ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Εντοπισμός και επιλογή υπηρεσιών ιστού με QoS** **Discovery and selection of web services with QoS**
Ονοματεπώνυμο Φοιτητή	**Παναγιώτης Ανδρεάτος**
Πατρώνυμο	**Γεράσιμος**
Αριθμός Μητρώου	**ΜΠΣΠ/2201**
Επιβλέπων	**Ευάγγελος Σακκόπουλος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Μάιος 2024**

Τριμελής Εξεταστική Επιτροπή

Ιωάννης Τασούλας	Διονύσιος Σωτηρόπουλος	Ευάγγελος Σακκόπουλος
Επίκουρος Καθηγητής	Επίκουρος Καθηγητής	Αναπληρωτής Καθηγητής

Περίληψη

Σκοπός της παρούσας εργασίας είναι ο σχεδιασμός και η ανάπτυξη μίας λύσης, η οποία με βάση δεδομένα που έχουν συλλεχθεί από τον χρήστη με άμεσο ή έμμεσο τρόπο καθώς και μετρήσεις ποιοτικών χαρακτηριστικών(Quality of Service, QoS) για ένα σύνολο υπηρεσιών ιστού, θα προτείνει την καταλληλότερη για αυτόν υπηρεσία.

Το παραδοτέο είναι ένα Node API το οποίο υποστηρίζει 2 διαφορετικές προσεγγίσεις για την επίλυση του συγκεκριμένου προβλήματος: μία που βασίζεται στα Συστήματα Συστάσεων με βάση το Περιεχόμενο(Content-based Recommender Systems) και μία που βασίζεται στα Συστήματα Συστάσεων που χρησιμοποιούν Συνεργατικό Φιλτράρισμα(Collaborative Filtering Recommender Systems).

Στο κείμενο αυτό που συνοδεύει την υλοποίηση, θα γίνει παρουσίαση των διάφορων τεχνολογιών-όρων που εμπλέκονται στην υλοποίηση και του τρόπου λειτουργίας του API, καθώς και ανάλυση των αλγορίθμων που χρησιμοποιήθηκαν. Τέλος, θα παρατεθούν στατιστικές μετρήσεις που πραγματοποιήθηκαν για τις 2 προσεγγίσεις και τα συμπεράσματα στα οποία αυτές οδήγησαν για την ποιότητα και την ποικιλία των παραγόμενων συστάσεων.

Abstract

The purpose of this thesis is the design and development of a solution, which based on data collected from the user in a direct or indirect way and Quality of Service (QoS) measurements for a set of web services, will propose the most suitable service for him.

The deliverable is a Node API that supports 2 different approaches to solve this problem: one based on Content-based Recommender Systems and one based on Collaborative Filtering Recommender Systems.

In this document attached to the implementation, we will present the different technologies-terms involved in the implementation and how the API works, as well as an analysis of the algorithms used. Finally, statistical measurements carried out for the 2 approaches and the conclusions they led to regarding the quality and variety of the generated recommendations will be presented.

Περιεχόμενα

Κατάλογοι στοιχείων

Κατάλογος εικόνων

Κατάλογος κομματιών κώδικα

Κατάλογος διαγραμμάτων

Κεφάλαιο 1 – Εισαγωγή

1.1 Υπηρεσία Ιστού(WS)

Οι υπηρεσίες ιστού είναι ένας τύπος λογισμικού διαδικτύου που χρησιμοποιούν τυποποιημένα πρωτόκολλα ανταλλαγής μηνυμάτων και διατίθενται από τον διακομιστή ιστού ενός παρόχου υπηρεσιών εφαρμογών για χρήση από έναν πελάτη ή άλλα προγράμματα που βασίζονται στον ιστό. Μια υπηρεσία ιστού υποστηρίζει μια συγκεκριμένη εργασία ή ένα σύνολο εργασιών. Μια επίσημη περιγραφή προσδιορίζει την υπηρεσία ιστού και περιλαμβάνει όλες τις λεπτομέρειες που απαιτούνται για την αλληλεπίδραση με αυτήν, όπως μορφές μηνυμάτων και πρωτόκολλα. Αυτό επιτρέπει σχεδόν σε οποιαδήποτε στοίβα υλικού ή λογισμικού να χρησιμοποιεί την υπηρεσία, ανεξάρτητα από την υποκείμενη πλατφόρμα και τη γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την κατασκευή της υπηρεσίας. Η ανάγκη για υπηρεσίες ιστού προέκυψε καθώς όλες οι μεγάλες πλατφόρμες είχαν πρόσβαση στο διαδίκτυο, αλλά οι διαφορετικές πλατφόρμες δεν μπορούσαν να αλληλεπιδράσουν μεταξύ τους.

Οι υπηρεσίες ιστού κατασκευάζονται με τη χρήση ανοικτών προτύπων και πρωτοκόλλων για την αλληλεπίδραση με διάφορες εφαρμογές. Τα πρωτόκολλα που χρησιμοποιούν οι υπηρεσίες ιστού περιλαμβάνουν τα εξής:

- **Extensible Markup Language**. Η XML χρησιμοποιείται για την επισήμανση, κωδικοποίηση και αποκωδικοποίηση δεδομένων.
- **Simple Object Access Protocol**. Το SOAP είναι ένα πρωτόκολλο διαδικτυακών υπηρεσιών βασισμένο στην XML που χρησιμοποιείται για τη μεταφορά δεδομένων με τη χρήση μηνυμάτων SOAP.
- **Web Services Description Language**. Η WSDL χρησιμοποιείται για να πει στην εφαρμογή-πελάτη τι περιλαμβάνεται στην υπηρεσία ιστού και πώς να συνδεθεί.
- **Representational State Transfer**. Βασισμένο στο HTTP, το REST παρέχει διαλειτουργικότητα μεταξύ συσκευών και του διαδικτύου για εργασίες που βασίζονται σε διεπαφές προγραμματισμού εφαρμογών (API).
- **Universal Description, Discovery and Integration**. Το UDDI είναι ένα πρότυπο βασισμένο στην XML που απαριθμεί και περιγράφει λεπτομερώς τις υπηρεσίες που είναι διαθέσιμες σε μια εφαρμογή.

Οι περισσότερες υπηρεσίες ιστού λειτουργούν με μια τυπική συμπεριφορά πελάτη-εξυπηρετητή. Η διαδικασία περιλαμβάνει τα ακόλουθα τρία βήματα:

1. Ο πελάτης στέλνει ένα αίτημα στον διακομιστή που περιλαμβάνει λεπτομέρειες και δεδομένα που απαιτεί η υπηρεσία ιστού. Τα δεδομένα μπορούν να είναι σε οποιαδήποτε κοινή μορφή, όπως JSON ή XML.
2. Η πλευρά του διακομιστή λαμβάνει και αυθεντικοποιεί το αίτημα, αναλύει τις απαιτούμενες λεπτομέρειες, επεξεργάζεται το αίτημα και αποκτά πρόσβαση σε τυχόν κατάλληλα αποτελέσματα.
3. Ο διακομιστής αποκτά πρόσβαση σε τυχόν κατάλληλα αποτελέσματα και τα στέλνει πίσω στην εφαρμογή-πελάτη, η οποία τα εμφανίζει σε μορφή και στυλ κατάλληλο για την εφαρμογή.

Το κλειδί σε αυτή τη διαδικασία είναι ότι η εφαρμογή-πελάτης είναι χαλαρά συνδεδεμένη(δηλαδή δεν έχει καμία απολύτως σχέση) με τον διακομιστή ή τις υπηρεσίες δεδομένων.

1.2 Ποιότητα Υπηρεσιών(QoS)

Στον σημερινό ψηφιακό κόσμο, οι υπηρεσίες ιστού διαδραματίζουν κρίσιμο ρόλο στην παροχή της υποδομής για αμέτρητες εφαρμογές και πλατφόρμες. Ωστόσο, για να είναι αυτές οι υπηρεσίες πραγματικά αποτελεσματικές, πρέπει να δίνουν προτεραιότητα στην ποιότητα των υπηρεσιών (QoS) που προσφέρουν. Οι μετρικές QoS είναι ένα σύνολο ποιοτικών μέτρων για το πόσο ελκυστική είναι μια υπηρεσία ιστού για τις εφαρμογές που ενδέχεται να την χρησιμοποιήσουν. Οι βασικές μετρικές QoS για τις υπηρεσίες ιστού είναι οι εξής:

- **Διαθεσιμότητα**. Αντιπροσωπεύει την πιθανότητα η υπηρεσία αυτή να είναι έτοιμη για άμεση χρήση. Η χαμηλή διαθεσιμότητα μιας υπηρεσίας Ιστού περιορίζει τη χρησιμότητα αυτής της υπηρεσίας.
- **Προσβασιμότητα**. Αντιπροσωπεύει την πιθανότητα ότι αυτή η υπηρεσία είναι σε θέση να διεκπεραιώσει ένα αίτημα υπηρεσίας Ιστού. Μια υπηρεσία Ιστού μπορεί να είναι διαθέσιμη αλλά να μην είναι προσβάσιμη από μια εφαρμογή-πελάτη.
- **Απόδοση**. Αντιπροσωπεύει τόσο την καθυστέρηση όσο και την απόδοση μιας υπηρεσίας Ιστού. Η καθυστέρηση περιγράφει πόσο γρήγορα (χρόνος διαδρομής σε χιλιοστά του δευτερολέπτου) ανταποκρίνεται η υπηρεσία μόλις κληθεί από την εφαρμογή-πελάτη. Η απόδοση περιγράφει τον αριθμό των αιτήσεων της υπηρεσίας Web που εξυπηρετούνται μέσα σε ένα χρονικό διάστημα.
- **Συμμόρφωση**. Αντιπροσωπεύει πόσο καλά η υπηρεσία συμμορφώνεται με τα δηλωμένα χαρακτηριστικά της υπηρεσίας.
- **Ασφάλεια**. Αντιπροσωπεύει τη χρησιμότητα και την ισχύ των μηχανισμών που παρέχει η υπηρεσία για την υποστήριξη της ασφάλειας και της ιδιωτικότητας καθ' όλη τη διάρκεια μιας αλληλεπίδρασης.
- **Αξιοπιστία**. Αποτελεί ένα συγκεντρωτικό μέτρο της συνολικής ικανότητας της υπηρεσίας να διατηρεί την ποιότητά της. Αυτή η μετρική εκφράζεται συνήθως σε όρους αποτυχιών ανά μονάδα χρόνου (π.χ. ημέρα, εβδομάδα, μήνας ή έτος) και αντιπροσωπεύει τη συνολική αξιοπιστία της υπηρεσίας ιστού.

Για να εξασφαλίσουν εξαιρετικό QoS για τις υπηρεσίες ιστού, οι οργανισμοί πρέπει να εφαρμόζουν ισχυρές στρατηγικές παρακολούθησης και βελτιστοποίησης. Τελικά, η εξαιρετική QoS όχι μόνο εξασφαλίζει την ικανοποίηση των χρηστών, αλλά συμβάλλει επίσης στην επιτυχία και τη δημοφιλία των διαδικτυακών υπηρεσιών σε ένα ολοένα και πιο ανταγωνιστικό περιβάλλον.

1.3 Στόχος εργασίας

Στόχος της παρούσας εργασίας είναι η δημιουργία ενός Node API, που θα περιλαμβάνει τις ακόλουθες λειτουργίες:

- **Φόρτωμα δεδομένων.** Τα δεδομένα αυτά θα είναι αποθηκευμένα σε τοπικά αρχεία excel και θα διαβάζονται από το API κατά την εκκίνηση της λειτουργίας του.
- **Εγγραφή χρήστη.** Ο χρήστης θα πραγματοποιεί ένα http post request σε συγκεκριμένο endpoint του API παρέχοντας στο request body τα στοιχεία σύνδεσης του. Στη συνέχεια, θα επιχειρείται η αποθήκευση των στοιχείων αυτών σε μία βάση δεδομένων. Τέλος, στο χρήστη θα επιστρέφεται ένα μήνυμα που θα τον ενημερώνει για την έκβαση της προσπάθειας αυτής.
- **Είσοδος χρήστη.** Ο χρήστης θα πραγματοποιεί ένα http post request σε συγκεκριμένο endpoint του API παρέχοντας στο request body τα στοιχεία σύνδεσης με τα οποία εγγράφηκε στο σύστημα. Στη συνέχεια, θα ελέγχεται κατά πόσο υπάρχουν αποθηκευμένα στοιχεία σύνδεσης στη βάση δεδομένων που να ταυτίζονται με αυτά που έδωσε ο χρήστης. Αν ναι, ο χρήστης θα αποκτά πρόσβαση σε όλες τις λειτουργίες του API. Διαφορετικά, θα ενημερώνεται για την αποτυχία εισόδου.
- **Παραγωγή συστάσεων με βάση το περιεχόμενο.** Ο χρήστης θα πραγματοποιεί ένα http post request σε συγκεκριμένο endpoint του API παρέχοντας στο request body τις τιμές που επιθυμεί να έχουν συγκεκριμένα ποιοτικά χαρακτηριστικά των υπηρεσιών ιστού που θα του προταθούν. Στη συνέχεια, η εφαρμογή συνδυάζοντας τις τιμές αυτές με μέρος των δεδομένων που φορτώθηκαν από τα excel αρχεία και εφαρμόζοντας κατάλληλους αλγορίθμους θα παράγει συστάσεις. Τέλος, οι συστάσεις αυτές θα επιστρέφονται στο χρήστη.
- **Παραγωγή συστάσεων με συνεργατικό φιλτράρισμα.** Ο χρήστης θα έχει πρόσβαση στη λειτουργία αυτή μόνο εφόσον έχει πραγματοποιήσει είσοδο στο σύστημα. Εφόσον έχει πρόσβαση, η έναρξη της λειτουργίας θα πυροδοτείται με ένα http get request σε συγκεκριμένο endpoint του API. Στη συνέχεια, η εφαρμογή, αξιοποιώντας μέρος των δεδομένων που φορτώθηκαν από τα excel αρχεία και εφαρμόζοντας κατάλληλους αλγορίθμους, θα παράγει συστάσεις. Τέλος, οι συστάσεις αυτές θα επιστρέφονται στο χρήστη.

1.4 Δομή εργασίας

Στα επόμενα κεφάλαια παρατίθεται το θεωρητικό πλαίσιο για τα ΣΣ, παρουσιάζεται ο τρόπος υλοποίησης-εκτέλεσης των λειτουργιών του API και διατυπώνονται τα συμπεράσματα στα οποία κατέληξε η παρούσα εργασία. Πιο συγκεκριμένα:

- **Κεφάλαιο 2.** Παραθέτει μία θεωρητική επισκόπηση των ΣΣ και των τεχνικών παραγωγής συστάσεων.
- **Κεφάλαιο 3.** Παρουσιάζει τον τρόπο δημιουργίας και την δομή όλων των μερών που συμμετέχουν στην υλοποίηση.
- **Κεφάλαιο 4.** Επεξηγεί τον τρόπο υλοποίησης των λειτουργιών του API παραθέτοντας διαγράμματα και εξηγώντας αναλυτικά τον κώδικα που έχει υλοποιηθεί για το σκοπό αυτό.
- **Κεφάλαιο 5.** Παρουσιάζει τον τρόπο εκτέλεσης των λειτουργιών του API.
- **Κεφάλαιο 6.** Παραθέτει τις στατιστικές μετρήσεις που πραγματοποιήθηκαν για τα ΣΣ της υλοποίησης καθώς και τα συμπεράσματα που εξάχθηκαν με βάση αυτές.

Κεφάλαιο 2 – Επισκόπηση ΣΣ

Στο Διαδίκτυο, όπου ο αριθμός των επιλογών είναι δυσθεώρητος, υπάρχει ανάγκη φιλτραρίσματος, ιεράρχησης και αποτελεσματικής παροχής σχετικών πληροφοριών, προκειμένου να αμβλυνθεί το πρόβλημα της υπερφόρτωσης πληροφοριών, το οποίο έχει δημιουργήσει δυνητικό πρόβλημα σε πολλούς χρήστες. Τα συστήματα συστάσεων επιλύουν αυτό το πρόβλημα αναζητώντας σε μεγάλο όγκο δυναμικά παραγόμενων πληροφοριών για την παροχή εξατομικευμένου περιεχομένου και υπηρεσιών στους χρήστες. Τα οφέλη που προσφέρουν τα συστήματα αυτά είναι πολλαπλά και για αυτό η ανάγκη να χρησιμοποιηθούν αποτελεσματικές και ακριβείς τεχνικές συστάσεων είναι επιτακτική.

Το σύστημα συστάσεων ορίζεται ως μια στρατηγική λήψης αποφάσεων για χρήστες σε σύνθετα περιβάλλοντα πληροφοριών. Έχουν αναπτυχθεί διάφορες προσεγγίσεις για τη δημιουργία συστημάτων συστάσεων που μπορούν να χρησιμοποιήσουν συνεργατικό φιλτράρισμα, φιλτράρισμα βάσει περιεχομένου ή υβριδικό φιλτράρισμα. Το συνεργατικό φιλτράρισμα συνιστά στοιχεία εντοπίζοντας άλλους χρήστες με παρόμοιο γούστο· χρησιμοποιεί τη γνώμη τους για να συστήσει αντικείμενα στον ενεργό χρήστη. Το φιλτράρισμα βάσει περιεχομένου συνιστά στοιχεία με βάση τα χαρακτηριστικά και τις πληροφορίες του χρήστη αγνοώντας τις συνεισφορές άλλων χρηστών. Το υβριδικό φιλτράρισμα συνδυάζει δύο ή περισσότερες τεχνικές φιλτραρίσματος με ποικίλους τρόπους, προκειμένου να αυξηθούν η ακρίβεια και η απόδοση των συστημάτων συστάσεων.

Η διαδικασία παραγωγής συστάσεων περιλαμβάνει τις ακόλουθες φάσεις:

1. **Φάση συλλογής πληροφοριών.** Σε αυτή τη φάση συλλέγονται οι απαραίτητες πληροφορίες των χρηστών για τη δημιουργία ενός προφίλ χρήστη ή μοντέλου για τις εργασίες πρόβλεψης. Οι πληροφορίες αυτές περιλαμβάνουν χαρακτηριστικά του χρήστη, συμπεριφορές και περιεχόμενο το οποίο προσπελάζει. Για τη συλλογή των πληροφοριών αυτών χρησιμοποιούνται οι ακόλουθες τεχνικές:
   - **Άμεση ανατροφοδότηση.** Το σύστημα προτρέπει τον χρήστη να δώσει βαθμολογίες για τα στοιχεία προκειμένου να κατασκευάσει και να βελτιώσει το μοντέλο του. Η τεχνική αυτή προσφέρει πιο αξιόπιστα δεδομένα, αλλά απαιτεί προσπάθεια από το χρήστη.
   - **Έμμεση ανατροφοδότηση.** Το σύστημα συμπεραίνει αυτόματα τις προτιμήσεις του χρήστη παρακολουθώντας τις διάφορες ενέργειες του, όπως το ιστορικό των αγορών, ιστορικό πλοήγησης κτλ. Η τεχνική αυτή δεν απαιτεί προσπάθεια από το χρήστη αλλά προσφέρει λιγότερο αξιόπιστα δεδομένα.
   - **Υβριδική ανατροφοδότηση.** Συνδυάζει τα πλεονεκτήματα των 2 προηγούμενων τεχνικών προκειμένου να ελαχιστοποιηθούν οι αδυναμίες τους. Αυτό μπορεί να επιτευχθεί πχ επιτρέποντας στο χρήστη να παρέχει άμεση ανατροφοδότηση μόνο όταν το επιθυμεί.
2. **Φάση εκμάθησης.** Σε αυτή τη φάση εφαρμόζεται ένας αλγόριθμος εκμάθησης για το φιλτράρισμα και την εκμετάλλευση των δεδομένων που συλλέγονται στην προηγούμενη φάση.
3. **Φάση πρόβλεψης/σύστασης.** Συνιστά ή προβλέπει τι είδους προϊόντα μπορεί να προτιμά ο χρήστης.

2.1 Φιλτράρισμα βάσει περιεχομένου

Η τεχνική φιλτραρίσματος βάσει περιεχομένου είναι ένας αλγόριθμος που εξαρτάται από τον τομέα και δίνει μεγαλύτερη έμφαση στην ανάλυση των χαρακτηριστικών των προϊόντων προκειμένου να παράξει προβλέψεις. Στην τεχνική αυτή, η σύσταση γίνεται με βάση το προφίλ του χρήστη χρησιμοποιώντας χαρακτηριστικά που εξάγονται από το περιεχόμενο προϊόντων τα οποία έχουν αξιολογηθεί στο παρελθόν από το χρήστη. Τα προϊόντα που συνιστώνται στο χρήστη είναι αυτά με τη μεγαλύτερη συσχέτιση με προϊόντα που έχει αξιολογήσει θετικά στο παρελθόν. Χρησιμοποιεί διαφορετικούς τύπους μοντέλων για την εύρεση της ομοιότητας μεταξύ των προϊόντων προκειμένου να παράγει ουσιαστικές συστάσεις. Παραδείγματα τέτοιων μοντέλων είναι το Term Frequency Inverse Document Frequency(TF/IDF), ο ταξινομητής Naive Bayes, δέντρα αποφάσεων και νευρωνικά δίκτυα. Η διατύπωση συστάσεων γίνεται με εκμάθηση του υποκείμενου μοντέλου είτε με στατιστική ανάλυση είτε με τεχνικές μηχανικής μάθησης. Παραδείγματα συστημάτων που χρησιμοποιούν αυτή την τεχνική είναι το News Dude, το CiteSeer και το LIBRA.

Η ανωτέρω τεχνική έχει πολλά πλεονεκτήματα αλλά και σημαντικά μειονεκτήματα. Τα πλεονεκτήματα της είναι:

- Δυνατότητα σύστασης νέων προϊόντων ακόμα και αν δεν υπάρχουν αξιολογήσεις από τους χρήστες για αυτά.
- Άμεση προσαρμογή σε αλλαγές στις προτιμήσεις των χρηστών.
- Διαχείριση καταστάσεων όπου διαφορετικοί χρήστες δεν χρησιμοποιούν τα ίδια προϊόντα, αλλά μόνο πανομοιότυπα προϊόντα σύμφωνα με τα εγγενή τους χαρακτηριστικά.
- Διασφάλιση ιδιωτικότητας χρηστών.
- Παροχή εξηγήσεων για το πως παράγονται οι συστάσεις στους χρήστες.

Τα μειονεκτήματα της είναι:

- Η αποτελεσματικότητα της εξαρτάται από τη διαθεσιμότητα περιγραφικών δεδομένων για τα προϊόντα.
- Υπερβολική εξειδίκευση περιεχομένου. Οι χρήστες λαμβάνουν συστάσεις μόνο για προϊόντα που είναι παρόμοια με αυτά που καθορίζονται στα προφίλ τους.

2.2 Συνεργατικό φιλτράρισμα

Το συνεργατικό φιλτράρισμα είναι μία τεχνική πρόβλεψης ανεξάρτητη από τον τομέα για υλικό που δεν μπορεί να περιγραφεί εύκολα και επαρκώς από μεταδεδομένα(π.χ. ταινίες). Η τεχνική αυτή λειτουργεί με τη δημιουργία μιας βάσης δεδομένων(μήτρα χρήστη-προϊόντος) των προτιμήσεων των χρηστών για τα αντικείμενα. Στη συνέχεια, ταιριάζει χρήστες με σχετικά ενδιαφέροντα και προτιμήσεις υπολογίζοντας ομοιότητες μεταξύ των προφίλ τους. Τέτοιοι χρήστες δημιουργούν μια ομάδα που ονομάζεται γειτονιά. Ένα χρήστης λαμβάνει συστάσεις για τα αντικείμενα που δεν έχει αξιολογήσει πριν, αλλά τα οποία έχουν ήδη αξιολογηθεί θετικά από τους χρήστες στη γειτονιά του. Η τεχνική αυτή μπορεί να παράξει 2 ειδών συστάσεις: πρόβλεψη της αξιολόγησης του χρήστη για ένα προϊόν και έναν κατάλογο με τα κορυφαία N προϊόντα που θα αρέσουν περισσότερο στον χρήστη. Παραδείγματα συστημάτων που χρησιμοποιούν αυτή την τεχνική είναι το Ringo, το GroupLens και το Amazon. Η τεχνική συνεργατικού φιλτραρίσματος μπορεί να χωριστεί σε δύο κατηγορίες: βάσει μνήμης και βάσει μοντέλου.

Στο συνεργατικό φιλτράρισμα βάσει μνήμης τα στοιχεία που έχουν βαθμολογηθεί από τον χρήστη στο παρελθόν παίζουν ένα σχετικό ρόλο στην αναζήτηση ενός γείτονα. Αυτό μπορεί να επιτευχθεί με τεχνικές είτε βασισμένες στο χρήστη είτε βασισμένες σε προϊόντα. Οι τεχνικές βασισμένες στο χρήστη ακολουθούν τα εξής βήματα:

1. Υπολογισμός της ομοιότητας μεταξύ των χρηστών συγκρίνοντας τις αξιολογήσεις τους για το ίδιο προϊόν.
2. Υπολογισμός της προβλεπόμενης βαθμολογίας για ένα προϊόν από τον ενεργό χρήστη ως σταθμισμένος μέσος όρος των αξιολογήσεων του προϊόντος από χρήστες παρόμοιους με τον ενεργό χρήστη. Τα βάρη είναι οι ομοιότητες των χρηστών με το προϊόν-στόχο.

Οι τεχνικές βασισμένες σε προϊόντα χρησιμοποιούν την ομοιότητα μεταξύ αντικειμένων και όχι την ομοιότητα μεταξύ χρηστών. Υπολογίζουν τις προβλέψεις ακολουθώντας τα εξής βήματα:

1. Δημιουργία μοντέλου με ομοιότητες προϊόντων ανακτώντας όλα τα προϊόντα που έχουν αξιολογηθεί από έναν ενεργό χρήστη.
2. Καθορισμός ομοιότητας ανακτηθέντων προϊόντων με το προϊόν-στόχο.
3. Επιλογή των k πιο όμοιων προϊόντων.
4. Υπολογισμός της προβλεπόμενης βαθμολογίας για το προϊόν-στόχος ως σταθμισμένος μέσος όρος της αξιολόγησης των ενεργών χρηστών για τα k επιλεγμένα προϊόντα.

Στο συνεργατικό φιλτράρισμα βάσει μοντέλου οι προηγούμενες αξιολογήσεις χρησιμοποιούνται για την εκμάθηση ενός μοντέλου. Η διαδικασία δημιουργίας του μοντέλου μπορεί να γίνει με τη χρήση τεχνικών μηχανικής μάθησης ή εξόρυξης δεδομένων. Αυτές οι τεχνικές μπορούν να προτείνουν γρήγορα ένα σύνολο προϊόντων, καθώς χρησιμοποιούν ένα προϋπολογισμένο μοντέλο και έχουν αποδείξει ότι παράγουν παρόμοια αποτελέσματα με τις τεχνικές συστάσεων με βάση τη γειτονιά. Παραδείγματα τέτοιων τεχνικών είναι οι τεχνικές μείωσης διαστάσεων(π.χ. Singular Value Decomposition), τεχνικές συμπλήρωσης μητρών, μέθοδοι λανθάνουσας σημασιολογίας, παλινδρόμηση και ομαδοποίηση.

Το συνεργατικό φιλτράρισμα έχει πολλά πλεονεκτήματα αλλά και σημαντικά προβλήματα. Τα πλεονεκτήματά του είναι:

- Μπορεί να αποδώσει σε τομείς όπου δεν υπάρχει πολύ υλικό για τα προϊόντα καθώς και όπου το υλικό αυτό είναι δύσκολο να αναλυθεί από ένα υπολογιστικό σύστημα.
- Έχει την δυνατότητα να παρέχει τυχαία συστάσεις, κάτι που σημαίνει ότι μπορεί να προτείνει προϊόντα που ενδιαφέρουν το χρήστη χωρίς το υλικό τους να είναι αποθηκευμένο στο προφίλ του.

Τα προβλήματα του είναι:

- **Πρόβλημα ψυχρής εκκίνησης.** Αυτό αναφέρεται σε μια κατάσταση όπου ένα σύστημα συστάσεων δεν έχει επαρκείς πληροφορίες σχετικά με ένα χρήστη ή για ένα προϊόν προκειμένου να κάνει σχετικές προβλέψεις.
- **Πρόβλημα αραιότητας δεδομένων.** Αυτό είναι το πρόβλημα που εμφανίζεται λόγω της έλλειψης επαρκών πληροφοριών, δηλαδή μόνο όταν ελάχιστα από το συνολικό αριθμό προϊόντων που είναι διαθέσιμα στη βάση δεδομένων έχουν αξιολογηθεί από χρήστες.
- **Επεκτασιμότητα.** Ο υπολογισμός συνήθως αυξάνεται γραμμικά με τον αριθμό των χρηστών και των προϊόντων.
- **Συνωνυμία.** Είναι η τάση των πολύ παρόμοιων προϊόντων να έχουν διαφορετικά ονόματα ή καταχωρήσεις.

2.3 Υβριδικό φιλτράρισμα

Το υβριδικό φιλτράρισμα είναι μία τεχνική που συνδυάζει διαφορετικές τεχνικές συστάσεων κάτι που συμβάλει στην αποφυγή ορισμένων περιορισμών και προβλημάτων που έχουν τα απλά συστήματα συστάσεων και εν τέλει στην καλύτερη βελτιστοποίηση του συστήματος. Ο συνδυασμός των προσεγγίσεων μπορεί να γίνει με έναν από τους ακόλουθους τρόπους:

- **Σταθμισμένη υβριδοποίηση.** Συνδυάζει τα αποτελέσματα διαφορετικών συστημάτων συστάσεων για τη δημιουργία μιας λίστας συστάσεων ή προβλέψεων ενσωματώνοντας τις βαθμολογίες από κάθε μία από τις χρησιμοποιούμενες τεχνικές μέσω ενός γραμμικού τύπου.
- **Εναλλασσόμενη υβριδοποίηση.** Το σύστημα αλλάζει σε μία από τις τεχνικές παραγωγής συστάσεων σύμφωνα με μια ευρετική που αντανακλά την ικανότητα του αντίστοιχου συστήματος συστάσεων να παράξει μια καλή εκτίμηση.
- **Κλιμακωτή υβριδοποίηση.** Εφαρμόζει μια επαναληπτική βελτίωση για την κατασκευή μιας σειράς προτίμησης μεταξύ διαφορετικών προϊόντων. Οι συστάσεις μιας τεχνικής βελτιώνονται από μια άλλη τεχνική συστάσεων.
- **Μικτή υβριδοποίηση.** Τα μικτά υβρίδια συνδυάζουν αποτελέσματα διαφορετικών τεχνικών σύστασης ταυτόχρονα, αντί να έχουν μία μόνο σύσταση ανά προϊόν. Κάθε στοιχείο έχει πολλαπλές συστάσεις που σχετίζονται με αυτό από διαφορετικές τεχνικές συστάσεων.
- **Συνδυασμός χαρακτηριστικών.** Τα χαρακτηριστικά που παράγονται από μια συγκεκριμένη τεχνική συστάσεων τροφοδοτούνται σε άλλη τεχνική συστάσεων.
- **Βελτίωση χαρακτηριστικών.** Η τεχνική χρησιμοποιεί τις αξιολογήσεις και άλλες πληροφορίες που παρήχθησαν από την προηγούμενο σύστημα συστάσεων και απαιτεί επίσης πρόσθετη λειτουργικότητα.
- **Μετα-επίπεδο.** Το εσωτερικό μοντέλο που παράγεται από μία τεχνική συστάσεων χρησιμοποιείται ως είσοδος για μια άλλη.

Κεφάλαιο 3 – Παρουσίαση τεχνολογιών εφαρμογής

3.1 Datasets

Στο πλαίσιο της παρούσας εργασίας δημιουργήθηκαν δύο datasets, ένα για την παραγωγή συστάσεων με φιλτράρισμα βάσει περιεχομένου και ένα για την παραγωγή συστάσεων με συνεργατικό φιλτράρισμα. Η διαδικασία αυτή πραγματοποιήθηκε με την έκδοση 2401 του Microsoft Excel.

3.1.1 Dataset για φιλτράρισμα βάσει περιεχομένου

Ως βάση χρησιμοποιήθηκε το dataset που βρίσκεται στον ακόλουθο σύνδεσμο: http://wsdream.github.io/dataset/wsdream_dataset1.html. Το εν λόγω dataset είναι σε μορφή txt, περιγράφει αποτελέσματα αξιολόγησης QoS σε πραγματικό κόσμο για 5.825 υπηρεσίες ιστού από 339 χρήστες και περιλαμβάνει τους ακόλουθους πίνακες:

- **userlist:** περιέχει πληροφορίες για τους 339 χρήστες. Πιο συγκεκριμένα, για κάθε χρήστη παρατίθενται τα ακόλουθα: User ID, IP Address, Country, Continent, AS, Latitude, Longitude, Region, City.
- **wslist:** περιέχει πληροφορίες για τις 5.825 υπηρεσίες ιστού. Πιο συγκεκριμένα, για κάθε υπηρεσία ιστού παρατίθενται τα ακόλουθα: Service ID, WSDL Address, Service Provider, IP Address, Country, Continent, AS, Latitude, Longitude, Region, City.
- **rtMatrix:** 339 x 5825 πίνακας με τον χρόνο απόκρισης της κάθε υπηρεσίας
- ιστού ανά χρήστη.
- **tpMatrix:** 339 x 5825 πίνακας με την απόδοση της κάθε υπηρεσίας ιστού ανά χρήστη.

Αξιοποιώντας τους πίνακες wslist, rtMatrix και tpMatrix δημιουργήθηκε ένα dataset υπηρεσιών ιστού όπου για κάθε μία από αυτές περιέχονται πληροφορίες και αποτελέσματα αξιολόγησης QoS. Πιο συγκεκριμένα:

- Οι πίνακες wslist, rtMatrix και tpMatrix μεταφέρθηκαν στο excel ακολουθώντας τις οδηγίες που περιέχονται στον ακόλουθο σύνδεσμο: https://www.adinstruments.com/support/knowledge-base/how-can-comma-separated-list-be-converted-cells-column-lt .
- Από τις 5.825 υπηρεσίες ιστού επιλέχθηκαν 981 και από τις πληροφορίες που περιέχονται στον πίνακα wslist για αυτές χρησιμοποιήθηκαν όλες εκτός από το Region και το City.
- Για κάθε μία από τις επιλεγμένες υπηρεσίες ιστού προστέθηκαν τα πεδία Response Time, Throughput και Operations.
- Το πεδίο Response Time είναι ο χρόνος απόκρισης μίας υπηρεσίας ιστού. Η τιμή του πεδίου αυτού ανά επιλεγμένη υπηρεσία ιστού είναι ο μέσος όρος της αντίστοιχης στήλης του πίνακα rtMatrix.
- Το πεδίο Throughput είναι η απόδοση μίας υπηρεσίας ιστού. Η τιμή του πεδίου αυτού ανά επιλεγμένη υπηρεσία ιστού είναι ο μέσος όρος της αντίστοιχης στήλης του πίνακα tpMatrix.
- Το πεδίο Operations περιγράφει τις λειτουργίες που επιτελεί μία υπηρεσία ιστού. Επειδή για τις υπηρεσίες ιστού του dataset δεν υπήρχε κάποιο έγγραφο που να περιγράφει τις λειτουργίες που επιτελούν, η τιμή του πεδίου αυτού ανά επιλεγμένη υπηρεσία ιστού τέθηκε με βάση το WSDL Address.

Service ID	WSDL Address	Service Provider	IP Address	Country	IP No.	AS	Latitude	Longitude	Throughput	Response Time	Operations
0	http://ewave.no-ip.com/ECallws/CinemaData.asmx?WSDL	no-ip.com	8.23.224.110	United States	135782510	AS3356 Level 3 Communications	38	-97	1.52	2.07	Cinema Data
1	http://ewave.no-ip.com/ECallws/StadiumSinchronization.a	no-ip.com	8.23.224.110	United States	135782510	AS3356 Level 3 Communications	38	-97	13.95	0.31	Stadium Synchronization
2	http://ewave.no-ip.com/ECallWS/CinemaSinchronization.a	no-ip.com	8.23.224.110	United States	135782510	AS3356 Level 3 Communications	38	-97	20.18	0.35	Cinema Synchronization
3	http://ewave.no-ip.com/ECallws/StadiumData.asmx?WSDI	no-ip.com	8.23.224.110	United States	135782510	AS3356 Level 3 Communications	38	-97	7.21	0.31	Stadium Data
4	http://ewave.no-ip.com/ECallws/BuyerData.asmx?WSDL	no-ip.com	8.23.224.110	United States	135782510	AS3356 Level 3 Communications	38	-97	7.24	0.30	Buyer Data
5	http://aquest.dyndns.org/CaptchaAudioWS/CaptchaAudio.	dyndns.org	204.13.248.1	United States	3.423E+09	AS33517 Dynamic Network Servic	40.7904	-74.0246	9.17	0.90	Captcha Audio
6	http://www.utn.edu.ar/WebServices/ListaTics.asmx?WSDL	utn.edu.ar	null	Argentina	0	null	-45.588	-69.07	2.56	1.21	Lista Tics
7	http://www.utn.edu.ar/WebServices/WSAcademico.asmx?	utn.edu.ar	null	Argentina	0	null	-45.588	-69.07	120.90	1.03	Academiko
8	http://200.45.113.149/SC.BlackBerryClients.WS/Service.as	200.45.113.149	200.45.113.1	Argentina	3.358E+09	AS7303 Telecom Argentina S.A.	-34.588	-58.6725	4.09	0.52	BlackBerry Clients
9	http://xml.dev.hoteldo.com/HotelDoInterface.asmx?WSDL	hoteldo.com	200.59.145.1	Argentina	3.359E+09	AS11664 Techtel LMDS Comunica	-34.471	-58.5078	3.44	1.27	Hotel Do Interface
10	http://www.librosar.com.ar/portal/servicioonix.asmx?WSI	librosar.com.ar	null	Argentina	0	null	null	null	3.65	0.99	Servicionix
11	http://ba.mobilenik.com.ar/dbbrowser/dbbrowser.asmx?W	mobilenik.com.ar	null	Argentina	0	null	null	null	2.46	1.09	Db Browser
12	http://www.mslatam.com/latam/msdn/comunidad/dce/es	mslatam.com	65.55.39.10	Argentina	1.094E+09	AS8075 Microsoft Corp	47.6801	-122.1206	18.03	3.37	FSD Statistics
13	http://www.neodynamic.com/Products/Demos/BCWebSar	neodynamic.com	190.2.60.187	Argentina	3.188E+09	AS16814 S.A.	-32.947	-60.6393	11.48	1.71	Barcode Professional
14	http://www.ssat.com.ar/googleearth1.1/service.asmx?wsc	ssat.com.ar	200.127.117.	Argentina	3.364E+09	AS10481 Prima S.A.	-34.777	-58.4069	1.67	0.87	Google Earth
15	http://www.transportesjoselito.com/atlas/modal/TiempoS	transportesjoselito	200.58.118.1	Argentina	3.359E+09	AS27823 Dattatec.com	-32.947	-60.6393	2.70	1.25	Tiempo Service
16	http://wsatebara.com.ar/ServicioWeb.asmx?WSDL	wsatebara.com.ar	190.247.40.1	Argentina	3.204E+09	AS10318 S.A.	-34.588	-58.6725	7.83	1.67	Servicio Web
18	http://www.enterpriseconnect.gov.au/_vti_bin/BusinessDa	enterpriseconnect.	210.193.179.	Australia	3.536E+09	AS17477 Macquarie Telecom	-27	133	4.65	1.13	Business Data Catalog
19	http://www.enterpriseconnect.gov.au/_vti_bin/People.asn	enterpriseconnect.	210.193.179.	Australia	3.536E+09	AS17477 Macquarie Telecom	-27	133	11.29	0.29	People
20	http://www.enterpriseconnect.gov.au/_vti_bin/Authentica	enterpriseconnect.	210.193.179.	Australia	3.536E+09	AS17477 Macquarie Telecom	-27	133	11.23	0.31	Authentication
21	http://www.industry.gov.au/_vti_bin/BusinessDataCatalog	industry.gov.au	210.193.176.	Australia	3.536E+09	AS17477 Macquarie Telecom	-35.276	149.1344	3.77	1.36	Business Data Catalog
22	http://www.industry.gov.au/_vti_bin/People.asmx?wsdl	industry.gov.au	210.193.176.	Australia	3.536E+09	AS17477 Macquarie Telecom	-35.276	149.1344	11.22	0.35	People
23	http://www.industry.gov.au/_vti_bin/Authentication.asmx	industry.gov.au	210.193.176.	Australia	3.536E+09	AS17477 Macquarie Telecom	-35.276	149.1344	11.31	0.31	Authentication
24	http://www.microsoftsharepoint.com/venue/_vti_bin/Shar	microsoftsharepoir	203.19.66.79	Australia	3.407E+09	AS9268 Over The Wire Pty Ltd	-27	133	4.59	1.01	Sharepoint Email
25	http://www.microsoftsharepoint.com/venue/_vti_bin/Busi	microsoftsharepoir	203.19.66.79	Australia	3.407E+09	AS9268 Over The Wire Pty Ltd	-27	133	11.22	0.30	Business Data Catalog
26	http://www.microsoftsharepoint.com/agenda/_vti_bin/Au	microsoftsharepoir	203.19.66.79	Australia	3.407E+09	AS9268 Over The Wire Pty Ltd	-27	133	11.21	0.30	Authentication

Εικόνα 1: Dataset για φιλτράρισμα βάσει περιεχομένου

3.1.2 Datasets για συνεργατικό φιλτράρισμα

Ως βάση χρησιμοποιήθηκε το dataset που βρίσκεται στον ακόλουθο σύνδεσμο: https://qwsdata.github.io/qws2.html. Το εν λόγω dataset είναι σε μορφή txt και περιλαμβάνει μετρήσεις QoS που έγιναν σε 2.507 υπηρεσίες ιστού. Πιο συγκεκριμένα, για κάθε μία από αυτές παρέχονται τα ακόλουθα στοιχεία:

- **Response Time:** χρόνος αποστολής μηνύματος και λήψης απάντησης.
- **Availability:** αριθμός επιτυχημένων κλήσεων / συνολικές κλήσεις.
- **Throughput:** συνολικός αριθμός κλήσεων για μια δεδομένη χρονική περίοδο.
- **Successability:** αριθμός απαντήσεων / αριθμός αιτήσεων.
- **Reliability:** λόγος του αριθμού των μηνυμάτων σφάλματος προς το σύνολο των μηνυμάτων.
- **Compliance:** ο βαθμός στον οποίο ένα έγγραφο WSDL ακολουθεί τις προδιαγραφές WSDL.
- **Best Practices:** ο βαθμός στον οποίο μια υπηρεσία Ιστού ακολουθεί το WS-I Basic Profile.
- **Latency:** χρόνος που χρειάζεται ο διακομιστής για να επεξεργαστεί μια συγκεκριμένη αίτηση.
- **Documentation:** μέτρο τεκμηρίωσης (δηλ. ετικέτες περιγραφής) στο WSDL.
- **Service Name:** όνομα της υπηρεσίας ιστού.
- **WSDL Address:** διεύθυνση του αρχείου WSDL στο διαδίκτυο.

Αξιοποιώντας το παραπάνω dataset δημιουργήθηκαν ένα dataset με κατηγοριοποιημένες και βαθμολογημένες υπηρεσίες ιστού καθώς επίσης και τέσσερα datasets με αξιολογήσεις χρηστών για υπηρεσίες ιστού. Για το dataset με κατηγοριοποιημένες και βαθμολογημένες υπηρεσίες ιστού ακολουθήθηκε η εξής διαδικασία:

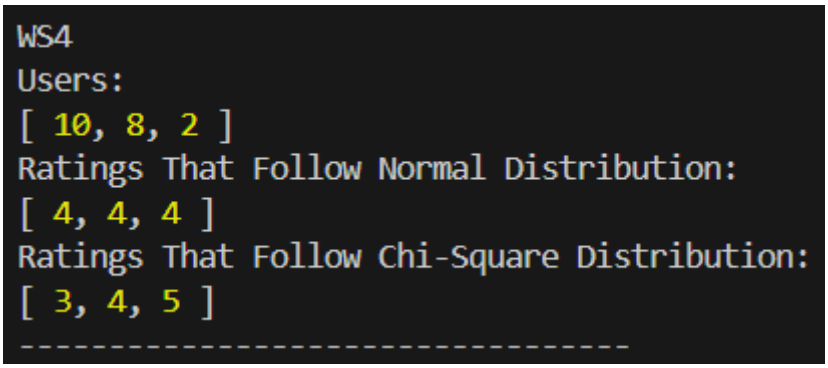
- Το QWS dataset μεταφέρθηκε στο excel ακολουθώντας τις οδηγίες που περιέχονται στον ακόλουθο σύνδεσμο: https://www.adinstruments.com/support/knowledge-base/how-can-comma-separated-list-be-converted-cells-column-lt.
- Από τις 2.507 υπηρεσίες ιστού επιλέχθηκαν 117.
- Οι επιλεγμένες υπηρεσίες ιστού κατατάχθηκαν σε μία από τις ακόλουθες κατηγορίες: Weather, Stock, Hotel, Bank, News, Music, SMS, Credit Card, Phone και Email. Η κατηγοριοποίηση έγινε με βάση τις τιμές των πεδίων Service Name και WSDL Address.
- Για τις επιλεγμένες υπηρεσίες ιστού υπολογίσθηκε μια βαθμολογία που κυμαίνεται από 1 έως 5. Η διαδικασία ανά υπηρεσία ιστού ήταν η εξής:
    1. Υπολογίσθηκε ο μέσος όρος των τιμών των πεδίων Availability, Successability και Reliability. Η υπολογισθείσα τιμή κυμαινόταν στο [1, 100].
    2. Η παραπάνω τιμή ανάχθηκε στο [1, 5] με τον εξής μετασχηματισμό: $[1, 100] \xrightarrow{*5} [5, 500] \xrightarrow{/100} [1, 5]$
    3. Η τιμή που προέκυψε από το βήμα 2 στρογγυλοποιήθηκε χρησιμοποιώντας τη συνάρτηση ROUND του excel. Το τελικό αποτέλεσμα ήταν μία ακέραια τιμή από 1 έως 5.
- Δημιουργήθηκε ένα dataset με τις επιλεγμένες υπηρεσίες ιστού. Για κάθε μία από αυτές αναφέρεται το Service Name, το WSDL Address, η βαθμολογία και η κατηγορία που κατατάχθηκαν.

Service Name	WSDL Address	Rating	Category
cweather	http://www.mail-archive.com/axis-user@xml.apache.org/msg02797/cweather.wsdl	3	Weather
ndfdXML	http://www.weather.gov/forecasts/xml/SOAP_server/ndfdXMLserver.php?wsdl	4	Weather
PluralsightWeather	http://www.pluralsight.com/services/weatherservice/weatherservice.asmx?WSDL	4	Weather
ndfdXML	http://weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl	4	Weather
DOTSFastWeather	http://trial.serviceobjects.com/fw/FastWeather.asmx?WSDL	4	Weather
WeatherStationService	http://www.pathfinder-xml.com/development/WSDL/WeatherStationService.wsdl	4	Weather
DOTSFastWeather	http://ws2.serviceobjects.net/fw/fastweather.asmx?wsdl	4	Weather
ndfdXML	http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl)	4	Weather
Service	http://ejse.com/WeatherService/Service.asmx?wsdl	4	Weather
GlobalWeather	http://www.webservicex.com/globalweather.asmx?WSDL	4	Weather
ndfdXML	http://weather.gov/forecasts/xml/SOAP_server/ndfdXMLserver.php?wsdl	4	Weather
Weather	http://www.deeptraining.com/webservices/weather.asmx?wsdl	4	Weather
BulportWeatherService	http://services.bulport.com/weather/weather.wsdl	4	Weather
USWeather	http://www.webservicex.net/usweather.asmx?WSDL	4	Weather
Service	http://www.ejse.com/WeatherService/Service.asmx?wsdl	4	Weather
ndfdXML	http://oscar.snapps.com/portal.nsf/WeatherService?WSDL	4	Weather
DelayedStockQuote	http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx?wsdl	3	Stock
StockServices	http://www.html2xml.nl/Services/Stocks/Version1/StockServices.asmx?wsdl	4	Stock
StockQuote	http://www.webservicex.net/stockquote.asmx?WSDL	5	Stock
stock	http://www.wopos.com/webservice/Stock.asmx?wsdl	4	Stock
RealTimeStockQuotes	http://ws.strikeiron.com/RealTimeStockQuotes?WSDL	4	Stock
StockHistoryService	http://www.flash-db.com/services/ws/stockHistory.wsdl	4	Stock
net.xmethods.services.stockquote.StockQuoteService	http://xml.afpc.tamu.edu/StockQuoteService.wsdl	4	Stock
StockQuoteServiceService	http://piv-login2.osc.edu/axis/services/stock-wss-01?wsdl	4	Stock
StockQuotes	http://ws.strikeiron.com/SwanandMokashi/StockQuotes?WSDL	4	Stock

Εικόνα 2: Dataset με κατηγοριοποιημένες και βαθμολογημένες υπηρεσίες ιστού

Για τα datasets με αξιολογήσεις χρηστών για υπηρεσίες ιστού ακολουθήθηκε η εξής διαδικασία:

- Χρησιμοποιήθηκαν οι ίδιες υπηρεσίες ιστού με το dataset με κατηγοριοποιημένες και βαθμολογημένες υπηρεσίες ιστού.
- Όσον αφορά τους χρήστες, αναπτύχθηκαν δύο παραλλαγές. Στη πρώτη παραλλαγή θεωρήθηκε ότι κάθε χρήστης μπορεί να ενδιαφέρεται για πολλαπλές κατηγορίες υπηρεσιών ιστού, ενώ στην δεύτερη παραλλαγή θεωρήθηκε ότι κάθε χρήστης ενδιαφέρεται μόνο για μία κατηγορία υπηρεσιών ιστού. Και στις δύο παραλλαγές αποφασίστηκε ότι ο μέγιστος αριθμός χρηστών που μπορούν να αξιολογήσουν μία υπηρεσία ιστού είναι δέκα.
- Οι αξιολογήσεις χρηστών αποφασίστηκε να είναι ακέραιοι αριθμοί που κυμαίνονται από 1 έως 5 και παράχθηκαν με τη βοήθεια ενός Node script. Το script αυτό ακολουθεί την εξής διαδικασία ανά υπηρεσία ιστού:
    1. Παράγει ένα τυχαίο ακέραιο αριθμό που κυμαίνεται από 1 έως 10 και αναπαριστά τον αριθμό χρηστών που έχουν αξιολογήσει την υπηρεσία.
    2. Επιλέγει τυχαία ποιοι χρήστες έχουν αξιολογήσει την υπηρεσία. Κάθε χρήστης αναπαρίσταται από έναν ακέραιο αριθμό από 1 έως 10.
    3. Χρησιμοποιώντας το npm πακέτο *probability-distributions* και τη μέθοδό του *rnorm*, παράγει αξιολογήσεις χρηστών που ακολουθούν την κανονική κατανομή. Ως μέση τιμή τίθεται η βαθμολογία της υπηρεσίας και η τυπική απόκλιση προσαρμόζεται αναλόγως με τη βαθμολογία αυτή, ώστε να εξασφαλιστεί ότι οι παραγόμενες αξιολογήσεις ανήκουν στο [1, 5]. Τέλος, οι αξιολογήσεις αυτές στρογγυλοποιούνται.
    4. Χρησιμοποιώντας το npm πακέτο *probability-distributions* και τη μέθοδό του *rchisq*, παράγει αξιολογήσεις χρηστών που ακολουθούν την κατανομή x^2 με ένα βαθμό ελευθερίας και παράμετρο μη κεντρικότητας ίση με τη βαθμολογία της υπηρεσίας μειωμένη κατά 1. Τέλος, κάθε μία από τις παραγόμενες αξιολογήσεις μετασχηματίζεται σε έναν ακέραιο από 1 έως 5 ως εξής:
    $x > 0 \xrightarrow{\lfloor x \rfloor} x \in N \xrightarrow{x\ mod\ 5} x \in \{0, 1, 2, 3, 4\} \xrightarrow{x+1} x \in \{1, 2, 3, 4, 5\}$
- Δημιουργήθηκαν τέσσερα datasets με αξιολογήσεις χρηστών για υπηρεσίες ιστού. Στο πρώτο dataset χρησιμοποιήθηκε η πρώτη παραλλαγή χρηστών και αξιολογήσεις που ακολουθούσαν την κανονική κατανομή. Στο δεύτερο dataset χρησιμοποιήθηκε η δεύτερη παραλλαγή χρηστών και αξιολογήσεις που ακολουθούσαν την κανονική κατανομή. Στο τρίτο dataset χρησιμοποιήθηκε η πρώτη παραλλαγή χρηστών και αξιολογήσεις που ακολουθούσαν την κατανομή x^2. Τέλος, στο τέταρτο dataset χρησιμοποιήθηκε η δεύτερη παραλλαγή χρηστών και αξιολογήσεις που ακολουθούσαν την κατανομή x^2.



Εικόνα 3: Τμήμα εξόδου Node script που παράγει αξιολογήσεις χρηστών για υπηρεσίες ιστού

Username	WS1	WS2	WS3	WS4	WS5	WS6	WS7	WS8	WS9	WS10	WS11	WS12	WS13	WS14	WS15	WS16	WS17	WS18	WS19	WS20	WS21	WS22
user1		4	5	4	5		3		4		4	4		4			3			4		
user2	3	4	4		3		5	5			4		5	4	4		3		5	4	4	
user3		4	4	4				4	4				3			4			5	4	4	
user4		5		4				5	4	3	4						3	4	5	4	5	
user5	3	4		4	5		4				4	5	4				2		5	4	4	3
user6	4	4		4				4			3	5									4	4
user7	3	4	3	5	4		4	4	5		4	5	4			4	3		5	5	4	4
user8	2	4	3			5	4	4		5	4	4		4	4			4		4	4	
user9		4		4			4	5		4	4					4	4	5	5	5	4	5
user10	3	5		4		4	4				4		3				3				4	4

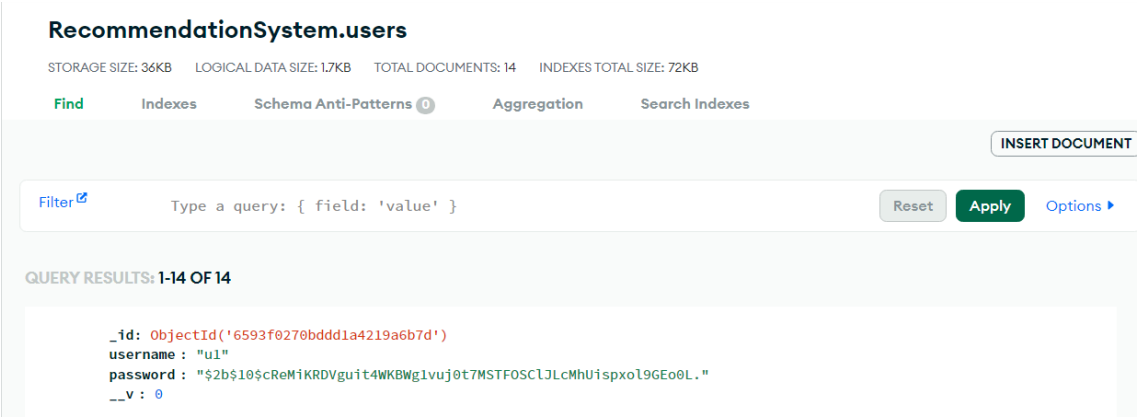
Εικόνα 4: Dataset με αξιολογήσεις χρηστών της πρώτης παραλλαγής που ακολουθούν την κανονική κατανομή

Username	Type	WS1	WS2	WS3	WS4	WS5	WS6	WS7	WS8	WS9	WS10	WS11	WS12	WS13	WS14	WS15	WS16	WS17	WS18	WS19	WS20	WS21
u1	Weather	4	4	3		4	4		5	4			4		4	4						
u2	Weather	3	4		4		4			4	3	4	5				4					
u3	Weather	3	4			3	5		4	4	4	5	3	4								
u4	Weather		3	4		5	4	5		5	3		4		4							
u5	Weather	3				4	4		4		4		4			4						
u6	Weather	3		3			5		5	3	4	4			4		5					
u7	Weather		4						4			4	5									
u8	Weather		3	5	3		4		4	4	4		4	4	5							
u9	Weather	3				4	4			4			4									
u10	Weather		4		5		4		4		5		4									
u11	Stock																	3		5	4	
u12	Stock																		4	5		
u13	Stock																			4	4	4
u14	Stock																	2		5	4	
u15	Stock																	3		5	4	
u16	Stock																	3			4	
u17	Stock																	3	4	5	3	4
u18	Stock																			5	4	
u19	Stock																	3	5	5	5	
u20	Stock																		4	5	4	4

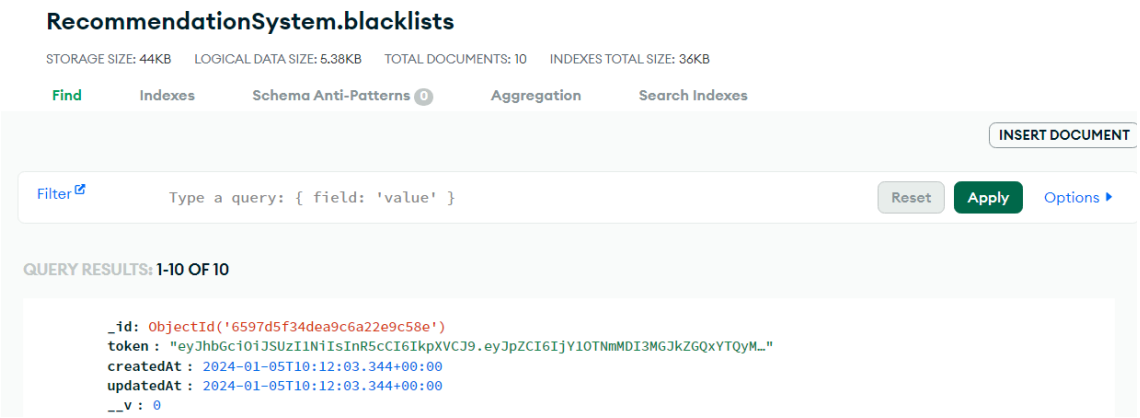
Εικόνα 5: Dataset με αξιολογήσεις χρηστών της δεύτερης παραλλαγής που ακολουθούν την κανονική κατανομή

3.2 Βάση δεδομένων

Στο πλαίσιο της παρούσας εργασίας δημιουργήθηκε ένα cloud database στον MongoDB Atlas. Περιλαμβάνει δύο collections, που ονομάζονται *users* και *blacklists*. Στο collection *users* περιέχονται τα στοιχεία σύνδεσης των χρηστών(username, password) στο Node API. Στο collection *blacklists* περιέχονται κωδικοποιημένα JWT, τα οποία δημιουργήθηκαν από το Node API, αλλά δεν θεωρούνται πλέον έγκυρα.



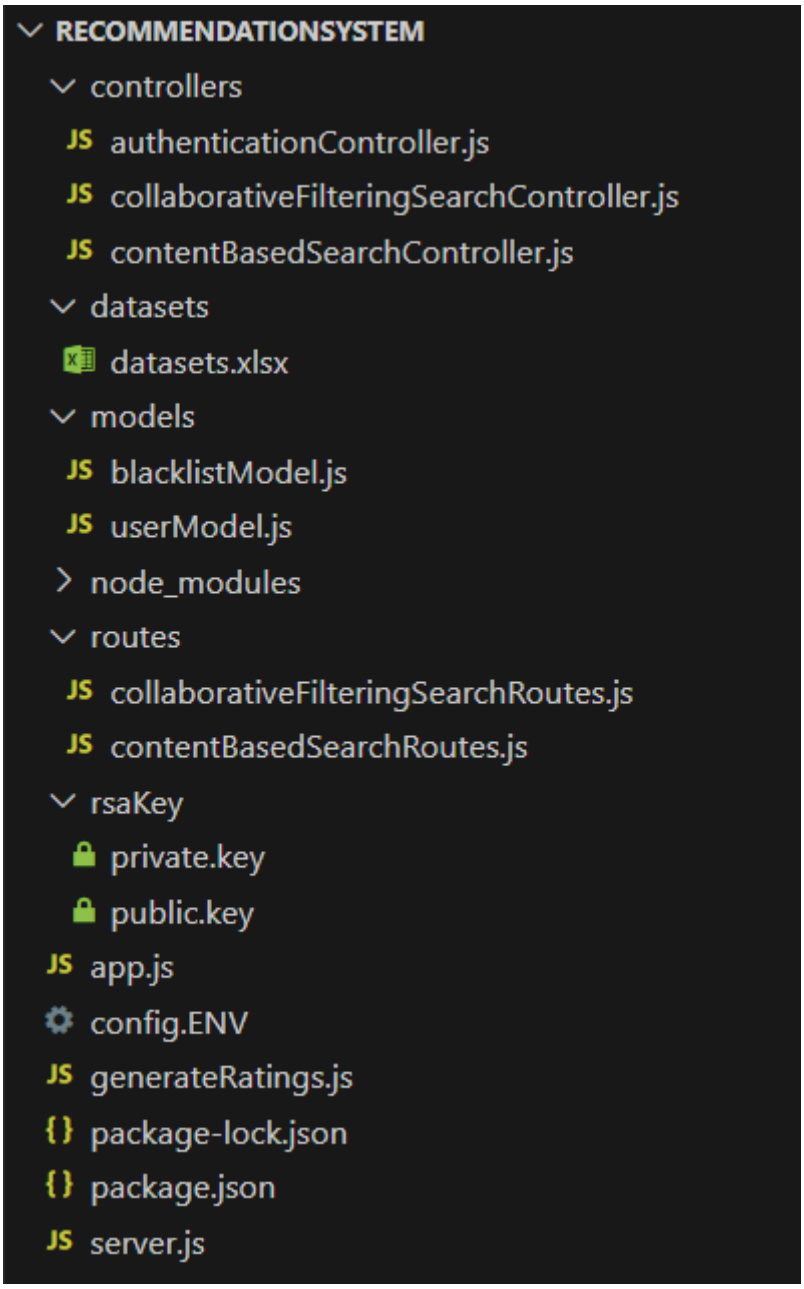
Εικόνα 6: Collection "users" του cloud database



Εικόνα 7: Collection "blacklists" του cloud database

3.3 Δομή Node project

Το Node project που δημιουργήθηκε στο πλαίσιο της παρούσας εργασίας έχει την ακόλουθη δομή:



Εικόνα 8: Δομή Node project

Όπως φαίνεται και στην παραπάνω εικόνα, το Node project αποτελείται από τα ακόλουθα μέρη:

- **Φάκελος *controllers*.** Περιέχει αρχεία με μεθόδους που χειρίζονται τα http requests προς το API. Πιο συγκεκριμένα, το *authenticationController.js* χειρίζεται τα http requests που σχετίζονται με αυθεντικοποίηση(π.χ. εγγραφή χρήστη), το collaborativeFilteringSearchController.js χειρίζεται τα http requests που σχετίζονται με παραγωγή συστάσεων με φιλτράρισμα βάσει περιεχομένου και το contentBasedSearchController.js χειρίζεται τα http requests που σχετίζονται με παραγωγή συστάσεων με συνεργατικό φιλτράρισμα.
- **Φάκελος *datasets.*** Περιέχει το excel αρχείο *datasets.xlsx*. Το αρχείο αυτό περιλαμβάνει σε διαδοχικά υπολογιστικά φύλλα τα datasets που δημιουργήθηκαν για το φιλτράρισμα βάσει περιεχομένου και το συνεργατικό φιλτράρισμα(περισσότερα στο Κεφάλαιο 3.1).
- **Φάκελος *models.*** Περιέχει αρχεία τα οποία χειρίζονται την ανάκτηση/αποθήκευση πληροφοριών στο cloud database(περισσότερα στο Κεφάλαιο 3.2). Πιο συγκεκριμένα το *userModel.js* αλληλεπιδρά με το collection *users* και το *blacklistModel.js* αλληλεπιδρά με το collection *blacklists*.
- **Φάκελος *routes*.** Περιέχει αρχεία τα οποία καθορίζουν τα endpoints του API, τα http requests που μπορούν να υποβληθούν σε αυτά, καθώς και ποιες μέθοδοι των αρχείων του φακέλου *controllers* θα χειριστούν αυτά τα requests.
- **Φάκελος *rsaKey.*** Περιέχει ένα ζεύγος κλειδιών RSA(ιδιωτικό-δημόσιο) των 4096 bits που χρησιμοποιούνται για την κωδικοποίηση και την αποκωδικοποίηση JWT's.
- **Αρχείο *app.js*.** Δημιουργεί την εφαρμογή Express.
- **Αρχείο *config.ENV*.** Περιέχει τις μεταβλητές περιβάλλοντος του project.
- **Αρχείο *generateRatings.js*.** Πρόκειται για ένα Node script το οποίο παράγει αξιολογήσεις χρηστών για υπηρεσίες ιστού που χρησιμοποιήθηκαν για τη δημιουργία datasets(περισσότερα στο Κεφάλαιο 3.1.2). Δεν σχετίζεται με τη λειτουργία του API.
- **Αρχείο *server.js*.** Συνδέεται με το cloud database και εκκινεί έναν http εξυπηρετητή. Έτσι ξεκινά η λειτουργία του API.

Κεφάλαιο 4 – Ανάλυση λειτουργιών εφαρμογής

4.1 ΣΣ με φιλτράρισμα βάσει περιεχομένου

4.1.1 Θεωρητικό υπόβαθρο

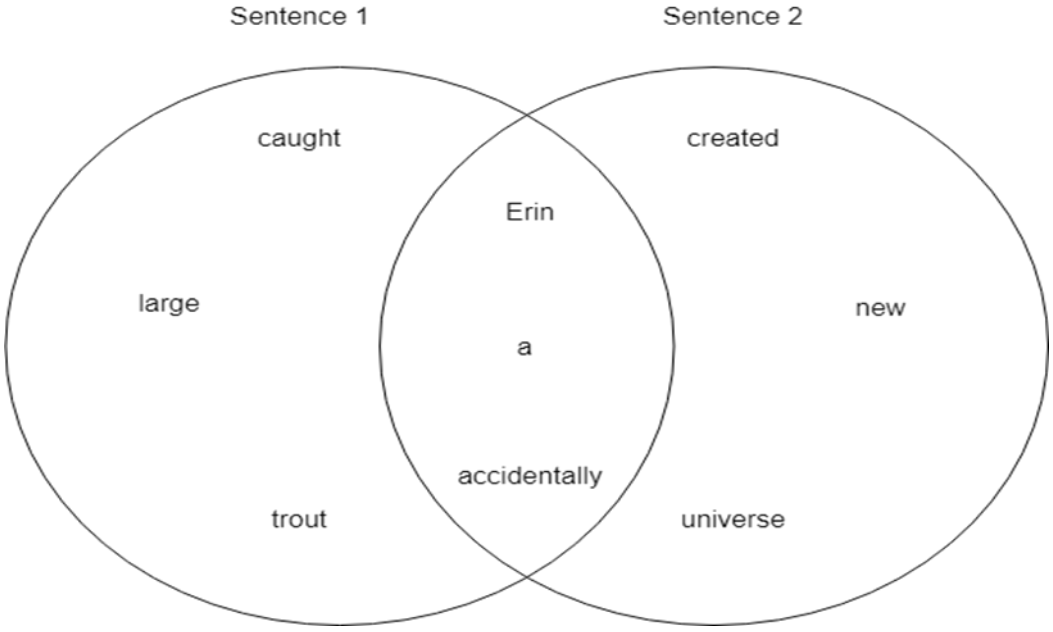
Σε γενικές γραμμές, οι μηχανές συστάσεων ουσιαστικά αναζητούν να βρουν στοιχεία που έχουν ομοιότητες. Μπορούμε να σκεφτούμε την ομοιότητα ως την εύρεση συνόλων με σχετικά μεγάλη τομή. Μπορούμε να πάρουμε οποιαδήποτε συλλογή στοιχείων, όπως έγγραφα, ταινίες, ιστοσελίδες κ.λπ., και να συλλέξουμε ένα σύνολο χαρακτηριστικών που ονομάζονται "shingles".

Shingles

Πρόκειται για μια πολύ βασική και ευρεία έννοια. Στην περίπτωση κειμένου, θα μπορούσαν να είναι χαρακτήρες, μονογράμματα ή διγράμματα. Απλώς ανάγουν τα κείμενα σε σύνολα στοιχείων ώστε να μπορεί να υπολογιστεί η ομοιότητα μεταξύ των συνόλων. Για καλύτερη κατανόηση, ας πάρουμε δύο προτάσεις και ας τις μετατρέψουμε σε σύνολα από shingles:

- Sentence 1 = "Erin accidentally caught a large trout", Shingles = [Erin, accidentally, caught, a, large, trout]
- Sentence 2 = "Erin accidentally created a new universe", Shingles = [Erin, accidentally, created, a, new, universe]

Τώρα, μπορούμε να βρούμε την ομοιότητα μεταξύ αυτών των προτάσεων εξετάζοντας μια οπτική αναπαράσταση της τομής των shingles μεταξύ των δύο συνόλων. Σε αυτό το παράδειγμα, ο συνολικός αριθμός (ένωση) των shingles είναι 12, και 3 αποτελούν μέρος της τομής. Θα υπολογίσουμε την ομοιότητα ως $\frac{3}{12} = \frac{1}{4}$.



Διάγραμμα 1: Αναπαράσταση τομής shingles μεταξύ δύο συνόλων

Λογικά, για να κάνουμε συστάσεις για υπηρεσίες ιστού θα πρέπει να έχουμε μια στήλη για την περιγραφή κάθε υπηρεσίας ιστού και μια γραμμή για κάθε λέξη που συναντάμε. Δεδομένου ότι

οι υπηρεσίες ιστού μπορεί να διαφέρουν πολύ, θα έχουμε πολλές κενές γραμμές για κάθε στήλη, εξ ου και η αραιότητα. Για να κάνουμε συστάσεις, θα υπολογίζαμε την ομοιότητα μεταξύ κάθε γραμμής βλέποντας ποιες λέξεις είναι κοινές.

	Description of Web Service 1	Description of Web Service 2	…	Description of Web Service n
Word 1	1	0	…	0
Word 2	0	1	…	1
…	…	…	…	…
Word n	0	0	…	1

Για να αντιμετωπίσουμε το πρόβλημα των κενών γραμμών, θα χρησιμοποιήσουμε τον αλγόριθμο LSH για να μειώσουμε τις διαστάσεις.

Αλγόριθμος LSH

Αυτή η τεχνική μπορεί να χρησιμοποιηθεί ως βάση για πιο σύνθετες μηχανές συστάσεων, συμπιέζοντας τις σειρές σε "υπογραφές", ή ακολουθίες ακεραίων αριθμών, που μας επιτρέπουν να συγκρίνουμε υπηρεσίες ιστού χωρίς να χρειάζεται να συγκρίνουμε ολόκληρα σύνολα λέξεων. Ο LSH χρησιμοποιείται για την εκτέλεση Αναζητήσεων Πλησιέστερου Γείτονα με βάση μια απλή έννοια της ομοιότητας: Δύο στοιχεία είναι παρόμοια εάν η τομή των συνόλων τους είναι αρκετά μεγάλη. Αυτή είναι η ίδια ακριβώς έννοια με την ομοιότητα Jaccard των συνόλων (η ομοιότητα Jaccard ορίζεται ως η τομή δύο συνόλων διαιρούμενη με την ένωση των δύο συνόλων). Το μεγάλο πλεονέκτημα αυτού του αλγορίθμου είναι η δυνατότητα κλιμάκωσης με τη χρήση μιας τεχνικής δάσους καθώς αυξάνεται ο αριθμός των στοιχείων. Όπως αναφέρθηκε προηγουμένως, ο στόχος είναι η εύρεση συνόλων που είναι παρόμοια με ένα σύνολο ερωτήματος. Η γενική μέθοδος είναι η εξής:

1. Κατακερματισμός στοιχείων έτσι ώστε παρόμοια στοιχεία να μπαίνουν στον ίδιο κάδο με μεγάλη πιθανότητα.
2. Περιορισμός της αναζήτησης ομοιότητας στον κάδο που σχετίζεται με το στοιχείο ερωτήματος.

MinHash

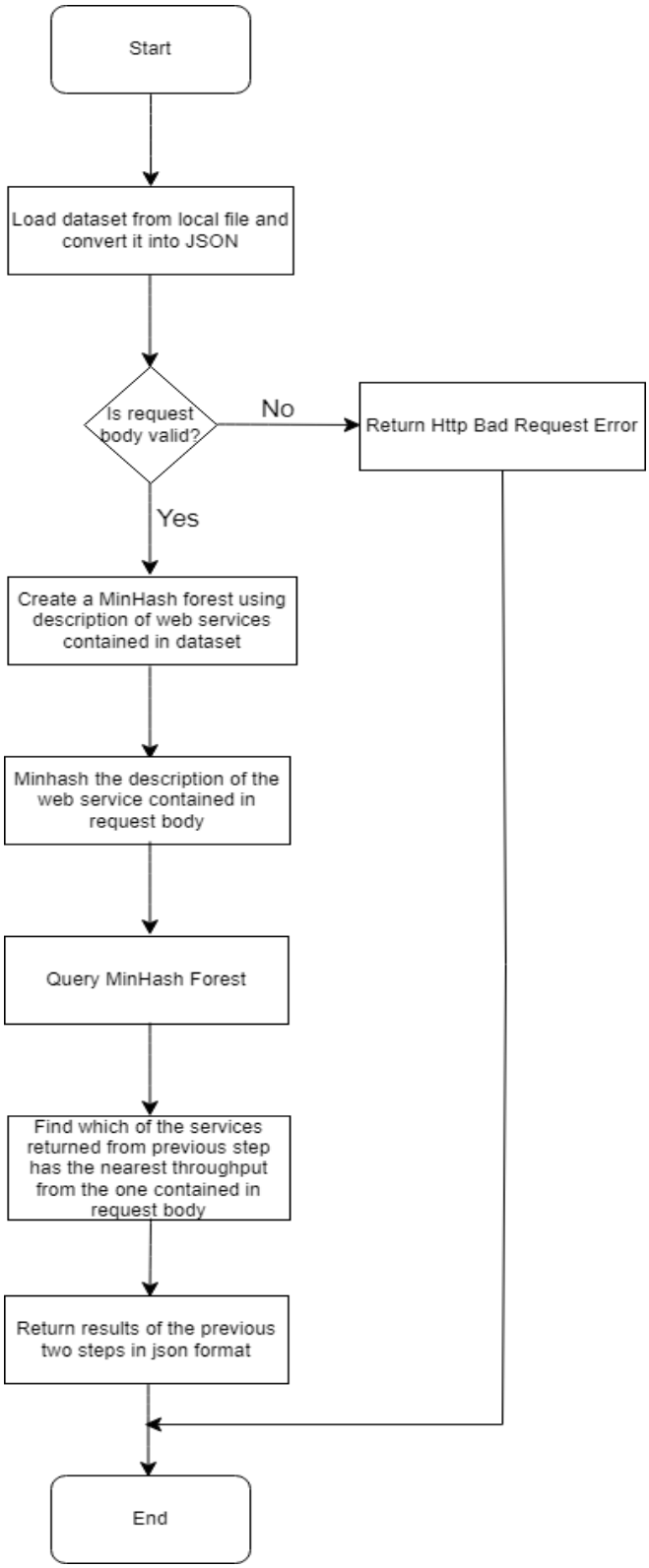
Ο στόχος του MinHash είναι να αντικαταστήσει ένα μεγάλο σύνολο με μια μικρότερη "υπογραφή" που εξακολουθεί να διατηρεί την υποκείμενη μετρική ομοιότητας. Για να δημιουργηθεί μια υπογραφή MinHash για κάθε σύνολο:

1. Μεταθέστε τυχαία τις γραμμές του πίνακα με τα shingles.
2. Για κάθε σύνολο, ξεκινήστε από την αρχή και βρείτε τη θέση του πρώτου shingle που εμφανίζεται στο σύνολο, δηλαδή του πρώτου shingle με 1 στο κελί του. Χρησιμοποιήστε αυτόν τον αριθμό για να αναπαραστήσετε το σύνολο. Αυτή είναι η "υπογραφή".
3. Επαναλάβετε όσες φορές επιθυμείτε, προσθέτοντας κάθε φορά το αποτέλεσμα στην υπογραφή του συνόλου.

Μετά από αυτή τη διαδικασία, η περιγραφή κάθε υπηρεσίας ιστού θα αντιπροσωπεύεται από μια υπογραφή MinHash, όπου ο αριθμός των γραμμών είναι πλέον πολύ μικρότερος από τον αριθμό των γραμμών του αρχικού πίνακα shingle.

4.1.2 Σχηματική απεικόνιση

Το παρακάτω διάγραμμα απεικονίζει τα βήματα που ακολουθεί το ΣΣ με φιλτράρισμα βάσει περιεχομένου της υλοποίησης για την παραγωγή συστάσεων.



Διάγραμμα 2: Διάγραμμα ροής για ΣΣ με φιλτράρισμα βάσει περιεχομένου

4.1.3 Υλοποίηση

Από το σύνολο των datasets που δημιουργήθηκαν στο πλαίσιο της παρούσας εργασίας, για τη λειτουργία αυτή θα χρησιμοποιηθεί το dataset για φιλτράρισμα βάσει περιεχομένου(περισσότερα στο Κεφάλαιο 3.1.1). Σε αυτό το dataset περιέχεται μία λίστα με υπηρεσίες ιστού για κάθε μία από τις οποίες παρέχονται τα ακόλουθα στοιχεία: Service ID, WSDL Address, Service Provider, IP Address, Country, Continent, AS, Latitude, Longitude, Throughput, Response Time και Operations. Αποφασίστηκε η αναζήτηση υπηρεσιών ιστού να γίνεται σε δύο στάδια. Στο πρώτο στάδιο θα γίνεται αναζήτηση με βάση το στοιχείο Operations στο σύνολο των υπηρεσιών ιστού του dataset. Στο δεύτερο στάδιο θα γίνεται αναζήτηση με βάση το στοιχείο Throughput στις υπηρεσίες ιστού που προέκυψαν από το πρώτο στάδιο. Τόσο οι τιμές των στοιχείων Operations, Throughput για τις οποίες θα γίνεται η αναζήτηση όσο και το πλήθος των υπηρεσιών ιστού που θα επιστρέφονται από το πρώτο στάδιο θα ορίζονται από τον χρήστη. Η εκτέλεση αυτής της λειτουργίας του API γίνεται με ένα http post request στο endpoint http://localhost:4000/api/v1/contentBasedSearch/.

Εισαγωγή dataset

Κατά την εκκίνηση του API, το dataset για φιλτράρισμα βάσει περιεχομένου φορτώνεται από το τοπικό αρχείο excel του Node project(περισσότερα στο Κεφάλαιο 3.3). Για διευκόλυνση προσπέλασης των δεδομένων, το dataset μετατρέπεται σε JSON. Τα παραπάνω επιτυγχάνονται αξιοποιώντας το npm πακέτο *xlsx*.

```
1. // Read excel file containing the dataset for this search technique.
2. const workbook = xlsx.readFile(path.join(__dirname, "../", 'datasets/datasets.xlsx'))
3. // Gather the names of all sheets contained in excel file.
4. const workbookSheets = workbook.SheetNames
5. // Convert the data contained in first sheet(our dataset) into JSON.
6. const dataset = xlsx.utils.sheet_to_json(workbook.Sheets[workbookSheets[0]])
```

Κομμάτι κώδικα 1: Εισαγωγή dataset για φιλτράρισμα βάσει περιεχομένου

Έλεγχος request body

Μόλις το API λάβει το http post request του χρήστη στο κατάλληλο endpoint, ελέγχει κατά πόσο στο request body παρέχονται οι απαραίτητες τιμές για την εκτέλεση του αλγορίθμου. Πιο συγκεκριμένα, το request body πρέπει να περιέχει μόνο τα ακόλουθα πεδία:

- **operation.** Πρόκειται για την τιμή του στοιχείου Operations της υπηρεσίας ιστού προς αναζήτηση. Η τιμή που παρέχει ο χρήστης για το πεδίο αυτό πρέπει να είναι τύπου string.
- **throughput.** Πρόκειται για την τιμή του στοιχείου Throughput της υπηρεσίας ιστού προς αναζήτηση. Η τιμή που παρέχει ο χρήστης για το πεδίο αυτό πρέπει να είναι ένας πραγματικός αριθμός.
- **numOfRecommendations.** Πρόκειται για το πλήθος των υπηρεσιών ιστού που θα επιστραφούν από το πρώτο στάδιο αναζήτησης υπηρεσιών ιστού. Η τιμή που παρέχει ο χρήστης για το πεδίο αυτό πρέπει να είναι ένας ακέραιος αριθμός.

Αν το request body πληροί τις παραπάνω προϋποθέσεις, ο αλγόριθμος προχωρά στην εκτέλεση του επόμενου βήματος. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http Bad Request Error και ο αλγόριθμος τερματίζεται. Οι περιγραφόμενοι έλεγχοι στο request body γίνονται με τη βοήθεια του npm πακέτου *ajv*.

```
exports.checkRequestBody = (req,res,next) => {
    try {
        // Validate request body against a json schema.
        const schema = {
            type: "object",
            properties: {
                operation: {type: "string"},
                numOfRecommendations: {type: "integer"},
                throughput: {type: "number"}
            },
            required: ["operation"],
            required: ["numOfRecommendations"],
            required: ["throughput"],
            additionalProperties: false,
        }
        const ajv = new Ajv()
        const validate = ajv.compile(schema)
        const valid = validate(req.body)
        // If request body is invalid, return Bad Request Error.
        if (!valid) {
            return res.status(400).json({
                message: 'Request has unexpected format'
            })
        }
        next()
    } catch (err) {
        return res.status(500).json({
            message: 'Internal Server Error'
        })
    }
}
```

Κομμάτι κώδικα 2: Έλεγχος request body στο ΣΣ με φιλτράρισμα βάσει περιεχομένου

Προεπεξεργασία δεδομένων

Για να είναι δυνατή η χρήση της τιμής του πεδίου operation(που περιέχεται στο request body) για την αναζήτηση υπηρεσιών ιστού, πρέπει να μετασχηματιστεί. Αρχικά, διασπάται σε λέξεις-shingles χρησιμοποιώντας το Javascript object *Intl.Segmenter*. Στη συνέχεια, όλοι οι χαρακτήρες που περιέχονται στις λέξεις αυτές μετατρέπονται σε πεζούς. Τέλος, χρησιμοποιώντας τη κλάση *MinHash* του npm πακέτου *minhashjs*, εφαρμόζεται ο αλγόριθμος MinHash για 128 μεταθέσεις στο σύνολο των λέξεων-shingles στο οποίο διασπάστηκε η τιμή του πεδίου operation.

```
function minHashText(text) {
    // Split text to words and return an array containing them in lower case.
    // Source: https://dev.to/kamonwan/the-right-way-to-break-string-into-words-in-javascript-3jp6
    const segmenter = new Intl.Segmenter([], { granularity: 'word' })
    const segmentedText = segmenter.segment(text)
    const tokens = [...segmentedText].filter(s => s.isWordLike).map(s => s.segment.toLowerCase())
    // Minhash the text on all of its words.
    let m = new MH.MinHash(num_perm=permutations)
    tokens.forEach(token => {
        m.update(token)
    })
    //Return MinHashed text.
    return m
}
```

Κομμάτι κώδικα 3: Προεπεξεργασία δεδομένων στο ΣΣ με φιλτράρισμα βάσει περιεχομένου

Δημιουργία MinHash Forest

Η δημιουργία της δομής αυτής αποσκοπεί στο να κάνει δυνατή την αναζήτηση υπηρεσιών ιστού με βάση το στοιχείο Operations χρησιμοποιώντας τον αλγόριθμο LSH. Για τη δημιουργία της ακολουθούνται τα εξής βήματα:

1. Προσπελάζονται όλες οι υπηρεσίες ιστού που περιέχονται στο dataset για φιλτράρισμα βάσει περιεχομένου και για κάθε μία από αυτές ανακτάται η τιμή του στοιχείου Operations και εφαρμόζεται σε αυτή την τιμή ο ίδιος μετασχηματισμός που εφαρμόστηκε στη τιμή του πεδίου operation του request body.
2. Χρησιμοποιώντας την κλάση *ForestLSH* του npm πακέτου *minhashjs*, δημιουργείται ένα MinHash Forest από την μετασχηματισμένη τιμή του στοιχείου Operations όλων των υπηρεσιών ιστού του dataset.
3. Εισάγεται ευρετήριο στο MinHash Forest για να καταστεί δυνατή η αναζήτηση.

```
function createMinhashForest() {
    console.time('Time to build forest')
    let minhash = []
    // Access all web services contained in dataset.
    dataset.forEach(ws => {
        // MinHash the value of the field "Operations" for each web service.
        minhash.push(minHashText(ws.Operations));
    })
    // Create a MinHash LSH Forest.
    const forest = new F.MinHashLSHForest(num_perm=permutations)
    // Add all MinHashed strings into the index.
    for (let i = 0; i < minhash.length; i++) {
        forest.add(i,minhash[i])
    }
    // Make MinHash LSH Forest searchable.
    forest.index()
    console.timeEnd('Time to build forest')
    return forest;
}
```

Κομμάτι κώδικα 4: Δημιουργία MinHash Forest

Αναζήτηση υπηρεσιών ιστού

Σε πρώτο στάδιο πραγματοποιείται αναζήτηση στο MinHash Forest με βάση τη μετασχηματισμένη τιμή του πεδίου operation του request body. Για τον καθορισμό του πλήθους των αποτελεσμάτων που θα επιστραφούν, χρησιμοποιείται η τιμή του πεδίου numOfRecommendations του request body. Το επιστρεφόμενο αποτέλεσμα είναι ένας πίνακας με τις θέσεις των υπηρεσιών ιστού που πληρούν το κριτήριο αναζήτησης στο MinHash Forest και κατ' επέκταση στο dataset(η εισαγωγή των υπηρεσιών ιστού στο MinHash Forest έγινε με τη ίδια σειρά που είναι αποθηκευμένες στο dataset). Αξιοποιώντας τις επιστρεφόμενες θέσεις, ανακτώνται τα αποθηκευμένα στοιχεία των αντίστοιχων υπηρεσιών ιστού στο dataset με εξαίρεση το στοιχείο Operation και καταχωρούνται στον πίνακα *ws*.

Σε δεύτερο στάδιο εντοπίζεται ποια από τις υπηρεσίες ιστού που προέκυψαν από το πρώτο στάδιο έχει το πιο κοντινό Throughput με την τιμή του πεδίου throughput του request body. Τα αποθηκευμένα στοιχεία αυτής της υπηρεσίες ιστού με εξαίρεση το στοιχείο Operations καταχωρούνται στον πίνακα *service*.

Η εκτέλεση του αλγορίθμου ολοκληρώνεται με την επιστροφή στον χρήστη των πινάκων *ws* και *service*.

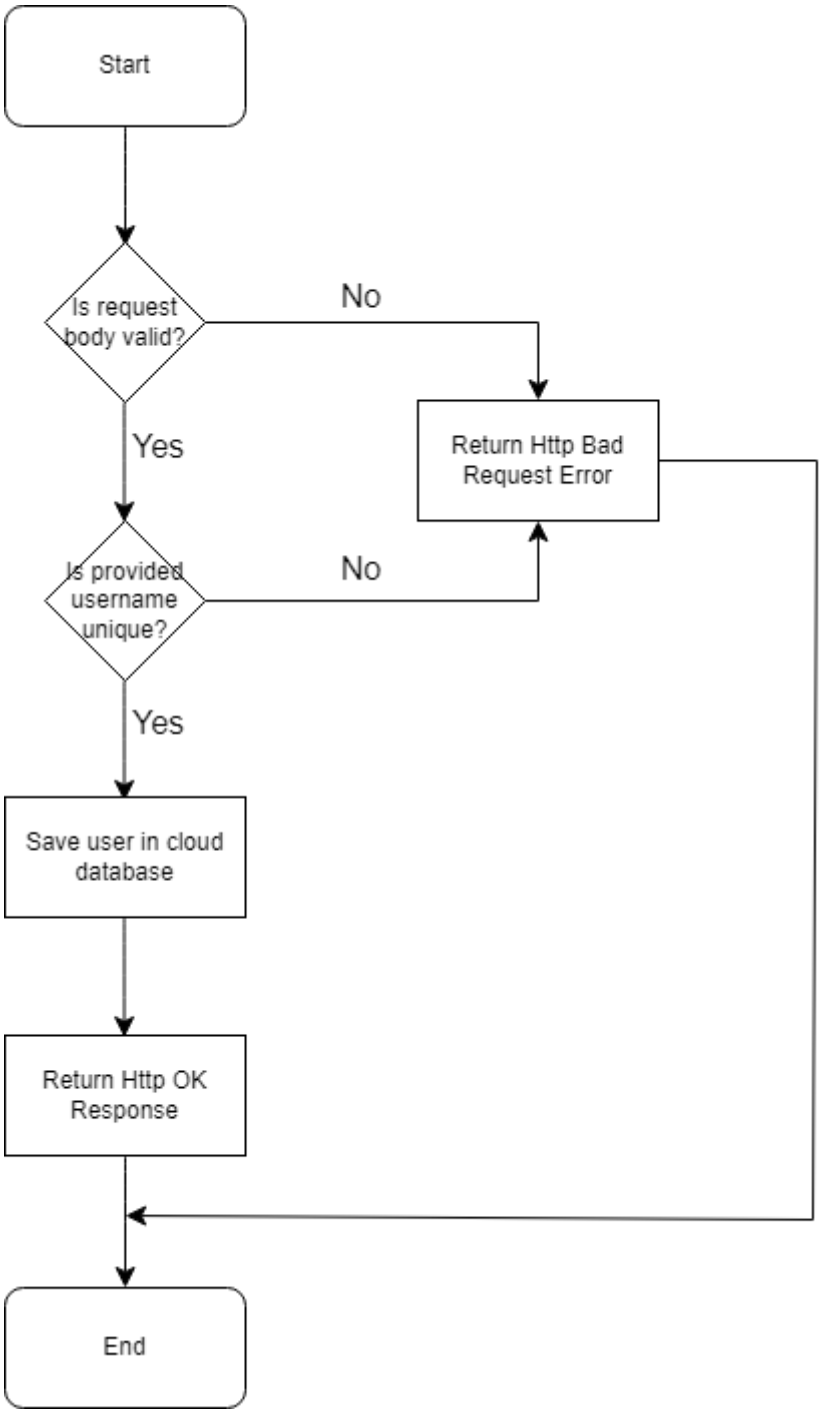
```
1.  // Query the created above forest using the values retrieved from request body.
2.  let results = forest.query(m,numOfRecommendations)
3.  // Gather all the stored info for recommended web services and find which of
4.  // them has the nearest throughput from the one contained in request body.
5.  let ws = []
6.  let minValue = Number.MAX_VALUE
7.  let minIndex = -1
8.  results.forEach(result => {
9.      // Clone json object and remove field "Operations".
10.     let service = JSON.parse(JSON.stringify(dataset[result]))
11.     delete service['Operations']
12.     ws.push(service)
13.     if (Math.abs(service.Throughput - throughput) < minValue) {
14.         minValue = Math.abs(service.Throughput - throughput)
15.         minIndex = result
16.     }
17. })
18. console.timeEnd('Time to query forest')
19. let service = []
20. // Check if any recomendations were found.
21. if (minIndex != -1) {
22.     // Clone json object and remove field "Operations".
23.     let temp = JSON.parse(JSON.stringify(dataset[minIndex]))
24.     delete temp['Operations']
25.     service.push(temp)
26. }
27. // Return the results in json format
28. res.status(200).json({
29.     recommendations: {
30.         basedOnOperation: ws,
31.         basedOnThroughput: service
32.     }
33. })
```

Κομμάτι κώδικα 5: Αναζήτηση υπηρεσιών ιστού στο ΣΣ με φιλτράρισμα βάσει περιεχομένου

4.2 Εγγραφή χρήστη

4.2.1 Σχηματική απεικόνιση

Το παρακάτω διάγραμμα απεικονίζει τα βήματα που ακολουθούνται για την εγγραφή ενός νέου χρήστη στο σύστημα.



Διάγραμμα 3: Διάγραμμα ροής για εγγραφή χρήστη

4.2.2 Υλοποίηση

Η εκτέλεση της λειτουργίας αυτής γίνεται με ένα http post request στο endpoint http://localhost:4000/api/v1/collaborativeFilteringSearch/register.

Έλεγχος request body

Μόλις το API λάβει το http post request του χρήστη στο κατάλληλο endpoint, ελέγχει κατά πόσο στο request body παρέχονται οι απαραίτητες τιμές για την εκτέλεση της λειτουργίας. Πιο συγκεκριμένα, το request body πρέπει να περιέχει μόνο τα ακόλουθα πεδία:

- **username, password.** Πρόκειται για τα στοιχεία σύνδεσης του χρήστη στο σύστημα. Οι τιμές που παρέχει ο χρήστης για τα πεδία αυτά πρέπει να είναι τύπου string.

Αν το request body πληροί τις παραπάνω προϋποθέσεις, η εκτέλεση της λειτουργίας συνεχίζεται. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http Bad Request Error και η λειτουργία τερματίζεται. Οι περιγραφόμενοι έλεγχοι στο request body γίνονται με τη βοήθεια του npm πακέτου ajv.

```
exports.checkRequestBody = (req,res,next) => {
    try {
        // Validate request body again a json schema.
        const schema = {
            type: "object",
            properties: {
                username: {type: "string"},
                password: {type: "string"}
            },
            required: ["username"],
            required: ["password"],
            additionalProperties: false,
        }
        const ajv = new Ajv()
        const validate = ajv.compile(schema)
        const valid = validate(req.body)
        // If request body is invalid, return Bad Request Error.
        if (!valid) {
            return res.status(400).json({
                message: 'Request has unexpected format'
            })
        }
        next()
    } catch (err) {
        return res.status(500).json({
            message: 'Internal Server Error'
        })
    }
}
```

Κομμάτι κώδικα 6: Έλεγχος request body στην εγγραφή χρήστη

Έλεγχος μοναδικότητας χρήστη

Ελέγχεται αν στο collection *users* του cloud database υπάρχει εγγραφή-χρήστης με ίδιο username με αυτό που περιέχεται στο request body. Αν όχι, η εκτέλεση της λειτουργίας συνεχίζεται κανονικά. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http Bad Request Error και η λειτουργία τερματίζεται.

Αποθήκευση χρήστη στη βάση δεδομένων

Η διαδικασία αυτή πραγματοποιείται σε δύο στάδια. Σε πρώτο στάδιο η τιμή του πεδίου password που παρέχεται στο request body μετασχηματίζεται για λόγους ασφάλειας. Ο

μετασχηματισμός αυτός γίνεται με τη βοήθεια του npm πακέτου *bcrypt* και περιλαμβάνει τα ακόλουθα βήματα:

- Παραγωγή μίας τυχαίας συμβολοσειράς, η οποία ονομάζεται salt.
- Εφαρμογή συνάρτησης κατακερματισμού στη συμβολοσειρά που προκύπτει από τη συνένωση της τιμής του πεδίου password και του salt. Η έξοδος της συνάρτησης αποτελεί τη μετασχηματισμένη τιμή του πεδίου password.

```
1.  // Hash user's password before saving it to the database.
2.  userSchema.pre("save", function (next) {
3.      const user = this
4.
5.      if (!user.isModified("password")) return next()
6.      bcrypt.genSalt(10, (err, salt) => {
7.          if (err) return next(err)
8.
9.          bcrypt.hash(user.password, salt, (err, hash) => {
10.             if (err) return next(err)
11.
12.             user.password = hash;
13.             next()
14.         })
15.     })
16. })
```

Κομμάτι κώδικα 7: Μετασχηματισμός password με τη χρήση συνάρτησης κατακερματισμού

Σε δεύτερο στάδιο δημιουργείται μία νέα εγγραφή στο collection *users* του cloud database με το username που παρείχε ο χρήστης στο request body και το μετασχηματισμένο password. Η εκτέλεση της λειτουργίας ολοκληρώνεται με την επιστροφή στο χρήστη ενός Http OK Response.

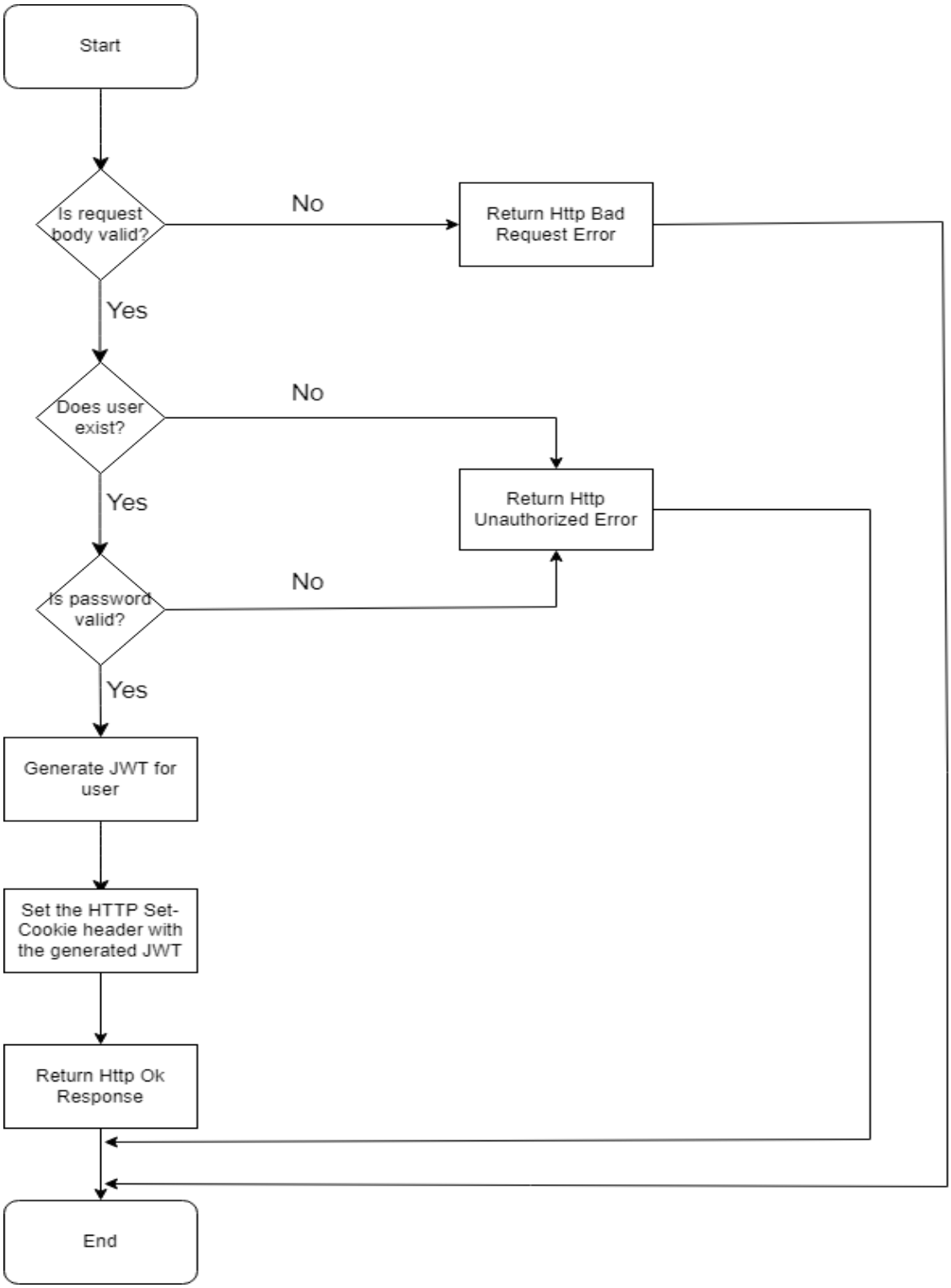
```
1.  exports.register = async (req, res) => {
2.      // Get required variables from request body.
3.      const {username, password} = req.body
4.      try {
5.          // Create an instance of a User.
6.          const newUser = new User({
7.              username,
8.              password,
9.          })
10.         // Check if a user with the provided username already exists.
11.         const userExists = await User.findOne({username})
12.         //If such a user exists, return Bad Request Error.
13.         if (userExists) {
14.             return res.status(400).json({
15.                 message: 'It seems that you already have an account, please log in instead.'
16.             })
17.         }
18.         //Otherwise, save user to database.
19.         const savedUser = await newUser.save()
20.         return res.status(200).json({
21.             message: 'Your account has been successfully created.'
22.         })
23.     } catch (err) {
24.         return res.status(500).json({
25.             message: 'Internal Server Error'
26.         })
27.     }
28. }
```

Κομμάτι κώδικα 8: Εγγραφή χρήστη

4.3 Είσοδος χρήστη

4.3.1 Σχηματική απεικόνιση

Το παρακάτω διάγραμμα απεικονίζει τα βήματα που ακολουθούνται για την είσοδο ενός χρήστη στο σύστημα.



Διάγραμμα 4: Διάγραμμα ροής για είσοδο χρήστη

4.3.2 Υλοποίηση

Η εκτέλεση της λειτουργίας αυτής γίνεται με ένα http post request στο endpoint http://localhost:4000/api/v1/collaborativeFilteringSearch/login.

Έλεγχος request body

Μόλις το API λάβει το http post request του χρήστη στο κατάλληλο endpoint, ελέγχεται κατά πόσο στο request body παρέχονται οι απαραίτητες τιμές για την εκτέλεση της λειτουργίας. Πιο συγκεκριμένα, το request body πρέπει να περιέχει μόνο τα ακόλουθα πεδία:

- **username, password.** Πρόκειται για τα στοιχεία σύνδεσης με τα οποία ο χρήστης εγγράφηκε στο σύστημα. Οι τιμές που παρέχει ο χρήστης για τα πεδία αυτά πρέπει να είναι τύπου string.

Αν το request body πληροί τις παραπάνω προϋποθέσεις, η εκτέλεση της λειτουργίας συνεχίζεται. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http Bad Request Error και η λειτουργία τερματίζεται. Οι περιγραφόμενοι έλεγχοι στο request body γίνονται με τη βοήθεια του npm πακέτου ajv.

```
1.  exports.checkRequestBody = (req,res,next) => {
2.      try {
3.          // Validate request body again a json schema.
4.          const schema = {
5.              type: "object",
6.              properties: {
7.                  username: {type: "string"},
8.                  password: {type: "string"}
9.              },
10.             required: ["username"],
11.             required: ["password"],
12.             additionalProperties: false,
13.         }
14.         const ajv = new Ajv()
15.         const validate = ajv.compile(schema)
16.         const valid = validate(req.body)
17.         // If request body is invalid, return Bad Request Error.
18.         if (!valid) {
19.             return res.status(400).json({
20.                 message: 'Request has unexpected format'
21.             })
22.         }
23.         next()
24.     } catch (err) {
25.         return res.status(500).json({
26.             message: 'Internal Server Error'
27.         })
28.     }
29. }
```

Κομμάτι κώδικα 9: Έλεγχος request body στην είσοδο χρήστη

Έλεγχος ύπαρξης χρήστη

Ελέγχεται αν στο collection *users* του cloud database υπάρχει εγγραφή-χρήστης με τιμή στο πεδίο username ίδια με αυτή που παρέχει ο χρήστης στο request body. Αν ναι, ανακτώνται οι τιμές των πεδίων _id και password για την συγκεκριμένη εγγραφή και η εκτέλεση της λειτουργίας συνεχίζεται. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http Unauthorized Error και η λειτουργία τερματίζεται.

Έλεγχος εγκυρότητας password

Όπως αναφέρθηκε στην εγγραφή χρήστη, στο cloud database δεν αποθηκεύεται αυτούσιο το password αλλά ένας μετασχηματισμός του για λόγους ασφάλειας. Επομένως, πρέπει να διαπιστωθεί αν η τιμή του πεδίου password που ανακτήθηκε από το cloud database στο προηγούμενο βήμα προήλθε από το password που παρέχει ο χρήστης στο request body. Αυτό γίνεται με τη μέθοδο *compare()* του npm πακέτου *bcrypt*. Αν το password που παρέχει ο χρήστης είναι έγκυρο, η εκτέλεση της λειτουργίας συνεχίζεται. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http Unauthorized Error και η λειτουργία τερματίζεται.

Δημιουργία JWT

Με τη βοήθεια του npm πακέτου *jsonwebtoken*, δημιουργείται ένα JWT για τον χρήστη. Στο payload του JWT περιέχεται η τιμή του πεδίου _id που ανακτήθηκε από το cloud database. Το JWT υπογράφεται χρησιμοποιώντας τον αλγόριθμο RS256 και το ιδιωτικό κλειδί RSA του Node project(περισσότερα στο Κεφάλαιο 3.3) και έχει διάρκεια ζωής 20 λεπτών.

```
1.  // Create a Json Web Token for user.
2.  userSchema.methods.generateJWT = function() {
3.      return jwt.sign({id: this._id},
4.          RSA_PRIVATE_KEY, {
5.              algorithm: 'RS256',
6.              expiresIn: '20m'
7.          })
8.  }
```

Κομμάτι κώδικα 10: Δημιουργία JWT για τον χρήστη

Δημιουργία Session για τον χρήστη

Για το σκοπό αυτό χρησιμοποιήθηκαν τα HTTP Cookies. Πρόκειται για ένα μικρό κομμάτι δεδομένων που στέλνει ένας διακομιστής στο πρόγραμμα περιήγησης ιστού ενός χρήστη. Το πρόγραμμα περιήγησης αποθηκεύει το cookie και το στέλνει πίσω στον ίδιο διακομιστή με μεταγενέστερα αιτήματα. Αξιοποιώντας τη μέθοδο *res.cookie()* του Express, δημιουργείται ένα HTTP Set-Cookie header με το JWT του χρήστη. Η εκτέλεση της λειτουργίας ολοκληρώνεται με την επιστροφή στο χρήστη ενός Http OK Response στο οποίο περιέχεται και το header που δημιουργήθηκε.

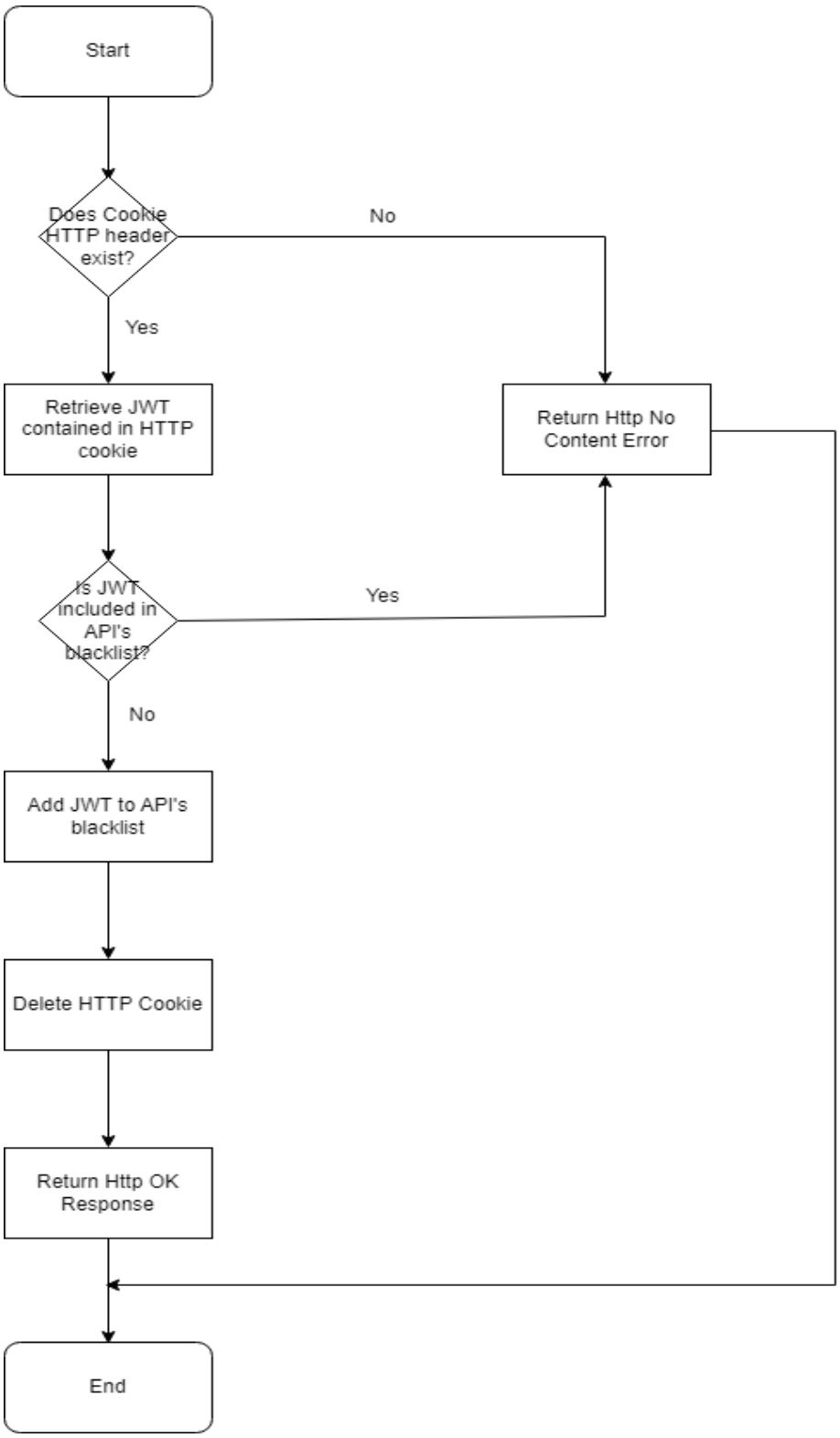
```
1.  exports.login = async (req,res) => {
2.      // Get required variables from request body.
3.      const {username, password} = req.body
4.      try {
5.          // Check if user exists. If no, return Unauthorized Error.
6.          const user = await User.findOne({username}).select('+password')
7.          if (!user) {
8.              return res.status(401).json({
9.                  message: 'No such user.'
10.             })
11.         }
12.         // If user exists, validate password.
13.         const valid = await bcrypt.compare(password, user.password)
14.         // If password is not valid, return Unauthorized Error
15.         if (!valid) {
16.             return res.status(401).json({
17.                 message: 'No such user.'
18.             })
19.         }
20.         // Otherwise, generate a unique token for the user
21.         // and send the token to the client.
22.         let options = {
23.             maxAge: 20 * 60 * 1000, // 20 minutes
24.             httpOnly: true,
25.             secure: true,
26.             sameSite: "None"
27.         }
28.         const token = user.generateJWT()
29.         res.cookie('SessionID', token, options)
30.         res.status(200).json({
31.             message: 'You have successfully logged in.'
32.         })
33.     } catch (err) {
34.         res.status(500).json({
35.             message: 'Internal Server Error'
36.         })
37.     }
38. }
```

Κομμάτι κώδικα 11: Είσοδος χρήστη

4.4 Αποσύνδεση χρήστη

4.4.1 Σχηματική απεικόνιση

Το παρακάτω διάγραμμα απεικονίζει τα βήματα που ακολουθούνται για την αποσύνδεση ενός χρήστη από το σύστημα.



Διάγραμμα 5: Διάγραμμα ροής για αποσύνδεση χρήστη

4.4.2 Υλοποίηση

Η εκτέλεση της λειτουργίας αυτής γίνεται με ένα http get request στο endpoint http://localhost:4000/api/v1/collaborativeFilteringSearch/logout.

Ανάκτηση Cookie HTTP header

Αξιοποιώντας τη μέθοδο *req.headers()* του Express, διερευνάται η ύπαρξη Cookie HTTP header στο request του χρήστη. Αν υπάρχει, τότε ανακτάται η τιμή του και η εκτέλεση της λειτουργίας συνεχίζεται. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http No Content Error και η λειτουργία τερματίζεται.

Ανάκτηση JWT χρήστη

Η ανακτηθείσα τιμή του Cookie HTTP header έχει την ακόλουθη δομή:

{cookie-name}={cookie-value};{cookie-options}. Το JWT του χρήστη είναι το {cookie-value}. Για την ανάκτησή του ακολουθούνται τα εξής βήματα:

- Χρησιμοποιώντας τη string μέθοδο *split()* με παράμετρο το '=', η ανακτηθείσα τιμή χωρίζεται στα ακόλουθα δύο μέρη: {cookie-name} και {cookie-value};{cookie-options}. Από αυτά, συγκρατείται το δεύτερο μέρος, καθώς εκεί περιέχεται η επιθυμητή τιμή.
- Χρησιμοποιώντας τη string μέθοδο *split()* με παράμετρο το ';', η τιμή που προέκυψε από το προηγούμενο βήμα χωρίζεται στα ακόλουθα δύο μέρη: {cookie-value} και {cookie-options}. Από αυτά, συγκρατείται το πρώτο μέρος που αποτελεί το JWT του χρήστη.

Έλεγχος εγκυρότητας JWT χρήστη

Για να είναι έγκυρο το JWT του χρήστη, θα πρέπει να μην περιέχεται στη μαύρη λίστα του API(collection *blacklists* του cloud database). Για το σκοπό αυτό, ελέγχεται αν υπάρχει στο εν λόγω collection εγγραφή που η τιμή του πεδίου token να ταυτίζεται με το JWT του χρήστη. Αν δεν υπάρχει, η εκτέλεση της λειτουργίας συνεχίζεται. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http No Content Error και η λειτουργία τερματίζεται.

Προσθήκη JWT χρήστη στη μαύρη λίστα του API

Δημιουργείται μία νέα έγγραφή στο collection *blacklists* του cloud database με το JWT του χρήστη.

Διαγραφή HTTP Cookie

Αξιοποιώντας τη μέθοδο *res.clearCookie()* του Express, διαγράφεται από το πρόγραμμα περιήγησης ιστού του χρήστη το Http Cookie που δημιουργήθηκε κατά την είσοδο του στο σύστημα. Η εκτέλεση της λειτουργίας ολοκληρώνεται με την επιστροφή στο χρήστη ενός Http OK Response.

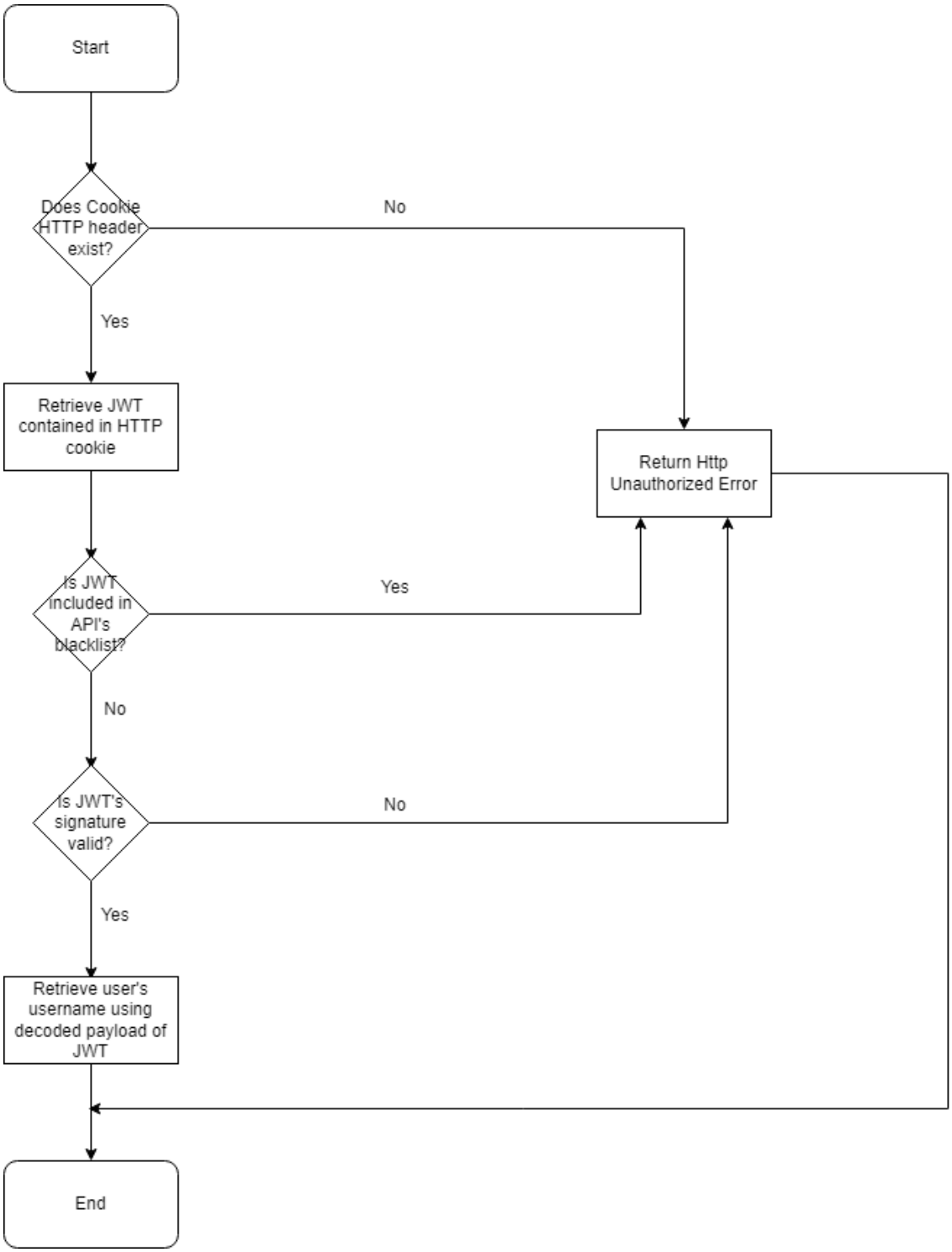
```
1.  exports.logout = async (req,res) => {
2.      try {
3.          // Get the session cookie from request header.
4.          const authHeader = req.headers['cookie']
5.          // If there is no cookie in request header,
6.          // return No Content Error.
7.          if (!authHeader) return res.sendStatus(204)
8.          // If there is, split the cookie string to get
9.          // the actual jwt token.
10.         const cookie = authHeader.split('=')[1]
11.         // This split removes cookie options.
12.         const accessToken = cookie.split(';')[0];
13.         // Check if retrieved token is blacklisted.
14.         const blacklisted = await Blacklist.findOne({token: accessToken})
15.         // If yes, return No Content Error.
16.         if (blacklisted) return res.sendStatus(204)
17.         // Otherwise, save retrieved token to Blacklist collection.
18.         const newBlacklist = new Blacklist({token: accessToken})
19.         await newBlacklist.save()
20.         // Clear cookie
21.         res.clearCookie('SessionID')
22.         // Return OK Response.
23.         res.status(200).json({
24.             message: 'You are logged out.'
25.         })
26.     } catch (err) {
27.         res.status(500).json({
28.             message: 'Internal Server Error'
29.         })
30.     }
31. }
```

Κομμάτι κώδικα 12: Αποσύνδεση χρήστη

4.5 Αυθεντικοποίηση χρήστη

4.5.1 Σχηματική απεικόνιση

Το παρακάτω διάγραμμα απεικονίζει τα βήματα που ακολουθούνται για να ελεγχθεί κατά πόσο ο χρήστης έχει πραγματοποιήσει είσοδο στο σύστημα.



Διάγραμμα 6: Διάγραμμα ροής για αυθεντικοποίηση χρήστη

4.5.2 Υλοποίηση

Σκοπός της λειτουργίας αυτής είναι να εξακριβωθεί κατά πόσο ο χρήστης έχει πραγματοποιήσει είσοδο στο σύστημα και κατ' επέκταση έχει δικαίωμα να χρησιμοποιήσει το ΣΣ με συνεργατικό φιλτράρισμα.

Ανάκτηση Cookie HTTP header

Αξιοποιώντας τη μέθοδο *req.headers()* του Express, διερευνάται η ύπαρξη Cookie HTTP header στο request του χρήστη. Αν υπάρχει, τότε ανακτάται η τιμή του και η εκτέλεση της λειτουργίας συνεχίζεται. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http Unauthorized Error.

Ανάκτηση JWT χρήστη

Η ανακτηθείσα τιμή του Cookie HTTP header έχει την ακόλουθη δομή:

{cookie-name}={cookie-value};{cookie-options}. Το JWT του χρήστη είναι το {cookie-value}. Για την ανάκτησή του ακολουθούνται τα εξής βήματα:

- Χρησιμοποιώντας τη string μέθοδο *split()* με παράμετρο το '=', η ανακτηθείσα τιμή χωρίζεται στα ακόλουθα δύο μέρη: {cookie-name} και {cookie-value};{cookie-options}. Από αυτά, συγκρατείται το δεύτερο μέρος, καθώς εκεί περιέχεται η επιθυμητή τιμή.
- Χρησιμοποιώντας τη string μέθοδο *split()* με παράμετρο το ';', η τιμή που προέκυψε από το προηγούμενο βήμα χωρίζεται στα ακόλουθα δύο μέρη: {cookie-value} και {cookie-options}. Από αυτά, συγκρατείται το πρώτο μέρος που αποτελεί το JWT του χρήστη.

Έλεγχος εγκυρότητας JWT χρήστη

Για να είναι έγκυρο το JWT του χρήστη, θα πρέπει να μην περιέχεται στη μαύρη λίστα του API(collection *blacklists* του cloud database). Για το σκοπό αυτό, ελέγχεται αν υπάρχει στο εν λόγω collection εγγραφή που η τιμή του πεδίου token να ταυτίζεται με το JWT του χρήστη. Αν δεν υπάρχει, η εκτέλεση της λειτουργίας συνεχίζεται. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http Unauthorized Error.

Έλεγχος υπογραφής JWT χρήστη

Αξιοποιώντας τη μέθοδο *verify()* του npm πακέτου *jsonwebtoken*, ελέγχεται η εγκυρότητα της υπογραφής του JWT. Για το σκοπό αυτό, παρέχονται ως παράμετροι στην εν λόγω μέθοδο τόσο ο αλγόριθμος που χρησιμοποιήθηκε για τη δημιουργία της υπογραφής(RS256) όσο και το δημόσιο κλειδί RSA του Node project(για τη δημιουργία της υπογραφής χρησιμοποιήθηκε το ιδιωτικό κλειδί RSA του Node project). Αν η υπογραφή του JWT είναι έγκυρη, η μέθοδος επιστρέφει το payload του JWT που περιέχει την τιμή του πεδίου _id της εγγραφής του collection *users* του cloud database με τα στοιχεία σύνδεσης του χρήστη. Χρησιμοποιώντας την τιμή αυτή, ανακτάται και αποθηκεύεται στο request το username του χρήστη. Διαφορετικά, επιστρέφεται στο χρήστη ένα Http Unauthorized Error.

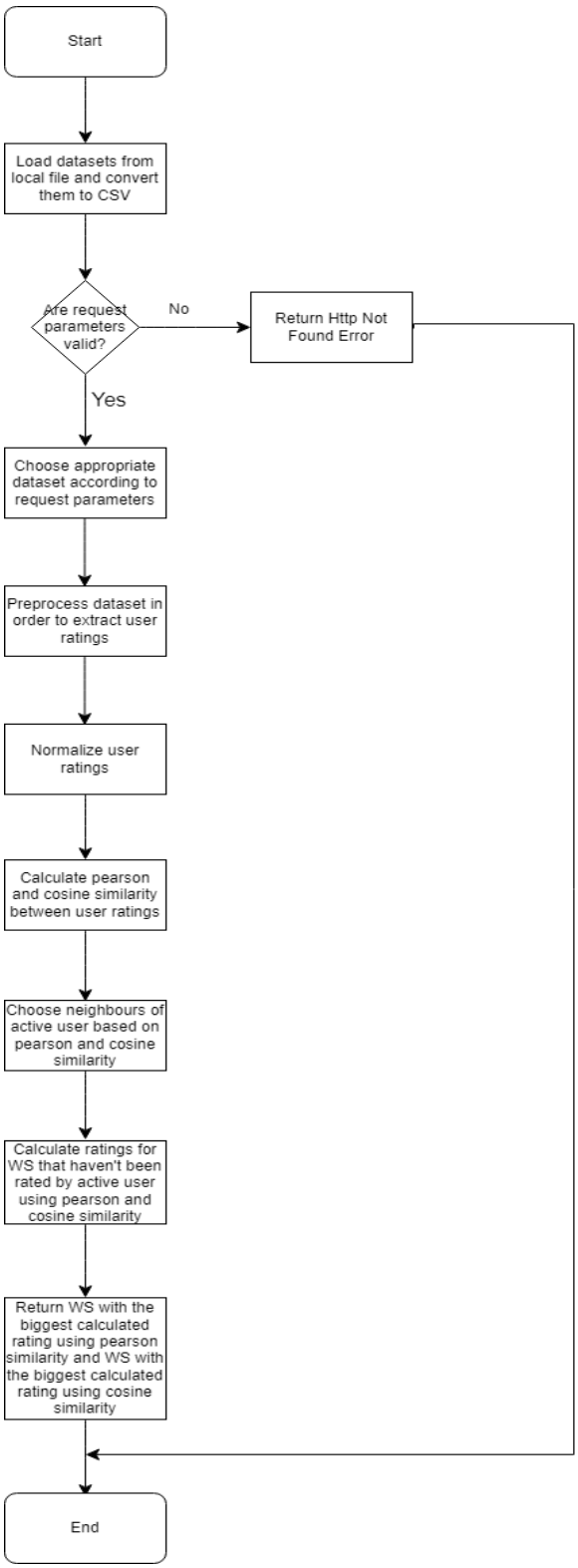
```
1.  exports.verify = async (req, res, next) => {
2.      try {
3.          // Get the session cookie from request header.
4.          const authHeader = req.headers['cookie']
5.          // If there is no cookie in request header,
6.          // return Unauthorized Error.
7.          if (!authHeader) return res.sendStatus(401)
8.          // If there is, split the cookie string to get
9.          // the actual jwt token.
10.         const cookie = authHeader.split('=')[1]
11.         // This split removes cookie options.
12.         const accessToken = cookie.split(';')[0];
13.         // Check if retrieved token is blacklisted.
14.         const blacklisted = await Blacklist.findOne({token: accessToken})
15.         // If yes, return Unauthorized Error and ask for re-authentication.
16.         if (blacklisted) {
17.             return res.status(401).json({
18.                 message: 'This session has expired. ' +
19.                 'Please login.'
20.             })
21.         }
22.         // Verify token.
23.         jwt.verify(accessToken, RSA_PUBLIC_KEY, {
24.             algorithms: ['RS256']},
25.             async (err, decoded) => {
26.                 if (err) {
27.                     return res.status(401).json({
28.                         message: 'This session has expired. ' +
29.                         'Please login.'
30.                     })
31.                 }
32.                 const user = await User.findById(decoded.id)
33.                 req.user = user.username
34.                 next()
35.             })
36.     } catch (err) {
37.         return res.status(500).json({
38.             message: 'Internal Server Error'
39.         })
40.     }
41. }
```

Κομμάτι κώδικα 13: Αυθεντικοποίηση χρήστη

4.6 ΣΣ με συνεργατικό φιλτράρισμα

4.6.1 Σχηματική απεικόνιση

Το παρακάτω διάγραμμα απεικονίζει τα βήματα που ακολουθεί το ΣΣ με συνεργατικό φιλτράρισμα της υλοποίησης για την παραγωγή συστάσεων.



Διάγραμμα 7: Διάγραμμα ροής για ΣΣ με συνεργατικό φιλτράρισμα

4.6.2 Υλοποίηση

Για τη λειτουργία αυτή, αναπτύχθηκε ένα ΣΣ με συνεργατικό φιλτράρισμα βάσει μνήμης που χρησιμοποιεί τεχνικές βασισμένες στο χρήστη. Η πρόσβαση στην εν λόγω λειτουργία επιτρέπεται μόνο αν έχει προηγηθεί είσοδος χρήστη στο σύστημα και η εκτέλεση της γίνεται με ένα http get request στο endpoint http://localhost:4000/api/v1/collaborativeFilteringSearch/{distribution}/{version}. Από το σύνολο των datasets που δημιουργήθηκαν στο πλαίσιο της παρούσας εργασίας, αξιοποιήθηκαν τα datasets για συνεργατικό φιλτράρισμα(περισσότερα στο Κεφάλαιο 3.1.2). Σε κάθε εκτέλεση της λειτουργίας χρησιμοποιούνται το dataset με κατηγοριοποιημένες και βαθμολογημένες υπηρεσίες ιστού για την επιστροφή πληροφοριών στο χρήστη σχετικά με τις προτεινόμενες υπηρεσίες ιστού και ένα από τα τέσσερα datasets με αξιολογήσεις χρηστών για την εκτέλεση του αλγορίθμου. Η επιλογή ανάμεσα στα τέσσερα datasets με αξιολογήσεις χρηστών σε κάθε εκτέλεση της λειτουργίας γίνεται με βάση τις τιμές των παραμέτρων {distribution} και {version}.

Εισαγωγή datasets

Κατά την εκκίνηση του API, τα datasets για συνεργατικό φιλτράρισμα φορτώνονται από το τοπικό αρχείο excel του Node project(περισσότερα στο Κεφάλαιο 3.3). Για διευκόλυνση προσπέλασης των δεδομένων, τα datasets με αξιολογήσεις χρηστών μετατρέπονται σε CSV και το dataset με κατηγοριοποιημένες και βαθμολογημένες υπηρεσίες ιστού μετατρέπεται σε JSON. Τα παραπάνω επιτυγχάνονται αξιοποιώντας το npm πακέτο *xlsx.*

```
1.  // Read excel file containing the dataset for this search techique.
2.  const workbook = xlsx.readFile(path.join(__dirname, "../", 'datasets/datasets.xlsx'))
3.  // Gather the names of all sheets contained in excel file.
4.  const workbookSheets = workbook.SheetNames
5.  // Convert the data contained in second sheet(list of web services) into JSON.
6.  const wsList = xlsx.utils.sheet_to_json(workbook.Sheets[workbookSheets[1]])
7.  // Convert the data contained in third sheet(user ratings that follow normal distribution-Version 1) into CSV.
8.  const ratingsNDV1 = xlsx.utils.sheet_to_csv(workbook.Sheets[workbookSheets[2]])
9.  // Convert the data contained in fourth sheet(user ratings  that follow normal distribution-Version 2) into CSV.
10. const ratingsNDV2 = xlsx.utils.sheet_to_csv(workbook.Sheets[workbookSheets[3]])
11. // Convert the data contained in fourth sheet(user ratings  that follow chi-square distribution-Version 1) into
12. // CSV.
13. const ratingsCSDV1 = xlsx.utils.sheet_to_csv(workbook.Sheets[workbookSheets[4]])
14. // Convert the data contained in fourth sheet(user ratings  that follow chi-square distribution-Version 2) into
15. // CSV.
16. const ratingsCSDV2 = xlsx.utils.sheet_to_csv(workbook.Sheets[workbookSheets[5]])
```

Κομμάτι κώδικα 14: Εισαγωγή datasets για συνεργατικό φιλτράρισμα

Έλεγχος παραμέτρων request

Μόλις το API λάβει το http get request του χρήστη στο κατάλληλο endpoint, ελέγχεται αν οι παράμετροι {distribution} και {version} έχουν έγκυρες τιμές. Πιο συγκεκριμένα:

- **{distribution}.** Πρόκειται για την κατανομή που ακολουθούν οι αξιολογήσεις χρηστών. Οι επιτρεπτές τιμές είναι normal(για κανονική κατανομή) και chi-square(για κατανομή x^2).
- **{version}.** Πρόκειται για την παραλλαγή χρηστών που χρησιμοποιείται. Οι επιτρεπτές τιμές είναι 1(για την πρώτη παραλλαγή - κάθε χρήστης μπορεί να ενδιαφέρεται για πολλαπλές κατηγορίες υπηρεσιών ιστού) και 2(για την δεύτερη παραλλαγή - κάθε χρήστης ενδιαφέρεται μόνο για μία κατηγορία υπηρεσιών ιστού).

Αν οι παράμετροι του request πληρούν τις παραπάνω προυποθέσεις, τότε με βάση τις τιμές τους επιλέγεται ποιο από τα τέσσερα datasets με αξιολογήσεις χρηστών θα χρησιμοποιηθεί στη συνέχεια(π.χ. distribution=normal, version = 1 => dataset με αξιολογήσεις χρηστών πρώτης

παραλλαγής που ακολουθούν την κανονική κατανομή) καθώς και ποια μέθοδος προεπεξεργασίας θα εφαρμοστεί σε αυτό το dataset.

```
exports.checkParameters = (req, res, next) => {
    const distribution = req.params.distribution
    const version = req.params.version
    // Check if parameters have valid values.
    // If no, return Not Found Error.
    if ((distribution != 'normal' && distribution != 'chi-square') || (version != '1' && version != '2')) {
        return res.sendStatus(404)
    }
    // Otherwise, according to parameters, call the corresponding preprocess method
    // with the appropriate arguements.
    if (distribution == 'normal' && version == '1') {
        [req.ratingsOnly, req.activeUserIndex] = preprocessV1(req.user, ratingsNDV1)
    } else if (distribution == 'normal' && version == '2') {
        [req.ratingsOnly, req.activeUserIndex] = preprocessV2(req.user, ratingsNDV2)
    } else if (distribution == 'chi-square' && version == '1') {
        [req.ratingsOnly, req.activeUserIndex] = preprocessV1(req.user, ratingsCSDV1)
    } else {
        [req.ratingsOnly, req.activeUserIndex] = preprocessV2(req.user, ratingsCSDV2)
    }
    next()
}
```

Κομμάτι κώδικα 15: Έλεγχος παραμέτρων request

Προεπεξεργασία dataset

Αναπτύχθηκαν δύο μέθοδοι για το σκοπό αυτό, μία για την πρώτη παραλλαγή χρηστών και μία για την δεύτερη παραλλαγή χρηστών. Σκοπός και των 2 μεθόδων είναι η εξαγωγή από το dataset με αξιολογήσεις χρηστών μία μήτρας χρήστη-υπηρεσίας ιστού όπου κάθε γραμμή της αναπαριστά έναν χρήστη, κάθε στήλη της αναπαριστά μία υπηρεσία ιστού και κάθε στοιχείο της (i, j) είναι η αξιολόγηση του χρήστη i για την υπηρεσία ιστού j(με πιθανές τιμές: 1, 2, 3, 4, 5) ή 0 σε περίπτωση που ο χρήστης i δεν έχει αξιολογήσει την υπηρεσία ιστού j.

Προεπεξεργασία dataset με αξιολογήσεις χρηστών πρώτης παραλλαγής

Όπως αναφέρθηκε προηγουμένως, το dataset έχει μετατραπεί σε CSV με κάθε γραμμή του CSV να περιέχει τις τιμές της αντίστοιχης γραμμής του excel χωρισμένες μεταξύ τους με κόμμα. Αρχικά, το dataset μετασχηματίζεται από CSV σε δισδιάστατη μήτρα ακολουθώντας την εξής διαδικασία:

1. Το CSV μετασχηματίζεται σε μία μονοδιάστατη μήτρα με κάθε στοιχείο της που βρίσκεται σε μία θέση k να περιέχει την k-οστή γραμμή του CSV.
2. Κάθε στοιχείο της μήτρας που προέκυψε από το προηγούμενο βήμα μετασχηματίζεται σε μία μονοδιάστατη μήτρα με κάθε στοιχείο της που βρίσκεται σε μία θέση l να περιέχει την l-οστή τιμή της αντίστοιχης γραμμής του CSV. Έτσι, προκύπτει μία μήτρα μητρών όπου κάθε στοιχείο της που βρίσκεται σε μία θέση m περιέχει μία μονοδιάστατη μήτρα με τις τιμές της m-οστής γραμμής του CSV.

Στη συνέχεια, αφαιρείται η πρώτη γραμμή της δισδιάστατης μήτρας, καθώς περιέχει τα ονόματα των στηλών του excel. Αυτό έχεις ως αποτέλεσμα, κάθε γραμμή της δισδιάστατης μήτρας να έχει την ακόλουθη μορφή: [username, αξιολόγηση υπηρεσίας ιστού 1, αξιολόγηση υπηρεσίας ιστού 2, …, αξιολόγηση υπηρεσίας ιστού 117]. Μετά, εντοπίζεται και αποθηκεύεται η γραμμή της δισδιάστατης μήτρας στην οποία περιέχονται οι αξιολογήσεις του ενεργού χρήστη(αυτού που έχει πραγματοποιήσει είσοδο στο σύστημα). Έπειτα, από κάθε γραμμή της δισδιάστατης μήτρας αφαιρείται το πρώτο στοιχείο(δηλαδή το username του χρήστη στον οποίο αντιστοιχούν οι

αξιολογήσεις). Έτσι, η δισδιάστατη μήτρα έχει πλέον την επιθυμητή δομή, αλλά οι τιμές της είναι σε μορφή string. Για να διορθωθεί αυτό, γίνονται τα εξής:

1. Η απουσία αξιολόγησης ενός χρήστη για μία υπηρεσία ιστού αναπαρίσταται με ένα empty string(''). Κάθε empty string εντός της δισδιάστατης μήτρας αντικαθίσταται από την τιμή '0'.
2. Κάθε τιμή της δισδιάστατης μετατρέπεται από string σε αριθμό.

```
1.  function preprocessV1(user, ratings) {
2.      // Convert CSV to 2d Array.
3.      const rows = ratings.split('\n')
4.      const twoDimensionalArray = rows.map(row => row.split(','))
5.      // Delete the first line of the array
6.      // (it contains the names of the columns).
7.      const ratingsFull = twoDimensionalArray.slice(0)
8.      ratingsFull.splice(0,1)
9.      // Find the index of the active user.
10.     const activeUserIndex = ratingsFull.findIndex(r => r.includes(user))
11.     // Retrieve only the ratings of the users.
12.     let ratingsOnly = ratingsFull.map(rf => rf.slice(1))
13.     // Replace null ratings with 0.
14.     ratingsOnly = ratingsOnly.map(ro => ro.map(r => r.replace('','0')))
15.     // Convert all ratings to numbers.
16.     ratingsOnly = ratingsOnly.map(ro => ro.map(r => Number(r)))
17.     return [ratingsOnly, activeUserIndex]
18.
19. }
```

Κομμάτι κώδικα 16: Προεπεξεργασία dataset με αξιολογήσεις χρηστών της πρώτης παραλλαγής

Προεπεξεργασία dataset με αξιολογήσεις χρηστών δεύτερης παραλλαγής

Εφαρμόστηκαν τα ίδια δύο πρώτα βήματα με την παραπάνω μέθοδο και από αυτά προέκυψε μία δισδιάστατη μήτρα κάθε γραμμή της οποίας έχει την ακόλουθη μορφή: [username, κατηγορία υπηρεσιών ιστού, αξιολόγηση υπηρεσίας ιστού 1, αξιολόγηση υπηρεσίας ιστού 2, …, αξιολόγηση υπηρεσίας ιστού 117]. Στη συνέχεια, εντοπίζεται η γραμμή της δισδιάστατης μήτρας στην οποία περιέχονται οι αξιολογήσεις του ενεργού χρήστη, ανακτάται και αποθηκεύεται η κατηγορία υπηρεσιών ιστού για την οποία αυτός ενδιαφέρεται. Μετά, δημιουργείται μία νέα δισδιάστατη μήτρα στην οποία περιέχονται μόνο οι αξιολογήσεις των χρηστών που ενδιαφέρονται για την ίδια κατηγορία υπηρεσιών ιστού με τον ενεργό χρήστη. Έπειτα, από κάθε γραμμή της νέας δισδιάστατης μήτρας αφαιρούνται τα δύο πρώτα στοιχεία(δηλαδή το username του χρήστη στον οποίο αντιστοιχούν οι αξιολογήσεις και η κατηγορία υπηρεσιών ιστού για την οποία ενδιαφέρεται). Έτσι, η δισδιάστατη μήτρα έχει πλέον την επιθυμητή δομή, αλλά οι τιμές της είναι σε μορφή string. Για να διορθωθεί αυτό, ακολουθούνται τα ίδια βήματα με την παραπάνω μέθοδο.

```
function preprocessV2(user, ratings) {
    // Convert CSV to 2d Array.
    const rows = ratings.split('\n')
    const twoDimensionalArray = rows.map(row => row.split(','))
    // Delete the first line of the array
    // (it contains the names of the columns).
    const datasetArray = twoDimensionalArray.slice(0)
    datasetArray.splice(0,1)
    // Find the type of the active user(logged in user).
    const row = datasetArray.findIndex(r => r.includes(user))
    const type = datasetArray[row][1]
    // Find all the users that are the same type as the active one.
    const ratingsFull = datasetArray.filter((user) => user[1] == type)
    // Find the index of the active user.
    const activeUserIndex = ratingsFull.findIndex(r => r.includes(user))
    // Retrieve only the ratings of the users.
    let ratingsOnly = ratingsFull.map(rf => rf.slice(2))
    // Replace null ratings with 0.
    ratingsOnly = ratingsOnly.map(ro => ro.map(r => r.replace('','0')))
    // Convert all ratings to numbers.
    ratingsOnly = ratingsOnly.map(ro => ro.map(r => Number(r)))
    return [ratingsOnly, activeUserIndex]
}
```

Κομμάτι κώδικα 17: Προεπεξεργασία dataset με αξιολογήσεις χρηστών της δεύτερης παραλλαγής

Κανονικοποίηση αξιολογήσεων χρηστών

Η εφαρμογή του εν λόγω μετασχηματισμού επιφέρει μεγαλύτερη αντικειμενικότητα στη διαδικασία αξιολόγησης. Αν R : [10] x [117] → {0, 1, 2, 3, 4, 5} είναι η μήτρα χρήστη-υπηρεσίας ιστού με τις αξιολογήσεις, τότε η αξιολόγηση του χρήστη i για την υπηρεσία ιστού j με i ∈ [10] και j ∈ [117] κανονικοποιείται χρησιμοποιώντας την ακόλουθη συνάρτηση:

$$n(i, j) = r_{ij} - \frac{\sum_{w=1}^{117} r_{iw}}{117}.$$

Ουσιαστικά, η κανονικοποιημένη τιμή μιας αξιολόγησης χρήστη για μία υπηρεσία ιστού προκύπτει από τη διαφορά μεταξύ της αξιολόγησης και του μέσου όρου των αξιολογήσεων του χρήστη για όλες τις υπηρεσίες ιστού. Η υλοποίηση της κανονικοποίησης γίνεται ακολουθώντας τα εξής βήματα:

1. Η δισδιάστατη μήτρα που πρόεκυψε από την προεπεξεργασία του dataset προσπελάζεται ανά γραμμή-χρήστη.
2. Για κάθε γραμμή, υπολογίζεται ο μέσος όρος των στοιχείων της.
3. Κάθε γραμμή προσπελάζεται ανά στοιχείο-αξιολόγηση και για κάθε στοιχείο υπολογίζεται η διαφορά μεταξύ αυτού και του μέσου όρου των στοιχείων της γραμμής.
4. Τα αποτελέσματα ανά γραμμή αποθηκεύονται σε μία νέα μήτρα.

```
function normalizeRatings(ratings) {
    // Normalize ratings.
    let normalizedRatings = []
    ratings.forEach(ra => {
        // Calculate the average of user ratings.
        let avg = ra.reduce((a,b) => a + b) / ra.length
        // Subtract from each rating the average calculated above.
        normalizedRatings.push(ra.map(r => r - avg))
    })
    return normalizedRatings}
```

Κομμάτι κώδικα 18: Κανονικοποίηση αξιολογήσεων χρηστών

Υπολογισμός ομοιότητας μεταξύ χρηστών

Ο υπολογισμός ομοιότητας μεταξύ δύο χρηστών γίνεται μέσω της σύγκρισης των αξιολογήσεων τους. Από τις διάφορες μετρικές που υπάρχουν για το σκοπό αυτό, αποφασίστηκε να χρησιμοποιηθούν η συσχέτιση Pearson και η ομοιότητα συνημιτόνου. Η χρήση δύο διαφορετικών μετρικών αποσκοπεί στο να εξεταστεί ο βαθμός που η μετρική επηρεάζει την ποιότητα και την ποικιλία των συστάσεων που παράγονται.

Συσχέτιση Pearson

Αν X:[1] x [117] → {0, 1, 2, 3, 4, 5} και Y:[1] x [117] → {0, 1, 2, 3, 4, 5} είναι μήτρες που περιέχουν τις αξιολογήσεις δύο χρηστών για τις 117 υπηρεσίες ιστού, τότε η συσχέτιση Pearson μεταξύ τους υπολογίζεται χρησιμοποιώντας την ακόλουθη συνάρτηση:

$$pcc(X, Y) = \frac{\sum_{i=1}^{117}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{117}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{117}(y_i - \bar{y})^2}}.$$

Η παραπάνω συνάρτηση έχει πεδίο τιμών το [-1, 1] με το 1 να αναπαριστά την τέλεια θετική συσχέτιση και το -1 να αναπαριστά την τέλεια αρνητική συσχέτιση. Οι μήτρες X, Y περιέχουν τις αξιολογήσεις των χρηστών πριν την κανονικοποίηση, καθώς ο εν λόγω μετασχηματισμός πραγματοποιείται από τον τύπο της συσχέτισης Pearson. Η υλοποίηση της διαδικασίας αυτής γίνεται ως εξής:

1. Η δισδιάστατη μήτρα που πρόεκυψε από την προεπεξεργασία του dataset προσπελάζεται ανά γραμμή-χρήστη.
2. Για κάθε γραμμή, υπολογίζεται η συσχέτιση Pearson με όλες τις γραμμές της μήτρας(συμπεριλαμβανομένου και του εαυτού της). Ο υπολογισμός αυτός πραγματοποιείται αξιοποιώντας το npm πακέτο *calculate-correlation.*
3. Τα αποτελέσματα ανά γραμμή αποθηκεύονται σε μία νέα δισδιάστατη μήτρα.

```
1.  function calculatePearsonSimilarity(ratings) {
2.      // Calculate pearson similarity between user ratings.
3.      // For details, check: https://www.npmjs.com/package/calculate-correlation.
4.      let similarityArray = []
5.      for (let i = 0; i < ratings.length; i++) {
6.          let similarityArrayLine = []
7.          for (let j = 0; j < ratings.length; j++) {
8.              if (i == j) {
9.                  similarityArrayLine.push(1)
10.             } else {
11.                 similarityArrayLine.push(calculateCorrelation(ratings[i], ratings[j]))
12.             }
13.         }
14.         similarityArray.push(similarityArrayLine)
15.     }
16.     return similarityArray
17. }
```

Κομμάτι κώδικα 19: Υπολογισμός συσχέτισης Pearson μεταξύ των αξιολογήσεων των χρηστών

Ομοιότητα συνημιτόνου

Αν X:[1] x [117] → {0, 1, 2, 3, 4, 5} και Y:[1] x [117] → {0, 1, 2, 3, 4, 5} είναι μήτρες που περιέχουν τις αξιολογήσεις δύο χρηστών για τις 117 υπηρεσίες ιστού, τότε η ομοιότητα συνημιτόνου μεταξύ τους υπολογίζεται χρησιμοποιώντας την ακόλουθη συνάρτηση:

$$cs(X, Y) = \frac{\sum_{i=1}^{117} x_i y_i}{\sqrt{\sum_{i=1}^{117} x_i^2}\sqrt{\sum_{i=1}^{117} y_i^2}}$$

Η παραπάνω συνάρτηση έχει πεδίο τιμών το [-1, 1] με το 1 να αναπαριστά την τέλεια θετική συσχέτιση και το -1 να αναπαριστά την τέλεια αρνητική συσχέτιση. Οι μήτρες X, Y περιέχουν τις αξιολογήσεις των χρηστών πριν την κανονικοποίηση, καθώς σε διαφορετική περίπτωση η τιμή της ομοιότητας συνημιτόνου θα ταυτιζόταν με την τιμή της συσχέτισης Pearson. Η υλοποίηση της διαδικασίας αυτής γίνεται ως εξής:

1. Η δισδιάστατη μήτρα που πρόεκυψε από την προεπεξεργασία του dataset προσπελάζεται ανά γραμμή-χρήστη.
2. Για κάθε γραμμή, υπολογίζεται η ομοιότητα συνημιτόνου με όλες τις γραμμές της γραμμές της μήτρας(συμπεριλαμβανομένου και του εαυτού της). Ο υπολογισμός αυτός πραγματοποιείται αξιοποιώντας το npm πακέτο *compute-cosine-similarity*.
3. Τα αποτελέσματα ανά γραμμή αποθηκεύονται σε μία νέα δισδιάστατη μήτρα.

```
function calculateCosineSimilarity(ratings) {
    // Calculate cosine similarity betweeen user ratings.
    //For details, check: https://www.npmjs.com/package/compute-cosine-similarity.
    let similarityArray = []
    for (let i = 0; i < ratings.length; i++) {
        let similarityArrayLine = []
        for (let j = 0; j < ratings.length; j++) {
            similarityArrayLine.push(similarity(ratings[i], ratings[j]))
        }
        similarityArray.push(similarityArrayLine)
    }
    return similarityArray
}
```

Κομμάτι κώδικα 20: Υπολογισμός ομοιότητας συνημιτόνου μεταξύ των αξιολογήσεων των χρηστών

Επιλογή γειτόνων του ενεργού χρήστη

Η επιλογή των γειτόνων του ενεργού χρήστη γίνεται με τη χρήση της μεθόδου του κατωφλιού, δηλαδή επιλέγονται οι χρήστες που το μέτρο ομοιότητας τους με τον ενεργό χρήστη ξεπερνά ένα προκαθορισμένο όριο(κατώφλι). Στην περίπτωση μας, η επιλογή γειτόνων έγινε δύο φορές, μία με βάση τη συσχέτιση Pearson και μία με βάση την ομοιότητα συνημιτόνου. Και στις δύο περιπτώσεις κατώφλι θεωρήθηκε το 0,1. Η υλοποίηση της διαδικασίας αυτής γίνεται ως εξής:

1. Η γραμμή της δισδιάστατης μήτρας που προέκυψε από τον υπολογισμό της ομοιότητας μεταξύ των χρηστών(με συσχέτιση Pearson ή ομοιότητα συνημιτόνου) και περιέχει τις μετρήσεις ομοιότητας του ενεργού χρήστη προσπελάζεται ανά στοιχείο - ομοιότητα με άλλο χρήστη.
2. Κάθε στοιχείο υποβάλλεται στους ακόλουθους ελέγχους:
    - Η θέση του στοιχείου στη γραμμή να μην ταυτίζεται με τον αριθμό της γραμμής. Αν δεν ισχύει, το στοιχείο αυτό αποτελεί την ομοιότητα του ενεργού χρήστη με τον εαυτό του.
    - Η τιμή του στοιχείου να είναι μεγαλύτερη ή ίση από την τιμή του κατωφλίου.

    Αν το στοιχείο περάσει επιτυχώς τους παραπάνω ελέγχους, τότε η θέση του προστίθεται σε μία μονοδιάστατη μήτρα με τους γείτονες του ενεργού χρήστη. Διαφορετικά, απορρίπτεται και ελέγχεται το επόμενο στοιχείο.

```
function chooseNeighbours(similarities, userIndex, threshold) {
    // Find similar neighbours using the threshold method.
    let neighbours = []
    for (let i = 0; i < similarities.length; i++) {
        if (i != userIndex && similarities[i] >= threshold) {
            neighbours.push(i)
        }
    }
    return neighbours
}
```

Κομμάτι κώδικα 21: Επιλογή γειτόνων του ενεργού χρήστη

Επιλογή υπηρεσιών ιστού και ανάκτηση απαραίτητων δεδομένων για βαθμολόγηση

Ενδιαφερόμαστε για τις υπηρεσίες ιστού που δεν έχουν αξιολογηθεί από τον ενεργό χρήστη. Για τον εντοπισμό τους χρησιμοποιείται η μήτρα χρήστη-υπηρεσίας ιστού με τις αξιολογήσεις που προέκυψε από την προεπεξεργασία του dataset. Πιο συγκεκριμένα, προσπελάζεται η γραμμή της μήτρας που περιέχει τις αξιολογήσεις του ενεργού χρήστη και για κάθε στοιχείο-αξιολόγηση ελέγχεται η ισότητα με το μηδέν. Τα στοιχεία που έχουν τιμή 0 δεν έχουν αξιολογηθεί από τον ενεργό χρήστη και η θέση τους αποθηκεύεται σε μία μονοδιάστατη μήτρα.

```
// Find the web services that have not been rated by the active user.
const nonRatedWS = ratingsOnly[activeUserIndex].reduce((ws, current, index) => {
    if (current == 0) {
        ws.push(index)
    }
    return ws
}, [])
```

Κομμάτι κώδικα 22: Εντοπισμός υπηρεσιών ιστού που δεν έχουν αξιολογηθεί από τον ενεργό χρήστη

Για αυτές τις υπηρεσίες ιστού, θα υπολογιστεί μία βαθμολογία με βάση τις αξιολογήσεις των γειτόνων. Για να συμβεί αυτό, θα πρέπει να ανακτηθούν οι αξιολογήσεις των γειτόνων για αυτές τις υπηρεσίες ιστού, οι συσχετίσεις Pearson και οι ομοιότητες συνημιτόνων των γειτόνων με τον ενεργό χρήστη. Η ανάκτηση των δεδομένων αυτών υλοποιείται ως εξής:

1. Η μονοδιάστατη μήτρα με τους γείτονες του ενεργού χρήστη προσπελάζεται ανά στοιχείο-γείτονα.
2. Για κάθε γείτονα, προσπελάζεται η μονοδιάστατη μήτρα με τις επιλεγμένες υπηρεσίες ιστού και για κάθε μία από αυτές ανακτάται η κανονικοποιημένη αξιολόγηση του γείτονα για αυτή και αποθηκεύεται σε μονοδιάστατη μήτρα.
3. Αφού ολοκληρωθεί η προσπέλαση της μήτρας με τις επιλεγμένες υπηρεσίες, η μονοδιάστατη μήτρα με τις αξιολογήσεις του γείτονα για τις υπηρεσίες αυτές αποθηκεύεται σε δισδιάστατη μήτρα.
4. Ανακτάται και αποθηκεύεται σε μονοδιάστατη μήτρα η ομοιότητα του ενεργού χρήστη με τον τρέχοντα γείτονα.

Από τη διαδικασία αυτή προκύπτουν οι εξής μήτρες:

- Η μήτρα γείτονα-επιλεγμένης υπηρεσίας ιστού, που δημιουργήθηκε χάρη στην εκτέλεση του βήματος 3 για όλους τους γείτονες και περιέχει τις κανονικοποιημένες αξιολογήσεις των γειτόνων του ενεργού χρήστη για τις επιλεγμένες υπηρεσίες ιστού.
- Μία μονοδιάστατη μήτρα με τις ομοιότητες του ενεργού χρήστη με τους γείτονες του.

Η διαδικασία εκτελείται δύο φορές, μία φορά θεωρώντας ως μέτρο ομοιότητας τη συσχέτιση Pearson και μία θεωρώντας ως μέτρο ομοιότητας την ομοιότητα συνημιτόνου.

```
1.  function retrieveNeighboursRatingsAndSimilarities(ratings, neighbours, ws, similarities, userIndex) {
2.      // Retrieve normalized ratings of active user's neighbours for the services
3.      // that he/she has not rated.
4.      let neighboursRatings = []
5.      // Retrieve similarity scores of neighbours.
6.      let neighboursSimilarities = []
7.      neighbours.forEach(n => {
8.          const neighbourRatings = ws.map(s => ratings[n][s])
9.          neighboursRatings.push(neighbourRatings)
10.         neighboursSimilarities.push(similarities[userIndex][n])
11.     })
12.     return [neighboursRatings, neighboursSimilarities]
13. }
```

Κομμάτι κώδικα 23: Ανάκτηση απαραίτητων δεδομένων για βαθμολόγηση επιλεγμένων υπηρεσιών ιστού

Βαθμολόγηση επιλεγμένων υπηρεσιών ιστού

Για το σκοπό αυτό, θα χρησιμοποιηθεί ο σταθμισμένος μέσος όρος. Έστω n το πλήθος των γειτόνων του ενεργού χρήστη, k το πλήθος των επιλεγμένων υπηρεσιών ιστού και u ∈ [10] ο ενεργός χρήστης. Αν S: [1] x [n] → R είναι η μήτρα με τις ομοιότητες του ενεργού χρήστη με τους γείτονες του, V: [n] x [k] → R είναι η μήτρα με τις κανονικοποιημένες αξιολογήσεις των γειτόνων του ενεργού χρήστη για τις επιλεγμένες υπηρεσίες ιστού και R: [10] x [117] → {0, 1, 3, 4, 5} είναι η μήτρα με τις αξιολογήσεις των χρηστών για τις υπηρεσίες ιστού, τότε ο βαθμός μίας επιλεγμένης υπηρεσίας ιστού i με i ∈ [k] υπολογίζεται από την συνάρτηση:

$$\text{score}(u, i) = \bar{r_u} + \frac{\sum_{j=1}^{n} s_j V_{ji}}{\sum_{j=1}^{n} s_j}.$$

Η βαθμολόγηση των επιλεγμένων υπηρεσιών ιστού γίνεται ως εξής:

1. Η μήτρα με τις κανονικοποιημένες αξιολογήσεις των γειτόνων του ενεργού χρήστη για τις επιλεγμένες υπηρεσίες ιστού προσπελάζεται ανά γραμμή-γείτονα.
2. Για κάθε γραμμή-γείτονα, όλα τα στοιχεία-αξιολογήσεις που περιέχει πολλαπλασιάζονται με την ομοιότητα του ενεργού χρήστη με αυτό το γείτονα. Τα αποτελέσματα αποθηκεύονται σε δισδιάστατη μήτρα.
3. Υπολογίζεται το άθροισμα των στοιχείων κάθε στήλης-επιλεγμένης υπηρεσίας ιστού της δισδιάστατης μήτρας που προέκυψε από το βήμα 2. Τα αποτελέσματα αποθηκεύονται σε μονοδιάστατη μήτρα.
4. Υπολογίζεται και αποθηκεύεται το άθροισμα των ομοιοτήτων του ενεργού χρήστη με τους γείτονες του.
5. Προσπελάζεται ανά στοιχείο η μονοδιάστατη μήτρα που προέκυψε από το βήμα 3, η τιμή κάθε στοιχείου της διαιρείται με το άθροισμα που υπολογίστηκε στο βήμα 4 και στη συνέχεια στο αποτέλεσμα προστίθεται ο μέσος όρος των αξιολογήσεων του ενεργού χρήστη. Όλες οι υπολογισθείσες τιμές αποθηκεύονται σε μονοδιάστατη μήτρα και αποτελούν τους βαθμούς των επιλεγμένων υπηρεσιών ιστών.

Στη μονοδιάστατη μήτρα που προέκυψε εντοπίζεται και επιστρέφεται η θέση του μεγαλύτερου βαθμού. Σε περίπτωση που περισσότερες από μία υπηρεσίες ιστού έχουν πετύχει τον μεγαλύτερο βαθμό, εντοπίζεται και επιστρέφεται η θέση που βρίσκεται πιο κοντά στην αρχή της μήτρας. Η διαδικασία βαθμολόγησης εκτελείται δύο φορές, μία χρησιμοποιώντας ως μετρική ομοιότητας την συσχέτιση Pearson και μία χρησιμοποιώντας ως μετρική ομοιότητας την ομοιότητα συνημιτόνου.

```
1.  function rateWS(ratings, similarities, avg, message) {
2.      // Multiply neighbour's ratings with the corresponding similarity score.
3.      let temp = ratings.map((rat, index) => rat.map((r) => r * similarities[index]))
4.      // Add above results per web service.
5.      temp = temp.reduce((acc, current) => acc.map((r, index) => r + current[index]))
6.      // Find the sum of elements in similarities array.
7.      const sum = similarities.reduce((a, b) => a + b)
8.      // Divide elements of temp array by above sum and add the user's average rating.
9.      const scores = temp.map((v) => avg + v/sum)
10.     console.log(message)
11.     console.log(scores)
12.     // Find the max score.
13.     const max = Math.max(...scores)
14.     // Return the index of max element in scores array.
15.     return scores.indexOf(max)
16. }
```

Κομμάτι κώδικα 24: Βαθμολόγηση επιλεγμένων υπηρεσιών ιστού και εντοπισμός μεγαλύτερης βαθμολογίας

Επιστροφή συστάσεων

Όπως αναφέρθηκε νωρίτερα, το παραπάνω βήμα εκτελείται δύο φορές, μία για κάθε μετρική ομοιότητας και κάθε φορά επιστρέφεται η θέση της μεγαλύτερης βαθμολογίας και κατ' επέκταση η θέση της υπηρεσίας ιστού στην οποία αυτή αντιστοιχεί στη μονοδιάστατη μήτρα με τις επιλεγμένες υπηρεσίες ιστού. Προσπελάζοντας αυτές τις θέσεις της μήτρας με τις επιλεγμένες υπηρεσίες ιστού, ανακτάται η θέση των επιλεγμένων υπηρεσιών ιστού στη δισδιάστατη μήτρα χρήστη-υπηρεσίας ιστού που προέκυψε από την προεπεξεργασία του dataset. Η θέση αυτή χρησιμοποιείται για την ανάκτηση και επιστροφή των αποθηκευμένων πληροφοριών για αυτές τις υπηρεσίες ιστού στο dataset με κατηγοριοποιημένες και βαθμολογημένες υπηρεσίες ιστού.

Κεφάλαιο 5 – Παρουσίαση λειτουργιών εφαρμογής

5.1 Εγκατάσταση και εκτέλεση εφαρμογής

Για την εγκατάσταση των npm πακέτων που χρησιμοποιεί το Node project, ο χρήστης θα πρέπει μέσω της γραμμής εντολών αφού εισέλθει στον φάκελο του project να εκτελέσει την εντολή *npm install*. Αφού γίνουν οι απαραίτητες εγκαταστάσεις, ο χρήστης μπορεί να τρέξει την εφαρμογή χρησιμοποιώντας την εντολή *npm start*. Αν δεν υπάρξει κάποιο πρόβλημα, στη γραμμή εντολών θα εμφανιστεί μήνυμα που θα ενημερώνει τον χρήστη ότι το API είναι διαθέσιμο. Για την πραγματοποίηση των http requests στα endpoints του API προτείνεται η χρήση του εργαλείου Postman.

5.2 ΣΣ με φιλτράρισμα βάσει περιεχομένου

Η εκτέλεση της λειτουργίας αυτής γίνεται με ένα http post request στο endpoint http://localhost:4000/api/v1/contentBasedSearch/. Το request body πρέπει να έχει την ακόλουθη δομή:

```
{
    "operation": "weather",
    "numOfRecommendations": 3,
    "throughput": 5.2
}
```

Εικόνα 9: Request body στο ΣΣ με φιλτράρισμα βάσει περιεχομένου

Η λειτουργία αυτή επιστρέφει ένα http response και εκτυπώνει κάποιες πληροφορίες στη γραμμή εντολών. Όσον αφορά τη γραμμή εντολών, εκτυπώνει τους χρόνους σε ms που χρειάστηκαν για τη δημιουργία και την αναζήτηση στο MinHash Forest.

```
Time to build forest: 229.149ms
Time to query forest: 3.003ms
```

Εικόνα 10: Έξοδος γραμμής στο ΣΣ με φιλτράρισμα βάσει περιεχομένου

Όσον αφορά το http response, περιέχει τις συστάσεις που παράγονται με βάση το request body.

```
{
    "recommendations": {
        "basedOnOperation": [
            {
                "Service ID": 642,
                "WSDL Address":
"http://www.webxml.com.cn/WebServices/WeatherWS.asmx?wsdl",
                "Service Provider": "webxml.com.cn",
                "IP Address": "61.147.124.120",
                "Country": "China",
                "IP No.": 1033075832,
                "AS": "AS23650 AS Number for CHINANET jiangsu province backbone",
                "Latitude": 32.0617,
                "Longitude": 118.7778,
                "Throughput": 352.74638348082595,
```

```
                "Response Time": 0.5630884955752211
            },
            {
                "Service ID": 710,
                "WSDL Address": "http://ws365.net/ws/weather.asmx?WSDL",
                "Service Provider": "ws365.net",
                "IP Address": "67.222.147.241",
                "Country": "China",
                "IP No.": 1138660337,
                "AS": "AS30496 Colo4Dallas LP",
                "Latitude": 32.8148,
                "Longitude": -96.8705,
                "Throughput": 9.525592920353978,
                "Response Time": 0.6401769911504429
            },
            {
                "Service ID": 786,
                "WSDL Address":
"http://www.wapit.cn/WebServices/Weather.asmx?WSDL",
                "Service Provider": "wapit.cn",
                "IP Address": "211.137.251.71",
                "Country": "China",
                "IP No.": 3549035335,
                "AS": "AS9808 Guangdong Mobile Communication Co.Ltd.",
                "Latitude": 45.75,
                "Longitude": 126.65,
                "Throughput": 3.287442477876105,
                "Response Time": 2.170005899705015
            }
        ],
        "basedOnThroughput": [
            {
                "Service ID": 786,
                "WSDL Address":
"http://www.wapit.cn/WebServices/Weather.asmx?WSDL",
                "Service Provider": "wapit.cn",
                "IP Address": "211.137.251.71",
                "Country": "China",
                "IP No.": 3549035335,
                "AS": "AS9808 Guangdong Mobile Communication Co.Ltd.",
                "Latitude": 45.75,
                "Longitude": 126.65,
                "Throughput": 3.287442477876105,
                "Response Time": 2.170005899705015
            }
        ]
    }
}
```


Εικόνα 11: Http response στο ΣΣ με φιλτράρισμα βάσει περιεχομένου

5.3 Εγγραφή χρήστη

Η εκτέλεση της λειτουργίας αυτής γίνεται με ένα http post request στο endpoint http://localhost:4000/api/v1/collaborativeFilteringSearch/register. Στο request body πρέπει να καθορίζονται τα στοιχεία σύνδεσης(username, password) του χρήστη στο σύστημα.

```
{
    "username": "u1",
    "password": "j8yKsHrM65tTTFx"
}
```

Εικόνα 12: Request body για εγγραφή χρήστη

Η λειτουργία αυτή επιστρέφει ένα http response με ένα μήνυμα που ενημερώνει τον χρήστη για την έκβαση της λειτουργίας.

```
{
    "message": "Your account has been successfully created."
}
```

Εικόνα 13: Http response για επιτυχημένη εγγραφή χρήστη

5.4 Είσοδος χρήστη

Η εκτέλεση της λειτουργίας αυτής γίνεται με ένα http post request στο endpoint http://localhost:4000/api/v1/collaborativeFilteringSearch/login. Στο request body πρέπει να καθορίζονται τα στοιχεία σύνδεσης(username, password) με τα οποία ο χρήστης εγγράφηκε στο σύστημα. Εφόσον επιβεβαιωθούν τα παρεχόμενα στοιχεία σύνδεσης, επιστρέφεται ένα http response με μήνυμα που ενημερώνει για την επιτυχή έκβαση της λειτουργίας και δημιουργείται ένα HTTP Cookie για τον χρήστη στο πρόγραμμα περιήγησης ιστού του. Διαφορετικά, επιστρέφεται ένα http response με μήνυμα που ενημερώνει για την αποτυχημένη έκβαση της λειτουργίας.

```
{
    "message": "You have successfully logged in."
}
```

Εικόνα 14: Http response για επιτυχημένη είσοδο χρήστη

```
SessionID=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2MjAwY2M0MTQ3MWZi
MDk5M2I2Nzk2MiIsImlhdCI6MTcxMzM3NzI2NiwiZXhwIjoxNzEzMzc4NDY2fQ.PgQ0e37saMTS
oQPI3ajcZuWUr5VmwmnPcOUe6czt-CrtDa8YmwauiIDKMy614FMQyi4AaQik1QgAc_4-
sboNKkkZ3b2pOlPGaVaICMtBlsFGSOAKeiNz7GjbN-NqI-hTBUoiVnjPhthJ3qFjiai-
nVGWp3DTkWekbytv60i5Qmfr202qzTPsnVHz_a0-qtonFMeRUi1vzYj-amud-d_C-
1HLtxJPJ8TXxXIo1otkarXzOFtmhwBa7teUEfjqr-
eOnhoBTEYtqjemhZ42ZPOsfaQcem_1jGLNk5CrJcBwxECMu4W3-
Ko5apPcOgvmyANE4D8gmWk56I2QycvswonswQ; Path=/; Secure; HttpOnly;
Expires=Wed, 17 Apr 2024 18:27:46 GMT;
```

Εικόνα 15: HTTP Cookie χρήστη

```
{
    "message": "No such user."
}
```

Εικόνα 16: Http response για αποτυχημένη είσοδο χρήστη

5.5 Αποσύνδεση χρήστη

Η εκτέλεση της λειτουργίας αυτής γίνεται με ένα http get request στο endpoint http://localhost:4000/api/v1/collaborativeFilteringSearch/logout. Το HTTP Cookie που

δημιουργήθηκε για τον χρήστη κατά την είσοδο του διαγράφεται και επιστρέφεται ένα http response με μήνυμα που ενημερώνει για την έκβαση της λειτουργίας.

```
{
    "message": "You are logged out."
}
```

Εικόνα 17: Http response για αποσύνδεση χρήστη

5.6 ΣΣ με συνεργατικό φιλτράρισμα

Η εκτέλεση της λειτουργίας αυτής γίνεται με ένα http get request στο endpoint http://localhost:4000/api/v1/collaborativeFilteringSearch/{distribution}/{version}. Για να έχει πρόσβαση ο χρήστης στη λειτουργία αυτή θα πρέπει να έχει πραγματοποιήσει είσοδο στο σύστημα. Η λειτουργία αυτή επιστρέφει ένα http response και εκτυπώνει κάποιες πληροφορίες στη γραμμή εντολών. Όσον αφορά την γραμμή εντολών, εκτυπώνει τα αποτελέσματα των διαδοχικών βημάτων του αλγόριθμου που χρησιμοποιείται. Όσον αφορά το http response, περιέχει τις συστάσεις που παράγονται.

```
2a)Pearson Similarities:
[
  [
             1, 0.132894599,
    0.046873153, 0.095483124,
    0.290075847, 0.044676098,
    0.223202401, 0.191765188,
    0.035378824, 0.078181452
  ],
  [
    0.132894599,           1,
    0.190612968, 0.248947565,
    0.189834983, 0.078045803,
    0.208173653,   0.2477088,
    0.141309012, 0.164847083
  ],
```

Εικόνα 18: Τμήμα εξόδου γραμμής εντολών στο ΣΣ με συνεργατικό φιλτράρισμα

```
{
    "recommendationBasedOnPearsonSimilarity": {
        "Service Name": "GlobalWeather",
        "WSDL Address":
"http://www.webservicex.com/globalweather.asmx?WSDL",
        "Rating": 4,
        "Category": "Weather"
    },
    "recommendationBasedOnCosineSimilarity": {
        "Service Name": "GlobalWeather",
        "WSDL Address":
"http://www.webservicex.com/globalweather.asmx?WSDL",
        "Rating": 4,
        "Category": "Weather"
    }
}
```

Εικόνα 19: Http response στο ΣΣ με συνεργατικό φιλτράρισμα

Κεφάλαιο 6 – Συμπεράσματα

6.1 ΣΣ με φιλτράρισμα βάσει περιεχομένου

Όπως αναφέρθηκε προηγουμένως, η αναζήτηση υπηρεσιών ιστού γίνεται σε δύο στάδια. Στο πρώτο στάδιο γίνεται αναζήτηση με βάση μία περιγραφή των λειτουργιών στο σύνολο των υπηρεσιών ιστού του dataset. Στο δεύτερο στάδιο γίνεται αναζήτηση με βάση την απόδοση στις υπηρεσίες ιστού που προέκυψαν από το πρώτο στάδιο. Επομένως, η ποιότητα και η ποικιλία των παραγόμενων συστάσεων καθορίζεται εν πολλοίς από το πρώτο στάδιο αναζήτησης. Επειδή δεν βρέθηκε κάποιο dataset υπηρεσιών ιστού που να περιέχει περιγραφή των λειτουργιών τους, η περιγραφή των λειτουργιών κάθε μίας από τις υπηρεσίες ιστού του dataset γράφτηκε με βάση τη διεύθυνση WSDL. Ως συνέπεια, οι περιγραφές αυτές δεν είναι ιδιαίτερα ακριβείς και κατ' επέκταση οδηγούν σε παραγωγή συστάσεων που είναι αρκετά πιθανό να μην ανταποκρίνονται στις προσδοκίες των χρηστών. Επίσης, οι περιγραφές αυτές είναι μικρές(μερικές λέξεις η κάθε μία), μοιάζουν μεταξύ τους(περιέχουν αρκετά συχνά κοινές λέξεις) και κάποιες υπηρεσίες ιστού έχουν ίδιες περιγραφές(από τις 981 περιγραφές που περιέχονται στο dataset μόνο οι 788 είναι διαφορετικές). Επομένως, η ποικιλία των συστάσεων που παράγεται δεν είναι ιδιαίτερα μεγάλη.

6.2 ΣΣ με συνεργατικό φιλτράρισμα

Όπως αναφέρθηκε προηγουμένως, η σύσταση υπηρεσιών ιστού γίνεται με βάση αξιολογήσεις χρηστών που είναι "όμοιοι" με τον ενεργό χρήστη. Χρησιμοποιούνται δύο διαφορετικές μετρικές ομοιότητας και δύο παραλλαγές χρηστών. Επειδή δεν βρέθηκε dataset με αξιολογήσεις χρηστών για υπηρεσίες ιστού, δημιουργήθηκε ένα ακολουθώντας την εξής διαδικασία:

- Αξιοποιώντας μετρήσεις QWS υπολογίστηκε μία βαθμολογία για κάθε υπηρεσία ιστού.
- Με βάση τη βαθμολογία αυτή, παράχθηκαν αξιολογήσεις χρηστών χρησιμοποιώντας δύο στατιστικές κατανομές(κανονική και x^2).

Από τα παραπάνω προκύπτει ότι η ποιότητα και η ποικιλία των συστάσεων πιθανώς να επηρεάζονται από τις μετρικές ομοιότητας, τις παραλλαγές χρηστών και τις στατιστικές κατανομές που χρησιμοποιήθηκαν. Για να διαπιστωθεί κατά πόσο αυτό ισχύει, πραγματοποιήθηκαν στατιστικές μετρήσεις. Ως δείγμα για τις μετρήσεις αυτές, χρησιμοποιήθηκαν οι συστάσεις του συστήματος για 12 χρήστες, 4 εκ των οποίων ήταν της πρώτης παραλλαγής και οι υπόλοιποι ήταν της δεύτερης παραλλαγής. Για κάθε έναν από τους χρήστες αυτούς καταγράφηκαν οι συστάσεις για όλους τους δυνατούς συνδυασμούς μετρικών ομοιότητας και στατιστικών κατανομών($2^2 = 4$ συνδυασμοί).

Όσον αφορά την ποιότητα συστάσεων, μετρήθηκε ο μέσος όρων των βαθμολογιών των συστάσεων-υπηρεσιών ιστού για όλους τους δυνατούς συνδυασμούς μετρικών ομοιότητας, στατιστικών κατανομών και παραλλαγών χρηστών. Τα αποτελέσματα παρατίθενται στον ακόλουθο πίνακα:

Distribution	Similarity Metric	Version	Average Rating Of Recommendations
normal	pearson	1	4.5
normal	cosine	1	4.25
chi-square	pearson	1	4
chi-square	cosine	1	4
normal	pearson	2	3.875
normal	cosine	2	3.875
chi-square	pearson	2	3.875
chi-square	cosine	2	3.875

Όσον αφορά την ποικιλία συστάσεων, μετρήθηκε κατά πόσο η εναλλαγή μετρικών ομοιότητας ή/και στατιστικών κατανομών για τον ίδιο χρήστη οδηγεί σε διαφοροποίηση της σύστασης. Τα αποτελέσματα παρατίθενται στον ακόλουθο πίνακα:

Distribution	Similarity Metric	Version	Same Recommendation	Different Recommendation	% Of "Same Recommendation"
same	different	1	5	3	62.5
different	same	1	2	6	25
different	different	1	1	7	12.5
same	different	2	16	0	100
different	same	2	14	2	87.5
different	different	2	14	2	87.5

Με βάση τις παραπάνω μετρήσεις, προέκυψαν τα ακόλουθα συμπεράσματα:

- **Παραλλαγές χρηστών.** Οι συστάσεις της πρώτης παραλλαγής είναι πιο ποιοτικές από τις συστάσεις της δεύτερης(έχουν κατά μέσο όρο καλύτερη βαθμολόγηση). Επίσης, η πρώτη παραλλαγή παράγει μεγαλύτερη ποικιλία συστάσεων από τη δεύτερη ανεξαρτήτως κατανομής και μετρικής ομοιότητας. Η υπεροχή της πρώτης παραλλαγής πιθανώς οφείλεται στο ότι ο μέσος χρήστης σε αυτή έχει βαθμολογήσει περισσότερα web services από ότι στην δεύτερη. Ως αποτέλεσμα, το σύστημα συστάσεων στην πρώτη παραλλαγή τροφοδοτείται με περισσότερα δεδομένα.
- **Στατιστικές κατανομές.** Όσον αφορά την ποιότητα των συστάσεων, η χρήση βαθμολογιών που ακολουθούν την κανονική κατανομή οδηγεί σε συστάσεις με λίγο μεγαλύτερη μέση βαθμολογία από ότι η χρήση βαθμολογιών που ακολουθούν κατανομή x^2. Επίσης, οι κατανομές επηρεάζουν σε σημαντικό βαθμό την ποικιλία συστάσεων που παράγεται. Πιο συγκεκριμένα, με σταθερή κατανομή παράγεται μικρή ποικιλία συστάσεων ενώ με μεταβαλλόμενη κατανομή παράγεται μεγαλύτερη ποικιλία συστάσεων.
- **Μετρικές ομοιότητας.** Οι μετρικές ομοιότητας δεν έχουν σημαντική επίδραση στην ποιότητα και την ποικιλία των συστάσεων.

Βιβλιογραφία

[1] [Security] Prototype Pollution in sheetJS. (2023, Δεκέμβριος 2). Ανάκτηση από Github: https://github.com/SheetJS/sheetjs/issues/2822

[2] Achjanis, K. (2023, Δεκέμβριος 8). *The Right Way to Split String Into Words in JavaScript*. Ανάκτηση από DEV Community: https://dev.to/kamonwan/the-right-way-to-break-string-into-words-in-javascript-3jp6

[3] Al-Masri, E. (2023, Νοέμβριος 1). *QWS Dataset (The Quality of Service for Web Services Dataset)*. Ανάκτηση από https://qwsdata.github.io/

[4] Al-Masri, E., & Mahmoud, Q. H. (2007). Discovering the best web service. *ACM 16th International Conference on World Wide Web (WWW)*, (σσ. 1257-1258).

[5] Al-Masri, E., & Mahmoud, Q. H. (2007). QoS-based Discovery and Ranking of Web Services. *IEEE 16th International Conference on Computer Communications and Networks (ICCCN)*, (σσ. 529-534).

[6] Al-Masri, E., & Mahmoud, Q. H. (2008). Investigating web services on the world wide web. *ACM 17th international conference on World Wide Web(WWW '08)*, (σσ. 795-804).

[7] Asher, M. (2023, Νοέμβριος 30). *Probability distribution demo and tests*. Ανάκτηση από statisticsblog: https://statisticsblog.com/probability-distributions/

[8] Awati, R. (2024, Απρίλιος 19). *What is UDDI and how does it work? – TechTarget Definition*. Ανάκτηση από TechTarget: https://www.techtarget.com/whatis/definition/UDDI-Universal-Description-Discovery-and-Integration

[9] Bigelow, S. J. (2024, Μάρτιος 30). *What are Web Services? | Definition from TechTarget*. Ανάκτηση από TechTarget: https://www.techtarget.com/searchapparchitecture/definition/Web-services

[10] Burckhardt, P. (2024, Φεβρουάριος 5). *compute-cosine-similarity - npm*. Ανάκτηση από npmjs: https://www.npmjs.com/package/compute-cosine-similarity

[11] How can a comma separated list be converted into cells in a column for Lt? (2023, Νοέμβριος 5). Ανάκτηση από ADInstruments: https://www.adinstruments.com/support/knowledge-base/how-can-comma-separated-list-be-converted-cells-column-lt

[12] How to Validate JSON Input with NodeJS (Schema). (2023, Δεκέμβριος 20). Ανάκτηση από Scriptable: https://scriptable.com/javascript/how-to-validate-json-input-with-nodejs-schema/

[13] HTTP - Wikipedia. (2024, Μάρτιος 30). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/HTTP

[14] Internet Protocol - Wikipedia. (2024, Απρίλιος 19). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Internet_Protocol

[15] JSON - Wikipedia. (2024, Μάρτιος 30). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/JSON

[16] JSON Web Token - Wikipedia. (2024, Απρίλιος 19). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/JSON_Web_Token(19/4/2024)

[17] Kandel, K. (2024, Ιανουάριος 5). *Measuring Execution Time in NodeJS*. Ανάκτηση από Medium: https://medium.com/@kirankandel007/measuring-execution-time-in-nodejs-1d4179eeb860

[18] Locality-sensitive hashing - Wikipedia. (2024, Απρίλιος 19). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Locality-sensitive_hashing

[19] McLendon, R. (2023, Δεκέμβριος 5). *Building a Recommendation Engine with Locality-Sensitive Hashing (LSH) in Python*. Ανάκτηση από Learn Data Science: https://www.learndatasci.com/tutorials/building-recommendation-engine-locality-sensitive-hashing-lsh-python/

[20] minhashjs - npm. (2023, Δεκέμβριος 10). Ανάκτηση από npmjs: https://www.npmjs.com/package/minhashjs

[21] Nandan, A. (2023, Δεκέμβριος 3). *How to Read Excel Files in Node.js*. Ανάκτηση από Javascripti in Plain English: https://javascript.plainenglish.io/how-to-read-excel-file-in-node-js-db3dbd39580b

[22] Node.js - Run Javascript Everywhere. (2024, Μάρτιος 30). Ανάκτηση από nodejs: https://nodejs.org/en

[23] npm - Wikipedia. (2024, Απρίλιος 19). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Npm

[24] Recommender system - Wikipedia. (2024, Μάρτιος 30). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Recommender_system

[25] REST - Wikipedia. (2024, Απρίλιος 19). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/REST

[26] Revived PlanetB | Syntax Highlight Code in Word Documents. (2024, Απρίλιος 10). Ανάκτηση από planetb: https://planetb.troye.io/

[27] RSA Algorithm in Cryptography - GeeksforGeeks. (2024, Απρίλιος 19). Ανάκτηση από GeeksforGeeks: https://www.geeksforgeeks.org/rsa-algorithm-cryptography/

[28] Sahi, S. (2024, Μάρτιος 30). *Understanding QoS Metrics of Web Services: Ensuring Exceptional User Experiences*. Ανάκτηση από Medium: https://medium.com/@sonalisahi/understanding-qos-metrics-of-web-services-ensuring-exceptional-user-experiences-433c119f60db

[29] Sheets, S. D. (2023, Δεκέμβριος 1). *xlsx - npm*. Ανάκτηση από npmjs: https://www.npmjs.com/package/xlsx

[30] SOAP - Wikipedia. (2024, Απρίλιος 19). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/SOAP

[31] Staff, w. (2023, Νοέμβριος 5). *Easy Ways to Change a Comma to Dot in Excel: 11 Steps*. Ανάκτηση από wikiHow: https://www.wikihow.com/Change-a-Comma-to-Dot-in-Excel

[32] Syntax Highlighter for Word. (2024, Απρίλιος 17). Ανάκτηση από syntax-highlighter: https://www.syntax-highlighter.dev

[33] Vara, R. (2024, Ιανουάριος 30). *calculate-correlation - npm*. Ανάκτηση από npmjs: https://www.npmjs.com/package/calculate-correlation

[34] Web service - Wikipedia. (2024, Μάρτιος 30). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Web_service

[35] Web Services Description Language - Wikipedia. (2024, Απρίλιος 19). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Web_Services_Description_Language

[36] What is an API? - Application Programming Interface Explained - AWS. (2024, Μάρτιος 30). Ανάκτηση από Amazon Web Services: https://aws.amazon.com/what-is/api/

[37] What is Express.js? | *Codeacademy*. (2024, Απρίλιος 19). Ανάκτηση από Codeacademy: https://www.codecademy.com/article/what-is-express-js

[38] What Is Quality of Service (QoS) in Networking? | *Fortinet*. (2024, Μάρτιος 30). Ανάκτηση από Fortinet: https://www.fortinet.com/resources/cyberglossary/qos-quality-of-service

[39] WS-DREAM: Towards Open Datasets and Source Code for Web Service Research. (2023, Νοέμβριος 15). Ανάκτηση από http://wsdream.github.io/dataset/wsdream_dataset1.html

[40] XML - Wikipedia. (2024, Απρίλιος 19). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/XML

[41] Zheng, Z., Zhang, Y., & Lyu, M. R. (2014). Investigating QoS of Real-World Web Services. *IEEE Transactions on Services Computing, 7*(1), σσ. 32-39.

[42] M, J. (2024, Ιανουάριος 15). *Build a Login and Logout API using Express.js (Node.js)*. Ανάκτηση από Dev Community: https://dev.to/m_josh/build-a-jwt-login-and-logout-system-using-expressjs-nodejs-hd2

[43] Do, Minh-Phung & Nguyen, Dung & Nguyen, Loc. (2010). Model-based approach for Collaborative Filtering

[44] Isinkaye, Folasade & Folajimi, Yetunde & Ojokoh, Bolanle. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal. 16. 10.1016/j.eij.2015.06.005

Γλωσσάρι

Συντομογραφία	Όρος	Σημασία
WS	Web Service	• Υπηρεσία που προσφέρεται από μια ηλεκτρονική συσκευή σε μια άλλη ηλεκτρονική συσκευή, επικοινωνώντας μεταξύ τους μέσω του Διαδικτύου. • Διακομιστής που εκτελείται σε μια συσκευή υπολογιστή, ο οποίος ακούει αιτήσεις σε μια συγκεκριμένη θύρα μέσω ενός δικτύου και παρέχει έγγραφα ιστού.
QoS	Quality of Service	Η χρήση μηχανισμών ή τεχνολογιών που λειτουργούν σε ένα δίκτυο για τον έλεγχο της κυκλοφορίας και τη διασφάλιση της απόδοσης κρίσιμων εφαρμογών με περιορισμένη χωρητικότητα δικτύου.
API	Application Programming Interface	Μηχανισμός που επιτρέπει σε δύο τμήματα λογισμικού να επικοινωνούν μεταξύ τους χρησιμοποιώντας ένα σύνολο ορισμών και πρωτοκόλλων.
ΣΣ	Σύστημα Συστάσεων	Υποκατηγορία συστήματος φιλτραρίσματος πληροφοριών που παρέχει προτάσεις για στοιχεία που αφορούν περισσότερο έναν συγκεκριμένο χρήστη.
Node	Node.js	Ένα δωρεάν, ανοιχτού κώδικα, cross-platform περιβάλλον εκτέλεσης Javascript που επιτρέπει στους προγραμματιστές να δημιουργούν διακομιστές, διαδικτυακές εφαρμογές, εργαλεία γραμμής εντολών και scripts.
JSON	Javascript Object Notation	Μία ανοικτή τυποποιημένη μορφή αρχείου και μορφή ανταλλαγής δεδομένων που χρησιμοποιεί κείμενο αναγνώσιμο από τον άνθρωπο για την αποθήκευση και τη μετάδοση

		αντικειμένων δεδομένων που αποτελούνται από ζεύγη χαρακτηριστικών-τιμών και πίνακες (ή άλλες σειριοποιήσιμες τιμές).
HTTP	HyperText Transfer Protocol	Πρωτόκολλο επιπέδου εφαρμογής στο μοντέλο της σουίτας πρωτοκόλλων του Διαδικτύου για κατανεμημένα, συνεργατικά, υπερμεσικά συστήματα πληροφοριών. Αποτελεί το θεμέλιο της επικοινωνίας δεδομένων για τον Παγκόσμιο Ιστό, όπου τα έγγραφα υπερκειμένου περιλαμβάνουν υπερσυνδέσμους προς άλλους πόρους στους οποίους ο χρήστης μπορεί εύκολα να έχει πρόσβαση, για παράδειγμα με ένα κλικ του ποντικιού ή με το πάτημα της οθόνης σε ένα πρόγραμμα περιήγησης ιστού.
XML	Extensible Markup Language	Μια γλώσσα σήμανσης και μια μορφή αρχείου για την αποθήκευση, τη μετάδοση και την ανακατασκευή αυθαίρετων δεδομένων. Ορίζει ένα σύνολο κανόνων για την κωδικοποίηση εγγράφων σε μορφή που είναι αναγνώσιμη τόσο από τον άνθρωπο όσο και από υπολογιστές.
SOAP	Simple Object Access Protocol	Είναι μια προδιαγραφή πρωτοκόλλου ανταλλαγής μηνυμάτων για την ανταλλαγή δομημένων πληροφοριών κατά την υλοποίηση υπηρεσιών ιστού σε δίκτυα υπολογιστών. Χρησιμοποιεί το XML Information Set για τη μορφή του μηνύματος και βασίζεται σε πρωτόκολλα επιπέδου εφαρμογής, συχνότερα το Hypertext Transfer Protocol (HTTP), αν και ορισμένα παλαιά συστήματα επικοινωνούν μέσω του Simple Mail Transfer Protocol (SMTP), για τη διαπραγμάτευση και τη μετάδοση μηνυμάτων.

WSDL	Web Services Description Language	Είναι μια γλώσσα περιγραφής διεπαφών βασισμένη σε XML, η οποία χρησιμοποιείται για την περιγραφή της λειτουργικότητας που προσφέρεται από μια υπηρεσία ιστού.
REST	Representational State Transfer	Είναι ένα αρχιτεκτονικό στυλ λογισμικού που δημιουργήθηκε για να καθοδηγήσει το σχεδιασμό και την ανάπτυξη της αρχιτεκτονικής του Παγκόσμιου Ιστού. Το REST ορίζει ένα σύνολο περιορισμών για το πώς πρέπει να συμπεριφέρεται η αρχιτεκτονική ενός κατανεμημένου συστήματος υπερμέσων σε κλίμακα Διαδικτύου, όπως ο Παγκόσμιος Ιστός.
UDDI	Universal Description, Discovery and Integration	Είναι ένα πρότυπο βασισμένο στην XML για την περιγραφή, δημοσίευση και εύρεση πληροφοριών σχετικά με υπηρεσίες ιστού. Είναι επίσης ένα μητρώο για τις επιχειρήσεις που παρέχουν διαδικτυακές υπηρεσίες για να καταχωρούν τον εαυτό τους και να βρίσκουν συνεργάτες στο Διαδίκτυο.
IP	Internet Protocol	Είναι το πρωτόκολλο επικοινωνίας επιπέδου δικτύου της σουίτας πρωτοκόλλων Διαδικτύου για την αναμετάδοση δεδομένων μέσω των ορίων του δικτύου. Η λειτουργία δρομολόγησής του επιτρέπει τη δικτύωση μέσω του Διαδικτύου και ουσιαστικά δημιουργεί το Διαδίκτυο.
npm	Node Package Manager	Είναι ένας διαχειριστής πακέτων για τη γλώσσα προγραμματισμού JavaScript που συντηρείται από τη Microsoft's npm, Inc. Αποτελεί τον προεπιλεγμένο διαχειριστή πακέτων για το περιβάλλον εκτέλεσης JavaScript Node.js και περιλαμβάνεται ως συνιστώμενο χαρακτηριστικό

		στο πρόγραμμα εγκατάστασης του Node.js.
JWT	Json Web Token	Είναι ένα προτεινόμενο πρότυπο του Διαδικτύου για τη δημιουργία δεδομένων με προαιρετική υπογραφή ή/και προαιρετική κρυπτογράφηση, το ωφέλιμο φορτίο των οποίων περιέχει JSON που βεβαιώνει κάποιον αριθμό ισχυρισμών. Τα tokens υπογράφονται είτε χρησιμοποιώντας ένα ιδιωτικό μυστικό είτε ένα δημόσιο/ιδιωτικό κλειδί.
RSA	Rivest–Shamir–Adleman	Είναι ένας αλγόριθμος ασύμμετρης κρυπτογραφίας. Ο ασύμμετρος σημαίνει στην πραγματικότητα ότι λειτουργεί με δύο διαφορετικά κλειδιά, δηλ. δημόσιο κλειδί και ιδιωτικό κλειδί. Όπως περιγράφει το όνομα, το δημόσιο κλειδί δίνεται σε όλους και το ιδιωτικό κλειδί παραμένει ιδιωτικό.
Express	Express.js	Είναι το πιο δημοφιλές πλαίσιο ιστού για το Node.js. Έχει σχεδιαστεί για την κατασκευή διαδικτυακών εφαρμογών και API και έχει χαρακτηριστεί ως το de facto πρότυπο πλαίσιο διακομιστή για το Node.
LSH	Locality-Sensitive Hashing	Είναι μια ασαφής τεχνική κατακερματισμού που κατακερματίζει παρόμοια στοιχεία εισόδου στους ίδιους "κάδους" με υψηλή πιθανότητα. Ο αριθμός των κάδων είναι πολύ μικρότερος από το σύμπαν των πιθανών στοιχείων εισόδου. Δεδομένου ότι παρόμοια στοιχεία καταλήγουν στους ίδιους κάδους, αυτή η τεχνική μπορεί να χρησιμοποιηθεί για την ομαδοποίηση δεδομένων και την αναζήτηση του πλησιέστερου γείτονα.