Αυτόνομο κοινωνικό δίκτυο βασισμένο σε τεχνολογίες αποκέντρωσης

Νικολαΐδης Παναγιώτης[1] και Φανάκης Απόστολος[2]

[1]nkpanagi@auth.gr, 7491
[2]apostolof@auth.gr, 8261

Επιβλέπων Χρήστος Ε. Δημάκης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πολυτεχνική Σχολή

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Θεσσαλονίκη 2022

Περίληψη

Τις τελευταίες δεκαετίες, η ραγδαία ανάπτυξη του διαδικτύου μετέβαλε ριζικά τις ανθρώπινες κοινωνίες, μέσω μίας πληθώρας ψηφιακών εφαρμογών, οι οποίες, στη συντριπτική τους πλειοψηφία, προσφέρονται από παρόχους υπηρεσιών υπολογιστικού νέφους, ακολουθώντας την αρχιτεκτονική πελάτη-εξυπηρετητή.

Μολονότι αυτό το μοντέλο υλοποίησης έχει αποδειχθεί ιδιαίτερα λειτουργικό και έχει βελτιωθεί αξιοσημείωτα ανά τα χρόνια, η συγκεντρωτική του λογική συνοδεύεται από μία σειρά προβλημάτων. Πρώτον, οι χρήστες καλούνται να εμπιστευθούν τα προσωπικά τους δεδομένα σε μία εξωτερική οντότητα. Εκείνη, διατηρώντας τον πλήρη έλεγχο επί αυτών, αποκτάει τη δυνατότητα να τα επεξεργάζεται, να τα διαμοιράζεται και να τα λογοκρίνει, είτε για να εξυπηρετήσει τα συμφέροντά της, είτε για να συμμορφωθεί με άλλες αρχές που της ασκούν εξουσία. Επιπλέον, απουσιάζει η εγγύηση της διαθεσιμότητας των δεδομένων, καθώς, ανά πάσα στιγμή, ο εξυπηρετητής μπορεί να αποσυνδεθεί για αόριστο χρονικό διάστημα και λόγω ποικίλων αιτιών, όπως κάποιας κυβερνοεπίθεσης ή κάποιας φυσικής καταστροφής.

Αυτοί είναι μερικοί βασικοί λόγοι που συνετέλεσαν στην ταχεία ανάπτυξη ενός συνόλου καινοτόμων λογισμικών ανοιχτού κώδικα, τα οποία βασίζονται σε τεχνολογίες όπως το blockchain και τα δίκτυα ομότιμων κόμβων. Τα παραπάνω, αν και βρίσκονται σε σχετικά πρώιμο στάδιο, αποτελούν ήδη ισχυρά εργαλεία δημιουργίας κατανεμημένων και αποκεντρωμένων εφαρμογών.

Στόχος της παρούσας διπλωματικής εργασίας είναι η υλοποίηση μίας αυτόνομης κοινωνικής πλατφόρμας, η οποία, αξιοποιώντας τεχνολογίες αποκέντρωσης, αφενός θα επιστρέφει την κυριότητα των προσωπικών δεδομένων στον χρήστη, αφετέρου θα παρέχει τη δυνατότητα διενέργειας διαφανών δημοκρατικών ψηφοφοριών. Αυτά μέσα σε ένα πλαίσιο ανθεκτικό, τόσο σε σφάλματα και επιθέσεις, όσο και σε απόπειρες λογοκρισίας και παραποίησης.

Η αναπτυχθείσα πιλοτική εφαρμογή «Concordia» προσεγγίζει τον παραπάνω στόχο συνδυάζοντας τις τεχνολογίες Ethereum και IPFS, ώστε να ορίσει έναν αποκεντρωμένο ψηφιακό χώρο, τόσο σε αρχιτεκτονικό όσο και πολιτικό επίπεδο.

Λέξεις-Κλειδιά: Αποκεντροποίηση, Ethereum, Blockchain, Έξυπνο Συμβόλαιο, Αποκεντρωμένη Εφαρμογή, IPFS, OrbitDB, React, Redux, Jenkins

Abstract

In recent decades, the rapid growth of the internet has radically changed society, through a plethora of digital applications, the vast majority of which are offered by cloud computing service providers, following the client-server architecture.

Although this implementation model has proven to be highly functional and has improved significantly over the years, its centralized logic is accompanied by a number of problems. Firstly, users are required to trust their personal data to an external entity. Maintaining full control over them, the latter gains the ability to process, share and censor them, either to serve its own interests or to comply with other authorities in power. Furthermore, there is no guarantee of data availability, as, at any time, the server can be disconnected indefinitely and for a variety of reasons, such as a cyber attack or a natural disaster.

These are some of the key factors that have led to the rapid development of a wide range of innovating open source software, that are based on technologies such as blockchain and peer-to-peer networks. The aforementioned technologies, although at a relatively early stage, are already powerful tools for creating distributed and decentralized applications.

The goal of this thesis is the implementation of an autonomous social platform, which, by utilizing decentralization technologies, on the one hand will return the ownership of the data to the end user, on the other hand will provide transparent democratic voting processes. These in a context resistant to both faults and attacks, as well as attempts at censorship and falsification.

The developed proof of concept application "Concordia" approaches the above goal by combining Ethereum and IPFS, in order to define a digital space, that is decentralized both at architectural and political level.

Keywords: Decentralization, Ethereum, Blockchain, Smart Contract, Decentralized Application, IPFS, OrbitDB, React, Redux, Jenkins

Ευχαριστίες

Σε αυτό το σημείο θα θέλαμε να ευχαριστήσουμε εγκάρδια όλους εκείνους που συνέβαλλαν στην εκπόνηση της παρούσας εργασίας:

- Τον επιβλέποντα καθηγητή μας, κ. Δημάκη Χρήστο, για την ευκαιρία που μας έδωσε να ασχοληθούμε με το συγκεκριμένο θέμα και την εμπιστοσύνη που μας έδειξε από την αρχή μέχρι το τέλος.
- Τη Νικολαΐδου Μελίνα, για τη σχεδίαση του λογότυπου της εφαρμογής και τη δημιουργία σημαντικού τμήματος των σχημάτων του παρόντος εγγράφου.
- Τις οικογένειες και τους φίλους μας, για την αμέριστη υλική και ηθική υποστήριξη που μας προσέφεραν καθ' όλη τη διάρκεια των σπουδών μας.
- Ο ένας τον άλλον, για την άρτια επικοινωνία και συνεργασία, καθώς και για την υπομονή και την επιμονή, χαρακτηριστικά καθοριστικής σημασίας για την επιτυχή πορεία της διπλωματικής.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Η αλματώδης ανάπτυξη του διαδικτύου διαμόρφωσε ένα ολοκαίνουργιο τοπίο σε κάθε τομέα της ανθρώπινης δραστηριότητας, παρέχοντας ένα αναρίθμητο πλήθος εφαρμογών και υπηρεσιών. Τα μέσα κοινωνικής δικτύωσης, το ηλεκτρονικό ταχυδρομείο, η ψηφιακή ειδησεογραφία, ο διαμοιρασμός αρχείων και οι υπηρεσίες πολυμέσων ροής, αποτελούν ορισμένα από τα σημαντικότερα - και πλέον αναπόσπαστα - κομμάτια, που συνθέτουν την ψηφιακή πτυχή της σύγχρονης καθημερινότητας.

Κατά κύριο λόγο, το μοντέλο που ακολουθούν οι παραπάνω τεχνολογίες είναι αυτό της αρχιτεκτονικής πελάτη-εξυπηρετητή (client–server architecture) και προσφέρονται από παρόχους υπηρεσιών υπολογιστικού νέφους (cloud computing service providers). Αυτό σημαίνει ότι οι απαραίτητες λειτουργίες τους, δηλαδή η επεξεργασία (processing), η αποθήκευση των δεδομένων (storage) και το πρωτόκολλο επικοινωνίας (communication protocol) υλοποιούνται επί ενός συγκεντρωτικού (centralized) πλαισίου, κάτι που τους προσδίδει ορισμένα αξιοσημείωτα πλεονεκτήματα (π.χ. ευκολία ανάπτυξης, συντήρησης και αποσφαλμάτωσης).

Στις μέρες μας, ωστόσο, παρατηρείται παράλληλα μία τάση δημιουργίας εφαρμογών που ακολουθούν αποκεντρωτικά μοντέλα λειτουργίας, στα οποία το processing και το storage κατανέμονται σε ένα σύνολο κόμβων που επικοινωνούν ομότιμα. Εντός, λοιπόν, αυτής της τάσης, αναπτύσσονται με ταχείς ρυθμούς διάφορα λογισμικά, τα οποία συνθέτουν ένα νέο, αποκεντρωτικό οικοσύστημα. Αυτό περιλαμβάνει (μεταξύ άλλων) τόσο καινοτόμα πρωτόκολλα αποθήκευσης δεδομένων (π.χ. IPFS), όσο και πλατφόρμες ανάπτυξης και εκτέλεσης αποκεντρωμένων εφαρμογών (π.χ. Ethereum blockchain).

1.2 Περί αποκέντρωσης

Αν και ο όρος «αποκέντρωση» χρησιμοποιείται ευρέως στην επιστήμη των υπολογιστών και στα κρυπτοοικονομικά[1] (cryptoeconomics), συνήθως ορίζεται αρκετά ασαφώς[2]. Στην πραγματικότητα η αποκέντρωση (ή, αντίστοιχα, ο συγκεντρωτισμός) μπορεί να τοποθετηθεί πάνω σε τρεις ξεχωριστούς άξονες, οι οποίοι είναι σε γενικές γραμμές ανεξάρτητοι ο ένας από τον άλλον. Αυτοί έχουν ως εξής:

α) **Αρχιτεκτονική** αποκέντρωση: Από πόσους φυσικούς υπολογιστές αποτελείται ένα σύστημα; Πόσοι από αυτούς μπορούν, ανά πάσα στιγμή, να χαλάσουν και εκείνο να αντέξει;

β) **Πολιτική** αποκέντρωση: Πόσα άτομα ή οργανισμοί ελέγχουν τους υπολογιστές από τους οποίους αποτελείται το σύστημα;

γ) **Λογική** αποκέντρωση: Η διεπαφή και οι δομές δεδομένων του συστήματος μοιάζουν περισσότερο με ένα μονολιθικό αντικείμενο ή ένα άμορφο σμήνος; Αν, δηλαδή, το σύστημα (συμπεριλαμβανομένων των παρόχων και των χρηστών) «κοπεί στη μέση», θα συνεχίσουν τα δύο μισά να λειτουργούν πλήρως ως ανεξάρτητες μονάδες;

Για παράδειγμα, το BitTorrent είναι αποκεντρωτικό ως προς όλους τους άξονες, ενώ ένα CDN (Content Delivery Network), είναι μόνο αρχιτεκτονικά και λογικά, αφού ελέγχεται από κάποια εταιρεία. Φυσικά, η έννοια μπορεί να γενικευθεί και να μιλάμε για αποκέντρωση μίας επιχείρησης (συνήθως πλήρως συγκεντρωτική) ή μίας γλώσσας (συνήθως πλήρως αποκεντρωτική).

Η επιλογή της δομής ενός συστήματος ως προς την αποκέντρωσή του βασίζεται στις εκάστοτε ανάγκες και στους στόχους του. Μερικά ισχυρά πλεονεκτήματα που διατυπώνονται συχνά για τα αποκεντρωτικά συστήματα είναι τα εξής:

- **Ανοχή σε σφάλματα**: Τα αρχιτεκτονικά αποκεντρωμένα συστήματα είναι λιγότερο πιθανό να αποτύχουν τυχαία, επειδή βασίζονται σε πολλά ξεχωριστά στοιχεία που είναι απίθανο να παρουσιάσουν σφάλματα ταυτόχρονα.

- **Αντοχή σε επιθέσεις**: Το κόστος μίας επίθεσης, που έχει ως στόχο την καταστροφή ή τον χειρισμό ενός αποκεντρωτικού συστήματος, είναι πολύ ακριβό. Αυτό συμβαίνει επειδή δεν υπάρχει κάποιο ευαίσθητο κεντρικό σημείο στο οποίο να μπορεί να πραγματοποιηθεί μία επίθεση, η οποία να έχει κόστος πολύ χαμηλότερο από το οικονομικό μέγεθος του περιβάλλοντος συστήματος.

- **Απουσία ανάγκης εκχώρησης εμπιστοσύνης**: Σε ένα ιδανικό πολιτικά αποκεντρωμένο σύστημα οι χρήστες δε χρειάζεται να εμπιστεύονται κάποια κεντρική αρχή για την επεξεργασία και την αποθήκευση των δεδομένων.

[1] Τα «κρυπτοοικονομικά» είναι η πρακτική επιστήμη της δημιουργίας κατανεμημένων συστημάτων, οι ιδιότητες των οποίων εξασφαλίζονται με οικονομικά κίνητρα, ενώ οι οικονομικοί τους μηχανισμοί είναι κρυπτογραφικά εγγυημένοι.[1]

- **Αντίσταση σε συμπαιγνίες**: είναι πολύ πιο δύσκολο για τους συμμετέχοντες σε αποκεντρωμένα συστήματα να συνεργαστούν για να ενεργήσουν με τρόπο που τους ωφελεί σε βάρος άλλων συμμετεχόντων.

Ιδιαίτερα τα τελευταία χρόνια, παρατηρείται μία έντονη ανάγκη υλοποίησης αποκεντρωμένων εφαρμογών (decentralized applications), οι οποίες, πέρα από τα αρχιτεκτονικά πλεονεκτήματα που τις χαρακτηρίζουν (π.χ. σταθερότητα, ασφάλεια, επεκτασιμότητα), αποσκοπούν στην επίτευξη πολιτικής αποκέντρωσης. Αυτό πηγάζει τόσο από την ανάγκη προάσπισης των αρχών που καταστρατηγούνται όταν τα δεδομένα υπάγονται στον έλεγχο κάποιας κεντρικής διαχείρισης (π.χ. της ελευθερίας του λόγου, της ανωνυμίας και της ιδιωτικότητας του χρήστη), όσο και από την ανάγκη δημιουργίας διαδικασιών που απαιτούν εγκυρότητα και αυθεντικότητα, όπως όσων σχετίζονται με την αυτοδιαχείριση και την άμεση δημοκρατία. Ως απόγειο των παραπάνω, μπορούν να θεωρηθούν οι λεγόμενες *αποκεντρωτικές αυτόνομες οργανώσεις* (decentralized autonomous organizations ή DAOs), οι οποίες αποτελούν μία μορφή αλγοκρατικής[2] οργάνωσης βασισμένης σε τεχνολογίες αποκέντρωσης και, κυρίως, στο blockchain.

1.3 Ορισμός του προβλήματος

Οι περισσότερες διαδεδομένες πλατφόρμες επικοινωνίας (κοινωνικά δίκτυα, mailing lists, forums κ.ά.) είναι ως επί το πλείστον συγκεντρωτικής μορφής, πράγμα το οποίο καθιστά αναγκαία την ύπαρξη κεντρικών αρχών που να τις διαχειρίζονται και να τις συντηρούν.

Παρά τα θετικά της χαρακτηριστικά, η κεντροποιημένη λογική ενός τέτοιου συστήματος αφενός συνοδεύεται από ποικίλα μειονεκτήματα τεχνικής φύσεως (αρχιτεκτονικός συγκεντρωτισμός), αφετέρου εγείρει σοβαρούς προβληματισμούς σχετικά με τη διαχείριση των προσωπικών δεδομένων των χρηστών από τις κεντρικές αρχές (πολιτικός συγκεντρωτισμός). Τα βασικότερα από τα παραπάνω θα μπορούσαν να συνοψιστούν ως εξής:

- Έλλειψη **ασφάλειας**: Τα προσωπικά δεδομένα των χρηστών μπορεί να υποκλαπούν εξαιτίας κάποιας κυβερνοεπίθεσης.
- Έλλειψη **διαθεσιμότητας**: Το σύστημα μπορεί να σταματήσει να λειτουργεί προσωρινά ή μόνιμα για τεχνικούς, οικονομικούς ή νομικούς λόγους.
- Έλλειψη **εμπιστοσύνης**: Οι κεντρικές αρχές έχουν τη δυνατότητα να παρακολουθούν τους χρήστες, να διαβάζουν, ή ακόμα και να διαρρέουν τα προσωπικά τους δεδομένα εν αγνοία των τελευταίων. Οι δε χρήστες δε διαθέτουν κανέναν τρόπο με τον οποίον να μπορούν να τις εμπιστευθούν με βεβαιότητα.

[2] Ο όρος «αλγοκρατία» (algocracy) αναφέρεται σε εναλλακτικές μορφές διακυβέρνησης που βασίζονται στη χρήση αλγορίθμων.[3]

- Έλλειψη εγγύησης της **αυθεντικότητας** των δεδομένων: Οι κεντρικές αρχές έχουν τη δυνατότητα να τροποποιούν τα δεδομένα κατά βούληση κάτι που έχει ως αποτέλεσμα να μην υπάρχει εγγύηση ως προς την αυθεντικότητα όσων βλέπουν οι χρήστες.
- Έλλειψη εγγύησης της **ελευθερίας του λόγου**: Οι κεντρικές αρχές έχουν τη δυνατότητα να λογοκρίνουν τα δεδομένα, είτε βάσει των συμφερόντων τους, είτε βάσει υποχρεώσεών τους προς τρίτους.

Επιπλέον, όπως γίνεται φανερό, οι αδυναμίες του συστήματος ως προς τον πολιτικό άξονα το καθιστούν ακατάλληλο να παρέχει στους χρήστες αυθεντικές και επικυρώσιμες δημοκρατικές διαδικασίες. Τέτοιου είδους διαδικασίες θα μπορούσε να ήταν από απλές ψηφοφορίες, μέχρι σύνθετες διαδικασίες αυτοδιαχείρισης της πλατφόρμας.

1.4 Στόχος της διπλωματικής

Στόχος της παρούσας διπλωματικής εργασίας είναι η δημιουργία μίας αυτόνομης κοινωνικής πλατφόρμας, η οποία, βασιζόμενη σε τεχνολογίες αποκέντρωσης, θα λειτουργεί ανεξάρτητα από κεντρικές αρχές, παρέχοντας στους χρήστες της πλήρη ελευθερία του λόγου και κυριότητα επί των δεδομένων τους. Παράλληλα, θα παρέχει μία πλατφόρμα για ανώνυμες και αυθεντικές ψηφοφορίες, εν δυνάμει ικανών να αποτελέσουν ένα έγκυρο, έμπιστο και άμεσα δημοκρατικό βήμα λήψης αποφάσεων.

Η proof of concept (PoC) εφαρμογή που αναπτύχθηκε για την επίτευξη του παραπάνω στόχου ονομάζεται Concordia[3] και λειτουργεί μέσω ενός συνδυασμού αποκεντρωτικών τεχνολογιών. Πιο συγκεκριμένα, στον επεξεργαστικό πυρήνα της και σαν σημείο αναφοράς αξιοποιεί το Ethereum blockchain, ενώ για την αποθήκευση του μεγαλύτερου όγκου των δεδομένων χρησιμοποιεί το IPFS μέσω της OrbitDB. Η δε διεπαφή του χρήστη υλοποιείται με σύγχρονες μεθόδους web development σε JavaScript (React, Redux κ.ά.).

1.5 Μεθοδολογία της διπλωματικής

Έναυσμα της παρούσας εργασίας αποτέλεσε η παρατηρήση της αρχιτεκτονικής δομής των σύγχρονων διαδικτυακών εφαρμογών και η ανάγκη διερεύνησης των επιπτώσεών της στον τελικό χρήστη.

Αρχικά, ορίστηκε με σαφήνεια το πρόβλημα (ενότητα 1.3) και ο στόχος της διπλωματικής (ενότητα 1.4), λαμβάνοντας την απόφαση να περιοριστεί στον τομέα των μέσων κοινωνικής δικτύωσης και της ψηφιακής δημοκρατίας.

[3]Η Concordia είναι η θεά της αρχαίας Ρωμαϊκής θρησκείας που προσωποποιεί την ομόνοια. Στην ελληνική μυθολογία ταυτίζεται με τη θεότητα Ομόνοια ή τη θεά Αρμονία.

Στη συνέχεια, πραγματοποιήθηκε έρευνα του χώρου των αποκεντρωμένων τεχνολογιών και ξεκίνησε η διαδικασία της σχεδίασης της εφαρμογής, μέσω της επιλογής του μοντέλου της τεχνολογικής στοίβας και του κατάλληλου λογισμικού σε κάθε επίπεδό της.

Ακολούθησε η διαδικασία υλοποίησης της πιλοτικής πλατφόρμας Concordia, με στόχο να καταστεί ο αρχικός σχεδιασμός πραγματοποιήσιμος.

Τέλος, εξήχθησαν συμπεράσματα και διατυπώθηκαν πιθανές μελλοντικές επεκτάσεις για την εφαρμογή.

1.6 Τυπογραφικές παραδοχές

Το παρόν έγγραφο αποτυπώνεται με τη γραμματοσειρά Linux Libertine O[4], ενώ για τα κομμάτια κώδικα χρησιμοποιείται η Hack[5]. Το μέγεθος του κυρίως κειμένου είναι 12pt και το διάστιχό του είναι επαυξημένο του προκαθορισμένου κατά το ήμισυ για άνεση κατά την ανάγνωση.

Καταβάλλεται η μέγιστη δυνατή προσπάθεια για τη χρήση ελληνικών όρων, όπου αυτό είναι εφικτό, με τους αντίστοιχους αγγλικούς να τους συνοδεύουν σε ακόλουθες παρενθέσεις. Τα εισαγωγικά που χρησιμοποιούνται είναι τα διπλά γωνιώδη (« »), τόσο για ελληνικούς, όσο και για ξενόγλωσσους χαρακτηρισμούς.

Επίσης, αριθμούνται επί της συνολικής έκτασης της εργασίας οι λεζάντες των σχημάτων και των πινάκων, οι υποσημειώσεις και οι βιβλιογραφικές αναφορές, με τις τελευταίες να παρατίθενται στο τέλος του εγγράφου.

Τέλος, επισημαίνεται ότι η συγγραφή της αναφοράς πραγματοποιήθηκε στο ηλεκτρονικό τυπογραφικό σύστημα LaTeX. Ο πηγαίος της κώδικας μπορεί να βρεθεί στο αντίστοιχο αποθετήριο κώδικα της διπλωματικής εργασίας[6].

1.7 Οργάνωση κεφαλαίων

Η παρούσα διπλωματική εργασία οργανώνεται σε κεφάλαια, ενότητες και υποενότητες, όπως αυτά διατυπώνονται στα Περιεχόμενα. Πιο συγκεκριμένα:

- Στο εισαγωγικό **Κεφάλαιο 1** γίνεται μία σύντομη ανάλυση του όρου «αποκέντρωση», μία περιγραφή του προβλήματος και μία παρουσίαση του στόχου της εργασίας.
- Το **Κεφάλαιο 2** σχετίζεται με το θεωρητικό υπόβαθρο, το οποίο περιλαμβάνει όλες τις έννοιες που είναι απαραίτητες για την κατανόηση των διαδικασιών της σχεδίασης και της υλοποίησης της εφαρμογής.
- Στο **Κεφάλαιο 3** αναλύεται η διαδικασία της σχεδίασης της εφαρμογής.

[4] `https://libertine-fonts.org/`
[5] `https://sourcefoundry.org/hack/`
[6] `https://gitlab.com/ecentrics/thesis-report`.

- Στο **Κεφάλαιο 4** περιγράφεται η διαδικασία υλοποίησης της πιλοτικής εφαρμογής Concordia.
- Στο **Κεφάλαιο 5** παρουσιάζονται τα συμπεράσματα της εργασίας (5.1), καθώς και διάφορες πιθανές μελλοντικές επεκτάσεις (5.2).
- Το **Παράρτημα Α΄** περιέχει στιγμιότυπα οθόνης της υλοποιημένης εφαρμογής.
- Το **Παράρτημα Β΄** περιλαμβάνει πίνακες με στατιστικά του αναπτυχθέντα κώδικα.
- Τέλος, παρατίθενται οι **βιβλιογραφικές αναφορές** που χρησιμοποιήθηκαν στο κείμενο.

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

2.1 Συναρτήσεις κατακερματισμού

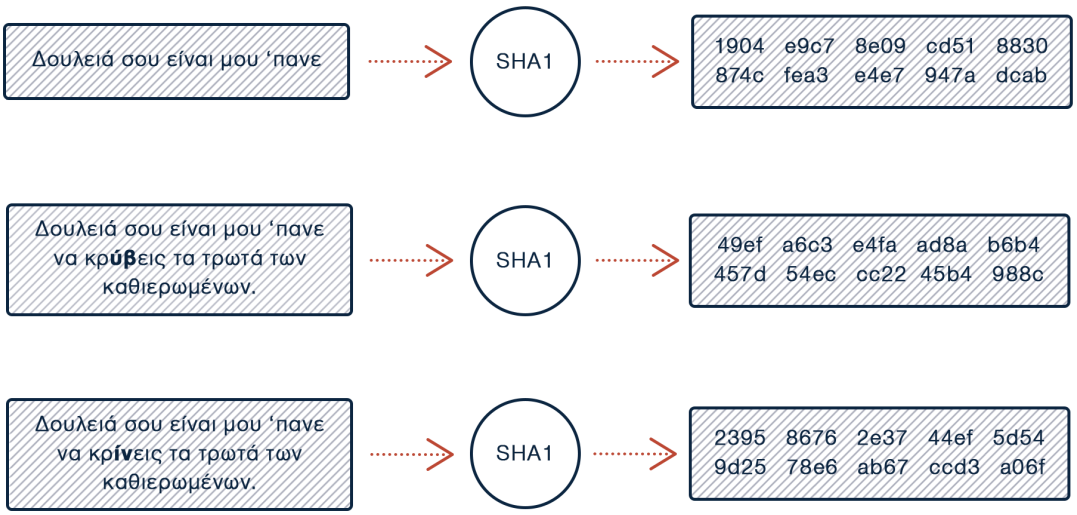
Οι κρυπτογραφικές συναρτήσεις κατακερματισμού (cryptographic hash functions) είναι ειδική κατηγορία συναρτήσεων κατακερματισμού σχεδιασμένες για χρήση στην κρυπτογραφία. Αποτελούν μαθηματικές συναρτήσεις που δέχονται ως είσοδο δεδομένα τυχαίου μεγέθους και επιστρέφουν συμβολοσειρές σταθερού μήκους.

Σχήμα 2.1: Λειτουργία συνάρτησης κατακερματισμού

Οι τιμές που επιστρέφει η συνάρτηση κατακερματισμού ονομάζονται τιμές κατακερματισμού (hash values, digests ή απλά hashes). Μία ιδανική κρυπτογραφική συνάρτηση κατακερματισμού έχει τις εξής βασικές ιδιότητες:

- Είναι ντετερμινιστική, δηλαδή η ίδια είσοδος παράγει πάντα την ίδια έξοδο.
- Είναι μη αντιστρέψιμη, δηλαδή είναι πρακτικά ανέφικτο να υπολογιστεί η είσοδος δεδομένης μιας εξόδου.
- Είναι αμφιμονοσήμαντη (1-1), δηλαδή σε δύο διαφορετικές εισόδους αντιστοιχούν πάντα δύο εντελώς διαφορετικές έξοδοι.

- Είναι αποδοτική, δηλαδή ο υπολογισμός του hash οποιασδήποτε εισόδου είναι γρήγορος.



Σχήμα 2.2: Παράδειγμα λειτουργίας συνάρτησης κατακερματισμού

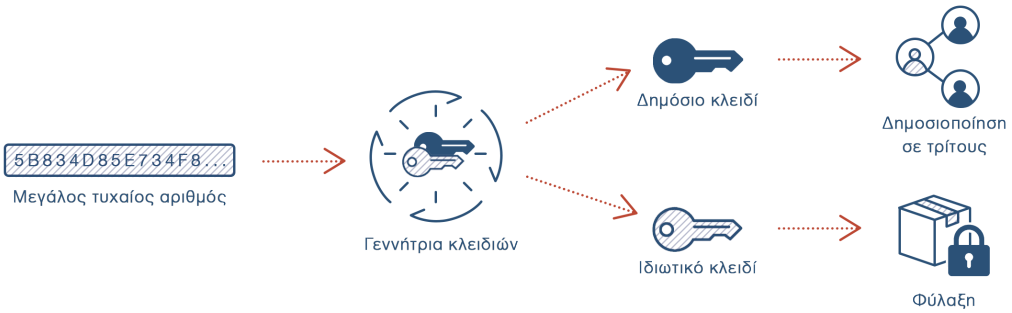
Μία από τις δημοφιλέστερες οικογένειες κρυπτογραφικών αλγορίθμων κατακερματισμού είναι αυτή των Secure Hash Algorithms (SHA), η οποία περιλαμβάνει τους SHA-0, SHA-1, SHA-2 και SHA-3.

2.2 Ασύμμετρη κρυπτογραφία

Η ασύμμετρη κρυπτογραφία (asymmetric cryptography) ή κρυπτογραφία δημόσιου κλειδιού (public-key cryptography) αποτελεί κρυπτογραφικό σύστημα που βασίζεται στη χρήση ενός ζεύγους κλειδιών (key pair), του *δημόσιου* (public key) και του *ιδιωτικού* (private key). Αυτά τα κλειδιά είναι μαθηματικά συνδεδεμένα ως εξής:

- Το ιδιωτικό κλειδί δε μπορεί να προκύψει γνωρίζοντας το δημόσιό του
- Ό,τι κρυπτογραφηθεί από το ένα μπορεί να αποκρυπτογραφηθεί μόνο από το άλλο

Η δημιουργία ενός ζεύγους κλειδιών επιτυγχάνεται μέσω μιας *γεννήτριας κλειδιών* (key generation function), η οποία χρησιμοποιεί ειδικούς αλγορίθμους (π.χ. RSA), δεχόμενη ως είσοδο έναν τυχαίο αριθμό. Από τα παραχθέντα κλειδιά, το δημόσιο γνωστοποιείται σε τρίτους, ενώ το ιδιωτικό παραμένει μυστικό.



Σχήμα 2.3: Παραγωγή ασύμμετρου ζεύγους κλειδιών

Ο χρήστης μπορεί να χρησιμοποιήσει τα κλειδιά για δύο βασικούς σκοπούς:

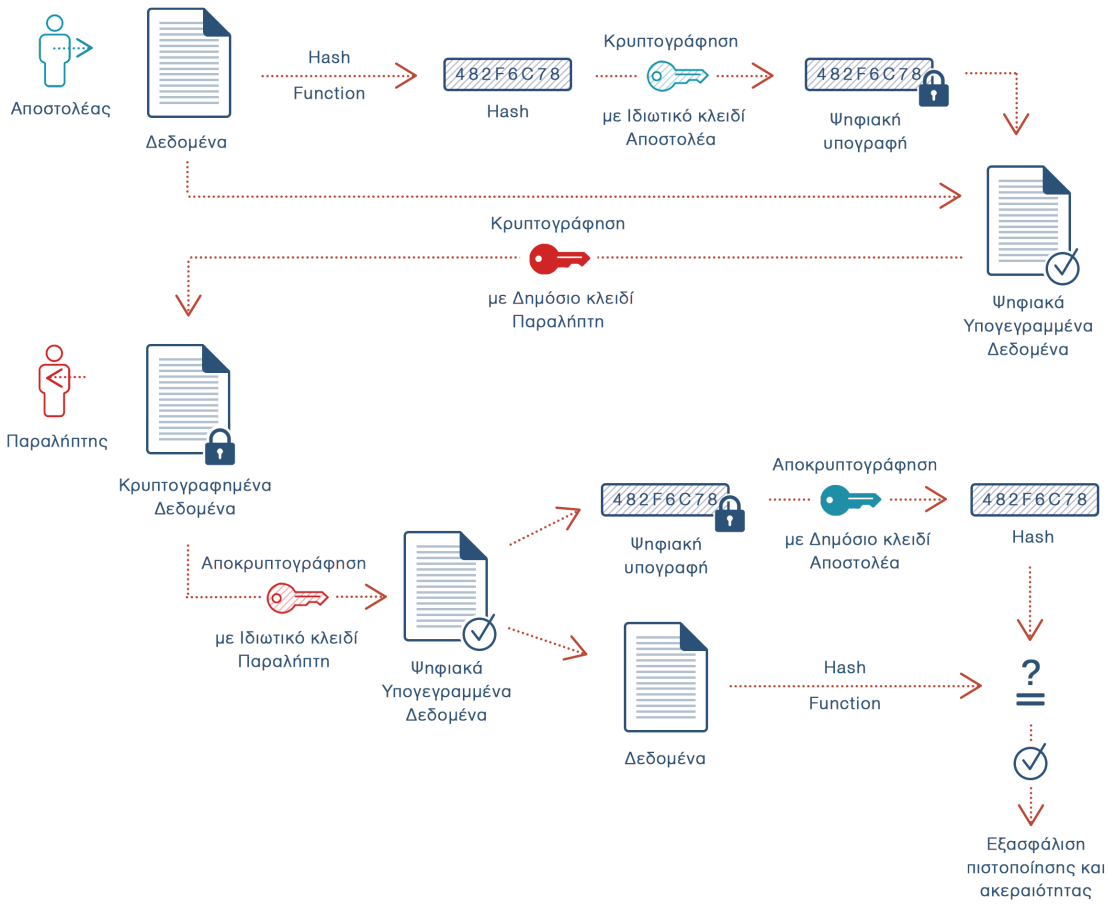
α) Για να αποκρυπτογραφήσει μηνύματα άλλων χρηστών, οι οποίοι τα κρυπτογράφησαν χρησιμοποιώντας το δημόσιο κλειδί του. Με αυτόν τον τρόπο εξασφαλίζεται η *εμπιστευτικότητα* (confidentiality).

β) Για να υπογράψει ψηφιακά ένα μήνυμα, κρυπτογραφώντας το hash των δεδομένων του με το ιδιωτικό του κλειδί. Έτσι, ο παραλήπτης του μηνύματος μπορεί μέσω της ληφθείσας *ψηφιακής υπογραφής* (digital signature):

   i. Να επαληθεύσει την ταυτότητα του αποστολέα, αποκρυπτογραφώντας επιτυχώς την ψηφιακή υπογραφή με το δημόσιο κλειδί του τελευταίου. Εξασφαλίζεται έτσι η *πιστοποίηση* (authenticity) της προέλευσης των δεδομένων.

   ii. Να επιβεβαιώσει ότι το μήνυμα έφτασε αναλλοίωτο, εφόσον το hash των δεδομένων συμπίπτει με το hash εντός της ψηφιακής υπογραφής. Με αυτόν τον τρόπο εξασφαλίζεται η *ακεραιότητα* (integrity) των δεδομένων.

Με τον συνδυασμό των παραπάνω, λέμε ότι δύο χρήστες μπορούν να επικοινωνούν μεταξύ τους με *κρυπτογράφηση απ' άκρη σ' άκρη* (end to end encryption).



Σχήμα 2.4: Κρυπτογράφηση απ' άκρη σ' άκρη

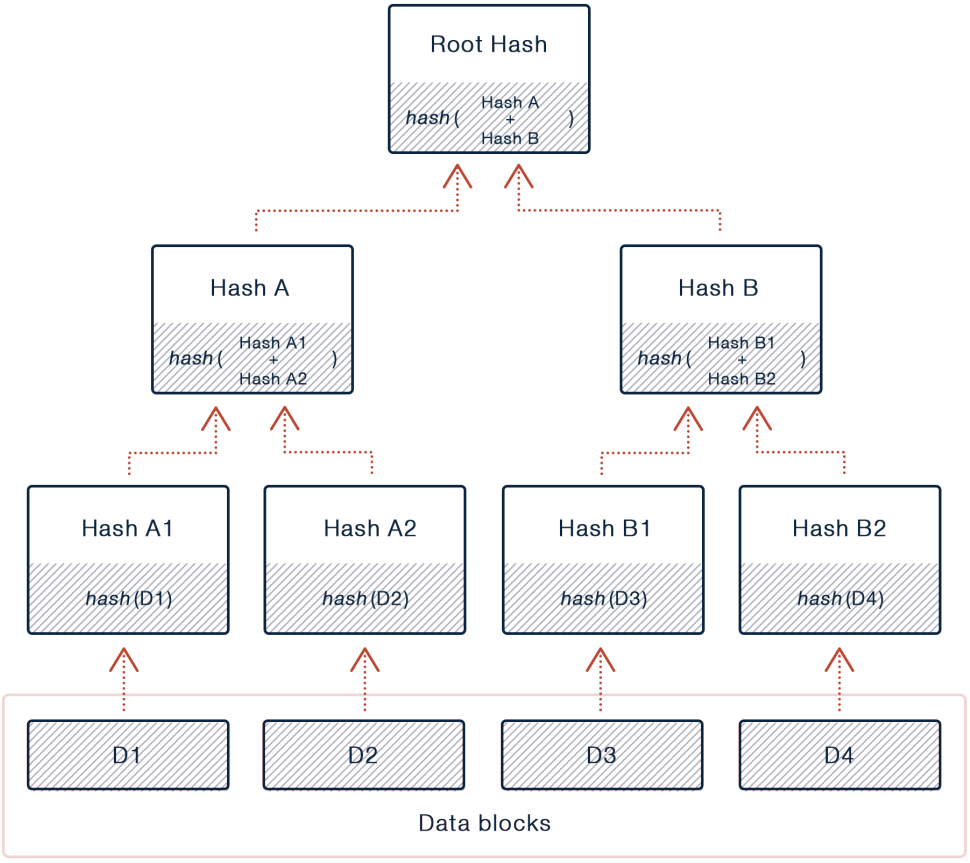
Μία προσέγγιση στην κρυπτογραφία δημόσιου κλειδιού είναι η κρυπτογραφία ελλειπτικής καμπύλης (Elliptic-Curve Cryptography ή ECC). Η ECC βασίζεται στην αλγεβρική δομή των ελλειπτικών καμπυλών σε πεπερασμένα πεδία και υπερέχει της non-EC κρυπτογραφίας, καθώς επιτρέπει τη δημιουργία μικρότερων κλειδιών με ισοδύναμη ασφάλεια. Ένα από τα πρωτόκολλά της είναι ο Elliptic Curve Digital Signature Algorithm (ECDSA), ο οποίος χρησιμοποιείται για την ψηφιακή υπογραφή δεδομένων και αποτελεί το EC-ανάλογο του DSA (Digital Signature Algorithm).[4]

2.3 Δένδρα Merkle

Ένα δένδρο Merkle (Merkle tree ή hash tree) είναι μία δενδρική δομή δεδομένων, η οποία απαρτίζεται από φύλλα (leaf nodes) που περιέχουν hashes από blocks δεδομένων και από άλλους κόμβους (non-leaf nodes) που περιέχουν τα hashes των θυγατρικών τους. Στην κορυφή του δένδρου βρίσκεται ο ριζικός κόμβος με το λεγόμενο root hash.[5]

Η πιο συνηθισμένη υλοποίηση είναι το δυαδικό (binary) δένδρο Merkle, το οποίο περιλαμ-

βάνει δύο θυγατρικούς κόμβους (child nodes) κάτω από κάθε γονικό (non-leaf) κόμβο, και είναι αυτό που αναλύεται στη συνέχεια.



Σχήμα 2.5: Παράδειγμα δυαδικού δένδρου Merkle

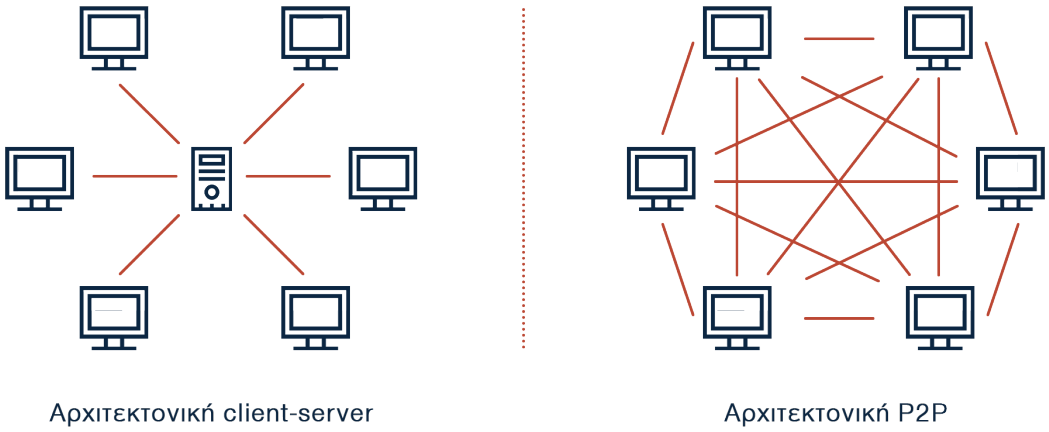
Τα Merkle trees επιτρέπουν την αποδοτική και ασφαλή επαλήθευση των περιεχομένων που ανήκουν σε σετ δεδομένων μεγάλου μεγέθους. Η βασική ιδιότητα είναι ότι για κάθε σετ δεδομένων υπάρχει ακριβώς ένα πιθανό δένδρο, το οποίο δε γίνεται να τροποποιηθεί χωρίς να αλλάξει ταυτόχρονα και το root hash.

Έτσι, μέσω των λεγόμενων Merkle proofs, μπορούμε:

- Να αποφανθούμε εάν κάποια δεδομένα ανήκουν στο δένδρο, υπολογίζοντας (για το εκάστοτε leaf node) hashes πλήθους ανάλογου του λογαρίθμου του συνολοικού αριθμού των φύλλων.
- Να αποδείξουμε συνοπτικά την εγκυρότητα ενός τμήματος κάποιου σετ δεδομένων, χωρίς να χρειαστεί να αποθηκεύσουμε ολόκληρο το σύνολο δεδομένων.
- Να διασφαλίσουμε την εγκυρότητα ενός συγκεκριμένου συνόλου δεδομένων εντός ενός μεγαλύτερου σύνολου, χωρίς να χρειαστεί να αποκαλυφθεί το περιεχόμενο οποιουδήποτε εκ των δύο.[6]

2.4 Δίκτυα Ομότιμων Κόμβων

Τα δίκτυα ομότιμων κόμβων ή Peer-to-Peer (P2P) networks αποτελούν μία κατανεμημένη αρχιτεκτονική δικτύων, οι συμμετέχοντες (κόμβοι) της οποίας μοιράζονται ένα τμήμα των πόρων τους, με στόχο την παροχή κάποιας υπηρεσίας (π.χ. τον διαμοιρασμό περιεχομένου). Εν αντιθέσει με συγκεντρωτικά δίκτυα τύπου client-server, οι κόμβοι (nodes) έχουν απευθείας πρόσβαση στους πόρους, χωρίς τη διαμεσολάβηση ενδιάμεσων οντοτήτων. Οι συμμετέχοντες ενός τέτοιου δικτύου είναι, δηλαδή, ταυτόχρονα, τόσο πάροχοι, όσο και αιτούντες των πόρων και της παρεχόμενης υπηρεσίας.[7]



Σχήμα 2.6: Αρχιτεκτονικές δικτύων client-server και P2P

Τα P2P networks μπορούν να χωριστούν σε δύο κατηγορίες:

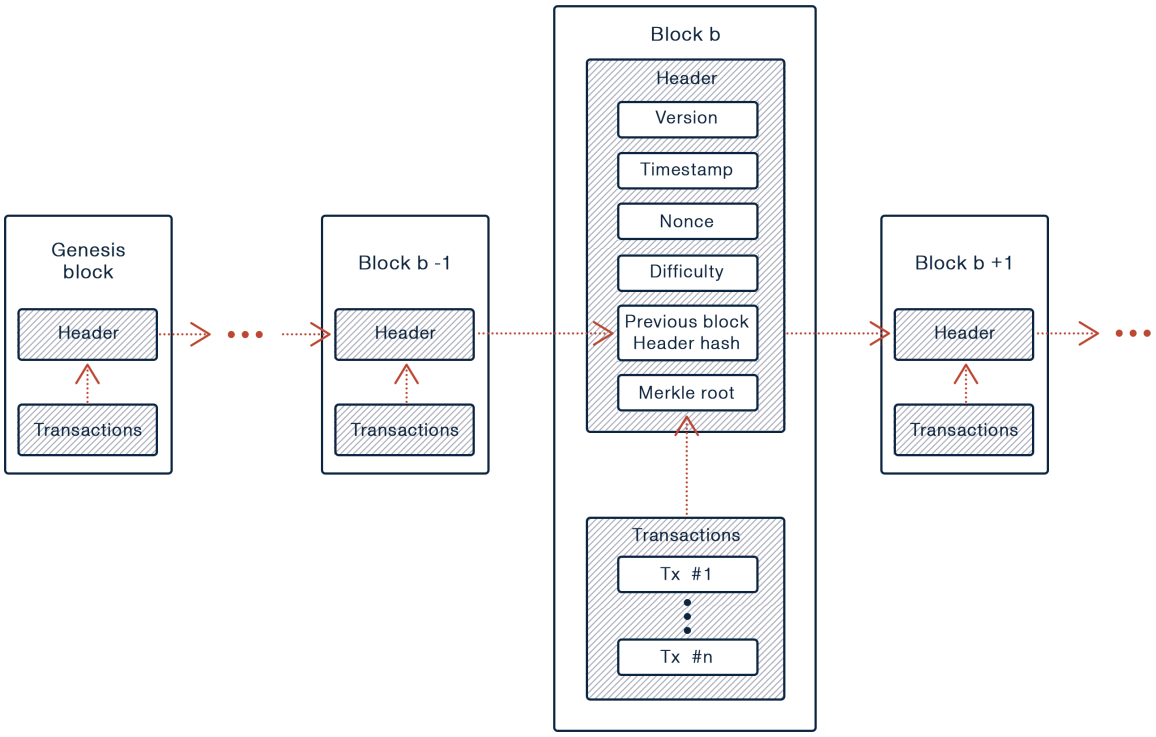
- Στα «Καθαρά» (Pure) P2P networks, στα οποία ισχύει ότι η αφαίρεση ενός τυχαίου κόμβου από το δίκτυο δεν προκαλεί κάποιο πρόβλημα σε αυτό.
- Στα «Υβριδικά» (Hybrid) P2P networks, στα οποία συμμετέχουν επιπλέον και κεντρικές οντότητες, παρέχοντας απαραίτητα τμήματα των προσφερόμενων υπηρεσιών.

Από εδώ και στο εξής, εάν δεν αναφέρεται ρητά η κατηγορία κάποιου P2P network, θα εννοείται ότι ανήκει στην πρώτη.

2.5 Blockchain

Το blockchain αποτελεί μία διανεμημένη δημόσια σειρά δεδομένων, που διατηρεί έναν αμετάβλητο ως προς το ιστορικό του κατάλογο (immutable ledger) ψηφιακών συναλλαγών (digital transactions) ενός αγαθού (asset), π.χ. ενός νομίσματος (currency ή token). Περιγράφηκε για πρώτη φορά το 2008 από ένα άτομο (ή μία ομάδα ανθρώπων) με το ψευδώνυμο Satoshi Nakamoto, αποτελώντας τη βάση του κρυπτονομίσματος (cryptocurrency) Bitcoin.[8]

Δομικό στοιχείο του blockchain είναι το μπλοκ (block), το οποίο περιέχει μία ομάδα έγκυρων συναλλαγών που έχουν κατακερματιστεί και κωδικοποιηθεί σε ένα δένδρο Merkle, το hash του προηγούμενου μπλοκ και μερικά ακόμα μεταδεδομένα (π.χ. nonce, timestamp). Έτσι, κάθε νέο μπλοκ «δείχνει» στο προηγούμενό του μέσω του hash, επιβεβαιώνοντας την ακεραιότητά του, με τα διαδεχόμενα μπλοκ να σχηματίζουν τελικά μία αλυσίδα, μέχρι το αρχικό μπλοκ, το οποίο είναι γνωστό ως το μπλοκ γένεσης (genesis block).[9]



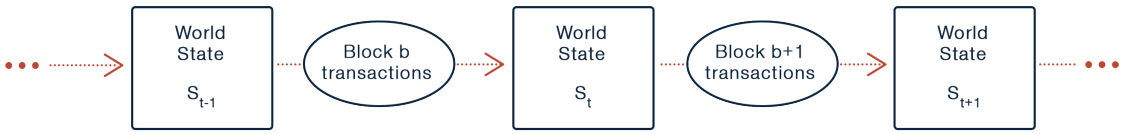
Σχήμα 2.7: Blockchain ως αλυσίδα από block

Ως προς την κυριότητα επί αυτού, το blockchain συνήθως[7] δεν ελέγχεται από κάποια κεντρική οντότητα, αλλά διατηρείται από ένα δημόσιο P2P δίκτυο. Οι κόμβοι (nodes) του δικτύου συμμορφώνονται συλλογικά με ένα πρωτόκολλο συναίνεσης (consensus) για την επικοινωνία και την επικύρωση νέων μπλοκ. Για παράδειγμα, στο Bitcoin, το consensus επιτυγχάνεται μέσω ενός Proof of Work (PoW) αλγορίθμου, όπου οι κόμβοι (miners) ανταγωνίζονται ο ένας τον άλλον για το ποιος θα λύσει πρώτος ένα σύνθετο αλγοριθμικό πρόβλημα που συσχετίζεται με το εκάστοτε block. Αυτός που θα τα καταφέρει επιβραβεύεται για την επεξεργαστική ισχύ που δαπάνησε με ένα ποσό από bitcoin. Εκείνα είναι εν μέρει νέα νομίσματα που κόβονται ή «εξορύσσονται» εκείνη τη στιγμή (όπως ορίζεται από το πρωτόκολλο), αλλά και όσα τέλη (fees) κατέβαλαν οι κόμβοι για να πραγματοποιήσουν τις συναλλαγές του μπλοκ. Αξίζει να σημειωθεί πως δεν είναι αναγκαίο να διαθέτει κανείς ολόκληρο το blockchain (το οποίο είναι ιδιαίτερα ογκώδες) - δηλαδή έναν πλήρη κόμβο - για να επικοινωνήσει με το δίκτυο, αλλά αρκεί να χρησιμοποιήσει έναν light node που απλά να αναμεταδίδει τις επιθυμητές συναλλαγές.

[7] Υπάρχουν και κάποιες υλοποιήσεις ιδιωτικών blockchain που, όμως, δε θα μας απασχολήσουν.

Η διευθυνσιοδότηση σε ένα blockchain επιτυγχάνεται αξιοποιώντας την κρυπτογραφία δημόσιου κλειδιού. Το πρωτόκολλο του εκάστοτε blockchain ορίζει έναν αλγόριθμο για την παραγωγή ζευγών κλειδιών (π.χ. ECDSA στο Bitcoin). Το δημόσιο από αυτά ορίζει τη διεύθυνση, ενώ το ιδιωτικό παραμένει μυστικό, υπό την κατοχή του χρήστη. Με αυτό τον τρόπο προκύπτει ένα πρακτικά ανεξάντλητο πλήθος πιθανών έγκυρων δημόσιων διευθύνσεων (π.χ. 2^{160} για το Bitcoin), στις οποίες οι χρήστες μπορούν να στέλνουν και να λαμβάνουν ποσά του εκάστοτε κρυπτονομίσματος. Για την αποστολή ενός ποσού, δηλώνουν το fee που επιθυμούν να καταβάλουν και υπογράφουν την επιθυμητή συναλλαγή με το ιδιωτικό τους κλειδί. Η συναλλαγή αναμεταδίδεται στο δίκτυο και παραμένει στο transaction pool μέχρις ότου να γίνει αποδεκτή και να συμπεριληφθεί στο επόμενο block.

Από τεχνική σκοπιά, το blockchain μπορεί να θεωρηθεί ως μία μηχανή καταστάσεων βασισμένη σε συναλλαγές (transaction-based state machine). Δηλαδή, ξεκινάει από μία αρχική κατάσταση (genesis state), η οποία τροποποιείται σταδιακά με κάθε block, και περιλαμβάνει ανά πάσα στιγμή τις διευθύνσεις με τα ποσά των νομισμάτων που τις αντιστοιχούν.



Σχήμα 2.8: Blockchain ως μηχανή καταστάσεων

Σύμφωνα με την ανάλυση της ενότητας 1.2, το blockchain είναι αρχιτεκτονικά και πολιτικά αποκεντρωτικό, καθώς δε διαθέτει δομικά κάποιο κεντρικό σημείο αποτυχίας, ούτε ελέγχεται από κάποιον. Ωστόσο, είναι λογικά συγκεντρωτικό, αφού υπάρχει μία κοινά αποδεκτή κατάσταση και το σύστημα συμπεριφέρεται μακροσκοπικά ως ένας ενιαίος υπολογιστής.

2.6 Ethereum

Σχήμα 2.9: Ethereum logo

Το Ethereum είναι ένα δημόσιο blockchain ανοιχτού κώδικα με εγγενές κρυπτονόμισμα το Ether (ETH). Παρέχει μία προγραμματιστική πλατφόρμα με ενσωματωμένη μία Turing-complete γλώσσα προγραμματισμού, που μπορεί να χρησιμοποιηθεί για τη δημιουργία αποκεντρωμένων εφαρμογών (Decentralized Applications ή DApps) μέσω της χρήσης «έξυπνων συμβολαίων» (smart contracts).[10]

2.6.1 Λογαριασμοί

Στο Ethereum blockchain οι λογαριασμοί αποτελούν οντότητες οι οποίες μπορούν να δέχονται, να κρατούν και να στέλνουν ETH και tokens, καθώς και να αλληλεπιδρούν με smart contracts.[11] Χωρίζονται σε δύο κατηγορίες: στους λογαριασμούς εξωτερικής κατοχής (externally owned accounts ή EOAs) και στους λογαριασμούς συμβολαίων (contract accounts).

Οι λογαριασμοί εξωτερικής κατοχής δημιουργούνται χωρίς κόστος και ελέγχονται μέσω ιδιωτικών κλειδιών. Μπορούν να πραγματοποιούν μόνο συναλλαγές μεταφοράς ETH ή κάποιου token.

Από την άλλη, οι λογαριασμοί συμβολαίων απαιτούν κάποιο κόστος δημιουργίας, καθώς χρησιμοποιούν αποθηκευτικό χώρο επί του blockchain, ενώ ελέγχονται αποκλειστικά από τον κώδικά τους. Οι συναλλαγές που πραγματοποιούν προς άλλους λογαριασμούς είναι μόνο σαν αντίδραση σε μία εισερχόμενη συναλλαγή, όπως ορίζει ο κώδικας του smart contract τους.

Πιο αναλυτικά, τα πεδία που διαθέτουν οι λογαριασμοί στο Ethereum είναι τα εξής:

- Το **nonce**: ένας μετρητής που υποδεικνύει τον αριθμό των απεσταλμένων συναλλαγών του λογαριασμού. Αυτό διασφαλίζει ότι οι συναλλαγές διεκπεραιώνονται μόνο μία φορά. Σε έναν λογαριασμό συμβολαίου, αυτός ο αριθμός αντιπροσωπεύει τον αριθμό των συμβολαίων που δημιουργήθηκαν από εκείνον.
- Το **balance**: το ποσό σε ETH που διαθέτει ο λογαριασμός, μετρημένο σε wei (1 ETH = 10^{18} wei).
- Το **codeHash**: ένα hash που αναφέρεται στον κώδικα του λογαριασμού (contract code). Είναι χαρακτηριστικό των λογαριασμών συμβολαίων (στους λογαριασμούς εξωτερικής κατοχής είναι hash μίας κενής συμβολοσειράς) και, σε αντίθεση με τα άλλα πεδία, αφότου οριστεί παραμένει αμετάβλητο.
- Το **storageRoot**: το root hash του δένδρου Merkle των αποθηκευμένων δεδομένων του smart contract, εφόσον πρόκειται για έναν λογαριασμό συμβολαίου (δε χρησιμοποιείται σε λογαριασμούς εξωτερικής κατοχής).

Η δημιουργία των λογαριασμών επιτυγχάνεται μέσω της παραγωγής ενός ζεύγους κλειδιών, αξιοποιώντας τον ECDSA (βλ. ενότητα 2.2). Έτσι, το ιδιωτικό κλειδί χρησιμοποιείται για να υπογράφονται ψηφιακά οι συναλλαγές, ενώ το δημόσιο ορίζει τη δημόσια διεύθυνση του λογαριασμού (είναι της μορφής «0x + τα 20 τελευταία bytes του Keccak-256[8] hash του δημόσιου κλειδιού»).

Κατά κύριο λόγο, οι χρήστες παράγουν και διαχειρίζονται τα ιδιωτικά κλειδιά των λογαριασμών εξωτερικής κατοχής μέσω ενός πορτοφολιού (wallet). Τα wallets αποτελούν εφαρμογές, οι οποίες δείχνουν το balance του λογαριασμού και παρέχουν τη δυνατότητα αποστολής/ λήψης

[8]Ο Keccak-256 αποτελεί τον κρυπτογραφικό αλγόριθμο κατακερματισμού που χρησιμοποιείται στο Ethereum. Πρόκειται για τον αλγόριθμο SHA3-256 πριν την τυποποίησή του από το NIST.

ETH και tokens από/ σε αυτόν. Ορισμένα προτοφόλια προσφέρουν περαιτέρω λειτουργίες, σημαντικότερη εκ των οποίων είναι η διασύνδεση του λογαριασμού με αποκεντρωμένες εφαρμογές. Επί του παρόντος, το πιο διαδεδομένο πορτοφόλι τέτοιου τύπου είναι το MetaMask[9].

2.6.2 Smart Contracts

Με λίγα λόγια, τα smart contracts αποτελούν αυτόνομα κομμάτια κώδικα, τα οποία είναι αποθηκευμένα στο blockchain και ενεργοποιούνται μέσω συναλλαγών. Κληρονομούν ιδιότητες του blockchain, όπως τη διαφάνεια (transparency), την επαληθευσιμότητα (verifiability) και την αμεταβλητότητα (immutability).

Ένα παράδειγμα της καθημερινότητας που μπορεί να παρομοιασθεί λειτουργικά με smart contract είναι ο αυτόματος πωλητής.[12] Ένας αυτόματος πωλητής ορίζεται ως ένα αυτόνομο μηχάνημα που διανέμει αγαθά ή παρέχει υπηρεσίες, όταν εισαχθεί σε αυτόν κάποιο χρηματικό ποσό και επιλεχθεί ένα διαθέσιμο προϊόν. Οι αυτόματοι πωλητές είναι προγραμματισμένοι να εκτελούν συγκεκριμένους κανόνες, που θα μπορούσαν να οριστούν σε ένα συμβόλαιο. Με όμοιο τρόπο, σε ένα smart contract μπορούν να κωδικοποιηθούν αυθαίρετες συναρτήσεις μετάβασης, επιτρέποντας τη δημιουργία μίας πληθώρας αποκεντρωμένων εφαρμογών.

Όπως προαναφέρθηκε στην υποενότητα 2.6.1, τα smart contracts εντάσσονται σε contract accounts και απαιτούν την καταβολή τελών για τη δημιουργία τους, αφού χρειάζεται να εγγράψουν επί του blockchain τον κώδικα και τα αρχικά δεδομένα τους.

Επιπλέον, ο κώδικάς τους ενεργοποιείται μονάχα όταν δεχθούν μία έγκυρη συναλλαγή από κάποιον άλλον λογαριασμό (είτε εξωτερικής κατοχής, είτε συμβολαίου). Η συναλλαγή αυτή εμπεριέχει, πέρα από το απαιτούμενο fee, ένα συνοδευτικό μήνυμα με πληροφορίες σχετικές με τη συνάρτηση που πρέπει να εκτελεστεί. Η δε συνάρτηση μπορεί να πραγματοποιεί ποικίλες διαφορετικές ενέργειες, όπως την ανάγνωση και την εγγραφή στην εσωτερική αποθήκευση, την πραγματοποίηση νέων συναλλαγών, ή, ακόμα, τη δημιουργία νέων συμβολαίων.

Η σύνταξη των smart contracts γίνεται κατά βάση σε γλώσσες υψηλού επιπέδου και, συγκεκριμένα, στις Solidity, Vyper και Fe[10]. Πριν την εγγραφή τους στο blockchain, μεταγλωττίζονται σε εμηνεύσιμο από την EVM bytecode, η οποία είναι υπεύθυνη για την αποθήκευση και την εκτέλεσή του (βλ. υποενότητα 2.6.5).

2.6.3 DApps

Οι DApps στο οικοσύστημα του Ethereum είναι εφαρμογές οι οποίες συνδυάζουν smart contracts και frontend UIs. Είναι ντετερμινιστικές, Turing-complete και εκτελούνται απομονωμένα στην EVM.[11]

Πέρα από τα θετικά χαρακτηριστικά των DApps που αναλύθηκαν στην ενότητα 1.2 (ανοχή σε σφάλματα, αντοχή σε επιθέσεις, απουσία ανάγκης εκχώρησης εμπιστοσύνης, αντίσταση σε

[9] `https://metamask.io/`

[10] `https://soliditylang.org/`, `https://github.com/vyperlang/vyper` και `https://fe-lang.org/`

συμπαιγνίες), τα Ethereum DApps διαθέτουν επιπλέον όλα τα πλεονεκτήματα των blockchain και των smart contract, όπως μηδενικό downtime, πλήρη ακεραιότητα δεδομένων και επαληθεύσιμη συμπεριφορά.

Ωστόσο, χαρακτηρίζονται και από μία σειρά αξιοσημείωτων μειονεκτημάτων, όπως τα παρακάτω:

- Δυσκολία συντήρησης: Συντηρούνται δυσκολότερα από τις συγκεντρωτικές εφαρμογές, εξαιτίας της αμεταβλητότητας του κώδικα και των δεδομένων επί του blockchain.
- Επιβάρυνση απόδοσης: Υπάρχει τεράστια επιβάρυνση απόδοσης (performance overhead) και η κλιμάκωση (scaling) είναι πολύ δύσκολη, καθώς απαιτείται όλοι οι κόμβοι να εκτελούν και να αποθηκεύουν όλες τις συναλλαγές.
- Συμφόρηση δικτύου: Επί του παρόντος, το δίκτυο μπορεί να επεξεργαστεί μόνο περίπου 10-15 συναλλαγές ανά δευτερόλεπτο. Εάν οι συναλλαγές αποστέλλονται με ταχύτερο ρυθμό από αυτόν, θα αυξάνονται παράλληλα και οι μη επιβεβαιωμένες συναλλαγές που αναμένουν να εκτελεστούν.
- Κακή εμπειρία χρήστη: Επί του παρόντος, είναι δύσκολο για τον μέσο τελικό χρήστη να αλληλεπιδράσει με το blockchain με ευκολία και ασφάλεια, καθώς απαιτούνται ενέργειες όπως η εγκατάσταση ειδικών εργαλείων για τη διασύνδεση με αυτό, η δημιουργία wallet, η διαφύλαξη του ιδιωτικού του κλειδιού και η προσθήκη ETH για την εξόφληση των τελών.

Παρ' όλα τα μειονεκτήματα, τα οποία μετριάζονται με τον καιρό μέσω συνεχών αναβαθμίσεων της πλατφόρμας, υπάρχει ήδη ένα ευρύ φάσμα εφαρμογών που μπορούν να υλοποιηθούν στο Ethereum, αξιοποιώντας τα ισχυρά του πλεονεκτήματα. Οι εφαρμογές αυτές μπορούν να διακριθούν σε τρεις κατηγορίες:

α) Οικονομικές εφαρμογές, οι οποίες παρέχουν στους χρήστες ισχυρούς τρόπους διαχείρισης και σύναψης συμβάσεων χρησιμοποιώντας τα χρήματά τους. Αυτό περιλαμβάνει υπονομίσματα, χρηματοοικονομικά παράγωγα, συμβάσεις αντιστάθμισης κινδύνου, πορτοφόλια αποταμίευσης, διαθήκες και ακόμα και ορισμένες κατηγορίες συμβάσεων εργασίας πλήρους κλίμακας.

β) Ημι-οικονομικές εφαρμογές, όπου εμπλέκονται χρήματα, αλλά η λειτουργία τους εμπεριέχει παράλληλα και μία αξιοσημείωτη μη νομισματική πλευρά. Ένα τέτοιο παράδειγμα είναι οι αυτόματες πληρωμές για λύσεις σε υπολογιστικά προβλήματα (βλ. Gitcoin).

γ) Μη οικονομικές εφαρμογές, όπως η αποκεντρωμένη αποθήκευση δεδομένων, συστήματα ταυτότητας (identity) και φήμης (reputation), διαδικτυακές ψηφοφορίες και αποκεντρωμένη διακυβέρνηση. Οι τελευταίες εισάγουν και την έννοια των Αποκεντρωμένων Αυτόνομων Οργανώσεων (Decentralized Autonomous Organizations ή DAOs), οντοτήτων που ενεργούν αυτόνομα, χωρίς την ανάγκη διαμεσολάβησης κάποιας συγκεντρωτικής (centralized) αρχής.[10]

2.6.4 Tokens

Η λέξη «token» προέρχεται από τη λέξη «tācen» των Παλαιών Αγγλικών και σημαίνει σημάδι ή σύμβολο. Στην καθημερινότητα, ο όρος χρησιμοποιείται, κατά κύριο λόγο, για την περιγραφή αντικειμένων ειδικού σκοπού, τα οποία μοιάζουν με κέρματα και έχουν ασήμαντη εγγενή αξία (π.χ. μάρκες παιχνιδιών). Συνήθως, η χρήση των tokens περιορίζεται σε συγκεκριμένες επιχειρήσεις, οργανισμούς ή τοποθεσίες, πράγμα το οποίο σημαίνει ότι δεν ανταλλάσσονται εύκολα και τυπικά έχουν μόνο μία λειτουργία.[13]

Ωστόσο, στο Ethereum blockchain η έννοια των tokens επεκτείνεται και επαναπροσδιορίζεται. Πλέον δεν υπάρχουν περιορισμοί χρήσης και ιδιοκτησίας, ενώ η αξία τους ποικίλει, ανάλογα με το τι αντιπροσωπεύουν (π.χ. νομίσματα, περιουσιακά στοιχεία, ή δικαιώματα πρόσβασης). Μπορούν, δε, να ανταλλαχθούν σε παγκόσμιο επίπεδο, για άλλα tokens ή για άλλα νομίσματα.

Τα tokens που ορίζονται σε Ethereum smart contracts μπορούν να έχουν μία ή περισσότερες από τις παρακάτω χρήσεις:

- Νόμισμα (currency)
- Πόρος (resource), π.χ. για το διαμοιρασμό CPU ή storage εντός ενός δικτύου
- Περιουσιακό στοιχείο (asset), εγγενούς ή εξωγενούς, υλικού ή άυλου (π.χ. χρυσός, πετρέλαιο, ενέργεια, ακίνητο)
- Πρόσβαση (access), ψηφιακή ή φυσική (π.χ. forum, δωμάτιο ξενοδοχείου)
- Μετοχικό κεφάλαιο (equity) ενός ψηφιακού οργανισμού (π.χ. μίας DAO) ή μίας νομικής οντότητας (π.χ. μίας εταιρείας)
- Ψηφοφορία (voting), παρέχοντας δικαιώματα ψήφου σε ένα ψηφιακό ή νομικό σύστημα
- Συλλεκτικό (collectible), ψηφιακό ή φυσικό
- Ταυτότητα (identity), ψηφιακής ή νομικής φύσεως
- Πιστοποίηση (attestation), π.χ. πτυχίο, πιστοποιητικό γέννησης
- Χρησιμότητα (utility), για πρόσβαση ή πληρωμή μίας υπηρεσίας

Ένα σημαντικό κριτήριο κατηγοριοποίησης των tokens είναι η εναλλαξιμότητα (fungibility) τους. Ένα token είναι εναλλάξιμο όταν οποιαδήποτε μονάδα του μπορεί να αντικατασταθεί με μία άλλη χωρίς καμία διαφορά στην αξία ή τη λειτουργία του. Από την άλλη πλευρά, ένα non-fungible token (NFT) αντιπροσωπεύει ένα μοναδικό υλικό ή άυλο στοιχείο και, επομένως, είναι μη εναλλάξιμο.

Τέλος, τα tokens συχνά ακολουθούν κάποια καθορισμένα standards στην υλοποίησή τους. Τα δημοφιλέστερα από αυτά είναι τα ERC-20 και ERC-777 (για fungible tokens), το ERC-721 (για NFTs) και το ERC-1155 (για semi-fungible tokens ή SFTs).

2.6.5 EVM

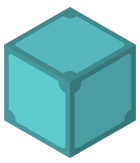
Τα smart contracts (και, κατ' επέκταση, οι DApps) εκτελούνται από την εικονική μηχανή του Ethereum (Ethereum Virtual Machine ή EVM). Η EVM αποτελεί μία quasi[11]–Turing-complete μηχανή καταστάσεων αρχιτεκτονικής βασισμένης σε στοίβα (stack-based architecture). Σε υψηλό επίπεδο, η EVM μπορεί να θεωρηθεί ως ένας παγκόσμιος αποκεντρωμένος υπολογιστής που περιέχει εκατομμύρια εκτελέσιμα αντικείμενα, το καθένα με τη δική του μόνιμη αποθήκη δεδομένων.

Η EVM αποθηκεύει όλες τις τιμές της μνήμης σε μια στοίβα και λειτουργεί με μέγεθος λέξης 256 bit, κυρίως για τη διευκόλυνση των εγγενών λειτουργιών κατακερματισμού και ελλειπτικής καμπύλης. Διαθέτει ένα σύνολο διευθυνσιοδοτήσιμων στοιχείων δεδομένων:

- Έναν αμετάβλητο κώδικα προγράμματος σε εικονική μνήμη ROM, φορτωμένο με τον bytecode του smart contract προς εκτέλεση.
- Μία πτητική (volatile) μνήμη, με κάθε θέση ρητά αρχικοποιημένη στο μηδέν.
- Ένας χώρος μόνιμης αποθήκευσης, που είναι μέρος του Ethereum state, επίσης αρχικά μηδενισμένος.

Υπάρχει, επίσης, ένα σύνολο μεταβλητών περιβάλλοντος και δεδομένων, τα οποία είναι διαθέσιμα κατά την εκτέλεση.[13]

2.7 IPFS



Σχήμα 2.10: IPFS logo

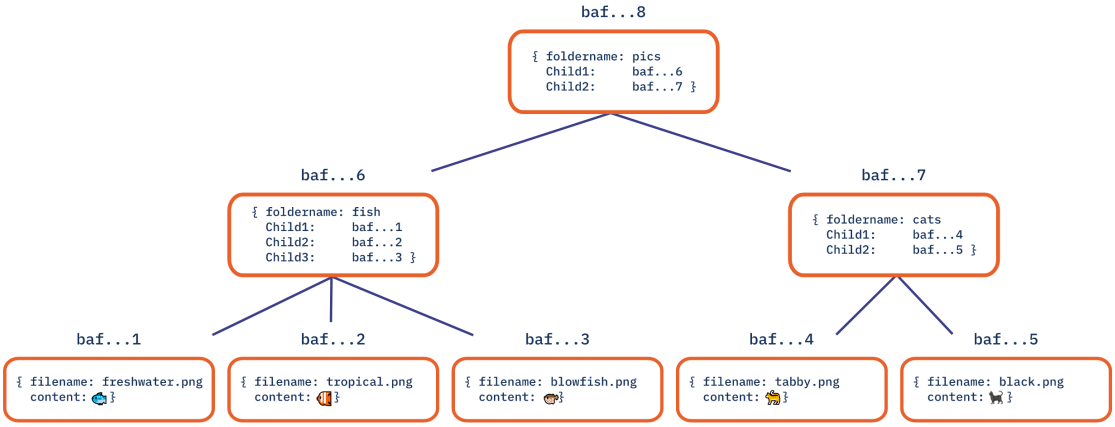
Το IPFS (InterPlanetary File System) είναι *ένα P2P πρωτόκολλο υπερμέσων, σχεδιασμένο για να διατηρήσει και να αυξήσει τη γνώση της ανθρωπότητας κάνοντας το διαδίκτυο αναβαθμίσιμο, ανθεκτικό και πιο ανοιχτό.*[14] Πρακτικά πρόκειται για ένα κατανεμημένο σύστημα για αποθήκευση και πρόσβαση σε αρχεία, ιστότοπους, εφαρμογές και δεδομένα. Το περιεχόμενο είναι προσβάσιμο μέσω ενός δικτύου ομότιμων κόμβων που βρίσκονται οπουδήποτε στον κόσμο, οι οποίοι ενδέχεται να να αποθηκεύουν πληροφορία, να τη μεταφέρουν (relay nodes) ή και τα δύο.[15]

Ο τρόπος λειτουργίας του IPFS βασίζεται στα εξής:

[11] «Quasi» («σχεδόν») επειδή όλες οι διαδικασίες εκτέλεσης περιορίζονται σε έναν πεπερασμένο αριθμό υπολογιστικών βημάτων από την ποσότητα gas που είναι διαθέσιμη για οποιαδήποτε εκτέλεση ενός smart contract.

- **Μοναδική ταυτοποίηση μέσω διευθυνσιοδότησης περιεχομένου (content addressing)**. Το περιεχόμενο δεν προσδιορίζεται από την τοποθεσία του (π.χ. https://...), αλλά από το τι περιλαμβάνει. Κάθε κομμάτι περιεχομένου έχει ένα μοναδικό αναγνωριστικό (Content ID ή CID), το οποίο είναι το hash του σε μορφή multihash (`https://multiformats.io/multihash/`).

- **Σύνδεση περιεχομένου μέσω κατευθυνόμενων άκυκλων γράφων (Directed Acyclic Graphs ή DAGs)**. Το IPFS αξιοποιεί DAGs (και συγκεκριμένα Merkle DAGs), μίας δομής δεδομένων της οποίας κάθε κόμβος έχει ως μοναδικό αναγνωριστικό το hash του περιεχομένου του (το CID).



Σχήμα 2.11: Παράδειγμα Merkle DAG[12]

Στο παραπάνω Merkle DAG τα baf...i αποτελούν τα CID των αρχείων και των φακέλων. Το δένδρο δημιουργείται από κάτω προς τα πάνω, υπολογίζοντας κάθε φορά τα κατάλληλα hashes/CIDs. Σε περίπτωση που το περιεχόμενο ενός κόμβου αλλάξει, τότε αλλάζει τόσο το CID του, όσο και τα CIDs όλων των προγόνων του.

- **Ανακάλυψη περιεχομένου μέσω κατανεμημένων πινάκων κατακερματισμού (Distributed Hash Tables ή DHTs)**. Ο DHT είναι ένα κατανεμημένο σύστημα για την αντιστοίχιση κλειδιών σε τιμές. Στο IPFS αποτελεί το θεμελιώδες συστατικό του συστήματος δρομολόγησης περιεχομένου και λειτουργεί ως διασταύρωση μεταξύ καταλόγου και συστήματος πλοήγησης. Πρακτικά πρόκειται για ένα πίνακα που αποθηκεύει ποιος έχει ποια δεδομένα και, μέσω του οποίου, ο χρήστης βρίσκει τον peer που έχει αποθηκευμένο το επιθυμητό περιεχόμενο.

Κάτι που θα πρέπει να σημειωθεί είναι πως, σαν προεπιλογή, οι IPFS κόμβοι αντιμετωπίζουν τα αποθηκευμένα δεδομένα ως προσωρινή μνήμη (cache), πράγμα που σημαίνει ότι δεν υπάρχει καμία εγγύηση ότι εκείνα θα συνεχίσουν να παραμένουν αποθηκευμένα σε αυτούς. Για την αποφυγή της διαγραφής τους από τον garbage collector, τα δεδομένα θα πρέπει να σημανθούν

[12] `https://proto.school/merkle-dags/`

ως σημαντικά, μέσω του «καρφιτσώματος» (pinning). Έτσι, για την ομαλή λειτουργία π.χ. μίας DApp που βασίζεται στο IPFS, θα πρέπει το περιεχόμενό της να είναι pinned από αρκετούς peers και/ή να γίνεται χρήση κάποιου pinning service, ώστε να εξασφαλίζεται διαθεσιμότητά του.

Βασικό πλεονέκτημα του IPFS είναι ότι ο αποκεντρωτικός του χαρακτήρας δίνει τη δυνατότητα να παρέχεται το περιεχόμενο από πολλούς κόμβους, οι οποίοι βρίσκονται σε διαφορετικές τοποθεσίες και δεν υπάγονται σε κάποια συγκεκριμένη κεντρική αρχή. Με αυτόν τον τρόπο, τα δεδομένα είναι πιο ανθεκτικά τόσο από άποψη διαθεσιμότητας (αν ένας κόμβος αποσυνδεθεί, θα υπάρχει κάποιος άλλος), όσο και από άποψη αντοχής στη λογοκρισία. Μπορεί, επίσης, να ανακτώνται γρηγορότερα, εφόσον τα διαθέτουν κάποιοι κοντινοί peers, πράγμα ιδιαίτερα πολύτιμο για κοινότητες που είναι καλά δικτυωμένες τοπικά αλλά δεν έχουν καλή σύνδεση με το ευρύτερο διαδίκτυο.

Κεφάλαιο 3

Σχεδίαση εφαρμογής

Σε αυτό το κεφάλαιο περιγράφεται η διαδικασία σχεδίασης της εφαρμογής Concordia, από τη σύλληψη της ιδέας και την επιλογή της τεχνολογικής στοίβας, μέχρι τον ορισμό της αρχιτεκτονικής της και τον διαχωρισμό του προγραμματιστικού έργου σε sprints.

3.1 Σύλληψη της ιδέας

Η σύλληψη της ιδέας για τη δημιουργία της εφαρμογής της παρούσας διπλωματικής εργασίας έχει ως εφαλτήριο την αναγνώριση ενός διδιάστατου προβλήματος.

Η πρώτη διάσταση εστιάζει στον χώρο των μέσων κοινωνικής δικτύωσης. Εκεί παρατηρείται αδιαμφισβήτητη επικράτηση πλατφορμών επικοινωνίας συγκεντρωτικής μορφής (π.χ. Facebook, Twitter, Instagram), ενώ προσπάθειες δημιουργίας αντίστοιχων αποκεντρωτικών εφαρμογών βρίσκονται σε πρώιμα στάδια, τόσο ανάπτυξης, όσο και υιοθέτησης από το ευρύ κοινό. Όπως αναλύθηκε και στην ενότητα 1.3, η τρέχουσα αυτή κατάσταση θέτει αξιοσημείωτα προβλήματα τεχνικής φύσεως (έλλειψη ασφάλειας και διαθεσιμότητας) και, κυρίως, πολιτικής (έλλειψη εμπιστοσύνης, εγγύησης της αυθεντικότητας των δεδομένων και της ελευθερίας του λόγου).

Η δεύτερη διάσταση εστιάζει στον χώρο της ψηφιακής δημοκρατίας (digital democracy). Συγκεκριμένα, παρατηρείται έλλειψη εργαλείων, ικανών να παρέχουν τη δυνατότητα διενέργειας αυθεντικών δημοκρατικών διαδικασιών. Ψηφοφορίες και αυτοδιαχείριση εντός συστημάτων κεντροποιημένης λογικής αδυνατούν, για αρχιτεκτονικούς λόγους, να εξασφαλίσουν τις απαραίτητες θεμελιώδεις ιδιότητες τέτοιων διαδικασιών, δηλαδή της ανωνυμίας και της επαληθευσιμότητας.

Αυτές οι παρατηρήσεις αποτέλεσαν την έμπνευση για τη δημιουργία μίας εφαρμογής, η οποία, μέσω ενός προτεινόμενου συνδυασμού αποκεντρωτικών τεχνολογιών, να ορίσει έναν ψηφιακό χώρο που θα έρθει αντιμέτωπος με το παραπάνω πρόβλημα. Έτσι, κεντρικός στόχος της πιλοτικής εφαρμογής Concordia, είναι να αποτελέσει μία αυτόνομη κοινωνική πλατφόρμα, που θα κατοχυρώνει στους χρήστες της ελευθερία του λόγου και πλήρη κυριότητα επί των δεδομένων τους. Επιπλέον, θα παρέχει τη δυνατότητα διενέργειας αυθεντικών, ανώνυμων ψηφοφοριών, κάτι που θα την καθιστά ένα αξιόπιστο δημοκρατικό βήμα για τη λήψη αποφάσεων εντός των

αυτοδιαχειριζόμενων κοινοτήτων της.

3.2 Τεχνολογική στοίβα

Ξεκινώντας τη σχεδίαση της πλατφόρμας, πραγματοποιήθηκε έρευνα για την επιλογή της τεχνολογικής της στοίβας (technology stack). Αυτή αποφασίστηκε να ακολουθήσει μία προσαρμοσμένη για τα δεδομένα μορφή τριμερούς διάταξης[13] και να χωριστεί σε τρία λογικά επίπεδα (tiers):

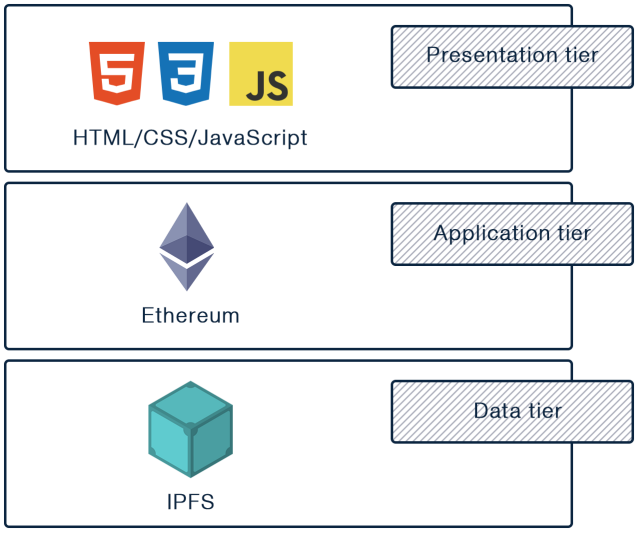
α) **Presentation tier**: Αποτελεί τη διεπαφή του χρήστη (user interface ή UI), μέσω της οποίας ο τελευταίος αλληλεπιδρά με την εφαρμογή. Για την εκπλήρωση των προδιαγραφών, το μοναδικό απαραίτητο χαρακτηριστικό αυτού του τμήματος είναι να μπορεί να εκτελείται αυτούσιο από τη συσκευή του τελικού χρήστη, δηλαδή να μην απαιτείται η ύπαρξη κάποιου εξυπηρετητή για τη λειτουργία του. Λαμβάνοντας, επιπροσθέτως, υπόψιν τις ανάγκες και τους περιορισμούς των λογισμικών των άλλων δύο επιπέδων, το παρόν κομμάτι αποφασίστηκε να σχεδιαστεί ως μία client-side web application σε HTML/CSS/JavaScript.

β) **Application tier**: Πρόκειται για το επίπεδο που πραγματοποιεί την επεξεργασία (processing) της εφαρμογής. Εδώ επιλέχθηκαν το blockchain και τα smart contracts, καθώς τα πλεονεκτήματά τους, όπως αυτά περιγράφηκαν στο κεφάλαιο 2, αρμόζουν απόλυτα με τις ιδιαίτερες απαιτήσεις της εφαρμογής. Συγκεκριμένα, επιλέχθηκε η πλατφόρμα του Ethereum, καθώς αποτελεί τον πρωτοπόρο στο χώρο, διαθέτοντας την ισχυρότερη κοινότητα και την δυνατότητα δημιουργίας πλήρως λειτουργικών εφαρμογών.

γ) **Data tier**: Το τμήμα αυτό είναι υπεύθυνο για την αποθήκευση του κύριου όγκου των δεδομένων (storage). Για την επίτευξη πλήρους αρχιτεκτονικής αποκέντρωσης των δεδομένων επιλέχθηκε το IPFS (βλ. ενότητα 2.7), το οποίο διανέμει το περιεχόμενο της εφαρμογής στους peers που συμμετέχουν σε αυτήν, χωρίς να απαιτεί κάποιο κεντρικό σημείο. Έτσι, κάθε χρήστης θα έχει πλήρη κυριότητα επί των δεδομένων του, ενώ, επιπλέον, θα συμμετέχει στην πλατφόρμα διαμοιράζοντας τα δεδομένα άλλων χρηστών.

Τελικά, με τη διασύνδεση των προαναφερθέντων τεχνολογιών, προκύπτει σχηματικά η ακόλουθη διάταξη:

[13] Η τριμερής διάταξη (three-tier architecture) διαχωρίζει μία εφαρμογή σε τρία ανεξάρτητα λειτουργικά επίπεδα και αποτελεί την κυρίαρχη επιλογή για διατάξεις παραδοσιακών εφαρμογών πελάτη-εξυπηρετητή.



Σχήμα 3.1: Επιλεγμένη τεχνολογική στοίβα

3.3 Μεθοδολογία σχεδίασης

Στον χώρο της τεχνολογίας λογισμικού υπάρχουν διάφορες μεθοδολογίες σχεδίασης οι οποίες έχουν μεταξύ τους κοινά στοιχεία. Αυτό καθιστά δύσκολο τον προσδιορισμό μίας μόνο μεθοδολογίας η οποία ακολουθείται πιστά σε κάθε έργο. Συνήθως, οι ομάδες που αναπτύσσουν το λογισμικό ακολουθούν μία μίξη από διάφορα εργαλεία, όπου αυτά κρίνονται βολικά για τους στόχους της ομάδας.

Κατά την σχεδίαση και την υλοποίηση του κώδικα ακολουθήθηκαν διάφορες τεχνικές και μοτίβα ανάπτυξης. Κατά βάση χρησιμοποιήθηκαν Agile μέθοδοι όπως το Kanban και Scrum και, αργότερα στην ανάπτυξη, το DevOps μοντέλο για διαρκή ενσωμάτωση (Continuous Integration) και διαρκή εγκατάσταση (Continuous Deployment).

Για την παρούσα εργασία, πραγματοποιήθηκε ανάλυση και σχεδιασμός των επιμέρους μονάδων εργασίας (tasks) πριν την έναρξη της διαδικασίας ανάπτυξης του κώδικα. Τα tasks που προδιαγράφηκαν ήταν συνήθως epics[14] τα οποία αργότερα χωρίστηκαν σε επιμέρους, μικρότερα tasks. Ορίστηκαν επίσης ορόσημα (milestones) τα οποία βοήθησαν ιδιαίτερα στην ιεράρχηση και προτεραιοποίηση των tasks.

Το Kanban είναι μία μέθοδος οργάνωσης έργων και οπτικοποίησης των μονάδων εργασίας (tasks) που απαιτούνται για την ολοκλήρωσή τους. Στο Kanban ορίζονται τα βασικά στάδια της ροής ενός task και χρησιμοποιούνται οπτικά μέσα ώστε να γίνει ιχνηλάτηση τόσο της συνολικής κατάστασης του έργου, όσο και συγκεκριμένων-μεμονωμένων tasks καθώς αυτά προοδεύουν. Για κάθε στάδιο ολοκλήρωσης ορίζεται μία ξεχωριστή ουρά εργασιών (στήλη), για παράδειγμα «σε αναμονή», «σε εξέλιξη», «ολοκληρωμένο». Χρησιμοποιούνται οπτικά σινιάλα (χρώματα, tags και άλλα) για τον διαχωρισμό και την γρήγορη κατανόηση των σημαντικότερων γνωρισμάτων

[14]Τα epics είναι μεγάλες μονάδες εργασίας οι οποίες αφορούν κάποιο βασικό χαρακτηριστικό. Ο διαχωρισμός τους σε επιμέρους tasks αναβάλλεται με σκοπό την καλύτερη κατανόηση των αναγκών τους.

των tasks, για παράδειγμα ξεχωριστό tag για κάθε υπηρεσία στην οποία αναφέρεται το task. Επίσης, ορίζονται όρια στον αριθμό των tasks που μπορούν να είναι ταυτόχρονα σε εξέλιξη.

Μία άλλη Agile μέθοδος είναι το Scrum. Το Scrum χρησιμοποιεί και επεκτείνει το Kanban. Η βασικές διαφορές του με το Kanban είναι ότι στο Scrum υπάρχουν πιο αυστηρές διαδικασίες. Ορίζονται προγραμματιστικοί κύκλοι (sprints) οι οποίοι έχουν συγκεκριμένες ημερομηνίες έναρξης και λήξης και συγκεκριμένους στόχους οι οποίοι αντικατοπτρίζονται σε στόχους ολοκλήρωσης ορισμένων tasks. Οι ρόλοι είναι σαφέστεροι, με κάθε μέλος της ομάδας να αναλαμβάνει διαφορετικές ευθύνες στην οργάνωση και εκτέλεση. Για την διαδικασία ανάπτυξης, υπήρξε πολύ χρήσιμη η χρήση του Scrum σε περιόδους όπου ήταν αναγκαία η ταχύτατη ανάπτυξη καίριων μερών του συστήματος. Λόγω της αυστηρότητας που επιβάλλεται από αυτό, ειδικά σε ό,τι αφορά τις προθεσμίες ολοκλήρωσης των επιμέρους tasks αλλά και του συνολικού sprint.

Καθώς η αναπτυξιακή διαδικασία ωριμάζει και η πλατφόρμα αποτελεί ένα βιώσιμο προϊόν, είναι χρήσιμη η ύπαρξη ενός συστήματος που να διευκολύνει την άμεση δημιουργία και δημοσίευση των νεότερων εκδόσεων. Μερικές εξαιρετικές μέθοδοι για την απρόσκοπτη και αυτοματοποιημένη επίτευξη του στόχου αυτού ορίζονται από το DevOps. Με τον όρο DevOps (development operations) αναφέρεται μία κουλτούρα σχεδίασης και ανάπτυξης λογισμικού που ορίζει τους ρόλους, τις διαδικασίες και τεχνολογίες της με σκοπό την συνεχή δημιουργία αξίας για τους χρήστες. Το DevOps έχει πολύ στενή σχέση με το Agile και αποτελεί την συνέχιση της νοοτροπίας αυτής στον χώρο.

Μία από τις πιο χρήσιμες πτυχές του DevOps, η οποία χρησιμοποιήθηκε στην διπλωματική, είναι το CI/CD το οποίο περιγράφει τις διαδικασίες αυτοματοποίησης των εργασιών ενσωμάτωσης (integration), ελέγχου (testing), παράδοσης (delivery) και εγκατάστασης (deployment) του προϊόντος. Μέσω των διαδικασιών αυτών αφαιρείται η ανάγκη ανθρώπινης αλληλεπίδρασης για την ολοκλήρωση των σταδίων αυτών, ενώ επιτυγχάνεται η διαρκής και απρόσκοπτη διάθεση της τελευταίες έκδοσης της πλατφόρμας στους χρήστες. Ορίζονται επίσης διαδικασίες δημιουργίας περιβάλλοντος ελέγχου (staging deploys) το οποίο αποτελεί σημαντικό βοήθημα στον έγκαιρο εντοπισμό λαθών του κώδικα.

3.4 Κατηγορίες χρηστών

Οι χρήστες (actors) της πλατφόρμας χωρίζονται σε πρωτεύοντες ή ενεργούς και δευτερεύοντες ή παθητικούς. Πρωτεύοντες χρήστες είναι εκείνοι που εκκινούν διεργασίες στο σύστημα. Δευτερεύοντες είναι οι χρήστες με τους οποίους αλληλεπιδρά το σύστημα, αλλά οι ίδιοι δεν εκκινούν διεργασίες σε αυτό. Συνολικά οι χρήστες που συμμετέχουν στο σύστημα είναι οι:

- Επισκέπτες
- Εγγεγραμμένα μέλη
- Συμβόλαια κοινοτήτων

3.4.1 Ενεργοί χρήστες

Οι ενεργοί χρήστες στο σύστημα είναι οι επισκέπτες και τα εγγεγραμμένα μέλη.

Όλοι οι χρήστες στο σύστημα είναι αρχικά επισκέπτες. Οι επισκέπτες έχουν τη δυνατότητα να βλέπουν το περιεχόμενο της κοινότητας, αλλά δε μπορούν να συμμετέχουν δημιουργώντας νέο περιεχόμενο (δημοσιεύοντας νέα θέματα ή μηνύματα). Επίσης, δε μπορούν να συμμετέχουν στις ψηφοφορίες της κοινότητας ή να ψηφίσουν τα μηνύματα.

Όταν ένας επισκέπτης εγγράφεται στο σύστημα, αποκτά έναν μοναδικό, αύξοντα αριθμό χρήστη και αποτελεί πλέον εγγεγραμμένο μέλος της κοινότητας. Τα εγγεγραμμένα μέλη έχουν τα δικαιώματα των επισκεπτών και μπορούν επιπλέον να προσθέσουν περιεχόμενο στην πλατφόρμα μέσω της δημιουργίας νέων θεμάτων, της δημοσίευσης μηνυμάτων και της ψήφισης στις ψηφοφορίες στις οποίες έχουν δικαίωμα.

3.4.2 Παθητικοί χρήστες

Παθητικοί χρήστες τους συστήματος είναι τα συμβόλαια των κοινοτήτων. Τα συμβόλαια αυτά δεν εκκινούν διεργασίες στο σύστημα και δεν αλληλεπιδρούν με αυτό άμεσα. Αποτελούν αυτόνομες εξωτερικές οντότητες, οι οποίες ορίζουν τους χρήστες κοινοτήτων μέσω της διάθεσης αναγνωριστικών token στα μέλη τους. Συγκεκριμένα, μέσω του διαμοιρασμού των token, καθορίζουν ποιοι χρήστες της πλατφόρμας έχουν δικαίωμα ψήφου στις ψηφοφορίες που αφορούν την κοινότητα.

3.4.3 Σύνοψη χρηστών

Συμπερασματικά προκύπτουν δύο διακριτές κατηγορίες ενεργών χρηστών με ξεχωριστά δικαιώματα όπως φαίνεται στο παρακάτω σχήμα:

	Δικαιώματα								
Κατηγορία χρήστη	*Προβολή θεμάτων*	*Προβολή μηνυμάτων*	*Προβολή ψηφοφοριών*	*Προβολή ψήφων μηνυμάτων*	*Δημιουργία θεμάτων*	*Δημιουργία μηνυμάτων*	*Δημιουργία ψηφοφοριών*	*Ψήφιση σε ψηφοφορίες*	*Ψήφιση μηνυμάτων*
Επισκέπτες	✓	✓	✓	✓	✗	✗	✗	✗	✗
Εγγεγραμμένα μέλη	✓	✓	✓	✓	✓	✓	✓*	✓*	✓

* Μόνο στις κοινότητες στις οποίες κατέχει το αντίστοιχο token και σε αυτές οι οποίες δεν έχουν ορισμένο token.

Πίνακας 3.1: Δικαιώματα χρήσης ανά κατηγορία χρήστη

3.5 Απαιτήσεις λογισμικού

Στην παρούσα ενότητα περιγράφονται οι βασικές απαιτήσεις λογισμικού (software requirements) της εφαρμογής.

Η πρώτη κατηγορία είναι αυτή των Λειτουργικών Απαιτήσεων (ΛΑ), η οποία αναφέρεται στη συμπεριφορά του συστήματος, δηλαδή στον τρόπο που θα αντιδρά και στις εξόδους που θα παράγει ανάλογα με τις εισόδους.

<ΛΑ-1> Ο χρήστης πρέπει να μπορεί να εγγραφεί στην εφαρμογή με τον Ethereum λογαριασμό του.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να εγγραφεί στην εφαρμογή, πατώντας το κουμπί «Sign Up» και συμπληρώνοντας τα απαραίτητα πεδία σύμφωνα με τις οδηγίες. Το πεδίο «Username» είναι υποχρεωτικό να συμπληρωθεί με το επιθυμητό username, το οποίο ορίζεται με μοναδικό τρόπο. Σε περίπτωση που ο χρήστης εισάγει μη διαθέσιμο Username, το σύστημα θα πρέπει να μην επιτρέπει στον χρήστη να συνεχίσει και να προβάλει αντίστοιχο μήνυμα λάθους. Επιπλέον, υπάρχουν τα προαιρετικά πεδία «Profile picture URL» και «Location», στα οποία ο χρήστης μπορεί να εισάγει μία εικόνα προφίλ και την τοποθεσία του αντίστοιχα.

User Priority (5/5): Η απαίτηση είναι ύψιστης προτεραιότητας για τους επισκέπτες, καθώς μόνο μέσω της εγγραφής μπορούν να χρησιμοποιήσουν τα υπόλοιπα χαρακτηριστικά της εφαρμογής (όπως φαίνεται στον πίνακα 3.1).

Technical Priority (5/5): Η απαίτηση είναι ύψιστης σημασίας για το σύστημα, επειδή επηρεάζει τη λειτουργικότητά του.

<ΛΑ-2> Ο χρήστης πρέπει να μπορεί συνδέεται στην εφαρμογή, εφόσον είναι εγγεγραμμένος.

Περιγραφή: Το σύστημα πρέπει να διαπιστώνει αυτόματα εάν το τρέχον Ethereum address έχει λογαριασμό στην εφαρμογή και εάν ναι, να συνδέει να τον χρήστη, ανακτώντας το Username του από το blockchain και προβάλλοντας το στο μενού.

User Priority (5/5): Αυτή η απαίτηση είναι ύψιστης προτεραιότητας για τους χρήστες, καθώς μέσω της σύνδεσης ενεργοποιούνται τα χαρακτηριστικά της δημιουργίας θεμάτων και δημοσίευσης μηνυμάτων.

Technical Priority (5/5): Η απαίτηση είναι ύψιστης σημασίας για το σύστημα, επειδή επηρεάζει τη λειτουργικότητά του.

<ΛΑ-3> Το σύστημα πρέπει να δημιουργεί τις απαραίτητες βάσεις δεδομένων και να τις συγχρονίζει με το δίκτυο.

Περιγραφή: Το σύστημα πρέπει να δημιουργεί τις απαραίτητες OrbitDB βάσεις δεδομένων, εάν αυτές δεν υπάρχουν ήδη τοπικά. Όταν ο χρήστης ξεκλειδώνει τον Ethereum λογαριασμό του, το σύστημα θα πρέπει να τον προτρέπει να υπογράψει με το ιδιωτικό του κλειδί τη συναλλαγή δημιουργίας της OrbitDB Identity του. Αυτή θα εξασφαλίζει τη γνησιότητα των βάσεών του και των δεδομένων που εκείνες θα περιέχουν. Επιπλέον, τοπικές βάσεις δεδομένων θα πρέπει να συγχρονίζονται με τις βάσεις άλλων IPFS κόμβων και να διατηρούνται ενημερωμένες.

User Priority (5/5): Η απαίτηση αυτή είναι ύψιστης σημασίας για τους χρήστες, καθώς η πλειοψηφία των δεδομένων της εφαρμογής διατηρούνται σε αυτές τις βάσεις.

Technical Priority (5/5): Η παρούσα απαίτηση είναι ύψιστης σημασίας για το σύστημα, καθώς οι περισσότερες θεμελιώδεις λειτουργίες της εφαρμογής προϋποθέτουν την αποθήκευση δεδομένων σε OrbitDB βάσεις.

<ΛΑ-4> Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να δημιουργεί θέματα (topics).

Περιγραφή: Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να δημιουργεί νέα θέματα. Αυτό το επιτυγχάνει πατώντας το κουμπί «New Topic», συμπληρώνοντας τα υποχρεωτικά πεδία της φόρμας («Topic subject» και «First post content»), πατώντας το κουμπί «Create Topic» και επιβεβαιώνοντας τη συναλλαγή στο Ethereum.

User Priority (5/5): Αυτή η απαίτηση είναι υψηλής σημασίας καθώς επιτελεί έναν από τους βασικούς στόχους της πλατφόρμας.

Technical Priority (5/5): Η απαίτηση είναι υψηλής σημασίας για τον ίδιο λόγο.

<ΛΑ-5> Ο χρήστης πρέπει να μπορεί να περιηγείται στα θέματα μίας κοινότητας.

Περιγραφή: Το σύστημα πρέπει να μπορεί να προβάλλει τα δημιουργημένα θέματα μίας κοινότητας στην αρχική οθόνη της. Ο χρήστης πρέπει να μπορεί να περιηγείται σε αυτά πατώντας πάνω τους και, έπειτα, χρησιμοποιώντας τα βέλη πλοήγησης, να περιηγηθεί στο ιστορικό των μηνυμάτων του θέματος.

User Priority (5/5): Η απαίτηση αυτή είναι υψηλής σημασίας, αφού επιτρέπει στους επισκέπτες να έχουν πρόσβαση στο δημοσιευμένο υλικό της πλατφόρμας.

Technical Priority (5/5): Πρόκειται για απαίτηση υψηλής σημασίας, επειδή αποτελεί βασικό χαρακτηριστικό για τη χρηστικότητα της πλατφόρμας.

<ΛΑ-6> Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να δημιουργεί μηνύματα (posts).

Περιγραφή: Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να δημιουργεί μηνύματα στο θέμα που επιθυμεί. Αυτό επιτυγχάνεται συμπληρώνοντας το πεδίο νέου μηνύματος στην οθόνη του θέματος, πατώντας το κουμπί «Post» και επιβεβαιώνοντας τη συναλλαγή στο Ethereum.

User Priority (5/5): Αυτή η απαίτηση είναι ύψιστης σημασίας για τους χρήστες, επειδή αποτελεί ένα από τα βασικότερα χαρακτηριστικά της πλατφόρμας.

Technical Priority (5/5): Η απαίτηση αυτή είναι ύψιστης σημασίας για το σύστημα, καθώς αποτελεί θεμελιώδες κομμάτι για την επίτευξη του βασικότερου στόχου της, δηλαδή της δημιουργίας διαλόγου.

<ΛΑ-7> Ο χρήστης πρέπει να μπορεί να τροποποιεί τα μηνύματά του.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να τροποποιεί τα μηνύματά του. Αυτό το επιτυγχάνει επιλέγοντας το κουμπί επεξεργασίας στο εκάστοτε μήνυμα, πραγματοποιώντας τις επιθυμητές τροποποιήσεις και πατώντας το κουμπί επιβεβαίωσης. Στη συνέχεια, το σύστημα τροποποιεί το περιεχόμενο του μηνύματος στη βάση δεδομένων του χρήστη. Σε περίπτωση που ο χρήστης αλλάξει γνώμη κατά τη διάρκεια της διαδικασίας της επεξεργασίας, μπορεί να πατήσει το κουμπί ακύρωσης και να αναιρέσει τις αλλαγές που πραγματοποίησε.

User Priority (4/5): Η απαίτηση αυτή αποτελεί σημαντικό χαρακτηριστικό, καθώς παρέχει στους χρήστες άμεσο έλεγχο επί των δεδομένων τους.

Technical Priority (3/5): Αυτή η απαίτηση είναι μέτριας σημαντικότητας για το σύστημα, επειδή αυτό θα μπορούσε να είναι λειτουργικό χωρίς το χαρακτηριστικό της επεξεργασίας μηνυμάτων.

<ΛΑ-8> Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να ψηφίζει σε μηνύματα άλλων χρηστών.

Περιγραφή: Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να υπερψηφίζει ή να καταψηφίζει μηνύματα άλλων χρηστών. Αυτό το επιτυγχάνει πατώντας τα παρακείμενα κουμπιά «+» ή «-» αντίστοιχα και επιβεβαιώνοντας τη συναλλαγή στο Ethereum (οι ψήφοι αποθηκεύονται εκεί). Η διαδικασία ισχύει και για την τροποποίηση ή την αφαίρεση μίας ψήφου από τον χρήστη.

User Priority (3/5): Η παρούσα απαίτηση είναι μέτριας σημασίας για τους χρήστες, καθώς αποτελεί ένα χρήσιμο αλλά όχι απαραίτητο χαρακτηριστικό.

Technical Priority (2/5): Η απαίτηση είναι χαμηλής σημασίας για τη λειτουργικότητα του συστήματος. Ωστόσο, τα δημιουργημένα δεδομένα μπορεί να είναι χρήσιμα σε μελλοντική επέκταση της εφαρμογής (π.χ. για τον υπολογισμό της εμπιστοσύνης των χρηστών).

<ΛΑ-9> Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να δημιουργεί ψηφοφορίες (polls).

Περιγραφή: Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να δημιουργεί ψηφοφορίες στις κοινότητες που του το επιτρέπουν. Αυτό το επιτυγχάνει πατώντας «Add Poll» στην οθόνη δημιουργία θέματος και συμπληρώνοντας τα απαραίτητα πεδία.

User Priority (5/5): Η απαίτηση είναι ύψιστης σημασίας για τους χρήστες, καθώς οι δημοκρατικές διαδικασίες αποτελούν μία από τις κύριες χρήσεις της πλατφόρμας.

Technical Priority (5/5): Η απαίτηση είναι ύψιστης σημασίας για το σύστημα, επειδή αποτελεί βασική προδιαγραφή του.

<ΛΑ-10> Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να ψηφίζει σε ψηφοφορίες.

Περιγραφή: Ο εγγεγραμμένος χρήστης πρέπει να μπορεί να ψηφίζει σε ψηφοφορίες, σύμφωνα με τους εκάστοτε κανόνες της. Σε κοινότητες που το απαιτούν, ο χρήστης θα πρέπει να διαθέτει το αντίστοιχο voting token για να έχει το δικαίωμα ψήφου.

User Priority (5/5): Η απαίτηση είναι ύψιστης σημασίας για τους χρήστες, καθώς οι δημοκρατικές διαδικασίες αποτελούν μία από τις κύριες χρήσεις της πλατφόρμας.

Technical Priority (5/5): Η απαίτηση είναι ύψιστης σημασίας για το σύστημα, επειδή αποτελεί βασική προδιαγραφή του.

<ΛΑ-11> Ο χρήστης πρέπει να μπορεί να διαγράφει τα τοπικά δεδομένα.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να διαγράφει τα τοπικά δεδομένα. Αυτό το επιτυγχάνει πατώντας στο κουμπί «Clear databases» του μενού και επιβεβαιώνοντας τη διαγραφή μέσω ενός pop-up διαλόγου.

User Priority (2/5): Η απαίτηση αυτή είναι χαμηλής σημασία για τους χρήστες, διότι αποτελεί απλά μία διευκόλυνση για τη διαγραφή των δεδομένων που έχουν αποθηκεύσει τοπικά.

Technical Priority (2/5): Η απαίτηση αυτή είναι χαμηλής σημασίας για το σύστημα.

<ΛΑ-12> Ο χρήστης πρέπει να μπορεί να δημιουργεί κοινότητες.

Περιγραφή: Ο χρήστης πρέπει να μπορεί να δημιουργεί κοινότητες, πατώντας το κουμπί «Create community» και συμπληρώνοντας τα απαραίτητα πεδία.

User Priority (4/5): Η απαίτηση είναι μεγάλης σημασίας για τους χρήστες, καθώς παρέχει την ευελιξία της δημιουργίας κοινοτήτων.

Technical Priority (4/5): Πρόκειται για απαίτηση μεγάλης σημασίας για την πλατφόρμα, επειδή έτσι γενικεύει τη χρήση της σε περισσότερες κοινότητες, προσελκύοντας μεγαλύτερο αριθμό χρηστών.

<ΛΑ-13> Κατά τη δημιουργία κοινότητας, ο χρήστης πρέπει να έχει τη δυνατότητα να ορίσει ένα contract που θα παρέχει προσαρμοσμένα tokens για αυτήν.

Περιγραφή: Κατά τη δημιουργία κοινότητας, ο χρήστης πρέπει να έχει τη δυνατότητα να ορίσει ένα contract που θα παρέχει προσαρμοσμένα tokens για αυτήν. Τα tokens αυτά θα διαμοιράζονται με τον τρόπο που επιθυμεί η κοινότητα και θα είναι εκείνα τα οποία θα καθορίζουν τους έγκυρους ψηφοφόρους της.

User Priority (4/5): Αυτή η απαίτηση είναι μεγάλης σημασίας, καθώς παρέχει στις κοινότητες τη δυνατότητα διενέργειας επιβεβαιώσιμων ανώνυμων ψηφοφοριών.

Technical Priority (4/5): Η απαίτηση είναι μεγάλης σημασίας για το σύστημα, διότι παρέχει στις κοινότητες την απαιτούμενη αυτονομία στον ορισμό των δημοκρατικών διαδικασιών τους.

Η δεύτερη κατηγορία είναι αυτή των Μη Λειτουργικών Απαιτήσεων (ΜΛΑ). Περιλαμβάνει απαιτήσεις αρχιτεκτονικής σημασίας, οι οποίες καθορίζουν κριτήρια ή περιορισμούς του τρόπου λειτουργίας του συστήματος και σχετίζονται με χαρακτηριστικά όπως η αποδοτικότητα, η αξιοπιστία και η ευχρηστία του.

<ΜΛΑ-1> Η πλατφόρμα πρέπει να είναι κατά το δυνατόν αρχιτεκτονικά αποκεντρωμένη.

Περιγραφή: Οι τεχνολογίες στις οποίες βασίζεται η πλατφόρμα πρέπει ιδανικά να μη δημιουργούν κεντρικά σημεία. Επιπλέον, ο κώδικας και η δημόσια διάθεση του πρέπει να γίνονται με αποκεντρωμένο τρόπο.

User Priority (5/5): Η αρχιτεκτονική αποκέντρωση της πλατφόρμας αποτελεί απαίτηση ύψιστης προτεραιότητας για τον χρήστη, καθώς διασφαλίζει την πολιτική αποκέντρωση και, έτσι, τους κύριους στόχους που έχουν οριστεί.

Technical Priority (5/5): Η αρχιτεκτονική αποκέντρωση της πλατφόρμας αποτελεί, απαίτηση ύψιστης σημασίας για το σύστημα, καθώς καθιστά το ίδιο ασφαλές σε επιθέσεις και τα δεδομένα μόνιμα διαθέσιμα στους χρήστες.

<**ΜΛΑ-2**> Τα fees για τη χρήση του Ethereum blockchain πρέπει να ελαχιστοποιούνται.

Περιγραφή: Τα τέλη συναλλαγών που πρέπει να καταβάλλονται για τη χρήση του Ethereum blockchain εξαρτώνται άμεσα τόσο από τον όγκο των δεδομένων προς αποθήκευση, όσο και από τους κύκλους επεξεργασίας των smart contracts της εφαρμογής. Ως προς τα δεδομένα, οι προγραμματιστές θα πρέπει να μεριμνούν ώστε ο κύριος όγκος τους να αποθηκεύεται επί του IPFS, ενώ επί του blockchain να αποθηκεύονται μόνο όσα πραγματικά χρειάζονται. Ως προς την απαιτούμενη επεξεργαστική ισχύ, πρέπει να βελτιστοποιείται ο κώδικας των smart contracts, έτσι ώστε οι διάφορες λειτουργίες τους να εκτελούνται με τους λιγότερους δυνατούς επεξεργαστικούς κύκλους.

User Priority (4/5): Η απαίτηση αυτή είναι μεγάλης σημασίας για τους χρήστες καθώς ναι μεν δεν είναι απαραίτητη για τη χρήση της αλλά είναι ιδιαίτερα σημαντική για την ένταξη χρηστών με χαμηλότερες οικονομικές δυνατότητες.

Technical Priority (5/5): Η απαίτηση αυτή είναι μεγάλης σημασίας για το σύστημα διότι αποτελεί σημαντικό παράγοντα που επιδρά στην προσέλκυση και τη διατήρηση ενεργών χρηστών.

<**ΜΛΑ-3**> Τα contracts της εφαρμογής πρέπει να είναι αναβαθμίσιμα.

Περιγραφή: Τα contracts της εφαρμογής πρέπει μπορούν να αναβαθμιστούν, έτσι ώστε να μπορούν να προστίθενται λειτουργίες και να διορθώνονται σφάλματα. Η αναβαθμισιμότητά τους θα πρέπει να επιτυγχάνεται με μεθόδους που να μην υπονομεύουν τη λειτουργικότητα των προηγούμενων εκδόσεων.

User Priority (2/5): Η απαίτηση αυτή είναι χαμηλής σημασίας για τους χρήστες, καθώς αφορά την ανάπτυξη και όχι τη χρήση της.

Technical Priority (5/5): Η απαίτηση αυτή είναι υψηλής σημασίας για το σύστημα, επειδή προσφέρει τη δυνατότητα αποσφαλμάτωσης του, καθώς και την υλοποίηση νέων χαρακτηριστικών.

3.6 Σενάρια χρήσης

Βασικό μέρος της σχεδίασης της πλατφόρμας ήταν η καταγραφή των απαιτήσεων η οποία έγινε στην προηγούμενη ενότητα (3.5) καθώς και η σχεδίαση και ανάπτυξη των σεναρίων χρήσης. Τα σενάρια χρήσης αντιστοιχίζουν πιθανές ενέργειες των χρηστών με αποκρίσεις του συστήματος. Μέσω της αντιστοίχισης αυτής παρουσιάζεται η λειτουργικότητα του συστήματος

και περιγράφονται τόσο οι λειτουργικές όσο και οι μη λειτουργικές απαιτήσεις του συστήματος.

Παρατίθενται εδώ τα σενάρια χρήσης που δίνουν τις απαραίτητες πληροφορίες για την κατανόηση της λειτουργίας του συστήματος.

3.6.1 Σενάριο χρήσης 1: Εγγραφή χρήστη

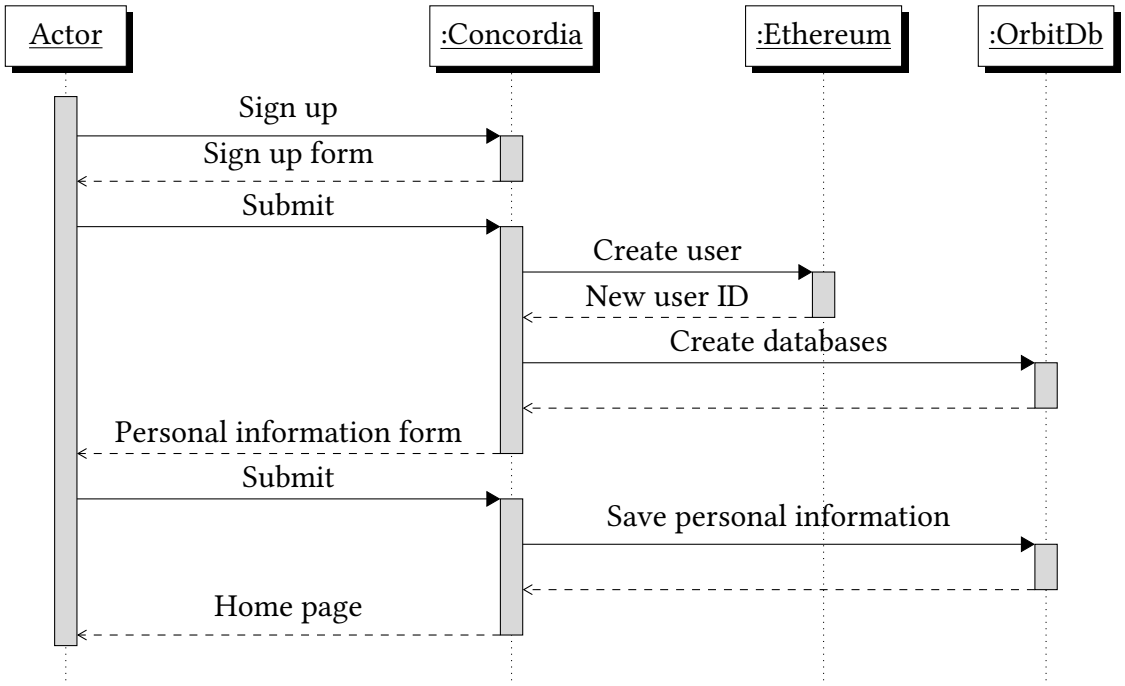
Το σενάριο χρήσης 1, <ΣΧ-1>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για την εγγραφή ενός χρήστη στο σύστημα. Στους πίνακες 3.2 και 3.3 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-1> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.2 φαίνεται το διάγραμμα της βασικής ροής.

	Εγγράφομαι στο σύστημα
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο επισκέπτης να μπορεί να εγγραφεί στο σύστημα ως χρήστης.
Αναφορά ΛΑ	**<ΛΑ-1>**, **<ΛΑ-3>**
Αναφορά ΜΛΑ	**<ΜΛΑ-2>**
Πυροδότηση δραστηριότητας	Ο επισκέπτης πατάει το κουμπί εγγραφή.
Προϋπόθεση	Ο επισκέπτης πρέπει να έχει ανοίξει την σελίδα της εφαρμογής.

Πίνακας 3.2: Σενάριο χρήσης 1, εγγραφή χρήστη στο σύστημα.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί εγγραφή.	Το σύστημα εμφανίζει την φόρμα «Εγγραφή Χρήστη».
2	Ο χρήστης συμπληρώνει τα πεδία και πατάει το κουμπί «Υποβολή».	Το σύστημα εισάγει νέο χρήστη στο blockchain.
3	-	Το σύστημα δημιουργεί τις προσωπικές βάσεις βάσεις δεδομένων OrbitDb του χρήστη.
4	-	Το σύστημα εμφανίζει την φόρμα «Πληροφορίες Χρήστη».
5	Ο χρήστης συμπληρώνει τις προσωπικές του πληροφορίες και πατάει το κουμπί «Υποβολή».	Το σύστημα εισάγει τις πληροφορίες χρήστη στην προσωπική του βάση OrbitDb.
Μετέπειτα κατάσταση:	Το σύστημα μεταβαίνει στην αρχική σελίδα της εφαρμογής.	

Πίνακας 3.3: Σενάριο χρήσης 1 - Βασική ροή



Σχήμα 3.2: Σενάριο χρήσης 1 - Διάγραμμα βασικής ροής

Το <ΣΧ-1> περιέχει επίσης τρεις εναλλακτικές ροές που μπορεί να προκύψουν βάσει των επιλογών του χρήστη και οι οποίες περιγράφονται στους πίνακες 3.4, 3.5 και 3.6.

Εναλλακτική ροή 1: Τα στοιχεία χρήστη είναι λανθασμένα.		
Εφόσον ο χρήστης στη γραμμή 2 δεν συμπληρώσει το πεδίο ονόματος χρήστη ή συμπληρώσει ένα όνομα χρήστη το οποίο είναι ήδη σε χρήση στο σύστημα, το σύστημα πρέπει να επιστρέψει σχετικό μήνυμα σφάλματος.		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	-	Το σύστημα εμφανίζει μήνυμα σφάλματος.
Το σύστημα επιστρέφει στη γραμμή 1 της βασικής ροής.		

Πίνακας 3.4: Σενάριο χρήσης 1 - Εναλλακτική ροή 1

Εναλλακτική ροή 2: Ο χρήστης πατάει το κουμπί «Άκυρο».		
Εφόσον ο χρήστης στη γραμμή 2 της Βασικής Ροής επιλέξει «Άκυρο» το σύστημα επιστρέφει στην αρχική σελίδα της εφαρμογής.		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί «Άκυρο»	Το σύστημα επιστρέφει στην αρχική σελίδα της εφαρμογής.
Το σενάριο χρήσης τερματίζεται.		

Πίνακας 3.5: Σενάριο χρήσης 1 - Εναλλακτική ροή 2

Εναλλακτική ροή 3: Ο χρήστης πατάει το κουμπί «Παράληψη».		
Εφόσον ο χρήστης στη γραμμή 5 της Βασικής Ροής επιλέξει «Παράληψη» το σύστημα επιστρέφει στην αρχική σελίδα της εφαρμογής.		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί «Παράληψη»	Το σύστημα επιστρέφει στην αρχική σελίδα της εφαρμογής.
Το σενάριο χρήσης τερματίζεται.		

Πίνακας 3.6: Σενάριο χρήσης 1 - Εναλλακτική ροή 3

3.6.2 Σενάριο χρήσης 2: Σύνδεση χρήστη

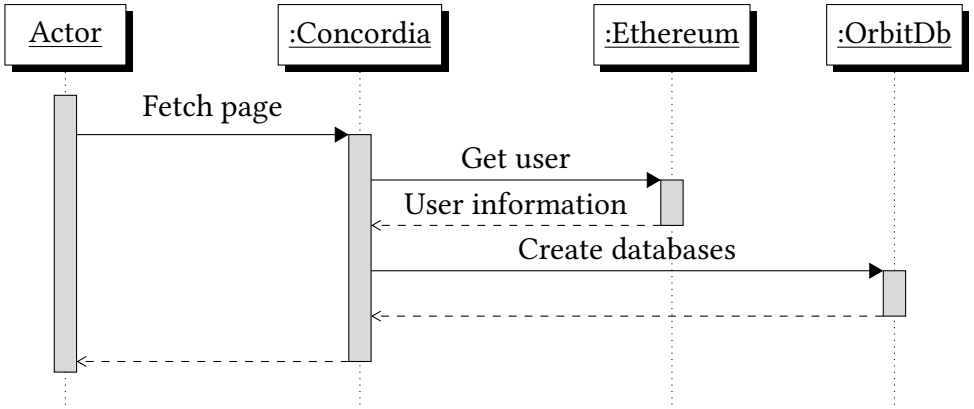
Το σενάριο χρήσης 2, <ΣΧ-2>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για την σύνδεση ενός χρήστη στο σύστημα. Στους πίνακες 3.7 και 3.8 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-2> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.3 φαίνεται το διάγραμμα της βασικής ροής.

Συνδέομαι στο σύστημα	
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο χρήστης να συνδέεται αυτόματα στο σύστημα.
Αναφορά ΛΑ	**<ΛΑ-2>**
Αναφορά ΜΛΑ	-
Πυροδότηση δραστηριότητας	-
Προϋπόθεση	Ο χρήστης πρέπει να έχει ανοίξει την σελίδα της εφαρμογής.

Πίνακας 3.7: Σενάριο χρήσης 2, σύνδεση χρήστη στο σύστημα.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	-	Το σύστημα ανακτά τις πληροφορίες του χρήστη από το blockchain.
2	-	Το σύστημα δημιουργεί τις προσωπικές βάσεις βάσεις δεδομένων OrbitDb του χρήστη.
Μετέπειτα κατάσταση:	Το σύστημα παραμένει στην αρχική σελίδα της εφαρμογής.	

Πίνακας 3.8: Σενάριο χρήσης 2 - Βασική ροή



Σχήμα 3.3: Σενάριο χρήσης 2 - Διάγραμμα βασικής ροής

3.6.3 Σενάριο χρήσης 3: Δημιουργία νέου θέματος

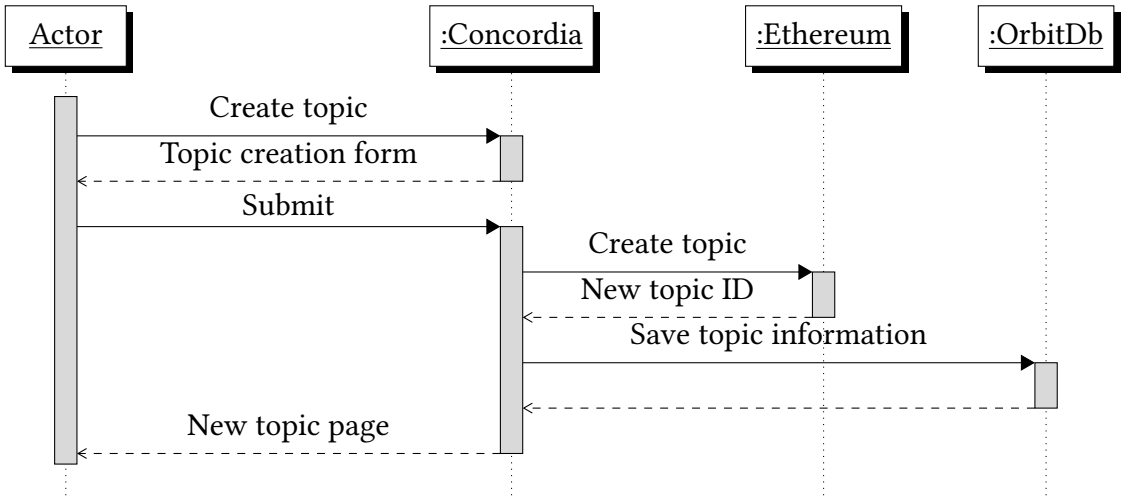
Το σενάριο χρήσης 3, <ΣΧ-3>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για την δημιουργία ενός θέματος. Στους πίνακες 3.9 και 3.10 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-3> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.4 φαίνεται το διάγραμμα της βασικής ροής.

Δημιουργώ νέο θέμα	
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο χρήστης να μπορεί να δημιουργήσει νέο θέμα.
Αναφορά ΛΑ	**<ΛΑ-4>**, **<ΛΑ-9>**
Αναφορά ΜΛΑ	**<ΜΛΑ-2>**
Πυροδότηση δραστηριότητας	Ο χρήστης πατάει το κουμπί δημιουργίας νέου θέματος.
Προϋπόθεση	Ο χρήστης να έχει συνδεθεί στην εφαρμογή και να βρίσκεται στην αρχική σελίδα.

Πίνακας 3.9: Σενάριο χρήσης 3, δημιουργία νέου θέματος.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί δημιουργίας νέου θέματος.	Το σύστημα εμφανίζει την φόρμα «Δημιουργία Θέματος».
2	Ο χρήστης συμπληρώνει τα πεδία και πατάει το κουμπί «Υποβολή».	Το σύστημα εισάγει νέο θέμα στο blockchain.
3	-	Το σύστημα εισάγει τις πληροφορίες του θέματος στην προσωπική βάση OrbitDb του χρήστη.
Μετέπειτα κατάσταση:	Το σύστημα μεταβαίνει στην σελίδα του νέου θέματος.	

Πίνακας 3.10: Σενάριο χρήσης 3 - Βασική ροή

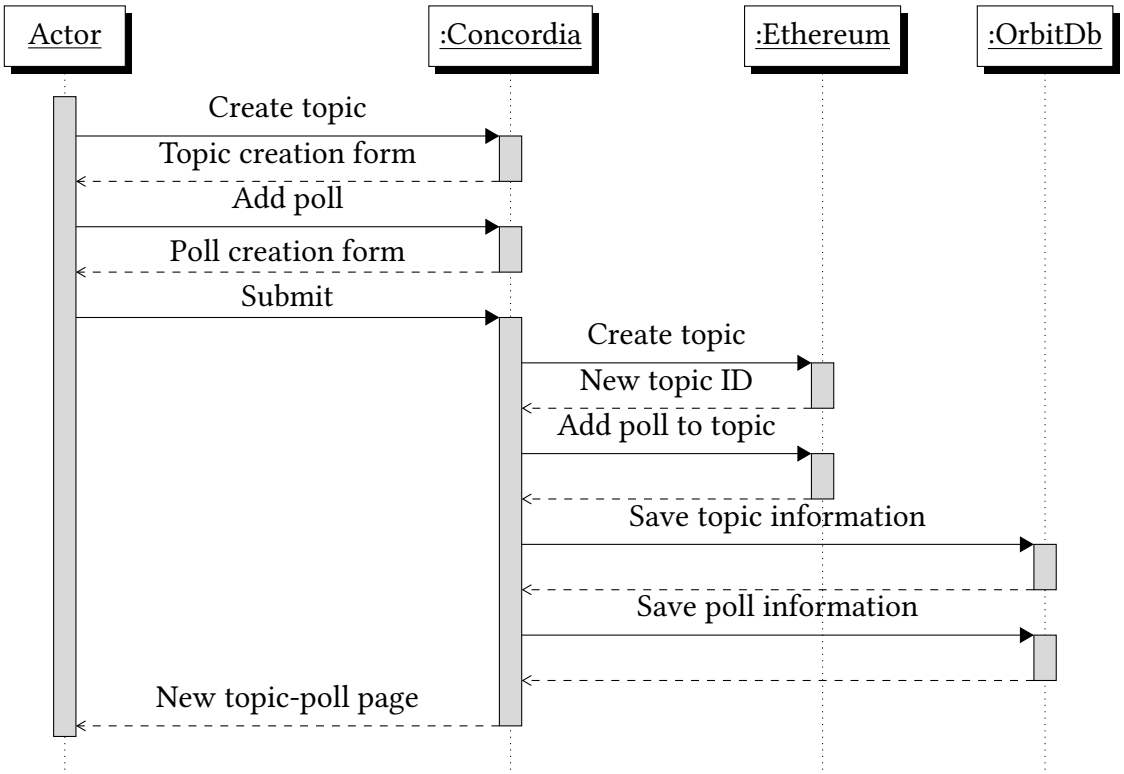


Σχήμα 3.4: Σενάριο χρήσης 3 - Διάγραμμα βασικής ροής

Το <ΣΧ-3> περιέχει επίσης δύο εναλλακτικές ροές που μπορεί να προκύψουν βάσει των επιλογών του χρήστη και οι οποίες περιγράφονται στους πίνακες 3.11 και 3.12. Η εναλλακτική ροή 1 φαίνεται επίσης στο σχήμα 3.5 όπου παρουσιάζεται το διάγραμμα ροής της.

Εναλλακτική ροή 1: Ο χρήστης δημιουργεί ψηφοφορία.		
Εφόσον ο χρήστης στη γραμμή 2 της Βασικής Ροής επιλέξει «Προσθήκη Ψηφοφορίας» το σύστημα ανανεώνει την σελίδα προσθέτοντας τα επιπλέον πεδία της φόρμας «Δημιουργία Ψηφοφορίας».		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης, αφού συμπληρώσει τη φόρμα «Δημιουργία Θέματος», πατάει το κουμπί «Προσθήκη ψηφοφορίας»	Το σύστημα ανανεώνει τη σελίδα με τα πεδία της φόρμας «Δημιουργία Ψηφοφορίας».
2	Ο χρήστης συμπληρώνει τα πεδία και πατάει το κουμπί «Υποβολή».	Το σύστημα εισάγει το νέο θέμα καθώς και τη νέα ψηφοφορία στο blockchain.
3	-	Το σύστημα εισάγει τις πληροφορίες του θέματος και της ψηφοφορίας στις προσωπικές βάσεις OrbitDb του χρήστη.
Το σύστημα μεταβαίνει στην σελίδα του νέου θέματος.		

Πίνακας 3.11: Σενάριο χρήσης 3 - Εναλλακτική ροή 1



Σχήμα 3.5: Σενάριο χρήσης 3 - Διάγραμμα εναλλακτικής ροής 1

Εναλλακτική ροή 2: Ο χρήστης πατάει το κουμπί «Άκυρο».		
Εφόσον ο χρήστης στη γραμμή 2 της Βασικής Ροής ή στη γραμμή 2 της Εναλλακτικής Ροής 1 επιλέξει «Άκυρο» το σύστημα επιστρέφει στην αρχική σελίδα της εφαρμογής.		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί «Άκυρο»	Το σύστημα επιστρέφει στην αρχική σελίδα της εφαρμογής.
Το σενάριο χρήσης τερματίζεται.		

Πίνακας 3.12: Σενάριο χρήσης 3 - Εναλλακτική ροή 2

3.6.4 Σενάριο χρήσης 4: Ανάκτηση θέματος

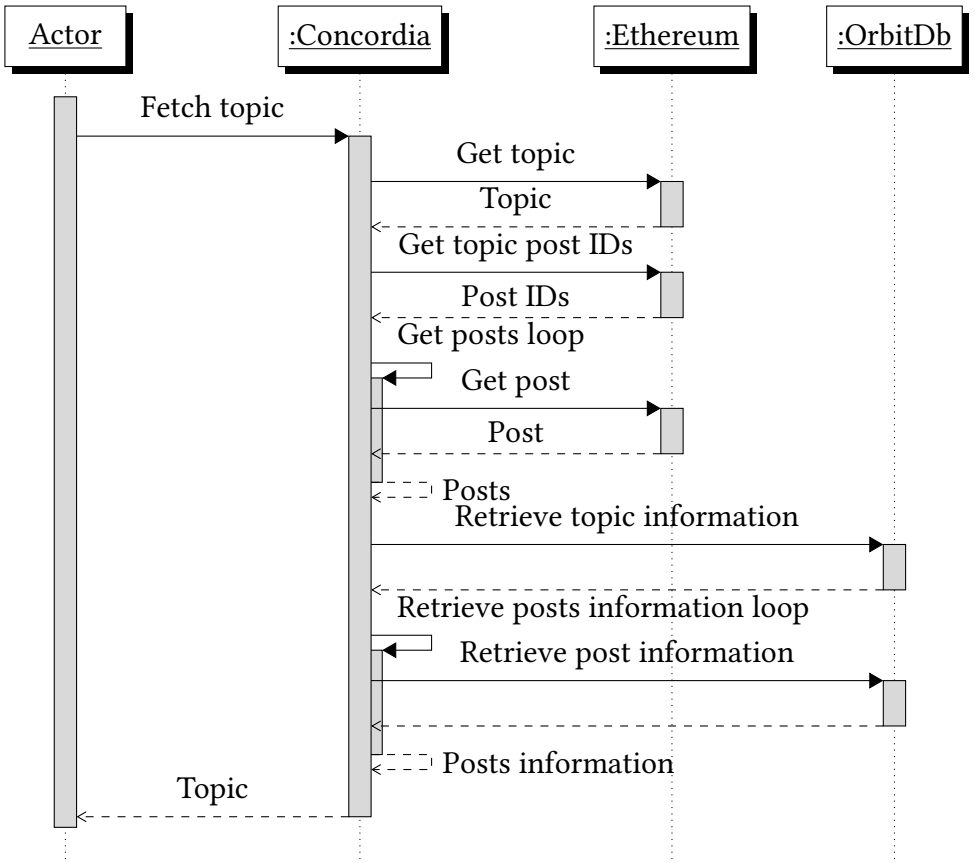
Το σενάριο χρήσης 4, <ΣΧ-4>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για την ανάκτηση ενός θέματος. Στους πίνακες 3.13 και 3.14 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-4> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.6 φαίνεται το διάγραμμα της βασικής ροής.

Ανακτώ ένα θέμα	
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο επισκέπτης ή ο χρήστης να μπορεί να ανακτήσει ένα θέμα.
Αναφορά ΛΑ	**<ΛΑ-5>**
Αναφορά ΜΛΑ	-
Πυροδότηση δραστηριότητας	Ο επισκέπτης ή χρήστης πατάει σε ένα από τα θέματα.
Προϋπόθεση	Ο επισκέπτης ή χρήστης πρέπει να έχει ανοίξει την σελίδα της εφαρμογής.

Πίνακας 3.13: Σενάριο χρήσης 4, ανάκτηση θέματος.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει σε ένα από τα θέματα της λίστας.	Το σύστημα ανακτά τις πληροφορίες του θέματος από το blockchain.
2	-	Το σύστημα ανακτά τα μηνύματα του θέματος αντιγράφοντας τις προσωπικές βάσεις OrbitDb των συγγραφέων.
Μετέπειτα κατάσταση:	Το σύστημα μεταβαίνει στην σελίδα του θέματος.	

Πίνακας 3.14: Σενάριο χρήσης 4 - Βασική ροή



Σχήμα 3.6: Σενάριο χρήσης 4 - Διάγραμμα βασικής ροής

Το <ΣΧ-4> περιέχει επίσης μία εναλλακτική ροή που μπορεί να προκύψει βάσει των επιλογών του χρήστη και η οποία περιγράφεται στον πίνακα 3.15. Η εναλλακτική ροή 1 φαίνεται επίσης στο σχήμα 3.7 όπου παρουσιάζεται το διάγραμμα ροής της.

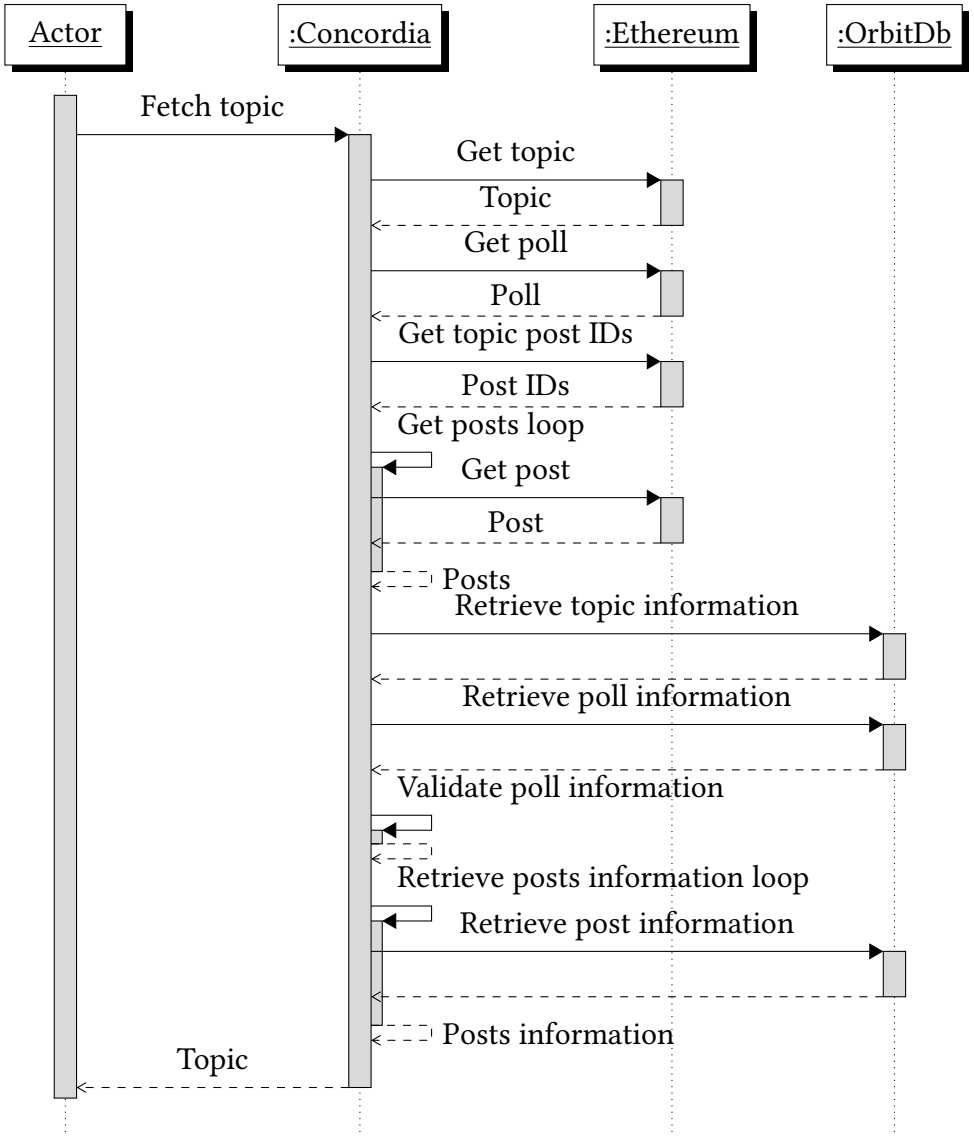
Εναλλακτική ροή 1: Το θέμα περιέχει ψηφοφορία.

Εφόσον το θέμα που ανακτήθηκε στη γραμμή 1 της Βασικής Ροής περιέχει ψηφοφορία ανακτώνται οι πληροφορίες της.

Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	-	Το σύστημα ανακτά τα μηνύματα του θέματος αντιγράφοντας τις προσωπικές βάσεις OrbitDb των συγγραφέων.
2	-	Το σύστημα ανακτά την ψηφοφορία από το blockchain.
3	-	Το σύστημα ανακτά τις πληροφορίες της ψηφοφορίας αντιγράφοντας την προσωπική βάση OrbitDb του συγγραφέα.
4	-	Το σύστημα επιβεβαιώνει τις πληροφορίες της ψηφοφορίας με βάση το hash που έχει ανακτηθεί από το blockchain.

Το σενάριο χρήσης τερματίζεται.

Πίνακας 3.15: Σενάριο χρήσης 4 - Εναλλακτική ροή 1



Σχήμα 3.7: Σενάριο χρήσης 4 - Διάγραμμα εναλλακτικής ροής 1

3.6.5 Σενάριο χρήσης 5: Δημιουργία νέου μηνύματος

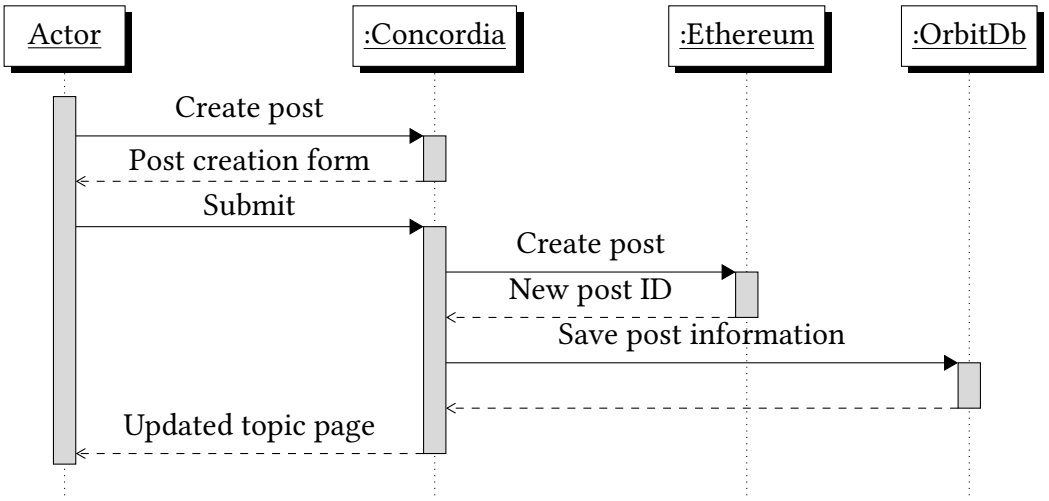
Το σενάριο χρήσης 5, <ΣΧ-5>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για την δημιουργία ενός μηνύματος. Στους πίνακες 3.16 και 3.17 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-5> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.8 φαίνεται το διάγραμμα της βασικής ροής.

Δημιουργώ νέο μήνυμα	
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο χρήστης να μπορεί να δημιουργήσει νέο μήνυμα.
Αναφορά ΛΑ	**<ΛΑ-6>**
Αναφορά ΜΛΑ	**<ΜΛΑ-2>**
Πυροδότηση δραστηριότητας	Ο χρήστης πατάει το κουμπί δημιουργίας νέου μηνύματος.
Προϋπόθεση	Ο χρήστης να έχει συνδεθεί στην εφαρμογή και να βρίσκεται στην σελίδα ενός θέματος.

Πίνακας 3.16: Σενάριο χρήσης 5, δημιουργία νέου μηνύματος.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί δημιουργίας νέου μηνύματος.	Το σύστημα εμφανίζει την φόρμα «Δημιουργία Μηνύματος».
2	Ο χρήστης συμπληρώνει τα πεδία και πατάει το κουμπί «Υποβολή».	Το σύστημα εισάγει νέο μήνυμα στο blockchain.
3	-	Το σύστημα εισάγει τις πληροφορίες του μηνύματος στην προσωπική βάση OrbitDb του χρήστη.
Μετέπειτα κατάσταση:	Το σύστημα παραμένει στη σελίδα του θέματος εμφανίζοντας το νέο μήνυμα.	

Πίνακας 3.17: Σενάριο χρήσης 5 - Βασική ροή



Σχήμα 3.8: Σενάριο χρήσης 5 - Διάγραμμα βασικής ροής

Το <ΣΧ-5> περιέχει επίσης μία εναλλακτική ροή που μπορεί να προκύψει βάσει των επιλογών του χρήστη και η οποία περιγράφεται στον πίνακα 3.18.

Εναλλακτική ροή 1: Ο χρήστης πατάει το κουμπί «Άκυρο».		
Εφόσον ο χρήστης στη γραμμή 2 της Βασικής Ροής επιλέξει «Άκυρο» το σύστημα επιστρέφει στη σελίδα του θέματος.		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί «Άκυρο»	Το σύστημα επιστρέφει στη σελίδα του θέματος.
Το σενάριο χρήσης τερματίζεται.		

Πίνακας 3.18: Σενάριο χρήσης 5 - Εναλλακτική ροή 1

3.6.6 Σενάριο χρήσης 6: Τροποποίηση μηνύματος

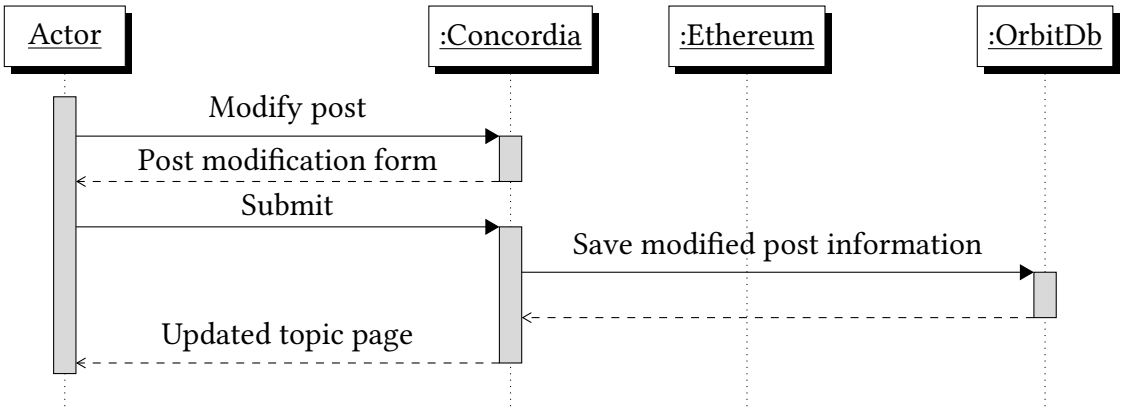
Το σενάριο χρήσης 6, <ΣΧ-6>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για τη τροποποίηση ενός μηνύματος. Στους πίνακες 3.19 και 3.20 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-6> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.9 φαίνεται το διάγραμμα της βασικής ροής.

Τροποποιώ ένα μήνυμα	
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο χρήστης να μπορεί να τροποποιήσει τα μηνύματά του.
Αναφορά ΛΑ	**<ΛΑ-7>**
Αναφορά ΜΛΑ	-
Πυροδότηση δραστηριότητας	Ο χρήστης πατάει το κουμπί τροποποίησης του μηνύματος.
Προϋπόθεση	Ο χρήστης να έχει συνδεθεί στην εφαρμογή και να βρίσκεται στην σελίδα του θέματος που περιέχει το μήνυμά του.

Πίνακας 3.19: Σενάριο χρήσης 6, τροποποίηση μηνύματος.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί τροποποίησης του μηνύματος.	Το σύστημα εμφανίζει την φόρμα «Τροποποίηση Μηνύματος».
2	Ο χρήστης συμπληρώνει τα πεδία και πατάει το κουμπί «Υποβολή».	Το σύστημα τροποποιεί τις πληροφορίες του μηνύματος στην προσωπική βάση OrbitDb του χρήστη.
Μετέπειτα κατάσταση:	Το σύστημα παραμένει στη σελίδα του θέματος εμφανίζοντας το τροποποιημένο μήνυμα.	

Πίνακας 3.20: Σενάριο χρήσης 6 - Βασική ροή



Σχήμα 3.9: Σενάριο χρήσης 6 - Διάγραμμα βασικής ροής

Το <ΣΧ-6> περιέχει επίσης μία εναλλακτική ροή που μπορεί να προκύψει βάσει των επιλογών του χρήστη και η οποία περιγράφεται στον πίνακα 3.21.

Εναλλακτική ροή 1: Ο χρήστης πατάει το κουμπί «Άκυρο».		
Εφόσον ο χρήστης στη γραμμή 2 της Βασικής Ροής επιλέξει «Άκυρο» το σύστημα επιστρέφει στη σελίδα του θέματος.		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί «Άκυρο»	Το σύστημα επιστρέφει στη σελίδα του θέματος.
Το σενάριο χρήσης τερματίζεται.		

Πίνακας 3.21: Σενάριο χρήσης 6 - Εναλλακτική ροή 1

3.6.7 Σενάριο χρήσης 7: Ψήφιση σε ψηφοφορία

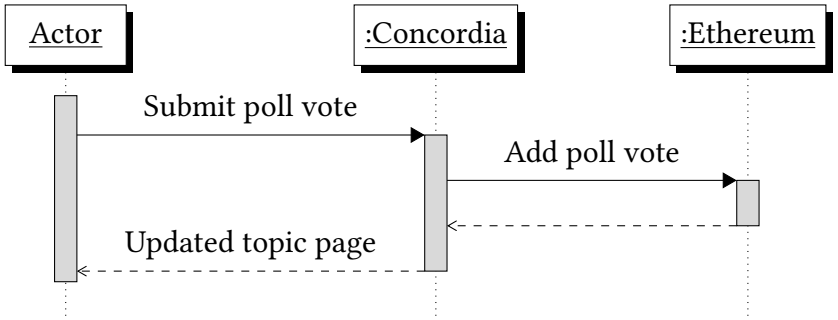
Το σενάριο χρήσης 7, <ΣΧ-7>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για την ψήφιση σε μία ψηφοφορία. Στους πίνακες 3.22 και 3.23 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-7> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.10 φαίνεται το διάγραμμα της βασικής ροής.

Ψηφίζω σε ψηφοφορία	
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο χρήστης να μπορεί να ψηφίσει σε μία ψηφοφορία.
Αναφορά ΛΑ	**<ΛΑ-10>**
Αναφορά ΜΛΑ	**<ΜΛΑ-2>**
Πυροδότηση δραστηριότητας	Ο χρήστης πατάει το κουμπί ψηφοφορίας.
Προϋπόθεση	Ο χρήστης να έχει συνδεθεί στην εφαρμογή και να βρίσκεται στην σελίδα ενός θέματος το οποίο περιλαμβάνει ψηφοφορία.

Πίνακας 3.22: Σενάριο χρήσης 7, ψήφιση σε ψηφοφορία.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί της επιλογής που επιθυμεί να ψηφίσει και πατάει το κουμπί «Υποβολή».	Το σύστημα εισάγει νέα ψήφο στο blockchain.
Μετέπειτα κατάσταση:	Το σύστημα ανανεώνει τις πληροφορίες της ψηφοφορίας.	

Πίνακας 3.23: Σενάριο χρήσης 7 - Βασική ροή



Σχήμα 3.10: Σενάριο χρήσης 7 - Διάγραμμα βασικής ροής

3.6.8 Σενάριο χρήσης 8: Ψήφιση μηνύματος

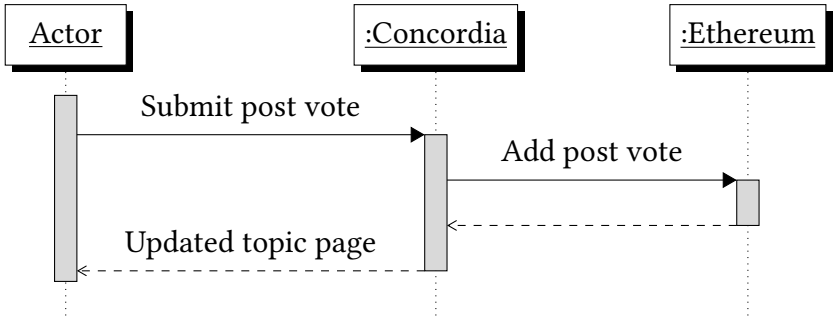
Το σενάριο χρήσης 8, <ΣΧ-8>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για την ψήφιση σε ένα μήνυμα. Στους πίνακες 3.24 και 3.25 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-8> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.11 φαίνεται το διάγραμμα της βασικής ροής.

Ψηφίζω σε μήνυμα	
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο χρήστης να μπορεί να υπερψηφίσει ή καταψηφίσει ένα μήνυμα.
Αναφορά ΛΑ	**<ΛΑ-8>**
Αναφορά ΜΛΑ	**<ΜΛΑ-2>**
Πυροδότηση δραστηριότητας	Ο επισκέπτης πατάει το κουμπί υπερψήφισης ή καταψήφισης.
Προϋπόθεση	Ο χρήστης να έχει συνδεθεί στην εφαρμογή και να βρίσκεται στην σελίδα ενός θέματος το οποίο περιλαμβάνει τουλάχιστον ένα μήνυμα το οποίο δεν έχει δημιουργήσει ο ίδιος.

Πίνακας 3.24: Σενάριο χρήσης 8, ψήφιση μηνύματος.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει στο κουμπί υπερψήφισης μηνύματος.	Το σύστημα εισάγει νέα ψήφο μηνύματος στο blockchain.
Μετέπειτα κατάσταση:	Το σύστημα ανανεώνει τις ψήφους του μηνύματος.	

Πίνακας 3.25: Σενάριο χρήσης 8 - Βασική ροή



Σχήμα 3.11: Σενάριο χρήσης 8 - Διάγραμμα βασικής ροής

3.6.9 Σενάριο χρήσης 9: Διαγραφή τοπικών δεδομένων

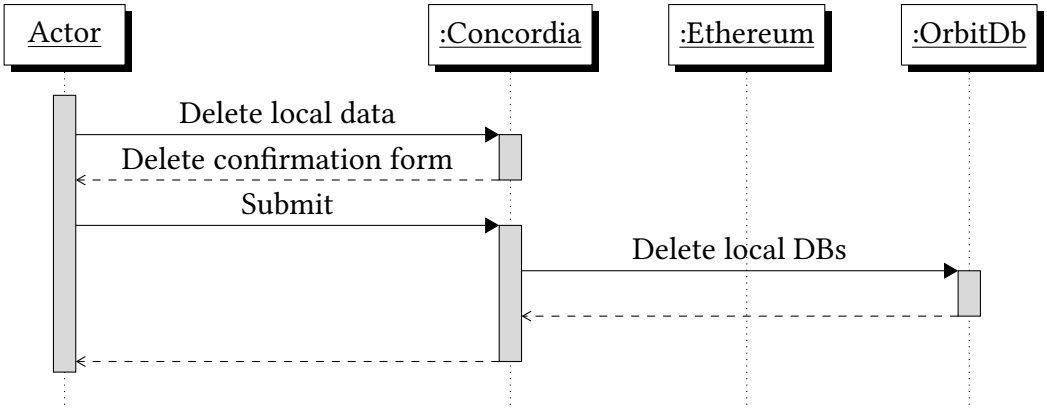
Το σενάριο χρήσης 9, <ΣΧ-9>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για τη διαγραφή των τοπικών δεδομένων. Στους πίνακες 3.26 και 3.27 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-9> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.12 φαίνεται το διάγραμμα της βασικής ροής.

Διαγράφω τα τοπικά δεδομένα	
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο επισκέπτης να μπορεί να διαγράψει τα τοπικά δεδομένα που αποθηκεύονται στο σύστημά του από την εφαρμογή.
Αναφορά ΛΑ	**<ΛΑ-11>**
Αναφορά ΜΛΑ	-
Πυροδότηση δραστηριότητας	Ο επισκέπτης πατάει το κουμπί διαγραφής των τοπικών δεδομένων.
Προϋπόθεση	Ο επισκέπτης πρέπει να έχει ανοίξει την σελίδα της εφαρμογής.

Πίνακας 3.26: Σενάριο χρήσης 9, διαγραφή τοπικών δεδομένων.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο επισκέπτης πατάει το κουμπί διαγραφής των τοπικών δεδομένων.	Το σύστημα εμφανίζει την φόρμα «Επιβεβαίωση Διαγραφής Τοπικών Δεδομένων».
2	Ο επισκέπτης συμπληρώνει το πεδίο και πατάει το κουμπί «Υποβολή».	Το σύστημα διαγράφει όλες τις τοπικές βάσεις OrbitDb που χρησιμοποιούνται από την εφαρμογή.
Μετέπειτα κατάσταση:	Το σύστημα παραμένει πραγματοποιεί ανανέωση της σελίδας.	

Πίνακας 3.27: Σενάριο χρήσης 9 - Βασική ροή



Σχήμα 3.12: Σενάριο χρήσης 9 - Διάγραμμα βασικής ροής

3.6.10 Σενάριο χρήσης 10: Δημιουργία κοινότητας

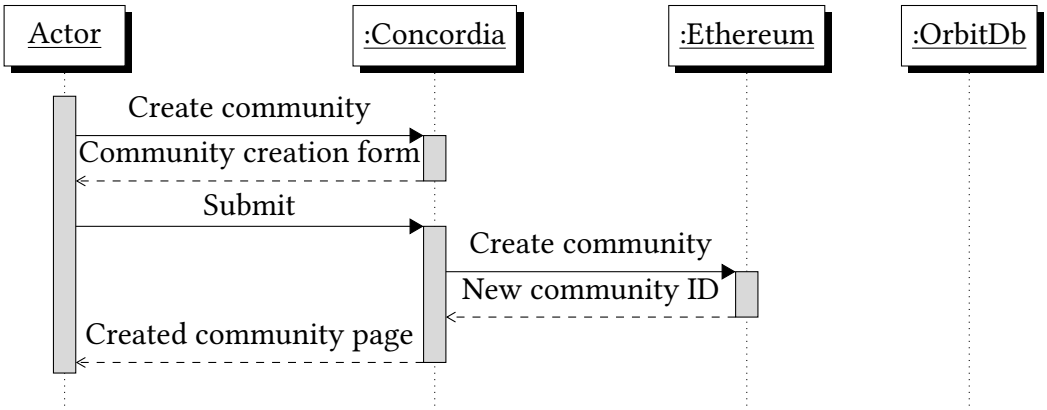
Το σενάριο χρήσης 10, <ΣΧ-10>, περιγράφει τις διαδοχικές ενέργειες που εκτελούνται για την δημιουργία μίας κοινότητας. Στους πίνακες 3.28 και 3.29 παρατίθενται οι βασικές πληροφορίες του <ΣΧ-10> και οι ενέργειες της βασικής ροής αντίστοιχα, ενώ στο σχήμα 3.13 φαίνεται το διάγραμμα της βασικής ροής.

	Δημιουργώ νέα κοινότητα
Σύντομη περιγραφή	Στόχος του σεναρίου χρήσης είναι ο χρήστης να μπορεί να δημιουργήσει νέα κοινότητα.
Αναφορά ΛΑ	**<ΛΑ-12>**, **<ΛΑ-13>**
Αναφορά ΜΛΑ	**<ΜΛΑ-2>**
Πυροδότηση δραστηριότητας	Ο χρήστης πατάει το κουμπί δημιουργίας νέας κοινότητας.
Προϋπόθεση	Ο χρήστης να έχει συνδεθεί στην εφαρμογή και να βρίσκεται στην αρχική σελίδα.

Πίνακας 3.28: Σενάριο χρήσης 10, δημιουργία νέας κοινότητας.

Βασική ροή		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί δημιουργίας νέας κοινότητας.	Το σύστημα εμφανίζει την φόρμα «Δημιουργία Κοινότητας».
2	Ο χρήστης συμπληρώνει τα πεδία και πατάει το κουμπί «Υποβολή».	Το σύστημα δημιουργεί νέα κοινότητα στο blockchain.
Μετέπειτα κατάσταση:	Το σύστημα μεταβαίνει στην σελίδα της νέας κοινότητας.	

Πίνακας 3.29: Σενάριο χρήσης 10 - Βασική ροή

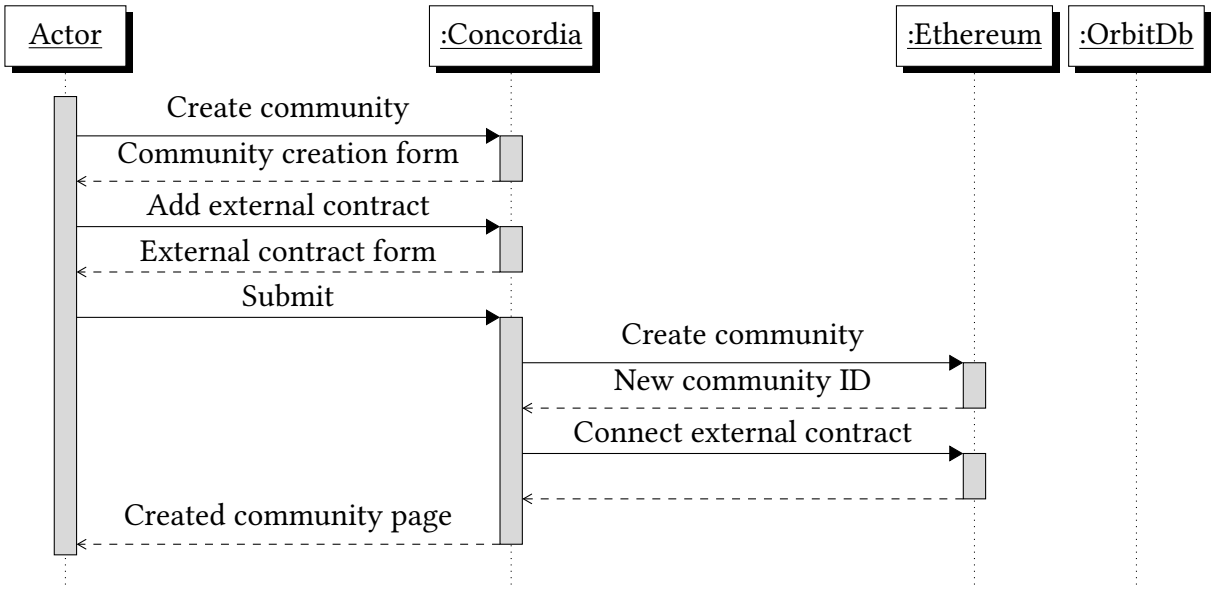


Σχήμα 3.13: Σενάριο χρήσης 10 - Διάγραμμα βασικής ροής

Το <ΣΧ-10> περιέχει επίσης δύο εναλλακτικές ροές που μπορεί να προκύψουν βάσει των επιλογών του χρήστη και οι οποίες περιγράφονται στους πίνακες 3.30 και 3.31. Η εναλλακτική ροή 1 φαίνεται επίσης στο σχήμα 3.14 όπου παρουσιάζεται το διάγραμμα ροής της.

Εναλλακτική ροή 1: Ο χρήστης ορίζει εξωτερικό contract για την κοινότητα.		
Εφόσον ο χρήστης στη γραμμή 2 της Βασικής Ροής επιλέξει «Προσθήκη Συμβολαίου» το σύστημα ανανεώνει την σελίδα προσθέτοντας τα επιπλέον πεδία της φόρμας «Σύνδεση Συμβολαίου».		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης, αφού συμπληρώσει τη φόρμα «Δημιουργία Κοινότητας», πατάει το κουμπί «Προσθήκη ψηφοφορίας»	Το σύστημα ανανεώνει τη σελίδα με τα πεδία της φόρμας «Σύνδεση Συμβολαίου».
2	Ο χρήστης συμπληρώνει τα πεδία και πατάει το κουμπί «Υποβολή».	Το σύστημα δημιουργεί την νέα κοινότητα στο blockchain και την συνδέει με το εξωτερικό contract.
Το σύστημα μεταβαίνει στην σελίδα της νέας κοινότητας.		

Πίνακας 3.30: Σενάριο χρήσης 10 - Εναλλακτική ροή 1



Σχήμα 3.14: Σενάριο χρήσης 3 - Διάγραμμα εναλλακτικής ροής 1

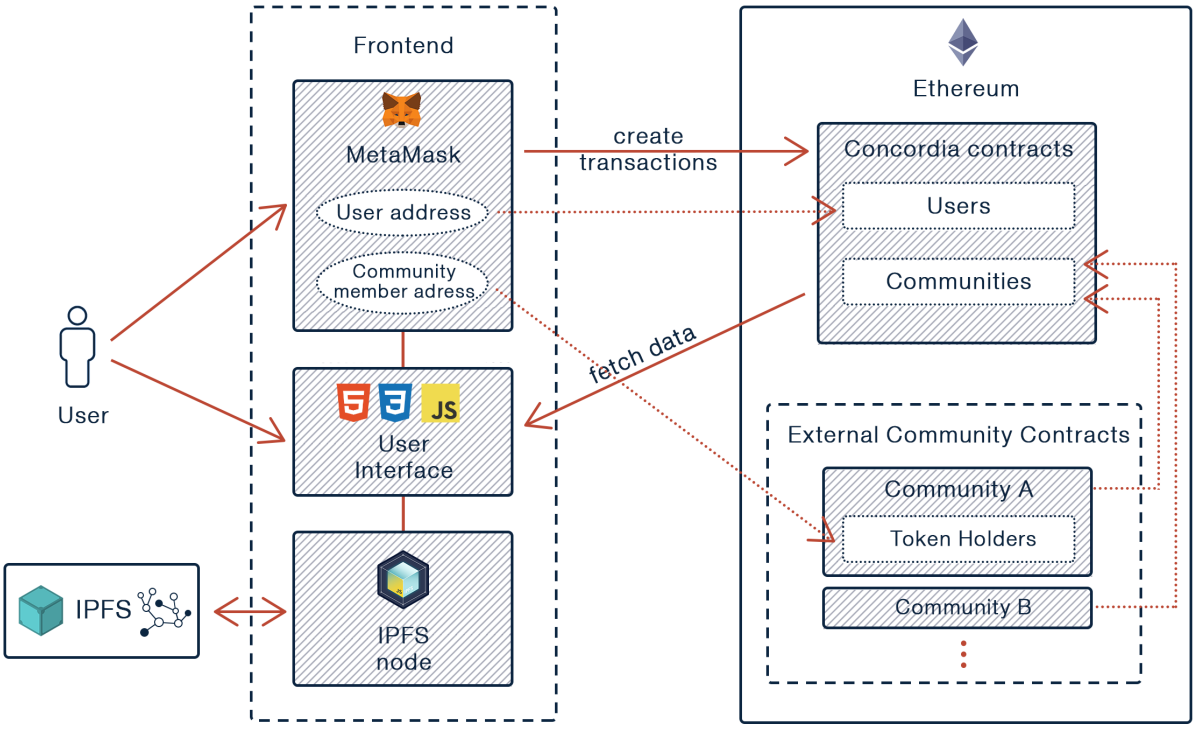
Εναλλακτική ροή 2: Ο χρήστης πατάει το κουμπί «Άκυρο».		
Εφόσον ο χρήστης στη γραμμή 2 της Βασικής Ροής ή στη γραμμή 2 της Εναλλακτικής Ροής 1 επιλέξει «Άκυρο» το σύστημα επιστρέφει στην αρχική σελίδα της εφαρμογής.		
Γραμμή	**Ενέργεια χρήστη συστήματος**	**Απάντηση Συστήματος**
1	Ο χρήστης πατάει το κουμπί «Άκυρο»	Το σύστημα επιστρέφει στην αρχική σελίδα της εφαρμογής.
Το σενάριο χρήσης τερματίζεται.		

Πίνακας 3.31: Σενάριο χρήσης 10 - Εναλλακτική ροή 2

3.7 Αρχιτεκτονική σχεδίαση

Στο κεφάλαιο αυτό περιγράφεται η αρχιτεκτονική του συστήματος, όπως προέκυψε από την επιλεγμένη τεχνολογική στοίβα και τις προαναφερθείσες απαιτήσεις του. Θα πρέπει να σημειωθεί ότι η παρουσιαζόμενη αρχιτεκτονική είναι πρώιμη και δεν αποτελεί την τελική υλοποίηση της πλατφόρμας, η οποία περιγράφεται στο κεφάλαιο 4.

Συνοπτικά, η αρχιτεκτονική του συστήματος αποτυπώνεται στο παρακάτω διάγραμμα:



Σχήμα 3.15: Αρχιτεκτονική του συστήματος (στάδιο σχεδίασης)

Αξίζει να σημειωθούν τα εξής:

- Ο κώδικας του frontend εκτελείται αποκλειστικά στο σύστημα του χρήστη, χωρίς να απαιτείται κάποιος εξυπηρετητής. Δηλαδή, ο χρήστης αρκεί απλά να έχει τον κώδικα αποθηκευμένο στον υπολογιστή του.

- Ο χρήστης αλληλεπιδρά άμεσα με το UI και το MetaMask. Το MetaMask αποτελεί browser add-on, το οποίο διαχειρίζεται τα ιδιωτικά κλειδιά Ethereum του χρήστη και πραγματοποιεί τις συναλλαγές του τελευταίου με τα smart contracts. Στην προκειμένη περίπτωση, περιέχει τα κλειδιά που σχετίζονται αφενός με τη διεύθυνση με την οποία ο χρήστης εγγράφεται στην πλατφόρμα, αφετέρου με τις διευθύνσεις που περιέχουν τα NFTs των κοινοτήτων στις οποίες ανήκει και έχει δικαιώματα ψήφου.

- Στο frontend εκτελείται στο παρασκήνιο ένας κόμβος για το IPFS. Αυτός συνδέεται με άλλους κατάλληλους κόμβους, διαμοιράζοντας τον κύριο όγκο των δεδομένων της εφαρμογής (π.χ. του περιεχομένου των μηνυμάτων).

- Τέλος, στο Ethereum blockchain υπάρχουν τόσο τα contracts της εφαρμογής, όσο και τα εξωτερικά contracts που παρέχουν τα tokens των κοινοτήτων. Τα μεν λειτουργούν ως το σημείο αναφοράς της εφαρμογής, επί του οποίου εκτελούνται οι ενέργειες και αποθηκεύονται οι μεταβλητές που είναι απολύτως απαραίτητες για τη λειτουργία της πλατφόρμας (π.χ. εγγεγραμμένοι χρήστες, δημιουργημένες κοινότητες). Τα δε, δημιουργούνται από εξωτερικές οντότητες, οι οποίες ορίζουν κατά τη βούλησή τους τον ακριβή τρόπο δημιουργίας και διαμοιρασμού των tokens τους στους χρήστες.

3.8 Προδιαγραφή μεθόδου υλοποίησης και χρονοπρογραμματισμός

Κατά τον χρονοπρογραμματισμό ακολουθήθηκαν οι τακτικές που ορίζει το Scrum. Το συνολικό προγραμματιστικό έργο χωρίστηκε σε επιμέρους, διακριτούς στόχους και κάθε στόχος αντιστοιχήθηκε σε ένα Sprint. Τα Sprints αποτελούνται από επιμέρους διαχωρισμό της εργασίας σε epic tasks. Σε αυτό το στάδιο χρονοπρογραμματισμού δεν έγινε αναλυτικότερη περιγραφή των επιμέρους tasks, κάθε epic χωρίστηκε σε tasks κατά το αρχικό στάδιο της υλοποίησης του.

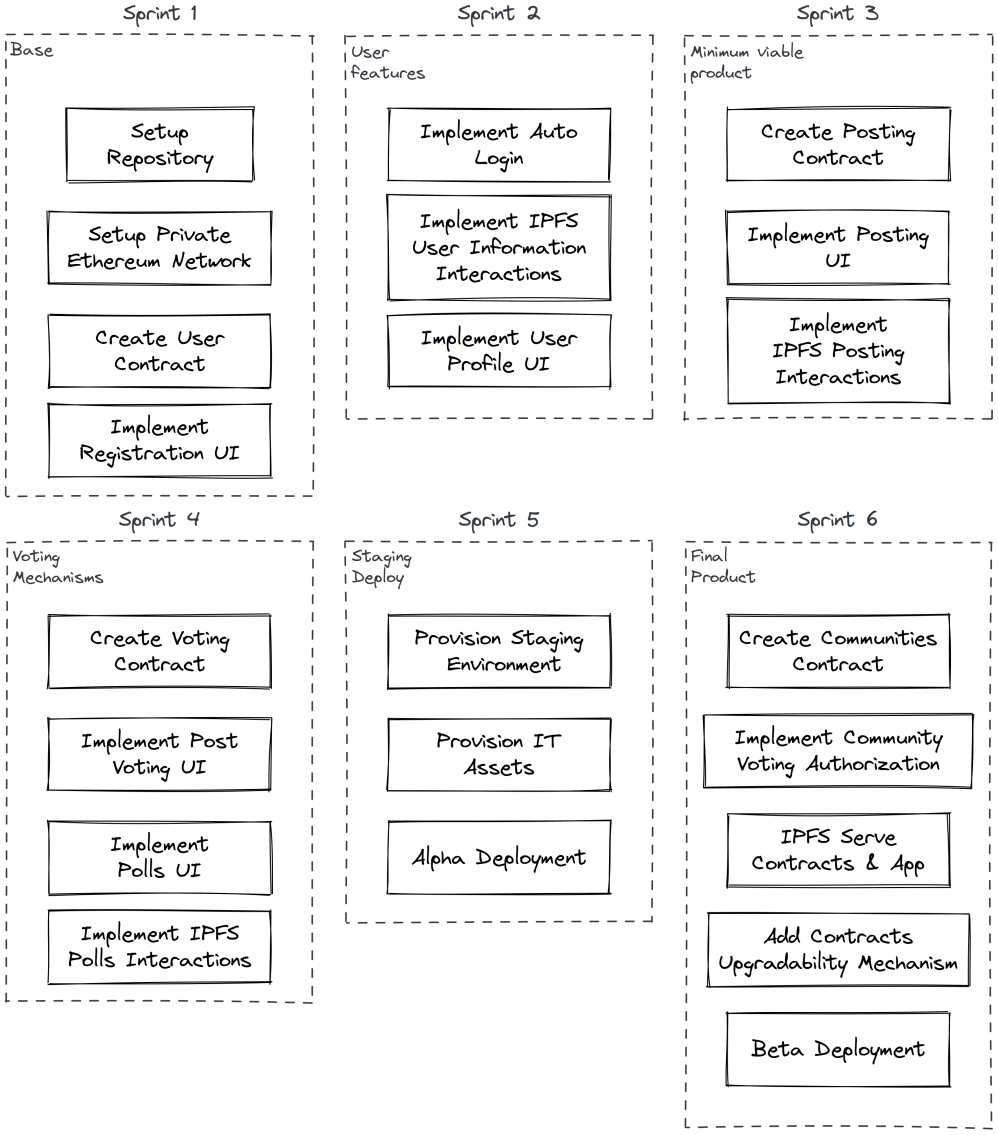
Ως σημαντικότερος στόχος της ανάπτυξης ορίζεται η δημιουργία ενός ελάχιστου βιώσιμου προϊόντος (Minumum Viable Product - MVP). Σε αυτό τον στόχο περιλαμβάνονται πιο στοιχειώδεις λειτουργίες μίας πλατφόρμας επικοινωνίας οι οποίες την κάνουν χρήσιμη, η δυνατότητα εγγραφής, δημιουργίας θεμάτων και μηνυμάτων και ανάγνωσης του υπάρχοντος περιεχομένου. Επειδή ο στόχος αυτός περιέχει από μόνος του σημαντική περιπλοκότητα και δυσκολία κρίθηκε αναγκαίος ο περαιτέρω διαχωρισμός του σε τρία Sprints.

Στο πρώτο Sprint ορίστηκε ο στόχος της δημιουργίας μίας βάσης κώδικα (codebase), της εξοικείωσης με τα προγραμματιστικά εργαλεία του οικοσυστήματος των DApps και της επιτυχής δημιουργίας του πρώτου contract. Στο δεύτερο Sprint ο στόχος ορίστηκε ως η δημιουργία

των τεχνικών χαρακτηριστικών που αφορούν τους χρήστες της πλατφόρμας και που οι ίδιοι (οι χρήστες) έχουν συνηθίσει να περιμένουν από μία τέτοια πλατφόρμα. Στο τρίτο Sprint συμπεριλήφθηκαν τα τεχνικά χαρακτηριστικά που απομένουν ώστε να δημιουργηθεί το MVP.

Τα επόμενα τρία Sprints χτίζουν διαδοχικά πάνω στην υπάρχουσα δουλειά και υποδομή. Στο τέταρτο μέρος εργασίας ως στόχος ορίστηκε η προσθήκη των χαρακτηριστικών ψηφοφορίας πάνω στα μηνύματα και δημιουργίας ψηφοφοριών θεμάτων (polls). Το επόμενο Sprint περιλαμβάνει εργασίες δημιουργίας υποδομής και την πρώτη ημι-δημόσια εγκατάσταση της εφαρμογής σε περιβάλλον δοκιμής. Το τελευταίο Sprint αποτελεί το τελικό προϊόν και περιέχει tasks σχετικά με την δημιουργία κοινοτήτων και την beta εγκατάσταση της εφαρμογής.

Εποπτικά, η διαδικασία της υλοποίησης περιγράφεται στο παρακάτω σχήμα (σχήμα 3.16).



Σχήμα 3.16: Διαχωρισμός σε sprints

Κεφάλαιο 4

Υλοποίηση εφαρμογής

Στο παρόν κεφάλαιο αναλύονται οι μέθοδοι και οι τεχνολογίες που αναπτύχθηκαν και χρησιμοποιήθηκαν τόσο για την υλοποίηση της ίδιας της εφαρμογής Concordia, όσο και για τη διαμορφώση του συνολικού προγραμματιστικού περιβάλλοντος επί του οποίου αυτή δημιουργήθηκε.

4.1 Μεθοδολογία υλοποίησης

Για την επίτευξη των στόχων που ορίστηκαν και την οργάνωση της εργασίας που απαιτείται σε διαχειρίσιμα μέρη, χρησιμοποιήθηκαν διάφορα εργαλεία και μέθοδοι ανάπτυξης λογισμικού, όπως το σύστημα ελέγχου εκδόσεων (version control system) Git, η μέθοδος οργάνωσης Scrum και οι διαδικασίες ανάπτυξης DevOps. Τα εργαλεία αυτά είναι δοκιμασμένα και έχουν εδραιωθεί στη σύγχρονη ανάπτυξη λογισμικού.

Μέσα από την χρήση των παραπάνω εργαλείων επιτυγχάνεται η ομαλή συνεργασία στην ανάπτυξη του λογισμικού. Κάθε μέλος της ομάδας δύναται να εργαστεί ανεξάρτητα και χωρίς την ανάγκη διαρκούς επικοινωνίας με τα υπόλοιπα μέλη. Οι στόχοι είναι ορισμένοι, σαφείς και χωρισμένοι σε διαχειρίσιμα μέρη τα οποία δεν καταβάλλουν τα μέλη. Ταυτόχρονα, έχοντας ως έδρα καθιερωμένα πρότυπα ανάπτυξης, παρέχεται φορμαλισμός και έτοιμες μέθοδοι επίλυσης προβλημάτων, γεγονός που λειτουργεί καταλυτικά και βοηθά στην αποφυγή τελμάτων κατά τη συγγραφή του κώδικα.

Το Git είναι δωρεάν λογισμικό ανοιχτού κώδικα το οποίο επιτρέπει και επικουρεί την απρόσκοπτη ανάπτυξη λογισμικού από πολλαπλά μέλη μίας ομάδας, ταυτόχρονα και διανεμημένα. Αυτό επιτυγχάνεται παρέχοντας ένα πλαίσιο από εργαλεία τα οποία βοηθούν την διαχείριση και ενσωμάτωση των διαφορετικών εκδόσεων του κώδικα τις οποίες αναπτύσσει κάθε μέλος της ομάδας ξεχωριστά. Υπάρχουν διάφορα μοντέλα χρήσης του Git και πιο συγκεκριμένα της δυνατότητας που δίνει για δημιουργία, ανάπτυξη και ένωση (merge) κλαδιών (branches).

Για τους σκοπούς της παρούσας διπλωματικής χρησιμοποιήθηκε το μοντέλο GitHub flow.[16] Το μοντέλο αυτό ορίζει ότι κάθε προγραμματιστής ανοίγει ένα νέο branch για τη ανάπτυξη ενός χαρακτηριστικού της εφαρμογής ή τη διόρθωση ενός μέρους του κώδικα. Έπειτα, όταν η δουλειά

έχει ολοκληρωθεί, δημιουργείται ένα αίτημα ένωσης (pull request) με το βασικό κλαδί ανάπτυξης (develop) της εφαρμογής. Η δουλειά υπόκειται σε αξιολόγηση από την υπόλοιπη ομάδα (review) και όταν κριθεί ότι ικανοποιεί τις ανάγκες του έργου, το branch γίνεται merge με το develop. Όταν το develop φτάσει σε ικανό σημείο σταθερότητας και αλλαγών, γίνεται merge με το branch παραγωγής (master). Από το master δημιουργούνται οι τελικές εκδόσεις της εφαρμογής οι οποίες διανέμονται για χρήση στην παραγωγή (production versions), ενώ από το develop δημιουργούνται οι δοκιμαστικές εκδόσεις αιχμής της εφαρμογής οι οποίες χρησιμοποιούνται κατά τον έλεγχο (staging versions).

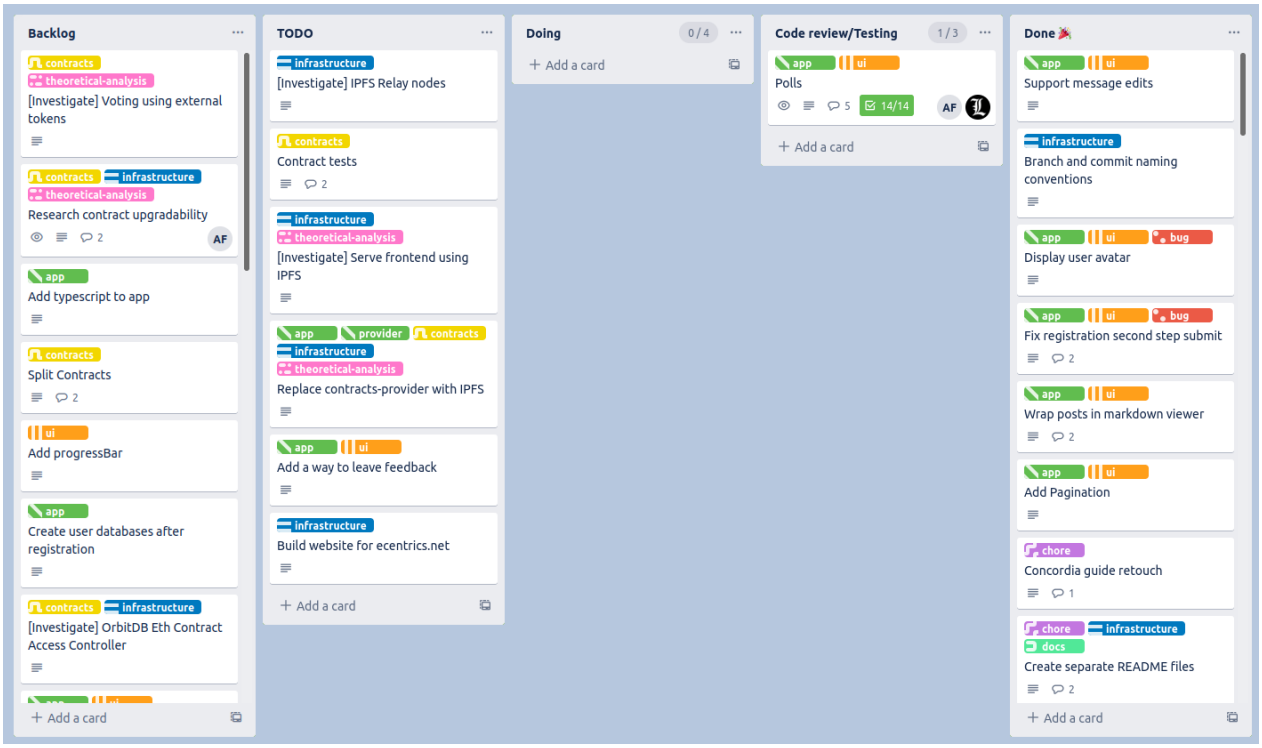
Το Scrum είναι μία μέθοδος οργάνωσης στην οποία ο επιμελητής του Scrum (Scrum master) διαχωρίζει τα ανεξάρτητα μέρη εργασίας (tasks) που πρέπει να υλοποιηθούν για την ολοκλήρωση των στόχων ενός project. Τα μέρη αυτά περιγράφονται αναλυτικά μαζί με τις απαιτήσεις τους και κατατίθενται σε μία λίστα εργασιών (backlog). Έπειτα, μέσα από συσκέψεις (meetings), επιλέγεται ένας αριθμός από tasks τα οποία ορίζουν το επόμενο προγραμματιστικό κύκλο (sprint). Κάθε task ανατίθεται σε κάποιο μέλος για υλοποίηση. Για το Sprint ορίζεται μία χρονική διάρκεια, στόχος της οποίας είναι η περάτωση όλων των tasks πριν τη λήξη της. Στο τέλος της προθεσμίας που ορίστηκε για το Sprint τα μέλη της ομάδας αποτιμούν τα αποτελέσματα και ορίζουν το επόμενο Sprint. Η διαδικασία επαναλαμβάνεται έως ότου το έργο ολοκληρωθεί.

Λόγω του πολύ μικρού μεγέθους της ομάδας, το Scrum ακολουθήθηκε ελαστικά. Συγκεκριμένα, δεν ορίστηκε ένας συγκεκριμένος επιμελητής του board αλλά κάθε μέλος της ομάδας φρόντιζε για τον ορισμό και την περιγραφή ενός μέρους των tasks. Τα sprints δεν ήταν συνεχόμενα και δεν είχαν πάντα τον ίδιο χρόνο εκτέλεσης αλλά προσαρμόζονταν ανάλογα με τις εκάστοτε ανάγκες και τον χρόνο των μελών. Κατά βάση, χρησιμοποιήθηκε η μέθοδος Kanban (που χρησιμοποιείται από το ίδιο το Scrum), για την οπτικοποίηση των tasks. Τα tasks χωρίστηκαν κατά κύριο λόγο στις παρακάτω λίστες:

- «Αναμονής» (backlog), η οποία περιέχει tasks τα οποία δεν έχουν ακόμα εισαχθεί σε κάποιο sprint
- «Ενεργού sprint» (sprint/todo), που περιλαμβάνει tasks τα οποία συμμετέχουν στο ενεργό (τρέχον) sprint
- «Εκτέλεσης» (in progress/doing), η οποία περιλαμβάνει tasks για τα οποία έχει ξεκινήσει η ανάπτυξη από κάποιο μέλος της ομάδας
- «Ελέγχου και αξιολόγησης» (testing/code review), η οποία περιέχει tasks των οποίων η ανάπτυξη έχει ολοκληρωθεί και βρίσκονται στο στάδιο ελέγχου (testing) ή αναμονής σε pull request
- «Ολοκλήρωσης» (done), που περιλαμβάνει tasks τα οποία έχουν τελειώσει, δηλαδή των οποίων η ανάπτυξη έχει ολοκληρωθεί και το pull request έχει γίνει merge

Επίσης, ορίστηκαν στις λίστες οι μέγιστοι αριθμοί από tasks που μπορούν τα υπάρχουν σε κάθε χρονική στιγμή (π.χ. μέχρι τέσσερα tasks στην λίστα εκτέλεσης). Αυτό έγινε για ενθάρρυνση της ολοκλήρωσης των tasks από τα μέλη, σε αντίθεση με την εγκατάλειψή τους σε ημιτελή κατάσταση της ανάπτυξης για την ανάληψη κάποιου νέου task.

Για την υλοποίηση του Scrum χρησιμοποιήθηκε η διαδικτυακή υπηρεσία Trello[15], στιγμιότυπο της οποίας φαίνεται στο παρακάτω σχήμα:



Σχήμα 4.1: Στιγμιότυπο οθόνης της διαδικτυακής υπηρεσίας Trello

Κατά την διαδικασία της ανάπτυξης του κώδικα, εφαρμόστηκαν επίσης οι τακτικές που ορίζονται από το DevOps σε ό,τι αφορά το deployment των υπηρεσιών. Το DevOps ορίζει διάφορα εργαλεία που αποσκοπούν στην απρόσκοπτη, αυτοματοποιημένη και γρήγορα ενσωμάτωση του κώδικα από το στάδιο της συγγραφής μέχρι την ολοκλήρωση και εγκατάσταση. Τα εργαλεία που χρησιμοποιήθηκαν εδώ είναι:

- Συνεχής έλεγχος (continuous testing)
- Συνεχής ολοκλήρωση (continuous integration)
- Συνεχής παράδοση (continuous delivery)
- Συνεχής εγκατάσταση (continuous deployment)

Για την υλοποίηση των τακτικών αυτών επιλέχθηκε μετά από εκτενή έρευνα η πλατφόρμα Jenkins. Το Jenkins συνδυάστηκε με την πλατφόρμα εικονοποίησης Docker ώστε να ακολουθηθούν οι τελευταίες ενδεδειγμένες πρακτικές της βιομηχανίας. Έγινε συγγραφή του αρχείου

[15] `https://trello.com/`

Jenkinsfile το οποίο περιγράφει με κώδικα την ροή εργασιών (pipeline) που πρέπει να ακολουθηθεί μετά από κάθε αλλαγή στον κώδικα. Η εκτέλεση του pipeline πραγματοποιείται αυτόματα από το Jenkins.

Το pipeline αποτελείται από στάδια και βήματα τα οποία φαίνονται στο σχήμα 4.2:

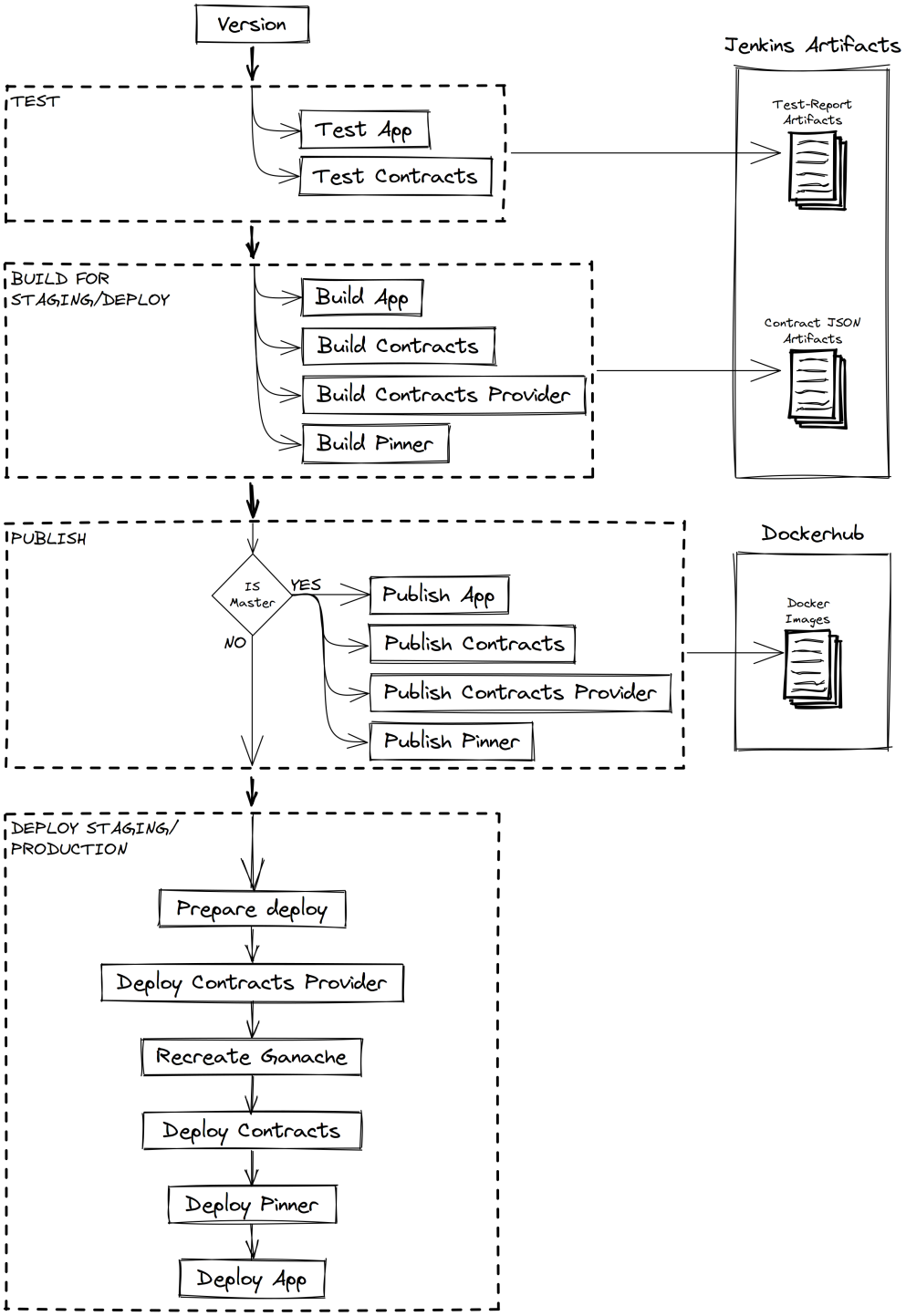
α) Αρχικά εκτελείται το βήμα «Version», το οποίο συλλέγει στοιχεία σχετικά με την εκτέλεση του pipeline όπως το κλαδί του κώδικα που πυροδότησε τη ροή και ποια από τα πακέτα λογισμικού που περιλαμβάνονται στο git repository περιέχουν αλλαγές.

β) Έπειτα εκτελείται το στάδιο «TEST» το οποίο περιέχει δύο βήματα που εκτελούνται παράλληλα και πραγματοποιούν έλεγχο του κώδικα των πακέτων.

γ) Αν το κλαδί πυροδότησης είναι ένα feature branch η ροή σταματά εδώ, ενώ αν πρόκειται για ένα από τα βασικά κλαδιά (master ή develop) τότε η ροή συνεχίζει με το στάδιο «BUILD» στο οποίο εκτελούνται παράλληλα τα βήματα που χτίζουν τα docker images των πακέτων εκείνων τα οποία περιέχουν αλλαγές.

δ) Στο στάδιο «PUBLISH», αν το κλαδί πυροδότησης είναι το κύριο κλαδί παραγωγής (master), τότε εκτελούνται παράλληλα βήματα τα οποία δημοσιεύουν τα docker images που δημιουργήθηκαν στο αποθετήριο Dockerhub.

ϵ) Τέλος, εκτελείται το στάδιο «DEPLOY», κατά το οποίο πραγματοποιείται η εγκατάσταση των υπηρεσιών στο ανάλογο περιβάλλον, staging για το κλαδί develop και production για το κλαδί master.



Σχήμα 4.2: Διάγραμμα ροής εργασιών Jenkins

Με τη χρήση του Jenkins αυτοματοποιείται με μεγάλη ευκολία ένα σημαντικό μέρος των διαδικασιών ανάπτυξης και δημοσίευσης του κώδικα. Με τη χρήση του συγκεκριμένου pipeline γίνεται σίγουρο ό,τι σε κάθε αλλαγή, ασχέτως του κλαδιού ανάπτυξης ο κώδικας ελέγχεται και τα αποτελέσματα των tests είναι αποθηκευμένα και διαθέσιμα για ανάλυση. Ακόμα, για το κλαδί develop, αυτοματοποιείται η ολοκλήρωση των πακέτων και η εγκατάστασή τους σε περιβάλλον δοκιμής (staging), γεγονός που διευκολύνει σημαντικά τις συλλογικές δοκιμές από την ομάδα

σε διαφορετικά περιβάλλοντα χρήσης (browsers). Τέλος, για το κλαδί master, αυτοματοποιείται η διαδικασία δημοσίευσης των docker images, μηδενίζοντας έτσι τον χρόνο που πρέπει να καταβάλουν τα μέλη της ομάδας σε αυτό.

4.2 Τεχνολογίες υλοποίησης

Η παρούσα ενότητα απαρτίζεται από υποενότητες, στις οποίες διατυπώνονται οι **σημαντικότερες** τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Όλες οι τεχνολογίες αποτελούν δωρεάν λογισμικό ανοιχτού κώδικα.

4.2.1 Τεχνολογίες σχετικές με το development

Σε αυτήν την υποενότητα περιγράφονται ορισμένα θεμελιώδη εργαλεία και frameworks που συνετέλεσαν στην ανάπτυξη της εφαρμογής.

Node.js

Σχήμα 4.3: Node.js logo

Το Node.js[16] είναι ένα περιβάλλον χρόνου εκτέλεσης JavaScript πολλαπλών πλατφορμών, το οποίο εκτελείται στη μηχανή V8[17] και παρέχει τη δυνατότητα εκτέλεσης κώδικα JavaScript εκτός περιηγητών ιστού. Επιτρέπει στους προγραμματιστές να χρησιμοποιούν JavaScript για τη σύνταξη εργαλείων γραμμής εντολών και τη δημιουργία κλιμακωτών διαδικτυακών εφαρμογών (κυρίως για εξυπηρετητές). Έχει αρχιτεκτονική βασισμένη σε συμβάντα (event-driven architecture), με δυνατότητα ασύγχρονης εισόδου/εξόδου (asynchronous I/O).[17]

Ένα από τα σημαντικότερα χαρακτηριστικά του Node.js είναι ο ενσωματωμένος διαχειριστής πακέτων του, ο οποίος ονομάζεται npm. Με τον npm γίνεται εφικτή η εγκατάσταση πακέτων (βιβλιοθηκών) από το μητρώο npm (npm registry[18]), καθώς και η οργάνωση και η διαχείρισή τους στα πλαίσια της ανάπτυξης μίας εφαρμογής που εξαρτάται από αυτά.

Το Node.js έχει το αποθετήριό του στο GitHub [19].

[16] `https://nodejs.org/`
[17] `https://v8.dev/`
[18] `https://www.npmjs.com/`
[19] `https://github.com/nodejs/node`

Docker

Σχήμα 4.4: Docker logo

Το Docker αποτελεί μία πλατφόρμα η οποία παρέχει λογισμικό εικονοποίησης (virtualization) στο επίπεδο του λειτουργικού συστήματος καθώς και ολοκληρωμένα συστήματα διαμοιρασμού και εκτέλεσης των παραγόμενων εικόνων.

Δίνει την δυνατότητα σύνθεσης εικονικών περιβαλλόντων λειτουργικού συστήματος τα οποία ονομάζονται εικόνες (images). Μέσα στις εικόνες είναι δυνατή η εκτέλεση προγραμμάτων σε ασφαλή, απομονωμένα και προβλέψιμα περιβάλλοντα τα οποία εγγυούνται τις ίδιες συνθήκες εκτέλεσης παντού. Έτσι, οι προγραμματιστές δεν χρειάζεται να ανησυχούν για το περιβάλλον εκτέλεσης του κώδικα και την ρύθμιση των παραμέτρων σε κάθε ξεχωριστή εγκατάσταση.

Ταυτόχρονα, η πλατφόρμα του Docker παρέχει συστήματα και τυποποιημένες μεθόδους για το πακετάρισμα των εικόνων, την μεταφόρτωση και την εκτέλεσή τους σε απομακρυσμένα συστήματα. Με αυτό τον τρόπο αποτελεί πολύτιμο εργαλείο το οποίο έχει γίνει το στάνταρ στη βιομηχανία λογισμικού για τον διαμοιρασμό και την εγκατάσταση ολοκληρωμένων εφαρμογών σε περιβάλλοντα δοκιμής (staging environments) και παραγωγής (production environment).

Τέλος, η δυνατότητα τοπικής εκτέλεσης των εικόνων στο σύστημα ανάπτυξης του κώδικα δίνει την ευκαιρία ελέγχου (testing) και αποσφαλμάτωσης (debug) τοπικά σε ένα περιβάλλον ίδιο με αυτό της εκτέλεσης. Αυτό είναι εξαιρετικά σημαντικό επειδή αποκλείει τυχών μεταβολές στην πορεία εκτέλεσης του προγράμματος που μπορεί να έρχονταν από την εκτέλεση σε ένα διαφορετικό περιβάλλον.

Jenkins

Σχήμα 4.5: Jenkins logo

Το Jenkins είναι ένας πλήρως παραμετροποιήσιμος και επεκτάσιμος διακομιστής αυτοματοποίησης (automation server). Ο διακομιστής μπορεί να αυτοματοποιήσει τις διαδικασίες ελέγχου, ολοκλήρωσης, παράδοσης και εγκατάστασης του κώδικα, υλοποιώντας έτσι βασικές διαδικασίες που ορίζει το DevOps, συνεχή έλεγχο (continuous testing), συνεχή ολοκλήρωση (continuous integration), συνεχή παράδοση (continuous delivery) και συνεχή εγκατάσταση (continuous

deployment). Επίσης, το Jenkins μπορεί να παραμετροποιηθεί μέσω των ρυθμίσεων που προσφέρει και των επεκτάσεων (plugins) που υπάρχουν ώστε να παρέχει τις δυνατότητες αυτές για οποιαδήποτε πλατφόρμα, γλώσσα και περιβάλλον ανάπτυξης.

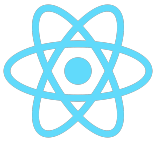
Στο Jenkins είναι δυνατός ο ορισμός με χρήση κώδικα (σε Groovy και στο DSL που παρέχεται από το Jenkins) πολλαπλών γραμμών εργασιών (pipeline). Οι γραμμές εργασιών συντίθενται από πολλαπλά βήματα τα οποία επιτελούν ξεχωριστούς στόχους προς το τελικό αποτέλεσμα της γραμμής. Τα βήματα μπορούν να τρέχουν σειριακά ή παράλληλα. Ενώ δίνεται η δυνατότητα εκτέλεσης σε πολλαπλά, διανεμημένα συστήματα καθώς και άλλες προχωρημένες λειτουργικότητες.

Το Jenkins συνδυάζεται αποτελεσματικά με την πλατφόρμα του Docker που περιγράφηκε προηγουμένως. Μέσω του συνδυασμού δίνεται η ευκαιρία της αυτοματοποίησης του μεγαλύτερου μέρους του DevOps σε ένα απολύτως προβλέψιμο περιβάλλον το οποίο παραμένει σταθερό από την ανάπτυξη του κώδικα μέχρι την τελική εγκατάσταση. Με αυτή την μέθοδο βελτιώνεται σημαντικά η αποτελεσματικότητα των ομάδων ανάπτυξης κώδικα.

4.2.2 Τεχνολογίες σχετικές με το UI

Στην παρούσα υποενότητα περιγράφονται όσες τεχνολογίες σχετίζονται με τη διεπαφή του χρήστη (UI), δηλαδή με το Presentation tier.

React



Σχήμα 4.6: React logo

Η React[20] αποτελεί βιβλιοθήκη JavaScript, η οποία χρησιμοποιείται για την κατασκευή διεπαφών χρήστη. Είναι δηλωτική (declarative) και βασίζεται σε components, τα οποία διαχειρίζονται την κατάστασή τους (state) και συντίθενται για να δημιουργήσουν πολύπλοκα διαδραστικά UIs.

Ένα σημαντικό εργαλείο για την ταχεία ανάπτυξη web εφαρμογών σε React είναι το Create React App[21]. Με τη χρήση μίας και μόνο εντολής (`npx create-react-app my-app`), εγκαθίσταται αυτόματα ένας development server σε περιβάλλον Node.js (ως μία μοναδική βιβλιοθήκη). Αυτός εμπεριέχει μία πληθώρα από build tools (π.χ. Webpack, Babel, ESLint), τα οποία προσφέρουν ισχυρές δυνατότητες, όπως άμεσα reloads και παραγωγή βελτιστοποιημένων bundles. Έτσι, η διαδικασία της υλοποίησης αποκτά ποικίλες διευκολύνσεις, χωρίς να απαιτεί την εκμάθηση, την χειροκίνητη εγκατάσταση και την προηγμένη διαμόρφωση των τεχνολογιών στο εσωτερικό.

[20] `https://reactjs.org/`
[21] `https://create-react-app.dev/`

Η React έχει το αποθετήριό της στο GitHub [22] και διατίθεται μέσω του μητρώου npm [23].

Redux

Σχήμα 4.7: Redux logo

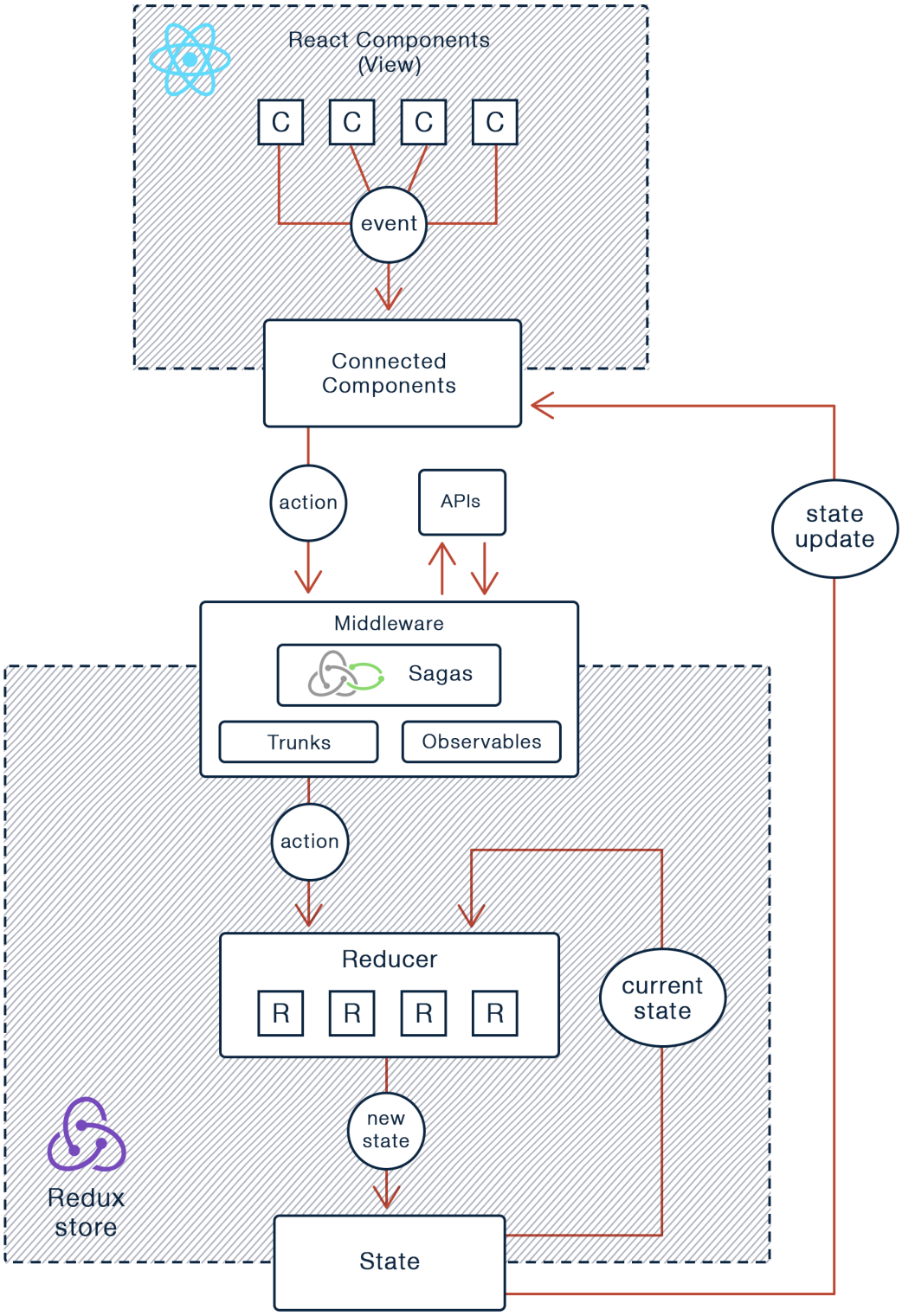
Το Redux[24] αποτελεί μία βιβλιοθήκη JavaScript, η χρήση της οποίας προσφέρει στην εφαρμογή ένα πλήρως διαχειρίσιμο global state.

Τα δομικά στοιχεία του Redux είναι τα εξής:

- **Actions**: Αντικείμενα τα οποία περιέχουν νέα πληροφορία για την τροποποίηση του state της εφαρμογής.
- **Reducers**: Συναρτήσεις οι οποίες λαμβάνοντας ένα action και διαβάζοντας το τρέχον state, εφαρμόζουν κάποια λογική για την παραγωγή ενός νέου state.
- **Store**: Το αντικείμενο στο οποίο βρίσκεται αποθηκευμένο το state της εφαρμογής. Η βασική ιδιότητα του state είναι ότι παραμένει αμετάβλητο και, για την ανανέωσή του, παράγεται πάντα ένα νέο state object μέσω των reducer.
- **Middleware**: Προαιρετικά κομμάτια κώδικα που λαμβάνουν actions πριν εκείνα φτάσουν στους reducers και εκτελούν κάποιο side effect. Συνήθως χρησιμοποιούνται για ενέργειες όπως logging και error reporting ή για να ενώσουν το Redux με εξωτερικά APIs.

Αν και το ίδιο το Redux είναι μικροσκοπικό σε μέγεθος, ο τρόπος υλοποίησής του έχει επιτρέψει τη δημιουργία ενός τεράστιου οικοσυστήματος εργαλείων και επεκτάσεων, τα οποία συνδέονται μαζί του ή βασίζονται σε αυτό. Για παράδειγμα, μία από τις κύριες χρήσεις του είναι η κατασκευή διεπαφών χρήστη σε συνδύασμό με άλλες βιβλιοθήκες, όπως με την React. Σε αυτήν την περίπτωση, συνδέεται μαζί της με το npm πακέτο `react-redux` και η λειτουργία του υπό ανάπτυξη UI προκύπτει ως εξής:

[22] `https://github.com/facebook/react/`
[23] `https://www.npmjs.com/package/react`
[24] `https://redux.js.org/`



Σχήμα 4.8: Λειτουργία του Redux σε συνδυασμό με React

Το Redux έχει το αποθετήριό του στο GitHub[25] και διατίθεται μέσω του μητρώου npm[26].

[25] `https://github.com/reduxjs/redux`
[26] `https://www.npmjs.com/package/redux`

Redux-Saga

Σχήμα 4.9: Redux-Saga logo

Το Redux-Saga[27] αποτελεί μία βιβλιοθήκη JavaScript του οικοσυστήματος του Redux. Πρόκειται για ένα Redux middleware, το οποίο χρησιμοποιεί ESG generator functions[28] για την εκτέλεση και διαχείριση ποικίλων ασύγχρονων side effect. Αυτές οι συναρτήσεις (sagas) παρέχουν μία πληθώρα επιλογών για την παράλληλη εκτέλεση κώδικα που μπορεί να σχετίζεται με εξωτερικά APIs, όπως με ένα blockchain ή μία βάση δεδομένων. Με αυτόν τον τρόπο, τα τελευταία μπορούν να συμπεριληφθούν στο κεντρικό Redux store και τη διαχείριση του συνολικού state της εφαρμογής.

Το Redux-Saga έχει το αποθετήριό του στο GitHub [29] και διατίθεται μέσω του μητρώου npm [30].

4.2.3 Τεχνολογίες σχετικές με το Ethereum

Στην παρούσα υποενότητα περιγράφονται εκείνες οι τεχνολογίες που σχετίζονται με το Ethereum, δηλαδή με το Application tier της τεχνολογικής στοίβας.

Truffle

Σχήμα 4.10: Truffle logo

Το Truffle[31] είναι ένα από τα δημοφιλέστερα Ethereum development frameworks και αποτελεί τμήμα της σουίτας Truffle.

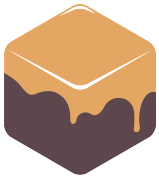
Μέσω του Truffle πραγματοποιείται η διαχείριση των έξυπνων συμβολαίων. Αυτή περιλαμβάνει τη δοκιμή, τη σύνδεση και τη μεταγλώττισή τους, καθώς και την ανάπτυξη τους στο blockchain.

Επίσης, το Truffle περιέχει πρόσθετα σχετικά εργαλεία, όπως διαδραστική κονσόλα για άμεση αλληλεπίδραση με τα contracts και εκτελεστής εξωτερικών σεναρίων (external script runner).

[27] https://redux.js.org/
[28] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function*
[29] https://github.com/redux-saga/redux-saga
[30] https://www.npmjs.com/package/redux-saga
[31] https://trufflesuite.com/truffle/

Έχει το αποθετήριό του στο GitHub [32] και διατίθεται μέσω του μητρώου npm [33].

Ganache

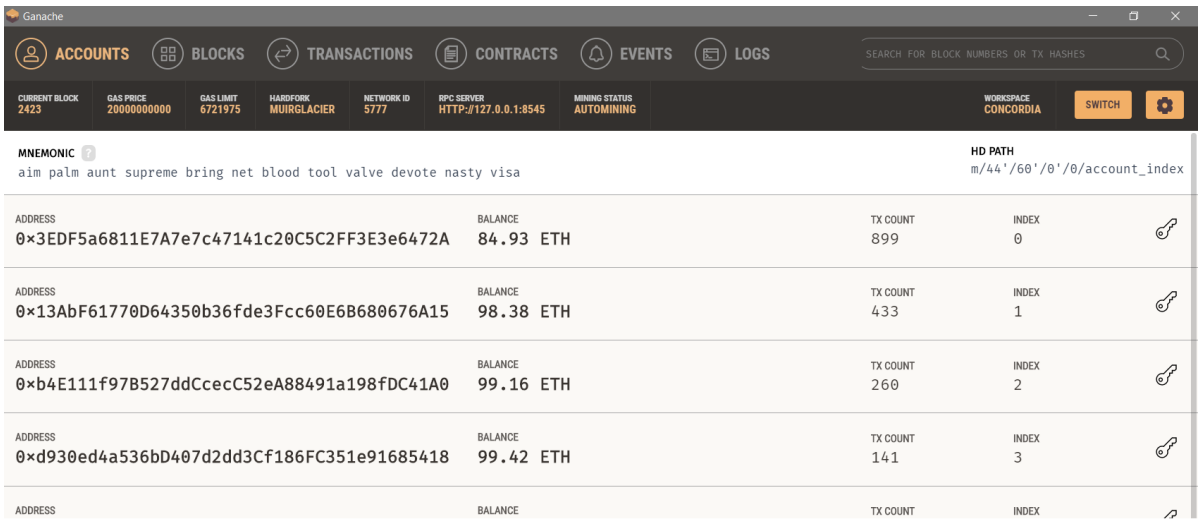


Σχήμα 4.11: Ganache logo

Το Ganache[34] είναι ένα λογισμικό που παρέχει ένα βοηθητικό προσωπικό Ethereum blockchain για ταχεία ανάπτυξη αποκεντρωμένων εφαρμογών και αποτελεί επίσης τμήμα της σουίτας Truffle. Διατίθεται τόσο ως desktop εφαρμογή με UI, όσο και ως CLI (command-line interface).

Το Ganache παρέχει ισχυρά εργαλεία για την ανάπτυξη έξυπνων συμβολαίων, όπως:

- Block explorer, μέσω του οποίου μπορούν να εξεταστούν λεπτομερώς όλα τα blocks και οι συναλλαγές που έλαβαν χώρα.
- Εξερεύνηση των εσωτερικών των contracts και των πυροδοτημένων event τους.
- Ενδελεχές αρχείο καταγραφής της εξόδου του blockchain, το οποίο περιλαμβάνει σημαντικές πληροφορίες για τον εντοπισμό σφαλμάτων.
- Δυνατότητα διαμόρφωσης του χρόνου εξόρυξης των block, έτσι ώστε να αρμόζει με τις εκάστοτε ανάγκες (αυτόματη εξόρυξη ή εξόρυξη σε προσαρμοσμένο χρονικό διάστημα).



Σχήμα 4.12: Ganache (desktop εφαρμογή)

[32] https://github.com/trufflesuite/truffle
[33] https://www.npmjs.com/package/truffle
[34] https://trufflesuite.com/ganache/

Το Ganache έχει το αποθετήριό του στο GitHub [35] και διατίθεται μέσω του μητρώου npm [36].

4.2.4 Τεχνολογίες σχετικές με το IPFS

Σε αυτήν την υποενότητα περιγράφονται όσες τεχνολογίες σχετίζονται με το IPFS (βλ. ενότητα 2.7), δηλαδή με το Data tier της τεχνολογικής στοίβας της εφαρμογής.

js-ipfs

Σχήμα 4.13: js-ipfs logo

Η υλοποίηση του IPFS που χρησιμοποείται στην εφαρμογή Concordia είναι αυτή σε JavaScript και ονομάζεται js-ipfs. Μέσω αυτής της βιβλιοθήκης, παρέχεται η δυνατότητα δημιουργίας ενός IPFS κόμβου, τόσο σε έναν Node.js server, όσο και σε ένα περιβάλλον browser.

Το js-ipfs έχει το αποθετήριό του στο GitHub [37] και διατίθεται μέσω του μητρώου npm [38].

OrbitDB

Σχήμα 4.14: OrbitDB logo

Η OrbitDB είναι μία P2P βάση δεδομένων ανοιχτού κώδικα. Χρησιμοποιεί το IPFS για την αποθήκευση των δεδομένων και το IPFS Pubsub για τον αυτόματο συγχρονισμό των βάσεων δεδομένων μεταξύ των peers. Είναι τελικά συνεπής (eventually consistent) και χρησιμοποιεί CRDTs (Conflict-Free Replicated Data Types) για συγχωνεύσεις βάσεων δεδομένων χωρίς συγκρούσεις, πράγμα που την καθιστά εξαιρετική επιλογή για DApps και offline-first web applications.[18]

Κάποια βασικά χαρακτηριστικά της είναι τα εξής:

- **Stores**: Η OrbitDB παρέχει διάφορους τύπους βάσεων (stores) για διαφορετικά μοντέλα δεδομένων και περιπτώσεις χρήσης:

[35] `https://github.com/trufflesuite/ganache`
[36] `https://www.npmjs.com/package/ganache`
[37] `https://github.com/ipfs/js-ipfs`
[38] `https://www.npmjs.com/package/ipfs`

- log: ένα αμετάβλητο (μόνο για προσάρτηση) ημερολόγιο με ανιχνεύσιμο ιστορικό.
- feed: ένα μεταβλητό αρχείο καταγραφής με ανιχνεύσιμο ιστορικό, του οποίου οι καταχωρήσεις μπορούν να προστεθούν και να αφαιρεθούν.
- keyvalue: μία βάση δεδομένων κλειδιών-τιμών.
- docs: μία βάση δεδομένων εγγράφων στην οποία μπορούν να αρχειοθετηθούν έγγραφα JSON με ένα καθορισμένο κλειδί.
- counter: μία βάση δεδομένων για καταμέτρηση συμβάντων.

Όλα τα stores υλοποιούνται πάνω στο `ipfs-log`, μία αμετάβλητη, operation-based CRDT για κατανεμημένα συστήματα, ενώ υπάρχει και η δυνατότητα δημιουργίας προσαρμοσμένων stores ανάλογα με την περίπτωση.

- **Address**: Κάθε βάση δεδομένων λαμβάνει κατά τη δημιουργία της μία διεύθυνση της μορφής `/orbitdb/CID/DATABASE_NAME`, όπου `CID` είναι το IPFS multihash του μανιφέστου της και `DATABASE_NAME` το όνομα της βάσης.[19]Το μανιφέστο είναι ένα IPFS object που περιέχει πληροφορίες της βάσης όπως το όνομα, τον τύπο και έναν δείκτη στον ελεγκτή πρόσβασης (access controller).
- **Identity**: Κάθε φορά που προστίθεται μία εγγραφή στη βάση υπογράφεται από τον δημιουργό της, ο οποίος προσδιορίζεται από μία ταυτότητα (identity). Το Identity object, πέρα από τον προεπιλεγμένο τρόπο λειτουργίας, μπορεί να προσαρμοστεί έτσι ώστε να συνδέεται με κάποιο εξωτερικό αναγνωριστικό. Η μορφή του έχει ως εξής[39]:

```
{
   id: '<the ID of the external identity>',
   // Auto-generated by OrbitDB
   publicKey: '<signing key used to sign OrbitDB entries>',
   signatures: {
     //Allows the owner of id to prove they own the private
     key associated with publicKey
     id: '<signature of id signed using publicKey>',
     //This links the two ids
     publicKey: '<signature of signatures.id + publicKey using
     id>'
   },
   type: 'orbitdb'
}
```

Σχήμα 4.15: OrbitDB Identity

[39]Βλ. και `https://github.com/orbitdb/orbit-db-identity-provider`

- **Access Control**: Κατά τη δημιουργία μίας βάσης μπορούν να οριστούν όσοι θα έχουν δικαίωμα εγγραφής σε αυτή, μέσω ενός ελεγκτή πρόσβασης (access controller). Ο ελεγκτής θα περιλαμβάνει τα public keys τους, τα οποία μπορούν να ανακτηθούν από το identity του καθενός. Από προεπιλογή και αν δεν ορίζεται διαφορετικά, δίνεται πρόσβαση εγγραφής μόνο στον δημιουργό της βάσης.

Η OrbitDB έχει το αποθετήριό της στο GitHub [40] και διατίθεται μέσω του μητρώου npm [41].

Libp2p

Σχήμα 4.16: Libp2p logo

Η libp2p είναι ένα αρθρωτό σύστημα πρωτοκόλλων, προδιαγραφών και βιβλιοθηκών που επιτρέπουν την ανάπτυξη p2p εφαρμογών. Αποτελεί το υποκείμενο επίπεδο δικτύου του IPFS.[15]

Ένα από τα υλοποιημένα πρωτόκολλα μεταφοράς δεδομένων της libp2p είναι το libp2p-webrtc-star[42]. Αποτελεί πρωτόκολλο μεταφοράς δεδομένων το οποίο υποστηρίζεται τόσο από Node.js servers, όσο και από browsers. Περιλαμβάνει, επίσης, έναν signalling server, που επιτρέπει τη γρήγορη ανακάλυψη και διασύνδεση των peers.

Το libp2p-webrtc-star έχει το αποθετήριό του στο GitHub [43] και διατίθεται μέσω του μητρώου npm [44].

4.3 Αρχιτεκτονική υλοποίησης

Το περιβάλλον ανάπτυξης της εφαρμογής υλοποιήθηκε χρησιμοποιώντας το μοντέλο αρχιτεκτονικής των μικροϋπηρεσιών. Το μοντέλο των μικροϋπηρεσιών βασίζεται στην αποδόμηση του συστήματος σε μικρές μονάδες, οι οποίες συνεργάζονται ώστε να προσφέρουν ένα ενιαίο αποτέλεσμα. Η προσέγγιση αυτή έχει πολλά πλεονεκτήματα σε σύγκριση με την ανάπτυξη μονολιθικών εφαρμογών. Ο βασικός λόγος για τον οποίο επιλέχθηκε η αρχιτεκτονική μικροϋπηρεσιών είναι η ευκολία που προσφέρει στη γρήγορη ανάπτυξη καινούριων χαρακτηριστικών, ταυτόχρονα από διαφορετικά μέλη μίας ομάδας, ασύγχρονα και χωρίς την ανάγκη συνεχούς επικοινωνίας και συνεννόησης μεταξύ τους. Αυτό συμβαίνει επειδή κάθε μέρος του συστήματος

[40] `https://github.com/orbitdb/orbit-db`
[41] `https://www.npmjs.com/package/orbit-db`
[42] `https://github.com/libp2p/js-libp2p-webrtc-star`
[43] `https://github.com/libp2p/js-libp2p-webrtc-star`
[44] `https://www.npmjs.com/package/libp2p-webrtc-star`

(υπηρεσία) είναι αυτόνομο και η ανάπτυξή του είναι διαχωρισμένη από το υπόλοιπο σύστημα με το οποίο είναι αδύναμα συνδεδεμένο (loosely coupled).

Το σύστημα του περιβάλλοντος ανάπτυξης συντίθεται από διάφορες μικροϋπηρεσίες, κάποιες από τις οποίες αναπτύχθηκαν στα πλαίσια αυτής της εργασίας, ενώ άλλες αποτελούν δωρεάν λογισμικό ανοιχτού κώδικα. Οι μικροϋπηρεσίες αυτές συνοψίζονται στον παρακάτω πίνακα:

Μικροϋπηρεσία	**Σύντομη περιγραφή - Αντικείμενο/Στόχος**
Concordia Application	Υπηρεσία με την οποία αλληλεπιδρούν οι χρήστες
Concordia Contracts Migrator	Υπηρεσία μεταφόρτωσης των συμβολαίων (contracts) στο blockchain
Concordia Pinner	Υπηρεσία καρφιτσώματος δεδομένων.
Concordia Contracts Provider	Υπηρεσία διαμοιρασμού των contract artifacts μέσω HTTP
Ganache	Τοπικό, ιδιωτικό Ethereum blockchain
Rendezvous Server	Υπηρεσία εύρεσης ομότιμων χρηστών

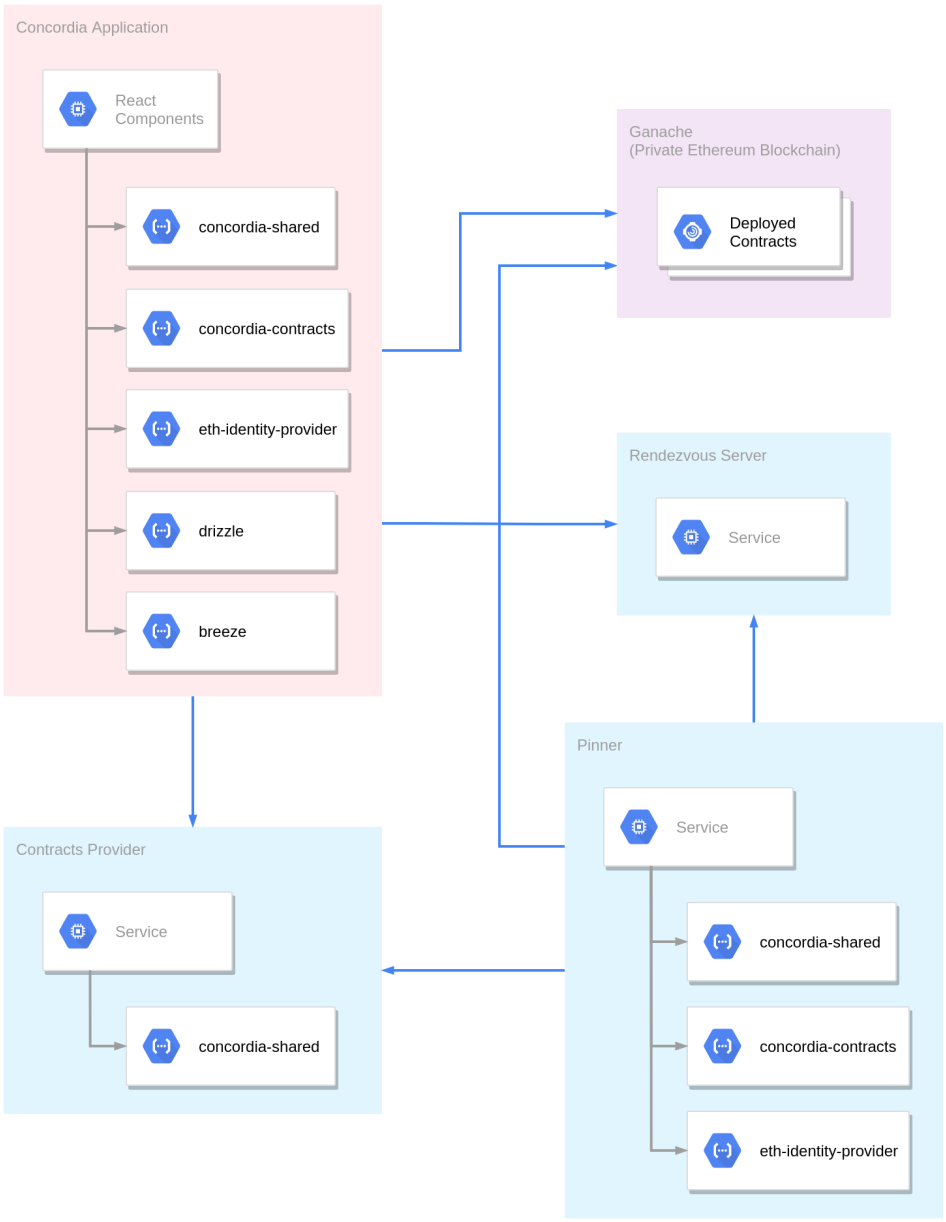
Πίνακας 4.1: Σύντομη περιγραφή των υπηρεσιών του περιβάλλοντος ανάπτυξης

Στα πλαίσια της εργασίας αναπτύχθηκαν επίσης διάφορα αρθρώματα (modules), κυρίως με τη μορφή βιβλιοθηκών JavaScript. Τα αρθρώματα χρησιμοποιούνται από τις υπηρεσίες για την επίτευξη των επιμέρους εργασιών. Η ανάπτυξη του λογισμικού σε ξεχωριστά αρθρώματα επιτρέπει την εύκολη επαναχρησιμοποίηση του κώδικα, καθώς και τον διαχωρισμό των αυτόνομων τμημάτων κώδικα. Τα αρθρώματα συνοψίζονται στον παρακάτω πίνακα:

Άρθρωμα	**Σύντομη περιγραφή - Αντικείμενο/Στόχος**
Άρθρωμα concordia-shared	Χρήσιμα εργαλεία και σταθερές συστήματος.
Άρθρωμα concordia-contracts	Μεταγλώττιση των contracts και διάθεση των artifacts.
Άρθρωμα eth-identity-provider	Δημιουργία μοναδικού αναγνωριστικού χρήστη για τη βάση OrbitDB.
Άρθρωμα drizzle	Βελτιωμένη προγραμματιστική διεπαφή επικοινωνίας με το blockchain.
Άρθρωμα breeze	Βελτιωμένη προγραμματιστική διεπαφή χρήσης της βάσης OrbitDB.

Πίνακας 4.2: Σύντομη περιγραφή αρθρωμάτων συστήματος.

Τα αρθρώματα και οι υπηρεσίες θα περιγραφούν σε μεγαλύτερη ανάλυση στα επόμενα κεφάλαια. Στο παρακάτω σχήμα φαίνεται η συνολική αρχιτεκτονική του συστήματος:



Σχήμα 4.17: Διάγραμμα αρχιτεκτονικής συστήματος

4.3.1 Αρθρώματα

Σε αυτό το κεφάλαιο θα περιγραφούν με μεγαλύτερη λεπτομέρεια τα αρθρώματα που αναπτύχθηκαν.

Άρθρωμα concordia-shared

Το άρθρωμα concordia-shared αποτελεί μία βιβλιοθήκη χρήσιμων εργαλείων και σταθερών. Εδώ περιέχεται όλο το λογισμικό το οποίο πρέπει ή είναι επιθυμητό να συμπεριφέρεται με τον ίδιο τρόπο συνολικά στο σύστημα, όπως για παράδειγμα μέθοδοι παραμετροποίησης των υπηρεσιών και μέθοδοι καταγραφής (logging). Το άρθρωμα αυτό χρησιμοποιείται από το άρθρωμα concordia-contracts καθώς και από τις υπηρεσίες Concordia Application, Concordia Pinner και

Concordia Contracts Provider.

Το άρθρωμα αυτό γίνεται διαθέσιμο για χρήση με τη μορφή τοπικής βιβλιοθήκης με τη χρήση της δυνατότητας διαχείρισης μοναδικού αποθετηρίου κώδικα (monorepo) yarn workspaces[45].

Άρθρωμα concordia-contracts

Το άρθρωμα αυτό επιτελεί δύο ενέργειες. Αρχικά, είναι το άρθρωμα στο οποίο αναπτύσσονται τα contracts που χρησιμοποιούνται από την εφαρμογή. Το άρθρωμα αυτό αναλαμβάνει τη μεταγλώττιση των contracts από κώδικα γλώσσας Solidity, στην κατάλληλη τελική μορφή JSON. Παρέχονται επίσης σενάρια ενεργειών (scripts) ώστε τα contracts να μεταφορτωθούν σε blockchain καθώς και στην υπηρεσία Concordia Contracts Provider. Αποτελεί επίσης βιβλιοθήκη η οποία μετά τη μεταγλώττιση και μεταφόρτωση των contracts σε blockchain παρέχει τα contract artifacts. Χρησιμοποιείται από τις υπηρεσίες Concordia Application και Concordia Pinner.

Το άρθρωμα αυτό γίνεται διαθέσιμο για χρήση με τη μορφή τοπικής βιβλιοθήκης με τη χρήση της δυνατότητας yarn workspaces.

Άρθρωμα eth-identity-provider

Η λειτουργία της βάσης OrbitDB επιτρέπει τη χρήση προσαρμοσμένων orbit-db-identity-provider, οι οποίοι θα δημιουργούν και θα επικυρώνουν τα μοναδικά αναγνωριστικά των χρήστών (OrbitDB Identity) βάσει προσαρμοσμένων εξωτερικών αναγνωριστικών (external identifier), όπως παρουσιάζεται στο σχήμα 4.15.

Στην περίπτωση της εφαρμογής Concordia είναι χρήσιμο να μπορούν να υπολογιστούν με ντετερμινιστικό τρόπο οι OrbitDB βάσεις δεδομένων του κάθε χρήστη, για λόγους απλότητας και εξοικονόμησης αποθηκευτικού χώρου επί του blockchain. Έτσι, αφού κάθε χρήστης ορίζεται μοναδικά μέσω της διεύθυνσης Ethereum με την οποία εγγράφεται και συνδέεται, αυτή θα πρέπει να αποτελεί και το εξωτερικό αναγνωριστικό στο πεδίο id της OrbitDB Identity.

Για αυτόν το λόγο υλοποιήθηκε το άρθρωμα eth-identity-provider, το οποίο:

- Παράγει ένα OrbitDB Identity για τον χρήστη, με id τον συνδυασμο του Ethereum address του και του address του κεντρικού contract της εφαρμογής[46]. Αυτό επιτυγχάνεται με την υπογραφή μίας συναλλαγής με το Ethereum private key του χρήστη, μέσω του MetaMask.
- Επικυρώνει τις OrbitDB Identity που απαιτούνται, εξασφαλίζοντας ότι υπογράφηκαν από τα Ethereum private key των κατόχων τους.
- Διασφαλίζει ντετερμινιστικές, υπολογίσιμες διευθύνσεις OrbitDB βάσεων για τον κάθε χρήστη.

[45] `https://yarnpkg.com/features/workspaces`

[46] Το δεύτερο εισήχθη για την αποφυγή προβλημάτων σε πολλαπλές αναπτύξεις συμβολαίων.

Το eth-identity-provider γίνεται διαθέσιμο για χρήση με τη μορφή βιβλιοθήκης μέσω του μητρώου λογισμικού npm [47], ενώ το αποθετήριό του βρίσκεται στο GitLab [48].

Άρθρωμα drizzle

Το άρθρωμα drizzle που χρησιμοποιείται στην υπηρεσία Concordia Application είναι μία τροποποιημένη έκδοση της JavaScript βιβλιοθήκης Drizzle (και συγκεκριμένα του @drizzle/store[49]), η οποία προσφέρεται από τη σουίτα εργαλείων Truffle. Η τροποποιημένη βιβλιοθήκη αναπτύχθηκε στα πλαίσια της διπλωματικής με στόχο τη διευκόλυνση της χρήσης του Drizle και την επιδιόρθωση προβληματικών σημείων της πρωτότυπης βιβλιοθήκης.

Το άρθρωμα drizzle υλοποιεί τις προγραμματιστικές διεπαφές μέσω των οποίων πραγματοποιείται η επικοινωνία της εφαρμογής με το blockchain. Για την επίτευξη της επικοινωνίας αυτής, η βιβλιοθήκη χρησιμοποιεί τη συλλογή βιβλιοθηκών web3.js η οποία αποτελεί τον πιο διαδεδομένο τρόπο διεπαφής με το blockchain σε αποκεντρωτικές εφαρμογές. Τελικά, παρέχει ένα Redux store, το οποίο συμπεριλαμβάνεται στο κεντρικό store της εφαρμογής.

Το άρθρωμα αυτό γίνεται διαθέσιμο για χρήση με τη μορφή βιβλιοθήκης μέσω του μητρώου λογισμικού npm [50], ενώ το αποθετήριό του βρίσκεται στο GitLab [51].

Άρθρωμα breeze

Το άρθρωμα αυτό αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας και αποτελεί μία βιβλιοθήκη περίβλημα (wrapper) της βιβλιοθήκης OrbitDB, η οποία παρέχει ένα Redux store.

Με τη συμπερίληψη του store του αρθρώματος στο κεντρικό Redux store της εφαρμογής, παρέχεται η δυνατότητα εκτέλεσης των λειτουργιών των OrbitDB βάσεων εντός του γενικότερου flow του frontend της εφαρμογής. Έτσι, οι προγραμματιστικές διεπαφές που προσφέρει η Orbit χρησιμοποιούνται πλέον μέσα από actions, reducers και middleware.

Το άρθρωμα αυτό γίνεται διαθέσιμο για χρήση με τη μορφή βιβλιοθήκης μέσω του μητρώου λογισμικού npm [52], ενώ το αποθετήριό του βρίσκεται στο GitLab [53].

[47] `https://www.npmjs.com/package/@ecentrics/eth-identity-provider`
[48] `https://gitlab.com/ecentrics/eth-identity-provider`
[49] `https://github.com/trufflesuite/drizzle/tree/develop/packages/store`
[50] `https://www.npmjs.com/package/@ecentrics/drizzle`
[51] `https://gitlab.com/ecentrics/drizzle`
[52] `https://www.npmjs.com/package/@ecentrics/breeze`
[53] `https://gitlab.com/ecentrics/breeze`

4.3.2 Concordia Application

Περιγραφή - Στόχοι υπηρεσίας

Σχήμα 4.18: Concordia logo

Η εφαρμογή Concordia (Concordia Application) εκθέτει τις γραφικές διεπαφές μέσω των οποίων αλληλεπιδρούν οι χρήστες με το σύστημα. Αποτελεί τον δίαυλο επικοινωνίας του τελικού χρήστη με το blockchain και με τη βάση OrbitDB. Η αρχιτεκτονική της υπηρεσίας φαίνεται στο σχήμα 4.19. Μέσω της εφαρμογής Concordia οι χρήστες μπορούν:

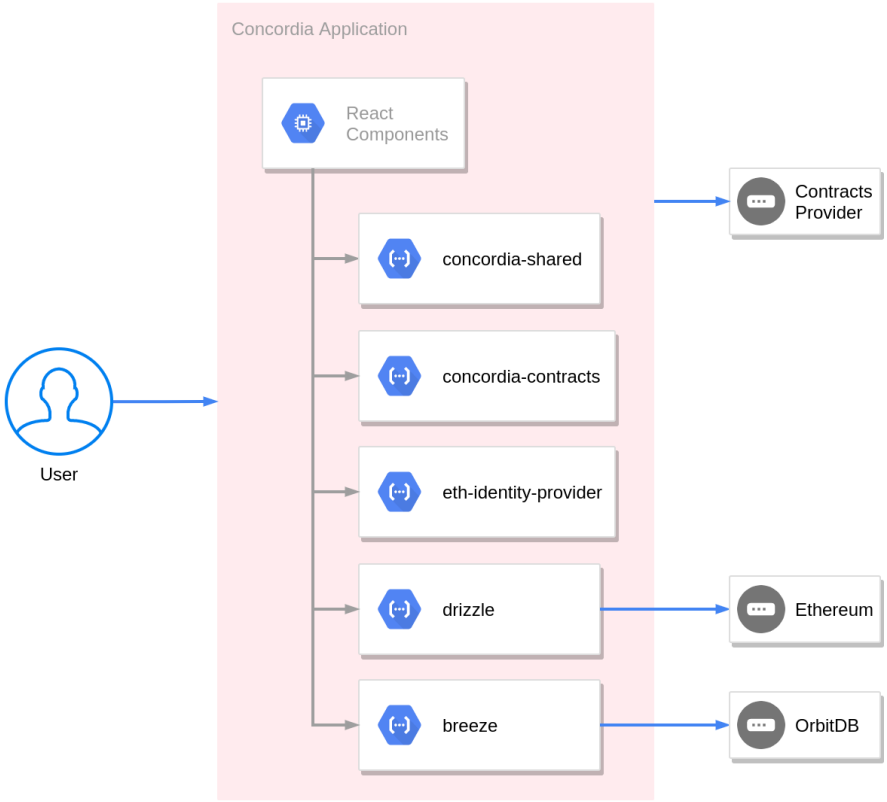
- Να περιηγούνται και να διαβάζουν το περιεχόμενο της πλατφόρμας
- Να δημιουργήσουν λογαριασμό χρήστη
- Να δημοσιεύουν και να τροποποιούν προσωπικές τους πληροφορίες, όπως η τοποθεσία και η εικόνα προφίλ
- Να δημιουργούν θέματα (topics)
- Να δημιουργούν και να τροποποιούν μηνύματα (posts)
- Να δημιουργούν ψηφοφορίες (polls), καθώς και να ψηφίζουν σε αυτές (εφόσον έχουν το δικαίωμα)
- Να υπερψηφίζουν (up-vote) ή να καταψηφίζουν (down-vote) μηνύματα άλλων χρηστών

Η υπηρεσία αποτελείται από κώδικα γραμμένο σε JavaScript, ο οποίος γίνεται διαθέσιμος στους τελικούς χρήστες με τη μορφή εφαρμογής διαδικτύου (web application) μέσω ενός διακομιστή (server). Παρόλο που η υπηρεσία προσφέρει τη γραφική διεπαφή χρήστη μόνο στην αγγλική γλώσσα, έχει παραμετροποιηθεί ώστε να είναι δυνατή η εύκολη μεταγλώττιση της χωρίς την ανάγκη πραγματοποίησης μεγάλων αλλαγών στον κώδικα.

Χρησιμοποιείται η βιβλιοθήκη React για την οργάνωση και ανάπτυξη των συνθετικών τμημάτων (components) του γραφικού περιβάλλοντος. Για το γραφικό περιβάλλον γίνεται χρήση του framework της Semantic UI[54]. Χρησιμοποιείται η βιβλιοθήκη Redux για τη διαχείριση κατάστασης της εφαρμογής (state management), καθώς και η βιβλιοθήκη Redux-Saga για τη διαχείριση ασύγχρονων παράπλευρων ενεργειών (side-effects) σε ένα σύστημα βασισμένο σε συμβάντα (event-based). Άλλες βιβλιοθήκες χρησιμοποιούνται για διάφορα μέρη της υπηρεσίας, ενώ

[54] `https://semantic-ui.com/`

χρησιμοποιούνται επίσης τα αρθρώματα που περιγράφηκαν προηγουμένως για την επίτευξη διαφορετικών στόχων. Ο πλήρης κατάλογος των βιβλιοθηκών και αρθρωμάτων μπορεί να βρεθεί στον κώδικα της υπηρεσίας στο παράρτημα.



Σχήμα 4.19: Αρχιτεκτονική υπηρεσίας Concordia Application

Για τη λειτουργία της υπηρεσία Concordia Application είναι απαραίτητα τα αντικείμενα (artifacts) που προκύπτουν από τη μεταγλώττιση των contracts και τη μεταφόρτωση/δημοσίευσή τους στο blockchain. Για την εισαγωγή των artifacts στην υπηρεσία έχουν αναπτυχθεί δύο μέθοδοι.

Η πρώτη μέθοδος είναι η μεταγλώττιση και μεταφόρτωση των contracts πριν την παραγωγή του πακέτου λογισμικού της υπηρεσίας για τελική χρήση (production build). Με αυτόν τον τρόπο η υπηρεσία θα έχει πρόσβαση στα artifacts μέσω της βιβλιοθήκης που παράγεται από το άρθρωμα concordia-contracts. Αυτή η μέθοδος έχει το μειονέκτημα ότι το τελικό πακέτο λογισμικού (production build) «δένεται» με όποια συγκεκριμένη έκδοση των contracts είναι διαθέσιμη κατά τη δημιουργία του πακέτου. Αυτό σημαίνει ότι σε ενδεχόμενη ενημέρωση των contracts πρέπει αναγκαστικά να δημιουργηθεί και νέα έκδοση του πακέτου λογισμικού της υπηρεσίας Concordia Application.

Για την αποφυγή του παραπάνω προβλήματος αναπτύχθηκε η δεύτερη μέθοδος προσκόμισης των contract artifacts, η οποία είναι η λήψη τους (download) από μία άλλη τοποθεσία στο διαδίκτυο. Σε αυτή τη μέθοδο, η εφαρμογή κατά την εκκίνησή της πραγματοποιεί ένα HTTP αίτημα (HTTP request) σε διεύθυνση η οποία δίνεται ως μεταβλητή περιβάλλοντος (environment

variable). Η απάντηση του αιτήματος αναμένεται να περιέχει τα artifacts ώστε η εφαρμογή να τα χρησιμοποιήσει.

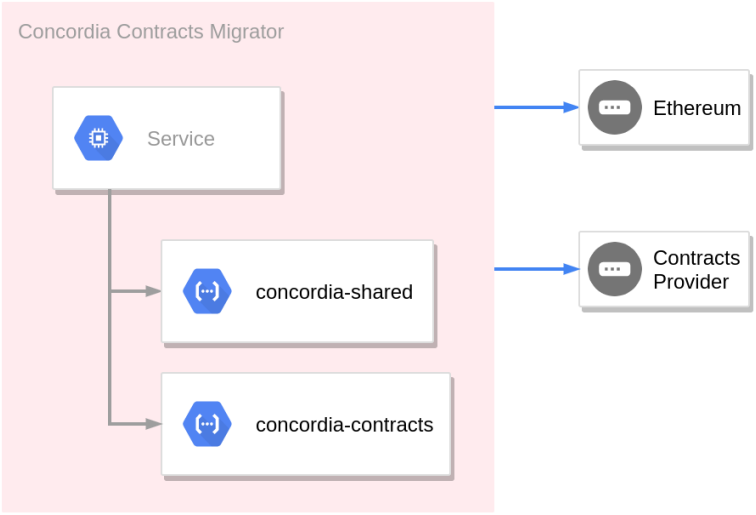
Διανομή

Η υπηρεσία Concordia Application πακετάρεται μαζί με τον διακομιστή nginx και γίνεται διαθέσιμη για χρήση ως εικόνα docker (docker image) μέσω του αποθετηρίου εικόνων dockerhub. Κατά την εκτέλεση της εικόνας οι χρήστες μπορούν μέσω μεταβλητών περιβάλλοντος να ορίσουν παραμέτρους της εκτέλεσης όπως η διεύθυνση του εξυπηρετητή (host location) της εφαρμογής και οι τοποθεσίες των υπηρεσιών Rendezvous Server και Contracts Provider.

4.3.3 Concordia Contracts Migrator

Περιγραφή - Στόχοι υπηρεσίας

Η υπηρεσία αυτή αποτελείται από ένα εκτελέσιμο πρόγραμμα γραμμής εντολών βασισμένο στο άρθρωμα concordia-contracts που αναλύθηκε σε προηγούμενη υποενότητα (4.3.1). Το πρόγραμμα, κατά την εκτέλεσή του, μεταγλωττίζει τα contracts και έπειτα τα μεταφορτώνει στο blockchain το οποίο είναι ορισμένο με χρήση μεταβλητών περιβάλλοντος. Τέλος, αν οι κατάλληλες μεταβλητές περιβάλλοντος είναι ορισμένες, το πρόγραμμα μεταφορτώνει τα τελικά artifacts σε αποθετήριο Concordia Contracts Provider. Η αρχιτεκτονική της υπηρεσίας φαίνεται στο παρακάτω σχήμα (σχήμα 4.20).



Σχήμα 4.20: Αρχιτεκτονική υπηρεσίας Concordia Contracts Migrator

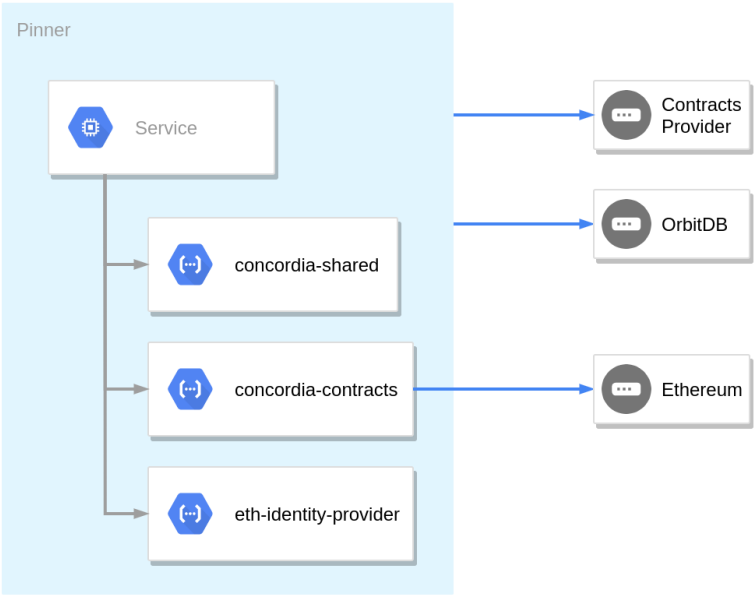
Διανομή

Η υπηρεσία αυτή γίνεται διαθέσιμη για χρήση ως docker image μέσω του αποθετηρίου εικόνων dockerhub. Οι χρήστες μπορούν χρησιμοποιώντας μεταβλητές περιβάλλοντος να αλλάξουν τη διεύθυνση του blockchain και την τοποθεσία της υπηρεσίας Contracts Provider στην οποία το πρόγραμμα θα μεταφορτώσει τα contracts και τα artifacts.

4.3.4 Concordia Pinner

Περιγραφή - Στόχοι υπηρεσίας

Η υπηρεσία καρφιτσώματος περιεχομένου (Concordia Pinner) αποτελεί μία εφαρμογή τερματικού (temrinal application/cmd application) η οποία στοχεύει στο καρφίτσωμα (pinning) του περιεχομένου που αποθηκεύεται στο IPFS μέσω της βάσης OrbitDB. Η υπηρεσία είναι γραμμένη στη γλώσσα προγραμματισμού JavaScript. Η αρχιτεκτονική της υπηρεσίας φαίνεται το σχήμα 4.21.



Σχήμα 4.21: Αρχιτεκτονική υπηρεσίας Concordia Pinner

Η υπηρεσία αυτή υλοποιήθηκε για να εγγυηθεί η διαθεσιμότητα του περιεχομένου του συστήματος που αποθηκεύεται στο IPFS (τίτλοι θεμάτων, περιεχόμενο μηνυμάτων και άλλα). Λόγω του τρόπου λειτουργίας του IPFS, το περιεχόμενο που αναρτούν οι χρήστες πρέπει να καρφιτσώνεται από άλλους χρήστες ή αυτόνομες εφαρμογές, όπως η υπηρεσία Concordia Pinner, ώστε να είναι διαθέσιμο. Αν το περιεχόμενο δεν καρφιτσωθεί, τότε θα είναι διαθέσιμο στους υπόλοιπους χρήστες μόνο από τον δημιουργό του, έτσι αν αυτός δεν είναι ενεργός στο δίκτυο, το περιεχόμενο θα είναι αδύνατο να βρεθεί.

Η υπηρεσία συνδέεται στο blockchain από όπου παρακολουθεί την κατάσταση του συστήματος και «ακούει» για νέους χρήστες, θέματα, μηνύματα και ψηφοφορίες. Η υπηρεσία συνδέεται επίσης στο IPFS, έτσι όταν δημιουργηθεί νέο περιεχόμενο στο σύστημα το καρφιτσώνει αυτόματα. Με αυτό τον τρόπο, διατηρώντας την υπηρεσία πάντα διαθέσιμη, για παράδειγμα εκτελώντας τη σε περιβάλλον διακομιστή (server), διαβεβαιώνεται η διαθεσιμότητα του περιεχομένου.

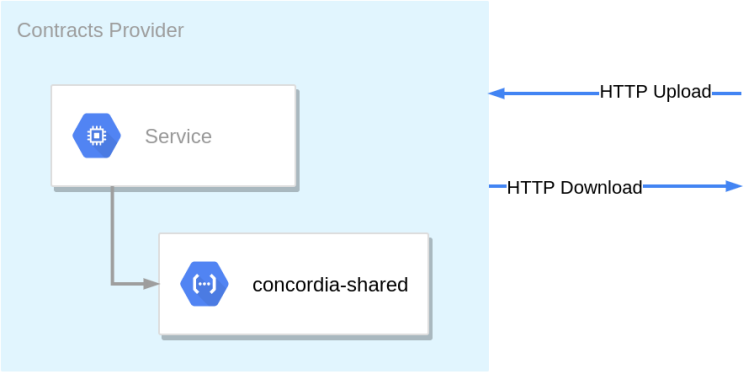
Διανομή

Η υπηρεσία αυτή γίνεται διαθέσιμη για χρήση ως docker image μέσω του αποθετηρίου εικόνων dockerhub. Κατά την εκτέλεση της εικόνας οι χρήστες μπορούν μέσω μεταβλητών περιβάλλοντος να ορίσουν παραμέτρους της υπηρεσίας όπως τη διεύθυνση του εξυπηρετητή (host location), τη διεύθυνση του blockchain, τις διαδρομές αποθήκευσης των δεδομένων στο σύστημα και τις τοποθεσίες των υπηρεσιών Rendezvous Server και Contracts Provider.

4.3.5 Concordia Contracts Provider

Περιγραφή - Στόχοι υπηρεσίας

Η υπηρεσία Contracts Provider αποτελεί μία βοηθητική υπηρεσία η οποία υλοποιεί ένα απλό αποθετήριο για τα contract artifacts. Είναι γραμμένη σε JavaScript και διαθέτει δύο HTTP endpoints, ένα για τη μεταφόρτωση (upload) των artifacts προς την υπηρεσία και ένα για τη λήψη (download) από την υπηρεσία. Η υπηρεσία υποστηρίζει επίσης την επισύναψη ετικετών στα artifacts, όπως η έκδοση (version) ή το κλαδί ανάπτυξης (branch, για παράδειγμα master/develop). Η αρχιτεκτονική της υπηρεσίας φαίνεται το σχήμα 4.22.



Σχήμα 4.22: Αρχιτεκτονική υπηρεσίας Concordia Contracts Provider

Η υπηρεσία χρησιμοποιείται σε μία προσπάθεια αποσύνδεσης της βασικής εφαρμογής που υλοποιεί η υπηρεσία Concordia Application από μία συγκεκριμένη έκδοση των contracts. Οι λόγοι που αυτό είναι επιθυμητό αναπτύχθηκαν στην περιγραφή της υπηρεσίας Concordia Application (υποενότητα 4.3.2). Ωστόσο, η υπηρεσία Contracts Provider αποτελεί σημείο κεντροποίησης του συστήματος, για το λόγο αυτό θεωρείται προσωρινή λύση η οποία θα μπορούσε να αντικατασταθεί από αποκεντρωτικές λύσεις όπως η μεταφόρτωση των artifacts στο IPFS και ο διαμοιρασμός τους από εκεί.

Διανομή

Η υπηρεσία αυτή γίνεται διαθέσιμη για χρήση ως docker image μέσω του αποθετηρίου εικόνων dockerhub. Οι χρήστες μπορούν χρησιμοποιώντας μεταβλητές περιβάλλοντος να αλλάξουν

παραμέτρους της εκτέλεσης, όπως τη διαδρομή αποθήκευσης των μεταφορτωμένων contract artifacts.

4.3.6 Ganache

Περιγραφή - Στόχοι υπηρεσίας

Η συγκεκριμένη υπηρεσία αξιοποιεί την CLI έκδοση του Ganache, δημιουργώντας ένα τοπικό ιδιωτικό Ethereum blockchain για τους σκοπούς ανάπτυξης της εφαρμογής Concordia.

Διανομή

Για τη χρήση της υπηρεσίας αυτής αναπτύχθηκε μία νέα εικόνα docker που βασίζεται στην επίσημη εικόνα που διατίθεται από τη σουίτα και προσθέτει μερικές χρήσιμες λειτουργικότητες όπως η δυνατότητα αποκάλυψης των κλειδιών που δημιουργούνται κατά την εκτέλεση. Η υπηρεσία γίνεται διαθέσιμη για χρήση ως docker image μέσω του αποθετηρίου εικόνων dockerhub. Η εικόνα παρέχει τη δυνατότητα τροποποίησης των παραμέτρων εκτέλεσης με χρήση μεταβλητών περιβάλλοντος. Με αυτό τον τρόπο οι χρήστες μπορούν να αλλάξουν τον αριθμό των λογαριασμών που θα δημιουργηθούν, το ποσό του Ether που θα λάβει κάθε λογαριασμός καθώς και άλλες μεταβλητές.

4.3.7 Rendezvous Server

Περιγραφή - Στόχοι υπηρεσίας

Η υπηρεσία Rendezvous Server παρέχει έναν signalling server, ο οποίος υιοθετεί το Libp2p πρωτόκολλο μεταφοράς δεδομένων libp2p-webrtc-star. Μέσω αυτού παρέχεται στους ομότιμους χρήστες (peers) η δυνατότητα ανακάλυψης των διευθύνσεων των υπόλοιπων χρηστών στο δίκτυο του IPFS.

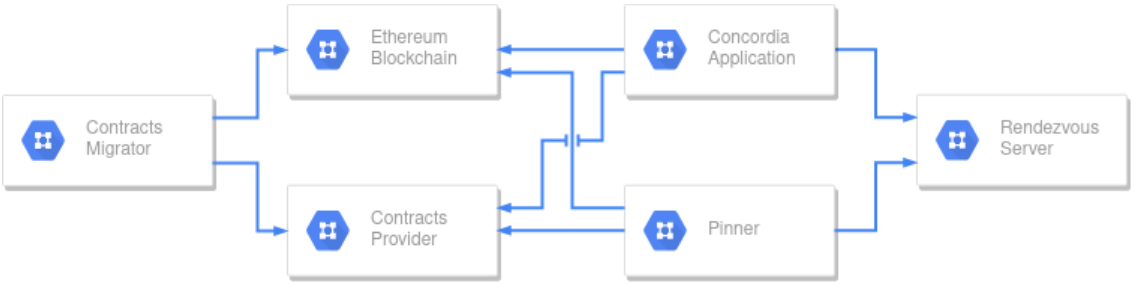
Διανομή

Η υπηρεσία αυτή είναι διαθέσιμη για χρήση από τους δημιουργούς της τόσο ως εφαρμογή μέσω του αποθετηρίου λογισμικού npm αλλά και ως docker image μέσω του αποθετηρίου εικόνων dockerhub.

4.3.8 Διασύνδεση υπηρεσιών

Στο μοντέλο των μικροϋπηρεσιών, βασικό χαρακτηριστικό είναι η επικοινωνία των ξεχωριστών υπηρεσιών και η ανταλλαγή μηνυμάτων για την επίτευξη των λειτουργικοτήτων του συστήματος. Σε αυτήν την υποενότητα θα αναλυθεί ο τρόπος με τον οποίο οι μικροϋπηρεσίες επικοινωνούν μεταξύ τους, καθώς και η φύση και το περιεχόμενο των μηνυμάτων που ανταλλάσουν.

Στο παρακάτω σχήμα (σχήμα 4.23) φαίνεται ο γράφος που οπτικοποιεί τα κανάλια επικοινωνίας μεταξύ των μικροϋπηρεσιών, καθώς και τα κανάλια επικοινωνίας των μικροϋπηρεσιών με το blockchain.



Σχήμα 4.23: Γράφος οπτικοποίησης των καναλιών επικοινωνίας των μικροϋπηρεσιών

Εδώ αναλύεται η επικοινωνία κάθε μικροϋπηρεσίας:

- **Contracts Migrator**: Η υπηρεσία εκτελεί αίτημα HTTP κατά την μεταφόρτωση των contracts στο Ethereum blockchain. Επίσης, εκτελεί αίτημα HTTP για την μεταφόρτωση των contract artifacts στην υπηρεσία Contracts Provider.
- **Concordia Application**: Η υπηρεσία εκτελεί αίτημα HTTP για την λήψη των contract artifacts από την υπηρεσία Contracts Provider, εκτελεί αιτήματα HTTP για την διενέργεια συναλλαγών στο Ethereum blockchain και, τέλος, δημιουργεί κανάλι επικοινωνίας UDP με την υπηρεσία Rendezvous Server, για την ανακάλυψη ομότιμων χρηστών (peers) στο δίκτυο IPFS.
- **Pinner**: Η υπηρεσία εκτελεί αίτημα HTTP για την λήψη των contract artifacts από την υπηρεσία Contracts Provider, εκτελεί αιτήματα HTTP για την ανανέωση και παρατήρηση της κατάστασης του contract στο Ethereum blockchain και τέλος δημιουργεί κανάλι επικοινωνίας UDP με την υπηρεσία Rendezvous Server για την ανακάλυψη peers στο δίκτυο IPFS.
- **Rendezvous Server**: Η υπηρεσία διατηρεί ανοιχτά κανάλια επικοινωνίας UDP με τους ομότιμους χρήστες, μέσω των οποίων ενημερώνει την λίστα των διαθέσιμων, ενεργών χρηστών.
- **Contracts Provider**: Η υπηρεσία δεν υποκινεί καμία επικοινωνία, παρά μόνο απαντά σε αιτήματα επικοινωνίας από άλλες υπηρεσίες.

4.3.9 Ροή πληροφορίας

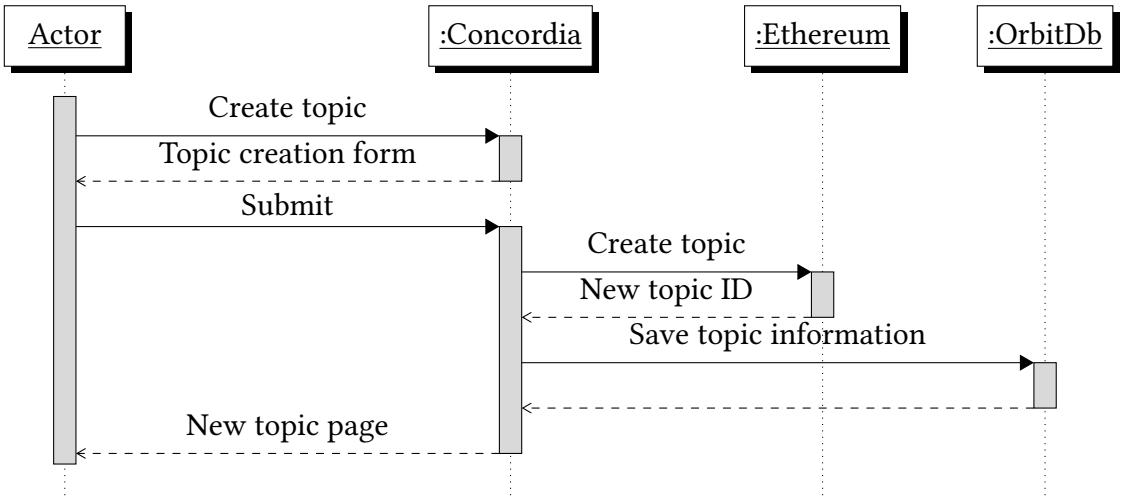
Στην παρούσα υποενότητα θα αναλυθεί η ροή της πληροφορίας στο σύστημα. Λόγω των πολλαπλών υπηρεσιών, της κατάτμησης την πληροφορίας και των διαφορετικών σημείων αποθήκευσης της, η ροή της πληροφορίας στο σύστημα ακολουθεί ένα σχετικά περίπλοκο μονοπάτι (σε σχέση με κλασσικές, μονολιθικές, κεντροποιημένες εφαρμογές).

Αρχικά θα γίνει αναφορά στη διαδικασία αποθήκευσης των νέων πληροφοριών. Η μοναδική πηγή παραγωγής δεδομένων στο σύστημα είναι οι χρήστες και κατ' επέκταση η υπηρεσία Concordia Application, εφόσον είναι η μοναδική υπηρεσία με την οποία αυτοί αλληλεπιδρούν. Τα δεδομένα που δημιουργούν οι χρήστες (πληροφορίες χρηστών, τίτλοι θεμάτων και περιεχόμενο μηνυμάτων) κατατέμνονται πριν αποθηκευτούν. Η πληροφορία που εισάγεται στο σύστημα διαχωρίζεται σε δύο μέρη. Στο blockchain αποθηκεύεται ένας δείκτης προς τα δεδομένα, ενώ τα πραγματικά δεδομένα αποθηκεύονται στη βάση OrbitDB. Ο δείκτης εκτός από την άμεση χρησιμότητα στην εύρεση των δεδομένων, παρέχει και την έμμεση λειτουργικότητα της δημιουργίας απαραίτητων μεταδομένων όπως ο αριθμός των θεμάτων στο σύστημα ή των μηνυμάτων σε ένα θέμα.

Από την πλευρά της εύρεση των πληροφοριών στο σύστημα, η ροή είναι ως εξής. Αρχικά, είναι απαραίτητη η αναζήτηση στο blockchain για την εύρεση του δείκτη προς τα δεδομένα. Έπειτα, τα δεδομένα μπορούν να ανακτηθούν μέσω του IPFS από τον εκάστοτε χρήστη ή από κάποιον Pinner.

Τέλος, παρακάτω δίνεται ένα παράδειγμα εισαγωγής πληροφορίας στο σύστημα και έπειτα ανάκτησης της ίδιας πληροφορίας.

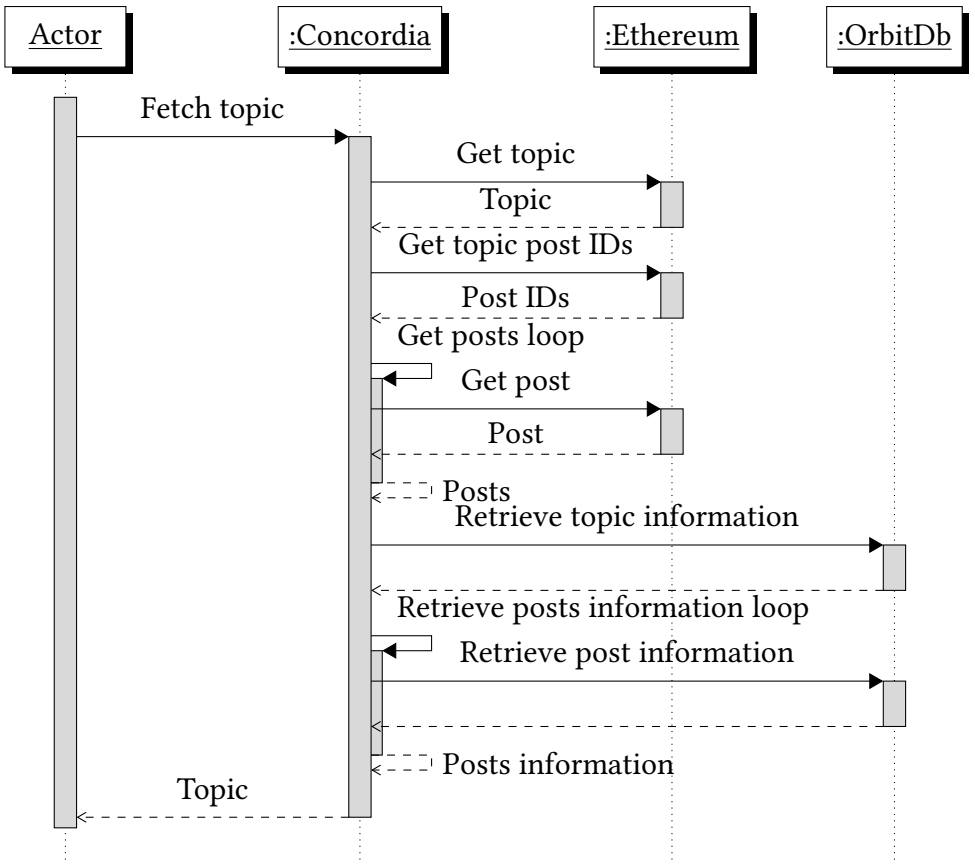
Έστω, χρήστης που δημιουργεί νέο θέμα. Τα δεδομένα που παράγονται είναι ο τίτλος του θέματος και το περιεχόμενο του πρώτου μηνύματος. Μεταδεδομένα της δημιουργίας είναι η διεύθυνση του δημιουργού του θέματος. Για την αποθήκευση του θέματος στο σύστημα δημιουργείται πρώτα συναλλαγή στο blockchain ώστε να δημιουργηθεί μία νέα εγγραφή στον πίνακα των θεμάτων. Η εγγραφή αυτή δεν περιέχει τίποτα παρά μόνο τη διεύθυνση του δημιουργού χρήστη. Αν η συναλλαγή είναι επιτυχής, θα επιστραφεί ο αύξων αριθμός του νέου θέματος. Έπειτα, στην προσωπική βάση OrbitDB του χρήστη και στον πίνακα των θεμάτων θα προστεθεί εγγραφή με αναγνωριστικό τον αύξων αριθμό του θέματος όπου θα αποθηκευτούν τα δεδομένα του τίτλου και πρώτου μηνύματος. Στο σχήμα 4.24 παρουσιάζεται γραφικά η διαδικασία.



Σχήμα 4.24: Διάγραμμα ακολουθίας δημιουργίας θέματος

Έστω, χρήστης που επιθυμεί να διαβάσει το προηγούμενο θέμα. Αρχικά, πρέπει να διαβα-

στούν τα μεταδεδομένα του συγκεκριμένου θέματος από το blockchain. Έπειτα, διαβάζονται από το blockchain οι αύξοντες αριθμοί των μηνυμάτων που έχουν δημοσιευτεί στο θέμα αυτό. Σε μία τελευταία ανάκτηση από το blockchain διαβάζονται τα μεταδομένα του κάθε μηνύματος. Έπειτα, η πληροφορία αυτή εμπλουτίζεται από τα δεδομένα του θέματος και των μηνυμάτων, τα οποία ανακτώνται από τις προσωπικές βάσεις Orbit κάθε χρήστη. Στο σχήμα 4.25 φαίνεται το διάγραμμα ροής της πληροφορίας κατά την ανάκτηση πληροφοριών από το σύστημα.



Σχήμα 4.25: Διάγραμμα ακολουθίας εύρεσης και ανάκτησης θέματος

4.4 Προβλήματα ανάπτυξης

Σε αυτήν την ενότητα περιγράφονται οι μεγαλύτερες δυσκολίες που αντιμετωπίστηκαν κατά την ανάπτυξη της πλατφόρμας. Αυτές μπορεί να αναφέρονται σε τεχνικά θέματα, αλλά και στις κοινωνικές και πολιτισμικές συνθήκες που επικρατούν στον χώρο των DApps και των crypto γενικότερα.

Μία από τις μεγαλύτερες τροχοπέδες που καθυστέρησε σοβαρά την ανάπτυξη ήταν η πρωιμότητα των βιβλιοθηκών και των εργαλείων ανάπτυξης. Οι βασικότερες βιβλιοθήκες που χρησιμοποιήθηκαν ήταν σε πρώτο ή δεύτερο πειραματικό στάδιο (alpha και beta phase αντίστοιχα). Συγκεκριμένα:

- Όλα τα εργαλεία της σουίτας Truffle ήταν σε alpha phase κατά την ανάπτυξη (κάποια έχουν περάσει σε beta πλέον).

- Το IPFS (συγκεκριμένα η βιβλιοθήκη js-ipfs) βρίσκεται ακόμα σε alpha έκδοση.
- Η OrbitDB βρίσκεται ακόμα σε alpha phase.
- Η γλώσσα των contracts, Solidity, ακόμα δεν έχει βγάλει version 1.0 καθώς αλλάζει διαρκώς με breaking changes[55].

Αυτή η έλλειψη ώριμων βιβλιοθηκών και εργαλείων προκάλεσε μείζονα προβλήματα. Συχνά έπρεπε να διορθωθούν προβλήματα των βιβλιοθηκών, ή να γίνει δουλειά που να τα παρακάμπτει. Άλλες φορές απαιτήθηκαν πολλές ώρες αποσφαλμάτωσης και δοκιμών ώστε να λειτουργήσουν τα χαρακτηριστικά που υπόσχονταν τα εργαλεία.

Ένα άλλο πρόβλημα ήταν η έλλειψη εργαλείων για ορισμένες διαδικασίες. Δύο βασικά παραδείγματα αυτού αποτελούν πρώτον η έλλειψη υποστήριξης για integration/end-to-end testing των contracts κατά την ανάπτυξη (πλέον υπάρχουν κάποιες λύσεις) και δεύτερον η έλλειψη έτοιμων διαδικασιών, plugins και integrations του Jenkins με τα εργαλεία ανάπτυξης και ειδικά με τη σουίτα Truffle.

Σε παρόμοια κατάσταση βρίσκεται και η γενική συναίνεση σχετικά με τα best practices. Σε διάφορα μέρη της ανάπτυξης παρατηρήθηκε ότι δεν υπήρχε κάποια διαμορφωμένη άποψη στην κοινότητα και κάθε ομάδα ανάπτυξης εφάρμοζε την δική της ιδέα. Αυτό καθιστά δύσκολη την ανάπτυξη από αρχάριους προγραμματιστές χωρίς καθοδήγηση. Ένα άλλο σχετικό πρόβλημα που παρατηρήθηκε είναι ότι στον χώρο υπάρχει ακόμα πολύς θόρυβος, δηλαδή σημαντικό μέρος των πηγών που βρίσκονται στο διαδίκτυο είναι αντικρουόμενες ή σε πολλές περιπτώσεις οι προτάσεις τους απορρίπτονται από την κοινότητα.

Τελικώς, ένα μη τεχνικό ζήτημα που έπρεπε να αντιμετωπιστεί είναι η αβεβαιότητα της βιωσιμότητας, εξέλιξης και αποδοχής της τεχνολογίας blockchain και των εφαρμογών που βασίζονται σε αυτήν από το ευρύ κοινό. Αυτό συναίνεσε αρνητικά, καθώς δημιούργησε μία επιτακτικότητα προσοχής της εμπειρίας του χρήστη (UX), κάτι που φυσιολογικά δεν αποτελεί σημαντικό μέρος της ανάπτυξης ενός PoC. Η ανάγκη για προσοχή του UX πηγάζει από την ανάγκη για συγκράτηση των χρηστών (user retention) με στόχο την αντιστροφή της αβεβαιότητας και την παροχή μίας γνησίως ευχάριστης εμπειρίας.

4.5 Χαρακτηριστικά που υλοποιήθηκαν

Όπως αναλύθηκε στο προηγούμενο κεφάλαιο, κατά την υλοποίηση εμφανίστηκαν διάφορα προβλήματα που δεν είχαν προβλεφθεί και τα οποία προκάλεσαν καθυστερήσεις στην ολοκλήρωση των tasks. Λόγω των καθυστερήσεων αυτών έγιναν διάφορες αναδιαμορφώσεις του προγραμματισμού των Sprint καθώς και διαπραγματεύσεις της σημαντικότητας των χαρακτηριστικών. Από τον επανασχεδιασμό και τις προσαρμογές αυτές προέκυψαν μερικές αλλαγές στο τελικό σετ των χαρακτηριστικών της πλατφόρμας σε σχέση με ό,τι είχε αρχικά προδιαγραφεί.

[55] Από τη σελίδα του πηγαίου κώδικα `https://github.com/ethereum/solidity`

Τα χαρακτηριστικά και οι αντίστοιχες Λειτουργικές Απαιτήσεις που τελικά υλοποιήθηκαν είναι:

- Η εγγραφή χρήστη και η δημιουργία των τοπικών βάσεων του, όπως περιγράφονται στις **<ΛΑ-1>** & **<ΛΑ-3>** και στο σενάριο χρήσης 3.6.1.
- Η αυτόματη είσοδος χρήστη, όπως περιγράφεται στη **<ΛΑ-2>** και στο σενάριο χρήσης 3.6.2.
- Η δημιουργία θέματος και η δημιουργία ψηφοφοριών, όπως περιγράφονται στις **<ΛΑ-4>** & **<ΛΑ-9>** και στο σενάριο χρήσης 3.6.3.
- Η περιήγηση στα υπάρχοντας θέματα, όπως περιγράφεται στη **<ΛΑ-5>** και στο σενάριο χρήσης 3.6.4.
- Η δημοσίευση μηνύματος, όπως περιγράφεται στη **<ΛΑ-6>** και στο σενάριο χρήσης 3.6.5.
- Η επεξεργασία μηνύματος, όπως περιγράφεται στη **<ΛΑ-7>** και στο σενάριο χρήσης 3.6.6.
- Η ψήφιση σε ψηφοφορία, όπως περιγράφεται στη **<ΛΑ-10>** και στο σενάριο χρήσης 3.6.7.
- Η ψήφιση σε μηνύματα, όπως περιγράφεται στη **<ΛΑ-8>** και στο σενάριο χρήσης 3.6.8.
- Η διαγραφή των τοπικών δεδομένων, όπως περιγράφεται στη **<ΛΑ-11>** και στο σενάριο χρήσης 3.6.9.

Τα παραπάνω αντιστοιχούν σε 11 ολοκληρωμένες από τις 13 προδιαγεγραμμένες ΛΑ ή πλήρωση 84.6%, ποσοστό που θεωρείται από τους συγγραφείς επαρκές για την εξαγωγή συμπερασμάτων για τον χώρο των DApps και υπερβάλλον για τα πλαίσια ενός PoC. Στο παράρτημα 5.2.4 παρατίθενται τα στιγμιότυπα οθόνης των υλοποιημένων χαρακτηριστικών.

Το χαρακτηριστικό το οποία παραλήφθηκε είναι η δημιουργία κοινοτήτων και ο ορισμός εξωτερικών contracts για τα tokens τους, όπως περιγράφονται στις **<ΛΑ-12>** & **<ΛΑ-13>** και στο σενάριο χρήσης 3.6.10.

Όσον αφορά στις Μη Λειτουργικές Απαιτήσεις, η **<ΜΛΑ-1>**, η οποία απαιτεί τη μεγιστοποίηση της αποκέντρωσης της πλατφόρμας, εκπληρώθηκε σε ικανοποιητικό βαθμό, παρότι φαινομενικά το σύστημα περιλαμβάνει σημεία κεντροποίησης για ορισμένες λειτουργίες του. Ενώ, δηλαδή, η διάθεση του frontend της εφαρμογής και των contract artifacts, καθώς και η διαδικασία ανακάλυψης των IPFS peers πραγματοποιούνται μέσω υπηρεσιών που εκτελούνται σε κεντρικούς εξυπηρετητές, στην πραγματικότητα είτε απλά παρέχουν σημαντικές διευκολύνσεις στο περιβάλλον ανάπτυξης, είτε αποτελούν προσωρινά εμπόδια που σχετίζονται με τους τρέχοντες περιορισμούς των υποκείμενων τεχνολογιών. Έτσι, από τη μία πλευρά ο κώδικας της εφαρμογής είναι εφικτό να διατίθεται στους χρήστες με αποκεντρωμένο τρόπο (π.χ. μέσω του ίδιου του IPFS), από την άλλη η παρούσα ανάγκη ύπαρξης των rendezvous server για το peer discovery μένει να επιλυθεί από την προγραμματιστική ομάδα του IPFS.

Επίσης, η ΜΛΑ που αφορά στην ελαχιστοποίηση των fees (**<ΜΛΑ-2>**) ακολουθήθηκε κατά το δυνατόν σε ολόκληρες τις διαδικασίες του σχεδιασμού και της υλοποίησης. Αυτό επιτεύχθηκε τόσο με την αποθήκευση των απολύτως απαραίτητων δεδομένων επί του blockchain, όσο και με τη βελτιστοποίηση του κώδικα των smart contracts, ώστε να εκτελείται με τον μικρότερο δυνατό αριθμό υπολογιστικών κύκλων.

Τέλος, η ΜΛΑ που σχετίζεται με την αναβαθμισιμότητα των contracts (**<ΜΛΑ-3>**) καταστρατηγήθηκε, λόγω του επιπρόσθετου χρόνου που θα απαιτούσε μία τέτοια υλοποίηση. Ωστόσο, υπάρχουν ήδη πλήρως λειτουργικές μέθοδοι για την επίτευξη αυτού του στόχου, με διαθέσιμα plugin[56] που μπορούν να ενσωματωθούν απευθείας στις υπάρχοντες ροές εργασίας.

4.5.1 Διαφορές σχεδιασμού-υλοποίησης

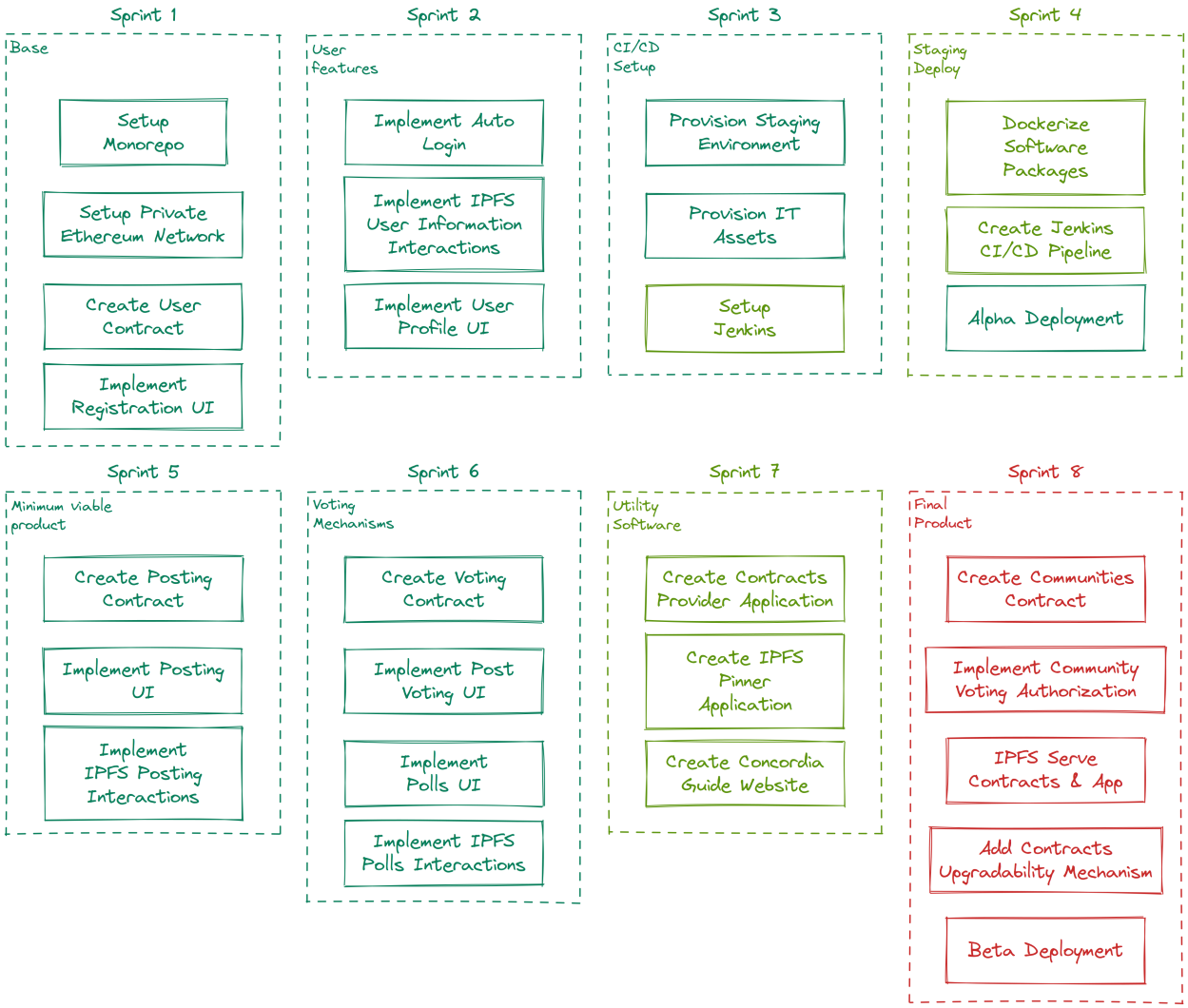
Σημαντικές διαφορές υπήρξαν επίσης στην διαδικασία υλοποίησης τόσο όσον αφορά στον αριθμό και στις λειτουργίες των διαφορετικών πακέτων λογισμικού όσο και τον χρονοπρογραμματισμό. Προστέθηκαν τρεις νέες υπηρεσίες, η υπηρεσία «Concordia Contracts Provider», ο προσαρμοσμένος IPFS pinner «Concordia Pinner» και η ιστοσελίδα «Concordia Guide».

Η ανάγκη για τα νέα πακέτα λογισμικού προέκυψε κατά την πορεία υλοποίησης της διπλωματικής και προστέθηκαν στον χρονοπρογραμματισμό που είχε γίνει στην αρχή της εργασίας. Στην προσαρμογή αυτή βοήθησαν ιδιαίτερα οι Agile τακτικές που ακολουθήθηκαν και η προσαρμοστικότητα που προσφέρει το Scrum σε μεταβαλλόμενες απαιτήσεις.

Τέλος, κατά την υλοποίηση έγινε γρήγορα αντιληπτή η αξία που προσφέρουν ένα δοκιμαστικό περιβάλλον (staging environment) σε συνδυασμό με ένα CI/CD σύστημα. Για το λόγο αυτό λήφθηκε η απόφαση να μεταφερθεί το sprint που αφορούσε αυτά πολύ νωρίτερα στην διαδικασία υλοποίησης, ώστε να μεγιστοποιηθεί η χρήση του.

Εποπτικά, η διαδικασία της υλοποίησης περιγράφεται στο παρακάτω σχήμα (σχήμα 4.26). Με σκούρο πράσινο χρώμα εμφανίζονται τα tasks τα οποία υπήρχαν στο χρονοπρογραμματισμό από τη αρχή και υλοποιήθηκαν, με ανοιχτό πράσινο αυτά τα οποία δεν υπήρχαν στον αρχικό προγραμματισμό αλλά υλοποιήθηκαν και με κόκκινο αυτά τα οποία δεν υλοποιήθηκαν.

[56] `https://docs.openzeppelin.com/upgrades-plugins/1.x/`



Σχήμα 4.26: Διαχωρισμός σε sprints

Κεφάλαιο 5

Συμπεράσματα και ανοιχτά θέματα

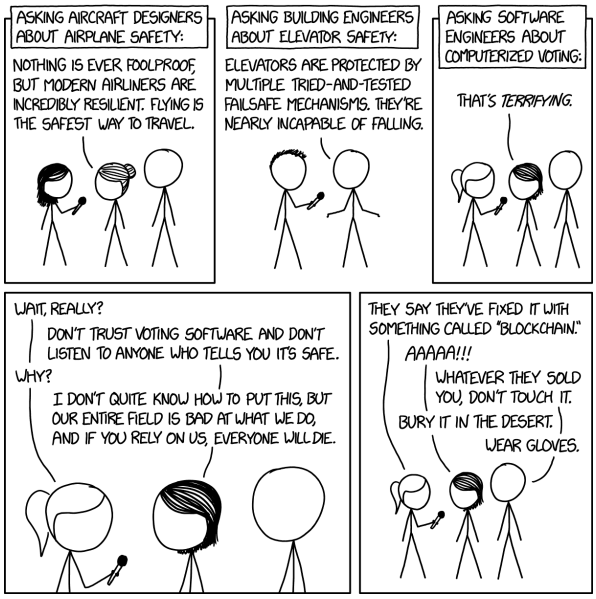
5.1 Συμπεράσματα

Συνοψίζοντας, μέσω της ανάπτυξης της πιλοτικής εφαρμογής Concordia γίνεται φανερό ότι είναι εφικτή η υλοποίηση μίας πλήρως αποκεντρωμένης κοινωνικής πλατφόρμας, η οποία να εκπληρώνει τον στόχο που τέθηκε στην ενότητα 1.4, σύμφωνα πάντα με τον σχεδιασμό του κεφαλαίου 3.

Μέσω της αρχιτεκτονικής αποκέντρωσης των τριών επιπέδων της τεχνολογικής στοίβας (βλ. ενότητα 3.2), δημιουργείται ένας πολιτικά αποκεντρωμένος ψηφιακός χώρος, ο οποίος κατοχυρώνει την ελευθερία του λόγου των συμμετεχόντων και παρέχει παντοδύναμες αυθεντικές δημοκρατικές διαδικασίες.

Ωστόσο, θα πρέπει να επισημανθεί ότι η εφαρμογή χαρακτηρίζεται από ορισμένα μειονεκτήματα, τα οποία σχετίζονται, κυρίως, με την πρώιμη κατάσταση ανάπτυξης των επιλεγμένων τεχνολογιών:

- Στο Application tier, μέσω της χρήσης του Ethereum, εισάγονται όλα εκείνα τα ζητήματα που συνοδεύουν επί του παρόντος το blockchain και τα smart contracts. Τα βασικότερα από αυτά είναι τα τέλη των συναλλαγών, η ανάγκη χρήσης επιπρόσθετων λογισμικών (π.χ. MetaMask) και η κλιμακοθετησιμότητά (scalability) των DApp. Σε γενικές γραμμές, το κλίμα στην παγκόσμια προγραμματιστική κοινότητα παραμένει αρκέτα πολωμένο ως προς το αν τελικά πλατφόρμες όπως το Ethereum θα μπορέσουν να ξεπεράσουν τα διάφορα προβλήματα και να ανταπεξέλθουν στις προσδοκίες.



Σχήμα 5.1: https://xkcd.com/2030/

- Στο Data tier, το IPFS και η OrbitDB αποτελούν επίσης ιδιαίτερα καινοτόμα λογισμικά και δε θεωρούνται ακόμα production-ready. Αυτό έχει ως αποτέλεσμα να εισάγουν με τη σειρά τους διάφορα προβλήματα, τα οποία σχετίζονται κυρίως με την εύρεση των peers (το οποίο βασίζεται προσωρινά σε signalling servers[57]) και το replication των δεδομένων.

Τέλος, τονίζεται πως, παρ' όλες τις τρέχουσες δυσκολίες, οι προγραμματιστικές κοινότητες των παραπάνω τεχνολογιών εργάζονται αδιάκοπα για τη βελτίωση τους, ενώ παρόμοια εναλλακτικά project μπορούν ανά πάσα στιγμή να αντικαταστήσουν αυτά που επιλέχθηκαν στην τρέχουσα υλοποίηση της τεχνολογικής στοίβας.

5.2 Ανοιχτά θέματα

5.2.1 Διαχείριση των τελών του Ethereum

Οι ανάγκες κάθε υπολογιστικού συστήματος σε πόρους που σχετίζονται με τις διάφορες λειτουργίες του (π.χ. επεξεργασία, αποθήκευση δεδομένων, δίκτυα) μεταφράζονται σε κάποιο οικονομικό κόστος. Στην περίπτωση της παρούσας εφαρμογής, ενώ η αποθήκευση των δεδομένων διαμοιράζεται αυτοβούλως ανάμεσα στους συμμετέχοντες κόμβους, η χρήση του Ethereum απαιτεί από τα μέλη την καταβολή τελών για τη δημιουργία συναλλαγών. Αν και αυτά τα τέλη είναι απαραίτητα για τη λειτουργία του blockchain και την προάσπισή του από επιθέσεις, αποτελούν ισχυρό εμπόδιο για την ένταξη των τελικών χρηστών στο οικοσύστημα των αποκεντρωμένων εφαρμογών του Ethereum.

[57] Βλ. και `https://github.com/libp2p/js-libp2p/issues/385`.

Στα πλαίσια της εφαρμογής Concordia, η λήψη μέτρων για τη διαχείριση των τελών θεωρείται υψίστης σημασίας. Ωστόσο, η συμπερίληψη ενός τέτοιου μηχανισμού θα περιέπλεκε εξαιρετικά τον σχεδιασμό της και, ως εκ τούτου, λήφθηκε η απόφαση να συμπεριληφθεί ως πρόταση για μελλοντική της επέκταση. Ένας τέτοιος μηχανισμός θα παρείχε τη δυνατότητα στα μέλη της πλατφόρμας να τη χρησιμοποιούν χωρίς να κατέχουν ή να δαπανούν ETH. Αυτό θα ήταν εφικτό μέσω της χρήσης «μετασυναλλαγών»[58], οι οποίες θα μεταβίβαζαν την αποπληρωμή των τελών στις κοινότητες που ανήκουν οι χρήστες.

Αυτή τη στιγμή υπάρχουν ήδη προσεγγίσεις υλοποιήσεων τέτοιου είδους μηχανισμών, όπως το Gas Station Network[59], ενώ η προγραμματιστική ομάδα του Ethereum εργάζεται ενεργά για την εγγενή υποστήριξη αυτής της δυνατότητας από την ίδια την πλατφόρμα.

5.2.2 Διανομή των Ethereum token

Στον φυσικό κόσμο, η έγκυρη και ανώνυμη διανομή ενός συνόλου μοναδικών πιστοποιητικών αυθεντικοποίησης στα μέλη μίας κοινότητας θα μπορούσε να ήταν μία διαδικασία, η οποία να απαιτούσε την φυσική παρουσία των χρηστών και την επιλογή ενός λαχνού-πιστοποιητικού από μία κληρωτίδα. Σε αυτήν την περίπτωση θα έπρεπε είτε να υπήρχε ολομέλεια και, έτσι, διαμοιρασμός της εμπιστοσύνης σε όλα τα μέλη, είτε να υπήρχε μεταβίβαση της εμπιστοσύνης σε μία επιτροπή.

Στον ψηφιακό κόσμο, το παραπάνω ζήτημα αποτελεί μία ιδιαίτερη πρόκληση με ποικίλες προσεγγίσεις σχετικά με την επιλογή των συστημάτων που θα χρησιμοποιηθούν, καθώς και των οντοτήτων στις οποίες θα εκχωρηθεί εμπιστοσύνη.

Στην παρούσα εφαρμογή, η υλοποίηση μηχανισμών για την ανώνυμη διανομή των Ethereum token των κοινοτήτων με τρόπο που να μην απαιτείται η εκχώρηση εμπιστοσύνης σε τρίτους, τέθηκε εκτός του πλαισίου της εργασίας, εξαιτίας της παρέκκλισης από το κεντρικό θέμα και της πολυπλοκότητας της. Όπως είναι σχεδιασμένη αυτήν τη στιγμή, η Concordia δύναται να υποστηρίξει ποικίλες αφηρημένες διαδικασίες οι οποίες να κατοχυρώνουν την εγκυρότητα των εκάστοτε μελών, αλλά όχι την ανωνυμία τους. Εκείνη, όσο η διαδικασία βασίζεται σε κάποια κεντρική οντότητα αυθεντικοποίησης, δε μπορεί να διασφαλιστεί, καθώς θα απαιτεί πάντα την εκχώρηση εμπιστοσύνης από τον τελικό χρήστη στα υπολογιστικά συστήματα της πρώτης. Η εμφάνιση του προβλήματος οφείλεται στο γεγονός ότι η ανωνυμοποίηση των πιστοποιητικών θα πρέπει να λάβει χώρα εντός των των προαναφερθέντων συστημάτων, τα οποία, ως επί το πλείστον, θα είναι συγκεντρωτικής λογικής.

Για παράδειγμα, έστω ότι μία κεντρική αρχή με δικό της σύστημα αυθεντικοποίησης αρχιτεκτονικής πελάτη-εξυπηρετητή αποφασίζει να συμμετάσχει στην πλατφόρμα της Concordia, δημιουργώντας μία κοινότητα και ορίζοντας ένα εξωτερικό smart contract για τα token των μελών της. Ο μηχανισμός διανομής των token θα μπορούσε να ήταν η εγγραφή του χρήστη στο

[58] Μετασυναλλαγή (meta-transaction) θεωρείται μία συναλλαγή που υπογράφεται από τον Ethereum λογαριασμό του χρήστη και προωθείται σε κάποιον τρίτο για να την εκτελέσει επί του blockchain.

[59] `https://opengsn.org/`

κεντρικό σύστημα της αρχής, η δήλωση μίας Ethereum διεύθυνσής του και η αποστολή ενός token αυθεντικοποίησης σε αυτήν. Κάτι τέτοιο θα έδινε τη δυνατότητα στους διαχειριστές του συστήματος να εντοπίζουν με ευκολία τις πραγματικές ταυτότητες των μελών της κοινότητας πίσω από κάθε token της, αίροντας, έτσι, την ανωνυμία των τελευταίων.

Λύση στο παραπάνω πρόβλημα μπορεί να επέλθει μόνο με την υλοποίηση μίας διαδικασίας ανωνυμοποίησης των token επί του blockchain. Αυτό απαιτεί την ύπαρξη ενός μηχανισμού στο οικοσύστημα του Ethereum, ο οποίος να παρέχει τη δυνατότητα μεταφοράς των token αποκρύπτοντας τις διευθύνσεις προέλευσης και προορισμού τους. Έτσι, οι χρήστες απλώς θα μετακινούσαν τα token που αρχικά παρέλαβαν σε μία διεύθυνση μη προσδιορίσιμη από τρίτους.

Στο ευρύτερο οικοσύστημα των blockchain υπάρχουν ήδη υλοποιήσεις που προσφέρουν αυτήν την δυνατότητα επί του εγγενούς τους νομίσματος (π.χ. Monero, Zcash), ενώ διάφορες ομάδες εργάζονται ενεργά για την ανάπτυξη τέτοιων μηχανισμών και στο Ethereum. Αν και υπάρχουν διαφοροποιήσεις στις προσεγγίσεις τους, η κύρια τεχνολογία στην οποία βασίζονται είναι αυτή των λεγόμενων «zero knowledge proof», με επικρατέστερα πρωτόκολλα τα zk-SNARK και zk-STARK. Ως μία ήδη λειτουργική λύση τύπου μίκτη συναλλαγών θα μπορούσε να θεωρηθεί ο Tornado[60], ο οποίος παρέχει τη δυνατότητα ανώνυμης μεταφοράς ETH ή ERC20 token αξιοποιώντας τα zk-SNARK.[20]

5.2.3 Εναλλακτικά συστήματα ψηφοφορίας

Επί του παρόντος, η εφαρμογή λειτουργεί αποκλειστικά αμεσοδημοκρατικά, με καθολικές ψηφοφορίες και ισάξιες ψήφους, όπως ορίζεται από το έξυπνο συμβόλαιο `Voting.sol`.

Τροποποιώντας την τρέχουσα υλοποίηση, η πλατφόρμα μπορεί να υποστηρίξει εναλλακτικά συστήματα ψηφοφορίας, μέσω της ανάπτυξης προσαρμοσμένων έξυπνων συμβολαίων. Αυτό θα έχει ως αποτέλεσμα να παρέχεται στην κάθε κοινότητα η δυνατότητα να ορίσει το voting smart contract που επιθυμεί, ανάλογα με τις ανάγκες και τις επιδιώξεις της.

Μερικά συστήματα ψηφοφοριών που μπορούν να ενσωματωθούν (και ακόμα και να συνδυαστούν) στην πλατφόρμα είναι τα παρακάτω:

- Ψήφοφορία με σειρά προτίμησης (ranked voting): το εκλογικό σώμα θα έχει τη δυνατότητα να ψηφίζει με σειρά προτίμησης μεταξύ των διαθέσιμων επιλογών των ψηφοφοριών.
- Πολλαπλή ψήφος: ορισμένοι ψηφοφόροι θα έχουν ισχυρότερη ψήφο βάσει κάποιου κριτηρίου (π.χ. βάσει του reputation τους).
- Έμμεση ψηφοφορία: κάθε μέλος θα ορίζει αντιπρόσωπο για μία ή περισσότερες ψηφοφορίες, για ορισμένο ή αόριστο χρονικό διάστημα.
- Μερική ψηφοφορία: επιβολή περιρισμών στο δικαίωμα ψήφου, βάσει κάποιου κριτηρίου (π.χ. ημερομηνία εγγραφής χρήστη).

[60] `https://tornado.cash/`

Φυσικά τα παραπάνω είναι καθαρά ενδεικτικά, πράγμα που σημαίνει ότι θα είναι εφικτό να μπορούν να δημιουργούνται εντελώς διαφορετικά συμβόλαια συστημάτων ψηφοφορίας, ανά πάσα στιγμή και από τον οποιονδήποτε. Η δε σύνδεσή τους με την εφαρμογή θα είναι όμοια με τα ήδη υλοποιημένα, αφού η μοναδική απαιτούμενη ενέργεια θα είναι η αποθήκευση ενός pointer προς το voting contract της εκάστοτε κοινότητας.

5.2.4 Συστήματα απόδοσης εμπιστοσύνης

Μία επιπλέον προσθήκη στην εφαρμογή μπορεί είναι ένα σύστημα απόδοσης εμπιστοσύνης (reputation system). Μέσω ενός reputation system, οι χρήστες μπορούν να κερδίζουν ή να χάνουν βαθμούς εμπιστοσύνης, με τον τρόπο που ορίζεται από το εκάστοτε smart contract.

Ορισμένες ενδεικτικές χρήσεις του είναι η συνεργασία του με τους μηχανισμούς που περιγράφονται στις υποενότητες 5.2.1 και 5.2.3. Για παράδειγμα, η ισχύς της ψήφου ενός μέλους μίας κοινότητας ή το ποσό των τελών που καλείται να καταβάλλει στο Ethereum θα μπορούσαν να υπολογίζονται αναλογα με τον βαθμό εμπιστοσύνης που έχει αποκτήσει.

Υιοθετώντας την αφηρημένη λογική που περιγράφηκε στα συστήματα ψηφοφορίας της προηγούμενης παραγράφου, είναι εφικτό να παρέχεται η δυνατότητα σε κάθε κοινότητα να επιλέγει μεταξύ ενός συνόλου διαφορετικών συστημάτων απόδοσης εμπιστοσύνης για τα μέλη της, μέσω εναλλακτικών reputation smart contract. Ήδη υπάρχει μία πλούσια γκάμα τέτοιων συστημάτων που μπορούν να υλοποιηθούν επί του Ethereum, με την ταξινομία τους να ορίζεται επί μίας πληθώρας ανεξάρτητων διαστάσεων.[21] Ωστόσο, η περαιτέρω ανάλυση τους, είναι θέμα που εκτείνεται πέρα από τα πλαίσια της παρούσας διπλωματικής εργασίας.

Παραρτήματα

Παράρτημα Α΄

Στιγμιότυπα οθόνης πλατφόρμας

Παράρτημα Β΄

Στατιστικά κώδικα

Στο παρόν παράρτημα παρατίθενται πίνακες με στατιστικά στοιχεία του κώδικα της εφαρμογής Concordia, καθώς και των υλοποιημένων βιβλιοθηκών. Συγκεκριμένα, πραγματοποιήθηκε καταμέτρηση των αρχείων και των γραμμών κώδικα μέσω του προγραμμάτος cloc[61], διαδικασία στην οποία αγνοήθηκαν αυτόματα configuration και auto-generated αρχεία (π.χ. yarn.lock, .gitignore).

Concordia `https://gitlab.com/ecentrics/concordia`				
Γλώσσα	**Αρχεία**	**Κενές γραμμές**	**Σχόλια**	**Κώδικας**
JSX	54	510	14	4491
JavaScript	81	302	84	2035
Groovy	1	76	32	673
Solidity	5	142	20	553
CSS	26	87	8	426
JSON	10	0	0	352
Markdown	7	157	0	352
Dockerfile	5	68	51	128
Bourne Shell	9	27	10	117
make	1	10	8	77
YAML	1	3	0	30
SVG	6	0	0	25
HTML	1	3	23	16
diff	1	0	8	7
Σύνολο	208	1385	258	9282

Πίνακας Β.1: Concordia - στατιστικά κώδικα

[61] `https://github.com/AlDanial/cloc`

drizzle https://gitlab.com/ecentrics/drizzle				
Γλώσσα	**Αρχεία**	**Κενές γραμμές**	**Σχόλια**	**Κώδικας**
JavaScript	36	281	137	1448
JSON	1	0	0	16
Markdown	1	2	0	4
Σύνολο	38	283	137	1468

Πίνακας Β.2: drizzle - στατιστικά κώδικα

breeze https://gitlab.com/ecentrics/breeze				
Γλώσσα	**Αρχεία**	**Κενές γραμμές**	**Σχόλια**	**Κώδικας**
JavaScript	16	105	56	583
JSON	1	0	0	17
Markdown	1	2	0	4
Σύνολο	18	107	56	604

Πίνακας Β.3: breeze - στατιστικά κώδικα

eth-identity-provider https://gitlab.com/ecentrics/eth-identity-provider				
Γλώσσα	**Αρχεία**	**Κενές γραμμές**	**Σχόλια**	**Κώδικας**
JavaScript	4	36	23	211
JSON	1	0	0	13
Markdown	1	2	0	4
Σύνολο	6	38	23	228

Πίνακας Β.4: eth-identity-provider - στατιστικά κώδικα

Βιβλιογραφία

[1] *Μάθετε για το Ethereum*. URL: https://ethereum.org/el/learn/ (επίσκεψη 16/03/2021).

[2] Vitalik Buterin. *The Meaning of Decentralization*. 6 Φεβ. 2017. URL: https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274.

[3] A. Aneesh. *Virtual Migration*. 2006.

[4] Don Johnson, Alfred Menezes και Scott Vanstone. «The Elliptic Curve Digital Signature Algorithm (ECDSA)». Στο: *International Journal of Information Security* (Αύγ. 2001). DOI: 10.1007/s102070100002. URL: https://doi.org/10.1007/s102070100002.

[5] Wikipedia. *Merkle tree*. URL: https://en.wikipedia.org/wiki/Merkle_tree.

[6] Belavadi Prahalad. *Merkle proofs Explained*. 7 Ιαν. 2018. URL: https://medium.com/crypto-0-nite/merkle-proofs-explained-6dd429623dc5.

[7] R. Schollmeier. «A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications». Στο: *Proceedings First International Conference on Peer-to-Peer Computing*. 2001, σσ. 101–102. DOI: 10.1109/P2P.2001.990434.

[8] Satoshi Nakamoto. «Bitcoin: A Peer-to-Peer Electronic Cash System». Στο: *Cryptography Mailing list at https://metzdowd.com* (31 Οκτ. 2008).

[9] Wikipedia. *Blockchain*. URL: https://en.wikipedia.org/wiki/Blockchain.

[10] Vitalik Buterin. *Ethereum Whitepaper*. 2013. URL: https://ethereum.org/en/whitepaper (επίσκεψη 28/06/2021).

[11] Ethereum community. *Ethereum documentation*. URL: https://ethereum.org/en/developers/docs/ (επίσκεψη 05/09/2021).

[12] Nick Szabo. «Formalizing and Securing Relationships on Public Networks». Στο: *First Monday* 2.9 (Σεπτ. 1997). DOI: 10.5210/fm.v2i9.548. URL: https://journals.uic.edu/ojs/index.php/fm/article/view/548.

[13] Gavin Wood Andreas M Antonopoulos. *Mastering Ethereum: Building Smart Contracts and DApps*. O'Reilly Media, 2018. ISBN: 1491971940.

[14] *IPFS*. URL: https://ipfs.io/.

[15] *IPFS documentation*. URL: https://docs.ipfs.io/.

[16] GitHub Guides. *Understanding the GitHub flow*. URL: https://guides.github.com/introduction/flow/.

[17] Wikipedia. *Node.js*. URL: https://en.wikipedia.org/wiki/Node.js.

[18] *OrbitDB*. URL: https://orbitdb.org.

[19] *Getting Started with OrbitDB*. URL: https://github.com/orbitdb/orbit-db/blob/main/GUIDE.md.

[20] *Privacy on Ethereum*. URL: https://docs.ethhub.io/ethereum-roadmap/privacy/ (επίσκεψη 12/12/2021).

[21] Ferry Hendrikx, Kris Bubendorfer και Ryan Chard. «Reputation systems: A survey and taxonomy». Στο: *Journal of Parallel and Distributed Computing* 75 (Αύγ. 2014). DOI: 10.1016/j.jpdc.2014.08.004.