ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

“ ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΜΕΤΡΙΚΩΝ ΑΠΟΔΟΣΗΣ ΕΡΕΥΝΗΤΩΝ ΚΑΙ ΕΡΕΥΝΗΤΙΚΩΝ ΟΜΑΔΩΝ SCHOLARBOOK”

Του φοιτητή
Παπακωνσταντίνου Γεώργιου
Αρ. Μητρώου: 174929

Επιβλέπων
Ουγιάρογλου Στέφανος, Επ. Καθηγητής

7 Ιουνίου 2023

Τίτλος Π.Ε.:Διαδικτυακή εφαρμογή για την παρακολούθηση μετρήσεων απόδοσης ερευνητών και ερευνητικών ομάδων.
Κωδικός Δ.Ε. 22265
Ονοματεπώνυμο φοιτητή/ών: Παπακωνσταντίνου Γεώργιου
Ονοματεπώνυμο εισηγητή: Ουγιάρογλου Στέφανος
Ημερομηνία ανάληψης Δ.Ε.: 17-10-2022
Ημερομηνία περάτωσης Δ.Ε.: 24-05-2023

Περίληψη

Η πληροφορική και η οργάνωση των δεδομένων είναι δύο απαραίτητοι παράγοντες για την ανάπτυξη και την πρόοδο σε πολλούς τομείς της ζωής, και ιδίως στον τομέα της επιστήμης και της έρευνας. Η πληθώρα των δεδομένων που δημιουργούνται σε αυτό τον τομέα κάθε μέρα καθιστά απαραίτητη τη χρήση προηγμένων τεχνολογιών για τη συλλογή, την ανάλυση και την οργάνωσή τους.
Η επιστημομετρία είναι ένας κλάδος της επιστήμης που ασχολείται με τη μέτρηση και την αξιολόγηση της επιστημονικής έρευνας. Ερευνά τις διάφορες μετρικές που μπορούν να χρησιμοποιηθούν για την αξιολόγηση της ποιότητας της έρευνας, καθώς και του αντίκτυπου που έχει στην κοινότητα των ερευνητών και στην κοινωνία γενικότερα. Στόχος της επιστημομετρίας είναι η βελτίωση της ποιότητας της έρευνας, καθώς και η προώθηση της ανάπτυξης της επιστημονικής γνώσης. Η επιστημομετρία χρησιμοποιεί συχνά τεχνολογίες της πληροφορικής για τη συλλογή, ανάλυση και παρουσίαση των δεδομένων.
Το Google Scholar αποτελεί μια από της πιο γνωστές πλατφόρμες όπου υπάρχουν δεδομένα επιστημομετρίας. Ο λόγος που ξεχωρίζει ο συγκεκριμένος ιστότοπος είναι επειδή παρέχει δεδομένα ελεύθερα χωρίς κάποια συνδρομή καθώς και συλλέγει ερευνητικά έργο από ποικίλες πηγές του διαδικτύου. Παρά την πληθώρα των δεδομένων η εργαλειοθήκη του ιστότοπου δεν παρέχει λειτουργίες για την παρακολούθηση συγκεκριμένων ερευνητών όπου μπορούν να ενδιαφέρουν έναν χρήστη. Έτσι η παρακολούθηση μεμονωμένων ερευνητών απαιτεί την επαναλαμβανόμενη αναζήτηση τους. Ακόμα, δεν παρέχει δημιουργία ομάδων από ερευνητές όπου μπορεί για παράδειγμα να απαρτίζουν μια ερευνητική ομάδα ή ένα τμήμα πανεπιστημιούπολης. Αυτό έχει ως αποτέλεσμα να γίνεται δύσκολη η σύγκριση μεμονωμένων ερευνητών ενώ η σύγκριση μεταξύ δύο τμημάτων να καθιστάτε αδύνατη.
Η παρούσα εφαρμογή που αναπτύχθηκε στα πλαίσια της διατριβής έχει ως σκοπό να επεκτείνει την εργαλειοθήκη του Google Scholar και να δώσει λύση στα προβλήματα οργάνωσης και σύγκρισης. Οι λειτουργίες της εφαρμογής περιλαμβάνουν τη συλλογή δεδομένων από την πλατφόρμα Google Scholar, τη δημιουργία προφίλ ερευνητών και ομάδων ερευνητών, την παρακολούθηση των δημοσιεύσεων, των αναφορών, του δείκτη h, καθώς και τη σύγκριση των δεδομένων μεταξύ διαφορετικών ομάδων ερευνητών. Με τη χρήση αυτής της εφαρμογής, οι ενδιαφερόμενοι χρήστες μπορούν να παρακολουθούν τις επιστημονικές επαφές τους και να δημιουργούν ομάδες με άλλους ερευνητές. Με στόχο τη σύγκριση ερευνητών εντός της ομάδας αλλά και τη σύγκριση μεταξύ ομάδων. Η εφαρμογή προσφέρει ένα ολοκληρωμένο οικοσύστημα όπου η διαδικασία συλλογής, οργάνωσης, σύγκρισης δεδομένων με σκοπό την εξαγωγή πορίσματός γίνεται με μεγάλη ευκολία. Αυτό μπορεί να οδηγήσει σε μια βαθύτερη κατανόηση των τελευταίων εξελίξεων στον τομέα της έρευνας και στην ανάπτυξη νέων ιδεών και προτάσεων έρευνας.

« Web-bases application for monitoring performance metrics of researches and research teams »

« George Papakonstantinou »

Abstract

Information technology and data organization are two essential factors for development and progress in many areas of life, especially in science and research. The abundance of data generated in this field every day makes it necessary to use advanced technologies to collect, analyze and organize them.
Scientometrics is a branch of science that deals with the measurement and evaluation of scientific research. It explores the various metrics that can be used to assess the quality of research and its impact on the research community and society at large. The aim of scientometrics is to improve the quality of research and to promote the development of scientific knowledge. Scientometrics often uses information technology to collect, analyze and present data.
Google Scholar is one of the most well-known platforms where epistemometric data exists. The reason this site stands out is because it provides data freely without any subscription and collects research work from various sources on the web. Despite the abundance of data the site's toolbox does not provide functions to track specific researchers where they may be of interest to a user. Thus tracking individual researchers requires their repeated search. It also does not provide for grouping of researchers where they may for example form a research group or a campus department. This has the effect of making it difficult to compare individual researchers while making comparison between two departments impossible.
The present application developed in the context of the thesis aims to expand the Google Scholar toolbox and provide a solution to the problems of organization and comparison. The functions of the application include collecting data from the Google Scholar platform, creating profiles of researchers and research groups, tracking publications, citations, h-index, as well as comparing data between different groups of researchers. By using this app, interested users can keep track of their scientific contacts and create groups with other researchers. Aiming to compare researchers within the group but also to compare between groups. This can lead to a deeper understanding of the latest developments in the research field and the development of new research ideas and proposals.

Ευχαριστίες

Ξεκινώντας, θα ήθελα να ευχαριστήσω τον κ. Στέφανο Ουγιάρογλου για την αφοσίωση του και τη βοήθεια του καθ όλη την ανάπτυξη της πτυχιακής εργασίας. Έκανε τον στόχο της εργασίας ξεκάθαρο και προσδιορισμένο από την πρώτη στιγμή. Η παρουσία του στην εργασία έκανε ομαλή την ανάπτυξη από την αρχή ως το τέλος.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Επιστημομερτία

Επιστημομετρία [1] είναι ο κλάδος μελέτης όπου με τη χρήση ποσοτικών μεθόδων και μετρικών, αναλύει και μετράει το αντίκτυπο της επιστημονικής και ακαδημαϊκής βιβλιογραφίας. Αναφέρεται ατομικά σε κάθε ερευνητή και περιλαμβάνει τον αριθμό των δημοσιεύσεων, τις συνολικές αναφορές μια δημοσίευσης καθώς και το αντίκτυπο της έρευνας πάνω στον επιστημονικό κλάδο. Η επιστημομετρία μπορεί να εφαρμοστεί τόσο ατομικά όσο και συλλογικά όπως για παράδειγμα σε ιδρύματα για την αξιολόγηση της παραγωγικότητας και του αντίκτυπου μιας έρευνας. Μπορεί επίσης να χρησιμοποιηθεί για τον εντοπισμό τάσεων και προτύπων στην επιστημονική έρευνα και για τον εντοπισμό τομέων στους οποίους απαιτείται περαιτέρω έρευνα.

Μερικά παραδείγματα των τύπων ερωτημάτων που η επιστήμη της επιστήμης μπορεί να βοηθήσει να απαντηθούν περιλαμβάνουν:

- Πόσες ερευνητικές εργασίες έχουν δημοσιευθεί για ένα συγκεκριμένο θέμα;
- Ποιες χώρες και ποια ιδρύματα είναι τα πιο ενεργά στη δημοσίευση έρευνας για ένα συγκεκριμένο θέμα;
- Ποιες ερευνητικές εργασίες για ένα συγκεκριμένο θέμα έχουν αναφερθεί περισσότερο;
- Τι αντίκτυπο μπορεί να έχει η χρηματοδότηση μιας έρευνας σχετικά με την ποιότητα του αποτελέσματος;
- Ποιο είναι το αντίκτυπος της έρευνας στην κοινωνία και την οικονομία;

Στην επιστημομετρία ενώ υπάρχει η ικανότητα να ερευνηθούν καθολικά οι πτυχές μιας τεχνολογίας και της δυναμικής της πάνω στην επιστήμη, η εξέλιξη της απαρτίζετε κυρίως στις μετρικές [2] των δημοσιεύσεων (publications), των αναφορών (citations) καθώς και του δείκτη h (h-index). Με τον όρο δημοσιεύσεων ορίζεται το σύνολο των άρθρων, επιστημονικών κειμένων, περιοδικών, δοκιμίων και ότι άλλο μπορεί να θεωρηθεί ως συνεισφορά πάνω σε έναν επιστημονικό κλάδο όπου έχει πραγματοποιήσει ένας ερευνητής. Με τον όρο αναφορά ορίζουμε τον αριθμό των φορών που έχει αναφερθεί μια δημοσίευση από άλλους ερευνητές. Ο δείκτης h πρόκειται για μια μέτρηση που συνδυάζει δεδομένα αναφορών και δημοσιεύσεων για τη μέτρηση της παραγωγικότητας και του αντίκτυπου ενός ερευνητή πάνω στην επιστήμη. Ο δείκτης h υπολογίζεται ταξινομώντας τις δημοσιεύσεις ενός ερευνητή με τη σειρά του αριθμού των αναφορών που έχουν λάβει και στη συνέχεια λαμβάνοντας τον υψηλότερο αριθμό όπου ο ερευνητής έχει τουλάχιστον h δημοσιεύσεις.

1.2 Διαδικτυακοί ιστότοποι βιβλιογραφίας

Ισότοποι όπου παρέχουν μια συλλογή από επιστημονικά βιβλία, άρθρα, περιοδικά και οποιοδήποτε ερευνητικό έργο πάνω στην επιστήμη ονομάζονται Διαδικτυακοί ιστότοποι βιβλιογραφίας. Η ραγδαία ανάπτυξη των παραπάνω ιστοτόπων ξεκίνησε στα τέλη της δεκαετίας του 1960 ώστε να καλυφθεί η ανάγκη

οργανωμένης συλλογής και καταγραφή βιβλιογραφικών δεδομένων. Αυτές οι διαδικτυακές πλατφόρμες περιλαμβάνουν χρήσιμες πληροφορίες για τα βιβλία, τα άρθρα και τα επιστημονικά περιοδικά, όπως για παράδειγμα τους συγγραφείς όπου εργάστηκαν, τη θεματολογία τους, την ημερομηνία δημοσίευσης και τη διαθεσιμότητα. Πέρα από τις βασικές πληροφορίες μερικές πλατφόρμες παρέχουν ακόμα και πρόσβαση σε αυτά τα ερευνητικά άλλοτε με τη χρήση συνδρομής και άλλοτε με προσωρινό δανεισμό ή αγορά. Μερικοί από τους διαδικτυακούς ιστοτόπους βιβλιογραφίας είναι το Google Scholar,το Scopus, το Web of Science, το JSTOR και το Project MUSE.

Πιο συγκεκριμένα το Scopus [3] και το Web of Science [4] απαρτίζουν τις μεγαλύτερες πηγές δεδομένων για την αναζήτηση ερευνητικών άρθρων, κυρίως στον τομέα των φυσικών επιστημών, της βιοϊατρικής, της μηχανικής. Παρέχοντας μεγάλο όγκο δεδομένων ποσοτικά αλλά και χρονολογικά, καλύπτοντας χρονολογίες έως και 100 χρόνια πριν, έχοντας καταχωρημένα ερευνητικά άρθρα, περιοδικά (journals), καθώς και από άλλες πηγές, όπως παρουσιάσεις σε επιστημονικά συνέδρια και τεχνικά έγγραφα. Το Scopus περιλαμβάνει περισσότερα από 70.000 επιστημονικά περιοδικά, συνέδρια και βιβλία από όλο τον κόσμο. Το Web of Science είναι ένα ιδιαίτερα γνωστό εργαλείο για τη μέτρηση της επιστημονικής δραστηριότητας των ανθρώπων, ομάδων και ιδρυμάτων, περιλαμβάνοντας περισσότερα από 12.000 επιστημονικά περιοδικά, συνέδρια και βιβλία από όλο τον κόσμο. Τέλος, οι παραπάνω υπηρεσίες απευθύνονται κυρίως σε επαγγελματίες, καθώς η πρόσβαση σε αυτές προαπαιτεί συνδρομή.

Αντίστοιχη πλατφόρμα με διαφορετική προσέγγιση καθώς και παροχή ελεύθερης πρόσβασης είναι η διαδικτυακή μηχανή αναζήτησης Google scholar [5]. Το Google scholar είναι μία υπηρεσία της Google, που ξεκίνησε πειραματικά τον Νοέμβριο του 2004, παρέχει πρόσβαση σε διαδικτυακά ακαδημαϊκά περιοδικά και βιβλία με κριτές, έγγραφα συνεδρίων, διατριβές και διπλωματικές εργασίες, προδημοσιεύσεις, περιλήψεις, τεχνικές εκθέσεις και οποιαδήποτε άλλη επιστημονική βιβλιογραφία, συμπεριλαμβανομένων δικαστικών αποφάσεων και διπλωμάτων ευρεσιτεχνίας. Η υπηρεσία κάνει χρήση web crawler, ή καλύτερα ρομπότ ιστού, ώστε να εντοπιστούν και να καταχωρηθούν τα δεδομένα της. Τέλος, αξίζει να σημειωθεί ότι η βιβλιοθήκη του πανεπιστημίου του Μίσιγκαν και άλλες βιβλιοθήκες των οποίων οι συλλογές σαρώθηκαν από την Google για τα Google Books και το Google Scholar διατήρησαν αντίγραφα των σαρώσεων και τα χρησιμοποίησαν για τη δημιουργία της ψηφιακής βιβλιοθήκης HathiTrust .

1.3 Κίνητρο

Τα τελευταία χρόνια ο κλάδος της πληροφορικής έχει σημειώσει ραγδαία πρόοδο και έχει προσφέρει σημαντικές βελτιώσεις τόσο ατομικά όσο και συλλογικά. Από μικρές βελτιώσεις στον τρόπο ζωής έως αυτοματοποιήσεις σε μαζικές γραμμές παραγωγής. Πλέον, με ευκολία θα μπορούσαμε να πούμε ότι η πληροφορία αποτελεί ένα είδος νομίσματος και μάλιστα με πολύ ισχυρή θέση. Ο κάτοχος της μπορεί να διεξάγει διαφημιστικές καμπάνιες κάνοντας στοχευμένο μάρκετινγκ, να βελτιστοποιήσει την εμπειρία των χρηστών είτε να προβλέψει αύξηση στη ζήτηση ενός αγαθού. Κύριος παράγοντάς για να επιτευχθεί κάτι τέτοιο πέρα από την κατοχή των δεδομένων είναι και η οργάνωση τους. Σε περίπτωση όπου δεν καλύπτεται αυτή η προϋπόθεση η οπτικοποίηση των δεδομένων και η εξαγωγή πορίσματος γίνεται σχεδόν ακατόρθωτη. Πέρα από τον εμπορικό τομέα, ο παραπάνω κανόνας ισχύει και στον ερευνητικό.

Η παρούσα πτυχιακή εργασία στοχεύει στην οργάνωση των δεδομένων πάνω στον κλάδο της επιστημομετρίας, δηλαδή ως προς τις μεταβλητές αναφορά, δημοσίευση και h-index. Έχει ως σκοπό να διευκολύνει την παρακολούθηση ερευνητών συμπεριλαμβανομένου του έργου τους, όπου τα δεδομένα τους υπάρχουν στην υπηρεσία Google Scholar. Όπως είναι ευρέως γνωστό από την επιστημονική κοινότητα τα δεδομένα τους είναι ελεύθερα και προσβάσιμα προς όλους. Πέρα από τον μεγάλο όγκο των δεδομένων, ο τρόπος όπου η Google Scholar προβάλει τα δεδομένα καθιστά οποιαδήποτε σύγκριση για τη δημιουργία πορίσματος χρονοβόρα και πολύπλοκη. Πιο συγκεκριμένα η υπηρεσία σχεδιάστηκε ώστε να προβάλει τα δεδομένα σε φυσικούς χρήστες. Επιπρόσθετα, δεν παρέχει τα δεδομένα της σε κάποια μορφή όπως json ή csv καθιστώντας απαραίτητη την εξαγωγή δεδομένον (scraping) σε περίπτωση όπου υπάρχει ανάγκη εφαρμογής αλγορίθμου. Επίσης, κάθε ερευνητής ενός ακαδημαϊκού τμήματος μπορεί να δημιουργήσει προφίλ και να φαίνεται το έργο του. Όμως δεν υπάρχει κάτι αντίστοιχο για το συνολικό έργο ενός τμήματος, δηλαδή δεν δίνεται η δυνατότητα παρουσίασης δεδομένων μια επιστημονικής ομάδας ή ενός ερευνητικού εργαστηρίου ή ακαδημαικού τμήματος. Με άλλα λόγια, ενώ το Google Scholar περιλαμβάνει τα δεδομένα δεν υπάρχει δυνατότητα γα αξιοποίηση αυτών μέσα από την πλατφόρμα όπως και δεν υπάρχει η προβολή αυτών σε μεγαλύτερη κλίμακα πέρα από την ατομική.

1.4 Συνεισφορά

Με κίνητρο τις αδυναμίες του Google Scholar σε συνδυασμό τη βελτίωση παρακολούθησης και οργάνωσης δεδομένων επιστημομετρίας η παρούσα συνεισφορά της πτυχιακής εργασίας είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής, με όνομα scholarbook. Με κύριο σκοπό την οργάνωση των δεδομένων που σχετίζονται με την επιστημονική έρευνα και την επιστημομετρία. Αρχικά, το πρόβλημα αποθήκευσης δεδομένων ανά χρήστη και προσωπικών δεδομένων. Επίσης, κύριο πρόβλημα της πλατφόρμας Google Scholar είναι η αδυναμία παρακολούθησης συγκεκριμένων ερευνητών και η εύκολη πρόσβαση σε αυτούς. Εξίσου σημαντικό πρόβλημα είναι η ομαδοποίηση ερευνητών σε ομάδες με σκοπό τόσο την οργάνωση όσο και τη σύγκριση του έργου τους και το αντίκτυπο τους στην επιστήμη.

Με βάση τα παραπάνω προβλήματα προκύπτουν τα ζητούμενα της εργασίας ώστε να επιλυθούν. Αρχικά, η αποθήκευση προσωπικών δεδομένων και προτιμήσεων επιτυγχάνεται με τη δημιουργία λογαριασμού για κάθε χρήστη. Επιπλέων, σε μορφή επαφής προσφέρεται η δυνατότητα παρακολούθησης ενός μεμονωμένου ερευνητή με βάση τις μετρικές της επιστημομετρίας, καθώς και η προβολή των δημοσιεύσεών του. Επιπλέον, για την καλύτερη οργάνωση και σύγκριση δίνεται η δυνατότητα στους χρήστες

να οργανώνουν και να ομαδοποιούν τις επαφές σε ομάδες. Σε κάθε ομάδα, οι μετρικές των ερευνητών προβάλλονται σε γράφημα, ώστε να μπορεί να γίνει σύγκριση των μελών της ίδιας ομάδας ανά χρονιά.

1.5 Οργάνωση εργασίας

Στο κεφάλαιο 2 παρουσιάζεται η πλατφόρμα του Google Scholar, μαζί με τις μετρικές όπου χρησιμοποιεί για την παρουσία του κάθε ερευνητή. Στη συνέχεια, γίνεται μια επεξήγηση για τη σημασία της κάθε μετρικής της επιστημομετρίας. Στο τέλος του κεφαλαίου παρουσιάζονται μερικά από τα δεδομένα του Google Scholar, τόσο για τον τρόπο προβολής του, δηλαδή τη γραφική διεπαφή, όσο και για τον τρόπο αναπαράστασης σε HTML.

Στο κεφάλαιο 3 προβάλλονται οι τεχνολογίες και τεχνικές όπου έχουν χρησιμοποιηθεί στην παρούσα διατριβή. Στην αρχή παρουσιάζεται η έννοια web scrapping μαζί με τους τρόπου όπου μπορεί να επιτευχθεί καθώς και τους τρόπους τόσο για επίτευξή του στόχου όσο και τρόπους για την αποφυγή scrapping ενός συστήματος. Στη συνέχεια παρουσιάζονται οι τεχνολογίες που χρησιμοποιούνται στην εφαρμογή και κλείνει με την παρουσίαση του εργαλείου git για τη διαχείριση πολλαπλών εκδόσεων.

Στο επόμενο κεφάλαιο 4 γίνεται λόγος για το τρόπο σχεδίασης της εφαρμογής δηλαδή το πρακτικό κομμάτι. Παρουσιάζεται η δομή του έργου, ο τρόπος λειτουργίας και επικοινωνίας όλης της εφαρμογής. Επίσης, προβάλλονται το σημαντικά μέρη του κώδικα ώστε ένας άνθρωπος με την παρουσία ή την απουσία γνώσεων προγραμματισμού να έχει τη δυνατότητα κατανόησης του έργου.

Επιπροσθέτως στο πέμπτο κεφάλαιο γίνεται παρουσίαση της γραφικής διεπαφής. Παρουσιάζεται το περιβάλλον όπου είναι ορατό στον χρήστη, οι αλληλεπιδράσεις όπου μπορεί να κάνει καθώς και το αποτέλεσμα όπου επιφέρουν αυτές. Κατά σειρά χωρισμένο σε κεφάλαια προβάλλονται οι κύριες λειτουργίες του έργου. Ενώ το κεφάλαιο κλείνει συνοψίζοντας τα πιθανά σενάρια χρήσης και η χρησιμότητα όπου μπορούν να επιφέρουν.Τέλος, στο τελευταίο κεφάλαιο γίνεται μια σύνοψή όλης της εργασίας και παρουσιάζονται οι μελλοντικές επεκτάσεις όπου μπορούν να γίνουν στην παρούσα διατριβή.

Κεφάλαιο 2ο: Δεδομένα επιστημομετρίας

2.1 Google Scholar

Το Google Scholar [5] αποτελεί μια μηχανή αναζήτησης του διαδικτύου, προσβάσιμη από όλους, σκοπός του είναι η ευρετηρίαση κειμένων και μεταδιδόμενων που προκύπτουν από την επιστημονική κοινότητα. Η πρώτη κυκλοφορία του Google Scholar έγινε τον Νοέμβριο του 2004 σε πειραματικά στάδια. Το ευρετήριο περιλαμβάνει διαδικτυακά περιοδικά, έγγραφα συνεδρίων, διπλωματικές εργασίες καθώς και οτιδήποτε άλλο θεωρείτε διατριβή πάνω σε έναν επιστημονικό κλάδο. Το ευρετήριο για την ενσωμάτωση δεδομένων χρησιμοποιεί έναν web crawler για τον εντοπισμό και την ενσωμάτωση καινούργιων επιστημονικών έργων. Για να γίνει αποδεκτή η ενσωμάτωση μιας έρευνας πρέπει να πληρεί συγκεκριμένα κριτήρια.

- Κύρια προϋπόθεσή για τον εντοπισμό του έργου ενός επιστήμονα είναι η διαθεσιμότητα του στο διαδίκτυο.
- Πρέπει το έργο να είναι δημοσιευμένο σε έναν εγκεκριμένο και αξιολογημένο από ομότιμους χώρο. Πρακτικά θα μπορούσε να είναι είτε ένα εγκεκριμένο ερευνητικό περιοδικό είτε μια ακαδημαϊκή δημοσίευση.
- Πρέπει να περιέχει επιστημονικό περιεχόμενο καθώς η ίδια η υπηρεσία έχει σχεδιαστεί για την ευρετηρίαση ακαδημαϊκής βιβλιογραφίας, οπότε τα έργα που έχουν κυρίως διαφημιστικό ή εμπορικό χαρακτήρα είναι λιγότερο πιθανό να συμπεριληφθούν.
- Πρέπει να είναι σωστά μορφοποιημένο και να παρατίθεται σωστά. Το Google Scholar βασίζεται στη δομή και τη μορφοποίηση της ακαδημαϊκής βιβλιογραφίας για την ακριβή ευρετηρίαση και κατάταξη του περιεχομένου. Έργα που είναι κακώς μορφοποιημένα ή δεν έχουν τις κατάλληλες παραπομπές ενδέχεται να έχουν λιγότερες πιθανότητες να ευρετηριαστούν.

Ιστορικά η ιδέα του Google Scholar [5] προέκυψε τυχαία από μια συνομιλία του Alex Verstak και του Anurag Acharya, υπάλληλοι της Google πάνω στην κατασκευή του κύριο ευρετηρίου της. Το κίνητρο τους ήταν να κάνουν τους ανθρώπους όπου επιλύουν προβλήματα πιο αποτελεσματικούς, κάνοντας την επιστημονική γνώση πιο προσιτή και ακριβέστερη. Ο παραπάνω στόχος αντικατοπτρίζεται στη διαφημιστική καμπάνια τις τότε εποχής όπου περιείχε τη φράση ”Stand on the shoulders of giants” σε ελεύθερη μετάφραση ”Στηρίζοντας στους ώμους γιγάντων”. Ατάκα όπου προέρχεται από τον Bernard of Chartres και αποτελεί αναφορά σε όλη την επιστημονική κοινότητα όπου συμβάλει στους τομείς κατά τους αιώνες, προσφέροντας στήριγμα για καινούργια κατορθώματα της ανθρωπότητας.

Το έναυσμα για τα χαρακτηριστικά του Google Scholar ήταν η συλλογές του Πανεπιστημίου του Michigan. Βέβαια, με την πάροδο του χρόνου αυτά τα χαρακτηριστικά έχουν καλλιεργηθεί. Συγκεκριμένα, το 2006 προστέθηκε στις λειτουργίες του η εισαγωγή βιβλιογραφικών παραπομπών από το RefWorks, το RefMan, το EndNote και το BibTeX. Στο επόμενο έτος, ξεκίνησε μια προσπάθεια ψηφιοποίησης βιβλίων και άρθρων όπου δεν περιείχαν μεταδεδομένα με τη συγκατάθεση των εκδοτών τους, ξεχωριστή από το Google Books [6]. Το 2013 προστέθηκε μια λειτουργία για τους συνδεδεμένους χρήστες όπου επέτρεπε την αποθήκευση των αποτελεσμάτων μιας αναζήτησης σε μια προσωπική συλλογή, στην οποία μπορεί να γίνει

επιπλέων αναζήτηση καθώς και οργάνωση με ετικέτες. Ταυτόχρονα με την ανάπτυξη του Scholar, αναπτύχθηκαν υπηρεσίες από άλλες εταιρείες, μερικοί ανταγωνιστές του που παρουσιάστηκαν ανά τα έτη είναι το Scirus, το CiteSeer και η ακαδημαϊκή αναζήτηση του Microsoft Windows Live.

Συγκριτικά με τις περισσότερες αντίστοιχες υπηρεσίες όπου η κατάταξη των αποτελεσμάτων προκύπτει από μία μεταβλητή. Όπως την ημερομηνία δημοσίευσης ή ο αριθμός αναφορών το Google Scholar χρησιμοποιεί έναν εξελιγμένο αλγόριθμο κατάταξης όπου χρησιμοποιεί πολλαπλές μεταβλητές, η καθεμία με διαφορετική βαρύτητα. Η μεγαλύτερη βαρύτητα δίνετε στον αριθμό παραπομπών και στη συνέχεια στον τίτλο. Το παραπάνω συμπέρασμα προέκυψε έπειτα από έρευνα όπου στις αναζητήσεις με βάση τον συγγραφέα ή το έτος, το Scholar επέστρεφε πρώτα αποτελέσματα τα άρθρα με υψηλό αριθμό αναφορών.

2.2 Δεδομένα

Το Google Scholar παρέχει τη δυνατότητα προβολής και παρακολούθησης των στατιστικών ενός ερευνητή. Τα δεδομένα που παρέχει είναι οι δημοσιεύσεις (publications), οι αναφορές (citations) καθώς και ο δείκτης h. Αυτά τα δεδομένα παρέχονται ελεύθερα στους επισκέπτες της πλατφόρμας χωρίς να χρειάζεται περαιτέρω σύνδεση ή πληρωμή. Η καταχώρηση των δεδομένων γίνεται αυτόματα στην πλατφόρμα με τη χρήση crawlers χωρίς αυτό να σημαίνει ότι δεν υπάρχει η δυνατότητα επεξεργασίας. Σε περίπτωση όπου ένας ερευνητής επιθυμεί να παραμετροποιήσει τα δεδομένα του, όπως για παράδειγμα την εναλλαγής τίτλου μιας δημοσίευσης, πρέπει να έχει ταυτοποιήσει τον λογαριασμό του με το ερευνητικό προφίλ και να είναι συνδεδομένως.

Η παροχή δημοσιεύσεων παρέχει τη δυνατότητα προβολής σε κατάλογο τα άρθρα, τις εργασίες και το ευρύτερο ερευνητικό έργο όπου έχει δημοσιευτεί από έναν μελετητή. Αυτό το εργαλείο μπορεί να φανεί χρήσιμο σε περιπτώσεις όπου υπάρχει ανάγκη αναζήτησης ενός συγκεκριμένου άρθρου ή ακόμα την προβολή εξειδίκευσης ενός μελετητή πάνω στον επιστημονικό του τομέα.

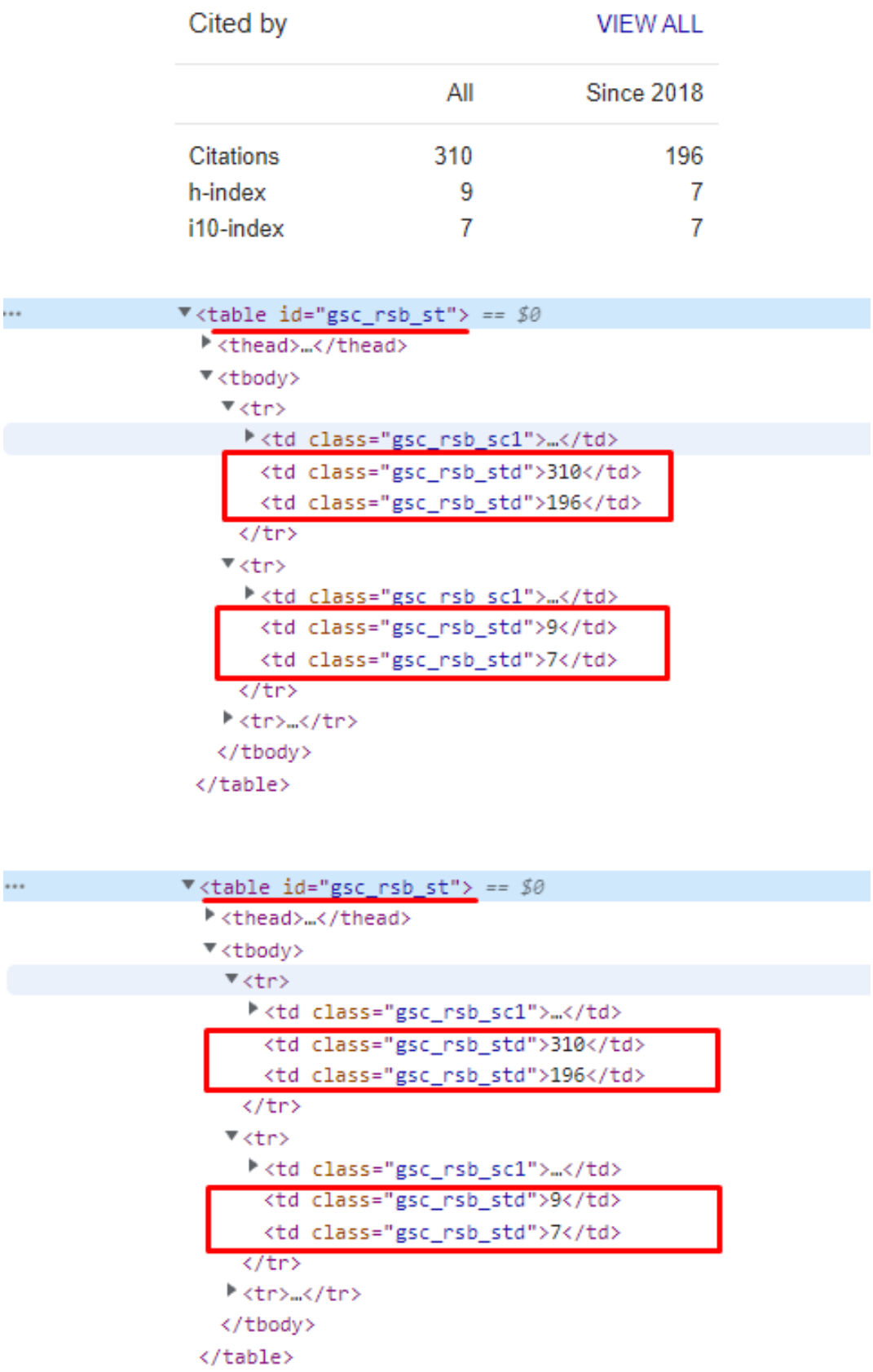
Η δυνατότητα παραπομπών-αναφορών του Google Scholar παρέχει μια λύση για την προβολή των αναφορών όπου έχουν γίνει πάνω σε ένα δημοσιευμένα άρθρο από άλλους μελετητές. Πιο συγκεκριμένα ο όρος αναφορά δείχνει τις συνολικές φορές όπου ένα ερευνητικό έργο χρησιμοποιήθηκε για τη δημιουργία καινούργιου έργου και προστέθηκε στη βιβλιογραφία. Κάτι τέτοιο μπορεί να φανερώσει τη χρησιμότητα και την απήχηση ενός αντίστοιχου έργου.

Ο δείκτης h [5] ή αλλιώς δείκτης Hirsch του Google Scholar, είναι ένας αριθμός όπου χρησιμοποιείται για να μετρήσει τη συνεισφορά και τη σημαντικότητα της επιστημονικής του εργασίας ενός επιστήμονα. Ο δείκτης h αξιολογεί την επιρροή ενός ερευνητή βασιζόμενος στον αριθμό και την ποιότητα των αναφορών που έχουν λάβει τα άρθρα του. Ο τρόπος υπολογισμού του είναι ο εξής: λαμβάνουμε υπόψη τον συνολικό αριθμό άρθρων του ερευνητή και τα ταξινομούμε με βάση τον αριθμό των αναφορών που έχουν λάβει. Ο δείκτης h αντιστοιχεί στον μεγαλύτερο αριθμό h, όπου τουλάχιστον h άρθρα έχουν τουλάχιστον h αναφορές.

Σε πιο απλά λόγια, ο δείκτης h μας δίνει μια εικόνα για το πόσο επιρρεαστικός είναι ένας ερευνητής με βάση τον αριθμό των αναφορών που έχουν λάβει τα άρθρα του. Όταν λέμε ότι ένας ερευνητής έχει δείκτη h ίσο με 10, αυτό σημαίνει ότι έχει τουλάχιστον 10 άρθρα που έχουν λάβει τουλάχιστον 10 αναφορές. Έτσι, με αυτήν τη μέτρηση μπορούμε να αξιολογήσουμε την επιρροή και τη σημαντικότητα του έργου

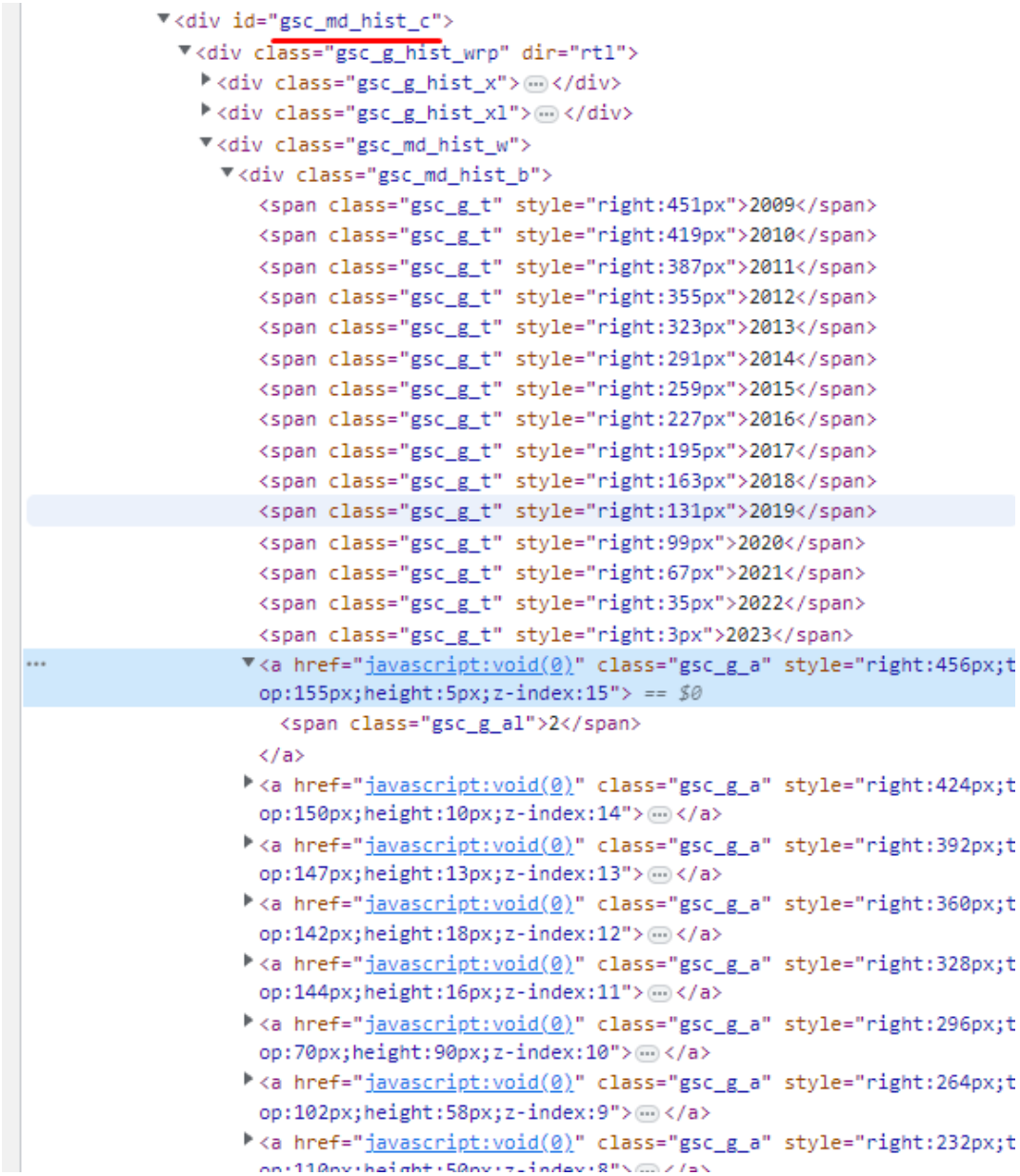
ενός ερευνητή στην επιστημονική κοινότητα.

Πέρα από τη χρησιμότητα αυτού, ο δείκτης h έρχεται με συγκεκριμένους περιορισμούς. Για παράδειγμα, υπάρχει περίπτωση ανακρίβειας του συγκεκριμένου δείκτη με αποτέλεσμα να μην αντανακλά την πραγματική απήχησης ενός ερευνητή, επειδή επηρεάζεται από παράγοντες όπως η αυτοαναφορά. Παρά τους περιορισμούς της κάθε μετρικής οι λειτουργίες του Google Scholar είναι ένα χρήσιμο εργαλείο για ερευνητές, επιστήμονες, βιβλιοθηκονόμους και ενδιαφερόμενους για την εύρεση και προβολή την επιστημονικής βιβλιογραφίας και τη συνεισφορά του κάθε ερευνητή.



Σχήμα 2.1: Δομή DOM από την ιστοσελίδα Google Scholar για την εξαγωγή δεδομένων συνολικών αναφορών και δείκτη h.

Στο σχήμα 2.1 παρουσιάζεται το DOM όπου περιέχει τα δεδομένα των συνολικών και τον τελευταίων 5 ετών των αναφορών και του δείκτη h. Το πρώτο βήμα για την ανάκτηση της πληροφορίας είναι να γίνει ένα αίτημα τύπου GET με το πρωτόκολλο https στη διεύθυνση `"https://scholar.google.com/citations?hl=en&user=CeWUj1YAAAAJ"`. Το αίτημα μας περιέχει στη μεταβλητή user το αναγνωριστικό ή αλλιώς scholar id του ερευνητή. Στη συνέχεια με τη βιβλιοθήκη beautiful soup [7] κάνουμε αναζήτηση του table element με id=”gsc_rsb_st”. Αφού βρεθεί γίνεται αναζήτηση στα στοιχεία td μέσα στο table με τη χρήση των κλάσεων. Η πληροφορία βρίσκεται στο κείμενο του κάθε στοιχείου. Για παράδειγμα, για την εξαγωγή των συνολικών αναφορών, στο table γίνεται αναζήτηση του td στοιχείο με class =”gsc_rsb_std” και εξάγεται το κείμενο, όπου στην περίπτωση του σχήματος 2.1 είναι ίσο με 310.



Σχήμα 2.2: Δομή DOM από την ιστοσελίδα Google Scholar για την εξαγωγή δεδομένων αναφορών ανά έτος.

Στο σχήμα 2.2 παρουσιάζεται το DOM όπου περιέχει τα δεδομένων των αναφορών ενός ερευνητή α-

νά έτος. Το πρώτο βήμα για την ανάκτηση της πληροφορίας είναι να γίνει ένα αίτημα τύπου GET με το πρωτόκολλο https στη διεύθυνση `"https://scholar.google.com/citations?hl=en&user=CeWUj1YAAAAJ#d=gsc_md_hist"`. Το αίτημα μας περιέχει στη μεταβλητή user το αναγνωριστικό ή αλλιώς scholar id του ερευνητή καθώς και με την εισαγωγή του #d=gsc_md_hist στον σύνδεσμο εμφανίζεται το ιστόγραμμα με τις αναφορές του ερευνητή ανά έτος. Αφού ολοκληρωθεί το αίτημα γίνεται αναζήτηση του div element με id=”gsc_md_hist_b”. Αφού βρεθεί με τη χρήση κανονικοποιημένων εκφράσεων (regex) συλλέγονται σε δύο διαφορετικές μεταβλητές όλα τα στοιχεία όπου ταιριάζουν στην έκφραση ”px”>(\d+)” και στην έκφραση ”<a class=”gsc_g_a” href=” javascript : void(0)”. Αναλυτικότερα στο σχήμα 2.2 γίνεται συλλογή όλων των span που περιέχουν το έτος και όλων τον a όπου περιέχουν τις αναφορές. Έπειτα από μελέτη της πλατφόρμας είναι γνωστό ότι το τρέχον έτος θα είναι το span όπου περιέχει το ” style =” right :3px”” και για κάθε προηγούμενη χρονιά προστίθενται 32px στο style. Επίσης, για τα a είναι γνωστό ότι η τρέχουσα χρονιά θα περιέχει το ” style =” right :8px”” και για κάθε προηγούμενη χρονιά προστίθενται 32px στο style. Για κάθε χρόνο εξάγεται το κείμενο από το span ενώ για κάθε αναφορά εξάγεται το κείμενο από παιδί στοιχείο του a. Με αυτό τον τρόπο γίνεται η αντιστοιχία ετών με αναφορές. Σε περίπτωση όπου δεν αντιστοιχιστεί το έτος με την αναφορά, αυτομάτως είναι γνωστό ότι οι αναφορές εκείνο το έτος ήταν ίσες με μηδέν.

Σχήμα 2.3: Δομή DOM από την ιστοσελίδα Google Scholar για την εξαγωγή δημοσιεύσεων.

Στο σχήμα 2.3 παρουσιάζεται το DOM όπου περιέχει τα δεδομένων των δημοσιεύσεων. Αρχικά γίνεται ένα αίτημα τύπου GET με το πρωτόκολλο https στη διεύθυνση `"https://scholar.google.com/citations?hl=en&user=CeWUj1YAAAAJ&cstart=0&pagesize=100"`.Το αίτημα μας περιέχει στη μεταβλητή user το αναγνωριστικό ή αλλιώς scholar id του ερευνητή, στη μεταβλητή cstart το αρχικό citation και στη μεταβλητή pagesize το πλήθος δημοσιεύσεων όπου θα φορτωθούν.Δηλαδή στη συγκεκριμένη περίπτωση φορτώνονται οι πρώτες 100 δημοσιεύσεις. Οπότε για την εξαγωγή όλων των δημοσιεύσεων απαιτούνται πολλαπλά αιτήματα με διαφορετικές τιμές στη μεταβλητή cstart. Αφού ολοκληρωθεί το αίτημα γίνεται αναζήτηση του table element με id=”gsc_a_t”. Κάθε tr μέσα στο table αποτελεί και μία δημοσίευση του ερευνητή. Ο τίτλος της δημοσίευσης είναι το κείμενο μέσα στο a element με class=”gsc_a_at”. Ενώ τα επόμενα δύο div element με class=”gs_gray” περιέχει τους ερευνητές όπου βοήθησαν στην ολοκλήρωση του έργου καθώς και το συνέδριο όπου παρουσιάστηκε. Ακόμα, στο κείμενο του a element με class=”gsc_a_ac gs_ibl” βρίσκονται οι αναφορές αυτής της δημοσίευσης. Επίσης, στο κείμενο του span element με class=”gsc_a_h gsc_a_hc gs_ibl” βρίσκεται η χρονολογία όπου δημο-

σιεύτηκε το άρθρο. Οι συνολικές δημοσιεύσεις του ερευνητή προκύπτουν απτό άθροισμα του πλήθους όλως των δημοσιεύσεων που βρέθηκαν. Τέλος, για την αντιστοιχία ετών με δημοσιεύσεις προστίθενται οι δημοσιεύεις με κοινό έτος στο span element με class =”gsc_a_h gsc_a_hc gs_ibl”.

Κεφάλαιο 3ο: Μοντέλα και Τεχνολογίες

3.1 Web Scraping

Το web scraping, επίσης γνωστό ως web harvesting ή web data extraction, είναι η διαδικασία συλλογής και εξαγωγής δεδομένων από ιστοσελίδες. Πρόκειται για μια διαδικασία όπου μπορεί να γίνει είτε χειροκίνητα είτε αυτοματοποιημένα με τη χρήση ενός bot ή ενός web crawler.Η συλλογή δεδομένων γίνεται έχοντας πρόσβαση στον Παγκόσμιο Ιστό (www) και με την υποβολή αιτήσεων HTTP στον διακομιστή μιας ιστοσελίδας, τη λήψη της σελίδας HTML ή XML και την ανάλυση των δεδομένων για να εξαχθούν τα επιθυμητά στοιχεία. Το web scraping μπορεί να είναι χρήσιμο για μια ποικιλία σκοπών, όπως η συλλογή δεδομένων για έρευνα ή ανάλυση, η αυτοματοποίηση μιας εργασίας που συμπεριλαμβάνει δεδομένα από πολλές ιστοσελίδες ή η δημιουργία ενός αντίγραφου μιας ιστοσελίδας για προβολή εκτός σύνδεσης. [8]

3.1.1 Τεχνικές για scraping

Ανάλογα από το επίπεδο του χρήστη, τη δομή της ιστοσελίδας ή ακόμα τον σκοπό που αποσκοπεί αυτή η διαδικασία το web scraping μπορεί να επιτευχθεί με διάφορους τρόπους.

- Ανθρώπινη αντιγραφή-επικόλληση: Στην πιο απλή του μορφή ένας χρήστης όπου επιθυμεί να συλλέξει δεδομένα μπορεί να τα αντιγράψει και να τα επικολλήσει από έναν διαδικτυακό ιστότοπο απευθείας σε ένα έγγραφο κειμένου (.txt) ή σε ένα υπολογιστικό φύλλο (excel). Μάλιστα σε μερικές περιπτώσεις η καλύτερη scaping μεθοδολογία δεν μπορεί να ξεπεράσει την ανθρώπινη κρίση, με αποτέλεσμα η αντιγραφή-επικόλληση να είναι η μοναδική μεθοδολογία όπου επιφέρει λύση σε προβλήματα όπου δεν μπορούν να αυτοματοποιηθούν.
- Εύρεση μοτίβου κειμένου: Μια ακόμα απλή, αλλά εξίσου χρήσιμη προσέγγιση είναι η εξαγωγή δεδομένων με την grep εντολών ή με τη δημιουργία regular expression χρησιμοποιώντας μια γλώσσα προγραμματισμού.
- HTTP αιτήματα: Στατικές και δυναμικές σελίδες μπορούν να φορτωθούν αιτώντας HTTP ερωτήματα και στην συνέχεια να επεξεργαστούν με μια γλώσσα προγραμματισμού.
- Ανάλυση του DOM: Μια από τις πιο συχνές μορφές, όπου γίνετε ενσωμάτωση ενός προγράμματος περιήγησης, για παράδειγμα Google Chrome, Mozilla με σκοπό το φόρτωμα της ιστοσελίδας.Σε δεύτερο χρόνο εξαγωγή της θεμιτής πληροφορίας αναλύοντας τη δομή του δέντρου DOM.
- Κάθετη συνάθροιση (vertical aggregation): Αναφέρεται στην ενέργεια συλλογής δεδομένων από πολλαπλές πηγές για έναν συγκεκριμένο τομέα και συγκέντρωση της πληροφορίας σε μια κοινή βάση δεδομένων. Τα τελευταία χρόνια έχουν ξεπροβάλει εταιρείες όπου μεταφράσουν την πληροφορία σε στατιστικές αναλύσεις και προβλέπουν μελλοντικές μεταβολές της οικονομίας.
- Μηχανική μάθηση : Τα τελευταία χρόνια έχουν γίνει προσπάθειες όπου με τη χρήση μηχανικής μάθησης προσπαθούν να εντοπίσουν και να εξάγουν πληροφορίες ερμηνεύοντας τις σελίδες οπτικά όπως θα έκανε ένας άνθρωπος.

- Λογισμικό : Υπάρχουν αρκετά εργαλεία λογισμικού όπου παρέχουν λογισμικό το οποίο μπορεί να αναλύσει αυτόματα τη δομή δεδομένων μιας σελίδες. Με την κατάλληλη παραμετροποίηση έχουν τη δυνατότητα να εξάγουν πληροφορία και να την αποθηκεύσουν σε μια βάση δεδομένων, χωρίς να είναι απαραίτητη συγγραφή χειροκίνητου κώδικα. Τέλος, ορισμένα από αυτά τα λογισμικά παρέχουν απευθείας πρόσβαση σε API.

3.1.2 Διαθέσιμες Βιβλιοθήκες

Οι ιστοσελίδες κατασκευάζονται με τη χρήση γλωσσών σήμανσης κειμένου (HTML και XHTML) και συχνά περιέχουν πλήθος χρήσιμων δεδομένων σε μορφή κειμένου. Ωστόσο, οι περισσότερες ιστοσελίδες έχουν σχεδιαστεί για τους τελικούς χρήστες δηλαδή τους επισκέπτες της σελίδας και όχι για εύκολη αυτοματοποιημένη χρήση. Ως αποτέλεσμα, με την πάροδο του χρόνου έχουν ξεπροβάλει πολλά εργαλεία και βιβλιοθήκες διαθέσιμα για την παραπάνω διαδικασία ώστε να κάνουν την εξαγωγή δεδομένων όσο πιο εύκολη και προσιτή. Μερικές δημοφιλείς είναι οι εξής:

- Beautiful Soup (Python): μια από τις πιο γνωστές βιβλιοθήκες για την ανάλυση εγγράφων HTML και XML.
- Cheerio (JavaScript): μια βιβλιοθήκη που επιτρέπει τον χειρισμό της HTML με τον ίδιο τρόπο από αυτό της jQuery.
- Scrapy (Python): ένα εργαλείο όπου διευκολύνει τη δημιουργία και τη συντήρηση web scrapers.
- Puppeteer (JavaScript): μια βιβλιοθήκη που παρέχει υψηλού επιπέδου API για τον έλεγχο ενός headless προγράμματος περιήγησης Chrome ή Chromium. Μπορεί να χρησιμοποιηθεί για web scraping, καθώς και για την αυτοματοποίηση εργασιών και τη δοκιμή εφαρμογών ιστού.
- Selenium (πολλές γλώσσες): μια βιβλιοθήκη που σας επιτρέπει να ελέγχετε ένα πρόγραμμα περιήγησης ιστού μέσω κώδικα, καθιστώντας εύκολη την εξαγωγή πληροφορίας από ιστοσελίδες που απαιτούν JavaScript για να φορτώσουν το περιεχόμενό τους.

3.1.3 Λόγοι για scraping και τρόπος λειτουργίας

Το web scrapping [8] μπορεί να χρησιμοποιηθεί ως κύριο στοιχείο εφαρμογών που χρησιμοποιούνται για ευρετηρίαση ιστού, εξόρυξη ιστού και εξόρυξη δεδομένων, παρακολούθηση αλλαγών τιμών στο διαδίκτυο και σύγκριση τιμών, παρακολούθηση κριτικών προϊόντων (για την παρακολούθηση του ανταγωνισμού), συλλογή καταχωρίσεων ακινήτων, παρακολούθηση δεδομένων καιρού, ανίχνευση αλλαγών στον ιστότοπο, έρευνα, παρακολούθηση της παρουσίας και της φήμης στο διαδίκτυο και ενσωμάτωση διαδικτυακών δεδομένων.

Οι νεότερες μορφές web scrapping περιλαμβάνουν τη συλλογή δεδομένων απευθείας από τους διακομιστές ιστού, χωρίς για χρειάζεται να αντλήσουν την πληροφορία μέσα από τη HTML. Για παράδειγμα, αφού γίνει το http request και ο διακομιστής απαντήσει, τότε το scraping γίνεται απευθείας στο παραγόμενο πακέτο, όπου συνήθως είναι της μορφής json ή xml. Με αυτή την τεχνική το scraping καθιστάτε γρηγορότερο αλλά σε αρκετές περιπτώσεις χρειάζεται λιγότερη συντήρηση καθώς η html μιας σελίδας αλλάζει συχνότερα από τις απαντήσεις του διακοσμητή.

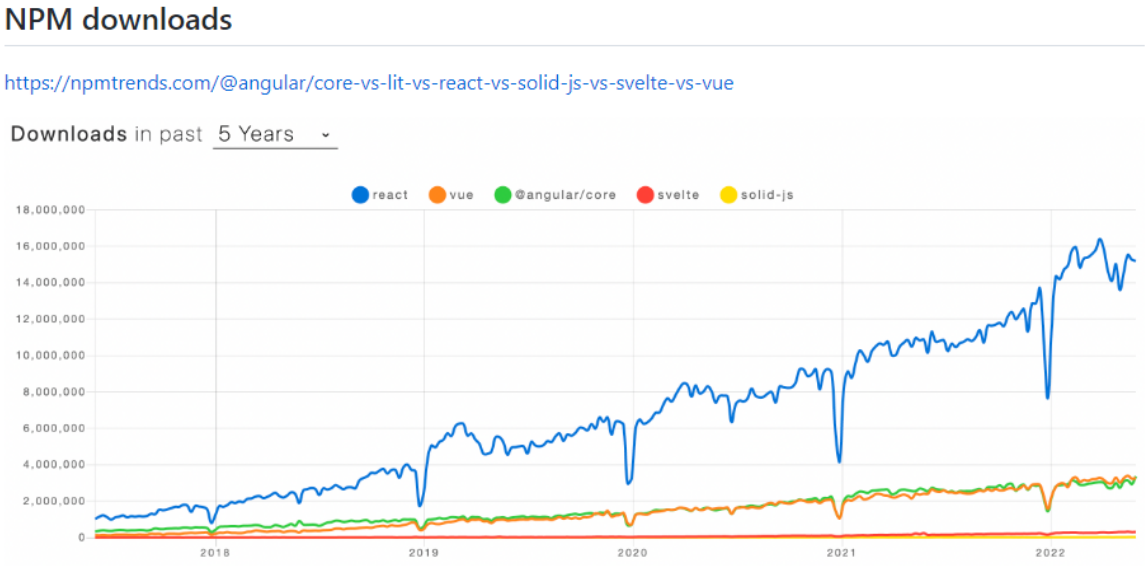
3.1.4 Τεχνικές για την αποτροπή scraping

Το web scraping [8] παρά την ραγδαία εξέλιξη του, αποτελεί από αρκετές ιστοσελίδες μη θεμιτό τρόπο για προβολή δεδομένων. Για αυτό τον λόγο έχουν εξελιχθεί και οι τεχνικές για την απότρεψη αυτού του σκοπού [9] [10]. Μερικές από τις τεχνικές είναι οι εξής:

- Χρήστη CAPTCHAS:Τα ευρέως γνωστά CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) δημιουργήθηκαν με σκοπό να αποτρέψουν την πρόσβαση μη φυσικών χρηστών σε ιστοτόπους. Συνήθως τα CAPTCHAs περιέχουν μια πρόκληση εύκολη προς τον άνθρωπο αλλά δύσκολη για να τη λύσει ένα πρόγραμμα, όπως το αποτέλεσμα από μια μαθηματική πράξη είτε την ολοκλήρωση ενός παζλ πριν σου δώσουν πρόσβαση στην πληροφορία.Αξίζει να σημειωθεί ότι το πιο πρόσφατο CAPTCHA της google αναγνωρίζει τον τύπο του χρήστη βάση τον τρόπο όπου περιηγείτε.
- Περιορισμός ρυθμού: Ορισμένοι διακομιστές παρακολουθούν τον αριθμό αιτήσεων που πραγματοποιούνται από μια συγκεκριμένη IP διεύθυνση και θέτουν ένα όριο αιτημάτων όπου επεξεργάζονται μέσα σε ένα χρονικό πλαίσιο, προκειμένου να αποτραπεί το scraping. Αυτό μπορεί να είναι ένας αποτελεσματικός τρόπος για να αποτρέψετε τα bots από το να κατακλύζουν τον διακομιστή με αιτήσεις και να επιβραδύνουν την υπηρεσία σας άσκοπα.
- Αποκλεισμός IP διευθύνσεων: Στην περίπτωση όπου ένας διακομιστής εντοπίσει ότι μια συγκεκριμένη διεύθυνση IP κάνει μεγάλο αριθμό αιτήσεων σε μικρό χρόνο, το οποίο μεταφράζετε σε ασυνήθιστη συμπεριφορά χρήστη. Τότε μπλοκάρει τη συγκεκριμένη διεύθυνση και δεν απαντάει στα αιτήματα της.
- Χρήση cookies: Ορισμένοι ιστότοποι κάνοντας χρήση των cookies παρακολουθούν την συμπεριφορά των χρηστών. Εάν ένας scraper δεν είναι σε θέση να χειριστεί σωστά τα cookies και εντοπιστεί από τον διακομιστή τότε μπλοκάρετε η διεύθυνση του.
- Χρήση ανίχνευσης χρήστη: Γνωστές βιβλιοθήκες για scraping πραγματοποιούν αιτήματα με συγκεκριμένο τρόπο. Σε περίπτωση όπου ο διακομιστής εντοπίσει κάποιο από τα μοτίβα της βιβλιοθήκης και αναγνωρίσει ότι ο χρήστης που κάνει το αίτημα είναι scraper ή crawler τότε μπλοκάρει την αίτηση.
- Χρήση κρυπτογράφησης: Μερικές ιστοσελίδες κρυπτογραφούν το περιεχόμενο τους με σκοπό να το κάνουν πιο δύσκολη τη διαδικασία εξαγωγής δεδομένων.Για παράδειγμα, μπορεί μια σελίδα να χρησιμοποιήσει τεχνικές κωδικοποίησης της ιστοσελίδας με JavaScript ώστε το περιεχόμενο της σελίδας να είναι δύσκολο να αποκωδικοποιηθεί εκτός αν εκτελεστεί από ένα πρόγραμμα περιήγησης.
- Χρήση συνδρομής ή απαίτηση σύνδεσης: Ορισμένες ιστοσελίδες μπορεί να απαιτούν την πληρωμή συνδρομής ή τη σύνδεση με έναν έγκυρο λογαριασμό ώστε να παρέχουν πρόσβαση στους χρήστης στο περιεχόμενο.
- Νομική δράση: Σε ορισμένες καταστάσεις, διαδικτυακοί τόποι όπου εντοπίσουν έναν scraper και βρουν το άτομο όπου ευθύνεται μπορούν να προβούν σε νομικούς φορείς για την απόσταση πληροφορίας χωρίς θεσμοθετημένο πλαίσιο.

3.2 React

Η React [11] είναι μια βιβλιοθήκη, ανοιχτού κώδικα, υλοποίησης UI βασισμένη στη JavaScript. Ξεκίνησε η ανάπτυξη της το 2011 από τον Jordan Walke και εκδόθηκε το 2013 από το Facebook και μια κοινότητα προγραμματιστών ανοιχτού κώδικα. Παρά το γεγονός ότι η React είναι μια βιβλιοθήκη και όχι μία γλώσσα, όπου αναφορικά με το M-V-C μοντέλο καλύπτει το επίπεδο προβολής (V) έχει καταφέρει να αναδυθεί στην κλίμακα των τεχνολογιών όπου χρησιμοποιούνται για την ανάπτυξη ιστοσελίδων. Αξίζει να σημειωθεί, παρά το γεγονός ότι η βιβλιοθήκη δημιουργήθηκε για να παρέχει λύσεις στη δημιουργία ιστοσελίδων η ανάπτυξη της δε σταμάτησε εκεί. Καθώς προσφέρει διάφορες επεκτάσεις, όπως το Flux για την οπτικοποίηση των λειτουργιών μιας εφαρμογές όπως επίσης τη React Native για την υλοποίηση εφαρμογών κινητών συσκευών.



Σχήμα 3.1: Αριθμός συνολικών λήψεων από JavaScript βιβλιοθήκες, τα τελευταία 5 χρόνια.

3.2.1 Ο χειρισμός της React πάνω στο dom

Το εικονικό DOM [11] (Virtual document object model) είναι το σημείο αναφοράς για τη βιβλιοθήκη React. Είναι ένα μικρού μεγέθους αντίγραφο του κανονικού DOM που δημιουργείται από το πρόγραμμα περιήγησης, με τη διαφορά ότι φορτώνεται και διατηρείται στη μνήμη. Οι αλλαγές όπου προκαλούνται σε μια ιστοσελίδα κατά την περιήγηση, αντικατοπτρίζονται πρώτα στο εικονικό DOM και ύστερα γίνονται μεταβολές στο πραγματικό DOM μόνο στα σημεία όπου χρειάζεται, χωρίς να χρειάζεται το φόρτωμα ολόκληρου του DOM. Πράγμα όπου καθιστά την βιβλιοθήκη αυτόματος ταχύτερη από τις συμβατικές υλοποιήσεις.

Η React JS [11] είναι υλοποιημένη με τέτοιο τρόπο ώστε η ροή δεδομένων να είναι μονής κατεύθυνσης. Η μονόπλευρη ροή από πάνω προς τα κάτω σχέσης ”πατέρα-παιδιού”, παρέχει έλεγχο ανάμεσα στις καταστάσεις των μοντέλων ενώ ταυτόχρονα καθορίζει τον τρόπο που τα δεδομένα μεταφέρονται και τροποποιούνται μέσα σε μια εφαρμογή. Πιο συγκεκριμένα, στη μονόπλευρη κατεύθυνση τα δεδομένα παρέρχονται από τους κεντρικούς κόμβους στους ενδότερους.Όταν κάποιο στοιχείο από τα δεδομένα υποστεί αλλαγή, μπορεί να προκαλέσει αλλαγές στην εμφάνισης της εφαρμογής, βέβαια η εφαρμογή

δεν μπορεί να διαχειριστεί τα δεδομένα μόνη της. Στη συγκεκριμένες περίπτωση, η ροή εξασφαλίζει απλούστερο και ξεκάθαρο σχεδιασμό για τον κόμβο όπου θα διαχειρίζεται τις αλλαγές της εφαρμογής κατά τη μεταβολή των δεδομένων.

Η React [11] μοντελοποιεί τη δημιουργία διαδικτυακών εφαρμογών χρησιμοποιώντας τα λεγόμενα συστατικά (components). Τα συστατικά κατά κόρων είναι html tags τα οποία δημιουργούνται στο εικονικό dom και στη συνέχεια μεταφράζονται στο κανονικό. Η διαφορά τους είναι ότι υπάρχει η δυνατότητα μεταφοράς δεδομένων μέσα στα συστατικά καθώς επίσης τα συστατικά δίνουν τη δυνατότητα να χωριστεί η γραφική διεπαφή σε μικρά αυτόνομα κομμάτια.

Τελευταίο χαρακτηριστικό είναι η δυνατότητα σύνταξης JSX [11]. Σημαίνει JavaScript XML και στην πραγματικότητα είναι επέκταση της JavaScript όπου καθορίζει έναν εναλλακτικό τρόπο σύνταξης. Τα πλεονεκτήματα αυτής τις σύνταξης είναι η ταχύτητα, η ασφάλεια και η απλότητα καθώς από το JSX κείμενο προκύπτει κατά τη μεταγλώττιση κώδικας JavaScript.

3.2.2 Πλεονεκτήματα και μειονεκτήματα

Όπως όλες οι βιβλιοθήκες με τη σειρά της και η React [11] δημιουργούνται για να παρέχουν μια λύση σε ένα πρόβλημα. Στις περισσότερες περιπτώσεις η κεντρική θεματολογία των προβλημάτων είναι ο τρόπος όπου μπορεί να γίνει αύξηση της παραγωγικότητας και μείωση του χρόνου υλοποίησης μια εφαρμογής. Το αντίκτυπο που έχει η βιβλιοθήκη σχετικά με τις παραπάνω μεταβλητές φαίνεται στα πλεονεκτήματα και τα μειονεκτήματα της.

Η επιλογή της React κατά την ανάπτυξη μιας εφαρμογής μπορεί να έχει πολλά πλεονεκτήματα. Ένα από αυτά είναι η χρήση του εικονικού DOM αντί του κανονικού DOM, το οποίο είναι ταχύτερο και βελτιώνει την απόδοση της εφαρμογής. Επίσης, η React είναι ανοιχτού κώδικα, πράγμα που σημαίνει ότι ο κώδικας είναι ελεύθερα διαθέσιμος και μπορεί να τροποποιηθεί από τους χρήστες ανάλογα με τις ανάγκες τους.

Η React καθιστά επίσης εφικτό ένα καταπληκτικό UI, προσφέροντας διάφορα εργαλεία και δυνατότητες για τη δημιουργία εντυπωσιακών διεπαφών χρήστη. Επιπλέον, η React είναι φιλική προς τις μηχανές αναζήτησης, ένα σημαντικό πλεονέκτημα για την εύρεση και την προβολή της εφαρμογής σε μηχανές αναζήτησης.

Επιπροσθέτως παρέχει τη δυνατότητα διευκόλυνσης της διαχείρισης μεγάλων εφαρμογών και η αύξηση της αναγνωσιμότητας του κώδικα. Αυτό είναι δυνατόν χάρη στην αναδιάταξη της αρχιτεκτονικής της εφαρμογής σε μικρά και ανεξάρτητα συστατικά. Τα συστατικά αυτά μπορούν να επαναχρησιμοποιηθούν και να αντικατασταθούν εύκολα, καθιστώντας τη διαχείριση μιας μεγάλης εφαρμογής πολύ πιο εύκολη.

Ένας ακόμα λόγος για την επιλογή της React είναι η δυνατότητα μετατροπής του κώδικα σε εφαρμογή για κινητά. Αυτό είναι εφικτό χάρη στο γεγονός ότι η React παρέχει τα εργαλεία για τη δημιουργία εφαρμογών που είναι συμβατές με τα περισσότερα κινητά συστήματα λειτουργίας.

Η React χρησιμοποιεί μια σύγχρονη σύνταξη JavaScript, τη JSX, η οποία επιτρέπει την ενσωμάτωση HTML κώδικα μέσα στον κώδικα της εφαρμογής. Αυτό καθιστά τη συγγραφή κώδικα ευκολότερη και πιο αποδοτική.

Ένας από τους κύριους λόγους που κάνουν τη React μια επιλογή δημοφιλή για την ανάπτυξη εφαρμογών είναι το μεγάλο και συνεχώς αναπτυσσόμενο οικοσύστημα της. Αυτό προσφέρει στους προγραμματιστές μια πληθώρα από βιβλιοθήκες και εργαλεία που μπορούν να χρησιμοποιήσουν για την ανάπτυξη πιο προηγμένων εφαρμογών. Αυτό μειώνει σημαντικά το χρόνο που απαιτείται για την ανάπτυξη και διατήρηση του κώδικα, καθώς οι προγραμματιστές δε χρειάζεται να εφεύρουν από την αρχή τον τροχό σε κάθε νέο έργο.

Η πληθώρα αυτών των βιβλιοθηκών περιλαμβάνει εργαλεία που καλύπτουν όλα τα επίπεδα της εφαρμογής, από το διαχειριστικό τμήμα έως τον πυρήνα της εφαρμογής. Αυτό καθιστά ευκολότερη την ανάπτυξη εφαρμογών που είναι σύνθετες καθώς και αυτές όπου εξελίσσονται συνεχόμενα με τον χρόνο.

Πέρα από τα πλεονεκτήματα η βιβλιοθήκη React έρχεται και με μερικούς περιορισμούς. Αρχικά, η React παρέχει μόνο τη βιβλιοθήκη γραφικής διεπαφής (UI), ενώ για την υλοποίηση μιας πλήρους εφαρμογής χρειάζεται επιπλέον βιβλιοθήκες για τη διαχείριση του state, των αιτημάτων προς τον server και του routing. Για την υλοποίηση πραγματικά πολύπλοκων εφαρμογών με πολλά δεδομένα, η React απαιτεί τη χρήση πολλαπλών βιβλιοθηκών, κάτι που μπορεί να αυξήσει το μέγεθος του κώδικα και τον χρόνο φόρτωσης της εφαρμογής.

Η React χρησιμοποιεί JSX, μια σύνταξη που συνδυάζει τη JavaScript με το HTML. Αυτό μπορεί να οδηγήσει στον κίνδυνο του inline scripting, δηλαδή της εισαγωγής κώδικα JavaScript απευθείας στο HTML, κάτι που ορισμένοι προγραμματιστές θεωρούν μη ασφαλές και μπορεί να καταστήσει τον κώδικα πιο δύσκολο στη διαχείριση και τη συντήρηση.

Η React χρησιμοποιεί συναρτησιακή προγραμματιστική προσέγγιση αντί για μια βασισμένη σε κλάσεις προσέγγιση, και αυτό μπορεί να δυσκολεύει τους προγραμματιστές που είναι εξοικειωμένοι με τη δομή της προγραμματιστικής προσέγγισης βασισμένης σε κλάσεις.

3.3 Python

Η Python [12] αποτελεί μια γλώσσα προγραμματισμού όπου αναπτύχθηκε μεταξύ τα τέλη της δεκαετίας του 1980 και στις αρχές της δεκαετίας του 1990 με δημιουργό τον Guido van Rossum, στο Εθνικό Ερευνητικό Ινστιτούτο Μαθηματικών και Επιστήμης Υπολογιστών στην Ολλανδία. Η δημιουργία της βασίστηκε σε χαρακτηριστικά άλλων προγραμματιστικών γλωσσών μερικές από αυτές είναι η ABC, C, C++, Algol-68 και Unix shell. Ο πηγαίος κώδικας της Python είναι διαθέσιμος κάτω από την αιγίδα της Γενικής Άδειας Δημόσιας Χρήσης (GPL) η οποία προστατεύει πλήρως τα πνευματικά της δικαιώματα. Σαν γλώσσα συντηρείται πλέον από μια ομάδα του ινστιτούτου όπου τη διευθύνει ο δημιουργός της και από το 2008 μέχρι σήμερα βρισκόμαστε την έκδοση της Python 3. Αξίζει να σημειωθεί ότι οι εκδόσεις της γλώσσα δεν είναι συμβατές με αυτές της προηγούμενης αν και αυτό δεν αποτρέπει εταιρείες όπου φέρουν τον χαρακτηρισμό ”τεχνολογικοί γίγαντες” όπως η Cisco, η IBM, η Mozilla, η Google, η Quora, η Hewlett-Packard, η Dropbox και η Qualcomm να χρησιμοποιήσουν τη συγκεκριμένη γλώσσα. Αιτία για αυτή τη συνήθεια είναι η απλότητα της, σε συνδυασμό με την έμφαση στην αναγνωσιμότητα και την αποδοτικότητα.

Το πεδίο εφαρμογής της Python [12] όπως και η κοινότητά της έχει επεκταθεί πέρα από συμβατικές χρήσεις όπως η χρήση της σε εφαρμογές ιστού ή εφαρμογές υπολογιστή.Πλέον η Python μπορεί να χρη-

σιμοποιηθεί για επεξεργασία εικόνας και σχεδίαση γραφικών, σε επιστημονικές εφαρμογές, σε παιχνίδια και τρισδιάστατα γραφικά, στην ανάπτυξη λογισμικού καθώς και η δομή της έγινε πηγή έμπνευσης για άλλες γλώσσες όπως η Swift και η Cobra. Ο προφανής λόγος για την επιτυχία της Python ήταν τα χαρακτηριστικά της, μερικά από αυτά είναι τα εξής:

- Απλή: Συγκριτικά με άλλες γλώσσες προγραμματισμού, το συντακτικό της γλώσσας θυμίζει τη γραφή απλών αυστηρών προτάσεων κάτι αντίστοιχο με ψευδοκώδικα. Αυτό, επιτρέπει στους χρήστες της να επικεντρωθούν στον πραγματικό σκοπό τους, δηλαδή την επίλυση του προβλήματος, παρά τη σύνταξη της γλώσσας.

- Εύκολη εκμάθηση: Η Python έχει, συγκριτικά με άλλες γλώσσες όπως Java και C, πιο ομαλή καμπύλη εκμάθησης λόγο του συντακτικού της.

- Ανοιχτού κώδικα: Η Python μαζί με τις περισσότερες διαθέσιμες βιβλιοθήκες είναι ανοιχτού κώδικα πράγμα όπου δίνει τη δυνατότητα πρόσβασης και μετατροπής του πηγαίου κώδικα.

- Υψηλό επίπεδο: Είναι μια γλώσσα προγραμματισμού όπου λειτουργεί αφαιρετικά με τις λεπτομέρειες στη σύνταξη. Συγκριτικά με τις υπόλοιπες γλώσσες προγραμματισμού χαμηλού επιπέδου, το συντακτικό της παρέχει εκφράσεις παρόμοιες με αυτές της φυσικής γλώσσας επίσης παρέχει αυτοματοποιημένες λύσεις σε σημαντικούς τομείς των υπολογιστικών συστημάτων, όπως για παράδειγμα η κατανομή πόρων.

- Δυναμική δήλωση μεταβλητών: Οι τύποι των μεταβλητών, των πεδίων των αντικειμένων δε χρειάζεται να δηλωθούν κατά τη συγγραφή του προγράμματος. Αυτό γίνεται επειδή η Python ορίζει τον τύπο της μεταβλητής κατά την εκτέλεση του προγράμματος σύμφωνα με την τιμή όπου θα έχει. Πιο συγκεκριμένα αν κατά την εκτέλεση του προγράμματος μια μεταβλητή πάρει τιμή με χαρακτήρες, εσωτερικά θα δηλωθεί ως αλφαριθμητική (String).

- Διασταυρούμενης πλατφόρμας (cross platform): Η Python μπορεί να εγκατασταθεί σε λειτουργικά συστήματα όπως Windows, Linux και Mac OS. Ένα πρόγραμμα μπορεί να εκτελεστεί σε οποιαδήποτε από τα παραπάνω λειτουργικά συστήματα χωρίς να χρειάζεται καμία αλλαγή.

- Χρήση διερμηνευτή: Υπάρχουν δύο τύποι για να ταξινομηθούν οι γλώσσες μηχανής ανάλογα με τον τρόπο λειτουργίας αυτές που λειτουργούν με μεταγλωττιστές και αυτές με διερμηνέα. Ένα πρόγραμμα γραμμένο σε C, η οποίο χρησιμοποιεί μεταγλωττιστή, απαιτεί μετατροπή του πηγαίου κώδικα σε γλώσσα αναγνώσιμη από τον υπολογιστή, δηλαδή σε γλώσσα μηχανής (δυαδικός κώδικας, μηδέν και ένα). Ο μεταγλωττισμένος κώδικας στη συνέχεια φορτώνεται στη μνήμη ενός υπολογιστή για να εκτελεστεί. Αντίθετα, η Python δεν απαιτεί αυτή τη μετατροπή, το πρόγραμμα εκτελείται απευθείας από τον πηγαίο κώδικα. Πιο συγκεκριμένα, εσωτερικά ο κώδικας μετατρέπεται σε μια ενδιάμεση μορφή γνωστή ως κώδικα byte και στη συνέχεια μεταφράζετε στην αντίστοιχη γλώσσα της εκάστοτε μηχανής. Αποτέλεσμα του χαρακτηριστικού είναι ότι δε χρειάζεται έλεγχος και ανησυχία για το φόρτωμα στη μνήμη.

- Εναλλακτικοί τρόποι προγραμματισμού: Υποστηρίζει διάφορες προσεγγίσεις προγραμματισμού κατά την υλοποίηση, όπως ο λειτουργικός, ο διαδικαστικός και ο αντικειμενοστραφής προγραμματισμός.

- Επεκτάσιμη: Ο κώδικας Python μπορεί να ενσωματωθεί σε ένα πρόγραμμα γραμμένο σε C και το αντίστροφο. Για παράδειγμα, αν χρειάζεται ένα επιμέρους κομμάτι κώδικα από ένα πρόγραμμα να εκτελεστεί γρήγορα, υπάρχει η δυνατότητα να γραφτεί σε C/C++ και στη συνέχεια να χρησιμοποιηθεί το αποτέλεσμα στην Python.

- Ευρύ οικοσύστημα βιβλιοθηκών: Η ενσωματωμένη βιβλιοθήκη της είναι τεράστια και καλύπτει ένα μεγάλο φάσμα διευκολύνσεων. Αυτή η βιβλιοθήκη περιέχει οντότητες γραμμένες κυρίως σε Python και C όπου παρέχουν πρόσβαση σε λειτουργίες του συστήματος και προσφέρουν λύσει σε συχνά προβλήματα του προγραμματισμού. Μερικά από αυτά είναι η επεξεργασία κειμένου, τύποι δεδομένων σε αριθμητικά και μαθηματικά στοιχεία, κρυπτογραφία και πρωτόκολλα υποστήριξης διαδικτύου.

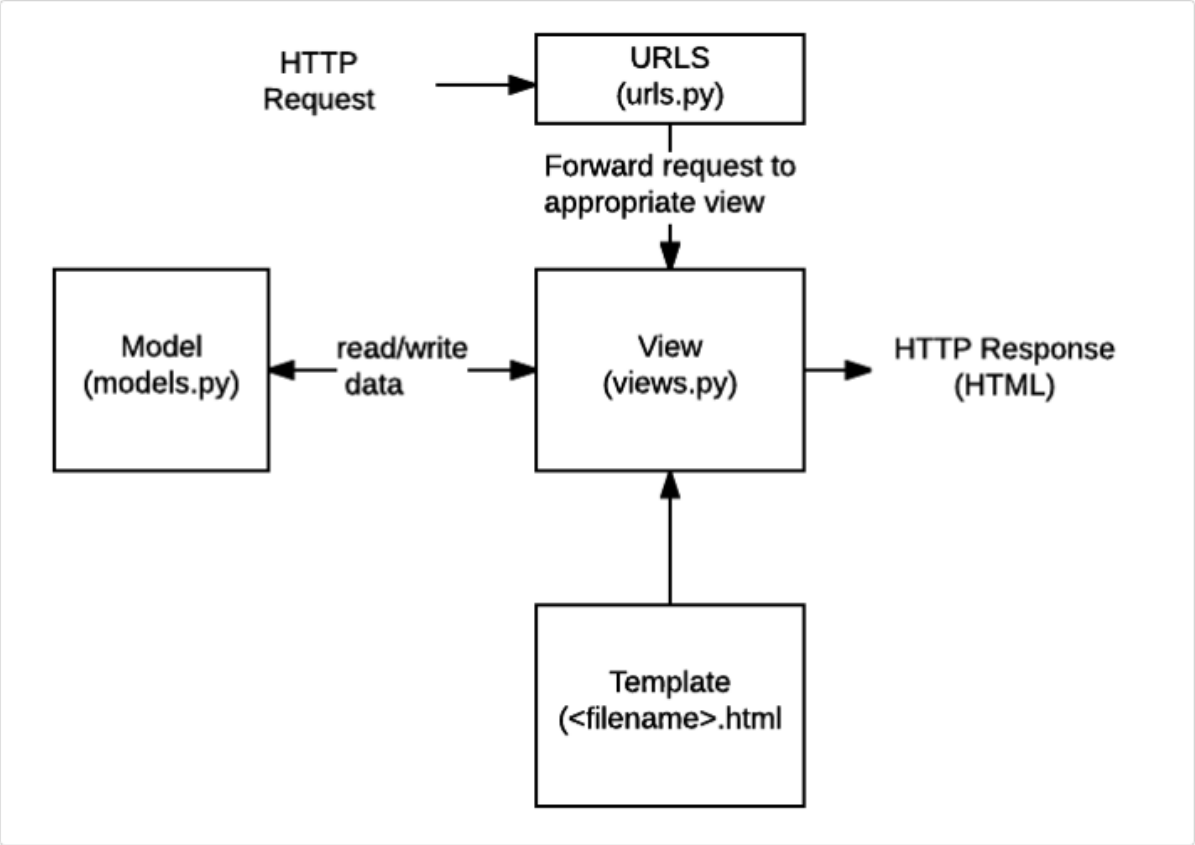
- Συλλογή σκουπιδιών: Μπορεί να χειριστεί την κατανομή και την αποδέσμευση μνήμης. Με άλλα λόγια, κατά την κατασκευή μεταβλητών και αντικειμένων ο προγραμματιστής δε χρειάζεται να ελέγξει τις περιπτώσεις όπου πρέπει να γίνει ανακατανομή ή αποδέσμευση μνήμης και μπορεί να εστιάσει καθαρά στη συγγραφή κώδικα.

3.4 Django

Το Django είναι ένα πλαίσιο συγγραφής κώδικα υψηλού επιπέδου, χτισμένο σε Python όπου επιτρέπει την ταχεία ανάπτυξη εφαρμογών ιστού.Παρέχεται δωρεάν σε οποιοδήποτε επιθυμεί να το χρησιμοποιήσει και υπάρχει από το 2005. Κύριο χαρακτηριστικό είναι ότι εστιάζει να αυτοματοποιεί τα βασικά χαρακτηριστικά μιας εφαρμογής ώστε ο προγραμματιστής να στοχεύει στη συγγραφή κώδικα χωρίς να χρειάζεται να ανακαλύψει ξανά τον τροχό.

Η αρχική ανάπτυξη του ξεκίνησε το 2003 και ολοκληρώθηκε το 2005 από μια ομάδα προγραμματιστών όπου κύριο τους μέλημα ήταν η δημιουργία και η συντήρηση ιστοσελίδων εφημερίδας. Η δημιουργία του προέκυψε τυχαία καθώς η ομάδα δημιουργούσε καινούργιες ιστοσελίδες, ξεκίνησε να επαναχρησιμοποιεί κοινούς κώδικες και πρότυπα σχεδίασης. Αυτός ο κώδικας στη συνέχεια μετατράπηκε σε ένα πρωτόκολλο πλαισίου ανάπτυξης ιστού, του οποίου του δόθηκε το όνομα ”Django” τον Ιούλιο του 2005. Η ανάπτυξη και η βελτίωση του συνέχισε με ορόσημο την κυκλοφορία της πρώτης έκδοσης (1.0), τον Σεπτέμβριο του 2008, μέχρι την έκδοση 4.0 που κυκλοφόρησε το 2022. Κάθε καινούργια έκδοση στοχεύει στη βελτιστοποίηση της αρχικής προσφέροντας καινούργια λειτουργικότητα, διορθώσεις σφαλμάτων, υποστήριξη νέων τύπων βάσεων δεδομένων και προσθήκη κλάσεων και συναρτήσεων με σκοπό τη μείωση του απαιτούμενου κώδικα για την υλοποίηση μιας διαδικασίας.

Η δομή του Django βασίζεται στο μοντέλο MVT (Model-View-Template). Στο παράδειγμα όπου υπάρχει μία κοινότυπη ιστοσελίδα όπου η μεταφορά δεδομένων γίνεται με το πρωτόκολλο HTTP, τα δεδομένα αποθηκεύονται σε μια βάση δεδομένων και η επικοινωνία γίνεται από έναν απλό περιηγητή ιστού. Όταν ληφθεί ένα αίτημα, η εφαρμογή καθορίζει τη λειτουργία βάση της διεύθυνσης URL σε συνδυασμό με τις πληροφορίες που παρέχει το αίτημα όπως και τον τύπο του αιτήματος (π.χ. POST, GET). Έπειτα από την κατάλληλη διεργασία η εφαρμογή επιστρέφει την απάντηση στο αίτημα το οποίο συνήθως δημιουργεί δυναμικά μια σελίδα HTML. Η παραπάνω διαδικασία ”αιτήματος-απάντησης” χειρίζεται από το Django με τη χρήση του MVT. Το αίτημα αρχικά περνάει από το αρχείο urls.py το οποίο περιέχει χαρτογράφο ανακατεύθυνσης URL για το σημείο του αρχείου views.py όπου θα κληθεί. Το views.py χρησιμοποιεί το models.py είτε για να διαβάσει είτε να γράψει δεδομένα και στη συνέχεια προβάλει την πληροφορία στο κατάλληλο αρχείο από τα πρότυπα (Template). Το models.py όπου αναπαριστά τα μοντέλα είναι αντικείμενα Python όπου καθορίζουν τη δομή των δεδομένων και τους μηχανισμούς διαχείρισης τους. Πρότυπα είναι μια συλλογή αρχείων, συνήθως HTML, χρησιμοποιώντας ένα πρότυπο και δεδομένα από το μοντέλο δημιουργείτε η γραφική διεπαφή μια εφαρμογής.



Σχήμα 3.2: Αναπαράσταση δομής του πλαισίου Django για τη διαχείριση HTTP αιτήματος

Ένα από τα βασικά χαρακτηριστικά του Django είναι η ιδέα του ”μπαταρίες συμπεριλαμβάνονται”, που μεταφράζεται ότι περιέχει πολλές ενσωματωμένες λειτουργίες. Πιο συγκεκριμένα, περιλαμβάνει υποστήριξη για τον χειρισμό φορμών, διαθέτει πάνελ διαχείρισης για τροποποίηση περιεχομένου στη βάση καθώς και αυτόματη δρομολόγηση.

Ένα από τα κύρια προτερήματα του Django είναι η εκτεταμένη καταγραφή των λειτουργιών και δυνατοτήτων που παρέχει. Αυτό βοηθάει στην εκμάθηση του συγκεκριμένου μοντέλου καθώς και στην εύρεση λύσης σε σύνηθες προβλήματα. Ένα ακόμα βασικό προτέρημα του Django είναι το ORM (Object Relational Mapper), το οποίο επιτρέπει στους προγραμματιστές κάνοντας χρήση κώδικα να αλληλεπιδρούν και να σχεδιάζουν το μοντέλο και τις συσχετίσεις στη σχεσιακή βάση δεδομένων. Κάτι τέτοιο διευκολύνει τη δημιουργία ερωτημάτων βάσης δεδομένων (queries) καθώς εξαλείφει τον κίνδυνο επιθέσεων injection.

Ένα επιπλέον χαρακτηριστικό του Django είναι ότι προσφέρει ισχυρό σύστημα ασφάλειας.Διαθέτει ένα ενσωματωμένο σύστημα όπου αποτρέπεις τις πιο γνωστές επιθέσεις όπως το cross-site scripting, cross-site request forgery και SQL injections.Ακόμα, διαθέτει έτοιμη τη λειτουργία εγγραφής και σύνδεσης, η οποίο πραγματοποιείται εύκολα ώστε να ανταποκρίνεται στις ανάγκες κάθε εφαρμογής.

Οι δυνατότητες του Django όπως και η επεκτασιμότητα του είναι πολλές, πράγμα που το καθιστά κατάλληλο για διαχείριση υψηλών επιπέδων κίνησης και δεδομένων.Εξάλλου δεν είναι τυχαία η επιλογή του συγκεκριμένου πλαισίου από εταιρείες όπως το Pinterest, το Instagram και οι Washington Times. Αποδεδειγμένα το Django παρέχει αξιοπιστία και αποτελεσματικότητα σε υλοποιήσεις ιστού.

Παρά τα ποικίλα πλεονεκτήματα του, υπάρχουν περιπτώσεις έργων όπου το Django δεν είναι η καταλληλότερη επιλογή.Το Django προσφέρει ένα ολοκληρωμένο πακέτου πλαισίου ιστού που έχει σχεδιαστεί από προγραμματιστές για προγραμματιστές, προσφέροντας τη μεγαλύτερη ευκολία κατά τη μετατροπή της ιδέας στην ολοκλήρωση του έργου. Ως εργαλείο ιστού μπορεί να χειριστεί πολύπλοκες υλοποιήσεις, βέβαια για μικρότερα έργα μπορεί να χαρακτηριστεί ως υπερβολικό. Επίσης, ή χρήση ενός συγκεκριμένου ORM για τη δομή της βάσης και του συστήματος αρχείων του έργου δημιουργεί την επιβολή σχεδιαστικών αποφάσεων στον προγραμματιστή. Για αρχάριους προγραμματιστές αυτή η επιβολή διευκολύνει την ανάπτυξη ιστοσελίδων, βέβαια για προγραμματιστές όπου επιθυμούν περισσότερο έλεγχο στην αρχιτεκτονική δεν την καθιστά ιδανική επιλογή. [13]

3.5 Beautiful Soup

Το Beautiful Soup [7] είναι ένα δυνατό εργαλείο, παρέχεται ως επιπλέων βιβλιοθήκη της Python, για τη διευκόλυνση της διαδικασίας scraping και την ανάλυση εγγράφων HTML και XML. Το κοινό όπου απευθύνεται είναι κυρίως προγραμματιστές και ερευνητές για την εξαγωγή συμπερασμάτων από την ανάλυση δεδομένων. Το εργαλείο επιτυγχάνει τον σκοπό του με τον αναλυτή LXML και μπορεί μα διαχειριστεί πολλούς διαφορετικούς τύπους εγγράφων, με κανονική ή ακόμα και με λανθασμένη σήμανση.

Ο βασικός λόγος προτίμησης της βιβλιοθήκης είναι η δυνατότητα αναζήτησης στη δομή των εγγράφων, όπως για παράδειγμα το δέντρο DOM. Η βιβλιοθήκη έρχεται με ενσωματωμένα εργαλεία όπου καθιστούν την παραπάνω αναζήτηση εύκολη και προσιτή χωρίς να χρειάζεται μεγάλη εμπειρία. Για παράδειγμα, υπάρχουν οι συναρτήσεις find() και find all() όπου δίνουν τη δυνατότητα αναζήτησης στοιχείων με συγκεκριμένο χαρακτηριστικό. Πιο συγκεκριμένα, σε ένα html αρχείο μπορούμε να αναζητήσουμε βάση της κλάσης ενός πεδίου. Επίσης, στην αναζήτηση της υπάρχει η δυνατότητα χρήσης επιλογέων CSS το οποίο προσφέρει πιο πολύπλοκα ερωτήματα με καλύτερες επιδώσεις.

Ένα ακόμα χαρακτηριστικό της βιβλιοθήκης είναι η ικανότητα αναζήτησης και χειρισμού διαφορετικών τύπου αρχείων και δεδομένων. Πιο συγκεκριμένα, η βιβλιοθήκη μπορεί να χειριστεί έγγραφα HTML και XML άλλα και τύπους σήμανσης όπως RSS και Atom feeds. Το παραπάνω χαρακτηριστικό την καθιστά αξιοσημείωτη επιλογή για της περισσότερες περιπτώσεις scraping ιστού.

Πέρα από τις δυνατότητες αναζήτησης και χειρισμού, η βιβλιοθήκη παρέχει μια συλλογή εργαλείων για την επεξεργασία των δεδομένων. Παρέχει τη δυνατότητα τροποποίησης της δομής ενός αρχείο. Πιο συγκεκριμένα, μπορεί να προσθέσει ή να αφαιρέσει ετικέτες όπως για παράδειγμα να τροποποιήσει της κλάσεις από ένα αρχείο HTML. Η παραπάνω δυνατότητα επιτρέπει τη δημιουργία και την αποθήκευση προσαρμοσμένων αρχείων HTML ή XML.

Εκτός από τα τεχνικά χαρακτηριστικά η βιβλιοθήκη διαθέτει σωστά δομημένη καταγραφή των δυνατοτήτων της καθώς και της οδηγίες για τη σωστή χρήση. Επίσης, με την πάροδο των χρόνων έχει αναπτύξει μεγάλη και ενεργή κοινότητα χρηστών. Αυτή η κοινότητά συμβάλει με τη συγγραφή κώδικα για τη βελτιστοποίηση της βιβλιοθήκης και παρέχει υποστήριξη σε τρίτους.

Παρά τα πολλά πλεονεκτήματα του, το Beautiful Soup έρχεται και με μερικούς περιορισμούς. Ένα σημαντικό μειονέκτημα είναι η διαχείριση δυναμικών ιστοσελίδων οι οποίες χρησιμοποιούν JavaScript ώστε να φορτώσουν το περιεχόμενο τους. Το Beautiful Soup δεν μπορεί να εκτελέσει JavaScript και να

φορτώσει το περιεχόμενο, επομένως δεν αποτελεί το κατάλληλο εργαλείο για αυτή την περίπτωση. Για την παραπάνω διαδικασία βιβλιοθήκες όπως η Scrapy και Selenium καθίστανται καταλληλότερες.

Συμπερασματικά, το Beautiful Soup είναι ένα ισχυρό εργαλείο που μπορεί να φανεί χρήσιμο σε αρκετές περιπτώσεις scraping. Η δυνατότητα αναζήτησης και πλοήγησης σε διαφορετικές δομές δέντρων και τύπους δεδομένων σε συνδυασμό με τη φιλικότητα της προς τους νέους χρήστες, το καθιστούν εξαιρετικό εργαλείο για αντίστοιχες εργασίες. Παρά ταύτα, η απουσία χειρισμού δυναμικών σελίδων μπορεί να γίνει αποτρεπτικός παράγοντας και να καθίστανται διαφορετικά εργαλεία καταλληλότερα για scaping.

3.6 Βάσεις δεδομένων και MySql

Ως βάσεις δεδομένων ορίζονται οι οργανωμένες συλλογές δεδομένων όπου δομούνται με προκαθορισμένο τρόπο και αποθηκεύονται σε ψηφιακή μορφή. Το πρόβλημα όπου στοχεύουν να επιλύουν οι βάσεις δεδομένων είναι η αποθήκευση, η διαχείριση και η ανάκτηση δεδομένων από μικρή έως μεγάλη κλίμακα. Η επίτευξή του παραπάνω στόχου γίνεται με την αποθήκευση δεδομένων με δομημένο και προκαθορισμένο τρόπο. Η πληροφορία όπου αποθηκεύεται σε μια βάση μπορεί να είναι κείμενα, κατάλογοι, οικονομικές συναλλαγές ή ακόμα οποιαδήποτε πληροφορία όπου μπορεί να εκφραστεί σε ψηφιακή μορφή. Επίσης, οι εφαρμογές της βάσης καθιστάτε απαραίτητη σε εφαρμογές διαδικτύου όπου υπάρχει η ανάγκη για αποθήκευση δεδομένων. Ανάλογες περιπτώσεις είναι ιστοσελίδες όπου παρέχουν ηλεκτρονικές αγορές, τραπεζικές συναλλαγές και κοινωνική δικτύωση. Κύριο μέλημα πέρα από τη δομημένη αποθήκευση δεδομένων είναι η παροχή εύκολης πρόσβασης και χειραγώγησης σε αυτά, χωρίς να κινδυνεύει η ακεραιότητα και η συνέπεια τους. Ακόμα, σε πλαίσια ερευνάς από τα δεδομένα μιας βάσης μπορούν να προκύψουν στατιστικά στοιχεία όπως γραφήματα με την τιμή προϊόντων για μία χρονική περίοδο ή ακόμα λογικά άλματα όπως για παράδειγμα η πρόβλεψη αύξησης ή μείωσης τιμών για μια συγκεκριμένη χρονική περίοδο.

Οι τύποι των βάσεων ποικίλουν και η χρησιμότητα της καθεμίας διαφέρει ανάλογα με την περίπτωση, για παράδειγμα υπάρχουν οι σχεσιακές βάσεις δεδομένων SQL, οι μη σχεσιακές NoSQL, οι βάσεις όπου χρησιμοποιούν γράφους. Επιπλέων, για τη διαχείριση τους έχουν αναπτυχθεί συστήματα διαχείρισης βάσεων δεδομένων (DBMS) όπως η MySQL, PostgreSQL και Oracle. [14]

Η MySQL είναι ένα από τα πιο γνωστά συστήματα διαχείρισης βάσεων δεδομένων, η δημιουργία, η ανάπτυξη και η συντήρηση προέκυψε εξ' ολοκλήρου από την Oracle Corporation. Κύριο χαρακτηριστικό της διάδοσης και υιοθέτησης από προγραμματιστές είναι πως το λογισμικό διανέμετε ελεύθερα και χαρακτηρίζετε ως ανοιχτού κώδικα. Η MySQL ανήκει στις SQL βάσεις δεδομένων δηλαδή στις βάσεις όπου βασίζονται στη σχεσιακότητα. Αυτό το μοντέλο αποτελείτε από την αποθήκευση δεδομένων σε πίνακες ορισμένους με γραμμές και στήλες. Η δομής της παρέχει την αποτελεσματική αναζήτηση και διαχείριση δεδομένων, ενώ ταυτόχρονα διασφαλίζει την ακεραιότητα τους. Όπως προς αναφέρθηκε η MySQL ανήκει στις SQL (Structured Query Language) βάσεις όπου για την αποθήκευση δεδομένων υπάρχει η δυνατότητα ορισμού σχέσης και σύνδεσης μεταξύ διαφορετικών στηλών.

Η δυνατότητα προσθήκης σχέσεων παρέχει τη δυνατότητα επεκτασιμότητας στην αποθήκευση δεδομένων κατά συνέπεια και στις επιπρόσθετες δυνατότητες μιας εφαρμογής. Αυτό την καθιστά κατάλληλη για τη διαχείριση μικρών ιστοσελίδων έως υπηρεσίες αποθήκευσης μεγάλων όγκων δεδομένων. Επίσης,

η προσαρμοστικότητα της αποτελεί σημείο αναφοράς για τη MySQL καθώς παρέχει τη δυνατότητα δημιουργίας συναρτήσεων, εκτέλεση αποθηκευμένων συναρτήσεων (stored procedures) και ενεργοποιητών (triggers). Σημαντικό χαρακτηριστικό της MySQL είναι η υποστήριξη πολλαπλών τύπων δεδομένων από συνηθισμένους τύπους όπως αλφαριθμητικούς χαρακτήρες, άκαιρους αριθμούς έως αντικείμενα τύπου json.

Οι δυνατότητες της και η συνεχής εξέλιξη έγιναν πόλος έλξης για προγραμματιστές με αποτέλεσμα πλέον η κοινότητα της MySQL να απαρτίζεται από πολλούς χρήστες. Ως αποτέλεσμα να υπάρχουν ποικίλοι τρόποι για την εκμάθηση της, είτε με γραπτό κείμενο είτε με βίντεο, ενώ ταυτόχρονα υπάρχουν διαδικτυακοί ιστότοποι για την αντιμετώπιση προβλημάτων. Πέρα από την παροχή βοήθειας το έργο της κοινότητάς αποτελείται και απτή ανάπτυξη καινούργιων εκδόσεων και δημιουργία πρόσθετων λειτουργιών.

Η MySQL έγινε γνωστή για την απόδοση και την αξιοπιστία όπου παρέχει. Η δομή της την επιτρέπει να διαχειρίζεται μεγάλο όγκο πληροφορίας και έχει την ικανότητα να διαχειριστεί ταυτόχρονα ερωτήματα από χρήστες. Το παραπάνω την καθιστά ιδανική επιλογή για υλοποιήσεις όπου απαιτούν γρήγορη ανάκτηση και διαχείριση δεδομένων, ένα αντίστοιχο παράδειγμα είναι η εξυπηρέτηση συναλλαγών σε πραγματικό χρόνο.

Συμπερασματικά, η MySQL είναι ένα ισχυρό εργαλείο όπου μπορεί να εξυπηρετήσει οποιαδήποτε σκοπό χωρίς κανέναν περιορισμό. Παρέχει ένα ευέλικτο σύστημα όπου μπορεί να προσαρμοστεί σε κάθε κατάσταση. Η επεκτασιμότητα της την καθιστά ιδανική επιλογή για πολλούς προγραμματιστές. Ακόμα, οι επιδώσεις της σε συνδυασμό με την αξιοπιστία κάνουν τη MySQL να είναι από τα πιο ανταγωνιστικά συστήματα διαχείρισης βάσεων δεδομένων. Επιπλέων, η ενεργή κοινότητά, ο μεγάλος όγκος δεδομένων για τα τεχνικά της χαρακτηριστικά σε συνδυασμό ότι είναι ανοιχτού κώδικα την καθιστά κατάλληλη για μεγάλους οργανισμούς. Η MySQL είναι ένα σύστημα διαχείρισης βάσεων δεδομένων όπου παρέχει μεγάλη γκάμα δυνατοτήτων χωρίς κάποιο μειονέκτημα και αυτό την καθιστά εξαιρετική επιλογή για περιπτώσεις όπου υπάρχει η ανάγκη για αποθήκευση και διαχείριση μεγάλων όγκων δεδομένων. [15]

3.7 Git

Το Git είναι ένα κατανεμημένο σύστημα όπου δίνει τη δυνατότητα ελέγχου εκδόσεων σε ένα αρχείο, αναφαίρετε κυρίως σε προγραμματιστές αν και μπορεί να χρησιμοποιηθεί σε οποιαδήποτε περίπτωση υπάρχει ανάγκη για έλεγχο εκδόσεων. Το Git έχει φέρει επανάσταση στον τομέα της πληροφορικής και στα αντίστοιχα επαγγέλματα, καθώς δίνει τη δυνατότητα ταυτόχρονης επεξεργασίες ενός αρχείου καθώς και διαχείριση και συλλογή κώδικα από πολλούς προγραμματιστές πάνω σε ένα έργο. Αναπτύχθηκε το 2005 με δημιουργό τον Linus Torvals ο οποίος μάλιστα είναι γνωστός στον χώρο της πληροφορικής καθώς ανέπτυξε και υλοποίησε το λειτουργικό πρόγραμμα Linux.

Το σημαντικότερο χαρακτηριστικό και λόγος δημιουργίας του git είναι η ικανότητα να καταγράφει πολλαπλές εκδόσεις ενός αρχείου. Αυτό δίνει τη δυνατότητα στους χρήστες του να επιστρέψουν σε παλαιότερες εκδόσεις του αρχείου με την εκτέλεση μιας εντολής. Κάτι τέτοιο μπορεί να φανεί πολύ χρήσιμο σε περιπτώσεις προγραμματισμού όπου εμφανίζονται σφάλματα. Επίσης, μέσα από το git δίνεται η δυνατότητα ταυτόχρονης εργασίας πάνω στο ίδιο αρχείο, χωρίς να υπάρχει μετά η ανάγκη χειροκίνητης

ενσωμάτωσης των αλλαγών. Η παραπάνω δυνατότητα επιτυγχάνεται λόγο της αρχιτεκτονικής του git. Πιο συγκεκριμένα ο κάθε χρήστης έχει κατεβασμένο στον υπολογιστή του ένα αντίγραφο ολόκληρου του έργου, οι οποιαδήποτε αλλαγές που θα επιβληθούν γίνονται τοπικά και στη συνέχεια μπορούν να ανέβουν στο κεντρικό αποθετήριο (repository). Αυτή η δομή αναιρεί την προϋπόθεση ενός κεντρικού διακομιστή, κάνοντας αυτομάτως το Git αξιόπιστο και ανθεκτικό.

Εξίσου σημαντικό χαρακτηριστικό του Git είναι η παροχή διακλαδώσεων (branching) πάνω σε ένα έργο. Πιο συγκεκριμένα ένα έργο υπάρχει η δυνατότητα να υπάρχει μια κεντρική διακλάδωση και οι υπόλοιπες διακλαδώσεις να ανήκουν σε προγραμματιστές όπου αναπτύσσουν το έργο. Αυτό παρέχει την ικανότητα ταυτόχρονης επεξεργασίας του έργου χωρίς να επηρεάζεται η κεντρική διακλάδωση αλλά ούτε και οι υπόλοιπες, όπου μπορεί να είναι σε διαφορετικές εκδόσεις. Κάτι τέτοιο επιτρέπει στους χρήστες την πειραματική ανάπτυξη καινούργιων ιδιοτήτων και λειτουργιών, χωρίς να υπάρχει κίνδυνος για την κύρια διακλάδωση κώδικα. Επίσης, το παραπάνω κάνει πολύ πιο προσιτή τη συνεργασία και τον έλεγχο κώδικα, καθώς η μεταφορά από μία διακλάδωση στην άλλη μπορεί να γίνει με την εκτέλεση μιας εντολής. Τέλος, το ίδιο χαρακτηριστικό μπορεί να φανεί χρήσιμο όταν υπάρχει η ανάγκη διαφορετικών προσεγγίσεων για την επίλυση ενός προβλήματος.

Το Git έρχεται μαζί με μια μεγάλη ποικιλία από εργαλεία όπου διευκολύνουν τη συνεργασία με τους υπόλοιπους χρήστες του ίδιου έργου. Υπάρχει η δυνατότητα εύκολου διαμοιρασμού αλλαγών κώδικα χρησιμοποιώντας μια δυνατότητα όπου ονομάζεται ”pull requests”. Το παραπάνω εργαλείο παρέχει τη δυνατότητα επιθεώρησης, συζήτησης και ελέγχου κώδικα πριν αυτός ενσωματωθεί στην κύρια διακλάδωση (branch). Η παραπάνω ιδιότητα προσφέρει δυνατό ένα κλίμα συνεργασίας κατά την ανάπτυξη λογισμικού ενώ ταυτόχρονα διευκολύνει τον εντοπισμό και τη διόρθωση πιθανών σφαλμάτων πριν συγχωνευτούν στο κυρίως κώδικα.

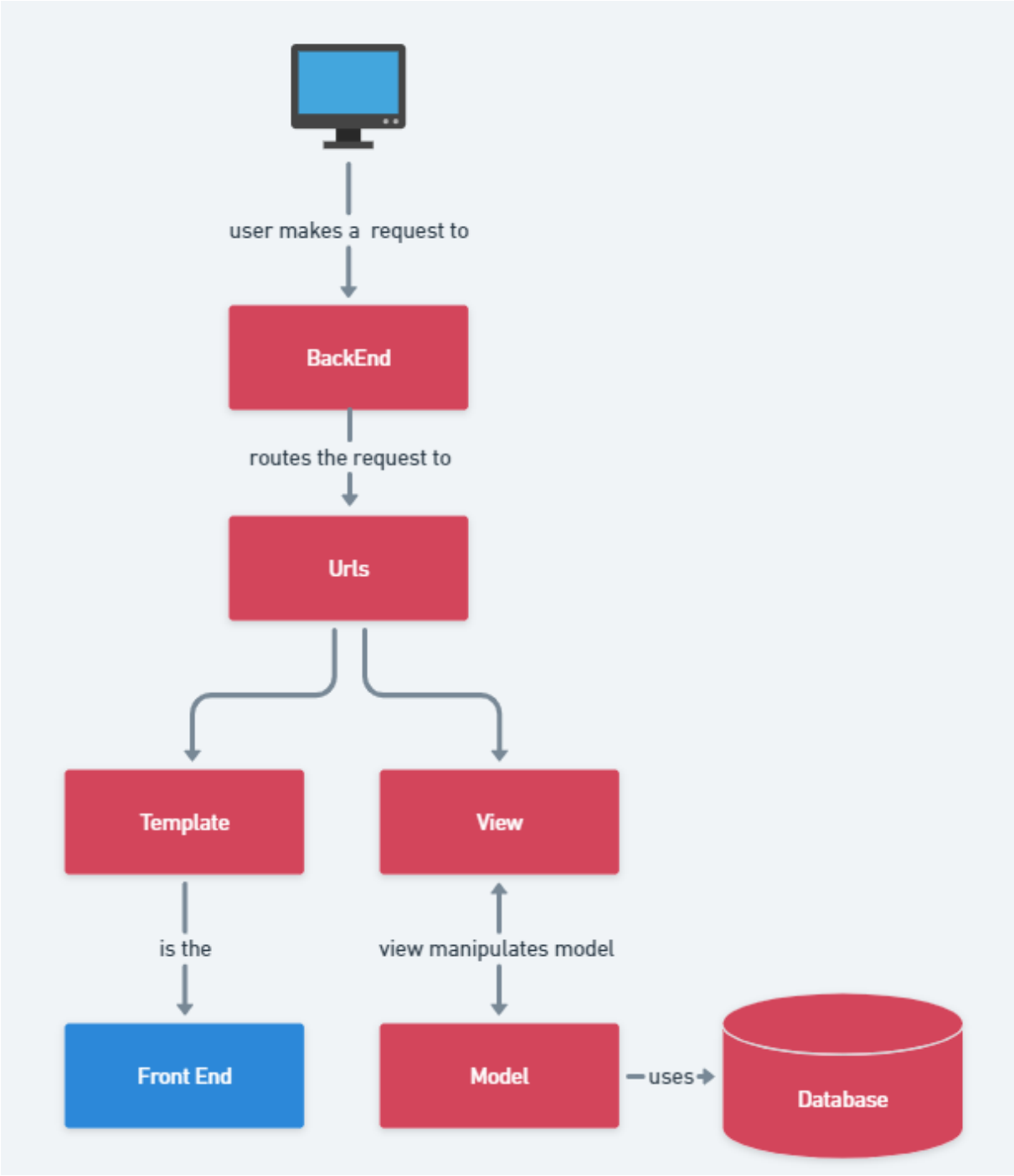
Γύρο από το git έχει δημιουργηθεί ένα οικοσύστημα όπου ενσωματώνει εξωτερικά εργαλεία και πλατφόρμες μερικές από αυτές είναι το Gitlab, το Github και το Bitbucket, τα οποία προσφέρουν μεγαλύτερη ευκολία στις λειτουργίες του και στο διαμοιρασμό κώδικα. Οι παραπάνω πλατφόρμες προσφέρουν επιπλέων χαρακτηριστικά όπως για παράδειγμα ο εντοπισμός προβλημάτων ή προβολή συγκρούσεων κατά την ενσωμάτωση δύο διακλαδώσεων. Επίσης, αναφορικά σε έργα και υλοποιήσεις ανοιχτού κώδικα διευκολύνουν τον διαμοιρασμό και την κοινή χρήση όπως επίσης και την ανάπτυξη καινούργιων δυνατοτήτων σε αυτά.

Από τη χρονιά έναρξης μέχρι και σήμερα το Git έχει σημειώσει σημαντική πρόοδο και έχει εγκαθιδρυθεί ως βασικό εργαλείο για την ανάπτυξη λογισμικού και τη συγγραφή κώδικα. Οι δυνατότητες τους όπως ο έλεγχος εκδόσεων, οι διακλαδώσεις και οι συγχωνεύσεις καθιστούν το Git ένα ανεκτίμητο και αναγκαίο εργαλείο για κάθε επαγγελματία στον χώρο της πληροφορικής. Συμπερασματικά, το Git είναι ένα ισχυρό εργαλείο που έφερε την επανάσταση και αύξησε την παραγωγικότητα στην ανάπτυξη λογισμικού. Οι δυνατότης που αναφέρθηκαν σε συνδυασμό με την ενσωμάτωση του σε πλατφόρμας δείχνουν τη σημαντικότητα και την αναγκαιότητα αυτού του εργαλείου. [16] [17]

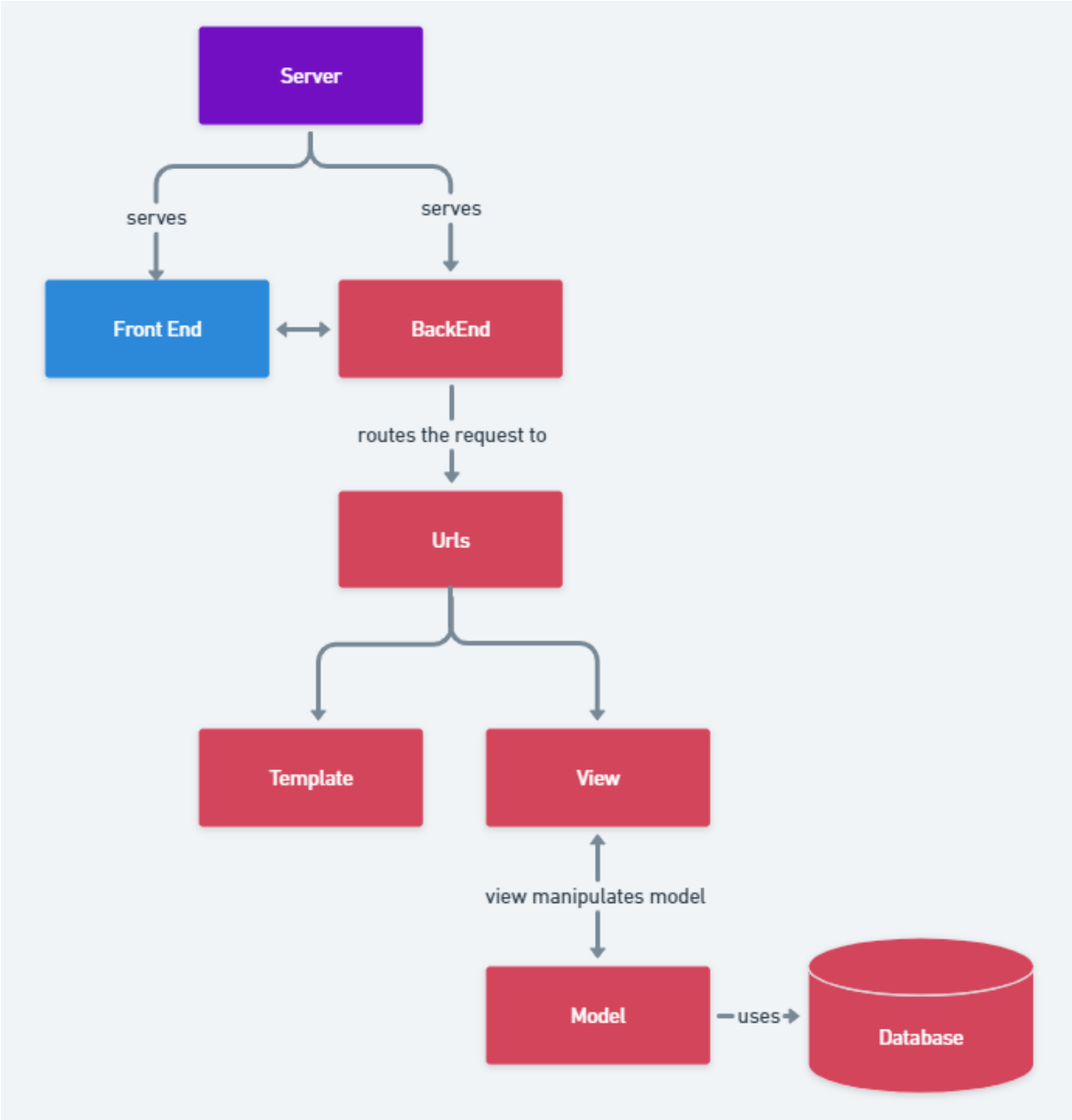
Κεφάλαιο 4ο: Σχεδίαση και Υλοποίηση της εφαρμογής

4.1 Αρχιτεκτονική της εφαρμογής

Η συλλογική παρουσίαση της αρχιτεκτονικής παρουσιάζεται στο σχήμα 4.1 με κόκκινο χρώμα απαρτίζονται τα μέρη του backend επομένως της django ενώ με μπλε αυτά της react. Στο παρακάτω σχήμα παρουσιάζονται όλα τα επιμέρους τμήματα όπου συντελούν στην υλοποίηση της εφαρμογής. Όταν γίνει ένα αίτημα προς την εφαρμογή δηλαδή προς το back-end το πρώτο πράγμα όπου γίνεται είναι η αντιστοιχία του συνδέσμου με την επιμέρους λειτουργία. Ο κόμβος όπου πραγματοποιεί το routing ονομάζεται urls και σύμφωνα με τη μορφή του αιτήματος καθορίζει αν θα πρέπει να προβάλει κάποια γραφική διεπαφή και να ενεργοποιήσει το template, να επεξεργαστεί τα μοντέλα είτε να απορρίψει το αίτημα. Στην παρούσα εργασία όλα τα αίτημα όπου έχουν τη μορφή ”/*” προβάλουν το frontend και αφήνουν τη react να διαχειριστεί εσωτερικά το routing δηλαδή αυτά αποτελούν αιτήματα όπου ενεργοποιείται το template και προβάλλεται γραφική διεπαφή στον χρήστη. Σε αντίθεση με τα αιτήματα της μορφής ”/api/*/” όπου αποτελούν επιμέρους λειτουργίες για την εμφάνιση και διαχείριση της πληροφορίας. Αυτά ενεργοποιούν τα views τα οποία με τη σειρά τους επεξεργάζονται τα μοντέλα και σε συνέχεα τη βάση δεδομένων. Αξίζει να σημειωθεί ότι αναφορικά με το σχήμα 4.2 ότι παρουσιάζεται ένας εναλλακτικός τρόπος αρχιτεκτονικής. Επί της ουσίας υπάρχει η δυνατότητα στην παρούσα εργασία το frontend από το backend να είναι ξεχωριστά καθώς ο τρόπος λειτουργίας τους είναι αυτόνομος. Με αποτέλεσμα να υπάρχει η δυνατότητα αυτά τα τμήματα να τρέχουν σε διαφορετικό port πάνω στο ίδιο μηχάνημα ή ακόμα και σε διαφορετικά μηχανήματα.



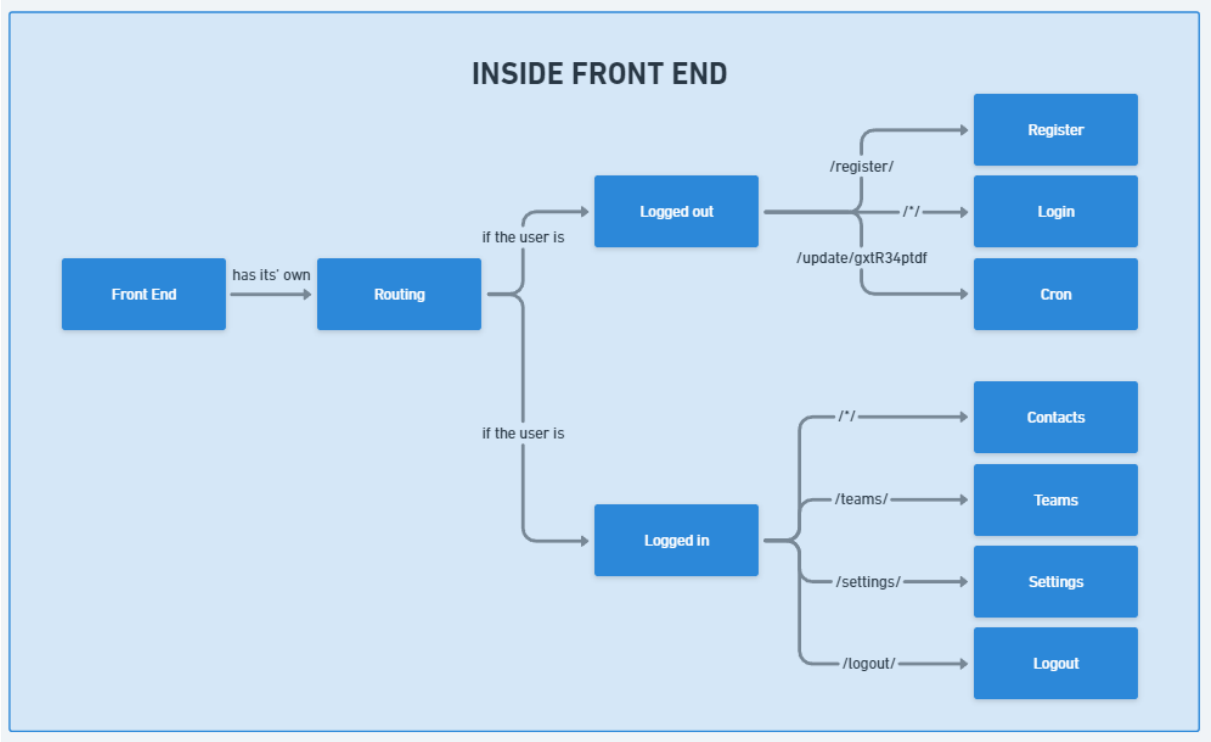
Σχήμα 4.1: Τμηματική αναπαράσταση αρχιτεκτονικής της εφαρμογής.



Σχήμα 4.2: Εναλλακτική αναπαράσταση αρχιτεκτονικής της εφαρμογής.

Στο σχήμα 4.3 παρουσιάζονται τα κύρια τμήματα για τον κόμβο της αρχιτεκτονικής Frontend. Όπως έχει προ αναφερθεί το frontend παρέχεται μέσα από το template της django, λειτουργεί ανεξάρτητα και διαχειρίζεται αυτόνομα τις λειτουργίες του. Στο σχήμα παρουσιάζεται το routing όπου διαχειρίζεται η django ανάλογα με τον τύπο του αιτήματος καθώς και την κατάσταση του χρήστη. Κύριος διαχωρισμός είναι αν ο χρήστης είναι συνδεδεμένος. Η κατάσταση αυτή καθορίζεται από το backend και διαχειρίζεται με session. Στην περίπτωση όπου ο χρήστης είναι αποσυνδεδεμένος μπορεί να έχει πρόσβαση μόνο στις γραφικές διεπαφές του login,register και του cron job. Αξίζει να σημειωθεί πως το cron job αποτελεί κόμβο όπου μόνο όποιος γνωρίζει τον σύνδεσμο μπορεί να έχει πρόσβαση σε αυτό. Οι συνδεδεμένει χρήστες έχουν πρόσβαση στις γραφικές διεπαφές των επαφών, των ομάδων, των ρυθμίσεων καθώς και της αποσύνδεσης. Αξίζει να σημειωθεί πως στην περίπτωση διαδρομής όπου δεν αντιστοιχίζεται με το rounting τότε ανάλογα με την κατάσταση του χρήστη θα προβληθεί η διαδρομή όπου αντιστοιχίζεται

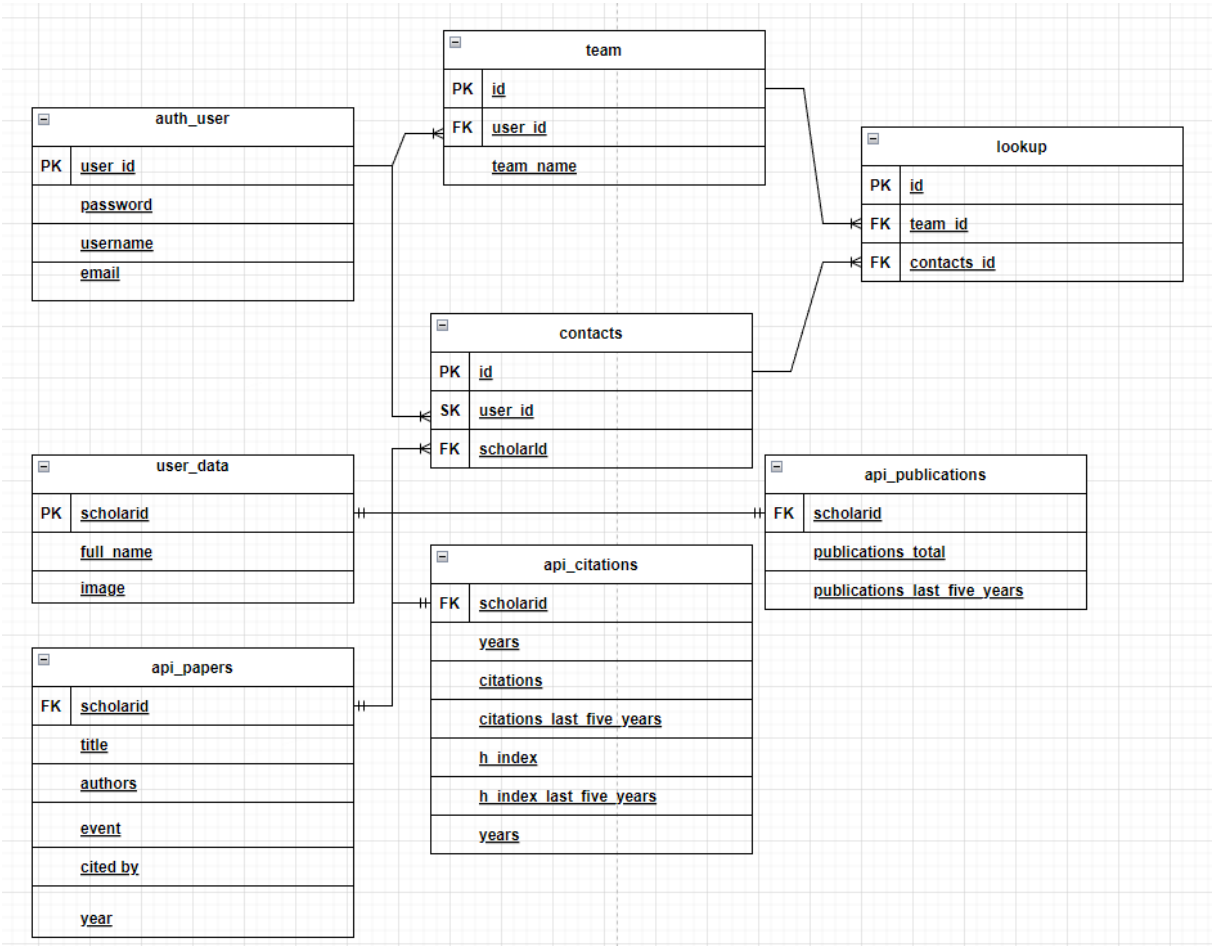
στον σύνδεσμο με μορφή ”/*/”. Δηλαδή εμφανίζεται το Login και το Contact στην εκάστοτε περίπτωση.



Σχήμα 4.3: Τμηματική αναπαράσταση αρχιτεκτονικής της γραφικής διεπαφής της εφαρμογής

Η αρχιτεκτονική της εφαρμογής χωρίζεται σε τρία επιμέρους τμήματα τα οποία αποτελούν διαφορετικό έργο. Αρχικά, για την αποθήκευση των δεδομένων έχει επιλεχθεί σχεσιακή βάση δεδομένων και πιο συγκεκριμένα γίνεται η χρήση της MySQL. Για την επεξεργασία των δεδομένων και τη διαχείριση ερωτημάτων http δηλαδή για το επιμέρους κομμάτι του διακομιστή (backend) έχει επιλεχθεί το Django με τη γλώσσα Python. Για τη γραφική διεπαφή αλληλεπίδρασης του χρήστη (frontend) έχει υλοποιηθεί με τη χρήση JavaScript βιβλιοθήκης επονομαζόμενη React. Αξίζει να σημειωθεί ότι κάθε τμήμα υλοποιεί διαφορετική λειτουργία αν και υπάρχει άμεση εξάρτηση μεταξύ τους ώστε να λειτουργήσει η εφαρμογή.

Η δομή της βάσης έγινε εξολοκλήρου με τη δημιουργία κλάσεων αντικειμένων στο μοντέλο, καθώς το Django παρέχει ORM (Object–relational mapping). Το Django παρέχει την παραπάνω δυνατότητα μόνο σε σχεσιακές βάσεις δεδομένων και υποστηρίζει ποικίλες βάσεις όπως MySQL, PostgreSQL και SQLite. Ο καθορισμός των σχέσεων όπως ένα προς πολλά ή πολλά προς πολλά καθορίζεται αποκλειστικά από τα μοντέλα της Django χωρίς να υπάρχει κανένας περιορισμός στις δυνατότητες που προσφέρουν οι σχετικές βάσεις. Επίσης, οι αλλαγές της βάσης που μπορεί να προκύψουν σε μελλοντικές εκδώσεις καθορίζονται από το Django με τα λεγόμενα migrations και δίνετε η δυνατότητα μεταφοράς από το ένα migration σε άλλο. Τέλος, για τον παραπάνω λόγο οποιαδήποτε αλλαγή στη δομή της βάσης προτείνετε να γίνετε μέσα από το Django και τη συγγραφή κώδικα ώστε να εξασφαλιστεί η σταθερότητα της εφαρμογής.



Σχήμα 4.4: Σχεδίαση δομής και συσχετίσεων βάσης MySQL

Στο σχήμα 4.4 παρουσιάζεται το σχήμα τις βάσης με τους απαραίτητους πίνακες, τα πεδία τους καθώς και τις συσχετίσεις μεταξύ πινάκων. Αρχικά ο πίνακας με όνομα auth_user είναι ο πίνακας των χρηστών όπου περιέχει τα πεδία τους. Συσχετίζεται με σχέση ένα προς πολλά με τους πίνακες team και contacts σε περίπτωση διαγραφής μιας εγγραφής απτό πίνακα διαγράφονται αυτόματα και οι συσχετιζόμενες εγγραφές στους πίνακες. Ο πίνακας contacts περιέχει πρωτεύων κλειδί έναν αύξων αριθμό καθώς και δύο ξένα κλειδιά ένα για τη συσχέτιση του χρήστη και ένα για την συσχέτιση με το user_data. Με αυτό τον τρόπο γίνεται η αποθήκευση των επαφών του χρήστη χωρίς να υπάρχουν πολλαπλές εγγραφές για τα δεδομένα των ερευνητών. Στη συνέχεια, ο πίνακας team περιέχει το όνομα της ομάδας και το αναγνωριστικό του χρήστη ώστε να προκύπτουν οι ομάδες σε ποιον χρήστη ανήκουν. Η σύνδεση μεταξύ ομάδας και επαφών ορίζεται στον πίνακα με όνομα lookup όπου περιέχει το αναγνωριστικό της ομάδας και το αναγνωριστικό της επαφής με συσχέτιση ένα προς πολλά. Τα δεδομένα των ερευνητών αποθηκεύονται σε 4 διαφορετικούς πίνακες οι οποίοι είναι συνδεδεμένοι με συσχέτιση ένα προς ένα με τον κεντρικό πίνακα user_data. Σε περίπτωση διαγραφής μιας εγγραφής απτό τον πίνακα τότε διαγράφονται αυτόματα και τα οι εγγραφές με αυτή τη συσχέτιση.

Η επιμέρους λειτουργία επεξεργασίας και αποθήκευσης δεδομένων όπως και η διαχείριση ερωτημάτων http επονομαζόμενη ως backend έγινε με τη χρήση της γλώσσας Python και του πλαισίου Django. Πιο συγκεκριμένα αυτή η λειτουργία διαχειρίζεται τα δεδομένα της βάσης δεδομένων ανάλωνα με τις αλληλε-

πιδράσεις του χρήστη. Για παράδειγμα, μερικές από τις αλληλεπιδράσεις είναι η σύνδεση-αποσύνδεση, η προσθήκη επαφής και η δημιουργία ομάδας. Ενώ ταυτόχρονα στην περίπτωση όπου ένας χρήστης προσθέσει μια επαφή τότε το backend καθιστάτε υπεύθυνο για τα δεδομένα του ερευνητή σε περίπτωση όπου μεταβληθούν μέσα από το Google Scholar. Επίσης, όπως έχει γίνει αναφορά, το Django βασίζεται στο μοντέλο MVT. Αξίζει να σημειωθεί πως στην υλοποίηση που έχει πραγματοποιηθεί υπάρχει η δυνατότητα το τμήμα της γραφικής διεπαφής είτε να παρέχεται από το Django είτε να παρέχεται αυτόνομα έχοντας το σε διαφορετικό κανάλι (port).

Το κομμάτι της γραφικής διεπαφής (frontend) και αλληλεπίδρασης μεταξύ ανθρώπου και μηχανής υλοποιήθηκε με τη χρήση της βιβλιοθήκης React. Η React καθιστάτε υπεύθυνη για να παρέχει στον χρήστη το γραφικό περιβάλλον όπου τα δεδομένα είναι ορατά στον χρήστη. Επίσης, πέρα από την προβολή δεδομένων η react διαχειρίζεται τα αντίστοιχα ερωτήματα όπου πρέπει να κλιθούν προς τον διακοσμητή ανάλογα με την αλληλεπίδραση του χρήστη. Αξίζει να σημειωθεί πως παρόλο που η εφαρμογή προορίζεται για ερευνητές, το γραφικό περιβάλλον έχει σχεδιαστεί με τρόπο ώστε χρήστες με ελάχιστη γνώση να μπορούν να περιηγηθούν.

4.2 Υλοποίηση backend

```python
from django.urls import path
from . import views

urlpatterns = [
    path('csrf/', views.get_csrf, name='api-csrf'),
    path('login/', views.login_view, name='api-login'),
    path('register/', views.register_view, name='api-register'),
    path('logout/', views.userlogout, name='api-logout'),
    path('session/', views.session_view, name='api-session'),
    path('overview/', views.overview, name='api-overview'),
    path('addContact/', views.addContact, name='api-addContact'),
    path('getContacts/', views.getContacts, name='api-getContacts'),
    path('removeContact/', views.removeContact, name='api-removeContact'),
    path('addTeam/', views.addTeam, name='api-addTeam'),
    path('getTeams/', views.getTeams, name='api-getTeams'),
    path('deleteTeam/', views.deleteTeam, name='api-deleteTeam'),
    path('getGroupContacts/', views.getGroupContacts, name='api-getGroupContacts'),
    path('addToGroup/', views.addToGroup, name='api-addToGroup'),
    path('removeFromGroup/', views.removeFromGroup, name='api-removeFromGroup'),
    path('getGroup/', views.getGroup, name='api-getGroup'),
    path('update/', views.update, name='api-update'),
    path('getPapers/', views.getPapers, name='api-getPapers'),
    path('getTeamPapers/', views.getTeamPapers, name='api-getTeamPapers'),
    path('deleteAccount/', views.deleteAccount, name='api-deleteAccount'),
    # path('images/', views.images, name='api-images'),
]
```

Σχήμα 4.5: Αρχείο urls.py όπου περιέχει μοτίβα URL

Για την επεξεργασία των δεδομένων και τη διαχείριση της βάσης το backend βασίζεται σε δομή API. Πιο συγκεκριμένα το API ορίζει τα επιμέρους κομμάτια κώδικα καθώς και τις λειτουργίες οπού θα ενεργοποιηθούν λαμβάνοντας συγκεκριμένα αιτήματα, σχήμα 4.5. Στην εφαρμογή ο κεντρικός κόμβος όπου κάνει τον παραπάνω έλεγχο και καθορίζει τον εκτελέσιμο κώδικα είναι το αρχείο (urls.py) όπου παρουσιάζεται στην παρακάτω εικόνα. Επί της ουσίας το αρχείο περιέχει μία μεταβλητή με όνομα ”urlpatterns” η οποία είναι μια λίστα αντικειμένων. Το αντικείμενο τύπου ”path” είναι μοτίβο IRL όπου περιέχει τρία πεδία. Στο πρώτο πεδίο υπάρχει η κατάληξη του url από τα http αιτήματα όπου δέχεται, σε περίπτωση όπου γίνει αίτημα με σύνδεσμο όπου δεν υπάρχει στον πίνακα, απευθείας με επιστρέφεται απάντηση με σφάλμα 404. Κάθε διεύθυνση URL σε μία συγκεκριμένη συνάρτηση του μοντέλου ”views”.Οι περισσότερες από τις παραπάνω λειτουργίες προσδιορίζονται από το όνομα τους και προϊδεάζουν τον προγραμματιστή για τη λειτουργία τους. Το τελευταίο πεδίο, το οποίο μπορεί να παραλειφθεί δέχεται ένα όνομα με αλφαριθμητικούς χαρακτήρες, όπου μπορεί να χρησιμοποιηθεί εσωτερικά για αναφορά στο μοτίβο.

```python
def get_csrf(request):
    response = JsonResponse({'detail': 'CSRF cookie set'})
    response['X-CSRFToken'] = get_token(request)
    return response

@require_POST
def register_view(request):
  data = json.loads(request.body)
  status = {'flag' : False ,'message' : ''}
  if User.objects.filter(username = data['username']).exists():
    status['message'] = "Username already exists"
  else:
    try:
      status['flag'] = True
      status['message'] = 'Account created successfully'
      user = User.objects.create_user(username= data['username'], email= data['email'], password = data['password'])
    except KeyError:
       status['message'] = 'Something went wrong, please try different password'
  return JsonResponse(status)


@require_POST
def login_view(request):
    data = json.loads(request.body)
    username = data.get('username')
    password = data.get('password')

    if username is None or password is None:
        return JsonResponse({'response' : False })
    user = authenticate(username=username, password=password)
    if user is None:
        return JsonResponse({'response' : False })
    login(request, user)
    user = User.objects.get(username = data.get('username'))
    request.session['userId'] = user.pk
    request.session['username'] = user.username
    return JsonResponse({'response' : True })


@require_POST
def userlogout(request):
  status = {'logout' : False}
  if session_view(request):
    logout(request)
    status['logout'] = True
  return JsonResponse(status)
```

Σχήμα 4.6: Αρχείο views.py και αναπαράσταση βασικών συναρτήσεων για λειτουργίες εγγραφής, σύνδεσης και αποσύνδεσης

Στο σχήμα 4.6 υπάρχει ένα στιγμιότυπο από το αρχείο views.py. Η πρώτη συνάρτηση με όνομα get csrf επιστρέφει το csrf token. Πιο συγκεκριμένα, όλα τα υπόλοιπα αιτήματα όπου γίνονται προς το API αν δεν περιέχουν στα δεδομένα κεφαλής (header) το συγκεκριμένο token δε γίνονται δεκτά. Δηλαδή το συγκεκριμένο λειτουργεί ως φίλτρο για http requests από τρίτους. Σε αυτό το σημείο αξίζει να σημειωθεί πως πέρα από το παραπάνω το Django παρέχει παραμετροποίηση των CORS ώστε να αποκλείει αιτήματα από ip διευθύνσεις όπου δε γνωρίζει. Επίσης, τα αιτήματα προς τον διακοσμητή γίνονται με τη μέθοδο POST και η επικοινωνία μεταξύ ερώτησης και απάντησης γίνεται με την κωδικοποίηση json. Στη συνέχεια, υπάρχει η μέθοδος register view η συγκεκριμένη μέθοδος λαμβάνει τρία πεδία ένα όνομα χρήστη, τον κωδικό όπου πρέπει να είναι τουλάχιστον 8 χαρακτήρες με ένα κεφαλαίο, ένα μικρό, ένα νούμερο και έναν ειδικό και ένα ηλεκτρονικό ταχυδρομείο. Σε περίπτωση όπου δεν υπάρχει το όνομα χρήστη και ο κωδικός πληροί τα κριτήρια, το API επιστρέφει τη μεταβλητή flag ίση με True και στη message το μήνυμα επιτυχίας. Αν κάτι πάει προκύψει και δεν μπορεί να δημιουργηθεί ο χρήστης τότε επιστρέφει False και το ανάλογο μήνυμα αντίστοιχα. Στην τρίτη συνάρτηση υπάρχει η σύνδεση του χρήστη όπου σε περίπτωση που σταλούν σωστά το όνομα χρήστη και ο κωδικός η σύνδεση γίνεται με επιτυχία. Στην παραπάνω συνάρτηση αποθηκεύουμε στην περίοδο όπου ο χρήστης είναι συνδεδεμένος το id του και το όνομα χρήστη (username). Στην τελευταία συνάρτηση γίνεται έλεγχος αν υπάρχει ανοιχτό session και στη συνέχει διαγράφεται μαζί με τις μεταβλητές όπου αποθηκεύσαμε στη σύνδεση.

```python
from django.db import models
from django.contrib.auth.models import User
from .UserData import UserData
from .Teams import Teams


class Contacts(models.Model):
    userId = models.ForeignKey(User, on_delete=models.CASCADE)
    scholarId = models.ForeignKey(UserData, on_delete=models.CASCADE)


    def getFullName( scholarId):
      contactName = UserData.objects.filter(scholarId=scholarId).values('fullName')
      return contactName[0]['fullName']

    def setScholarId(user, scholarId):

      data = {'flag' : True, 'message' : ''}
      if Contacts.objects.filter(userId = user).filter(scholarId=scholarId):
        data['message']= f'Contact with {Contacts.getFullName(scholarId)} already exists'

      else:
        userData = UserData.checkScholarId(scholarId)
        if userData:
          contact = Contacts(userId = user,scholarId=userData)
          contact.save()
          data['message']= f'Contact {Contacts.getFullName(scholarId)} added successfully'
        else:
           data['message']='Wrong scholarId'
      return data

```

Σχήμα 4.7: Μοντέλο Contacts και οι μέθοδοι διαχείρισης επαφών.

Η κλάση Contacts αποτελεί τη βασική κλάση για τον χειρισμό των επαφών και αναπαριστά τον πίνακα contacts στο σχήμα 4.4. Περιέχει δύο πεδία όπου δημιουργούν τη διασύνδεση μεταξύ ενός χρήστη και ενός ερευνητή όπου υπάρχει στον πίνακα ”user data”. Με τη χρήση της Django στις γραμμές 8 και 9 4.7 ορίζονται τα παραπάνω πεδία καθώς επίσης ορίζεται η λειτουργία σε περίπτωση διαγραφής. Πιο συγκεκριμένα, είτε γίνει διαγραφή του λογαριασμού του χρήστη είτε γίνει διαγραφή του ερευνητή από τη βάση δεδομένων. Τότε αυτομάτως διαγράφετε η εγγραφή από τον πίνακα contacts.

Σε περίπτωση όπου ο χρήστης επιθυμεί να προσθέσει μια επαφή, υπάρχουν πιθανές περιπτώσεις εισαγωγής όπου πρέπει να συμπεριληφθούν. Αναλυτικότερα, υπάρχουν τρία σενάρια εισαγωγής επαφής. Αρχικά, υπάρχει το σενάριο καταχώρησης επαφής όπου υπάρχει ήδη. Δεύτερον υπάρχει η περίπτωση εισαγωγής λάθος scholarId. Επίσης, υπάρχει η περίπτωση εισαγωγής σωστού scholarId, το οποίο δεν αποτελεί υπάρχον επαφή του χρήστη. Στο αναφερόμενο σενάριο υπάρχουν οι υποπεριπτώσεις ο ερευνητής να υπάρχει στη βάση δεδομένων ή να μην υπάρχει και να πρέπει να προστεθεί.

Για να καλυφθούν όλες οι πιθανές περιπτώσεις εισαγωγής επαφής, η διαχείριση γίνεται από τη μέθοδο ”setScholarId”. Αρχικά, δέχεται δύο παραμέτρους σε δύο μεταβλητές, τον χρήστη όπου επιθυμεί να προσθέσει επαφή και τον μοναδικό scholarId του ερευνητή. Στη μέθοδο, ο πρώτος έλεγχος που γίνεται είναι αν το συγκεκριμένο αναγνωριστικό υπάρχει ήδη στις επαφές του χρήστη. Αν δεν υπάρχει, γίνεται κλήση στη μέθοδο checkScholarId του μοντέλου UserData για την επαλήθευση του αναγνωριστικού. Αν το κλειδί του ερευνητή βρεθεί και υπάρχουν τα δεδομένα στη βάση τότε γίνεται καταχώριση της επαφής και δημιουργείτε μια καινούργια εγγραφή στον πίνακα των επαφών. Ειδάλλως, το αναγνωριστικό δεν αντιστοιχίζεται και δε γίνεται ολοκλήρωση της καταχώρησης. Σε οποιαδήποτε περίπτωση η μέθοδος επιστρέφει αν έγινε η καταχώρηση καθώς και ένα μήνυμα ώστε να ενημερωθεί ο χρήστης.

```python
class UserData(models.Model):
  scholarId = models.CharField(primary_key=True,max_length = 30)
  fullName= models.CharField(max_length = 100, blank=True)
  image = models.CharField(max_length = 200, blank=True)
  # Elegxw an uparxei to scholarId,an uparxei epistrefw True sto api

# An den uparxei kai  to scholarId einai valid dhmiourgw thn eggrafh st
on pinaka userData, Citations,Publications kai epistrefw True sto api
  # An den einai valid epistrefw false
  def checkScholarId(scholarId):

    try:
      if UserData.objects.filter( scholarId=scholarId ).exists():
        return UserData.objects.filter(scholarId=scholarId).get()
      else:
        id = scholarId
        url = 'https://scholar.google.gr/citations?hl=el&user='+id+
'#d=gsc_md_hist'

        ua=UserAgent()
        hdr = {'User-Agent': ua.random,
            'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
            'Accept-Charset': 'ISO-8859-1,utf-8;q=0.7,*;q=0.3',
            'Accept-Encoding': 'none',
            'Accept-Language': 'en-US,en;q=0.8',
            'Connection': 'keep-alive'}

        req = Request(url, headers=hdr)
        uClient = urlopen(req)
        page_html = uClient.read()
        uClient.close()
        page_soup = soup(page_html, "html.parser")
        containers = page_soup.find("div",{"id":"gsc_prf_pua"})
        img = containers.find("img")

        if re.match('/citations/.+', img['src']):
          imgPath = 'https://scholar.google.com' + img['src']
        else:
          imgPath = img['src']

        containers = page_soup.find("div",{"id":"gsc_prf_in"})
        userDataObj = UserData(scholarId=scholarId,fullName =containers
.text,image= imgPath)
        userDataObj.save()
        Publications.setPublications(userDataObj, scholarId)
        Citations.setCitations(userDataObj, scholarId)
        return userDataObj
    except:
      return False


  def update(scholarId):
    url = 'https://scholar.google.gr/citations?hl=el&user='+scholarId+
'#d=gsc_md_hist'
    ua=UserAgent()
    hdr = {'User-Agent': ua.random,
            'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
            'Accept-Charset': 'ISO-8859-1,utf-8;q=0.7,*;q=0.3',
            'Accept-Encoding': 'none',
            'Accept-Language': 'en-US,en;q=0.8',
            'Connection': 'keep-alive'}

    req = Request(url, headers=hdr)
    uClient = urlopen(req)
    page_html = uClient.read()
    uClient.close()
    page_soup = soup(page_html, "html.parser")
    containers = page_soup.find("div",{"id":"gsc_prf_pua"})
    img = containers.find("img")
    containers = page_soup.find("div",{"id":"gsc_prf_in"})
    object = UserData.objects.filter(scholarId = scholarId)
    object.update(scholarId=scholarId,fullName =containers.text,image=
img['src'])
    Publications.setPublications(object, scholarId)
    Citations.setCitations(object, scholarId)
```

Σχήμα 4.8: Μοντέλο UserData και λειτουργίες διαχείρισης δεδομένων ερευνητών

Το μοντέλο UserData απαρτίζεται από τρία πεδία το scholarId του ερευνητή όπου είναι και πρωτεύων κλειδί, το ονοματεπώνυμο και τον σύνδεσμο για τη φωτογραφία. Από μοντέλο Contacts κατά την καταχώρηση μιας επαφής γίνεται κλήση στη μέθοδο checkScholarId. Η συγκεκριμένη μέθοδος δέχεται το αναγνωριστικό του ερευνητή. Στην αρχή κάνει αναζήτηση αν υπάρχουν αποθηκευμένες στη βάση οι πληροφορίες του ερευνητή, και σε περίπτωση όπου υπάρχει τις επιστρέφει. Σε περίπτωση όπου δεν υπάρχουν αποθηκευμένες γίνεται http αίτημα στην πλατφόρμα του Google Scholar με το αναγνωριστικό του. Στη συνέχεια, γίνεται αναζήτηση για τον σύνδεσμο της φωτογραφίας καθώς και για το πλήρες όνομα του ερευνητή. Στην παραπάνω ενέργεια υπάρχουν δύο πιθανά σενάρια. Το πρώτο είναι το scholarId να αντιστοιχιστεί με τον ερευνητή και στη συνέχεια αποθηκεύεται στη βάση και γίνεται κλήση στα μοντέλα Publications, Citations ώστε να αποθηκευτούν τα στατιστικά του. Στη δεύτερη περίπτωση το scholarId δεν είναι έγκυρο και δεν αντιστοιχίζεται με κάποιον ερευνητή. Τότε κατά την αναζήτηση της φωτογραφίας δημιουργείτε εξαίρεση την οποία τη διαχειρίζεται η μέθοδος και επιστρέφει False.

Το μοντέλο UserData περιέχει τη μέθοδο update η οποία είναι υπεύθυνη για τη λειτουργία του cron job ώστε να ενημερώνονται τα πεδία των ερευνητών. Ο τρόπος λειτουργίας της είναι παρόμοιος με αυτόν της μεθόδου checkScholarId, με διαφορά ότι δεν υπάρχει το σενάριο λανθασμένου scholarId. Για την αποθήκευση της φωτογραφίας γίνεται αναζήτηση σε ένα html στοιχείο div με αναγνωριστικό id ”gsc prf in”. Στη συνέχεια γίνεται δεύτερη αναζήτηση στο στοιχείο για να βρεθεί το img tag. Αφού βρεθεί, ο σύνδεσμος της φωτογραφίας βρίσκεται στο scr του στοιχείου. Επίσης, για την εύρεση του ονόματος, του ερευνητή, γίνεται αναζήτηση στο div με αναγνωριστικό id ”gsc prf in”. Η επιθυμητή πληροφορία βρίσκεται στο κείμενο του. Κατά την εύρεση των παραπάνω στοιχείων, γίνεται ενημέρωση της βάσης δεδομένων καθώς και γίνεται κλήση ώστε να ενημερωθούν τα στατιστικά στοιχεία του.

```python
class Publications(models.Model):
  scholarId = models.OneToOneField(to='api.UserData', on_delete=models.CASCADE)
  publications_total = models.SmallIntegerField()
  publications_last_five_years = models.SmallIntegerField()
  years = models.JSONField()

  def setPublications(userDataObj, scholarId):
    # data = json.loads(request.body)
    id = scholarId
    todays_date = date.today()
    curyear=todays_date.year
    pubs = {}
    result = {'years':  {}}

    for y in range(curyear-40, curyear+1):
      pubs[y]=0

    total=0
    total2=0
    for x in range(0, 2000, 100):
      url = 'https://scholar.google.gr/citations?hl=el&user='+id+'&cstart='+str(x)+'&pagesize=100'

      ua=UserAgent()
      hdr = {'User-Agent': ua.random,
          'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
          'Accept-Charset': 'ISO-8859-1,utf-8;q=0.7,*;q=0.3',
          'Accept-Encoding': 'none',
          'Accept-Language': 'en-US,en;q=0.8',
          'Connection': 'keep-alive'}

      req = Request(url, headers=hdr)
      uClient = urlopen(req)
      page_html = uClient.read()
      uClient.close()

      #...
      #Data extraction inside for loop

      if(Publications.objects.filter(scholarId = scholarId).exists()):
      object = Publications.objects.filter(scholarId = scholarId)
      object.update(publications_total = total, publications_last_five_years = total5,years = json_year)
    else:
      object = Publications(scholarId = userDataObj, publications_total = total,
publications_last_five_years = total5,years = json_year)
      object.save()
```

Σχήμα 4.9: Μοντέλο Publications και λειτουργίες διαχείρισης δεδομένων ερευνητών

Για την καταχώρηση των στατιστικών δεδομένων των δημοσιεύσεων, δημιουργήθηκε το μοντέλο Publications. Περιέχει το πεδίο scholarId με συσχέτιση ένα προς ένα στο πρωτεύων κλειδί του μοντέλου UserData σχήμα 4.8, ώστε να διατηρείται η μοναδικότητα. Πιο συγκεκριμένα, να αποφευχθεί η περίπτωση διπλών εγγραφών στη βάση με το ίδιο αναγνωριστικό. Επίσης, κατά τη διαγραφή ενός στοιχείου του μοντέλου UserData, γίνεται αυτόματη διαγραφή του εξαρτημένου στοιχείου στο μοντέλο των δημοσιεύσεων. Επιπρόσθετα, υπάρχει το πεδίο publications total και publications last five years όπου είναι άκαιροι αριθμού και περιέχουν τις συνολικές δημοσιεύσεις και τις συνολικές δημοσιεύσεις των τελευταίων πέντε ετών του ερευνητή. Το τελευταίο πεδίο με όνομα years είναι τύπου json και περιέχει ένα σύνολο συσχετίσεων κλειδιού-τιμής, έχοντας ως κλειδί το έτος και τιμή τον αριθμό των δημοσιεύσεων για τη συγκεκριμένη χρονιά.

Το μοντέλο Publications περιέχει μόνο μία μέθοδο η οποία επαναχρησιμοποιείται τόσο για την αποθή-

κευση ενός στοιχείου όσο και για την ενημέρωση αυτού. Η συγκεκριμένη μέθοδος δέχεται το αντικείμενο UserData όπου πρόκειται να συσχετιστεί καθώς και το scholarId του ερευνητή. Στη συνέχεια, προετοιμάζει αρχικοποιεί τις μεταβλητές όπου θα αποθηκευτούν τα δεδομένα του ερευνητή. Ύστερα ξεκινάει μια επαναληπτική διαδικασία ώστε να πραγματοποιηθούν αιτήματα στην πλατφόρμα Google Scholar και να φορτωθούν οι συνολικές δημοσιεύσεις. Σε κάθε επανάληψη υπολογίζονται οι δημοσιεύσεις και αποθηκεύονται αθροιστικά. Αφού ολοκληρωθεί το scraping των δεδομένων, γίνεται έλεγχος για την ύπαρξη ή μη του συγκεκριμένου αναγνωριστικού στη βάση δεδομένων. Σε περίπτωση όπου δεν υπάρχει τότε γίνεται καταχώρηση καινούργιας εγγραφής, ειδάλλως ενημερώνεται η εγγραφή όπου βρέθηκε πως υπάρχει.

```python
class Citations(models.Model):
    scholarId = models.OneToOneField(to='api.UserData', on_delete=models.CASCADE)
    years = models.JSONField()
    citations = models.SmallIntegerField()
    citations_last_five_years = models.SmallIntegerField()
    h_index = models.SmallIntegerField()
    h_index_last_five_years = models.SmallIntegerField()

    def setCitations(userDataObj, scholarId):
        result = {'years':  {}, 'citations': 0, 'citations_last_five_years': 0,
                  'h_index': 0, 'h_index_last_five_years': 0}
        id = scholarId
        url = 'https://scholar.google.gr/citations?hl=el&user='+id+'#d=gsc_md_hist'

        ua=UserAgent()
        hdr = {'User-Agent': ua.random,
            'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
            'Accept-Charset': 'ISO-8859-1,utf-8;q=0.7,*;q=0.3',
            'Accept-Encoding': 'none',
            'Accept-Language': 'en-US,en;q=0.8',
            'Connection': 'keep-alive'}

        req = Request(url, headers=hdr)
        uClient = urlopen(req)
        page_html = uClient.read()
        uClient.close()
        page_soup = soup(page_html, "html.parser")
        containers = str(page_soup)
        m = re.search('<div class="gsc_md_hist_b">(.+?)</div>',containers,re.DOTALL)

        #...
        #Data extraction from HTML into object result

        if (Citations.objects.filter(scholarId=scholarId).exists()):
            object = Citations.objects.filter(scholarId=scholarId)
            object.update(
                        citations = result['citations'],
                        citations_last_five_years=result['citations_last_five_years'],
                        years=json_year,
                        h_index=result['h_indexs'],
                        h_index_last_five_years=result['h_index_last_five_years']
                        )
        else:
            object = Citations(
                scholarId=userDataObj,
                citations = result['citations'],
                citations_last_five_years=result['citations_last_five_years'],
                years=json_year,
                h_index=result['h_indexs'],
                h_index_last_five_years=result['h_index_last_five_years'])
            object.save()
```

Σχήμα 4.10: Μοντέλο Citations και λειτουργίες διαχείρισης δεδομένων ερευνητών

Για την καταχώρηση των στατιστικών δεδομένων των αναφορών, δημιουργήθηκε το μοντέλο Citations σχήμα 4.10 . Περιέχει το πεδίο scholarId με συσχέτιση ένα προς ένα στο πρωτεύων κλειδί του μοντέλου UserData, ώστε να διατηρείται η μοναδικότητα. Επίσης, κατά τη διαγραφή ενός στοιχείου του μοντέλου

UserData, γίνεται αυτόματη διαγραφή του εξαρτημένου στοιχείου στο μοντέλο των αναφορών. Επιπρόσθετα, υπάρχουν κατά σειρά τα πεδία για την αποθήκευση των αναφορών ανά έτος σε μορφή json. Τα πεδία citations και citations last five years όπου σε ακέραια μορφή έχουν τον συνολικό αριθμώ των αναφορών και των αριθμό των τελευταίων πέντε ετών αντίστοιχα. Ακόμα, υπάρχουν τα πεδία h index και h index last five years όπου σε ακέραια μορφή έχουν τον δείκτη h και των δείκτη h των τελευταίων πέντε ετών.

Το μοντέλο Citations περιέχει μόνο μία μέθοδο η οποία επαναχρησιμοποιείται τόσο για την αποθήκευση καινούργιας εγγραφής στη βάση όσο για την ενημέρωση της. Η συγκεκριμένη μέθοδος δέχεται το αντικείμενο UserData όπου πρόκειται να συσχετιστεί και το scholarId του ερευνητή. Στη συνέχεια, αρχικοποιεί τις μεταβλητές όπου θα αποθηκευτούν τα δεδομένα του ερευνητή. Στη συνέχεια πραγματοποιείτε ένα http αίτημα στην πλατφόρμα Google Scholar με το αναγνωριστικό του ερευνητή και στη συνέχεια γίνεται η εξαγωγή των δεδομένων. Αναλυτικότερα, το παραπάνω αίτημα γίνεται κατά την πλοήγηση του χρήστη στην ιστοσελίδα μέσα από έναν περιηγητή ιστού και εμφανίζει στον χρήστη τις αναφορές του ερευνητή. Αφού ολοκληρωθεί το scraping των δεδομένων, γίνεται έλεγχος για την ύπαρξη ή μη του συγκεκριμένου αναγνωριστικού στη βάση δεδομένων. Σε περίπτωση όπου δεν υπάρχει τότε γίνεται καταχώρηση καινούργιας εγγραφής, ειδάλλως ενημερώνεται η εγγραφή όπου βρέθηκε πως υπάρχει.

4.3 Υλοποίηση frontend

```jsx
function App() {
  const ctx = useContext(UserProvider);

  return (
    <BrowserRouter>
      <ToastContainer
          position="top-right"
          autoClose={3000}
          hideProgressBar={false}
          newestOnTop={false}
          closeOnClick
          rtl={false}
          pauseOnFocusLoss
          draggable
          pauseOnHover
          theme="dark"
        />
      <Routes>
        <Route path="/update/gxtR34ptdf" element={!ctx.isLoggedIn && <Cron />} />
        <Route path="/*" element={!ctx.isLoggedIn &&  <Login></Login>  } />
        <Route path="/register" element={<Register />} />
      </Routes>

      {ctx.isLoggedIn &&
      <main className="d-flex flex-row" style={{'background' : "#151929" ,'minHeight' : '100vh'}  }>
        <Menu></Menu>
        <Routes>
          <Route path="/*" element={ctx.isLoggedIn &&  <Contacts></Contacts> } />
          <Route path="/teams" element={ctx.isLoggedIn &&  <Teams></Teams> } />
          <Route path="/settings" element={ctx.isLoggedIn &&  <Settings></Settings> } />
          <Route path="/logout" element={ctx.isLoggedIn && <Logout></Logout>} />
        </Routes>
      </main>
      }

    </BrowserRouter>
  );
}

export default App;
```

Σχήμα 4.11: Μοντέλο διαχείρισης συνδέσμων και χειρισμού διεπαφών.

Το αρχείο του σχήματος 4.11 ονομάζεται App.js, η συνάρτηση App() είναι το κύριο σημείο εισόδου της εφαρμογής και περιλαμβάνει το κεντρικό σκελετό της διαδικτυακής σελίδας. Η εφαρμογή αυτή δημιουργεί ένα δυναμικό μενού πλοήγησης για τον χρήστη. Η εφαρμογή χρησιμοποιεί το BrowserRouter και το Routes από τη βιβλιοθήκη react-router-dom για τη δρομολόγηση των σελίδων. Επιπλέον, χρησιμοποιεί πολλά στοιχεία της γλώσσας JSX για την επιστροφή των απαραίτητων στοιχείων HTML και τον συντονισμό των δεδομένων και της λογικής της εφαρμογής.

Με τη χρήση της βιβλιοθήκης React Router καθορίζεται σε αυτό το αρχείο η διαχείριση διεπαφών. Η εφαρμογή χρησιμοποιεί το Context API της React για να διαχειριστεί τα στοιχεία του χρήστη και επίσης χρησιμοποιεί τη βιβλιοθήκη react-toastify για να εμφανίσει μηνύματα ειδοποιήσεων στο χρήστη. Ανάλογα με το αν ο χρήστης έχει συνδεθεί ή όχι, η εφαρμογή εμφανίζει διαφορετικά στοιχεία στην οθόνη.

Γενικά, η εφαρμογή αποτελείται από τρεις διαθέσιμες σελίδες: Σελίδα σύνδεσης (Login), Σελίδα εγγραφής (Register) και Σελίδα επαφών (Contacts) καθώς και άλλες διαθέσιμες σελίδες (Teams, Settings, Logout) που εμφανίζονται μόνο εάν ο χρήστης είναι συνδεδεμένος. Επίσης, υπάρχει μια σελίδα Cron που εμφανίζεται μόνο αν ο χρήστης δεν είναι συνδεδεμένος και εισάγει τον ακριβή σύνδεσμο.

```
export const UserProvider = (props) => {
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [csrf, setCsrf] = useState('')
  const [httpRequest, setHttpRequest] = useState('http://localhost:8000/')

  const getCSRF = () => {
    //gets CSRF token from django and sets the cookie
  };

  useEffect(() => {
    getCSRF();
    getSession();
  },[])

  const getSession = () => {
    //gets the Session in order to find if the user is still logged in
  }

  const logoutHandler = () => {setIsLoggedIn(false)}
  const loginHandler = () => {setIsLoggedIn(true)}

  return(
    <UserContext.Provider value={{csrf: csrf ,isLoggedIn: isLoggedIn, httpRequest : httpRequest, onLogout:
logoutHandler, onLogin: loginHandler, getCSRF : getCSRF, getSession  : getSession }}>
      {props.children}
    </UserContext.Provider>
  )
}

export default UserContext
```

Σχήμα 4.12: Κλάση React Context με όνομα UserContext για την παροχή των περιεχομένων της σε κάθε του παιδί.

Στο σχήμα 4.12 παρουσιάζεται το κομμάτι κώδικα όπου αποτελεί την υλοποίηση ενός React Context, το πεδίο ορατότητας όπου έχουν τα δεδομένα καθώς και οι συναρτήσεις είναι γενικό και υπάρχει πρόσβαση σε αυτό από όλα τα υποστοιχεία του. Στη συγκεκριμένη περίπτωση, το context περιέχει το αρχείο App.js με αποτέλεσμα τα δεδομένα του να έχουν ορατότητα από όλες τις συναρτήσεις της εφαρμογής.

Η συνάρτηση UserProvider είναι ένας πάροχος που χρησιμοποιείται για να ενημερώσει τις συνιστώσες που βρίσκονται κάτω από αυτό με τα δεδομένα του context και τις μεθόδους. Το αντικείμενο UserContext παρέχει μια ενιαία δομή δεδομένων και μεθόδων για τη διαχείριση της κατάστασης του χρήστη, ενώ το export default UserContext χρησιμοποιείται για να εξάγεται το αντικείμενο Context και να χρησιμοποιηθεί από άλλες συνιστώσες που χρειάζονται πρόσβαση στις μεταβλητές του Context ή στις μεθόδους του.

Ορίζεται ένα προεπιλεγμένο αντικείμενο τύπου UserContext με τις ιδιότητες csrf, isLoggedIn, onLogout,

onLogin, getCSRF, getSession, και httpRequest.Κατά σειρά στην αρχή υπάρχει το csrf token όπου χρησιμοποιείται στα header από κάθε http αίτημα ώστε να επεξεργαστεί από τη διακοσμητή. Στη συνέχεια, η μεταβλητή isLoggedIn καθορίζει αν ο χρήστης είναι συνδεδομένος στην εφαρμογή καθώς οι συναρτήσεις onLogout, onLogin καθορίζουν τον χειρισμό που σχετίζονται με το login και το logout. Επίσης, υπάρχουν οι μέθοδοι getCSRF, getSession όπου χρησιμοποιούνται για τη διαχείριση του csrf token και τη διαχείριση στην περίπτωση όπου ο χρήστης παρέμεινε συνδεδεμένος. Η μεταβλητή httpRequest καθορίζει το σημείο όπου θα γίνονται τα αιτήματα. Στη συνέχεια, ορίζεται ο UserProvider, ο οποίος χρησιμοποιεί τον useState hook για να δημιουργήσει δύο καταχωρητές κατάστασης (isLoggedIn και csrf) και τον useEffect hook για να εκτελέσει τις συναρτήσεις getCSRF και getSession κατά τη δημιουργία του UserProvider component, με την παράμετρο [] να υποδηλώνει ότι θα εκτελεστούν μόνο μία φορά.

Τέλος, επιστρέφεται το UserContext.Provider component, το οποίο περιβάλλει όλα τα παιδιά του, παρέχοντας το περιεχόμενό του στο κάθε παιδί που το χρησιμοποιεί, χρησιμοποιώντας το value prop. Το UserContext εξάγεται για να χρησιμοποιηθεί αλλού στην εφαρμογή.

```
const Contacts = () =>{
  const userCtx = useContext(UserProvider);
  const [scholarId, setScholarId] = useState('')
  const [contactList, setContactList] = useState([])
  const [ fetchingContact, setFetchingContact ] = useState(false)

  useEffect( () =>{
    if(userCtx.csrf){
      getContacts()
    }
  },[userCtx.csrf])



  const getContacts =() =>{
    //Fetches all contacts of the user
  }

  const addContact = () =>{
    //Adds a contact to the user
    //Sends http request with scholar ID
    //Prints proper message
  }


  const removeContact = (e)=> {
    //Removes a contact from the user
    //Sends http request with scholar ID
    //Prints proper message
  }


  return(
    <div className={classes.container}>
      <div className={classes.addContactContainer}>
        <h1  className={classes.heading}>Add contacts</h1>
        <div className="d-flex flex-row flex-wrap align-items-center w-100 justify-content-center gap-3
mb-3">
          <input className={classes.greenInput} onInput={e => setScholarId(e.target.value)} value=
{scholarId} />
          { !fetchingContact && <button onClick={addContact} className={classes.addButton}>Add
contact</button> }
          { fetchingContact && <HashLoader
            color="#42B883"
            className={classes.spinner}
            loading={true}
            size={25}
            aria-label="Loading Spinner"
            data-testid="loader"
          /> }
        </div>
        <div className={classes.contactListContainer}>
          {contactList &&  contactList.map( (contact) =>{
            return(<ContactList key={contact.scholarId} contact={contact} removeContact={removeContact}
/>)
          } )}
        </div>
      </div>
    </div>
  )

}

export default Contacts
```

Σχήμα 4.13: Κλάση προβολής και διαχείρισης επαφών

```
const ContactList = (props) => {
  const [isWatching, setIsWatching] = useState(false);
  const [showPapers, setShowPapers] = useState(false)
  const toggleIsWatching = () =>{setIsWatching(!isWatching)}
  const toggleShowPapers = () =>{setShowPapers(!showPapers)}

  return (
    <div className={classes.container}>
      <div className="row my-2">
        <div className="col-12 col-md-3">
          <div className="d-flex h-100 text-center align-items-center justify-content-center">
            <img src={props.contact.image} className={classes.image}></img>
          </div>
        </div>
        <div className="col-12 col-md-6 ">
          <div className="d-flex h-100 text-center align-items-center justify-content-center">
            <div className={classes.contactInfo}>
              <span>{props.contact.fullName}</span>
              <span>{props.contact.scholarId}</span>
            </div>
          </div>
        </div>
        <div className="col-12 col-md-3">
          <div className="d-flex h-100 text-center align-items-center justify-content-center">
            <div className={classes.contactListActionsContainer}>
              <button onClick={(e) => setIsWatching(!isWatching)}>
                <FontAwesomeIcon icon={faChartLine} />
              </button>
              <button onClick={(e) =>  setShowPapers(!showPapers)}>
                <FontAwesomeIcon icon={faFileLines} />
              </button>
              <button
                onClick={props.removeContact}
                id={props.contact.scholarId}
              >
                <FontAwesomeIcon icon={faTrash} />
              </button>
            </div>
          </div>
        </div>
      </div>
      <Papers  scholarId={props.contact.scholarId} onClose={toggleShowPapers}  isOpen={showPapers} />
      <Overview scholarId={props.contact.scholarId} onClose={toggleIsWatching}  isOpen={isWatching} />
    </div>
  );
};
export default ContactList;
```

Σχήμα 4.14: Κλάση προβολής δεδομένων επαφής

Στο σχήμα 4.13 παρουσιάζεται η κλάση όπου με όνομα ”Contacts”, όπου είναι υπεύθυνη για την προβολή και τη διαχείριση επαφών ενός χρήστη. Η συνιστώσα εισάγει το περιβάλλον χρήστη, το οποίο περιλαμβάνει το CSRF του χρήστη για αυθεντικοποίηση, καθώς και άλλες εξαρτήσεις, όπως το HashLoader από τη βιβλιοθήκη react-spinners και το toast από τη βιβλιοθήκη react-toastify. Στο σώμα της συνιστώσας, ορίζονται οι μεταβλητές κατάστασης και οι μεθόδοι όπως η getContacts, η addContact και η removeContact, οι οποίες επικοινωνούν με ένα API για να ανακτήσουν, να προσθέσουν ή να αφαιρέσουν επαφές. Επίσης, εισάγονται τα απαραίτητα modules όπως το useContext, το useState και το useEffect και η μορφοποίηση της HTML από το αρχείο Contacts.module.css.

Αρχικά, με τη χρήση του useState, δημιουργούνται τρεις μεταβλητές κατάστασης (state variables), δη-

λαδή το scholarId, contactList, fetchingContact. Η τιμή τους μπορεί να αλλάξει στον κώδικα με την αντίστοιχη εντολή set. Οι επαφές προέρχονται από API με τη χρήση της μεθόδου getContacts. Επίσης, η προσθήκη και η αφαίρεσή επαφών γίνεται με τις μεθόδους addContact και removeContact αντίστοιχα. Η σελίδα επαφών ανανεώνεται αυτόματα όταν ο χρήστης προσθέσει ή αφαιρέσει μια επαφή. Εμφανίζεται μια λίστα με τις υπάρχουσες επαφές και ο χρήστης μπορεί να προσθέσει νέες επαφές με το πάτημα ενός κουμπιού ”Add contact”. Επίσης, υπάρχει ένα spinner που εμφανίζεται κατά τη διάρκεια της φόρτωσης κατά την προσθήκη μιας νέας επαφής.

Η μέθοδος Contact για κάθε επαφή του χρήστη περιέχει ένα στοιχείου του μοντέλου Contact List σχήμα 4.14. Αυτή η μέθοδος περιέχει μια λίστα με επαφές, κάθε μία από τις οποίες εμφανίζεται η φωτογραφία, το αντίστοιχο όνομα και αριθμό αναγνωριστικού ενός ερευνητή (scholarId). Επίσης, υπάρχουν κουμπιά που επιτρέπουν στον χρήστη να δει τα δεδομένα του ερευνητή του επιλεγμένου ερευνητή, καθώς και να διαγράψει την επαφή από τη λίστα.

Τα κουμπιά εμφανίζονται με εικονίδια. Χρησιμοποιούνται hooks της React όπως το useState για να διαχειριστούν την κατάσταση των αναδυόμενων παραθύρων όπου περιέχουν τα δεδομένα τους. Συγκεκριμένα, οι μέθοδοι toggleIsWatching και toggleShowPapers χρησιμοποιούνται για τον έλεγχο εμφάνισης των δεδομένων ή των διαθέσιμων έγγραφων κάποιας επαφής.

```
const Papers = (props) => {
  const userCtx = useContext(UserProvider);
  const [papers, setPapers] = useState([]);
  const [page, setPage] = useState(0)
  const [limit, setLimit] = useState(10)
  const [isLast, setIsLast] = useState(false)
  useEffect(() => {
    if( props.isOpen && !isLast ){
      fetchData()
    }
  }, [page,props.isOpen ])

  const fetchData = () => {//fetches data from api};
  const calculatePercentage = (i) => {//function that creates a dynamic progress}
  let timer;
  const countProgress = (e) => {//function to handle scrollbar progress};
  const handleClick = (e) =>{
    if(e.target.classList.contains('close')){
      props.onClose()
    }
  }

  return (
    <div  onClick={(e) => handleClick(e)}
      className={`${classes.fixedContainer} ${props.isOpen ? classes.isOpen : ""} close`}>
      <div className={classes.container} onScroll={(e) => countProgress(e)}>
      <button onClick={() => {props.onClose()}} className={classes.closeButton}>
        <FontAwesomeIcon icon={faClose} />
      </button>
        <div>
          {papers &&
            papers.map((paper, index) => {
              return (
                <div key={`${props.scholarId}${index}`} className={`${classes.paper} row g-0`}>
                  <div className="d-flex flex-column col-9">
                    <span className={`fs-5`}>Title : {paper["title"]}</span>
                    { paper["authors"] != '' &&
                       <span className={`${classes.smallText}`}>
                       Authors : {paper["authors"]}
                     </span>
                     }
                    { paper["event"] != '' &&
                      <span className={`${classes.smallText}`}>
                      Event : {paper["event"]}
                    </span>
                     }
                  </div>
                  <div className={`col-2 d-flex flex-column align-items-center`}>
                    {paper["citedBy"] > 0 && (
                      <>
                        <span>Cited By</span>
                        <span>{paper["citedBy"]}</span>
                      </>
                    )}
                  </div>
                  <div className={`col-1 d-flex flex-column align-items-center`}>
                    {paper["year"] > 0 && (
                      <>
                        <span>Year</span>
                        <span>{paper["year"]}</span>
                      </>
                    )}
                  </div>
                </div>
              );
            })}
        </div>
      </div>
    </div>
  );
};

export default Papers;
```

Σχήμα 4.15: Κλάση προβολής δημοσιεύσεων επαφής

Η συνάρτηση Papers είναι ο κόμβος για την προβολή των συνολικών δημοσιεύσεων μιας επαφής. Περιέχει βασικές μεταβλητές βασισμένες στην κατάσταση (state) της React για τη σωστή προβολή και διαχείριση των δημοσιεύσεων. Οι μεταβλητές της συνάρτησης με εξαίρεση την πρώτη μεταβλητή στο σχήμα 4.15, όπου χρησιμοποιείται για τα αιτήματα στον διακοσμητή, βασίζονται στο hook της react με όνομα useState.

Οι μεταβλητές για τη διαχείριση των δεδομένων είναι η papers, η page, η limit και η isLast. Η πρώτη μεταβλητή αποτελεί έναν πίνακα αντικειμένων, όπου κάθε αντικείμενο αποτελεί μια δημοσίευση του επιβλέπων ερευνητή. Βάση της συγκεκριμένες μεταβλητής φορτώνεται στο γραφικό περιβάλλων μία γραμμή για κάθε δημοσίευση του, τα δεδομένα που προβάλλονται είναι ο τίτλος της δημοσίευσης, τα ονόματα των υπόλοιπων ερευνητών όπου συντέλεσαν στην εργασία, το συνέδριο όπου παρουσιάστηκε η εργασία καθώς και τις συνολικές αναφορές της και τον χρονολογία δημοσίευσης. Η μεταβλητή page μαζί με την limit καθορίζουν το όριο της κάθε σελίδας καθώς και την τρέχουσα σελίδα με σκοπό τα δεδομένα κάθε ερευνητή να φορτώνονται σταδιακά και να μην επιβαρύνονται οι επιδώσεις του συστήματος στην περίπτωση όπου ένας ερευνητής έχει μεγάλο πλήθος δημοσιεύσεων. Όταν το api επιστρέψει όλες τις δημοσιεύσεις τότε η μεταβλητή isLast αλλάζει τιμή από ψευδή σε αληθή ώστε να μη δημιουργηθεί καινούργιο αίτημα προς τον διακοσμητή, με σελίδα και όριο όπου δε θα επιστρέψει δεδομένα. Στη συνέχεια οι συναρτήσεις countProgress και calculatePercentage καθορίζουν τη συνθήκη ανάλογα με την πλοήγηση του χρήστη ώστε να ζητήσουν και να φορτώσουν δεδομένα στη διεπαφή. Τέλος, στη συνάρτηση Papers με τη χρήση τον props δίνεται πρόσβαση σε εξωτερική συνάρτηση ώστε να εξαφανίζεται η γραφική διεπαφή.

```
const Overview = (props) => {
  const userCtx = useContext(UserProvider)
  const [citationChartData, setCitationChartData] = useState([])
  const [publicationChartData, setPublicationChartData] = useState([])
  const [citations, setCitations] = useState([])
  const [publications, setPublications] = useState([])
  const [hIndex, setHIndex] = useState([])

  useEffect(() => {
    if(props.isOpen && hIndex.length == 0 ){
      fetchData()
    }
  }, [userCtx.csrf, props.scholarId,props.isOpen]);

  const fetchData = () => {//fetches and saves data from api};
  const citationsCharts = (
    // <ResponsiveContainer className={classes.chartContainer}  aspect={2}>
    <ResponsiveContainer width="98%" height={300}>
      <AreaChart  data={citationChartData}>
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis dataKey="name" />
        <YAxis domain={[0, "dataMax + 5"]} />
        <Tooltip  contentStyle={{backgroundColor: "#151929",color: "#42b883"}}  />
        <Area
          type="monotone"
          dataKey="citations"
          stroke="#42B883"
          fill="#5fe7aa59"
        />
      </AreaChart>
    </ResponsiveContainer>
  );

  const publicationsCharts = (
    //same as citations with different data
  );

  const handleClick = (e) =>{
    if(e.target.classList.contains('close')){
      props.onClose()
    }
  }

  return (
    <div onClick={(e) => handleClick(e)}
      className={`${classes.fixedContainer} ${props.isOpen ? classes.isOpen : ""} close`}
    >
      <div className={classes.container}>
      <button
        onClick={() => {
          props.onClose()
        }}
        className={classes.fixedContainerCloseButton}
      >
        <FontAwesomeIcon icon={faClose} />
      </button>
        <div className="row my-3 g-0">
          <div className="col-12 ">
            <h1 className={classes.title}>Citations</h1>
            {citationsCharts}
          </div>
          <div className="col-12 ">
            <div className="d-flex flex-column justify-content-center align-items-center">
              <OverviewStats data={citations} />
              <OverviewStats data={hIndex} />
            </div>
          </div>
        </div>
        <div className="row g-0">
          <div className="col-12 ">
            <h1 className={classes.title}>Publications</h1>
            {publicationsCharts}
          </div>
          <div className="col-12 ">
            <div className="d-flex justify-content-center align-items-center">
              <OverviewStats data={publications} />
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default Overview;
```

Σχήμα 4.16: Κλάση προβολής δεδομένων σχετικά με τις επιδόσεις ενός ακαδημαϊκού συγγραφέα

Στη συνάρτηση Overview στο σχήμα 4.16, γίνεται εισαγωγή βιβλιοθηκών για την προβολή δεδομένων σε γράφημα όπως η βιβλιοθήκη Area, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer από τη βιβλιοθήκη recharts. Επιπλέον, χρησιμοποιεί τη συσκευασία UserProvider από το store/user-context για να παρέχει πληροφορίες στα επιμέρους στοιχεία του παρατηρητή σχετικά με το χρήστη και τα δεδομένα του.

Ο βασικός σκοπός αυτού του κώδικα είναι να κάνει μια αίτηση στον εξυπηρετητή με μια συγκεκριμένη μέθοδο (POST) για να λάβει στοιχεία σχετικά με τις επιδόσεις ενός ακαδημαϊκού συγγραφέα. Η απάντηση αυτής της αίτησης περιλαμβάνει στοιχεία σχετικά με τις αναφορές, τις δημοσιεύσεις και τον δείκτη Η του συγγραφέα και επιστρέφονται στο χρήστη με τη μορφή γραφημάτων και στατιστικών πληροφοριών.

Στη συνέχεια, η συνάρτηση ”Overview” δημιουργείται ως σταθερή συνάρτηση, που δέχεται μια παράμετρο με όνομα ”props”. Στο εσωτερικό της συνάρτησης, δηλώνονται μερικές μεταβλητές χρησιμοποιώντας τις συναρτήσεις useState και useContext. Η συνάρτηση useEffect χρησιμοποιείται για την ανάκτηση δεδομένων από έναν server κατά την αρχικοποίηση του component ή όταν τοποθετούνται νέες τιμές στο props.scholarId ή στο userCtx.csrf ή όταν αλλάζει η τιμή του props.isOpen. Η μέθοδος fetchData χρησιμοποιείται για τη λήψη δεδομένων από τον διακομιστή, με το οποίο επικοινωνεί μέσω μιας αίτησης POST στο URL api/overview/.

4.4 Cronjob

```
const Cron = () =>{
  const userCtx = useContext(UserProvider);
  const [ result, setResult ] = useState("Fetching.....")
  const updateDB = () =>{

      fetch(`${userCtx.httpRequest}api/update/`, {
        credentials: "include",
        })
        .then((res) => res.json())
        .then((data) => {
          if ( data['flag'] ){
            setResult("All went  well")
            toast("All went  well")
          }else{
            setResult("Something went wrong")
            toast("Something went wrong")
          }
        })
        .catch((err) => {
          setResult("Something went wrong")
          toast("Something went wrong")
        });
  }
  useEffect( ()=>{updateDB()},[] )
  return(<div>{result}</div>)
```

Σχήμα 4.17: Κλάση για την ενημέρωση δεδομένων της βάσης.

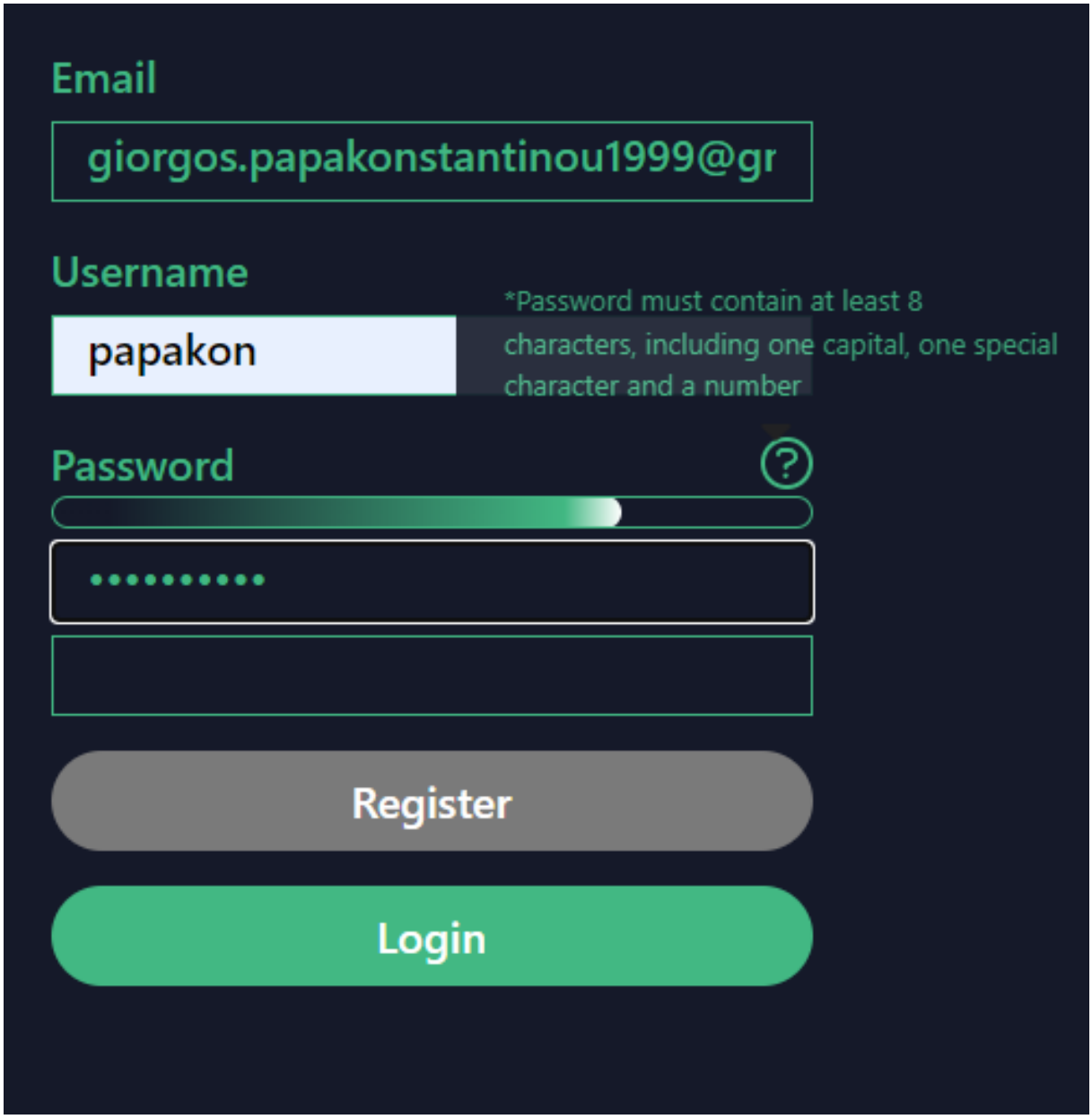
Ο κώδικας του σχήματος 4.17 αναφέρεται στη δημιουργία ενός στοιχείου της διεπαφής χρήστη που ονομάζεται ”Cron”. Το useState χρησιμοποιείται για τη διαχείριση μιας κατάστασης (state) που αντιπροσωπεύει το κείμενο που θα εμφανίζεται στην οθόνη (”Fetching.....” ή ”All went well” ή ”Something went wrong”). Τέλος, το useEffect χρησιμοποιείται για την εκτέλεση της συνάρτησης updateDB όταν η συνιστώσα φορτώνεται.

Η συνάρτηση updateDB καλεί τη μέθοδο fetch για να καλέσει μια διαδικτυακή υπηρεσία (API) που βρίσκεται στη διεύθυνση ”api/update/” και περιμένει την απάντηση από την υπηρεσία. Η παραπάνω κλήση έχει ως αποτέλεσμα την ενημέρωση των στατιστικών δεδομένων των ερευνητών όπου υπάρχουν στη βάση δεδομένων. Αν η απάντηση περιλαμβάνει ένα flag (flag===true), τότε εμφανίζεται το κείμενο ”All went well” και ένα μήνυμα toast που ειδοποιεί το χρήστη ότι όλα πήγαν καλά δηλαδή τα δεδομένα των ερευνητών έχουν ενημερωθεί. Αν το flag δεν είναι true, τότε εμφανίζεται το κείμενο ”Something

went wrong” και ένα αντίστοιχο μήνυμα toast.

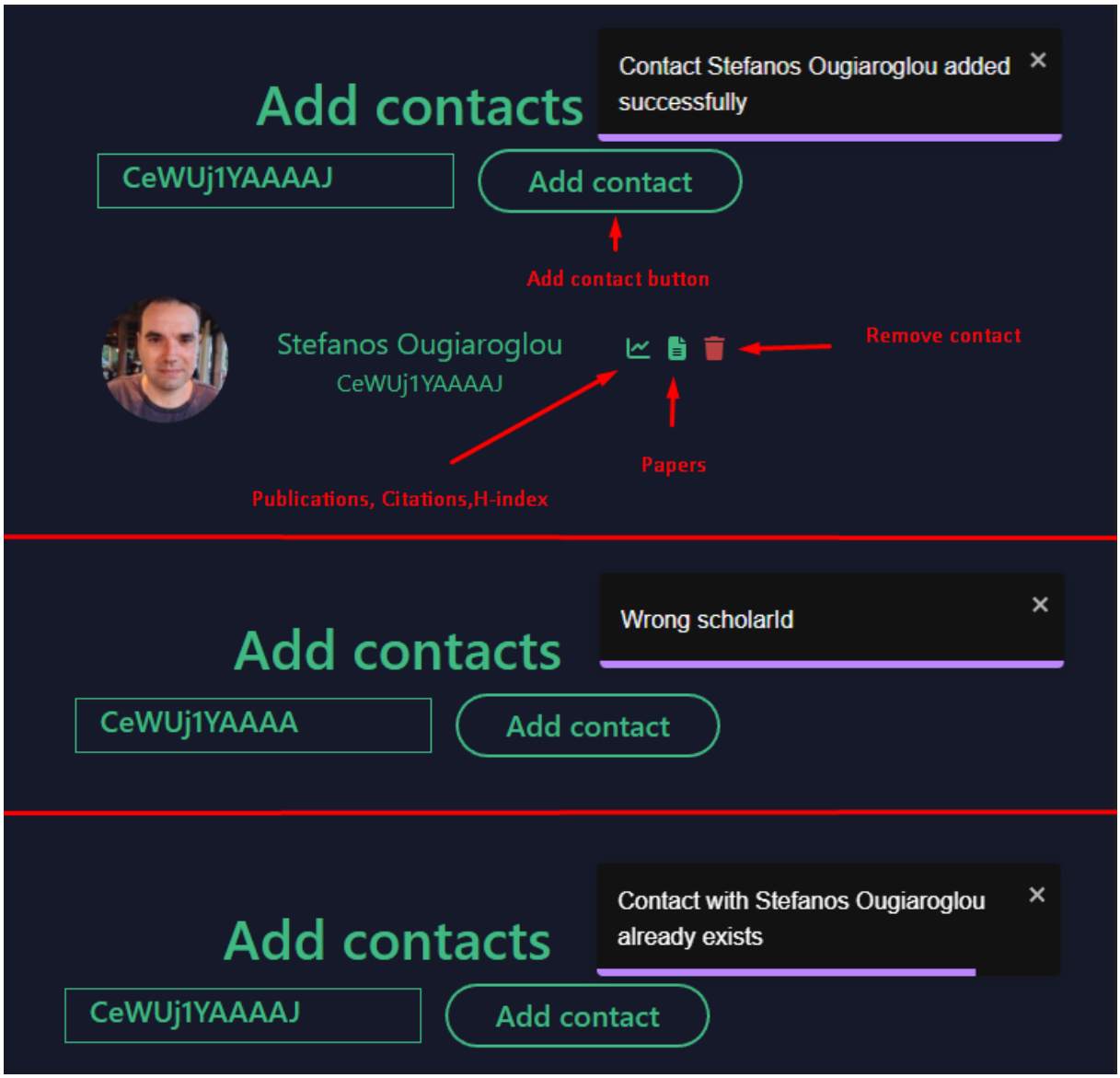
Κεφάλαιο 5ο: Παρουσίαση της Εφαρμογής

5.1 Δημιουργία επαφών



Σχήμα 5.1: Γραφική διεπαφή για τη δημιουργία λογαριασμού.

Για την πρόσβαση στην εφαρμογή είναι προαπαιτούμενο η δημιουργία λογαριασμού.Στο σχήμα 5.1 παρουσιάζεται η γραφική διεπαφή για την δημιουργία λογαριασμού. Τα πεδία όπου απαιτούνται για τη δημιουργία λογαριασμού είναι το ηλεκτρονικό ταχυδρομείο, ένα όνομα χρήστη και ο κωδικός με συγκεκριμένα χαρακτηριστικά. Για τον κωδικό υπάρχει μια μπάρα όπου βαθμολογεί την ισχύ του. Αφού συμπληρωθούν τα στοιχεία σωστά το κουμπί καταχώρησης της φόρμας ενεργοποιείται.Όταν ο χρήστης καταχωρήσει τη φόρμα επιστρέφεται και αντίστοιχο μήνυμα για την επιτυχία ή μη της ενέργειας.

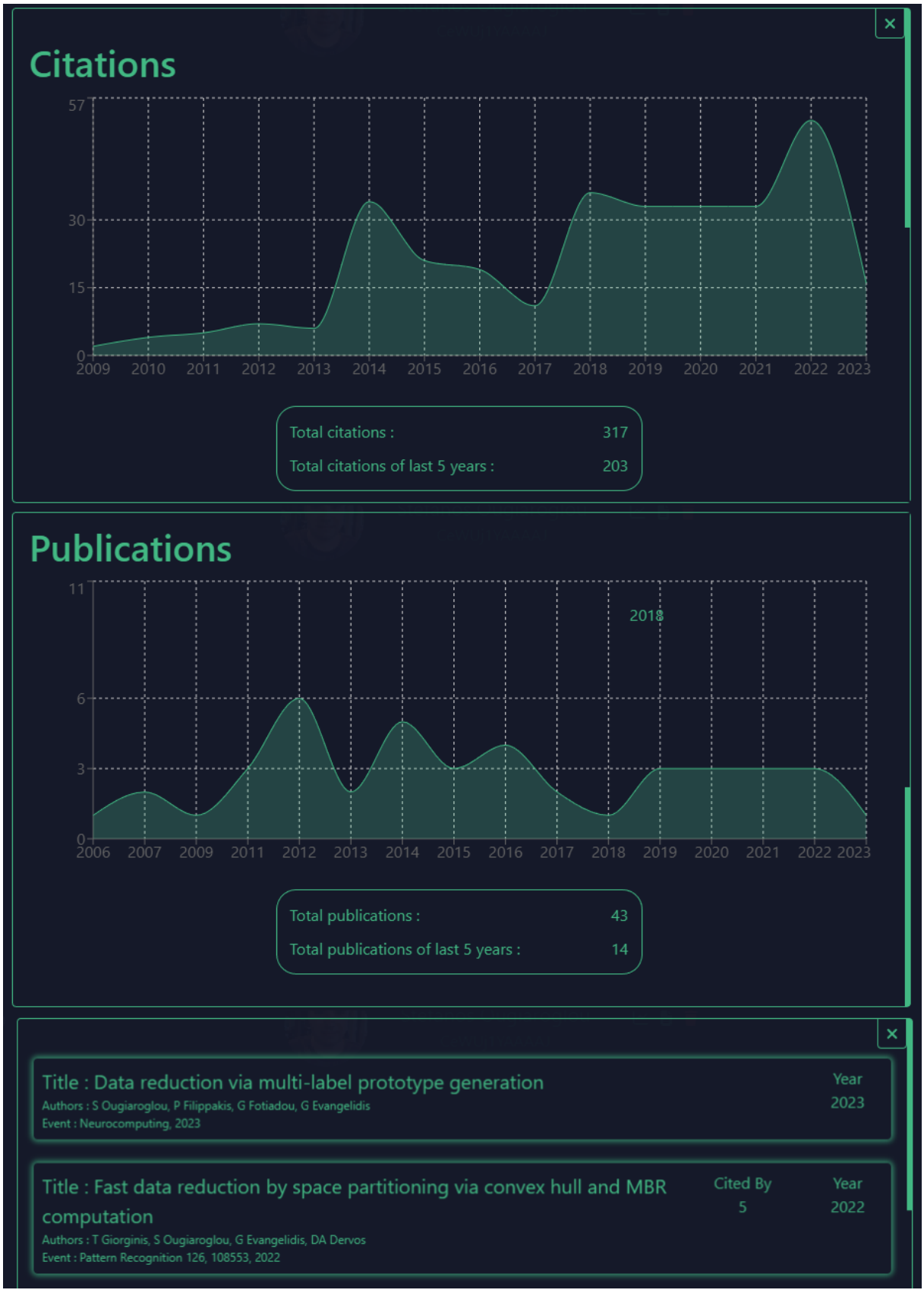


Σχήμα 5.2: Γραφική διεπαφή για την προσθήκη και προβολή επαφής.

Αφού γίνει ολοκλήρωση της εγγραφής και σύνδεση με τον λογαριασμό η πρώτη σελίδα όπου εμφανίζεται στον χρήστη είναι η σελίδα προσθήκης επαφής δηλαδή αυτή του σχήματος 5.2. Για την προσθήκη επαφής ο χρήστης το μοναδικό στοιχείο όπου χρειάζεται να εισάγει στη φόρμα είναι το αναγνωριστικό του ερευνητή ή αλλιώς Scholar Id. Μετά την εισαγωγή του αναγνωριστικού και το πάτημα του κουμπιού για την καταχώρηση της επαφής, το κουμπί εξαφανίζεται μέχρι να υπάρχει απάντηση από τον διακοσμητή.

Κατά την καταχώρηση επαφής υπάρχουν τρία πιθανά σενάρια. Το πρώτο είναι αυτό όπου το αναγνωριστικό αντιστοιχίζεται με ερευνητή του Google Scholar ο οποίος δεν υπάρχει στις επαφές του χρήστη. Μετά την εξαγωγή δεδομένων της επαφής σε περίπτωση όπου δεν υπάρχουν ήδη στη βάση δεδομένων εμφανίζετε μήνυμα επιτυχίας στον χρήστη καθώς η επαφή του με τη φωτογραφία του ερευνητή, το ονοματεπώνυμο, το αναγνωριστικό καθώς και τρία κουμπιά για τη διαχείριση της επαφής.Κατά σειρά εμφάνισης τα κουμπιά είναι υπεύθυνα για την προβολή των μετρικών του Google Scholar, την παρουσίαση των διαθέσιμων άρθρων του ερευνητή καθώς και τη διαγραφή της επαφής. Η δεύτερη περίπτωση

χρήσης είναι η εισαγωγή λανθασμένου Scholar Id όπου εμφανίζεται στην περίπτωση που το εισαγόμενο αναγνωριστικό δεν αντιστοιχίζεται σε ερευνητή. Σε αυτή την περίπτωση ο χρήστης ενημερώνεται με το κατάλληλο λεκτικό μήνυμα. Υπάρχει η περίπτωση εισαγωγής αναγνωριστικού ερευνητή όπου υπάρχει στις επαφές του χρήστη.Τότε ο χρήστης ενημερώνεται με το κατάλληλο λεκτικό μήνυμα καθώς επίσης αναφέρεται και το ονοματεπώνυμο του ερευνητή όπου έγινε προσπάθεια επανεισαγωγής.

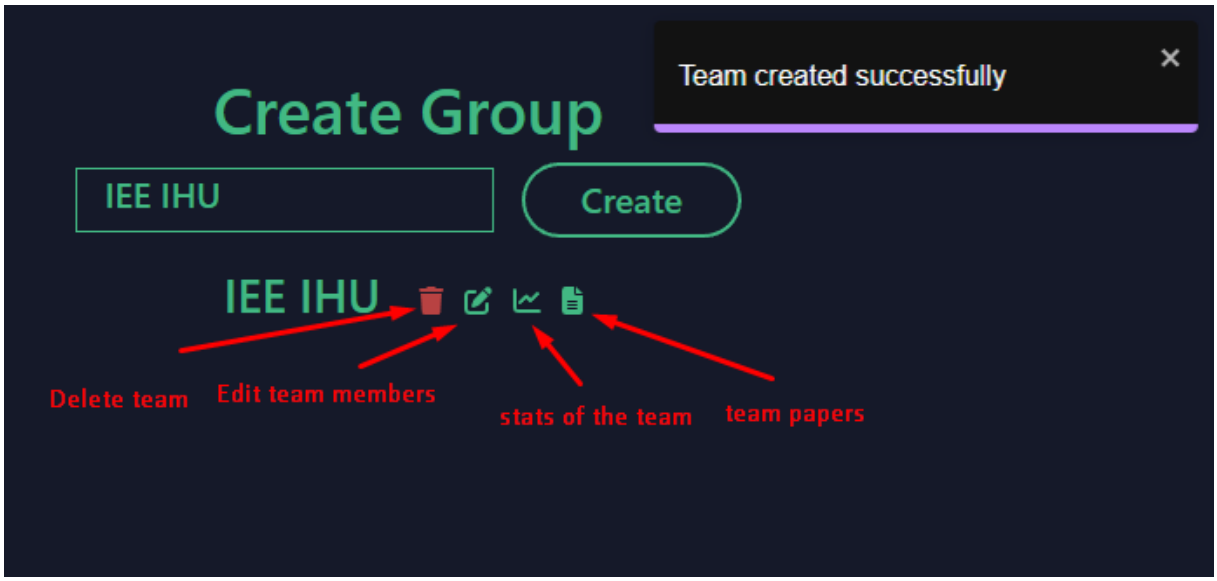


Σχήμα 5.3: Γραφική διεπαφή για τις μετρικές και τις δημοσιεύσεις του ερευνητή.

Αναφορικά με το σχήμα 5.3 υπάρχει γραφική αναπαράσταση των δεδομένων του ερευνητή. Πιο συγκε-

κριμένα κατά το πάτημα του πρώτου κουμπιού τα δεδομένα της επαφής προβάλλονται σε αναδυόμενο παράθυρο όπου περιέχονται σε γράφημά οι δημοσιεύσεις και οι αναφορές ανά έτος ενώ τα συνολικά δεδομένα και τα δεδομένα των τελευταίων πέντε ετών προβάλλονται σε μορφή πίνακα. Στο κάτω μέρος του σχήματος υπάρχει η γραφική αναπαράσταση των διαθέσιμων ερευνητικών δημοσιεύσεων της επαφής. Για κάθε δημοσίευση προβάλλεται ο τίτλος, οι συγγραφείς, η ημερίδα παρουσίασης καθώς και σε περίπτωση όπου υπάρχουν το έτος της και τις συνολικές αναφορές της δημοσίευσης.

5.2 Δημιουργία ομάδας



Σχήμα 5.4: Γραφική διεπαφή για τις μετρικές και τις δημοσιεύσεις του ερευνητή.

Για τη δημιουργία ομάδας το μοναδικό χαρακτηριστικό όπου χρειάζεται να εισάγει ο χρήστης είναι το όνομα της ομάδας. Κατά το πάτημα του κουμπιού δημιουργίας όπου εμφανίζεται στο σχήμα 5.4. υπάρχει ενημέρωση στην οθόνη του χρήστη για την επιτυχή εισαγωγή ομάδας.Επίσης, εμφανίζεται στην οθόνη η καινούργια ομάδα με τέσσερα κουμπιά για την επεξεργασία της.

Κατά σειρά του σχήματος από αριστερά στα δεξιά το πρώτο κουμπί διαγράφει την ομάδα. Το δεύτερο κουμπί εμφανίζει ένα αναδευόμενο παράθυρο όπου περιέχει όλες τις επαφές του χρήστη με σκοπό να προσθέσει κάποια από αυτές στην ομάδα. Αφού γίνει εισαγωγή των ατόμων στην ομάδα ο χρήστης μπορεί να δει και να συγκρίνει τα μέλη της ομάδας βάση των μετρικών του Google Scholar με το πάτημα του τρίτου κουμπιού. Ακόμα κατά το πάτημα του τέταρτου κουμπιού εμφανίζεται ένα αναδυόμενο παράθυρο όπου περιέχει τις δημοσιεύσεις της ομάδας.

Σχήμα 5.5: Γραφική διεπαφή για τις μετρικές και τις δημοσιεύσεις του ερευνητή.

Αναφορικά με το σχήμα 5.5 και τις διαθέσιμες λειτουργίες της ομάδας, ο χρήστης μπορεί να εισάγει επαφές ή να αφαιρέσει απτή ομάδα καθώς και να συγκρίνει τις διαθέσιμες μετρικές. Για την προσθήκη ή αφαίρεση ατόμων ο χρήστης θα πρέπει να πατήσει στο δεύτερο κουμπί όπου εμφανίζονται οι επαφές του. Στη συνέχεια στο αναδυόμενο παράθυρο οι επαφές όπου δεν έχει προσθέσει στην ομάδα εμφανίζονται με πράσινο εικονίδιο προσθήκης ενώ αν υπάρχει ήδη εμφανίζεται με κόκκινο εικονίδιο αφαίρεσης. Τα παραπάνω εικονίδια αποτελούν κουμπιά τα οποία κατά το πάτημα κάνουν αίτημα για την προσθήκη ή την αφαίρεση της επαφής. Στην απάντηση του αιτήματος εμφανίζεται αντίστοιχο μήνυμα στον χρήστη.

Αφού προστεθούν επαφές στην ομάδα ο χρήστης μπορεί να παρακολουθήσει τα δεδομένα τους πατώντας το τρίτο κουμπί. Για κάθε χρήστη δημιουργείται ένα δεκαεξαδικό χρώμα με το οποίο ”ζωγραφίζονται” τα δεδομένα του για καλύτερη παρακολούθηση τους. Η προβολή των δημοσιεύσεων και των αναφορών γίνονται με τη χρήση γραφήματος. Κάθε γραμμή του χρήστη εμφανίζεται με το αντίστοιχο χρώμα όπου του έχει ανατεθεί, κατά το σύρσιμο του κέρσορα πάνω στο γράφημά παρουσιάζεται ο αριθμός των δημοσιεύσεων ή των αναφορών όλης της ομάδας καθώς και ατομικά του κάθε ερευνητή. Επίσης, τα συνολικά δεδομένα της ομάδας εμφανίζονται σε μορφή πίνακα.

Για την προβολή της κάθε δημοσίευσης η αναπαράσταση γίνεται σε μορφή λίστας με κάθε δημοσίευση ως αντικείμενο της λίστας. Για κάθε αντικείμενο περιέχεται ο τίτλος της δημοσίευσης, οι συγγραφείς, το συνέδριο όπου παρουσιάστηκε, καθώς επίσης σε περίπτωση όπου υπάρχει, ο αριθμός των αναφορών και η χρονιά δημοσίευσης. Ακόμα αξίζει να σημειωθεί ότι στην περίπτωση όπου μία δημοσίευση υπάρχει σε δύο ερευνητές της ίδιας ομάδας. Τότε γίνεται διασταύρωση τη δημοσίευσης ώστε να μην εμφανίζεται πολλαπλές φορές και περιέχει το εικονίδιο των ερευνητών όπου συντέλεσαν στη δημοσίευση. Τέλος, τα αντικείμενα της λίστας είναι ταξινομημένα κατά το έτος δημοσίευσης.

5.3 Σενάρια χρήσης

Η παρούσα διατριβή ως στόχο την επίλυση προβλημάτων οργάνωσης στον κλάδο της επιστημομετρίας. Οι χρήστες της εφαρμογής μπορούν να δημιουργήσουν λογαριασμό ώστε να αποθηκεύονται οι ενέργειες και οι προτιμήσεις τους. Ένας χρήστης μπορεί να δημιουργήσει την προσωπική του ατζέντα με τις δικές τους επαφές από ερευνητές όπου τον ενδιαφέρουν. Επίσης, για κάθε ερευνητή υπάρχει η δυνατότητα προβολής των μετρικών της επιστημομετρίας τόσο σε γράφημα ανά έτος όσο και συνολικά. Ακόμα δίνεται δυνατότητα παρακολούθησης κάθε δημοσίευσης με στοιχεία όπως ο τίτλος, οι συντελεστές όπου βοήθησαν, το συνέδριο όπου παρουσιάστηκε τις συνολικές αναφορές και τη χρονιά της δημοσίευσης.

Πέρα από το ατομικό κομμάτι οι εργαλειοθήκη της εφαρμογής επεκτείνεται και σε συλλογικό. Αρχικά υπάρχει η δυνατότητα οργάνωσης των επαφών σε ομάδες. Σε κάθε ομάδα μπορούν να προστεθούν οι επιστήμονες που βρίσκονται στην ατζέντα του χρήστη. Ο χρήστης μπορεί να προσθέσει και να αφαιρέσει άτομα στην ομάδα άμεσα. Επίσης, για κάθε ομάδα υπάρχει η δυνατότητα προβολής των στατιστικών των επιστημών. Σε κάθε ομάδα ο χρήστης μπορεί να δει και να συγκρίνει τις αναφορές και τις δημοσιεύσεις στα μέλη της ανά χρονιά, αλλά ακόμα μπορεί να δει και το συλλογικό έργο, σε μορφή γραφήματος. Επιπρόσθετος, ο χρήστης έχει πρόσβαση στα βασικά χαρακτηριστικά των δημοσιεύσεων μιας ομάδας όπως ακριβώς και στην περίπτωση της επαφής. Στην περίπτωση κοινής δημοσίευσης μεταξύ δύο ή περισσότερων μελών της ομάδας η δημοσίευση εμφανίζεται μία φορά μαζί όμως με τα εικονίδια των συντελεστών.

Κεφάλαιο 6ο: Συμπεράσματα και Μελλοντικές επεκτάσεις

6.1 Συμπεράσματα

Η παρούσα πτυχιακή αποσκοπούσε στην επέκταση της εργαλειοθήκης του Google Scholar. Πέρα από τη συλλογή δεδομένων, αποτελεί προσπάθεια για την καλύτερη οργάνωση των δεδομένων. Πιο συγκεκριμένα, οι χρονοβόρες διαδικασίες όπως η αναζήτησης και προσπέλασης ερευνητών ελαχιστοποιούνται. Οι χρήστες της πλατφόρμας έχουν τη δυνατότητα προβολής συγκεκριμένων ερευνητών όπου ενδιαφέρονται με τη μορφή επαφής. Ακόμα για τη συλλογή δεδομένων μιας επαφής ο χρήστης δε χρειάζεται να αφιερώσει χρόνο καθώς τα δεδομένα συλλέγονται και ενημερώνονται αυτόματα. Επίσης, η εφαρμογή κάνει δυνατή τη δημιουργία ομάδας και την ενσωμάτωση επαφής σε αυτή. Τόσο στην επαφή όσο και στην ομάδα τα δεδομένα οπτικοποιούνται παρέχοντας τα συνολικά αποτελέσματα των μετρικών αλλά και τα δεδομένα ανά χρονιά σε μορφή διαγράμματος. Οι διαφορές μεταξύ ερευνητών οι ομάδων γίνονται ορατές με άνεση χωρίς να απαιτείτε μεγάλος κόπος από τον χρήστη. Η εφαρμογή προσφέρει ένα ολοκληρωμένο οικοσύστημα όπου η διαδικασία συλλογής, οργάνωσης, σύγκρισης δεδομένων με σκοπό την εξαγωγή πορίσματός γίνεται με μεγάλη ευκολία. Παρόμοιες υλοποιήσεις, ευκολύνουν την εκτίμηση ενός ερευνητή ή ερευνητικού κέντρου ώστε να γίνεται φανερή η απόδοση και το αντίκτυπο μιας διατριβής. Μελλοντικά μπορεί να χρησιμοποιηθεί για τη δημιουργία κατατάξεων ακαδημαϊκών τμημάτων που θα επιφέρει χρηματοδοτήσεις για την αύξηση της ακαδημαϊκής δραστηριότητας.

Αξίζει να σημειωθεί πως ο σκοπός της εφαρμογής δεν αναπτύχθηκε για την αντικατάσταση διαδικτυακών τόπων βιβλιογραφίας. Καθώς σε αυτές παρέχουν δεδομένων και είναι εξαιρετικά προσβάσιμες. Ωστόσο, η εφαρμογή συνδυάζει και παρουσιάζει τα πιο σημαντικά δεδομένα που απαιτούνται για τη σύγκριση των ερευνητών. Ο βασικός σκοπός της είναι η παροχή μιας χρήσιμης εργαλειοθήκης όπου διευκολύνει τη συλλογή δεδομένων και μειώνει την περιπλοκότητα σε περιπτώσεις σύγκρισης .

6.2 Μελλοντικές επεκτάσεις

Η παρούσα διατριβή αποβλέπει στην ολοκληρωμένη οργάνωση και επέκταση των δεδομένων της επιστημομετρίας, με στόχο τη βελτίωση των δυνατοτήτων του Google Scholar. Αν και η παρούσα προσπάθεια αποτελεί μια αρχή για τον εν λόγω στόχο, υπάρχουν πολλές προτάσεις που μπορούν να επεκτείνουν και να εμβαθύνουν τη διατριβή μας.

Με στόχο την ανάπτυξη και υλοποίηση νέων τεχνικών και εργαλείων που θα επιτρέπουν τη συλλογή, ανάλυση και κατηγοριοποίηση μεγάλου όγκου δεδομένων επιστημομετρικής πληροφορίας. Μπορεί να έχει ως τελικό αποτέλεσμα τη δημιουργία ενός συνολικού καταλόγου των ερευνητικών εργασιών και των επιστημονικών πηγών που υπάρχουν σε διάφορες πλατφόρμες και θα διευκολύνει την αναζήτηση και αξιολόγηση της επιστημονικής πληροφορίας από τον χρήστη.

Η αρχική προσέγγιση θα μπορούσε να εμπλουτιστεί με μια σειρά επιπλέον ιδεών για την ανάπτυξη της πτυχιακής εργασίας. Πρώτον, οι επιστήμονες θα μπορούσαν να έχουν πρόσβαση στα δεδομένα τους όχι μόνο μέσω της εφαρμογής, αλλά και να τα λαμβάνουν τοπικά σε μορφή csv. Αυτό θα διευκόλυνε τους χρήστες να αποθηκεύσουν τα δεδομένα τους στην τοπική τους συσκευή για ευκολότερη πρόσβαση και επεξεργασία.

Περαιτέρω, μια ενδιαφέρουσα ιδέα είναι η δημιουργία διαφορετικών τύπων χρηστών μέσα στην εφαρμογή. Μετά από διασταύρωση, οι χρήστες θα μπορούσαν να κατηγοριοποιηθούν σε διάφορες κατηγορίες, όπως "επιστήμονες", "φοιτητές" και άλλους. Έτσι, οι χρήστες που είναι επιστήμονες θα είχαν πρόσβαση σε επιπλέον δυνατότητες, όπως η δυνατότητα να επεξεργαστούν τα δεδομένα τους, είτε διορθώνοντάς τα είτε ανεβάζοντας δημοσιεύσεις που δεν υπάρχουν στο Google Scholar. Αυτό θα βοηθούσε τους χρήστες να έχουν πλήρη έλεγχο στα δεδομένα τους.

Επίσης, μια ιδέα για την επέκταση της εφαρμογής είναι η προσθήκη ενός συστήματος αξιολόγησης των εργασιών των επιστημόνων. Αυτό θα βοηθούσε τους χρήστες της εφαρμογής να αξιολογήσουν την ποιότητα των δημοσιεύσεών, ενώ θα παρείχε επίσης στους επιστήμονες στοχευμένα σχόλια για τη βελτίωση των επιστημονικών τους εργασιών.

Επιπλέον, μπορεί να δημιουργηθεί μια διαδικτυακή κοινότητα για τους χρήστες της εφαρμογής, όπου οι επιστήμονες θα μπορούν να ανταλλάξουν απόψεις, να συζητούν ερευνητικά θέματα και να δημοσιεύουν τις εργασίες τους για σχολιασμό από άλλους επιστήμονες. Μέσω αυτής της κοινότητας, οι επιστήμονες θα μπορούν να διαμορφώνουν την επιστημονική τους άποψη, να αναπτύσσουν ιδέες και να συνεργάζονται με άλλους επιστήμονες στον τομέα τους.

Επιπρόσθετα, η εφαρμογή μπορεί να επικεντρωθεί στην εισαγωγή πολλαπλών πλατφορμών προέλευσης δεδομένων, προκειμένου να αυξηθούν οι πληροφορίες που διαθέτουν οι ερευνητές. Τέλος, μια ενδιαφέρουσα πρόταση είναι η κοινοποίηση των ομάδων των χρηστών στην εφαρμογή, ώστε δυο χρήστες να μπορούν να συνεργάζονται και να επεξεργάζονται τα ίδια δεδομένα. Με αυτόν τον τρόπο, θα ενθαρρύνεται η συνεργασία και η ανταλλαγή γνώσεων μεταξύ των επιστημόνων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] I. CiteDrive, "Scientometrics," 2022.

[2] J. Mingers and L. Leydesdorff, "A review of theory and practice in scientometrics," *European Journal of Operational Research*, vol. 246, no. 1, pp. 1–19, 2015.

[3] Elsevier, "Scopus," 2022.

[4] W. of science, "Web of science," 2022.

[5] F. R. Jensenius, M. Htun, D. J. Samuels, D. A. Singer, A. Lawrence, and M. Chwe, "The benefits and pitfalls of google scholar," *PS: Political Science amp; Politics*, vol. 51, no. 4, p. 820–824, 2018.

[6] E. Jones, "Google books as a general research collection," *Library Resources Technical Services*, vol. 54, pp. 77–89, 04 2010.

[7] L. Richardson, "Beautiful soup documentation," *April*, 2007.

[8] B. Zhao, *Web Scraping*, pp. 1–3. 05 2017.

[9] S. Singh and A. Haque, "Anti-scraping application development," 08 2015.

[10] K. Turk, S. Pastrana, and B. Collier, “A tight scrape: methodological approaches to cybercrime research data collection in adversarial environments,” 06 2020.

[11] C. Deshpande, “The best guide to know what is react [updated],” Nov 2022.

[12] V. Thangarajah, “Python current trend applications-an overview,” 10 2019.

[13] “Django introduction - learn web development: Mdn.”

[14] L. Liu and M. T. Özsu, eds., *Encyclopedia of Database Systems*. Springer Reference, New York: Springer, 2009.

[15] MySQL, “Mysql official cite,” 2022.

[16] S. Chacon and B. Straub, *Pro git: Everything you need to know about Git*. Apress, second ed., 2014.

[17] T. Point, “Git fast version control,” 2022.