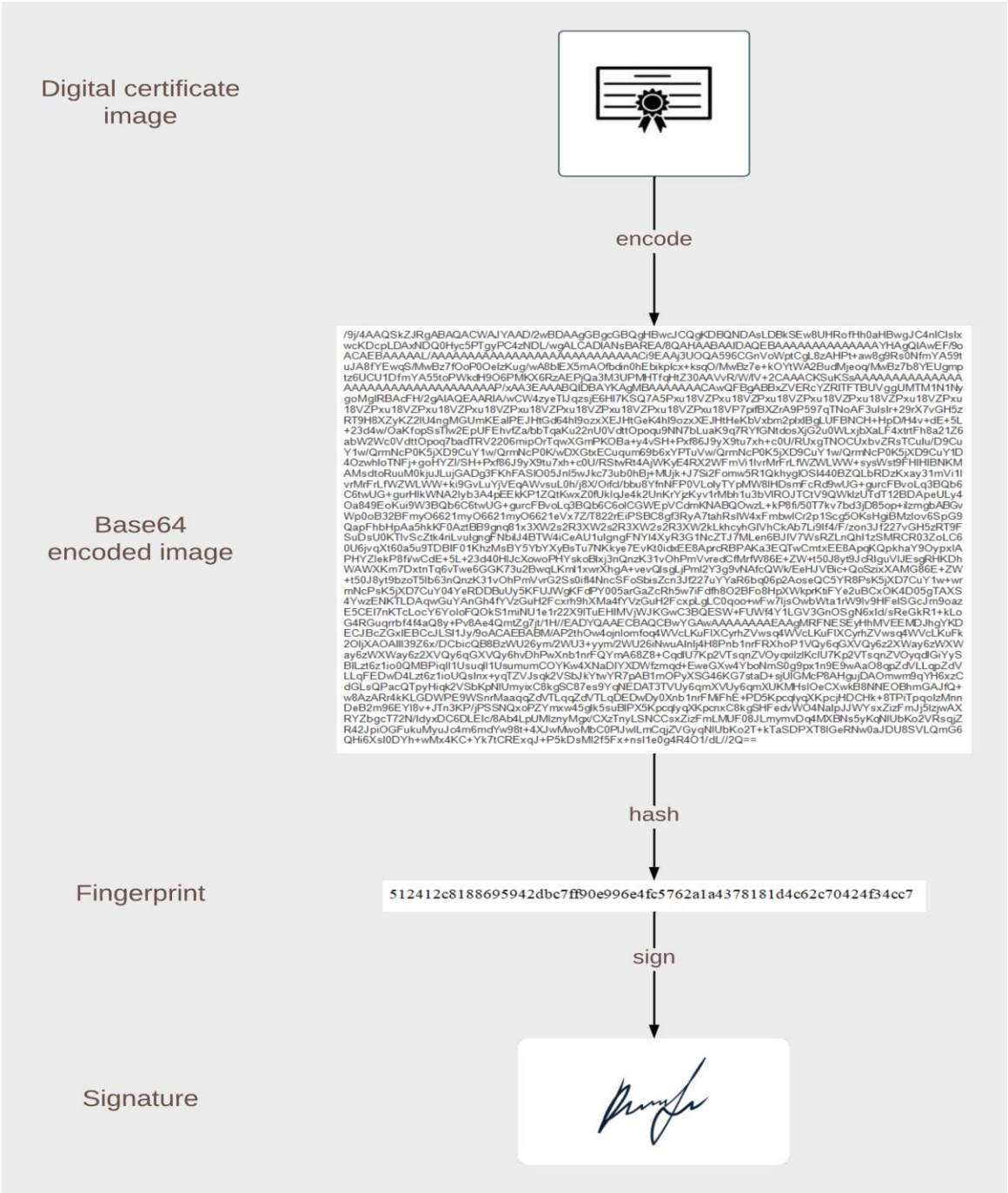
### 4.6.3 Δημιουργία συμβολαίου

Για την δημιουργία ενός έξυπνου συμβολαίου χρειαζόμαστε τον κώδικα του συμβολαίου μεταφρασμένο. Στην συνέχεια εκτελούμε μία συναλλαγή αφήνοντας το πεδίο του παραλήπτη κενό και συμπεριλαμβάνοντας στο πεδίο των δεδομένων τον μεταφρασμένο κώδικα του συμβολαίου. Η κενή μεταβλητή του παραλήπτη δηλώνει ότι εκτελούμε μια συναλλαγή δημιουργίας συμβολαίου. Στην συνέχεια με την επιτυχή εκτέλεση της συναλλαγής μας επιστρέφεται η απόδειξη εκτέλεσης της που περιέχει την διεύθυνση του έξυπνου συμβολαίου και την οποία θα χρησιμοποιούμε από εδώ και στο εξής για να αλληλεπιδράμε με αυτό.

### 4.6.4 Επικοινωνία με ένα συμβόλαιο

Για να επικοινωνήσει κάποιος χρήστης με ένα έξυπνο συμβόλαιο πρέπει να γνωρίζει την διεύθυνση του και στην συνέχεια να εκτελέσει μια συναλλαγή προς αυτή την διεύθυνση. Στην συναλλαγή πρέπει να τοποθετήσει την διεύθυνση του συμβολαίου ως παραλήπτη και στο πεδίο των δεδομένων να συμπεριλάβει την συνάρτηση που επιθυμεί να εκτελέσει και τις παραμέτρους που θέλει να περάσει στην συνάρτηση.

### 4.6.5 Αποθήκευση πιστοποιητικού σε έξυπνο συμβόλαιο




**Εικόνα 17: Παραγωγή αποτυπώματος και υπογραφής πιστοποιητικού**

Για την αποθήκευση ενός πιστοποιητικού στο Ethereum δεν αποθηκεύουμε απευθείας το πιστοποιητικό στο έξυπνο συμβόλαιο. Στην παραπάνω εικόνα βλέπουμε την διαδικασία μετατροπής του πιστοποιητικού στα δεδομένα του αποτυπώματος (fingerprint) και της υπογραφής του (signature) τα οποία θα χρησιμοποιήσουμε για την αποθήκευση του στο έξυπνο συμβόλαιο. Αρχικά η εικόνα του πιστοποιητικού κωδικοποιείται σύμφωνα με την κωδικοποίηση base64. Στην συνέχεια τα κωδικοποιημένα αυτά δεδομένα περνάνε από μία συνάρτηση κατακερματισμού και παράγεται το αποτύπωμα του πιστοποιητικού και τέλος το αποτύπωμα αυτό υπογράφεται με το ιδιωτικό κλειδί του χρήστη.

Η συνάρτηση κατακερματισμού που χρησιμοποιείται είναι η SHA-256 [25] και αυτό που κάνει πρακτικά είναι να απεικονίζει τα δεδομένα εισόδου σε συμβολοσειρές 256 bit.

Η συνάρτηση αυτή έχει τις δύο παρακάτω σημαντικές ιδιότητες:

1) Είναι μη αντιστρέψιμη, δηλαδή δεν μπορεί να παραχθεί η είσοδος με δεδομένη την έξοδο
2) Για δύο διαφορετικές εισόδους είναι απίθανο να παραχθεί το ίδιο αποτέλεσμα. Όσο μεγαλύτερο το μήκος των συμβολοσειρών εξόδου τόσο μειώνεται αυτή η πιθανότητα.

Αυτό που τελικά αποθηκεύεται στο blockchain είναι η αντιστοίχιση αποτύπωμα-υπογραφή. Έχοντας αυτά τα δεδομένα μπορούμε εύκολα να επαληθεύσουμε το πιστοποιητικό.

Με αυτό τον τρόπο αποθήκευσης επιτυγχάνονται κάποια σημαντικά αποτελέσματα.

1) Τα δεδομένα που αποθηκεύουμε στο blockchain κοστίζουν και επομένως αποθηκεύοντας μόνο το hash, δηλαδή 64 χαρακτήρες μειώνουμε σημαντικά το χρηματικό κόστος που απαιτείται για την αποθήκευση ενός πιστοποιητικού.
2) Καθώς δεν αποθηκεύουμε το πιστοποιητικό στο blockchain αλλά το αποτύπωμά του ασφαλίζουμε τα προσωπικά δεδομένα του χρήστη, αφού η διαδικασία όπως είπαμε είναι μη αντιστρέψιμη και επομένως δεν μπορεί κάποιος να τα παραβιάσει. Έτσι ικανοποιούμε την μη λειτουργική απαίτηση της ιδιωτικότητας των δεδομένων.
3) Έχοντας αποθηκευμένη την ψηφιακή υπογραφή μαζί με το αποτύπωμα μπορούμε να επαληθεύσουμε την ταυτότητα του ιδιοκτήτη του πιστοποιητικού. Επίσης ικανοποιούμε και την μη λειτουργική απαίτηση της ακεραιότητας των δεδομένων καθώς οποιαδήποτε αλλαγή στα δεδομένα του πιστοποιητικού καθιστά την υπογραφή άκυρη.

### 4.6.6 Κώδικας συμβολαίου

Στην παρακάτω εικόνα φαίνεται ο κώδικας του έξυπνου συμβολαίου.

```
pragma solidity >=0.4.22 <0.9.0;

contract IssuedCertificates {
    address public owner;
    mapping (string => string) public signedCertificates;

    constructor()  {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    function addCertificate(string memory hash, string memory signature) public onlyOwner {
        signedCertificates[hash] = signature;
    }

    function deleteCertificate(string memory hash) public onlyOwner {
        delete signedCertificates[hash];
    }
}
```

**Εικόνα 18: Κώδικας έξυπνου συμβολαίου**

Μεταβλητές

owner

Καθορίζει τον δημιουργό του συμβολαίου και αρχικοποιείται κατά την δημιουργία του με την τιμή msg.sender που εκφράζει την διεύθυνση που εκτέλεσε την συναλλαγή δημιουργίας του συμβολαίου.

signedCertificates

Ένα σύνολο από υπογεγραμμένα πιστοποιητικά αποθηκευμένα κατά την αντιστοιχία αποτύπωμα-υπογραφή. Το αποτύπωμα του πιστοποιητικού λειτουργεί ως κλειδί για την ανάκτηση της υπογραφής.

Modifier

onlyOwner

Ένας τροποποιητής που απαιτεί ο αποστολέας της συναλλαγής που εκτελείται να είναι ο δημιουργός του συμβολαίου. Μπορεί να τοποθετηθεί σε συναρτήσεις και επιτρέπει την εκτέλεσή τους μόνο εφόσον ικανοποιούνται οι συνθήκες του.

Συναρτήσεις

addCertificate

Δέχεται ως ορίσματα το αποτύπωμα και την υπογραφή ενός πιστοποιητικού και τα αποθηκεύει στο συμβόλαιο.

deleteCertificate

Δέχεται ως όρισμα το αποτύπωμα ενός πιστοποιητικού και το διαγράφει από τα αποθηκευμένα πιστοποιητικά του συμβολαίου.

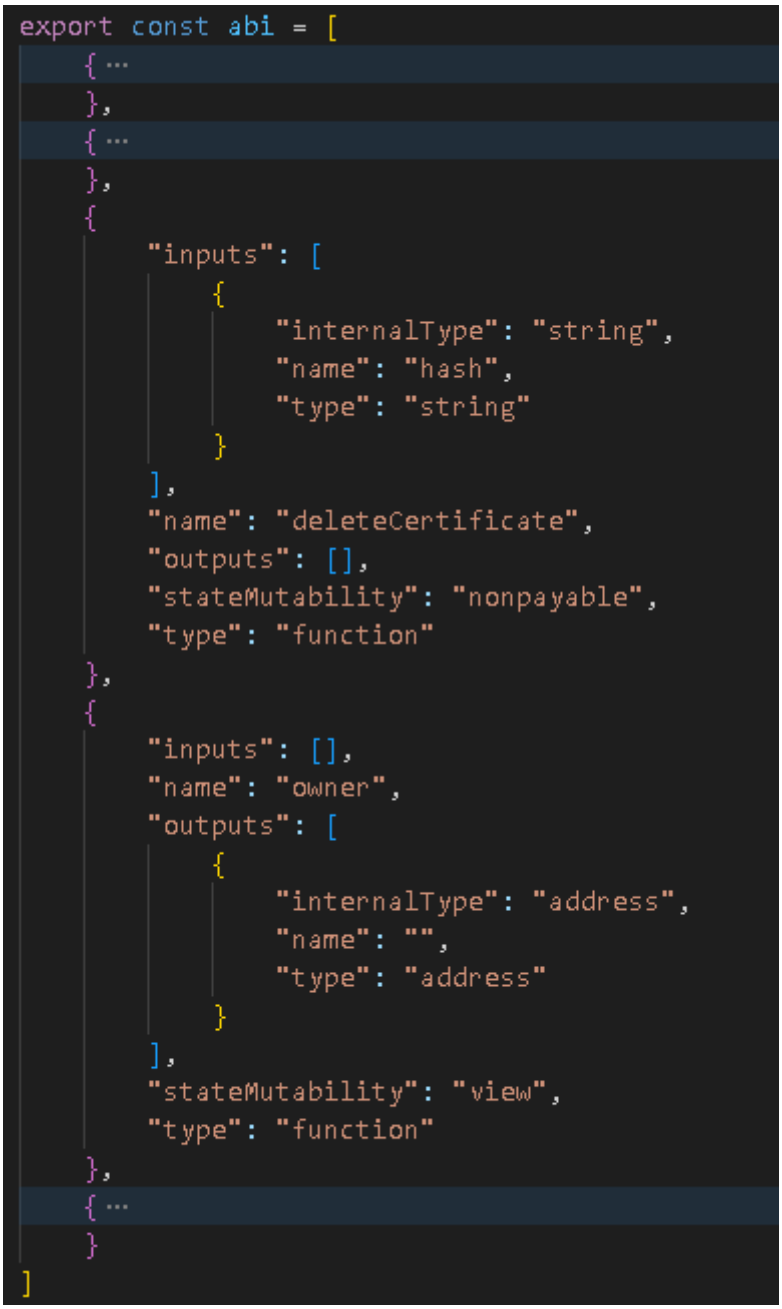
Και οι δύο παραπάνω συναρτήσεις έχουν τον modifier onlyOwner δηλαδή μόνο ο δημιουργός του συμβολαίου έχει την δυνατότητα να προσθέσει και να διαγράψει πιστοποιητικά σε αυτό.

Στην εφαρμογή μας χρειαζόμαστε την ανάκτηση των τιμών owner και των αποθηκευμένων πιστοποιητικών στην μεταβλητή signedCertificates. Δεν χρειάζεται να δημιουργήσουμε επιπλέον συναρτήσεις για αυτό τον σκοπό καθώς η Solidity παράγει αυτόματα συναρτήσεις ανάκτησης για τις δημόσιες μεταβλητές ενός συμβολαίου.

### 4.6.7 ABI συμβολαίου

Στην παρακάτω εικόνα φαίνεται το ABI (Application Binary Interface) του συμβολαίου. Το ABI αποτελεί αντικείμενο περιγραφής των συναρτήσεων του συμβολαίου και των ορισμάτων που αυτές δέχονται. Είναι απαραίτητο ώστε ο χρήστης που διενεργεί μια συναλλαγή να μπορεί να καθορίσει τα δεδομένα που όταν το συμβόλαιο διαβάσει, θα αναγνωρίσει και θα καλέσει την κατάλληλη συνάρτηση με τα κατάλληλα ορίσματα.

Βλέπουμε την συνάρτηση deleteCertificate που δέχεται ως είσοδο ένα string με όνομα hash καθώς και την συνάρτηση owner η οποία δημιουργήθηκε αυτόματα για την αντίστοιχη δημόσια μεταβλητή και επιστρέφει μια μεταβλητή τύπου address, δηλαδή το δημόσιο κλειδί του ιδιοκτήτη του συμβολαίου.




**Εικόνα 19: ABI έξυπνου συμβολαίου**

### 4.6.8 Εκτέλεση συναρτήσεων του συμβολαίου με χρήση της βιβλιοθήκης Web3

Παρακάτω βλέπουμε τον κώδικα που αποτελεί το σημείο σύνδεσης του front-end με το blockchain και χρησιμοποιεί την βιβλιοθήκη web3.js για την εκτέλεση συναρτήσεων του έξυπνου συμβολαίου.

```
import Web3 from 'web3';
import {code, abi} from '../contract/Contract_abi_code.js';

export default function create_contract(account, provider){
    let web3 = new Web3(provider);
    let contract = new web3.eth.Contract(abi);
    contract.options.from = account;
    let obj = {};

    obj.deploy = async function (){
        let newContract = await contract.deploy({data : code}).send();
        return newContract.options.address;
    }

    obj.addCertificate = async function (contractAddress, hash, signature){
        contract.options.address = contractAddress;
        let receipt = await contract.methods.addCertificate(hash, signature).send();
        return receipt;
    }

    obj.deleteCertificate = async function (contractAddress, hash){
        contract.options.address = contractAddress;
        let receipt = await contract.methods.deleteCertificate(hash).send();
        return receipt;
    }

    obj.getOwner = async function (contractAddress){
        contract.options.address = contractAddress;
        let owner = await contract.methods.owner().call();
        return owner;
    }

    obj.getSignedCertificate = async function (contractAddress, hash){
        contract.options.address = contractAddress;
        let signature = await contract.methods.signedCertificates(hash).call();
        return signature;
    }
    obj.ecRecover = function (hash, signature){
        return web3.eth.personal.ecRecover(hash, signature);
    }
    return obj;
}
```

**Εικόνα 20: Αντικείμενο εκτέλεσης λειτουργιών του έξυπνου συμβολαίου**

Αρχικά εισάγουμε τρία πράγματα την βιβλιοθήκη web3, τον μεταφρασμένο κώδικα (code) και το ABI του συμβολαίου. Βλέπουμε ότι εξάγεται η συνάρτηση create_contract η οποία δέχεται ως ορίσματα τον λογαριασμό του χρήστη (account) και έναν πάροχο (provider) και επιστρέφει ένα αντικείμενο του οποίο διαθέτει ως μεθόδους όλες τις ενέργειες που αφορούν ένα έξυπνο συμβόλαιο. Ο πάροχος είναι αυτός ο οποίος είναι συνδεδεμένος με το blockchain και θα εκτελεί τις συναλλαγές και στην περίπτωσή μας ως πάροχος δίνεται το API του MetaMask.

Στην αρχή της συνάρτησης ορίζουμε την τοπική μεταβλητή contract. Η μεταβλητή αυτή αποτελεί το αντικείμενο που αναπαριστά το συμβόλαιό μας και δημιουργείται με την χρήση της συνάρτησης web3.eth.Contract της βιβλιοθήκης δίνοντας ως όρισμα το ABI του συμβολαίου. Επίσης ορίζουμε ως αποστολέα για τις συναλλαγές του συμβολαίου που θα εκτελούνται τον λογαριασμό του χρήστη.

Στην συνέχεια ορίζονται οι διάφορες συναρτήσεις οι οποίες εκτελούν συναλλαγές που αφορούν το συμβόλαιο και επιστρέφουν πίσω τα αποτελέσματα που χρειαζόμαστε. Αξίζει να παρατηρήσουμε την πρώτη συνάρτηση που έχει το όνομα deploy και αποτελεί την συνάρτηση δημιουργίας του συμβολαίου. Όπως είχαμε αναφέρει χρησιμοποιεί στο πεδίο data της συναλλαγής τον μεταφρασμένο κώδικα του συμβολαίου.

## 4.7 Διαγράμματα ροής λειτουργιών

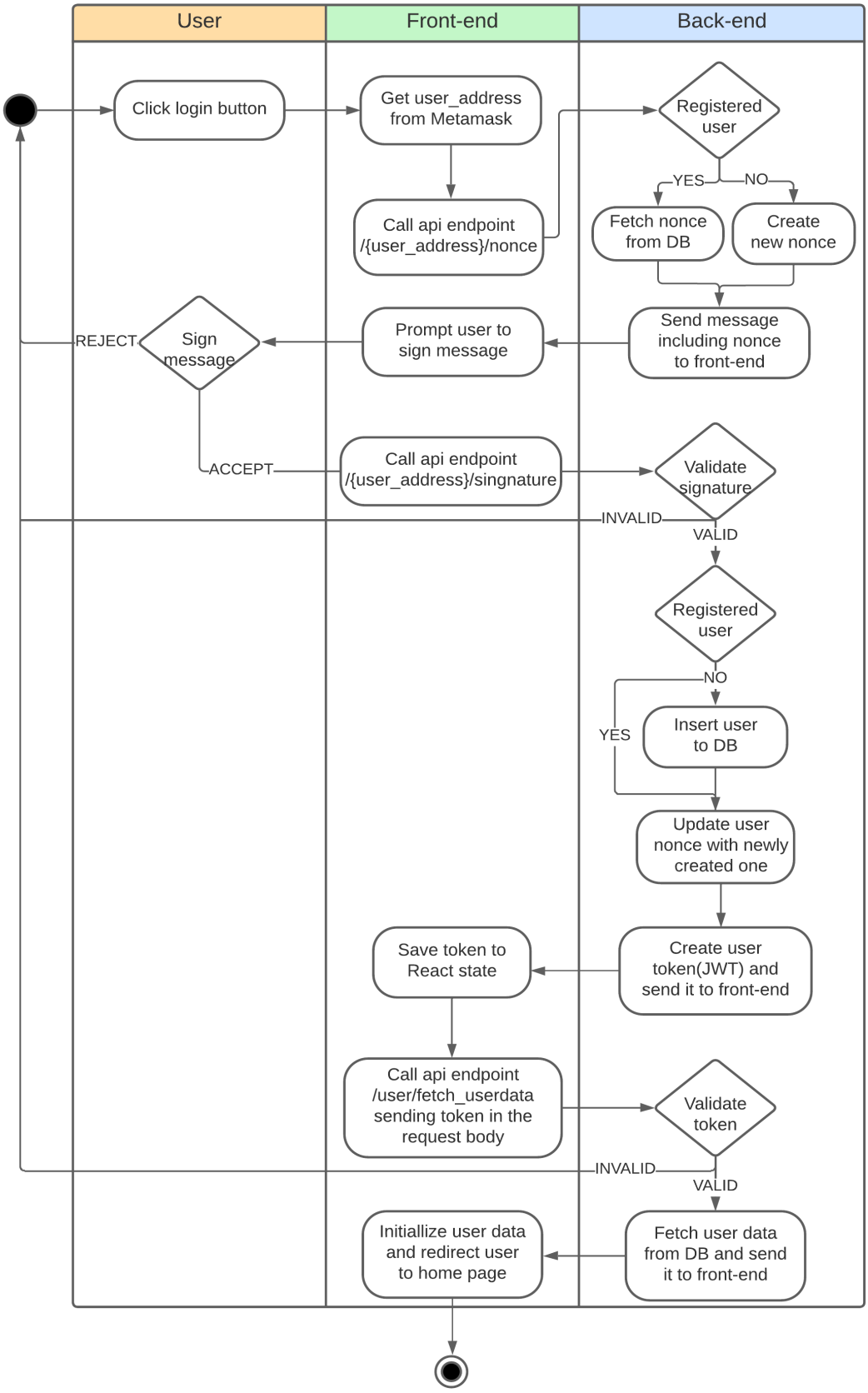
Έχοντας αναλύσει τα επιμέρους υποσυστήματα της εφαρμογής μας, θα δούμε τώρα όλες τις λειτουργίες που εκτελεί, τα βήματά τους καθώς και τα μηνύματα επικοινωνίας των υποσυστημάτων κατά την εκτέλεση αυτών των λειτουργιών.

### 4.7.1 Είσοδος στην εφαρμογή

Η πρώτη λειτουργία οποιουδήποτε χρήστη για την χρήση της εφαρμογής είναι η ταυτοποίηση του και είσοδος σε αυτή.

Διαδικασία εισόδου και ταυτοποίησης

Πρέπει ο χρήστης να έχει εγκατεστημένο το MetaMask αλλιώς το κουμπί εισόδου τον οδηγεί σε εγκατάσταση του. Μόλις το εγκαταστήσει το κουμπί εισόδου οδηγεί στην ταυτοποίησή του. Για την είσοδο του ο χρήστης θα πρέπει να είναι συνδεδεμένος στον λογαριασμό του στο MetaMask.




**Εικόνα 21: Διάγραμμα εισόδου στην εφαρμογή**

Μόλις ο χρήστης πατήσει το κουμπί εισόδου γίνεται ανάκτηση της διεύθυνσης του από το MetaMask και στην συνέχεια κλήση στο API endpoint /{user_address}/nonce. Το back-end ελέγχει αν ο χρήστης είναι εγγεγραμμένος στο σύστημα ή αν συνδέεται πρώτη φορά. Στην πρώτη περίπτωση το nonce του χρήστη ανακτάται από την βάση ενώ στην δεύτερη δημιουργείται εκείνη τη στιγμή. Στην συνέχεια επιστρέφεται στον χρήστη ένα μήνυμα προς υπογραφή που περιλαμβάνει το nonce. Τότε αναδύεται το παράθυρο του MetaMask και ζητάει από τον χρήστη να υπογράψει το μήνυμα. Αν ο χρήστης το υπογράψει γίνεται νέα κλήση αυτή τη φορά στο API endpoint /{user_address}/signature και στο σώμα του αιτήματος περιέχεται η υπογραφή του μηνύματος και το μήνυμα. Το back-end ελέγχει την υπογραφή και αν την επαληθεύσει επιστρέφει πίσω ένα token το οποίο θα χρησιμοποιείται για οποιοδήποτε αίτημα του χρήστη στο μέλλον. Επίσης αν ο χρήστης συνδέεται πρώτη φορά τότε εισάγεται η διεύθυνση του και το nonce στη βάση αλλιώς απλά ανανεώνεται το nonce του χρήστη. Στην συνέχεια το front-end αποθηκεύει το token και εκτελεί νέο αίτημα για την λήψη των δεδομένων του χρήστη, στο API endpoint /user/fetch_userdata περνώντας το token στο σώμα του αιτήματος. Το back-end ελέγχει το token και αν το επαληθεύσει στέλνει πίσω όλα τα δεδομένα του χρήστη. Το front-end λαμβάνει τα δεδομένα και ανακατευθύνει τον χρήστη στην αρχική σελίδα της εφαρμογής.

Δημιουργία token

Το token δημιουργείται από το back-end για τον χρήστη με την χρήση της βιβλιοθήκης jsonwebtoken. Ακολουθεί το πρότυπο token JWT και αποτελείται από τρία τμήματα, την κεφαλίδα, τα δεδομένα και την υπογραφή. Η κεφαλίδα περιέχει κάποια δεδομένα για το token όπως χρόνος ζωής, ημερομηνία έκδοσης κ.α. Τα δεδομένα περιέχουν τα δεδομένα που χρειάζονται για την ταυτοποίηση του χρήστη στο σύστημα και στην προκειμένη περίπτωση χρησιμοποιείται το δημόσιο κλειδί του. Η υπογραφή είναι υπογραφή των παραπάνω από το back-end με βάση κάποια μυστική φράση, ώστε να εντοπίζεται αλλοίωση των δεδομένων και να ακυρώνεται το token.

Περιορισμός πρόσβασης σε πόρους

Για τον περιορισμό πρόσβασης του χρήστη λοιπόν λαμβάνουμε το token του και ελέγχουμε την εγκυρότητα του. Αν επαληθευτεί τότε δίνουμε στην χρήστη πρόσβαση σε δεδομένα που αντιστοιχούν στο δημόσιο κλειδί του. Ο έλεγχος αυτός γίνεται από middleware που δημιουργήθηκε με την βοήθεια του Express και ενεργοποιείται σε κάθε endpoint που ξεκινάει με το πρόθεμα /user. Το πρόθεμα αυτό δηλώνει συνδεδεμένο χρήστη.

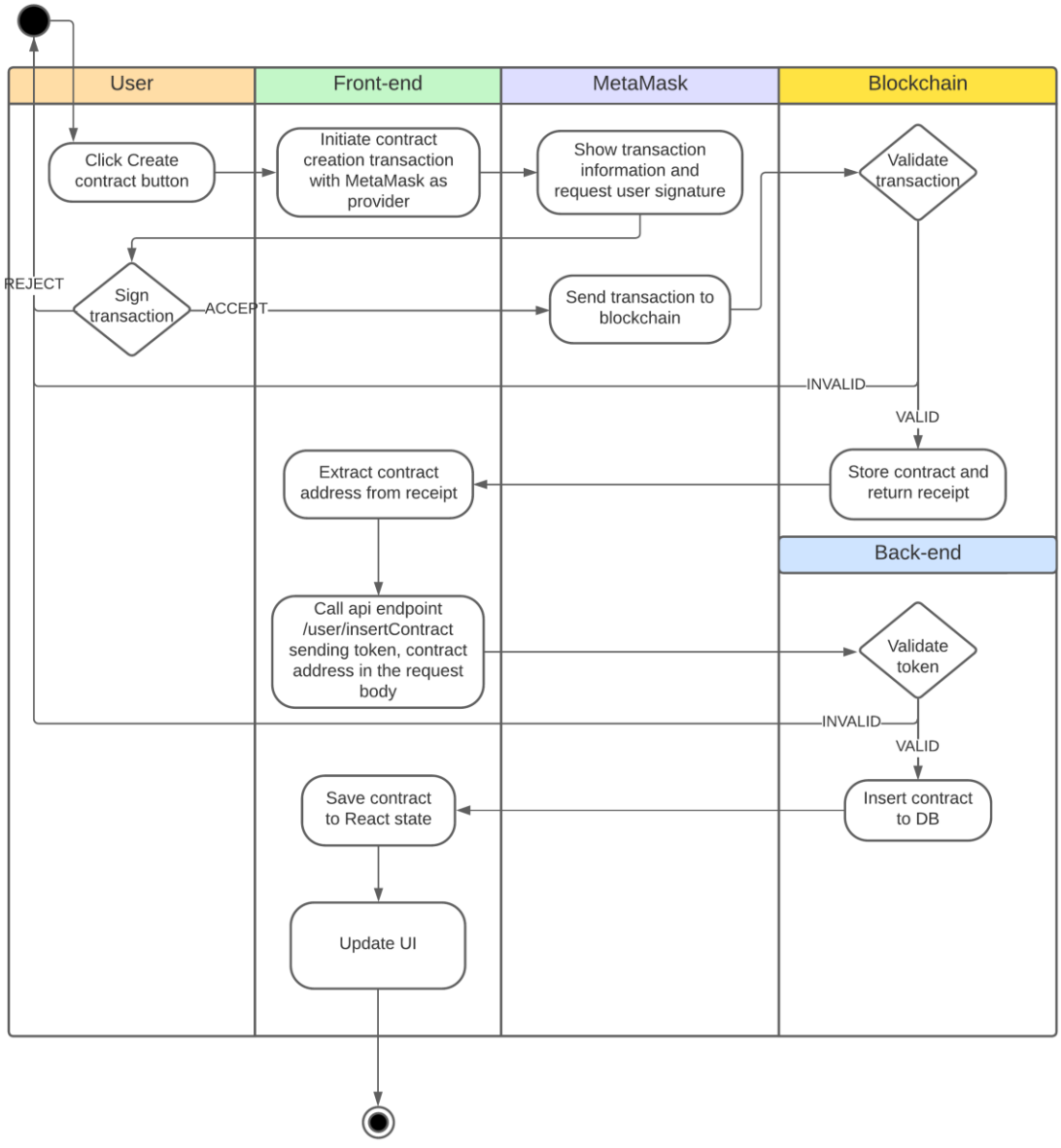
```
//verifies token for every request starting with /user
app.use('/user', (req, res, next) =>{
    let token = req.body.jwt;
    req.token = verify_token (token, SECRET);
    if (!req.token){
        res.status(404).json({
            success : false,
            error: "INVALID TOKEN"
        });
    }
    else{
        next();
    }
});
```

**Εικόνα 22: Middleware επαλήθευσης token**

Προβολή ιδιόκτητων ή εκδοθέντων πιστοποιητικών

Μετά την είσοδο του χρήστη στο σύστημα και στην αρχική σελίδα μπορεί να πλοηγηθεί στην γραφική διεπαφή και να δει τα πιστοποιητικά του (ιδιόκτητα και εκδοθέντα) ή να εκτελέσει κάποιες ενέργειες. Επίσης μπορεί να δει αιτήματα που του έχουν γίνει για τα διάφορα πιστοποιητικά και να αλληλεπιδράσει με αυτά. Για παράδειγμα μπορεί να υπογράψει ένα αίτημα έκδοσης πιστοποιητικού που του έχει σταλεί.

### 4.7.2 Δημιουργία συμβολαίου




**Εικόνα 23: Διάγραμμα δημιουργίας συμβολαίου**

Για να εκδώσει πιστοποιητικά κάποιος χρήστης πρέπει πρώτα να έχει δημιουργήσει ένα έξυπνο συμβόλαιο στο blockchain στο οποίο αυτά θα αποθηκευτούν. Για να γίνει αυτό πρέπει να πλοηγηθεί στην σελίδα των πιστοποιητικών που έχει εκδώσει και να πατήσει το

κουμπί Create contract. Τότε αναδύεται το παράθυρο του MetaMask το οποίο δείχνει στον χρήστη την συναλλαγή δημιουργίας συμβολαίου. Αν αποδεχτεί και υπογράψει την συναλλαγή, τότε αυτή στέλνεται στο blockchain όπου επαληθεύεται και αν είναι έγκυρη το έξυπνο συμβόλαιο δημιουργείται. Το front-end λαμβάνει την απόδειξη της συναλλαγής που περιέχει την διεύθυνση του συμβολαίου που δημιουργήθηκε και στην συνέχεια γίνεται κλήση στο API endpoint /user/insertContract με το παρακάτω αντικείμενο:

```
{
        jwt : {json web token},
        contractAddress : {contract address}
}
```

Όπως βλέπουμε το αντικείμενο αυτό περιλαμβάνει το token και την διεύθυνση του νέου συμβολαίου που δημιουργήθηκε. Στην συνέχεια αφού γίνει ταυτοποίηση του χρήστη με επαλήθευση του token, το νέο συμβόλαιο αποθηκεύεται στην βάση. Τέλος ανανεώνεται η λίστα των συμβολαίων του χρήστη στην γραφική διεπαφή.

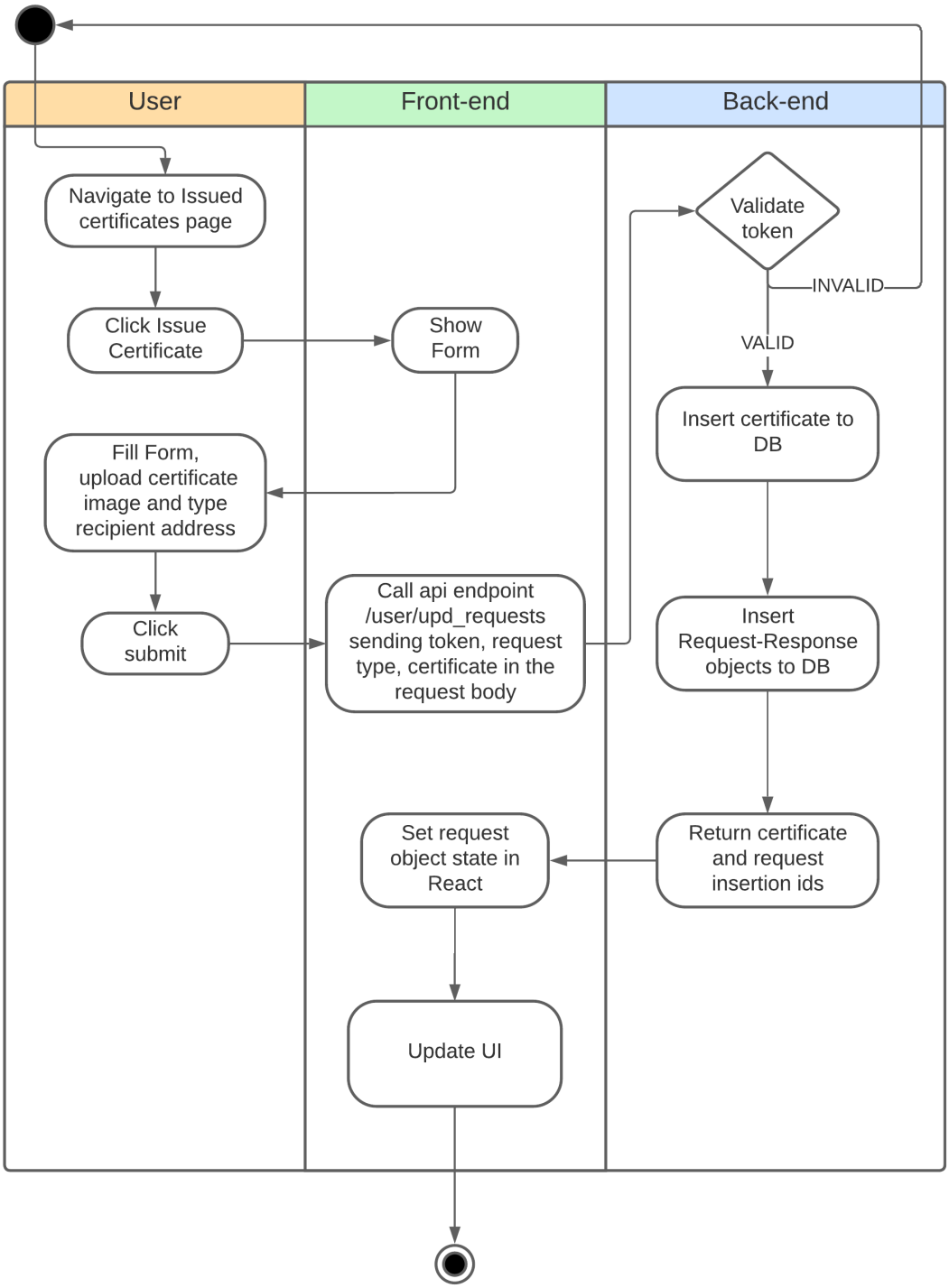
Μπορούμε να δημιουργήσουμε και άλλα συμβόλαια και να επιλέξουμε από το dropdown menu της γραφικής διεπαφής ποιο συμβόλαιο θα χρησιμοποιείται για τα νέα πιστοποιητικά που θα εκδίδουμε κάθε φορά.

### 4.7.3 Έκδοση πιστοποιητικού

Για την έκδοση ενός πιστοποιητικού ακολουθούνται τρία βήματα.

1) Αρχικά ο εκδότης του πιστοποιητικού δημιουργεί ένα αίτημα προς τον παραλήπτη που περιλαμβάνει το πιστοποιητικό και αποθηκεύεται στην βάση.
2) Στην συνέχεια ο παραλήπτης διαβάζει από την βάση το νέο αίτημα που τον αφορά, βλέπει ότι είναι αίτημα έκδοσης πιστοποιητικού και επιλέγει αν θα το υπογράψει ή όχι. Αν το υπογράψει αυτομάτως δηλώνει ότι συμφωνεί με τους ισχυρισμούς του πιστοποιητικού. Στην συνέχεια το αίτημα με την προσθήκη της υπογραφής του χρήστη αποθηκεύεται στην βάση.
3) Ο εκδότης διαβάζει το αίτημα από την βάση και αντιλαμβάνεται ότι ο χρήστης το έχει υπογράψει και οφείλει τώρα να το αποθηκεύσει στο συμβόλαιό του στο blockchain. Αν η υποβολή στο blockchain πετύχει τότε το πιστοποιητικό αποθηκεύεται στην βάση και είναι πλέον προσβάσιμο από τον εκδότη και τον παραλήπτη του.

Αίτημα έκδοσης πιστοποιητικού



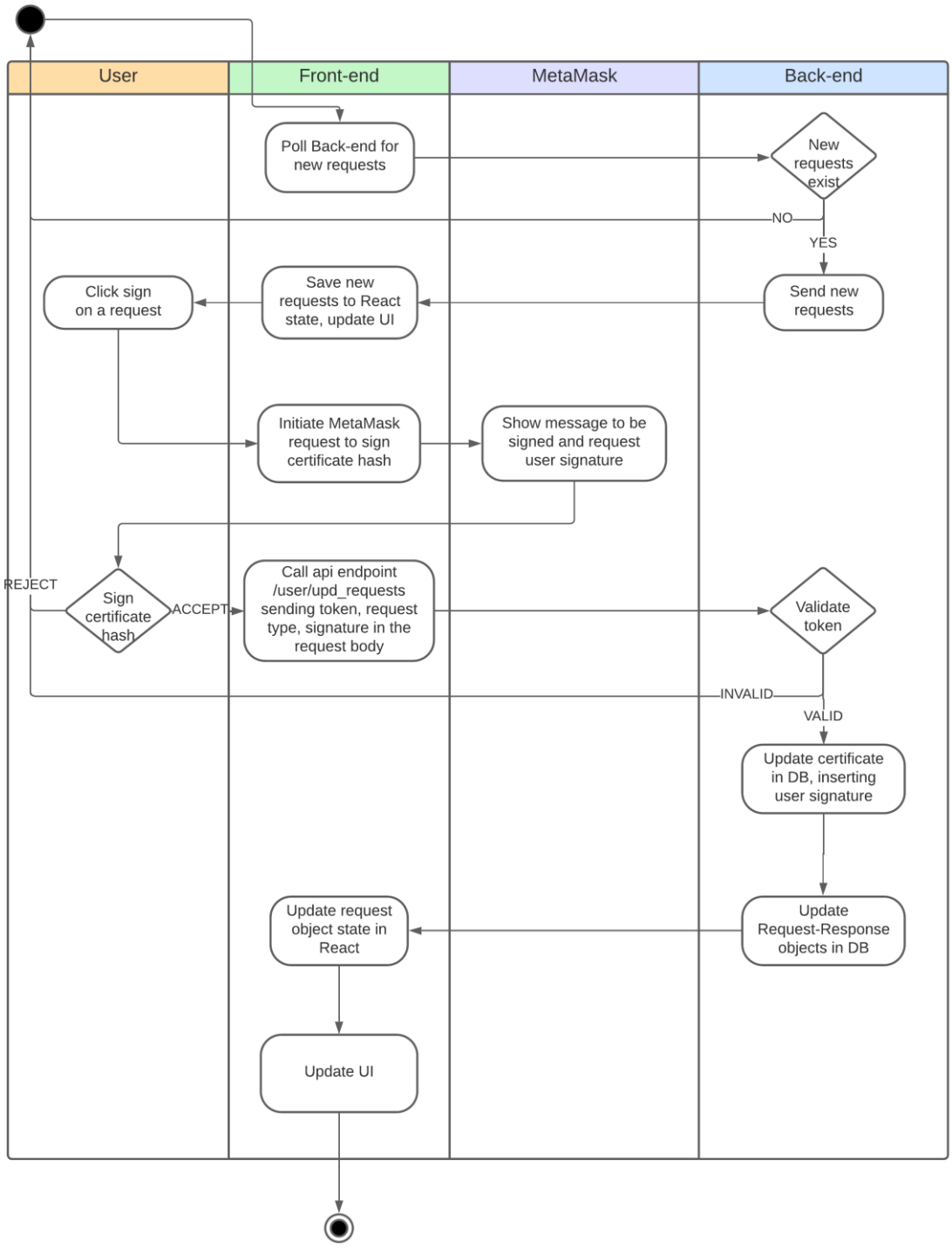

**Εικόνα 24: Διάγραμμα έκδοσης πιστοποιητικού**

Ο εκδότης του πιστοποιητικού συμπληρώνει την φόρμα έκδοσης. Αρχικά ανεβάζει το πιστοποιητικό και στην συνέχεια πληκτρολογεί την διεύθυνση του παραλήπτη. Τότε γίνεται κλήση στο API endpoint /user/upd_requests με το παρακάτω αντικείμενο:

```
{
        type : ‘ISS’,
        jwt : {json web token},
        data : {
                cert : {certificate}
        }
}
```

Όπως βλέπουμε το αντικείμενο αυτό περιλαμβάνει το token, τον τύπο του αιτήματος ‘ISS’ που δηλώνει την έκδοση πιστοποιητικού (ISSUE) και το πιστοποιητικό. Στην συνέχεια αφού γίνει ταυτοποίηση του χρήστη με επαλήθευση του token, εισάγεται στην βάση το νέο πιστοποιητικό και τα αντικείμενα Request-Response που αναπαριστούν το αίτημα του. Τέλος δημιουργείται στην γραφική διεπαφή του το αντικείμενο που αντιπροσωπεύει το αίτημα του και θα ανανεώνεται καθώς αυτό θα μεταβαίνει σε άλλες καταστάσεις.

Υπογραφή του πιστοποιητικού από τον παραλήπτη




**Εικόνα 25: Διάγραμμα υπογραφής πιστοποιητικού**

Το παραπάνω διάγραμμα εκτός από την υπογραφή του πιστοποιητικού από τον παραλήπτη δείχνει και την διαδικασία με την οποία η γραφική διεπαφή του χρήστη ανανεώνεται με τα νέα αιτήματα που έχουν γίνει προς αυτόν.

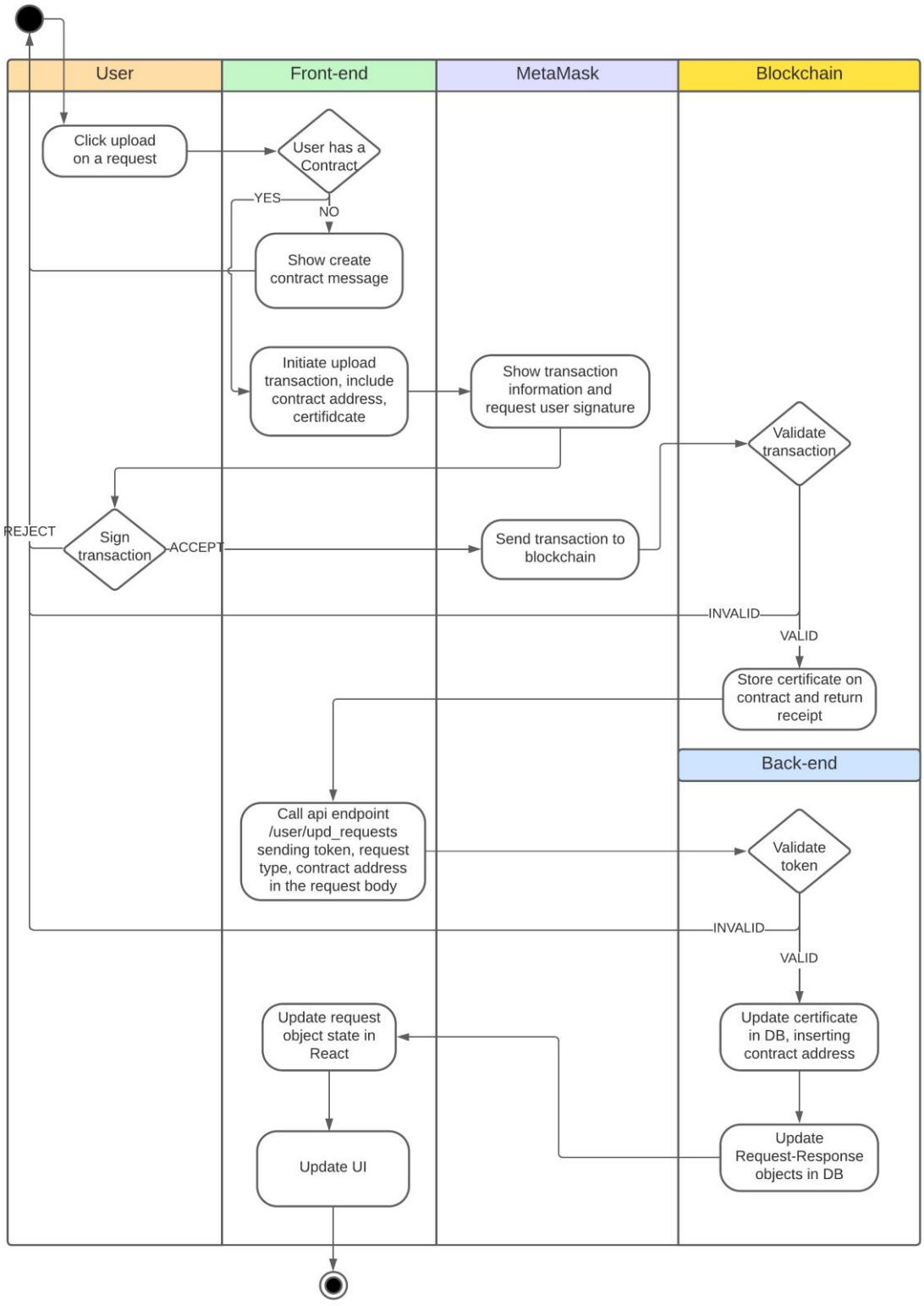
Ας υποθέσουμε ότι ενώ είμαστε συνδεδεμένοι κάποιος δημιουργεί ένα νέο αίτημα έκδοσης πιστοποιητικού για εμάς, το οποίο είναι τώρα αποθηκευμένο στη βάση. Το front-end ρωτάει συνέχεια το back-end για νέα αιτήματα και αν υπάρχουν τα λαμβάνει ως απάντηση και ανανεώνει την γραφική διεπαφή.

Ο χρήστης μπορεί να επιλέξει να υπογράψει κάποιο από τα αιτήματα έκδοσης πιστοποιητικού που του έχουν σταλεί. Τότε αναδύεται το παράθυρο του MetaMask το οποίο δείχνει στον χρήστη το μήνυμα που θα υπογράψει, το hash δηλαδή του πιστοποιητικού. Αν ο χρήστης το υπογράψει τότε γίνεται κλήση στο API endpoint /user/upd_requests με το παρακάτω αντικείμενο:

```
{
        type : ‘ISS_SIGNED’,
        jwt : {json web token},
        data : {
                cert : {certificate},
                signature : {user signature},
                req_id : { request id}
        }
}
```

Όπως βλέπουμε το αντικείμενο αυτό περιλαμβάνει το token, τον τύπο του αιτήματος ‘ISS_SIGNED’, την υπογραφή του χρήστη, το πιστοποιητικό και το id του αιτήματος που αφορά αυτή η κλήση. Η μετάβαση του αιτήματος στην νέα κατάσταση δηλώνει ότι το αρχικό αίτημα έχει υπογραφεί από τον χρήστη. Στην συνέχεια αφού γίνει ταυτοποίηση του χρήστη με επαλήθευση του token, το πιστοποιητικό που υπάρχει στην βάση ανανεώνεται με την υπογραφή του. Ακόμα ανανεώνονται τα αντικείμενα Request-Response καθώς το αρχικό αίτημα έχει μεταβεί σε νέα κατάσταση. Τέλος ανανεώνεται το αντικείμενο του αιτήματος στην γραφική διεπαφή του χρήστη.

Αποθήκευση πιστοποιητικού στο blockchain




**Εικόνα 26: Διάγραμμα αποθήκευσης πιστοποιητικού στο blockchain**

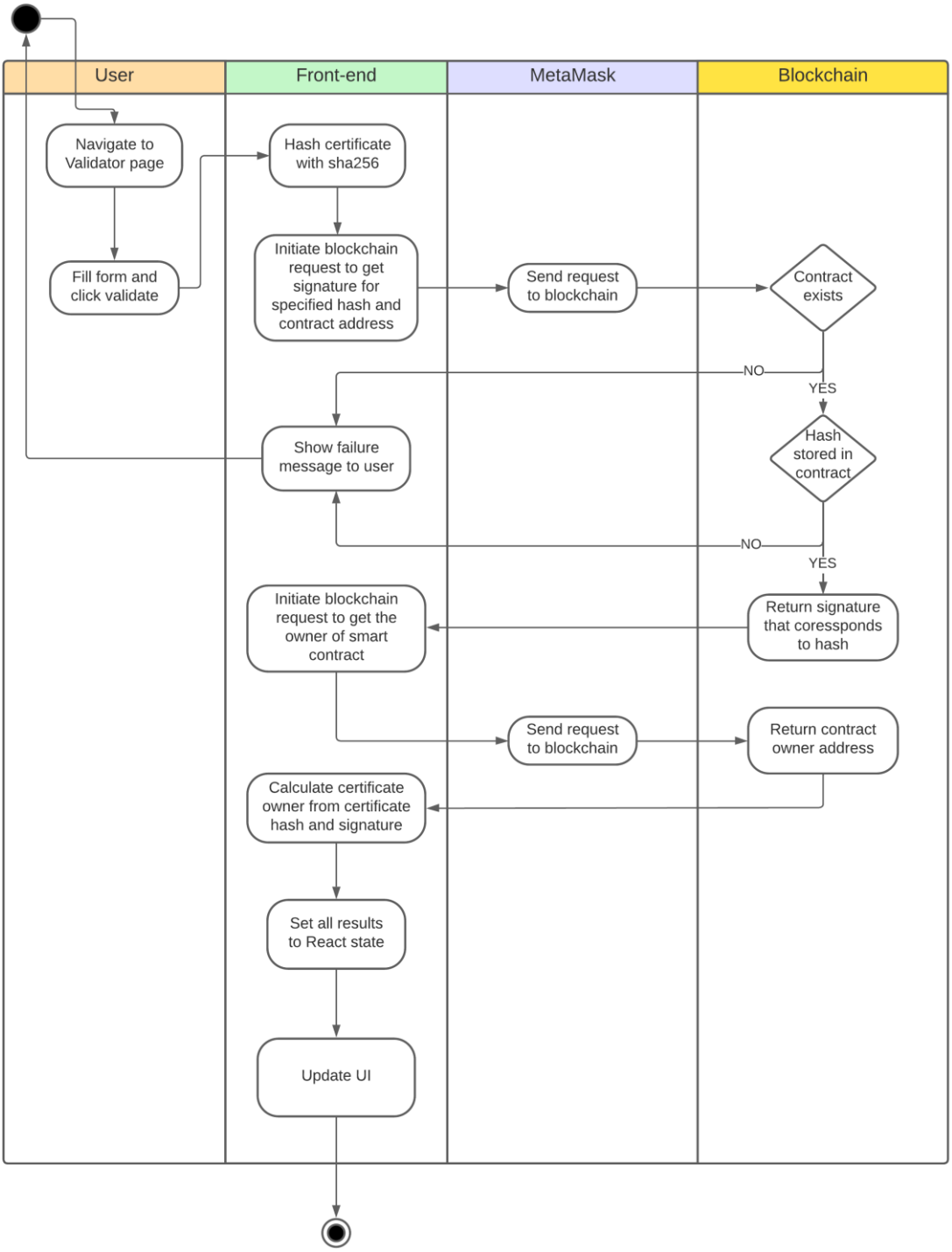
Υποθέτουμε ότι ο εκδότης κάποιου πιστοποιητικού έχει λάβει την υπογραφή του παραλήπτη για το αρχικό αίτημα του. Τώρα μπορεί να αποθηκεύσει το πιστοποιητικό στο blockchain.

Για να γίνει αυτό πρέπει προφανώς να έχει ήδη δημιουργήσει ένα έξυπνο συμβόλαιο στο blockchain. Αν αυτό δεν έχει γίνει το σύστημα δείχνει στον χρήστη μήνυμα προτροπής δημιουργίας συμβολαίου. Αν ο χρήστης διαθέτει ήδη κάποιο συμβόλαιο τότε αναδύεται το παράθυρο του MetaMask που δείχνει στον χρήστη την συναλλαγή που πρόκειται να εκτελεστεί. Αν αποδεχτεί και υπογράψει την συναλλαγή, τότε αυτή στέλνεται στο blockchain όπου επαληθεύεται και αν είναι έγκυρη το αποτύπωμα του πιστοποιητικού μαζί με την υπογραφή αποθηκεύονται στο έξυπνο συμβόλαιο. Το front-end λαμβάνει την απόδειξη της συναλλαγής και στην συνέχεια γίνεται κλήση στο API endpoint /user/upd_requests με το παρακάτω αντικείμενο:

```
{
        type : ‘ISS_UPL’,
        jwt : {json web token},
        data : {
                cert : {certificate},
                contractAddress : {contract address},
                req_id : { request id}
        }
}
```

Όπως βλέπουμε το αντικείμενο αυτό περιλαμβάνει το token, τον τύπο του αιτήματος ‘ISS_UPL’, την διεύθυνση του συμβολαίου, το πιστοποιητικό και το id του αιτήματος που αφορά αυτή η κλήση. Η μετάβαση του αιτήματος στην νέα αυτή κατάσταση δηλώνει ότι το πιστοποιητικό έχει αποθηκευτεί στο blockchain. Στην συνέχεια αφού γίνει ταυτοποίηση του χρήστη με επαλήθευση του token, το πιστοποιητικό που υπάρχει στην βάση ανανεώνεται με την διεύθυνση του συμβολαίου στο οποίο αποθηκεύτηκε. Ακόμα ανανεώνονται τα αντικείμενα Request-Response καθώς το αίτημα έχει μεταβεί σε νέα κατάσταση. Τέλος ανανεώνεται το αντικείμενο του αιτήματος στην γραφική διεπαφή του χρήστη.

### 4.7.4 Επαλήθευση πιστοποιητικού




**Εικόνα 27: Διάγραμμα επαλήθευσης πιστοποιητικού**

Για να επαληθεύσει ένα πιστοποιητικό ο χρήστης πρέπει να έχει την εικόνα/έγγραφο του πιστοποιητικού και την διεύθυνση του συμβολαίου στην οποία είναι αποθηκευμένο. Πλοηγείται στην σελίδα Validator συμπληρώνει με τα παραπάνω στοιχεία την φόρμα, κάνει κλικ στο κουμπί Validate και εμφανίζεται το αποτέλεσμα της επαλήθευσης στην οθόνη.

Για να γίνει η επαλήθευση το front-end υπολογίζει το hash του πιστοποιητικού και ζητάει από το blockchain την υπογραφή που αντιστοιχεί σε αυτό το hash και υπάρχει στο

συμβόλαιο που δόθηκε. Αν το συμβόλαιο αυτό υπάρχει και περιέχει το hash που ζητήθηκε τότε επιστρέφεται η υπογραφή, αλλιώς εμφανίζεται στον χρήστη μήνυμα σφάλματος. Στην συνέχεια γίνεται αίτημα που ζητάει την διεύθυνση του ιδιοκτήτη του συμβολαίου. Τέλος από το hash και την υπογραφή, υπολογίζεται το δημόσιο κλειδί δηλαδή η ταυτότητα του χρήστη που υπέγραψε το πιστοποιητικό.

Τα παρακάτω στοιχεία παρουσιάζονται στην γραφική διεπαφή:

- Διεύθυνση ιδιοκτήτη συμβολαίου
- Hash ή αποτύπωμα πιστοποιητικού
- Υπογραφή του αποτυπώματος
- Διεύθυνση του χρήστη που υπέγραψε το αποτύπωμα

Τι γνωρίζουμε με αυτά τα στοιχεία;

1) Το πιστοποιητικό που λάβαμε και υπολογίσαμε το αποτύπωμά του είναι γνήσιο. Αν το πιστοποιητικό είχε παραποιηθεί τότε το αποτύπωμα που θα υπολογίζαμε θα ήταν διαφορετικό με αυτό που υπάρχει στο συμβόλαιο και αντιστοιχεί στο αρχικό πιστοποιητικό. Επομένως έτσι εντοπίζουμε αμέσως πιστοποιητικά που δεν είναι γνήσια.
2) Την διεύθυνση του εκδότη του πιστοποιητικού. Καθώς το αποτύπωμα του πιστοποιητικού είναι αποθηκευμένο στο συγκεκριμένο συμβόλαιο, ο ιδιοκτήτης του θα είναι και ο εκδότης του πιστοποιητικού.
3) Την διεύθυνση του χρήστη που υπέγραψε το αποτύπωμα του πιστοποιητικού δηλαδή την διεύθυνση του ιδιοκτήτη του.

Μπορούμε εύκολα να ελέγξουμε αν ο χρήστης που μας έστειλε το πιστοποιητικό είναι όντως ο ιδιοκτήτης του βάζοντας τον να υπογράψει κάποιο μήνυμα με το ιδιωτικό του κλειδί. Στην συνέχεια από το μήνυμα και την υπογραφή υπολογίζουμε το δημόσιο κλειδί του και αν αυτό ταυτίζεται με το κλειδί που υπέγραψε το αποτύπωμα του πιστοποιητικού, γνωρίζουμε ότι ο χρήστης είναι ο πραγματικός ιδιοκτήτης του.

Έτσι ακόμα και αν κάποιος κακόβουλος χρήστης έχει στα χέρια του πιστοποιητικό άλλου και την διεύθυνση του έξυπνου συμβολαίου στην οποία είναι αποθηκευμένο, δεν μπορεί να παριστάνει ότι είναι ο ιδιοκτήτης του.

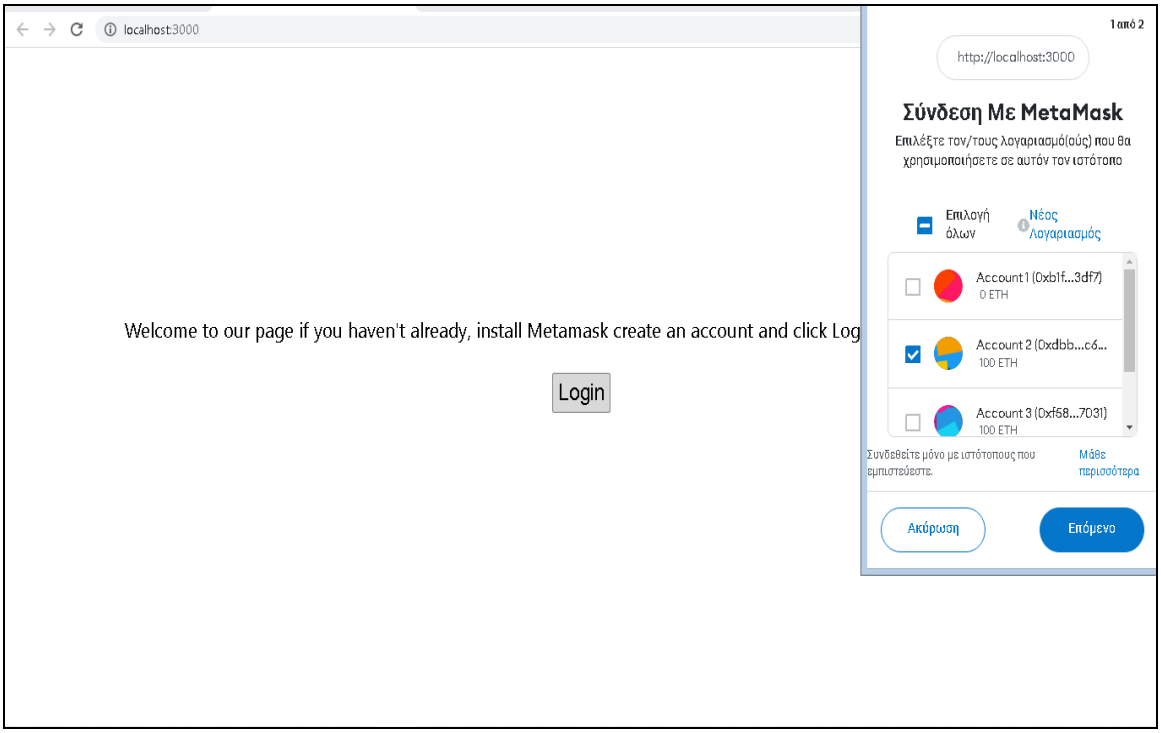
Ωστόσο αν κάποιος κακόβουλος χρήστης αποκτήσει το ιδιωτικό κλειδί άλλου, μπορεί να εκτελέσει οποιαδήποτε κακόβουλη ενέργεια στο σύστημα εκ μέρους του. Επομένως είναι πολύ σημαντικό οι χρήστες να προστατεύουν τα ιδιωτικά τους κλειδιά.

# 5 Παρουσίαση λειτουργιών εφαρμογής με στιγμιότυπα

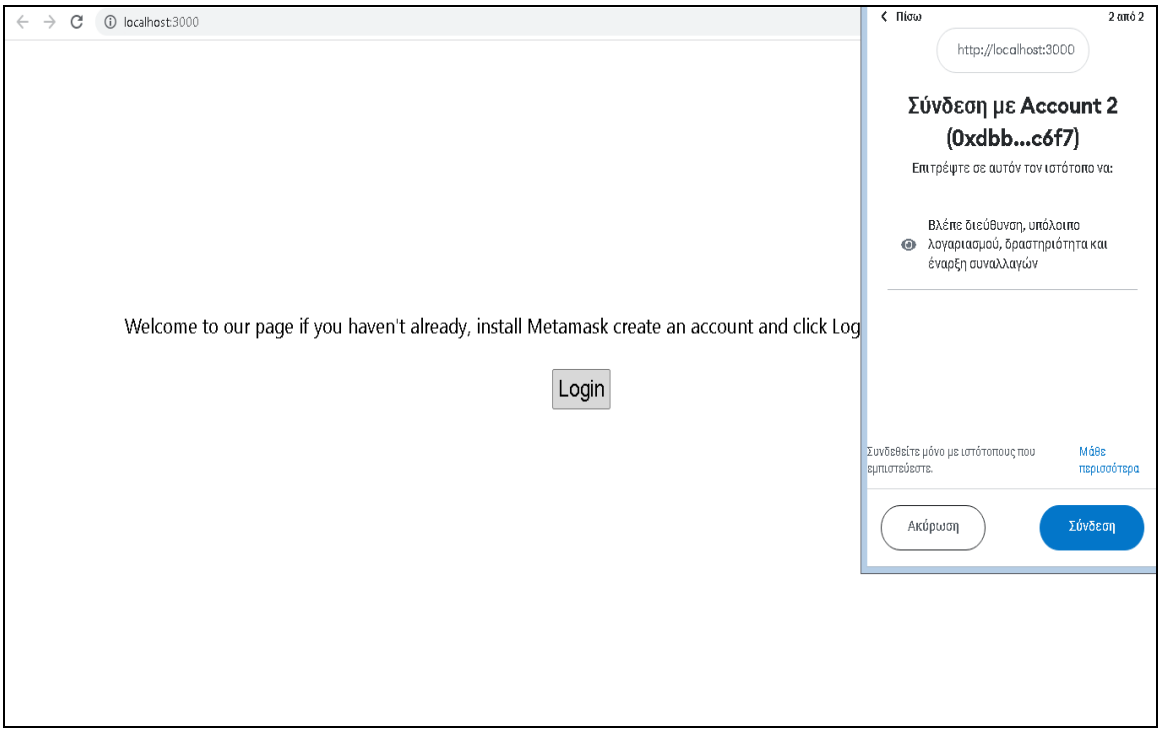
Σε αυτή την ενότητα θα γίνει επίδειξη της εφαρμογής με στιγμιότυπα από την εκτέλεση των λειτουργιών της. Θα γίνει σύνδεση στην εφαρμογή με δύο διαφορετικούς λογαριασμούς χρήστη, τον Λογαριασμό 2 και τον Λογαριασμό 3 οι οποίοι θα αλληλεπιδρούν δημιουργώντας συμβόλαια, εκδίδοντας πιστοποιητικά ο ένας στον άλλο και ελέγχοντάς τα.

## 5.1 Σύνδεση με Λογαριασμό 2

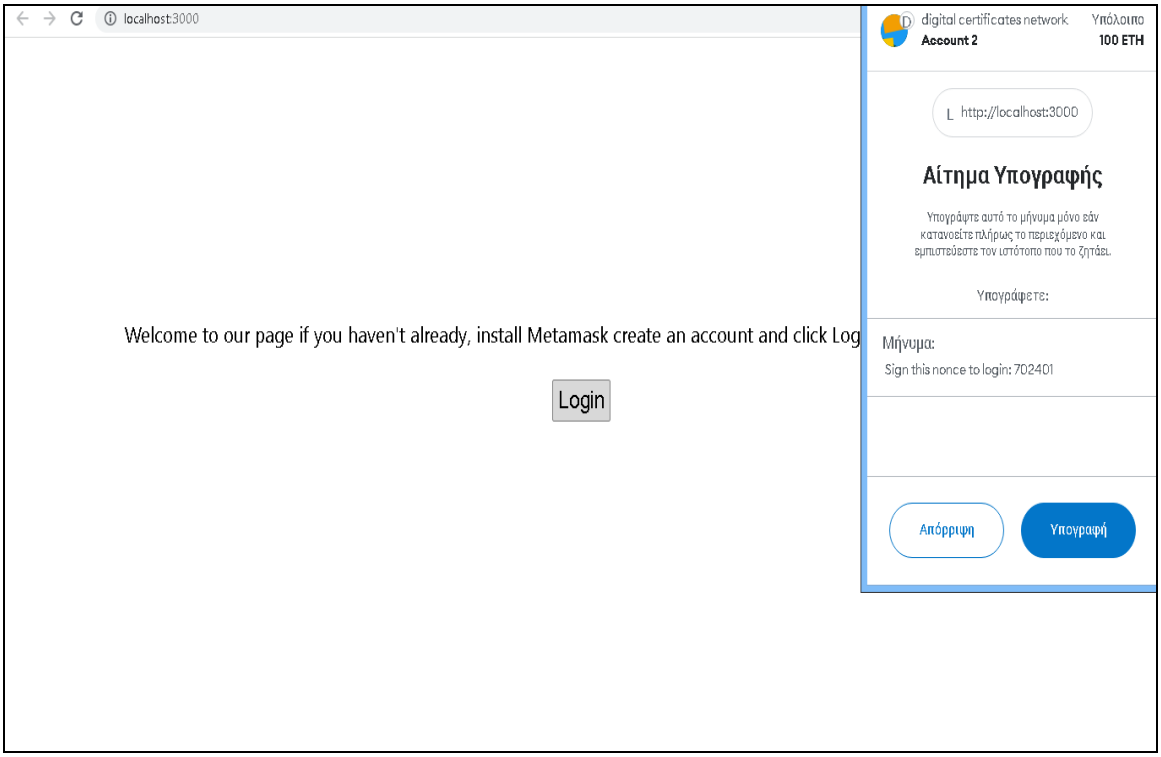
Μόλις πατήσουμε το κουμπί Login αναδύεται το παράθυρο του MetaMask. Επιλέγουμε τον Λογαριασμό 2 και πατάμε Επόμενο.



Στην συνέχεια πατάμε Σύνδεση δίνοντας στην εφαρμογή πρόσβαση στον Λογαριασμό 2.

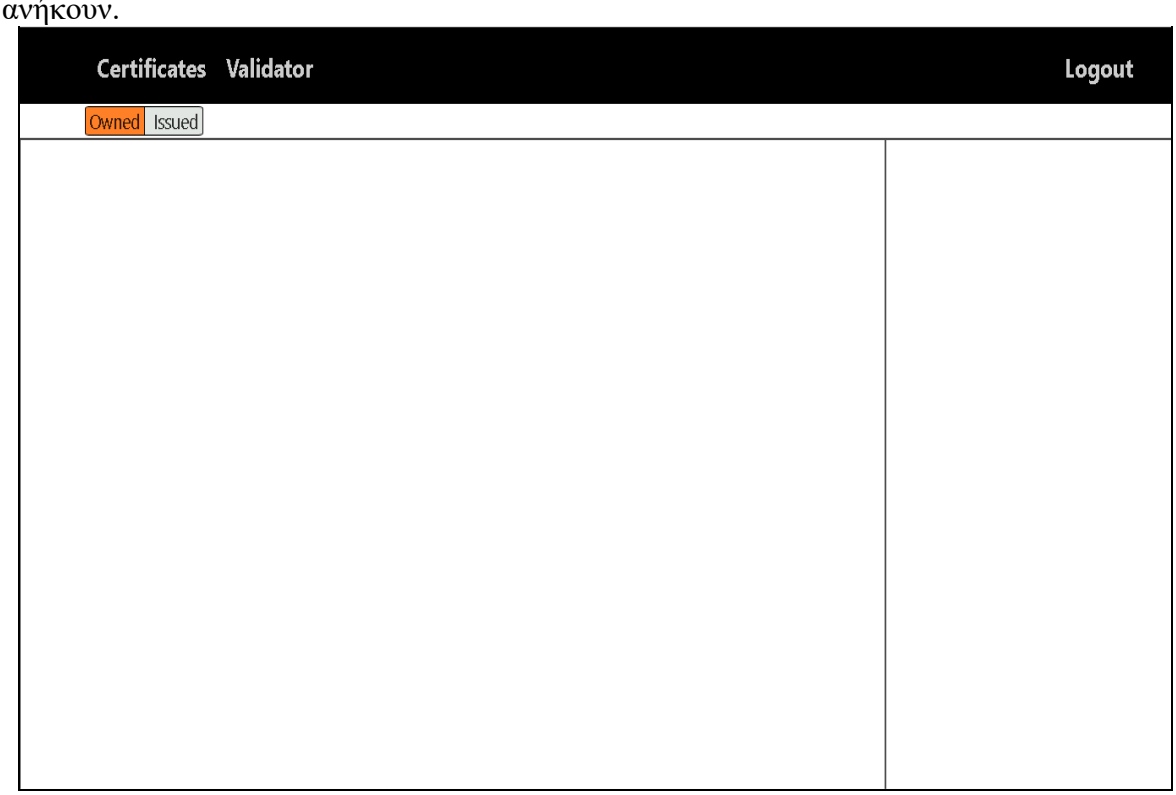



Έρχεται από το back-end το μήνυμα σύνδεσης που περιλαμβάνει το nonce και πρέπει να το υπογράψουμε, επομένως πατάμε Υπογραφή.

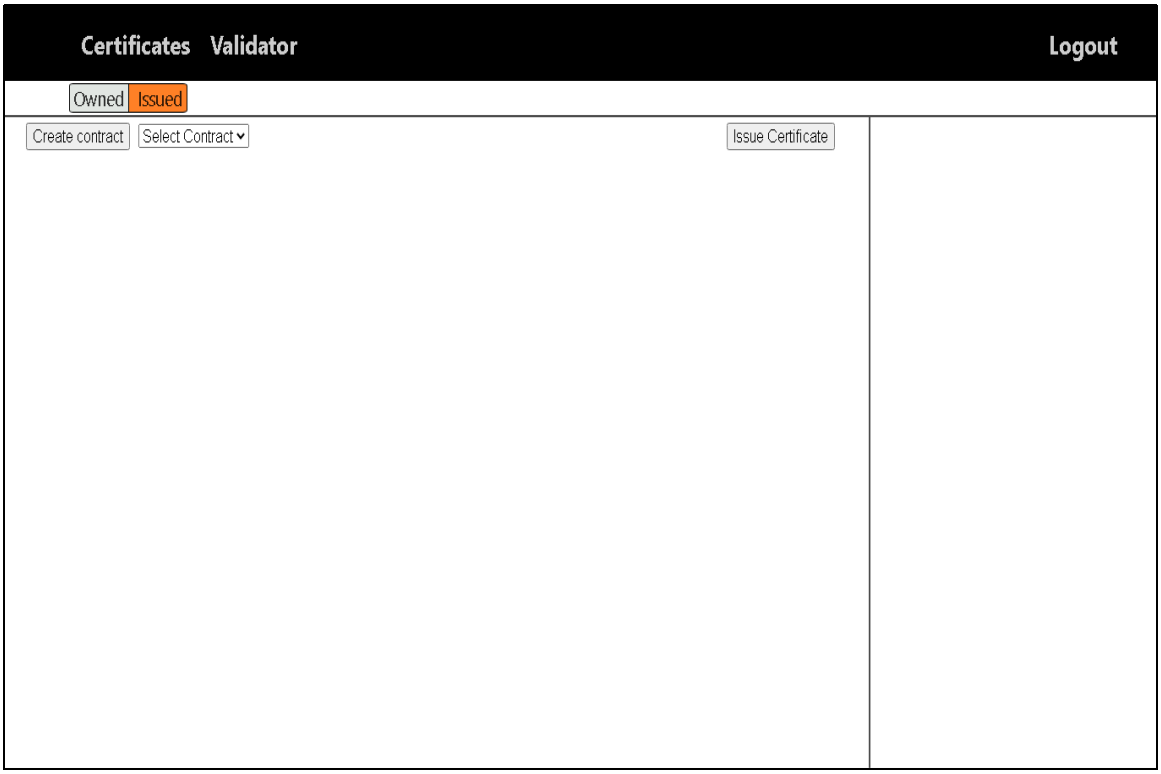


## 5.2 Πλοήγηση στις σελίδες της εφαρμογής

Πλέον είμαστε συνδεδεμένοι και βλέπουμε την αρχική σελίδα της εφαρμογής, βρισκόμαστε δηλαδή στην σελίδα Owned certificates. Εδώ μπορούμε να δούμε τα πιστοποιητικά που μας ανήκουν.



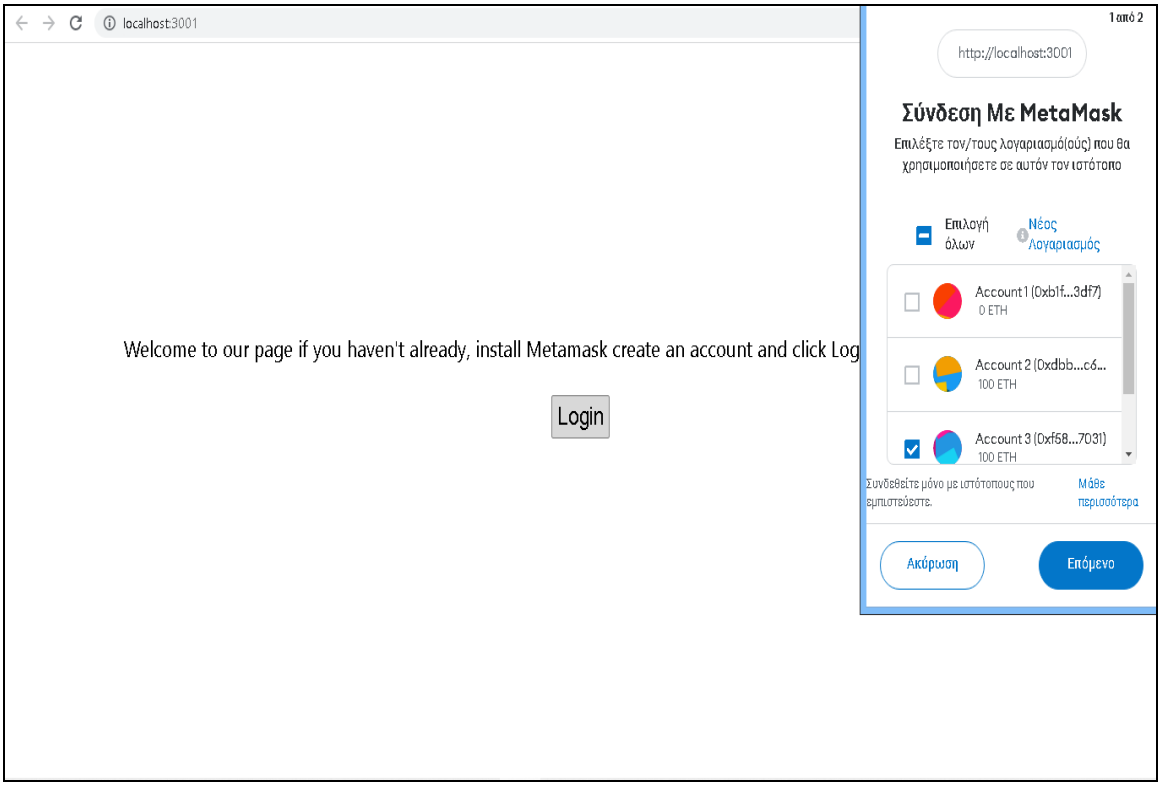

Στην συνέχεια πλοηγούμαστε στην σελίδα Issued certificates πατώντας το κουμπί Issued. Εδώ μπορούμε να δούμε τα πιστοποιητικά που έχουμε εκδώσει για άλλους.



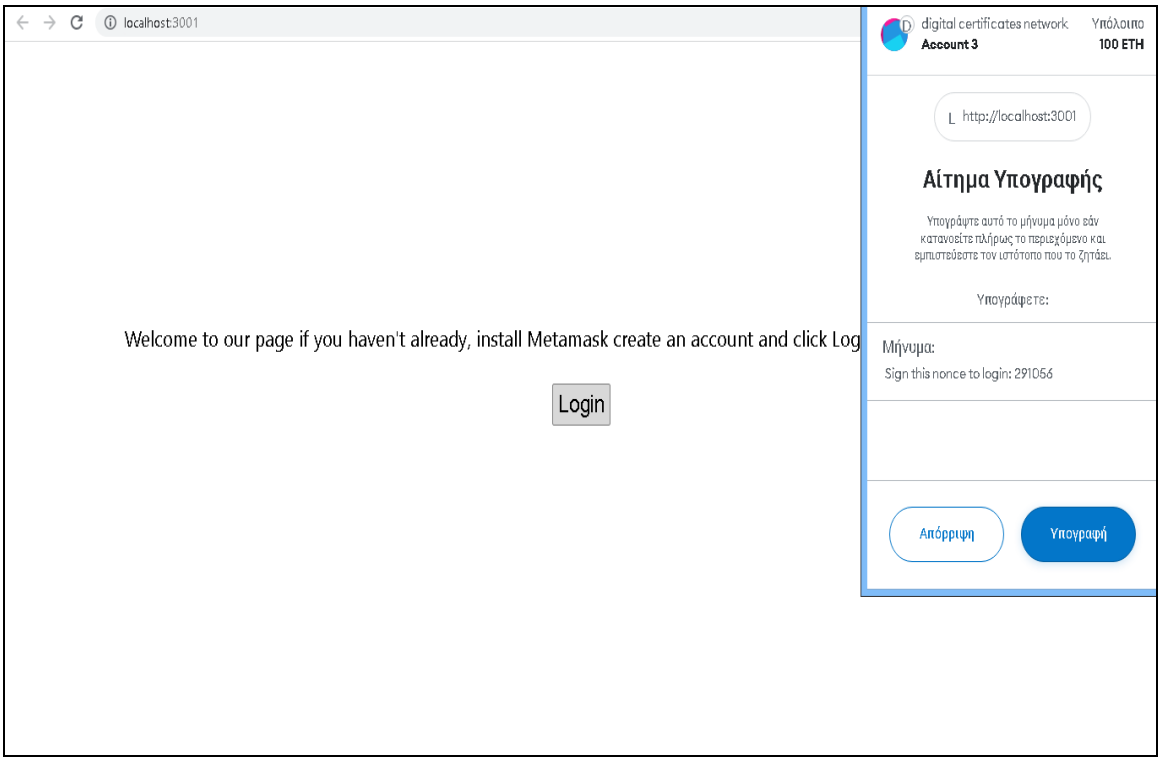
Έπειτα πατάμε την επιλογή Validator του αρχικού μενού και πλοηγούμαστε στην σελίδα Validator όπου γίνεται ο έλεγχος των πιστοποιητικών.


## 5.3 Σύνδεση με Λογαριασμό 3

Όπως και προηγουμένως συνδεόμαστε στην εφαρμογή αυτή την φορά όμως με τον Λογαριασμό 3.



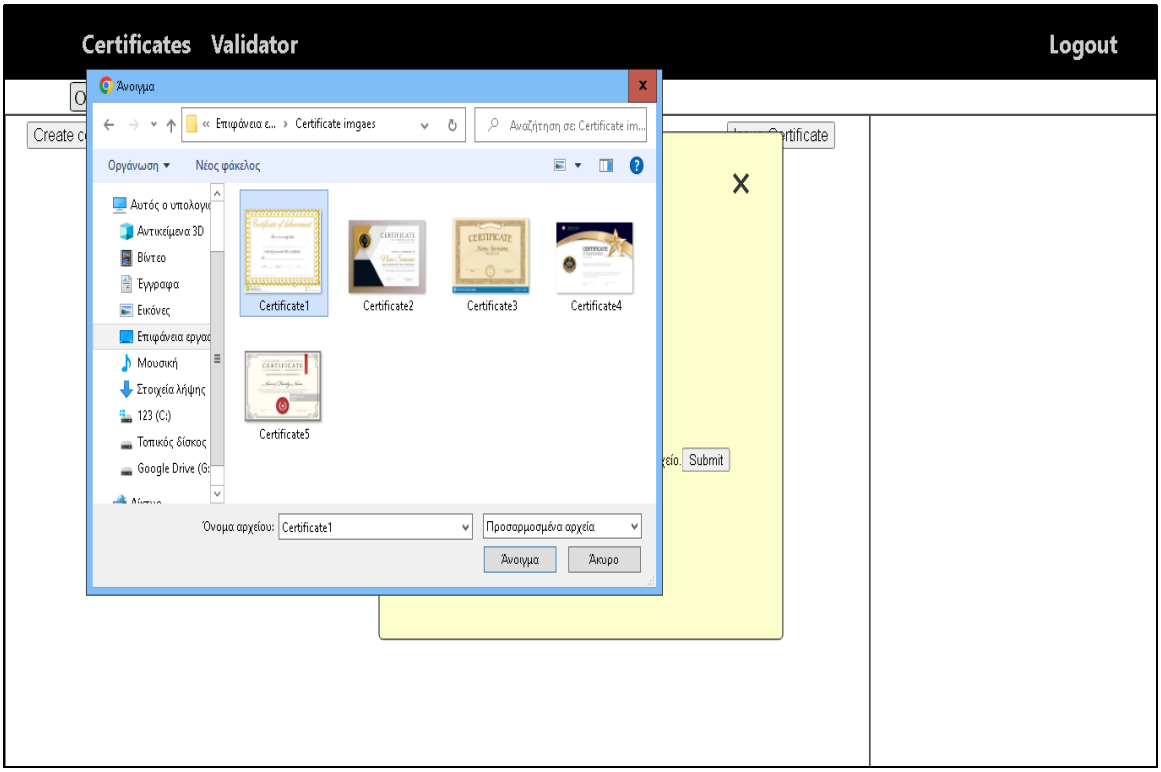
Βλέπουμε ότι πάλι μας επιστρέφεται μήνυμα προς υπογραφή το οποίο υπογράφουμε και συνδεόμαστε.



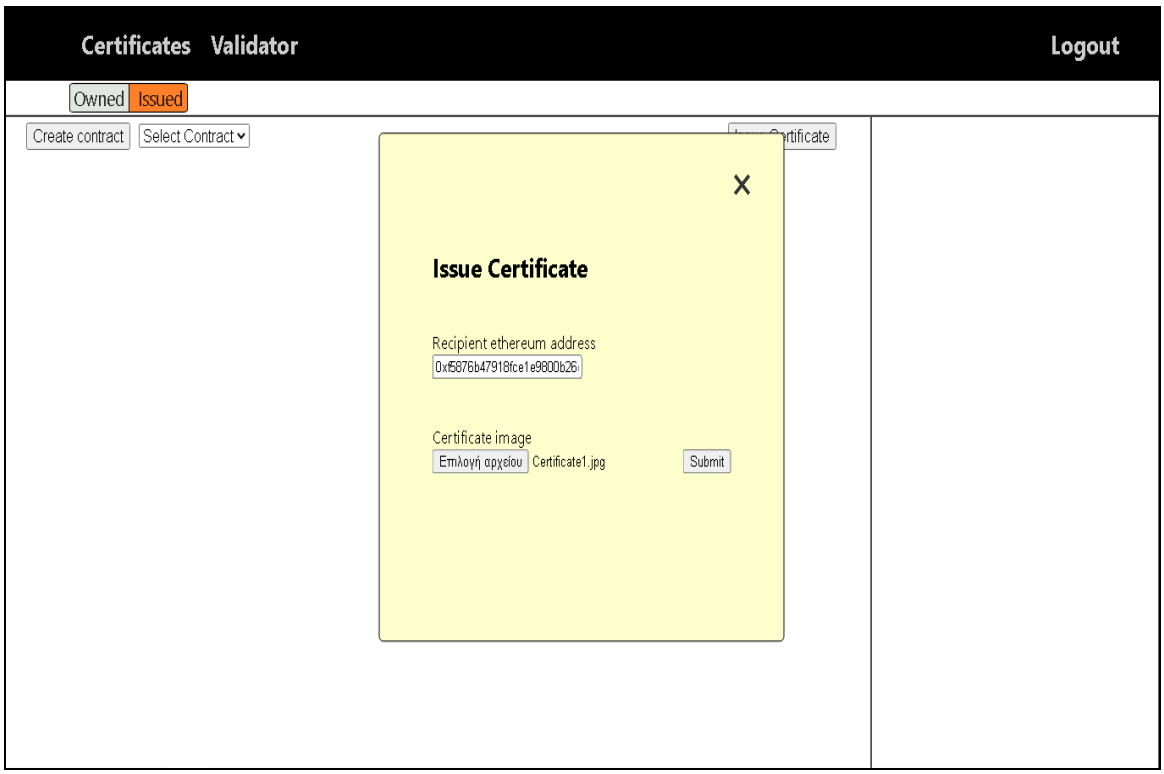

## 5.4 Έκδοση πιστοποιητικών από Λογαριασμό 2 σε Λογαριασμό 3

Πατάμε το κουμπί Issue Certificate και βλέπουμε την φόρμα που πρέπει να συμπληρώσουμε.

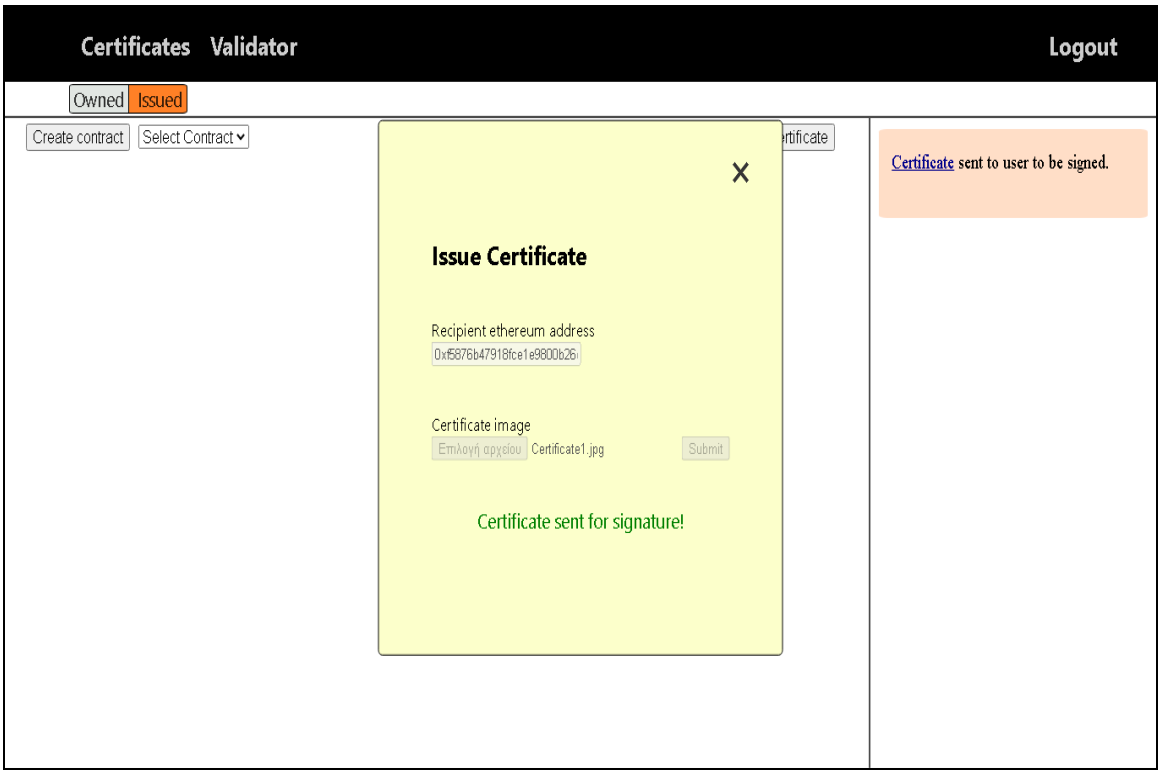
Επιλέγουμε το πρώτο πιστοποιητικό.



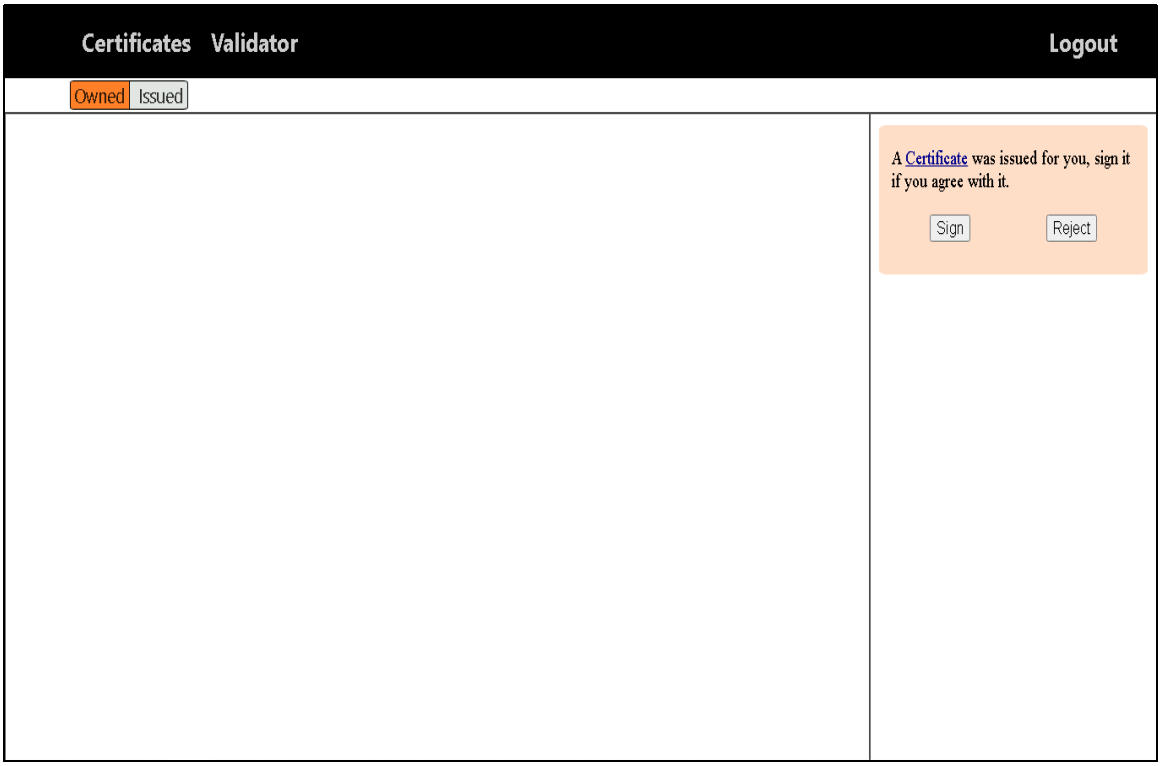

Συμπληρώνουμε την διεύθυνση του παραλήπτη του πιστοποιητικού, την διεύθυνση δηλαδή του Λογαριασμού 3.



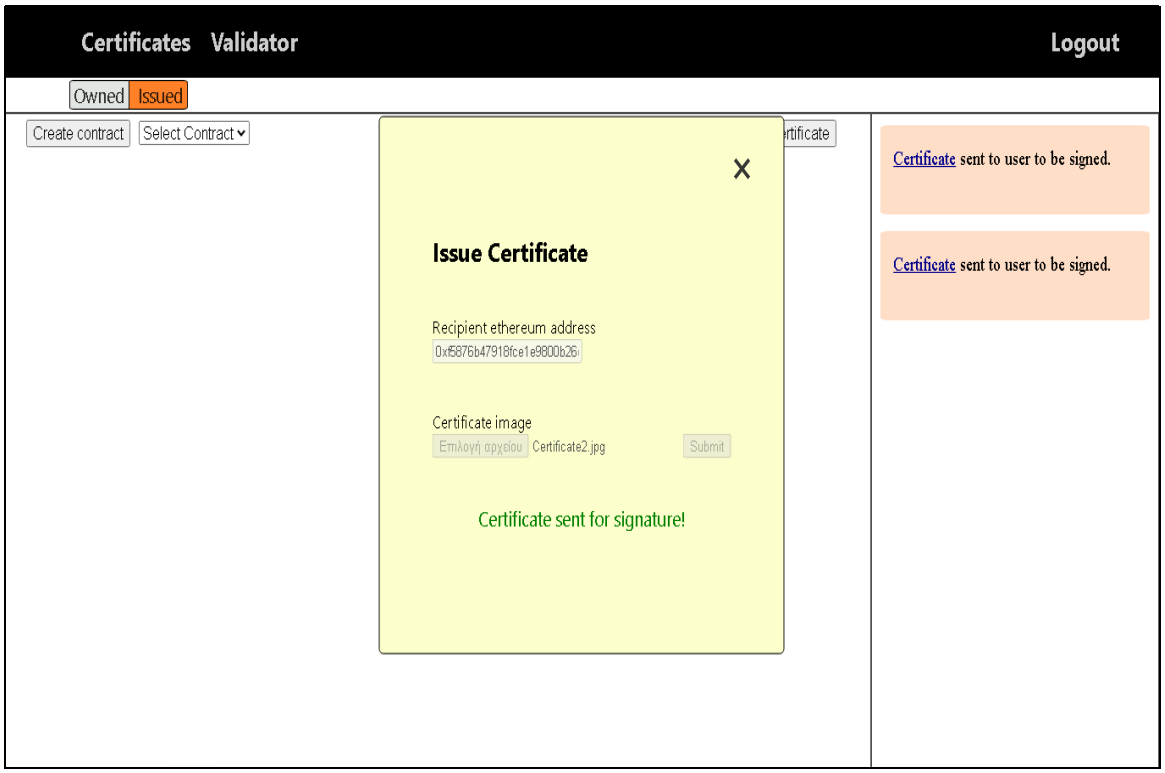
Πατάμε Submit και βλέπουμε ότι το αίτημα έκδοσης έχει σταλεί επιτυχώς στον παραλήπτη προς υπογραφή. Επίσης στο δεξί τμήμα της εικόνας βλέπουμε το αντικείμενο που αντιπροσωπεύει αυτό το αίτημα και θα αλλάζει καθώς το αίτημα θα μεταβαίνει από μία κατάσταση στην επόμενη.



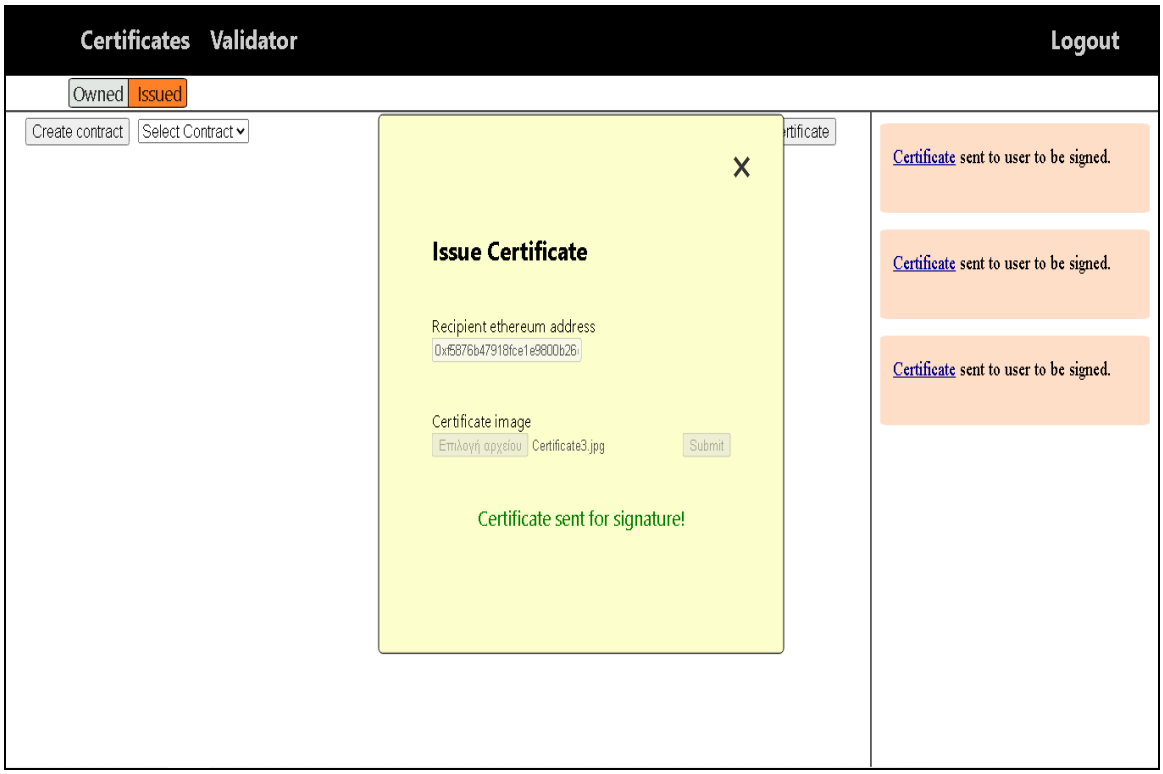

Βλέπουμε ότι το αίτημα έχει εμφανιστεί στην γραφική διεπαφή του Λογαριασμού 3 και δίνονται στον χρήστη οι επιλογές υπογραφής ή απόρριψης.



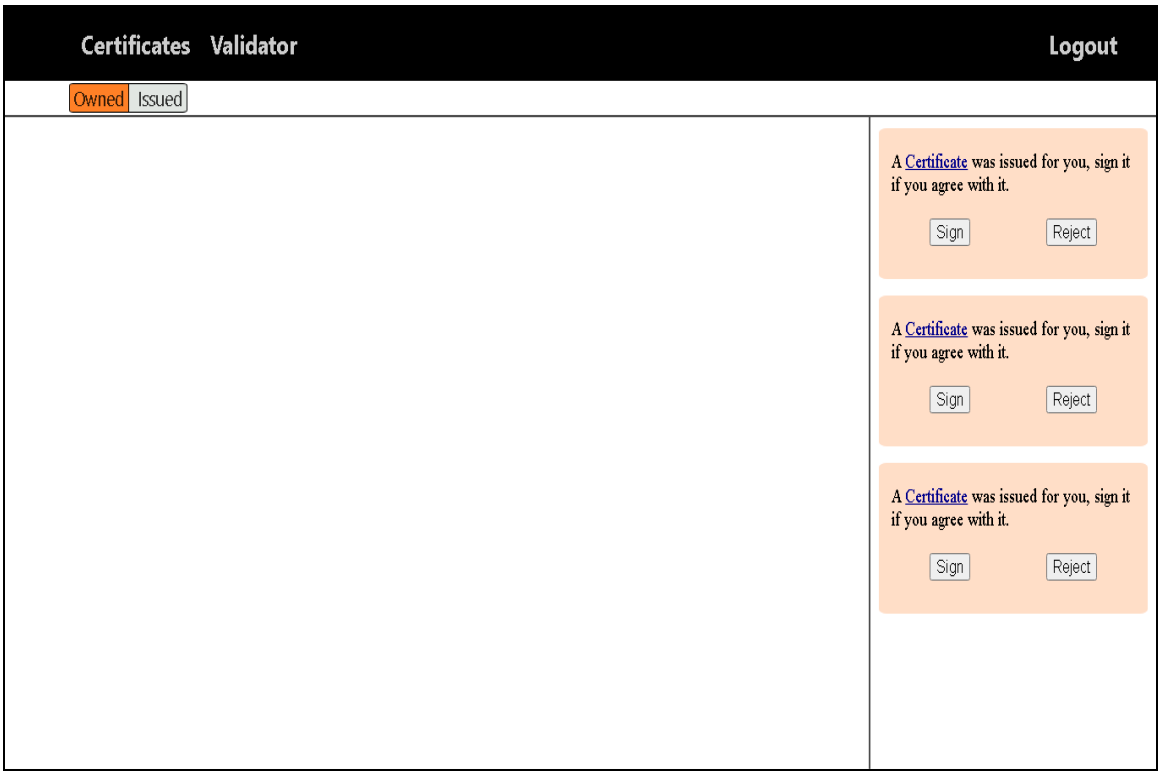
Έκδοση δεύτερου πιστοποιητικού, βλέπουμε πως προστίθεται ένα ακόμα αίτημα στην γραφική διεπαφή.




Έκδοση τρίτου πιστοποιητικού, προστίθεται και το τρίτο αίτημα στην γραφική διεπαφή.

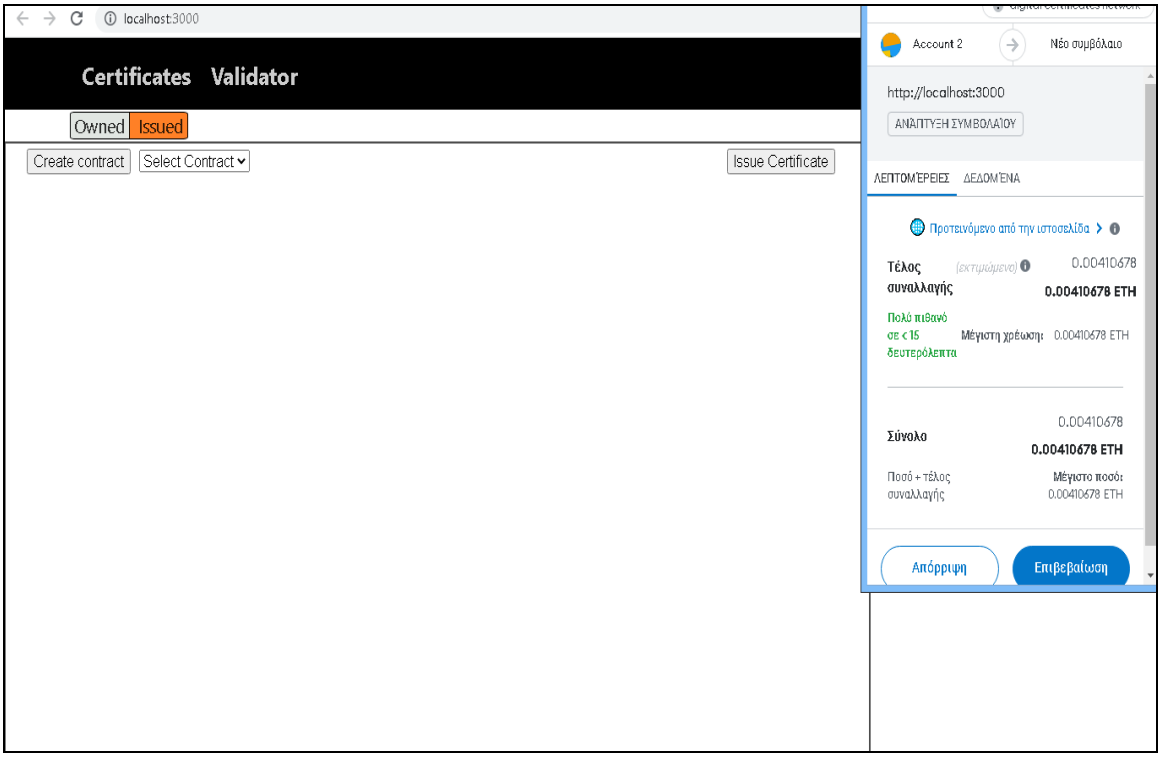


Βλέπουμε την γραφική διεπαφή του Λογαριασμού 3 μετά την έκδοση προς αυτόν τριών πιστοποιητικών.

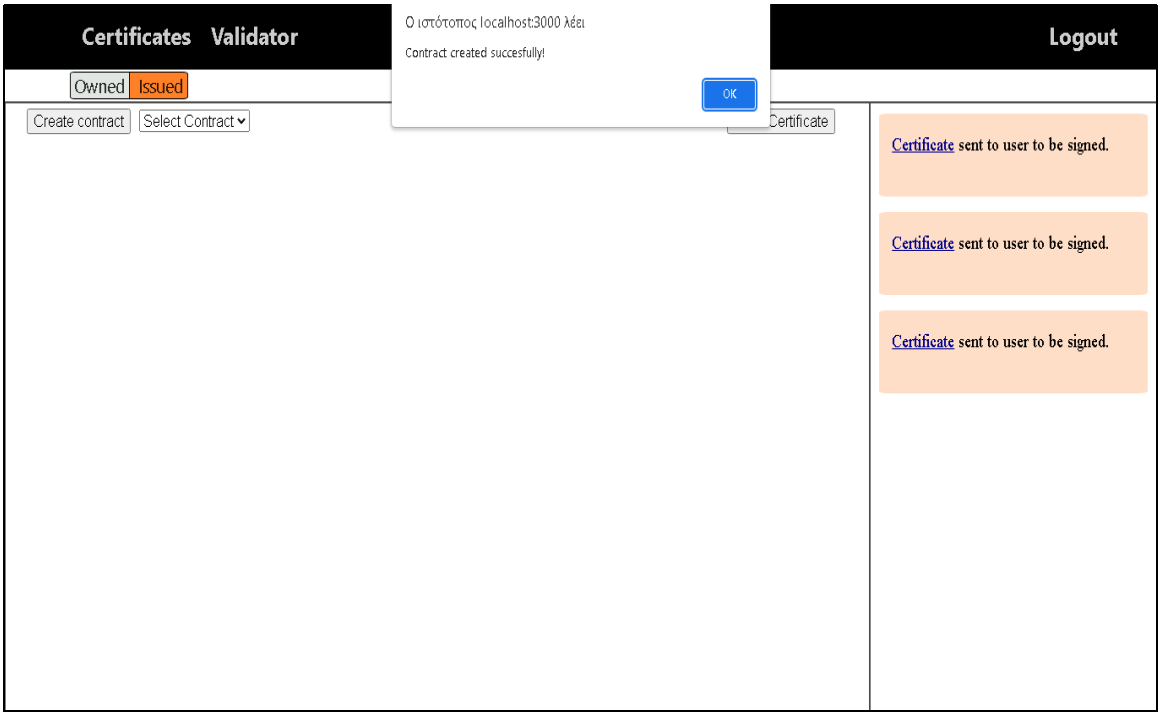



## 5.5 Δημιουργία συμβολαίου από τον Λογαριασμό 2

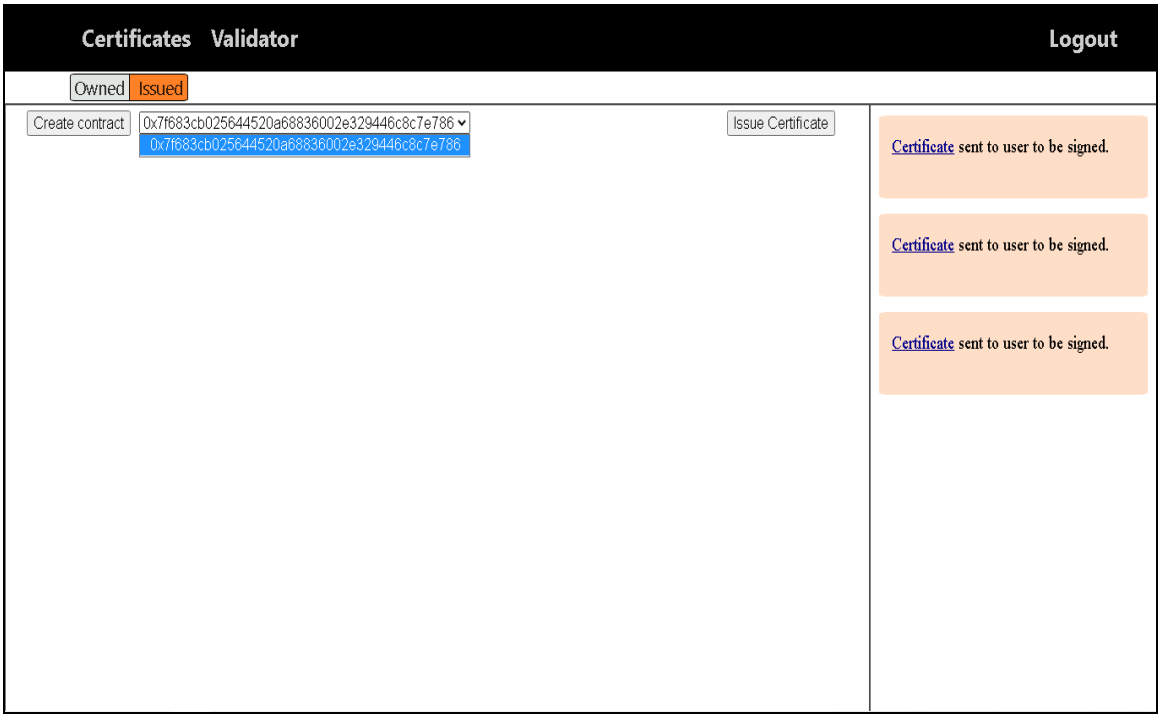
Ο χρήστης πατάει Create contract και αναδύεται το παράθυρο του MetaMask δείχνοντας την συναλλαγή ανάπτυξης συμβολαίου και ζητώντας επιβεβαίωση.



Πατάμε επιβεβαίωση και βλέπουμε το μήνυμα επιτυχούς δημιουργίας του συμβολαίου.

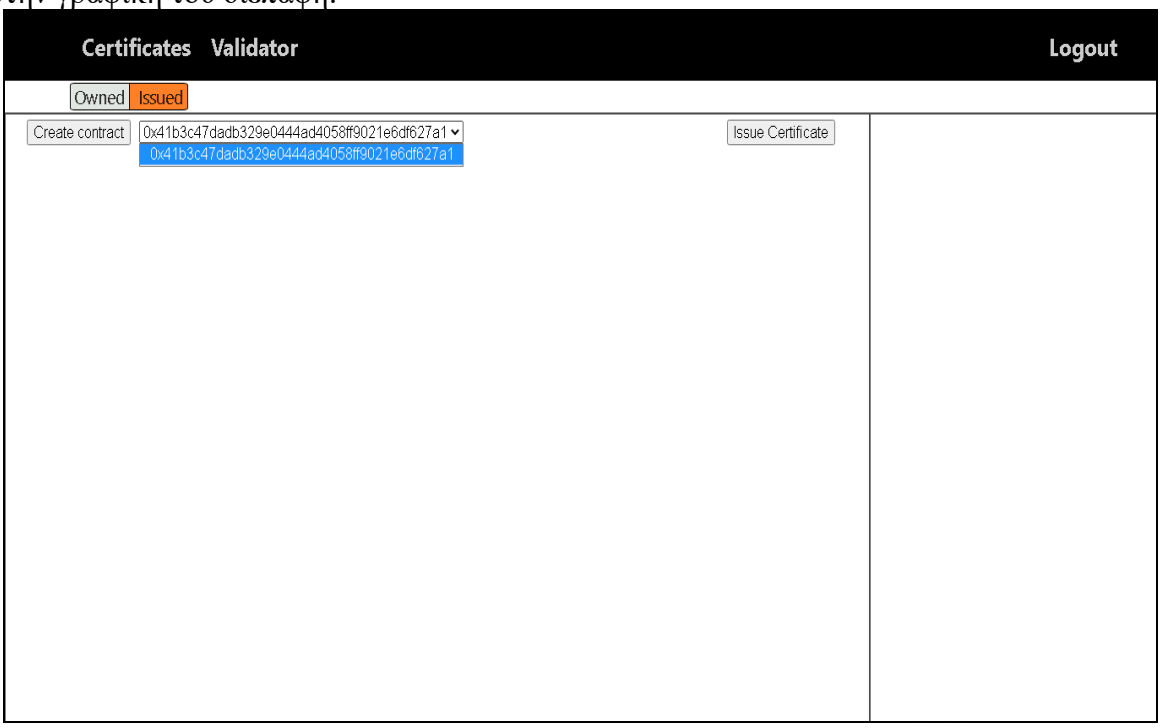



Βλέπουμε στην γραφική διεπαφή μας την διεύθυνση του συμβολαίου που μόλις δημιουργήσαμε.



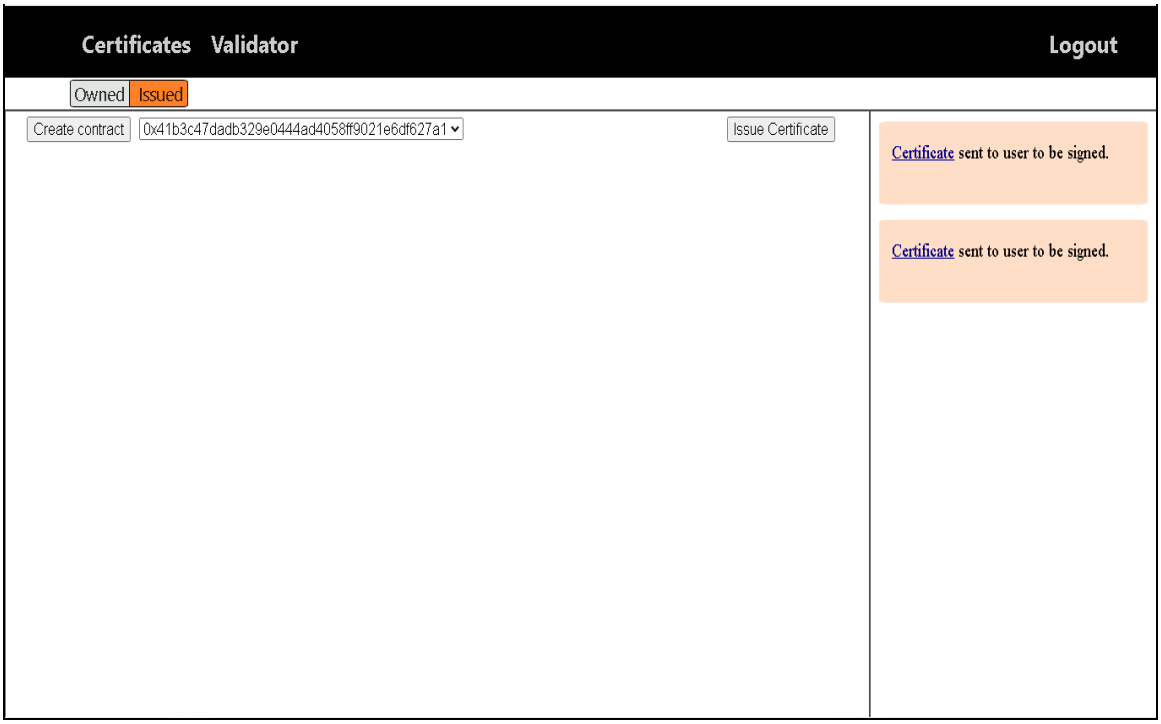
## 5.6 Δημιουργία συμβολαίου από τον Λογαριασμό 3

Όμοια με προηγουμένως ο Λογαριασμός 3 δημιουργεί ένα συμβόλαιο το οποίο βλέπουμε στην γραφική του διεπαφή.



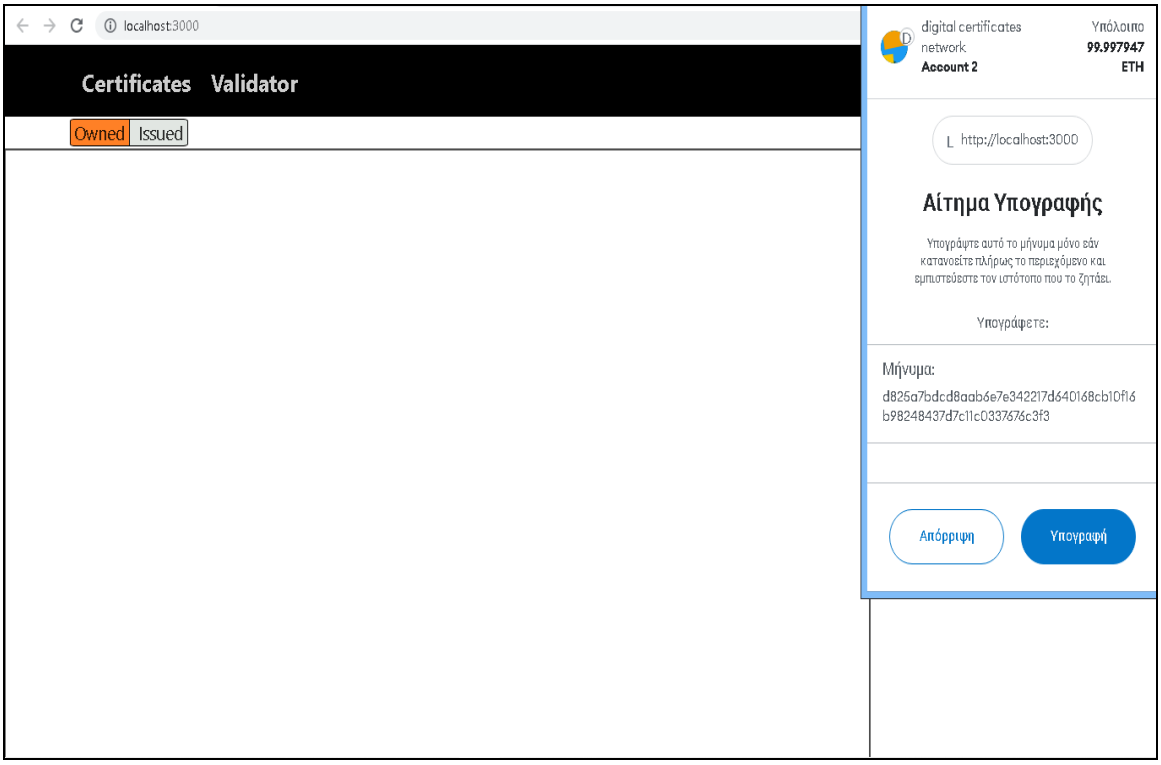

## 5.7 Έκδοση πιστοποιητικών από Λογαριασμό 3 σε Λογαριασμό 2

Όμοια με προηγουμένως, ο Λογαριασμός 3 εκδίδει το τέταρτο και το πέμπτο πιστοποιητικό στον Λογαριασμό 2 και βλέπουμε τα δύο αντικείμενα που εμφανίζονται στην γραφική του διεπαφή.

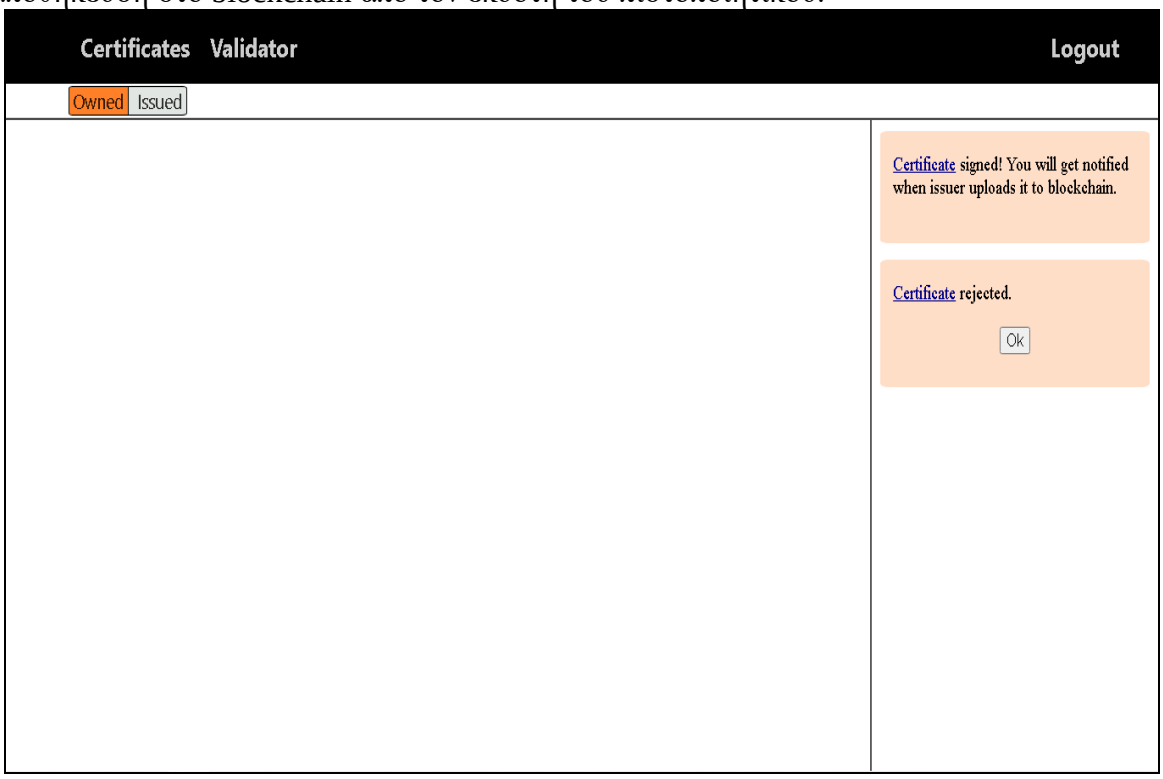


## 5.8 Υπογραφή πιστοποιητικού από τον Λογαριασμό 2

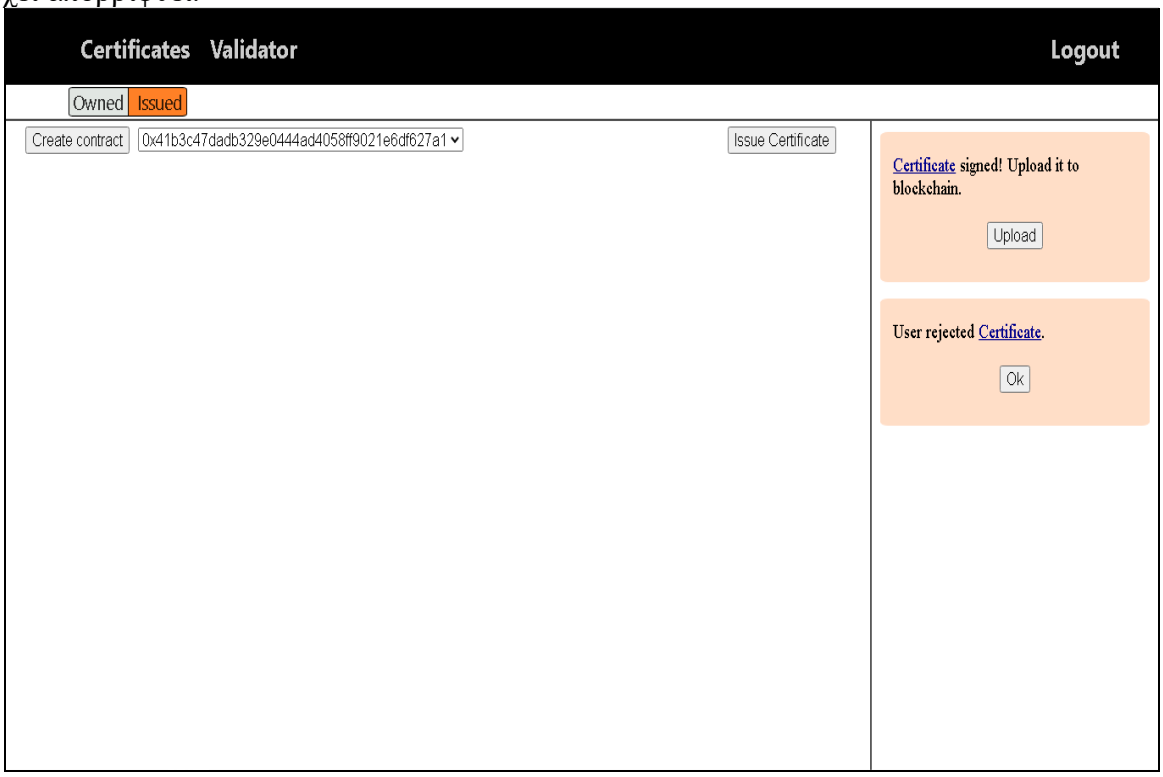
Ο Λογαριασμός 2 υπογράφει το τέταρτο πιστοποιητικό και απορρίπτει το πέμπτο. Βλέπουμε το μήνυμα προς υπογραφή που περιέχει το αποτύπωμα του πιστοποιητικού.




Βλέπουμε πως ανανεώνονται τα αιτήματα του Λογαριασμού 2 στην γραφική διεπαφή του. Το αίτημα για το πιστοποιητικό που υπέγραψε του δείχνει πως τώρα αναμένεται η αποθήκευση στο blockchain από τον εκδότη του πιστοποιητικού.

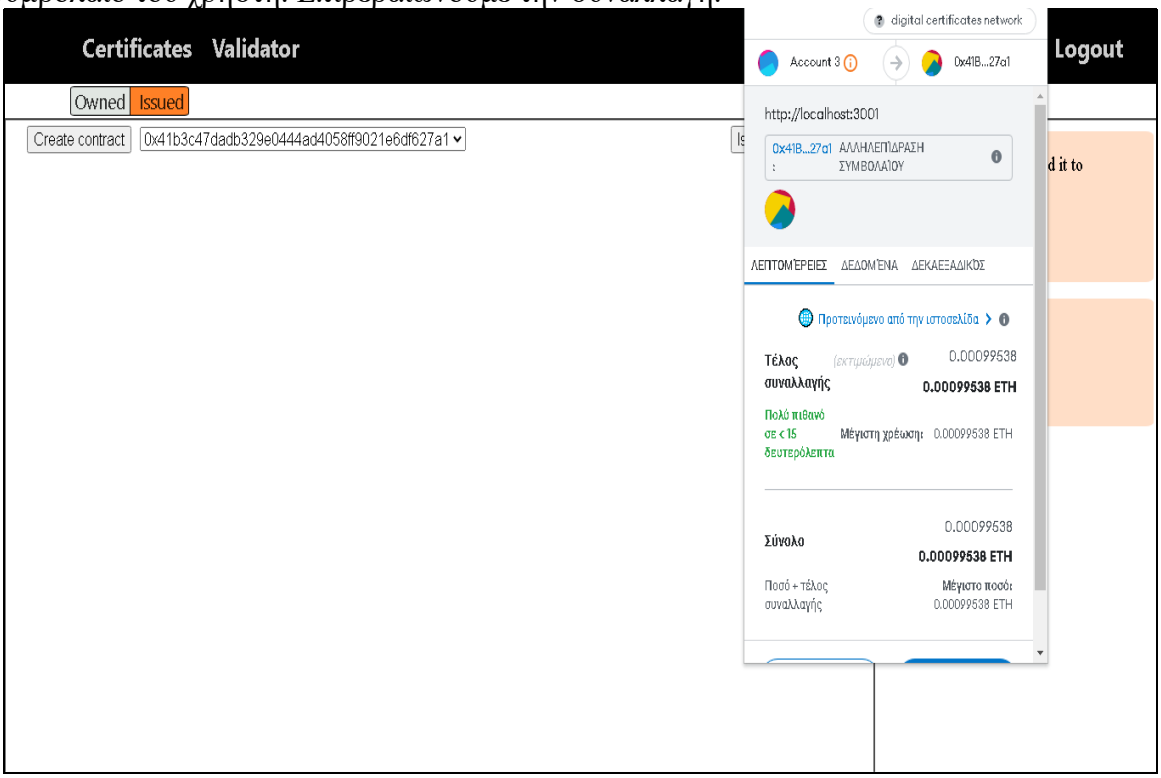


Παρακάτω φαίνεται η ανανεωμένη γραφική διεπαφή του Λογαριασμού 3. Βλέπουμε ότι το πιστοποιητικό που έχει υπογραφεί είναι έτοιμο για αποθήκευση στο blockchain ενώ το άλλο έχει απορριφθεί.

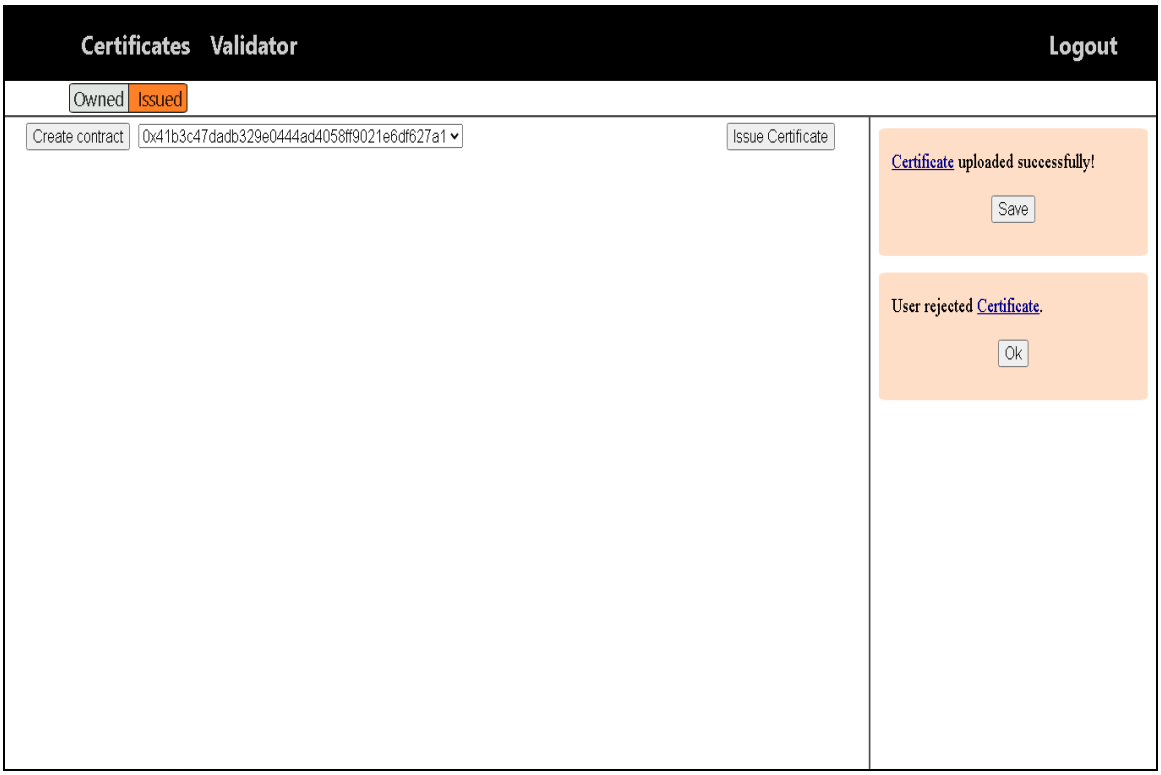



## 5.9 Αποθήκευση του πιστοποιητικού στο blockchain από τον Λογαριασμό 3

Ο Λογαριασμός 3 πατάει Upload για να αποθηκεύσει το υπογεγραμμένο πιστοποιητικό και βλέπουμε το παράθυρο του MetaMask που δείχνει την συναλλαγή που αλληλεπιδρά με το συμβόλαιο του χρήστη. Επιβεβαιώνουμε την συναλλαγή.



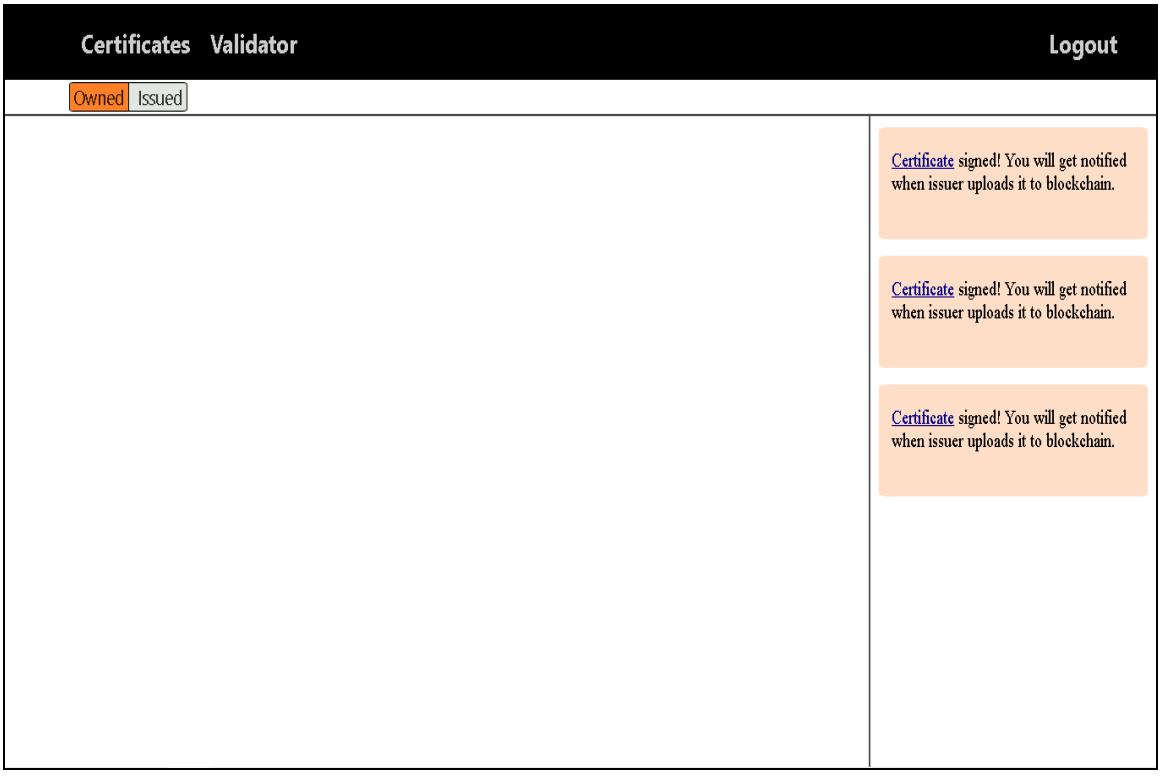
Το πιστοποιητικό αποθηκεύεται στο blockchain επιτυχώς όπως φαίνεται και από το πρώτο αίτημα της γραφικής διεπαφής. Τώρα πατάμε Save στο πρώτο αίτημα και Ok στο δεύτερο.



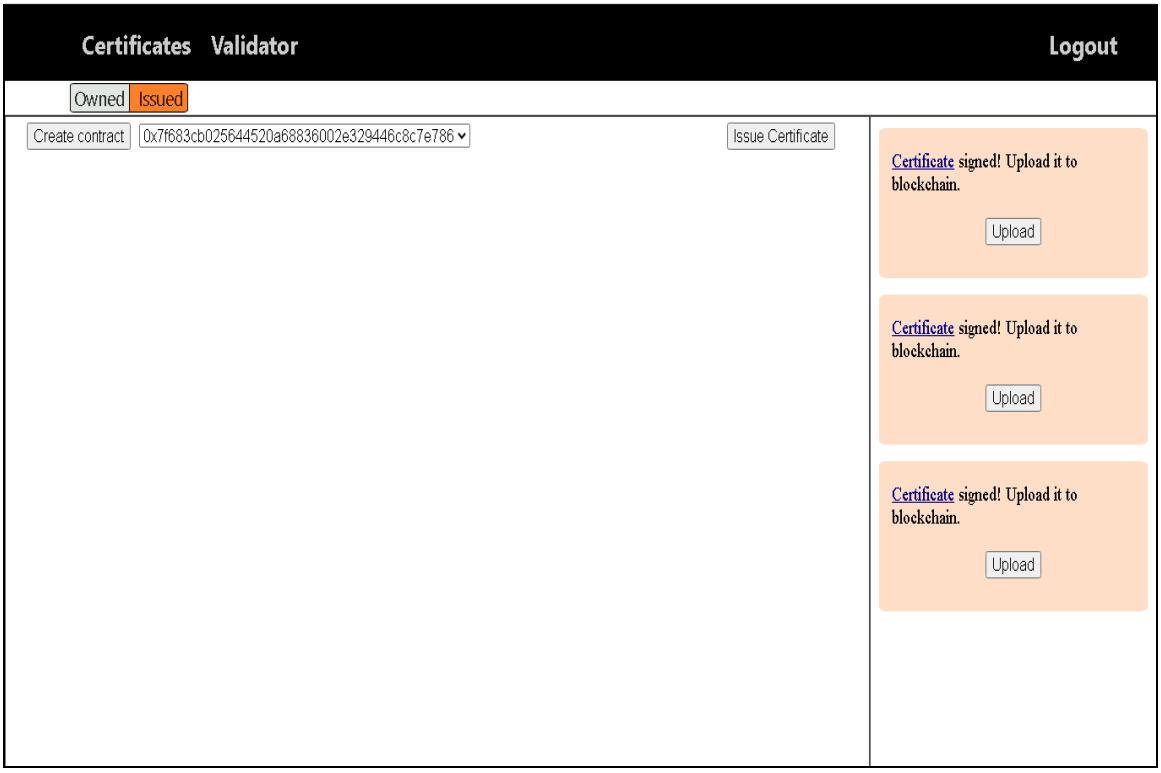

Βλέπουμε την ανανεωμένη γραφική διεπαφή του Λογαριασμού 3 όπου φαίνεται και το πιστοποιητικό που μόλις αποθηκεύτηκε στο blockchain.

Βλέπουμε το πιστοποιητικό και στην γραφική διεπαφή του Λογαριασμού 2 αλλά αυτή τη φορά στην καρτέλα Owned certificates καθώς το πιστοποιητικό αυτό του ανήκει.


## 5.10 Υπογραφή όλων των πιστοποιητικών από τον Λογαριασμό 3

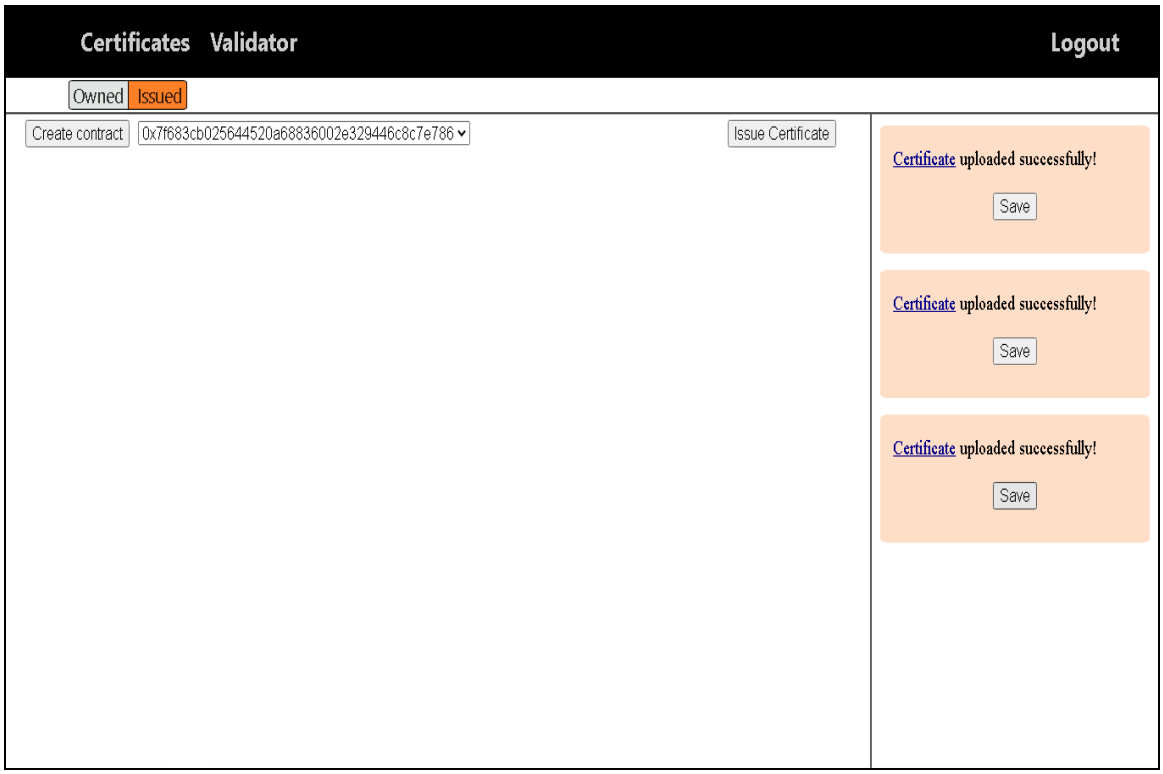


Εικόνα της γραφικής διεπαφής του Λογαριασμού 2 με τα τρία πιστοποιητικά που μόλις υπογράφηκαν έτοιμα να αποθηκευτούν στο blockchain.

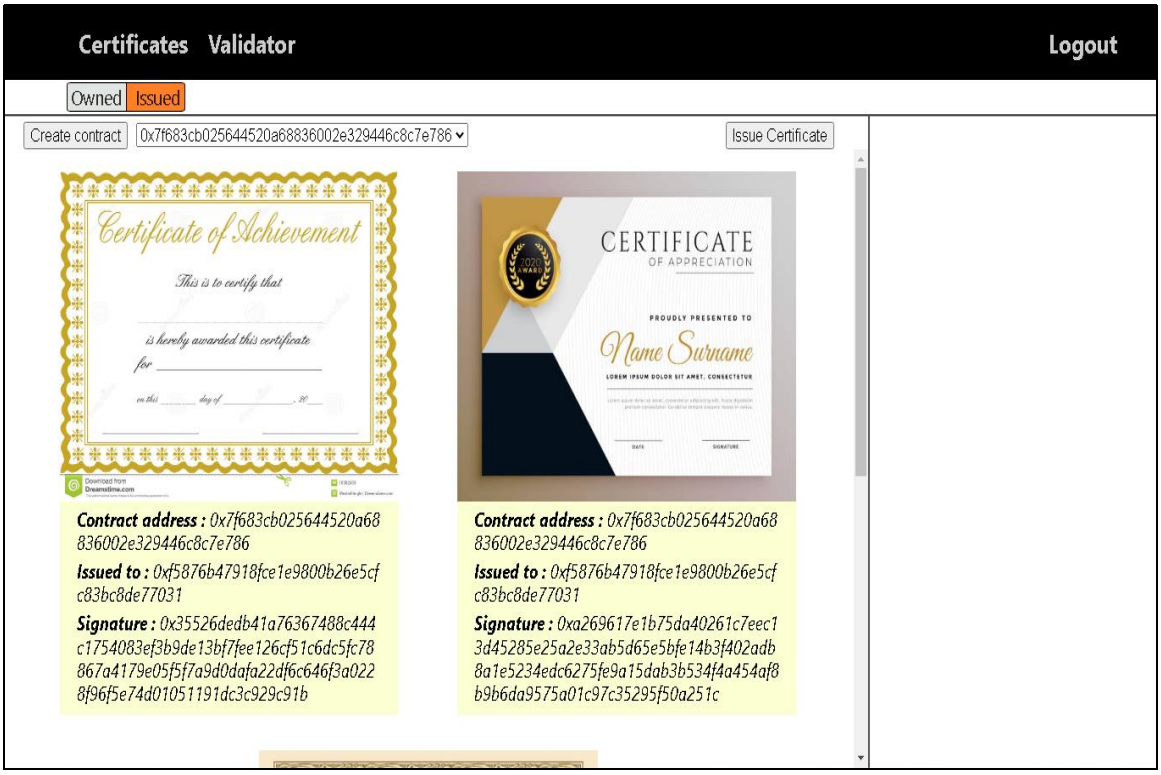


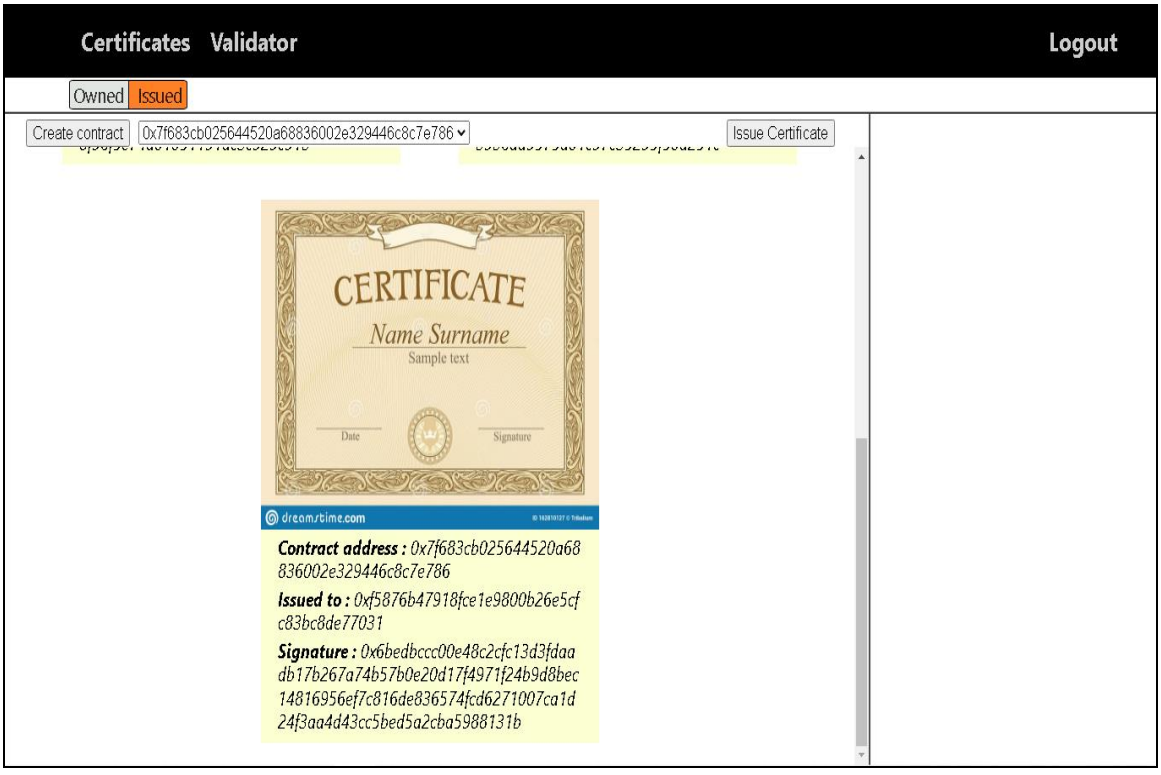

## 5.11 Αποθήκευση όλων των υπογεγραμμένων πιστοποιητικών στο blockchain από τον Λογαριασμό 2

Ο Λογαριασμός 2 αποθηκεύει όλα τα πιστοποιητικά στο blockchain και στην συνέχεια πατάει σε όλα Save.

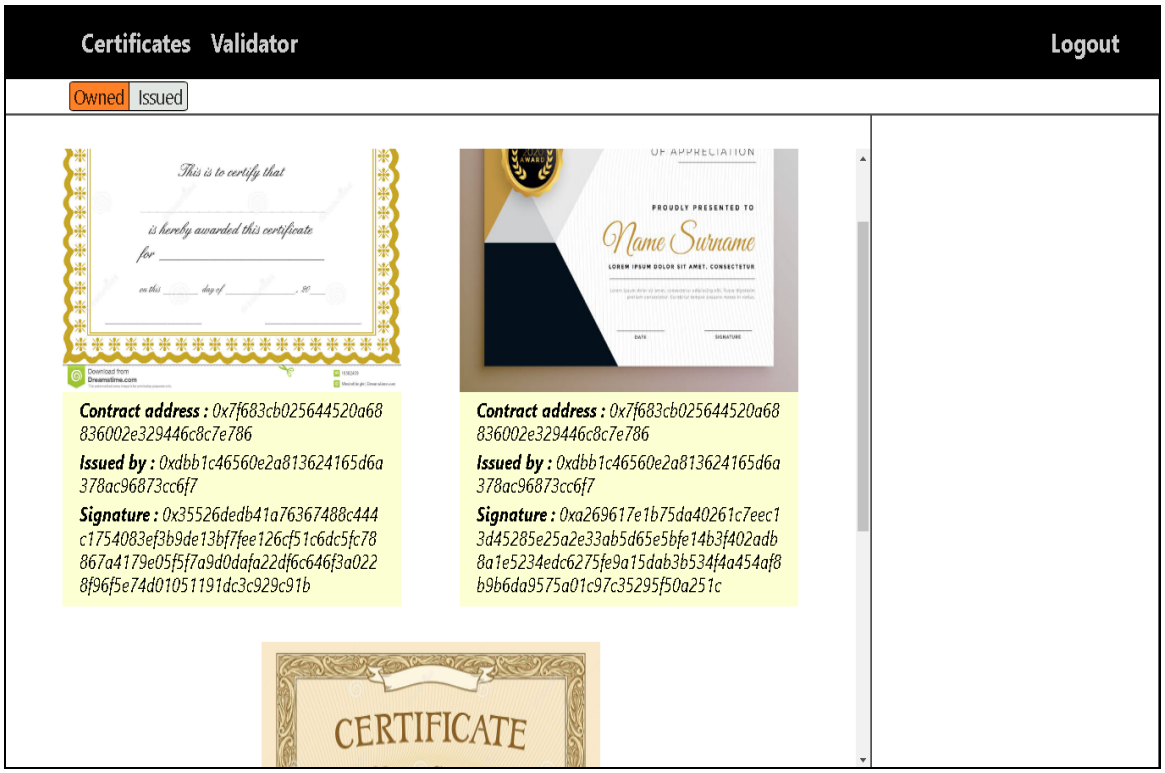


Βλέπουμε τώρα τα τρία πιστοποιητικά που φαίνονται στην γραφική του διεπαφή.

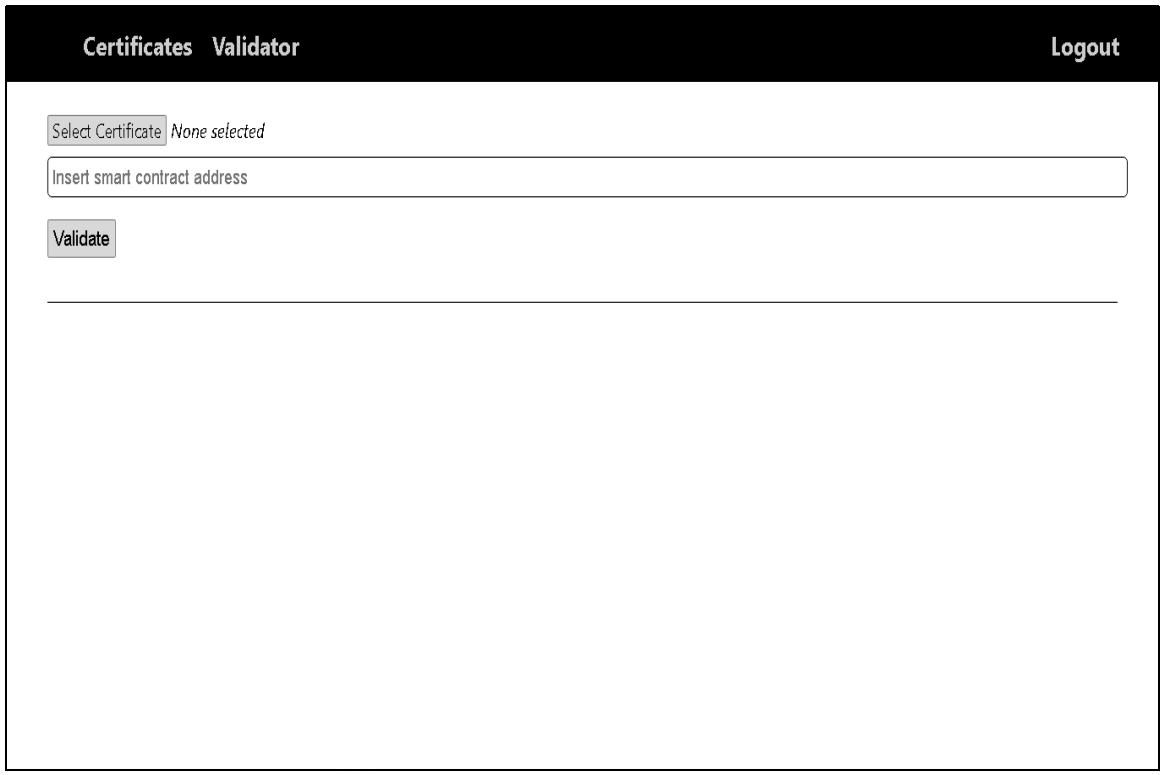


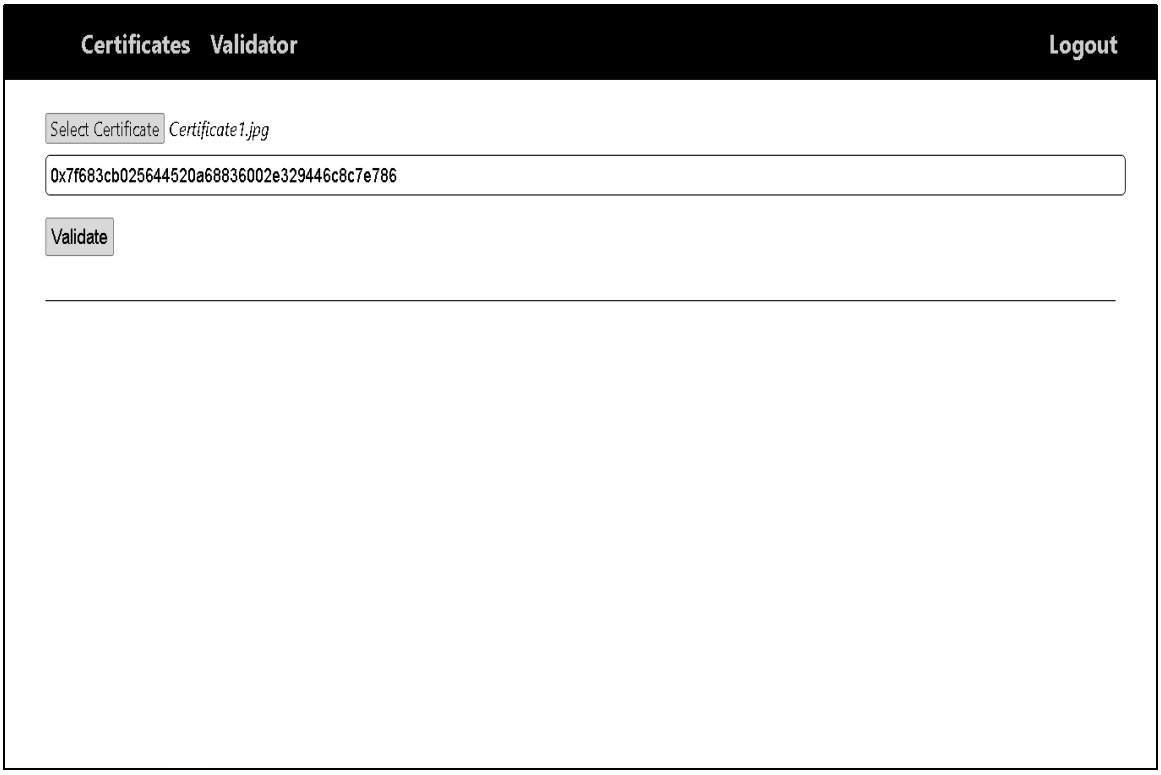
Τα ίδια πιστοποιητικά φαίνονται και στην γραφική διεπαφή του Λογαριασμού 3 αλλά στην καρτέλα Owned certificates.



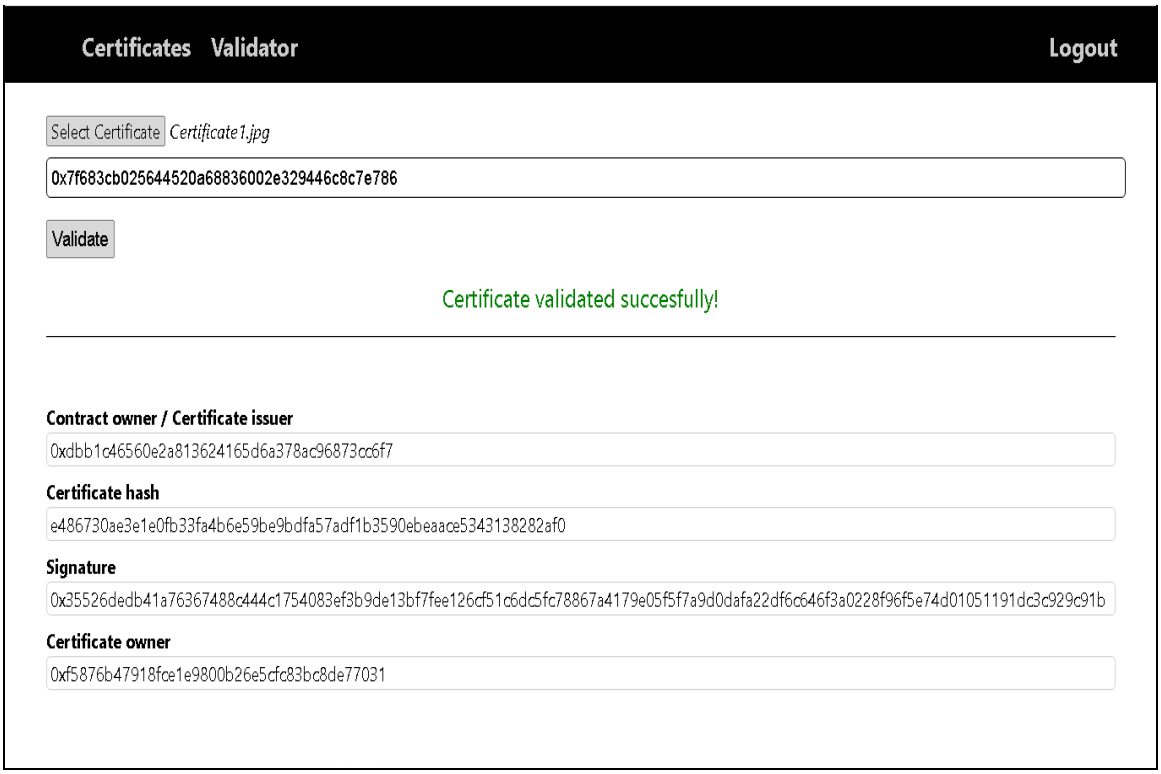

## 5.12 Επαλήθευση πιστοποιητικού από τον Λογαριασμό 3



Αρχικά εισάγει την διεύθυνση του έξυπνου συμβολαίου και το πρώτο πιστοποιητικό.




Πατάει Validate και βλέπουμε το αποτέλεσμα της επιτυχούς επαλήθευσης του πιστοποιητικού. Φαίνονται ο εκδότης του πιστοποιητικού-ιδιοκτήτης του συμβολαίου, το αποτύπωμα του συμβολαίου, η υπογραφή του αποτυπώματος και ο ιδιοκτήτης του πιστοποιητικού.



# Επίλογος

## Συμπεράσματα

Η τεχνολογία του blockchain παρουσιάζει μεγάλη ανάπτυξη και με την καινοτομία του Ethereum, που άνοιξε τον δρόμο για την ανάπτυξη αποκεντρωμένων εφαρμογών αποτελεί την βάση για την νέα γενιά του διαδικτύου που χτίζεται. Όντας ακόμα σε πρώιμο στάδιο τίποτα δεν είναι σταθερό και προκαθορισμένο και νέα πεδία εφαρμογής και τρόποι χρήσης συνεχώς παρουσιάζονται.

Στην παρούσα διπλωματική εργασία σχεδιάστηκε και αναπτύχθηκε μια διαδικτυακή εφαρμογή ψηφιακών πιστοποιητικών με χρήση του blockchain και των έξυπνων συμβολαίων. Δημιουργήθηκε ως λειτουργικό πρωτότυπο που θα αποτελέσει την βάση για ένα νέο τρόπο πιστοποίησης, ψηφιακό, αποκεντρωμένο και ασφαλή. Στόχος της εργασίας ήταν τόσο η μελέτη και κατανόηση της τεχνολογίας του blockchain και των εφαρμογών της στα πλαίσια του Web3, όσο και η τριβή με σύγχρονες τεχνολογίες και εργαλεία ανάπτυξης αποκεντρωμένων εφαρμογών και η δημιουργία μιας.

Η εφαρμογή που τελικά υλοποιήθηκε εκπληρώνει τις λειτουργικές απαιτήσεις της αποθήκευσης, έκδοσης και επαλήθευσης ψηφιακών πιστοποιητικών καθώς και τις μη λειτουργικές απαιτήσεις της ιδιωτικότητας, διαφάνειας, ακεραιότητας των δεδομένων και εύκολης και άμεσης επαλήθευσης των πιστοποιητικών.

## Μελλοντικό έργο

Όπως ειπώθηκε η εφαρμογή αποτελεί ένα λειτουργικό πρωτότυπο που ενσαρκώνει την ιδέα. Κατά την ανάπτυξή της έγιναν παραδοχές και σαφώς δεν μπορεί να θεωρηθεί πλήρης και έτοιμη για λειτουργία σε πραγματικές συνθήκες. Επομένως δίνεται η δυνατότητα περαιτέρω ανάπτυξης και επέκτασης της.

Μία παραδοχή που έγινε αφορούσε τον τρόπο αποθήκευσης των πιστοποιητικών. Τα πιστοποιητικά όλων των χρηστών αποθηκεύονται στην βάση της εφαρμογής. Η απόφαση αυτή έγινε καθαρά για την απλοποίηση της υλοποίησης ωστόσο είναι αντίθετη στην αρχική ιδέα. Κατά την αρχική ιδέα τα πιστοποιητικά αποτελούν προσωπική ιδιοκτησία των χρηστών επομένως μόνο οι ίδιοι θα πρέπει να είναι κάτοχοί τους. Η εφαρμογή πρέπει να αποθηκεύει μόνο τα αποτυπώματα των πιστοποιητικών τα οποία θα χρησιμοποιεί για την επαλήθευσή τους, όχι όμως τα ίδια τα πιστοποιητικά. Μελέτη του τρόπου αποθήκευσης των πιστοποιητικών αποτελεί ένα μονοπάτι επέκτασης της εφαρμογής. Πιθανή λύση θα μπορούσε να είναι η αποθήκευση στις προσωπικές συσκευές των χρηστών όμως θα πρέπει να μελετηθούν τα ζητήματα της απώλειας των πιστοποιητικών καθώς και της μη διαθεσιμότητας σε περίπτωση χρήσης άλλης συσκευής. Επιπλέον οι εκδότες των πιστοποιητικών όπως για παράδειγμα τα ακαδημαϊκά ιδρύματα έχουν την ανάγκη πρόσβασης σε αυτά σε ορισμένες περιπτώσεις, γεγονός που θα πρέπει να συνυπολογιστεί.

Μια άλλη παραδοχή ήταν η τοποθέτηση εκτός του πεδίου μελέτης της εφαρμογής, του προβλήματος της αντιστοίχισης των δημόσιων κλειδιών των χρηστών με τις φυσικές ή νομικές οντότητες. Κατεύθυνση μελέτης και επέκτασης της εφαρμογής μπορεί να αποτελέσει η υλοποίηση αυτής της αντιστοίχισης. Το ζήτημα μελετήθηκε σε κάποιο βαθμό και δύο πιθανές λύσεις θεωρήθηκαν. Η πρώτη είναι η υλοποίηση πιστοποιητικών ταυτότητας από Certificate authorities, η οποίο βέβαια προϋποθέτει την ύπαρξη έμπιστης τρίτης οντότητας και καθιστά το σύστημα λιγότερο αποκεντρωμένο. Η δεύτερη είναι η ανάπτυξη μιας ροής ταυτοποίησης, με αποστολή μηνύματος που περιέχει στοιχεία απόδειξης της ταυτότητας του φυσικού προσώπου, όπως για παράδειγμα η φοιτητική

ταυτότητα και είναι υπογεγραμμένο από το ιδιωτικό κλειδί του χρήστη. Σε αυτή την περίπτωση βέβαια βασιζόμαστε σε χρήση φυσικών πιστοποιητικών ταυτότητας με τα όποια μειονεκτήματα αυτά φέρουν.

Τέλος η εφαρμογή μπορεί να επεκταθεί άμεσα με την υλοποίηση νέων λειτουργιών όπως λειτουργία διαγραφής πιστοποιητικών μετά το πέρας της ημερομηνίας ισχύος τους ή λειτουργία ανανέωσης πεδίων πιστοποιητικών σε περίπτωση λαθών. Καθώς αναφερόμαστε σε τεχνολογία που είναι σε πρώιμο στάδιο και συνεχώς εξελίσσεται, όντας δημιουργικοί και παρατηρώντας το περιβάλλον μπορούμε να εντοπίζουμε τις νέες απαιτήσεις και ανάγκες που προκύπτουν και να επεκτείνουμε το σύστημα.

# Βιβλιογραφία


[1] "A Brief History of Certification – TestOut Continuing Education." https://testoutce.com/blogs/it-insights-blog/160401479-a-brief-history-of-certification

[2] "Public-key cryptography - Wikipedia." https://en.wikipedia.org/wiki/Public-key_cryptography.

[3] "Εγκεκριμένοι Πάροχοι Υπηρεσιών Εμπιστοσύνης: Η αναγκαιότητα των Παρόχων Υπηρεσιών Εμπιστοσύνης." https://howto.gov.gr/mod/book/view.php?id=1224&chapterid=1439.

[4] S. Haber and W. Scott Stornetta, "How to time-stamp a digital document," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 537 LNCS, pp. 437–455, 1991, doi: 10.1007/3-540-38424-3_32/COVER.

[5] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System".

[6] U. W. Chohan, "The Double Spending Problem and Cryptocurrencies," *SSRN Electronic Journal*, Jan. 2021, doi: 10.2139/SSRN.3090174.

[7] "Sci-Hub | Blockchain technology overview | 10.6028/nist.ir.8202." https://sci-hub.se/https://doi.org/10.6028/NIST.IR.8202.

[8] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.".

[9] "Gasper | ethereum.org." https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/gasper/.

[10] "Ethereum Virtual Machine (EVM) | ethereum.org." https://ethereum.org/en/developers/docs/evm/.

[11] "Introduction to smart contracts | ethereum.org." https://ethereum.org/en/developers/docs/smart-contracts/.

[12] "Solidity — Solidity 0.8.19 documentation." https://docs.soliditylang.org/en/v0.8.19/

[13] "Ganache - Truffle Suite." https://trufflesuite.com/ganache/.

[14] "The crypto wallet for Defi, Web3 Dapps and NFTs | MetaMask." https://metamask.io/.

[15] "web3.js - Ethereum JavaScript API — web3.js 1.0.0 documentation." https://web3js.readthedocs.io/en/v1.8.2/index.html.

[16] "React – The library for web and native user interfaces." https://react.dev/.

[17] "Writing Markup with JSX – React." https://react.dev/learn/writing-markup-with-jsx#jsx-putting-markup-into-javascript.

[18] "About | Node.js." https://nodejs.org/en/about.

[19] "Express - Node.js web application framework." https://expressjs.com/.

[20] "JSON-RPC API | ethereum.org." https://ethereum.org/en/developers/docs/apis/json-rpc/.

[21] "Fetch API - Web APIs | MDN." https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API.

[22] "What is RESTful API? - RESTful API Explained - AWS." https://aws.amazon.com/what-is/restful-api/?trk=faq_card.

[23] "mysql - npm."

https://www.npmjs.com/package/mysql.
[24] “Data URLs - HTTP | MDN.” https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URLs.
[25] “SHA-2 - Wikipedia.” https://en.wikipedia.org/wiki/SHA-2.