Πανεπιστήμιο Πειραιώς - Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών
Προηγμένα Συστήματα Πληροφορικής
Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Ψηφιοποίηση εγγράφων και διασφάλιση γνησιότητας με χρήση της τεχνολογίας Blockchain. Document digitation and validation using Blockchain technology
Ονοματεπώνυμο Φοιτητή	Δαβλιάνη Ιωάννα
Πατρώνυμο	Νικόλαος
Αριθμός Μητρώου	ΜΠΣΠ20010
Επιβλέπων	Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής

Ημερομηνία παράδοσης: Ιούνιος 2022

Τριμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης	**Μαρία Βίρβου**	**Κωνσταντίνος Πατσάκης**
Αναπληρωτής Καθηγητής	**Καθηγήτρια**	**Αναπληρωτής Καθηγητής**

Ευχαριστίες

Για την υλοποίηση της παρούσας διπλωματικής εργασίας, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Ευθύμιο Αλέπη, για την εμπιστοσύνη του, καθώς και τον υποψήφιο διδάκτορα κ. Σπύρο Παπαδημητρίου για την καθοριστική του καθοδήγηση καθόλη τη διάρκεια της φοίτησής μου.

Επίσης, θέλω να ευχαριστήσω θερμά τον συμφοιτητή μου Ηλία Παπανικολάου, με τον οποίο συνεργαστήκαμε για την ολοκλήρωση της εργασίας.

Τέλος, θέλω να ευχαριστήσω την οικογένεια μου, για την υποστήριξη της, σε όλη τη διάρκεια του εκπαιδευτικού μου βίου.

Περίληψη

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια του Προγράμματος Μεταπτυχιακών Σπουδών «Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης», και έχει ως στόχο την υλοποίηση web εφαρμογής, για την ψηφιοποίηση εγγράφων βαθμολογίας φοιτητών και τίτλων σπουδών πανεπιστημίου, αξιοποιώντας την τεχνολογία blockchain.

Έχει αναπτυχθεί κατάλληλο περιβάλλον χρήστη front-end σε angular framework προκειμένου να ανεβάζει αρχεία κατάλληλα εξουσιοδοτημένος υπάλληλος πανεπιστημίου, ο οποίος έχει επίσης δικαίωμα αναζήτησης των φοιτητών, των εγγράφων τους αλλά και των πληροφοριών όλων των blocks του blockchain. Οι φοιτητές μπορούν με την είσοδό τους στην εφαρμογή να κάνουν αναζήτηση των δικών τους εγγράφων και να δουν πληροφορίες των blocks που είναι εγγεγραμμένη η δική τους πληροφορία. Επίσης οι φοιτητές μπορούν να δουν στοιχεία επικοινωνίας των υπαλλήλων - καθηγητών του πανεπιστημίου. Οι ρόλοι των χρηστών μπορούν να επεξεργαστούν από τον διαχειριστή (admin) του συστήματος, ο οποίος μπορεί να τροποποιήσει το ρόλο των εγγεγραμμένων χρηστών.

Το backend έχει αναπτυχθεί σε αρχιτεκτονική microservices με Spring Webflux, χρησιμοποιώντας ασύγχρονες κλήσεις προς τα microservices, ενώ χρησιμοποιήθηκε MongoDB για ασύγχρονη αναζήτηση και αποθήκευση των εγγράφων. Εκτός από την απευθείας κλήση μεταξύ των microservices χρησιμοποιήθηκε message broker (RabbitMQ) ώστε να διασφαλιστεί η απρόσκοπτη επικοινωνία μεταξύ τους, και να εξασφαλιστεί ότι δεν θα υπάρξει απώλεια δεδομένων. Για την αποθήκευση εγγράφων, είναι αποδεκτά αρχεία τύπου pdf αλλά και PNG, JPG τα οποία μετατρέπονται σε PDF με τη χρήση OCR, ενώ για το tracing των requests χρησιμοποιήθηκε zipkin.

Τέλος για την ασφάλεια της εφαρμογής χρησιμοποιήθηκε keycloak server ο οποίος εκδίδει Json Web Tokens (JWT) για κάθε κλήση των συνδεδεμένων χρηστών στο front-end και επιβεβαιώνει την εγκυρότητα του στο back-end. Ο server χρησιμοποιεί postgres βάση δεδομένων για την αποθήκευση των χρηστών.

Abstract

The presented thesis was developed within the Master's Degree Programme "Advanced Computer Systems - Software Development and Artificial Intelligence" framework. Its objective was to implement a web application that digitises student grades and university degrees using blockchain technology.

It integrates a friendly front-end user interface developed in the Angular framework to upload files by a suitably authorised university employee, who also has the right to search students, their documents and information of all blockchain blocks. Upon entering the application, students can explore their documents and view the information of the blocks that host them. Also, they can view the contact information of the employees - professors of the university. User roles are editable by the administrator (admin) of the system, who can modify the role of registered users.

The backend has been developed in a microservices architecture with Spring Webflux, using asynchronous calls to microservices. MongoDB was used for asynchronous document search and storage. In addition to direct calls between microservices, a message broker (RabbitMQ) was used to ensure seamless communication between them and to ensure that there would be no data loss. For storing documents, PDF files are accepted as well as PNG and JPG files which are converted to PDF using OCR, while Zipkin was used for tracing requests.

Finally, the Keycloak server was used for the application's security, which issues Json Web Tokens (JWT) for each call of the connected users on the front-end and confirms its validity on the backend. The server uses the Postgres database for storing the users.

Πίνακας περιεχομένων

Περιεχόμενα

Εισαγωγή

Αναδρομή

Το blockchain είναι μια αποκεντρωμένη τεχνολογία διαχείρισης συναλλαγών και δεδομένων που αναπτύχθηκε πρώτα για το κρυπτονόμισμα Bitcoin. Το ενδιαφέρον για την τεχνολογία blockchain έχει αυξηθεί από τότε που δημιουργήθηκε η ιδέα το 2008. Ο λόγος για το ενδιαφέρον για το blockchain είναι ότι ενσωματώνει χαρακτηριστικά που παρέχουν ασφάλεια, ανωνυμία και ακεραιότητα δεδομένων, χωρίς την πραγματοποίηση ελέγχων των συναλλαγών από τρίτους οργανισμούς. Ως εκ τούτου, δημιουργεί ενδιαφέρουσες ερευνητικές περιοχές, ειδικά από την άποψη των τεχνικών προκλήσεων και περιορισμών.

Ένα δίκτυο blockchain αποτελείται από μία distributed βάση δεδομένων, που διατηρεί μία συνεχώς αυξανόμενη λίστα εγγραφών που επιβεβαιώνονται από τους κόμβους που συμμετέχουν σε αυτή. Τα δεδομένα καταγράφονται σε ένα public ledger, συμπεριλαμβανομένων των πληροφοριών για κάθε συναλλαγή που ολοκληρώθηκε. Οι πληροφορίες για κάθε συναλλαγή που ολοκληρώθηκε είναι διαθέσιμες σε όλους τους κόμβους. Όλη η λειτουργία είναι αυτοματοποιημένη και δεν απαιτεί επέμβαση από τρίτο. Αυτό το χαρακτηριστικό καθιστά το σύστημα πιο διαφανές από τις κεντρικές συναλλαγές που περιλαμβάνουν τρίτους. Επιπλέον, οι κόμβοι στο blockchain είναι όλοι ανώνυμοι, γεγονός που καθιστά ασφαλέστερο για άλλους κόμβους την επιβεβαίωση των συναλλαγών. Το Bitcoin ήταν η πρώτη εφαρμογή που εισήγαγε την τεχνολογία blockchain και δημιούργησε ένα αποκεντρωμένο περιβάλλον για κρυπτονομίσματα, όπου οι συμμετέχοντες μπορούν να αγοράζουν και να ανταλλάσσουν αγαθά με ψηφιακό χρήμα.

Από το 2008 μέχρι σήμερα έχουν αναπτυχθεί πάνω από 10.000 διαφορετικά κρυπτονομίσματα, βασισμένα στην ίδια κεντρική ιδέα, με παραλλαγές στον τρόπο υλοποίησης, ενώ αναμένεται ο αριθμός αυτός να αυξηθεί ακόμα περισσότερο, καθώς θεωρείται ότι είμαστε ακόμα στην αρχή της υιοθέτησης της τεχνολογίας.

Πεδία εφαρμογής

Παρόλο που η χρήση της τεχνολογίας blockchain χρησιμοποιείται εκτεταμένα και είναι διαδεδομένη κυρίως σε κρυπτονομίσματα, οι ιδιότητες της ασφάλειας, διαφάνειας και ανωνυμίας το καθιστούν κατάλληλο και σε άλλα πεδία εφαρμογής τα οποία συνεχώς αναπτύσσονται και επεκτείνονται.

Το blockchain έχει βρει πρόσφορο έδαφος στον τομέα της οικονομίας εκτός από τις οικονομικές συναλλαγές με κρυπτονομίσματα, για την ασφάλεια συναλλαγών, προστασία από ξέπλυμα βρώμικου χρήματος κ.α. Ωστόσο, έχει αρχίσει να διεισδύει και σε άλλους τομείς όπως:

- **Υγεία**: Τα δεδομένα υγείας που είναι κατάλληλα για το blockchain περιλαμβάνουν γενικές πληροφορίες όπως η ηλικία, το φύλο και βασικά δεδομένα ιατρικού ιστορικού, όπως ιστορικό εμβολιασμών ή άλλες πληροφορίες. Από μόνη της, καμία από αυτές τις πληροφορίες δεν θα μπορούσε να προσδιορίσει συγκεκριμένα οποιονδήποτε συγκεκριμένο ασθενή, κάτι που του επιτρέπει να αποθηκευτεί σε μια κοινόχρηστη αλυσίδα μπλοκ στην οποία θα μπορούσαν να έχουν πρόσβαση πολλά άτομα χωρίς ανησυχίες για την παραβίαση του απορρήτου.

  Καθώς οι εξειδικευμένες συνδεδεμένες ιατρικές συσκευές γίνονται πιο διαδεδομένες και συνδέονται όλο και περισσότερο με το αρχείο υγείας ενός ατόμου, το blockchain μπορεί να συνδέσει αυτές τις συσκευές με αυτό το αρχείο. Οι συσκευές θα μπορούν να αποθηκεύουν τα δεδομένα που δημιουργούνται σε μια αλυσίδα μπλοκ υγειονομικής περίθαλψης και να τα προσαρτούν σε προσωπικά ιατρικά αρχεία. Ένα βασικό ζήτημα που αντιμετωπίζουν επί του παρόντος οι συνδεδεμένες ιατρικές συσκευές είναι η αποθήκευση των δεδομένων που δημιουργούν — αλλά το blockchain θα μπορούσε να είναι η λύση.

- **Πολυμέσα**: Οι εταιρείες πολυμέσων έχουν ήδη αρχίσει να υιοθετούν τεχνολογία blockchain για την εξάλειψη απάτης, τη μείωση του κόστους και ακόμη και την προστασία των δικαιωμάτων πνευματικής ιδιοκτησίας (IP) περιεχομένου – όπως οι δίσκοι μουσικής. Σύμφωνα με το MarketWatch, η παγκόσμια αγορά blockchain στα μέσα ενημέρωσης και την ψυχαγωγία εκτιμάται ότι θα φτάσει τα 1,54 δισεκατομμύρια δολάρια έως το 2024.

  Μια πλατφόρμα που έχει τραβήξει τα φώτα της δημοσιότητας αξιοποιώντας την τεχνολογία blockchain για τα πολυμέσα, είναι η Eluvio, Inc. Η Eluvio Content Fabric, που κυκλοφόρησε επίσημα το 2019, χρησιμοποιεί τεχνολογία blockchain για να επιτρέπει στους παραγωγούς περιεχομένου να διαχειρίζονται και να διανέμουν premium βίντεο σε καταναλωτές και επιχειρηματικούς εταίρους χωρίς δίκτυα παράδοσης περιεχομένου.

- **Ενέργεια**: Η τεχνολογία blockchain θα μπορούσε να χρησιμοποιηθεί για την εκτέλεση συναλλαγών προμήθειας ενέργειας, αλλά και για να παρέχει περαιτέρω τη βάση για διαδικασίες μέτρησης, χρέωσης και εκκαθάρισης. Άλλες πιθανές εφαρμογές περιλαμβάνουν τεκμηρίωση ιδιοκτησίας, διαχείριση περιουσιακών στοιχείων, εγγυήσεις προέλευσης, δικαιώματα εκπομπής και πιστοποιητικά ανανεώσιμων πηγών ενέργειας.

- **Διαχείριση αρχείων:** Οι εθνικές, πολιτειακές και τοπικές κυβερνήσεις είναι υπεύθυνες για τη διατήρηση των αρχείων των ατόμων, όπως ημερομηνίες γέννησης και θανάτου, οικογενειακές καταστάσεις ή μεταβιβάσεις περιουσίας. Ωστόσο, η

διαχείριση αυτών των δεδομένων μπορεί να είναι δύσκολη, και μέχρι σήμερα ορισμένα από αυτά τα αρχεία υπάρχουν μόνο σε έντυπη μορφή. Αρκετές φορές, οι πολίτες πρέπει να μεταβαίνουν στα γραφεία της τοπικής αυτοδιοίκησης για να προβούν σε αλλαγές, κάτι που είναι χρονοβόρο, περιττό και απογοητευτικό. Η τεχνολογία blockchain θα μπορούσε να απλοποιήσει την τήρηση των αρχείων και να καταστήσει τα δεδομένα πολύ πιο ασφαλή.

- **Εκλογές**: Η τεχνολογία blockchain έχει τη δυνατότητα να κάνει τη διαδικασία ψηφοφορίας πιο εύκολα προσβάσιμη, βελτιώνοντας παράλληλα την ασφάλεια τους. Ακόμα κι αν κάποιος είχε πρόσβαση στο τερματικό με σκοπό να αλλοιώσει τα αποτελέσματα, δεν θα μπορούσε να επηρεάσει άλλους κόμβους. Κάθε ψήφος θα αποδίδοταν σε μία ταυτότητα και, καθώς η δυνατότητα δημιουργίας πλαστής ταυτότητας είναι αδύνατη, οι αρμόδιοι υπάλληλοι θα μπορούσαν να μετρήσουν τις ψήφους πιο αποτελεσματικά.

- **IoT**: Το Blockchain μπορεί να μεταμορφώσει πρακτικές σε διάφορους τομείς του IoT, όπως στις αλυσίδες εφοδιασμού, παρακολουθώντας τη θέση των αγαθών που αποστέλλονται και διασφάλιση ότι παραμένουν εντός καθορισμένων συνθηκών

Αυτά είναι ορισμένα από τα πεδία εφαρμογής της τεχνολογίας, τα οποία συνεχώς αυξάνονται με νέες λύσεις και καινοτομίες.

Αποθήκευση αρχείων

Μέχρι σήμερα για την βεβαίωση γνησιότητας ενός εγγράφου χρησιμοποιούνται σφραγίδες, υπογραφές, υδατογραφήματα και άλλες τεχνικές οι οποίες όμως παραβιάζονται σχετικά εύκολα. Με το blockchain όμως αυτό καθίσταται πρακτικά αδύνατο.

Υπάρχουν δύο τρόποι με τους οποίους μπορεί το blockchain να χρησιμοποιηθεί για την αποθήκευση αρχείων:

- Μέσα στο blockchain (on-chain): όλα τα δεδομένα αποθηκεύονται μέσα σε κάθε block στην αλυσίδα. Έτσι εάν συμβεί το οτιδήποτε, π.χ. μία επίθεση, τα δεδομένα μπορούν να ανακτηθούν και να χρησιμοποιηθουν. Όμως αυτό έχει ένα μεγάλο μειονέκτημα, καθώς είναι πολύ κοστοβόρο και χρονοβόρο, καθώς τα αρχεία έχουν μεγάλο όγκο πληροφορίας.
- Εκτός του blockchain (off-chain): μόνο τα metadata αποθηκεύονται στην αλυσίδα, ενώ το ίδιο το αρχείο αποθηκεύεται αλλού, είτε αποκεντρωμένα είτε όχι. Τα δεδομένα που αποθηκεύονται στο blockchain σε περίπτωση απώλειας ενδεχομένως να μην μπορούν να αποκατασταθούν, ωστόσο αυτή η λύση είναι είναι αποδοτική και οικονομική, και αποτελεί την πιο συμφέρουσα επιλογή.

Σχετικές έρευνες

Η τεχνολογία του blockchain, λόγω των πολλών πεδίων εφαρμογής και των όσων προσφέρει σε σχέση με παραδοσιακές τεχνικές αποθήκευσης και πιστοποίησης εγγράφων, έχει γίνει αντικείμενο περαιτέρω έρευνας και μελέτης. Η έρευνα [11] περιγράφει ότι το blockchain είναι ένα δημόσιο καθολικό που διανέμεται και αποκεντρώνεται και χρησιμοποιείται για τη διενέργεια συναλλαγών στο διαδίκτυο σε ολόκληρο το δίκτυο συστημάτων υπολογιστών. Είναι μια τεχνολογία που υιοθετήθηκε για την ασφάλεια των δεδομένων. Η μη τροποποιήσιμη ιδιότητα του blockchain βοηθά να ξεπεραστεί το πρόβλημα της πλαστογραφίας εγγράφων. Περίπου ένα εκατομμύριο φοιτητές αποφοιτούν κάθε χρόνο και οι αρχές έκδοσης των εγγράφων φαίνεται να έχουν παραβιαστεί για τα διαπιστευτήρια ασφαλείας των δεδομένων των φοιτητών. Λόγω έλλειψης αποτελεσματικού μηχανισμού κατά της παραχάραξης, τα πλαστά έγγραφα συχνά περνούν απαρατήρητα. Η χρήση φυσικών αντιγράφων εγγράφων δημιουργεί τεράστιο κόστος, καθώς περιλαμβάνει χειροκίνητη επαλήθευση, αποθήκευση χαρτιού και μη αυτόματο έλεγχο. Διάφορες υπηρεσίες αποτυγχάνουν να επαληθεύσουν τη γνησιότητα των εγγράφων και δημιουργούν κενά. Σε αυτή την εργασία, σε μια προσπάθεια να λυθεί το παραπάνω πρόβλημα, σχεδιάστηκε ένα ιδιωτικό δίκτυο blockchain πολλαπλών κόμβων χρησιμοποιώντας το δίκτυο Ethereum και δημιουργήθηκε ένας αποθηκευτικός χώρος εκτός αλυσίδας (IPFS), για την αποθήκευση των εγγράφων. Αξιολογήθηκε επίσης η απόδοση του blockchain Ethereum αναλύοντας τον αντίκτυπο διαφόρων παραμέτρων όπως το διαφορετικό επίπεδο δυσκολίας, το φορτίο, το μέγεθος δικτύου και οι αλγόριθμοι consensus.

Η έρευνα [12] περιγράφει πως η διαχείριση εκπαιδευτικών πιστοποιητικών συχνά αντιμετωπίζει προβλήματα σχετικά με το χειρισμό εγγράφων, την απώλεια δεδομένων ή κακόβουλες ενέργειες. Ειδικά στην περίπτωση των έντυπων πιστοποιητικών, η προέλευση και η ακεραιότητα των πιστοποιητικών είναι δύσκολο να επαληθευτεί. Επιπλέον, τέτοια έγγραφα μπορεί να χαθούν ή να καταστραφούν. Η επανέκδοση πιστοποιητικών μπορεί να είναι δαπανηρή, δύσκολη ή αδύνατη, π.χ. εάν ο οργανισμός έχει κλείσει. Παρόλο που η έκδοση και η υπογραφή εγγράφων επιλύει ψηφιακά ορισμένα από αυτά τα ζητήματα, εξακολουθεί να υπάρχει η απαίτηση για την ύπαρξη κεντρικών αξιόπιστων υποδομών και, τελικά, να είναι δύσκολη η επαλήθευση και η ανάκτηση χαμένων εγγράφων. Σε αυτή την εργασία, παρουσιάζεται το SPROOF, μια πλατφόρμα για την έκδοση, διαχείριση και επαλήθευση ψηφιακών εγγράφων σε ένα δημόσιο blockchain. Στην προτεινόμενη προσέγγιση, όλα τα δεδομένα που απαιτούνται για την επαλήθευση των εγγράφων και οι εκδότες αποθηκεύονται αποκεντρωμένα, με διαφάνεια και προστατεύεται η ακεραιότητα. Ακολουθώντας τις αρχές του Web of Trust, οι εκδότες μπορούν να επιβεβαιώσουν ο ένας τον άλλον με αποκεντρωμένο τρόπο. Επιπλέον, λαμβάνονται υπόψη ζητήματα επεκτασιμότητας και απορρήτου.

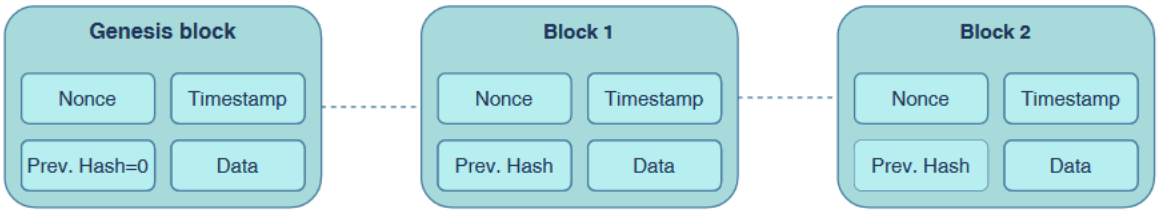
Η έρευνα [18] περιλαμβάνει τη συστηματική βιβλιογραφική ανασκόπηση αναφορικά με την τεχνολογία blockchain στην εκπαίδευση, Με αυτόν τον τρόπο, στοχεύει να προσφέρει μια λεπτομερή κατανόηση όσον αφορά τα οφέλη, τα εμπόδια, την εφαρμογή της τεχνολογίας blockchain και τους μελλοντικούς τομείς όπου η τεχνολογία blockchain μπορεί να εφαρμοστεί σε άλλους τομείς της εκπαίδευσης. Η ανάλυση αποκαλύπτει ότι η τεχνολογία blockchain στην εκπαίδευση εξακολουθεί να είναι ένας νέος κλάδος, αλλά έχει πολλές δυνατότητες να ωφελήσει τον εκπαιδευτικό τομέα γενικότερα.

Τέλος παρέχει ένα υπόβαθρο για τα εκπαιδευτικά ιδρύματα και τους ερευνητές ώστε να εξερευνήσουν άλλους τομείς στους οποίους μπορεί να εφαρμοστεί η τεχνολογία blockchain. Η εργασία εξετάζει την εφαρμογή της τεχνολογίας blockchain στην εκπαίδευση με τη βοήθεια βιβλιομετρικής ανάλυσης. Αυτή είναι μια από τις πρώτες γνωστές μελέτες που ανασκόπησαν την τεχνολογία blockchain εντοπίζοντας τα οφέλη της, τα εμπόδια και την παρούσα εφαρμογή τεχνολογίας blockchain. Με βάση την ανάλυση, προσδιορίζονται επίσης μελλοντικές περιοχές εφαρμογής, όπως η έκδοση ψηφιακών εγγράφων βαθμολογίας και ασφαλής διακίνηση των εγγράφων αυτών.

Blockchain

Χαρακτηριστικά

Το blockchain είναι μία αλυσίδα από blocks, όπου το καθένα περιέχει κρυπτογραφημένη πληροφορία του προηγούμενου block. Το πρώτο block της αλυσίδας, το οποίο δεν περιέχει προηγούμενη πληροφορία, ονομάζεται genesis. Στη συνέχεια το 2ο block, περιέχει το hash (SHA-256) του 1ου, το 3ο περιέχει το hash του 2ου κ.ο.κ. Η γενική αρχή είναι ότι το n+1 block περιέχει την πληροφορία του n block (εικόνα 1).



- Εικόνα 1 -

Το hash του κάθε block, παράγεται από την πληροφορία που αποθηκεύεται σε αυτό, τη χρονική στιγμή που δημιουργήθηκε, το hash του προηγούμενου block και τον αριθμό nonce ο οποίος παράγεται μέσα από τη διαδικασία κρυπτογράφησης.

Τα δύο δομικά χαρακτηριστικά της τεχνολογίας blockchain είναι τα consensus και immutability .

Immutability

Το immutability είναι η ικανότητα ενός δικτύου blockchain να αποτρέπει την αλλαγή των συναλλαγών που έχουν ήδη επιβεβαιωθεί. Αυτές οι συναλλαγές συνήθως σχετίζονται συχνά με τη μεταφορά κρυπτονομισμάτων, ενώ μπορεί επίσης να αναφέρονται στην εγγραφή άλλων μορφών ψηφιακών δεδομένων.

Η ακεραιότητα των δεδομένων που αποθηκεύονται στο blockchain είναι εξασφαλισμένη αφού το περιεχόμενό τους δεν μπορεί να αλλάξει μόλις η πληροφορία προστεθεί στο block. Αλλαγή της πληροφορίας θα σήμαινε ότι πρέπει να αλλάξουν όλα τα blocks που προστεθηκαν στην αλυσίδα μετά από το block που πρέπει να τροποποιηθεί. Όπως αναφέρθηκε, κάθε block περιέχει κρυπτογραφημένη την πληροφορία του προηγούμενου block, το timestamp της χρονικής στιγμής που δημιουργήθηκε το block, και την πληροφορία που θα αποθηκευτεί. Το timestamp λειτουργεί ως απόδειξη ότι η πληροφορία που περιέχει το κάθε block, υπήρχε πριν τη δημιουργία του block. Τα blocks συνθέτουν μία αλυσίδα, συνεχώς αυξανόμενη, όπου το τελευταίο block κάθε φορά περιέχει κρυπτογραφημένη

πληροφορία όλων των προηγούμενων. Έτσι η τροποποίηση με οποιονδήποτε τρόπο της πληροφορίας που περιέχει το κάθε block είναι αδύνατη καθώς για να τροποποιηθεί η πληροφορία στο n block της αλυσίδας θα πρέπει να τροποποιηθεί και το block n-1, n-2 κλπ.

Consensus

Το consensus είναι η ικανότητα των κόμβων εντός του δικτύου blockchain να συμφωνούν για την κατάσταση του δικτύου και για την εγκυρότητα των συναλλαγών. Συνήθως, η διαδικασία επίτευξης συναίνεσης ολοκληρώνεται μέσω ενός αλγόριθμου συναίνεσης.

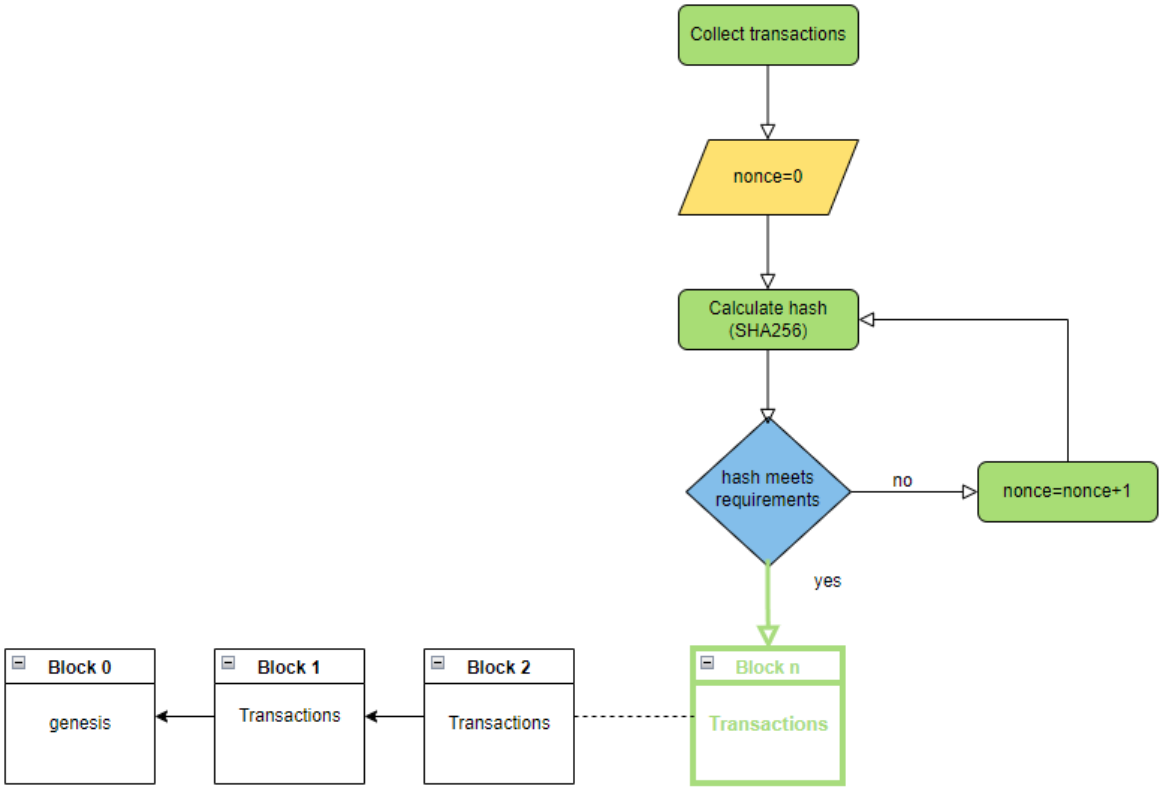
Ανάλογα με τον τρόπο λειτουργίας του μηχανισμού consensus διακρίνουμε δύο είδη, το proof of work και proof of stake.

Proof of work

Το Proof of Work (PoW) είναι μία μορφή απόδειξης στην οποία το ένα μέρος, στην συγκεκριμένη περίπτωση το block που πρέπει να προστεθεί στην αλυσίδα, αποδεικνύει ότι έχει γίνει μία συγκεκριμένη υπολογιστική προσπάθεια. Οι επαληθευτές στη συνέχεια μπορούν με ελάχιστη προσπάθεια από μέρους τους να επιβεβαιώσουν αυτή την προσπάθεια. Ο σκοπός των αλγορίθμων proof-of-work, δεν είναι να αποδείξουν ότι εκτελέστηκε συγκεκριμένη εργασία ή ότι λύθηκε ένα υπολογιστικό παζλ, αλλά να αποτρέψει κακόβουλη χρήση, απαιτώντας υψηλές απαιτήσεις κατανάλωσης ενέργειας και υλικού εξοπλισμού ώστε να παραβιαστεί η αλυσίδα, κάνοντας την παραβίαση πρακτικά αδύνατη.

Το Proof of Work βασίζεται στο nonce του κάθε block. Όταν χτίζεται ένα block, καταβάλλεται προσπάθεια να υπολογιστεί ο αριθμός nonce, ο οποίος αν συνδυαστεί με τα υπόλοιπα δεδομένα του block, δηλαδη την πληροφορία των transactions, το timestamp και το hash του προηγούμενου block, θα παράξει το hash που τηρεί τις προδιαγραφές για να μπει το block στην αλυσίδα. Κάθε φορά που δοκιμάζεται ένα nonce που δεν παράγει το σωστό hash, απορρίπτεται και παράγεται το επόμενο μέχρι να βρεθεί το κατάλληλο (εικόνα 2).

Όταν στη διαδικασία συμμετέχουν πολλοί κόμβοι που προσπαθούν να βρουν το nonce, όπως συμβαίνει και με το bitcoin, ο κόμβος που θα βρει τη λύση επιβραβεύεται.



- Εικόνα 2 -

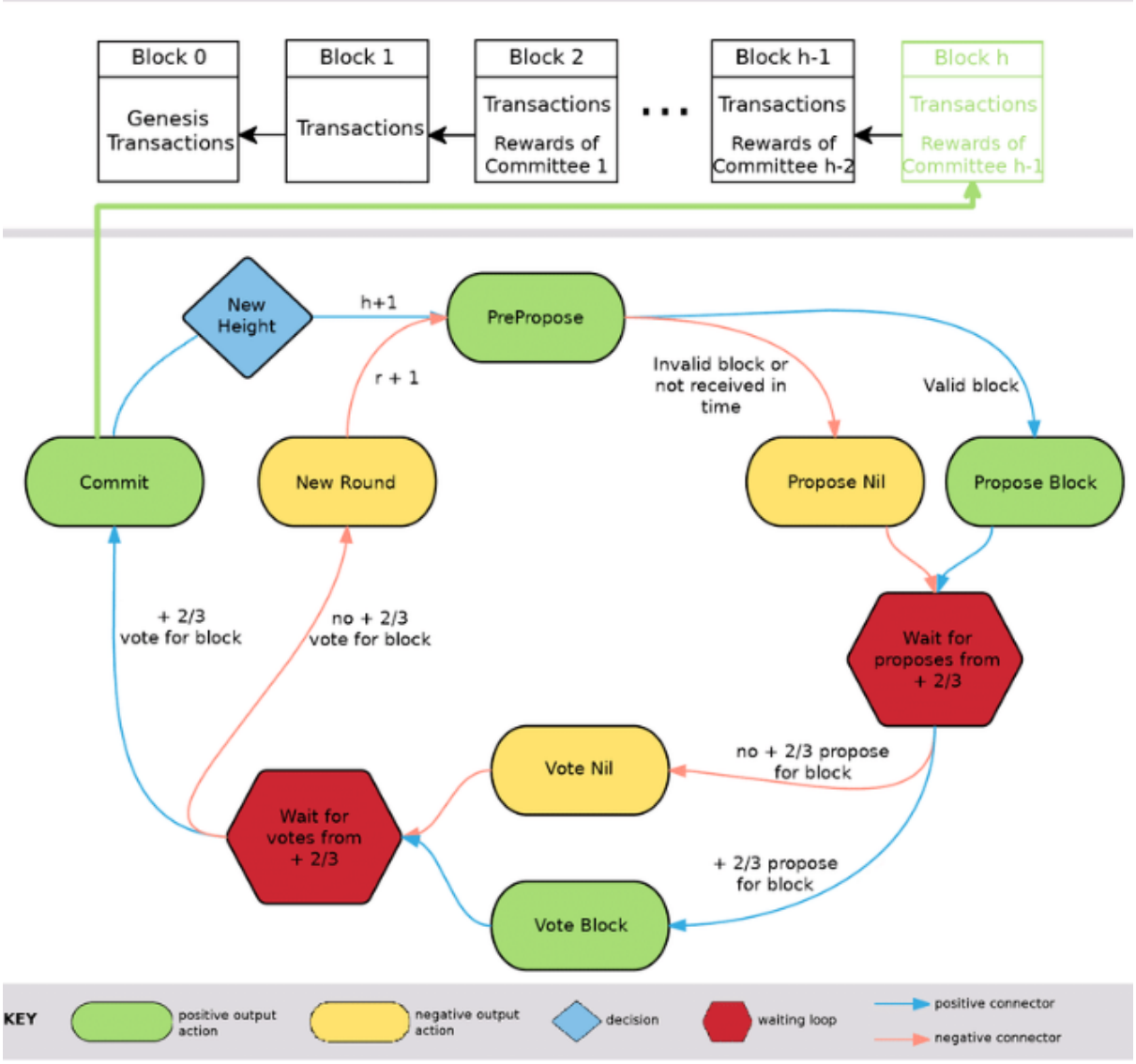
Proof of stake

Το Proof of Stake (PoS) είναι ένας άλλος μηχανισμός επαλήθευσης και προσθήκης του block στην αλυσίδα, ο οποίος χρησιμοποιείται σε δημόσια δίκτυα κρυπτονομισμάτων. Με τη συγκεκριμένη μέθοδο οι ιδιοκτήτες κρυπτονομισμάτων, επικυρώνουν τις συναλλαγές block με βάση των αριθμό των νομισμάτων που διαθέτει ο καθένας. Το proof of stake θεωρείται λιγότερο επικίνδυνο όσον αφορά την πιθανότητα επίθεσης στο δίκτυο, καθώς καθιστά μια επίθεση λιγότερο συμφέρουσα.

Το PoS μειώνει την υπολογιστική δύναμη που απαιτείται για να γίνει επαλήθευση των block και των transactions που πραγματοποιούνται. Οι ιδιοκτήτες κρυπτονομισμάτων, προσφέρουν τα κρυπτονομίσματά τους ως εγγύηση για την επικύρωση ενός block και γίνονται validators. Στη συνέχεια επιλέγονται τυχαία για να χτίσουν το επόμενο block. Αυτό το σύστημα επιλέγει τυχαία ποιος θα χτίσει το επόμενο block αντί να βασίζεται στον ανταγωνισμό σχετικά με το ποιος θα βρει πρώτος το nonce. Για να συμμετέχει στη διαδικασία κάποιος ως validator, πρέπει να διαθέσει μία συγκεκριμένη ποσότητα κρυπτονομισμάτων.

Για παράδειγμα το Ethereum, το οποίο είναι το 2ο πιο διαδεδομένο κρυπτονόμισμα μετά το Bitcoin, απαιτεί 32 eth για να μπορέσει κάποιος να συμμετέχει.

Τα blocks επικυρώνονται από περισσότερους από έναν validators και όταν ένα συγκεκριμένος αριθμός validators επαληθεύσει ότι το block είναι ακριβές, οριστικοποιείται και κλείνει (εικόνα 3). Τα δίκτυα που λειτουργούν με αυτό τον τρόπο χρησιμοποιούν διαφορετικούς μηχανισμούς επιβεβαίωσης των block.

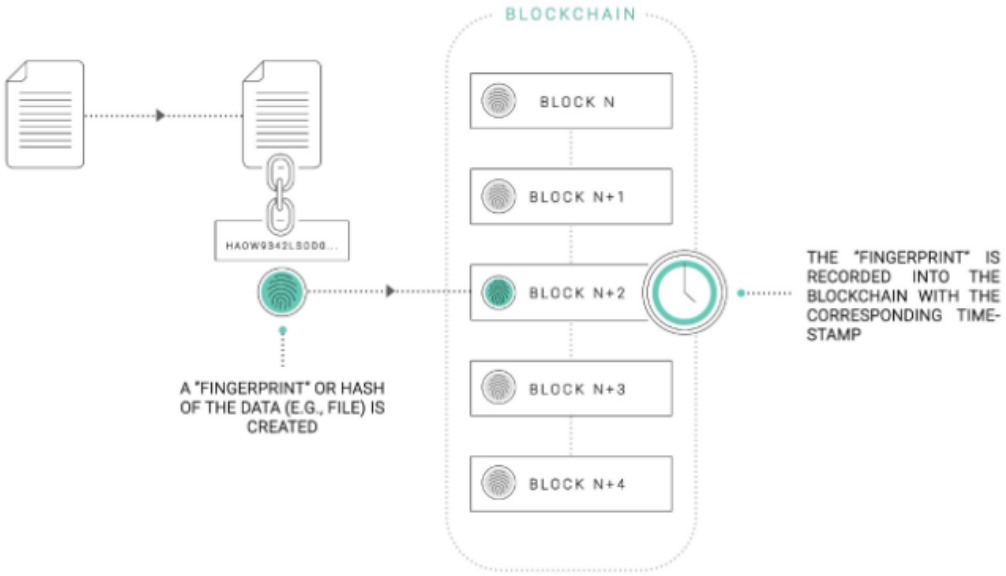


- Εικόνα 3 –

Proof of existence

Το Proof of Existence (PoE) δεν έχει να κάνει με την δημιουργία νέων block στην αλυσίδα αλλά με την επιβεβαίωση της πληροφορίας που τα ήδη υπάρχοντα blocks περιέχουν. Χρησιμοποιώντας το blockchain για επιβεβαίωση γνησιότητας εγγράφων όπως Εκπαιδευτικά Πιστοποιητικά, Άδειες Οδήγησης, Μητρώα Υγείας, Μητρώα Υπηρεσιών Υπαλλήλων, Πράξεις Πώλησης και Μητρώα Περιουσίας, Πιστοποιητικά γέννησης και θανάτου, φορολογικές δηλώσεις και ούτω καθεξής, η πληροφορία τους περιέχεται στο blockchain. Ωστόσο, υπάρχει το ενδεχόμενο να γίνει κατάχρηση αυτής της διευκόλυνσης δημιουργώντας και υποβάλλοντας πλαστά έγγραφα. Ο οργανισμός ή το άτομο που λαμβάνει τα έγγραφα, δεν είναι σε θέση να επαληθεύσει τη γνησιότητα και την ακεραιότητα των υποβληθέντων εγγράφων, καθιστώντα την επαλήθευση αυτών μια πρόκληση. Για την αντιμετώπιση αυτού του προβλήματος, η τεχνολογία blockchain μπορεί να αξιοποιηθεί. Η τεχνολογία blockchain παρέχει αυθεντικότητα εγγράφων, ιδιοκτησία, αμετάβλητο και στεγανότητα στα δεδομένα που καταγράφονται σε αυτό. Από αυτά τα ψηφιακά έγγραφα, τα εκπαιδευτικά πιστοποιητικά είναι μια από τις πιο σημαντικές οντότητες που πρέπει να διασφαλιστούν. Η C-DAC το θεώρησε ως μια σημαντική περίπτωση χρήσης και εφάρμοσε μια λύση που βασίζεται σε blockchain για αυτό. Ο οργανισμός που εκδίδει τα εκπαιδευτικά πιστοποιητικά θα πρέπει να εκδίδει πιστοποιητικά μέσω blockchain. Μόλις ένα πιστοποιητικό καταγραφεί στο blockchain, είναι μόνιμο. Οποιοσδήποτε μπορεί να το επαληθεύσει παρέχοντας το μοναδικό αναγνωριστικό του πιστοποιητικού, όπως αριθμό καταλόγου μαθητή, αναγνωριστικό συναλλαγής ή κρυπτογραφικό κατακερματισμό του πιστοποιητικού (hash). Αυτό διευκολύνει τη διαδικασία επαλήθευσης για τους παραλήπτες εγγράφων σχετικά με τη γνησιότητα και την ακεραιότητα του εγγράφου. Επίσης, εξαλείφει τον ρόλο τρίτων εταιρειών επαλήθευσης, γεγονός που εξοικονομεί κόστος και αυξάνει την αποτελεσματικότητα της επαλήθευσης.

Η διαδικασία αυτή, κατά την οποία γίνεται έλεγχος αν υπάρχει στο blockchain το έγγραφο που μας ενδιαφέρει, περιγράφει το Proof of Existence.


BLOCKCHAIN
BLOCK N
BLOCK N+1
BLOCK N+2
BLOCK N+3
BLOCK N+4
THE "FINGERPRINT" IS RECORDED INTO THE BLOCKCHAIN WITH THE CORRESPONDING TIME-STAMP
A "FINGERPRINT" OR HASH OF THE DATA (E.G., FILE) IS CREATED

- Εικόνα 4 –

Τεχνολογίες υλοποίησης

Η παρούσα εργασία αξιοποιεί την τεχνολογία του blockchain και χρησιμοποιεί Proof of Work δημιουργώντας ένα ασφαλές περιβάλλον για την αποθήκευση και πιστοποίηση της γνησιότητας των εγγράφων.

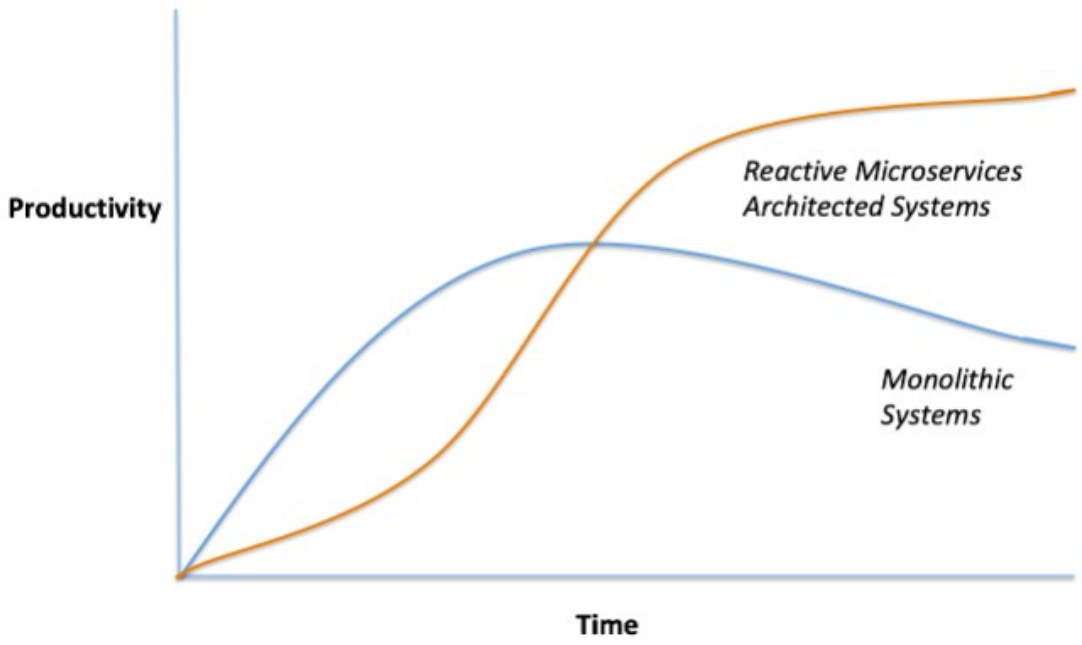
Γενικά

Η εφαρμογή αναπτύχθηκε σε αρχιτεκτονική reactive microservices με Spring Webflux. Οι συναρτήσεις όλων των services, επιστρέφουν αντικείμενα τύπου Mono και Flux, και αντίστοιχα οι controllers όλων των services επιστρέφουν αντικείμενα ίδιου τύπου, τα οποία είναι ασύγχρονα και επιστρέφουν τιμή όταν αυτή γίνει διαθέσιμη. Όλες οι κλήσεις μεταξύ των services είναι non-blocking, έτσι ώστε να μη δημιουργείται συμφόρηση και τελικά αποτυχία του συστήματος σε περίπτωση φόρτου.

Reactive Microservices

Η αρχιτεκτονική των microservices αφορά ανεξάρτητες εφαρμογές, όπου η κάθε μία αναφέρεται ως ένα ξεχωριστό service που αναλαμβάνει μία ξεχωριστή διεργασία και επικοινωνεί με τις υπόλοιπες ώστε να επιτευχθεί το business logic.

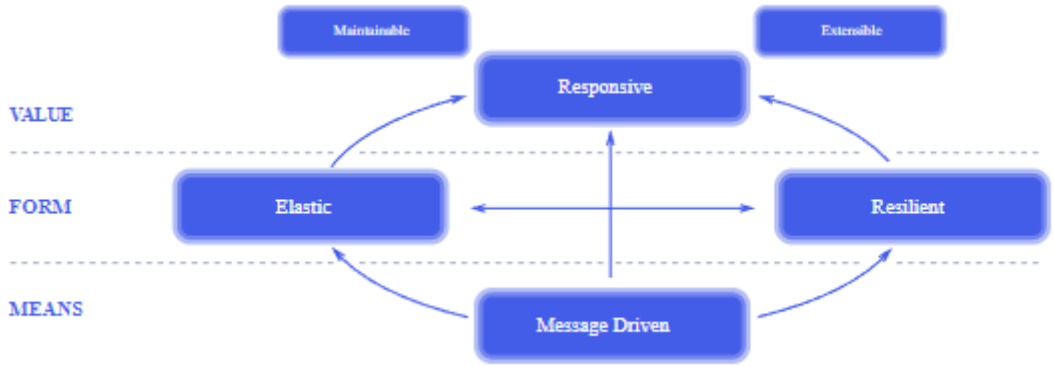
Η απόδοση της αρχιτεκτονικής των reactive microservices σε σχέση με τις μονολιθικές εφαρμογές φαίνεται στο σχήμα παρακάτω (εικόνα 5).



- Εικόνα 5 -

Τα μεγάλα συστήματα αποτελούνται από μικρότερα και, επομένως, εξαρτώνται από τις reactive ιδιότητες των επιμέρους services. Αυτό σημαίνει ότι τα reactive συστήματα εφαρμόζουν αρχές σχεδιασμού, ώστε αυτές οι ιδιότητες να εξασφαλίζονται σε όλα τα επίπεδα κλίμακας, καθιστώντας τα λειτουργικά. Τα μεγαλύτερα συστήματα στον κόσμο βασίζονται σε αρχιτεκτονικές που βασίζονται σε αυτές τις ιδιότητες ώστε να εξυπηρετούν καθημερινά τις ανάγκες δισεκατομμυρίων ανθρώπων.

Οι θεμελιώδεις ιδιότητες είναι αυτές της απόκρισης (Responsive) , ανθεκτικότητας (Resilient), ελαστικότητας (Elastic) και οδηγούμενη από μηνύματα (Message Driven) (εικόνα 6).



- Εικόνα 6 -

Responsive: Με τον όρο αυτό εννοούμε ότι η υπηρεσία θα πρέπει να ανταποκρίνεται έγκαιρα, όσο το δυνατόν γρηγορότερα, και να μην αφήνει ποτέ τους clients ή τις υπηρεσίες που με οποιονδήποτε τρόπο επικοινωνούν να κολλάνε. Η ανταπόκριση είναι ο ακρογωνιαίος λίθος της χρηστικότητας και της χρησιμότητας βοηθώντας στο γρήγορο εντοπισμό των προβλημάτων και την αποτελεσματική αντιμετώπισή τους. Τα συστήματα απόκρισης επικεντρώνονται στην παροχή γρήγορων και συνεπών χρόνων απόκρισης, χρησιμοποιώντας αξιόπιστα ανώτερα επιτρεπτά χρονικά όρια απόκρισης, ώστε να παρέχουν σταθερή ποιότητα υπηρεσιών καθώς μια αστοχία συστήματος μπορεί να προκαλέσει αλυσιδωτή αντίδραση αστοχιών. Αυτή η συνεπής συμπεριφορά με τη σειρά της απλοποιεί τον χειρισμό σφαλμάτων, δημιουργεί εμπιστοσύνη στον τελικό χρήστη και ενθαρρύνει την περαιτέρω αλληλεπίδραση.

Resilient: Το σύστημα ανταποκρίνεται σε περίπτωση αποτυχίας. Αυτό ισχύει για όλα τα services - καθώς κάθε service που δεν είναι resilient δεν θα ανταποκρίνεται μετά από αποτυχία. Η ιδιότητα αυτή επιτυγχάνεται με την δημιουργία αντιγράφων και τις διαδικασίες

του περιορισμού, της απομόνωσης και της ανάθεσης. Οι βλάβες περιέχονται σε κάθε component. Απομονώνοντας κάθε component από το άλλο, διασφαλίζουμε ότι τμήματα του συστήματος μπορούν να αποτύχουν και να ανακάμψουν χωρίς να διακυβεύεται το συνολικό σύστημα. Η ανάκτηση κάθε component ανατίθεται σε ένα άλλο (εξωτερικό) component και η συνεχής διαθεσιμότητα εξασφαλίζεται με δημιουργία αντιγράφων, όπου είναι απαραίτητο. Έτσι ο client δεν επιβαρύνεται με τον χειρισμό των αστοχιών του συστήματος, αφού οι υπηρεσίες παραμένουν ενεργές ακόμα και σε περίπτωση αποτυχίας. Το σύστημα μπορεί να ανταποκριθεί επειδή μπορεί να εντοπίσει ότι ένα ασύγχρονο response δεν επιστρέφει στο χρόνο που έχει οριστεί και να στείλει το κατάλληλο response (circuit breaker)

Elastic: Το σύστημα ανταποκρίνεται σε μεγάλο φόρτο αιτημάτων. Τα reactive συστήματα μπορούν να αντιδράσουν σε αλλαγές στον ρυθμό αποστολής αιτημάτων, αυξάνοντας ή μειώνοντας τους πόρους που διατίθενται για την εξυπηρέτηση αυτών των αιτημάτων. Αυτό συνεπάγεται σχεδιασμό που δεν έχει κεντρικά σημεία συμφόρησης, με αποτέλεσμα τη δυνατότητα τεμαχισμού ή δημιουργίας αντιγράφων των components και διαμοιρασμό των αιτημάτων μεταξύ τους. Τα reactive συστήματα υποστηρίζουν προγνωστικούς αλγόριθμους κλιμάκωσης και παρέχουν κατάλληλες μετρήσεις ζωντανής απόδοσης, επιτυγχάνοντας την ελαστικότητα με οικονομικά αποδοτικό τρόπο.

Message Driven: Τα reactive συστήματα βασίζονται στην ασύγχρονη μετάδοση μηνυμάτων για να δημιουργήσουν ένα όριο μεταξύ των components που διασφαλίζει χαλαρή σύζευξη και απομόνωση. Αυτό το όριο παρέχει επίσης το μέσο για την ανάθεση αστοχιών ως μηνύματα. Η χρήση σαφούς διέλευσης μηνυμάτων επιτρέπει τη διαχείριση φορτίου, την ελαστικότητα και τον έλεγχο της ροής διαμορφώνοντας και παρακολουθώντας τις ουρές μηνυμάτων στο σύστημα και εφαρμόζοντας μεθόδους επαναφοράς όταν είναι απαραίτητο. Τα μηνύματα ως μέσο επικοινωνίας καθιστούν δυνατή τη διαχείριση της αποτυχίας σε ένα σύμπλεγμα ή σε έναν μόνο κεντρικό υπολογιστή. Η επικοινωνία χωρίς αποκλεισμό επιτρέπει στους παραλήπτες να καταναλώνουν πόρους μόνο ενώ είναι ενεργοί, οδηγώντας σε μικρότερη επιβάρυνση του συστήματος.

Όταν σχεδιάζονται με τις παραπάνω αρχές (reactive principles), τα microservices έχουν τα ακόλουθα χαρακτηριστικά:

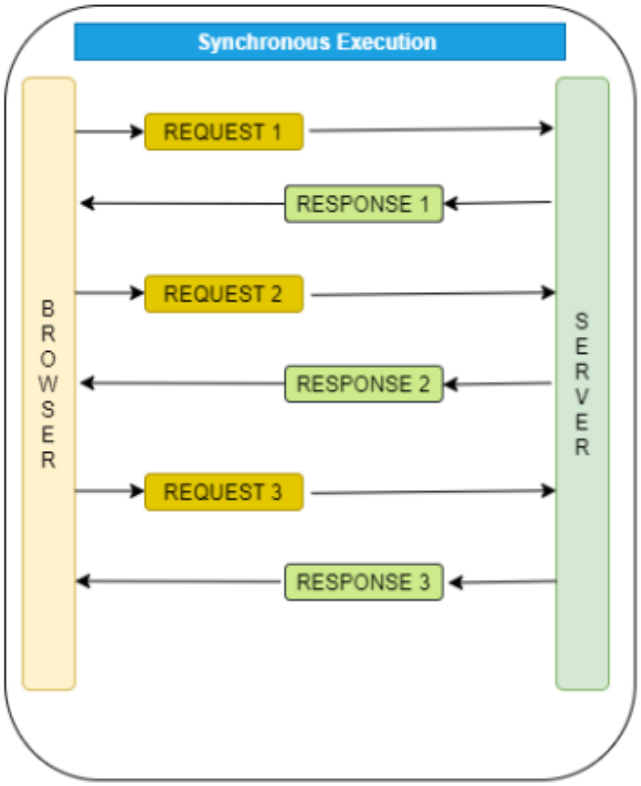
- Απομόνωση, όπου κάθε service είναι πλήρως αποσυνδεδεμένο από την υπόλοιπη εφαρμογή, έτσι ώστε ο κύκλος ζωής του, συμπεριλαμβανομένης της αποτυχίας, να μην έχει αντίκτυπο σε άλλα microservices του συστήματος.
- Αυτονομία, όπου κάθε service ενεργεί και λαμβάνει αποφάσεις ανεξάρτητα από τα υπόλοιπα, και επιστρέφει αποτελέσματα μέσω ενός API.
- Ενιαία ευθύνη, όπου κάθε service επιτελεί ένα ρόλο.

- Event driven, όπου κάθε service παράγει, καταναλώνει και αντιδρά σε γεγονότα.
- Κινητικότητα (και δυνατότητα διεύθυνσης), όπου ένα service μπορεί να μετακινηθεί κατά το χρόνο εκτέλεσης, αλλά είναι διαθέσιμο με τον ίδιο τρόπο, ανεξάρτητα από τη θέση του.
- Διαχειρίζονται αποκλειστικά και μόνο τη δική τους κατάσταση.

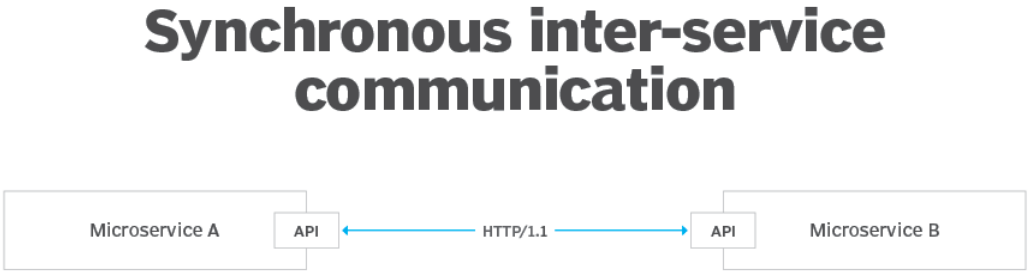
Επικοινωνία των microservices

Σύγχρονη (blocking)

Σε μια σύγχρονη (blocking) επικοινωνία, το service που στέλνει ένα request, περιμένει μια απάντηση (response) πριν στείλει το επόμενο μήνυμα. Η επικοινωνία πραγματοποείται με τη χρήση της αρχιτεκτονικής REST. Κάθε request, δεσμεύει ένα thread στο service στο οποίο έχει στείλει το request, και το thread αυτό αναλαμβάνει να στείλει το response. Αν υπάρχουν πολλά requests ταυτόχρονα στο ίδιο service, ή αν το τελευταίο αποτύχει να ολοκληρώσει ένα request, πιθανόν το σύστημα να μην μπορεί να εξυπηρετήσει άλλα requests. Αυτό σημαίνει ότι όλα τα υπόλοιπα services, πρέπει να περιμένουν να ολοκληρωθεί το ένα αίτημα για να αποδεσμευτεί ένα thread, με αποτέλεσμα να υπάρχει καθυστέρηση σε όλο το σύστημα (εικόνες 7 & 8).



- Εικόνα 7 -



- Εικόνα 8 -

Ένα σύστημα, με μία σειρά από services, όπου το ένα εξαρτάται από το άλλο, λειτουργεί καλά εφόσον λειτουργούν όλες οι υπηρεσίες. Όταν ένα service αποτύχει, μπορεί να προκαλέσει μία αλυσίδα αποτυχιών που θα οδηγήσουν σε αδυναμία εξυπηρέτησης του τελικού χρήστη. Ομοίως, εάν ένα service είναι αργό, τότε ο χρόνος απόκρισης του συστήματος μεγαλώνει και η εξυπηρέτηση του τελικού χρήστη επιβραδύνεται.

Εφαρμόζοντας σύγχρονη επικοινωνία μεταξύ των microservices, είναι ουσιαστικά σα να έχει φτιαχτεί μία μονολιθική εφαρμογή, οπότε ακυρώνονται τα οφέλη αυτής της αρχιτεκτονικής.

Ασύγχρονη (non-blocking)

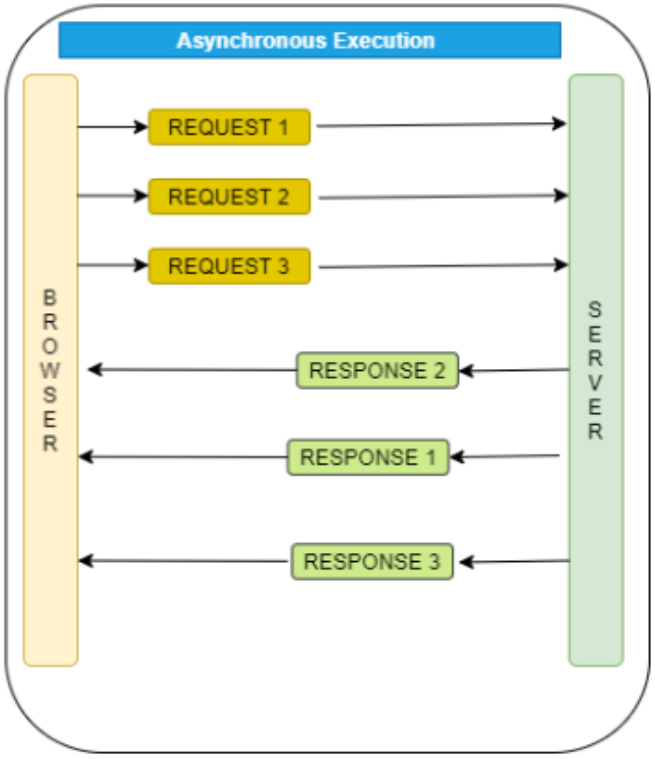
Σε μια ασύγχρονη (non-blocking) επικοινωνία τα μηνύματα αποστέλλονται χωρίς να περιμένουν απάντηση (εικόνα 9). Αυτό είναι κατάλληλο για κατανεμημένα συστήματα και συνήθως απαιτεί έναν message broker για τη διαχείριση των μηνυμάτων (εικόνα 10).

Ένα service που στέλνει ένα request, περιμένει από κάποιο άλλο ένα response αλλά χωρίς να δεσμεύει κάποιο thread.

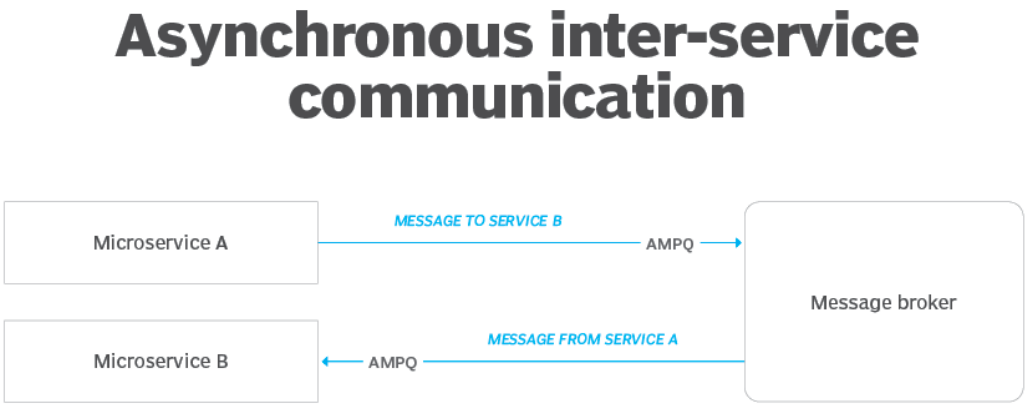
Η ασύγχρονη επικοινωνία είναι εξ ορισμού μη αποκλειστική και υποστηρίζει καλύτερη κλιμάκωση, αφού χρησιμοποιώντας ένα μοντέλο publisher/subscriber, πολλά microservices μπορούν να εγγραφούν για να λαμβάνουν ενημερώσεις από events. Επιπλέον, σε περίπτωση που κάποιο service παρουσιάσει πρόβλημα και σταματήσει να λειτουργεί, οι μηχανισμοί ασύγχρονης επικοινωνίας παρέχουν διάφορες τεχνικές ανάκτησης και είναι γενικά καλύτεροι στον χειρισμό σφαλμάτων. Εάν για κάποιο λόγο το service που πρέπει να λάβει το μήνυμα τεθεί εκτός λειτουργίας ή για οποιονδήποτε άλλο λόγο δεν είναι προσβάσιμο, τα υπόλοιπα microservices μπορούν να στέλνουν μηνύματα είτε μέσω του message broker, είτε απευθείας, τα οποία θα ληφθούν όταν το service επαναλειτουργήσει. Επίσης, όταν χρησιμοποιούνται message brokers αντί για πρωτόκολλο REST, τα microservices δεν χρειάζεται πραγματικά να γνωρίζονται μεταξύ τους. Ένα νέο service μπορεί να προστεθεί

ακόμη και αφού ένα παλιό λειτουργεί για μεγάλο χρονικό διάστημα. Έτσι έχουμε μεγαλύτερη ανεξαρτησία μεταξύ των microservices.

Όταν εφαρμόζεται ασύγχρονη επικοινωνία μεταξύ των microservices, είναι σύνηθες να χρησιμοποιείται ένας message broker. Ο message broker διασφαλίζει την σταθερή και αξιόπιστη επικοινωνία μεταξύ διαφορετικών microservices, ενώ ο διαχειριστής του συστήματος μπορεί να ελέγξει μέσω αυτού την κατάσταση των μηνυμάτων και να τα διαχειριστεί σε περίπτωση που χρειαστεί να επέμβει. Τα πιο γνωστά message brokers είναι RabbitMQ, Kafka και Redis.



- Εικόνα 9 -



- Εικόνα 10 -

Ο τύπος επικοινωνίας που θα επιλεχθεί σε κάθε σύστημα, πρέπει να λαμβάνει υπόψη διαφορετικές παραμέτρους, όπως τον τρόπο δομής των microservices, την υποδομή που είναι διαθέσιμη, την καθυστέρηση, την κλίμακα, τις εξαρτήσεις και τον σκοπό της επικοινωνίας. Η δημιουργία της ασύγχρονης επικοινωνίας μπορεί να είναι πιο περίπλοκη και απαιτεί την προσθήκη περισσότερων στοιχείων στο project, αλλά τα πλεονεκτήματα της υπερτερούν των μειονεκτημάτων.

Angular

Η Angular είναι ένα framework, το οποίο χρησιμοποιείται για client εφαρμογές με χρήση HTML και Typescript. Τα βασικά δομικά στοιχεία της είναι τα components τα οποία είναι οργανωμένα σε modules. Τα modules συγκεντρώνουν συναφή κώδικα σε λειτουργικά σύνολα. Μια εφαρμογή έχει πάντα τουλάχιστον ένα root module που επιτρέπει το bootstrapping, και τυπικά έχει πολλά περισσότερα feature modules (εικόνα 11).

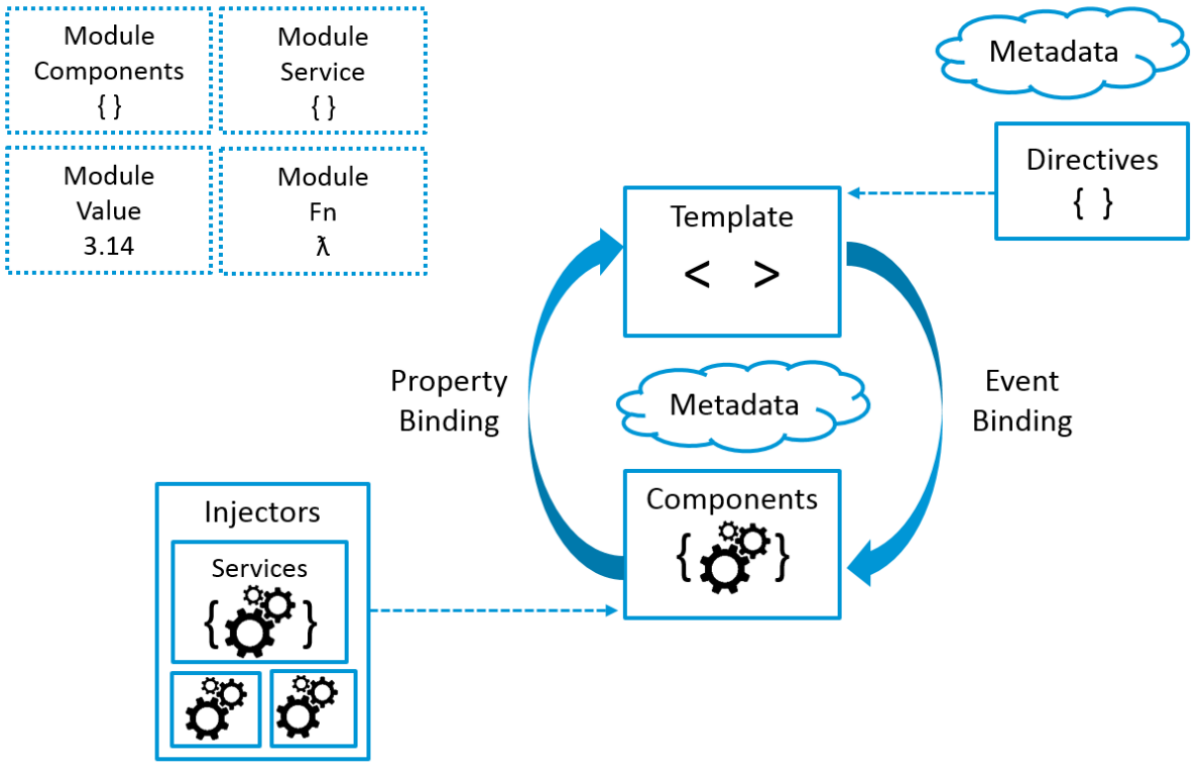
Τα components προσδιορίζουν views, τα οποία είναι υποσύνολα της κάθε σελίδας που εμφανίζεται στο χρήστη, και μπορούν να τροποποιηθούν σύμφωνα με τη λογική και τα δεδομένα του προγράμματος. Τα components χρησιμοποιούν services, τα οποία παρέχουν συγκεκριμένες λειτουργίες, που δεν σχετίζονται άμεσα με τα views. Τα services μπορούν να γίνουν inject στα components σαν dependencies καθιστώντας τον κώδικα αρθρωτό, επαναχρησιμοποιήσιμο και αποδοτικό (modular - reusable - efficient).

Τα components, modules και services είναι κλάσεις που χρησιμοποιούν decorators. Αυτοί οι decorators αποδίδουν έναν τύπο, και παρέχουν metadata που λένε στην Angular πώς να τα χρησιμοποιεί.

Τα metadata σε ένα service, δίνουν την πληροφορία που χρειάζεται η Angular για να γίνουν διαθέσιμα μέσω του dependency injection (DI).

Τα components μίας εφαρμογής, συνήθως προσδιορίζουν πολλά views, οργανωμένα ιεραρχικά. Η Angular διαθέτει router service, το οποίο παρέχει πλοήγηση μεταξύ των views.

Επιπλέον η Angular χρησιμοποιεί observables σαν interface για τον χειρισμό των ασύγχρονων λειτουργιών. Προσαρμοσμένα events στέλνουν δεδομένα τύπου observable τα οποία διαχειρίζονται όπως χρειάζεται κάθε φορά.

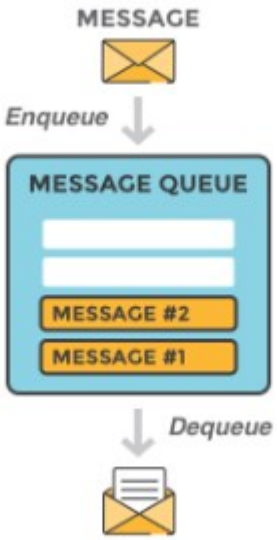


- Εικόνα 11 - Αρχιτεκτονική Angular –

Message broker

Rabbitmq

Το RabbitMQ είναι ένα message broker, δηλαδή ένα πρόγραμμα το οποίο διαχειρίζεται μηνύματα που στέλνονται μεταξύ των services. Τα μηνύματα αυτά μπορεί να περιέχουν πληροφορία σχετικά με μία διαδικασία, ή μια λειτουργία που πρέπει να γίνει από διαφορετικό service από αυτό το οποίο το πυροδότησε. Επίσης μπορεί ο αποδέκτης του μηνύματος αυτού να είναι σε άλλο server ή να πρόκειται για ένα απλό μήνυμα κειμένου. Το RabbitMQ αποθηκεύει τα μηνύματα που στέλνονται από ένα service σε μία λίστα (ουρά), μέχρι να συνδεθεί κάποιο άλλο service το οποίο κάνει register σαν listener, δηλαδή περιμένει να δεχτεί τα κατάλληλα μηνύματα. Μόλις το RabbitMQ αναγνωρίσει ότι υπάρχει service το οποίο περιμένει να δεχτεί τα μηνύματα που έχει αποθηκεύσει, τα προωθεί (εικόνα 12).



- Εικόνα 12 -

Η δομή μιας ουράς μηνυμάτων (message queue) είναι απλή. Η εφαρμογή που στέλνει το μήνυμα λέγεται producer, ενώ όσες εφαρμογές δέχονται τα μηνύματα μέσω του broker, λέγονται consumers (εικόνα 13). Ένα service μπορεί να είναι ταυτόχρονα και producer αλλά και consumer, αφού μπορεί να στέλνει μηνύματα ενώ συγχρόνως να δέχεται από άλλες εφαρμογές. Όλα τα μηνύματα μένουν στη μνήμη του broker μέχρι να καταναλωθούν από τους consumers.

- Εικόνα 13 -

Περιπτώσεις χρήσης

Η ουρά μηνυμάτων επιτρέπει στους διακομιστές να ανταποκρίνονται γρήγορα σε αιτήματα αντί να αναγκάζονται να εκτελούν επί τόπου διαδικασίες με μεγάλους πόρους που μπορεί να καθυστερήσουν τον χρόνο απόκρισης. Η ουρά μηνυμάτων είναι επίσης καλή όταν πρέπει να σταλεί ένα μήνυμα σε πολλούς καταναλωτές ή να εξισορροπηθούν τα φορτία μεταξύ των modules.

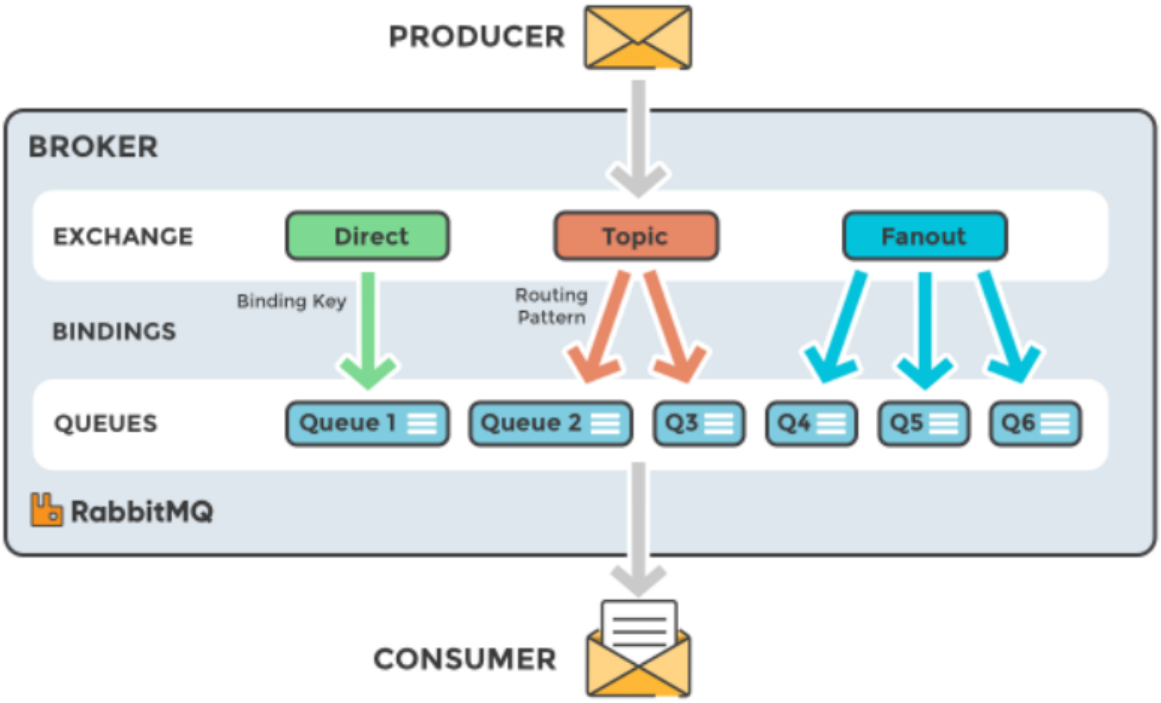
Ο consumer βγάζει ένα μήνυμα από την ουρά και ξεκινά την επεξεργασία του. Την ίδια ώρα, ο producer στέλνει νέα μηνύματα, τα οποία προστίθενται στην ουρά. Ο consumer μπορεί να βρίσκεται σε έναν εντελώς διαφορετικό διακομιστή από τον producer, ή μπορεί να βρίσκεται στον ίδιο διακομιστή. Το αίτημα μπορεί να δημιουργηθεί σε μία γλώσσα προγραμματισμού και να διεκπεραιωθεί σε άλλη γλώσσα προγραμματισμού. Το θέμα είναι ότι οι δύο εφαρμογές θα επικοινωνούν μόνο μέσω των μηνυμάτων που στέλνουν η μία στην άλλη, πράγμα που σημαίνει ότι ο αποστολέας και ο παραλήπτης έχουν χαλαρή σύζευξη.

Exchanges

Τα μηνύματα δεν δημοσιεύονται απευθείας σε μια ουρά. Αντίθετα, ο producer στέλνει μηνύματα σε ένα exchange. Ένα exchange είναι υπεύθυνο για τη δρομολόγηση των μηνυμάτων σε διαφορετικές ουρές με τη βοήθεια κλειδιών δρομολόγησης. Τα κλειδιά δρομολόγησης είναι ένας σύνδεσμος μεταξύ μιας ουράς και ενός exchange (εικόνα 14).

Υπάρχουν διαφορετικά είδη exchanges :

- **Direct**: Τα μηνύματα δρομολογούνται σε ουρές των οποίων τα κλειδιά δρομολόγησης ταυτίζονται με τα κλειδιά δρομολόγησης των μηνυμάτων. Για παράδειγμα αν το κλειδί δρομολόγησης της ουράς είναι “pdfProcess”, ένα μήνυμα που θα σταλεί με αυτό το κλειδί θα προστεθεί σε αυτή την ουρά και θα δρομολογηθεί από αυτή.
- **Fanout**: Το fanout exchange δρομολογεί τα μηνύματα σε όλες τις ουρές που έχουν εγγραφεί σε αυτό.
- **Topic**: Το topic exchange συνδυάζει τα κλειδιά δρομολόγησης και του routing pattern.
- **Headers**: Τα headers exchanges χρησιμοποιούν τα header attributes για τη δρομολόγηση των μηνυμάτων.



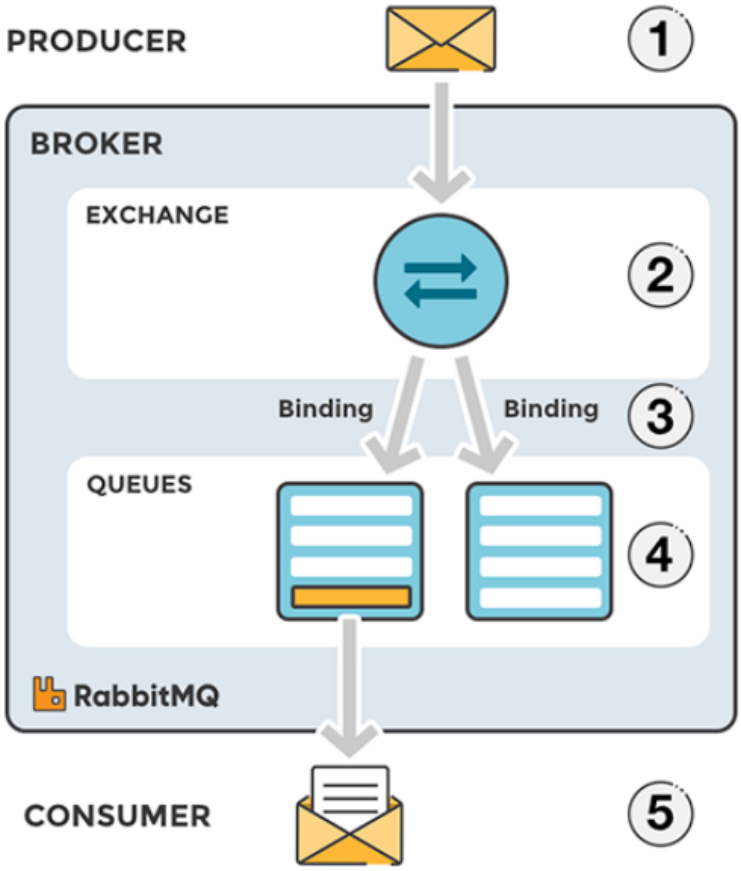
- Εικόνα 14 -

Ροή μηνυμάτων

Τα βήματα που ακολουθούνται για την αποστολή ενός μηνύματος από μία εφαρμογή σε μία άλλη είναι τα ακόλουθα:

1. Ο producer δημοσιεύει ένα μήνυμα στο exchange. Κατά την αποστολή καθορίζεται ο τύπος του μηνύματος.
2. Το exchange λαμβάνει το μήνυμα και πλέον είναι υπεύθυνο να το δρομολογήσει, λαμβάνοντας υπόψη παραμέτρους όπως το κλειδί δρομολόγησης, και τον τύπο του ανταλλακτηρίου.
3. Δημιουργούνται συσχετίσεις από το exchange στις ουρές και δρομολογούνται τα μηνύματα ανάλογα με τις παραμέτρους του μηνύματος.
4. Το ανταλλακτήριο δρομολογεί το μήνυμα στις ουρές, ανάλογα με τα χαρακτηριστικά του μηνύματος.
5. Τα μηνύματα παραμένουν στην ουρά μέχρι να τα χειριστεί ένας consumer.
6. Ο consumer καταναλώνει και διαχειρίζεται το μήνυμα.

Η εικόνα 15 αποτυπώνει τα παραπάνω βήματα.



- Εικόνα 15 –

Reactive database

Reactive MongoDB

Η MongoDB είναι μία βάση δεδομένων γενικής χρήσης, η οποία διατηρεί τα καλύτερα χαρακτηριστικά των σχεσιακών και των NoSQL βάσεων δεδομένων. Αντί για πίνακες σχεσιακών βάσεων δεδομένων, χρησιμοποιεί έγγραφα (documents). Αντί της αποθήκευσης δεδομένων σε στήλες και γραμμές, οι βάσεις δεδομένων εγγράφων μπορούν να αποθηκεύουν δεδομένα ως JSON (JavaScript Object Notation). Η βάση μπορεί να αποθηκεύσει οποιονδήποτε τύπο δεδομένων και η δομή των εγγράφων μπορεί να τροποποιηθεί εύκολα. Μπορούν να προστεθούν νέα πεδία χωρίς να επηρεαστούν άλλα έγγραφα, ενώ μπορούν να μοντελοποιηθούν δεδομένα σε οποιαδήποτε τρόπο που ταιριάζει στην εφαρμογή, για παράδειγμα, ως ζεύγη κλειδιών-τιμών, ως ακμές ή κόμβοι ενός γράφου, ή ως εμφωλευμένες δομές που αναπαριστούν σχέσεις.

GridFs

Μία από τις κύριες αρχές σχεδιασμού της ReactiveMongo είναι ότι όλα τα έγγραφα μπορούν να σταλούν και να ληφθούν χρησιμοποιώντας το Producer/Consumer pattern γνωστό ως Enumerator/Iteratee pattern.

Ένας iteratee είναι consumer δεδομένων, επεξεργάζεται κομμάτια δεδομένων. Ένας iteratee μπορεί να βρεθεί σε μία από τις εξής καταστάσεις:

α) ολοκληρωμένος (που σημαίνει ότι ενδέχεται να μην επεξεργάζεται περισσότερα κομμάτια),

β) συνέχεια (δέχεται περισσότερα δεδομένα),

γ) error (παρουσιάστηκε σφάλμα).

Μετά από κάθε βήμα, επιστρέφει ένα νέο instance του iteratee που μπορεί να βρίσκεται σε διαφορετική κατάσταση. Ένας enumerator είναι πηγή δεδομένων το οποίο όταν εφαρμόζεται σε ένα iteratee, θα το τροφοδοτήσει ανάλογα με την κατάστασή του.

Σε ένα παράδειγμα ανάκτησης πολλών εγγράφων, η δημιουργία μίας τεράστιας λίστας εγγράφων μπορεί να γεμίσει πολύ γρήγορα την προσωρινή μνήμη του συστήματος, επομένως δεν είναι καλή επιλογή. Χρησιμοποιώντας ένα blocking iterator μπορεί να δημιουργήσουμε καθυστερήσεις ενώ ένας non-blocking ασυγχρονος iterator είναι δύσκολα διαχειρίσιμος. Εκεί εφαρμόζονται οι Iteratees και οι Enumerators. Ουσιαστικά η βάση (MongoDB) είναι ο producer (enumerator) ενώ ο κώδικας - εφαρμογή ο consumer (iteratee).

Το ίδιο συμβαίνει και και για το GridFS. Κατά την αποθήκευση ενός αρχείου, το GridFS είναι ο consumer (iterate) ενώ κατά την ανάκτηση είναι ο producer (enumerator)

Χρησιμοποιώντας iteratees και enumerators, μπορούμε να κάνουμε συνεχή ροή αρχείων, από τον client στον server και τη βάση δεδομένων, διατηρώντας χαμηλή χρήση μνήμης και με τρόπο non-blocking.

Το GridFS χρησιμοποιείται στη ReactiveMongoDb για αποθήκευση αρχείων που είναι μεγαλύτερα από 16mb (εικόνες, βίντεο κλπ). Το GridFS χρησιμοποιεί δύο διαφορετικά collections, τα files και τα chunks. Τα metadata του αρχείου αποθηκεύονται στα files, ενώ το περιεχόμενο χωρίζεται σε chunks των 256kB. Η ανάκτηση των αρχείων γίνεται εκτελώντας αναζητήσεις στα files, και στη συνέχεια ενώνοντας τα chunks που αποτελούν το αρχείο.

Σε ορισμένες περιπτώσεις, η αποθήκευση μεγάλων αρχείων μπορεί να είναι πιο αποτελεσματική σε μια βάση δεδομένων MongoDB παρά σε ένα σύστημα αρχείων σε επίπεδο συστήματος. Εάν το σύστημα αρχείων περιορίζει τον αριθμό των αρχείων που μπορούν να αποθηκευτούν, το GridFS βοηθάει να αποθηκευτούν όσα αρχεία χρειάζονται. Όταν χρειάζεται πρόσβαση σε πληροφορίες από τμήματα μεγάλων αρχείων χωρίς να χρειάζεται να φορτωθούν ολόκληρα αρχεία στη μνήμη, το GridFS μπορεί να ανακαλέσει τμήματα αρχείων χωρίς να διαβάσει ολόκληρο το αρχείο στη μνήμη. Αν πρέπει να διατηρηθούν τα αρχεία και τα metadata αυτόματα συγχρονισμένα και αναπτυγμένα σε διάφορα συστήματα και εγκαταστάσεις, μπορεί να χρησιμοποιηθεί το GridFS. Όταν

χρησιμοποιούνται γεωγραφικά κατανεμημένα σύνολα αντιγράφων, η MongoDB μπορεί να διανείμει αρχεία και τα metadata τους αυτόματα σε μια σειρά από παρουσίες και εγκαταστάσεις mongodb.

Το GridFS δεν είναι κατάλληλο εάν χρειάζεται να ενημερωθεί το περιεχόμενο ολόκληρου του αρχείου ατομικά. Εναλλακτικά, μπορούν να αποθηκευτούν πολλές εκδόσεις κάθε αρχείου και να καθοριστεί η τρέχουσα έκδοση του αρχείου στα metadata. Μπορεί να ενημερωθεί το πεδίο metadata που υποδεικνύει την κατάσταση "πιο πρόσφατη" σε μια ατομική ενημέρωση μετά τη μεταφόρτωση της νέας έκδοσης του αρχείου και αργότερα να αφαιρεθεί από τις προηγούμενες εκδόσεις, εάν χρειάζεται.

Στην περίπτωση αποθήκευσης εγγράφων η ιδιότητά του αυτή μας είναι πολύ χρήσιμη, καθώς δεν θέλουμε να είναι εφικτή η τροποποίηση ενός εγγράφου που έχει ήδη αποθηκευτεί.

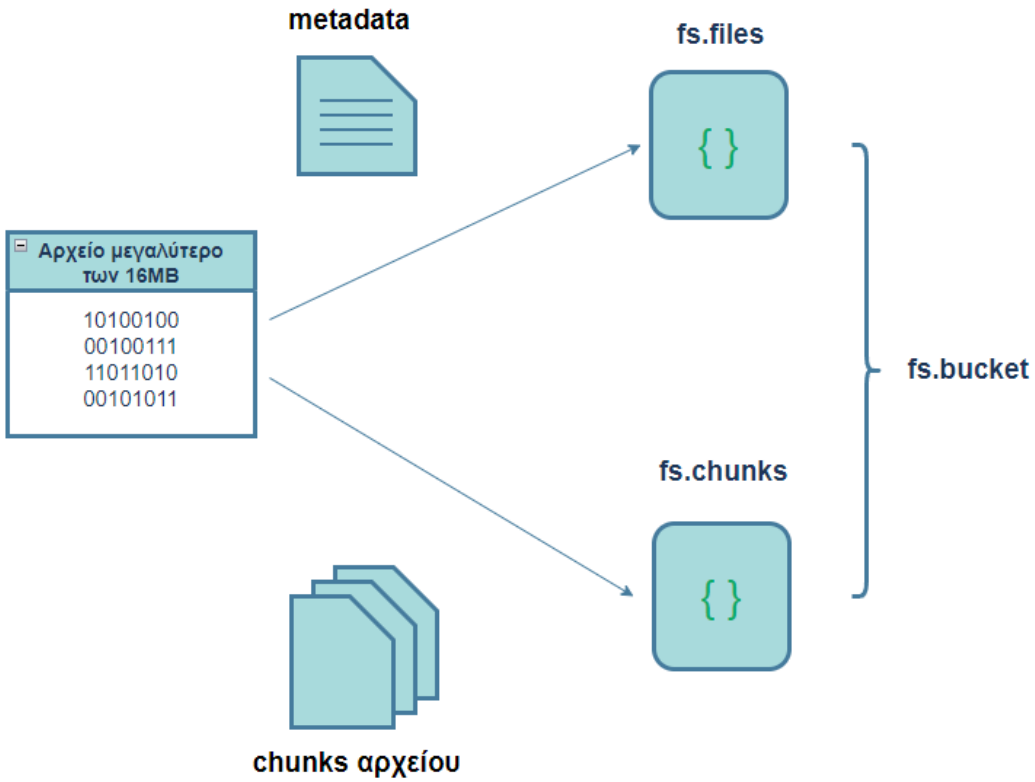
Πώς λειτουργεί το GridFS:

Το GridFS οργανώνει αρχεία σε έναν κάδο (bucket), μια ομάδα συλλογών MongoDB που περιέχει τα chunks των αρχείων και τις πληροφορίες που τα περιγράφουν. Το bucket περιέχει τις ακόλουθες συλλογές, που ονομάζονται χρησιμοποιώντας τη σύμβαση που ορίζεται στην προδιαγραφή GridFS:

- Η συλλογή των **chunks** αποθηκεύει τα κομμάτια δυαδικού αρχείου.
- Η συλλογή **files** αποθηκεύει τα metadata του αρχείου.

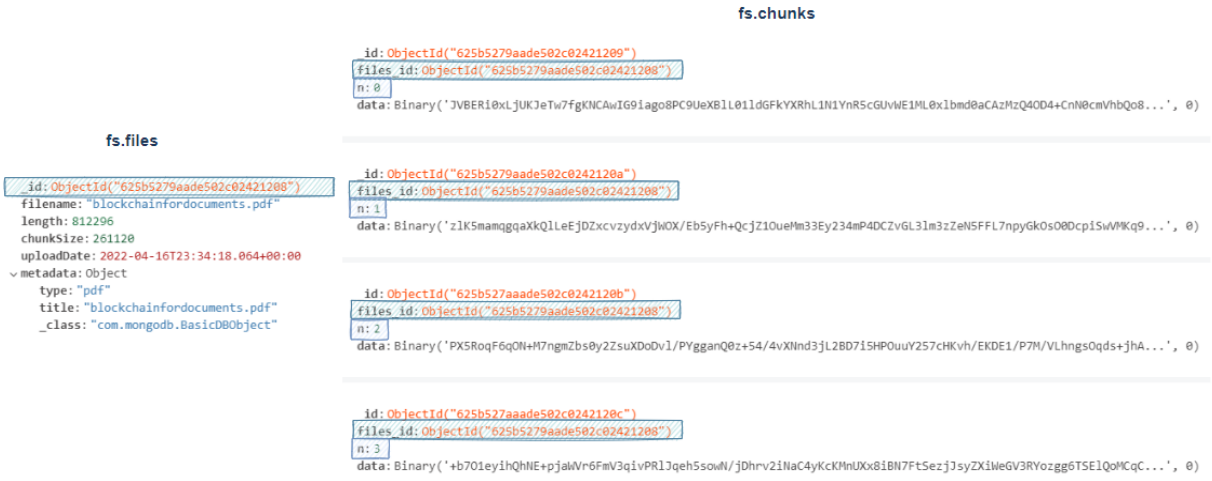
Όταν δημιουργείται ένα νέο GridFS bucket, το πρόγραμμα οδήγησης δημιουργεί τις προηγούμενες συλλογές, με πρόθεμα το προεπιλεγμένο όνομα κάδου «fs», εκτός εάν καθοριστεί διαφορετικό όνομα. Το πρόγραμμα οδήγησης δημιουργεί επίσης ένα ευρετήριο σε κάθε συλλογή για να διασφαλίσει τη γρήγορη ανάκτηση των αρχείων και των σχετικών metadata. Το πρόγραμμα οδήγησης δημιουργεί το bucket του GridFS στην πρώτη λειτουργία εγγραφής μόνο εάν δεν υπάρχει ήδη και δημιουργεί ευρετήρια μόνο εάν δεν υπάρχουν και όταν ο κάδος είναι άδειος.

Κατά την αποθήκευση αρχείων με το GridFS, το πρόγραμμα οδήγησης χωρίζει τα αρχεία σε μικρότερα κομμάτια, καθένα από τα οποία αντιστοιχεί σε ένα ξεχωριστό έγγραφο στη συλλογή chunks. Δημιουργεί επίσης ένα έγγραφο στη συλλογή files που περιέχει ένα αναγνωριστικό αρχείου (file id), όνομα αρχείου (file name) και άλλα metadata αρχείου. Το αρχείο μπορεί να ανέβει από τη μνήμη ή από stream. Το παρακάτω διάγραμμα (εικόνα 16), αποτυπώνει πώς το GridFS χωρίζει τα αρχεία κατά τη μεταφόρτωσή τους σε έναν κάδο.



- Εικόνα 16 -

Τα chunks του κάθε αρχείου περιέχουν το αναγνωριστικό id του file ώστε να υπάρχει η συσχέτιση για την ανάκτηση του αρχείου αλλά και τη σειρά με την οποία πρέπει να ανακτηθούν. Παρακάτω βλέπουμε πώς αποθηκεύεται ένα αρχείο στη MongoDB (εικόνα 17).



- Εικόνα 17 -

OCR

Η τεχνολογία OCR (Optical Character Recognition), γνωστή και ως αναγνώριση κειμένου, αναγνωρίζει κείμενο από φωτογραφίες ή σκαναρισμένα αρχεία και εικόνες. Η τεχνολογία OCR απομονώνει κάθε γράμμα ξεχωριστά, τα ανασυντάσσει σε λέξεις και στη συνέχεια σχηματίζει προτάσεις, εξαλείφοντας την ανάγκη για παρέμβαση του χρήστη.

Τα συστήματα OCR χρησιμοποιούν ενσωματώνουν ευφυείς μεθόδους έξυπνης αναγνώρισης χαρακτήρων (intelligent character recognition - ICR), όπως αναγνώριση διαφορετικών γλωσσών ή χειρόγραφα.

Αρχικά, η τεχνολογία OCR μετατρέπει το αρχείο σε ασπρόμαυρο. Η εικόνα αναλύεται για φωτεινές και σκοτεινές περιοχές, όπου οι φωτεινές περιοχές αναγνωρίζονται σαν φόντο ενώ οι σκοτεινές αναγνωρίζονται σαν γράμματα που πρέπει να αναγνωριστούν. Σε αυτό το στάδιο αναγνωρίζεται ένας χαρακτήρας, μία λέξη ή ένα κομμάτι κειμένου κάθε φορά. Στη συνέχεια, οι χαρακτήρες αναγνωρίζονται χρησιμοποιώντας είτε αναγνώριση προτύπων (pattern recognition) είτε αναγνώριση χαρακτηριστικών (feature recognition).

Ο εντοπισμός χαρακτηριστικών πραγματοποιείται όταν το OCR εφαρμόζει κανόνες σχετικά με τα χαρακτηριστικά ενός συγκεκριμένου γράμματος ή αριθμού για την αναγνώριση χαρακτήρων στο σαρωμένο έγγραφο. Τα χαρακτηριστικά περιλαμβάνουν τον αριθμό γωνιακών γραμμών, διασταυρούμενων γραμμών ή καμπυλών σε έναν χαρακτήρα. Για παράδειγμα, το κεφαλαίο γράμμα "A" αποθηκεύεται ως δύο διαγώνιες γραμμές που συναντώνται με μια οριζόντια γραμμή στη μέση. Όταν αναγνωρίζεται ένας χαρακτήρας, μετατρέπεται σε κωδικό ASCII (Αμερικανικός Τυπικός Κώδικας για Ανταλλαγή Πληροφοριών) που χρησιμοποιούν τα συστήματα υπολογιστών για να χειριστούν περαιτέρω χειρισμούς.

Ένα πρόγραμμα OCR αναλύει επίσης τη δομή μιας εικόνας εγγράφου. Χωρίζει τη σελίδα σε στοιχεία όπως μπλοκ κειμένων, πίνακες ή εικόνες. Οι γραμμές χωρίζονται σε χαρακτήρες και μετά σε λέξεις. Μόλις ξεχωρίσουν οι χαρακτήρες, το πρόγραμμα τους συγκρίνει με ένα σύνολο εικόνων μοτίβων. Μετά την επεξεργασία όλων των πιθανών αντιστοιχιών, το πρόγραμμα παρουσιάζει το αναγνωρισμένο κείμενο.

Tesseract

Για την εφαρμογή OCR χρησιμοποιήθηκε η βιβλιοθήκη Tesserac,t η οποία ακολουθεί μία τυπική επεξεργασία του κειμένου βήμα βήμα. Αναλυτικά:

α) Το πρώτο βήμα είναι η ανάλυση των συνιστωσών, και η αποθήκευση των περιγραμμάτων των στοιχείων που απαρτίζουν το κείμενο. Με την ανάλυση των περιγραμμάτων είναι εύκολο να αναγνωριστεί το κείμενο, όσο εύκολο είναι και στην περίπτωση που μετατρέπεται σε ασπρόμαυρο. Σε αυτό το στάδιο τα περιγράμματα μετατρέπονται σε blobs.

β) Στη συνέχεια τα blobs οργανώνονται σε γραμμές κειμένου, και ελέγχεται αν πρόκειται για χειρόγραφο ή για τυπογραφημένο κείμενο. Ακολούθως, οι γραμμές αναλύονται σε λέξεις ανάλογα με τα κενά που εντοπίζονται ανάμεσά τους. Το τυπογραφημένο κείμενο χωρίζεται αμέσως σύμφωνα με το μέγεθος των χαρακτήρων που έχουν χρησιμοποιηθεί, ενώ το χειρόγραφο με καθορισμένους και ασαφείς χώρους ανάμεσα στις λέξεις.

γ) Στο επόμενο βήμα εφαρμόζεται ένα διπλό πέρασμα. Στο πρώτο πέρασμα γίνεται προσπάθεια να αναγνωριστεί κάθε λέξη με τη σειρά της. Κάθε λέξη που έχει ικανοποιητική αναγνώριση, μεταβιβάζεται σε ένα προσαρμοστικό ταξινομητή ως δεδομένα εκπαιδευσης. Ο προσαρμοστικός ταξινομητής στη συνέχεια μπορεί να αναγνωρίσει με μεγαλύτερη ακρίβεια το κείμενο στη σελίδα. Ακολουθεί το 2ο πέρασμα όπου γίνεται προσπάθεια να αναγνωριστούν οι λέξεις που δεν αναγνωρίστηκαν την πρώτη φορά.

δ) Τέλος, γίνεται ανάλυση των ασαφών σημείων και προσπάθεια να αναγνωριστεί κείμενο γραμμένο με μικρότερο μέγεθος.

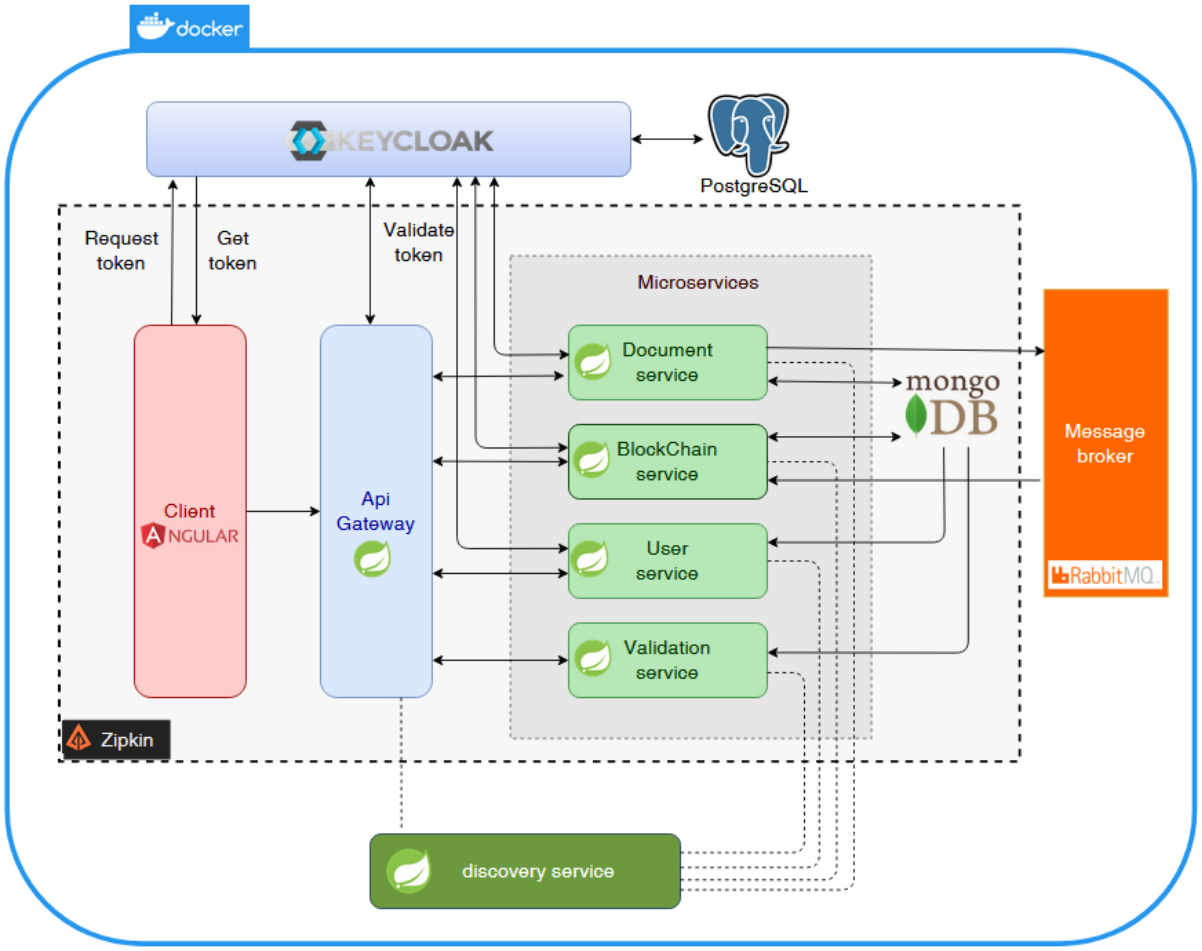
Εφαρμογή

Δομή

Η εφαρμογή απαρτίζεται από τα εξής modules:

- Api-Gateway: Αποτελεί το μοναδικό σημείο σύνδεσης με τον client. Όλα τα requests περνάνε από το api-gateway, και από εκεί δρομολογούνται στο κατάλληλο service για επεξεργασία.
- Naming - service: Κάθε microservice, είναι απαραίτητο να γνωρίζει την IP κάθε άλλου service της εφαρμογής. Αυτό όμως δεν είναι δυνατό να γίνει χωρίς το discovery server, ο οποίος λειτουργεί σαν registry για όλες τις διευθύνσεις των microservices.
- User-service: Χρησιμοποιείται για αναζήτηση των χρηστών με κριτήρια, αλλά και διαχείριση με βάση τον ρόλο που έχουν.
- Validation-service: Είναι το μόνο service το οποίο δεν απαιτεί να έχει γίνει login, και χρησιμεύει για να γίνει επιβεβαίωση της γνησιότητας του εγγράφου.
- Document-service: Αποθηκεύει τα έγγραφα που ανεβάζει ο εξουσιοδοτημένος χρήστης, αφού ολοκληρωθεί η διαδικασία του OCR, αλλά και επιστρέφει έγγραφα και σχετικές με αυτά πληροφορίες, που αναζητά κάποιος συνδεδεμένος χρήστης.
- Blockchain-service: Χτίζει τα blocks που απαρτίζουν το blockchain και διατηρεί την πληροφορία αναλλοίωτη.
- Angular web app (client) : Αποτελεί το γραφικό περιβάλλον το οποίο μπορεί να χρησιμοποιήσει ο χρήστης, είτε με την ιδιότητα του υπαλλήλου του πανεπιστημίου, είτε ως φοιτητής αλλά και ως απλός επισκέπτης για να επιβεβαιώσει τη γνησιότητα κάποιου εγγράφου.

Επιπλέον έχει χρησιμοποιηθεί Keycloak server για την αυθεντικοποίηση του χρήστη, με βάση PostgreSQL, βάση MongoDB για την αποθήκευση των documents και των blocks, RabbitMQ για αποστολή μηνυμάτων μεταξύ των services της εφαρμογής, zipkin για το tracing των requests και όλα αυτά έγιναν deploy σε docker-compose (εικόνα 18).



- Εικόνα 18 -

Τα images της εφαρμογής είναι διαθέσιμα στο https://hub.docker.com/u/dlnioanna.

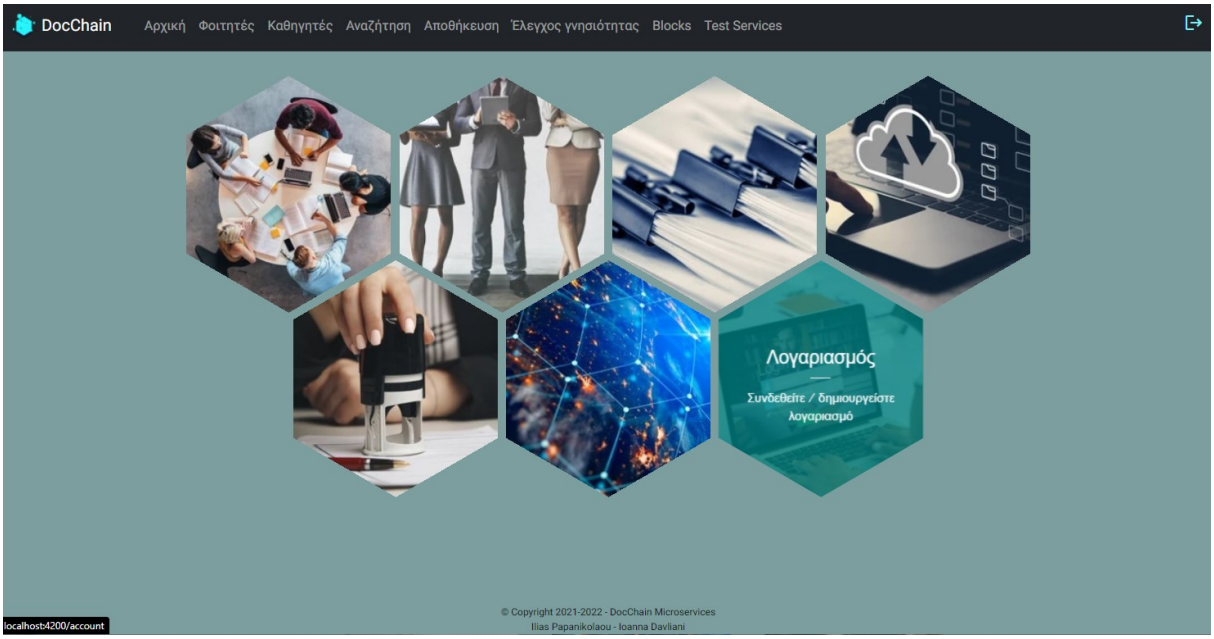
Front-end

Η αρχική σελίδα (εικόνα 19) έχει μενού περιήγησης όπου ο χρήστης θα πρέπει να κάνει login για να περιηγηθεί και αναλόγως το ρόλο του, θα είναι διαθέσιμες επιπλέον λειτουργίες στις καρτέλες. Η μόνη λειτουργία που δεν απαιτεί σύνδεση και είναι ανοιχτή χωρίς στοιχεία χρήστη είναι ο «Έλεγχος γνησιότητας».

- Εικόνα 19 -

Σύνδεση - εγγραφή χρήστη

Αρχικά ο χρήστης μπορεί να κάνει εγγραφή με ρόλο USER. Ο admin του συστήματος θα πρέπει να του αποδόσει ρόλο STUDENT ή EMPLOYEE. Σε περίπτωση που ο χρήστης προσπαθήσει να χρησιμοποιήσει κάποια υπηρεσία ενώ δεν είναι συνδεδεμένος, ή αν το χρονικό όριο της σύνδεσής του έχει λήξει, η εφαρμογή θα τον ανακατευθύνει στη σελίδα σύνδεσης. Σε περίπτωση που έχει ήδη λογαριασμό, απαιτείται απλά να συνδεθεί με τα στοιχεία του (εικόνα 20).

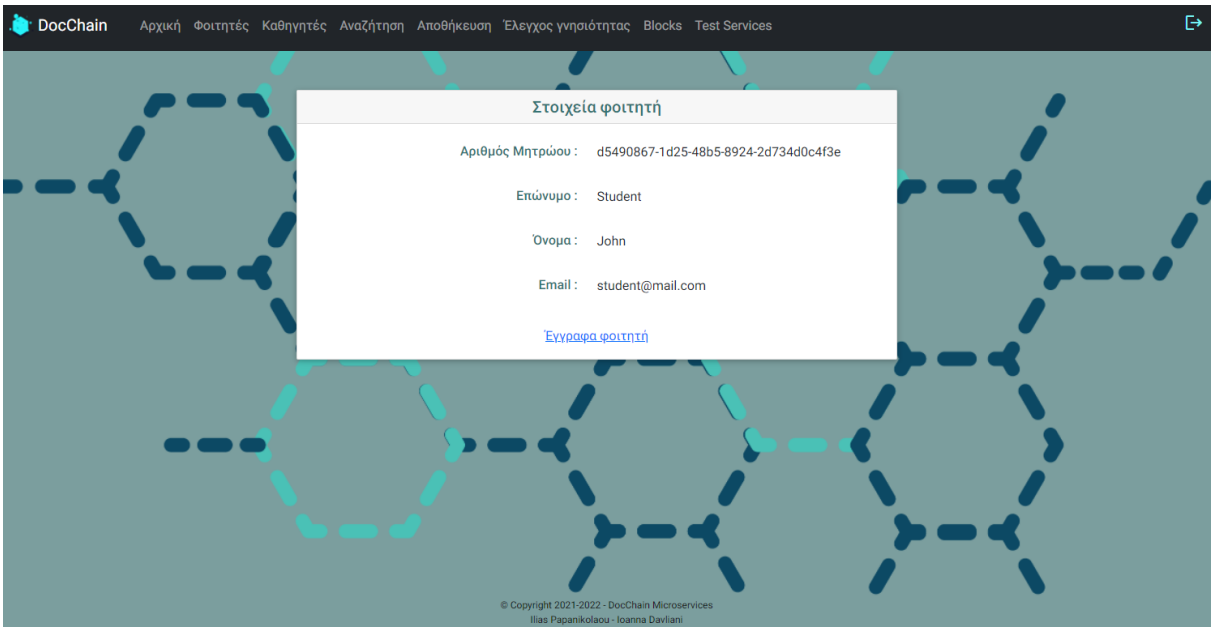


- Εικόνα 20 -

Φοιτητές

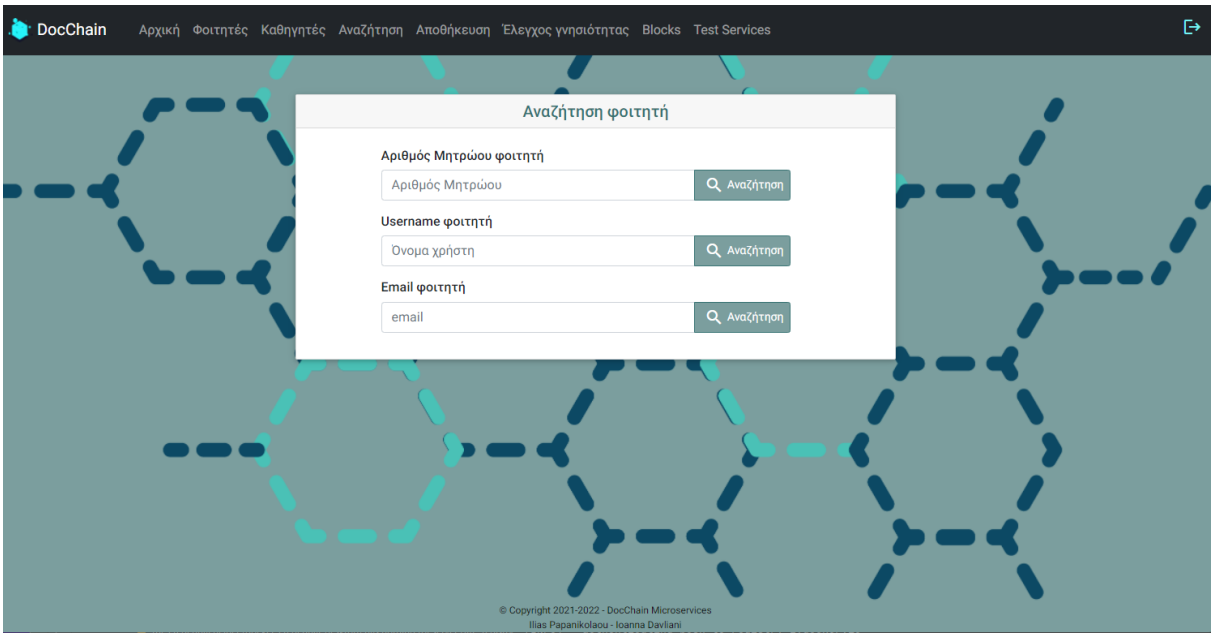
Αν ο χρήστης που θα συνδεθεί είναι φοιτητής στην αντίστοιχη καρτέλα (εικόνα 21) αυτή θα δει τα στοιχεία του (εικόνα 22).

- Εικόνα 21 –



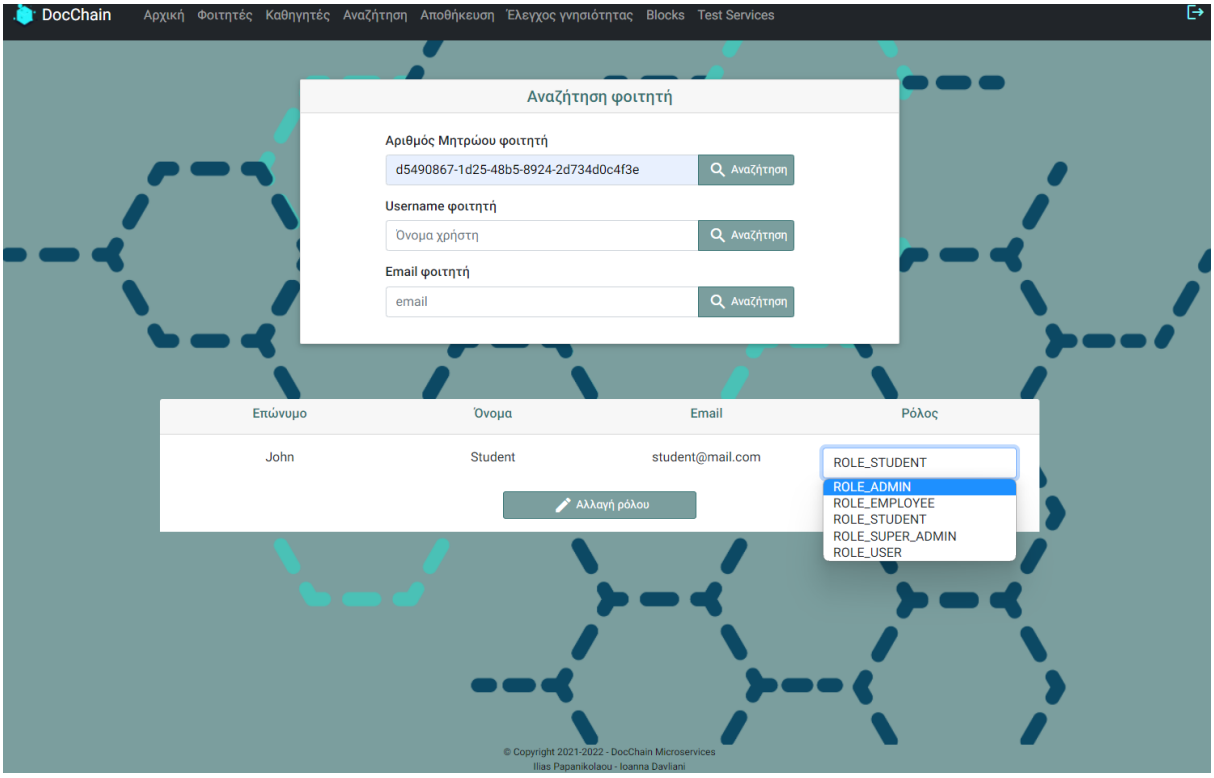
- Εικόνα 22 -

Αν πρόκειται για υπάλληλο του Πανεπιστημίου, στην ίδια καρτέλα, μπορεί να αναζητήσει κάποιον φοιτητή με κριτήρια : (α) τον αριθμό μητρώου φοιτητή, δηλαδή το id που του έχει αποδοθεί από το keycloak κατά την εγγραφή του, (β) το username του το οποίο είναι μοναδικό και (γ) το email του το οποίο επίσης είναι μοναδικό (εικόνα 23).



- Εικόνα 23 -

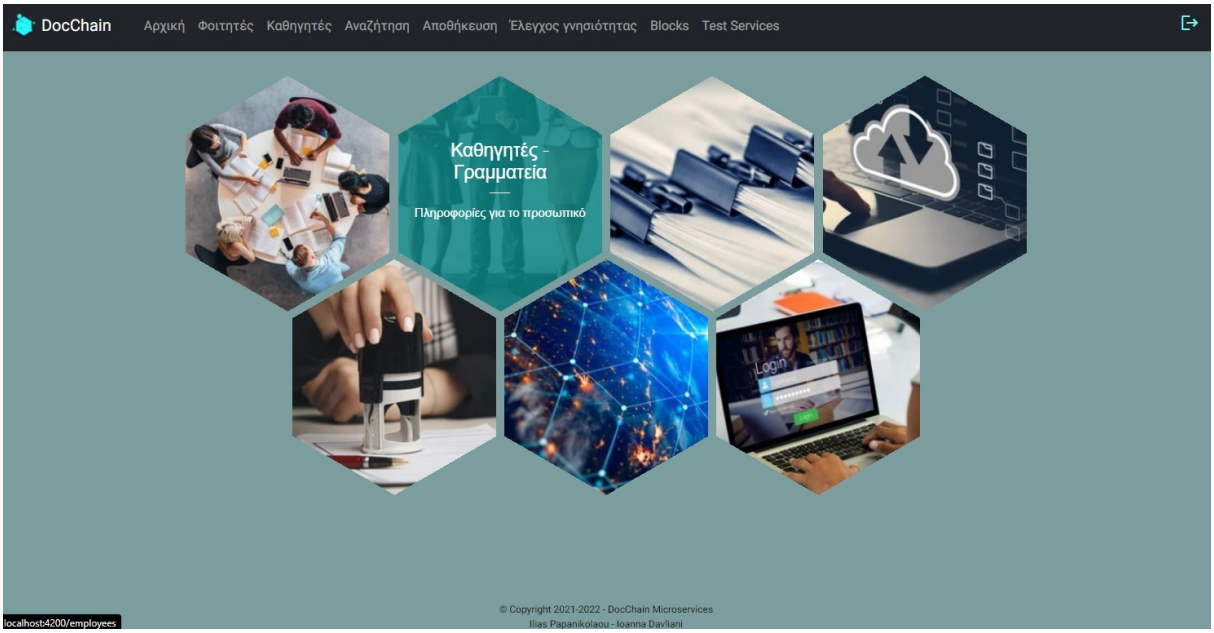
Αν συνδεθεί κάποιος admin σε αυτή την καρτέλα, έχει την ίδια επιλογή με τον υπάλληλο του πανεπιστημίου, με τη διαφορά ότι στα αποτελέσματα αναζήτησης μπορεί να επεξεργαστεί το ρόλο του φοιτητή (εικόνα 24).



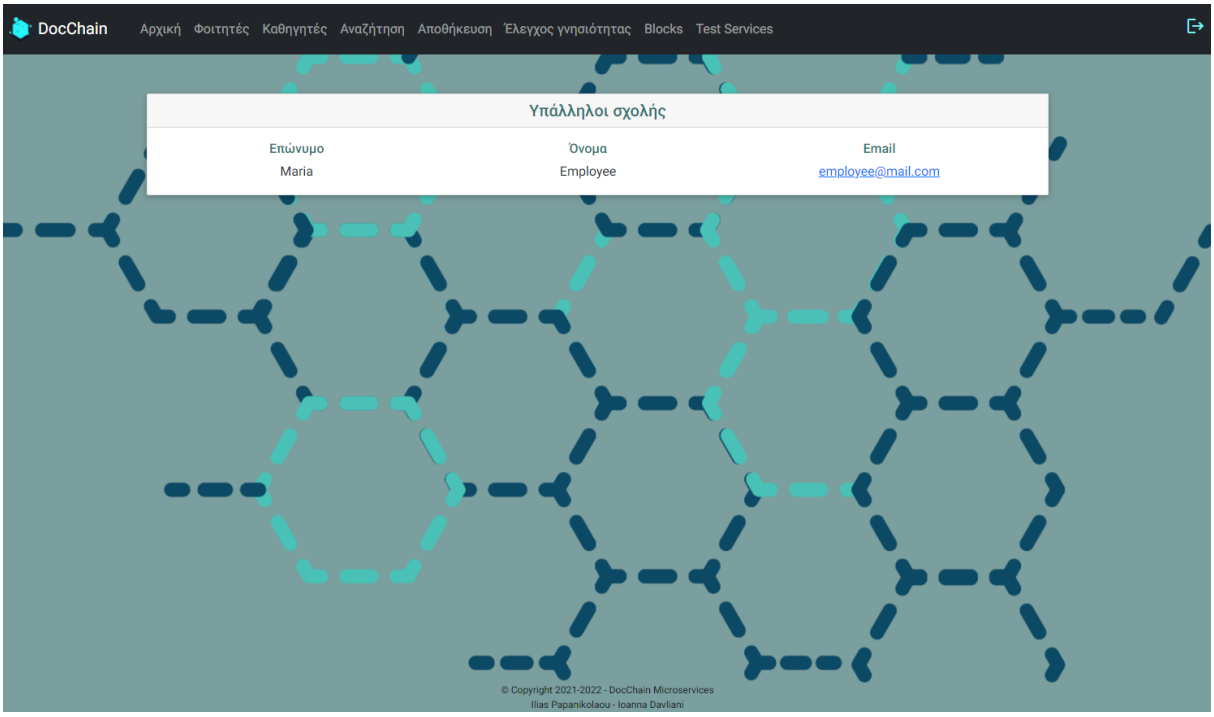
- Εικόνα 24 -

Καθηγητές

Αν ο χρήστης που θα συνδεθεί είναι φοιτητής στην καρτέλα των καθηγητών (εικόνα 25), θα δει τα ονόματα των καθηγητών του πανεπιστημίου, καθώς και στοιχεία επικοινωνίας (εικόνα 26).

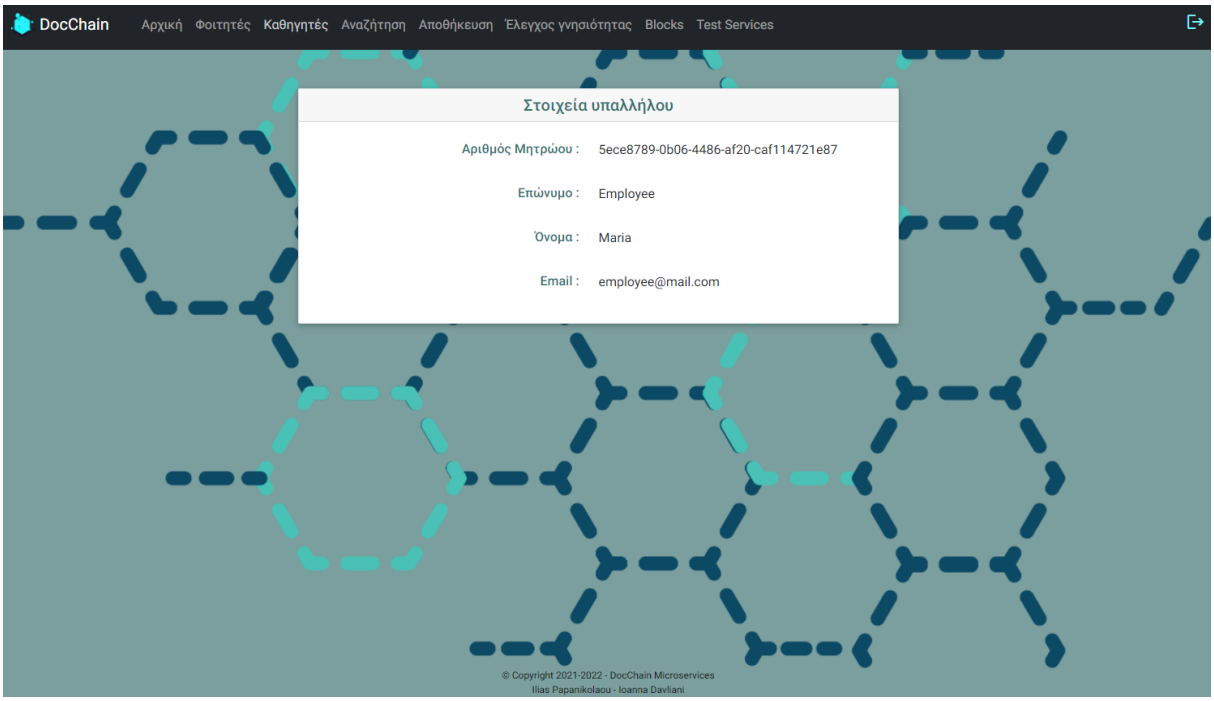


- Εικόνα 25 -



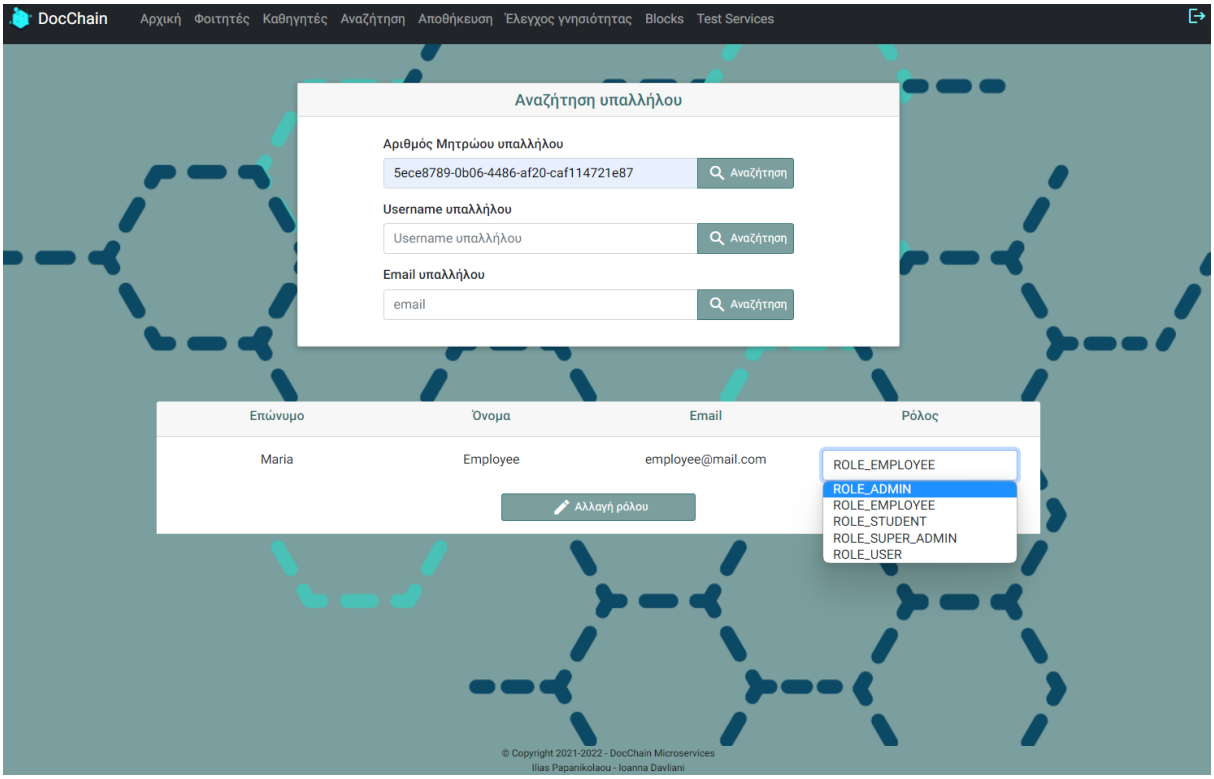
- Εικόνα 26 -

Αντίστοιχα αν πρόκειται για υπάλληλο ή καθηγητή θα δει τα δικά του στοιχεία (εικόνα 27).



- Εικόνα 27 -

Ο διαχειριστής του συστήματος όπως στην καρτέλα του φοιτητή μπορεί να κάνει αναζήτηση, έτσι κι εδώ μπορεί να αναζητήσει με τα ίδια κριτήρια κάποιον καθηγητή και να επεξεργαστεί τον ρόλο του (εικόνα 28).

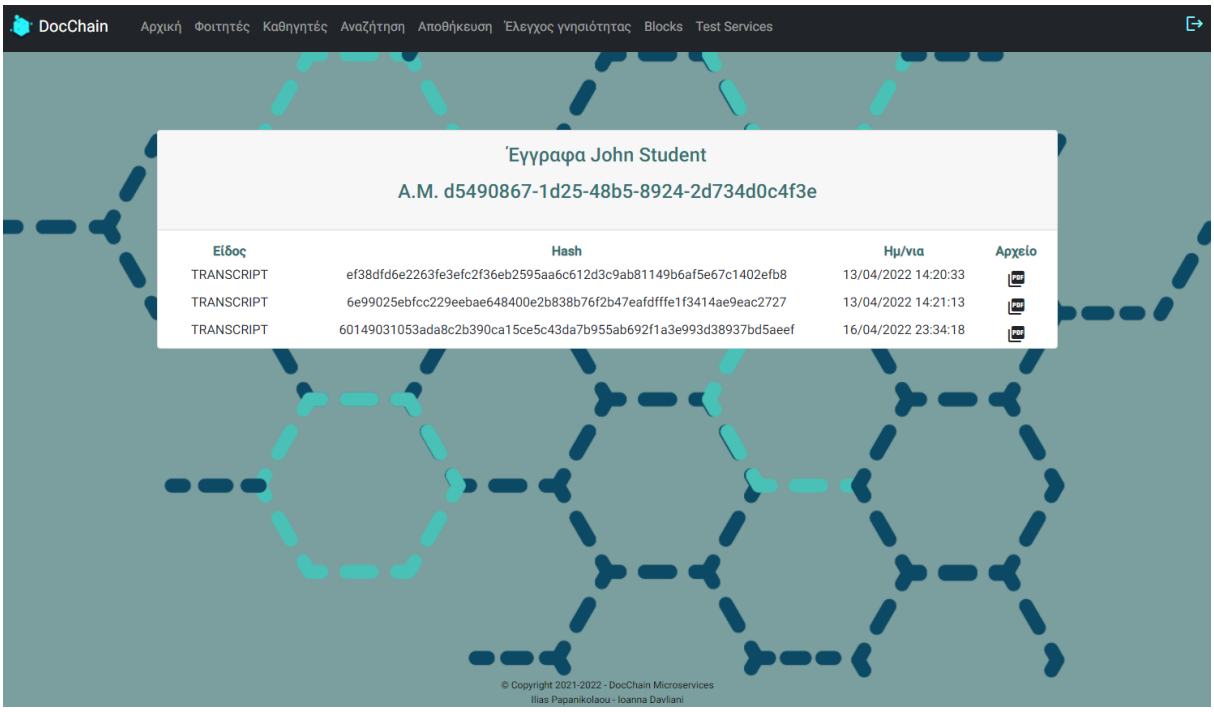


- Εικόνα 28 -

Αναζήτηση αρχείων

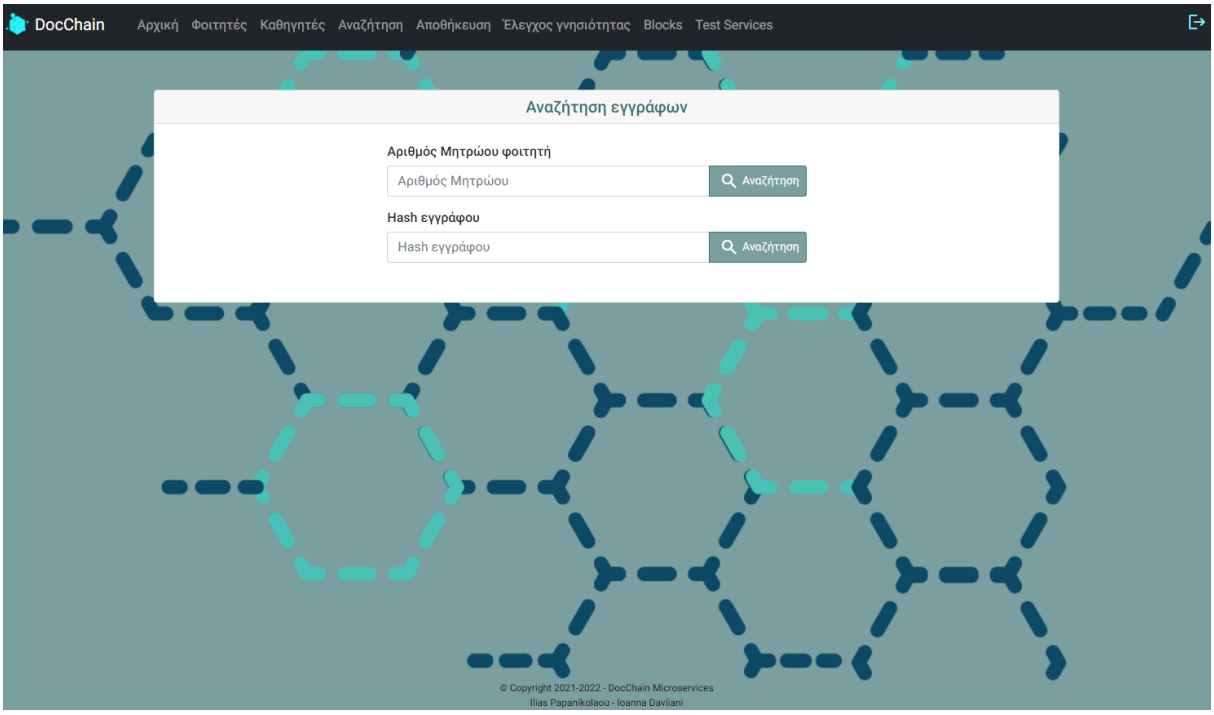
Στη συγκεκριμένη σελίδα (εικόνα 29) οι φοιτητές μπορούν να δουν τα έγγραφα που υπάρχουν διαθέσιμα για τους ίδιους, τα οποία κάποιος υπάλληλος της σχολής έχει ανεβάσει και έχουν αποθηκευτεί στο blockchain. Στους φοιτητές δεν επιτρέπεται να κάνουν κάποια αναζήτηση αλλά να δουν όλα τα έγγραφα που τους αφορούν (εικόνα 30).

- Εικόνα 29 -



- Εικόνα 30 -

Οι συνδεδεμένοι υπάλληλοι - καθηγητές και διαχειριστές του συστήματος μπορούν να κάνουν αναζήτηση με τον αριθμό μητρώου φοιτητή, ώστε να δουν όλα τα καταχωρημένα έγγραφα για αυτον τον φοιτητή, ή με το hash του εγγράφου που τους ενδιαφέρει (εικόνα 31).



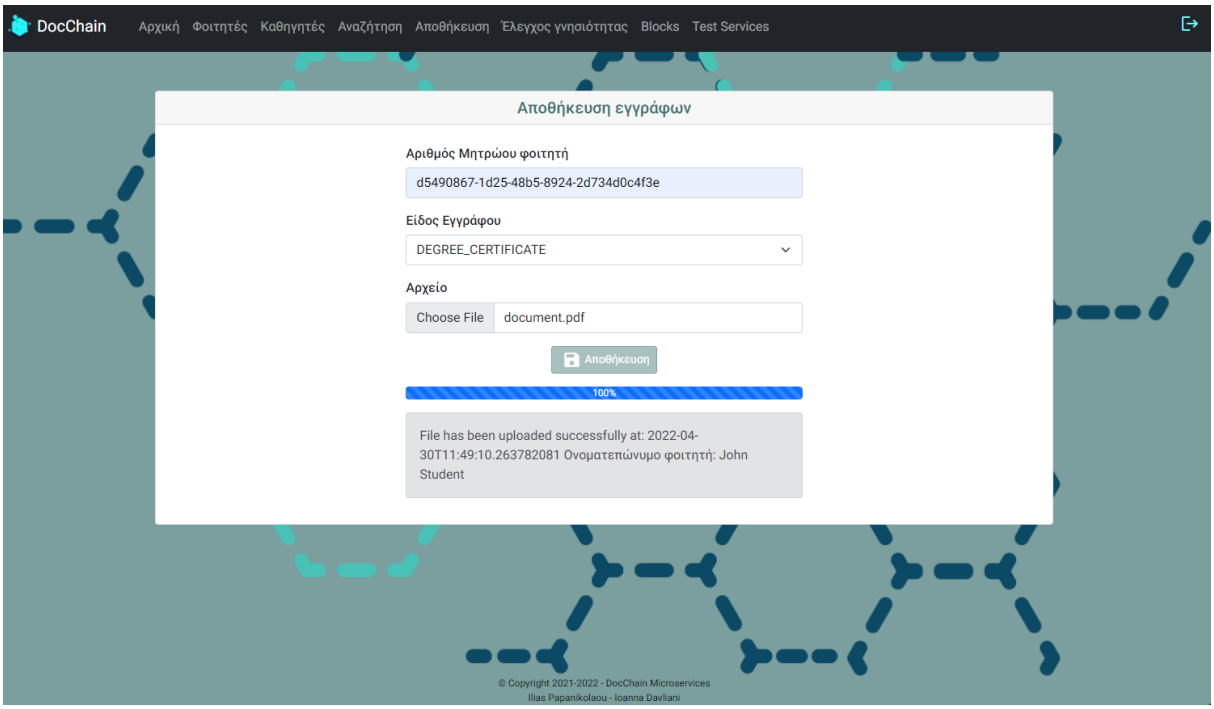
- Εικόνα 31 -

Αποθήκευση αρχείων

Οι φοιτητές έχουν μεν πρόσβαση στη σελίδα (εικόνα 32), αλλά τους εμφανίζεται μήνυμα να απευθυνθούν στη γραμματεία της σχολής, καθώς δεν έχουν δικαίωμα να ανεβάσουν έγγραφα.

- Εικόνα 32 -

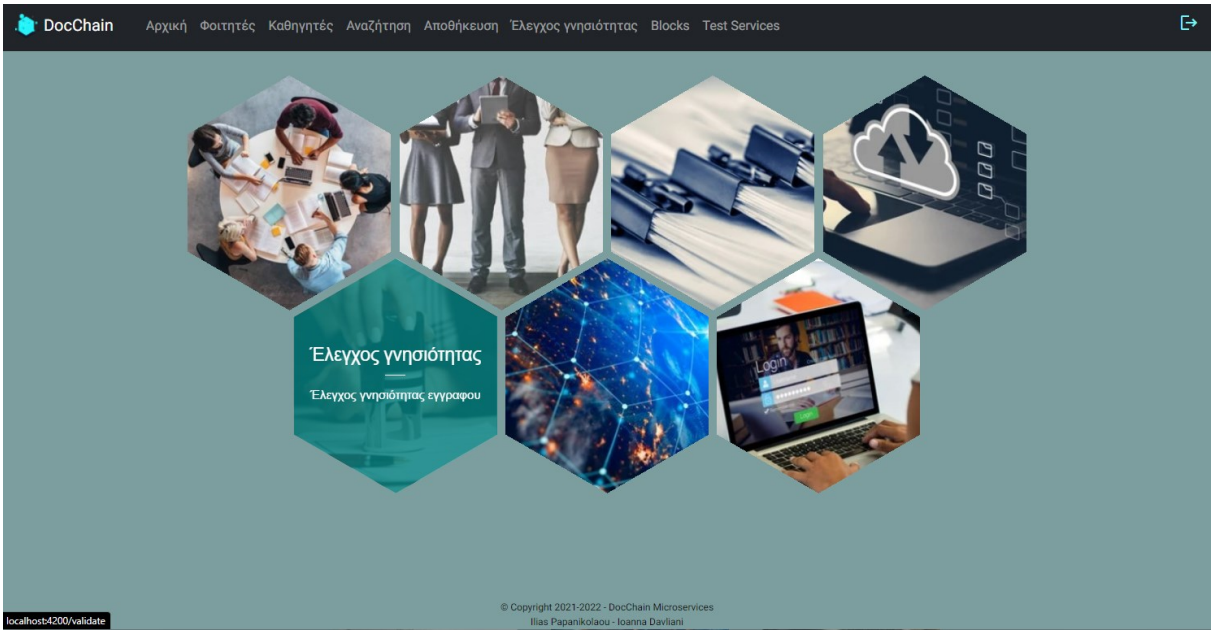
Αντίθετα οι υπάλληλοι και ο admin, βλέπουν την παρακάτω φόρμα (εικόνα 33), όπου υποχρεωτικά συμπληρώνουν τον αριθμό μητρώου φοιτητή, τον τύπο εγγράφου που ανεβάζουν και τέλος επιλέγουν το αρχείο, τύπου pdf ή εικόνα. Μόλις το αρχείο ανέβει επιτυχώς εμφανίζεται σχετικό μήνυμα. Σε περίπτωση αποτυχίας επίσης εμφανίζεται μήνυμα που ενημερώνει τον χρήστη ότι το ανέβασμα απέτυχε.



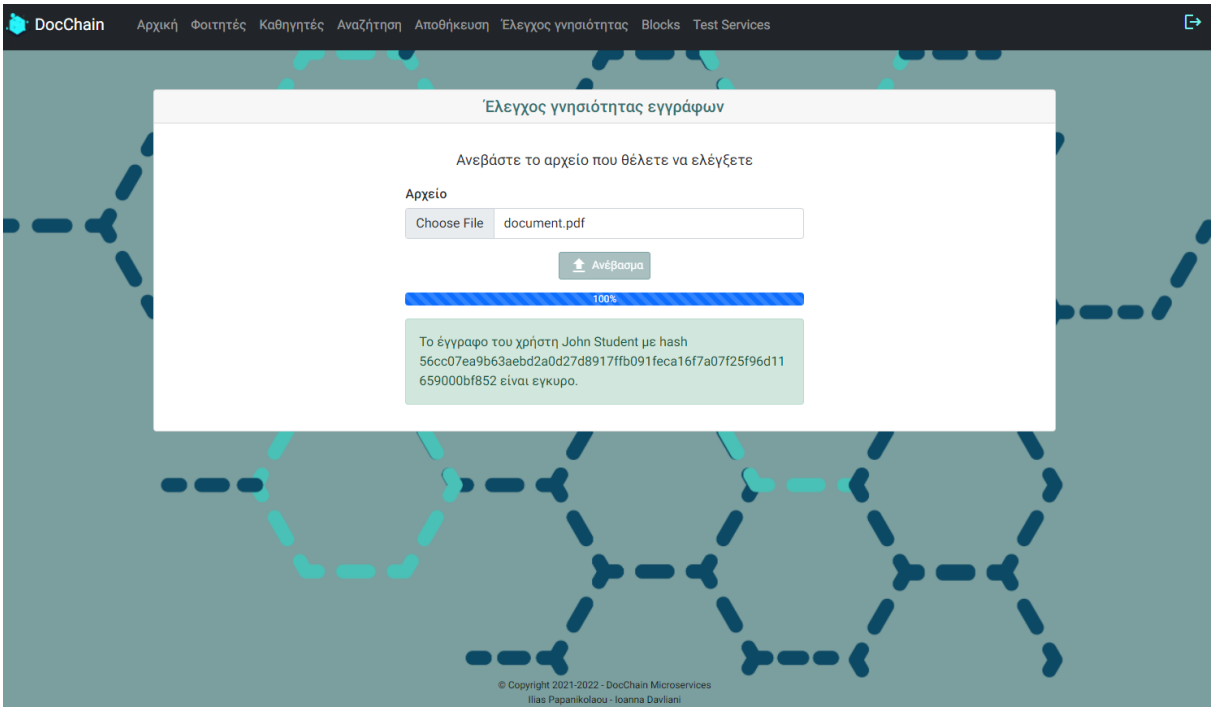
- Εικόνα 33 –

Έλεγχος γνησιότητας

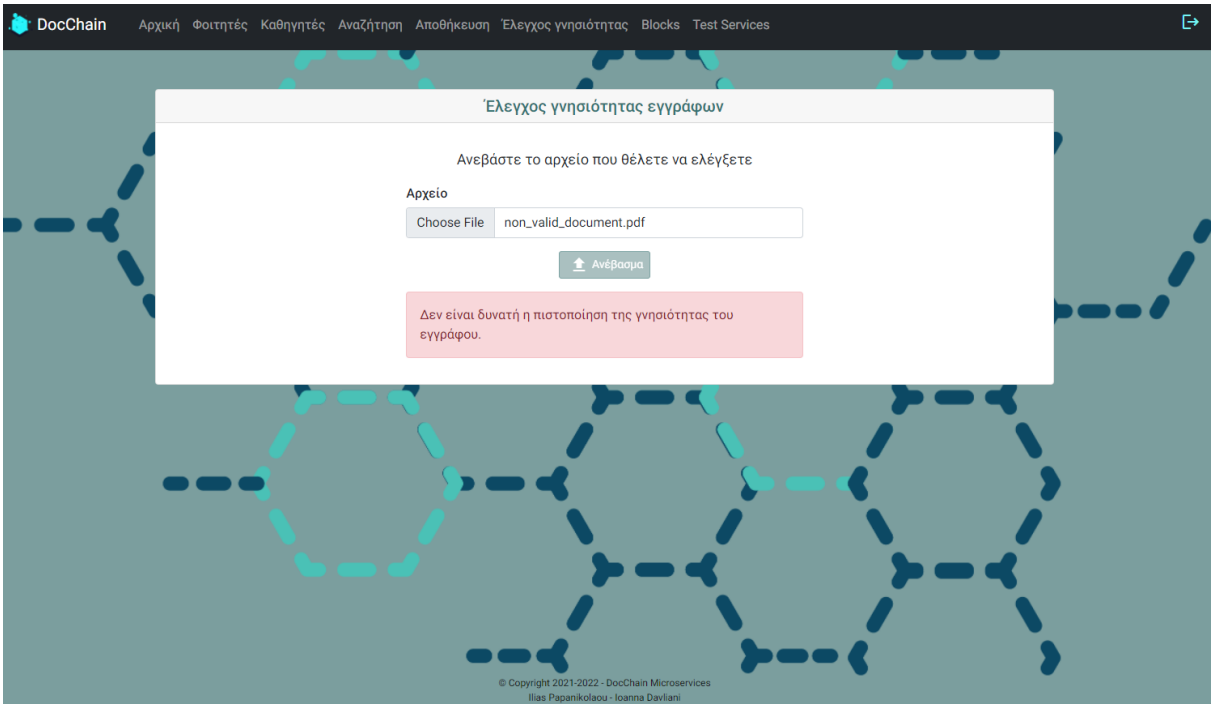
Όπως αναφέρθηκε η σελίδα ελέγχου γνησιότητας (εικόνα 34) είναι η μόνη μου δεν απαιτεί αυθεντικοποίηση του χρήστη. Είναι ανοιχτή ώστε οποιοσδήποτε θέλει να ελέγξει τη γνησιότητα ενός εγγράφου, να μπορεί απλά να ανεβάζει το αρχείο που έχει στα χέρια του και θα παίρνει σαν απάντηση αν το αρχείο έχει πιστοποιηθεί και σε θετική περίπτωση, θα εμφανίζεται και το όνομα του φοιτητή στον οποίο ανήκει, όπως φαίνεται παρακάτω (εικόνες 35 & 36).



- Εικόνα 34 -



- Εικόνα 35 -



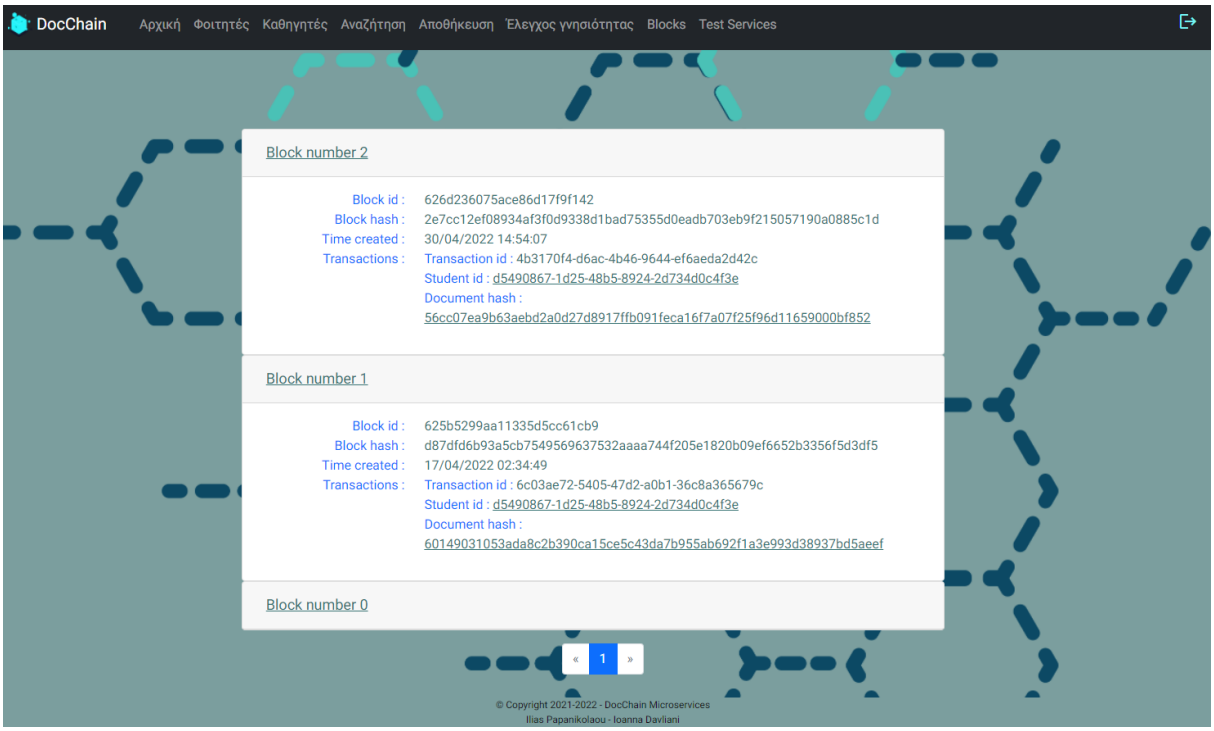
- Εικόνα 36 -

Blockchain

Η τελευταία σελίδα (εικόνα 37) μας δίνει πληροφορίες για όλα τα blocks που υπάρχουν.

- Εικόνα 37 -

Στην περίπτωση που κάποιος είναι συνδεδεμένος σαν φοιτητής μπορεί να δει τα blocks τα οποία έχουν πληροφορία σχετικά με δικά του έγγραφα, ενώ ένας υπάλληλος του πανεπιστημίου μπορεί να δει όλα τα blocks που υπάρχουν στη βάση (εικόνα 38). Επιλέγοντας το link student id ή document hash, οδηγείται στην αναζήτηση εγγράφων όπου εμφανίζονται όλα τα έγγραφα του επιλεγμένου φοιτητή ή το συγκεκριμένο έγγραφο αντιστοιχα.



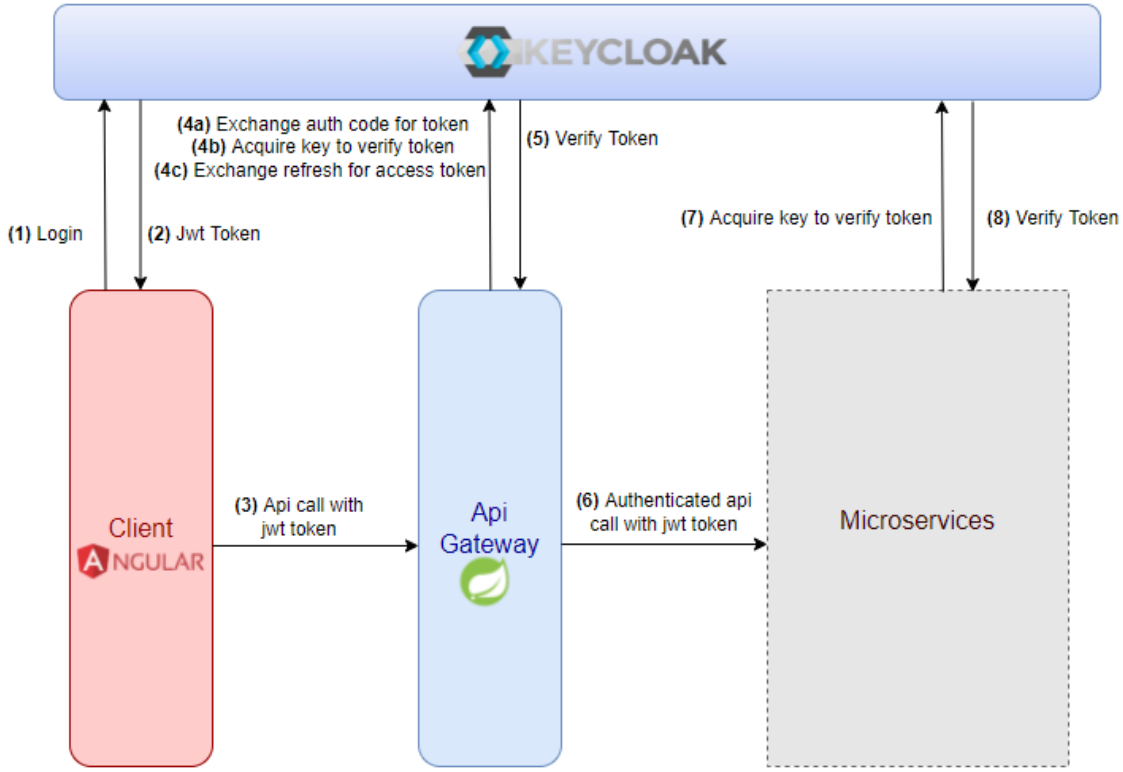
- Εικόνα 38 -

Back-end

Το backend αποτελείται από 6 microservices, τα api-gateway, naming-service, blockchain-service, document-service, user-service, validation-service.

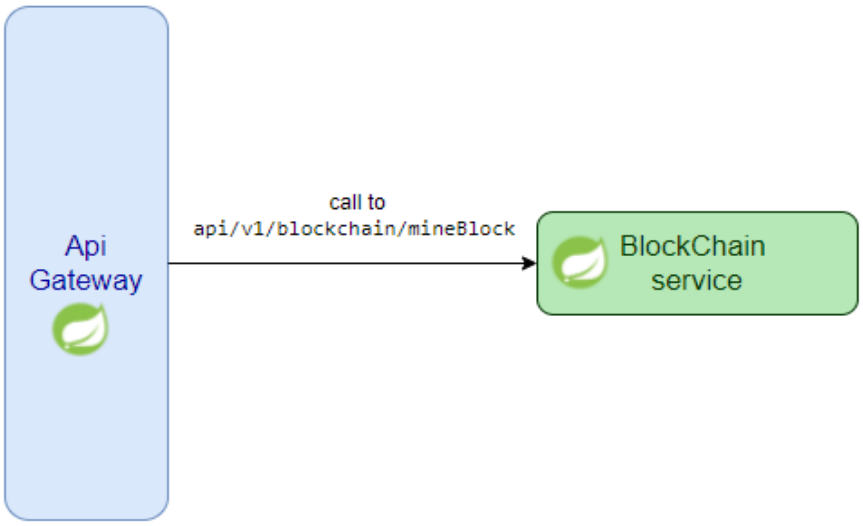
Api-gateway

Είναι το σημείο σύνδεσης με το front-end. Όλα τα requests περνάνε από το api-gateway, αφού τα endpoints των microservices δεν είναι γνωστά στον client, επομένως δεν μπορεί να κάνει απευθείας κλήση σε κάποιο άλλο microservice. Το api-gateway δουλεύει σαν reverse proxy από εκεί δρομολογούνται όλα τα requests στο κατάλληλο service για επεξεργασία. Επίσης είναι το πρώτο σημείο εφαρμογής του spring security (εικόνα 39).



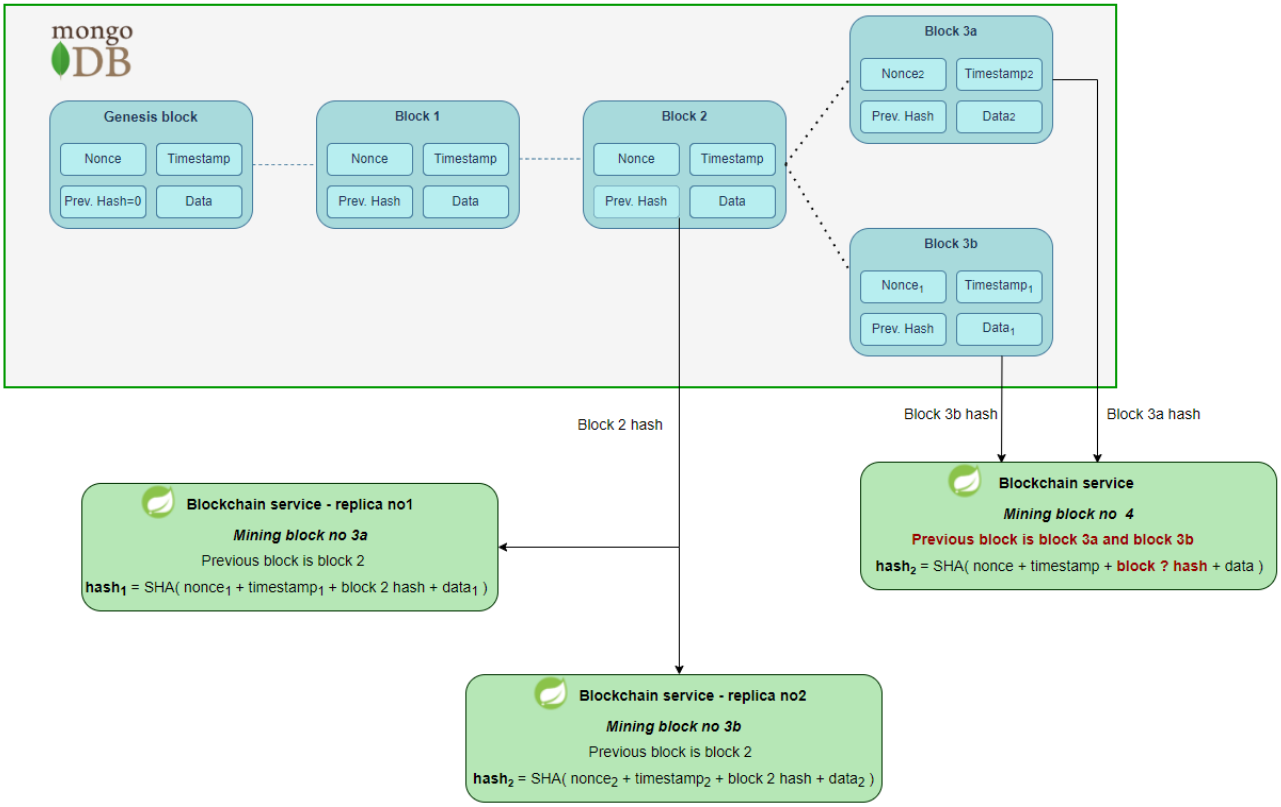
- Εικόνα 39 -

Στην εφαρμογή αξιοποιούμε το γεγονός ότι επικοινωνεί απευθείας με τα endpoints των υπόλοιπων services και του έχουμε αναθέσει να κάνει mine ένα block κάθε 10 λεπτά, με τη χρήση scheduler και καλώντας το mineblock endpoint blockchain-service (εικόνα 40). Ένας άλλος λόγος που ορίστηκε ο scheduler στο api-gateway και όχι απευθείας στο blockchain-service, είναι ότι λόγω του load balancing, αν το blockchain-service γίνει scale και δημιουργηθούν κι άλλα αντίγραφα αυτού του service, θα υπάρξει αυτόματη διαχείριση ως προς το ποιο στιγμιότυπο του service θα κάνει το mine.



- Εικόνα 40-

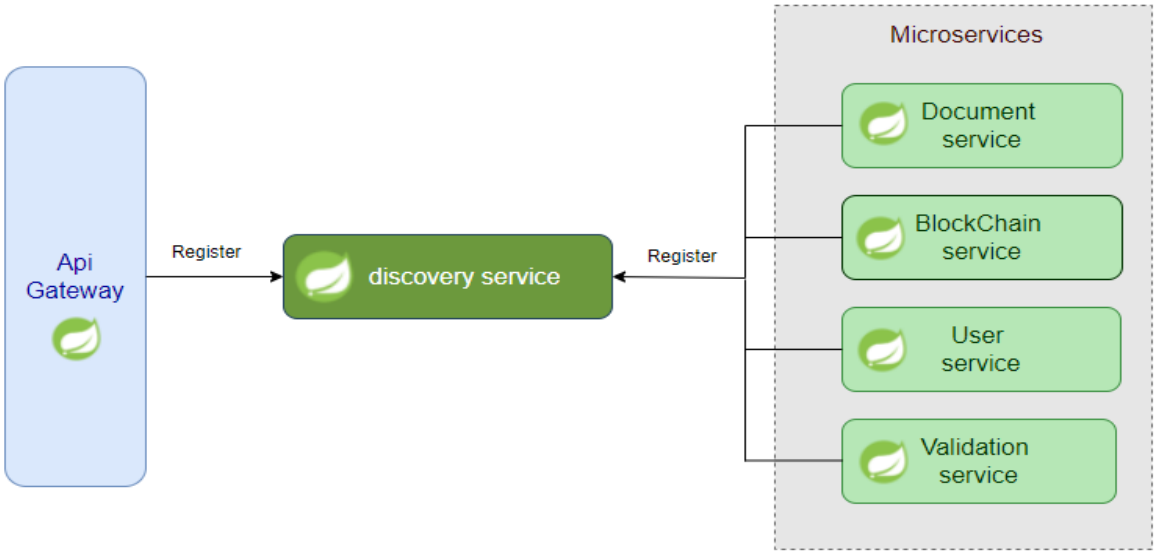
Ειδικά η διαδικασία του mining, δεν μπορεί να γίνει ταυτόχρονα από όλα τα replicas services, επειδή κάθε φορά χρειαζόμαστε την πληροφορία του τελευταίου block της αλυσίδας. Σε περίπτωση λοιπόν που 2 blockchain-services ξεκινούσαν να κάνουν mine θα έφτιαχναν από 1 block έγκυρο, όμως μόλις γινόταν η αποθήκευση θα υπήρχε ασυνέχεια στην αλυσίδα (εικόνα 41). Αναθέτοντας την έναρξη του mining στο api-gateway, ακόμα και αν δημιουργηθουν αντίγραφα του blockchain-service, θα καλούνται με χρήση του αλγορίθμου round-robin, εξασφαλίζοντας σωστή λειτουργία, αφού αν κάποιο service τεθεί εκτός λειτουργίας θα δημιουργηθεί το block από κάποιο άλλο, ενώ λόγω της χρονικής απόστασης των 10 λεπτών μεταξύ του mine ενός νέου block, είναι αδύνατο να ξεκινήσει το χτίσιμο ενός νέου block χωρίς να έχει ολοκληρωθεί το προηγούμενο.



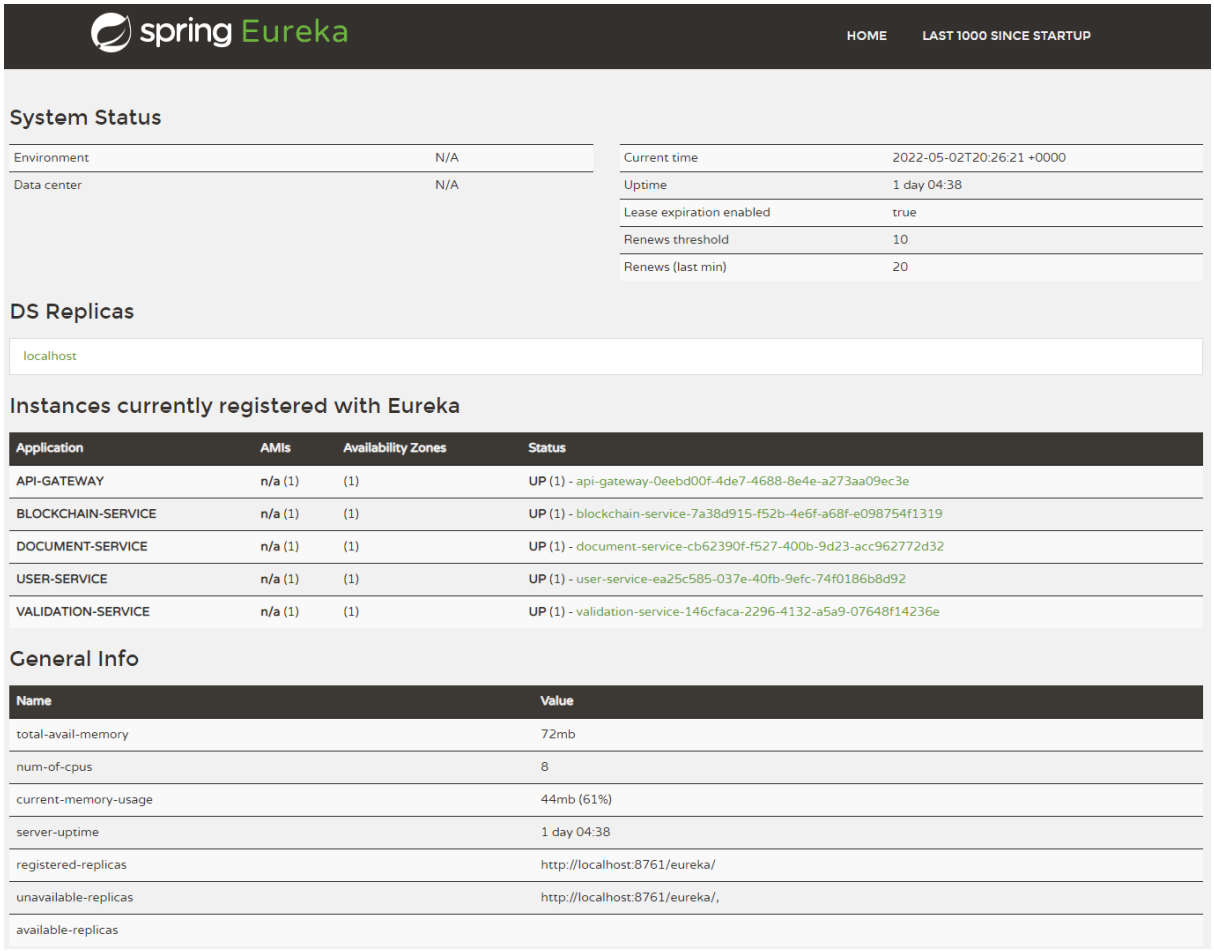
- Εικόνα 41 -

Naming server

Κάθε service κάνει register στο συγκεκριμένο στο naming-server το οποίο κρατάει την πληροφορία για το που υπάρχει το κάθε service (εικόνα 42).Στη συνέχεια δίνει την πληροφορία αυτή στα υπόλοιπα για να επικοινωνούν μεταξύ τους κάθε φορά που πρέπει να στείλουν κάποιο request . Παρακάτω φαίνεται η πληροφορία που κρατάει ο eureka server την οποία μοιράζεται με τα υπόλοιπα services (εικόνα 43).



- Εικόνα 42 -

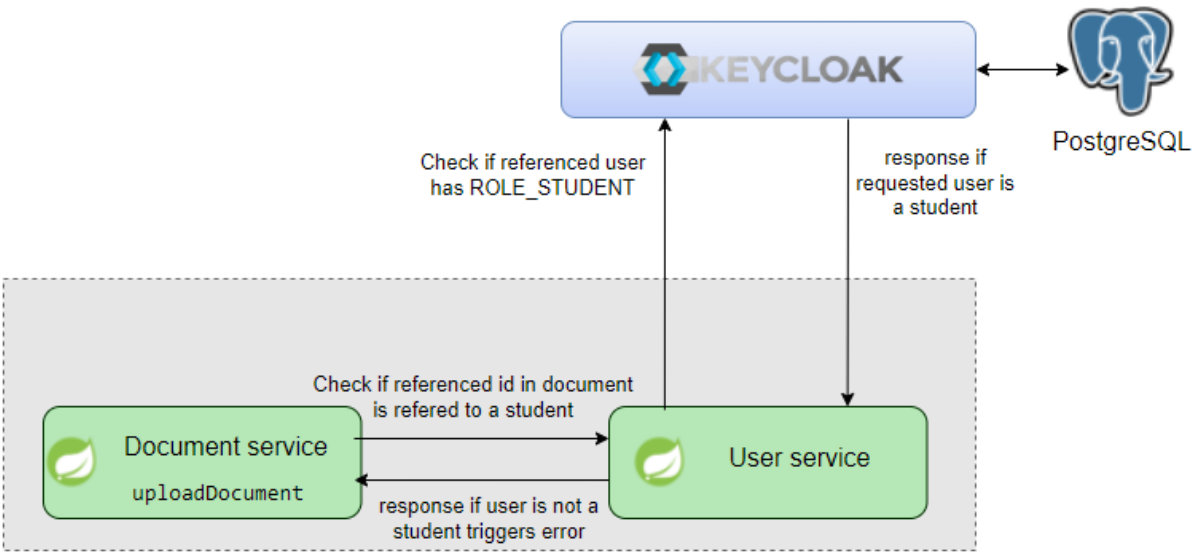


- Εικόνα 43 -

User service

Το user service χρησιμεύει στην διαχειριση των χρηστών και στην αναζήτηση και διαχείριση των φοιτητών και υπαλλήλων του πανεπιστημίου. Επικοινωνεί απευθείας με το keycloak server ώστε να κάνει αναζήτηση των χρηστών στην PostgreSQL.

Τα υπόλοιπα services μπορούν να επικοινωνήσουν με το user-service για να ελέγξουν οποιαδήποτε πληροφορία έχει να κάνει με φοιτητή ή υπάλληλο ή συνδεδεμένο χρήστη της εφαρμογής. Συγκεκριμένα το document-service επικοινωνεί με το user-service (εικόνα 44), για να ελέγξει αν τα έγγραφα που ανεβαίνουν αφορούν κάποιον φοιτητή, ώστε να μην μπορεί να γίνει αλόγιστη χρήση από κάποιον και να αποφευχθεί η αποθήκευση εγγράφων που δεν εχουν να κάνουν με έγγραφα φοιτητών.



- Εικόνα 44 -

Document service

Το document-service συνδέεται με τη MongoDB και χρησιμεύει για την αποθήκευση και αναζήτηση των εγγράφων που αποθηκεύει ο χρήστης (υπάλληλος πανεπιστημίου) και το hash των οποίο εγγράφεται στην αλυσίδα του blockchain.

Το ανέβασμα των αρχείων, ενώ γίνεται ασύγχρονα, για να βρεθεί στο hash του εγγράφου, θα πρέπει να ολοκληρωθεί η λήψη των FilePart που αποστέλλει ο client, ώστε να γίνει προσωρινή του αποθήκευση, κατάλληλη επεξεργασία OCR και στη συνέχεια αποθήκευση στη βάση. Ενώ όπως αναφέρθηκε, η αρχιτεκτονική του project είναι ασύγχρονη, σε αυτό το σημείο ήταν αναγκαίο να χρησιμοποιηθεί blocking μεθοδος, η οποία όμως δεν επιβαρύνει σε μεγάλο βαθμό την απόδοση του προγράμματος.

Όταν ο υπάλληλος του πανεπιστημίου ανεβάζει ένα έγγραφο για αποθήκευση, αρχικά γίνεται επικοινωνία με το user-service για να ελεγχθεί ότι το έγγραφο που ανεβαίνει αφορά φοιτητή του πανεπιστημίου (εικόνα - 44). στη συνέχεια εάν το έγγραφο που ανεβαίνει είναι μορφής *.pdf, μπορεί να συνεχίσει η διαδικασία για την εύρεση του SHA χωρίς περεταίρω επεξεργασία, όμως σε περίπτωση που το έγγραφο είναι εικόνα τύπου *.png ή *.jpg, γίνεται επεξεργασία OCR, ώστε το έγγραφο να αποθηκευτεί σε μορφή PDF/A.

Ο τύπος PDF/A είναι μια τυποποιημένη έκδοση ISO του Portable Document Format (PDF) που ειδικεύεται για χρήση στην αρχειοθέτηση και τη μακροπρόθεσμη διατήρηση ηλεκτρονικών εγγράφων. Το PDF/A διαφέρει από το PDF επειδή απαγορεύει λειτουργίες ακατάλληλες για μακροπρόθεσμη αρχειοθέτηση, όπως η σύνδεση γραμματοσειρών (σε αντίθεση με την ενσωμάτωση γραμματοσειρών) και η κρυπτογράφηση.

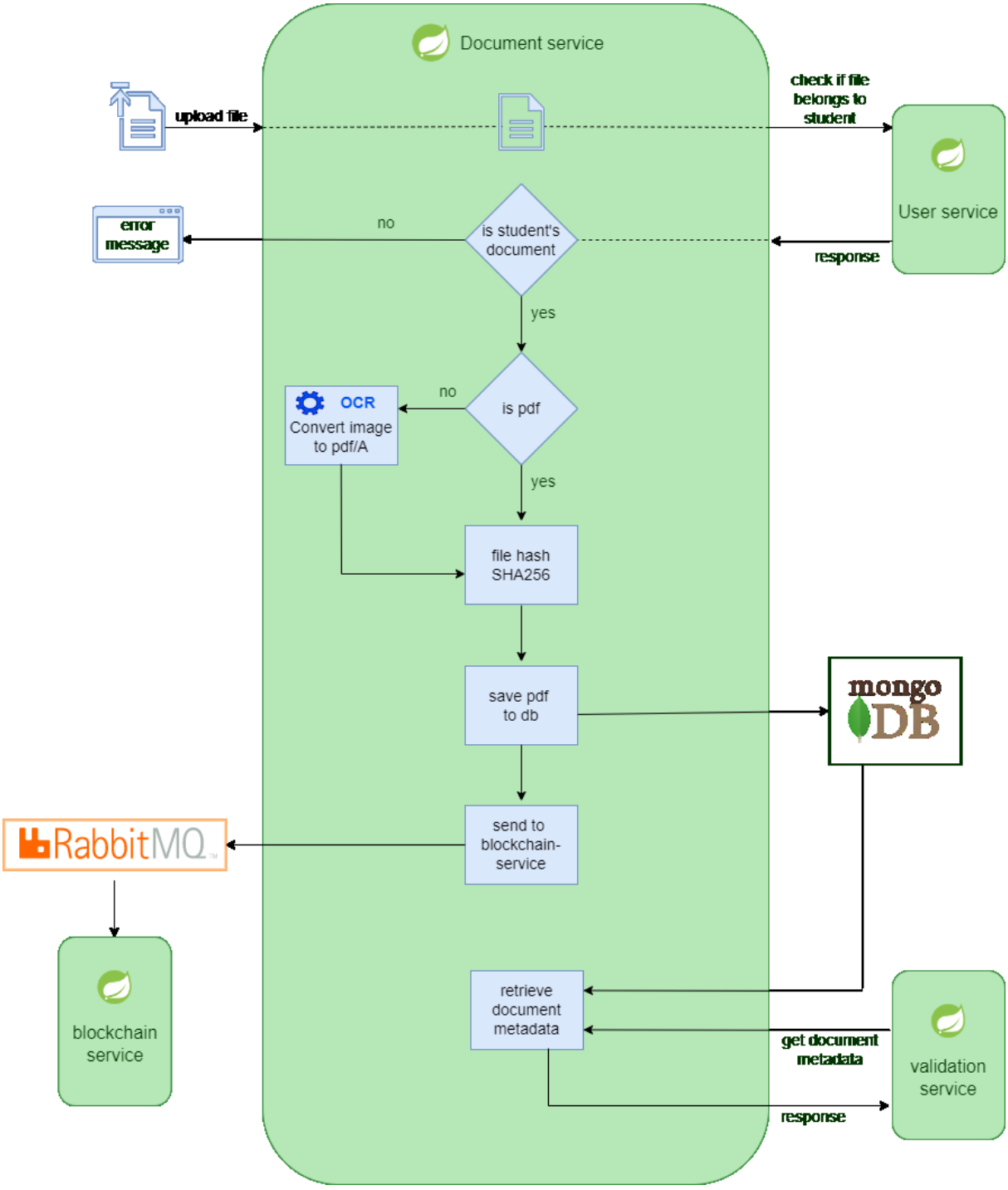
Στο σημείο αυτό, το έγγραφο το οποίο θέλουμε να αποθηκεύσουμε, είναι μορφής *.pdf και το επόμενο βήμα είναι να βρούμε το hash του αρχείου. Το hash, είναι μία μοναδική συμβολοσειρά που παράγεται με βάση τον κρυπτογραφικό αλγόριθμο SHA-256, ο οποίος δεν μπορεί να δουλέψει αντίστροφα, δηλαδή γνωρίζοντας τη μοναδική αυτή συμβολοσειρά, δεν είναι δυνατό να βρούμε τα δεδομένα που την παρήγαγαν. Αν αλλάξει έστω και ένα γράμμα από το περιεχόμενο του κειμένου, το νέο hash που θα παραχθεί θα είναι εντελώς διαφορετικό από το προηγούμενο. Το όνομα του αρχείου δεν επηρεάζει το hash, αφού αυτό παράγεται μόνο από το περιεχόμενό του. Έτσι εάν το hash 2 αρχείων είναι το ίδιο, τότε πρόκειται για το ίδιο αρχείο. Σε αυτή την αρχή βασίζεται και το validation-service, όπως θα εξηγήσουμε παρακάτω.

Στη συνέχεια το έγγραφο αποθηκεύεται στη βάση, χρησιμοποιώντας το GridFS, το οποίο όπως αναφέρθηκε, δεν επιτρέπει update της πληροφορίας που έχει αποθηκευτεί, και έτσι συντελεί στην ασφάλεια του συστήματος, ώστε να μην μπορέσει κάποιος να τροποποιήσει το περιεχόμενο των εγγράφων. Το hash κάθε εγγράφου που έχει εγγραφεί στο blockchain, δεν μπορεί να αλλάξει καθώς τότε το έγγραφο παύει να είναι έγκυρο.

Επιτρέποντας στον χρήστη να κάνει update θα άλλαζε το hash οπότε θα έπρεπε να τροποποιηθεί το block που περιέχει το συγκεκριμένο έγγραφο, κάτι που είναι αδύνατο. Επομένως αποτελεί μία επιπλέον δικλείδα ασφαλείας ώστε τα αποθηκευμένα έγγραφα, να μην είναι εφικτό να τροποποιηθούν ακόμα και σε περίπτωση κακόβουλης ενέργειας.

Μόλις το έγγραφο αποθηκευτεί στη βάση, θα πρέπει να εγγραφεί η σχετική πληροφορία στο blockchain. Αυτό θα γίνει από το blockchain-service, όμως θα πρέπει να ενημερωθεί από το document-service. Η ενημέρωση αυτή γίνεται με τη χρήση του κατάλληλου message-broker, όπου χρησιμοποιώντας το docChain-exchange και προσθέτοντας το μήνυμα στο document-queue, γίνεται η αποστολή ενός αντικειμένου τύπου AppDocument, το οποίο περιέχει το id (αριθμό μητρώου) του φοιτητή τον οποίο αφορά το έγγραφο, το ονοματεπώνυμό του, το id του υπαλλήλου που έκανε την καταχώριση, το είδος του εγγράφου (π.χ. βαθμολογία, πτυχίο κλπ), το id του εγγράφου ώστε να μπορεί να γίνει αναζήτηση στη βάση, αλλά και το hash του εγγράφου.

Επίσης το document service επικοινωνεί με το validation service, ώστε μόλις επικυρωθεί κάποιο έγγραφο στο blockchain, να συλλέξει τα δεδομένα του εγγράφου που θέλουμε να εμφανιστούν στον τελικό χρήστη.


Document service
upload file
check if file belongs to student
User service
error message
no
is student's document
response
yes
OCR
Convert image to pdf/A
no
is pdf
yes
file hash SHA256
save pdf to db
mongo DB
RabbitMQ
send to blockchain-service
blockchain service
retrieve document metadata
get document metadata
validation service
response

- Εικόνα 45 -

Blockchain service

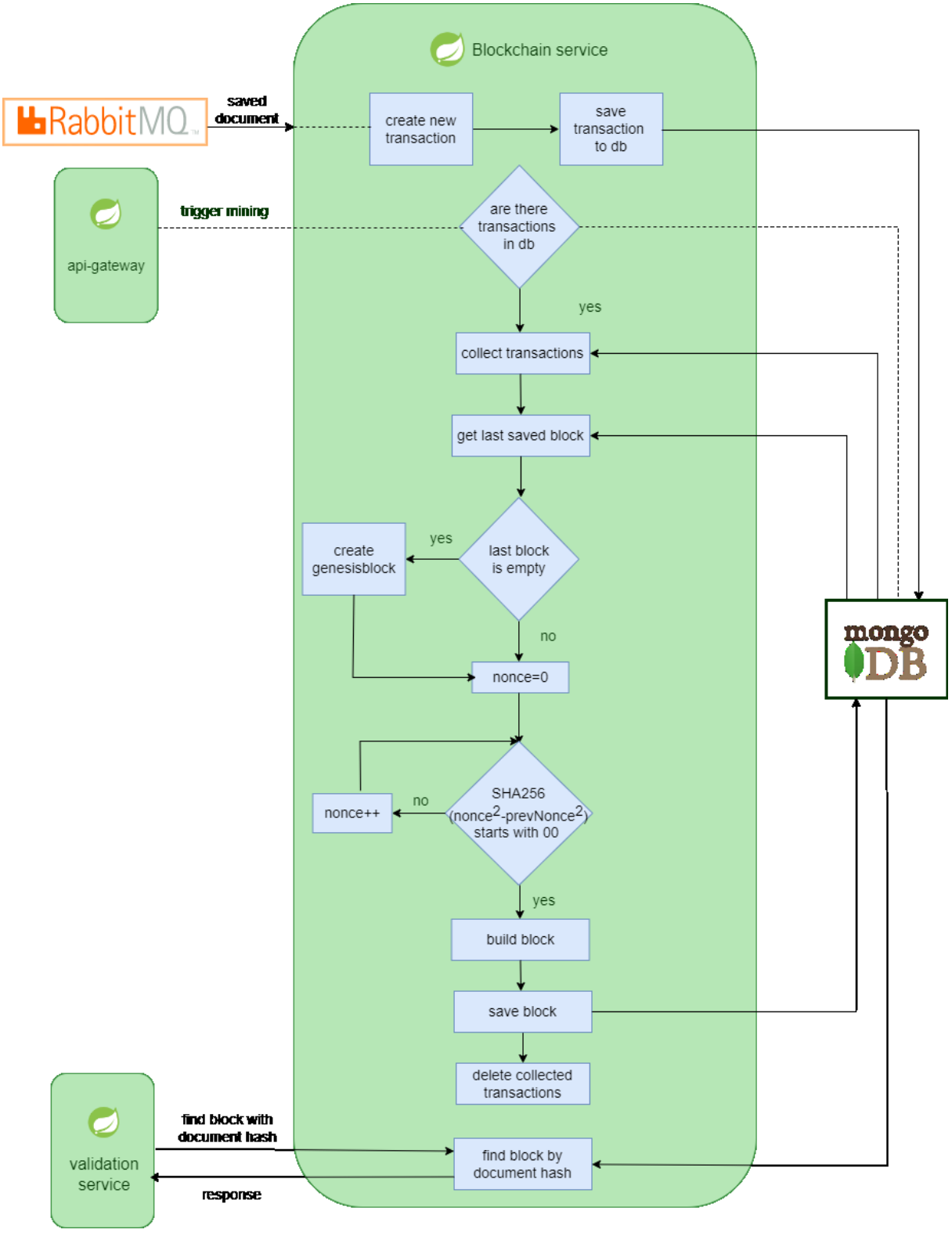
Το blockchain-service αποτελεί τον πυρήνα της εφαρμογής, αφού εδώ χτίζεται το blockchain. Αρχικά μόλις το document-service στείλει μήνυμα μέσω του RbbitMQ με την πληροφορία του εγγράφου που μόλις αποθηκεύτηκε, το blockchain-service, μέσω του listener που έχει ρυθμισμένο, ώστε να λαμβάνει τα μηνύματα που προστίθενται στο document-queue του docChain-exchange, ενημερώνεται και λαμβάνοντας το μήνυμα δημιουργεί ένα νέο transaction το οποίο περιέχει το id του φοιτητή και το hash του εγγράφου. Τα transactions αποθηκεύονται στη βάση για να είναι διαθέσιμα κατά τη δημιουργία νέου block, αφού μόλις ληφθούν από το blockchain-service, διαγράφονται από το document-queue και δεν μπορούν να ανακτηθούν από αυτό.

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, κάθε δέκα λεπτά γίνεται mine ένα νέο block. Κάθε δέκα λεπτά γίνεται έλεγχος αν υπάρχουν αποθηκευμένα transactions στη βάση. Αν δεν υπάρχουν νέα transactions δεν δημιουργείται νέο block, αφού κάτι τέτοιο θα σήμαινε κατασπατάληση των πόρων για να αποθηκευτεί ένα block που δεν περιέχει πληροφορία. Αντίθετα αν βρεθούν νέα transactions ξεκινάει η διαδικασία του mining.

Στην περίπτωση που το mining γίνεται για πρώτη φορα, πρώτα θα εισαχθεί στην αλυσίδα το genesis block, το οποίο δεν περιέχει πληροφορία. Επειδή κάθε block που εισάγεται στην αλυσίδα, βασίζεται στο προηγούμενο το genesis block θα έχει σαν hash του προηγούμενου block τιμή 0, και σκοπός του είναι να χρησιμεύσει στην παραγωγή του επόμενου block.

Στην περίπτωση λοιπόν, που βρεθούν αποθηκευμένα transactions, αυτά συλλέγονται, και η λίστα αυτή των transactions, αποτελεί την πληροφορία που αποθηκεύεται σε κάθε block και μαζί με το nonce, timestamp και το index του block παράγουν το hash του block.

Μόλις γίνει η συλλογή των transactions ξεκινάει η προσπάθεια να βρεθεί το κατάλληλο nonce για να χτιστεί τελικά το block. Η απαίτηση που υπάρχει για να θεωρηθεί κατάλληλο το nonce και ολοκληρωθεί το mining είναι το hash της διαφοράς των τετραγώνων των hash, 2 διαδοχικών block, να ξεκινάει με δύο μηδενικά, δηλαδή να είναι της μορφής “00XXXXXXXXXXXXXXXXXX”. Μόλις βρεθεί το nonce που καλύπτει αυτή την απαίτηση, έχει ολοκληρωθεί και το proof of work και το block προστίθεται στο τέλος της αλυσίδας.


Blockchain service
RabbitMQ
saved document
create new transaction
save transaction to db
api-gateway
trigger mining
are there transactions in db
yes
collect transactions
get last saved block
create genesisblock
yes
last block is empty
no
nonce=0
nonce++
no
SHA256 (nonce2-prevNonce2) starts with 00
yes
build block
save block
delete collected transactions
mongo DB
validation service
find block with document hash
find block by document hash
response

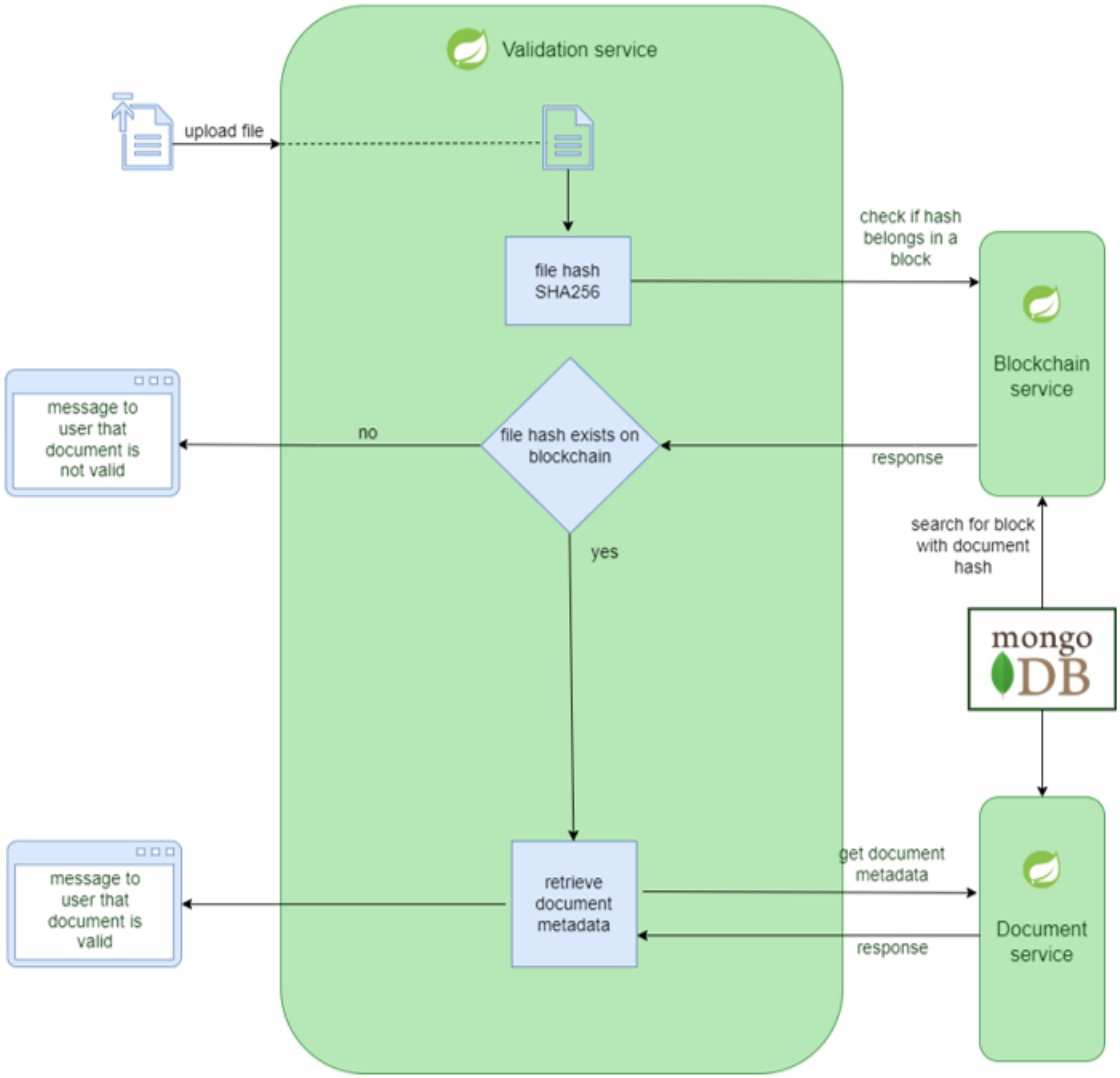
- Εικόνα 46 -

Validation service

Το validation-service χρησιμοποιείται για τον έλεγχο των εγγράφων. Κάθε χρήστης της εφαρμογής, χωρίς να είναι συνδεδεμένος ή να έχει εγγραφεί, μπορεί να ανεβάσει ένα αρχείο pdf, για να ελεγχθεί η γνησιότητά του.

Και εδώ, όπως και στο document-service, τα αρχεία ανεβαίνουν ασύγχρονα και μόλις ολοκληρωθεί το ανέβασμα, παίρνουμε το hash του εγγράφου. Στη συνέχεια το validation-service επικοινωνεί με το blockchain-service ώστε να ελέγξει ότι το hash υπάρχει μέσα σε κάποιο block και αν όντως βρεθεί, επικοινωνεί με το document service ώστε να ανακτήσει και τα υπόλοιπα στοιχεία (metadata) που πρέπει να εμφανιστούν στον χρήστη, και δεν αποθηκεύονται στο block.

Μόλις γίνει ο παραπάνω έλεγχος, εμφανίζεται μήνυμα στον χρήστη που επιβεβαιώνει ή όχι τη γνησιότητα του εγγράφου.



- Εικόνα 47 -

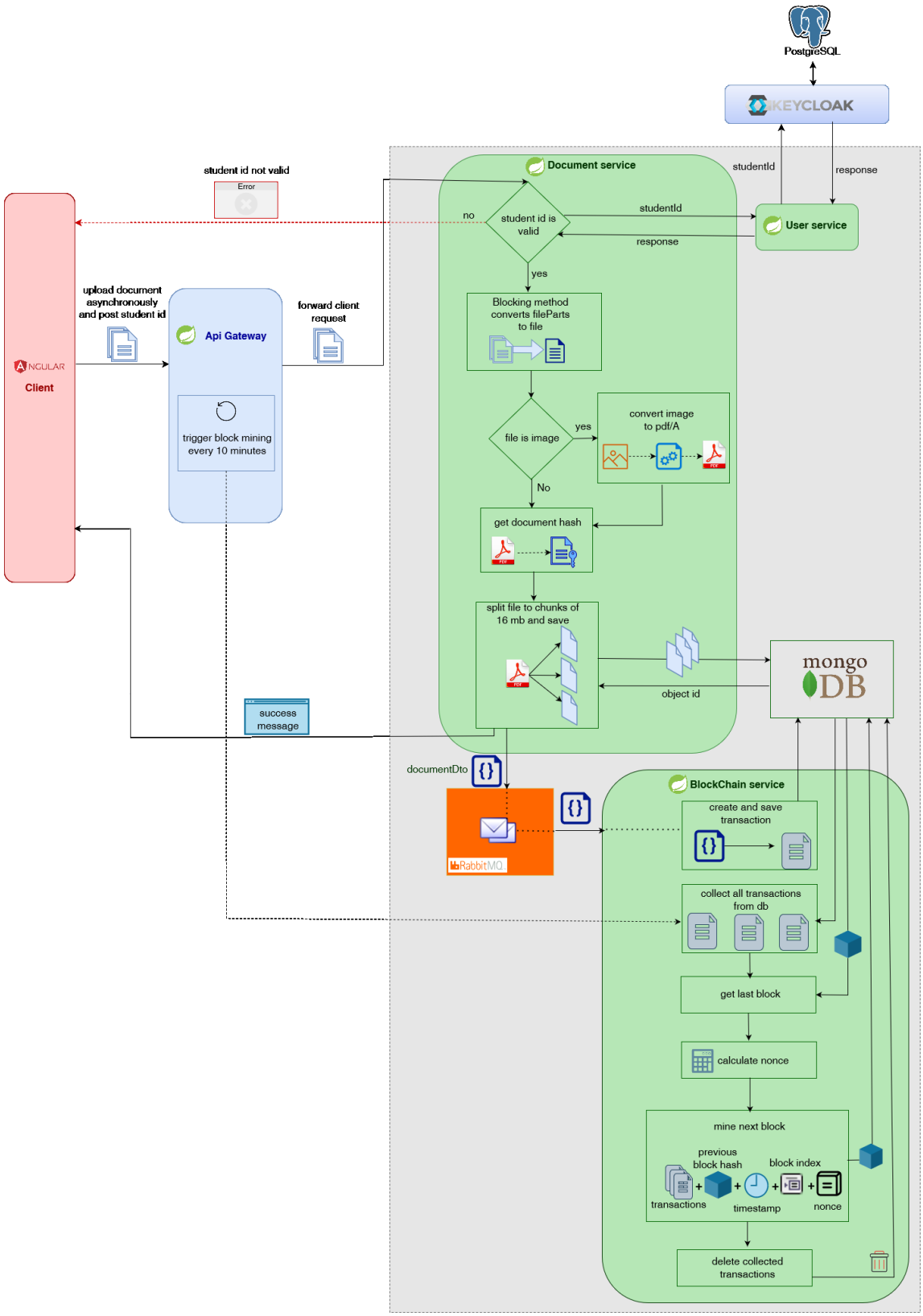
Παράδειγμα χρήσης

Upload document

Παρακάτω φαίνεται αναλυτικά η διαδικασία αποθήκευσης εγγράφου από κάποιον υπάλληλο του πανεπιστημίου (εικόνα 48). Συγκεκριμένα, όταν ο χρήστης ανεβάζει ένα έγγραφο, αυτό στέλνεται ασύγχρονα από τον client στο backend, και αφού περάσει από τη πρώτη δικλείδα ασφαλείας, το api-gateway, προωθείται στο document-service. Εκεί με κλήση στο user service ελέγχεται εάν ο αριθμός μητρώου φοιτητή τον οποίο έχει καταχωρήσει ο χρήστης, ανήκει όντως σε φοιτητή. Σε περίπτωση που δεν ανήκει σε φοιτητή, εμφανίζεται μήνυμα λάθους στον καταχωρητή, ενώ στην περίπτωση που το μητρώο ανήκει όντως σε φοιτητή, συλλέγονται τα fileParts που έχουν σταλεί και συγκροτουν το αρχείο που στάλθηκε αρχικά. Στη συνέχεια γίνεται έλεγχος εάν το αρχείο είναι εικόνα ή pdf και σε περίπτωση που είναι εικόνα, μέσω OCR μετατρέπεται σε pdf/A. Σε αυτό το σημείο το αρχείο είναι σίγουρα pdf και βρίσκουμε το hash του εγγράφου, το οποίο αφού σπάσει σε chunks των 16mb, με χρήση του gridFS, αποθηκεύεται στη βάση. Μόλις το blockchain service ολοκληρώσει αυτή τη διαδικασία, στέλνει μήνυμα επιτυχούς αποθήκευσης στον χρήστη και ένα μήνυμα με ένα DTO που περιέχει τις απαραίτητες πληροφορίες στο message broker, ώστε να προωθηθεί η πληροφορία στο blockchain service.

Μόλις το blockchain service λάβει το DTO που έστειλε το document service, δημιουργεί ένα αντικείμενο transaction που περιέχει πληροφορίες σχετικά με την καταχώρηση του εγγράφου.

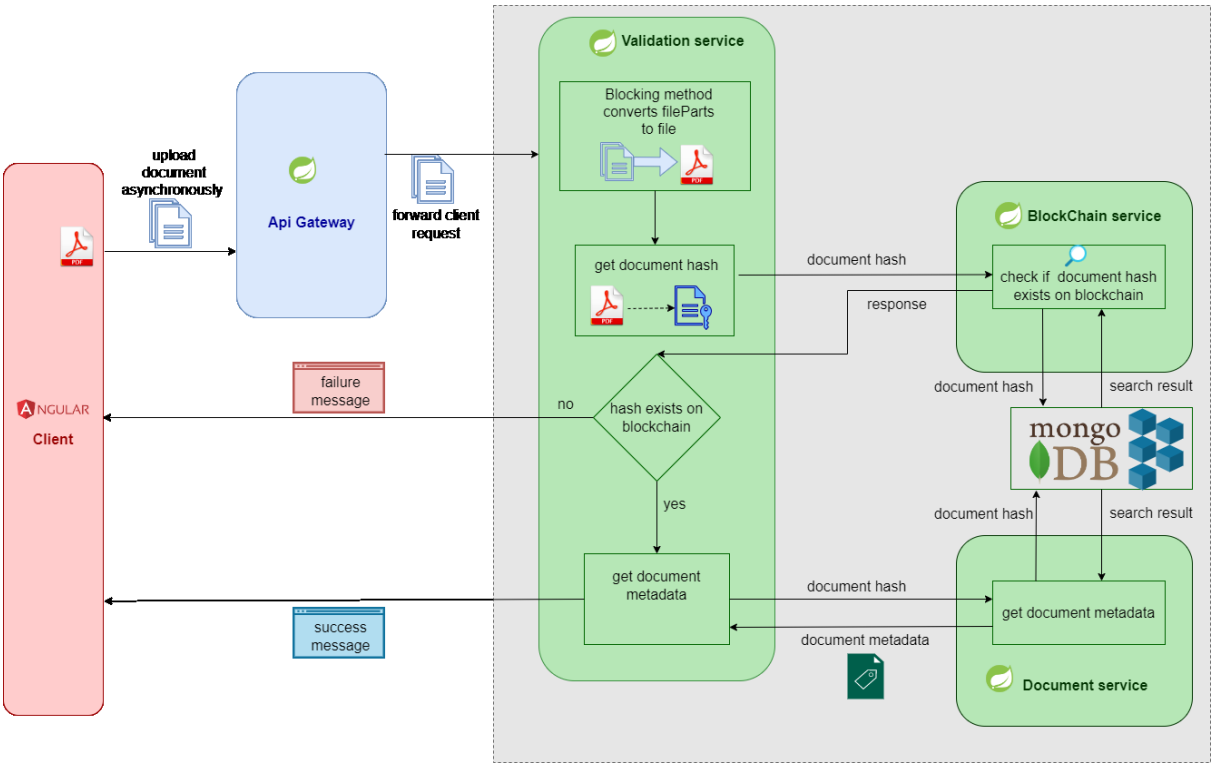
Το api gateway είναι προγραμματισμένο ώστε κάθε 10 λεπτά, να κάνει μία κλήση στο blockchain service, προκειμένου να αρχίσει το χτίσιμο του επόμενου μπλοκ. Μόλις γίνει αυτή η κλήση, το blockchain service συλλέγει όλα τα transactions που υπάρχουν στη βάση εκείνη τη στιγμή, και αφού πάρει τα δεδομένα του τελευταίου block που είναι αποθηκευμένο, ξεκινάει τη διαδικασία υπολογισμού του nonce. Μόλις βρεθεί το Nonce που καλύπτει την απαίτηση που έχει οριστεί, ξεκινάει το χτίσιμο του block, και μόλις αυτό ολοκληρωθεί διαγράφονται τα transactions που έχουν συλλεχθεί, προκειμένου να μη συμπεριληφθούν 2η φορά σε κάποιο επόμενο block.


PostgreSQL
KEYCLOAK
Document service
studentId
response
student id not valid
Error
no
student id is valid
studentId
response
User service
yes
upload document asynchronously and post student id
forward client request
Api Gateway
Blocking method converts fileParts to file
ANGULAR
Client
trigger block mining every 10 minutes
file is image
yes
convert image to pdf/A
No
get document hash
split file to chunks of 16 mb and save
object id
mongo DB
success message
documentDto
RabbitMQ
BlockChain service
create and save transaction
collect all transactions from db
get last block
calculate nonce
mine next block
previous block hash
block index
transactions
timestamp
nonce
delete collected transactions

- Εικόνα 48 -

Validate document

Κατά τη βεβαίωση γνησιότητας του εγγράφου, ο χρήστης θα ανεβάσει το αρχείο pdf του οποίου τη γνησιότητα θέλει να διαπιστώσει. Το αρχείο στέλνεται ασύγχρονα από τον client στο backend, και αφού περάσει από το api-gateway, προωθείται στο validation-service. Εκεί συλλέγονται τα fileParts με και συγκροτείται το αρχείο που έστειλε ο χρήστης. Στη συνέχεια παράγεται το hash του αρχείου και γίνεται κλήση στο blockchain-service ώστε να γίνει αναζήτηση στο blockchain του συγκεκριμένου hash. Αν το hash του εγγράφου δεν βρεθεί σε κάποιο transaction του blockchain, τότε ο χρήστης λαμβάνει μήνυμα ότι δεν ήταν δυνατή η πιστοποίηση της γνησιότητας. Σε περίπτωση που βρεθεί το hash μέσα στο blockchain, τότε γίνεται κλήση στο document-service, ώστε να βρεθούν και τα υπόλοιπα metadata του αρχείου. Αφού βρεθούν και οι πληροφορίες του εγγράφου εμφανίζεται στον χρήστη μήνυμα επιβεβαίωσης της γνησιότητας του εγγράφου.



- Εικόνα 49 -

Μελλοντικές επεκτάσεις

Η διαχείριση εγγράφων με χρήση της τεχνολογίας blockchain, επιδέχεται πολλές επεκτάσεις ως προς τον τρόπο αποθήκευσης και διαχείρισης της πληροφορίας που επεξεργάζεται.

Αποκέντρωση

Το σύστημα που έχει δημιουργηθεί, αποθηκεύει τις πληροφορίες σε μία κεντρική βάση. Με τον ίδιο τρόπο αποθηκεύεται και το ίδιο το blockchain, κάτι που σημαίνει ότι ο διαχειριστής του συστήματος έχει πλήρη πρόσβαση σε αυτό. Αντίθετα στα μεγάλα δημόσια δίκτυα, προτιμάται η αποκεντρωμένη αποθήκευση. Μία τέτοια τροποποίηση θα άλλαζε σημαντικά την ασφάλεια του συστήματος.

Σε ένα αποκεντρωμένο σύστημα δεν μπορεί να υπάρχει παρέμβαση τρίτου στα δεδομένα, επομένως αυτά δεν μπορούν να αλλάξουν με κανέναν τόπο, ούτε και να διαγραφούν, αφού η πληροφορία θα αποθηκεύεται ταυτόχρονα σε πολλούς κόμβους. Ακόμα κι αν τροποποιηθούν κακόβουλα τα δεδομένα ενός κόμβου, αυτά δεν θα μπορούν να επαληθευτούν από τους υπόλοιπους κόμβους του δικτύου.

Διόρθωση πληροφορίας

Η σύνταξη και αποθήκευση των εγγράφων, γίνεται από κάποιον υπάλληλο του πανεπιστημίου. Όλες οι ενέργειες στις οποίες απαιτείται ανθρώπινη παρέμβαση εμπεριέχουν τον κίνδυνο λάθους. Τι συμβαίνει όμως όταν ένα έγγραφο το οποίο τελικά δεν είναι έγκυρο λόγω λάθους, αποθηκεύεται στο blockchain και έτσι σε περίπτωση ελέγχου φαίνεται έγκυρο; Σε αυτή την περίπτωση θα πρέπει να υπάρχει μηχανισμός που να ελέγχει την περίπτωση, όπου το έγγραφο αφού αποθηκεύτηκε, σημάνθηκε ως άκυρο.

Δεδομένου ότι δεν μπορεί να γίνει update της πληροφορίας, θα πρέπει να μπορεί να ανέβει ξανά το έγγραφο με το σωστό περιεχόμενο, το οποίο με κάποιο μοναδικό κλειδί να δείχνει στο προηγούμενο, και με αυτόν τον τρόπο να το ακυρώνει. Ουσιαστικά πρόκειται για ισοτικότητα εγγράφου, όπου όταν αναγνωρίζεται η σύνδεση με κάποιο ήδη αποθηκευμένο έγγραφο, να αναγνωρίζεται σαν έγκυρο μόνο το τελευταίο.

Ψηφιακή υπογραφή

Τα περισσότερα ψηφιακά παραγόμενα έγγραφα επικυρώνονται μέσω ψηφιακής υπογραφής. Θα μπορούσε να αναπτυχθεί κώδικας ώστε να γίνεται διαχείριση των υπογραφών των υπαλλήλων που εμπλέκονται στην έκδοση ενός εγγράφου, ώστε να υπάρχει διασφάλιση και για την έκδοση του εγγράφου και όχι μόνο για την αποθήκευση του.

Κλειδί για προβολή

Στην εποχή του GDPR η διαφύλαξη των δεδομένων είναι πολύ σημαντική, και επομένως τα αποθηκευμένα έγγραφα δεν θα πρέπει να είναι διαθέσιμα σε όλους. Θα μπορούσε να προστεθεί στην εφαρμογή έλεγχος ώστε να είναι δυνατή ή βεβαίωση γνησιότητας και η προβολή και αποθήκευση του εγγράφου, μόνο με την άδεια του φοιτητή, χρησιμοποιώντας κάποιο προσωρινό κωδικό (OTP), ή κάποιο δυναμικά δημιουργημένο σύνδεσμο προς την εφαρμογή, το οποίο να είναι έγκυρο για μικρό χρονικό διάστημα και να γίνονται διαθέσιμα μόνο με την έγκριση του φοιτητή.

Βιβλιογραφία

1. Where Is Current Research on Blockchain Technology?—A Systematic Review
2. Proof-of-Stake (PoS) Definition
3. Blockchain - Wikipedia
4. Proof of work - Wikipedia
5. What Is a Nonce? A No-Nonsense Dive into Proof of Work - CoinCentral.
6. Blockchain Based Proof of Existence (PoE) Application for Educational Certificate Verification | SpringerLink
7. Does Proof of Existence establish Provenance? | by Maurizio Greco | Chronicled
8. Where Is Current Research on Blockchain Technology?—A Systematic Review
9. The growing list of applications and use cases of blockchain technology in business and life
10. How To Use Blockchain To Store Data [Multiple Options Available] | upGrad blog
11. Haveri, P., Rashmi, U. B., Narayan, D. G., Nagaratna, K., & Shivaraj, K. (2020, July). EduBlock: Securing Educational Documents using Blockchain Technology. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE.
12. Brunner, C., Knirsch, F., & Engel, D. (2019). SPROOF: A Platform for Issuing and Verifying Documents in a Public Blockchain. In ICISSP (pp. 15-25).
13. Introduction to Angular concepts
14. The Reactive Manifesto
15. Reactive Microservices with Spring WebFlux | by Suraj Batuwana
16. Traditional architectures versus Reactive Microservices
17. Part 1: RabbitMQ for beginners - What is RabbitMQ? - CloudAMQP.
18. Bhaskar, P., Tiwari, C. K., & Joshi, A. (2020). Blockchain in education management: present and future applications. Interactive Technology and Smart Education.
19. How Synchronous REST Turns Microservices Back into Monoliths – The New Stack
20. Communication in a microservice architecture | Microsoft Docs
21. Redis, Kafka or RabbitMQ: Which MicroServices Message Broker To Choose?.
22. GridFS — Java
23. GridFS — MongoDB Manual
24. Iteratees
25. Writing Reactive Apps with ReactiveMongo and Play, Pt. 3 - GridFS
26. What Is Optical Character Recognition (OCR)? | IBM
27. https://github.com/tesseract-ocr/docs/blob/main/tesseracticdar2007.pdf
28. https://arxiv.org/pdf/1805.09441.pdf
29. MongoDB Architecture Guide

30. Thread Behavior In Synchronous And Asynchronous Method
31. https://livebook.manning.com/book/spring-security-in-action/chapter-19/22
32. https://livebook.manning.com/book/spring-security-in-action/chapter-19/
33. PDF/A - Wikipedia
34. Decentralized vs. Centralized: A Detailed Comparison - 101 Blockchains
35. What Is Injectors in Angular?