Επιβράβευση με την υποστήριξη υποδομής Blockchain

ΕΠΙΒΛΕΠΟΝ
ΚΑΘΗΓΗΤΗΣΕΛΕΝΗ
ΑΙΚΑΤΕΡΙΝΗ ΛΕΛΙΓΚΟΥ

ΦΟΙΤΗΤΗΣ

ΦΡΑΝΣ ΤΖΕΜΟΛΛΑΡΙ
70147288

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
	ΛΕΛΙΓΚΟΥ ΑΙΚΑΤΕΡΙΝΗ ΕΛΕΝΗ		Eleni Aikaterini Leligkou Digitally signed by Eleni Aikaterini Leligkou Date: 2023.07.25 13:01:59 +03'00'
	ΔΡΟΣΟΣ ΧΡΗΣΤΟΣ		Dimitrios Kantzos Digitally signed by Dimitrios Kantzos Date: 2023.07.26 11:35:06 +03'00'
	ΚΑΝΤΖΟΣ ΔΗΜΗΤΡΙΟΣ		Christos Drosos Digitally signed by Christos Drosos Date: 2023.07.20 22:08:22 +03'00'

ΠΕΡΙΕΧΟΜΕΝΑ

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Τζεμολλάρι Φρανς του Φερντινάντ, με αριθμό μητρώου 70147288 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Βιομηχανικής Σχεδίασης & Παραγωγής του Τμήματος Μηχανικών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Φρανς Τζεμολλάρι

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά θα ήθελα να ευχαριστήσω θερμά και από τα βάθη της καρδιάς μου τον Δημήτριο Σταματάκη, έναν καταπληκτικό άνθρωπο, καθώς ήταν από την αρχή εώς το τέλος δίπλα μου καθ' όλη την διπλωματική εργασία. Με της σωστή και κατάλληλη καθοδήγηση του, παρακολουθώντας συνεχώς την πορεία μου, όποτε αυτό έχρηζε αναγκαίο.

Έπειτα θέλω να ευχαριστήσω την κυρία Ελένη Αικατερίνη Λελίγκου, η οποία δέχθηκε να συνεργαστεί μαζί μου και να μου δώσει την ευκαιρία να ασχοληθώ και να εξερευνήσω ένα άκρος ενδιαφέρων και συναρπαστικό κομμάτι του προγραμματισμού.

Συνεχίζοντας, θα ήθελα να ευχαριστήσω τον Ιωάννη Χριστίδη, ο οποίος ήταν ο πρώτος άνθρωπος που μου μίλησε, με ενέταξε σε αυτό τον κλάδο προγραμματισμού, μου εξήγησε με τι ασχολείται αυτός ο κλάδος και με έφερε σε επαφή με τους αρμόδιους (όπως προανέφερα τον Δημήτριο Σταματάκη).

Τέλος, θα ήθελα να ευχαριστήσω οικογένεια και φίλους, που με τον δικό τους μοναδικό τρόπο ο καθένας, με βοήθησαν ανταπεξέλθω κούραση και δυσκολίες για να βρεθώ σε αυτό σημείο που βρίσκομαι σήμερα.

Περίληψη

Η εργασία με τον τίτλο "Επιβράβευση με την υποστήριξη υποδομής Blockchain" επικεντρώνεται στην ανάπτυξη ενός ιστοχώρου τύπου φόρουμ (Forum) με χρήση της τεχνολογίας blockchain.

Συγκεκριμένα, έχετε δημιουργήσει έναν ιστότοπο που επιτρέπει τη σύνδεση μέσω του Metamask, χρησιμοποιώντας κώδικα, και έχετε δημιουργήσει τα αντίστοιχα πορτοφόλια και τη blockchain υποδομή.

Η λειτουργία του ιστοτόπου σας είναι τέτοια, ώστε όταν ένας χρήστης συνδέεται και συνδέει το πορτοφόλι του, αν δημιουργεί ένα νέο άρθρο στο φόρουμ, αμείβεται με 10 κέρματα (coins) ως ανταμοιβή.

Επίσης, όταν προσθέτει σχόλιο σε υπάρχον άρθρο, λαμβάνει 5 κέρματα σαν ανταμοιβή.

Μέσω της υποδομής blockchain, η επιβράβευση των χρηστών γίνεται με ασφάλεια και αξιοπιστία, καθώς κάθε δραστηριότητα και συναλλαγή καταγράφεται σε ένα κατανεμημένο και αναλλοίωτο βιβλίο επί του blockchain.

Αυτό ενισχύει τη διαφάνεια και την εμπιστοσύνη στο σύστημα, προσφέροντας στους χρήστες ένα κίνητρο για συμμετοχή και συνεισφορά στην κοινότητα του φόρουμ σας.

Συνοψίζοντας, η εργασία σας επιτυγχάνει την επιβράβευση των χρηστών μέσω της υποστήριξης υποδομής blockchain, προσφέροντας μια ασφαλή και διαφανή εμπειρία σε ένα φόρουμ με αμοιβές για τις δραστηριότητες των χρηστών.

Summary

The work titled "Rewarding with Blockchain Infrastructure Support" focuses on developing a forum-type website using blockchain technology.

Specifically, you have created a website that allows users to connect via Metamask using code, and you have established the corresponding wallets and blockchain infrastructure.

The functionality of your website is such that when a user connects and links their wallet, they are rewarded with 10 coins for creating a new article on the forum.

Additionally, when they add a comment to an existing article, they receive 5 coins as a reward.

Through the blockchain infrastructure, user rewards are carried out securely and reliably, as every activity and transaction is recorded in a distributed and immutable ledger on the blockchain.

This enhances transparency and trust in the system, providing users with an incentive to participate and contribute to your forum community.

In summary, your work achieves user rewards through the support of blockchain infrastructure, offering a secure and transparent experience in a forum with rewards for user activities.

1.Εισαγωγή στις βασικές έννοιες του Block Chain

1.1 Τι είναι το Block Chain

Το blockchain είναι ένα καινοτόμο τεχνολογικό σύστημα που έχει επανασχεδιάσει τον τρόπο με τον οποίο λειτουργούν οι συναλλαγές και οι εγγραφές δεδομένων. Σε αντίθεση με τις παραδοσιακές κεντρικές βάσεις δεδομένων, το blockchain λειτουργεί με ένα αποκεντρωμένο σύστημα, το οποίο επιτρέπει σε πολλούς υπολογιστές να συνεργαστούν για την επαλήθευση και την ασφάλεια των συναλλαγών.

Στη βάση του, το blockchain αποτελείται από ένα αλυσιδωτό σύνολο μπλοκ. Κάθε μπλοκ περιέχει μια λίστα από συναλλαγές και ένα μοναδικό αναγνωριστικό, γνωστό ως "hash", που αντιπροσωπεύει το περιεχόμενο του μπλοκ.

Οι μπλοκ συνδέονται μεταξύ τους μέσω των hash, δημιουργώντας μια αλυσίδα με χρονική σειρά των συναλλαγών.

Η κύρια λειτουργία του blockchain είναι η επιβεβαίωση και η ασφάλεια των συναλλαγών. Κάθε συναλλαγή που γίνεται στο blockchain πρέπει να επαληθεύεται από τους κόμβους του δικτύου πριν προστεθεί σε ένα μπλοκ.

Αυτό δημιουργεί μια ανεξάρτητη, αξιόπιστη και διαφανή διαδικασία επαλήθευσης, καθώς οι συμμετέχοντες στο δίκτυο έχουν την ίδια εκδοχή των δεδομένων.

Ένα ακόμα σημαντικό χαρακτηριστικό του blockchain είναι η αποκέντρωση. Αντί να εξαρτώνται από έναν κεντρικό φορέα ελέγχου, τα δεδομένα του blockchain διανέμονται και αποθηκεύονται σε πολλούς κόμβους σε όλο το δίκτυο. Αυτό το χαρακτηριστικό το καθιστά ανθεκτικό σε επιθέσεις και αποτρέπει την αλλοίωση ή την απώλεια δεδομένων.

Επιπλέον, οι blockchain τεχνολογίες μπορούν να χρησιμοποιηθούν για την εκτέλεση έξυπνων συμβολαίων. Ένα έξυπνο συμβόλαιο είναι ένα πρόγραμμα που εκτελεί αυτόματα συμφωνημένες ενέργειες όταν εκπληρούνται συγκεκριμένες προϋποθέσεις.

Το blockchain παρέχει το πλαίσιο για την ασφαλή εκτέλεση και την επαλήθευση αυτών των συμβολαίων, χωρίς την ανάγκη εμπιστοσύνης σε έναν ενδιάμεσο.

Το blockchain έχει ευρεία εφαρμογή σε πολλούς τομείς, όπως οι οικονομικές συναλλαγές, η αλυσίδα εφοδιασμού, η ψηφιακή ταυτότητα, η υγεία και πολλοί άλλοι.

Η δυνατότητα να δημιουργηθούν αξιόπιστες, ασφαλείς και αυτόνομες διαδικτυακές πλατφόρμες έχει κάνει το blockchain έναν σημαντικό παράγοντα της ψηφιακής μετάβασης.

Συνοψίζοντας, το blockchain είναι μια τεχνολογία που ανατρέπει τον τρόπο με τον οποίο λειτουργούν οι συναλλαγές και οι εγγραφές δεδομένων. Με την αποκέντρωση, την επαλήθευση και την ασφάλεια που παρέχει, το blockchain αποτελεί έναν σημαντικό πυλώνα της ψηφιακής εποχής και προσφέρει αμέτρητες δυνατότητες για την ανακατασκευή των διαδικτυακών εφαρμογών.

[2][3][4]

Το Blockchain είναι μια μορφή αποθήκευσης συναλλαγών που λειτουργεί ως αποκεντρωμένο σύστημα διαχείρισης. Αυτό σημαίνει ότι πολλοί συμμετέχοντες, γνωστοί και ως κόμβοι, επιβεβαιώνουν και επικυρώνουν τις συναλλαγές.

Σύμφωνα με έρευνες [5][6], το blockchain μπορεί να θεωρηθεί ως ένα είδος τεχνολογίας κατανεμημένου καθολικού συστήματος που προσφέρει ασφάλεια στις αρχειοθετημένες πληροφορίες, όπως πιστοποιητικά, αποτρέποντας την παραβίασή τους και την αλλοίωσή τους.

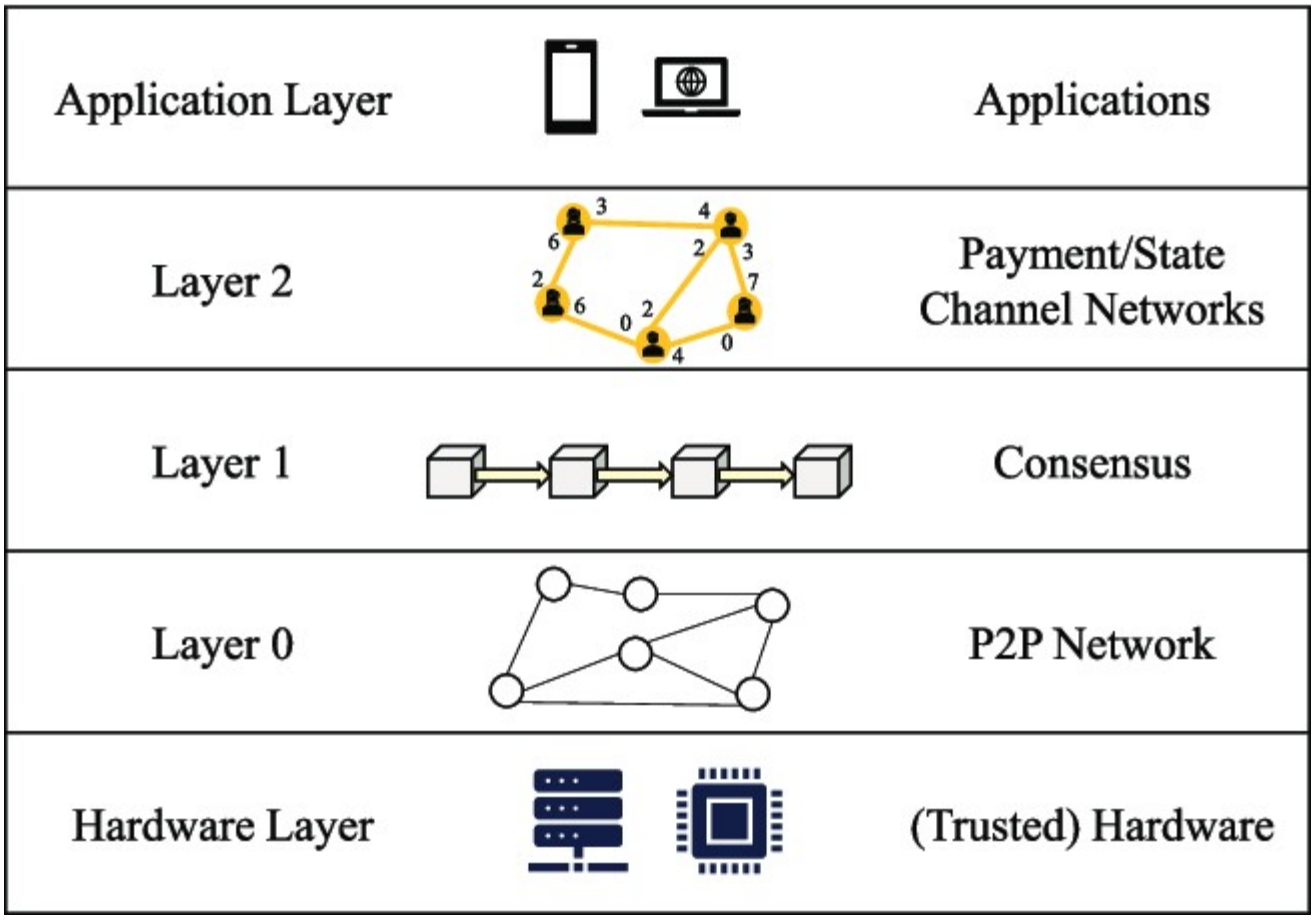
Αυτό επιτυγχάνεται μέσω της πλήρους καταγραφής και αποκάλυψης όλων των συναλλαγών στους συμμετέχοντες του δικτύου [7].

Επιπλέον, το blockchain αναμένεται να επανασχεδιάσει τις οικονομίες και την κοινωνία, μειώνοντας το κόστος και προσφέροντας οικονομικές συναλλαγές χωρίς την ανάγκη για αξιόπιστα τρίτα μέρη [8][9].

Επιπλέον, έρευνα [10] έχει αναφέρει ότι η τεχνολογία blockchain μπορεί να χρησιμοποιηθεί για την καταγραφή συναλλαγών, την αποθήκευση ιατρικών αρχείων, τη δημιουργία δεσμευτικών συμφωνιών, την παρακολούθηση της κυκλοφορίας εμπορευμάτων, την αποθήκευση πιστωτικών αρχείων, την παρακολούθηση της πορείας έργων τέχνης και τον έλεγχο πληρωμών μέσω της προμηθευτικής αλυσίδας, μεταξύ πολλών άλλων διαδικασιών.

Για να κατανοήσετε την αρχιτεκτονική του blockchain, θα πρέπει να γνωρίζετε ότι αποτελείται από τρία επίπεδα (layers) [11]:

1. Επίπεδο υλικού (Hardware Layer): Αποτελείται από τους κόμβους του δικτύου, οι οποίοι συμμετέχουν στη διαδικασία συναίνεσης για την επικύρωση ή απόρριψη νέων συναλλαγών, χρησιμοποιώντας την υπολογιστική ισχύ τους. Επίσης, αποθηκεύουν το ιστορικό των συναλλαγών που έχουν πραγματοποιηθεί στο δίκτυο blockchain.
2. Επίπεδο λογιστικού (Ledger Layer): Σε αυτό το επίπεδο ανήκουν οι συναλλαγές και τα μπλοκ που τις περιέχουν. Αποτελεί το καθολικό κατανεμημένο καταμερισμένο βιβλίο μεταξύ όλων των συμμετεχόντων.
3. Επίπεδο εφαρμογών (Application Layer): Σε αυτό το επίπεδο ανήκουν οι εφαρμογές, ιστοσελίδες και προϊόντα που αξιοποιούν την τεχνολογία blockchain για τη δημιουργία αξίας.



Εικόνα 1.1 Τα τρία επίπεδα της αρχιτεκτονικής των Blockchain.

Με την κατανόηση της αρχιτεκτονικής του blockchain, μπορείτε να αποκτήσετε μια σφαιρική εικόνα της τεχνολογίας αυτής και να κατανοήσετε τα πλεονεκτήματα που συνοδεύουν.

1.2 Βασικές Λειτουργείες

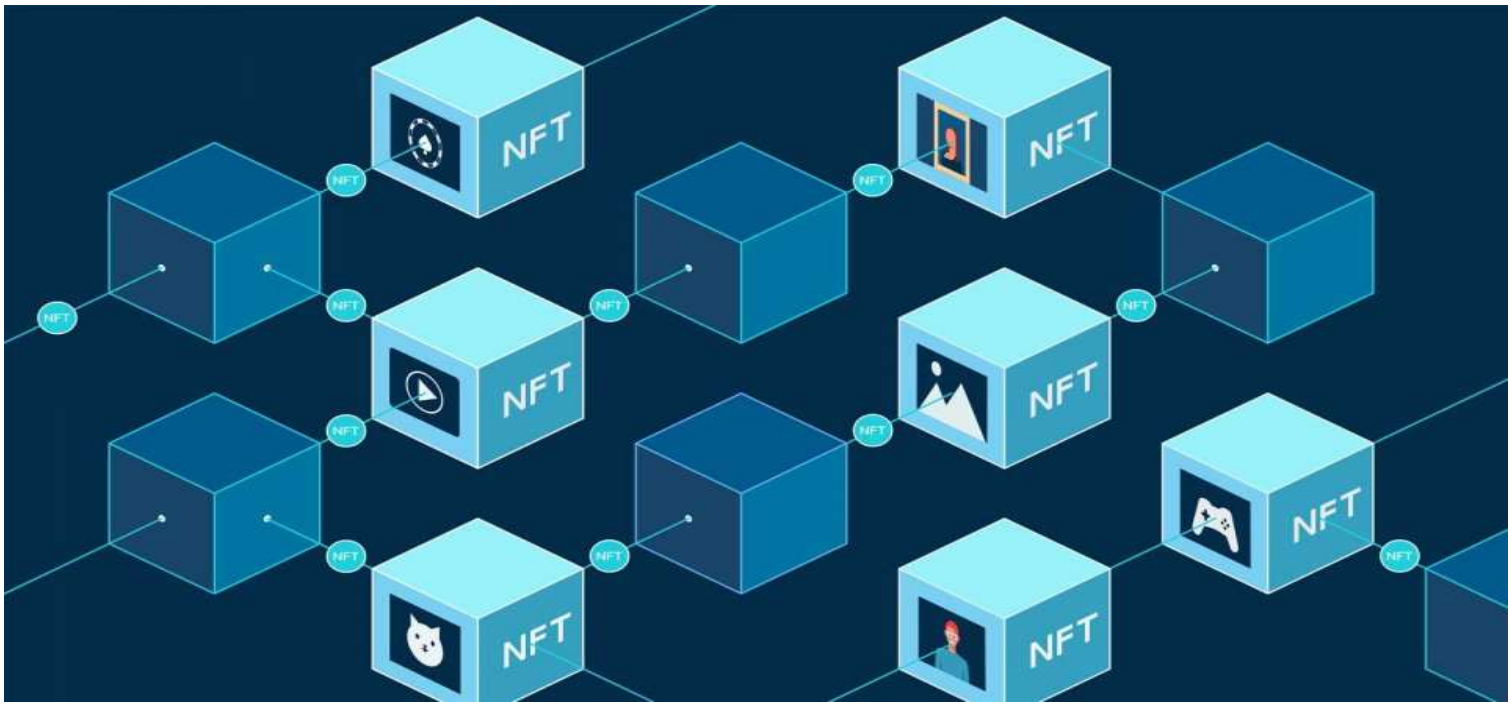
Το blockchain προσφέρει ορισμένα σημαντικά πλεονεκτήματα σε σχέση με τις παραδοσιακές μεθόδους καταχώρησης και διαχείρισης δεδομένων. Ας εξετάσουμε ορισμένες από αυτές:

1. Αποκέντρωση και ανεξαρτησία: Το blockchain λειτουργεί χωρίς την ανάγκη ενός κεντρικού αρχείου ή φορέα ελέγχου.
   Αυτό σημαίνει ότι κάθε συμμετέχωντας κόμβος έχει το ίδιο αντίγραφο του blockchain και μπορεί να επαληθεύσει τις συναλλαγές. Αυτή η ανεξαρτησία και αποκέντρωση καθιστά το σύστημα πιο αξιόπιστο και ανθεκτικό σε επιθέσεις.
2. Ασφάλεια και ακεραιότητα: Οι συναλλαγές στο blockchain είναι κρυπτογραφημένες και συνδέονται μεταξύ τους με μοναδικά hash.
   Αυτό δημιουργεί μια αλυσίδα που είναι πολύ δύσκολο να παραβιαστεί ή να αλλοιωθεί. Επιπλέον, η επαλήθευση των συναλλαγών από πολλούς κόμβους καθιστά την απάτη και την αλλοίωση σχεδόν αδύνατες.

3. Διαφάνεια: Το blockchain είναι μια δημόσια κατανεμημένη βάση δεδομένων, που σημαίνει ότι όλοι οι συμμετέχοντες έχουν πρόσβαση στις συναλλαγές και τις εγγραφές. Αυτό διασφαλίζει υψηλότερο επίπεδο διαφάνειας και εμπιστοσύνης, καθώς οι συναλλαγές μπορούν να ελεγχθούν από οποιονδήποτε ενδιαφερόμενο.
4. Ταχύτητα και αποδοτικότητα: Μολονότι το blockchain μπορεί να απαιτεί περισσότερο χρόνο για την επιβεβαίωση μιας συναλλαγής σε σχέση με τις παραδοσιακές μεθόδους, έχουν αναπτυχθεί καινοτόμοι τρόποι για τη βελτίωση της απόδοσης.
   Για παράδειγμα, υπάρχουν ιδιόκτητα blockchain δίκτυα που επιτρέπουν γρηγορότερες συναλλαγές και επεξεργασία μεγαλύτερου αριθμού δεδομένων.

Το blockchain έχει τη δυνατότητα να αλλάξει τον τρόπο λειτουργίας των πολλών τομέων, προσφέροντας αυξημένη ασφάλεια, διαφάνεια, επίδοση και ευκαιρίες για καινοτομία. Καθώς η τεχνολογία εξελίσσεται, αναμένεται να προκύψουν ακόμα περισσότερες εφαρμογές και προοπτικές για το blockchain.

[2][3][4]



Εικόνα 1.2 What are the blockchains. Διαθέσιμο στο διαδίκτυο.

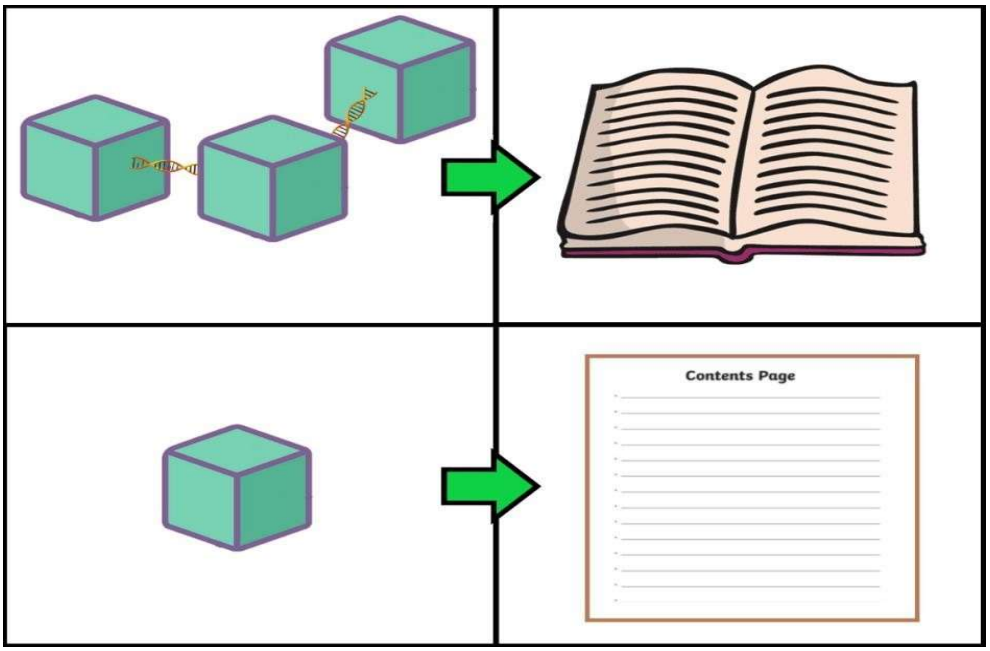
1.3 Ιστορία

Η πρώτη ιστορικά εφαρμοσμένη τεχνολογία blockchain ήταν η ψηφιακή νόμισμα Bitcoin, που δημιουργήθηκε από τον Satoshi Nakamoto. Σε αυτό το έργο, προτάθηκε μια λύση σε ένα γνωστό πρόβλημα, το πρόβλημα του Διπλοζυγίου Ξοδεύματος, που ήταν γνωστό από το 1980.

Το πρόβλημα αυτό αναφέρεται σε μια κατάσταση όπου ένα ψηφιακό νόμισμα μπορεί να χρησιμοποιηθεί περισσότερα από μια φορά. Οι λύσεις που προτάθηκαν πριν από την εφεύρεση της τεχνολογίας της αλυσίδας δεν ήταν ικανοποιητικές και είχαν περιορισμένη εφαρμογή.

Η τεχνολογία της blockchain είχε την ικανότητα να επιλύσει αυτό το πρόβλημα και να παρέχει ένα αξιόπιστο σύστημα για τη μεταφορά και την αποθήκευση ψηφιακού νομίσματος. Από τότε, η τεχνολογία της blockchain

έχει εξελιχθεί και έχει εφαρμοστεί σε διάφορους τομείς, όπως οι οικονομικές συναλλαγές, οι συμβάσεις έξυπνων και εφαρμογών συστήματος, η αλυσίδα εφοδιασμού και πολλοί άλλοι.[1]



Εικόνα 1.3 Παρομοίωση Βιβλίου με Blockchain. Δημιουργός:Δημήτριος Σταματάκης

1.4 Εφαρμογές Block Chain

Μπορούμε να εξετάσουμε κάποιες από τις πιο γνωστές εφαρμογές του blockchain:

1. Κρυπτονομίσματα: Το πιο γνωστό παράδειγμα είναι το Bitcoin, το οποίο είναι η πρώτη και πιο διάσημη κρυπτονόμισμα. Τα κρυπτονομίσματα χρησιμοποιούν το blockchain για την εκτέλεση και την επαλήθευση των συναλλαγών μεταξύ των χρηστών χωρίς την ανάγκη ενός κεντρικού φορέα.

2. Διαχείριση εφοδιαστικής αλυσίδας: Το blockchain μπορεί να χρησιμοποιηθεί για την αποτελεσματική και ασφαλή διαχείριση της αλυσίδας εφοδιασμού.

   Μπορεί να καταγράψει το ιστορικό των προϊόντων από την αρχική παραγωγή έως την παράδοση στον τελικό καταναλωτή, εξασφαλίζοντας την ποιότητα, την προέλευση και την ακεραιότητα των προϊόντων.

3. Ψηφιακή ταυτότητα: Οι τεχνολογίες blockchain μπορούν να χρησιμοποιηθούν για τη δημιουργία ασφαλών και αξιόπιστων ψηφιακών ταυτοτήτων.
   Αυτό μπορεί να βοηθήσει στην επαλήθευση της ταυτότητας ατόμων, την αποθήκευση προσωπικών δεδομένων με ασφάλεια και τη διευκόλυνση της ανταλλαγής πληροφοριών μεταξύ διάφορων οντοτήτων.

4. Έξυπνα συμβόλαια: Το blockchain μπορεί να υποστηρίξει την υλοποίηση έξυπνων συμβολαίων, τα οποία είναι αυτόματα εκτελούμενα και ελέγχονται από το σύστημα. Αυτά τα συμβόλαια μπορούν να

επιτρέψουν την αυτόματη μεταφορά πόρων, την εκτέλεση συναλλαγών και την εφαρμογή κανόνων με βάση τις προκαθορισμένες συνθήκες.

Αυτές είναι μόνο μερικές από τις εφαρμογές του blockchain που υπάρχουν σήμερα. Η τεχνολογία ανοίγει τον δρόμο για νέες καινοτομίες και αναμένεται να επηρεάσει τον τρόπο με τον οποίο λειτουργούν πολλοί τομείς της κοινωνίας. [1]

1.5 Blockchain-Βιβλίο

Ένα blockchain αναπαριστάται ως ένα σύνολο υπολογιστών (κόμβων) και μια αποθήκευση συναλλαγών[12]. Το blockchain αποτελείται από δεδομένα τα οποία έχουν οργανωθεί σε μονάδες, γνωστές και ως μπλοκ.

Κάθε μπλοκ παρομοιάζεται με μια σελίδα σε ένα λογιστικό βιβλίο. Κάθε "σελίδα" έχει έναν μοναδικό αριθμό και είναι εύκολο να ελεγχθεί εάν μια σελίδα (ή ένα μπλοκ) έχει διαγραφεί, απλά εξετάζοντας τους αριθμούς των σελίδων.

Αν δεν υπάρχει συνέχεια στους αριθμούς, τότε κάποια σελίδα έχει αφαιρεθεί. Κάθε "σελίδα" περιέχει δεδομένα (συναλλαγές) τα οποία έχουν οργανωθεί με λογική και χρονολογική σειρά.

Οι πληροφορίες αυτές κρυπτογραφούνται χρησιμοποιώντας κρυπτογραφικούς αλγορίθμους, προκειμένου να διασφαλιστεί η απόρρητη φύση των δεδομένων του χρήστη και η ακεραιότητα των δεδομένων.

Τα μπλοκ ενώνονται μεταξύ τους, σχηματίζοντας ένα μοναδικό βιβλίο. Και όπως σε μια βιβλιοθήκη, μπορεί να υπάρχουν πολλά διαφορετικά βιβλία, ή με άλλα λόγια, πολλά διαφορετικά είδη blockchain.

Το blockchain λειτουργεί μέσω ενός συστήματος υποκοπής (consensus mechanism), το οποίο είναι υπεύθυνο για την επιβεβαίωση της εγκυρότητας των συναλλαγών και την επίτευξη συμφωνίας μεταξύ των κόμβων.

Ένα δημοφιλές σύστημα υποκοπής είναι το Proof of Work (PoW), όπου οι κόμβοι πρέπει να επιλύουν ένα πρόβλημα υπολογιστικής πολυπλοκότητας για να πιστοποιήσουν μια συναλλαγή και να προσθέσουν ένα μπλοκ στο blockchain.

Μια άλλη μέθοδος είναι το Proof of Stake (PoS), όπου η επιλογή του επόμενου μπλοκ γίνεται βάσει του ποσοστού των νομισμάτων που κατέχει κάθε κόμβος.

Συνοψίζοντας, το blockchain είναι ένα σύστημα που χρησιμοποιείται για την ασφαλή και αξιόπιστη αποθήκευση και μεταφορά δεδομένων. Με τη βοήθεια της κρυπτογραφίας και των συστημάτων υποκοπής, το blockchain παρέχει ανθεκτικότητα, ασφάλεια και ανακλητότητα, καθιστώντας το ένα αξιόπιστο εργαλείο για πολλές εφαρμογές όπως τις κρυπτονομίσματα, τις συμβολαιακές συναλλαγές και την αποθήκευση δεδομένων.

1.6 Χαρακτηριστικά Blockchain

1. Ένα blockchain είναι ένα σύστημα υπολογιστών (κόμβων) και ένα μέρος αποθήκευσης συναλλαγών.
2. Ένα blockchain αποτελείται από δεδομένα που έχουν οργανωθεί σε μονάδες ή πιο συγκεκριμένα σε blocks.
3. Κάθε block μοιάζει με μια σελίδα από ένα λογιστικό βιβλίο.
4. Κάθε "σελίδα" είναι αριθμημένη και είναι πολύ εύκολο να εντοπιστεί εάν κάποια σελίδα (ή μπλοκ) έχει διαγραφεί κοιτάζοντας τους αριθμούς των σελίδων, διότι δεν θα υπάρχει συνέχεια.
5. Κάθε «σελίδα» έχει υλικό (συναλλαγές) οργανωμένο με λογική και χρονολογική σειρά.
6. Οι πληροφορίες κρυπτογραφούνται με χρήση κρυπτογραφικών αλγορίθμων για να διασφαλιστεί ότι το απόρρητο του χρήστη δεν διακινδυνεύει και τα δεδομένα δεν μπορούν να αλλοιωθούν.
7. Οι σελίδες είναι όλες ραμμένες μεταξύ τους για να κάνουν ένα μόνο βιβλίο.
8. Μπορεί να υπάρχει μεγάλος αριθμός διαφορετικών ειδών blockchain.
9. Κάθε μπλοκ συνδέεται με το προηγούμενο με λογική σειρά.
10. Δεν μπορεί να αφαιρεθεί ένα μεμονωμένο μπλοκ χωρίς να αλλοιωθεί ολόκληρη η αλυσίδα από την αρχή έως το συγκεκριμένο μπλοκ.
11. Απαιτείται επαλήθευση σε κάθε μπλοκ που προστίθεται στην αλυσίδα.
12. Η εισαγωγή ενός μπλοκ είναι αδύνατη εκτός εάν η πλειοψηφία του δικτύου συμφωνεί ότι είναι γνήσιο και έχει επιβεβαιωθεί ως έγκυρο στοιχείο της αλυσίδας.
13. Η αλλαγή ήδη επικυρωμένων δεδομένων είναι εξαιρετικά δύσκολη.
14. Η ασφάλεια του blockchain είναι άμεσα συνδεδεμένη με τον συνολικό αριθμό των κόμβων και το μέγεθος της αλυσίδας.
15. Κάθε μπλοκ πρέπει να αποτελείται τουλάχιστον από ένα hash, παρτίδες από πρόσφατα επικυρωμένες συναλλαγές και το hash του προηγούμενου μπλοκ.

[12]

1.7 Αναπτυξιακές τάσεις

Ας εξετάσουμε κάποια πρόσφατα αναπτυξιακά τάσεις και προοπτικές για το blockchain:

1. DeFi (Αξιοποίηση Χρηματοοικονομικών Εφαρμογών): Το blockchain έχει δημιουργήσει ένα ολόκληρο οικοσύστημα χρηματοοικονομικών εφαρμογών που γνωρίζει ως DeFi. Αυτές οι εφαρμογές επιτρέπουν την παροχή χρηματοοικονομικών υπηρεσιών, όπως δανεισμό, ανταλλαγή, συναλλαγές με παράγωγα και πολλά άλλα, χωρίς την ανάγκη για παραδοσιακούς χρηματοοικονομικούς οργανισμούς.

2. NFTs (Μη Αντικατάστατα Καλά): Το blockchain έχει επίσης προωθήσει την ανάπτυξη των μη αντικατάστατων καλών (NFTs).

   Τα NFTs είναι μοναδικά ψηφιακά αντικείμενα που μπορούν να αντιπροσωπεύουν οποιοδήποτε είδος περιεχομένου, όπως έργα τέχνης, συλλεκτικά αντικείμενα, εικονογραφημένα αρχεία και πολλά άλλα. Το blockchain διασφαλίζει την αυθεντικότητα, την ιστορία και την ιδιοκτησία των NFTs.

3. Διάδραση του δημοσίου τομέα: Πολλές κυβερνήσεις και δημόσιες αρχές εξετάζουν το πώς το blockchain μπορεί να ενισχύσει την αποτελεσματικότητα, τη διαφάνεια και την ασφάλεια στον δημόσιο τομέα.
   Παραδείγματα περιλαμβάνουν την αναλογική καταγραφή ψήφων, τη διαχείριση των δημόσιων καταγραφών και τη βελτιωμένη διαχείριση των πόρων.

4. Προστασία δεδομένων και ιδιωτικότητα: Ορισμένα blockchain δίκτυα έχουν αναπτύξει μηχανισμούς για την προστασία των δεδομένων και την εκτέλεση προγραμμάτων με ιδιωτικότητα. Αυτό επιτρέπει την ασφαλή και εμπιστευτική ανταλλαγή πληροφοριών χωρίς να αποκαλύπτονται σε όλους οι λεπτομέρειες των συναλλαγών.

Αυτές είναι μερικές μόνο από τις πρόσφατες τάσεις και προοπτικές στον τομέα του blockchain. Η τεχνολογία συνεχίζει να εξελίσσεται και να παρέχει νέες δυνατότητες, ενώ επιδρά σε διάφορους τομείς της οικονομίας και της κοινωνίας. [1]

1.8 Κύριες Προκλήσεις

Ας εξετάσουμε μερικές από τις κύριες προκλήσεις που αντιμετωπίζει το blockchain και ορισμένες τάσεις που επηρεάζουν τον τομέα:

1. Κλιμακωσιμότητα: Το blockchain αντιμετωπίζει προκλήσεις σχετικά με την κλιμακωσιμότητα, δηλαδή την ικανότητά του να υποστηρίξει μεγάλους όγκους συναλλαγών.
   Η επεξεργαστική ικανότητα και οι απαιτήσεις αποθήκευσης μπορούν να αποτελέσουν πρόβλημα σε κάποια blockchain δίκτυα, και προσπάθειες γίνονται για τη βελτίωση της κλιμακωσιμότητας μέσω της ανάπτυξης νέων πρωτοκόλλων και τεχνολογιών.

2. Ασφάλεια: Η ασφάλεια είναι ένας σημαντικός παράγοντας για το blockchain, καθώς το σύστημα πρέπει να προστατεύει τις συναλλαγές και τα δεδομένα από κακόβουλες επιθέσεις.
   Είναι σημαντικό να αναπτύσσονται μηχανισμοί ασφάλειας, όπως κρυπτογραφία και μηχανισμοί συναίνεσης, για να διασφαλίζεται η ακεραιότητα του συστήματος.

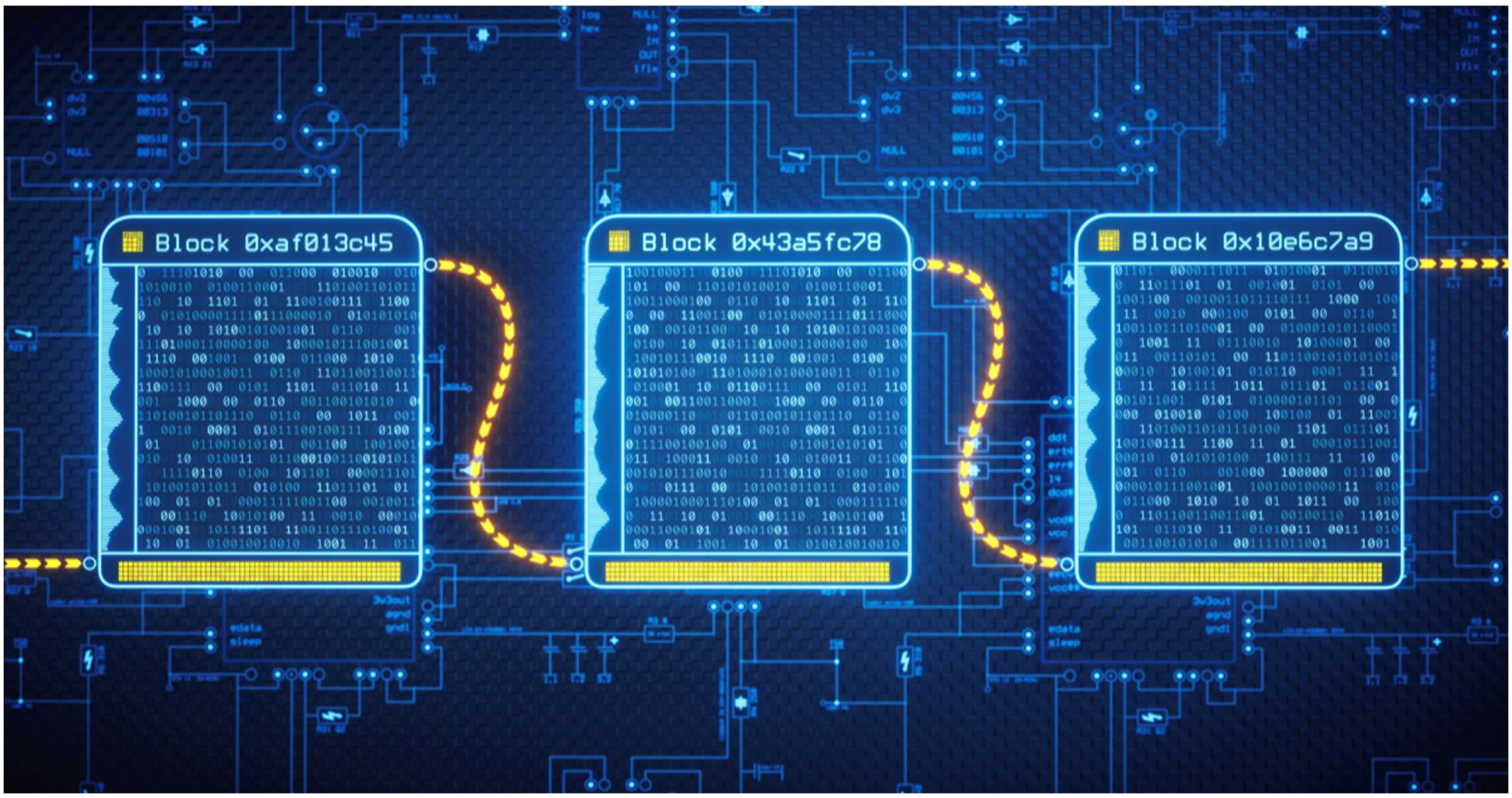
3. Νομικό πλαίσιο και ρύθμιση: Η τεχνολογία blockchain έχει νέες νομικές και ρυθμιστικές προκλήσεις. Οι υπάρχουσες νομικές δομές συχνά δεν είναι προσαρμοσμένες για τις ειδικές απαιτήσεις του blockchain.
   Επομένως, οι ρυθμιστικές αρχές ανά τον κόσμο εξετάζουν τη θέσπιση νέων νόμων και κανονισμών για να αντιμετωπίσουν τα θέματα αυτά και να προστατεύσουν τους χρήστες του blockchain.

4. Περιβάλλοντας ζητήματα: Οι διαδικασίες εξόρυξης κρυπτονομισμάτων (mining) σε ορισμένα blockchain δίκτυα μπορεί να επηρεάσουν το περιβάλλον, καθώς απαιτούν υψηλή κατανάλωση ενέργειας.

Η ανάπτυξη πιο βιώσιμων και περιβαλλοντικά φιλικών τεχνολογιών είναι μια τάση που έχει εμφανιστεί πρόσφατα, με σκοπό τη μείωση των επιπτώσεων στο περιβάλλον.

Αυτές είναι μερικές από τις προκλήσεις και τάσεις που επηρεάζουν τον τομέα του blockchain. Καθώς η τεχνολογία συνεχίζει να αναπτύσσεται, αναμένεται να εμφανιστούν νέες προκλήσεις και ευκαιρίες που θα επηρεάσουν το μέλλον του blockchain.

Συνοψίζοντας, η τεχνολογία του blockchain έχει εμφανιστεί ως μια μεταστροφική δύναμη με το δυναμικό να επανασχεδιάσει διάφορους κλάδους και τομείς. Η αποκεντρωμένη και διαφανής φύση του παρέχει νέες ευκαιρίες για ασφαλείς και αποτελεσματικές συναλλαγές, διαχείριση δεδομένων και δημιουργία εμπιστοσύνης μεταξύ των συμμετεχόντων. Παρά τις προκλήσεις που αντιμετωπίζει, όπως η κλιμακωσιμότητα, η ασφάλεια και οι ρυθμιστικές προκλήσεις, οι συνεχείς εξελίξεις διαμορφώνουν το μέλλον του blockchain. [1]



Εικόνα 1.8 Σύνδεση μεταξύ των Blocks. Διαθέσιμο στο διαδίκτυο.

Καθώς προχωρούμε, είναι σημαντικό να προάγουμε τη συνεργασία μεταξύ των φορέων της αγοράς, των ρυθμιστικών αρχών και των καινοτόμων, προκειμένου να αξιοποιήσουμε πλήρως το δυναμικό της τεχνολογίας του blockchain.

Με την αντιμετώπιση των περιορισμών της και την αξιοποίηση των δυνατοτήτων της, μπορούμε να ανοίξουμε νέες προοπτικές για την οικονομική συμμετοχή, τη διαχείριση της ψηφιακής ταυτότητας, την βελτιστοποίηση των αλυσίδων εφοδιασμού και την ανανέωση των μοντέλων επιχειρήσεων.

Η τεχνολογία blockchain προκύπτει από ένα δίκτυο ανθρώπων που δημιουργούν και μοιράζονται κάτι κοινό. Το κύριο χαρακτηριστικό αυτού του δικτύου, που χρησιμοποιείται στην τεχνολογία της αλυσίδας και στη γλώσσα των υπολογιστών, ανήκει στην κατηγορία δικτύων ομότιμων κόμβων.

Το δίκτυο είναι αποκεντρωμένο και διανεμημένο ισόποσα. Αυτό σημαίνει ότι δεν υπάρχει κάποιο πρόσωπο στο δίκτυο που να υπερέχει έναντι κάποιου άλλου προσώπου κατά οποιονδήποτε τρόπο, οπότε δεν υπάρχει προτεραιότητα μεταξύ των προσώπων.

Τα πρόσωπα που συμμετέχουν στο δίκτυο δεν είναι ίδια, αλλά είναι ίσα μεταξύ τους αναφορικά με οποιαδήποτε διαδικασία εκλογής ή επιλογής μεταξύ τους. Στατιστικά, εάν τεθεί θέμα εκλογής κάποιου προσώπου, η εκλογή αυτή θα δίνει ίσα ποσοστά επιτυχίας σε κάθε ένα από αυτά τα πρόσωπα και θα εκτελείται τυχαία.

Όλα τα πρόσωπα στο δίκτυο blockchain δημιουργούν και μοιράζονται από κοινού ένα κοινό αρχείο. Η διαδικασία δημιουργίας και διατήρησης αυτού του αρχείου καθορίζεται και ελέγχεται από ένα Σύνταγμα κανόνων, που ονομάζεται Πρωτόκολλο Συναίνεσης.

Αυτοί οι κανόνες διατυπώνονται με τη βασική αρχή της εξαιρετικής ανάγκης για εμπιστοσύνη ανάμεσα σε αυτά τα άτομα.

Αυτό που σημαίνει η παραπάνω πρόταση αναφέρεται στην ανάγκη απόδειξης εμπιστοσύνης ανάμεσα στα άτομα του δικτύου μόνο σε εξαιρετικές περιπτώσεις και προτιμητέον καμία περίπτωση καθόλου.

Με τη σύνταξη ενός συνοπτικού Πρωτοκόλλου Συναίνεσης εξαλείφεται η δημιουργία συνθηκών που θα απαιτούσαν από τα άτομα να αποδείξουν την ακεραιότητά τους όσον αφορά τη συμμετοχή τους στο δίκτυο και, κατά συνέπεια, το δικαίωμά τους να συνυπάρχουν σε αυτό. [1]

1.9 Ο ρόλος των δεδομένων στις αλυσίδες κοινοποιήσεων και η λειτουργία τους σε ένα blockchain

Η φύση των δεδομένων σε μια αλυσίδα κοινοποιήσεων καθορίζεται από την ύπαρξη ή όχι συγκεκριμένης λειτουργίας των δεδομένων. Υπάρχουν τα σημαντικά δεδομένα που εγγυώνται την απρόσκοπτη λειτουργία ενός blockchain και τα εννοιολογικά δεδομένα που είναι αξιόλογα και χρήσιμα για τους κατόχους τους.

Στο Bitcoin, τα δεδομένα περιγράφουν την κατάσταση ενός τραπεζικού χαρτοφυλακίου. Με εφαρμογές όπως η πιστοποίηση προϊόντων, η διαφύλαξη εγγράφων και η σύνταξη συμβολαίων, οι αλυσίδες κοινοποιήσεων έχουν πολλούς ρόλους και αρμοδιότητες.

Στον τομέα της blockchain, τα άτομα έχουν αρμοδιότητες όπως η κατοχή και ενημέρωση των δεδομένων και το δικαίωμα υποβολής νέων μπλοκ.

Οι κόμβοι διακρίνονται σε απλούς κόμβους και διαχειριστές-κόμβους.

Οι διαχειριστές-κόμβοι μπορεί να είναι εξορύχοι ή επικυρωτές, ανάλογα με τη μέθοδο απόδειξης που χρησιμοποιούν.

Οι ρόλοι και οι αρμοδιότητες στο πλαίσιο μιας blockchain ποικίλουν ανάλογα με το συγκεκριμένο πρωτόκολλο και τον τύπο της αλυσίδας. Συνήθως, υπάρχουν κόμβοι που διακρατούν το μερίδιο των δεδομένων της αλυσίδας, ενώ άλλοι κόμβοι λειτουργούν ως διαχειριστές-κόμβοι ή εξορύχοι (miners) για την εξόρυξη νέων μπλοκ ή την επικύρωση συναλλαγών.

Όσον αφορά τη δομή των μπλοκ, κάθε μπλοκ περιέχει συνήθως τις εξής πληροφορίες:

a) α) Ένα κρυπτογραφικό αποτύπωμα του προηγούμενου μπλοκ στην αλυσίδα, το οποίο συνδέει το μπλοκ με τα προηγούμενα μπλοκ και εξασφαλίζει την ακεραιότητα της αλυσίδας. Αυτό το δεδομένο ανήκει στην κατηγορία των σημαντικών δεδομένων.

b) Ένα κρυπτογραφικό αποτύπωμα των συναλλαγών που περιέχονται στο μπλοκ. Αυτό το αποτύπωμα προκύπτει από τη λειτουργία μιας συνάρτησης κατακερματισμού πάνω στις συναλλαγές και εξασφαλίζει την ακεραιότητα των δεδομένων. Οι συναλλαγές ανήκουν στην κατηγορία των Χρήσιμων Δεδομένων, ενώ το κρυπτογραφικό τους αποτύπωμα ανήκει στην κατηγορία των Σημαντικών Δεδομένων.

   Τα δεδομένα αυτά εντάσσονται στο μπλοκ σε μια διάταξη που στη γλώσσα των υπολογιστών ονομάζεται δέντρο Merkle (Merkle tree), και θυμίζει ένα ανεστραμμένο δέντρο. Η κορυφή του ανεστραμμένου αυτού δέντρου ονομάζεται ρίζα Merkle (Merkle root). Η ρίζα Merkle λοιπόν προκύπτει και περιέχει πληροφοριακά το σύνολο των ενταγμένων στο μπλοκ, συναλλαγών.
   Να σημειωθεί ότι κάθε Κόμβος που λαμβάνει μέρος στη διαδικασία υποβολής νέου μπλοκ προς ένταξη στην αλυσίδα blockchain, δεν συμπεριλαμβάνει απαραίτητα τα ίδια χρήσιμα δεδομένα. Προϋπόθεση για τη συμπερίληψη μιας συναλλαγής σε έναν υποψήφιο προς ένταξη στην αλυσίδα, μπλοκ, είναι η παρουσία των δεδομένων αυτών στον διαχειριστή-κόμβο καθώς σχετίζεται με το επισυναπτόμενο σε αυτήν, κόμιστρο (transaction fee).
   Το κόμιστρο είναι κλάσμα του ψηφιακού νομίσματος που χρησιμοποιεί η εκάστοτε αλυσίδα. Όσο υψηλότερο είναι το επισυναπτόμενο από τον χρήστη της συναλλαγής κόμιστρο, τόσο υψηλότερα θα βρίσκεται στην λίστα συμπερίληψης συναλλαγών, αυξάνοντας τις πιθανότητες συντομότερης κοινοποίησης.

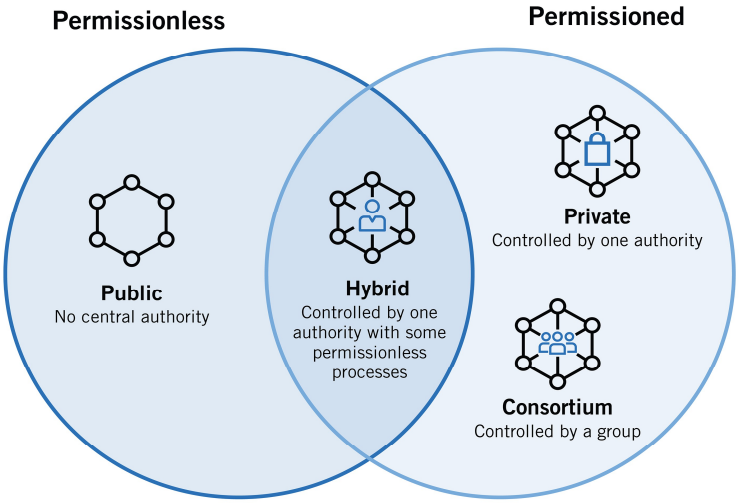
c) Τη χρονική του σφραγίδα ή χρόνο-σφραγίδα (timestamp). Πρόκειται για τον αριθμό των δευτερολέπτων που έχουν μεσολαβήσει από την 1η Ιανουαρίου του 1970 μέχρι την στιγμή της παραγωγής της χρονο-σφραγίδας.

d) Την απόδειξη εργασίας του μπλοκ (cryptographic nonce). Πρόκειται για έναν μοναδικά δημιουργημένο αριθμητικό λέξημα. Είναι μια σειρά σημαντικών δεδομένων άπαξ παραχθέντος με τυχαία διεργασία από τον κόμβο-διαχειριστή. Τα δεδομένα ενός αριθμητικού λεξήματος σε ένα μπλοκ που τηρεί τις προϋποθέσεις ένταξης σε μια αλυσίδα blockchain έχουν την εξής ιδιότητα.

   Η λειτουργία μιας συνάρτησης κατακερματισμού στο σύνολο των δεδομένων των προηγούμενων τριών ομάδων ενός μπλοκ συμπεριλαμβανομένου και του αριθμητικού λεξήματος αυτού, δημιουργεί την τελευταία ομάδα δεδομένων του μπλοκ.Σε περίπτωση χρήσης του αλγορίθμου απόδειξης μερίσματος χρησιμοποιείται συνήθως μία ή περισσότερες ψηφιακές υπογραφές που αποδεικνύουν ότι το μπλοκ έχει παραχθεί από τον/τους χρήστες που επέλεξε το δίκτυο για την εργασία αυτή με βάση τα μερίσματα που κατέχουν.

e) Κρυπτογραφικό αποτύπωμα του εν λόγω μπλοκ. Όπως και στην περίπτωση της ομάδας (α) το αποτύπωμα αυτό καθορίζει τη μοναδική ταυτότητα του μπλοκ αυτού. Να σημειωθεί ότι εάν ένα υποψήφιο προς ένταξη στην αλυσίδα blockchain, μπλοκ, πετύχει να ενταχθεί σε αυτήν, το κρυπτογραφικό του αποτύπωμα θα αποτελέσει την ομάδα (α) του επόμενου από αυτό, μπλοκ. [1]

1.10 Κατηγορίες Blockchain



Εικόνα 1.10 4 Κατηγορίες των Blockchain. Διαθέσιμο στο διαδίκτυο.

Τα δίκτυα blockchain μπορούν να κατηγοριοποιηθούν ανάλογα με το μοντέλο άδειας που χρησιμοποιούν. Σε ένα ελεύθερο δίκτυο (permissionless), οποιοσδήποτε έχει την ικανότητα να εκδώσει ένα νέο μπλοκ χωρίς να απαιτείται ειδική άδεια. Από την άλλη πλευρά, σε ένα εξουσιοδοτημένο δίκτυο (permissioned), μόνο επιλεγμένοι κόμβοι έχουν την άδεια να εκδίδουν νέα μπλοκ.[13]

Υπάρχουν επίσης τα δημόσια (public) και ιδιωτικά (private) blockchain. Στα δημόσια blockchain, οποιοσδήποτε μπορεί να συνδεθεί χωρίς να υπάρχει κάποιος έλεγχος. Από την άλλη πλευρά, στα ιδιωτικά blockchain, η πρόσβαση απαιτεί πρόσκληση ή την ικανοποίηση ορισμένων προϋποθέσεων.

Είναι σημαντικό να σημειωθεί ότι ένα δίκτυο blockchain μπορεί να συνδυάζει χαρακτηριστικά από διάφορες κατηγορίες. Δηλαδή, ένα blockchain μπορεί να είναι ταυτόχρονα ελεύθερο και δημόσιο, ή ελεύθερο και ιδιωτικό. Η επιλογή του μοντέλου άδειας και του επιπέδου προσβασιμότητας επηρεάζει τόσο τις συνθήκες για την έκδοση μπλοκ όσο και τη δυνατότητα σύνδεσης στο δίκτυο.

1.10.1 Ελεύθερα δίκτυα(Permissionless)

Τα αποκεντρωμένα δίκτυα blockchain χωρίς άδεια αναφέρονται σε συστήματα καθολικού χαρακτήρα που επιτρέπουν σε οποιονδήποτε να δημοσιεύει blocks χωρίς την απαίτηση άδειας. Συνήθως, αυτά τα συστήματα είναι ανοιχτού κώδικα και διατίθενται δωρεάν για λήψη.

Λόγω του γεγονότος ότι όλοι έχουν τη δυνατότητα να δημοσιεύουν block, οποιοσδήποτε μπορεί να αποκτήσει πρόσβαση στο blockchain και να πραγματοποιήσει συναλλαγές, συμπεριλαμβανομένων και των συναλλαγών σε ήδη δημοσιευμένα blocks.[13]

Μέσα σε ένα αποκεντρωμένο δίκτυο blockchain χωρίς άδεια, όλοι οι χρήστες έχουν τη δυνατότητα να διαβάζουν και να γράφουν στο καθολικό του.

Εξαιτίας του γεγονότος ότι αυτά τα δίκτυα είναι διαθέσιμα σε όλους, κακόβουλα άτομα μπορεί να προσπαθήσουν να εκμεταλλευτούν το σύστημα δημοσιεύοντας blocks με μη εξουσιοδοτημένο τρόπο.

Προκειμένου να αποτραπεί αυτή η κατάσταση, τα δίκτυα blockchain χωρίς άδεια συνήθως χρησιμοποιούν μ ια πολυμερή συμφωνία ή προσέγγιση "συναίνεσης". Αυτό σημαίνει ότι οι χρήστες πρέπει να επενδύουν (PoW) ή να διατηρούν πόρους (PoS) κατά τη δημοσίευση των μπλοκ. Αυτός ο μηχανισμός καθιστά δύσκολη την υπονόμευση του συστήματος από κακόβουλους χρήστες.

Παραδείγματα τέτοιων συναινετικών προσεγγίσεων είναι η απόδειξη εργασίας (PoW) και η απόδειξη συμμετοχής (PoS). Μέσω αυτών των μηχανισμών, τα δίκτυα blockchain χωρίς άδεια ενθαρρύνουν τη μη κακόβουλη δραστηριότητα, ανταμείβοντας τους δημιουργούς μπλοκ που συμμορφώνονται με το πρωτόκολλο, με εγγενή νομίσματα.[13]

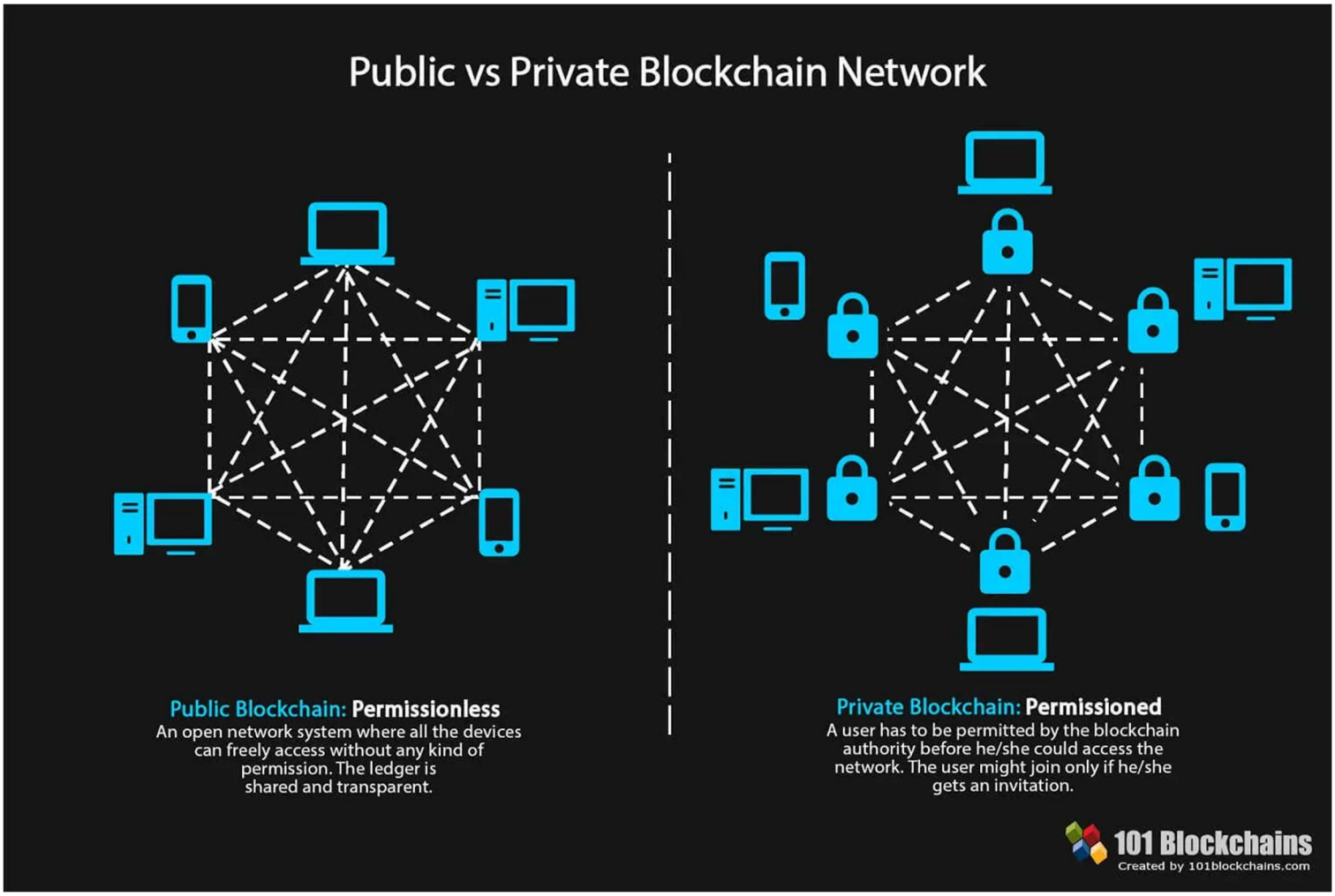
1.10.2 Εξουσιοδοτημένα Δίκτυα (Permissioned)

Τα εξουσιοδοτημένα δίκτυα blockchain είναι δίκτυα όπου οι χρήστες που δημοσιεύουν τα μπλοκ πρέπει να έχουν εγκριθεί από ένα εξουσιοδοτημένο μέλος. Αυτό είτε μπορεί να είναι ένα κεντρικό φορέα, είτε ένα αποκεντρωμένο σύνολο χρηστών. Με την εξουσιοδότηση των χρηστών, μπορεί να ελεγχθεί η πρόσβαση στο δίκτυο, καθώς και να οριστούν τα δικαιώματα έκδοσης συναλλαγών.[13]

Ένα εξουσιοδοτημένο δίκτυο blockchain μπορεί να επιτρέπει την προβολή του blockchain σε οποιονδήποτε ή να περιορίζει την πρόσβαση μόνο σε εξουσιοδοτημένα άτομα. Το λογισμικό του δικτύου μπορεί να είναι είτε ανοιχτού είτε κλειστού κώδικα. [13]

Τα εξουσιοδοτημένα δίκτυα blockchain μπορούν να παρέχουν την ίδια ιχνηλασιμότητα των ψηφιακών στοιχείων όπως τα ανεξάρτητα δίκτυα blockchain, καθώς και έναν κατανεμημένο και αξιόπιστο μηχανισμό αποθήκευσης δεδομένων. Χρησιμοποιούν επίσης μοντέλα συναίνεσης για την προσθήκη των μπλοκ, αλλά οι προσεγγίσεις μπορεί να διαφέρουν ανάλογα με το εξουσιοδοτημένο σύστημα που χρησιμοποιείται. [13]

Τα εξουσιοδοτημένα δίκτυα blockchain χρησιμοποιούνται σε πολλές εφαρμογές, όπως η χρηματοοικονομία, η υπηρεσία υγείας, η προμήθεια αλυσίδας, η κατανεμημένη διαχείριση της ενέργειας και πολλές άλλες. Η εξουσιοδοτημένη φύση των δικτύων αυτών προσφέρει επιπλέον επίπεδα ασφάλειας και ελέγχου, αλλά μπορεί επίσης να εισαγάγει περιορισμούς στην ευελιξία και την αποκεντρωμένη φύση του blockchain. [13]



Εικόνα 1,10,2 Ιδιωτικά & Δημόσια Δίκτυα. Διαθέσιμο στο Διαδίκτυο.

1.10.3 Δημόσια Δίκτυα (Public)

Τα δημόσια δίκτυα blockchain είναι ανοιχτά δίκτυα που επιτρέπουν σε οποιονδήποτε χρήστη να συμμετέχει και να συναλλαγεί στο δίκτυο χωρίς περιορισμούς ή έλεγχους. Αυτό σημαίνει ότι οποιοσδήποτε μπορεί να δημιουργήσει έναν λογαριασμό και να συναλλαγεί με άλλους χρήστες χωρίς να απαιτείται άδεια ή εξουσιοδότηση.

Τα δημόσια δίκτυα blockchain προσφέρουν αποκέντρωση, διαφάνεια και ανοικτή πρόσβαση σε όλους. Οι συναλλαγές και οι εγγραφές αποθηκεύονται δημόσια και είναι προσβάσιμες από όλους τους συμμετέχοντες στο δίκτυο.

Η ασφάλεια του δικτύου εξασφαλίζεται μέσω της κρυπτογραφίας και του μηχανισμού της συναίνεσης (consensus), όπως το Proof of Work ή το Proof of Stake.

Παραδείγματα δημοφιλών δημόσιων blockchain είναι το Bitcoin, το Ethereum, το Litecoin και το Ripple. Αυτά τα δίκτυα είναι ανοιχτά σε οποιονδήποτε ενδιαφερόμενο χρήστη και χρησιμοποιούνται για διάφορες εφαρμογές, όπως η κρυπτονομισματική, οι έξυπνες συμβάσεις και η αποθήκευση αξιών.

1.10.4 Ιδιωτικά Δίκτυα (Private)

Τα ιδιωτικά δίκτυα blockchain λειτουργούν με διαφορετικό τρόπο από τα δημόσια δίκτυα. Στα ιδιωτικά δίκτυα, η πρόσβαση και η συμμετοχή στο δίκτυο είναι περιορισμένες και απαιτούν την έγκριση ή την πρόσκληση από τα υπάρχοντα μέλη.

Αυτό επιτρέπει στους συμμετέχοντες να διατηρούν έναν έλεγχο και μια επιτήρηση του δικτύου και των δεδομένων τους.

Ένα ιδιωτικό δίκτυο blockchain συνήθως χρησιμοποιείται από επιχειρήσεις ή οργανισμούς που έχουν ειδικές απαιτήσεις ασφάλειας, εμπιστευτικότητας και ελέγχου. Οι συμμετέχοντες μπορούν να ορίσουν ποιοι έχουν πρόσβαση στο δίκτυο και στις πληροφορίες τους, και μπορούν να εφαρμόζουν προηγμένες μεθόδους αυθεντικοποίησης και κρυπτογράφησης για την προστασία των δεδομένων.

Ένα ιδιωτικό δίκτυο blockchain μπορεί να επιτρέπει στις επιχειρήσεις να διαχειρίζονται και να επικυρώνουν συναλλαγές μεταξύ των συμμετεχόντων με αυξημένη αποτελεσματικότητα και διαφάνεια. Επίσης, μπορεί να παρέχει τη δυνατότητα εφαρμογής έξυπνων συμβολαίων που εκτελούνται αυτόματα με βάση συγκεκριμένους κανόνες και συνθήκες.

Οι ιδιωτικές blockchain λύσεις είναι επίσης ελκυστικές για οργανισμούς που χρειάζονται να συνεργάζονται και να μοιράζονται δεδομένα μεταξύ τους, χωρίς να αποκαλύπτουν τις πλήρεις λεπτομέρειες των συναλλαγών σε τρίτους.

Συνολικά, τα ιδιωτικά δίκτυα blockchain προσφέρουν έναν τρόπο να αξιοποιηθούν οι πλεονεκτήματα της τεχνολογίας blockchain, όπως η ασφάλεια, η αυτονομία και η αξιοπιστία, ενώ ταυτόχρονα επιτρέπουν τον έλεγχο και την προστασία των ευαίσθητων δεδομένων.

1.11 Τρόποι πιστοποίησης και Αλγόριθμοι

Όπως γνωρίζουμε ήδη, το blockchain είναι ένα αποκεντρωμένο δίκτυο που προσφέρει τη δυνατότητα εμπιστοσύνης μεταξύ των συμμετεχόντων χωρίς την ανάγκη τρίτων μερών για την επίτευξη και την ασφάλεια του συστήματος.

Για να επιτευχθεί αυτό, το blockchain χρησιμοποιεί αλγορίθμους συναίνεσης. Η διαδικασία της συναίνεσης βασίζεται στην ιδέα της πραγματοποίησης συχνών ασφαλών αλλαγών στο κατανεμημένο καθολικό.

Η συγχρονισμένη λειτουργία των κόμβων αποτελεί μια κρίσιμη προσέγγιση που διασφαλίζει την ύπαρξη και την εκτέλεση κοινών καταστάσεων σύμφωνα με προκαθορισμένους κανόνες για την αλλαγή της κατάστασης του blockchain.

Λόγω του γεγονότος ότι η κατάσταση κατανέμεται σε πολλά αντίγραφα εντός του δικτύου, η εκτέλεση της κατάστασης θα οδηγήσει στην τελική παραγωγή ισοδύναμων αποτελεσμάτων.

Επομένως, τα αντίγραφα πρέπει να επικοινωνούν και να συμφωνούν για πιθανές ενημερώσεις της κατάστασης, χρησιμοποιώντας μια τεχνική συναίνεσης.

Ωστόσο, η ενσωμάτωση της συναίνεσης σε ένα αποκεντρωμένο σύστημα αποτελεί πρόκληση, λόγω της ανάγκης για ένα μοντέλο συναίνεσης που να είναι ανθεκτικό σε αντιξοότητες, ανοχή σε αποτυχίες, διαχείριση δικτύου, χειρισμός καθυστερήσεων και άλλες κρίσιμες ιδιότητες.[29]

Επιπλέον, ένας αλγόριθμος συναίνεσης πρέπει να περιλαμβάνει μέτρα ασφαλείας, όπως η επιτήρηση κακόβουλων κόμβων μέσω της εφαρμογής κανόνων, όπως η συγχρονισμένη αποστολή μηνυμάτων.

Η συναίνεση στα κατανεμημένα συστήματα, συμπεριλαμβανομένης της τεχνολογίας blockchain, είναι η διατήρηση τριών θεμελιωδών χαρακτηριστικών που εξασφαλίζουν την αποτελεσματικότητα του δικτύου.

Η συναίνεση δεν αφορά μόνο τη διατήρηση μιας συνεπούς παγκόσμιας κατάστασης συμφωνημένης σε ένα αποκεντρωμένο καθολικό, αλλά επίσης διασφαλίζει την ασφάλεια, τη λειτουργικότητα και την ανοχή σε σφάλματα του δικτύου.

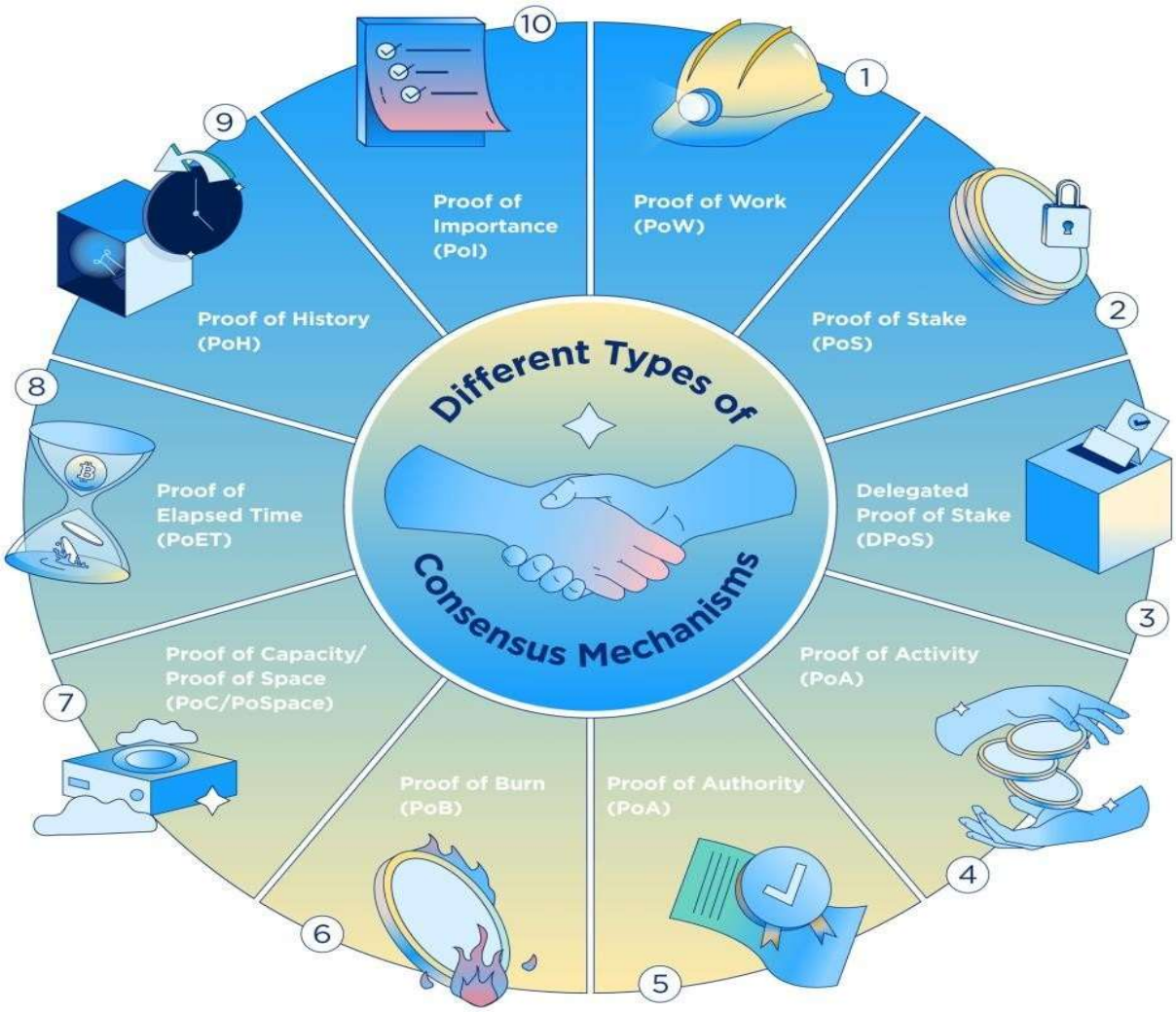
Κατά συνέπεια, οι αλγόριθμοι συναίνεσης μπορούν να αξιολογηθούν βάσει αυτών των χαρακτηριστικών.

Το μοντέλο συναίνεσης μπορεί να διατηρήσει την ασφάλεια, εξασφαλίζοντας ότι όλοι οι κόμβοι θα παράγουν πανομοιότυπα, συνεχή και νόμιμα αποτελέσματα. Η "Liveness" είναι η ικανότητα ενός μοντέλου συναίνεσης να καθοδηγεί τη συμβολή των κόμβων χωρίς σφάλματα κατά τη δημιουργία αξίας.

Για να επιτευχθεί ένας συγκεκριμένος βαθμός ανοχής σε σφάλματα, ο συναινετικός αλγόριθμος πρέπει να είναι σε θέση να αντιμετωπίζει τυχόν αποτυχίες των συμμετεχόντων κόμβων και να ανακάμπτει από αυτές.

Εν κατακλείδι, η επίτευξη συναίνεσης σε ένα κατανεμημένο σύστημα είναι πολύπλοκη διαδικασία, καθώς απαιτεί την αντιμετώπιση πολλών προκλήσεων, όπως η αντιμετώπιση αντιξοοτήτων, η ανοχή σε αποτυχίες και η διαχείριση καθυστερήσεων.

Οι αλγόριθμοι συναίνεσης πρέπει να σχεδιαστούν με προσεκτικό τρόπο για να εξασφαλίσουν την ασφάλεια, την αποτελεσματικότητα και την ανοχή σε σφάλματα του συστήματος.



Εικόνα 1.11 Διάφορα δημοφιλή είδη Αλγόριθμων Συναίνεσης. Διαθέσιμο στο διαδίκτυο

1.11.1 Proof of Stake (PoS)

Proof of Stake (PoS) είναι ένας μηχανισμός συναίνεσης που χρησιμοποιείται στην τεχνολογία του blockchain για την επιβεβαίωση και την ασφάλεια του δικτύου. Σε αντίθεση με το Proof of Work (PoW), όπου η επιλύση υπολογιστικώς απαιτητικών προβλημάτων προσφέρει συναίνεση, στο PoS η συναίνεση βασίζεται στην κατοχή και τη στάθμιση κρυπτονομισμάτων.

Στο PoS, οι συμμετέχοντες του δικτύου, γνωστοί ως "validators" ή "forgers", δεν ανταγωνίζονται μεταξύ τους για την επίλυση ενός υπολογιστικού προβλήματος.

Αντ' αυτού, η επιλογή του επόμενου block γίνεται βάσει του ποσοστού συνολικών κρυπτονομισμάτων που κατέχει ο κάθε validator.

Συγκεκριμένα, οι validators καταθέτουν ένα ποσό κρυπτονομίσματος ως "ποντίκι" (stake) και αναλαμβάνουν το ρόλο της επιβεβαίωσης των συναλλαγών. Η πιθανότητα να επιλέξουν τον επόμενο block για προσθήκη στο blockchain εξαρτάται από το μέγεθος του stake τους. Με άλλα λόγια, όσο μεγαλύτερο είναι το stake ενός validator, τόσο μεγαλύτερη είναι η πιθανότητά του να επιλέξει τον επόμενο block.

Ένα σημαντικό χαρακτηριστικό του PoS είναι η εξοικονόμηση ενέργειας. Επειδή δεν απαιτείται υπολογιστική ισχύς για την επίλυση προβλημάτων, η κατανάλωση ενέργειας στο PoS είναι πολύ χαμηλότερη σε σύγκριση με το PoW.

Αυτό καθιστά το PoS μια εναλλακτική λύση για την εξόρυξη κρυπτονομισμάτων με πιο βιώσιμο περιβάλλοντικό αποτύπωμα.

Ωστόσο, ο PoS είναι επίσης επιρρεπής σε επιθέσεις "51% attack". Αν ένας validator κατέχει πάνω από το 51% του συνόλου των κρυπτονομισμάτων, μπορεί να επηρεάσει τη λειτουργία του δικτύου.

Γι' αυτό το λόγο, ορισμένα συστήματα PoS χρησιμοποιούν διάφορους μηχανισμούς για την αντιμετώπιση αυτού του κινδύνου, όπως η τυχαιοποίηση της επιλογής των validators ή η επιβολή ποινών σε περίπτωση κακής συμπεριφοράς.

Συνολικά, το PoS προσφέρει μια εναλλακτική μέθοδο συναίνεσης για τη λειτουργία των blockchain. Βασίζεται στην κατοχή κρυπτονομισμάτων αντί για υπολογιστική ισχύ και προσφέρει πλεονεκτήματα όπως εξοικονόμηση ενέργειας και μειωμένο κόστος.

Ωστόσο, χρειάζεται προσεκτική σχεδίαση και ασφάλεια για την αντιμετώπιση πιθανών επιθέσεων.[15][16]

1.11.2 Proof of Stake Time (PoST)

Γενικά, το Proof of Stake (PoS) είναι ένας μηχανισμός συναίνεσης στο blockchain, όπου οι συναλλαγές επαληθεύονται και νέα μπλοκ προστίθενται στο blockchain από τους validators (επαληθευτές).

Αντί να βασίζεται στην υπολογιστική ισχύ, όπως στο Proof of Work (PoW), το PoS βασίζεται στην κατοχή κρυπτονομισμάτων.

Οι validators σε ένα σύστημα PoS επιλέγονται για να επιβεβαιώσουν συναλλαγές και να παράγουν νέα μπλοκ βάσει του ποσού κρυπτονομίσματος που κατέχουν και κατατίθενται ως εγγύηση (stake).

Η επιλογή των validators μπορεί να γίνει με τυχαιοποιημένο τρόπο ή με βάση το ποσό της εγγύησης τους.

Ένα σημαντικό πλεονέκτημα του PoS είναι η εξοικονόμηση ενέργειας σε σύγκριση με το PoW, καθώς δεν απαιτείται επίλυση πολύπλοκων υπολογιστικών προβλημάτων.

Επιπλέον, μειώνεται η ανάγκη για εξειδικευμένο εξοπλισμό (όπως γραφικές κάρτες) που συχνά απαιτείται για την εξόρυξη (mining) κρυπτονομισμάτων στο PoW.

Ωστόσο, το PoS έχει και ορισμένες προκλήσεις, όπως το πρόβλημα της "κατοχής της πλειοψηφίας" (the "nothing at stake" problem) και της "επίθεσης με μοίρα" (the "nothing at stake" attack).

Για την αντιμετώπιση αυτών των προκλήσεων, έχουν προταθεί διάφορες τεχνικές και πρωτόκολλα, όπως το Casper και το Ouroboros.

Σε συνοπτικές γραμμές, το Proof of Stake είναι ένας μηχανισμός συναίνεσης στο blockchain που βασίζεται στην κατοχή κρυπτονομίσματος αντί της υπολογιστικής ισχύος. Αυτός ο μηχανισμός επιτρέπει την επαλήθευση και την προσθήκη νέων μπλοκ στο blockchain, ενώ παρέχει πλεονεκτήματα όπως εξοικονόμηση ενέργειας και μειωμένο κόστος.[17][18]

1.11.3 Proof of Work (PoW)

Proof of Work (PoW) είναι ένας μηχανισμός συναίνεσης που χρησιμοποιείται στην τεχνολογία του blockchain για την επαλήθευση και την επιβεβαίωση των συναλλαγών και της ασφάλειας του δικτύου.

Ο PoW έχει εισαχθεί για πρώτη φορά από τον Satoshi Nakamoto, δημιουργό του Bitcoin, και αποτελεί έναν κρίσιμο πυλώνα για τη λειτουργία του δικτύου.

Ο σκοπός του PoW είναι να αποτρέπει κακόβουλες επιθέσεις και να διασφαλίσει ότι οι συναλλαγές που προστίθενται στο blockchain είναι έγκυρες και αμετάβλητες.

Για να επιτευχθεί αυτό, οι συμμετέχοντες του δικτύου, γνωστοί ως "miner" (ανθρώπινοι υπολογιστές ή ειδικοί υπολογιστικοί συστοιχίες), πρέπει να επιλύουν ένα πρόβλημα υπολογιστικής δυσκολίας.

Το πρόβλημα που πρέπει να επιλύσει ο miner στο PoW είναι η εύρεση μιας τιμής (nonce) που, σε συνδυασμό με τα δεδομένα της συναλλαγής και ενός προηγούμενου block, παράγει μια συγκεκριμένη κατανομή hash με συγκεκριμένο αριθμό μηδενικών στην αρχή.

Οι miners επιδιώκουν να επιλύσουν αυτό το πρόβλημα με δοκιμές και λάθη, χρησιμοποιώντας υπολογιστική ισχύ για να υπολογίσουν τη σωστή τιμή nonce.

Ο πρώτος miner που επιλύει το πρόβλημα και βρίσκει τη σωστή τιμή nonce, ανακαλύπτει ένα νέο block και το προσθέτει στο blockchain.

Ως ανταμοιβή για την εργασία του, λαμβάνει ένα ποσό κρυπτονομίσματος (όπως τα Bitcoin) και τις προμήθειες από τις συναλλαγές που περιέχονται στο block.

Ο PoW είναι σχεδιασμένος έτσι ώστε η επίλυση του προβλήματος να είναι πολύ υπολογιστικά απαιτητική και να απαιτεί πολύ χρόνο και πόρους.

Αυτό εξασφαλίζει ότι οι miners πρέπει να επενδύουν σημαντικούς πόρους για να επιλύσουν το πρόβλημα, προστατεύοντας έτσι το δίκτυο από κακόβουλες επιθέσεις. Τα δημοφιλέστερα blockchain που χρησιμοποιούν την συγκεκριμένη τεχνολογία είναι Bitcoin και Ethereum. [14]

Ωστόσο, ο PoW έχει και μειονεκτήματα, όπως την υψηλή κατανάλωση ενέργειας και τους υψηλούς χρόνους επιβεβαίωσης των συναλλαγών. Για αυτόν τον λόγο, άλλοι μηχανισμοί συναίνεσης, όπως το Proof of Stake (PoS), έχουν αναπτυχθεί για να αντιμετωπίσουν αυτά τα προβλήματα.

1.11.4 Proof of Burn (PoB)

Το Proof of Burn (PoB) είναι ένας μηχανισμός συναίνεσης που χρησιμοποιείται στο blockchain και βασίζεται στην καύση (burn) κρυπτονομίσματος ως ένδειξη δέσμευσης και συμμετοχής στο δίκτυο. Κατά το PoB, οι χρήστες καίνε (αποστέλλουν σε μια μη αναστρέψιμη διεύθυνση) μια ποσότητα κρυπτονομίσματος που ανήκει σε ένα συγκεκριμένο πρωτόκολλο ή δίκτυο.

Η καύση των κρυπτονομισμάτων αποτελεί μια απώλεια μόνιμης και αναστρέψιμης αξίας για τον αποστολέα. Με αυτόν τον τρόπο, ο αποστολέας δείχνει την προθυμία του να αφιερώσει τους πόρους του στο δίκτυο, καθιστώντας το PoB έναν μηχανισμό "κατανάλωσης" ή "καταναλωτή" κρυπτονομίσματος.

Μια από τις κύριες ιδέες πίσω από το PoB είναι ότι οι χρήστες που καίνε τα κρυπτονομίσματα έχουν μεγαλύτερη πιθανότητα να επιλεγούν ως validators για την επαλήθευση συναλλαγών και την παραγωγή νέων μπλοκ.

Ως αποτέλεσμα, αυτοί οι χρήστες αποκτούν δικαιώματα συναίνεσης ανάλογα με τηνποσότητα των κρυπτονομισμάτων που κάηκαν, παρουσιάζοντας έτσι το επίπεδο του ενδιαφέροντός τους και τη συμμετοχή τους στο δίκτυο.

Οι χαρακτηριστικές εφαρμογές του PoB περιλαμβάνουν την επιλογή validators, την απόδειξη επενδύσεων σε ένα έργο ή ένα πρωτόκολλο και την επιβράβευση των χρηστών που συνεισέφεραν στην ανάπτυξη του blockchain.

Συνολικά, το Proof of Burn είναι ένας μηχανισμός συναίνεσης που επιτρέπει στους χρήστες να αποδεικνύουν το ενδιαφέρον και τη συμμετοχή τους στο blockchain μέσω της καύσης κρυπτονομισμάτων, δημιουργώντας ένα οικονομικό κίνητρο για την ασφάλεια και την αποτελεσματική λειτουργία του δικτύου. [19][20]

1.11.5 Proof of Proof (PoP)

Το Proof of Proof (PoP) είναι ένας μηχανισμός συναίνεσης που επιτρέπει σε ένα νεοσύστατο blockchain, γνωστό ως blockchain κληρονομιάς ασφάλειας (Security Inheriting, SI), να κληρονομήσει μέτρα ασφαλείας από ένα άλλο blockchain, γνωστό ως blockchain Παροχής Ασφαλείας (Security Providing, SP), προκειμένου να επιτευχθεί επεκτασιμότητα.

Ο μηχανισμός κληρονομικότητας PoP είναι αποκεντρωμένος και δεν απαιτεί άδεια από το blockchain SP ή εξουσιοδότηση από κεντρικό δίκτυο ή ομοσπονδιακά ιδρύματα.

Ο PoP αποτρέπει τη συμμετοχή των χρηστών του blockchain SI που δεν συμβάλλουν στον εξορυκτικό διαγωνισμό του blockchain SP, καθώς απαιτεί από αυτούς να συμβάλλουν στη διατήρηση των εγγενών διακριτικών του blockchain.

Παράλληλα, η κληρονομικότητα δεν επιβάλλει τεχνικά ή μη τετριμμένα όρια στις αλυσίδες μπλοκ SI που υιοθετούν αυτό το πρωτόκολλο. Η κατάσταση που μεταδίδεται από το δίκτυο blockchain SP χρησιμοποιείται για την παροχή κινήτρων στο blockchain SI.[24][25][26]

1.11.6 Proof of History (PoH)

Το Proof of History (PoH) είναι ένας νέος μηχανισμός συναίνεσης που αναπτύχθηκε για τη χρήση σε blockchain δικτυκών συστημάτων. Η βασική ιδέα πίσω από το PoH είναι η χρήση μιας χρονικής σφραγίδας που αποδεικνύει τη σειρά και το χρονικό διάστημα μεταξύ των συμβάντων ή των μπλοκ σε ένα blockchain.

Στο PoH, υπάρχει ένας κεντρικός χρονοσφραγιστής (timestamp authority) που δημιουργεί και επαληθεύει μια σειρά από μοναδικές και ακατάλληλες για παραπληροφόρηση χρονικές σφραγίδες.

Οι χρονικές σφραγίδες αυτές συνδέονται με τις προηγούμενες και επόμενες σφραγίδες, δημιουργώντας μια αλυσίδα που αποτελεί ένα αξιόπιστο ιστορικό των γεγονότων στο δίκτυο.

Μέσω του PoH, τα μέλη του blockchain μπορούν να επαληθεύσουν τη σειρά των συμβάντων ή τη χρονική διάταξη των μπλοκ χωρίς να βασίζονται αποκλειστικά σε μηχανισμούς συναίνεσης όπως το Proof of Work ή το Proof of Stake.

Η τεχνολογία PoH μπορεί να χρησιμοποιηθεί για τη βελτίωση της απόδοσης, της κλιμάκωσης και της ασφάλειας των blockchain δικτύων.

Συνολικά, το Proof of History παρέχει έναν αξιόπιστο και αμετάβλητο χρονικό άξονα για τα γεγονότα σε ένα blockchain, καθιστώντας τον ένα σημαντικό παράγοντα για την αποτελεσματική και ασφαλή λειτουργία του δικτύου.[21][22][23]

1.11.7 Proof of Authority (PoA)

Το Proof of Authority (PoA) είναι ένας μηχανισμός συναίνεσης που χρησιμοποιείται στα blockchain για την επίτευξη σταθερότητας και ασφάλειας. Σε ένα δίκτυο PoA, οι κόμβοι που συμμετέχουν στην επικύρωση των συναλλαγών και τη δημιουργία νέων μπλοκ είναι προεπιλεγμένοι και αξιόπιστοι φορείς εξουσίας.

Στο PoA, η επικύρωση των συναλλαγών δεν βασίζεται στην επίλυση προβλημάτων απόδειξης εργασίας (Proof of Work) ή στην κατοχή συγκεκριμένου ποσού κρυπτονομίσματος (Proof of Stake).

Αντίθετα, οι κόμβοι που συμμετέχουν στο δίκτυο πιστοποιούνται με βάση την ταυτότητά τους και την εμπιστοσύνη που τους έχει αποδοθεί. Συνήθως, η εμπιστοσύνη βασίζεται σε εξωτερικούς παράγοντες, όπως η φήμη, η εμπειρία ή η εξουσία που έχουν οι κόμβοι στο δίκτυο.

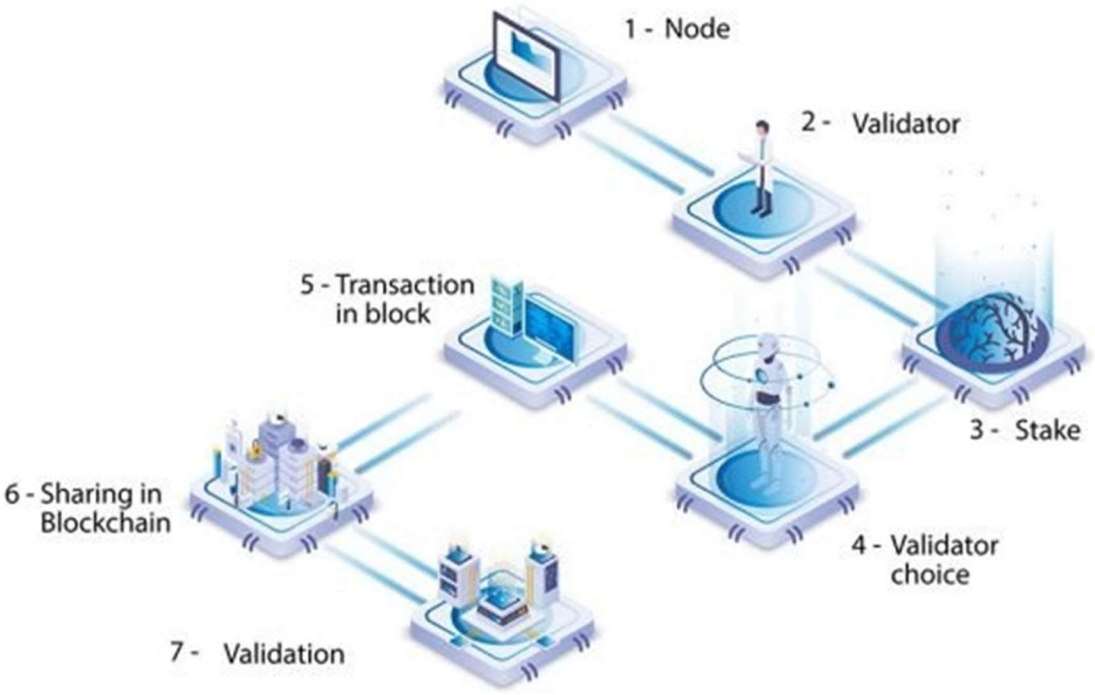
Οι κόμβοι που είναι εξουσιοδοτημένοι να επικυρώνουν συναλλαγές και να δημιουργούν μπλοκ στο δίκτυο PoA ονομάζονται "εξουσιοδοτημένοι κόμβοι" ή "εξουσιοδοτημένοι επόπτες".

Αυτοί οι κόμβοι είναι υπεύθυνοι για τη λειτουργία και την ασφάλεια του δικτύου. Οι συναλλαγές επικυρώνονται γρήγορα, καθώς δεν απαιτείται η επίλυση πολύπλοκων αλγορίθμων.

Ο PoA χρησιμοποιείται συχνά σε ιδιωτικά ή επιχειρησιακά blockchain όπου η εμπιστοσύνη και η ταχύτητα των συναλλαγών είναι πιο σημαντικές από την ανωνυμία και την ανεξαρτησία των συμμετεχόντων.

Ο PoA μπορεί να προσφέρει μεγαλύτερη απόδοση και κλιμάκωση σε σχέση με άλλους μηχανισμούς συναίνεσης, αλλά το κόστος είναι ότι το δίκτυο εξαρτάται από την εμπιστοσύνη στους εξουσιοδοτημένους κόμβους, κατάλληλη για ιδιωτικές κοινοπραξίες.[27][28]

Συνολικά, το Proof of Authority επιτρέπει την αποτελεσματική λειτουργία ιδιωτικών blockchain δικτύων, επιτρέποντας ταχείς χρόνους επιβεβαίωσης και επικύρωσης συναλλαγών, με έμφαση στην αξιοπιστία των συμμετεχόντων κόμβων.



Εικόνα 1.11 Blockchain Proof of Authority. Διαθέσιμο στο διαδίκτυο

2.Κρυπτογραφία και Blockchain

2.1 Εισαγωγή

Η κρυπτογραφία αποτελεί ένα σημαντικό στοιχείο των τεχνολογιών blockchain. Στο πλαίσιο των blockchain, η κρυπτογραφία χρησιμοποιείται για να εξασφαλίσει την ασφάλεια και την ακεραιότητα των δεδομένων που αποθηκεύονται και μεταδίδονται στο δίκτυο.

Οι κρυπτογραφικές τεχνικές που χρησιμοποιούνται στα blockchain βασίζονται στην ανάπτυξη μαθηματικών αλγορίθμων που επιτρέπουν την κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων.

Τα κρυπτογραφημένα δεδομένα μπορούν να αναγνωσθούν μόνο από τους αποδέκτες με τη χρήση του κατάλληλου κλειδιού πρόσβασης.

Οι τεχνολογίες blockchain χρησιμοποιούν κυρίως δύο κρυπτογραφικές τεχνικές:

- Κρυπτογραφία δημόσιου κλειδιού (Public Key Cryptography): Σε αυτήν την τεχνική, κάθε χρήστης έχει ένα ζεύγος κλειδιών, ένα δημόσιο και ένα ιδιωτικό κλειδί. Το δημόσιο κλειδί χρησιμοποιείται για την κρυπτογράφηση των δεδομένων, ενώ το ιδιωτικό κλειδί χρησιμοποιείται για την αποκρυπτογράφηση τους.
  Με αυτόν τον τρόπο, οι συναλλαγές και οι ενέργειες που γίνονται στο blockchain μπορούν να επαληθευθούν και να επικυρωθούν από τους συμμετέχοντες με τη χρήση των δημόσιων κλειδιών.

- 2. Κρυπτογραφικές συναρτήσεις κατακερματισμού (Cryptographic Hash Functions): Οι συναρτήσεις κατακερματισμού παίρνουν ένα αρχικό μήνυμα και το μετατρέπουν σε μια μοναδική ακολουθία χαρακτήρων, γνωστή ως κατακερματισμένη τιμή ή hash. Αυτή η μοναδική τιμή λειτουργεί ως ψηφιακή αποτύπωση του αρχικού μηνύματος. Ένας μικρός αλλαγή στο αρχικό μήνυμα θα έχει ως αποτέλεσμα μια εντελώς διαφορετική κατακερματισμένη τιμή. Οι συναρτήσεις κατακερματισμού χρησιμοποιούνται στα blockchain για τη δημιουργία των block hashes, τα οποία συμβάλλουν στην ακεραιότητα και την ασφάλεια των δεδομένων.

Η κρυπτογραφία συμβάλλει στην προστασία των δεδομένων στα blockchain, εξασφαλίζοντας την αυθεντικότητα, την εμπιστοσύνη και την ασφάλειά τους. Αποτελεί ένα κρίσιμο εργαλείο για τη διασφάλιση της αξιοπιστίας των blockchain και των συναλλαγών που πραγματοποιούνται σε αυτά.

2.2 Σύγχρονη Κρυπτογραφία

Η σύγχρονη κρυπτογραφία είναι ένας τομέας της κρυπτογραφίας που ασχολείται με τη μελέτη και την εφαρμογή αλγορίθμων κρυπτογράφησης και αποκρυπτογράφησης για την προστασία των πληροφοριών και των δεδομένων σε ψηφιακά συστήματα. Αποτελεί έναν σημαντικό τομέα της πληροφορικής και της ασφάλειας των πληροφοριών.

Στη σύγχρονη κρυπτογραφία, χρησιμοποιούνται δύο βασικές κατηγορίες κρυπτογραφικών αλγορίθμων:

- Συμμετρική Κρυπτογραφία: Στη συμμετρική κρυπτογραφία, ο αποστολέας και ο παραλήπτης χρησιμοποιούν το ίδιο κλειδί για την κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων.
  Ο αλγόριθμος κρυπτογράφησης παίρνει τα πρωτογενή δεδομένα και το κλειδί και παράγει το κρυπτογραφημένο κείμενο, ενώ ο αλγόριθμος αποκρυπτογράφησης χρησιμοποιεί το ίδιο κλειδί για να αποκρυπτογραφήσει το κρυπτογραφημένο κείμενο και να το επαναφέρει στα αρχικά δεδομένα. Η συμμετρική κρυπτογραφία είναι γρήγορη και αποδοτική, αλλά απαιτεί ασφαλές κανάλι για την ανταλλαγή του κλειδιού.

- Ασύμμετρη Κρυπτογραφία: Στην ασύμμετρη κρυπτογραφία, γνωστή επίσης και ως κρυπτογραφία με δημόσιο κλειδί, χρησιμοποιούνται ζεύγη κλειδιών, ένα δημόσιο κλειδί και ένα ιδιωτικό κλειδί.
  Ο αποστολέας χρησιμοποιεί το δημόσιο κλειδί του παραλήπτη για να κρυπτογραφήσει τα δεδομένα, και μόνο ο παραλήπτης μπορεί να αποκρυπτογραφήσει τα δεδομένα χρησιμοποιώντας το ιδιωτικό κλειδί του. Οι ασύμμετροι αλγόριθμοι είναι ασφαλείς και παρέχουν μηχανισμούς για τη δημιουργία και επαλήθευση των ψηφιακών υπογραφών. Ωστόσο, είναι πιο αργοί από τους συμμετρικούς αλγορίθμους.

Η σύγχρονη κρυπτογραφία χρησιμοποιείται ευρέως σε διάφορους τομείς, όπως η προστασία των προσωπικών δεδομένων, οι τραπεζικές συναλλαγές, οι ηλεκτρονικές υπογραφές, η ασφάλεια των δικτύων και των επικοινωνιών, καθώς και στα blockchain, όπως αναφέρατε προηγουμένως.

Οι αλγόριθμοι κρυπτογράφησης εξασφαλίζουν την εχεμύθεια, την ακεραιότητα και την αυθεντικότητα των δεδομένων, παρέχοντας ασφάλεια και εμπιστοσύνη στις ψηφιακές επικοινωνίες και συναλλαγές.

2.3 Συμμετρική Κρυπτογράφηση

Η συμμετρική κρυπτογράφηση (Symmetric Encryption) είναι μια μέθοδος κρυπτογραφίας όπου η ίδια κλειδιά χρησιμοποιούνται τόσο για την κρυπτογράφηση όσο και για την αποκρυπτογράφηση των δεδομένων.

Αυτό σημαίνει ότι ο αποστολέας και ο παραλήπτης κοινοποιούν το ίδιο κλειδί και το χρησιμοποιούν για την ασφαλή επικοινωνία.

Στη συμμετρική κρυπτογράφηση, το μήνυμα που επιθυμεί να κρυπτογραφηθεί (καθαρό κείμενο) συνδέεται με ένα κλειδί και μετατρέπεται σε μια ακανόνιστη μορφή, γνωστή ως κρυπτοκείμενο.

Ο παραλήπτης χρησιμοποιεί το ίδιο κλειδί για να αποκρυπτογραφήσει το κρυπτοκείμενο και να ανακτήσει το αρχικό καθαρό κείμενο.

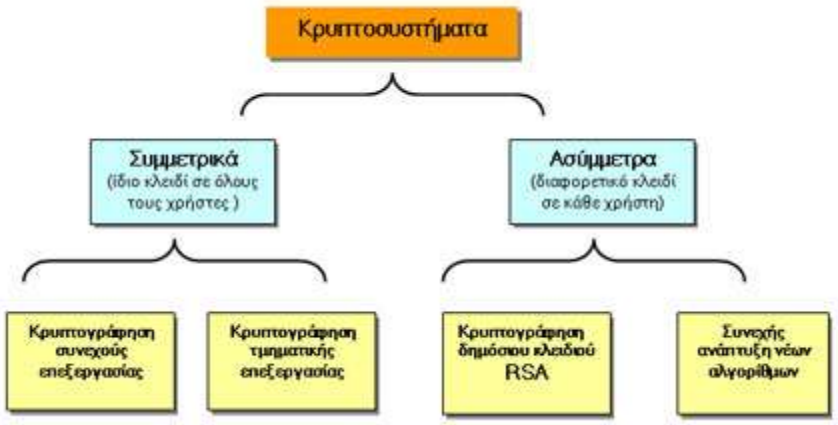
Οι αλγόριθμοι συμμετρικής κρυπτογράφησης είναι γρήγοροι και αποτελεσματικοί, καθιστώντας τη συμμετρική κρυπτογράφηση κατάλληλη για την προστασία μεγάλων όγκων δεδομένων. Ωστόσο, έχει την πρόκληση της ασφάλειας του κλειδιού.

Οποιοσδήποτε κατέχει το κλειδί μπορεί να κρυπτογραφήσει ή να αποκρυπτογραφήσει τα δεδομένα, επομένως απαιτείται ασφαλής διανομή του κλειδιού μεταξύ των εμπλεκομένων μερών.

Στο πλαίσιο των blockchain, η συμμετρική κρυπτογράφηση μπορεί να χρησιμοποιηθεί για την ασφαλή αποθήκευση των δεδομένων των μπλοκ ή για την αποκρυπτογράφηση των μεταδιδόμενων πληροφοριών μεταξύ των συμμετεχόντων στο δίκτυο.

Με τη χρήση συμμετρικής κρυπτογράφησης, τα δεδομένα μπορούν να προστατευτούν από ανεπιθύμητη πρόσβαση και αποκρυπτογραφηθούν μόνο από τους ορισμένους αποδέκτες που διαθέτουν το κατάλληλο κλειδί.[30]

Συνολικά, η συμμετρική κρυπτογράφηση είναι μια σημαντική τεχνική που εξασφαλίζει την εμπιστευτικότητα και την απομόνωση των δεδομένων στα blockchain, προσθέτοντας ένα επίπεδο ασφάλειας στις επικοινωνίες και τις ανταλλαγές πληροφοριών μεταξύ των συμμετεχόντων.



Εικόνα 2.3 Ανάλυση συμμετρικών και ασύμμετρων Κρυπτονομισμάτων. Διαθέσιμο στο διαδίκτυο

2.4 Ασύμμετρη Κρυπτογράφηση

Η ασύμμετρη κρυπτογράφηση (Asymmetric Encryption) είναι μια μέθοδος κρυπτογραφίας που χρησιμοποιεί ζεύγη κλειδιών για την κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων.

Τα ζεύγη κλειδιών αποτελούνται από ένα δημόσιο κλειδί και ένα ιδιωτικό κλειδί, τα οποία είναι μαθηματικά συσχετισμένα μεταξύ τους.

Ο αποστολέας χρησιμοποιεί το δημόσιο κλειδί του παραλήπτη για να κρυπτογραφήσει το μήνυμα. Κατόπιν, το κρυπτογραφημένο μήνυμα μπορεί να αποκρυπτογραφηθεί μόνο με το ιδιωτικό κλειδί που ανήκει στον παραλήπτη. Αυτή η διαδικασία εξασφαλίζει την εμπιστευτικότητα των δεδομένων, καθώς μόνο ο παραλήπτης μπορεί να αποκρυπτογραφήσει το μήνυμα με το ιδιωτικό του κλειδί.

Ένα ακόμη σημαντικό χαρακτηριστικό της ασύμμετρης κρυπτογράφησης είναι η ψηφιακή υπογραφή. Χρησιμοποιώντας το ιδιωτικό κλειδί του αποστολέα, μπορεί να υπογραφεί ένα μήνυμα, παρέχοντας έτσι την επιβεβαίωση της γνησιότητας και ακεραιότητας του αποστολέα.

Το ψηφιακό αυτό υπογεγραμμένο μήνυμα μπορεί στη συνέχεια να επαληθευθεί με το δημόσιο κλειδί του αποστολέα.

Στο πλαίσιο των blockchain, η ασύμμετρη κρυπτογράφηση χρησιμοποιείται για πολλούς σκοπούς, όπως η δημιουργία και επαλήθευση των ψηφιακών υπογραφών για την επιβεβαίωση της γνησιότητας των μπλοκ και των συναλλαγών, καθώς και η ασφαλής ανταλλαγή κλειδιών για την προστασία των επικοινωνιών μεταξύ των συμμετεχόντων.

Η ασύμμετρη κρυπτογράφηση παρέχει ένα αποτελεσματικό μέσο για τη διατήρηση της ασφάλειας και της εμπιστοσύνης στο περιβάλλον των blockchain.

2.5 Hash Functions

Οι συναρτήσεις κατακερματισμού (hash functions) είναι μαθηματικές λειτουργίες που αντιστοιχίζουν μια είσοδο (ή μήνυμα) σε ένα μοναδικό αναπαράσταση ταξινόμησης, που ονομάζεται hash value ή hash code. Η σημαντική ιδιότητα των hash functions είναι ότι είναι μη αναστρέψιμες, πράγμα που σημαίνει ότι δεν

μπορεί να εξαχθεί το αρχικό μήνυμα από την hash τιμή.Οι hash functions χρησιμοποιούνται ευρέως στην κρυπτογραφία και τα blockchain για διάφορους σκοπούς:

- Ακεραιότητα δεδομένων: Οι hash functions μπορούν να χρησιμοποιηθούν για να επαληθευτεί αν ένα αρχείο ή ένα μήνυμα έχει τροποποιηθεί. Μετά την εφαρμογή της hash function στο αρχικό μήνυμα, μπορούμε να ελέγξουμε αν η παραγόμενη hash τιμή αντιστοιχεί στην αρχική. Αν ακόμη και ένα μικρότερο μέρος του αρχικού μηνύματος τροποποιηθεί, η hash τιμή θα είναι εντελώς διαφορετική.

- Κρυπτογραφία: Οι hash functions χρησιμοποιούνται για τη δημιουργία ψηφιακών υπογραφών και την αποθήκευση κωδικών πρόσβασης. Αντί να αποθηκεύουμε άμεσα έναν κωδικό πρόσβασης, αποθηκεύουμε την hash τιμή του.
  Όταν κάποιος προσπαθεί να συνδεθεί, η εισαγωγή του κωδικού πρόσβασης συγκρίνεται με την αποθηκευμένη hash τιμή. Αυτό επιτρέπει την επαλήθευση της ορθότητας του κωδικού χωρίς να αποκαλύπτεται ο ίδιος ο κωδικός πρόσβασης.

- Δημιουργία διευθύνσεων blockchain: Στα blockchain, οι hash functions χρησιμοποιούνται για να δημιουργηθούν μοναδικές διευθύνσεις για τους χρήστες και τα μπλοκ. Η είσοδος της hash function περιλαμβάνει συνήθως στοιχεία όπως η δημόσια κλειδοθήκη του χρήστη ή τα δεδομένα του μπλοκ, και η hash τιμή που παράγεται χρησιμοποιείται ως η μοναδική διεύθυνση αναφοράς τους.

Οι hash functions προσφέρουν αποδοτικότητα, ασφάλεια και ανεξαρτησία από το μέγεθος των δεδομένων. Είναι ουσιώδες στοιχείο της κρυπτογραφίας και των blockchain, παρέχοντας ασφάλεια και εμπιστοσύνη στις διαδικασίες αποθήκευσης και επαλήθευσης των δεδομένων.

2.6 Digital Signatures

Η ψηφιακή υπογραφή (digital signature) είναι μια τεχνική της κρυπτογραφίας που Η ψηφιακή υπογραφή (digital signature) είναι μια τεχνική της κρυπτογραφίας που χρησιμοποιείται για την επαλήθευση της αυθεντικότητας, της ακεραιότητας και της μη-απαρνησιμότητας ενός μηνύματος ή εγγράφου σε ηλεκτρονική μορφή. Οι ψηφιακές υπογραφές επιτρέπουν σε έναν αποστολέα να αποδείξει ότι το μήνυμα που έχει υπογράψει προέρχεται από αυτόν και δεν έχει τροποποιηθεί κατά τη μεταφορά του.

Η διαδικασία της δημιουργίας ψηφιακής υπογραφής περιλαμβάνει τα εξής βήματα:

1. Κατακερματισμός (hashing): Αρχικά, το μήνυμα που θα υπογραφεί υποβάλλεται σε μια συνάρτηση κατακερματισμού (hash function) για να παραχθεί μια μοναδική hash τιμή. Αυτή η τιμή αντιπροσωπεύει το μήνυμα και λειτουργεί ως αποτύπωμα του.

2. Ιδιωτικό κλειδί (private key): Ο αποστολέας χρησιμοποιεί το ιδιωτικό κλειδί του για να δημιουργήσει μια ψηφιακή υπογραφή για την hash τιμή του μηνύματος. Το ιδιωτικό κλειδί είναι ένα μυστικό κλειδί που γνωρίζει μόνο ο αποστολέας και το χρησιμοποιεί για να υπογράψει τα μηνύματα.

3. Υπογραφή του μηνύματος: Ο αποστολέας εφαρμόζει μια μαθηματική λειτουργία στην hash τιμή του μηνύματος με το ιδιωτικό κλειδί του για να δημιουργήσει την ψηφιακή υπογραφή του μηνύματος. Αυτή η υπογραφή είναι μοναδική για το συγκεκριμένο μήνυμα και τον αποστολέα.

Για να επαληθευτεί η ψηφιακή υπογραφή, ο παραλήπτης ακολουθεί τα παρακάτω βήματα:

1. Λήψη του μηνύματος και της ψηφιακής υπογραφής: Ο παραλήπτης λαμβάνει το μήνυμα και την ψηφιακή υπογραφή που το συνοδεύει.

2. Υπολογισμός κατακερματισμού (hash): Ο παραλήπτης χρησιμοποιεί την ίδια hash function που χρησιμοποιήθηκε από τον αποστολέα για να υπολογίσει την hash τιμή του ληφθέντος μηνύματος.

3. Επαλήθευση της υπογραφής: Ο παραλήπτης χρησιμοποιεί το δημόσιο κλειδί του αποστολέα για να εφαρμόσει μια μαθηματική λειτουργία στην ψηφιακή υπογραφή. Αν η ψηφιακή υπογραφή ταιριάζει με την hash τιμή του μηνύματος, τότε η υπογραφή θεωρείται έγκυρη και το μήνυμα θεωρείται αυθεντικό και ακέραιο.

Οι ψηφιακές υπογραφές παρέχουν μια αξιόπιστη μέθοδο για την επαλήθευση της αυθεντικότητας και της ακεραιότητας των δεδομένων.

Επιπλέον, επιτρέπουν στον αποστολέα να αποδείξει την αυθεντικότητα του μηνύματος και να μην μπορεί να απαρνηθεί αργότερα ότι το υπέγραψε. Αυτή η τεχνική είναι κρίσιμη για την ασφάλεια των ηλεκτρονικών συναλλαγών και της ανταλλαγής πληροφοριών.

3.Βασικά είδη Κρυπτονομισμάτων

3.1 Εισαγωγή

Τα βασικά είδη κρυπτονομισμάτων είναι τα ακόλουθα:

1. Bitcoin (BTC): Το Bitcoin ήταν το πρώτο και πιο γνωστό κρυπτονόμισμα που δημιουργήθηκε. Λειτουργεί με την τεχνολογία blockchain και αποτελεί το πρότυπο για τις υπόλοιπες κρυπτονομίσεις. Χαρακτηρίζεται από υψηλή αξία και μεγάλη αγοραστική δύναμη.
2. Ethereum (ETH): Το Ethereum είναι μια πλατφόρμα ανοικτού κώδικα που υποστηρίζει τη δημιουργία και εκτέλεση έξυπνων συμβολαίων. Το κρυπτονόμισμα που χρησιμοποιείται στο δίκτυο Ethereum ονομάζεται Ether (ETH). Επιπλέον, το Ethereum αποτελεί και τη δεύτερη μεγαλύτερη αξία σε αγοραστική δύναμη μετά το Bitcoin.
3. Ripple (XRP): Το Ripple είναι μια πλατφόρμα πληρωμών και ανταλλαγής χρημάτων που επιτρέπει γρήγορες, ασφαλείς και οικονομικές συναλλαγές. Το κρυπτονόμισμα XRP χρησιμοποιείται στην πλατφόρμα για τη διεκπεραίωση συναλλαγών.

4. Litecoin (LTC): Το Litecoin είναι ένα κρυπτονόμισμα που βασίζεται στο πρωτόκολλο του Bitcoin, αλλά με κάποιες τροποποιήσεις. Έχει γρηγορότερους χρόνους επιβεβαίωσης συναλλαγών και μικρότερες προμήθειες συναλλαγής από το Bitcoin.
5. Bitcoin Cash (BCH): Το Bitcoin Cash δημιουργήθηκε από το διαχωρισμό του αλυσίδας μπλοκ του Bitcoin (hard fork). Έχει μεγαλύτερο μέγεθος μπλοκ και γρηγορότερους χρόνους επιβεβαίωσης συναλλαγών από το Bitcoin.

Εικόνα 3,1 Βασικά είδη Κρυπτονομισμάτων (Bitcoin). Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper 151(2014), 1–32 (2014) [33]

Αυτά είναι μερικά από τα βασικά είδη κρυπτονομισμάτων. Υπάρχουν επίσης πολλά άλλα κρυπτονομίσματα με διάφορα χαρακτηριστικά και λειτουργίες.

3.2 Token

Τα tokens στα κρυπτονομίσματα αναφέρονται σε ψηφιακά ενεργητικά που χρησιμοποιούνται εντός ενός συγκεκριμένου οικοσυστήματος κρυπτονομισμάτων. Τα tokens μπορούν να αναπαριστούν διάφορα πράγματα, όπως κρυπτονομίσματα, ψηφιακά περιουσιακά στοιχεία, δικαιώματα ή υπηρεσίες εντός του συστήματος.

Τα tokens μπορούν να λειτουργήσουν ως εναλλακτική μορφή χρήματος εντός του οικοσυστήματος και μπορούν να χρησιμοποιηθούν για την αγορά αγαθών και υπηρεσιών εντός της κοινότητας των χρηστών.

Επιπλέον, τα tokens μπορούν να χρησιμοποιηθούν για την εκτέλεση συναλλαγών, τη συμμετοχή σε ψηφοφορίες ή την απόκτηση πρόσβασης σε ειδικές λειτουργίες και υπηρεσίες εντός του οικοσυστήματος.

Ένα παράδειγμα γνωστού token είναι το ERC-20 token στο δίκτυο Ethereum, το οποίο χρησιμοποιείται για τη δημιουργία και λειτουργία έξυπνων συμβολαίων στο Ethereum. Άλλα παραδείγματα tokens περιλαμβάνουν τα Binance Coin (BNB) στην πλατφόρμα Binance και τα Stellar Lumens (XLM) στο δίκτυο Stellar.

Συνολικά, τα tokens στα κρυπτονομίσματα αντιπροσωπεύουν μια μορφή ψηφιακής αξίας ή χρήσης εντός ενός οικοσυστήματος και χρησιμοποιούνται για να διευκολύνουν τις συναλλαγές και τη λειτουργία του συστήματος.

3.3 Τυπικά Νομίσματα

Στα blockchain, τα τυπικά νομίσματα αναφέρονται στα ψηφιακά νομίσματα που λειτουργούν εντός ενός συγκεκριμένου blockchain. Αυτά τα νομίσματα μπορούν να χρησιμοποιηθούν για την ανταλλαγή αξίας και την πραγματοποίηση συναλλαγών εντός του δικτύου του συγκεκριμένου blockchain.

Στην περίπτωση του Bitcoin, το τυπικό νόμισμα είναι το Bitcoin (BTC). Οι χρήστες μπορούν να αγοράσουν, να πωλήσουν ή να ανταλλάξουν Bitcoins για αγαθά και υπηρεσίες. Οι συναλλαγές με Bitcoin καταγράφονται στο blockchain του Bitcoin, παρέχοντας διαφάνεια και ασφάλεια στις συναλλαγές.

Αντίστοιχα, σε άλλα blockchain, υπάρχουν επίσης τυπικά νομίσματα που χρησιμοποιούνται εντός του συγκεκριμένου οικοσυστήματος. Για παράδειγμα, στο Ethereum, το τυπικό νόμισμα είναι το Ether (ETH), ενώ στο Binance Smart Chain, το τυπικό νόμισμα είναι το Binance Coin (BNB). Κάθε blockchain έχει το δικό του νόμισμα ή token που χρησιμοποιείται για την ανταλλαγή αξίας και την εκτέλεση συναλλαγών εντός του οικοσυστήματος.

Συνολικά, τα τυπικά νομίσματα στα blockchain λειτουργούν ως ψηφιακά ενεργητικά που επιτρέπουν την ανταλλαγή αξίας και την εκτέλεση συναλλαγών εντός του συστήματος του συγκεκριμένου blockchain.

3.4 Μοναδικά Τεκμήρια – NFT

Το NFT αναφέρεται στα "Non-Fungible Tokens" (Μη-εναλλάξιμα Κρυπτονομίσματα) και αποτελεί ένα είδος ψηφιακού περιουσιακού στοιχείου που χρησιμοποιεί την τεχνολογία του blockchain για να εγγυηθεί τη μοναδικότητα, την αυθεντικότητα και την ανακλησιμότητά του.

Σε αντίθεση με τα κρυπτονομίσματα που είναι ανταλλάξιμα (όπως τα Bitcoin ή τα Ethereum), τα NFT είναι μοναδικά και μη-εναλλάξιμα. Κάθε NFT έχει μια μοναδική ψηφιακή υπογραφή που το διακρίνει από άλλα και επιβεβαιώνει την αυθεντικότητά του.

Τα NFT μπορούν να αντιπροσωπεύουν οποιοδήποτε ψηφιακό αγαθό ή στοιχείο, όπως έργα τέχνης, συλλεκτικά αντικείμενα, εικόνες, βίντεο, μουσική, εικονογραφήσεις, εικονογραφημένα GIF, εικονογραφημένα παιχνίδια και άλλα.

Η μοναδική φύση των NFT επιτρέπει στους δημιουργούς να επισημάνουν και να αποδείξουν την κυριότητα και την αυθεντικότητα των ψηφιακών τους δημιουργιών.

Η αξία ενός NFT καθορίζεται από τη ζήτηση του και την αναγνώριση της αξίας του από την αγορά. Οι συναλλαγές με NFT συνήθως γίνονται μέσω αγοραπωλησιών σε ανοιχτές αγορές ή δημοπρασίες σε πλατφόρμες που υποστηρίζουν τα NFT.

Τα NFT έχουν προκαλέσει μεγάλο ενδιαφέρον και έχουν εφαρμογές σε πολλούς τομείς, όπως η τέχνη, η μουσική, οι αθλητικές συλλογές και η ψυχαγωγία, παρέχοντας νέες ευκαιρίες για την αναγνώριση, την αμοιβή και την εμπορία ψηφιακών δημιουργιών.

Τα μοναδικά τεκμήρια (unique tokens) στα blockchain αναφέρονται σε ψηφιακά αντικείμενα που είναι μοναδικά και μη-εναλλάξιμα.

Αυτά τα τεκμήρια χρησιμοποιούν την τεχνολογία του blockchain για να εγγυηθούν τη μοναδικότητα, την αυθεντικότητα και την ανακλησιμότητα τους.

Εικόνα 2.6 Ανάλυση συμμετρικών και ασύμμετρων Κρυπτονομισμάτων. Andreas M. Antonopoulos and Dr. Gavin Wood, “Mastering Ethereum”, O’Reilly Media, Inc,United States of America, 2019 [32]

Ένα από τα πιο γνωστά παραδείγματα μοναδικών τεκμηρίων στα blockchain είναι τα Non-Fungible Tokens (NFTs) που αναφέρθηκαν προηγουμένως. Τα NFTs αντιπροσωπεύουν μοναδικά ψηφιακά αντικείμενα, όπως έργα τέχνης, εικόνες, βίντεο, μουσική και άλλα.

Κάθε NFT έχει μια μοναδική αναγνωριστική υπογραφή που το καθιστά μοναδικό στο blockchain, επιτρέποντας στους κατόχους να αποδεικνύουν την αυθεντικότητα και την κυριότητά τους.

Τα μοναδικά τεκμήρια μπορούν να έχουν ευρύ φάσμα εφαρμογών και χρησιμοποιούνται σε διάφορους τομείς.

Πέρα από την τέχνη και τον πολιτισμό, τα μοναδικά τεκμήρια μπορούν να χρησιμοποιηθούν στη μουσική βιομηχανία για τη δημιουργία και την ανταλλαγή μοναδικών άλμπουμ ή συναυλιών.

Επίσης, μπορούν να εφαρμοστούν στον αθλητισμό, για παράδειγμα, στη δημιουργία συλλογών ψηφιακών καρτών παικτών ή στην εκδήλωση ειδικών δικαιωμάτων πρόσβασης σε αθλητικά γεγονότα.

Οι μοναδικές τεκμηριώσεις μπορούν να προσφέρουν αξία στους δημιουργούς, δίνοντας τους τη δυνατότητα να αμελήσουν τα δικαιώματα πνευματικής ιδιοκτησίας και να αμεριστεύσουν τα έργα τους απευθείας στο κοινό.

Επίσης, μπορούν να παρέχουν νέες ευκαιρίες για τους επενδυτές και τους συλλέκτες να αγοράσουν, να πωλούν και να διαπραγματεύονται μοναδικά ψηφιακά αντικείμενα.

Συνολικά, τα μοναδικά τεκμήρια στα blockchain επιτρέπουν την αυθεντική, μοναδική και διαφανή καταγραφή και ανταλλαγή ψηφιακών αντικειμένων, ανοίγοντας νέους ορίζοντες για την ψηφιακή οικονομία και τον πολιτισμό.

Τα Non-Fungible Tokens (NFTs) είναι μοναδικά ψηφιακά αντικείμενα που χρησιμοποιούν την τεχνολογία του blockchain για να εγγυηθούν τη μοναδικότητα, την αυθεντικότητα και την ανακλησιμότητά τους. Ας δούμε λίγο πιο αναλυτικά τι είναι τα NFTs και πώς λειτουργούν:

1. Μοναδικότητα: Κάθε NFT έχει μια μοναδική αναγνωριστική ψηφιακή υπογραφή, η οποία το καθιστά μοναδικό στο blockchain. Αυτό σημαίνει ότι κάθε NFT είναι μοναδικό και διαφορετικό από άλλα.

2. Αυθεντικότητα: Η τεχνολογία του blockchain επιτρέπει την επαλήθευση της αυθεντικότητας ενός NFT. Καθώς κάθε NFT είναι συνδεδεμένο με ένα μοναδικό αναγνωριστικό και αποθηκεύεται σε ένα ανοιχτό και αναλλοίωτο κατανεμημένο λογαριασμό, μπορεί να επαληθευθεί η γνησιότητα και η προέλευση του.

3. Ανακλησιμότητα: Οι συναλλαγές με NFTs είναι μόνιμες και ανακλήσιμες. Αυτό σημαίνει ότι ο κάτοχος ενός NFT μπορεί να το μεταβιβάσει ή να το πωλήσει σε άλλον χρήστη, και οι αλλαγές καταγράφονται και αποθηκεύονται στο blockchain. Η ανακλησιμότητα δίνει στους κατόχους την ευελιξία να αλληλεπιδρούν με τα NFTs στο πλαίσιο των δυνατοτήτων και των περιορισμών που καθορίζονται από το έξυπνο συμβόλαιο του NFT.

Τα NFTs έχουν εφαρμογές σε πολλούς τομείς:

1. Τέχνη: Τα NFTs έχουν επαναπροσδιορίσει τον τρόπο που η τέχνη δημιουργείται, ανταλλάσσεται και αντιλαμβάνεται. Καλλιτέχνες μπορούν να δημιουργήσουν ψηφιακά έργα τέχνης ως NFTs και να τα πωλήσουν σε αγοραστές που ενδιαφέρονται να αποκτήσουν αποκλειστικά δικαιώματα ή να συλλέξουν έργα τέχνης.

2. Μουσική: Οι καλλιτέχνες μπορούν να κυκλοφορήσουν τη μουσική τους ως NFTs, προσφέροντας στους ακροατές τη δυνατότητα να αποκτήσουν αποκλειστικά δικαιώματα πρόσβασης στα έργα τους ή να συλλέξουν μοναδικές εκδόσεις τραγουδιών.

3. Αθλητικά συλλεκτικά: Οι ομάδες αθλητικών λιγκ και αθλητές μπορούν να δημιουργήσουν συλλεκτικά NFTs που αντιπροσωπεύουν πραγματικά αθλητικά αντικείμενα, όπως φωτογραφίες, βίντεο ή ψηφιακές κάρτες παικτών, προσφέροντας στους φανς τη δυνατότητα να τα αγοράσουν και να τα συλλέξουν.

4. Εικονικός κόσμος: Οι εικονικοί κόσμοι και οι παιχνιδοπλατφόρμες μπορούν να χρησιμοποιήσουν τα NFTs για την αντιπροσώπευση και την ανταλλαγή εικονικών αντικειμένων, όπως εικονικές γης, αντικείμενα εξοπλισμού ή εικονικά κτίρια.

Οι πωλήσεις NFT έχουν συχνά πραγματοποιηθεί μέσω online αγορών ή δημοπρασιών, με τη χρήση κρυπτονομισμάτων όπως το Ethereum. Οι συναλλαγές NFT μπορούν να περιλαμβάνουν ένα ποσοστό από τις πωλήσεις του εκάστοτε NFT που πηγαίνει στον δημιουργό, προσφέροντας νέες ευκαιρίες για καλλιτέχνες και δημιουργούς να επωφεληθούν από την αξία των δημιουργιών τους.

Παρόλο που τα NFTs έχουν γνωρίσει μεγάλη ανάπτυξη και ενδιαφέρον, είναι σημαντικό να σημειωθεί ότι η αξία τους μπορεί να υπόκειται σε μεγάλη αλληλεπίδραση και η αγορά τους μπορεί να είναι υψηλά αλλόκοτη και αβέβαιη.

4.Παρουσίαση του Forum

4.1 Web Interface

4.1.1 Εισαγωγή

Ένα ιστότοπος (website) είναι μια συλλογή ηλεκτρονικών σελίδων που βρίσκονται σε έναν κοινό ψηφιακό τόπο στον παγκόσμιο ιστό (World Wide Web). Ο ιστότοπος μπορεί να περιέχει διάφορο περιεχόμενο, όπως κείμενο, εικόνες, βίντεο, ήχο και άλλα πολυμέσα.

Ένα φόρουμ είναι μια ειδική μορφή ιστότοπο που επιτρέπει στους χρήστες να δημοσιεύουν, να συζητούν και να ανταλλάσσουν απόψεις και πληροφορίες με άλλους χρήστες. Συνήθως, τα φόρουμ περιλαμβάνουν διάφορες ενότητες ή θέματα συζητήσεων, γνωστά και ως "threads".

Οι χρήστες μπορούν να δημοσιεύουν νέα θέματα ή να απαντούν σε υπάρχοντα θέματα, δημιουργώντας έναν διάλογο.

Συνήθως, στα φόρουμ μπορούν να υπάρχουν συστήματα σχολιασμού και ανάρτησης (comments και posts), που επιτρέπουν στους χρήστες να σχολιάζουν τις αναρτήσεις ή τις απαντήσεις των άλλων χρηστών.

Αυτό δημιουργεί έναν δια δραστικό χώρο, όπου οι χρήστες μπορούν να συμμετέχουν σε συζητήσεις, να ανταλλάσσουν απόψεις και γνώμες, και να αποκτήσουν πληροφορίες από την κοινότητα του φόρουμ.

Ας εμβαθύνουμε λίγο περισσότερο στη λειτουργία ενός φόρουμ με σχόλια και αναρτήσεις.

Ένα φόρουμ λειτουργεί ως ένας χώρος συζήτησης και ανταλλαγής πληροφοριών, όπου οι χρήστες μπορούν να δημιουργήσουν λογαριασμό και να συμμετέχουν σε συζητήσεις που αφορούν συγκεκριμένα θέματα.

Οι χρήστες μπορούν να δημιουργούν νέα θέματα, γνωστά και ως "threads" ή "νήματα", σε διάφορες ενότητες του φόρουμ, και άλλοι χρήστες μπορούν να απαντήσουν σε αυτά τα θέματα.

Τα σχόλια και οι αναρτήσεις συνιστούν τα βασικά στοιχεία μιας διαλογής σε ένα φόρουμ. Οι χρήστες μπορούν να απαντούν σε θέματα που έχουν δημιουργηθεί από άλλους χρήστες, και επίσης μπορούν να ανταποκριθούν σε σχόλια που έχουν γίνει σε αυτά τα θέματα.

Αυτό δημιουργεί μια δυναμική συζήτηση, καθώς οι χρήστες μπορούν να ανταλλάσσουν απόψεις, να παρέχουν πληροφορίες, να θέτουν ερωτήσεις και να συνεισφέρουν στην κοινότητα του φόρουμ.

Τα σχόλια και οι αναρτήσεις μπορούν να περιέχουν κείμενο, συνδέσμους, εικόνες, βίντεο ή άλλα πολυμέσα. Οι χρήστες μπορούν να αντιδράσουν σε ένα σχόλιο ή μια ανάρτηση, συμφωνώντας ή διαφωνώντας με αυτήν, και μπορούν επίσης να κάνουν αναφορές σε άλλα μέλη της κοινότητας με τη χρήση αναφορικών σημάνσεων ή "tags".

Η συγκεκριμένη διάρθρωση των σχολίων και των αναρτήσεων εξαρτάται από την πλατφόρμα του φόρουμ. Ορισμένα φόρουμ ενδέχεται να έχουν ιεραρχική δομή, όπου οι αναρτήσεις σχετίζονται μεταξύ τους με τρόπο που δημιουργεί δέντρο συζήτησης.

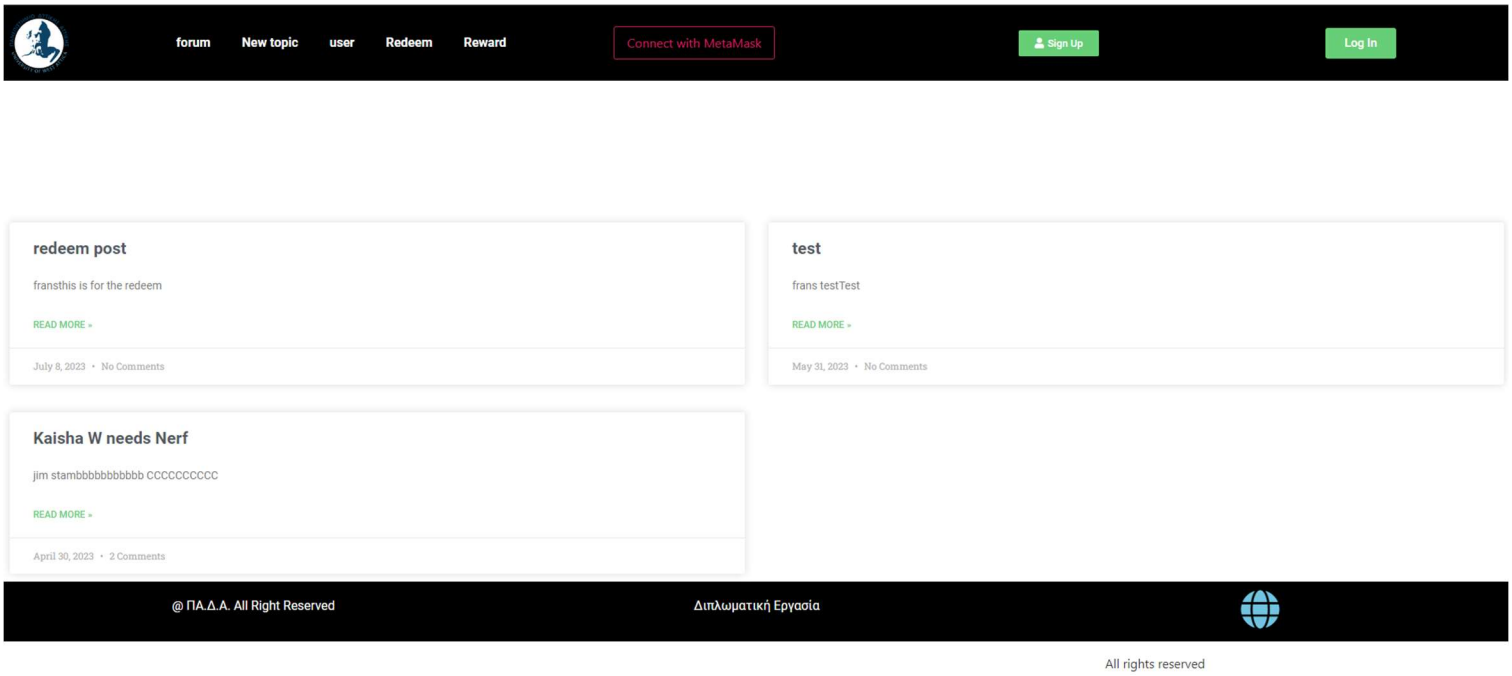
4.1.2 Επεξήγηση Φωτογραφιών Forum

.Ας εξηγήσουμε την φωτογραφία από κάτω με λεπτομέρειες:

Βλέπουμε την αρχική σελίδα του Forum, στην οποία δημοσιεύονται τα Posts που δημιουργούνται στο Site.

Επιπροσθέτως έχει την μπάρα πλοήγησης (nav menu) το οποίο περιέχει τις σελίδες του Website, οι οποίες είναι όλες υπερσύνδεσμοι και παραπέμπουν στο κατάλληλο section του Website.

Η μπάρα περιήγησης (navigation menu) περιέχει επίσης το κουμπί διασύνδεσης με το Metamask, ένα κουμπί που παραπέμπει στην σελίδα εισόδου λογαριασμού και άλλο ένα κουμπί που παραπέμπει στην σελίδα που δημιουργεί ένας χρήστης λογαριασμός.



Εικόνα 4.1.2.1 Φωτογραφία από την ιστοσελίδα που δημιουργήθηκε για την υλοποίηση της εφαρμογής. Από τον δημιουργό.

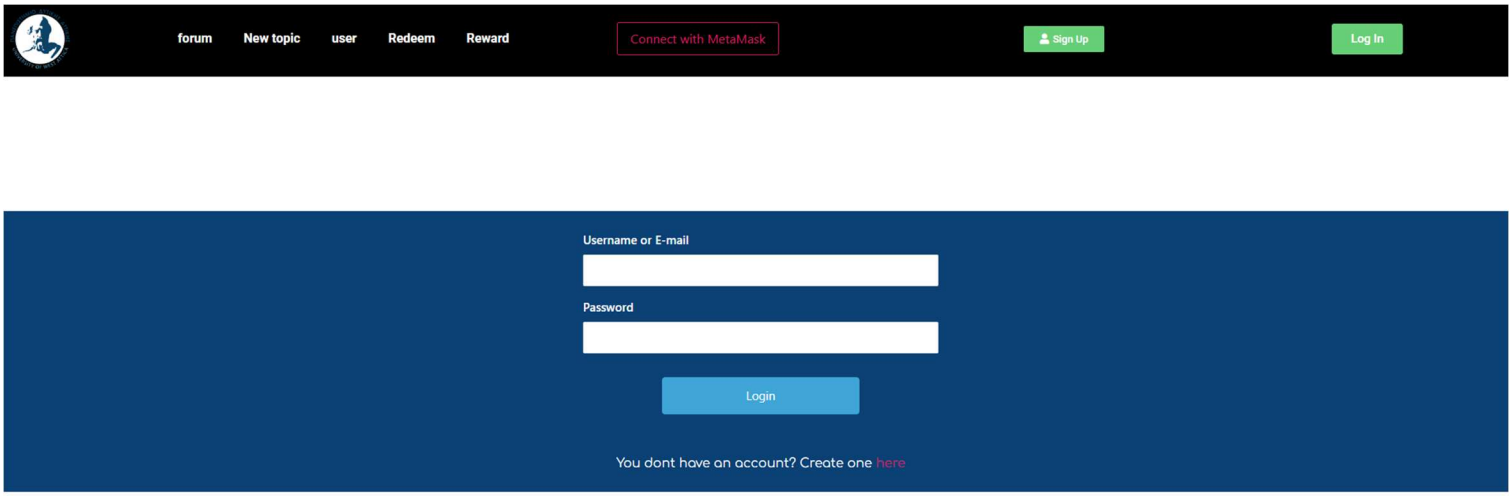
Στην φωτογραφία από πάνω βλέπουμε την συνέχεια της αρχικής σελίδας η οποία περιέχει και το Footer του Website.

Το footer του Website περιέχει τα πνευματικά δικαιώματα και έναν υπερσύνδεσμο με το Website της σχολής.

Στην φωτογραφεία παρακάτω μας παρουσιάζεται η σελίδα στην οποία γίνεται η σύνδεση του χρήστη στον λογαριασμό του (Log In).

Είναι μία φόρμα με τις απαραίτητες πληροφορίες που χρειάζεται η βάση δεδομένων του Website ώστε να κάνει την σωστή ταυτοποίηση του χρήστη με τα στοιχεία που είναι ήδη αποθηκευμένα.

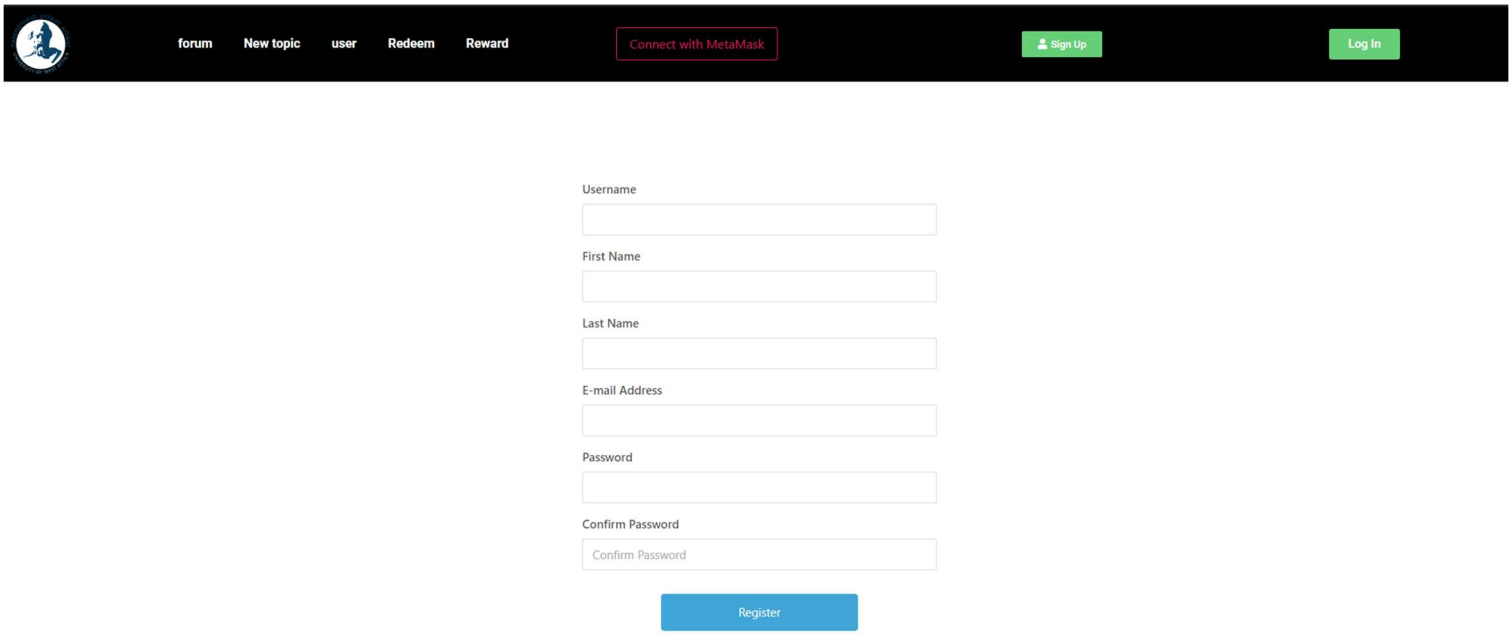
Παρέχει επίσης την δυνατότητα μεταφοράς στην σελίδα δημιουργίας χρήστη, αν και εφόσον ο χρήστης δεν έχει, ή ακόμα και την ανάκτηση κωδικού σε περίπτωση που ο χρήστης τον έχει ξεχάσει.



Εικόνα 4.1.2.2 Φωτογραφία από την ιστοσελίδα που δημιουργήθηκε για την υλοποίηση της εφαρμογής. Από τον δημιουργό.

Στην αποκάτω φωτογραφία είναι η φόρμα δημιουργίας account user, στην σελίδα σύνδεσης (Sign Up).

Έχει την κατάλληλη φόρμα με τα κατάλληλα στοιχεία ώστε μόλις γίνει η σύνδεση και η συμπλήρωση των κατάλληλων δεδομένων, να αποθηκεύσει αυτόματα τα στοιχεία του χρήστη η βάση δεδομένων.

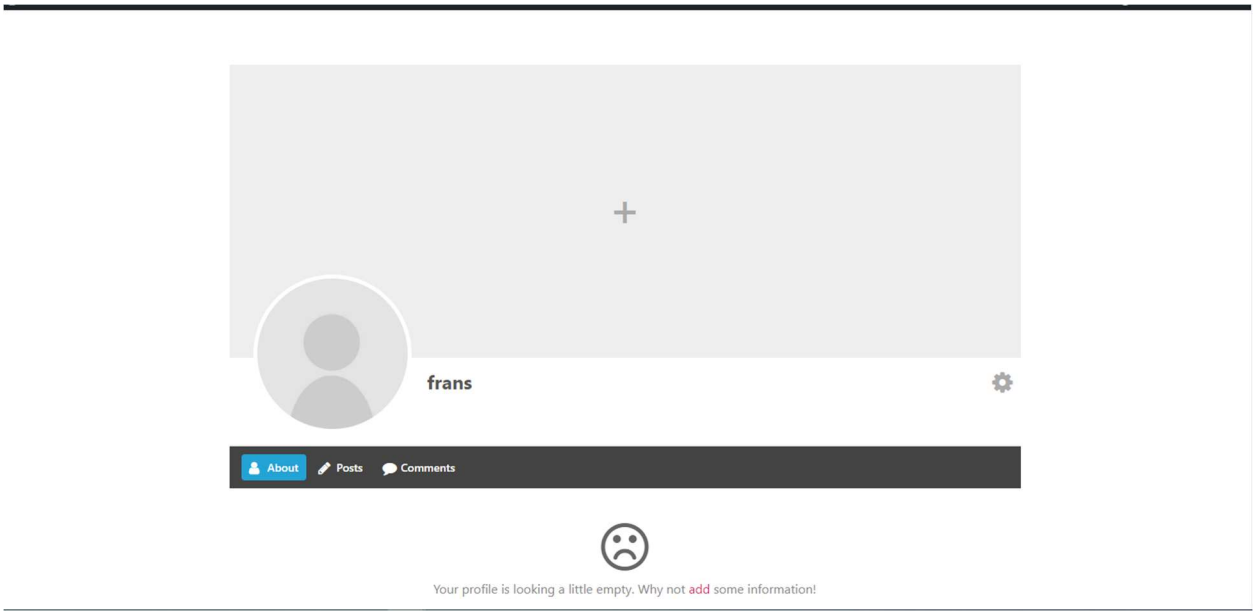


Εικόνα 4.1.2.3 Φωτογραφία από την ιστοσελίδα που δημιουργήθηκε για την υλοποίηση της εφαρμογής. Από τον δημιουργό.

Εδώ μας παρουσιάζεται η σελίδα του χρήστη (user), η οποία είναι προσβάσιμη μόνο εάν έχει κάνει είσοδο (log in) ο χρήστης, αλλιώς δεν μπορεί να επιλεγεί (clickable).

Σε αυτή την σελίδα ο χρήστης μπορεί να προσθέσει προσωπικές πληροφορίες, φωτογραφία και τα συναφή.

Είναι αρκετά χρήσιμη για παρουσίαση και για reliability του χρήστη.



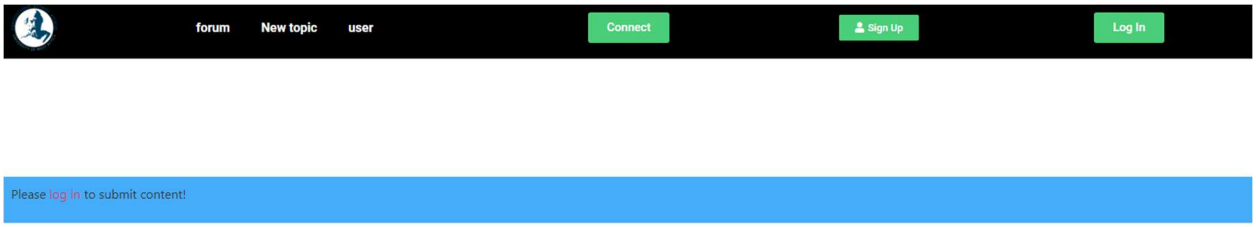
Εικόνα 4.1.2.4 Φωτογραφία από την ιστοσελίδα που δημιουργήθηκε για την υλοποίηση της εφαρμογής. Από τον δημιουργό.

Εδώ παρατηρούμε την φόρμα υποβολής νέας δημοσίευσης (Post).

Ο χρήστης που ενδιαφέρεται δίνει τις κατάλληλες πληροφορίες, τίτλο θέματος, σε ποιά κατηγορία θέλει να τοποθετηθεί το Post του, κατάλληλα tags ώστε να βοηθάνε στο search και Google search, μία ερώτηση επιβεβαίωσης ότι δεν είναι ρομποτ (spam bot) και τέλος το κείμενο που επιθυμεί να παραθέσει.

Εικόνα 4.1.2.5 Φωτογραφία από την ιστοσελίδα που δημιουργήθηκε για την υλοποίηση της εφαρμογής. Από τον δημιουργό.

Στην αποκάτω φωτογραφία παρατηρούμε τι συμβαίνει αν κάποιος έχει επισκεφτεί το Website χωρίς να κάνει login στον λογαριασμό του, οπότε δεν τον αφήνει να κάνει Post, καθώς δε θα έχει τα κατάλληλα στοιχεία ο κώδικας και η βάση δεδομένων ώστε να κάνουν την διασύνδεση με το πορτοφόλι και το account του χρήστη στο Metamask ώστε να δώσει την κατάλληλη ανταμοιβή.

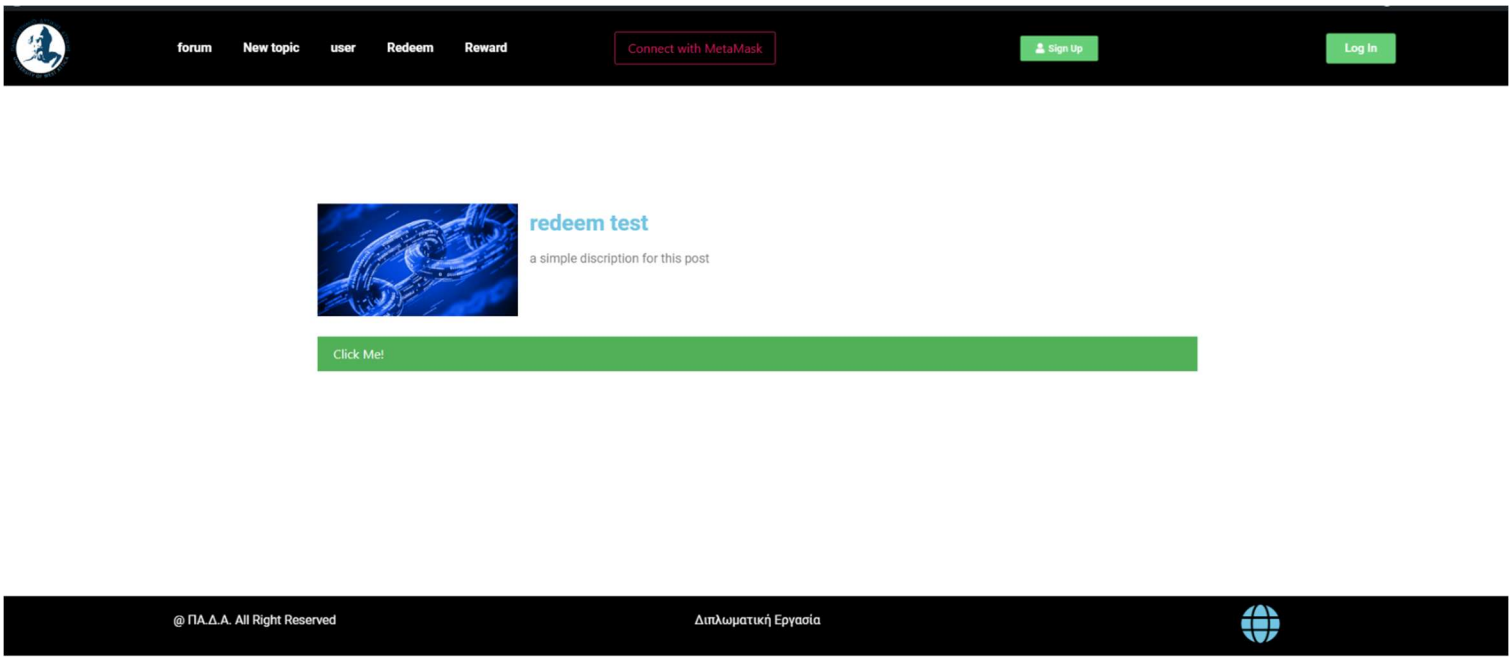


Εικόνα 4.1.2.6 Φωτογραφία από την ιστοσελίδα που δημιουργήθηκε για την υλοποίηση της εφαρμογής. Από τον δημιουργό.

Στην παρακάτω φωτογραφία παρατηρούμε την σελίδα στην οποία υπάρχει το Post με την κατάλληλη περιγραφή και τιμή (νομίσματα).

Σε αυτή την σελίδα μπορεί να κάνει ο χρήστης το redeem πατώντας το κουμπί στην σελίδα.

Το κουμπί θα κάνει τον έλεγχο ότι πληρεί τις προδιαγραφές (συνδεδεμένο Metamask,coin balance, ορθά στοιχεία – connectivity) και έπειτα θα συνεχίσει στην πραγματοποίηση της συναλλαγής.



Εικόνα 4.1.2.7 Φωτογραφία από την ιστοσελίδα που δημιουργήθηκε για την υλοποίηση της εφαρμογής. Από τον δημιουργό.

Στην παρακάτω φωτογραφία είναι η σελίδα του admin που θα κάνει το έλεγχο της συναλλαγής όταν ο πελάτης θα πηγαίνει να παραλάβει το προϊόν.

Είναι μία φόρμα που ελέγχει το μοναδικό κλειδί που έχει δημιουργηθεί από την συναλλαγή του πελάτη και το username του.

Με το submit button θα τρέξουν από πίσω οι συναλλαγές και οι ταυτοποιήσεις και μόλις ολοκληρωθεί η επικοινωνία θα πετάξει ανάλογο μήνυμα, συναλλαγή ορθή ή λάθος στοιχεία, και θα πράξει ανάλογα ο admin.

Εικόνα 4.1.2.8 Φωτογραφία από την ιστοσελίδα που δημιουργήθηκε για την υλοποίηση της εφαρμογής. Από τον δημιουργό.

4.2 Σχετικές πληροφορίες Website

Το Website έχει δημιουργηθεί με WordPress και Elementor.

Έχουν παραμετροποιηθεί αρκετές λειτουργείες, όπως ότι ο χρήστης μπορεί να γίνει αυτόματα editor και όχι απλός guest ή ότι μόλις γίνει το submit button του Post δημοσιεύεται αυτόματα, χωρίς να χρειάζεται να υπάρχει κάποιος editor στο dashboard του WordPress ώστε να προχωράει εκείνος την δημοσίευση.

Βέβαια αυτό μπορεί να επηρεάσει στο να δημοσιεύονται μη σχετικά πράματα με το Website ή και ακόμη Myleware or Phishing content.

Είναι αρκετά φιλικό προς τον κάθε χρήστη.

4.3 Εφαρμογή Κώδικα

4.3.1 Ένωση Metamask με WebSite

```
<script src="https://cdn.ethers.io/lib/ethers-5.4.4.min.js"></script>
<script>
(function($) {
// Create a provider using Infura
var provider = new ethers.providers.JsonRpcProvider("https://soccer.montyontherun.gr/");

// Request access to the user's Ethereum accounts
ethers.request({ provider: provider }).then(function(accounts) {
var userAccount = accounts[0];
console.log('Connected to Metamask with account:', userAccount);

// Use the account for further operations or trigger events
}).catch(function(error) {
console.log('Failed to connect to Metamask:', error.message);
});
})(jQuery);
</script>
```

Αυτός ο κώδικας χρησιμοποιεί τη βιβλιοθήκη ethers για να αλληλεπιδράσει με το δίκτυο Ethereum. Ας εξηγήσουμε τον κώδικα με λεπτομέρειες:

- Η πρώτη γραμμή του κώδικα φορτώνει τη βιβλιοθήκη ethers από το CDN του ethers.io. Αυτή η βιβλιοθήκη παρέχει λειτουργίες για την αλληλεπίδραση με το δίκτυο Ethereum. Μέσω αυτής της γραμμής, ενσωματώνουμε το ethers.min.js αρχείο στην ιστοσελίδα μας.

- Η δεύτερη γραμμή αρχίζει ένα ανώνυμο συνάρτηση που δέχεται το `$` ως παράμετρο. Αυτό το `$` αναπαριστά το jQuery αντικείμενο, το οποίο μπορεί να χρησιμοποιηθεί για ευκολία στην προσπέλαση των DOM στοιχείων στη σελίδα.

- Εντός της ανώνυμης συνάρτησης, δημιουργούμε έναν πάροχο (provider) με τη βοήθεια του Infura. Ο πάροχος απαιτείται για να αλληλεπιδράσετε με το δίκτυο Ethereum. Στην περίπτωση μας, χρησιμοποιούμε το Infura ως τον πάροχο μας.
- Ορίζουμε τη διεύθυνση URL του Infura ("https://soccer.montyontherun.gr/") ως παράμετρο στην κατασκευή του πάροχου (provider).

- Στη συνέχεια, χρησιμοποιούμε τη μέθοδο `ethers.request()` για να ζητήσουμε πρόσβαση στους Ethereum λογαριασμούς του χρήστη. Αυτή η μέθοδος επιστρέφει ένα Promise που αντιστοιχεί σε έναν πίνακα με τους λογαριασμούς.

  Χρησιμοποιούμε τον πρώτο λογαριασμό από τον πίνακα και τον αποθηκεύουμε στη μεταβλητή `userAccount`.

- Τέλος, καταγράφουμε τον συνδεδεμένο λογαριασμό στην κονσόλα και εκτελούμε περαιτέρω ενέργειες ή εκδηλώσεις όπως απαιτείται.

```
<script src="PATH_TO_METAMASK_JS"></script>
<script>
  function connectMetaMask() {
    if (typeof window.ethereum !== 'undefined') {
      window.ethereum.enable();
    } else {
      alert("Please install MetaMask to connect.");
    }
  }
</script>
```

<button onclick="connectMetaMask()">Connect with MetaMask</button>

Ο παραπάνω κώδικας περιλαμβάνει τις εξής λειτουργίες:

- Φόρτωση του Metamask: Ο κώδικας χρησιμοποιεί την εντολή `<script>` για να φορτώσει το Metamask JavaScript SDK από την διεύθυνση PATH_TO_METAMASK_JS. Αυτή η βιβλιοθήκη είναι υπεύθυνη για την επικοινωνία με την επέκταση Metamask στον περιηγητή.
- Συνάρτηση για σύνδεση με το Metamask: Ο κώδικας ορίζει μια συνάρτηση με όνομα `connectMetaMask()`. Αυτή η συνάρτηση ελέγχει αν υπάρχει ορισμένο το αντικείμενο `window.ethereum`, που αντιπροσωπεύει την επέκταση Metamask στον περιηγητή. Εάν το αντικείμενο `window.ethereum` υπάρχει, τότε ο χρήστης καλείται να επιτρέψει τη σύνδεση με το Metamask μέσω της εντολής `window.ethereum.enable()`. Σε διαφορετική περίπτωση, εμφανίζεται ένα απλό μήνυμα προειδοποίησης.
- Κουμπί σύνδεσης με το Metamask: Ο κώδικας προσθέτει ένα κουμπί στη σελίδα με κείμενο "Connect with MetaMask". Όταν γίνεται κλικ στο κουμπί, καλείται η συνάρτηση `connectMetaMask()` που συνδέεται με το Metamask.

Συνολικά, ο κώδικας αυτός παρέχει έναν απλό τρόπο για τον χρήστη να συνδεθεί με το Metamask κάνοντας κλικ στο κουμπί "Connect with MetaMask".

4.3.2 Κώδικας Rewarding Tool

<script src="https://cdn.ethers.io/lib/ethers-5.4.4.min.js"></script>

<script>

(async function($) {

// Create a provider using Infura

var provider = new ethers.providers.JsonRpcProvider("https://soccer.montyontherun.gr/");

// Request access to the user's Ethereum accounts

var accounts = await ethereum.request({ method: 'eth_requestAccounts' });

var userAccount = accounts[0];

console.log('Connected to Metamask with account:', userAccount);

// Assuming you have the contract ABI and address defined

const contractABI = […]

const contractAddress = "0xd9145CCE52D386f254917e481eB44e9943F39138";

// Create an ethers instance using the provider

const ethersProvider = new ethers.providers.Web3Provider(provider);

```
// Create a contract instance
const contract = new ethers.Contract(contractAddress, contractABI, ethersProvider.getSigner());

// Function to make a post and award points
async function makePost() {
try {
// Make the function call to the smart contract
await contract.makePost();

// Perform additional actions or show success message
console.log("Post made successfully!");
} catch (error) {
// Handle error
console.error("Error making a post:", error);
}
}

// Function to make a comment and award points
async function makeComment() {
try {
// Make the function call to the smart contract
await contract.makeComment();

// Perform additional actions or show success message
console.log("Comment made successfully!");
} catch (error) {
// Handle error
console.error("Error making a comment:", error);
}
}
```

// Call the functions

makePost();

makeComment();

})(jQuery);

</**script**>

Ας ρίξουμε μια πιο λεπτομερή ματιά στον κώδικα:

1. Στην πρώτη γραμμή του κώδικα, έχουμε τη δήλωση `<script src="https://cdn.ethers.io/lib/ethers-5.4.4.min.js"></script>`. Αυτή η γραμμή φορτώνει τη βιβλιοθήκη ethers μέσω του CDN του ethers.io. Αυτό επιτρέπει στην ιστοσελίδα να χρησιμοποιήσει τις λειτουργίες της βιβλιοθήκης ethers.

2. Στη δεύτερη γραμμή, έχουμε την ανώνυμη συνάρτηση που ξεκινά με `(function($) {`. Αυτή η συνάρτηση περικλείει όλον τον κώδικα που ακολουθεί και παρέχει το `$` ως παράμετρο, το οποίο αναπαριστά το jQuery αντικείμενο.

3. Εντός της ανώνυμης συνάρτησης, έχουμε τη δημιουργία ενός παρόχου (provider) με τη χρήση του Infura. Η γραμμή κώδικα `var provider = new ethers.providers.JsonRpcProvider("https://soccer.montyontherun.gr/");` δημιουργεί ένα αντικείμενο παρόχου (provider) χρησιμοποιώντας την κλάση `JsonRpcProvider` της βιβλιοθήκης ethers. Παρέχουμε τη διεύθυνση URL του Infura ("https://soccer.montyontherun.gr/") ως παράμετρο στον κατασκευαστή του παρόχου.

4. Στη συνέχεια, χρησιμοποιούμε τη μέθοδο `ethers.request()` για να ζητήσουμε πρόσβαση στους Ethereum λογαριασμούς του χρήστη. Αυτή η μέθοδος επιστρέφει ένα Promise που αντιστοιχεί σε έναν πίνακα με τους λογαριασμούς. Χρησιμοποιούμε τον πρώτο λογαριασμό από τον πίνακα και τον αποθηκεύουμε στη μεταβλητή `userAccount`.

5. Τέλος, καταγράφουμε τον συνδεδεμένο λογαριασμό στην κονσόλα και εκτελούμε περαιτέρω ενέργειες ή εκδηλώσεις όπως απαιτείται.

Συνεχίζοντας με την εξήγηση:

1. Η γραμμή κώδικα <script src="https://cdn.ethers.io/lib/ethers-5.4.4.min.js"></script>` φορτώνει τη βιβλιοθήκη ethers από το CDN του ethers.io. Αυτό μας επιτρέπει να χρησιμοποιήσουμε τις λειτουργίες και τις κλάσεις της βιβλιοθήκης ethers στην ιστοσελίδα μας.

2. Η ανώνυμη συνάρτηση που ακολουθεί `(function($) { ... })(jQuery);` περικλείει ολόκληρο τον κώδικα και τον εκτελεί αμέσως. Η παράμετρος `$` αναπαριστά το αντικείμενο jQuery.

3. Στο εσωτερικό της ανώνυμης συνάρτησης, δημιουργούμε έναν πάροχο (provider) χρησιμοποιώντας την κλάση `JsonRpcProvider` της βιβλιοθήκης ethers. Ο πάροχος παρέχει πρόσβαση σε ένα Ethereum δίκτυο μέσω JSON-RPC.

4. Στη γραμμή κώδικα `ethers.request({ provider: provider }).then(function(accounts) { ... }).catch(function(error) { ... });`, χρησιμοποιούμε τη μέθοδο `request()` της βιβλιοθήκης ethers για να ζητήσουμε πρόσβαση στους Ethereum λογαριασμούς του χρήστη. Η μέθοδος `request()` επιστρέφει ένα Promise που αντιστοιχεί σε έναν πίνακα με τους λογαριασμούς.

5. Εάν η αίτηση για πρόσβαση στους λογαριασμούς είναι επιτυχής, η συνάρτηση που δίνεται ως παράμετρος στη μέθοδο `then()` εκτελείται. Μέσα σε αυτήν τη συνάρτηση, παίρνουμε τον πρώτο λογαριασμό από τον πίνακα των λογαριασμών και τον αποθηκεύουμε στη μεταβλητή `userAccount`. Έπειτα, εκτελούνται οποιεσδήποτε άλλες ενέργειες ή εκδηλώσεις χρειάζονται με τον συνδεδεμένο λογαριασμό.

6. Εάν η αίτηση για πρόσβαση στους λογαριασμούς αποτύχει, η συνάρτηση που δίνεται ως παράμετρος στη μέθοδο `catch()` εκτελείται. Εδώ μπορείτε να προσθέσετε κατάλληλο κώδικα για να αντιμετωπίσετε το σφάλμα που παρουσιάστηκε κατά τη σύνδεση με τους λογαριασμούς.

Αυτή είναι η λεπτομερής εξήγηση του κώδικα που μας δώσατε. Παρακαλώ σημειώστε ότι η εξήγηση αναφέρεται στον προκαθορισμένο τρόπο χρήσης της βιβλιοθήκης ethers. Εάν υπάρχουν προσαρμογές ή ειδικές απαιτήσεις για την εφαρμογή σας, θα πρέπει να προσαρμόσετε τον κώδικα αναλόγως.

Αυτός ο κώδικας εκτελεί διάφορες ενέργειες σχετικά με ένα έξυπνο συμβόλαιο που λειτουργεί στο δίκτυο Ethereum. Ας αναλύσουμε τις βασικές λειτουργίες του κώδικα:

- Συμπεριλαμβάνεται η βιβλιοθήκη ethers.js από το CDN `https://cdn.ethers.io/lib/ethers-5.4.4.min.js`. Αυτή η βιβλιοθήκη παρέχει βοηθητικές λειτουργίες για τη διεπαφή με το δίκτυο Ethereum.

- Ορίζεται μια συνάρτηση που εκτελείται αυτόματα και αναμένει την πληροφορία προμηθευτή JSON-RPC. Αυτός ο προμηθευτής επικοινωνεί με το δίκτυο Ethereum μέσω του κόμβου Infura που παρέχεται στο `https://soccer.montyontherun.gr/`.

- Ανακτώνται οι λογαριασμοί Ethereum του χρήστη που είναι συνδεδεμένοι με την επέκταση Metamask. Αυτό γίνεται με τη χρήση της `ethers.request()`, η οποία ζητά την πρόσβαση στους λογαριασμούς από τον παροχέα.

- Ο πρώτος λογαριασμός αποθηκεύεται στη μεταβλητή `userAccount` και εμφανίζεται στην κονσόλα.

- Ορίζεται το ABI του έξυπνου συμβολαίου. Το ABI περιέχει πληροφορίες σχετικά με τις συναρτήσεις και τις μεταβλητές του συμβολαίου.

- Ορίζεται η διεύθυνση του συμβολαίου.

- Δημιουργείται μια νέα ιστοσελίδα χρησιμοποιώντας το `Web3Provider` από τη βιβλιοθήκη ethers.js. Αυτή η ιστοσελίδα παρέχει μια διεπαφή με το δίκτυο Ethereum.

- Δημιουργείται μια νέα ιστοσελίδα χρησιμοποιώντας τη `Contract` από τη βιβλιοθήκη ethers.js. Αυτή η ιστοσελίδα αντιπροσωπεύει το έξυπνο συμβόλαιο που θα αλληλεπιδρά με το δίκτυο Ethereum.

- Ορίζονται δύο συναρτήσεις, η `makePost()` και η `makeComment()`, που χρησιμοποιούνται για την αλληλεπίδραση με το έξυπνο συμβόλαιο. Και οι δύο συναρτήσεις παίρνουν τη διεύθυνση του χρήστη από τη Metamask και καλούν τις αντίστοιχες συναρτήσεις στο έξυπνο συμβόλαιο.

- Οι συναρτήσεις `makePost()` και `makeComment()` καλούνται για να εκτελέσουν τις αντίστοιχες ενέργειες στο έξυπνο συμβόλαιο.

- Στη συνάρτηση `makePost()`, πραγματοποιείται η εξής διαδικασία:
  - Ανακτούνται οι λογαριασμοί Ethereum του χρήστη μέσω του `web3.eth.getAccounts()`.
  - Η πρώτη διεύθυνση λογαριασμού αποθηκεύεται στη μεταβλητή `userAddress`.
  - Καλείται η συνάρτηση `makePost()` του έξυπνου συμβολαίου χρησιμοποιώντας τη μέθοδο `send()`.
  - Εμφανίζεται στην κονσόλα το μήνυμα "Post made successfully!" αν η εκτέλεση ολοκληρωθεί με επιτυχία, αλλιώς εμφανίζεται το αντίστοιχο μήνυμα λάθους.

- Στη συνάρτηση `makeComment()`, πραγματοποιείται η εξής διαδικασία:
  - Ανακτούνται οι λογαριασμοί Ethereum του χρήστη μέσω του `web3.eth.getAccounts()`.
  - Η πρώτη διεύθυνση λογαριασμού αποθηκεύεται στη μεταβλητή `userAddress`.
  - Καλείται η συνάρτηση `makeComment()` του έξυπνου συμβολαίου χρησιμοποιώντας τη μέθοδο `send()`.
  - Εμφανίζεται στην κονσόλα το μήνυμα "Comment made successfully!" αν η εκτέλεση ολοκληρωθεί με επιτυχία, αλλιώς εμφανίζεται το αντίστοιχο μήνυμα λάθους.

- Τέλος, καλούνται οι συναρτήσεις `makePost()` και `makeComment()` για να εκτελέσουν τις αντίστοιχες ενέργειες στο έξυπνο συμβόλαιο.

Αυτός ο κώδικας επιτρέπει την αλληλεπίδραση με ένα έξυπνο συμβόλαιο Ethereum μέσω της Metamask και της βιβλιοθήκης ethers.js. Ο χρήστης μπορεί να κάνει μια ανάρτηση ή ένα σχόλιο και να ανταμείβεται με κάποιους πόντους.

Ο κώδικας χρησιμοποιεί τις διεπαφές του Metamask και της ethers.js για να αναγνωρίσει την τρέχουσα διεύθυνση του χρήστη, να αλληλεπιδράσει με το έξυπνο συμβόλαιο και να εμφανίσει μηνύματα κατά την εκτέλεση των λειτουργιών.

Επιπροσθέτως:

- Ο κώδικας ξεκινά με την συμπερίληψη της βιβλιοθήκης ethers.js από ένα CDN (Content Delivery Network). Αυτή η βιβλιοθήκη παρέχει ένα σύνολο εργαλείων για την αλληλεπίδραση με το δίκτυο Ethereum.

- Εντός της άμεσα κληθείσας έκφρασης συνάρτησης `(async function($) { ... })(jQuery);`, ο κώδικας περικλείεται για να εξασφαλιστεί ότι θα εκτελεστεί μόνο όταν η βιβλιοθήκη jQuery είναι διαθέσιμη.

- Δημιουργείται ένας πάροχος (provider) με τη χρήση της κλάσης `JsonRpcProvider` από το ethers.js, ο οποίος συνδέεται στο δίκτυο Ethereum μέσω της διεύθυνσης URL `"https://soccer.montyontherun.gr/"`. Αυτός ο πάροχος θα χρησιμοποιηθεί για την αλληλεπίδραση με το δίκτυο Ethereum.

- Ζητείται πρόσβαση στους Ethereum λογαριασμούς του χρήστη χρησιμοποιώντας τη μέθοδο `ethers.request()`, περνώντας ως παράμετρο τον πάροχο. Οι επιστρεφόμενοι λογαριασμοί αποθηκεύονται στη μεταβλητή `accounts`.

- Ο πρώτος λογαριασμός στον πίνακα `accounts` ανατίθεται στη μεταβλητή `userAccount`, που αντιπροσωπεύει το συνδεδεμένο λογαριασμό Metamask.

- Ο κώδικας ορίζει το ABI (Application Binary Interface) του συμβολαίου ως έναν πίνακα

- Στη συνέχεια, καθορίζεται η διεύθυνση του συμβολαίου στη μεταβλητή `contractAddress`.

- Δημιουργείται μια νέα ιστοσελίδα (web3) με τη χρήση της κλάσης `Web3Provider` από το ethers.js, που λαμβάνει ως παράμετρο το `window.ethereum`. Αυτός ο web3 παρέχει τη δυνατότητα αλληλεπίδρασης με το δίκτυο Ethereum.

- Δημιουργείται μια νέα περίπτωση (instance) του συμβολαίου με τη χρήση της κλάσης `Contract` από το ethers.js, που λαμβάνει ως παραμέτρους το ABI του συμβολαίου και τη διεύθυνση του συμβολαίου.

- Ορίζονται δύο ασύγχρονες συναρτήσεις `makePost()` και `makeComment()`, που χρησιμοποιούνται για την καταχώρηση ενός νέου post και ενός νέου σχολίου αντίστοιχα.

- Η συνάρτηση `makePost()` αναλαμβάνει να καλέσει τη μέθοδο `makePost()` του συμβολαίου μέσω της συνάρτησης `send()`, περνώντας ως παράμετρο τη διεύθυνση του χρήστη. Έπειτα, εκτελεί πρόσθετες ενέργειες ή εμφανίζει ένα μήνυμα επιτυχίας.

- Η συνάρτηση `makeComment()` λειτουργεί με παρόμοιο τρόπο, καλώντας τη μέθοδο `makeComment()` του συμβολαίου.

- Κατόπιν, οι συναρτήσεις `makePost()` και `make Comment()` καλούνται για να εκτελέσουν τις αντίστοιχες ενέργειες.

<script>

```
// Configure Web3 provider and contract address
const web3 = new Web3("web3_provider_url");
const contractAddress =     "account{0}": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4";
const rewardingToolContract = new web3.eth.Contract(rewardingToolABI, contractAddress);
// Function to add points when redeeming a reward
async function redeemReward() {
  try {
    const accounts = await web3.eth.requestAccounts();
    const userAccount = accounts[0];
    // Add points when redeeming the reward
    await rewardingToolContract.methods.addPoints().send({ from: userAccount });
    // Call the existing redeemReward function
    redeemReward();
    alert("Points added successfully!");
  } catch (error) {
    console.error(error);
```

```
    alert("Failed to add points");
  }
 }
 // Function to add points when rewarding a user
 async function rewardUser() {
  try {
   const accounts = await web3.eth.requestAccounts();
   const userAccount = accounts[0];
   // Add points when rewarding the user
   await rewardingToolContract.methods.addPoints().send({ from: userAccount });
   // Call the existing rewardUser function
   rewardUser();
   alert("Points added successfully!");
  } catch (error) {
   console.error(error);
   alert("Failed to add points");
  }
 }
</script>
```

Ο κώδικας περιλαμβάνει τις εξής λειτουργίες:

1. Ορίζει τον πάροχο Web3 και τη διεύθυνση του συμβολαίου:

Αυτός ο κώδικας χρησιμοποιεί το CDN (Content Delivery Network) για να φορτώσει τη βιβλιοθήκη Web3 με την έκδοση 1.5.2. Επίσης, ορίζει τη διεύθυνση του συμβολαίου "rewardingTool" ως "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4".

2. Ορίζει το αντικείμενο σύμβασης για το συμβόλαιο "rewardingTool":

Αυτή η γραμμή κώδικα δημιουργεί ένα αντικείμενο σύμβασης Web3 για το συμβόλαιο "rewardingTool" με τη χρήση της διεύθυνσης συμβολαίου που ορίστηκε προηγουμένως.

3. Ορίζει τις συναρτήσεις `redeemReward` και `rewardUser`:

Αυτές οι συναρτήσεις περιέχουν τον κώδικα που εκτελείται όταν ο χρήστης επιλέγει να ανταμείψει ή να ανταλλάξει μια ανταμοιβή. Ο κώδικας περιλαμβάνει την αποστολή συναλλαγής για να προσθέσει πόντους στο συμβόλαιο "rewardingTool" και στη συνέχεια εμφανίζει ένα μήνυμα επιτυχίας ή αποτυχίας.

4.3.3.1 Εργαλείο Ανταμοιβής – Ανάκτηση Ανταμοιβής (Redeem)

```
<script src="https://cdn.ethers.io/lib/ethers-5.4.4.min.js"></script>
<script>
(function($) {
    // Create a provider using Infura
    var provider = new ethers.providers.JsonRpcProvider("https://soccer.montyontherun.gr/");
    // Request access to the user's Ethereum accounts
    ethers.request({ provider: provider }).then(function(accounts) {
        var userAccount = accounts[0];
        console.log('Connected to Metamask with account:', userAccount);
        // Use the account for further operations or trigger events
    }).catch(function(error) {
        console.log('Failed to connect to Metamask:', error.message);
    });
})(jQuery);
</script>
```

Ο παραπάνω κώδικας περιλαμβάνει δύο τμήματα.

Πρώτο μέρος:

- <script src="https://cdn.ethers.io/lib/ethers-5.4.4.min.js"></script>
  Αυτή η γραμμή κώδικα εισάγει τη βιβλιοθήκη ethers στο έγγραφο σας, χρησιμοποιώντας το CDN (Content Delivery Network) στη διεύθυνση "https://cdn.ethers.io/lib/ethers-5.4.4.min.js". Η βιβλιοθήκη ethers παρέχει ένα περιβάλλον JavaScript για τη διεπαφή με το Ethereum και την αλυσίδα των μπλοκ. Αυτός ο κώδικας πρέπει να είναι παρών για να μπορέσει να χρησιμοποιηθεί η βιβλιοθήκη ethers στο επόμενο μέρος του κώδικα.

Δεύτερο μέρος:

- **<script>**
  (function($) {

// Create a provider using Infura

var provider = new ethers.providers.JsonRpcProvider("https://soccer.montyontherun.gr/");

```
// Request access to the user's Ethereum accounts
ethers.request({ provider: provider }).then(function(accounts) {
var userAccount = accounts[0];
console.log('Connected to Metamask with account:', userAccount);
// Use the account for further operations or trigger events}).catch(function(error)
 {console.log('Failed to connect to Metamask:', error.message);});})(jQuery);
```

</script>

Αυτό το τμήμα κώδικα δημιουργεί έναν πάροχο (provider) χρησιμοποιώντας την υπηρεσία Infura. Ο πάροχος είναι υπεύθυνος για την παροχή πρόσβασης σε έναν κόμβο του Ethereum δικτύου. Στη συνέχεια, ζητείται η πρόσβαση στους Ethereum λογαριασμούς του χρήστη. Αν η πρόσβαση είναι επιτυχής, το πρώτο λογαριασμό επιστρέφεται και εμφανίζεται μήνυμα στην κονσόλα με τον συνδεδεμένο λογαριασμό.

Σημείωση: Ο κώδικας αυτός υποθέτει ότι ο χρήστης έχει εγκαταστήσει και συνδέσει το Metamask στον περιηγητή του. Επίσης, απαιτείται να υπάρχει ένας έγκυρος πάροχος Infura και να παραμετροποιηθεί στην γραμμή κώδικα που δημιουργεί τον πάροχο (provider).

```
<script>
  document.addEventListener('DOMContentLoaded', function() {
    // Connect to web3 provider
    const web3 = new Web3(window.ethereum);
    // Configure contract address and ABI
    const contractAddress = 'rewarding_tool_contract_address';
    const contractABI = ... // Load the ABI of your rewarding tool contract
    const rewardingToolContract = new web3.eth.Contract(contractABI, contractAddress);
    // Redeem button click handler
    redeemButton.addEventListener('click', async () => {
      try {
        const accounts = await web3.eth.requestAccounts();
        const userAccount = accounts[0];
        // Add points when redeeming the reward
        await rewardingToolContract.methods.addPoints().send({ from: userAccount });
        // Call the existing redeemReward function or perform any other necessary actions
```

```
    redeemReward();
    alert('Points added successfully!');}
      catch (error) {
    console.error(error);
    alert('Failed to add points');
  }
 });
});
```

</script>

Ο παραπάνω κώδικας προσθέτει μια λειτουργία για το κλικ του κουμπιού με την ταυτότητα "redeemButton". Ακολουθούν λεπτομερείς περιγραφές για κάθε τμήμα του κώδικα:

- `document.addEventListener('DOMContentLoaded', function() { ... });`: Αυτό το μέρος του κώδικα εκτελείται όταν το DOM της σελίδας έχει φορτωθεί πλήρως.
- `const web3 = new Web3(window.ethereum);`: Αυτή η γραμμή κώδικας δημιουργεί μια νέα περιοχή Web3 χρησιμοποιώντας τον πάροχο (provider) window.ethereum. Αυτός ο πάροχος χρησιμοποιείται για να συνδεθείτε με τον τρέχοντα πάροχο Web3 του περιηγητή.
- `const contractAddress = 'rewarding_tool_contract_address';`: Εδώ ορίζετε τη διεύθυνση του συμβολαίου σας που θέλετε να αλληλεπιδράσετε μαζί του.
- `const contractABI = ... // Load the ABI of your rewarding tool contract`: Σε αυτό το σημείο πρέπει να φορτώσετε το ABI (Application Binary Interface) του συμβολαίου σας. Το ABI καθορίζει τη δομή και τις μεθόδους του συμβολαίου σας.
- `const rewardingToolContract = new web3.eth.Contract(contractABI, contractAddress);`: Αυτή η γραμμή κώδικας δημιουργεί μια νέα συμβολή (contract) για το συμβόλαιο με το χρησιμοποιούμενο ABI και τη διεύθυνση που καθορίσατε.
- `redeemButton.addEventListener('click', async () => { ... });`: Αυτή η γραμμή κώδικας προσθέτει έναν ακροατή κλικ για το κουμπί με την ταυτότητα "redeemButton". Όταν γίνει κλικ στο κουμπί, η συνάρτηση που περικλείεται στο "async () => { ... }" θα εκτελεστεί.
- `const accounts = await web3.eth.requestAccounts();`: Αυτή η γραμμή κώδικας ζητά την πρόσβαση στους Ethereum λογαριασμούς του χρήστη μέσω της μεθόδου `requestAccounts()` της περιοχής Web3.
- `const userAccount = accounts[0];`: Αποθηκεύεται ο πρώτος λογαριασμός Ethereum του χρήστη στη μεταβλητή `userAccount`.
- `await rewardingToolContract.methods.addPoints().send({ from: userAccount });`: Αυτή η γραμμή κώδικας καλεί τη μέθοδο `addPoints()` του συμβολαίου `rewardingToolContract` και προσθέτει πόντους στον χρήστη. Η μέθοδος εκτελείται με τη χρήση του λογαριασμού `userAccount`.
- `redeemReward();`: Καλείται η συνάρτηση `redeemReward()` για να εκτελέσει τυχόν άλλες ενέργειες που απαιτούνται μετά την προσθήκη των πόντων.
- `alert('Points added successfully!');`: Εμφανίζεται ένα απλό μήνυμα ειδοποίησης στον χρήστη με το κείμενο "Points added successfully!".

Αυτός ο κώδικας επιτρέπει στον χρήστη να προσθέσει πόντους όταν πατάει το κουμπί με την ταυτότητα "redeemButton”.

```
<script>
  document.addEventListener('DOMContentLoaded', function() {
    // Connect to web3 provider
    const web3 = new Web3(window.ethereum);
    // Configure contract address and ABI
    const contractAddress =
'0x518674ab2b227e5f11e9084f615d57663cde47bce1ba168b4c19c7ee22a73d70';
    const contractABI = ... // Load the ABI of your rewarding tool contract
    const rewardingToolContract = new
web3.eth.Contract(0x518674ab2b227e5f11e9084f615d57663cde47bce1ba168b4c19c7ee22a73d70,
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4);
    // Redeem button click handler
    redeemButton.addEventListener('click', async () => {
      try {
        const accounts = await web3.eth.requestAccounts();
        const userAccount = accounts[0];
        // Add points when redeeming the reward
        await rewardingToolContract.methods.addPoints().send({ from: userAccount });
        // Call the existing redeemReward function or perform any other necessary actions
        redeemReward();
        alert('Points added successfully!');
      } catch (error) {
        console.error(error);
        alert('Failed to add points');
      }
    });
  });
</script>
```

Ο παραπάνω κώδικας περιλαμβάνει τις εξής λειτουργίες:

- Σύνδεση με τον πάροχο Web3: Ο κώδικας συνδέεται με τον πάροχο Web3 μέσω του `window.ethereum`, που είναι η διεπαφή του περιηγητή με τον πάροχο Web3.
- Ορισμός διεύθυνσης και ABI του συμβολαίου: Ορίζεται η διεύθυνση του συμβολαίου (`contractAddress`) και το ABI του συμβολαίου (`contractABI`). Πρέπει να αντικαταστήσετε τις παραμέτρους με τις πραγματικές τιμές για το συμβόλαιο σας.
- Περιστολή κουμπιού "Redeem": Ο κώδικας περικλείει τη λειτουργία του κουμπιού "Redeem" μεταξύ του `redeemButton.addEventListener('click', async () => { ... });`. Όταν γίνει κλικ στο κουμπί, η συνάρτηση που περιλαμβάνεται μέσα θα εκτελεστεί.
- Ανάκτηση λογαριασμού: Ο κώδικας χρησιμοποιεί τη μέθοδο `web3.eth.requestAccounts()` για να ζητήσει την πρόσβαση στους Ethereum λογαριασμούς του χρήστη. Η πρώτη λογαριασμός αποθηκεύεται στη μεταβλητή `userAccount`.
- Προσθήκη πόντων: Η μέθοδος `rewardingToolContract.methods.addPoints().send({ from: userAccount })` καλεί τη μέθοδο `addPoints()` του συμβολαίου `rewardingToolContract` για να προσθέσει πόντους στον χρήστη. Η μέθοδος εκτελείται με τη χρήση του λογαριασμού `userAccount`.
- Κλήση επιπλέον λειτουργιών: Μετά την προσθήκη των πόντων, μπορείτε να καλέσετε οποιαδήποτε επιπλέον λειτουργία που χρειάζεται να εκτελεστεί μετά από την προσθήκη των πόντων.
- Εμφάνιση ειδοποίησης: Ανάλογα με το αποτέλεσμα της εκτέλεσης, εμφανίζεται ένα απλό μήνυμα ειδοποίησης με το κείμενο "Points added successfully!" ή "Failed to add points!".

Συνολικά,

ο κώδικας επιτρέπει τη σύνδεση με τον πάροχο Web3, την επικοινωνία με ένα συμβόλαιο και την εκτέλεση μιας λειτουργίας (προσθήκη πόντων) όταν γίνεται κλικ σε ένα κουμπί "Redeem".

```
<style>
.blockchain-button {
  display: full-block;
  background-color: #4CAF50;
  color: white;
  padding: 10px 20px;
  font-size: 16px;
  border: none;
  cursor: pointer;
}
.blockchain-button:hover {
  background-color: #45a049;
}
</style>
```

```
<div class="blockchain-button">Click Me!</div>
<script>
jQuery(document).ready(function($) {
    // Function to check if a contract exists
    async function checkContractExists(provider, contractAddress) {
        // Add your code for checking contract existence here
        // Make sure to return a boolean value indicating whether the contract exists or not
    }
    // Function to initialize the contract and perform actions
    async function initializeContract() {
        const provider = window.ethereum;
        const contractAddress = 'YOUR_CONTRACT_ADDRESS';
        const contractABI = []; // Add your contract ABI here
        try {
            // Check if the user is connected to MetaMask
            const accounts = await provider.request({
                method: "eth_requestAccounts",
            });
            if (accounts.length === 0) {
                throw new Error("User is not connected to MetaMask");
            }
            // Create a Web3Provider using MetaMask's provider
            const ethersProvider = new ethers.providers.Web3Provider(provider);
            // Check if the contract exists
            const itExists = await checkContractExists(ethersProvider, contractAddress);
            if (!itExists) {
                throw new Error(`Contract with address: ${contractAddress}, does not exist!`);
            }
            // Get the signer from the provider
            const signer = await ethersProvider.getSigner();
            // Initialize the contract
            const contractInstance = new ethers.Contract(contractAddress, contractABI, signer);
```

```
        // Step 1: Approve transaction
        const approveTx = await
contractInstance.approve('0x76be496Efe62dC94c52bE475A20159ab215deedA', 50);
        await approveTx.wait();
        // Step 2: Get random number
        await getRandomNumber();
        // Step 3: Claim reward transaction
        const rewardClaimTx = await contractInstance.productClaimer(0);
        await rewardClaimTx.wait();
        // Step 4: Display success message
        console.log('Reward claimed successfully!');
        // Additional steps or actions can be added here
        console.log('All steps completed successfully!');
    } catch (error) {
        console.error('Error executing code snippets:', error);
    }
}
// Button click event handler
$('.blockchain-button').on('click', function(e) {
    e.preventDefault();
    initializeContract();
});
// Function to get random number
async function getRandomNumber() {
    try {
        const response = await fetch("https://express-oracle-rewarding-tool-
02e3806b3ce0.herokuapp.com/random-number");
        const data = await response.json();
        console.log(data);
    } catch (error) {
        console.error('Error fetching random number:', error);
    }
```

```
    }

    // Enqueue the JavaScript code
    (function($) {
        function enqueue_blockchain_script() {
            if (typeof elementorFrontend !== 'undefined') {
                elementorFrontend.hooks.addAction('frontend/element_ready/button.default',
function($element) {
                    var container = $element[0].getElementsByClassName('blockchain-button');
                    if (container.length > 0) {
                        var scriptTag = document.createElement('script');
                        scriptTag.src = '/path/to/your/custom.js';
                        scriptTag.type = 'text/javascript';
                        document.body.appendChild(scriptTag);
                    }
                });
            }
        }
        enqueue_blockchain_script();
    })(jQuery);
});
```

</script>

Ο παραπάνω κώδικας περιλαμβάνει τις εξής λειτουργίες:

- Ορισμός του στυλ για το κουμπί: Ο κώδικας περιγράφει το στυλ του κουμπιού με τη χρήση CSS. Ορίζονται χαρακτηριστικά όπως το χρώμα φόντου, οι διαστάσεις, η γραμματοσειρά και άλλα.
- Προετοιμασία του HTML κώδικα για το κουμπί: Ο κώδικας δημιουργεί ένα `<div>` στο οποίο εφαρμόζεται η CSS κλάση "blockchain-button". Αυτό το κουμπί μπορεί να προσαρμοστεί κατάλληλα με το περιεχόμενο που επιθυμείτε.
- Αρχικοποίηση του συμβολαίου και εκτέλεση ενεργειών: Ο κώδικας περιλαμβάνει μια σειρά λειτουργιών που εκτελούνται όταν γίνεται κλικ στο κουμπί "Click Me!". Αυτές οι λειτουργίες περιλαμβάνουν τη σύνδεση με τον πάροχο Web3, τον έλεγχο της ύπαρξης του συμβολαίου, την αρχικοποίηση του συμβολαίου, την εκτέλεση βημάτων και την εμφάνιση μηνυμάτων επιτυχίας ή σφάλματος.

- Αίτηση για απόκτηση τυχαίου αριθμού: Ο κώδικας περιλαμβάνει μια λειτουργία που αιτείται έναν τυχαίο αριθμό από έναν εξωτερικό πάροχο. Αυτό μπορεί να χρησιμοποιηθεί για διάφορους σκοπούς, ανάλογα με την εφαρμογή.
- Ουρά JavaScript κώδικα: Ο κώδικας περιέχει μια ουρά JavaScript που ενσωματώνεται στον κώδικα της σελίδας μέσω του jQuery. Αυτή η ουρά JavaScript είναι υπεύθυνη για την εκτέλεση του κώδικα όταν η σελίδα φορτώνεται.

Συνολικά, ο κώδικας αυτός δημιουργεί ένα κουμπί στη σελίδα που, όταν γίνεται κλικ, συνδέεται με τον πάροχο Web3, αλληλεπιδρά με ένα συμβόλαιο και εκτελεί μια σειρά λειτουργιών, ενδεχομένως επικοινωνώντας με εξωτερικούς παρόχους δεδομένων.

[37][38][39]

4.3.2.2 Εργαλείο Ανταμοιβής – Ανάκτηση Ανταμοιβής (Redeem) Button

```
<div class="custom-form">
  <div class="form-group">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>
  </div>
  <div class="form-group">
    <label for="code">Generated Code:</label>
    <input type="text" id="code" name="code" required>
  </div>
  <div class="form-group">
    <button id="submit" class="blockchain-button validate-button">Submit</button>
  </div>
</div>
<script>
jQuery(document).ready(function($) {
  // Validate button click event handler
  $('.validate-button').on('click', async function(e) {
    e.preventDefault();
    try {
```

```
// Call the validation transaction
const username = $('#username').val();
const code = $('#code').val();
// Function to check if a contract exists
async function checkContractExists(provider, contractAddress) {
  // Add your code for checking contract existence here
  // Make sure to return a boolean value indicating whether the contract exists or not
}
// Function to initialize the contract and perform actions
async function initializeContract() {
  const provider = window.ethereum;
  const contractAddress = 'YOUR_CONTRACT_ADDRESS';
  const contractABI = []; // Add your contract ABI here

  try {
    // Check if the user is connected to MetaMask
    const accounts = await provider.request({
      method: "eth_requestAccounts",
    });
    if (accounts.length === 0) {
      throw new Error("User is not connected to MetaMask");
    }
    // Create a Web3Provider using MetaMask's provider
    const ethersProvider = new ethers.providers.Web3Provider(provider);
    // Check if the contract exists
    const itExists = await checkContractExists(ethersProvider, contractAddress);
    if (!itExists) {
      throw new Error(`Contract with address: ${contractAddress}, does not exist!`);
    }
    // Get the signer from the provider
    const signer = await ethersProvider.getSigner();
    // Initialize the contract
```

```
        const contractInstance = new ethers.Contract(contractAddress, contractABI, signer);
        // Step 1: Approve transaction
        const approveTx = await
contractInstance.approve('0x76be496Efe62dC94c52bE475A20159ab215deedA', 50);
        await approveTx.wait();
        // Step 2: Get random number
        await getRandomNumber();
        // Step 3: Claim reward transaction
        const rewardClaimTx = await contractInstance.productClaimer(0);
        await rewardClaimTx.wait();

        // Step 4: Display success message
        console.log('Reward claimed successfully!');

        // Additional steps or actions can be added here
        console.log('All steps completed successfully!');
      } catch (error) {
        console.error('Error executing code snippets:', error);
      }
    }
    // Function to get random number
    async function getRandomNumber() {
      try {
        const response = await fetch("https://express-oracle-rewarding-tool-
02e3806b3ce0.herokuapp.com/random-number");
        const data = await response.json();
        console.log(data);
      } catch (error) {
        console.error('Error fetching random number:', error);
      }
    }
    const validationTx = await redeemerValidator(username, code, 0);
```

```
    const receipt = await validationTx.wait();
    if (receipt.status === 1) {
      // Show success message
      console.log('Validation successful!');
      initializeContract();
    } else {
      // Show error message
      console.log('Validation failed!');
    }
    // Wait for the transaction and display message accordingly
  } catch (error) {
    console.error('Error executing validation transaction:', error);
  }
 });
});
</script>
```

Ο κώδικας αυτός περιγράφει ένα προσαρμοσμένο φόρμα εισαγωγής δεδομένων με ένα κουμπί υποβολής. Ας αναλύσουμε τις λειτουργίες του:

1. Φόρμα εισαγωγής δεδομένων: Ο κώδικας περικλείει τη φόρμα μέσα σε ένα `<div>` με την κλάση "custom-form". Η φόρμα περιλαμβάνει δύο πεδία εισαγωγής με ετικέτες ("Username" και "Generated Code") και ένα κουμπί υποβολής.
2. Εκδήλωση κλικ στο κουμπί υποβολής: Ο κώδικας προσθέτει έναν ακροατή στο γεγονός κλικ στο κουμπί υποβολής με την κλάση "validate-button". Όταν γίνεται κλικ, η ακόλουθη συνάρτηση εκτελείται:
    - Λήψη των τιμών από τα πεδία εισαγωγής: Ο κώδικας ανακτά τις τιμές που έχουν εισαχθεί στα πεδία "Username" και "Generated Code".
    - Εκτέλεση συναλλαγής επικύρωσης: Ο κώδικας καλεί τη συνάρτηση `redeemerValidator()` για να εκτελέσει μια συναλλαγή επικύρωσης, περνώντας το όνομα χρήστη, τον κωδικό και την τιμή 0 ως παραμέτρους. Αυτή η συνάρτηση αναλαμβάνει να ελέγξει την εγκυρότητα των δεδομένων και να επιστρέψει μια συναλλαγή μεταβλητής.
    - Αναμονή για τη συναλλαγή και εμφάνιση ανάλογου μηνύματος: Ο κώδικας περιμένει για την ολοκλήρωση της συναλλαγής επικύρωσης και ελέγχει την κατάσταση της συναλλαγής. Αν η συναλλαγή έχει επιτυχή κατάσταση (status === 1), τότε εμφανίζεται ένα μήνυμα επιτυχίας ("Validation successful!") και καλείται η συνάρτηση `initializeContract()` για να αρχικοποιηθεί ο συμβόλαιος και να εκτελεστούν ενέργειες. Αν η συναλλαγή έχει αποτύχει κατάσταση, εμφανίζεται ένα μήνυμα σφάλματος ("Validation failed!").

3. Συναρτήσεις επικύρωσης και αρχικοποίησης συμβολαίου: Ο κώδικας περιέχει δύο ακόμη συναρτήσεις, την `checkContractExists()` και την `initializeContract()`, που αναλαμβάνουν τον έλεγχο της ύπαρξης του συμβολαίου και την αρχικοποίησή του αντίστοιχα. Αυτές οι συναρτήσεις χρησιμοποιούν τον πάροχο Web3Provider για να αλληλεπιδράσουν με το Metamask και το συμβόλαιο. Επίσης, περιλαμβάνουν βήματα όπως η έγκριση συναλλαγής, η λήψη τυχαίου αριθμού και η εκτέλεση συναλλαγής απαίτησης ανταμοιβής.
4. Επικοινωνία με τον εξυπηρετητή: Ο κώδικας περιέχει τη συνάρτηση `getRandomNumber()` που χρησιμοποιεί τη μέθοδο `fetch()` για να αιτηθεί έναν τυχαίο αριθμό από έναν εξυπηρετητή. Αυτή η συνάρτηση εκτελείται μέσα από τη συνάρτηση `initializeContract()` και μπορεί να προσαρμοστεί για να ανταποκρίνεται στις απαιτήσεις της εφαρμογής σας.

Συνολικά, ο κώδικας αυτός παρέχει μια λειτουργικότητα για την εισαγωγή δεδομένων σε μια φόρμα και την εκτέλεση συναλλαγών με το Metamask μέσω ενός έλεγχου επικύρωσης. Ανάλογα με τις απαιτήσεις της εφαρμογής σας, μπορείτε να προσαρμόσετε τις συναρτήσεις επικύρωσης και αρχικοποίησης για να περιλαμβάνουν τις κατάλληλες ενέργειες και να αλληλεπιδρούν με το συμβόλαιο σας.

Επιπλέον:

1. Εκδήλωση κλικ στο κουμπί υποβολής:
    - Όταν γίνεται κλικ στο κουμπί υποβολής με την κλάση "validate-button", εκτελείται η ακόλουθη συνάρτηση ως αντίδραση:
2. Συνάρτηση επικύρωσης:
    - Ανάλυση συνάρτησης "redeemerValidator()":
    - Η συνάρτηση αποθηκεύει τις τιμές των πεδίων εισαγωγής "Username" και "Generated Code".
    - Υπάρχει η ασύγχρονη κλήση συνάρτησης `redeemerValidator()` με τα παραπάνω δεδομένα. Αυτή η συνάρτηση μπορεί να προσαρμοστεί ανάλογα με τις απαιτήσεις της εφαρμογής.
    - Αναμένει την ολοκλήρωση της συνάρτησης `redeemerValidator()` και αποθηκεύει το αποτέλεσμα στη μεταβλητή `validationTx`.
    - Αν η συναλλαγή έχει επιτυχή κατάσταση (status === 1), τότε εμφανίζεται ένα μήνυμα επιτυχίας ("Validation successful!") και καλείται η συνάρτηση `initializeContract()` για να αρχικοποιηθεί ο συμβόλαιος και να εκτελεστούν ενέργειες.
    - Αν η συναλλαγή έχει αποτύχει κατάσταση, εμφανίζεται ένα μήνυμα σφάλματος ("Validation failed!").
3. Συνάρτηση αρχικοποίησης συμβολαίου:
    - Ανάλυση συνάρτησης "initializeContract()":
    - Ο κώδικας αρχικά ελέγχει αν ο χρήστης είναι συνδεδεμένος με το Metamask και αν δεν είναι, εμφανίζεται ένα μήνυμα λάθους.
    - Δημιουργείται ένας πάροχος Web3Provider χρησιμοποιώντας τον πάροχο του Metamask.
    - Ο κώδικας ελέγχει αν το συμβόλαιο με τη διεύθυνση `contractAddress` υπάρχει. Αυτή η λειτουργία πρέπει να προσαρμοστεί ανάλογα με την υλοποίηση του συμβολαίου.
    - Αν υπάρχει το συμβόλαιο, δημιουργείται ένας signer από τον πάροχο και αρχικοποιείται το συμβόλαιο χρησιμοποιώντας τη διεύθυνση `contractAddress` και το `contractABI`.
    - Ακολουθεί η εκτέλεση βημάτων, όπως η έγκριση μιας συναλλαγής, η λήψη ενός τυχαίου αριθμού και η εκτέλεση μιας συναλλαγής απαίτησης ανταμοιβής.

- Ο κώδικας εμφανίζει ένα μήνυμα επιτυχίας ("Reward claimed successfully!") και μπορούν να προστεθούν επιπλέον ενέργειες ή βήματα εδώ.

4. Συνάρτηση λήψης τυχαίου αριθμού:
    - Ανάλυση συνάρτησης "getRandomNumber()":
    - Η συνάρτηση χρησιμοποιεί τη μέθοδο `fetch()` για να αιτηθεί έναν τυχαίο αριθμό από έναν εξυπηρετητή.
    - Το αποτέλεσμα επιστρέφεται ως αντικείμενο δεδομένων JSON και εμφανίζεται στην κονσόλα.

Με τον παρεχόμενο κώδικα, δημιουργείται μια φόρμα που ζητά από τον χρήστη ένα όνομα χρήστη και έναν κωδικό, εκτελεί μια επικύρωση στο Metamask και στη συνέχεια εκτελεί μια ακολουθία ενεργειών, όπως η έγκριση μιας συναλλαγής, η λήψη ενός τυχαίου αριθμού και η εκτέλεση μιας συναλλαγής απαίτησης ανταμοιβής. Ανάλογα με τις απαιτήσεις σας, μπορείτε να προσαρμόσετε τις συναρτήσεις επικύρωσης και αρχικοποίησης για να περιλαμβάνουν τις αντίστοιχες ενέργειες και να αλληλεπιδρούν με το συμβόλαιο σας.

[37][38][39]

4.3.4 Login Verification

```
<script type="text/javascript">
    jQuery(document).ready(function($) {
        if ($('.login-success-popup').length > 0) {
            // Replace 'Your message here' with your desired message
            var message = 'You have logged in successfully!';
            // Display the message in a popup
            alert(message);
        }
    });
</script>
```

- Ο κώδικας που παρουσιάζεται είναι ένα απλό σενάριο JavaScript που εκτελείται όταν η σελίδα φορτώνεται και χρησιμοποιεί τη βιβλιοθήκη jQuery.
- Αν η σελίδα περιέχει ένα στοιχείο με την κλάση 'login-success-popup', τότε ο κώδικας εμφανίζει ένα απλό αναδυόμενο παράθυρο με ένα μήνυμα επιτυχίας. Το μήνυμα που εμφανίζεται στο παράθυρο είναι 'You have logged in successfully!' (Έχετε συνδεθεί με επιτυχία!).

Ο κώδικας αυτός μπορεί να χρησιμοποιηθεί για να εμφανίσει ένα μήνυμα επιτυχίας στον χρήστη όταν συνδεθεί στο σύστημα. Ανάλογα με την υλοποίηση της σελίδας, μπορείτε να τροποποιήσετε το μήνυμα ή να προσθέσετε περισσότερη λειτουργικότητα.

4.3.5 Redeen Verification

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.19 <0.9.0;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";

import "@openzeppelin/contracts/access/AccessControl.sol";

import "hardhat/console.sol";

// import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";

import "./IOracle.sol"; // Importing the Oracle's Interface

contract RewardingTool is AccessControl {

    // Contract Address (GENERA - Network): 0x300302fEc3D905eb66Cb7743C636F8741B72dB3a

    // using SafeERC20 for IERC20;

    // -- Global Score - START

    uint public baseReward;

    IERC20 private token; // This contract is our ERC-20 MGS Tokens

    IOracle private oracle; // This contract provides randomness

    address public contractAddress;

    address public owner;

    // address public erc20_addr;


    mapping(address => User) public users;

    mapping(string => address) public userNames;

    mapping(string => Service) public services;

    mapping(uint => Product) public products;


    uint public numUsers;

    uint public numServices;

    uint public numProducts;

    uint public numPendingProds;

    // bytes32 public constant DEFAULT_ADMIN_ROLE =

    //     keccak256("DEFAULT_ADMIN_ROLE");

    bytes32 public constant MANAGER_ROLE = keccak256("MANAGER_ROLE");

    // -- Global Score- END
```

```
constructor(IERC20 _token, /*address _tokenAddress,*/ IOracle _oracle) {
    baseReward = 10; // Applying a default value
    contractAddress = address(this);
    // erc20_addr = _tokenAddress;
    owner = msg.sender;
    token = _token;
    oracle = _oracle;
    _setupRole(DEFAULT_ADMIN_ROLE, owner);
    _setupRole(MANAGER_ROLE, address(this));
    // Just some automations cuz the contract is still under dev
    // Creating Services and ServiceEvents
    string[2] memory defaultServives = ["forum", "game"];
    string[3] memory defaultForumEvents = [
        "submitComment",
        "voteOnComment",
        "voteOnPost"
    ];
    uint8[3] memory defaultForumEventsMultis = [10, 3, 5];

    string[4] memory defaultGameEvents = [
        "TownHallUpgrade",
        "cardCreation",
        "successCardSale",
        "rankBased"
    ];
    uint8[4] memory defaultGameEventsMultis = [10, 3, 5, 1];

    for (uint i = 0; i < 2; i++) {
        string memory service = defaultServives[i];
        createServiceInternal(service);

        if (areEqual(service, "forum")) {
```

```
            for (uint j = 0; j < 3; j++) {
                createEventInternal(
                    defaultForumEvents[j],
                    "forum",
                    defaultForumEventsMultis[j]
                );
            }
        } else if (areEqual(service, "game")) {
            for (uint j = 0; j < 3; j++) {
                createEventInternal(
                    defaultGameEvents[j],
                    "game",
                    defaultGameEventsMultis[j]
                );
            }
        }
    }
    // Creating some Products
// -- Events Section - START
event UserCreation(
    uint indexed id,
    address indexed account,
    string indexed name
); // OK
event ServiceCreation(uint indexed id, string indexed name); // OK
event EventCreation(
    uint indexed id,
    string indexed name,
    string indexed serviceName
); // Ok

event PointsGained(address indexed account, uint indexed amount); // OK
```

```
    event PointsRedeemed(
        address indexed account,
        string indexed serviceName,
        uint indexed _productPrice,
        uint _totalPoints_before
    ); // ok

    event ProductCreation(
        string indexed name,
        uint indexed price,
        uint indexed amount,
        string location
    ); //TODO: Test this!

    event ProductAquired(uint indexed id, string indexed name); //TODO: Test this!

    event ProductClaimed(
        uint indexed id,
        string indexed name,
        uint indexed price
    ); //TODO: Test this!

    // -- Events Section - END

    // -- Structs Section - START

    struct Product {
        // Ex. It can be a physical, real or digital redeemable reward (brelock, in-game currency, free musceum tickets
        uint id;
        uint price;
```

```
        uint32 amount; // MAX_VALUE uint32, dec: 4,294,967,295 || hex: 0xFFFFFFFF
        bool isEmpty;
        bool isInfinite;
        bool isDisabled;
        string name;
        string location;
    }

    struct PendingProduct {
        uint id;
        uint productId;
        bytes32 collectionHash; // keccak256(username + productId + 6-digit nonce)
        bool isRedeemed;
    }
    modifier onlyOwner() {
        require(
            hasRole(DEFAULT_ADMIN_ROLE, msg.sender),
            "RewardingTool.sol: caller is not the Owner"
        );
        _;
    }
    modifier managerLevel() {
        require(
            hasRole(MANAGER_ROLE, msg.sender) ||
                hasRole(DEFAULT_ADMIN_ROLE, msg.sender),
            "RewardingTool.sol: caller is not a Manager or the Owner"
        );
        _;
    }       );
{…}       /*
            Example Code:
            // Create a new instance of the contract
```

```
        const tokenContract = new ethers.Contract(tokenAddress, erc20Abi, signer);
        const amountToApprove = ethers.utils.parseUnits('10.0', 18); // Change the amount as needed, '18' is the typical number of decimal places for ERC-20 tokens
        // Call the approve function
        const approvalTx = await tokenContract.approve(spenderAddress, amountToApprove);
        // Wait for it to be mined
        setIsLoading(true); // For React Users
        await approvalTx.wait();
        console.log('Tokens approved');
    */

    // 5. If user can buy the Product and allows the contract to transfer token on his/her behalf, transfer the required amount of tokens (particular_product.price) from his/her account to the ERC-20 Contract.
    token.transferFrom(msg.sender, address(this), particular_product.price);
    // 6. Generate an 6-digit Nonce in order to create a "collectionHash"
    uint32 randomNonce = oracle.randomNumber();
    // 7. Craete the "collectionHash"
    bytes32 _collectionHash = hashValues(
        current_user.name,
        particular_product.id,
        randomNonce
    );
    // 8. Create the PendingProduct object
    PendingProduct memory pendingProd = PendingProduct(
        numPendingProds,
        particular_product.id,
        _collectionHash,
        false
    );
    numPendingProds += 1;
{…}
  // -- Getter Functions - START
```

```
    function viewEvent(
        string memory _serviceName,
        string memory _eventName
    ) public view returns (ServiceEvent memory) {
        Service storage current_service = services[_serviceName];
        ServiceEvent storage current_event = current_service.events[_eventName];
        return current_event;
    }
    function viewUserPoints(address _userAddr) public view returns (uint) {
        // User storage current_user = users[_userAddr]; // Getting the requested user by using his/her wallet address
        return token.balanceOf(_userAddr);
    }
    function viewYourPoints() public view returns (uint) {
        // User storage you = users[msg.sender]; // Getting the requested user by using his/her wallet address
        return token.balanceOf(msg.sender);
    }
    function viewYourUnclaimedProds()
        public
        view
        returns (PendingProduct[] memory)
    {
        User storage you = users[msg.sender]; // Getting the requested user by using his/her wallet address
        return you.pendingProducts;
    }
    function getUserProducts(
        string memory _userName
    ) public view returns (PendingProduct[] memory userPendingProducts) {
        address currentUserAddr = userNames[_userName];
        User storage currentUser = users[currentUserAddr];
        require(
```

```
        bytes(_userName).length > 0,
        "getUserProducts: Invalid/Wrong Input"
    );
    require(
        bytes(currentUser.name).length > 0,
        "getUserProducts: User does not exist"
    );
    return currentUser.pendingProducts;
}
function getUserProducts(
    address _userAddr
) public view returns (PendingProduct[] memory userPendingProducts) {
    User storage currentUser = users[_userAddr]; // Getting the requested user by
    require(
        bytes(currentUser.name).length > 0,
        "getUserProducts: User does not exist"
    );
    return currentUser.pendingProducts;
}
function getAllProducts() public view returns (Product[] memory) {
    Product[] memory allProducts = new Product[](numProducts);
    for (uint i = 0; i < numProducts; i++) {
        allProducts[i] = products[i];
    }
    return allProducts;
}
// -- Getter Functions - END
// -- AccessControl Functions - START
// Function to assign Manager access level category
function assignManagerRole(address _account) public onlyOwner {
    grantRole(MANAGER_ROLE, _account);
    User storage currentUser = users[_account];
```

```
    currentUser.accessLevel = "manager";
}
// Function to assign Custom access level category
function assignRole(bytes32 _rule, address _account) public onlyOwner {
    grantRole(_rule, _account);
}
// Function to create new Access level categoryies
function setupRole(bytes32 _rule, address _account) public onlyOwner {
    _setupRole(_rule, _account);
}
// -- AccessControl Functions - END
// -- Utility Functions - START
function pointsCalc(uint _multiplier) internal view returns (uint) {
    return baseReward * _multiplier;
}
// This should be in the Solidity's std library...
// @dev Checks if 2 strings are equal. Returns true or false.
// @notice If you want to check if a string is set or not,
function areEqual(
    string memory string_1,
    string memory string_2
) internal pure returns (bool) {
    require(
        bytes(string_1).length >= 0 && bytes(string_2).length >= 0,
        "From: areEqual, probably inserted only one argument, two are required"
    );
    return
        keccak256(abi.encodePacked(string_1)) ==
        keccak256(abi.encodePacked(string_2));
}
function hashValues(
    string memory name,
```

```
        uint256 productId,
        uint nonce
    ) public pure returns (bytes32) {
        return keccak256(abi.encodePacked(name, productId, nonce));
    }
    // -- Utility Functions - END
    function checkOwnerRole(address _account) public view returns (bool) {
        return hasRole(DEFAULT_ADMIN_ROLE, _account);
    }
}
```

Ας αναλύσουμε τον παρεχόμενο κώδικα Solidity:

1. Η συμβολοσειρά `SPDX-License-Identifier: GPL-3.0` προσδιορίζει την άδεια χρήσης για το συμβόλαιο.
2. Εισαγωγή απαραίτητων βιβλιοθηκών και συμβολαίων:
    - Το συμβόλαιο εισάγει τις βιβλιοθήκες `IERC20` και `AccessControl` από την OpenZeppelin.
    - Εισάγει τη διεπαφή `IOracle` από το συμβόλαιο Oracle.
3. Το συμβόλαιο `RewardingTool` επεκτείνει το συμβόλαιο `AccessControl`:
    - Ορίζει μεταβλητές όπως `baseReward`, `token`, `oracle`, `contractAddress`, `owner`, `numUsers`, `numServices`, `numProducts`, `numPendingProds`.
    - Δημιουργεί δομές `User`, `ServiceEvent`, `Service` και `Product`.
    - Ορίζει μεταβλητές αποθήκευσης όπως `users`, `userNames`, `services`, `products`.
    - Περιέχει μετασχηματιστές και μεταβλητές ρόλων για τον έλεγχο πρόσβασης.
    - Ορίζει συναρτήσεις για την προσθήκη και αφαίρεση πόντων, την αγορά προϊόντων, τη δημιουργία χρηστών, υπηρεσιών, γεγονότων και προϊόντων.
    - Περιέχει επίσης λειτουργίες για την ανάγνωση πόντων χρήστη, ανάγνωση εκκρεμών προϊόντων χρήστη και ανάγνωση όλων των προϊόντων.
    - Περιέχει επίσης λειτουργίες για τη διαχείριση των ρόλων των χρηστών και των κανόνων πρόσβασης.
5. Τέλος, υπάρχουν μερικές βοηθητικές συναρτήσεις όπως `pointsCalc` για τον υπολογισμό των πόντων, `areEqual` για τη σύγκριση δύο συμβολοσειρών και `hashValues` για τον υπολογισμό του `collectionHash`.
6. Το έξυπνο συμβόλαιο "RewardingTool" είναι ένα συμβόλαιο Solidity που χρησιμοποιείται για την επιβράβευση χρηστών με επιπρόσθετα κέρδη (MGS Score Points) για συγκεκριμένες ενέργειες που πραγματοποιούν σε ένα σύστημα.

Το συμβόλαιο υλοποιεί τις εξής λειτουργίες:

1. Δημιουργία και διαχείριση χρηστών: Οι χρήστες μπορούν να δημιουργήσουν έναν λογαριασμό και να συνδέσουν το όνομά τους με τη διεύθυνση πορτοφολιού τους. Επίσης, το συμβόλαιο παρέχει δυνατότητες διαχείρισης των χρηστών όπως η εκχώρηση ρόλων και η ανάθεση επιπέδων πρόσβασης.

2. Δημιουργία και διαχείριση υπηρεσιών: Οι διαχειριστές μπορούν να δημιουργήσουν νέες υπηρεσίες που οι χρήστες μπορούν να χρησιμοποιήσουν για να κερδίσουν πόντους. Κάθε υπηρεσία έχει μια λίστα από γεγονότα (events) για τα οποία οι χρήστες μπορούν να κερδίσουν πόντους.
3. Εκχώρηση πόντων: Οι χρήστες μπορούν να κερδίσουν πόντους εκτελώντας συγκεκριμένες ενέργειες που σχετίζονται με τις υπηρεσίες. Οι πόντοι αυτοί μπορούν να χρησιμοποιηθούν αργότερα για την απόκτηση ανταμοιβών ή προϊόντων.
4. Απόκτηση προϊόντων: Οι χρήστες μπορούν να αποκτήσουν προϊόντα χρησιμοποιώντας τους πόντους που έχουν συγκεντρώσει. Όταν αποκτούν ένα προϊόν, οι πόντοι τους μειώνονται αντίστοιχα.
5. Έλεγχος πόντων και προϊόντων: Οι χρήστες μπορούν να ελέγξουν τον αριθμό των πόντων που έχουν συγκεντρώσει και τα ανεξόφλητα προϊόντα που έχουν αποκτήσει.

Επιπλέον, το συμβόλαιο παρέχει μηχανισμούς ασφάλειας και διαχείρισης ρόλων για τους διαχειριστές και τον ιδιοκτήτη του συμβολαίου. Οι διαχειριστές έχουν πρόσβαση σε επιπλέον λειτουργίες όπως η δημιουργία και η ενημέρωση υπηρεσιών, γεγονότων και προϊόντων, καθώς και η εκχώρηση πρόσθετων πόντων σε χρήστες.

Συνολικά, το συμβόλαιο RewardingTool παρέχει ένα σύστημα επιβράβευσης για τους χρήστες με τη χρήση κρυπτονομισμάτων (MGS Score Points) για τη συμμετοχή τους σε συγκεκριμένες δραστηριότητες και την αγορά προϊόντων.

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.19 <0.9.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

import "@openzeppelin/contracts/access/AccessControl.sol";

contract MyGreenScore is ERC20, AccessControl {

    // bytes32 public constant OWNER_ROLE = keccak256("OWNER_ROLE");

    bytes32 public constant MANAGER_ROLE = keccak256("MANAGER_ROLE"); // =>
0xasdfioha9hdf934yda9d73y

    address public owner;

    constructor() ERC20("MyGreenScore", "MGS") {

        owner = msg.sender;

        _mint(address(this), 1000000 * (10 ** uint256(decimals()))); // This is how much the contract gets

        _mint(owner, 50000 * (10 ** uint256(decimals()))); // This is the MGS Token that the Deployer gets

        _setupRole(DEFAULT_ADMIN_ROLE, owner);

        _setupRole(MANAGER_ROLE, address(this));

    }

    modifier onlyOwner() {

        require(
```

```
            hasRole(DEFAULT_ADMIN_ROLE, msg.sender),
            "MyGreenScore: caller is not the owner"
        );
    }
    function loadUpContract(address _addr) public onlyOwner {
        _mint(_addr, 2000000 * (10 ** uint256(decimals()))); // This is how much the Rewarding contract gets
    }

    function loadUpThisContract() public onlyOwner {
        _mint(address(this), 1000000 * (10 ** uint256(decimals())));
    }
    function mintAndTransferMGS(uint _amount, address _to) public onlyOwner {
        _mint(_to, _amount * (10 ** uint256(decimals()))); // This is how much the Rewarding contract gets
    }
    // Function to assign Manager access level category
    function assignManagerRole(address account) public onlyOwner {
        grantRole(MANAGER_ROLE, account);
    }
    // // Function to assign Custom access level category
    // function assignRole(bytes32 _rule, address _account) public onlyOwner {
    //     grantRole(_rule, _account);
    // }
    // TODO: Function to create new Access level categoryies
    // function setupRole(bytes32 _rule, address _account) public onlyOwner {
    //     _setupRole(_rule, _account);
    // }
}
```

Το παραπάνω κομμάτι κώδικα αντιπροσωπεύει ένα έξυπνο συμβόλαιο με το όνομα "MyGreenScore". Ας αναλύσουμε τις βασικές λειτουργίες του:

- Το συμβόλαιο κληρονομεί από το συμβόλαιο ERC20 της OpenZeppelin, το οποίο υλοποιεί τα πρότυπα ERC20 για τη δημιουργία ενός νέου κατανεμητή token.

- Επίσης, κληρονομεί από το συμβόλαιο AccessControl της OpenZeppelin, το οποίο παρέχει λειτουργίες διαχείρισης ρόλων για την πρόσβαση σε συγκεκριμένες λειτουργίες.

Οι βασικές λειτουργίες του συμβολαίου είναι οι εξής:

- Κατασκευαστής (constructor): Κατά την αρχικοποίηση του συμβολαίου, δημιουργείται ένας αρχικός αριθμός tokens με το σύμβολο "MGS" και το όνομα "MyGreenScore". Ο κατανεμητής (deployer) του συμβολαίου λαμβάνει 50,000 tokens, ενώ το συμβόλαιο λαμβάνει 1,000,000 tokens.

- Μόνο ο κάτοχος (owner) του συμβολαίου έχει πρόσβαση σε ορισμένες λειτουργίες, που προστατεύονται από τον μοντιφικατέρ (modifier) "onlyOwner".

- Υπάρχει μια λειτουργία "loadUpContract" που επιτρέπει στον κάτοχο να προσθέσει επιπλέον tokens σε μια διεύθυνση που καθορίζεται ως παράμετρος. Αυτή η διεύθυνση μπορεί να είναι η διεύθυνση ενός άλλου συμβολαίου (όπως το συμβόλαιο επιβράβευσης).

- Η λειτουργία "loadUpThisContract" επιτρέπει στον κάτοχο να προσθέσει επιπλέον tokens στο ίδιο το συμβόλαιο.

- Υπάρχει η λειτουργία "mintAndTransferMGS" που επιτρέπει στον κάτοχο να δημιουργήσει και να μεταφέρει νέα tokens σε μια διεύθυνση που καθορίζεται ως παράμετρος.

- Η λειτουργία "assignManagerRole" επιτρέπει στον κάτοχο να αναθέσει τον ρόλο του "MANAGER_ROLE" σε μια διεύθυνση. Οι διαχειριστές μπορούν να έχουν πρόσβαση σε συγκεκριμένες λειτουργίες που απαιτούν αυτόν τον ρόλο.

Είναι σημαντικό να σημειωθεί ότι ο κώδικας χρησιμοποιεί τη βιβλιοθήκη OpenZeppelin για την υλοποίηση των λειτουργιών ERC20 και AccessControl. Αυτή η βιβλιοθήκη είναι ευρέως αποδεκτή και χρησιμοποιείται συχνά για την ασφαλή ανάπτυξη έξυπνων συμβολαίων στο Ethereum.

Ας συνεχίσουμε με την ανάλυση του κώδικα και παρέχω περισσότερες πληροφορίες:

- Η γραμμή `bytes32 public constant MANAGER_ROLE = keccak256("MANAGER_ROLE");` ορίζει μια σταθερά με την τιμή του ρόλου "MANAGER_ROLE". Η σταθερά αυτή αναπαριστά τη μοναδική ταυτότητα του ρόλου στο συμβόλαιο.

- Η μεταβλητή `owner` αποθηκεύει τη διεύθυνση του κατόχου (owner) του συμβολαίου. Η τιμή αυτή ορίζεται κατά την αρχικοποίηση του συμβολαίου στον κατασκευαστή.

- Η συνάρτηση `onlyOwner` είναι ένας μοντιφικατέρ που επιβεβαιώνει ότι ο καλούντας την συνάρτηση είναι ο κάτοχος του συμβολαίου. Αν ο καλούντας δεν είναι ο κάτοχος, η εκτέλεση της συνάρτησης διακόπτεται και εμφανίζεται μήνυμα σφάλματος.

- Η συνάρτηση `loadUpContract` επιτρέπει στον κάτοχο να προσθέσει επιπλέον tokens σε μια διεύθυνση που καθορίζεται ως παράμετρος. Αυτή η διεύθυνση μπορεί να είναι η διεύθυνση ενός άλλου συμβολαίου, όπως το συμβόλαιο επιβράβευσης. Ο κάτοχος μπορεί να φορτώσει το συμβόλαιο με ένα προκαθορισμένο ποσό tokens.

- Η συνάρτηση `loadUpThisContract` επιτρέπει στον κάτοχο να προσθέσει επιπλέον tokens στο ίδιο το συμβόλαιο. Αυτό μπορεί να γίνει για να εξασφαλιστεί ότι το συμβόλαιο έχει αρκετά tokens για την κανονική του λειτουργία.

- Η συνάρτηση `mintAndTransferMGS` επιτρέπει στον κάτοχο να δημιουργήσει και να μεταφέρει νέα tokens σε μια διεύθυνση που καθορίζεται ως παράμετρος. Αυτό μπορεί να χρησιμοποιηθεί για να προσθέσει περισσότερα tokens σε άλλα συμβόλαια ή διευθύνσεις που απαιτούν πρόσθετη ποσότητα.

- Η συνάρτηση `assignManagerRole` επιτρέπει στον κάτοχο να αναθέσει τον ρόλο "MANAGER_ROLE" σε μια διεύθυνση. Αυτό επιτρέπει στη διεύθυνση αυτή να έχει πρόσβαση σε συγκεκριμένες λειτουργίες που απαιτούν αυτόν τον ρόλο.

4.3.6 Γενικές εξωτερικές διασυνδέσεις του Κώδικα

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.19;

contract Oracle {

    uint32 public randomNumber = 123456;


    function updateRandomNumber(uint32 _randomNumber) public {

        // This function should have adequate controls to prevent unauthorized updates

        randomNumber = _randomNumber;

    }

}
```

Έχουμε ένα απλό συμβόλαιο Oracle. Ας αναλύσουμε τον κώδικα:

- Η γραμμή `pragma solidity ^0.8.19;` ορίζει την έκδοση της Solidity που πρέπει να χρησιμοποιηθεί για το συμβόλαιο. Σε αυτήν την περίπτωση, απαιτείται η έκδοση 0.8.19 ή νεότερη.

- Το συμβόλαιο `Oracle` δηλώνει μια μεταβλητή `randomNumber` τύπου `uint32`, η οποία αποθηκεύει έναν τυχαίο αριθμό. Αρχικά, η τιμή της `randomNumber` ορίζεται ως 123456.

- Η συνάρτηση `updateRandomNumber` επιτρέπει την ενημέρωση της τιμής του `randomNumber`. Η νέα τιμή που παρέχεται ως παράμετρος `_randomNumber` αντιστοιχεί στην ενημερωμένη τιμή του τυχαίου αριθμού. Αυτή η συνάρτηση δεν έχει επιπλέον ελέγχους για την εξουσιοδότηση των ενημερώσεων, οπότε πρέπει να προστεθούν κατάλληλοι έλεγχοι για να αποτραπεί η μη εξουσιοδοτημένη ενημέρωση του τυχαίου αριθμού.

Αυτό είναι ένα απλό συμβόλαιο Oracle που αποθηκεύει έναν τυχαίο αριθμό και επιτρέπει την ενημέρωσή του. Ωστόσο, για να χρησιμοποιηθεί ως Oracle σε μια πραγματική εφαρμογή, θα πρέπει να προστεθούν περισσότεροι έλεγχοι και λειτουργίες ασφαλείας.

```
const hre = require("hardhat");

const { ethers } = require("hardhat");

const fs = require("fs"); // Node.js file system module for reading files

const path = require("path");

const readline = require("readline");

const { getAddresses } = require("./getAddresses");

function askForUser(query) {

  const rl = readline.createInterface({
```

```
    input: process.stdin,
    output: process.stdout,
  });
  return new Promise((resolve) =>
    rl.question(query, (ans) => {
      rl.close();
      resolve(ans);
    })
  );
}
function formatTokenAmount(amount, decimals = 18, precision = 4) {
  // Convert BigNumber to string and get the integer part
  const str = ethers.formatUnits(amount, decimals);
  // Split the string into integer and fractional parts
  const parts = str.split(".");
  // Combine integer part and the specified number of decimal places
  const formatted = parts[0] + "." + (parts[1] || "").slice(0, precision);
  return parseFloat(formatted).toFixed(precision);
}
async function main() {
  const [deployer] = await ethers.getSigners();
  console.log();
  console.log("====== Getting Balances ======");
  // Using Node's filesystem to get the 2 ABI's
  // Assuming the ABI's are stored in the 'build/contracts' folder with names
'YourERC20ContractName.json' and 'YourOracleContractName.json'
  // Getting the Paths
  const ERC20Path = path.resolve(
    __dirname,
    "../artifacts/contracts/MyGreenScore.sol/MyGreenScore.json"
  );
  const oraclePath = path.resolve(
```

```
  __dirname,
  "../artifacts/contracts/Oracle.sol/Oracle.json"
);
const servicesPath = path.resolve(
  __dirname,
  "../artifacts/contracts/Services.sol/Services.json"
);
const RewardingToolPath = path.resolve(
  __dirname,
  "../artifacts/contracts/RewardingTool.sol/RewardingTool.json"
);
// Getting the ABI's using the Paths
const ERC20ABI = JSON.parse(fs.readFileSync(ERC20Path)).abi;
const OracleABI = JSON.parse(fs.readFileSync(oraclePath)).abi;
const ServicesABI = JSON.parse(fs.readFileSync(servicesPath)).abi;
const RewardingToolABI = JSON.parse(fs.readFileSync(RewardingToolPath)).abi;
// Getting the Contracts' Addresses
const contractAddresses = getAddresses();
// 1. Create contract instances: ERC20
const ERC20ContractInstance = new ethers.Contract(
  contractAddresses.ERC20ContractAddress,
  ERC20ABI,
  deployer
);
// 2. Create contract instances: Oracle
const OracleContractInstance = new ethers.Contract(
  contractAddresses.OracleContractAddress,
  OracleABI,
  deployer
);
// 3. Create contract instances: Services
const ServicesContractInstance = new ethers.Contract(
```

```
  contractAddresses.ServicesContractAddress,
  ServicesABI,
  deployer
);
const RewardingToolContractInstance = new ethers.Contract(
  contractAddresses.RewardingToolAddress,
  RewardingToolABI,
  deployer
);

const address = await askForUser("Provide Address: ");
// const amountToGive = await askForAmount("Amount in MGS: ");
try {
  console.log("--------------------------------------------------------");
  console.log();
  console.log(`====== Commencing MGS Token Balance Retrieval ======`);
  console.log();
  console.log("--------------------------------------------------------");
  const balance = await ERC20ContractInstance.balanceOf(address);
  console.log("--------------------------------------------------------");
  console.log("Conserning This Address: ", address);
  console.log("--------------------------------------------------------");
  console.log();
  console.log("| MGS Balance: ", formatTokenAmount(balance), "  MGS");
  console.log("--------------------------------------------------------");
  console.log();
} catch (error) {
  console.log();
  console.log("--------------------------------------------------------");
  console.log("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^");
  console.log("This error occurred:");
  console.log(error);
```

```
    console.log();
    console.log("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^");
    console.log("--------------------------------------------------------");
  }
}
main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error);
    process.exit(1);
  });
module.exports = {
  main,
};
```

Παρακάτω είναι μια ανάλυση του κώδικα:

a. Η αρχή του κώδικα περιλαμβάνει τις απαιτούμενες βιβλιοθήκες και τα αρχεία που χρειάζονται.
b. Οι συναρτήσεις `askForUser` και `formatTokenAmount` είναι βοηθητικές συναρτήσεις για τη διευκόλυνση του χειρισμού ερωτημάτων προς το χρήστη και τη μορφοποίηση των ποσών των token.
c. Η συνάρτηση `main` είναι η κύρια λειτουργία του σεναρίου. Αρχικά, ανακτάται ο deployer (ο χρήστης που εκτελεί το σενάριο). Στη συνέχεια, γίνεται η ανάγνωση των αρχείων ABI για τα συμβόλαια MyGreenScore.sol, Oracle.sol, Services.sol και RewardingTool.sol. Τα αρχεία αυτά πρέπει να βρίσκονται στον ίδιο φάκελο με το σενάριο και να έχουν τα ονόματα που προσδιορίζονται στις μεταβλητές `ERC20Path`, `oraclePath`, `servicesPath` και `RewardingToolPath`.
d. Στη συνέχεια, δημιουργούνται αντικείμενα συμβολαίων για τα συμβόλαια ERC20, Oracle, Services και RewardingTool, χρησιμοποιώντας τα ABI και τις διευθύνσεις των συμβολαίων που ανακτήθηκαν από την `getAddresses`.
e. Η συνάρτηση `askForUser` χρησιμοποιείται για να ζητήσει από τον χρήστη να παράσχει μια διεύθυνση.
f. Στη συνέχεια, η συνάρτηση `balanceOf` του συμβολαίου ERC20 καλείται για να ανακτηθεί η ισορροπία του MyGreenScore token για την δοθείσα διεύθυνση.
g. Τέλος, η ισορροπία του MyGreenScore token εμφανίζεται στον καταληκτικό πίνακα.

Αυτός είναι ο βασικός κώδικας για την ανάγνωση της ισορροπίας του MyGreenScore token για μια δεδομένη διεύθυνση χρήστη.

```
import { useEffect, useState } from "wordpress";
import { ethers } from "ethers";
import axios from "axios";
import { axiosOracle } from "api/config";
import { useMetaMask } from "contexts/web3/MetaMaskContextProvider";
import { useGlobalContext } from "contexts/GlobalContextProvider";
import { toast } from "react-toastify";
// import { axiosOracle } from "api/config";
import {
  loginProcessHandler,
  noAccountWarning,
} from "utils/LoginProcessHandler";

export async function getNonce() {
  try {
    const response = await axiosOracle.get("/big-random-number");
    const nonce = response.data.randomBigNumber;
    return nonce;
  } catch (error) {
    console.error("⛔ (Express Oracle) Failed to fetch nonce:", error);
    toast.error(
      "We are experiencing issues with the Web Server, please try again later",
      {
        position: "top-center",
        autoClose: 13000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
        theme: "dark",
      }
```

```
    );
    return null;
  }
}
export function useWeb3Login() {
  const { hasProvider, provider, wallet } = useMetaMask();
  const { userData, setUserData, callContractFn } = useGlobalContext();
  const [isAuthenticated, setIsAuthenticated] = useState(false);
  const [signer, setSigner] = useState(null);

  useEffect(() => {
    async function fetchUserData() {
      try {
        const _userObj = await callContractFn("users", userData.wallet);
        console.log(_userObj);
        console.log("(Web3Login): The user obj from contract: ", _userObj);
        if (_userObj[2] === "") {
          noAccountWarning();
          setUserData((prev) => {
            return {
              ...prev,
              name: "No Account",
            };
          });
          return;
        }
        const _isManager = await callContractFn(
          "checkManagerRole",
          userData.wallet
        );
        const _isOwner = await callContractFn(
          "checkOwnerRole",
```

```
        userData.wallet
      );
      // const _pendingRewards = await callContractFn(
      //   "getUserProducts",
      //   userData.wallet
      // );

      setUserData((prev) => {
        return {
          ...prev,
          name: _userObj[2],
          // tokens: (parseInt(userTokens) / 1000).toFixed(2),
          // pendingRewards: _pendingRewards,
          accessLevel: _isManager ? "manager" : _isOwner ? "owner" : "",
        };
      });
    } catch (error) {
      console.error("Error fetching user data:", error);
    }
  }
  console.log(isAuthenticated);
  console.log(userData.wallet);
  if (isAuthenticated && userData.wallet) {
    console.log("I am Running you guys!!!");
    fetchUserData();
  }
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, [isAuthenticated, userData.wallet]);
const signMessage = async () => {
  // if (wallet.chainId !== 20231) {
  const isWeb3Ready = loginProcessHandler("login", hasProvider, wallet);
  if (!isWeb3Ready) return;
```

```
    const nonce = await getNonce();
    const message = `This is your generated random verification number: ${nonce}.\n \nNo action is required from your side concerning the random number. We simply show it for transparency reasons.\n \nBy allowing your Wallet to sign this message, using your private key, it can be proven that you are the true owner of this wallet, without the need for a password.\n \nPlease click "Sign" to proceed.`;
    console.log(provider);
    if (nonce && hasProvider) {
      const wallet = new ethers.providers.Web3Provider(provider);
      const _signer = wallet.getSigner();
      const _signerAddr = await _signer.getAddress();
      setSigner(_signer);
      const signedMessage = _signer.signMessage(message);
      try {
        console.log("The Message: ", message);
        const responsePromise = signedMessage
          .then((signedMessage) =>
            axiosOracle.post("/verify-signature", {
              nonce: message,
              //   nonce: "AAAAAA", // FOr testing: to get an Error
              userAddress: _signerAddr,
              signedMessage: signedMessage,
            })
          )
          .then((response) => {
            if (response.data.verified) {
              return response;
            } else {
              throw new Error("Signature verification failed");
            }
          });
        toast.promise(
          responsePromise,
          {
```

```
    pending: "Sign this message and await for verification...",
    success: {
      render: ({ data }) => {
        console.log(data.data);
        setIsAuthenticated(data.data.verified);
        if (data.data.verified) {
          setUserData((prev) => {
            return {
              ...prev,
              isLoggedIn: true,
              wallet: _signerAddr,
            };
          });
          return "Your signature is valid! Welcome ";
        } else {
          return "The signature is invalid!";
        }
      },
    },
    error: "Failed to verify signed message",
  },
  {
    position: "top-center",
    autoClose: 4000,
    hideProgressBar: false,
    closeOnClick: true,
    pauseOnHover: true,
    draggable: true,
    progress: undefined,
    theme: "colored",
  }
);
```

```
    } catch (error) {
      console.error("Failed to verify signed message:", error);
    }
  }
 };
 return { isAuthenticated, signMessage };
}
```

Παραπάνω είναι μια ανάλυση του κώδικα:

a. Οι απαιτούμενες βιβλιοθήκες `wordpress`, `ethers` και `axios` εισάγονται στο πρόγραμμα.
b. Ακολουθεί η εισαγωγή από τα context `MetaMaskContextProvider` και `GlobalContextProvider`. Αυτά τα context φαίνεται ότι παρέχουν πληροφορίες για το MetaMask και την καθολική κατάσταση της εφαρμογής αντίστοιχα.
c. Η συνάρτηση `getNonce` φαίνεται να ανακτά έναν αριθμό nonce από τον Oracle server.
d. Η συνάρτηση `useWeb3Login` φαίνεται να είναι ένας προσαρμοσμένος γίγνεσθαι συγκεκριμένα για την εφαρμογή πιστοποίησης και εισόδου μέσω του MetaMask. Η συνάρτηση χρησιμοποιεί το πλαίσιο MetaMaskContextProvider για να ανακτήσει πληροφορίες σχετικά με το MetaMask πάροχο, όπως η παροχή, το πορτοφόλι κ.λπ., και το GlobalContextProvider για να αποθηκεύσει και να ανακτήσει καθολικές πληροφορίες και καταστάσεις εφαρμογής.
e. Οι μεταβλητές `isAuthenticated` και `signer` χρησιμοποιούνται για να κρατήσουν την κατάσταση της ταυτοποίησης και τον υπογράφοντα.
f. Η συνάρτηση `fetchUserData` φαίνεται να ανακτά δεδομένα χρήστη από το συμβόλαιο και τα αποθηκεύει στο GlobalContextProvider.
g. Η συνάρτηση `signMessage` φαίνεται να χρησιμοποιείται για να υπογράψει ένα μήνυμα με τον ιδιωτικό κλειδί του χρήστη που παρέχεται από το MetaMask. Το μήνυμα αποστέλλεται στον Oracle server για επαλήθευση της υπογραφής. Η ταυτοποίηση επιτυγχάνεται αν η υπογραφή είναι έγκυρη και ο χρήστης θεωρείται ταυτοποιημένος.

Η κατανόηση της solidity που συνεπάγεται και η έμπνευση του κώδικα προέρχεται από εδώ

[40]

5.Συμπεράσματα και μελλοντικές επεκτάσεις

Η εφαρμογή επικεντρώθηκε σε διάφορα θέματα σχετικά με την τεχνολογία blockchain, τις έξυπνες συμβάσεις, τις εφαρμογές τους και τις διαδικασίες που μπορούν να ενσωματωθούν σε ένα οικοσύστημα blockchain.

Καταρχάς, εξετάσαμε τις έξυπνες συμβάσεις (smart contracts), που είναι προγράμματα που εκτελούνται αυτόματα όταν πληρούνται συγκεκριμένες προϋποθέσεις. Είναι αυτόνομες, αξιόπιστες και ασφαλείς, και μπορούν να χρησιμοποιηθούν για διάφορες εφαρμογές, όπως τον χειρισμό χρημάτων και των αποκεντρωμένων οργανισμών (DAOs).

Στη συνέχεια, εστιάσαμε στην εφαρμογή MyGreenScore, η οποία είναι μια εφαρμογή blockchain που χρησιμοποιεί έναν τοπικό token MGS (MyGreenScore) για την ανταμοιβή των χρηστών για περιβαλλοντικά φιλικές ενέργειες. Η εφαρμογή έχει ένα κέντρο ελέγχου πρόσβασης (Access Control Center), που διαχειρίζεται τους ρόλους και τις προνομιακές προσβάσεις των χρηστών, καθώς και ένα Oracle contract που χρησιμοποιείται για να ανακτήσει τυχαίους αριθμούς που χρησιμοποιούνται στην εφαρμογή.

Ακόμη, εστιάσαμε σε κομμάτια κώδικα JavaScript για τη σύνδεση με το blockchain, τον έλεγχο των ισορροπιών των χρηστών, την υπογραφή μηνυμάτων, και την επικοινωνία με έναν Oracle πάροχο για την ανάκτηση τυχαίων αριθμών.

Συμπεράσματα για μια εφαρμογή blockchain:

- Οι έξυπνες συμβάσεις (smart contracts) Οι έξυπνες συμβάσεις είναι κεντρικό στοιχείο της τεχνολογίας blockchain. Είναι προγράμματα που εκτελούνται αυτόνομα με βάση τις προκαθορισμένες συνθήκες, χωρίς την ανάγκη για ενδιάμεσους. Μπορούν να χρησιμοποιηθούν για να δημιουργήσουν αξιόπιστα και διαφανή συμβόλαια μεταξύ δύο ή περισσότερων μερών.- Η δημιουργία κέντρου ελέγχου πρόσβασης (Access Control Center) βοηθά στην διαχείριση των ρόλων και των προνομιακών προσβάσεων των χρηστών.
- Εφαρμογές blockchain: Η τεχνολογία blockchain εφαρμόζεται σε πολλούς τομείς, όπως οικονομία, υγεία, προμήθειες, αλυσίδες εφοδίων και πολλά άλλα. Η δυνατότητα ασφαλούς, διαφανούς και αυτόνομης λειτουργίας του blockchain κάνει τις εφαρμογές του να είναι αξιόπιστες και αποδοτικές.
- Κέντρο Ελέγχου Πρόσβασης (Access Control Center): Η δημιουργία ενός κέντρου ελέγχου πρόσβασης επιτρέπει τη διαχείριση των ρόλων και των δικαιωμάτων πρόσβασης των χρηστών. Αυτό είναι σημαντικό για να διασφαλιστεί ότι οι χρήστες έχουν μόνο τις απαραίτητες εξουσίες για τις ενέργειες που πραγματοποιούν στο blockchain.
- Oracle: Οι Oracle πάροχοι επιτρέπουν τις εφαρμογές να αλληλεπιδρούν με εξωτερικές πηγές δεδομένων, που μπορεί να είναι χρήσιμο για τη λήψη εξωτερικών πληροφοριών ή για την επαλήθευση δεδομένων που προέρχονται από έξω από το blockchain.
- Προστασία και ασφάλεια: Η τεχνολογία blockchain βασίζεται σε ισχυρά μηχανισμούς κρυπτογράφησης και επικύρωσης, παρέχοντας υψηλό επίπεδο ασφάλειας και προστασίας των δεδομένων.

Γενικότερα, συμπεραίνουμε πως η τεχνολογία blockchain αποτελεί μια καινοτόμο και υποσχόμενη λύση που μπορεί να ανατρέψει τον τρόπο λειτουργίας πολλών κοινωνικών και οικονομικών τομέων. Μέσω της εφαρμογής της σε διάφορους τομείς της κοινωνίας, μπορούμε να δημιουργήσουμε αξιόπιστα, διαφανή και αυτόνομα συστήματα, εξασφαλίζοντας ταυτόχρονα υψηλό επίπεδο ασφάλειας και προστασίας των δεδομένων. Ενώ η τεχνολογία blockchain συνεχίζει να εξελίσσεται, παρέχει νέες ευκαιρίες και λύσεις που έχουν τη δυνατότητα να αλλάξουν τον τρόπο λειτουργίας της κοινωνίας για το καλύτερο.

6.ΒΙΒΛΙΟΓΡΑΦΙΑ

1) el.wikipedia.org
2) blockchain.org
3) kriptomat.io
4) naftemporiki.gr
5) https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3052165
6) https://www.semanticscholar.org/paper/Governance-in-the-Blockchain-Economy%3A-A-Framework-Beck-M%C3%BCller-Bloch/78d24471d60cff29cd0421e5473d4cb31fc71555
7) Towards a Framework for Evaluation of Blockchain Implementations Implementations
8) https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3018371 .
9) M. Iansiti and K. R. Lakhani, ‘‘The truth about blockchain,’’ Harvard Bus. Rev., vol. 95, no. 1,pp. 118–127, 2017.
10) I. Milic. (2019). Blockchain Statistics and Facts That Will Make You Think: The Dawn of Hyper Capitalism. [Online]. Διαθέσιμο: https://fortunly.com/statistics/blockchain-statistics/#gref
11) (2019) Leonardo Maria, Gianluca Salviotti, Nico Abbatemarco, “Towards a Comprehensive Blockchain Architecture Continuum”, Proceedings of the 52nd Hawaii International Conference on System Sciences
12) Takashima, “The Ultimate Guide To The World Of Blockchain Technology, Bitcoin, Ethereum”, 2017.
13) Andreas M. Antonopoulos and Dr. Gavin Wood, “Mastering Ethereum”, O’Reilly Media, Inc,United States of America, 2019.
14) S. Nakamoto. (2017). Bitcoin: A Peer-to-Peer Electronic Cash System, Oct. 2008. [Online].Available: http://www.bitcoin.org/bitcoin.pdf
15) C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, ‘‘Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications, and opportunities,’’ IEEE Access, vol. 7, pp. 85727–85745, 2019.IEEE Access, vol. 7, pp. 85727– 85745, 2019
16) L. M. Bach, B. Mihaljevic, and M. Zagar, ‘‘Comparative analysis of blockchain consensus algorithms,’’ in Proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO),
17) May 2018, pp. 1545–1550
18) (2020). VeriCoin. [Online]. Available: https://vericonomy.ams3.cdn.digitaloceanspaces.com/documents/VeriCoin-Proof-of-Stake-TimeWhitepaper.pdf
19) (2020). VeroCoin. [Online]. Available: https://vericoin.info/vericoindigital-currency/
20) F. Tschorsch and B. Scheuermann, ‘‘Bitcoin and beyond: A technical survey on decentralized digital currencies,’’ IEEE Commun. Surveys Tuts., vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.

21) K. Karantias, A. Kiayias, and D. Zindros, ‘‘Proof-of-burn,’’ in Proc. Int. Conf. Financial
22) Cryptogr. Data Security., 2019, pp. 523–540.

23) L. Ismail and H. Materwala, ‘‘A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions,’’ Symmetry, vol. 11, no. 10, p. 1198, Sep. 2019.

24) A. Yakovenko, ‘‘Solana: A new architecture for a high-performance blockchain,’’ Solana
25) Labs, Tech. Rep., 2018.

26) A. Shahaab, B. Lidgey, C. Hewage, and I. Khan, ‘‘Applicability and appropriateness of distributed ledgers consensus protocols in public and private sectors: A systematic review,’’ IEEE Access, vol. 7, pp. 43622–43636, 2019

27) (2020). VeriBlock. [Online]. Available: https://mirror1.veriblock.org/ Proof-of Proof_and_VeriBlock_Blockchain_Protocol_Consensus_ Algorithm_and_Economic_Incentivization_v1 pdf

28) (2020). Veriblock. [Online]. Available: https://www.veriblock.com/

29) O. Samuel, N. Javaid, M. Awais, Z. Ahmed, M. Imran, and M. Guizani, ‘‘A blockchain model for fair data sharing in deregulated smart grids,’’ in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2019, pp. 1–7.

30) S. D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, ‘‘PBFT vsproof-of-authority: Applying the CAP theorem to permissioned blockchain,’’ in Proc. Italian Conf. Cyber Secur., 2018, pp. 1–11.
31) Andreas M. Antonopoulos and Dr. Gavin Wood, “Mastering Ethereum”, O’Reilly Media, Inc,United States of America, 2019
32) Massimo Bertaccini, “Cryptography Algorithms”, Packt> BIRMINGHAM—MUMBAI (2022)
33) Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper 151(2014), 1–32 (2014)
34) https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between
35) https://www.news18.com/news/business/cryptocurrency/cryptocurrency-price-today-bitcoin-zooms-over-22000-polkadot-solana-gain-up-to-8-in-a-day-5948155.html
36) https://mashable.com/deals/march-1-blockchain-development-bundle?test_uuid=05n6qXSf1roZ2M9nfgWNQzG&test_variant=a
37) https://github.com/jimzord12/RewardingTool-Frontend-Deploy/tree/master
38) https://github.com/jimzord12/Genera-SmartContracts/tree/version_2
39) https://github.com/jimzord12/simple-oracle-backend-express
40) https://www.youtube.com/watch?v=gyMwXuJrbJQ&t=23477s&ab_channel=freeCodeCamp.org