ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

ΥΠΗΡΕΣΙΑ ΕΠΙΒΕΒΑΙΩΣΗΣ ΣΥΝΑΛΛΑΓΩΝ ΣΕ ΔΙΚΤΥΑ BLOCKCHAIN ΠΟΥ ΒΑΣΙΖΟΝΤΑΙ ΣΕ EVM

Φοιτητής: Εμμανουήλ Σταθόπουλος
Αριθμός Μητρώου: 50343052

Επιβλέπων Καθηγητής

Δημήτριος Κόγιας
Ακαδημαϊκός Υπότροφος / Εντεταλμένος Διδάσκων

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, Φεβρουάριος 2024

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

A BLOCKCHAIN-BASED SERVICE FOR TRANSACTION CONFIRMATION IN EVM ENVIRONMENTS

Student: Emmanouil Stathopoulos
Registration Number: 50343052

Supervisor

Dimitrios Kogias
Adjunct Lecturer / Adjunct Academic Staff

ATHENS-EGALEO, February 2024

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Δημήτριος Κόγιας, Ακαδημαϊκός Υπότροφος / Εντεταλμένος Διδάσκων	Πατρικάκης Χαράλαμπος, Καθηγητής	Παπαδόπουλος Περικλής, Καθηγητής
Dimitrios Kogias Digitally signed by Dimitrios Kogias Date: 2024.04.10 17:10:27 +03'00' (Υπογραφή)	Digitally signed by Charalampos Patrikakis Date: 2024.04.15 11:52:57 +03'00' (Υπογραφή)	Periklis Papadopoulos Digitally signed by Periklis Papadopoulos Date: 2024.04.13 14:50:15 +03'00' (Υπογραφή)

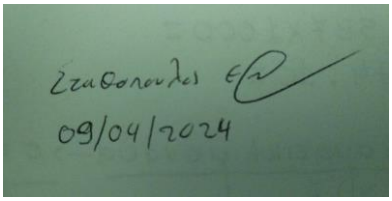
ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος … Εμμανουήλ Σταθόπουλος ... του … Φωτίου ..., με αριθμό μητρώου … 50343052 … φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της 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 διπλώματός μου.»

Ο Δηλών
Εμμανουήλ Σταθόπουλος



(Υπογραφή)

Περίληψη

Στην εργασία αυτή παρουσιάζεται διαδικτυακή υπηρεσία, η χρήση της οποίας επιτρέπει την πραγματοποίηση και αποπληρωμή ανώνυμων ηλεκτρονικών παραγγελιών μέσω του EVM blockchain δικτύου Polygon. Ως μέσο ανταλλαγής για την αποπληρωμή των παραγγελιών μπορούν να χρησιμοποιηθούν, είτε το εγγενές ψηφιακό νόμισμα του δικτύου (MATIC), είτε οι υποστηριζόμενες ψηφιακές μάρκες (USDC, USDT, agEUR). Η σύνδεση μεταξύ του blockchain και των εξυπηρετητών της υπηρεσίας επιτυγχάνεται μέσω της παρακολούθησης των έξυπνων συμβολαίων που έχουν αναπτυχθεί και καταχωρηθεί στο Polygon, για τον συγκεκριμένο σκοπό. Ως κίνητρο δημιουργίας της υπηρεσίας είναι η αναφορά και χρήση του διαδικτύου Web3, το οποίο αποτελείται από κατανεμημένα δίκτυα διαμοιρασμού αρχείων και δίκτυα blockchain. Το Web3, λόγο του τρόπου υλοποίησης του (ανοιχτός κώδικας), επιτρέπει στους χρήστες να ελέγχουν τις διαδικασίες που ακολουθούνται στο δίκτυο, ενώ παράλληλα διαφυλάσσει την ταυτότητα των χρηστών μέσω κρυπτογραφικών αλγορίθμων, παρέχοντας έτσι την ασφάλεια που υπολείπεται από το σημερινό διαδίκτυο (Web2).

Λέξεις – κλειδιά

διαδίκτυο 3, δίκτυα αλυσίδας μπλοκ, αποκεντρωμένα δίκτυα, κρυπτονομίσματα, ανοικτός κώδικας, προστασία προσωπικών δεδομένων

Abstract

This paper presents an internet service which can be used for creating and checking out orders anonymously, through the Polygon EVM blockchain network. As medium of exchange for the orders, clients can either use the network's native currency (MATIC) or the service supported tokens (USDC, USDT, agEUR). The connection between the blockchain and the service servers is achieved by monitoring smart contracts, that have been developed for this exactly purpose. This service was created having in mind that by using a blockchain network there could be a reference to Web3, which consists of distributed file-sharing networks and blockchain networks. Web3, structured as open source, allows users to audit the procedures followed on the network while safeguarding their true identity through cryptographic algorithms, providing the security that today's internet (Web2) lack.

Keywords

Web3, blockchain, decentralized networks, cryptocurrencies, open source, personal data protection

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

1 ΕΙΣΑΓΩΓΗ

Στην εργασία αυτή παρουσιάζεται διαδικτυακή υπηρεσία που επιτρέπει στους χρήστες να πραγματοποιούν και να αποπληρώνουν ανώνυμες ηλεκτρονικές παραγγελίες, μέσω blockchain.

Τα δίκτυα blockchain αποτελούν δίκτυα ομότιμων κόμβων, κατανεμημένου μηχανισμού συναίνεσης τα οποία, συντηρούνται από τους ίδιους τους, τούς χρήστες. Οι συναλλαγές που πραγματοποιούνται μέσω blockchain, αν και διαθέσιμες προς ανάγνωση από όλους (χρήστες ή μη χρήστες του δικτύου), δεν περιέχουν τα προσωπικά στοιχεία τον συναλλασμένων παρά μόνο χαρακτηριστικά που δηλώνουν ότι η συναλλαγή αφορά κάποιους συγκεκριμένους χρήστες.

1.1 Αντικείμενο της διπλωματικής εργασίας

Παρόλο που έχουν επενδυθεί και εξακολουθούν να επενδύονται μεγάλα ποσά χρήματος για την αύξηση της γεωγραφικής κάλυψης και γενικότερη βελτίωση των συνδέσεών του, όσο αφορά τον τρόπο λειτουργίας και δόμησής του, το διαδίκτυο (internet) παραμένει απαράλλακτο από την εποχή που έγινε εμπορικά διαθέσιμο.

Μελετώντας την τωρινή του μορφή (Web2), το διαδίκτυο υλοποιείται από ένα δίκτυο ιδιωτικών εξυπηρετητών οι οποίοι συνδέονται μεταξύ τους με άμεσο ή έμμεσο τρόπο. Αν και τα δεδομένα που ανταλλάσσονται σε αυτό (στο διαδίκτυο) είναι μη αξιοποιήσιμα/αναγνώσιμα από τους ενδιάμεσους εξυπηρετητές (λόγω συμμετρικής κρυπτογράφησης, π.χ. SSL και TSL) είναι αδύνατον να γνωρίζουμε που αυτά καταλήγουν, αφού η επεξεργασία και η αποθήκευσή τους γίνεται σε εξυπηρετητές, όπου ως χρήστες, δεν έχουμε καμία πρόσβαση. Οι υπηρεσίες αν και είναι υποχρεωμένες να δηλώνουν τον τρόπο διαχείρισης των δεδομένων που λαμβάνουν και να συμμορφώνονται με αυτόν, δεν αποκλείεται να χρησιμοποιούν τα δεδομένα που έχουν συλλέξει και για άλλους λόγους, χωρίς την συναίνεση των χρηστών. Έτσι είμαστε σε θέση να υποστηρίξουμε ότι το Web2, αν και αποτελεί ένα εργαλείο αμέτρητων δυνατοτήτων, είναι ένα εργαλείο το οποίο δεν μπορεί να ελεγχθεί. Από την άλλη όμως, το να υποστηρίζουμε ότι το πρόβλημα του διαδικτύου είναι ότι δεν μπορεί να ελεγχθεί, δεν δίνει την πλήρη εικόνα του προβλήματος, αφού αν και είναι θεωρητικά κατανεμημένο ανά τον πλανήτη, το διαδίκτυο στην πλειοψηφία του συγκεντρώνεται σε εταιρίες που εμπορεύονται την υπολογιστική ισχύ που χρησιμοποιείται από τις υπηρεσίες που στεγάζει, δηλαδή σε κέντρα δεδομένων (data centers). Άρα το διαδίκτυο, όχι μόνο δεν είναι ελέγξιμο, αλλά και διαθεσιμότητα του εξαρτάται κυρίως από τα κέντρα δεδομένων που το εξυπηρετούν.

Θέλοντας γενικότερα να αντιμετωπιστούν τα ζητήματα του διαδικτύου, σήμερα αναπτύσσετε ένα ασφαλέστερο μέσο δικτύωσης, που χαρακτηρίζεται ως Web3. Συνοπτικά, το Web3 αποτελείται από δίκτυα ομότιμων κόμβων (peer to peer networks) στα οποία οι χρήστες, μπορούν να λειτουργήσουν και ως εξυπηρετητές αλλά και να ελέγξουν (audit) τις υπηρεσίες που αυτό στεγάζει. Δηλαδή Web3 χαρακτηρίζεται το κατανεμημένο διαδίκτυο ανοιχτού κώδικα (open source) που υλοποιείται από δίκτυα διαμοιρασμού αρχείων (π.χ. το IPFS [1]) και δίκτυα blockchain και συντηρείται από τους ίδιους τού τους χρήστες. Εκτιμώντας λοιπόν ότι στα επόμενα χρόνια το Web3 θα αποτελεί το κύριο εργαλείο ανάπτυξης διαδικτυακών υπηρεσιών, σήμερα έχουμε την ευκαιρία να αναπτύξουμε και να δοκιμάσουμε εφαρμογές, προετοιμάζοντάς μας για τις εργασιακές απαιτήσεις του μέλλοντος.

1.2 Σκοπός και στόχοι

Στα πλαίσια της διπλωματικής αυτής εργασίας, δημιουργήθηκε υβριδική υπηρεσία εξυπηρέτησης ηλεκτρονικών πληρωμών (χρησιμοποιεί συνδυαστικά τα διαδίκτυα Web2 και Web3), με σκοπό τόσο την καθ' αυτό χρήση της, όσο και την γενικότερη ενημέρωση του αναγνώστη σχετικά με τις δυνατότητες που προσφέρει η τεχνολογία blockchain και το Web3.

Στόχος είναι ο αναγνώστης μέσω των περιεχομένων της εργασίας και χρησιμοποιώντας την υπηρεσία που παρουσιάζεται στην εργασία:

- Να αναγνωρίσει τα πλεονεκτήματα που επιφέρει η χρήση blockchain υπηρεσιών
- Να εξοικειωθεί με τα EVM δίκτυα blockchain και τα έξυπνα συμβόλαια
- Να διαπιστώσει το πόσο "εφικτή" είναι η ανάπτυξη μίας blockchain υπηρεσίας
- Να παρακινηθεί ως προς την περαιτέρω διερεύνηση γύρω από την τεχνολογία blockchain

1.3 Μεθοδολογία

Το διαδίκτυο όπως αναφέραμε υπό την τωρινή του μορφή, αποτελεί ένα εργαλείο αμέτρητων δυνατοτήτων αλλά ταυτόχρονα και ένα εργαλείο, το οποίο εξυπηρετεί την κακόβουλη χρήση. Η διαπίστωση αυτή τεκμηριώνεται από το γεγονός ότι, ενώ ως χρήστες μπορούμε να είμαστε θετικοί ότι τα δεδομένα που ανταλλάσσουμε με μία υπηρεσία καταναλώνονται αποκλειστικά από αυτή λόγο της κρυπτογράφησης των συνδέσεων, δεν υπάρχει καμία εξασφάλιση σχετικά με το πως η υπηρεσία διαχειρίζεται τα δεδομένα που λαμβάνει, καθώς και ότι δεν θα αλλάξει τον τρόπο διαχείρισης τους στο μέλλον.

Έχοντας εντοπίσει το παραπάνω ζήτημα, μαθαίνοντας ότι μία εταιρία "κολοσσός" στην εξυπηρέτηση ηλεκτρονικών πληρωμών αποφάσισε μονομερώς να αποκλείσει κάποιους χρήστες από την υπηρεσία τους (παρακρατώντας το σύνολο των χρημάτων τους), δημιουργήθηκε η ιδέα ανάπτυξης μίας υπηρεσίας, στην οποία οι συναλλαγές θα εξυπηρετούνται μέσω εγγυημένων διαδικασιών, διασφαλίζοντας παράλληλα την ανωνυμία των χρηστών της. Ανακαλύπτοντας ότι οι παραπάνω απαιτήσεις ικανοποιούνται πλήρως μέσω Web3 και συγκεκριμένα από τα EVM δίκτυα blockchain, διαπιστώθηκε ότι η ιδέα αυτή είναι τεχνολογικά υλοποιήσιμη. Έτσι, από την αρχική ιδέα προέκυψε μία υπηρεσία η οποία εξυπηρετεί τον ίδιο σκοπό με την προαναφερθείσα, έχοντας όμως ως κίνητρο την προστασία των χρηστών και την διαφάνεια των συναλλαγών.

Αρχικά λοιπόν πραγματοποιήθηκε διαδικτυακή έρευνα σχετικά με τα EVM blockchain δίκτυα, τα έξυπνα συμβόλαια και τη γλώσσα προγραμματισμού Solidity, η οποία είχε ως αποτέλεσμα την καταχώρηση των (αρχικών) έξυπνων συμβολαίων στο δοκιμαστικό δίκτυο Mumbai. Διαπιστώνοντας ότι η επικοινωνία με τα έξυπνα συμβόλαια θα πρέπει να πραγματοποιείται μέσω εξυπηρετητή, προέκυψε ότι η ανάπτυξή του θα πρέπει να γίνει στο περιβάλλον Node.js, χρησιμοποιώντας βιβλιοθήκη Web3.js και στη γλώσσα προγραμματισμού JavaScript. Ξεκινώντας την ανάπτυξη του εξυπηρετητή, διαπιστώθηκε ότι θα πρέπει να γίνουν τροποποιήσεις στα έξυπνα συμβόλαια και ότι χρειάζεται να αναπτυχθούν αρχεία που τοποθετούνται στον εξυπηρετητή του πελάτη. Τελειώνοντας την ανάπτυξη των αρχείων και τροποποίηση των έξυπνων συμβολαίων, πραγματοποιήθηκε έλεγχος της υπηρεσίας στο Mumbai και μετά τις απαραίτητες διορθώσεις, η υπηρεσία καταχωρήθηκε (και) στο δίκτυο του Polygon.

1.4 Δομή

Η εργασία αυτή δομείται σε έξι (6) διακριτές ενότητες, οι οποίες είναι αναπτυγμένες με τέτοιο τρόπο, ώστε ο αναγνώστης να μπορεί να ακολουθήσει τους συλλογισμούς του συγγραφέα, διατηρώντας παράλληλα μία λογική συνέχεια.

Στην πρώτη (1η) ενότητα, ο αναγνώστης εισάγεται στο θέμα της εργασίας λαμβάνοντας γνώση τους λόγους για τους οποίους αυτό θεωρείται ενδιαφέρον και επίκαιρο από τον συγγραφέα. Αμέσως μετά παρουσιάζονται οι επιδιώξεις της εργασίας αυτής και στη συνέχεια, αφού πρώτα ενημερωθεί ο αναγνώστης τον λόγο για τον οποίο αποφασίστηκε η δημιουργία της υπηρεσίας επιβεβαίωσης, παρουσιάζεται η μεθοδολογία ανάπτυξής της.

Στην δεύτερη (2η) ενότητα, μιας και η υπηρεσία που παρουσιάζεται αφορά εγχρήματες συναλλαγές μέσω blockchain, γίνεται αρχικά παρουσίαση του υλικό χρήματος, πως αυτό χρησιμοποιείται ως μέσο συναλλαγής και πως προκύπτει η αξία του. Μετά, γίνεται αναφορά στο άυλο χρήμα, παρουσιάζοντας το πως αυτό έχει διευκολύνει τις καθημερινές μας συναλλαγές αλλά και τα προβλήματά του. Τέλος, έχοντας παρουσιάζει το υλικό και άυλο χρήμα, γίνεται αναφορά στο ψηφιακό χρήμα, εκθέτοντας το πως αυτό διαφοροποιείται από το άυλο, πώς υποστηρίζεται από τα δίκτυα blockchain, σε ποιους απευθύνεται και ποιους σκοπούς εξυπηρετεί. Στο τέλος της ενότητας, οι τρείς μορφές χρήματος συνοψίζονται υπό την μορφή πίνακα.

Η επόμενη ενότητα (ενότητα 3), αφορά το blockchain. Κατά την εισαγωγή της ενότητας, γίνεται αναφορά με ποιόν τρόπο η συσσώρευση των λειτουργιών του διαδικτύου μέσω κεντρικών υπηρεσιών (Web2), επιφέρει την εμπορευματοποίηση των προσωπικών μας δεδομένων και στη συνέχεια παρουσιάζονται συνοπτικά τα περιεχόμενά της. Μετά την εισαγωγή, ακολουθεί παρουσίαση της τεχνολογίας blockchain αναφέροντας τεχνικά χαρακτηριστικά, με τρόπο εύκολο προς την παρακολούθηση. Η ενότητα ολοκληρώνεται με μία συνοπτική αναφορά στα δίκτυα Bitcoin, Ethereum και Polygon και την αναλυτική παρουσιαστή του κάθε δικτύου, ξεχωριστά.

Φτάνοντας την ενότητα τέσσερα (4), ο αναγνώστης είναι έτοιμος να κατανοήσει το πώς λειτουργεί η υπηρεσία επιβεβαίωσης, δηλαδή η υπηρεσία που παρουσιάζεται σε αυτή την εργασία. Αρχικά, γίνεται μια απλή περιγραφή της υπηρεσίας η οποία ακολουθείται από μία πιο λεπτομερή περιγραφή, στην οποία αναφέρονται τα "βήματα" που ακολουθούνται από την υπηρεσία. Τέλος, γίνεται μία σύντομη αναφορά στα μέσα τα οποία χρησιμοποιήθηκαν για την ανάπτυξή της.

Στην προτελευταία ενότητα (ενότητα 5) και έχοντας εξοικειωθεί με τις διαδικασίες - βήματα που υλοποιούν την υπηρεσία, γίνεται παρουσίαση του κώδικα της υπηρεσίας, ο οποίος οργανώνεται σε τρείς ομάδες. Η πρώτη ομάδα, αφορά τα έξυπνα συμβόλαια της υπηρεσίας, δηλαδή κώδικα που καταχωρείται στο δίκτυο blockchain, η δεύτερη ομάδα αφορά κώδικα που χρησιμοποιείται άμεσα από τον χρήστη, ενώ η τελευταία ομάδα αφορά τα αρχεία του χρησιμοποιούνται για την υποστήριξη της υπηρεσίας.

Η έκτη (6η) και τελευταία ενότητα αποτελεί τον επίλογο της εργασίας στην οποία περιγράφεται συνοπτικά ο τρόπος με ποιον τρόπο γίνετε η σύνδεση με την υπηρεσία, οι εκτιμήσεις σχετικά με αυτή, καθώς και η επόμενη επιδίωξη του συγγραφέα, όσο αφορά το Web3.

2 ΕΓΡΧΗΜΑΤΕΣ ΣΥΝΑΛΛΑΓΕΣ

Η διαδικασία της ανταλλαγής προϊόντων ή υπηρεσιών, είναι μία διαδικασία που παρατηρείται όχι μόνο στον άνθρωπο αλλά σε διάφορους έμβιους οργανισμούς. Ερευνητές που μελετούσαν την συμπεριφορά των αιγυπτιακών κροκοδείλων, διαπίστωσαν ότι έχουν αναπτύξει μία συναλλαγματική [2] σχέση με τα τοπικά παρυδάτια πτηνά κατά την οποία, τα πτηνά εισέρχονται στο στόμα των κροκοδείλων με σκοπό να τραφούν από τα υπολείμματα τροφής που βρίσκονται εγκλωβισμένα ανάμεσα στα δόντια τους, προσφέροντάς τους υπηρεσίες στοματικής υγιεινής.

Για τον άνθρωπο, η λέξη «συναλλαγή» [3], αναφέρεται στην διαδικασία κατά την οποία ένας πωλητής, παραχωρεί ένα αντικείμενο ή προσφέρει μία υπηρεσία σε έναν αγοραστή, έναντι κάποιου προκαθορισμένου χρηματικού ποσού. Παρόλο που το ποσό που πρόκειται να λάβει ο πωλητής είναι προσυμφωνημένο, δηλαδή προκαθορισμένης αξίας, δεν είναι απαραίτητα και προκαθορισμένης μορφής. Συχνά οι πωλητές δίνουν την δυνατότητα στους αγοραστές να επιλέξουν την μορφή του χρήματος, με σκοπό να αυξήσουν το αγοραστικό τους κοινό. Σε αυτή την ενότητα λοιπόν θα παρουσιαστούν οι βασικές μορφές χρήματος (υλικό και άυλο), τι είναι το ψηφιακό χρήμα και πως συνδέεται το ψηφιακό χρήμα με τα δίκτυα blockchain.

2.1 Υλικό χρήμα

Ξεκινώντας με την ποιο απλή μορφή χρήματος, ως υλικό χρήμα [4] χαρακτηρίζονται τα αντικείμενα τα οποία χρησιμοποιούνται από τα μέλη μιας κοινωνίας ως ανταλλακτικό μέσο. Για την ευκολότερη κατανόηση της παραπάνω φράσης, ας θέσουμε ένα παράδειγμα κατά το οποίο δύο παραγωγοί διαθέτουν προϊόντα προς πώληση, με τον πρώτο παραγωγό να διαθέτει πατάτες και τον δεύτερο παραγωγό να διαθέτει ελιές.

Οι παραγωγοί, με σκοπό να αυξήσουν την ποικιλία των αγαθών που διαθέτουν προς πώληση, θα ήθελαν να έχουν την δυνατότητα να εμπορεύονται περισσότερα είδη προϊόντων από αυτά που παράγουν. Για να το πετύχουν αυτό, αναπτύσσουν μεταξύ τους μία συμφωνία κατά την οποία, ο παραγωγός της πατάτας παραχωρεί ένα μέρος της σοδιάς του στον ελαιοπαραγωγό και λαμβάνει από αυτόν μέρος της δικής του σοδιάς, δηλαδή ελιές. Η παραπάνω διαδικασία, αποτελεί παράδειγμα εμπορικής συναλλαγής όπου δύο προϊόντα εγγενούς (πραγματικής) αξίας χρησιμοποιούνται ως υλικό χρήμα.

Στη συνέχεια, οι παραγωγοί πηγαίνουν στην τοπική αγορά, όπου διαθέτουν το εμπόρευμά τους προς πώληση έναντι μίας ποσότητας παραστατικού χρήματος (fiat money). Για να μπορέσει ο καταναλωτής να αγοράσει κάποιο προϊόν (π.χ. ελιές), αρκεί να έχει στη διάθεσή του την απαραίτητη ποσότητα παραστατικού χρήματος και να την διαθέσει τον πωλητή. Ως παραστατικό χρήμα περιγράφονται υλικά αντικείμενα (κέρματα και χαρτονομίσματα) αμελητέας πραγματικής αξίας, η αξία των οποίων προσδιορίζεται μέσω κυβερνητικής απόφασης. Η αξία τους είναι πρακτικά ονομαστική, και αρκετές φορές χαρακτηρίζονται ως «μετρητά».

Και για τις δύο συναλλαγές (συναλλαγή μεταξύ παραγωγών και συναλλαγή μεταξύ παραγωγού και καταναλωτή), η ευθύνη της σωστής διεκπεραίωσης βαραίνει τους συναλλασμένους, οι οποίοι θα πρέπει να επιβεβαιώσουν ότι τα προϊόντα ή/και τα χρήματα που έλαβαν είναι ποσοτικά και ποιοτικά ακέραια. Σημαντικό είναι να αναφερθεί ότι συνήθως, οι συναλλαγές υλικού χρήματος πραγματοποιούνται χωρίς να είναι αναγκαία η ανταλλαγή προσωπικών στοιχείων ή ευαίσθητων

πληροφοριών μεταξύ των συναλλασμένων, πράγμα που προστατεύει την ιδιωτικότητά τους.

2.2 Άυλο χρήμα

Σε αντίθεση με το υλικό παραστατικό χρήμα, που έχει κάποια πραγματική αξία λόγω των υλικών από το οποίο κατασκευάζεται, το άυλο χρήμα αποτελεί ένα είδος παραστατικού χρήματος με μηδενική πραγματική αξία, αφού δεν κατασκευάζεται από κάποιο υλικό αλλά αποτελεί λογιστικό προϊόν. Η αγοραστική αξία του άυλου παραστατικού χρήματος είναι ανά μονάδα ίση με αυτή του αντίστοιχου υλικού και προσδιορίζεται μέσω κρατικού διατάγματος, όπως ακριβώς γίνεται και για το υλικό. Κύριο χαρακτηριστικό του άυλου χρήματος είναι η διευκόλυνση των συναλλαγών, αφού τα συναλλασσόμενα μέρη μπορούν να πραγματοποιήσουν μεταφορά χρήματος απομακρυσμένα, χωρίς να είναι αναγκαία η φυσική τους παρουσία σε κάποια προσυμφωνημένη τοποθεσία.

Παρόλο όμως που η χρήση άυλου χρήματος διευκολύνει την πραγματοποίηση των συναλλαγών, δεν διατίθεται προς όλα τα μέλη της κοινωνίας, αφού οι συναλλαγές άυλου χρήματος εξυπηρετούνται από πιστωτικά ιδρύματα τα οποία διαθέτουν τις υπηρεσίες τους υπό όρους και προϋποθέσεις. Για να μπορέσει να χρησιμοποιήσει άυλο χρήμα, ο ενδιαφερόμενος καλείται να παραχωρήσει στο πιστωτικό ίδρυμα τα προσωπικά του στοιχεία καθώς και άλλες ευαίσθητες πληροφορίες. Έχοντας στη διάθεσή του τα παραπάνω, το ίδρυμα πραγματοποιεί μία σειρά ελέγχων και εφόσον δεν υπάρξει κάποιο πρόβλημα, επιτρέπετε στον ενδιαφερόμενο η χρήση του.

Θα πρέπει να αναφερθεί ότι όταν μία συναλλαγή πραγματοποιείται με χρήση άυλου χρήματος, ο έλεγχος της συναλλαγής βαραίνει το πιστωτικό ίδρυμα. Για το λόγο αυτό, τα συναλλασσόμενα μέλη είναι αναγκασμένα να εμπιστευτούν ότι η διαδικασία ελέγχου που πραγματοποιείται από το ίδρυμα είναι επαρκής, καθώς και να αποδεχτούν ότι οι πληροφορίες της συναλλαγής δύναται να κοινοποιηθούν σε τρίτους, ακούσια ή εκούσια, μέσω αυτού.

2.3 Ψηφιακό χρήμα

Πέρα του υλικού και του άυλου, υπάρχει μία κατηγορία χρήματος η οποία ονομάζεται ψηφιακό χρήμα [5]. Το ψηφιακό, αν και μοιάζει να εντάσσεται στην κατηγορία του άυλου χρήματος λόγω της μη φυσικής του υπόστασης, αποτελεί ξεχωριστή κατηγορία χρήματος, αφού η πρόσβαση σε αυτό γίνεται αποκλειστικά μέσω υπολογιστών ή/και ψηφιακών πορτοφολιών τα οποία έχουν πρόσβαση στο παγκόσμιο ιστό (worldwide web) ή σε κάποιο εξειδικευμένο δίκτυο και όχι μέσω κάποιου πιστωτικού ιδρύματος.

Όπως αναφέραμε, το ψηφιακό χρήμα είναι ένα ψηφιακό προϊόν, το οποίο δεν εξυπηρετείται από τις τράπεζες ή κάποιας άλλης μορφής πιστωτικού ιδρύματος (εξαιρουμένης της περίπτωσης όπου το ψηφιακό νόμισμα έχει δημιουργηθεί από κάποιο πιστωτικό ίδρυμα), αλλά μέσω συνδέσεων δικτύου. Για να γίνει πιο εύκολα κατανοητό το, τί ακριβώς είναι το ψηφιακό χρήμα ας θέσουμε ένα παράδειγμα το οποίο αφορά ένα ηλεκτρονικό παιχνίδι.

Ο χρήστης του παιχνιδιού έχοντας αφιερώσει αρκετές ώρες στη ίδια πίστα αποφασίζει να αγοράσει μέσω του παιχνιδιού μάρκες, ώστε να βοηθηθεί και να προχωρήσει στην επόμενη πίστα. Οι μάρκες αυτές εκδίδονται από την εταιρία που έχει κατασκευάσει το παιχνίδι και είναι διαθέσιμες αποκλειστικά για τους χρήστες των παιχνιδιών της. Όταν πραγματοποιείται η αγορά, ο χρήστης ξοδεύει μία ποσότητα άυλου χρήματος και λαμβάνει μια ποσότητα ψηφιακού χρήματος. Το ψηφιακό χρήμα που έλαβε αν και αντιστοιχεί σε μία ποσότητα παραστατικού χρήματος, δεν έχει

αγοραστική αξία, αφού η αξία του ψηφιακού χρήματος που έλαβε είναι καθαρά υποκειμενική και εξαρτάται άμεσα με το πόσο πολύ θέλει να συνεχίσει να παίζει το παιχνίδι και προχωρήσει στην επόμενη πίστα.

Μελετώντας τα παραπάνω διαπιστώνει κανείς ότι το ψηφιακό χρήμα είναι ένα είδος χρήματος μηδενικής αγοραστικής αξίας το οποίο εξυπηρετεί συγκεκριμένους στόχους. Το 2008 ο Satoshi Nakamoto [6] (πιθανολογείται ότι είναι ψευδώνυμο και όχι πραγματικό όνομα), δημοσίευσε ένα άρθρο με θέμα τη δημιουργία ενός ψηφιακού νομίσματος, το οποίο εξυπηρετείται από ένα δίκτυο ομότιμων κόμβων και ελέγχεται μέσω κρυπτογραφίας, με στόχο την γενικότερη διευκόλυνση των χρηματικών συναλλαγών.

Η τεχνολογία στην οποία αναφέρεται το άρθρο του Satoshi, η τεχνολογία blockchain, στηρίζεται στην ύπαρξή μίας "κατανεμημένης βάσης δεδομένων" ή αλλιώς στη διατήρηση ενός "κατανεμημένου σημειωματάριου" (distributed ledger) και του αλγοριθμικού ελέγχου των μεταβολών (της βάσης δεδομένων / του σημειωματάριου), μέσω κρυπτογραφικών διαδικασιών. Ο τρόπος με τον οποίο δομούνται τα δίκτυα blockchain θα παρουσιαστεί στην επόμενη ενότητα, αλλά στο σημείο αυτό είναι σημαντικό να αναφερθεί πως τα δίκτυα αυτά μεταξύ άλλων επιτρέπουν:

1. Την πραγματοποίηση ανώνυμων συναλλαγών ψηφιακού χρήματος
2. Την εξυπηρέτηση ανώνυμων συμφωνιών - συμβολαίων, αορίστου περιεχομένου
3. Την πιστοποίηση των συναλλαγών και τον έλεγχο των συμφωνιών μέσω κρυπτογραφίας
4. Την δυνατότητα ελέγχου της ακεραιότητας του δικτύου, από το ίδιο το δίκτυο

Λαμβάνοντας υπόψη τα παραπάνω, η χρήση ψηφιακού χρήματος μέσω δικτύων blockchain (χρήση κρυπτονομίσματος) μοιάζει να συνδυάζει τα θετικά της χρήσης υλικού και άυλου χρήματος, με το μόνο μειονέκτημα να εντοπίζεται στην υποκειμενική του αξία, το οποίο όμως "μειώνεται" όσο αυξάνεται η ζήτησή του. Για την ευκολότερη κατανόηση της παραπάνω πρότασης και συνοψίζοντας τα περιεχόμενα του ενότητας, καταγράφηκαν τα χαρακτηριστικά των υπό εξέταση μορφών χρήματος σε μορφή πίνακα.

ζήτημα	**Υλικό χρήμα**	**Άυλο χρήμα**	**Ψηφιακό χρήμα**
τρόπος πραγματοποίησης συναλλαγών	δια ζώσης	απομακρυσμένα	απομακρυσμένα
βάρος ελέγχου συναλλαγών	μεταξύ συναλλασμένων	πιστωτικό ίδρυμα	κατανεμημένος κρυπτογραφικός αλγόριθμος
δυνατότητα ανώνυμων συναλλαγών	ναι	όχι	ναι
αξία χρήματος	πραγματική, παραστατική	παραστατική	υποκειμενική
δυνατότητα χρήσης	από όλους	υπό προϋποθέσεις	από όλους

Πίνακας 1. Χαρακτηριστικά υλικού, άυλου και ψηφιακού χρήματος

3 BLOCKCHAIN

Ζώντας στην "ψηφιακή εποχή", βρισκόμαστε σε μία εποχή ορόσημο για την ανθρωπότητα. Από τον πρώτο ηλεκτρονικό υπολογιστή μέχρι το διαδίκτυο των πάντων, βιώνουμε τις αλλαγές που φέρνει η τεχνολογία στην καθημερινότητά μας, χωρίς απαραίτητα να το συνειδητοποιούμε. Παρόλο που ο ρυθμός ανάπτυξης της τεχνολογίας και η γενικότερη χρήση των τεχνολογικών μέσων δεν αποτελεί καθαυτό πρόβλημα, υπάρχουν εταιρίες αλλά και ιδιώτες όπου διαπιστώθηκε ότι χρησιμοποιούσαν τα διαθέσιμα τεχνολογικά μέσα, αθέμιτα, με σκοπό το προσωπικό όφελος.

Σύμφωνα με ένα άρθρο του NBCnews [7], ο διευθύνον σύμβουλος της εταιρίας Meta (πρώην Facebook) Mark Zuckerberg, αλλά και άλλα υψηλόβαθμα στελέχη της ίδιας εταιρίας, χρησιμοποιούσαν τα δεδομένα που συλλέγονταν μέσω της πλατφόρμας Facebook, ως μέσω εκβιασμού, προς τις συνεργαζόμενες (με την Meta) εταιρίες. Επίσης, στο ίδιο άρθρο αναφέρεται ότι η εταιρία παραχωρούσε πρόσβαση στα προσωπικά δεδομένα των χρηστών σε τρίτους, ανάλογα με τα χρήματα που ξόδευαν στην πλατφόρμα, την προσωπική τους σχέση με τον Zuckerberg και το πόσο πρόθυμοι ήταν να συμμετέχουν στη γενικότερη ανταλλαγή ευαίσθητων δεδομένων με την εταιρία του. Η αναφορά στο άρθρο του NBCnews αν και συγκεκριμένη, αποτελεί χαρακτηριστικό παράδειγμα των κινδύνων που επιφέρει η χρήση των σύγχρονων τεχνολογικών μέσων αλλά και η χρήση κεντρικών υπηρεσιών (centralized services).

Ως κεντρικές, χαρακτηρίζονται οι υπηρεσίες οι οποίες εξυπηρετούνται μέσω συγκεκριμένων εξυπηρετητών και δικτύων, διατηρούν τον κώδικά τους κρυφό και ανήκουν σε κάποια ομάδα ανθρώπων ή σε κάποια εταιρία. Λόγω του κρυφού χαρακτήρα των υπηρεσιών αυτών, οι χρήστες δεν έχουν τη δυνατότητα να γνωρίζουν επακριβώς τι συμβαίνει κατά τη χρήση της υπηρεσίας και είναι αναγκασμένοι να εμπιστευτούν τους ισχυρισμούς της εκάστοτε ομάδας ή εταιρίας καθώς και τα πορίσματα των ελεγκτικών αρχών. Αυτό σημαίνει ότι, η χρήση κεντρικών διαδικτυακών υπηρεσιών είναι πρακτικά μη ελεγχόμενη. Σήμερα, έχοντας εντοπίσει τα ζητήματα που προκύπτουν από τη χρήση κεντρικών υπηρεσιών, υπάρχουν προγραμματιστές που προσπαθούν να αναδημιουργήσουν τις υπάρχουσες υπηρεσίες ως αποκεντρωμένες (decentralized services).

Οι αποκεντρωμένες υπηρεσίες βασίζονται σε κατανεμημένα δίκτυα ομότιμων κόμβων, με τον κάθε κόμβο να λειτουργεί ανεξάρτητα από τους όμοιούς του, έχοντας όμως τα ίδια δικαιώματα και δυνατότητες με αυτούς. Τα ποιο γνωστά παραδείγματα αποκεντρωμένων υπηρεσιών, είναι τα peer to peer (p2p) δίκτυα torrent, στα οποία οι κόμβοι ανταλλάσσουν αρχεία μεταξύ τους και τα δίκτυα blockchain, όπου αποτελούν και το θέμα της ενότητας αυτής αλλά και το κύριο εργαλείο για την δημιουργία της υπηρεσίας επιβεβαίωσης, η οποία παρουσιάζεται σε αυτή την εργασία.

Αρχικά λοιπόν, θα γίνει μια συνοπτική παρουσίαση της τεχνολογίας blockchain. Έχοντας κατανοήσει τις δυνατότητες που προσφέρει η τεχνολογία αυτή, θα αναφερθούμε στο δίκτυο Bitcoin, όπου αποτελεί και το πρώτο πετυχημένο εμπορικά δίκτυο blockchain, τις δυνατότητές που προσφέρει καθώς και τον τρόπο λειτουργίας του. Στη συνέχεια, θα αναφερθούμε στο δίκτυο Ethereum, εστιάζοντας στο πως διαφοροποιείται από εκείνο του Bitcoin και τέλος στο δίκτυο Polygon, αναφέροντας τους λόγους επιλογής του ως "βάση" για την υπηρεσία επιβεβαίωσης.

Σκοπός της ενότητας είναι, ο αναγνώστης να κατανοήσει την τεχνολογία blockchain ως έννοια και όχι να εμβαθύνει από τεχνικής άποψης, αφού το δίκτυο Polygon και γενικότερα η τεχνολογία του blockchain αποτελούν το εργαλείο ανάπτυξης της υπηρεσίας και όχι το θέμα της εργασίας.

3.1 Τεχνολογία

Έχοντας ασχοληθεί με τα δίκτυα και γενικότερα με την τεχνολογία του blockchain, αντιλαμβάνεται κανείς ότι, η τεχνολογία αυτή έχει ως στόχο την δημιουργία αξιόπιστων υπηρεσιών, μέσω αυτοελεγχόμενων κατανεμημένων δικτύων. Για να φτάσουμε όμως στο συμπέρασμα αυτό, ας πάρουμε τα πράγματα από την αρχή, ξεκινώντας από την ερμηνεία της λέξης «blockchain».

Η λέξη «blockchain» [8], περιγράφει μία αλυσίδα (chain) από ψηφιακά πακέτα (blocks) τα οποία συνδέονται μεταξύ τους χρονολογικά, μέσω κρυπτογραφίας. Τα πακέτα της αλυσίδας, στη γενικότερή τους μορφή αποτελούνται από δύο τμήματα, με το πρώτο να περιλαμβάνει τις πληροφορίες σχετικά με την αλυσίδα (block header), όπως το μοναδικό αναγνωριστικό του πακέτου (block hash), το αναγνωριστικό του προηγούμενου χρονικά πακέτου (previous block hash) και την χρονική στιγμή (block timestamp) κατά την οποία δημιουργήθηκε το πακέτο και το δεύτερο τμήμα (block body), στο οποίο εμπεριέχονται οι μεταβολές της κατάστασης της αλυσίδας, δηλαδή οι συναλλαγές. Το αναγνωριστικό που περιέχεται στην κεφαλίδα (πρώτο τμήμα) του πακέτου, λειτουργεί ως απόδειξη/επιβεβαίωση για το ποιες είναι η μεταβολές που περιλαμβάνονται στο σώμα (δεύτερο τμήμα) του πακέτου και δημιουργείται μέσω αλγόριθμου κρυπτογράφησης. Χρησιμοποιώντας τις πληροφορίες τις κεφαλίδας, ο κάθε κόμβος του δικτύου οργανώνει τα blocks χρονολογικά, δημιουργώντας ένα τοπικό αντίγραφο της κατάστασης της αλυσίδας. Στη συνέχεια, ανά τακτά χρονικά διαστήματα, οι κόμβοι συγκρίνουν το αντίγραφό τους με αυτό των υπόλοιπων κόμβων, με σκοπό να εντοπίσουν τυχών μεταβολές στην κατάσταση της αλυσίδας και να τις συμπεριλάβουν στο τοπικό τους αντίγραφο. Στην περίπτωση που κατά το συγχρονισμό εντοπιστεί ασυμφωνία μεταξύ του αναγνωριστικού και του περιεχομένου του σώματος ενός πακέτου, οι κόμβοι που περιλαμβάνουν τα μη έγκυρα blocks τιμωρούνται, ανάλογα με τους κανόνες του δικτύου (αυτοέλεγχος).

Η δημιουργία των πακέτων και κατά επέκταση της αλυσίδας, αποτελείται από τέσσερις διαδοχικές διαδικασίες:

1) Συναλλαγή: Κατά τη διαδικασία της συναλλαγής, ο χρήστης καταχωρεί μία υπογεγραμμένη εντολή τροποποίησης της αλυσίδας. Η υπογραφή χρησιμοποιείται για να "αποδείξει" στο δίκτυο ότι ο εντολέας της συναλλαγής, έχει το δικαίωμα να τροποποιήσει την κατάσταση της αλυσίδας κατά τον τρόπο που περιγράφεται σε αυτή. Για την υπογραφή της συναλλαγής χρησιμοποιείται ένα ιδιωτικό κλειδί (private key), το οποίο ανήκει και είναι γνωστό μόνο στον εντολέα, ενώ ως εντολέας φαίνεται ένα δημόσιο κλειδί (public key), το οποίο προκύπτει από το ιδιωτικό. Σημαντικό είναι να αναφερθεί ότι τα κλειδιά, ιδιωτικά και δημόσια, δεν αντιστοιχίζονται άμεσα με την πραγματική ταυτότητα του εντολέα, επιτρέποντας την εκτέλεση ανώνυμων συναλλαγών.

2) Συλλογή: Η καταχώριση μιας συναλλαγής στο δίκτυο, δεν συνεπάγει την άμεση καταχώρησή της στην αλυσίδα. Μετά την υπογραφή, η συναλλαγή λαμβάνει έναν χαρακτηρισμό, ο οποίος δηλώνει ότι η εντολή τροποποίησης της αλυσίδας εκκρεμεί προς επικύρωση. Το δίκτυο, συλλέγει τις "προς επικύρωση" συναλλαγές και τις διαθέτει στους κόμβους του δικτύου, ξεκινώντας τη διαδικασία της επικύρωσης.

3) Επικύρωση: Η διαδικασία της επικύρωσης αποτελεί μια ανταγωνιστική διαδικασία για τους κόμβους του δικτύου. Ο κάθε κόμβος ενεργώντας "εγωιστικά", προσπαθεί να επικυρώσει αυτός τις μη καταχωρημένες στην αλυσίδα συναλλαγές, με σκοπό να ανταμειφθεί από το δίκτυο. Η μέθοδος της επικύρωσης, παρόλο που διαφοροποιείται ανάλογα με τον μηχανισμό συναίνεσης του δικτύου, έχει ως αποτέλεσμα την δημιουργία ενός νέου πακέτου (block), του οποίου η γενική μορφή περιγράφηκε προηγουμένως. Ο κόμβος, που καταφέρνει να επικυρώσει τις συναλλαγές, αρχικά περιλαμβάνει το νέο πακέτο στην τοπική του αλυσίδα και στη συνέχεια κοινοποιεί την "νέα" αλυσίδα στους υπόλοιπους κόμβους του δικτύου.

4) Συγχρονισμός: Φτάνοντας στο τέταρτο και τελευταίο βήμα, το δίκτυο ανακοινώνει το νέο block στους υπόλοιπους κόμβους οι οποίοι, αφού το ελέγξουν, το προσθέτουν στο τοπικό τους αντίγραφο. Μετά τον συγχρονισμό, η διαδικασία επέκτασης της αλυσίδας επαναλαμβάνεται από την αρχή, επεκτείνοντάς τη επ' αόριστόν (εφόσον υπάρχουν κόμβοι στο δίκτυο).

Συνοψίζοντας, αναφέραμε πως η τεχνολογία του blockchain περιλαμβάνει δίκτυα ομότιμων κόμβων, όπου ο κάθε κόμβος αποθηκεύει τοπικά την κατάσταση της αλυσίδας, δηλαδή το σύνολο των συναλλαγών του δικτύου, διατηρώντας της χρονολογική τους σειρά. Επίσης, αναφέραμε πως η κατάσταση της αλυσίδας μεταβάλλεται ανάλογα με το περιεχόμενο του σώματος των πακέτων που την αποτελούν, ενώ ο έλεγχος της κατάστασης της αλυσίδας γίνεται διασταυρώνοντας τα περιεχόμενα της κεφαλίδας και του σώματος, για κάθε πακέτο ξεχωριστά. Όσο αφορά την δημιουργία των πακέτων και την επέκταση της αλυσίδας, ακολουθούνται τέσσερα (4) διακριτά βήματα ή τέσσερεις διακριτές διαδικασίες, όπου αυτές είναι η συναλλαγή, η συλλογή, η επικύρωση και ο συγχρονισμός.

Ξεκινώντας με την περιγραφή της τεχνολογίας του blockchain, υποστηρίξαμε ότι η τεχνολογία αυτή παρέχει την δυνατότητα δημιουργίας αξιόπιστων, μη κεντρικών υπηρεσιών. Έχοντας απαντήσει στο ερώτημα "Ποιες διαδικασίες περιγράφουν την τεχνολογία blockchain;", είμαστε σε θέση να δικαιολογήσουμε την αρχική μας πρόταση, εστιάζοντας στη διαδικασία της επικύρωσης.

Η διαδικασία της επικύρωσης, όπως αναφέραμε προηγουμένως αποτελεί μια διαδικασία ανταγωνισμού για τους κόμβους του δικτύου. Ο κάθε κόμβος, είτε καταθέτοντας ποσότητες υπολογιστικής ισχύς (Proof of Work), είτε καταθέτοντας ποσά ψηφιακού χρήματος ως εγγύηση (Proof of Stake), είτε μέσω κάποιου άλλου μηχανισμού συναίνεσης (consensus mechanism), προσπαθεί να αποδείξει ότι είναι πιο αξιόπιστος από τους υπόλοιπους, με σκοπό να επιλεγεί από το δίκτυο ως επικυρωτής και να ανταμειφθεί. Το δίκτυο, σε κάθε κύκλο επέκτασης της αλυσίδας, επιλέγει ένα διαφορετικό κόμβο ως επικυρωτή ανάλογα με το ποσοστό αφοσίωσής του (ως προς τη συνολική αφοσίωση των κόμβων), κάνοντας το δίκτυο να μην έχει ένα μοναδικό σημείο αποτυχίας (no single point of failure), αφού η ευθύνη επικύρωσης των πακέτων διαμοιράζεται μεταξύ όλων των κόμβων. Στην περίπτωση που ο επιλαχών κόμβος εισάγει στην αλυσίδα ένα λανθασμένο block, τότε τιμωρείτε ανάλογα με το μηχανισμό συναίνεσης και το block που εισήγαγε στην αλυσίδα διαγράφεται, διατηρώντας την αλυσίδα πραγματική. Οι κόμβοι, έχοντας την ανταμοιβή και όχι την τιμωρία ως κίνητρο συμμετοχής, κάνουν την τεχνολογία blockchain ένα αξιόπιστο μέσο επικύρωσης συναλλαγών, το οποίο διαθέτει όλα του τα δεδομένα προς έλεγχο (open source), μιας και η ανάγνωση των περιεχομένων της αλυσίδας είναι ελεύθερη προς όλους τους χρήστες.

3.2 Δίκτυα

Η πρώτη, ιστορικά γνωστή, αναφορά σε ένα πρωτόκολλο όμοιο με αυτό του blockchain έγινε το 1982 με τη διατριβή του David L. Chaum, με τίτλο «Computer Systems Established, Maintained and Trusted by Mutually Suspicious Groups» [9]. Το 1991, οι W. Scott Stornetta και Stuart Haber επέκτειναν την ιδέα του Chaum με τη δημοσίευσή τους «How to Time-Stamp a Digital Document» [10], ωστόσο η ιδέα του blockchain έμεινε αναξιοποίητη μέχρι το 2009, όπου ο Satoshi Nakamoto (πιθανολογείτε ότι είναι ψευδώνυμο) κοινοποίησε το δίκτυο Bitcoin, το οποίο και είχε περιγράψει ένα (1) χρόνο νωρίτερα. Το δίκτυο του Satoshi είναι το πρώτο εμπορικά και πιθανώς ιστορικά δίκτυο blockchain και αποτελεί ορόσημο, αφού "απέδειξε" ότι η υλοποίηση δικτύων αυτού του είδους είναι εφικτή. Μετά την εμπορική και τεχνολογική επιτυχία του Bitcoin ακολούθησε η κοινοποίηση διάφορων νέων δικτύων, όπως το δίκτυο Ethereum [11] (2013-2015) που έδωσε στους χρήστες την δυνατότητα να καταχωρίσουν στο δίκτυο συναλλαγές αλλά και εκτελέσιμο από το δίκτυο κώδικα και το δίκτυο Polygon [12] (2017), το οποίο αποτελεί μία "επέκταση" του δικτύου Ethereum.

3.2.1 Bitcoin

Το δίκτυο Bitcoin [13], υποστηρίζει ένα κατανεμημένο δημόσιο κατάλογο (shared public ledger) στον οποίο εμπεριέχονται όλες οι πιστοποιημένες συναλλαγές του δικτύου, ταξινομημένες χρονολογικά. Η χρονολογική σειρά των συναλλαγών καθώς και τα δεδομένα της κάθε συναλλαγής, εξασφαλίζονται μέσω ενός κρυπτογραφικού αλγορίθμου και συγκεκριμένα του SHA256 (Secure Hash Algorithm 256). Για να μπορέσει κάποιος να καταχωρίσει μια συναλλαγή, θα πρέπει να έχει ένα Bitcoin πορτοφόλι (Bitcoin wallet), στο οποίο να έχει πιστωθεί και να είναι διαθέσιμη η απαιτούμενη ποσότητα BTC (το ψηφιακό νόμισμα ή κρυπτονόμισμα του δικτύου).

Το Bitcoin, ως υπηρεσία που αξιοποιεί την τεχνολογία Blockchain, έχει ως στόχο την δημιουργία μιας παγκόσμιας συναλλαγματικής πολιτικής, όπου:

1. Αντικαθιστά το σύνολο των παραστατικών νομισμάτων, με ένα παγκόσμιο "αμερόληπτο" νόμισμα.
2. Προστατεύει την ανωνυμία των συναλλασσόμενων αξιοποιώντας εικονικά πορτοφόλια, τα οποία δεν μπορούν να αντιστοιχιστούν (άμεσα) με την πραγματική ταυτότητα του ιδιοκτήτη (τους).
3. Πιστοποιεί και ελέγχει τις συναλλαγές, μέσω μιας κατανεμημένης λογικής.
4. Παύει την ανάγκη ύπαρξης κάποιας ελεγκτικής αρχής (π.χ. τράπεζα), αφού ο έλεγχος των συναλλαγών γίνεται από το ίδιο το δίκτυο (trustless transaction).

Εξετάζοντας στο ζήτημα του Bitcoin πορτοφολιού, το εικονικό αυτό πορτοφόλι αποτελείται από ένα ιδιωτικό κλειδί (private key) και μία δημόσια διεύθυνση (public address) η οποία προκύπτει από αυτό (από το ιδιωτικό κλειδί). Χρησιμοποιώντας τις εγγραφές του δικτύου και θέτοντας ως "είσοδο" μία δημόσια διεύθυνση, ο χρήστης, αλλά και το ίδιο το δίκτυο, έχει άμεση γνώση σχετικά με την ποσότητα BTC που αντιστοιχεί στο αντίστοιχο πορτοφόλι. Για να μπορέσει να γίνει μία συναλλαγή, δηλαδή μία μεταφορά BTC από ένα πορτοφόλι σε ένα άλλο, ο αποστολέας δημιουργεί μια μη υπογεγραμμένη εντολή η οποία περιλαμβάνει την ποσότητα BTC που θέλει να μεταφέρει, την δημόσια διεύθυνση του παραλήπτη και κάποιες άλλες απαραίτητες για τη διεκπεραίωση της

συναλλαγής πληροφορίες. Στη συνέχεια, χρησιμοποιώντας το ιδιωτικό του κλειδί, ο αποστολέας υπογράφει την μέχρι τώρα μη υπογεγραμμένη εντολή και την προωθεί στο δίκτυο προς επικύρωση και καταχώρηση.

Οι επικυρωτές [14] (validators ή miners) από μέρους τους, έχοντας στη διάθεσή τους όλες τις εκκρεμείς συναλλαγές, προσπαθούν να τις καταχωρίσουν στο δίκτυο ώστε να λάβουν μία ποσότητα BTC ως ανταμοιβή (mining) και να επεκτείνουν τη αλυσίδα. Ένας επικυρωτής για να μπορέσει να καταχωρίσει το νέο πακέτο (block) στην αλυσίδα, αρχικά ελέγχει τα περιεχόμενα του προς καταχώρηση πακέτου και αφού βεβαιωθεί για την αρτιότητά τους, ξεκινάει μία διαδικασία δοκιμών η οποία λέγετε (hashing).

Ο αλγόριθμος κρυπτογράφησης SHA256 [15] αποτελεί μία διαδικασία μετατροπής ενός μηνύματος αορίστου μήκους, σε ένα κείμενο σταθερού μήκους το οποίο αποτελείται από 64 (εξήντα τέσσερεις) χαρακτήρες του δεκαεξαδικού συστήματος αρίθμησης (hexadecimal numeral system). Ο SHA256 και γενικότερα οι αλγόριθμοι αυτού του είδους έχουν την ιδιότητα, η αντιστοίχιση μεταξύ εισόδου και εξόδου είναι εφικτή μόνο από την είσοδο προς την έξοδο (one way encryption). Αυτό σημαίνει ότι για να βρεθεί μια συγκεκριμένη έξοδος, θα πρέπει να δοκιμαστούν τυχαία μηνύματα εισόδου. Αν και θεωρητικά η εύρεση μιας συγκεκριμένης εξόδου είναι εφικτή, με τις τεχνολογίες και τα μέσα του σήμερα είναι πρακτικά αδύνατη, αφού ο χρόνος που απαιτείται για την διαδικασία δοκιμών είναι μεγάλος (χρειάζονται 2^{256} ή περίπου $1.158 * 10^{77}$ δοκιμές).

Κατά τη διαδικασία των δοκιμών (hashing) λοιπόν, οι επικυρωτές θέτοντας ως είσοδο δεδομένα του νέου block αλλά και έναν επιπλέον τυχαίο αριθμό στον SHA256, προσπαθούν να δημιουργήσουν μία έξοδο, η αριθμητική τιμή της οποίας να είναι ίση ή μικρότερη σε σχέση με έναν αριθμό "στόχο" [16]. Η διαδικασία αυτή γίνετε ξεχωριστά από τον κάθε επικυρωτή και είναι μία ανταγωνιστική διαδικασία, αφού μόνο ο πρώτος που θα καταφέρει να δημιουργήσει μία αποδεκτή έξοδο, θα αμειφθεί (σε BTC). Το δίκτυο Bitcoin είναι φτιαγμένο ώστε κάθε επέκταση της αλυσίδας να συμβαίνει περίπου κάθε 10 (δέκα) λεπτά [17]. Έτσι, γνωρίζοντας τον ρυθμό δοκιμών (hash rate) που πραγματοποιούνται από όλους τους επικυρωτές του δικτύου (συνολικά), το δίκτυο, σε κάθε νέο γύρο επέκτασης, θέτει ως στόχο τον "κατάλληλο" αριθμό ώστε η μέση συχνότητα επέκτασης να είναι η επιθυμητή. Θα πρέπει να σημειωθεί το hashing είναι μία ενεργειακά δαπανηρή διαδικασία, καθώς οι επικυρωτές χρησιμοποιούν υπολογιστές γενικού ή/και ειδικού σκοπού για συμμετέχουν στο δίκτυο (και να έχουν την ευκαιρία να επικυρώσουν συναλλαγές), οι οποίοι καταναλώνουν ηλεκτρική ενέργεια για να λειτουργήσουν. Τα δίκτυα που χρησιμοποιούν την μέθοδό hashing για την επέκτασή τους (όπως το Bitcoin), χαρακτηρίζονται ως Proof of Work και θεωρούν ως μέτρο αφοσίωσης την κατανάλωση ενεργείας. Όσο πιο πολύ ενέργεια καταναλώνει ένας επικυρωτής, τόσο πιο έμπιστος θεωρείται.

3.2.2 Ethereum

Το Ethereum [18] αποτελεί ένα blockchain δίκτυο, το οποίο ασφαλίζεται και υποστηρίζεται από τους κόμβους του μέσω του μηχανισμού συναίνεσης Proof of Stake. Σε αντίθεση με το δίκτυο Bitcoin, το οποίο υποστηρίζει ένα κατανεμημένο δημόσιο κατάλογο (shared public ledger), το Ethereum συντηρεί ένα εικονικό μηχάνημα κατάστασης, το Ethereum Virtual Machine (EVM). Το εικονικό αυτό μηχάνημα εκτός από τη δυνατότητα πραγματοποίησης συναλλαγών (μέσω μεταφοράς του ψηφιακού νομίσματος ETH), επιτρέπει στους χρήστες να καταχωρίσουν κώδικα σε αυτό και να αλλάξουν (μη αναστρέψιμα) την κατάστασή του αλλά και τις ίδιες του τις λειτουργίες.

Τα "κομμάτια" κώδικα στο EVM, χαρακτηρίζονται ως έξυπνα συμβόλαια (smart contracts) και αποτελούν το βασικό "εργαλείο" για την υπηρεσία που αναπτύχθηκε στα πλαίσια της διπλωματικής αυτής εργασίας.

Για να γίνει κατανοητό τι ακριβώς είναι το εικονικό μηχάνημα του Ethereum, ας δούμε αρχικά τι είναι εικονικό μηχάνημα ως έννοια. Τα εικονικά μηχανήματα [19] αποτελούν αυθαίρετους ψηφιακούς υπολογιστές (υπό τη μορφή λογισμικού), οι πόροι των οποίων (π.χ. αριθμός επεξεργαστών, χωρητικότητα μνήμης τυχαίας προσπέλασης) καθορίζονται από έναν υπερεπόπτη (hypervisor) και όχι από το ίδιο το υλικό (hardware). Ο υπερεπόπτης ενεργώντας ως ενδιάμεσος μεταξύ εικονικού και πραγματικού μηχανήματος, αρχικά συλλέγει και δεσμεύει τους πόρους του υλικού, ώστε στη συνέχεια να τους διαθέσει στα εικονικά μηχανήματα, τα οποία "τρέχουν" σε αυτό. Με άλλα λόγια, εικονικό μηχάνημα είναι ένα λογισμικό που εκτελείται σε ένα φυσικό μηχάνημα, έχει αντίστοιχες δυνατότητες και ιδιότητες με ένα φυσικό μηχάνημα, με την διαφορά ότι αντί να έχει πρόσβαση στο σύνολο των πόρων του υλικού υπολογιστή που το φιλοξενεί (host), έχει πρόσβαση σε ένα μέρος αυτών. Το Ethereum Virtual Machine [20] λοιπόν, είναι ένας κατανεμημένος εικονικός υπολογιστής ειδικού σκοπού, ο οποίος "τρέχει" στους υπολογιστές των επικυρωτών του δικτύου Ethereum, έχει τη δική αρχιτεκτονική, το δικό του ρεπερτόριο εντολών και κατ' επέκταση τη δική του assembly.

Το δίκτυο Ethereum με το εικονικό του μηχάνημα, όπως αναφέραμε και παραπάνω εκτός από την υποστήριξη συναλλαγών ψηφιακού χρήματος, υποστηρίζει την εκτέλεση Smart Contracts (Έξυπνων Συμβολαίων) [21] τα οποία είναι διατυπωμένα υπό την μορφή κώδικα. Για την δημιουργία των συμβολαίων, καθώς η συγγραφή σε assembly είναι δύσκολη και τις περισσότερες φορές απαγορευτική, έχουν αναπτυχθεί γλώσσες προγραμματισμού υψηλού επιπέδου (π.χ. Solidity, Vyper) [22], οι οποίες κάνουν την σύνταξη ευκολότερη. Ο χαρακτηρισμός «συμβόλαιο» πηγάζει από το γεγονός ότι, τα κομμάτια κώδικα που καταχωρούνται στο EVM είναι μη τροποποιήσιμα, όπως ακριβώς συμβαίνει και με τα νομικής φύσεως συμβόλαια.

Σχετικά με τον μηχανισμό συναίνεσης του δικτύου, το Ethereum αποτελεί ένα Proof of Stake [23] δίκτυο στο οποίο η αφοσίωση ενός επικυρωτή προσμετράτε ανάλογα με την ποσότητα ETH που έχει καταθέσει ως ενέχυρο (stake) σε αυτό. Σε αντίθεση με τα PoW δίκτυα όπου οι επικυρωτές ανταγωνίζονται μεταξύ τους για το ποιος θα καταχωρίσει πρώτος το νέο block στην αλυσίδα, στα PoS ο επικυρωτής υποδεικνύεται από το ίδιο το δίκτυο. Συγκεκριμένα, σε κάθε γύρο επέκτασης το δίκτυο επιλέγει μια επιτροπή, όπου το ένα της μέλος (το μέλος που ενεργεί ως επικυρωτής) δημιουργεί το νέο block της αλυσίδας, ενώ τα υπόλοιπα μέλη επικυρώνουν την ορθότητά του. Αν το block χαρακτηριστεί "ορθό" από την πλειοψηφία των μελών, τότε το νέο block προστίθεται στην αλυσίδα, ο επικυρωτής αμείβεται (σε ETH, για το δίκτυο Ethereum) και ξεκινάει ο επόμενος κύκλος επέκτασης. Αν όχι, τότε το μέλος που δημιούργησε το (μη έγκυρο) block τιμωρείται από το δίκτυο, συνήθως παρακρατώντας μία ποσότητα ψηφιακού νομίσματος από το προσωπικό του ενέχυρο. Γενικά, επειδή ο επικυρωτής ορίζεται από το ίδιο το δίκτυο, τα PoS δίκτυα δίνουν την δυνατότητα "μεγάλου" ρυθμού επέκτασης της αλυσίδας (1 block κάθε 12 δευτερόλεπτα για το δίκτυο Ethereum [24]), διατηρώντας την αξιοπιστία τους.

3.2.3 Polygon

Παρόλο που ο ρυθμός επέκτασης της αλυσίδας του Ethereum (PoS) επιτρέπει την εξυπηρέτηση περισσότερων συναλλαγών ανά μονάδα χρόνου σε σύγκριση με το δίκτυο Bitcoin (PoW), παρουσιάζεται συχνά το φαινόμενο όπου το κόστος πραγματοποίησης μιας συναλλαγής (gas fee) [25] είναι δυσανάλογο της υποκειμενικής ή αντικειμενικής αξίας της ίδιας της συναλλαγής. Αυτό συμβαίνει διότι συνήθως οι εντολές μεταβολής της κατάστασης το EVM σε κάθε γύρο επέκτασης, είναι περισσότερες από αυτές που μπορούν να υποστηριχθούν από το δίκτυο, δημιουργώντας έτσι συνωστισμό (network congestion). Το δίκτυο, για να μπορέσει να διαχειριστεί τον μεγάλο αριθμό (σε εκκρεμότητα) εντολών, αυξάνει το κόστος συναλλαγής (gas price), θεωρώντας ότι όσο πιο σημαντική είναι η εντολή τόσο πιο πολλά θα είναι διατεθειμένος να πληρώσει ο εντολέας της. Με ποιο απλά λόγια, είναι σαν να "λέει" το δίκτυο «Έχω ΧΧ θέσεις σε αυτό το block. Ποιος πληρώνει τα περισσότερα ώστε να συμπεριλάβω την εντολή του σε αυτό;». Το Polygon [26] λοιπόν, αποτελεί ένα EVM blockchain δίκτυο (το οποίο στην τωρινή του μορφή αποτελείται από επιμέρους δίκτυα), με σκοπό να αντιμετωπίσει τα ζητήματα που παρουσιάζονται στο Ethereum, όπως το φαινόμενο του συνωστισμού και κατά επέκταση το ζήτημα του υψηλού κόστους συναλλαγών.

Κατά τον σχεδιασμό ενός blockchain δικτύου λαμβάνονται υπόψη (υποχρεωτικά) οι έννοιες «ασφάλεια – security», «αποκεντροποίηση – decentralization», «επεκτασιμότητα – scalability», καθώς όλα τα δίκτυα θα πρέπει να:

1. Είναι ασφαλή από επιθέσεις τις οποιασδήποτε μορφής.
2. Μη ελέγχονται από κάποια αρχή ή ομάδα ανθρώπων.
3. Μην επηρεάζεται, η λειτουργία του δικτύου ή/και η εμπειρία των χρηστών, από παράγοντες όπως αυξημένη χρήση (συνωστισμός).

Δυστυχώς, οι τρεις παραπάνω έννοιες - στοιχεία είναι άρρηκτα "συνδεδεμένες" μεταξύ τους, με τρόπο τέτοιο ώστε η βελτιστοποίηση των δύο να επιφέρει τον υποβιβασμό του τρίτου. Το φαινόμενο αυτό αναφέρεται βιβλιογραφικά ως Blockchain Trilemma (το τρίλημμα του blockchain) [27].

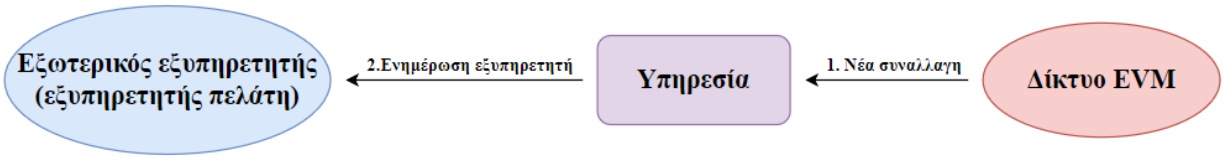
Πρακτικά και χωρίς να αναφερθούμε σε τεχνικές λεπτομέρειες, το Polygon έχοντας λίγους (σε αριθμό) κόμβους και θεωρώντας ότι οι του κόμβοι είναι αμερόληπτοι και ικανοί να εξασφαλίσουν την ασφάλεια του δικτύου, επιτυγχάνει υψηλό ρυθμό παραγωγής block (1 block κάθε 2 περίπου δευτερόλεπτα [28]), δηλαδή υψηλή επεκτασιμότητα. Αποτελεί ένα παράλληλο ή δευτέρου επιπέδου (Layer 2) [29] δίκτυο, το οποίο αρχικά ελέγχει τα block του μέσω των σχετικών διαδικασιών (συναλλαγή, επικύρωση και συγχρονισμός) και χρησιμοποιεί το δίκτυο Ethereum για να επαληθεύσει την κατάστασή του (ελέγξει την ορθότητα της κατάσταση του EVM του, άρα και το σύνολο των μεταβολών του καθώς και την ίδια την αμεροληψία των κόμβων του).

Λαμβάνοντας υπόψη τα παραπάνω, η βασική διαφορά μεταξύ των δικτύων Polygon και του Ethereum, που εντοπίζεται από το χρήστη, είναι το μειωμένο κόστος χρήσης του πρώτου έναντι του δευτέρου. Αν λοιπόν η ανάπτυξη της υπηρεσίας γίνει στο Ethereum είναι βέβαιο ότι, θα υπάρξουν φορές όπου οι χρήστες θα πρέπει να πληρώσουν ένα κόστος συναλλαγής (gas fee), συγκρίσιμο του συναλλασσόμενου ποσού. Για αυτό, επιλέγοντας το δίκτυο Polygon για την ανάπτυξη της υπηρεσίας, είμαστε θετικοί ότι ακόμα και κατά τις περιόδους συνωστισμού του δικτύου, το κόστος συναλλαγής θα τέτοιο ώστε να μην επηρεάζει την υπηρεσία (ως προς το κόστος χρήσης της).

4 ΥΠΗΡΕΣΙΑ ΕΠΙΒΕΒΑΙΩΣΗΣ

Για να μπορεί ο αναγνώστης να παρακολουθήσει την ανάπτυξη της υπηρεσίας, είναι απαραίτητο αρχικά να κατανοήσει την υπηρεσία ως διαδικασία, αλλά και τον αντικειμενικό της σκοπό.

Σε μία απλουστευμένη περιγραφή λοιπόν, η υπηρεσία αρχικά πραγματοποιεί την επιβεβαίωση των συναλλαγών που λαμβάνουν χώρα σε κάποιο EVM δικτύου (π.χ. Polygon) και στη συνέχεια προωθεί τα δεδομένα τους (των συναλλαγών) σε εξωτερικούς εξυπηρετητές, δηλαδή εξυπηρετητές που λειτουργούν ανεξάρτητα των δικτύων blockchain.



Εικόνα 1. Απλοποιημένη διαδικασία ενημέρωσης πελάτη για νέα συναλλαγή

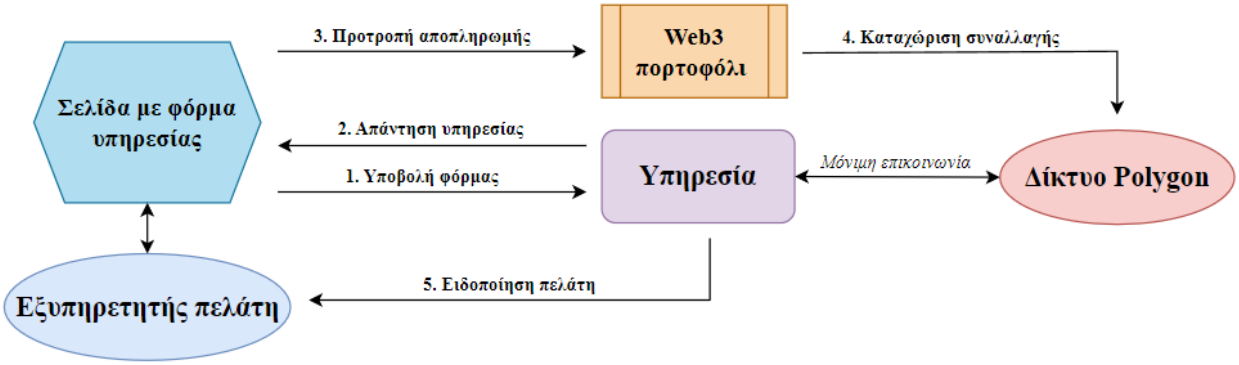
Η λέξη επιβεβαίωση χρησιμοποιείται για να περιγράψει την διαδικασία όπου μέσω των εγγραφών του δικτύου, λαμβάνεται (αδιαμφισβήτητα) γνώση σχετικά με το αν μία υπό εξέταση συναλλαγή έχει κατατεθεί στο δίκτυο ή όχι. Για να μπορεί να γίνει η επιβεβαίωση, η υπηρεσία πρέπει να διατηρεί μόνιμη σύνδεση με το δίκτυο EVM, ώστε να λαμβάνει γνώση για την κάθε (σχετική με την υπηρεσία) συναλλαγή που καταχωρείται σε αυτό.

4.1 Περιγραφή

Σε μία πιο λεπτομερή περιγραφή, η υπηρεσία που αναπτύχθηκε, επιτρέπει στους χρήστες της να διεκπεραιώνουν παραγγελίες, η πληρωμή των οποίων γίνεται μέσω του δικτύου Polygon.

Ο χρήστης (ή αλλιώς πελάτης της υπηρεσίας), έχοντας πρώτα πραγματοποιήσει τις απαραίτητες ρυθμίσεις ώστε να είναι σε θέση να λαμβάνει ειδοποιήσεις από την υπηρεσία, αρκεί να τοποθετήσει μία “ειδική” φόρμα στις σελίδες που τον ενδιαφέρει να κάνουν χρήση της υπηρεσίας. Κατά την υποβολή μιας παραγγελίας (μέσω της φόρμας) και μετά από την ανταλλαγή πληροφοριών μεταξύ της σελίδας του πελάτη και της υπηρεσίας, ο υποβολέας (τελικός χρήστης ή πελάτης του πελάτη), προτρέπεται να αποπληρώσει την παραγγελία του μέσω του Web3 πορτοφολιού του. Τα πορτοφόλια αυτού του είδους αποτελούν επεκτάσεις του περιηγητή οι οποίες, μεταξύ άλλων, δίνουν την δυνατότητα υποβολής συναλλαγών στο blockchain.

Μετά την επιτυχή καταχώριση της συναλλαγής από τον τελικό χρήστη (τον πελάτη, του πελάτη της υπηρεσίας) στο δίκτυο, χρησιμοποιώντας ένα μοναδικό αναγνωριστικό το οποίο περιλαμβάνεται τόσο στην παραγγελία όσο και στη υποβληθείσα συναλλαγή, γίνεται αντιστοίχιση συναλλαγής – παραγγελίας από την υπηρεσία και στη συνέχεια αποστέλλεται ειδοποίηση προς τον πελάτη (της υπηρεσίας), η οποία περιλαμβάνει τα δεδομένα της παραγγελίας και τα δεδομένα της συναλλαγής.



Εικόνα 2. Βήματα που ακολουθούνται για την εξυπηρέτηση μιας παραγγελίας

Έχοντας λάβει την ειδοποίηση και αξιοποιώντας τα δεδομένα της, ο πελάτης, είναι πλέον σε θέση να εκτελέσει οποιαδήποτε λειτουργία επιθυμεί, όπως για παράδειγμα τον χαρακτηρισμό της παραγγελίας ως «πληρωμένη» σε μία βάση δεδομένων, διεκπεραιώνοντας έτσι τη παραγγελία.

4.2 Μέσα ανάπτυξης

Για την ανάπτυξη της υπηρεσίας χρησιμοποιήθηκαν οι γλώσσες JavaScript, Solidity και PHP, με την συγγραφή των αρχείων να έχει γίνει στο εργαλείο ανάπτυξης κώδικα της Microsoft, Visual Studio Code, έχοντας κατεβάσει όλες τις απαραίτητες επεκτάσεις.

Η γλώσσα JavaScript χρησιμοποιήθηκε για την ανάπτυξη των αρχείων που σχετίζονται με τον εξυπηρετητή της υπηρεσίας, η εκτέλεση του οποίου γίνετε μέσω του περιβάλλοντος Node.js. Επίσης μεταξύ και άλλων βιβλιοθηκών, χρησιμοποιήθηκε η βιβλιοθήκη Web3.js, μέσω της οποίας γίνεται η επικοινωνία και αλληλεπίδραση με το EVM δίκτυο. Σχετικά με τα συμβόλαια του EVM, επειδή η καταχώρηση ενός έξυπνου συμβολαίου στο “πραγματικό” δίκτυο κοστίζει, πριν την οριστική καταχώρηση των συμβολαίων στο Polygon έγιναν δοκιμαστικές καταχωρήσεις στο δίκτυο Mumbai. Η συγγραφή των συμβολαίων έγινε αποκλειστικά σε Solidity, ενώ η γλώσσα PHP χρησιμοποιήθηκε για την ανάπτυξη του αρχείου, μέσω του οποίου γίνεται η ενημέρωση (σκανδαλισμός) των πελατών της υπηρεσίας.

4.2.1 Εργαλείο Visual Studio Code

Το Visual Studio Code είναι ένα εργαλείο επεξεργασίας κώδικα το οποίο εγγενώς υποστηρίζει τις γλώσσες JavaScript και TypeScript και το περιβάλλον Node.js, ενώ δύναται να υποστηρίξει και άλλες γλώσσες (π.χ. Solidity, PHP) μέσω των σχετικών επεκτάσεων [30].

4.2.2 Γλώσσα JavaScript

Η γλώσσα JavaScript είναι μία γλώσσα δέσμης ενεργειών (scripting language), που αποσκοπεί την στοχευμένη αλλαγή ή/και τροποποίηση των περιεχομένων μίας ιστοσελίδας, χωρίς να χρειάζεται να

γίνει η εκ νέου φόρτωσή της. Αν και η γλώσσα αυτή αναπτύχθηκε για τον παραπάνω σκοπό, μέσω του περιβάλλοντος Node.js και διατηρώντας την ίδια σύνταξη, μπορεί να χρησιμοποιηθεί για την δημιουργία εκτελέσιμων προγραμμάτων, όπως για παράδειγμα τη δημιουργία εξυπηρετητών. [31]

4.2.3 Περιβάλλον Node.js

Όπως αναφέραμε και παραπάνω, η γλώσσα JavaScript δημιουργήθηκε για να "δώσει ζωή" στις ιστοσελίδες, μέσω γραφικών και δυναμικής αλλαγής περιεχομένων. Το περιβάλλον Node.js, αποτελεί ένα (ασύγχρονο) χώρο εκτέλεσης JavaScript, το οποίο βασίζεται στη δημιουργία και παρακολούθηση γεγονότων και επιτρέπει την δημιουργία υπηρεσιών εκτός του περιβάλλοντος χρήστη (back-end) [32]. Η συγγραφή στο περιβάλλον Node.js διέπετε από τη λογική "Όταν συμβεί αυτό το γεγονός, τότε κάνε αυτές τις διαδικασίες και αν δεν συμβαίνει τίποτα, τότε απλά περίμενε".

4.2.4 Βιβλιοθήκη Web3.js

Η Web3.js, αποτελεί μια συλλογή από βιβλιοθήκες, όπου στο σύνολό τους δίνουν την δυνατότητα επικοινωνίας και αλληλεπίδρασης με τα EVM δίκτυα. Η σύνδεση με το δίκτυο ή πιο συγκεκριμένα με έναν κόμβο του δικτύου (node), μπορεί να πραγματοποιηθεί με χρήστη των πρωτοκόλλων HTTP, IPC ή μέσω WebSocket. Όσο αφορά την διαδικασία επικοινωνίας με τον κόμβο του δικτύου από την άποψη του προγραμματισμού, η βιβλιοθήκη περιλαμβάνει ένα περιβάλλον προκαθορισμένων εντολών (API), μέσω των οποίων γίνονται οι επιθυμητές ενέργειες. [33]

4.2.5 Γλώσσα Solidity

Η Solidity είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού, υψηλού επιπέδου, που χρησιμοποιείται για την συγγραφή έξυπνων συμβολαίων. Τα έξυπνα συμβόλαια αποτελούν κομμάτια κώδικα, τα οποία μεταβάλλουν την κατάσταση του EVM στα οποία έχουν κατατεθεί και είναι αμετάβλητα (immutable) ως προς το περιεχόμενό τους. [34]

4.2.6 Δίκτυο Mumbai

Το δίκτυο Mumbai αποτελεί ένα αντίγραφο του δικτύου Polygon, στο οποίο οι προγραμματιστές μπορούν να καταχωρίσουν έξυπνα συμβόλαια και μέσω αυτού να αναπτύξουν εφαρμογές, χωρίς κάποιο κόστος. Κατά κανόνα, πριν την καταχώρηση ενός έξυπνου συμβολαίου σε ένα "πραγματικό" δίκτυο (π.χ. στο Polygon), προηγούνται πολλαπλές καταχωρίσεις σε ένα δοκιμαστικό (π.χ. στο δίκτυο Mumbai). Όταν ο προγραμματιστής γίνει σίγουρος ότι το έξυπνο συμβόλαιό του λειτουργεί με τον αναμενόμενο τρόπο, μέσω των δοκιμών που έγιναν στο δοκιμαστικό δίκτυο, τότε προβαίνει σε μία εκ νέου καταχώριση στο "πραγματικό" δίκτυο, μειώνοντας έτσι το κόστος ανάπτυξης. [35]

4.2.7 Γλώσσα PHP

Η γλώσσα PHP (Personal Home Page), είναι μία γλώσσα δέσμης ενεργειών (scripting language) γενικού σκοπού, η οποία συνήθως χρησιμοποιείται για την δημιουργία ιστοσελίδων, μέσω εξυπηρετητών διαδικτύου. [36]

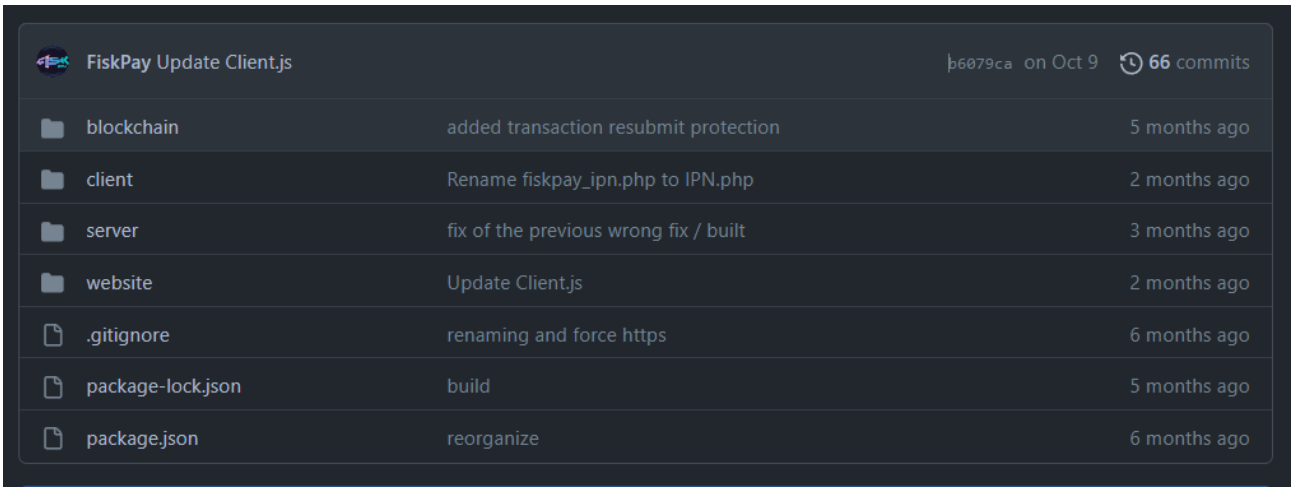
5 ΥΛΟΠΟΙΗΣΗ

Ερχόμενοι στην ενότητα της υλοποίησης και έχοντας κατανοήσει τα περιεχόμενα της προηγούμενης ενότητας, είμαστε έτοιμοι να αναφερθούμε στα αρχεία της υπηρεσίας.

Σύμφωνα με τον σκοπό που εξυπηρετούν, το σύνολο των αρχείων οργανώνονται σε τρεις (3εις) ομάδες με:

1. την πρώτη ομάδα να αφορά τα κομμάτια κώδικα τα οποία βρίσκονται (έχουν κατατεθεί) στο δίκτυο Polygon, δηλαδή τα έξυπνα συμβόλαια,
2. την δεύτερη ομάδα να αφορά τα αρχεία που χρειάζεται να ενσωματώσει ο χρήστης της υπηρεσίας (π.χ. ο κάτοχος ενός ηλεκτρονικού καταστήματος), για να μπορέσει να συνδεθεί με αυτή
3. και την τρίτη ομάδα να αφορά τα αρχεία, τα οποία εκτελούνται από τον εξυπηρετητή (server) της υπηρεσίας.

Όπως θα διαπιστωθεί και στη συνέχεια από τον αναγνώστη, η κατασκευή της υπηρεσίας αφορά την συγγραφή πολλών γραμμών κώδικα και δημιουργία αρκετών αρχείων, πράγμα που καθιστά την γραμμή προς γραμμή παρουσίαση μη πρακτική. Για τον λόγο αυτό, το σύνολο των αρχείων της υπηρεσίας είναι διαθέσιμα προς ανάγνωση μέσω του ιστότοπου Github.com . Επίσης κατά την μελέτη του εκάστοτε αρχείου, θα υπάρχει ο σχετικός υπερσύνδεσμος προς το αρχείο αυτό με σκοπό να μπορεί να γίνει παραπομπή από την περιγραφή, στον ίδιο τον κώδικα.



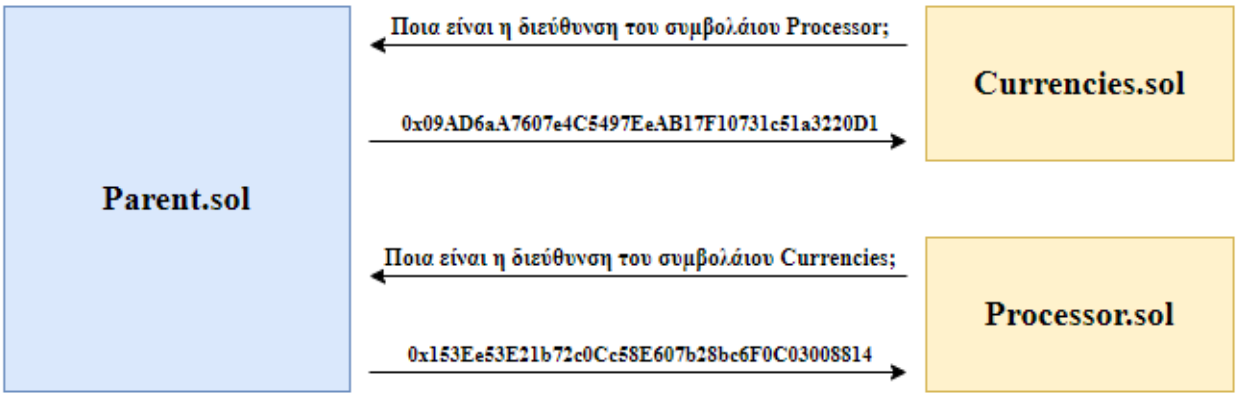
Εικόνα 3. Κεντρικός φάκελος υπηρεσίας επιβεβαίωσης, στο Github

5.1 Έξυπνα συμβόλαια

Ξεκινώντας την περιγραφή του κώδικα που καταχωρήθηκε στο Polygon, υπενθυμίζεται ότι στόχος της υπηρεσίας είναι η επιβεβαίωση των συναλλαγών, δηλαδή η εξακρίβωση ότι μία συναλλαγή έχει καταχωρηθεί επιτυχώς στο δίκτυο. Για να μπορέσει λοιπόν να γίνει η επιβεβαίωση, θα πρέπει να αναπτυχθεί ένα έξυπνο συμβόλαιο, το οποίο να λειτουργεί ως ενδιάμεσος μεταξύ των συναλλασσομένων και να δημιουργεί ένα γεγονός (event) για κάθε νέα συναλλαγή, το οποίο γεγονός και θα παρακολουθούμε.

Παρόλο που η δημιουργία ενός έξυπνου συμβολαίου είναι απλή σαν διαδικασία, θα πρέπει να λάβουμε υπόψη ότι μετά την κατάθεσή του στο δίκτυο, το περιεχόμενό του δεν μπορεί να μεταβληθεί. Με άλλα λόγια, ο κώδικας ενός καταχωρημένου έξυπνου συμβολαίου είναι αμετάβλητος (immutable). Στην περίπτωση που χρειάζεται να γίνει οποιαδήποτε αλλαγή, ο προγραμματιστής πρέπει να καταχωρήσει το νέο - τροποποιημένο έξυπνο συμβόλαιο στο δίκτυο και έχοντας τη διεύθυνση του νέου έξυπνου συμβολαίου, να προβεί στην αντικατάσταση της παλιάς διεύθυνσης με την καινούρια, σε όλα τα αρχεία ή/και έξυπνα συμβόλαια. Αν τώρα, υπάρχει κάποιο άλλο έξυπνο συμβόλαιο στο δίκτυο το οποίο χρησιμοποιεί την διεύθυνση του παλιού, θα πρέπει να γίνει αντικατάσταση και αυτού του έξυπνου συμβολαίου, όπου εάν η διεύθυνσή του χρησιμοποιείται από ένα τρίτο έξυπνο συμβόλαιο, θα πρέπει να γίνει αντικατάσταση και του τρίτου και ούτω καθ’ εξής.

Στη δική μας περίπτωση, θεωρώντας βέβαιο ότι κάποια στιγμή θα χρειαστεί να τροποποιήσουμε τον κώδικα κάποιου έξυπνου συμβολαίου, έπρεπε να βρεθεί μία λύση ώστε να μην χρειάζεται να γίνει αντικατάσταση της διεύθυνσης με τον τρόπο που περιγράφηκε. Η λύση που βρέθηκε, περιλαμβάνει ένα έξυπνο συμβόλαιο το οποίο λειτουργεί ως τηλεφωνικός κατάλογος. Σε έναν τηλεφωνικό κατάλογο αντιστοιχίζονται ονόματα με τους αντίστοιχους (τηλεφωνικούς) αριθμούς. Κατά όμοιο τρόπο, δημιουργήθηκε έξυπνο συμβόλαιο στο οποίο καταγράφονται τα ονόματα των έξυπνων συμβολαίων που χρησιμοποιούνται από την υπηρεσία μαζί με τις διευθύνσεις τους. Διατηρώντας το περιεχόμενο του έξυπνου συμβολαίου ενημερωμένο (δηλαδή τις αντιστοιχίσεις ονόματος – διεύθυνση), το μόνο που χρειάζεται είναι να “ζητήσουμε” από αυτό, την διεύθυνση του έξυπνου συμβολαίου που μας ενδιαφέρει, ονομαστικά. Το έξυπνο συμβόλαιο αυτό ονομάζεται **Parent.sol** και αποτελεί το βασικό έξυπνο συμβόλαιο της υπηρεσίας, αφού όλα τα έξυπνα συμβόλαια και τα αρχεία της υπηρεσίας, λαμβάνουν τις απαραίτητες για τη λειτουργία τους διευθύνσεις, μέσω αυτού.



Εικόνα 4. Εύρεση διεύθυνσης έξυπνου συμβολαίου, μέσω του έξυπνου συμβολαίου Parent

Μέχρι τώρα έχουμε αναφερθεί στο εγγενή ψηφιακό νόμισμα ή κρυπτονόμισμα MATIC, το οποίο χρησιμοποιείται ως μέσο συναλλαγής στο δίκτυο του Polygon. Στην πραγματικότητα όμως το MATIC, παρόλο που είναι το μοναδικό νόμισμα του δικτύου, δεν είναι το μοναδικό μέσο συναλλαγής. Τα EVM δίκτυα (όπως το Ethereum και το Polygon), δίνουν την δυνατότητα στους χρήστες να δημιουργήσουν μάρκες (tokens) [37] μέσω έξυπνων συμβολαίων, τα οποία ακολουθούν ένα συγκεκριμένο πρότυπο σύνταξης (ERC-20) [38] και χρησιμοποιούνται ως μέσο συναλλαγής το οποίο εξυπηρετεί μία συγκεκριμένη λειτουργία ή υπηρεσία. Για παράδειγμα οι μάρκες USDT (Tether) και USDC (USD Coin), έχουν ως λειτουργία την 1 προς 1 αντιστοίχιση με το δολάριο [39]. Δηλαδή ισχύει ότι 1 USDT ισούται με 1 USDC, το οποίο ισούται με 1 USD (δολάριο), επιτρέποντας έτσι την "μετατροπή" παραστατικού υλικού ή άυλου χρήματος, σε ψηφιακό ίσης αξίας. Έχοντας κατά νου ότι, η υπηρεσία θα πρέπει να επιβεβαιώνει την μεταφορά νομισμάτων αλλά και μαρκών, δημιουργήθηκε το έξυπνο συμβόλαιο **Currencies.sol**, στο οποίο περιλαμβάνονται όλες η μάρκες που υποστηρίζονται από την υπηρεσία. Παρόλο που θα ήταν εφικτό η υπηρεσία να επιβεβαιώνει όλες τις συναλλαγές, εξυπηρετώντας αποκλειστικά τις εγγεγραμμένες στο έξυπνο συμβόλαιο μάρκες (και το νόμισμα MATIC), μπορεί να διασφαλιστεί ότι οι συναλλαγές αφορούν μάρκες με ουσιαστικό λόγο ύπαρξης (δηλαδή μάρκες που εξυπηρετούν κάποια λειτουργία ή υπηρεσία).

Φτάνοντας στο έξυπνο συμβόλαιο που αναφέραμε στην αρχή αυτού το κεφαλαίου, το έξυπνο συμβόλαιο **Processor.sol** αποτελεί τον ενδιάμεσο (middleman) των συναλλαγών και είναι το πιο "απαιτητικό" ως προς την σύνταξη. Περιέχει διεπαφές (interfaces) μέσω των οποίων χρησιμοποιεί συναρτήσεις (functions) από άλλα συμβόλαια (π.χ. τα συμβόλαια Parent.sol και Currencies.sol), εσωτερικές συναρτήσεις που προωθούν τα νομίσματα ή τις μάρκες από το έξυπνο συμβόλαιο στον τελικό παραλήπτη, εκτελεί διαφόρους ελέγχους ασφαλείας και εκπέμπει γεγονότα (events) σε κάθε επιτυχή συναλλαγή.

Στη συνέχεια, έχοντας αναφέρει τα συμβόλαια της υπηρεσίας (που αφορούν την εργασία αυτή) και έχοντας περιγράψει την λειτουργία που εξυπηρετεί το καθένα, μπορούμε να προχωρήσουμε με το περιεχόμενο του κάθε έξυπνου συμβολαίου κάνοντας τα απαραίτητα σχόλια και επεξηγήσεις.

5.1.1 Parent.sol

Σύνδεσμος GitHub:
https://github.com/FiskPay/service/blob/dissertation/blockchain/Parent.sol

Σύνδεσμος στο PolygonScan:
https://polygonscan.com/address/0x163342FAe2bBe3303e5A9ADCe4BC9fb44d0FF062

Χρησιμοποιώντας το σύνδεσμο που παραπέμπει στο GitHub ως αναφορά, η σύνταξη του κώδικα ξεκινάει ορίζοντας την άδεια (license) (γραμμή 1) σύμφωνα με την οποία συγγράφεται το έξυπνο συμβόλαιο. Αμέσως μετά, δίνεται εντολή/οδηγία (γραμμή 3) στον μεταγλωττιστή (compiler) υποδεικνύοντας την έκδοση με την οποία θα γίνει η μεταγλώττιση (compile).

Πριν ξεκινήσει η σύνταξη του κώδικα χρειάζεται να δηλωθούν οι διεπαφές (interfaces) που χρησιμοποιούνται από το έξυπνο συμβόλαιο. Το συγκεκριμένο έξυπνο συμβόλαιο χρησιμοποιεί μόνο μία διεπαφή (interface), τη IGlobal (γραμμές 5 έως 8), όπου σε αυτή δηλώνεται η συνάρτηση (function) Name (των συμβολαίων Currencies και Processor). Η συνάρτηση Name επιστρέφει το όνομα του έξυπνου συμβολαίου που την εμπεριέχει, δηλαδή επιστέφει κείμενο (string). Θα πρέπει να τονιστεί ότι για να κληθεί μία απομακρυσμένη συνάρτηση, το έξυπνο συμβόλαιο που την καλεί, θα πρέπει να την καλέσει χρησιμοποιώντας την διεύθυνση του απομακρυσμένου έξυπνου συμβολαίου (γραμμή 114), εισάγοντας πάντα τις σωστές παραμέτρους εισόδου. Στην περίπτωσή μας η συνάρτηση Name, δεν έχει παραμέτρους εισόδου. Γενικά, αν γίνει προσπάθεια κλήσης μίας συνάρτησης από ένα απομακρυσμένο έξυπνο συμβόλαιο το οποίο δεν περιέχει την συνάρτηση αυτή ή οι μεταβλητές εισόδου που δίνονται (κατά την κλήση της συνάρτησης) είναι λανθασμένες, τότε η εκτέλεση επιφέρει σφάλμα και κατά επέκταση δεν τροποποιεί (εάν επρόκειτο να τροποποιήσει κατά την επιτυχή της εκτέλεση) το EVM.

Το έξυπνο συμβόλαιο λοιπόν ξεκινάει (γραμμή 10), δημιουργώντας κάποια γεγονότα (events) (γραμμές 14 έως 18), όπου κατά την επιτυχή εκτέλεση των συναρτήσεων που τα συμπεριλαμβάνουν, εκπέμπονται στο δίκτυο με σκοπό να ληφθούν από κάποια εφαρμογή εκτός blockchain. Αν και για το συγκεκριμένο έξυπνο συμβόλαιο δεν γίνεται κάποια παρακολούθηση των γεγονότων, είναι καλή τακτική να δημιουργούνται γεγονότα για πιθανή μελλοντική χρήση. Σημαντικό είναι να αναφέρουμε ότι στο έξυπνο συμβόλαιο Processor, παρακολουθούμε το γεγονός που εκπέμπεται κατά την διεκπεραίωση μίας συναλλαγής, το οποίο λαμβάνεται από τον εξυπηρετητή που “τρέχει” το αρχείο BackServer.js και χρησιμοποιείται για την επιβεβαίωσή της εκτός του blockchain. Περισσότερες επεξηγήσεις στη συνέχεια.

Συνεχίζοντας με το έξυπνο συμβόλαιο Parent, γίνονται κάποιες δηλώσεις μεταβλητών (γραμμές 25 και 26), σταθερών (γραμμές 28 και 29) ή καταστάσεων (states) όπως αναφέρονται στο blockchain. Κοιτώντας την σύνταξη των καταστάσεων, διαπιστώνεται ότι οι δύο πρώτες έχουν τον χαρακτηρισμό «private», ενώ οι δύο επόμενες έχουν τον χαρακτηρισμό «public». Ως private (ιδιωτικές) χαρακτηρίζονται οι μεταβλητές, οι σταθερές και οι συναρτήσεις οι οποίες καλούνται μόνο από το ίδιο το έξυπνο συμβόλαιο και δεν είναι προσβάσιμες από κάποιον τρίτο (είτε είναι έξυπνο συμβόλαιο, είτε είναι χρήστης). Αντιθέτως ως public (δημόσιες) χαρακτηρίζονται οι μεταβλητές, οι σταθερές και οι συναρτήσεις, οι οποίες δύναται να εκτελεστούν από όλους. Ο χαρακτηρισμός του δεύτερου ζεύγους μεταβλητών (γραμμές 28 και 29) ως «constant», δηλώνει ότι

η τιμή τους δεν γίνεται να μεταβληθεί/τροποποιηθεί, δηλαδή είναι σταθερή. Οι μεταβλητές owner και newOwner (γραμμές 25 και 26) κατά την καταχώριση του έξυπνου συμβολαίου στο δίκτυο, λαμβάνουν ως τιμή την διεύθυνση Ethereum που καταχωρεί το έξυπνο συμβόλαιο.

Προχωρώντας και εξετάζοντας τις επόμενες δύο (2) μεταβλητές (γραμμές 43 και 44), διαπιστώνεται ότι αφορούν μεταβλητές αντιστοίχισης (mapping). Οι μεταβλητές αυτές, πρακτικά συσχετίζουν δύο (2) καταστάσεις μεταξύ τους και στη συγκεκριμένη περίπτωση, η μεταβλητή nameToAddress (γραμμή 43) συσχετίζει ένα κείμενο (string) με μία διεύθυνση (address) Ethereum, με την μεταβλητή addressToName (γραμμή 44) να κάνει την "ανάποδη" αντιστοίχιση. Όπως έχουμε ήδη πει, το έξυπνο συμβόλαιο Parent λειτουργεί με τρόπο αντίστοιχο ενός τηλεφωνικού καταλόγου, δηλαδή "συνδέει" ονόματα με διευθύνσεις. Τα mappings, μπορούν να αντιμετωπιστούν ως λίστες με όλους τους πιθανούς συνδυασμούς της πρώτης μεταβλητής, οι οποίες αντιστοιχίζονται με την αρχική - προκαθορισμένη (default) τιμή της δεύτερης. Για παράδειγμα η μεταβλητή nameToAddress, λέμε ότι είναι μία λίστα που περιέχει όλους τους πιθανούς συνδυασμούς χαρακτήρων (και μηκών), όπου για τον κάθε συνδυασμό αποδίδεται η τιμή της μηδενικής διεύθυνσης Ethereum (0x0000000000000000000000000000000000000000).

Στις επόμενες 8 γραμμές (γραμμές 48 έως 55), γίνεται δήλωση ενός τροποποιητή (modifier). Οι τροποποιητές αποτελούν κομμάτια κώδικα τα οποία εκτελούνται κατά την αρχή ή το τέλος μίας συνάρτησης και αποσκοπούν την μείωση επαναλαμβανόμενου κώδικα στο έξυπνο συμβόλαιο. Ο τροποποιητής ownerOnly περιλαμβάνει κώδικα ο οποίος ελέγχει αν η διεύθυνση που κάνει χρήση μίας συνάρτησης (η οποία συνάρτηση χρησιμοποιεί τον τροποποιητή, βλέπε γραμμή 80), είναι ίση με την μεταβλητή owner. Υπενθυμίζεται ότι η μεταβλητή owner, (αρχικά) αποθηκεύει την διεύθυνση Ethereum που δημιουργεί το έξυπνο συμβόλαιο, δηλαδή τον ιδιοκτήτη του έξυπνου συμβολαίου. Το σύμβολο «_» (γραμμή 54) δηλώνει ότι η συνάρτηση που χρησιμοποιεί τον τροποποιητή, εκτελείται μετά τους ελέγχους του τροποποιητή.

Κοιτώντας τις επόμενες τρεις (3εις) συναρτήσεις (γραμμές 63 έως 76) αρχικά βλέπουμε ότι κατά την δήλωσή τους, περιλαμβάνουν τον χαρακτηρισμό «view». Ο χαρακτηρισμός αυτός σε μία συνάρτηση δηλώνει ότι οι συνάρτηση δεν τροποποιεί την κατάσταση του EVM, παρά μόνο διαβάζει από αυτό. Έτσι η συνάρτηση Owner (γραμμή 63) επιστέφει την διεύθυνση Ethereum του ιδιοκτήτη του έξυπνου συμβολαίου, η GetContractAddress (γραμμή 68) εισάγοντας ένα κείμενο επιστρέφει την σχετιζόμενη με αυτό διεύθυνση και η GetContractName (γραμμή 73) εισάγοντας μία διεύθυνση Ethereum, επιστρέφει το όνομα του συμβολαίου στο οποίο αντιστοιχεί η διεύθυνση. Αν δεν αντιστοιχεί κάποιο όνομα, τότε επιστέφει ένα κείμενο μηδενικού μήκους.

Τελειώνοντας με την περιγραφή του έξυπνου συμβολαίου Parent, οι συναρτήσεις ChangeOwnership (γραμμές 80 έως 85) και AcceptOwnership (γραμμές 87 έως 99), αφορούν την αλλαγή της ιδιοκτησίας του έξυπνου συμβολαίου, ενώ οι επόμενες τρεις (3εις) συναρτήσεις αφορούν την προσθήκη, την επεξεργασία και την διαγραφή των εγγραφών στον κατάλογο. Συγκεκριμένα, η AddContract (γραμμές 101 έως 124) κάνοντας τους απαραίτητους ελέγχους προσθέτει ένα νέο έξυπνο συμβόλαιο (το όνομα του έξυπνου συμβολαίου) και την αντίστοιχη διεύθυνση στον κατάλογο, η UpdateContract (γραμμές 126 έως 151), εφόσον υπάρχει το όνομα που εισάγεται σε αυτή (ως παράμετρος) και μετά από τους σχετικούς ελέγχους αντικαθιστά την παλιά διεύθυνση με την διεύθυνση που επίσης εισάγεται (ως παράμετρος) και τέλος η RemoveContract (γραμμές 153 έως 163) η οποία αφαιρεί τις αντιστοιχίσεις ονόματος - διεύθυνσης, εφόσον αυτές υπάρχουν.

Μελετώντας στις γραμμές 105, 116, 130 και 141 βλέπουμε ότι η κάθε γραμμή αφορά μία σύγκριση μεταξύ κειμένων. Η γλώσσα Solidity δεν υποστηρίζει την άμεση σύγκριση ακολουθίας χαρακτήρων (κειμένου). Για το λόγο αυτό, οι συγκρίσεις γίνονται αρχικά "πακετάροντας" το κάθε κείμενο μέσω της συνάρτησης abi.encodePacked και στη συνέχεια εισάγονται τα αποτελέσματα στην συνάρτηση (αλγόριθμος hashing) keccak256 η οποία επιστρέφει ένα hash. Αν οι τιμές των hash είναι ίδιες, τότε είναι και τα κείμενα. Αν όχι, τότε τα κείμενα είναι διαφορετικά (τα πεζά και τα κεφαλαία γράμματα αποδίδουν διαφορετικό αποτέλεσμα κατά το hashing).

Σχετικά με τις δύο τελευταίες συναρτήσεις (γραμμές 167 έως 176), η συνάρτηση receive (γραμμές 167 έως 171) εκτελείται όταν γίνεται αποστολή νομισμάτων στο έξυπνο συμβόλαιο χωρίς δεδομένα (data) και η fallback (γραμμές 173 έως 175) εκτελείται όταν γίνεται αλληλεπίδραση με το έξυπνο συμβόλαιο αποστέλλοντας δεδομένα, τα οποία δεν αντιστοιχούν σε κάποια από τις συναρτήσεις που περιέχονται στο έξυπνο συμβόλαιο. Οι συναρτήσεις receive και fallback, δεν χρειάζεται να χαρακτηριστούν ως «function», αφού αποτελούν εγγενή συναρτήσεις της γλώσσας Solidity. Ο χαρακτηρισμός «payable» κατά τον ορισμό της συνάρτησης receive δηλώνει ότι κατά την εκτέλεσή της, επιτρέπεται η αποστολή νομισμάτων προς το έξυπνο συμβόλαιο. Η γραμμή 177 δηλώνει το τέλος του έξυπνου συμβολαίου.

```
61  //------------------------------------------------------------------------// v GET FUNCTIONS
62
63      function Owner() public view returns(address){···
66      }
67      //
68      function GetContractAddress(string calldata _name) public view returns(address){···
71      }
72
73      function GetContractName(address _address) public view returns(string memory){···
76      }
77
78  //------------------------------------------------------------------------// v SET FUNTIONS
79
80      function ChangeOwnership(address _newOwner) public ownerOnly returns(bool){···
85      }
86
87      function AcceptOwnership() public returns(bool){···
99      }
100     //
101     function AddContract(string calldata _name, address _address) public ownerOnly returns(bool){···
124     }
125
126     function UpdateContract(string calldata _name, address _address) public ownerOnly returns(bool){···
151     }
152
153     function RemoveContract(string calldata _name) public ownerOnly returns(bool){···
163     }
```

Εικόνα 5. Συναρτήσεις έξυπνου συμβολαίου Parent

5.1.2 Currencies.sol

Σύνδεσμος GitHub:
https://github.com/FiskPay/service/blob/dissertation/blockchain/Payment/Currencies.sol

Σύνδεσμος στο PolygonScan:
https://polygonscan.com/address/0x153Ee53E21b72c0Cc58E607b28bc6F0C03008814

Διατηρώντας τη δομή του προηγούμενου έξυπνου συμβολαίου, η σύνταξη ξεκινάει με την απαραίτητη δήλωση άδειας (γραμμή 1) και στη συνέχεια την επιλογή της έκδοσης του μεταγλωττιστή (γραμμή 3). Αμέσως μετά δηλώνονται οι διεπαφές IParent (γραμμές 5 έως 10) και IERC20 (γραμμές 12 έως 15), με την πρώτη να περιλαμβάνει τις συναρτήσεις GetContractAddress, Owner και MATIC του έξυπνου συμβολαίου Parent και την δεύτερη την συνάρτηση symbol. Η symbol είναι μία συνάρτηση που υπάρχει σε όλα τα συμβόλαια μάρκας τα οποία είναι σύμφωνα με το πρότυπο ERC-20 και επιστρέφει το σύμβολο – αναγνωριστικό της. Για παράδειγμα, αν αναφερόμαστε στο έξυπνο συμβόλαιο της μάρκας USD Coin, η συνάρτηση symbol θα επιστρέψει την τιμή USDC.

Το έξυπνο συμβόλαιο Currencies (γραμμές 17 έως 161) ξεκινάει, με την σύνταξη των γεγονότων CurrencyAddition (γραμμή 21) και CurrencyRemoval (γραμμή 22). Έπειτα, δηλώνεται η σταθερή (constant) μεταβλητή pt (γραμμή 26), η οποία επιτρέπει την άμεση χρήση των λειτουργιών της διεπαφής IParent (βλέπε γραμμή 59). Αμέσως μετά δηλώνονται οι σταθερές διεύθυνσης parentAddress (γραμμή 32) και MATIC (γραμμή 34). Ο χαρακτηρισμός «immutable» σημαίνει ότι η μεταβλητή δεν είναι απαραίτητο να αρχικοποιηθεί κατά τη δήλωσή της, αλλά όταν λάβει τιμή, η τιμή της δεν γίνεται να ξανά αλλάξει. Τέλος, δηλώνονται η μεταβλητή κειμένου Name (γραμμή 42) (η οποία μπορεί να εκτελεστεί όπως μία συνάρτηση, αφού έχει τον χαρακτηρισμό «public»), ο πίνακας κειμένων currencies (γραμμή 44) και οι μεταβλητές αντιστοίχισης, symbolToAddress (γραμμή 52) και addressToSymbol (γραμμή 53). Ο τροποποιητής (modifier) ownerOnly (γραμμές 57 έως 63), εκτελεί την ίδια λειτουργία με τον αντίστοιχο του έξυπνου συμβολαίου Parent.

Κατά την δημιουργία ενός έξυπνου συμβολαίου μπορεί να χρειάζεται να αποθηκευτούν καταστάσεις (states), οι οποίες δεν είναι διαθέσιμες πριν την καταχώρηση του έξυπνου συμβολαίου, όπως για παράδειγμα τη χρονική στιγμή (timestamp) κατά την οποία καταχωρήθηκε το έξυπνο συμβόλαιο στο δίκτυο ή μία τιμή η οποία προέρχεται από κάποιο τρίτο έξυπνο συμβόλαιο. Για το λόγο αυτό, υπάρχει η εγγενής συνάρτηση constructor, η οποία εκτελείται (αποκλειστικά και μόνο μία φορά) κατά την καταχώριση του έξυπνου συμβολαίου στο δίκτυο και επιτρέπει την αποθήκευση αυτού του είδους των καταστάσεων σε μεταβλητές. Στη δική μας περίπτωση, η συνάρτηση constructor (γραμμές 67 έως 70), εκχωρεί την διεύθυνση που προκύπτει από την εκτέλεση της συνάρτησης MATIC του έξυπνου συμβολαίου Parent, στην μεταβλητή MATIC (η οποία δηλώθηκε στην γραμμή 34).

Η σύνταξη συνεχίζει με τη δήλωση των συναρτήσεων ανάγνωσης (view) GetCurrencyAddress (γραμμές 76 έως 79), GetCurrencySymbol (γραμμές 81 έως 84) και GetCurrencyList (γραμμές 86 έως 89). Η GetCurrencyAddress επιστρέφει την διεύθυνση της μάρκας (του έξυπνου συμβόλαιού της), στο οποίο αντιστοιχεί το σύμβολο που εισήχθη ως παράμετρος, ενώ η GetCurrencySymbol κάνει ακριβώς το ανάποδο, δηλαδή εισάγοντας την διεύθυνση του έξυπνου συμβολαίου της μάρκας

ως παράμετρο, επιστρέφει το σύμβολό της. Η συνάρτηση GetCurrencyList, επιστρέφει τα σύμβολα όλων των εγγεγραμμένων μαρκών (tokens) υπό την μορφή λίστας (array).

Πριν τις συναρτήσεις receive (γραμμές 151 έως 155), fallback (γραμμές 157 έως 160) και το τέλος του έξυπνου συμβολαίου (γραμμή 161), δηλώνονται οι συναρτήσεις AddCurrency (γραμμές 93 έως 120) και RemoveCurrency (γραμμές 122 έως 147), οι οποίες αφορούν την προσθήκη και την αφαίρεση μαρκών (token), αντίστοιχα. Μελετώντας την AddCurrency, αρχικά, γίνονται κάποιοι έλεγχοι ώστε να εξακριβωθεί ότι τα δεδομένα που εισάγονται (σύμβολο και διεύθυνση μάρκας), δεν είναι ήδη καταχωρημένα (γραμμές 97 έως 100) και στη συνέχεια γίνεται έλεγχος (γραμμές 102 έως 106) ώστε να επιβεβαιωθεί ότι η διεύθυνση που δόθηκε αντιστοιχεί σε έξυπνο συμβόλαιο. Έπειτα, γίνεται έλεγχος (γραμμές 108 έως 111) σύμφωνα με τον οποίο επαληθεύεται ότι η διεύθυνση του έξυπνου συμβολαίου που εισήχθη ως παράμετρος, αφορά μάρκα της οποίας το σύμβολο είναι ίδιο με το εισαγόμενο και εφόσον ισχύει αυτό, η νέα μάρκα προστίθεται ως υποστηριζόμενη (γραμμές 113 έως 116). Αναφορικά με την RemoveCurrency, στην αρχή ελέγχεται ότι το σύμβολο που εισήχθη αντιστοιχεί σε υποστηριζόμενη μάρκα (γραμμές 124 έως 127) και στην περίπτωση που ισχύει, αφαιρείται (γραμμές 129 έως 143).

```
76  >   function GetCurrencyAddress(string calldata _symbol) public view returns(address){···
79      }
80
81  >   function GetCurrencySymbol(address _addr) public view returns(string memory){···
84      }
85      //
86  >   function GetCurrencyList() public view returns(string[] memory){···
89      }
90
91  //------------------------------------------------------------------------// v SET FUNTIONS
92
93  >   function AddCurrency(string calldata _symbol, address _addr) public ownerOnly returns(bool){···
120     }
121
122 >   function RemoveCurrency(string calldata _symbol) public ownerOnly returns(bool){···
147     }
148
149 //------------------------------------------------------------------------// v DEFAULTS
150
151 >   receive() external payable{···
155     }
156
157 >   fallback() external{···
160     }
```

Εικόνα 6. Συναρτήσεις έξυπνου συμβολαίου Currencies

5.1.3 Processor.sol

Σύνδεσμος GitHub:
https://github.com/FiskPay/service/blob/dissertation/blockchain/Payment/Processor.sol

Σύνδεσμος στο PolygonScan:
https://polygonscan.com/address/0x09AD6aA7607e4C5497EeAB17F10731c51a3220D1

Όπως και στα δύο (2) παραπάνω αρχεία, στις πρώτες γραμμές διατυπώνεται η δήλωση άδειας (γραμμή 1), η εντολή προς τον μεταγλωττιστή (γραμμή 3) και μετά ακολουθεί δήλωση των διεπαφών IParent (γραμμή 5 έως 8), ICurrencies (γραμμή 10 έως 14), ISubscribers (γραμμές 16 έως 19) και IERC20 (γραμμές 21 έως 27). Η IParent (γραμμές 5 έως 8) (όπως και στο αρχείο Currencies.sol) περιέχει την συνάρτηση GetContractAddress (γραμμή 7), η οποία επιστρέφει την διεύθυνση που αντιστοιχεί στο κείμενο (δηλαδή το όνομα) το οποίο εισάγεται ως μεταβλητή είσοδού. Η ICurrencies, περιέχει τις συναρτήσεις GetCurrencyAddress (γραμμή 12) και GetCurrencySymbol (γραμμή13) του έξυπνου συμβολαίου Currencies, με την GetCurrencyAddress να επιστρέφει την διεύθυνση που αντιστοιχεί στο εισαγόμενο σε αυτή σύμβολο (κείμενο) και την GetCurrencySymbol να κάνει την ανάποδη αντιστοίχιση. Η διεπαφή ISubscribers αφορά το έξυπνο συμβόλαιο της υπηρεσίας που σχετίζεται με τους συνδρομητές, το οποίο δεν εξετάζεται στα πλαίσια της εργασίας. Παρόλα αυτά, η συνάρτηση AllowProcessing (γραμμή 18), εισάγοντας ως παραμέτρους τη διεύθυνση του παραλήπτη και το συναλλασσόμενο ποσό, επιστρέφει μια λογική τιμή (bool) από την οποία εξαρτάται αν θα επιτραπεί η συναλλαγή ή όχι. Η IERC20, περιέχει τις συναρτήσεις approve (γραμμή 23), allowance (γραμμή 24). balanceOf (γραμμή 25) και transferFrom (γραμμή 26) του πρότυπου ERC-20. Τονίζεται ότι οι συναρτήσεις approve και balanceOf, έχουν εισαχθεί από λάθος και δεν χρησιμοποιούνται. Όσο αφορά τις allowance και transferFrom, μέσω της πρώτης επιστέφεται η συνολική ποσότητα μαρκών (tokens) που μπορεί να μεταφέρει ο spender (διεύθυνση εκπρόσωπου) εκ μέρους του owner (διεύθυνση ιδιοκτήτη μαρκών), ενώ μέσω της συνάρτησης transferFrom γίνεται μεταφορά της ποσότητας value του ιδιοκτήτη (παράμετρος from), στον παραλήπτη (παράμετρος receiver), από τον χρήστη που εκτελεί την συνάρτηση, δηλαδή τον εκπρόσωπο.

Το έξυπνο συμβόλαιο λοιπόν ξεκινάει (γραμμή 29) με την δήλωση του γεγονότος Processed (γραμμή 33), το οποίο εκπέμπεται κατά την ολοκλήρωση κάθε συναλλαγής (γραμμή 145). Η παρακολούθηση του γεγονότος, δίνει την δυνατότητα να γνωρίζουμε όχι μόνο ότι πραγματοποιήθηκε μία συναλλαγή, αλλά και ποια είναι η συναλλαγή αυτή (συγκεκριμένα), μέσω των δεδομένων verification και timestamp τα οποία κοινοποιούνται από το γεγονός. Στις επόμενες γραμμές (γραμμές 37 έως 61) ακολουθούν δηλώσεις καταστάσεων και αμέσως μετά γίνεται δήλωση των ιδιωτικών (private) συναρτήσεων _transferMATIC (γραμμές 69 έως 72) και _transferToken (γραμμές 74 έως 88).

Η συνάρτηση _transferMATIC χρησιμοποιείται (αποκλειστικά) από το έξυπνο συμβόλαιο, ώστε να μεταφέρει τη ποσότητα MATIC που ορίζεται στην παράμετρο _amount, στην διεύθυνση _receiver. Αντίστοιχα, η συνάρτηση _TransferToken αφορά την αποστολή μάρκας (η οποία ορίζεται από την παράμετρο _symbol), ποσότητας _amount, προς την διεύθυνση_receiver. Συγκρίνοντας τις δύο συναρτήσεις, διαπιστώνετε ότι η δεύτερη περιέχει ορισμούς μεταβλητών (γραμμές 76, 78 και 79) και ελέγχους (γραμμές 81 και 84) όπου δεν υπάρχουν στην πρώτη. Αυτό συμβαίνει γιατί η

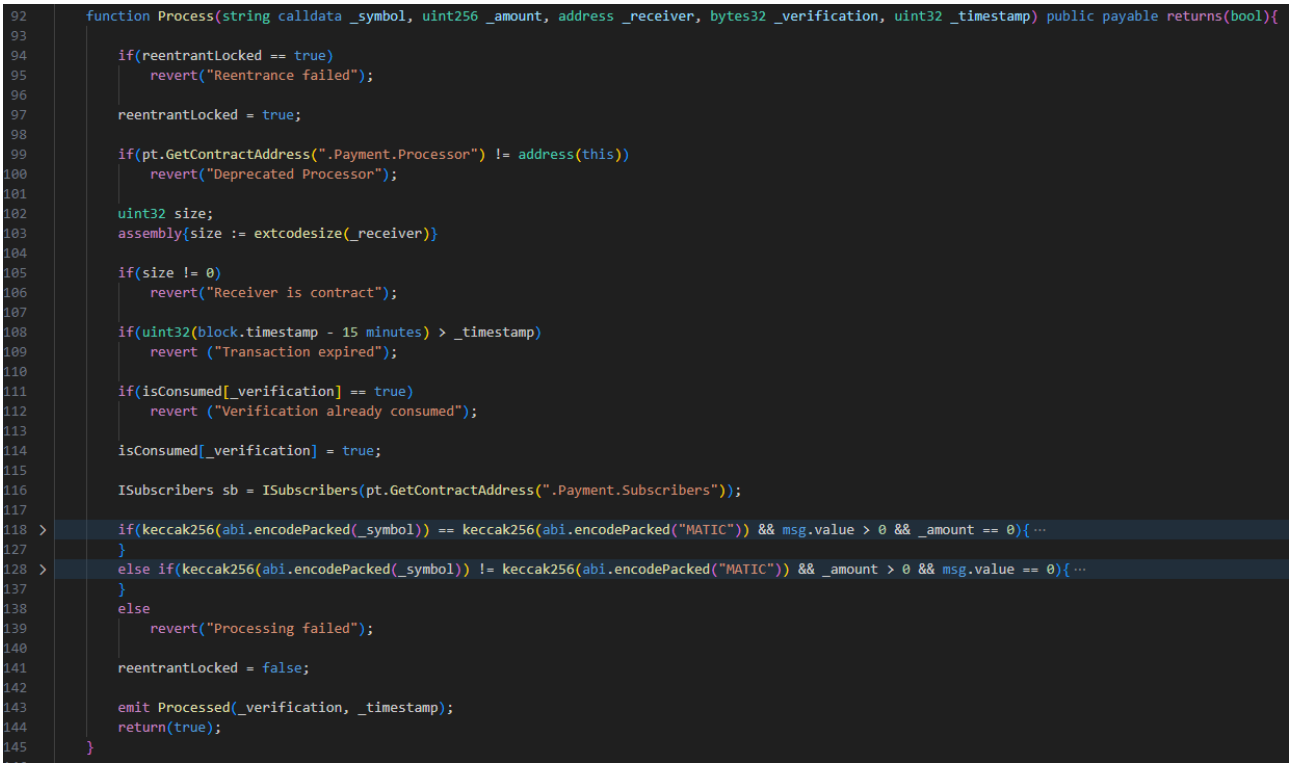
μεταφορά MATIC είναι εγγενής του EVM, ενώ η μεταφορά μάρκας γίνεται χρησιμοποιώντας το συμβόλαιό της (το έξυπνο συμβόλαιο της ίδιας της μάρκας). Περιγράφοντας λοιπόν την συνάρτηση _TransferToken, αρχικά δημιουργείται πρόσβαση στο έξυπνο συμβόλαιο Currencies (γραμμή 76) και στη συνέχεια χρησιμοποιώντας την συνάρτηση GetCurrencyAddress, λαμβάνεται (και αποθηκεύετε) η διεύθυνση που αντιστοιχεί στο έξυπνο σύμβολο της μάρκας (γραμμή 78). Θεωρώντας ότι η παραπάνω διεύθυνση αντιστοιχεί σε μάρκα (και ότι δεν είναι η μηδενική διεύθυνση), πραγματοποιείται πρόσβαση στο συμβόλαιό της (γραμμή 79) και πριν γίνει χρήση της οποιασδήποτε συνάρτησης, ελέγχεται ότι η διεύθυνση υποστηρίζεται από την υπηρεσία (γραμμή 81). Εφόσον, η διεύθυνση αντιστοιχεί σε υποστηριζόμενη μάρκα, τότε ελέγχεται αν το έξυπνο συμβόλαιο Processor έχει την άδεια (του αποστολέα) να μεταφέρει την ποσότητα _amount από τον αποστολέα (msg.sender) στην διεύθυνση _receiver (γραμμή 84) και στην περίπτωση που αυτό επιτρέπεται, γίνεται μεταφορά μέσω της συνάρτησης transferFrom του έξυπνου συμβολαίου της μάρκας (γραμμή 87).

Φτάνοντας στην συνάρτηση Process (γραμμές 94 έως 147), αρχικά θα αναφέρουμε τις παραμέτρους εισόδου, οι οποίες είναι οι:

1. _symbol: Το σύμβολο του νομίσματος (MATIC) ή μάρκας (token) που πρόκειται να μεταφερθεί.
2. _amount: Η ποσότητα μάρκας η οποία πρόκειται να μεταφερθεί. Αν η μεταφορά αφορά το νόμισμα MATIC, τότε λαμβάνει την τιμή μηδέν (0) και η ποσότητα εμπεριέχεται στην μεταβλητή msg.value, μιας και η συνάρτηση έχει τον χαρακτηρισμό «payable».
3. _receiver: Η διεύθυνση του παραλήπτη των νομισμάτων / μαρκών.
4. _verification: Το μοναδικό αναγνωριστικό της συναλλαγής (αποτέλεσμα του αλγορίθμου SHA256).
5. _timestamp: Η χρονική στιγμή (timestamp) που χρησιμοποιήθηκε για την δημιουργία του αναγνωριστικού.

Οι γραμμές 96 έως 99 και η γραμμή 143, αποσκοπούν την αποφυγή μίας συγκεκριμένης επίθεσης προς το έξυπνο συμβόλαιο και δεν θα επεξηγηθούν. Έτσι, ξεκινώντας την περιγραφή των λειτουργιών της συνάρτησης Process από τη γραμμή 101, λαμβάνοντας από το έξυπνο συμβόλαιο Parent τη διεύθυνση του έξυπνου συμβολαίου Processor και συγκρίνοντάς τη με τη δική του διεύθυνση, γίνεται έλεγχος ώστε να επιβεβαιωθεί ότι το τρέχον έξυπνο συμβόλαιο δεν έχει αντικατασταθεί (από μία νεότερη έκδοσή του). Στη συνέχεια ελέγχεται αν η διεύθυνση του παραλήπτη (_receiver) αντιστοιχεί σε έξυπνο συμβόλαιο (γραμμές 104 έως 108) και εφόσον δεν αντιστοιχεί, ελέγχεται αν η συναλλαγή έχει λήξει (γραμμή 110) καθώς και αν είναι διπλότυπη (γραμμή 113). Αν η συναλλαγή είναι μοναδική, τότε χαρακτηρίζεται ως «κατατεθειμένη» (γραμμή 116) και δημιουργείται πρόσβαση στο έξυπνο συμβόλαιο Subscribers μέσω της διεπαφής ISubscribers (γραμμή 118). Στις επόμενες γραμμές κώδικα (από 120 μέχρι και 129), αρχικά ελέγχονται, αν η συναλλαγή αφορά το νόμισμα MATIC και αν οι ποσότητες msg.value και _amount, αντίστοιχα, έχουν τιμές μεγαλύτερη και ίση του μηδενός. Εφόσον ο έλεγχος είναι αληθής (true), γίνεται έλεγχος της κατάστασης της συνδρομής του παραλήπτη (γραμμή 122) και ελέγχεται αν τα δεδομένα εισόδου της συνάρτησης Processor, μετά από επεξεργασία, επιφέρουν αποτέλεσμα, ίδιο με την παράμετρο _verification (γραμμή 125). Αν το αποτέλεσμα είναι ίδιο, γίνεται αποστολή

των νομισμάτων μέσω την συνάρτησης _transferMatic (γραμμή 128). Αντίστοιχες διαδικασίες, γίνονται για την μεταφορά των μαρκών (γραμμές 130 έως 139), ενώ μετά την επιτυχή μεταφορά (νομίσματος ή μάρκας), εκπέμπεται το γεγονός Processed (γραμμή 145), μαζί με τις παραμέτρους _verification και _timestamp. Ακολουθούν οι συναρτήσεις receive (γραμμές 151 έως 155) και fallback (γραμμές 157 έως 160) και το τέλος του έξυπνου συμβολαίου (γραμμή 161).

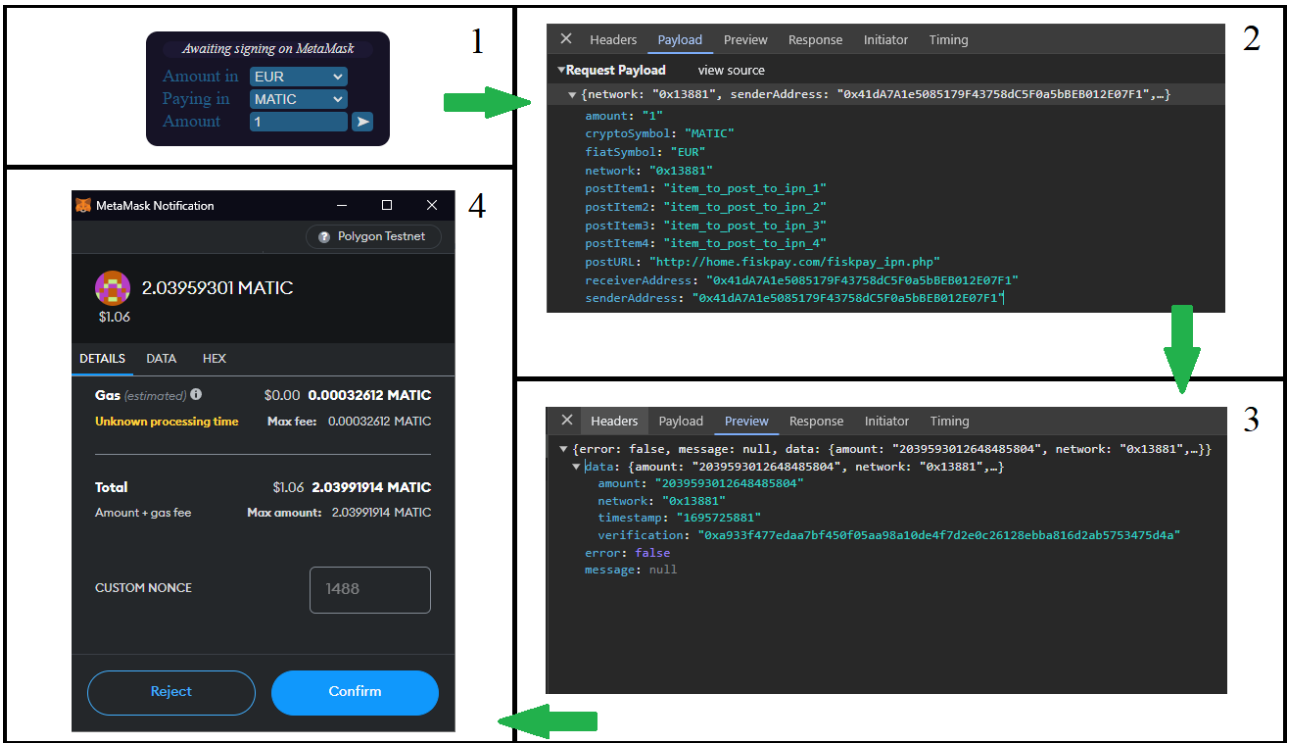


```
92  function Process(string calldata _symbol, uint256 _amount, address _receiver, bytes32 _verification, uint32 _timestamp) public payable returns(bool){
93
94      if(reentrantLocked == true)
95          revert("Reentrance failed");
96
97      reentrantLocked = true;
98
99      if(pt.GetContractAddress(".Payment.Processor") != address(this))
100         revert("Deprecated Processor");
101
102     uint32 size;
103     assembly{size := extcodesize(_receiver)}
104
105     if(size != 0)
106         revert("Receiver is contract");
107
108     if(uint32(block.timestamp - 15 minutes) > _timestamp)
109         revert ("Transaction expired");
110
111     if(isConsumed[_verification] == true)
112         revert ("Verification already consumed");
113
114     isConsumed[_verification] = true;
115
116     ISubscribers sb = ISubscribers(pt.GetContractAddress(".Payment.Subscribers"));
117
118     if(keccak256(abi.encodePacked(_symbol)) == keccak256(abi.encodePacked("MATIC")) && msg.value > 0 && _amount == 0){ ···
127     }
128     else if(keccak256(abi.encodePacked(_symbol)) != keccak256(abi.encodePacked("MATIC")) && _amount > 0 && msg.value == 0){ ···
137     }
138     else
139         revert("Processing failed");
140
141     reentrantLocked = false;
142
143     emit Processed(_verification, _timestamp);
144     return(true);
145 }
```

Εικόνα 7. Συνάρτηση Process, έξυπνου συμβολαίου Processor

5.2 Αρχεία πελάτη

Για να μπορεί να δημιουργηθεί μία παραγγελία και να υπογραφεί μία συναλλαγή, είναι απαραίτητη η ύπαρξη ενός αρχείου, το οποίο να δίνει τις παραπάνω δυνατότητες. Το αρχείο Client,js, μέσω μίας "ειδικής" φόρμας, επιτρέπει στον πελάτη να δημιουργεί παραγγελίες και να τις καταθέτει στην υπηρεσία, μέσω αιτημάτων POST. Κατά τη λήψη του αιτήματος POST από την υπηρεσία και μετά την εκτέλεση των απαιτούμενων διαδικασιών, αποστέλλονται ως απάντηση δεδομένα, τα οποία χρησιμοποιούνται από τη βιβλιοθήκη Web3.js, για την δημιουργία μίας νέας συναλλαγής στο Web3 πορτοφόλι (wallet) του τελικού χρήστη, προτρέποντάς τον να την υπογράψει και να την καταθέσει στο δίκτυο. Διευκρινίζεται ότι, ως τελικός χρήστης χαρακτηρίζεται ο επισκέπτης της ιστοσελίδας του πελάτη της υπηρεσίας. Σχετικά με το αίτημα POST, τα δεδομένα της φόρμας ελέγχονται σε πρώτο χρόνο, μέσω του αρχείου Client.js και μόνο αν είναι ορθά, επιτρέπεται η υποβολή του στην υπηρεσία. Παρόλο που γίνεται έλεγχος και κατά την παραλαβή των αιτημάτων, ο εκ των προτέρων έλεγχος αποσυμφορεί τον εξυπηρετητή της υπηρεσίας, ο οποίος δέχεται λιγότερα, μη ορθά ως προς το περιεχόμενο, αιτήματα. Στην περίπτωση που κάποιο από τα εισαγόμενα δεδομένα είναι μη αποδεκτό, ο τελικός χρήστης ενημερώνεται μέσω των μηνυμάτων της φόρμας.

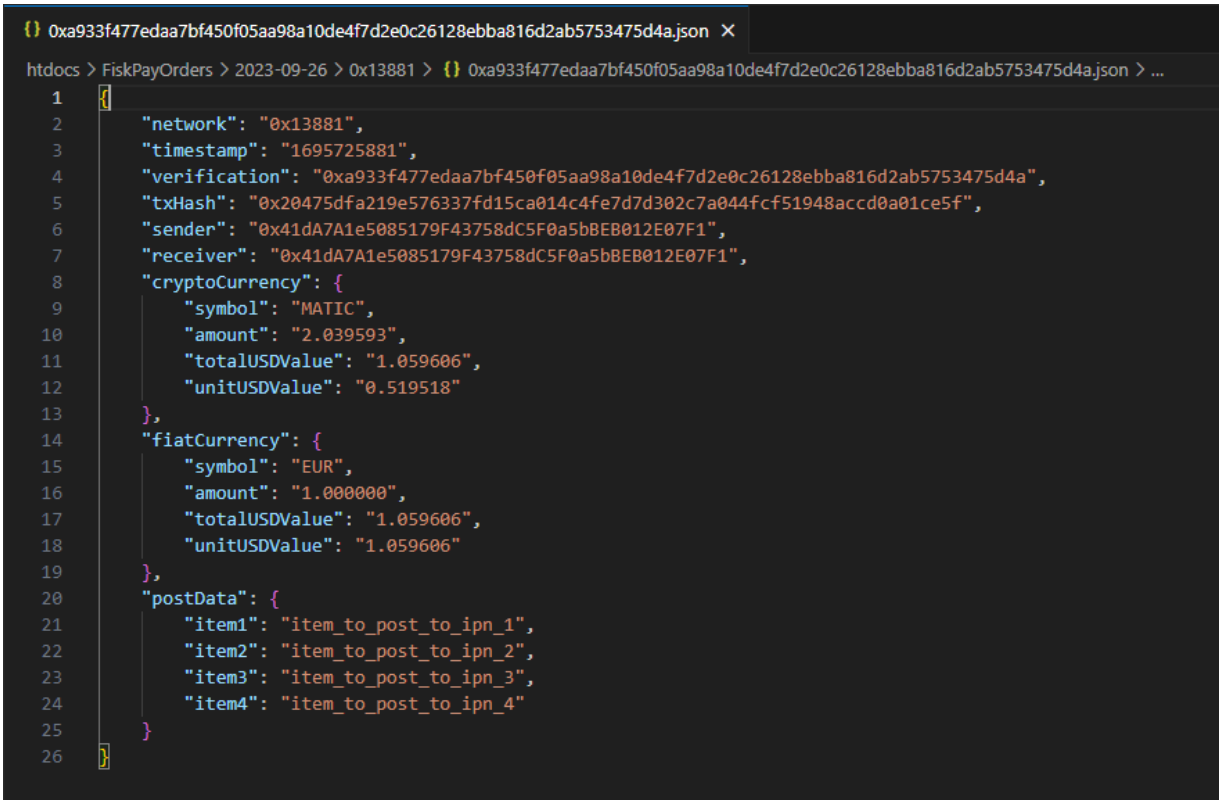


Εικόνα 8. Υποβολή (1) - δεδομένα (2) παραγγελίας, δεδομένα (3) - δημιουργία (4) συναλλαγής

Εν συνεχεία, μετά την υποβολή της συναλλαγής στο δίκτυο και την επιβεβαίωσή της από την υπηρεσία, εκκρεμεί η ενημέρωση του πελάτη, η οποία πραγματοποιείται μέσω των διαδικασιών σκανδαλισμού (trigger) και διεκδίκησης (claim). Κάθε φορά που ο πελάτης καταθέτει μία παραγγελία (order) στην υπηρεσία, οφείλει να υποδεικνύει ένα αρχείο (στον παγκόσμιο ιστό), ώστε αυτό να χρησιμοποιηθεί από την διαδικασία σκανδαλισμού. Η διαδικασία σκανδαλισμού πρακτικά

αφορά την αποστολή κωδικοποιημένων δεδομένων (προς το αρχείο του πελάτη), μέσω αιτήματος POST. Μετά την λήψη των δεδομένων, ο πελάτης είναι σε θέση να διεκδικήσει την παραγγελία αποστέλλοντας ένα αίτημα GET προς την υπηρεσία, το οποίο περιλαμβάνει τα δεδομένα που έλαβε μέσω του αιτήματος POST καθώς και την Ethereum διεύθυνσή του. Με την επιτυχή διεκδίκηση, λαμβάνονται τα δεδομένα της παραγγελίας και τα δεδομένα της συναλλαγής, τα οποία χρησιμοποιούνται από τον πελάτη κατά τη δική του βούληση.

Αν και η διαχείριση ενός αιτήματος POST και η πραγματοποίηση ενός αιτήματος GET είναι απλές ως διαδικασίες, το να γίνεται η ανάπτυξη του αρχείου σκανδαλισμού από τον πελάτη, δεν είναι καλή πρακτική. Για τον λόγο αυτό, δημιουργήθηκε το αρχείο IPN.php, μέσω του οποίου εξυπηρετούνται οι διαδικασίες σκανδαλισμού (λήψη αιτήματος POST) και διεκδίκησης (αποστολή αιτήματος GET), προσφέροντας στον πελάτη απευθείας τα δεδομένα που τον ενδιαφέρουν.



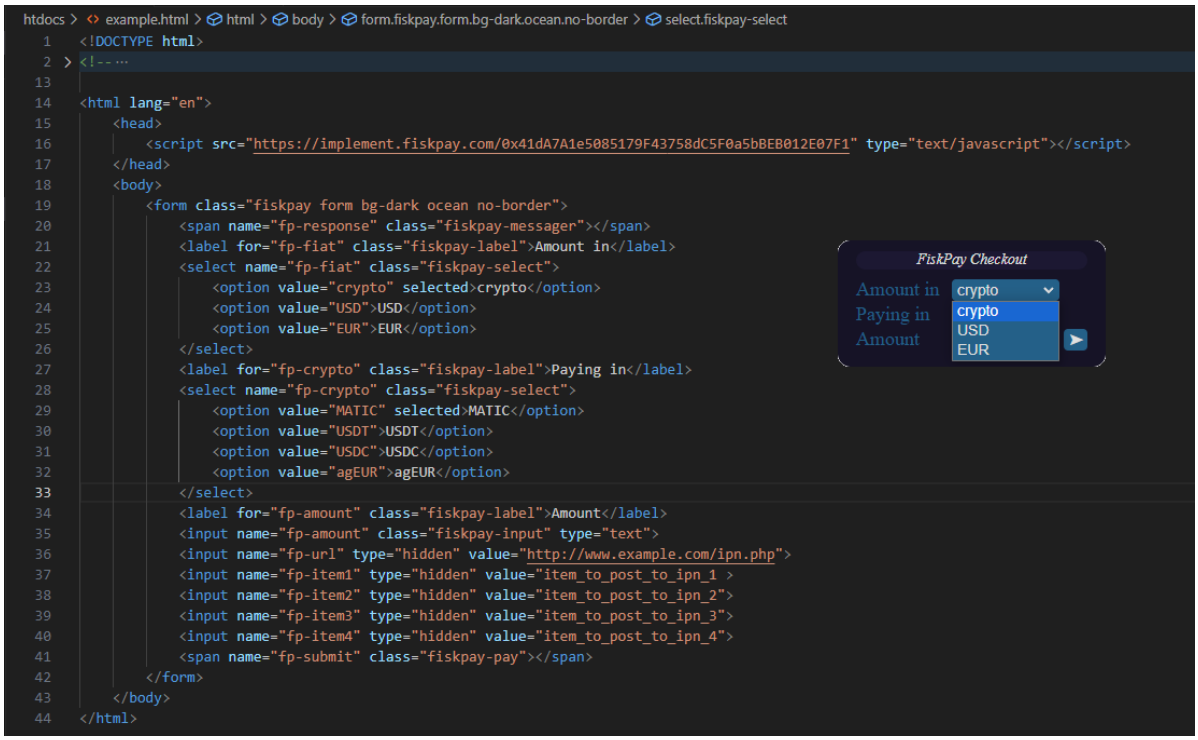
Εικόνα 9. Δεδομένα που αποστέλλονται κατά την επιτυχή διεκδίκηση παραγγελιών

5.2.1 Client.js

Σύνδεσμος GitHub:
https://github.com/FiskPay/service/blob/dissertation/website/Client.js

Σχετικά με το αρχείο Client.js και πριν την παρουσίαση του κώδικά του, χρειάζεται να γίνουν κάποιες διευκρινήσεις. Συγκεκριμένα, στην εισαγωγή της ενότητας, έγινε αναφορά σε μία "ειδική" φόρμα η οποία χρησιμοποιείται την δημιουργία των παραγγελιών. Μιας και η φόρμα δημιουργείται μέσω του υπό εξέταση αρχείου, ας ξεκινήσουμε μελετώντας τον τρόπο σύνταξής της.

Για να μπορέσει να "ενεργοποιηθεί" η φόρμα γίνεται εισαγωγή του αρχείου Client.js (γραμμή 16, Εικόνα 10.) μέσω εξωτερικού συνδέσμου, στον οποίο και καταχωρείται η Ethereum διεύθυνση του πελάτη. Με αυτό τον τρόπο όχι μόνο ο πελάτης χρειάζεται ελάχιστο χρόνο να ενσωματώσει το αρχείο Client.js στη σελίδα του, αλλά από μέρους της υπηρεσίας, μπορεί να γίνει ενημέρωση του αρχείου σε πραγματικό χρόνο και για όλους τους πελάτες, εφόσον κάτι τέτοιο κριθεί αναγκαίο. Σχετικά με την σύνταξη της φόρμας (γραμμές 19 έως 42), μέσω της κλάσης fiskpay και μερικών άλλων κλάσεων γίνεται αντίστοιχα, η παρακολούθηση της φόρμας από την υπηρεσία και η ρύθμιση των χρωμάτων της φόρμας (γραμμή 19). Στην πρώτη γραμμή εντός της φόρμας, γίνεται εισαγωγή του πλαισίου μηνυμάτων (γραμμή 20). Στη συνέχεια γίνεται επιλογή σχετικά με το αν το ποσό που θα εισαχθεί στη φόρμα (γραμμές 34 και 35), αντιστοιχεί στο κρυπτονόμισμα (ή μάρκα) που θα επιλεγεί μέσω της φόρμας (γραμμές 27 έως 33) ή το ποσό που εισάγεται αφορά δολάρια (USD) ή ευρώ (EUR) και θα πρέπει να γίνει μετατροπή του ποσού στην κατάλληλη ποσότητα κρυπτονομισμάτων (ή μαρκών), από την υπηρεσία. Τέλος γίνεται η εισαγωγή του συνδέσμου, προς το αρχείο σκανδαλισμού (IPN.php) του πελάτη (γραμμή 36), εισαγωγή των δεδομένων της παραγγελίας (γραμμές 37 έως 40) και το κουμπί υποβολής της φόρμας (γραμμή 41).



Εικόνα 10. Ενδεικτικός κώδικας και γραφικό αποτέλεσμα, φόρμας δημιουργίας παραγγελιών

Ερχόμενοι λοιπόν στο υπό επεξήγηση αρχείο, το αρχείο Client.js ξεκινάει με την εισαγωγή των βιβλιοθηκών Web3.js (γραμμές 1 έως 7) και SHA256.js (γραμμές 9 έως 15), μέσω εξωτερικών συνδέσμων. Η πρώτη βιβλιοθήκη χρησιμοποιείται (μεταξύ άλλων) για την επικοινωνία με το Web3 πορτοφόλι του περιηγητή του χρήστη, ενώ η δεύτερη επιτρέπει την χρήση του αλγορίθμου sha256, στο αρχείο. Έπειτα, γίνεται η δήλωση της σταθεράς url (γραμμή 17) και μερικών μεταβλητών (γραμμές 20 έως 24) και στη συνέχεια δηλώνονται οι λειτουργίες που πρέπει να εκτελεστούν κατά τη λήψη του αρχείου από την ιστοσελίδα του πελάτη, μέσω της συνάρτησης onload.

Κατά την λήψη του αρχείου και μέσω της συνάρτησης onload (γραμμές 26 έως 114), αρχικά λαμβάνεται γνώση για όλες τις φόρμες της υπηρεσίας, οι οποίες υπάρχουν στην ιστοσελίδα του πελάτη (γραμμή 28) και αποθηκεύεται ένας ψευδοτυχαίος αριθμός (γραμμή 29). Αμέσως μετά και για κάθε φόρμα που έχει εντοπιστεί (γραμμές 31 έως 97), γίνονται κάποιες καταχωρίσεις (γραμμές 33 έως 36) και στη συνέχεια λαμβάνονται οι εισαγόμενες μεταβλητές (γραμμές 38 έως 47), οι οποίες ελέγχονται ως προς την μοναδικότητά τους (γραμμές 49 έως 66). Στην περίπτωση που κάποια φόρμα περιέχει πολλαπλές καταχωρίσεις, γίνεται ενημέρωση του πελάτη (γραμμές 38 έως 72), αλλιώς γίνεται αποδοχή της φόρμας από την υπηρεσία (γραμμές 73 έως 97). Τέλος, αφού έχει γίνει ο έλεγχος όλων των φορμών, λαμβάνεται η διεύθυνση Ethereum από τον σύνδεσμο (γραμμή 100 αρχείου Client.js και γραμμή 16, Εικόνα 10.) η οποία μετά τους απαραίτητους ελέγχους (γραμμές 103 έως 106), αποθηκεύεται (γραμμές 107 έως 111).

Μετά την συμπλήρωση της φόρμας από τον τελικό χρήστη και κατά την υποβολή της φόρμας, εκτελείται η συνάρτηση Pay (γραμμές 116 έως 583), η οποία ξεκινάει μία αλληλουχία διαδικασιών.

Αρχικά γίνεται έλεγχος σχετικά με το αν υπάρχει κάποιο Web3 πορτοφόλι, ως επέκταση στον περιηγητή (γραμμές 141 έως 151). Αφότου βρεθεί η σχετική επέκταση, αποστέλλετε προς αυτή εντολή σύνδεσης με το δίκτυο του Polygon (γραμμές 153 έως 181) και γίνεται επιβεβαίωση ότι η εντολή ικανοποιήθηκε (γραμμές 183 έως 188). Έχοντας "τακτοποιήσει" το ζήτημα της σύνδεσης με το δίκτυο, γίνεται λήψη της διεύθυνσης Ethereum από την επέκταση του περιηγητή (γραμμές 190 και 191) και ρύθμιση του τί πρέπει να γίνει, στην περίπτωση που ο (τελικός) χρήστης αλλάξει την διεύθυνση Ethereum (γραμμές 193 έως 200) ή αλλάξει το δίκτυο (γραμμές 201 έως 205). Στη συνέχεια γίνεται η δήλωση των διεπαφών (interface) των συμβολαίων Parent, Processor, Currencies, Subscribers (χρησιμοποιείται για τις συνδρομές) και του πρότυπου ERC-20 (γραμμές 207 έως 211) και μετά μέσω της βιβλιοθήκης Web3 και της μεταβλητής web3Instance (γραμμή 213), πραγματοποιείται σύνδεση με τα συμβόλαια (γραμμές 225 έως 234). Σε αυτό το σημείο, είμαστε σε θέση να "ασχοληθούμε" με την φόρμα. Αφού λοιπόν γίνει ανάγνωση και αποθήκευση των δεδομένων της φόρμας (γραμμές 236 έως 244), ελέγχεται αν το κρυπτονόμισμα (ή μάρκα), που εισήχθη υποστηρίζεται από την υπηρεσία (γραμμές 246 έως 259) και αν τα δεδομένα που δόθηκαν έχουν αποδεκτές τιμές (γραμμές 161 έως 329). Στη συνέχεια γίνονται κάποιοι έλεγχοι που σχετίζονται με την ύπαρξη ή μη, συνδρομής (γραμμές 331 έως 347) και μετά δημιουργείται το αντικείμενο sendObject (γραμμή 349) το οποίο περιέχει το σύνολο των δεδομένων, που επρόκειτο θα σταλούν μέσω αιτήματος POST στην υπηρεσία (γραμμές 351 έως 362). Η υποβολή του αιτήματος POST γίνεται μέσω των εντολών open, setRequestHeadder και send (γραμμές 576 έως 578). Έπειτα, μετά τη λήψη της απάντησης (γραμμές 376 έως 574) από την υπηρεσία, ελέγχεται αν σε αυτή υπάρχει κάποιο μήνυμα λάθους (ή αν η ίδια η απάντηση είναι λάθος) (γραμμές 385 έως 403) και αν δεν υπάρχει, δηλώνεται η συνάρτηση process. Η συνάρτηση process, χρησιμοποιείται για την κατάθεση των συναλλαγών στο δίκτυο και την ενημέρωση του χρήστη, όσο αφορά την κατατεθειμένη συναλλαγή. Έτσι λοιπόν, μετά την ανάγνωση της απάντησης της υπηρεσίας

(γραμμές 501 έως 503), ανάλογα με το αν η πληρωμή αφορά το νόμισμα MATIC ή όχι (γραμμές 505 έως 519 και 520 έως 572, αντίστοιχα), γίνονται έλεγχοι σχετικά με το αν η συναλλαγή μπορεί να κατατεθεί επιτυχώς στο δίκτυο και αν αυτό είναι εφικτό, γίνεται σκανδαλισμός της επέκτασης του πελάτη, εισάγοντας τα δεδομένα της απάντησης στην συνάρτηση process (γραμμή 511 για MATIC και γραμμές 558 και 563 για διαφορετικό του MATIC).

Έχοντας καλύψει την συνάρτηση Pay και μετά την ανάπτυξή της, γίνεται εισαγωγή (inject) της σχεδίασης της φόρμας (γραμμές 585 έως 1138), στη σελίδα του πελάτη, ολοκληρώνοντας έτσι την περιγραφή του αρχείου Client.js.

5.2.2 IPN.php

Σύνδεσμος GitHub:
https://github.com/FiskPay/service/blob/dissertation/client/IPN.php

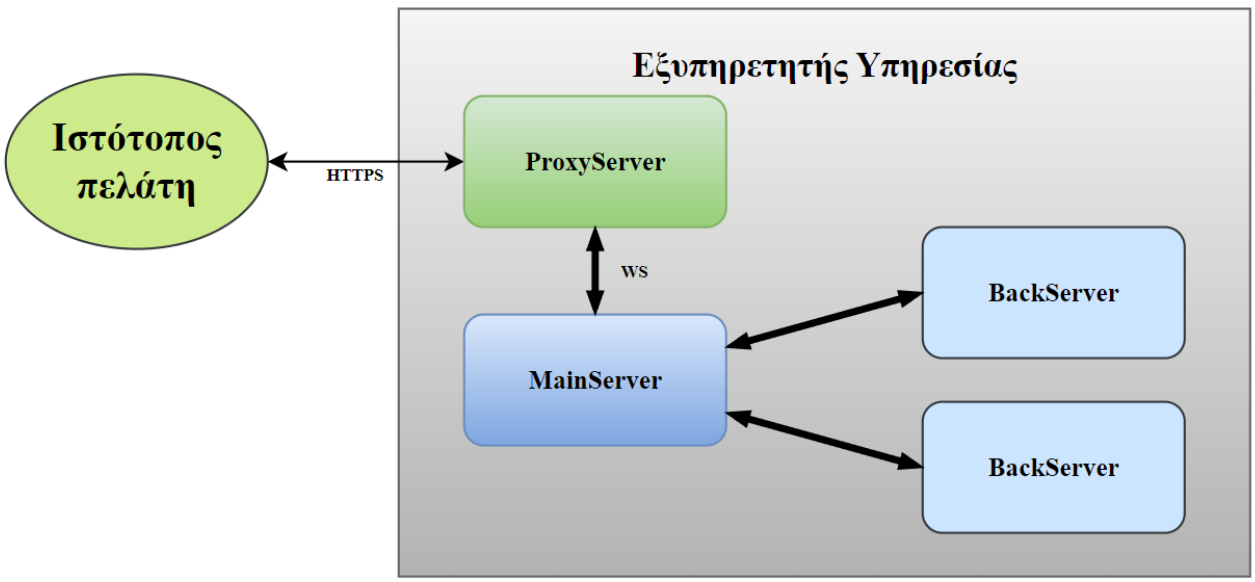
Το αρχείο IPN.php, όπως διατυπώθηκε νωρίτερα χρησιμοποιείται για τον σκανδαλισμό του πελάτη από την υπηρεσία και από μέρους του πελάτη, για την διεκδίκηση παραγγελιών. Υπενθυμίζεται ότι για την διεκδίκηση μίας παραγγελίας, χρησιμοποιούνται τα δεδομένα που λαμβάνονται από την υπηρεσία (μέσω του σκανδαλισμού) και η Ethereum διεύθυνση του πελάτη (της υπηρεσίας).

Αρχικά λοιπόν στο αρχείο δηλώνεται η διεύθυνση του πελάτη (γραμμή 11) και αν θέλουμε να γίνει καταγραφή των δεδομένων της παραγγελίας (γραμμή 14) μετά την διεκδίκησή της (γραμμές 89 έως 100). Έχοντας κάνει τις παραπάνω δηλώσεις, γίνεται έλεγχος του αν η πρόσβαση στο αρχείο γίνεται μέσω αιτήματος POST και εφόσον αυτό συμβαίνει, αποθηκεύονται τα δεδομένα του αιτήματος στην μεταβλητή $iPNData (γραμμή 24). Έπειτα, ελέγχεται αν τα δεδομένα της μεταβλητής $iPNData είναι κωδικοποιημένα ως BYTE64 και εφόσον είναι, αποστέλλεται αίτημα GET στην υπηρεσία (διεκδίκηση παραγγελίας), η απάντηση του οποίου αποθηκεύεται στην μεταβλητή $responseString (γραμμή 26). Γνωρίζοντας ότι η απάντηση θα πρέπει να είναι υπό μορφή JSON, στην επόμενη γραμμή, γίνεται ο σχετικός έλεγχος, επιβεβαιώνεται ότι στην απάντηση περιέχονται τα αντικείμενα error, message και data (γραμμή 28) και μετά επιβεβαιώνεται ότι η απάντηση περιέχει δεδομένα και όχι κάποιο σφάλμα (γραμμή 30). Έχοντας πραγματοποιήσει όλους τους απαραίτητους ελέγχους, γίνεται αποθήκευση των πληροφορίων της απάντησης στις αντίστοιχες μεταβλητές (γραμμές 32 έως 53) και δημιουργείται χώρος για την ανάπτυξη των επακόλουθων διαδικασιών από μεριά του πελάτη, ανάλογα με το δίκτυο που χρησιμοποιήθηκε για την συναλλαγή (γραμμές 59 έως 70 και 71 έως 83).

5.3 Αρχεία εξυπηρετητή

Ο εξυπηρετητής της υπηρεσίας, περιγράφεται συνοπτικά από τρεις (3εις) βασικές λειτουργίες. Αρχικά, λαμβάνει τη παραγγελία (order) από τον ιστότοπο του εκάστοτε πελάτη και μετά από επεξεργασία, την αποθηκεύει. Στη συνέχεια, υποδεικνύει στον (ιστότοπο του) πελάτη την συναλλαγή που αντιστοιχεί στην παραγγελία του, ώστε εκείνος να την προωθήσει στον τελικό χρήστη (τον πελάτη, του πελάτη της υπηρεσίας). Τέλος, ο εξυπηρετητής παρακολουθώντας το γεγονός Processed (του έξυπνου συμβολαίου Processor) για νέες συναλλαγές, σε κάθε νέα συναλλαγή, ενημερώνει τον κατάλληλο πελάτη (client), αποστέλλοντάς του τα δεδομένα της παραγγελίας μαζί με κάποιες επιπλέον πληροφορίες, όπως το κόστος και τις πληροφορίες της συναλλαγής στο δίκτυο.

Αν και για λόγους κατανόησης έχουμε αναφερθεί με τρόπο, ώστε ότι η υπηρεσία να φαίνεται ότι αποτελείται από ένα (1) μόνο εξυπηρετητή, στην πραγματικότητα αποτελείται από τρεις (3εις), τον **BackServer**, τον **MainServer** και τον **ProxyServer** (αρχεία BackServer.js, MainServer.js, ProxyServer.js, αντίστοιχα). Η τμηματοποίηση αυτή, έχει ως στόχο την απομόνωση των MainServer και BackServer από τον παγκόσμιο ιστό (ή τουλάχιστον την απόκρυψή τους από αυτόν), χρησιμοποιώντας τον ProxyServer ως ενδιάμεσο. Επειδή η επικοινωνία μεταξύ MainServer και πελάτη γίνεται αποκλειστικά μέσω αυτού, ο ProxyServer αποτελεί το μόνο γνωστό "σημείο" της υπηρεσίας, οπότε είναι και το μόνο στοιχείο που είναι άμεσα εκτεθειμένο σε επιθέσεις (π.χ. DDoS). Έτσι, ακόμα και αν γίνει κάποια επίθεση, οι άλλοι δύο, οι οποίοι "βρίσκονται" σε άλλο μηχάνημα ή σε διαφορετικά μεταξύ τους μηχανήματα, παραμένουν ασφαλείς, συνεχίζοντας να εκτελούν τις διεργασίες επιβεβαίωσης εκτός κινδύνου και διακόπτοντας μόνο τις διεργασίες που αφορούν τον ProxyServer (π.χ. λήψη νέας παραγγελίας από τον πελάτη).



Εικόνα 11. Εσωτερική δομή του εξυπηρετητή της υπηρεσίας

Εστιάζοντας στους άλλους δύο εξυπηρετητές, ο BackServer είναι υπεύθυνος για την παρακολούθηση του γεγονότος Processed και για τη μεταβίβαση των πληροφοριών (verification, timestamp) του κάθε γεγονότος στον MainServer, ο οποίος με τη σειρά του, αφού έχει αρχικά λάβει, επεξεργαστεί και αποθηκεύσει την παραγγελία του πελάτη, τον ειδοποιεί, στέλνοντάς του τα σχετικά δεδομένα, μέσω του ProxyServer. Γνωρίζοντας ότι αν για οποιοδήποτε λόγο η παρακολούθηση (σε πραγματικό χρόνο) του γεγονότος Processed αποτύχει, δεν θα μπορεί να γίνει επιβεβαίωση της συναλλαγής και κατά συνέπεια της παραγγελίας, η υπηρεσία προβλέπει την ύπαρξη δύο (2) BackServer, με τον καθένα από αυτούς να διατηρεί δύο (2) ξεχωριστές (έμμεσες, μέσω WebSocket εξωτερικών παρόχων) συνδέσεις με το δίκτυο Polygon, για λόγους πλεονασμού (redundancy). Σχετικά με τον τρόπο επικοινωνίας μεταξύ των εξυπηρετητών και μεταξύ του πελάτη και του ProxyServer, η επικοινωνία στην πρώτη περίπτωση γίνεται μέσω διαδικτυακών βυσμάτων (WebSocket - WS), ενώ στη δεύτερη μέσω αιτημάτων HTTPS.

5.3.1 BackServer.js

Σύνδεσμος GitHub:
https://github.com/FiskPay/service/blob/dissertation/server/BackServer.js

Όπως διατυπώθηκε προηγουμένως, ο BackServer είναι υπεύθυνος για την παρακολούθηση των συναλλαγών που πραγματοποιούνται μέσω του έξυπνου συμβολαίου Processor και την προώθηση των δεδομένων της συναλλαγής, στον MainServer, χρησιμοποιώντας μια σύνδεση WebSocket. Επίσης, έχουμε αναφέρει ότι η σύνδεση με το δίκτυο του Polygon, επιτυγχάνεται συνδεόμενοι σε WebSockets εξωτερικών παρόχων. Ας δούμε λοιπόν, πως υλοποιούνται τα παραπάνω…

Στις πρώτες γραμμές (γραμμές 1 και 2), γίνεται η εισαγωγή των κλάσεων dotenv (γραμμη 1) και sha256 (γραμμή 2). Η dotenv, δίνει την δυνατότητα εισαγωγής παραμέτρων σε ένα αρχείο, διατηρώντας την τιμή τους κρυφή και συνήθως χρησιμοποιείται για την απόκρυψη ευαίσθητων στοιχείων, όπως κωδικούς ασφάλειας, ενώ η sha256, επιτρέπει την χρήση του ομώνυμου αλγορίθμου hashing στο αρχείο. Στις επόμενες γραμμές (γραμμές 4 και 5) εισάγονται οι κλάσεις Listener (γραμμή 4) και DataLoop (γραμμή 5). Η κλάση Listener, αναπτύχθηκε αποκλειστικά για το συγκεκριμένο αρχείο και πραγματοποιεί τη σύνδεση στο blockchain μέσω της συνάρτησης connect (γραμμή 91), την παρακολούθηση της κατάστασης των συνδέσεων (γραμμές 29 έως 37) και την παρακολούθηση του έξυπνου συμβολαίου Processor (γραμμές 38 έως 52). Η DataLoop (η οποία χρησιμοποιείται και από άλλα αρχεία της υπηρεσίας), δημιουργεί ένα “κυκλικό” χώρο, προκαθορισμένων θέσεων στην μνήμη, στον οποίο αποθηκεύονται δεδομένα υπό το καθεστώς First In First Out (FIFO). Καλώντας την συνάρτηση exists (γραμμή 40), ελέγχεται αν το εισαγόμενο δεδομένο περιέχεται σε κάποια θέση του χώρου, ενώ η συνάρτηση push (γραμμή 43) τοποθετεί το δεδομένο (που εισάγεται στη συνάρτηση) σε μία “κενή” θέση ή στη θέση με τη αρχαιότερη εγγραφή. Έπειτα, εισάγονται η συνάρτηση dateTime (γραμμή 7) και η κλάση io (γραμμή 9), με την κλήση της πρώτης να επιστρέφει την τρέχουσα ημερομηνία και ώρα και τη δεύτερη επιτρέπει την σύνδεση (ως client) σε WebSocket.

Μετά την εισαγωγή των κλάσεων και συναρτήσεων, γίνεται η δήλωση των μεταβλητών και σταθερών. Αρχικά, για να έχουμε πρόσβαση στις παραμέτρους του dotenv, δηλώνεται η σταθερά myENV (γραμμή 11) και μέσω αυτής, αποθηκεύονται οι υπερσύνδεσμοι των εξωτερικών παρόχων

(που χρησιμοποιούνται για την σύνδεση στο Polygon) στην mainnetProviderURLs (γραμμή 13). Στη συνέχεια ακολουθούν διάφορες δηλώσεις (γραμμές 14 έως 27), όπου σε αυτές περιλαμβάνονται η δήλωση της transaction (γραμμή 22) για την αναφορά στη κλάση DataLoop, η δήλωση της listener (γραμμή 23) για την αναφορά στην κλάση Listener και η wsClient (γραμμή 24) η οποία αφορά την σύνδεση του BackServer, στο WebSocket του MainServer.

Η κλάση Listener, είναι γραμμένη έτσι ώστε όταν αλλάζει η κατάσταση των συνδέσεων (σύνδεση – αποσύνδεση) προς τους εξωτερικούς παρόχους (δηλαδή τις συνδέσεις προς το Polygon) ή λαμβάνεται ένα νέο γεγονός από το δίκτυο, να δημιουργεί γεγονότα. Συγκεκριμένα, κατά την αλλαγή της κατάστασης των συνδέσεων, εκπέμπεται το γεγονός providerChange, ενώ για κάθε επιτυχή συναλλαγή, εκπέμπεται το γεγονός newTransaction. Στις επόμενες γραμμές (από 29 έως 52), δηλώνονται οι διαδικασίες που εκτελούνται κατά τη λήψη των παραπάνω γεγονότων, χρησιμοποιώντας τη σταθερά listener. Η πρώτη διαδικασία (γραμμές 29 έως 37), αφορά την εκτύπωση της κατάστασης των συνδέσεων (στην κονσόλα του εξυπηρετητή) και η δεύτερη διαδικασία (γραμμές 38 έως 52) αφορά την ειδοποίηση του MainServer, όταν πραγματοποιείται μία νέα συναλλαγή. Η ειδοποίηση προς τον MainServer γίνεται σε πραγματικό χρόνο (γραμμή 45), εφόσον υπάρχει σύνδεση προς αυτόν, ενώ στην περίπτωση που δεν υπάρχει, οι πληροφορίες της συναλλαγής αποθηκεύονται (γραμμή 43) και αποστέλλονται, μόλις αποκατασταθεί η σύνδεση (γραμμή 74). Κατά αντίστοιχο τρόπο και μέσω της σταθεράς wsClient, διατυπώνονται οι διαδικασίες που αφορούν την επικοινωνία μεταξύ MainServer και BackServer (γραμμές 54 έως 87), ενώ η επικοινωνία ξεκινά με κλήση της συνάρτησης connect (γραμμή 89).

5.3.2 ProxyServer.js

Σύνδεσμος GitHub:
https://github.com/FiskPay/service/blob/dissertation/server/ProxyServer.js

Ο ProxyServer, όπως έχουμε αναφέρει, λειτουργεί ως διαβιβαστής μηνυμάτων μεταξύ του πελάτη και του MainServer. Όπως στον BackServer, έτσι και στον ProxyServer, η επικοινωνία με τον MainServer πραγματοποιείται μέσω WebSocket, ενώ η ανταλλαγή μηνυμάτων μεταξύ ProxyServer και πελάτη, γίνεται μέσω HTTPS. Αρχικά λοιπόν, γίνεται εισαγωγή της κλάσης dotenv (γραμμή 1), των κλάσεων που χρειάζονται για να πραγματοποιηθούν οι εξωτερικές συνδέσεις (γραμμές 2 έως 9) καθώς και η συνάρτηση dateTime (γραμμή 11). Για να μπορεί ο πελάτης να επικοινωνήσει με τον ProxyServer (μέσω HTTPS) και να του στείλει αιτήματα, θα πρέπει να "στηθεί" ένας εξυπηρετητής διαδικτύου (webserver). Στη συνέχεια λοιπόν και αφού δηλωθεί η σταθερά myENV (γραμμή 13), μέσω της κλάσης express (γραμμή 2), γίνεται ρύθμιση του HTTPS εξυπηρετητή (γραμμές 15 έως 40), ο οποίος και δημιουργείται στην επόμενη γραμμή (γραμμή 42). Στο σημείο αυτό θα πρέπει να τονιστεί ότι η επικοινωνία του πελάτη με τον webserver, δεν είναι εφικτή, μέχρις ότου γίνει κλήση της συνάρτησης listen (γραμμή 230). Οι τελευταίες δηλώσεις αφορούν τη δημιουργία των σταθερών httpAgent (γραμμή 44) και httpsAgent (γραμμή 45), οι οποίες χρησιμοποιούνται κατά την δημιουργία (νέων) αιτημάτων HTTP και HTTPS, (αντίστοιχα στις γραμμές 200 και 202) από ProxyServer προς τον πελάτη και την δήλωση της σταθεράς wsClient, για την επικοινωνία με τον MainServer. Τονίζεται ότι όταν η (αρχική) επικοινωνία γίνεται από τον πελάτη, χρησιμοποιείται πάντα ασφαλής σύνδεση (HTTPS), ενώ όταν η (αρχική) επικοινωνία γίνεται από τον ProxyServer (δηλαδή κατά την ειδοποίηση του πελάτη, ότι μία συναλλαγή καταχωρήθηκε επιτυχώς στο δίκτυο),

η σύνδεση (ως προς την ασφάλεια) εξαρτάται από τον εξυπηρετητή του πελάτη (γραμμή 199). Έχοντας κάνει την απαραίτητη "προεργασία", μπορεί πλέον να γίνει το "στήσιμο" του webserver που αναφέραμε προηγουμένως.

Ξεκινώντας από τον ιστότοπο του πελάτη, ο επισκέπτης, μέσω του αρχείου Client.js, ξεκινά τη διαδικασία (μίας νέας) συναλλαγής. Έχοντας συμπληρώσει τη σχετική φόρμα, κατά την υποβολή της, αποστέλλετε ένα αίτημα POST προς τον ProxyServer η διαχείριση του οποίου γίνεται μέσω των λειτουργιών της createOrder (γραμμές 55 έως 107), του εξυπηρετητή HTTPS. Αρχικά λοιπόν, έχοντας λάβει τα δεδομένα από το POST, ελέγχεται αν υπάρχει επικοινωνία με τον MainServer (γραμμή 57) και στην περίπτωση που δεν υπάρχει (γραμμές 62 έως 69), εμφανίζεται στον επισκέπτη το σχετικό μήνυμα μέσω του πεδίου μηνυμάτων της φόρμας (της ιστοσελίδας του πελάτη). Μετά τον έλεγχο και εφόσον υπάρχει επικοινωνία, αποστέλλονται τα δεδομένα του POST (γραμμή 75) στον MainServer και δημιουργείται μία υπόσχεση (Promise) (γραμμή 77), η οποία επιστρέφει (ετεροχρονισμένα) την απάντησή του (γραμμές 81 έως 91) ή ότι το αίτημα έληξε (γραμμές 93 έως 102), στην περίπτωση που δεν έχει ληφθεί απάντηση μετά από 15 (δεκαπέντε) δευτερόλεπτα. Η απάντηση προωθείται στον ιστότοπο του πελάτη, μέσω των τελευταίων γραμμών κώδικα (γραμμές 105 και 106) και αν η απάντηση είναι έγκυρη (δεν έχει λήξει το αίτημα), ο επισκέπτης (της ιστοσελίδας του πελάτη), καλείται να υπογράψει και να καταθέσει την συναλλαγή στο δίκτυο. Μετά την κατάθεση της συναλλαγής στο δίκτυο, ο BackServer αποστέλλει τα δεδομένα της συναλλαγής στον MainServer και εκείνος με τη σειρά του, μέσω των λειτουργιών του γεγονότος triggerCustomer (γραμμές 182 έως 210), αποστέλλει ένα αίτημα POST προς το αρχείο IPN.php, το οποίο φιλοξενείται στον εξυπηρετητή του πελάτη. Αμέσως και μέσω του αρχείου IPN.php, δημιουργείται ένα (ασφαλές) αίτημα GET από τον πελάτη προς τον ProxyServer, το οποίο εξυπηρετείται μέσω των λειτουργιών της claimOrder (γραμμές 108 έως 175). Η claimOrder δομείται με τρόπο αντίστοιχο της createOrder, με τη διαφορά ότι η απάντηση που λαμβάνεται από τον MainServer (γραμμή 134), περατώνει την παραγγελία, αποστέλλοντας τα δεδομένα της παραγγελίας και του δικτύου στον πελάτη. Σημειώνεται ότι ο HTTPS εξυπηρετητής, επιστρέφει τον κωδικό σφάλματος 403, σε κάθε άλλη περίπτωση (γραμμές 176 έως 180).

5.3.3 MainServer.js

Σύνδεσμος GitHub:
https://github.com/FiskPay/service/blob/dissertation/server/MainServer.js

Ο βασικός ή κύριος εξυπηρετητής MainServer, όπως πιθανώς προκύπτει και από το όνομά του, αποτελεί τον κόμβο συλλογής, επεξεργασίας, αποθήκευσης, διαχείρισης και προώθησης του συνόλου των δεδομένων της υπηρεσίας. Κατά την λήψη μιας παραγγελίας (γραμμές 69 έως 72), χρησιμοποιώντας πάντα τον ProxyServer ως ενδιάμεσο, δημιουργείται μία συναλλαγή, τα δεδομένα της οποίας προωθούνται στον πελάτη και από αυτόν, στον πελάτη του. Με την κατάθεση της συναλλαγής στο δίκτυο, ο BackServer στέλνει τα απαραίτητα δεδομένα (γραμμή 93 έως 106) στον MainServer και γίνεται κλήση της συνάρτησης trigger (γραμμές 28 έως 53), μέσω της οποίας αποστέλλεται (έμμεσα) ειδοποίηση στον πελάτη, ως προς την περάτωση της παραγγελίας. Έπειτα, χρησιμοποιώντας τα δεδομένα που έλαβε, ο πελάτης, διεκδικεί τη παραγγελία του (γραμμές 74 έως 92) και λαμβάνει ως απάντηση τα δεδομένα της παραγγελίας, καθώς και τα δεδομένα του blockchain. Η επικοινωνία με τους εξυπηρετητές ProxyServer και BackServer, γίνεται μέσω

συνδέσεων WebSocket, οι οποίες εξυπηρετούνται μέσω του MainServer (WS server).

Σε μια πιο λεπτομερή περιγραφή, αρχικά εισάγονται οι εξωτερικές κλάσεις και συναρτήσεις (γραμμές 1 έως 9), με την κλάση Orders (γραμμή 4), η οποία αναπτύχθηκε αποκλειστικά για το αρχείο MainServer.js, να χρησιμοποιείται για την διαχείριση των παραγγελιών και τις κλάσεις createServer (γραμμή 8) και Server (γραμμή 9) συνδυαστικά να χρησιμοποιούνται, για την δημιουργία του WebSocket server (γραμμές 25 και 26). Στις επόμενες γραμμές (γραμμές 11 έως 26), γίνεται δήλωση κάποιων σταθερών και αμέσως μετά δηλώνεται η συνάρτηση trigger (γραμμές 28 έως 53). Μέσω της trigger, γίνεται ο σκανδαλισμός των πελατών που "έχουν" επιβεβαιωμένες συναλλαγές, αλλά δεν έχουν λάβει γνώση για αυτό. Στην πρώτη γραμμή της συνάρτησης (γραμμή 30), αρχικά λαμβάνεται ο αριθμός τον πελατών (clients), του WebSocket server, που είναι συνδεδεμένοι ως «ProxyServer». Εφόσον υπάρχει τουλάχιστον ένας πελάτης (client) (γραμμή 32), δηλαδή τουλάχιστον ένας ProxyServer συνδεδεμένος με τον MainServer μέσω WebSocket, υπάρχει η δυνατότητα σκανδαλισμού του πελάτη της υπηρεσίας, η οποία γίνεται με την εκπομπή του γεγονότος triggerCustomer (γραμμή 43), αφού πρώτα γίνουν κάποιοι επιπλέον έλεγχοι (γραμμές 36, 38 και 40). Η σύνταξη του κώδικα συνεχίζει με τον προγραμματισμό του εξυπηρετητή WebSocket και τελειώνει με την ενεργοποίησή του.

Ο προγραμματισμός του WebSocket εξυπηρετητή (γραμμές 55 έως 154), ξεκινάει με τις λειτουργίες του γεγονότος connection, δηλαδή με την συμπεριφορά του εξυπηρετητή κατά τη σύνδεση ενός νέου πελάτη (client). Αμέσως μετά τη σύνδεση, ελέγχεται (γραμμή 62) αν η διεύθυνση IP του πελάτη περιλαμβάνεται στην άσπρη λίστα (whitelist), δηλαδή αν ο πελάτης είναι ο ProxyServer ή ένας BackServer και αν όχι, η σύνδεση διακόπτεται (γραμμή 64). Έχοντας εξασφαλίσει ότι ο συνδεδεμένος στον WebSocket server είναι ένας από τους παραπάνω, στις επόμενες γραμμές συγγράφεται η συμπεριφορά του εξυπηρετητή προς τον πελάτη (γραμμές 69 έως 153). Ξεκινώντας με τα γεγονότα createOrder (γραμμές 69 έως 72) και claimOrder (γραμμές 74 έως 92), τα οποία δημιουργούνται αποκλειστικά από τον εξυπηρετητή ProxyServer, η λήψη του πρώτου γίνεται ως προς την δημιουργία νέας παραγγελίας, ενώ η δεύτερη, ως προς την διεκδίκησή της. Κατά τη λήψη του γεγονότος createOrder και μέσω της συνάρτησης createOrder (γραμμή 71), της κλάσης Orders, αποθηκεύονται τα δεδομένα της παραγγελίας ως τοπικό αρχείο και λαμβάνονται τα δεδομένα της συναλλαγής (που συνδέονται με την παραγγελία, σταθερά responseObject), τα οποία προωθούνται στον ProxyServer, μέσω της εκπομπής του γεγονότος createOrderResponse (γραμμή 72). Η λήψη του claimOrder, γίνεται μετά τον σκανδαλισμό του πελάτη της υπηρεσίας και συγκεκριμένα μετά την αποστολή αιτήματος POST προς το αρχείο IPN.php (του πελάτη). Από το αρχείο IPN.php και αμέσως μετά τη λήψη του POST, δημιουργείται ένα GET αίτημα προς τον ProxyServer, το οποίο προωθείται στον MainServer, ως το γεγονός claimOrder. Με τη λήψη του claimOrder, αρχικά λαμβάνεται η θέση του αρχείου της παραγγελίας (γραμμή 76) και εφόσον υπάρχει το αρχείο (γραμμή 78), γίνεται διεκδίκηση (γραμμή 80) και αφαίρεση αυτής από την λίστα εκκρεμών παραγγελιών (γραμμή 82). Ανάλογα με το αν η διεκδίκηση έγινε επιτυχώς ή όχι, το αρχείο της παραγγελίας μεταφέρεται στην κατάλληλη θέση (γραμμές 84 έως 87) και το αποτέλεσμα (responseObject) προωθείται στον ProxyServer, με την εκπομπή του claimOrderResponse (γραμμή 89). Τα επόμενα δύο γεγονότα, newTransaction (γραμμές 93 έως 106) και newTransactionPacket (γραμμές 108 έως 140) σχετίζονται με την λήψη ειδοποιήσεων από τον/τους BackServer για νέες επιβεβαιωμένες συναλλαγές. Με τη λήψη του γεγονότος newTransaction, ελέγχεται αν η συναλλαγή έχει ήδη επιβεβαιωθεί (γραμμή 95) και αν όχι, χαρακτηρίζεται ως επιβεβαιωμένη (γραμμή 100) και γίνεται κλήση της συνάρτησης trigger (γραμμή 103). Η newTransactionPacket, αφορά την μη μεμονωμένη λήψη επιβεβαιωμένων συναλλαγών, δηλαδή τη λήψη πακέτου

επιβεβαιωμένων συναλλαγών και εκτελεί τις ίδιες λειτουργίες με την newTransaction.

Η ενεργοποίηση του WebSocket εξυπηρετητή (γραμμές 156 έως 175), γίνεται με την ενεργοποίηση του HTTP εξυπηρετητή, μιας και οι συνδέσεις WebSockets αποτελούν "αναβάθμιση" των συνδέσεων HTTP. Με την ενεργοποίηση λοιπόν του εξυπηρετητή και μετά από λίγη ώρα (γραμμές 160 έως 163) μέσω του setInterval (γραμμές 165 έως 174), λαμβάνονται οι εκκρεμείς συναλλαγές (γραμμή 167) και γίνεται σκανδαλισμός του εκάστοτε πελάτη (γραμμή 171), έναν προς έναν, ανά τακτά διαστήματα (γραμμή 172).

6 ΕΠΙΛΟΓΟΣ

Συνοπτικά, η υπηρεσία επιβεβαίωσης εξυπηρετεί διαδικτυακές παραγγελίες η αποπληρωμή των οποίων πραγματοποιείται στο Polygon, δηλαδή μέσω ενός EVM δικτύου blockchain. Οι πελάτες της υπηρεσίας, για να συνδεθούν με αυτή, χρειάζεται να τοποθετήσουν μία "ειδική" φόρμα στη σελίδα τους και ένα PHP αρχείο στον webserver τους. Ο τελικός χρήστης για να μπορέσει να αποπληρώσει την παραγγελία του μέσω της υπηρεσίας, θα πρέπει να έχει εγκαταστήσει στο πρόγραμμα περιήγησής του ένα Web3 wallet (π.χ. MetaMask) δηλαδή μία επέκταση, μέσω της οποίας πραγματοποιείται η επικοινωνία μεταξύ χρήστη και δικτύου Polygon.

Όπως αντιλαμβάνεται κανείς, η χρήση-απήχηση της υπηρεσίας, από τεχνολογικής άποψης, είναι άρρηκτα συνδεδεμένη με την απήχηση της Web3 τεχνολογίας. Το Web3 μη έχοντας λάβει (ακόμα;) εκτεταμένη δημοσιότητα από τα μέσα μαζικής ενημέρωσης, αποτελεί μια έννοια άγνωστη προς τη μεγαλύτερη μερίδα των χρηστών του διαδικτύου. Αυτό έχει ως αποτέλεσμα η υπηρεσίες που αναπτύσσονται ή/και έχουν αναπτυχθεί σε αυτό, να απευθύνονται σε μικρό αγοραστικό κοινό. Σκεπτόμενοι λοιπόν ότι ο κύριος περιορισμός για την καθαυτό χρήση της υπηρεσίας βρίσκεται στην τεχνολογία που χρησιμοποιεί, πέραν της επέκτασης της υπηρεσίας σε περισσότερα EVM δίκτυα, εκτιμάται ότι η κάποια περαιτέρω ανάπτυξη, επιδιώκοντας πλέον εμπορικούς σκοπούς, θα είναι άκαρπη.

Ως επόμενη επιδίωξη πάνω στο Web3, είναι η ανάπτυξη υπηρεσίας, μέσω της οποίας παίχτες διαδικτυακών παιχνιδιών, θα μπορούν να μεταφέρουν ψηφιακό χρήμα από και προς το παιχνίδι. Μέσω αυτής της (νέας) υπηρεσίας, τα συνδεδεμένα παιχνίδια θα έχουν τη επιλογή να δημιουργήσουν ένα «play to earn» μοντέλο παιχνιδιού, στο οποίο οι παίκτες έχουν και οικονομικό κίνητρο για να παίξουν, ενώ παράλληλα εξοικειώνονται με την blockchain τεχνολογία και το Web3.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

[1] IPFS Docs, 2023
Welcome to the IPFS docs [διαδικτυακό]
Πηγή: https://docs.ipfs.io/
Πρόσβαση: 11/9/2023

[2] J. Savedge, 2022
10 Examples of Animal Species Working Together in the Wild [διαδικτυακό]
Πηγή: https://www.treehugger.com/animal-species-working-together-in-wild
Πρόσβαση: 13/9/2023

[3] Wiktionary, 2023
Συναλλαγή [διαδικτυακό]
Πηγή: https://el.wiktionary.org/wiki/συναλλαγή
Πρόσβαση: 13/9/2023

[4] Euretirio, 2023
Χρήμα (Money) [διαδικτυακό]
Πηγή: https://euretirio.com/xrima/
Πρόσβαση: 13/9/2023

[5] J. Frankenfield, 2023
Digital Currency Types, Characteristics, Pros & Cons, Future Uses [διαδικτυακό]
Πηγή: https://www.investopedia.com/terms/d/digital-currency.asp
Πρόσβαση: 15/9/2023

[6] Μ. Μπίτζης, 2022
Νέα εργαλεία και εξελίξεις στις συναλλακτικές συνήθειες των πολιτών
Πηγή: https://polynoe.lib.uniwa.gr/xmlui/handle/11400/2553

[7] O. Solon & C. Farivar, 2019
Mark Zuckerberg leveraged Facebook user data to fight rivals and help friends, leaked documents show [διαδικτυακό]
Πηγή: https://www.nbcnews.com/mark-zuckerberg-leveraged-facebook-user-data
Πρόσβαση: 22/9/2023

[8] ICAEW, 2023
What is blockchain? [διαδικτυακό]
Πηγή: https://www.icaew.com/blockchain-articles/what-is-blockchain
Πρόσβαση: 24/9/2023

[9] D. L. Chaum, 1982
Computer Systems Established, Maintained and Trusted by Mutually Suspicious Groups
Πηγή: https://evervault.com/papers/chaum

[10] S. Haber & W. S. Stornetta, 1991
How to time-stamp a digital document
Πηγή: https://link.springer.com/article/10.1007/BF00196791

[11] Wikipedia, 2023
Ethereum [διαδικτυακό]
Πηγή: https://en.wikipedia.org/wiki/Ethereum
Πρόσβαση: 26/9/2023

[12] Wikipedia, 2023
Polygon (blockchain) [διαδικτυακό]
Πηγή: https://en.wikipedia.org/wiki/Polygon_(blockchain)
Πρόσβαση: 26/9/2023

[13] Bitcoin Project, 2023
How does Bitcoin work? [διαδικτυακό]
Πηγή: https://bitcoin.org/en/how-it-works
Πρόσβαση: 1/10/2023

[14] J. Frankenfield, 2023
What Is Block Time? What It Measures, Verification, and Example [διαδικτυακό]
Πηγή: https://www.investopedia.com/terms/b/block-time-cryptocurrency.asp
Πρόσβαση: 1/10/2023

[15] Travel Planner, 2023
How does SHA256 work? [διαδικτυακό]
Πηγή: https://medium.com/the-kickstarter/how-does-sha256-work-1dc89a665c32
Πρόσβαση: 1/10/2023

[16] Bitcoin Project, 2023
Blockchain [διαδικτυακό]
Πηγή: https://developer.bitcoin.org/reference/block_chain.html
Πρόσβαση: 2/10/2023

[17] Bitcoin Wiki, 2023
Why do I have to wait 10 minutes before I can spend money I received? [διαδικτυακό]
Πηγή: https://en.bitcoin.it/wiki/Help:FAQ
Πρόσβαση: 2/10/2023

[18] Ethereum Foundation, 2023
INTRO TO ETHEREUM [διαδικτυακό]
Πηγή: https://ethereum.org/en/developers/docs/intro-to-ethereum/
Πρόσβαση: 6/10/2023

[19] Google Cloud, 2023
What is a Virtual Machine? [διαδικτυακό]
Πηγή: https://cloud.google.com/learn/what-is-a-virtual-machine
Πρόσβαση: 8/10/2023

[20] Ethereum Foundation, 2023
ETHEREUM VIRTUAL MACHINE (EVM) [διαδικτυακό]
Πηγή: https://ethereum.org/en/developers/docs/evm/
Πρόσβαση: 8/10/2023

[21] Wikipedia, 2023
Smart Contract [διαδικτυακό]
Πηγή: https://en.wikipedia.org/wiki/Smart_contract
Πρόσβαση: 10/10/2023

[22] Ethereum Foundation, 2022
SMART CONTRACT LANGUAGES [διαδικτυακό]
Πηγή: https://ethereum.org/en/developers/docs/smart-contracts/languages/
Πρόσβαση: 10/10/2023

[23] J. Frankenfield, 2023
What Does Proof-of-Stake (PoS) Mean in Crypto [διαδικτυακό]
Πηγή: https://www.investopedia.com/terms/p/proof-stake-pos.asp
Πρόσβαση: 11/10/2023

[24] YCharts Inc, 2023
Ethereum Average Block Time (I:EBT) [διαδικτυακό]
Πηγή: https://ycharts.com/indicators/ethereum_average_block_time
Πρόσβαση: 13/10/2023

[25] Ethereum Foundation, 2023
Network feed [διαδικτυακό]
Πηγή: https://ethereum.org/en/gas/
Πρόσβαση: 18/10/2023

[26] Cointelegraph, 2023
Polygon blockchain explained: A beginner's guide to MATIC [διαδικτυακό]
Πηγή: https://cointelegraph.com/learn/polygon-blockchain-explained
Πρόσβαση: 18/10/2023

[27] M. Musharraf, 2021
What is the Blockchain Trilemma? [διαδικτυακό]
Πηγή: https://www.ledger.com/academy/what-is-the-blockchain-trilemma
Πρόσβαση: 18/10/2023

[28] PolygonScan, 2023
Polygon PoS Chain Average Block Time Chart [διαδικτυακό]
Πηγή: https://polygonscan.com/chart/blocktime
Πρόσβαση: 20/10/2023

[29] Ethereum Foundation, 2023
What is layer 2? [διαδικτυακό]
Πηγή: https://ethereum.org/en/layer-2/
Πρόσβαση: 20/10/2023

[30] Microsoft, 2023
Documentation for Visual Studio Code [διαδικτυακό]
Πηγή: https://code.visualstudio.com/docs
Πρόσβαση: 22/0/2023

[31] Mozilla, 2023
What is JavaScript [διαδικτυακό]
Πηγή: https://developer.mozilla.org/docs/Learn/JavaScript/First_steps/What_is_JavaScript
Πρόσβαση: 22/10/2023

[32] C. Kopecky, 2023
What is Node.js? A beginner's introduction to JavaScript runtime [διαδικτυακό]
Πηγή: https://www.educative.io/blog/what-is-nodejs
Πρόσβαση: 22/10/2023

[33] Readthedocs, 2023
web3.js – Ethereum JavaScript API [διαδικτυακό]
Πηγή: https://web3js.readthedocs.io/en/v1.10.0/
Πρόσβαση: 22/10/2023

[34] Solidity Team, 2023
Solidity [διαδικτυακό]
Πηγή: https://docs.soliditylang.org/en/v0.8.21/
Πρόσβαση: 22/10/2023

[35] Alchemy Insights Inc, 2022
Polygon's Mumbai Testnet: A Complete Guide [διαδικτυακό]
Πηγή: https://www.alchemy.com/overviews/mumbai-testnet
Πρόσβαση: 22/10/2023

[36] Wikipedia, 2023
PHP [διαδικτυακό]
Πηγή: https://en.wikipedia.org/wiki/PHP
Πρόσβαση: 22/10/2023

[37] J. Frankenfield, 2023
What Are Crypto Tokens, and How Do They Work? [διαδικτυακό]
Πηγή: https://www.investopedia.com/terms/c/crypto-token.asp
Πρόσβαση: 24/10/2023

[38] Ethereum Foundation, 2023
ERC-20 TOKEN STANDARD [διαδικτυακό]
Πηγή: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/
Πρόσβαση: 24/10/2023

[39] A. Hayes, 2023
Stablecoins: Definition, How They Work, and Types [διαδικτυακό]
Πηγή: https://www.investopedia.com/terms/s/stablecoin.asp
Πρόσβαση: 27/10/2023