ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Οργανωτής Ερευνητικών Δημοσιεύσεων Ατόμου/Ομάδας/Ιδρύματος

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΘΕΟΔΩΡΟΥ ΣΑΪΡΟΓΛΟΥ

Επιβλέπων : Αντώνιος Σιδηρόπουλος
Αναπληρωτής Καθηγητής, ΔΙ.ΠΑ.Ε

Θεσσαλονίκη, 30 Ιουνίου 2023

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Οργανωτής Ερευνητικών Δημοσιεύσεων Ατόμου/Ομάδας/Ιδρύματος

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΘΕΟΔΩΡΟΥ ΣΑΪΡΟΓΛΟΥ

Επιβλέπων : Αντώνιος Σιδηρόπουλος

Αναπληρωτής Καθηγητής ΔΙ.ΠΑ.Ε.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 30 Ιουνίου 2023.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Αντώνιος Σιδηρόπουλος Αναπληρωτής Καθηγητής ΔΙ.ΠΑ.Ε.	Στέφανος Ουγιάρογλου Επίκουρος Καθηγητής ΔΙ.ΠΑ.Ε.	Ευκλείδης Κεραμόπουλος Αναπληρωτής Καθηγητής ΔΙ.ΠΑ.Ε.

Θεσσαλονίκη, 30 Ιουνίου 20

Περίληψη

Στόχος της εργασίας είναι η δημιουργία ενός συστήματος το οποίο συλλέγει και οργανώνει με ημιαυτόματο τρόπο τις δημοσιεύσεις ενός Ερευνητή. Οι χρήστες του συστήματος θα είναι οι Ερευνητές. Η υλοποίηση αφορά μια κεντρική ιστοσελίδα, η οποία θα μπορεί να εμφανίζει συγκεντρωτικά στοιχεία και να δίνει την δυνατότητα στον χρήστη να "περιηγηθεί" στην βάση δεδομένων.

Η ιστοσελίδα είναι και η πύλη εισόδου για τους χρήστες-ερευνητές. Ο κάθε χρήστης δημιουργεί το προφίλ του, με βάση τα στοιχεία που έχει εισάγει στο κεντρικό σύστημα του ΔΙΠΑΕ (μέσω SSO), όπως ονοματεπώνυμο, τις ταυτότητές του: Orcid, ScopusID κτλ. Με χρήση εξωτερικών API συστημάτων συγκεντρώνεται η λίστα με τις δημοσιεύσεις του και καταχωρούνται στην βάση.

Τελικός στόχος είναι να χρησιμοποιείται το σύστημα από το Τμήμα Πληροφορικής και τα Ερευνητικά Εργαστήρια, αλλά και από ολόκληρο το Ίδρυμα ως ένα αξιόπιστο κεντρικό σύστημα καταγραφής δημοσιεύσεων.

Λέξεις Κλειδιά: οργανωτής, δημοσιεύσεις, εφαρμογή

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The aim of the work is to create a system which collects and organizes in a semi-automatic way the publications of a Researcher. The users of the system will be the Researchers. The implementation concerns a central website, which will be able to display data and enable the user to "browse" the database.

The website is also the entry portal for user-researchers. Each user creates his profile, based on the information he has entered in the IHU central system (via SSO), such as his name, his identities: Orcid, ScopusID, etc. Using external API systems, the list of his publications is gathered and are registered in the database.

The ultimate goal is the system to be used by the IT Department and Research Laboratories, but also by the entire Institution as a reliable central publication recording system.

Keywords: organizer, publications, webapp

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Οργανωτής Ερευνητικών Δημοσιεύσεων Ατόμου/Ομάδας/Ιδρύματος

Στις μέρες μας, υπάρχει πληθώρα από πλατφόρμες-πηγές, στις οποίες μπορεί ένας ερευνητής να αναζητήσει και να βρει το ερευνητικό-ακαδημαϊκό του έργο. Κάποιες από αυτές τις πλατφόρμες, συμπληρώνονται από τους ίδιους τους ερευνητές, ενώ άλλες πλατφόρμες συγκεντρώνουν ερευνητικό έργο από διάφορες πηγές.

Όπως γίνεται αντιληπτό, η πληθώρα πολλών και διαφορετικών πηγών για τις ίδιες πληροφορίες, δημοσιεύσεις στην περίπτωση μας, δύναται να δημιουργήσει περισσότερα προβλήματα από όσα μπορεί να λύσει. Αρχικά, τέτοιου είδους πλατφόρμες διαχειρίζονται μεγάλα ποσά δεδομένων, εγκυμονεί πάντα ο κίνδυνος διπλοεγγραφών, αποθήκευση λανθασμένων πληροφοριών, αποθήκευση πληροφοριών χαμηλής ποιότητας ή ακόμη και πολύ λίγων πληροφοριών για κάποιον ερευνητή ή δημοσίευσή του.

Συνεπώς, εάν κάποιος ερευνητής θέλει να «δείξει» το ερευνητικό του έργο, θα πρέπει να συγκεντρώσει υλικό από όλες τις διαθέσιμες πηγές και χειροκίνητα να “καθαρίσει” τις δημοσιεύσεις, να κρατήσει κάθε δημοσίευση μόνο μια φορά και μάλιστα να συγκεντρώσει όσες περισσότερες πληροφορίες υπάρχουν για κάθε δημοσίευση (σε περίπτωση διπλοτύπων με λιγότερες πληροφορίες).

1.2 Αντικείμενο διπλωματικής

Αντικείμενο της εργασίας είναι η δημιουργία ενός συστήματος το οποίο θα συλλέγει και θα οργανώνει με ημιαυτόματο τρόπο τις δημοσιεύσεις ενός ερευνητή. Οι χρήστες του συστήματος θα είναι οι ερευνητές.

Θα υπάρχει κεντρική ιστοσελίδα του συστήματος, η οποία θα μπορεί να εμφανίζει συγκεντρωτικά στοιχεία και να δίνει την δυνατότητα στον χρήστη να "περιηγηθεί" στην βάση δεδομένων. Επίσης, η ιστοσελίδα θα είναι και η πύλη εισόδου για τους χρήστες-ερευνητές. Ο κάθε χρήστης με βάση τα στοιχεία προφίλ που έχει καταχωρήσει στην κεντρική υπηρεσία ταυτοποίησης του ΜΠΗΣ/ΔΙΠΑΕ, όπως Ονοματεπώνυμο, τις ταυτότητές του: Orcid[9], ResearcherID, ScopusID θα μπορεί να δημιουργεί το λογαριασμό του. Με χρήση API κλήσεων θα "κατεβαίνει" η λίστα με τις δημοσιεύσεις του από δύο πηγές Scopus[8] και Orcid[9], θα μετασχηματίζονται σε μια γενική μορφή και αφού συγχωνευθούν σε μια λίστα θα καταχωρούνται στην βάση. Στο σύστημά μας θα έχουν πρόσβαση το προσωπικό του ΜΠΗΣ/ΔΙΠΑΕ μέσω της κεντρικής υπηρεσίας ταυτοποίησης.

Τελικός στόχος του συστήματος είναι να χρησιμοποιείται από το Τμήμα Πληροφορικής και τα Ερευνητικά Εργαστήρια, αλλά και από ολόκληρο το Ίδρυμα ως ένα αξιόπιστο κεντρικό σύστημα καταγραφής δημοσιεύσεων.

1.2.1 Συνεισφορά

Συνοψίζοντας, πέρα από την τελική υλοποίηση, η οποία αφορά μια διαδικτυακή εφαρμογή, απαιτούνται αρκετά προγενέστερα στάδια μέχρι το τελικό αποτέλεσμα.

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Αξιολογήσαμε διαφορετικές τεχνολογίες που θα βοηθούσαν στην υλοποίηση τόσο του Frontend όσο και του Backend.
2. Ερευνήσαμε κατάλληλες βιβλιοθήκες και στο Frontend και στο Backend που θα βοηθούσαν στην επίτευξη των επιθυμητών αποτελεσμάτων.
3. Μελετήσαμε τα εξωτερικά APIs του Scopus[8] και του Orcid[9], για να συγκεντρώσουμε την απαραίτητη πληροφορία.
4. Μορφοποιήσαμε τα συλλεγόμενα δεδομένα με τέτοιο τρόπο, ώστε να αποθηκεύονται οι πληροφορίες από το Scopus[8] και το Orcid[9] με την ίδια ακριβώς μορφή.
5. Υλοποιήσαμε αλγόριθμο συγχώνευσης δεδομένων έτσι ώστε τα δεδομένα από διαφορετικές πηγές-λίστες να μετατρέπονται σε μια και μόνο λίστα.

6. Πραγματοποιήσαμε διασύνδεση με τα συστήματα του ΜΠΗΣ/ΔΙΠΑΕ, έτσι ώστε ο κάθε χρήστης ερευνητής να δημιουργεί λογαριασμό και να συλλέγεται το ερευνητικό του έργο μέσα από μερικά clicks (SSO).
7. Κατασκευάσαμε διαδικτυακή εφαρμογή (Frontend και Backend), με διασύνδεση σε εσωτερικά APIs αλλά και εξωτερικά (IHU, SCOPUS[8], Orcid[9]). Η εφαρμογή είναι εύκολα προσβάσιμη και διαχειρίσιμη τόσο σε υπολογιστές, όσο και σε κινητά και ταμπλέτες.

1.3 Οργάνωση κειμένου

Πηγές σχετικές με το αντικείμενο της διπλωματικής παρουσιάζονται στο Κεφάλαιο 2 . Το Κεφάλαιο 3 συζητά τις τεχνολογίες που επιλέχθηκαν για την υλοποίηση της εφαρμογής καθώς και τις εξωτερικές πηγές δεδομένων που χρησιμοποιήθηκαν. Στο Κεφάλαιο 4 περιγράφεται όλη η αρχιτεκτονική της εφαρμογής. Στο Κεφάλαιο 5 εξετάζεται η βάση δεδομένων και ο τρόπος που στήθηκε. Στο Κεφάλαιο 6 αναλύεται ο μηχανισμός συγκέντρωσης δεδομένων (importer). Στο Κεφάλαιο 7 εξηγείται ενδελεχώς το Backend και το API που έχουν δημιουργηθεί. Στο Κεφάλαιο 8 παρουσιάζεται η δομή και η λειτουργία του Frontend. Στο Κεφάλαιο 9 γίνεται η μελέτη της ροής του χρήστη στην τελική εφαρμογή. Στο Κεφάλαιο 10 υπάρχει ο επίλογος.

2

Σχετικές εργασίες/τεχνολογίες

Υπάρχουν αρκετές τεχνολογίες και συστήματα που σχετίζονται με τη δική μας εργασία. Παρατίθενται πληροφορίες για τα σημαντικότερα σχετικά συστήματα, με λεπτομέρειες για το τι είναι το καθένα, με τι ασχολείται, τι πληροφορίες παρέχει, πως μπορεί κάποιος να έχει πρόσβαση σε αυτή.

2.1 Scopus

Το Scopus[8] είναι μια μεγάλη διεπιστημονική βάση δεδομένων επιστημονικής βιβλιογραφίας που διατηρείται από την Elsevier. Περιλαμβάνει άρθρα από περισσότερα από 20.000 περιοδικά με κριτές από διάφορους τομείς όπως η επιστήμη, η τεχνολογία, η ιατρική, οι κοινωνικές επιστήμες και οι τέχνες. Η βάση δεδομένων ενημερώνεται σε καθημερινή βάση και περιέχει περιλήψεις, παραπομπές και άρθρα πλήρους κειμένου. Το Scopus[8] παρέχει μια ολοκληρωμένη άποψη της επιστημονικής βιβλιογραφίας καλύπτοντας πηγές από όλο τον κόσμο και καλύπτοντας ένα ευρύ φάσμα χρονικών περιόδων.

Το Scopus ID είναι ένα μοναδικό αναγνωριστικό που εκχωρείται σε συγγραφείς επιστημονικών άρθρων που είναι ευρετηριασμένα στη βάση δεδομένων Scopus[8]. Το Scopus ID συνδέει όλα τα άρθρα ενός συγγραφέα μαζί, καθιστώντας ευκολότερη την παρακολούθηση της ερευνητικής τους απόδοσης και του αντίκτυπου. Το αναγνωριστικό Scopus[8] εκχωρείται με βάση το όνομα και τη σχέση του συγγραφέα και είναι μοναδικό για κάθε συγγραφέα.

Στη βάση του Scopus βρίσκονται μόνο ποιοτικά περιοδικά. Αυτός είναι και ο λόγος για τον οποίο το Scopus χρησιμοποιείται επίσημα και από πανεπιστήμια σε ολόκληρο τον κόσμο.

Το Scopus API είναι μια διεπαφή μέσω προγραμματισμού για τη βάση δεδομένων Scopus που επιτρέπει στους προγραμματιστές να έχουν πρόσβαση στα περιεχόμενα και τη λειτουργικότητά

της. Με το Scopus API, οι προγραμματιστές μπορούν να αναζητήσουν μέσω προγραμματισμού τη βάση δεδομένων, να ανακτήσουν άρθρα και να εξάγουν δεδομένα όπως πληροφορίες συντάκτη, αναφορές και περιλήψεις. Αυτό μπορεί να χρησιμοποιηθεί για τη δημιουργία προσαρμοσμένων εφαρμογών που αξιοποιούν τα δεδομένα στο Scopus[8] για διάφορους σκοπούς, όπως η παρακολούθηση του ερευνητικού αποτελέσματος ενός συγκεκριμένου συγγραφέα ή οργανισμού ή η παρακολούθηση του αντίκτυπου ενός συγκεκριμένου άρθρου ή συνόλου άρθρων.

Για να αποκτήσει κάποιος πρόσβαση στο Scopus API, πρέπει να εγγραφείτε για ένα κλειδί API στον ιστότοπο του Scopus[8]. Το κλειδί API παρέχει πρόσβαση στο API και επιτρέπει στον χρήστη να πραγματοποιεί ένα συγκεκριμένο πλήθος κλήσεων API ανά μήνα, ανάλογα με το πρόγραμμα που επιλέγει. Υπάρχουν διαθέσιμα διαφορετικά προγράμματα τιμολόγησης, που κυμαίνονται από μηδενικό έως και πολύ υψηλό κόστος.

Το Scopus[8] είναι μια ολοκληρωμένη βάση δεδομένων επιστημονικής βιβλιογραφίας που παρέχει μια ευρεία εικόνα του παγκόσμιου ερευνητικού έργου. Το Scopus ID και το Scopus API είναι δύο εργαλεία που βοηθούν στην αξιοποίηση αυτών των δεδομένων στο έπακρο, επιτρέποντας στους συγγραφείς να παρακολουθούν την ερευνητική τους παραγωγή και τον αντίκτυπό τους και στους προγραμματιστές να δημιουργούν προσαρμοσμένες εφαρμογές που αξιοποιούν τα δεδομένα αυτά.

2.2 Orcid

Το Orcid[9] (Open Researcher and Contributor ID) είναι ένας μη κερδοσκοπικός οργανισμός που παρέχει ένα μοναδικό αναγνωριστικό για ερευνητές και μελετητές. Δημιουργήθηκε για να βοηθήσει στην ταυτοποίηση της συνωνυμίας των ερευνητών και ήταν η πρώτη προσπάθεια μαζί με το ResearcherID. Το αναγνωριστικό, γνωστό ως Orcid ID, βοηθά να διασφαλιστεί ότι τα αποτελέσματα της έρευνας συνδέονται σωστά με το σωστό άτομο, μειώνοντας τις περιπτώσεις εσφαλμένης απόδοσης του συγγραφέα.

Το Orcid[9] λειτουργεί ως κεντρικό μητρώο για τους ερευνητές, παρέχοντάς τους ένα μέσο για να διατηρήσουν τα ερευνητικά τους προφίλ και να συνδέσουν την εργασία τους με διαφορετικά ερευνητικά συστήματα. Το μητρώο είναι ανοιχτό, διαφανές και καθοδηγείται από την κοινότητα, με συμμετοχή ερευνητών, εκδοτών, χρηματοδοτών και ακαδημαϊκών ιδρυμάτων σε όλο τον κόσμο. Πλέον οι περισσότεροι εκδοτικοί οίκοι, απαιτούν από τους συγγραφείς να αναφέρουν και το ORCID τους μαζί με τα υπόλοιπα στοιχεία τους.

Το Orcid API επιτρέπει στους χρήστες να ενσωματώνουν τη λειτουργικότητα Orcid[9] στα δικά τους συστήματα και ροές εργασίας. Το API παρέχει πρόσβαση στο μητρώο Orcid[9], συμπεριλαμβανομένων πληροφοριών όπως ονόματα ερευνητών, διευθύνσεις, βιογραφικά και αποτελέσματα έρευνας.

Για να αποκτήσει κάποιος πρόσβαση στο Orcid API, πρέπει πρώτα να δημιουργήσει έναν λογαριασμό Orcid[9] και να αποκτήσει ένα κλειδί API. Στη συνέχεια, μπορεί να χρησιμοποιήσει το API για να ανακτήσει πληροφορίες από το μητρώο Orcid[9], καθώς και για να ενημερώσει πληροφορίες στην εγγραφή Orcid[9]. Το API είναι δωρεάν για χρήση για μη εμπορικούς σκοπούς και υπάρχουν αρκετές διαθέσιμες επιλογές για εμπορική χρήση.

Το Orcid[9] παρέχει επίσης μια σειρά εργαλείων και πόρων για να βοηθήσει τους χρήστες να χρησιμοποιήσουν το API και να ενσωματώσουν το Orcid[9] στις εργασίες τους. Αυτοί οι πόροι περιλαμβάνουν τεκμηρίωση API, δείγμα κώδικα και υποστήριξη από την κοινότητα Orcid[9].

2.3 Google Scholar

Το Google Scholar είναι μια μηχανή αναζήτησης που ευρετηριάζει συγκεκριμένα επιστημονική βιβλιογραφία και ακαδημαϊκές πηγές. Παρέχει έναν τρόπο σε ερευνητές, φοιτητές και μελετητές να αναζητήσουν άρθρα, εργασίες συνεδρίων, διατριβές, βιβλία, προεκτυπώσεις και άλλες επιστημονικές εργασίες από όλους τους τομείς σπουδών.

Το Google Scholar λειτουργεί με ευρετηρίαση μεταδεδομένων και πλήρους κειμένου επιστημονικού υλικού που βρίσκεται στον Ιστό και επιτρέπει στους χρήστες να αναζητούν αυτήν τη βάση δεδομένων χρησιμοποιώντας λέξεις-κλειδιά και φράσεις. Τα αποτελέσματα αναζήτησης ταξινομούνται με βάση διάφορους παράγοντες, όπως η συνάφεια του περιεχομένου, η φήμη του συγγραφέα, ο τόπος δημοσίευσης και ο αριθμός των αναφορών. Αυτό δίνει τη δυνατότητα στους χρήστες να βρίσκουν γρήγορα τις πιο σχετικές και αξιόπιστες πηγές για την έρευνά τους.

Εκτός από τη μηχανή αναζήτησής του, το Google Scholar παρέχει επίσης μια σειρά από χαρακτηριστικά που το καθιστούν πολύτιμο εργαλείο για τους ερευνητές. Για παράδειγμα, οι χρήστες μπορούν να δημιουργήσουν μια προσωπική βιβλιοθήκη αποθηκευμένων άρθρων, να δημιουργήσουν ειδοποιήσεις για να ενημερώνονται για νέα έρευνα σε συγκεκριμένα θέματα και να παρακολουθούν τον αντίκτυπο της δουλειάς τους βλέποντας πόσο συχνά αναφέρεται από άλλους.

2.4 Aminer

Το Aminer είναι μια ολοκληρωμένη πλατφόρμα επιστημονικής έρευνας που παρέχει πρόσβαση σε μια εκτενή συλλογή επιστημονικών άρθρων, περιοδικών και άρθρων συνεδρίων. Λειτουργεί ως γέφυρα μεταξύ των ερευνητών και της επιστημονικής κοινότητας. Με την εκτεταμένη συλλογή επιστημονικών εργασιών του, το Aminer έχει γίνει μια πολύτιμη πηγή για όσους αναζητούν να συμβαδίζουν με τις τελευταίες εξελίξεις στον τομέα σπουδών τους.

Το Aminer API είναι ένα RESTful API που επιτρέπει στους προγραμματιστές να έχουν πρόσβαση στην τεράστια συλλογή επιστημονικών εργασιών και άλλων επιστημονικών εργασιών που φιλοξενούνται στο Aminer. Το API παρέχει πρόσβαση μέσω προγραμματισμού στην τεράστια συλλογή ερευνητικών εργασιών και δίνει τη δυνατότητα στους προγραμματιστές να δημιουργήσουν προσαρμοσμένες εφαρμογές που χρησιμοποιούν αυτά τα δεδομένα.

Το API Aminer λειτουργεί χρησιμοποιώντας πρωτόκολλο REST (Representational State Transfer). Αυτό επιτρέπει στους προγραμματιστές να αλληλοεπιδρούν με το API χρησιμοποιώντας τυπικά αιτήματα HTTP, όπως το GET και το POST. Το API υποστηρίζει επίσης μορφές δεδομένων όπως JSON και XML, διευκολύνοντας τους προγραμματιστές να εργαστούν με τα δεδομένα που επιστρέφονται από το API.

Για να αποκτήσει κάποιος πρόσβαση στο API του Aminer, πρέπει να δημιουργήσει έναν λογαριασμό στην πλατφόρμα Aminer. Μόλις αποκτήσει λογαριασμό, μπορεί να υποβάλει αίτηση για ένα κλειδί API, το οποίο θα επιτρέψει να ξεκινήσει να κάνετε αιτήματα API. Το κλειδί API λειτουργεί ως μοναδικό αναγνωριστικό για την εφαρμογή και χρησιμοποιείται για τον έλεγχο ταυτότητας κάθε αιτήματος που γίνεται στο API.

Το Aminer παρέχει πληθώρα πληροφοριών σχετικά με τον τρόπο χρήσης του API, συμπεριλαμβανομένης λεπτομερούς τεκμηρίωσης και δείγματος κώδικα. Αυτό διευκολύνει τους προγραμματιστές να αρχίσουν να χρησιμοποιούν το API, ακόμα κι αν δεν είναι εξοικειωμένοι με τα REST API ή τον τομέα επιστημονικής έρευνας.

2.5 DBLP

Η πλατφόρμα DBLP είναι μια δημόσια βάση δεδομένων που αφορά την περιοχή της επιστήμης των υπολογιστών και της πληροφορικής. Παρέχει μια εκτενή συλλογή από επιστημονικά άρθρα, συνεδρίες και βιβλία σχετικά με αυτό τον τομέα. Η βάση δεδομένων DBLP (Digital Bibliography & Library Project) χρησιμοποιείται ευρέως από ερευνητές, φοιτητές και

επαγγελματίες της πληροφορικής για να ενημερωθούν σχετικά με τις πρόσφατες εξελίξεις και έρευνες στον τομέα.

Η ιστοσελίδα προσφέρει πολλά οφέλη στους χρήστες της. Καταρχάς, παρέχει μια εύχρηστη διεπαφή αναζήτησης που επιτρέπει στους χρήστες να αναζητήσουν άρθρα και βιβλία με βάση διάφορα κριτήρια όπως οι συγγραφείς, οι λέξεις-κλειδιά και οι τίτλοι. Επιπλέον, παρέχει λεπτομερείς πληροφορίες για κάθε δημοσίευση, συμπεριλαμβανομένων περιλήψεων, πλήρων κειμένων και πληροφοριών συναφών με τους συγγραφείς. Η ιστοσελίδα παρέχει επίσης πρόσβαση σε πληροφορίες σχετικά με διάφορα συνέδρια και εκδηλώσεις σε όλο τον κόσμο. Αυτό επιτρέπει στους χρήστες να ενημερώνονται για τις τελευταίες εξελίξεις στον τομέα και να ανακαλύπτουν νέες ιδέες και τάσεις.

Όσον αφορά τη διαθεσιμότητα API για δεδομένα, η ιστοσελίδα DBLP παρέχει ένα δημόσιο API που επιτρέπει στους χρήστες να αποκτήσουν πρόσβαση στα δεδομένα της βάσης. Αυτό επιτρέπει στους προγραμματιστές να αναπτύξουν εφαρμογές που χρησιμοποιούν τις πληροφορίες της DBLP για αυτοματοποιημένη ανάκτηση δεδομένων, αναζήτηση και ανάλυση. Για να καταλάβει κανείς περισσότερα για την ιστοσελίδα, μπορεί επίσης να εξετάσει τη δομή των δεδομένων που παρέχονται και τους τρόπους με τους οποίους ανανεώνονται τα δεδομένα.

2.6 ResearchGate

Το ResearchGate αποτελεί μία πλατφόρμα κοινωνικής δικτύωσης που στοχεύει στην επιστημονική κοινότητα. Δημιουργήθηκε με σκοπό να ενώσει ερευνητές από διάφορα πεδία και να παρέχει πληροφορίες, εργαλεία και πόρους που ενισχύουν την επιστημονική έρευνα και συνεργασία

Το ResearchGate είναι μία πλατφόρμα που συνδέει ερευνητές από όλο τον κόσμο. Οι χρήστες μπορούν να δημιουργήσουν το προφίλ τους και να ανεβάσουν τα επιστημονικά τους άρθρα, διατριβές, βιβλία και άλλα ερευνητικά έργα. Μπορούν επίσης να αναζητήσουν και να συνδεθούν με άλλους ερευνητές, να συζητήσουν ιδέες και να συνεργαστούν σε επιστημονικά έργα. Επιπλέον, το ResearchGate παρέχει πληροφορίες για συνέδρια, εργαστήρια, και ειδικές εκδηλώσεις που αφορούν τον επιστημονικό κόσμο.

Οι χρήστες του ResearchGate έχουν πολλά οφέλη από τη χρήση της πλατφόρμας. Πρώτον, μπορούν να ανεβάσουν τα έργα τους και να τα κάνουν προσβάσιμα σε ευρεία επιστημονική κοινότητα. Αυτό διευκολύνει τη διάδοση των ερευνητικών αποτελεσμάτων και την ανταλλαγή γνώσεων. Δεύτερον, οι ερευνητές μπορούν να συνδεθούν με άλλους ερευνητές που ασχολούνται με παρόμοια θέματα και να συζητήσουν ιδέες, να λάβουν ανατροφοδότηση και

να δημιουργήσουν συνεργασίες. Τρίτον, οι χρήστες μπορούν να παρακολουθήσουν τις παραπομπές των έργων τους και να αξιολογήσουν την επιρροή της έρευνάς τους.

Ως πλατφόρμα κοινωνικής δικτύωσης, το ResearchGate παρέχει μία εφαρμογή προγραμματισμού διεπαφής (API) που επιτρέπει στους προγραμματιστές να αποκτήσουν πρόσβαση σε δεδομένα της πλατφόρμας. Αυτό επιτρέπει την ανάπτυξη εφαρμογών και εργαλείων που ωφελούν την επιστημονική κοινότητα. Μέσω του API, οι προγραμματιστές μπορούν να ανακτήσουν πληροφορίες για χρήστες, έργα, παραπομπές και άλλα στοιχεία που είναι διαθέσιμα στο ResearchGate.

Εκτός από τα βασικά χαρακτηριστικά που προσφέρει το ResearchGate, υπάρχουν και άλλες λειτουργίες που μπορούν να ωφελήσουν τους ερευνητές. Για παράδειγμα, οι χρήστες μπορούν να αναζητήσουν εργασίες και άλλα επιστημονικά έργα σε διάφορα πεδία, ενώ παράλληλα μπορούν να βρουν επιστημονικές ομάδες που συνδέονται με τα ενδιαφέροντά τους. Επιπλέον, το ResearchGate παρέχει μετρήσεις όπως ο δείκτης RG Score, που αξιολογεί την επιρροή ενός ερευνητή, καθώς και τη δυνατότητα αξιολόγησης και σχολιασμού έργων από άλλους χρήστες.

2.7 ACM Digital Library

Μια σημαντική πηγή για τους επιστήμονες και τους ερευνητές αποτελεί η πλατφόρμα ACM Digital Library. Η πλατφόρμα ACM DL είναι μια ψηφιακή βιβλιοθήκη που δημιουργήθηκε από το ACM (Association for Computing Machinery), μια παγκόσμια οργάνωση που απαρτίζεται από επιστήμονες της πληροφορικής και της πληροφοριακής επιστήμης. Σκοπός της ACM Digital Library είναι να παρέχει πρόσβαση σε επιστημονικά άρθρα, περιοδικά, πρακτικά συνεδρίων και άλλα περιεχόμενα που σχετίζονται με την πληροφορική.

Η ACM Digital Library παρέχει μια πληθώρα υπηρεσιών στους χρήστες της. Μπορούν να αναζητήσουν και να ανακτήσουν επιστημονικά άρθρα και περιοδικά, να περιηγηθούν σε συνέδρια και εκδηλώσεις που διοργανώνει το ACM, να προβάλλουν βίντεο και παρουσιάσεις, να αποθηκεύουν τα αγαπημένα τους άρθρα και να δημιουργούν ειδοποιήσεις για νέα περιεχόμενα.

Για τους προγραμματιστές και τους ερευνητές που επιθυμούν να αξιοποιήσουν τα δεδομένα της ACM Digital Library σε δικές τους εφαρμογές, υπάρχει διαθέσιμο ένα API. Το ACM Digital Library API παρέχει τη δυνατότητα αναζήτησης, ανάκτησης και προβολής περιεχομένου από την βιβλιοθήκη, επιτρέποντας έτσι την αυτοματοποίηση και την προηγμένη ανάλυση των δεδομένων.

Η ACM Digital Library οργανώνει τα περιεχόμενα της σε κατηγορίες και τομείς γνώσης, προσφέροντας έτσι μια δομημένη πλατφόρμα αναζήτησης. Οι χρήστες μπορούν να αναζητήσουν άρθρα με βάση τη θεματολογία, τον συγγραφέα και άλλα κριτήρια, διευκολύνοντας την ανεύρεση των επιθυμητών πληροφοριών.

Μια σημαντική πτυχή της ACM Digital Library είναι η δημιουργία μιας ενεργούς κοινότητας. Οι χρήστες μπορούν να σχολιάζουν, να συζητούν και να αλληλεπιδρούν με άλλους επιστήμονες, δημιουργώντας ένα περιβάλλον συνεργασίας και ανταλλαγής γνώσεων. Η ACM Digital Library προσφέρει διάφορα προνόμια στους συνδρομητές της, συμπεριλαμβανομένης της πρόσβασης σε όλα τα άρθρα και περιεχόμενα. Επιπλέον, πολλά ιδρύματα και πανεπιστήμια έχουν συνεργασίες με την ACM και παρέχουν πρόσβαση στην Digital Library στους φοιτητές και το προσωπικό τους.

2.8 Semantic Scholar

Το Semantic Scholar αντιπροσωπεύει μια πρωτοποριακή επιστημονική μηχανή αναζήτησης που διατίθεται δωρεάν στο διαδίκτυο. Βασιζόμενο σε προηγμένες τεχνολογίες ανάλυσης γλώσσας και μηχανικής μάθησης, το Semantic Scholar αποτελεί ένα εργαλείο απαραίτητο για ερευνητές και ακαδημαϊκούς σε όλον τον κόσμο. Πρόκειται για μια επιστημονική μηχανή αναζήτησης που αναπτύχθηκε από την ομάδα Επιστήμης των Δεδομένων στο Allen Institute for AI.

Ο σκοπός του Semantic Scholar είναι να προσφέρει ένα εύχρηστο εργαλείο για την αναζήτηση και ανακάλυψη επιστημονικών άρθρων. Η πλατφόρμα χρησιμοποιεί προηγμένους αλγορίθμους και τεχνικές επεξεργασίας γλώσσας για να αναλύει το περιεχόμενο των επιστημονικών άρθρων και να το οργανώνει με τρόπο που είναι ευανάγνωστος και κατανοητός για τον χρήστη.

Οι χρήστες του Semantic Scholar έχουν πρόσβαση σε ένα ευρύ φάσμα επιστημονικών δημοσιεύσεων από διάφορα επιστημονικά πεδία. Οι άνθρωποι μπορούν να αναζητήσουν άρθρα βάσει λέξεων-κλειδιών, συγγραφέων, περιοδικών και πεδίων έρευνας. Η πλατφόρμα παρέχει επίσης εκτεταμένες πληροφορίες για τα άρθρα, συμπεριλαμβανομένων των περιλήψεων, των παραπομπών και των συναφών άρθρων.

Το Semantic Scholar παρέχει ένα API που επιτρέπει στους προγραμματιστές να αποκτήσουν πρόσβαση στα δεδομένα της πλατφόρμας. Αυτό σημαίνει ότι μπορούν να χρησιμοποιήσουν τη λειτουργικότητα του Semantic Scholar στις δικές τους εφαρμογές ή ερευνητικές εργασίες. Το API παρέχει πληροφορίες για τα άρθρα, τους συγγραφείς, τις παραπομπές και άλλα μεταδεδομένα που βρίσκονται στη βάση δεδομένων του Semantic Scholar.

2.9 Web of Science

Το Web of Science είναι μια διαδικτυακή πλατφόρμα αναζήτησης και αναφοράς βιβλιογραφικών δεδομένων και επιστημονικών άρθρων. Παρέχει πρόσβαση σε μια ευρεία γκάμα ακαδημαϊκών περιοδικών, συνεδρίων, βιβλίων και άλλων επιστημονικών πηγών, και είναι ένα από τα πιο αξιόπιστα εργαλεία για την αξιολόγηση της επιστημονικής έρευνας.

Οι χρήστες του Web of Science έχουν πρόσβαση σε μια μεγάλη ποικιλία επιστημονικών περιοδικών και άρθρων που καλύπτουν πολλούς επιστημονικούς τομείς, όπως ιατρική, βιολογία, φυσικές επιστήμες, κοινωνικές επιστήμες, τεχνολογία και πολλούς άλλους. Οι χρήστες μπορούν να αναζητήσουν βάσει κλειδιών λέξεων, συγγραφέων, θεμάτων και άλλων παραμέτρων για να βρουν επιστημονικά άρθρα που είναι σχετικά με την έρευνά τους.

Μία από τις βασικές λειτουργίες του Web of Science είναι η δυνατότητα αναζήτησης με βάση τον αριθμό των αναφορών (citations) που έχει λάβει ένα άρθρο. Αυτό επιτρέπει στους χρήστες να εντοπίσουν τα πιο επιδραστικά και επικαιροποιημένα άρθρα σε ένα συγκεκριμένο πεδίο. Το Web of Science παρέχει επίσης μετρήσεις αναφοράς και επιδόσεων για συγγραφείς και περιοδικά, όπως το impact factor, το h-index και άλλα μετρικά.

Για τους ερευνητές και τους ακαδημαϊκούς, το Web of Science προσφέρει μια πληθώρα πληροφοριών και εργαλείων για τη διεξαγωγή προηγμένης έρευνας. Οι χρήστες μπορούν να παρακολουθούν την εξέλιξη της έρευνας σε συγκεκριμένα θέματα, να ανακαλύψουν νέες τάσεις και να ενημερώνονται για τα πιο πρόσφατα επιστημονικά ευρήματα. Επιπλέον, οι ερευνητές μπορούν να χρησιμοποιήσουν το Web of Science για να παράγουν αναφορές και παραθέσεις για τη δική τους έρευνα.

Το Web of Science παρέχει ένα δημόσιο API για την ανάκτηση δεδομένων. Το API επιτρέπει στους χρήστες να αναζητήσουν, να φιλτράρουν και να ανακτήσουν επιστημονικά άρθρα και άλλα δεδομένα απευθείας από την πλατφόρμα του Web of Science. Αυτή η δυνατότητα επιτρέπει την αυτοματοποίηση και την ευκολότερη πρόσβαση σε δεδομένα για ερευνητικούς και αναλυτικούς σκοπούς.

3

Τεχνολογικό - Θεωρητικό υπόβαθρο

Για να γίνει εφικτή η υλοποίηση της εργασίας χρησιμοποιήθηκαν διάφορες τεχνολογίες, βιβλιοθήκες κώδικα, τεχνικές και συλλογιστικές, οι οποίες θα αναλυθούν παρακάτω. Αρχικά θα εξεταστούν οι τεχνολογίες που επιλέχθηκαν για το Frontend, έπειτα για το Backend, αμέσως μετά θα δοθούν κάποιες πληροφορίες για τον Server που φιλοξενεί την εφαρμογή και τέλος θα παρουσιαστούν οι τρόποι πρόσβασης και κλήσης τόσο του Scopus API όσο και του Orcid[9] API.

3.1 Front End – React

Για την υλοποίηση του Frontend επιλέχθηκε η React[1]. Πρόκειται για μια JavaScript βιβλιοθήκη, που αναπτύχθηκε το 2011 από μηχανικούς λογισμικού του Facebook και χρησιμοποιείται ευρέως σε κολοσσούς της τεχνολογίας, όπως Netflix, PayPal, Instagram, Apple. Η React δίνει τη δυνατότητα στον προγραμματιστή:

- να δημιουργήσει ευέλικτα UIs (User Interfaces) και διαδραστικές web εφαρμογές
- να χρησιμοποιήσει ξεχωριστές τεχνικές για να επιτύχει τη μέγιστη ταχύτητα ακόμη και στα πιο απαιτητικά πληροφοριακά συστήματα,
- να διατηρήσει σταθερό τον κώδικα με την ίδια ποιότητα και απόδοση, ακόμη κι όταν τα δεδομένα του έργου project αλλάζουν.

Περισσότερες πληροφορίες σχετικά με τη React[1] μπορούν να βρεθούν στην επίσημη ιστοσελίδα της βιβλιοθήκης (https://reactjs.org/) [1]

Για να μπορέσει να γίνει η React[1] πιο ευέλικτη, χρησιμοποιήθηκαν περαιτέρω βιβλιοθήκες στο Frontend ή τεχνικές, καθεμιά για επιτύχει ένα συγκεκριμένο στόχο.

3.1.1 Material UI

Η Material UI[3] είναι μια βιβλιοθήκη της React[1] που προσφέρει μια ολοκληρωμένη σουίτα εργαλείων διεπαφής χρήστη για να βοηθήσει να δημιουργηθούν νέες λειτουργίες πιο γρήγορα. Με άλλα λόγια είναι μια UI βιβλιοθήκη που περιλαμβάνει έτοιμα κομμάτια κώδικα με στοιχεία όπως κουμπιά, φόρμες συμπλήρωσης, πεδία κειμένου, carousel φωτογραφιών και άλλα στοιχεία έτοιμα για χρήση. Περισσότερες πληροφορίες υπάρχουν στην επίσημη ιστοσελίδα (https://mui.com/)[3]

3.1.2 Routing

Η React[1] ως γνωστόν είναι ένα Single Page Application. Αυτό σημαίνει ότι δεν δημιουργεί διαφορετικές σελίδες αλλά όλο το περιεχόμενο παρουσιάζεται μέσω μιας και μόνο μιας HTML σελίδας. Για να μπορέσει λοιπόν να δοθεί η εντύπωση πολλών σελίδων αλλά και να δημιουργούνται διαφορετικές διευθύνσεις (URLs) χρησιμοποιήθηκε η βιβλιοθήκη της React[1], με όνομα React Router Dom[2]. Περισσότερες πληροφορίες υπάρχουν στην επίσημη ιστοσελίδα (https://v5.reactrouter.com/web/guides/quick-start)[2]

3.1.3 Session Storage

Για να μπορούν να κρατηθούν πληροφορίες και να ξαναχρησιμοποιηθούν στο Frontend ακόμα και μετά την ανανέωση της σελίδας από το χρήστη, οι πιο χρήσιμες πληροφορίες αποθηκεύονται στον περιηγητή ιστού και συγκεκριμένα στο session του χρήστη. Αυτή είναι μια τεχνική που επιτρέπει στο χρήστη να μην "χάνει" πληροφορίες που συγκέντρωσε στο παρασκήνιο με κάποια Web API κλήση, αν καταλάθος κλείσει την σελίδα (αλλά όχι τον περιηγητή ιστού) ή ανανεώσει τη σελίδα του. Αν κλείσει ο περιηγητής ιστού όμως, επειδή το session του χρήστη τερματίζεται, χάνεται και όλη η αποθηκευμένη πληροφορία.

Για παράδειγμα, ο χρήστης συνδέθηκε στην εφαρμογή μας, εάν κλείσει την εφαρμογή μας και ξαναμπεί (χωρίς να έχει κλείσει τον περιηγητή ιστού στο μεσοδιάστημα) ή αν ανανεώσει τη σελίδα, έχει αποθηκευτεί στο Session Storage ότι ο χρήστης έχει συνδεθεί καθώς και τα στοιχεία του, με αποτέλεσμα να μη χρειάζεται να επανασυνδεθεί αλλά να συνεχίσει απρόσκοπτα την περιήγησή του στη εφαρμογή μας. Αν ο χρήστης όμως κλείσει τελείως τον περιηγητή ιστού και προσπαθήσει να προσπελάσει την ιστοσελίδα μας, ο περιηγητής ιστού δεν τον αναγνωρίζει, αφού οι προηγούμενες πληροφορίες χάθηκαν με το session που τερματίστηκε, και ζητάει νέα σύνδεση από το χρήστη.

3.1.4 Encrypt – Decrypt

Πρόκειται για μια βιβλιοθήκη που επιτρέπει την κρυπτογράφηση και την αποκρυπτογράφηση κειμένων όπου και όπως χρειάζεται. Στην περίπτωσή μας, με το Session Storage, επειδή ζητείται ρητά η αποθήκευση πληροφοριών στον περιηγητή ιστού, κρυπτογραφούνται πρώτα για λόγους ασφαλείας, ενώ όταν το Frontend χρειάζεται να χρησιμοποιήσει αυτές τις πληροφορίες, αποκρυπτογραφούνται. Περισσότερες πληροφορίες υπάρχουν στην επίσημη ιστοσελίδα (https://jsh1400.github.io/string-encode-decode/)

3.2 Back End

Η αρχική προσέγγιση για την υλοποίηση του Backend ήταν η χρήση της Python, διότι παρέχει πολλές βιβλιοθήκες για επικοινωνία με συστήματα όπως το Scopus. Για υλοποίηση WebAPI σε Python χρησιμοποιούνται δύο frameworks το Django[6] και το FastAPI[5]. Έπειτα από ταυτόχρονη μελέτη και σύγκριση των δύο frameworks, προκρίθηκε η χρήση του FastAPI[5] ως η τελική λύση για το Backend της εφαρμογής.

3.2.1 Django

Το Django[6] είναι ένα δωρεάν και ανοικτού κώδικα web framework γραμμένο σε Python. Ονομάστηκε έτσι από τον Django Reinhardt, έναν κιθαρίστα της jazz manouche σκηνής από τη δεκαετία του 1930 μέχρι τις αρχές της δεκαετίας του 1950.

Το Django[6] είναι ένα Python web framework υψηλού επιπέδου, που ενθαρρύνει την ταχεία ανάπτυξη και τον καθαρό, ρεαλιστικό σχεδιασμό. Αφαιρεί μεγάλο μέρος της ταλαιπωρίας της ανάπτυξης κώδικα, ώστε να μπορεί να εστιάζει ο προγραμματιστής στη σύνταξη της εφαρμογής χωρίς να χρειάζεται να ανακαλύψει ξανά τον τροχό.

Τα βασικά χαρακτηριστικά είναι:

- Γρήγορη ανάπτυξη: Το Django[6] σχεδιάστηκε για να βοηθά τους προγραμματιστές να μεταφέρουν τις εφαρμογές από την ιδέα στην ολοκλήρωση όσο το δυνατόν γρηγορότερα.
- Ασφαλές: Το Django[6] λαμβάνει σοβαρά υπόψη την ασφάλεια και βοηθά τους προγραμματιστές να αποφεύγουν πολλά κοινά λάθη ασφαλείας.
- Επεκτάσιμο: Μερικοί από τους πιο πολυσύχναστους ιστοτόπους στον ιστό αξιοποιούν την ικανότητα του Django[6] να κλιμακώνεται γρήγορα και με ευελιξία.

Περισσότερες πληροφορίες στην επίσημη ιστοσελίδα (https://www.djangoproject.com/)[6]

3.2.2 FastAPI

Το FastAPI[5] είναι ένα σύγχρονο και γρήγορο (υψηλής απόδοσης) web framework για τη δημιουργία API με Python 3.7+.

Τα βασικά χαρακτηριστικά είναι:

- Γρήγορο: Πολύ υψηλή απόδοση, στο ίδιο επίπεδο με το NodeJS και το Go (χάρη στο Starlette και το Pydantic). Ένα από τα πιο γρήγορα διαθέσιμα Python framework.
- Γρήγορη κωδικοποίηση: Αύξηση της ταχύτητας ανάπτυξης κατά περίπου 200% έως 300%.
- Λιγότερα σφάλματα: Μείωση περίπου στο 40% των σφαλμάτων που προκαλούνται από ανθρώπους (προγραμματιστές).
- Εύκολο: Σχεδιασμένο για να είναι εύκολο στη χρήση και στην εκμάθηση. Λιγότερος χρόνος ανάγνωσης εγγράφων.
- Ισχυρό: Λήψη κώδικα έτοιμο για παραγωγή. Με αυτόματη διαδραστική τεκμηρίωση.
- Βασισμένο σε πρότυπα: Βασίζεται (και είναι πλήρως συμβατό με) τα ανοιχτά πρότυπα για API: OpenAPI (παλαιότερα γνωστά ως Swagger) και JSON Schema.

Περισσότερες πληροφορίες στην επίσημη ιστοσελίδα (https://fastapi.tiangolo.com/)[5]

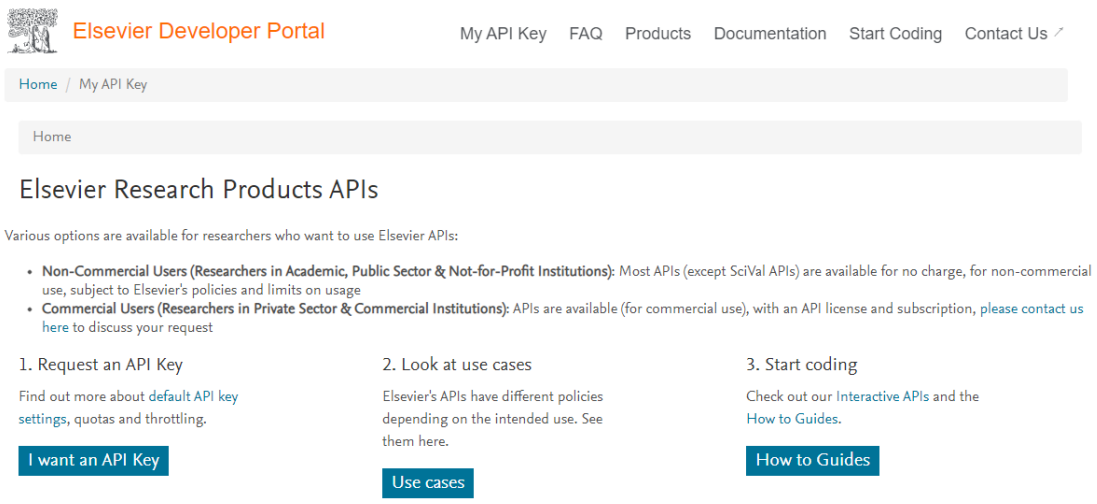
Παρά το γεγονός ότι και τα δύο frameworks μπορούν να βοηθήσουν στο χτίσιμο της εφαρμογής και είναι εξίσου σταθερά, το FastAPI[5] αποδεικνύεται πιο γρήγορο framework από το Django[6] και μάλιστα είναι γρηγορότερο όχι μόνο ως framework αλλά και στην εκμάθησή του.

3.3 Deployment – Apache Server

Για την διαδικτυακή παρουσία της εφαρμογής, επιλέχθηκε ένας Apache Server στο τομέα ΟΜΕΑ του τμήματος. Η έκδοση του Frontend εκτελείται ξεχωριστά από τα αρχεία του Backend οπότε θεωρητικά τα δύο μέρη της υλοποίησης, δημιουργήθηκαν και εκτελούνται ξεχωριστά.

3.4 Πρόσβαση στο Scopus API

Το Scopus[8] είναι διεθνής βιβλιογραφική βάση δεδομένων, η οποία πρωτοξεκίνησε το έτος 2004, και η οποία περιλαμβάνει περιλήψεις και παραπομπές για ακαδημαϊκά άρθρα από



Εικόνα 1 Περιβάλλον Scopus

έγκριτα επιστημονικά περιοδικά. Η συγκεκριμένη πλατφόρμα διαθέτει APIs για εξωτερική σύνδεση και προσπέλαση των δεδομένων τους.

Απαραίτητη προϋπόθεση για να μπορέσει κάποιος να συνδεθεί και να καλέσει APIs του Scopus[8] είναι να έχει δημιουργήσει ένα λογαριασμό ως Elsevier Developer. Έπειτα ο χρήστης πρέπει να παράγει ένα API key (https://dev.elsevier.com/index.jsp), που θα χρησιμοποιεί ως μοναδικό αναγνωριστικό στις κλήσεις που πραγματοποιεί, έτσι ώστε να αυθεντικοποιηθεί και να επιστρέψει η κλήση με όλα τα δεδομένα που ζητήθηκαν.

Ο λογαριασμός συνήθως απαιτεί την αυθεντικοποίηση μέσω κάποιου εκπαιδευτικού ιδρύματος, το οποίο έχει εξασφαλίσει πρόσβαση στην εν λόγω πλατφόρμα. ενώ οι κλήσεις στα APIs πραγματοποιούνται με επιτυχία ΜΟΝΟ εντός του δικτύου του ιδρύματος (εικονικού ή μη).

Εκτεταμένες πληροφορίες σχετικά με το Scopus API μπορούν να βρεθούν στην επίσημη σελίδα του ινστιτούτου (https://dev.elsevier.com/scopus.html).

Στη δική μας περίπτωση, πραγματοποιούνται GET κλήσεις στο URL:

```
"https://api.elsevier.com/content/search/scopus"
```

με παραμέτρους:

- το API key,
- query : au-id({το αναγνωριστικό του ερευνητή})
- view: Complete
- cursor: 0
- count: 25

Η εν λόγω κλήση επιστρέφει όλες τις διαθέσιμες πληροφορίες (view : Complete), του επιλεγμένου ερευνητή (query : au-id({το αναγνωριστικό του ερευνητή})), και επιστρέφει την πρώτη σελίδα αποτελεσμάτων (cursor: 0), μέχρι 25 αποτελέσματα η σελίδα (count: 25). Όπως είναι φυσιολογικό το σύστημα που δημιουργήσαμε, καλεί το API με τις κατάλληλες παραμέτρους όσες φορές χρειάζεται για να συλλέξει όλες τις πληροφορίες του ερευνητή, παρόλο που το API επιστρέφει μέχρι 25 δημοσιεύσεις τη φορά.

3.5 Πρόσβαση στο Orcid API

Το **Orcid[9]** (Open Researcher and Contributor ID) είναι μια πλατφόρμα που παρέχει έναν αλφαριθμητικό κωδικό για τον μοναδικό προσδιορισμό επιστημόνων και άλλων ακαδημαϊκών συγγραφέων, για την αναζήτηση της βιβλιογραφικής τους παραγωγής, όπως και άλλων στοιχείων που παρέχονται από τον συγγραφέα.

Το orcid.org διαθέτει και αυτό APIs για εξωτερική σύνδεση και προσπέλαση των δεδομένων τους. Όπως και στο Scopus[8], απαραίτητη προϋπόθεση για να μπορέσει κάποιος να συνδεθεί και να καλέσει APIs του Orcid[9] είναι να έχει δημιουργήσει ένα λογαριασμό στη σελίδα www.orcid.org και να παράγει ένα API key, που θα χρησιμοποιηθεί ως μοναδικό αναγνωριστικό στις κλήσεις που πραγματοποιούνται, έτσι ώστε να αυθεντικοποιηθεί ο χρήστης και να επιστρέψει η κλήση με όλα τα δεδομένα που ζητήθηκαν.

Σε αντίθεση με το Scopus[8], το Orcid[9] δεν απαιτεί κάποια σύνδεση με εκπαιδευτικό ίδρυμα για τη δημιουργία λογαριασμό, ούτε κάποια σύνδεση σε δίκτυο πανεπιστημιακού ιδρύματος (εικονικού ή μη) για επιτυχείς κλήσεις στα APIs τους. Αναλυτικές οδηγίες για όλη την παραπάνω διαδικασία μπορούν να βρεθούν εδώ: https://info.orcid.org/documentation/api-tutorials/api-tutorial-read-data-on-a-record/#easy-faq-2532

Εκτεταμένες πληροφορίες σχετικά με το ORCID API μπορούν να βρεθούν στην επίσημη σελίδα του οργανισμού (https://pub.orcid.org/v3.0/).

Στη δική μας περίπτωση για να λάβουμε όλες τις πληροφορίες για έναν ερευνητή απαιτούνται δύο στάδια.

Στο πρώτο στάδιο γίνεται μια GET κλήση στο URL:

```
"https://pub.orcid.org/v3.0/{userID}/works"
```

Όπου:

- σαν userID χρησιμοποιείται το αναγνωριστικό του ερευνητή που μας ενδιαφέρει,

- στα headers της κλήσης περιέχεται το API key που λάβαμε από το Orcid[9] σαν Bearer Authentication token.

Η παραπάνω κλήση επιστρέφει όλο το ερευνητικό έργο του ερευνητή επιστρέφοντας όμως λίγες πληροφορίες για κάθε δημοσίευση. Για παράδειγμα δεν επιστρέφονται οι co-authors. Για να μπορέσουμε να συλλέξουμε όσο το δυνατόν περισσότερες πληροφορίες για κάθε δημοσίευση, κρατάμε σε μια λίστα το αναγνωριστικό τους, και έπειτα γίνεται μια νέα κλήση που επιστρέφει περισσότερες πληροφορίες για κάθε δημοσίευση. Η νέα GET κλήση γίνεται στο URL:

```
"https://pub.orcid.org/v3.0/{userID}/works/{workIDs}"
```

Όπου:

- σαν userID χρησιμοποιείται το αναγνωριστικό του ερευνητή που μας ενδιαφέρει,
- σαν workIDs είναι η λίστα με όλα τα Ids των δημοσιεύσεων του ερευνητή,
- στα headers της κλήσης περιέχεται το API key που λάβαμε από το Orcid[9] σαν Bearer Authentication token.

Η λίστα των workIDs μπορεί να δεχτεί μέχρι 100 αναγνωριστικά τη φορά. Το σύστημα που έχουμε δημιουργήσει υπολογίζει κατάλληλα και σπάει τις δημοσιεύσεις του ερευνητή σε κατοστάδες και πραγματοποιεί όσες κλήσεις χρειάζονται, προκειμένου να συλλεχθεί όλη η απαραίτητη πληροφορία. Αρχιτεκτονική Εφαρμογής

Εδώ θα ακολουθήσει η ανάλυση του προβλήματος που διαπραγματεύεται η διπλωματική.

4

Αρχιτεκτονική Εφαρμογής

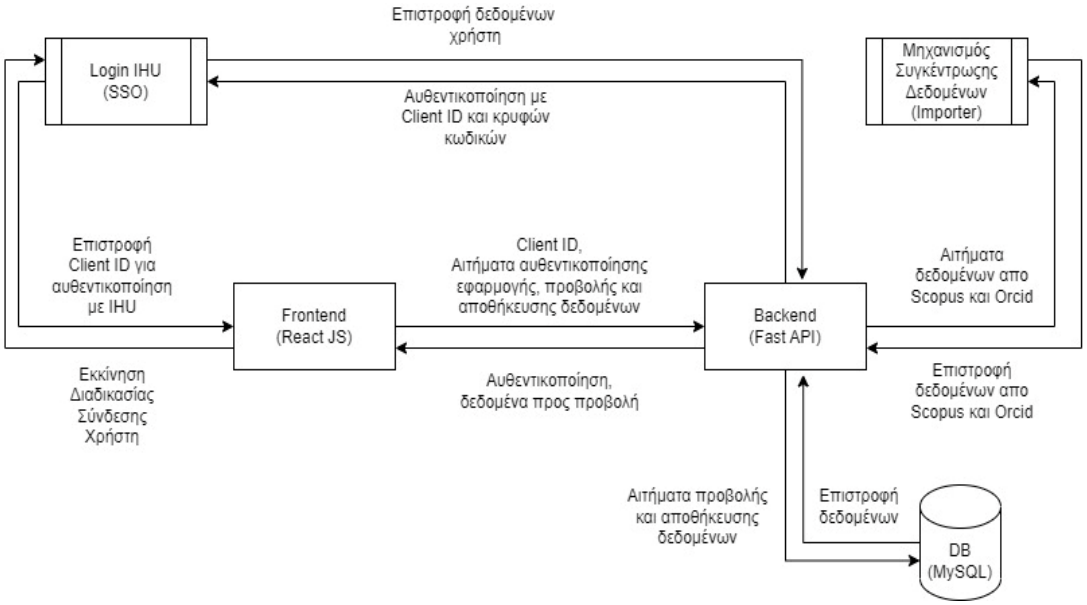
Στο κεφάλαιο αυτό, καταγράφονται η δομή του project, ο τρόπος που όλα τα στοιχεία της εφαρμογής αλληλοεπιδρούν μεταξύ τους, ο τρόπος με τον οποίο εκτελείται ο κώδικας και οι δομές που δημιουργήθηκαν.

4.1 Αρχιτεκτονική

Εδώ θα δούμε πως αλληλοεπιδρούν το Frontend, το Backend, η βάση και ο μηχανισμός συγκέντρωσης δεδομένων (importer) μεταξύ τους. Παρακάτω φαίνεται η λογική όλης της αρχιτεκτονικής που έχει υλοποιηθεί:

Όπως φαίνεται από το σχήμα παρακάτω, η εφαρμογή αποτελείται από πέντε επιμέρους στοιχεία, τα οποία συνεργάζονται αρμονικά μεταξύ τους. Τα στοιχεία αυτά είναι:

- Το Frontend,
- το Backend,
- τη βάση δεδομένων,
- τον μηχανισμό συγκέντρωσης δεδομένων (importer),
- και το μηχανισμό σύνδεσης χρήστη μέσα από τη δομή του πανεπιστημίου (Single Sign On).



Εικόνα 2 Αρχιτεκτονική Εφαρμογής

Σε πρώτη φάση ο χρήστης έρχεται σε επαφή με το Frontend, όπου του ζητείται να συνδεθεί στην εφαρμογή. Η σύνδεση πραγματοποιείται με τη βοήθεια δομής του πανεπιστημίου, η οποία επιστρέφει στο frontend όλα τα απαραίτητα στοιχεία του χρήστη. Έπειτα το Frontend στέλνει τα στοιχεία αυτά στο Backend μέσω κλήσης στο WebAPI, και το Backend αναλαμβάνει να επιβεβαιώσει την ταυτότητα του χρήστη που προσπαθεί να εισέλθει με τη βοήθεια δομής του πανεπιστημίου. Στην περίπτωση που επιβεβαιωθεί η ταυτότητα του χρήστη και εφόσον ο χρήστης δεν υπάρχει στη βάση δεδομένων, τότε προστίθεται στη βάση, συγκεντρώνονται όλα τα δεδομένα από εξωτερικές πηγές (Orcid, Scopus), αποθηκεύονται στη βάση δεδομένων και στέλνονται στο Frontend για προβολή. Αν ο χρήστης υπάρχει ήδη στη βάση δεδομένων, απλά επιστρέφονται όλα τα δεδομένα που έχουν συγκεντρωθεί.

Κάθε αλληλεπίδραση του χρήστη με την εφαρμογή απαιτεί την επικοινωνία και την ανταλλαγή δεδομένων μεταξύ Frontend, Backend και βάσης δεδομένων. Ο μηχανισμός συγκέντρωσης δεδομένων χρησιμοποιείται μόνο κατά την πρώτη σύνδεση-δημιουργία λογαριασμού των νέων χρηστών. Από τη άλλη ο μηχανισμός σύνδεσης μέσω δομής του πανεπιστημίου χρησιμοποιείται κάθε φορά που χρήστης συνδέεται στην εφαρμογή μας. Πρέπει να τονιστεί ότι από πλευράς Backend έχει υλοποιηθεί περαιτέρω μηχανισμός αυθεντικοποίησης, με το πρωτόκολλο Oauth2. Αυτό έχει ως αποτέλεσμα το Backend να στέλνει δεδομένα μόνο στους αυθεντικοποιημένους χρήστες που χρησιμοποιούν το Frontend.

Όπως γίνεται κατανοητό από το σχήμα το Frontend αλληλοεπιδρά άμεσα με το μηχανισμό σύνδεσης του πανεπιστημίου και με το Backend. Από την άλλη το Backend αλληλοεπιδρά

άμεσα με το Frontend, τη βάση δεδομένων και το μηχανισμό συγκέντρωσης δεδομένων (importer).

4.2 Εγκατάσταση

Η εγκατάσταση της εφαρμογής έχει γίνει σε server Apache του τομέα omea.iee.ihu.gr, ο οποίος έχει δοθεί από το ΔΙ.ΠΑ.Ε. Η λειτουργία της εφαρμογής σε περιβάλλον παραγωγής απαιτεί διαφορετική μεταχείριση για το Frontend και το Backend της εφαρμογής σε σύγκριση με το περιβάλλον ανάπτυξης.

4.2.1 Frontend

Για να μπορέσουμε να εκτελέσουμε το Frontend σε περιβάλλον παραγωγής, χρειάζεται να ανοίξουμε ένα τερματικό μέσα στο φάκελο FE του project μας.

Στο τερματικό τρέχουμε την εντολή npm run build, και δημιουργείται ένας φάκελος με όνομα build. Το μόνο που έχουμε να κάνουμε είναι να μεταφέρουμε τα περιεχόμενα αυτού του φακέλου στον server και να τα τοποθετήσουμε στο φάκελο με όνομα public_html.

4.2.2 Backend

Για να μπορέσουμε να εκτελέσουμε το Backend σε περιβάλλον παραγωγής, χρειάζεται να μεταφέρουμε το φάκελο BE από το repositoryμας στον server στον γονικό φάκελο.

Έπειτα σε ένα τερματικό στον server τρέχουμε την εντολή screen. Το Screen είναι ένας διαχειριστής παραθύρων πλήρους οθόνης που πολυπλέκει ένα φυσικό τερματικό μεταξύ πολλών διεργασιών, συνήθως διαδραστικών φλοιών. Έπειτα τρέχουμε στο τερματικό την εντολή uvicorn BE.thesis.main:app.

Απαραίτητη προϋπόθεση για να τρέξει σωστά η εφαρμογή στον server είναι να έχουν γίνει εγκατάσταση όλων των απαιτούμενων πακέτων και βιβλιοθηκών όπως ορίζονται στο αρχείο package.json για το Frontend και στο αρχείο requirement.txt για το Backend.

5

Βάση Δεδομένων

Στο κεφάλαιο αυτό, θα μελετηθεί η βάση δεδομένων, τα μοντέλα που χρησιμοποιήθηκαν για τη δημιουργία της και η χρησιμότητα του κάθε πίνακα.

Στη συγκεκριμένη εφαρμογή επιλεχθεί να υλοποιηθεί μια MySQL βάση δεδομένων. Όπως έχει αναφερθεί, στο Backend για τη δημιουργία του WebAPI χρησιμοποιήθηκε η τεχνολογία FastAPI, ένα framework της Python. Στην τεχνολογία αυτή η βάση δεδομένων δημιουργείται με τη βοήθεια μοντέλων. Το FastAPI υλοποιεί την ORM λογική ως προς τις βάσεις δεδομένων, χρησιμοποιώντας τη βιβλιοθήκη SQLAlchemy.

Η βάση δεδομένων της εφαρμογής αποτελείται από πέντε πίνακες:

- τον πίνακα users,
- τον πίνακα papers,
- τον πίνακα associaton_table για σύνδεση των πινάκων users και papers,
- τον πίνακα scopus,
- τον πίνακα orcid,

5.1 Πίνακας users

Στον πίνακα users αποθηκεύονται δεδομένα σχετικά με τον κάθε χρήστη που χρησιμοποιεί την εφαρμογή. Το μοντέλο που δημιουργεί τον πίνακα users φαίνεται παρακάτω:

```
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    uid = Column(String(50), unique=True, index=True)
    firstname = Column(String(255))
    lastname = Column(String(255))
    orc_id = Column(String(255), nullable=True)
    scopus_id = Column(String(255), nullable=True)
    email = Column(String(255))
    userPapers = relationship(
        "Papers",
        secondary=association_table,
        back_populates="creators")
```

Πίνακας 1 Μοντέλο για τη δημιουργία του πίνακα User στη βάση δεδομένων

Το παραπάνω μοντέλο, δημιουργεί έναν πίνακα στη βάση με όνομα users με τα εξής πεδία:

- id, μοναδικό αναγνωριστικό για κάθε χρήστη και το πρωτεύον κλειδί του πίνακα,
- uid, μοναδικό αναγνωριστικό για κάθε χρήστη σύμφωνα με τη δομή του πανεπιστημίου,
- firstname, το όνομα του χρήστη,
- lastname, το επώνυμο του χρήστη,
- orc_id, το μοναδικό αναγνωριστικό του χρήστη στην πλατφόρμα Orcid,
- scopus_id, το μοναδικό αναγνωριστικό του χρήστη στην πλατφόρμα Scopus,
- email, η διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη,
- userPapers, σύνδεση με τον πίνακα “Papers” με τη βοήθεια του πίνακα συσχέτισης (association table).

5.2 Πίνακας papers

Στον πίνακα papers αποθηκεύονται δεδομένα σχετικά με τις δημοσιεύσεις κάθε χρήστη. Είναι ο πίνακας τα δεδομένα του οποίου τροφοδοτούν το Frontend και προβάλλονται όλες οι απαραίτητες πληροφορίες σχετικά με τις δημοσιεύσεις του συνδεδεμένου χρήστη. Το μοντέλο που δημιουργεί τον πίνακα papers φαίνεται παρακάτω:

```
class Papers(Base):
      __tablename__ = 'papers'
      id = Column(Integer, primary_key=True, index=True)
      title = Column(String(600))
      publicationName = Column(String(600))
      description = Column(String(255))
      publicationType = Column(String(255))
      authors = Column(Text)
      link = Column(String(255))
      doi = Column(String(255))
      volume = Column(String(255))
      pageRange = Column(String(255))
      source = Column(String(255))
      publishedDate = Column(String(255))
      hidden = Column(Boolean, unique=False, default=True)
      creators = relationship(
      "User",
      secondary=association_table
      back_populates="userPapers")
```

Πίνακας 2 Μοντέλο για τη δημιουργία του πίνακα Paper στη βάση δεδομένων

Το παραπάνω μοντέλο, δημιουργεί έναν πίνακα στη βάση με όνομα papers με τα εξής πεδία:

- id, μοναδικό αναγνωριστικό για κάθε δημοσίευση και το πρωτεύον κλειδί του πίνακα,
- title, ο τίτλος της δημοσίευσης,
- publicationName, ο τίτλος της δημοσίευσης που εκδόθηκε,
- description, η περιγραφή της δημοσίευσης,
- publicationType, ο τύπος της δημοσίευσης,
- authors, οι συγγραφείς της δημοσίευσης,
- link, οι διαδικτυακοί σύνδεσμοι της δημοσίευσης,
- doi, μοναδικό αναγνωριστικό της δημοσίευσης σε παγκόσμιο επίπεδο,
- volume, η έκδοση του μέσου που εκδόθηκε η δημοσίευση,
- pageRange, το εύρος σελίδων στο μέσο που εκδόθηκε η δημοσίευση,
- source, η πηγή από την οποία συλλέχθηκε η δημοσίευση,
- publishedDate, η ημερομηνία έκδοσης της δημοσίευσης,
- hidden, εάν η δημοσίευση θα εμφανίζεται στον κύριο πίνακα του χρήστη,
- creators, σύνδεση με τον πίνακα “Users” με τη βοήθεια του πίνακα συσχέτισης (association table). Εκφράζει τον χρήστη για τον οποίο δημιουργήθηκε η εγγραφή

5.3 Πίνακας association_table

Στον πίνακα association_table αποθηκεύονται τα ids των χρηστών με τα ids των δημοσιεύσεών τους. Πρόκειται στην ουσία, για πίνακα που συνδέει τους χρήστες με τις δημοσιεύσεις τους. Το μοντέλο που δημιουργεί τον πίνακα association_table φαίνεται παρακάτω:

Το παραπάνω μοντέλο, δημιουργεί έναν πίνακα στη βάση με όνομα association. Πρόκειται στην ουσία για έναν πίνακα συσχέτισης μεταξύ των πινάκων Users και Papers, όπου συσχετίζονται οι δημοσιεύσεις με τους χρήστες/δημιουργούς τους.

```
association_table = Table('association', Base.metadata,
                          Column('users_id', ForeignKey(
                              'users.id'), primary_key=True),
                          Column('papers_id', ForeignKey(
                              'papers.id'), primary_key=True)
                          )
```

Πίνακας 3 Μοντέλο συσχέτισης των πινάκων User και Paper

5.4 Πίνακας scopus

Στον πίνακα scopus αποθηκεύονται οι δημοσιεύσεις των χρηστών που προήλθαν από το Scopus. Αφού συγκεντρωθούν και μορφοποιηθούν οι δημοσιεύσεις των χρηστών από το Scopus API αποθηκεύονται στον πίνακα scopus, έτσι ώστε σε επόμενο στάδιο να μπορέσουν να συγχωνευθούν με τον πίνακα papers. Το μοντέλο που δημιουργεί τον πίνακα scopus φαίνεται παρακάτω:

```
class ScopusDB(Base):
    __tablename__ = 'scopus'
    id = Column(Integer, primary_key=True, index=True)
    title = Column(String(600))
    publicationName = Column(String(600))
    description = Column(String(255))
    publicationType = Column(String(255))
    authors = Column(Text)
    link = Column(String(255))
    doi = Column(String(255))
    volume = Column(String(255))
    pageRange = Column(String(255))
    publishedDate = Column(String(255))
```

Πίνακας 4 Μοντέλο για τη δημιουργία του πίνακα Scopus στη βάση δεδομένων

Το παραπάνω μοντέλο, δημιουργεί έναν πίνακα στη βάση με όνομα scopus με τα εξής πεδία:

- id, μοναδικό αναγνωριστικό για κάθε δημοσίευση και το πρωτεύον κλειδί του πίνακα,
- title, ο τίτλος της δημοσίευσης,
- publicationName, ο τίτλος της δημοσίευσης που εκδόθηκε,

- description, η περιγραφή της δημοσίευσης,
- publicationType, ο τύπος της δημοσίευσης,
- authors, οι συγγραφείς της δημοσίευσης,
- link, οι διαδικτυακοί σύνδεσμοι της δημοσίευσης,
- doi, μοναδικό αναγνωριστικό της δημοσίευσης σε παγκόσμιο επίπεδο,
- volume, η έκδοση του μέσου που εκδόθηκε η δημοσίευση,
- pageRange, το εύρος σελίδων στο μέσο που εκδόθηκε η δημοσίευση,
- publishedDate, η ημερομηνία έκδοσης της δημοσίευσης,

5.5 Πίνακας orcid

Στον πίνακα orcid αποθηκεύονται οι δημοσιεύσεις των χρηστών που προήλθαν από το Orcid. Αφού συγκεντρωθούν και μορφοποιηθούν οι δημοσιεύσεις των χρηστών από το Orcid API αποθηκεύονται στον πίνακα orcid, έτσι ώστε σε επόμενο στάδιο να μπορέσουν να συγχωνευθούν με τον πίνακα papers. Το μοντέλο που δημιουργεί τον πίνακα orcid φαίνεται παρακάτω:

```
class OrcidDB(Base):
    __tablename__ = 'orcid'
    id = Column(Integer, primary_key=True, index=True)
    title = Column(String(600))
    publicationName = Column(String(600))
    description = Column(String(255))
    publicationType = Column(String(255))
    authors = Column(Text)
    link = Column(String(255))
    doi = Column(String(255))
    volume = Column(String(255))
    pageRange = Column(String(255))
    publishedDate = Column(String(255))
```

Πίνακας 5 Μοντέλο για τη δημιουργία του πίνακα Orcid στη βάση δεδομένων

Το παραπάνω μοντέλο, δημιουργεί έναν πίνακα στη βάση με όνομα orcid με τα εξής πεδία:

- id, μοναδικό αναγνωριστικό για κάθε δημοσίευση και το πρωτεύον κλειδί του πίνακα,
- title, ο τίτλος της δημοσίευσης,
- publicationName, ο τίτλος της δημοσίευσης που εκδόθηκε,
- description, η περιγραφή της δημοσίευσης,
- publicationType, ο τύπος της δημοσίευσης,
- authors, οι συγγραφείς της δημοσίευσης,

- link, οι διαδικτυακοί σύνδεσμοι της δημοσίευσης,
- doi, μοναδικό αναγνωριστικό της δημοσίευσης σε παγκόσμιο επίπεδο,
- volume, η έκδοση του μέσου που εκδόθηκε η δημοσίευση,
- pageRange, το εύρος σελίδων στο μέσο που εκδόθηκε η δημοσίευση,
- publishedDate, η ημερομηνία έκδοσης της δημοσίευσης,

6

Μηχανισμός Συγκέντρωσης Δεδομένων (Importer)

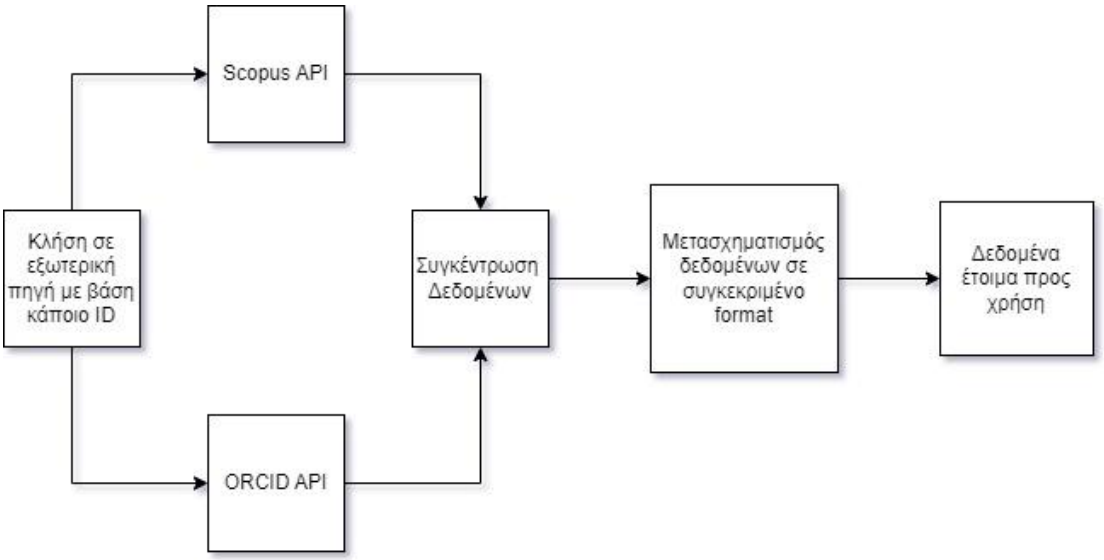
Στο κεφάλαιο αυτό, αναλύονται όλοι οι τρόποι και οι τεχνικές που έχουν χρησιμοποιηθεί με σκοπό τη συγκέντρωση δεδομένων από εξωτερικές πηγές και συγκεκριμένα το Scopus και το Orcid. Πέρα από αυτό αναλύεται και ο τρόπος με τον οποίο τα δεδομένα αυτά, καθαρίζονται και μετασχηματίζονται στην ίδια μορφή, από όποια πηγή και αν λήφθηκαν, έτσι ώστε να εισαχθούν στη βάση δεδομένων.

6.1 Συναρτήσεις ανάγνωσης και επεξεργασίας δεδομένων από Scopus και Orcid

Ο μηχανισμός κλήσης των εξωτερικών APIs μας βοηθάει να συγκεντρώσουμε όλα τα απαραίτητα δεδομένα (ακαδημαϊκό έργο) για έναν χρήστη, τα οποία αφού μορφοποιηθούν σε συγκεκριμένη μορφή θα πρέπει να εξεταστεί αν πρέπει να ενημερώσουν τη βάση και συγκεκριμένα τον πίνακα papers.

Επειδή έχουμε να συνδεθούμε με διαφορετικά εξωτερικά APIs, αυτά των Scopus και Orcid, έχουν δημιουργηθεί διαφορετικές συναρτήσεις για τη λήψη δεδομένων για την κάθε πηγή, σύμφωνα πάντα με τον τρόπο που προτείνεται από τον δημιουργό των εξωτερικών WebAPI.

Παρακάτω θα αναλυθεί όλη η λογική πίσω από τον μηχανισμό κλήσης εξωτερικών APIs και μορφοποίησης των δεδομένων που συλλέχθηκαν και θα εξηγηθούν ενδελεχώς όλες οι συναρτήσεις που υλοποιούν την λογική αυτή.



Εικόνα 3 Μηχανισμός ανάγνωσης και επεξεργασίας δεδομένων

Η λογική συλλογής και μορφοποίησης δεδομένων φαίνεται στην παρακάτω εικόνα:

Όπως φαίνεται παραπάνω, αρχικά επιτυγχάνεται η επικοινωνία με τις εξωτερικές πηγές και πραγματοποιούνται κλήσεις προς αυτές (Scopus, Orcid). Αφού συγκεντρωθούν τα δεδομένα από κάθε πηγή, μορφοποιούνται σε συγκεκριμένο format, όπως αυτό που χρησιμοποιείται για τις δημοσιεύσεις της εφαρμογής και έχει και ο πίνακας papers της βάσης δεδομένων. Αφού ολοκληρωθεί και η μορφοποίηση, τα δεδομένα είναι έτοιμα για κάθε χρήση. Στην περίπτωσή μας, εκτελείται ο μηχανισμός συγχώνευσης για να δούμε ποια από τις νέες δημοσιεύσεις που συγκεντρώθηκαν, πρέπει να προστεθούν ή να ενημερώσουν εγγραφές του πίνακα papers στη βάση δεδομένων και προβάλλεται στους χρήστες μέσω του Frontend.

Έπειτα θα εξηγηθούν ενδελεχώς όλες οι συναρτήσεις που υλοποιούν την λογική αυτή.

Πρόκειται για τις συναρτήσεις που βρίσκονται στον φάκελο BE/externalCalls, και πιο συγκεκριμένα συναρτήσεις στο αρχείο:

- externalAPIS.py, που περιέχει συναρτήσεις πρόσβασης στα APIs των Orcid και Scopus,

6.1.1 Ανάγνωση δεδομένων από το Scopus

Η συνάρτηση get_user_data_SCOPUS έχει σκοπό να συλλέξει όλο το ακαδημαϊκό έργο για ένα δοθέν scopus id – χρήστη. Επειδή το Scopus API είναι οργανωμένο με σελιδοποίηση, έχει δημιουργηθεί μηχανισμός πραγματοποιεί όσες διαδοχικές κλήσεις χρειάζονται για να συλλεχθεί όλο το ακαδημαϊκό έργο του συγκεκριμένου scopus id.

```
def get_user_data_SCOPUS(userID):
    print("SCOPUS API")
currentCursor = '*'
    previousCursor = 1
    docData = []
    docDataInitial = []
    while (previousCursor != currentCursor):

        parameters = {"apikey": scopus_key,
                      "query": "au-id({})".format(userID),
                      "view": "COMPLETE",
                      "cursor": currentCursor,
                      "count": 25,
                      }
        response = requests.get(scopus_url, params=parameters)
        if response.status_code == 200:
            response_JSON = response.json()
        # cursor for json pagination
            currentCursor = response_JSON["search-results"]["cur-
sor"]["@next"]
            previousCursor = response_JSON["search-results"]["cur-
sor"]["@current"]
            if currentCursor != previousCursor:

                res = response_JSON['search-results']['entry']
                authors = ''
                for index in range(len(res)):
                    entr = {}
                    docDataInitial.append(res[index])

                    if "author" in res[index].keys():
                        for idx in range(len(res[index]['author'])):
                            if idx == 0:
                                authors = res[index]['au-
thor'][idx]['authname']
                            else:
                                authors = authors + ", " + \
                                    res[index]['author'][idx]['authname']
```

Συνάρτηση 1 Ανάγνωση δεδομένων από το Scopus

Όρισμα:

- userID : μοναδικό αναγνωριστικό χρήστη.

Επιστροφή:

- docData: πίνακας όπου κάθε εγγραφή περιέχει το ακαδημαϊκό έργο του επιστήμονα με το δοθέν αναγνωριστικό προέρχονται από το Scopus.
-

Αλγόριθμος:

- Αρχικοποίηση μεταβλητών currentCursor με '*', previousCursor με 1, docData με κενή λίστα και docDataInitial με κενή λίστα.

- Βρόχος while που επαναλαμβάνεται όσο ο previousCursor δεν είναι ίσος με τον currentCursor,
- Εντός του βρόχου while, ορισμός παραμέτρων που θα χρησιμοποιηθούν στην API κλήση στο Scopus.
- Αίτημα GET στο καθορισμένο url API με τις παραμέτρους που ορίστηκαν παραπάνω.
- Αν η κλήση είναι επιτυχημένη (status=200), η απάντηση του API μετασχηματίζεται σε μορφή JSON και αποθηκεύεται η σχετική πληροφορία στις μεταβλητές, docData και docDataInitial.
- Ενημερώνονται οι τιμές των currentCursor και previousCursor από την απάντηση του API.
- Εάν η κλήση αποτύχει (status διαφορετικού του 200), εκτυπώνεται ένα μήνυμα λάθος και επιστρέφεται None.
- Τέλος του while βρόχου.
- Επιστροφή του πίνακα docData.

6.1.2 Ανάγνωση και επεξεργασία δεδομένων από το Orcid

Η συνάρτηση get_user_data_ORCID έχει σκοπό να συλλέξει όλο το ακαδημαϊκό έργο για ένα δοθέν orcid – χρήστη. Επειδή το Orcid API επιστρέφει όλες τις βασικές πληροφορίες για το ακαδημαϊκό έργο του δοθέντος orcid και παραλείπονται κάποιες (πχ η λίστα με τους συγγραφείς), έχει αναπτυχθεί μια επιπρόσθετη συνάρτηση που ονομάζεται get_user_data_detailed_ORCID. Η συνάρτηση αυτή συλλέγει όλες τις πληροφορίες που χρειάζονται για το ακαδημαϊκό έργο του orcid και έχουν αναπτυχθεί επιπρόσθετες συναρτήσεις επεξεργασίας των δεδομένων που συλλέγονται από αυτή την συνάρτηση.

Η συνάρτηση get_user_data_ORCID συλλέγει όλες τις βασικές πληροφορίες για το ακαδημαϊκό έργο του δοθέντος orcid και καταγράφει όλα τα αναγνωριστικά (ids) που χρειάζονται για να δοθούν ως όρισμα στην συνάρτηση get_user_data_detailed_ORCID, η οποία πραγματοποιεί νέες κλήσεις και συλλέγει όλα τα απαραίτητα δεδομένα που παραλείπονται από τη συνάρτηση get_user_data_ORCID.

```
def get_user_data_ORCID(userID):
    response = requests.get(f"{orcid_url}/{userID}/works", headers={
                            "Authorization": "Bearer {}".format(orcid_key),
"Accept": "application/vnd.orcid+json"})
    if response.status_code == 200:
        print("ORCID  API")
        response_JSON = response.json()
        tempWork = []
        for index in range(len(response_JSON['group'])):
            tempWork.append(response_JSON['group'][index]
                            ['work-summary'][0]['put-code'])
        if (len(tempWork) > 0):
            if (len(tempWork) > 100):
                splitedSize = 100  # items per chart
                tempWork_splited = [tempWork[x:x+splitedSize]
                                    for x in range(0, len(tempWork), split-
edSize)]
                finalData = []
                for item in tempWork_splited:
                    workData = ",".join(map(str, item))
                    finalData = finalData + \
                        get_user_data_detailed_ORCID(userID, workData)
                # with open("test.json", "w") as outfile:
                #     json.dump(finalData, outfile)
                return finalData
            else:
                work = ",".join(map(str, tempWork))
                tempOrcid = get_user_data_detailed_ORCID(userID, work)
                # Create files for testing
                # with open("Orcid_Edited.json", "w") as outfile:
                #     json.dump(tempOrcid, outfile)
                return tempOrcid
        else:
            return None
    else:
        print(
            f"There's a {response.status_code} error with your request")
        return None
```

Συνάρτηση 2 Ανάγνωση και επεξεργασία δεδομένων από το Orcid

```
                    entr['title'] = res[index]["dc:title"] if "dc:title" in
res[index].keys(
                    ) else None
                    entr['publicationName'] = res[index]["prism:publication-
Name"] if "prism:publicationName" in res[index].keys(
                    ) else None
                    entr['description'] = res[index]["subtypeDescription"]
if "subtypeDescription" in res[index].keys(
                    ) else None
                    entr['publicationType'] = res[index]["prism:aggregation-
Type"] if "prism:aggregationType" in res[index].keys(
                    ) else None
                    entr['authors'] = authors if "author" in res[in-
dex].keys(
                    ) else None
                    entr['link'] = res[index]["link"][2]['@href'] if res[in-
dex]["link"][2] else None

                    entr['doi'] = res[index]["prism:doi"] if "prism:doi" in
res[index].keys(
                    ) else None
                    entr['volume'] = res[index]["prism:volume"] if
"prism:volume" in res[index].keys(
                    ) else None
                    entr['pageRange'] = res[index]["prism:pageRange"] if
"prism:pageRange" in res[index].keys(
                    ) else None
                    entr['publishedDate'] = res[index]["prism:coverDate"] if
"prism:coverDate" in res[index].keys(
                    ) else None
                    docData.append(entr)
        else:
            print(
                f"There's a {response.status_code} error with your request")
            return None
    return docData
```

Συνάρτηση 3 Ανάγνωση και επεξεργασία δεδομένων από το Orcid

Όρισμα:

- userID: μοναδικό αναγνωριστικό χρήστη.

Επιστροφή:

- tempOrcid: πίνακας όπου κάθε εγγραφή περιέχει το ακαδημαϊκό έργο του επιστήμονα με το δοθέν αναγνωριστικό και προέρχονται από το Orcid.

Αλγόριθμος:

- Κλήση στο API του Orcid με τις κατάλληλες παραμέτρους,
- Αν επιτύχει η κλήση (status=200), η απάντηση μετατρέπεται σε μορφή JSON και εξάγεται η μεταβλητή put-code για κάθε μια εγγραφή της απάντησης του API.
- Καλείται έπειτα η συνάρτηση "get_user_data_detailed_ORCID" στην οποία περνιούνται ως ορίσματα το userID και τα work codes. Η συνάρτηση επιστρέφει τα ενημερωμένα και πιο λεπτομερή δεδομένα σχετικά με το ακαδημαϊκό έργο του χρήστη, προερχόμενο από την πηγή Orcid.

- Αν η κλήση αποτύχει (status διαφορετικό του 200), η συνάρτηση τυπώνει μήνυμα λάθους και επιστρέφει None.

Η συνάρτηση get_user_data_detailed_ORCID καλείται από τη συνάρτηση get_user_data_ORCID με σκοπό να συλλέξει πληροφορίες σε μεγαλύτερο βάθος. Στην συνάρτηση get_user_data_detailed_ORCID καλούνται άλλες συναρτήσεις που φαίνονται παρακάτω για να μορφοποιηθούν τα δεδομένα ώστε να είναι έτοιμα να εισαχθούν στην βάση δεδομένων.

```
def get_user_data_detailed_ORCID(userID, workIDs):
    res = requests.get(f"{orcid_url}/{userID}/works/{workIDs}", headers={
        "Authorization": "Bearer {}".format(orcid_key), "Accept": "applica-
tion/vnd.orcid+json"})
    if res.status_code == 200:
        res_JSON = res.json()
        res = res_JSON['bulk']
        docData = []
        for index in range(len(res)):
            data = res[index]["work"]
            entr = {}
            citation = citationHelper(
           a['citation'] else None)
            entr['title'] = data["title"]["title"]["value"] if data["title"]
else None
            entr['publicationName'] = data["journal-title"]["value"] if
data["journal-title"] else None
            entr['description'] = removeDashAndCapitalize(data["type"]) if
"type" in data.keys(
            ) else None
            entr['publicationType'] = removeDashAndCapitalize(data["type"])
if "type" in data.keys(
            ) else None
            entr['authors'] = authorsList(data["contributors"]["contribu-
tor"])
            entr['link'] = data["url"]["value"] if data["url"] else None
            entr['doi'] = data["external-ids"]["external-id"][0]['external-
id-value'] if data["external-ids"] and data["external-ids"]["external-
id"][0]['external-id-type'] == "doi" else None
            entr['volume'] = citation['volume'] if citation and "volume" in
citation.keys(
            ) else None
            entr['pageRange'] = citation['pages'] if citation and "pages" in
citation.keys(
            ) else None
            entr['publishedDate'] = dateHelper(
                data["publication-date"])
            docData.append(entr)
        return docData     data['citation']['citation-value'] if dat
    else:
        print(
            f"There's a {res.status_code} error with your request")
        return None
```

Συνάρτηση 4 Βοηθητική συνάρτηση διαχείρισης δεδομένων από το Orcid

Ορίσματα:

- userID: μοναδικό αναγνωριστικό χρήστη, workIDs = τα μοναδικά αναγνωριστικά για κάθε ακαδημαϊκό έργο του χρήστης ως προς την πηγή Orcid.

Επιστροφή:

- docData: πίνακας όπου κάθε εγγραφή περιέχει το ακαδημαϊκό έργο του επιστήμονα με το δοθέν αναγνωριστικό και προέρχονται από το Orcid .

Αλγόριθμος:

- Κλήση στο Orcid με όλες τις απαραίτητες παραμέτρους για λήψη του ακαδημαϊκού έργου του χρήστη με το δοθέν userID,
- Αν η κλήση επιτύχει (status=200) αρχικοποιείται ένας κενός πίνακας με όνομα"docData", έπειτα τρέχει μια loop όσο το μέγεθος του πίνακα που επιστρέφεται από την κλήση.
- Για κάθε επανάληψη δημιουργείται ένα κενό dictionary me όνομα "entr",
- έπειτα εξάγεται η απαραίτητη πληροφορία από το επιστρεφόμενο αποτέλεσμα της κλήσης όπως τίτλος, όνομα δημοσίευσης κτλ.
- Καλούνται κάποιες επιπλέον βοηθητικές συναρτήσεις, οι "citationHelper", “dateHelper”, “removeDashAndCapitalize”, "authorsList", που δίνουν στα δεδομένα μας την επιθυμητή μορφή,
- Όλη η εξαγόμενη πληροφορία αποθηκεύεται στο dictionary "entr" και το dictionary αυτό αποθηκεύεται στον πίνακα "docData".
- Αφού τελειώσουν όλοι οι βρόχοι επανάληψης επιστρέφεται ο πίνακας "docData".
- Αν η κλήση στο API δεν είναι επιτυχημένη (status διάφορο του 200), τυπώνεται μήνυμα λάθους και επιστρέφεται None.

Η συνάρτηση citationHelper καλείται από τη συνάρτηση get_user_data_detailed_ORCID και σκοπό έχει να μετασχηματίσει πληροφορίες σχετικές με μια συγκεκριμένη δημοσίευση.

```
def citationHelper(citation):
    if (citation):
        tempCitation = citation.split(',')
        temp = []
        for i in range(len(tempCitation)):
            if "volume" in tempCitation[i] or "pages" in tempCitation[i]:
                tempString = tempCitation[i].replace('{', "")
                tempString = tempString.replace('}', "")
                temp.append(tempString)
        if len(temp) > 0:
            citString = ",".join(temp)
            finalDict = dict(e.split(' = ') for e in citString.split(','))
            return finalDict
        else:
            return None
    else:
        return None
```

Συνάρτηση 5 Βοηθητική συνάρτηση διαχείρισης δημοσιεύσεων

Όρισμα:

- citation: ένα dictionary με πληροφορίες σχετικά με μια συγκεκριμένη δημοσίευση.

Επιστροφή:

- finalDict: το αρχικό dictionary μετασχηματισμένο στην κατάλληλη μορφή.

Αλγόριθμος:

- Η συνάρτηση αρχικά ελέγχει αν το όρισμα citation είναι κενό ή None.
- Αν το όρισμα citation δεν είναι κενό, η συνάρτηση σπάει τη μεταβλητή citation σε μια λίστα από strings χρησιμοποιώντας το κόμμα ',' σαν διαχωριστικό και αποθηκεύει το αποτέλεσμα σε μια μεταβλητή με όνομα "tempCitation".
- Έπειτα δημιουργείται μια κενή λίστα με όνομα "temp" και εκτελείται βρόχος επανάληψης μέσα στην λίστα "tempCitation".
- Σε κάθε επανάληψη ελέγχεται αν το string περιέχει τα substrings "volume" ή "pages".
- Αν περιέχεται, απομακρύνονται τα curly brackets από το string και το νέο string αποθηκεύεται στη λίστα "temp".
- Μετά τον επαναληπτικό βρόχο, ελέγχεται αν η λίστα "temp" είναι άδεια.
- Αν η μεταβλητή “temp” δεν είναι άδεια, δημιουργείται ένα νέο string με την ένωση των strings της μεταβλητής "temp" χρησιμοποιώντας το κόμμα ',' σαν διαχωριστή, και το αποτέλεσμα αποθηκεύεται στη μεταβλητή "citString".
- Έπειτα δημιουργείται ένα dictionary από τη μεταβλητή "citString" χωρίζοντας κάθε στοιχείο του citString σε ζευγάρι key-value χρησιμοποιώντας το ίσον ' = ' σαν διαχωριστή.
- Η συνάρτηση επιστρέφει τη μεταβλητή finalDict.
- Αν η λίστα"temp" είναι άδεια, η συνάρτηση επιστρέφει None.
- Αν το όρισμα "citation" είναι κενό ή δεν υπάρχει η συνάρτηση επιστρέφει None.

Η συνάρτηση dateHelper καλείται από τη συνάρτηση get_user_data_detailed_ORCID και σκοπό έχει να μετασχηματίσει τις ημερομηνίες για μια συγκεκριμένη δημοσίευση.

```
def dateHelper(dateFetched):
    if (dateFetched == None):
        return None
    else:
        year = dateFetched["year"]['value'] if dateFetched["year"] else
"1970"
        month = dateFetched["month"]['value'] if dateFetched["month"] else
"01"
        day = dateFetched["day"]['value'] if dateFetched["day"] else "01"
        return f"{year}-{month}-{day}"
```

Συνάρτηση 6 Βοηθητική συνάρτηση για την επεξεργασία ημερομηνιών

Όρισμα:

- dateFetched: ημερομηνία όπως επιστράφηκε από τις κλήσεις στο Orcid, αναμένεται ως dictionary.

Επιστροφή:

- finalDict: ημερομηνία ως string με τη μορφή yyyy—mm-dd.

Αλγόριθμος:

- Η συνάρτηση ελέγχει αν το όρισμα "dateFetched" είναι κενό.

- Αν το όρισμα "dateFetched" είναι κενό, η συνάρτηση επιστρέφει None.
- Αν το όρισμα "dateFetched" δεν είναι κενό, εξάγονται τα strings"year", "month", και "day" από το dictionary "dateFetched", ενώ εάν δεν βρεθούν τα παραπάνω strings ανατίθενται οι τιμές "1970", "01", και "01" αναλογικά.
- Η συνάρτηση επιστρέφει το string με τη μορφή "{year}-{month}-{day}".

Η συνάρτηση authorList καλείται από τη συνάρτηση get_user_data_detailed_ORCID και σκοπό έχει να δημιουργήσει ένα κείμενο διαχωρισμένο με κόμμα σχετικά με τους συγγραφείς μια συγκεκριμένης δημοσίευσης.

```
def authorsList(contributors):
    creators = ''
    if len(contributors) > 0:
        for idx in range(len(contributors)):
            if idx == 0:
                creators = contributors[idx]["credit-name"]["value"].re-
place(
                    ",", "")
            else:
                creators = creators + ', ' + \
                    contributors[idx]["credit-name"]["value"].replace(
                        ",", "")
        return creators
    else:
        return None
```

Συνάρτηση 7 Βοηθητική συνάρτηση επεξεργασίας συγγραφέων

Όρισμα:

- contributors: πίνακας από dictionaries με τους συγγραφείς ενός ακαδημαϊκού έργου.

Επιστροφή:

- creators: κείμενο διαχωρισμένο με κόμμα με τους δημιουργούς ενός ακαδημαϊκού έργου.

Αλγόριθμος:

- Αρχικοποιείται μια κενή μεταβλητή (string) με όνομα creators.
- Ελέγχεται αν το μέγεθος του πίνακα contributors είναι μεγαλύτερο από 0.
- Αν είναι, τρέχει ένας επαναληπτικός βρόχος σε κάθε στοιχείο της λίστας contributors, χρησιμοποιώντας τη μεταβλητή idx για να ελέγχεται ο τρέχοντας index της επανάληψης.
- Για κάθε στοιχείο της λίστας, ελέγχεται αν το idx είναι ίσο με το 0.
- Αν είναι, ανατίθεται η τιμή του κλειδιού "credit-name" key του τρέχοντος dictionary στη μεταβλητή creators, έχοντας αντικαταστήσει τυχόν κόμματα στο string με κενό string.
- Αν η μεταβλητή idx δεν είναι ίση με το 0, η συνάρτηση προσθέτει την τιμή του κλειδιού "credit-name" στην υπάρχουσα τιμή της μεταβλητής creators, χωρισμένη με κόμμα και κενό χαρακτήρα.
- Όταν ολοκληρωθούν οι επαναλήψεις η συνάρτηση επιστρέφει την τιμή της μεταβλητής creators.

- Εάν το μέγεθος της λίστας contributors είναι μικρότερη ή ίση με το 0, η συνάρτηση επιστρέφει None.

Η συνάρτηση removeDashAndCapitalize καλείται από τη συνάρτηση get_user_data_detailed_ORCID και σκοπό έχει να αφαιρέσει τις παύλες από ένα string και να το μετασχηματίσει σε Camelcase.

```
def removeDashAndCapitalize(text):
    text = text.title().replace("-", " ")
    return text
```

Συνάρτηση 8 Βοηθητική συνάρτηση επεξεργασίας κειμένου

Όρισμα:

- text: κείμενο σε μορφή string.

Επιστροφή:

- text: το αρχικό string έχοντας αντικαταστήσει το χαρακτήρα "-" με κενό χαρακτήρα και θέτοντας τον αρχικό χαρακτήρα του string με κεφαλαίο και μικρά όλους τους υπόλοιπους χαρακτήρες..

Αλγόριθμος:

- Στη μεταβλητή text, ανατίθεται το όρισμα text έχοντας αντικαταστήσει το χαρακτήρα "-" με κενό χαρακτήρα και θέτοντας τον αρχικό χαρακτήρα του string text με κεφαλαίο και μικρά όλους τους υπόλοιπους χαρακτήρες..
- Η συνάρτηση επιστρέφει την τροποποιημένη μεταβλητή "text".

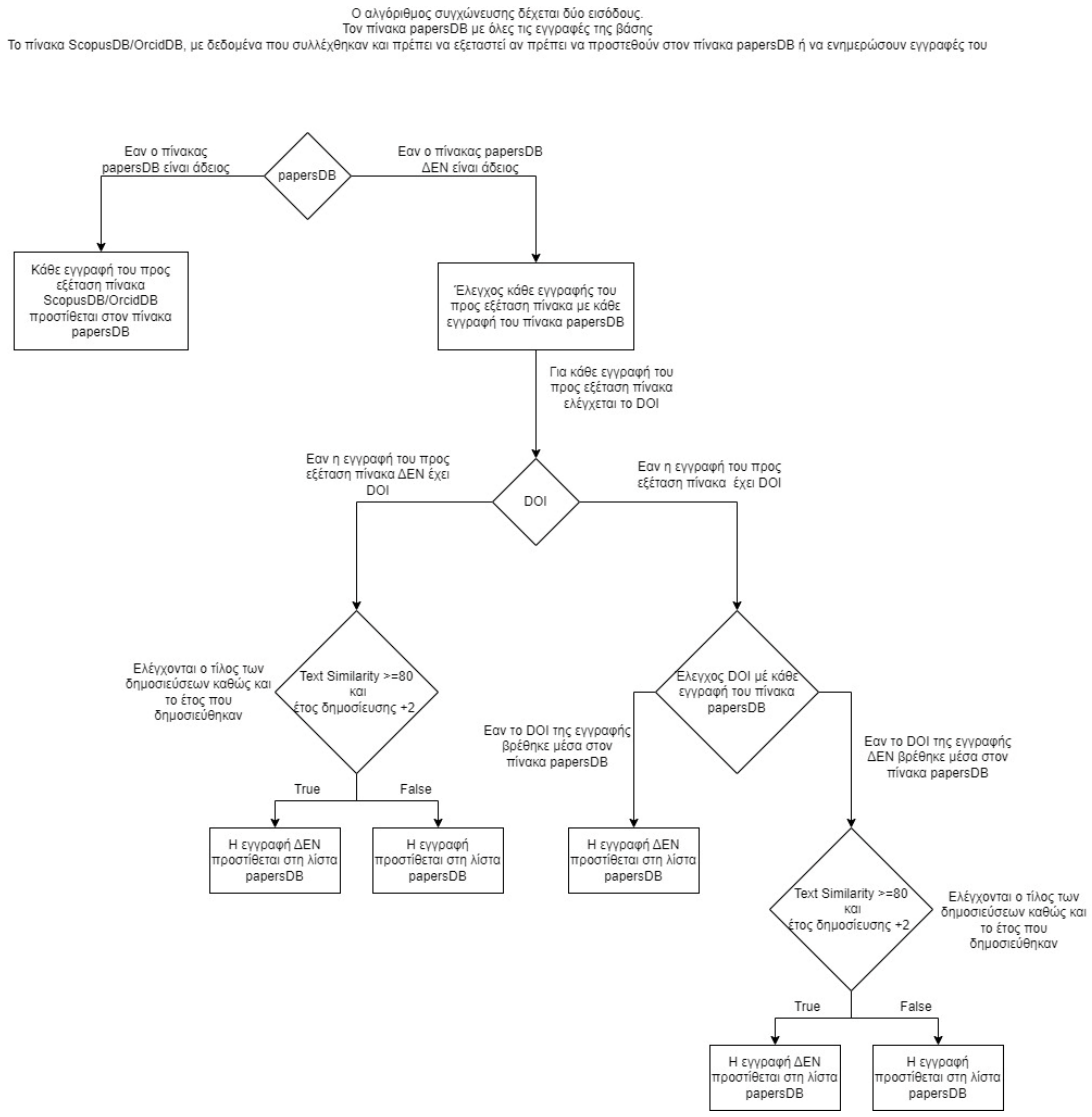
6.2 Αλγόριθμος Συγχώνευσης και Βοηθητικές Συναρτήσεις

Ο αλγόριθμος συγχώνευσης βοηθά να εξεταστεί εάν οι εγγραφές ενός πίνακα με δεδομένα που λήφθηκαν από εξωτερικές πηγές (Orcid, Scopus) πρέπει να προστεθούν ή να ενημερώσουν εγγραφές του πίνακα papers της βάσης δεδομένων μας. Ο πίνακας papers της βάσης, είναι υπεύθυνος για ότι βλέπουν οι χρήστες στο Frontend σχετικά με τις δημοσιεύσεις τους.

Γενικά, Ο αλγόριθμος συγχώνευσης δέχεται δύο εισόδους. Τον πίνακα papersDB με όλες τις εγγραφές της βάσης. Τον πίνακα ScopusDB/OrcidDB, με δεδομένα που συλλέχθηκαν και πρέπει να εξεταστεί αν πρέπει να προστεθούν στον πίνακα papersDB ή να ενημερώσουν εγγραφές του. Αυτό που επιστρέφεται από τον αλγόριθμο, είναι ο πίνακας papers της βάσης δεδομένων ενημερωμένος, ως προς τις εγγραφές που πρέπει να προστεθούν ή απλά να ενημερωθούν με νέα δεδομένα.

Παρακάτω θα αναλυθεί όλη η λογική πίσω από τον αλγόριθμο συγχώνευσης και θα εξηγηθούν ενδελεχώς όλες οι συναρτήσεις που υλοποιούν την λογική αυτή.

Η λογική συγχώνευσης φαίνεται στην παρακάτω εικόνα:



Εικόνα 4 Μηχανισμός συγχώνευσης δεδομένων

Αρχικά ελέγχεται αν ο πίνακας papersDB είναι άδειος ή όχι. Σε περίπτωση που είναι άδειος όλες οι εγγραφές του πίνακα που εξετάζεται προστίθενται στον πίνακα papersDB. Αν ο papersDB δεν είναι άδειος, τότε ελέγχονται οι εγγραφές του προς εξέταση πίνακα μια προς μια, επαναληπτικά.

Εάν η εγγραφή δεν έχει DOI τότε ελέγχεται ο τίτλος της δημοσίευσης και το έτος δημοσίευσης της εγγραφής αυτής με όλες τις εγγραφές του πίνακα papersDB. Για τους δύο τίτλους αφού αφαιρεθούν κενά, ειδικοί χαρακτήρες και μετατραπούν όλοι οι χαρακτήρες σε lowercase εάν δεν υπάρξει πλήρης ταύτιση, ο έλεγχος των τίτλων συνεχίζεται υπολογίζοντας την ομοιότητα μεταξύ τους. Για την ομοιότητα τίτλων επιστρέφεται ένα ποσοστό επί τοις εκατό, όπου 100% σημαίνει πλήρης ταύτιση. Ως προς το έτος δημοσίευσης ελέγχεται αν οι δημοσιεύσεις

εκδόθηκαν σε διάστημα ίσο η μικρότερο των 2 ετών. Αν λοιπόν η ομοιότητα των τίτλων είναι μεγαλύτερη ή ίση του 80% και οι δημοσιεύσεις εκδόθηκαν με διαφορά μικρότερη των δύο ετών η εγγραφή δεν προστίθενται στον πίνακα papersDB. Σε αντίθετη περίπτωση η εγγραφή προστίθεται στη λίστα papersDB.

Αν η εγγραφή έχει DOI, τότε ελέγχεται με κάθε εγγραφή του πίνακα papersDB. Εάν βρεθεί εγγραφή στον πίνακα papersDB με το ίδιο DOI, τότε η εγγραφή δεν προστίθενται στον πίνακα papersDB. Σε αντίθετη περίπτωση ο έλεγχος συνεχίζεται με τον τίτλο της δημοσίευσης και το έτος δημοσίευσης της εγγραφής αυτής με όλες τις εγγραφές του πίνακα papersDB.

Αν λοιπόν η ομοιότητα των τίτλων είναι μεγαλύτερη ή ίση του 80% και οι δημοσιεύσεις εκδόθηκαν με διαφορά μικρότερη των δύο ετών η εγγραφή δεν προστίθενται στον πίνακα papersDB. Σε αντίθετη περίπτωση η εγγραφή προστίθεται στη λίστα papersDB.

Παρακάτω θα εξηγηθούν ενδελεχώς όλες οι συναρτήσεις που υλοποιούν την λογική αυτή.

Πρόκειται για τις συναρτήσεις που βρίσκονται στον φάκελο BE/externalCalls, και πιο συγκεκριμένα συναρτήσεις στη αρχείο:

- mergeDBs.py, που διαμορφώνει τα δεδομένα που λήφθηκαν από Orcid και Scopus και τα συγχωνεύει σε μια οντότητα.

6.2.1 Μηχανισμός συγχώνευσης δημοσιεύσεων

Η συνάρτηση itemsToAdd υλοποιεί τη λογική συγχώνευσης όλων των δημοσιεύσεων. Χρησιμοποιεί τη συνάρτηση removeDuplicatesDB για να καθαρίσει τις λίστες-ορίσματα από διπλοεγγραφές και έπειτα με τη χρήση τη συνάρτησης mergeFunc γίνεται η συγχώνευση.

```
def itemsToAdd(papersList, scopusList, orcidList):
    toAdd = []
    if (len(scopusList) > 0):
        scopusRemovedDuplicates = removeDuplicatesDB(scopusList)
        merge1 = mergeFunc(
            papersList, scopusRemovedDuplicates, "Scopus")
        if (len(merge1) > 0):
            papersList = papersList+merge1
            toAdd = toAdd+merge1
    if (len(orcidList) > 0):
        orcidRemoveDuplicates = removeDuplicatesDB(orcidList)
        merge2 = mergeFunc(papersList, orcidRemoveDuplicates, "Orcid")
        if (len(merge2) > 0):
            toAdd = toAdd + merge2
        toAdd = removeDuplicatesDB(toAdd)
    return toAdd
```

Συνάρτηση 9 Μηχανισμός συγχώνευσης δημοσιεύσεων

Ορίσματα:

- papersList: πίνακας δημοσιεύσεων, αντίγραφο του πίνακα papers της βάσης που αποθηκεύεται πληροφορία αφού έχουν συγχωνευθεί οι πίνακες orcid και scopusID της βάσης,
- scopusList: πίνακας με τις δημοσιεύσεις όπως έχουν αντληθεί από την πηγή Scopus,
- orcidList: πίνακας με τις δημοσιεύσεις όπως έχουν αντληθεί από την πηγή Orcid.

Επιστροφή:

- toAdd: πίνακας με δημοσιεύσεις από τις λίστες scopusList και orcidList, οι οποίες πρέπει να εισαχθούν στο πίνακα papers της βάσης.

Αλγόριθμος:

- Αρχικοποίηση κενής λίστας με όνομα "toAdd"
- Έλεγχος εάν το μήκος του πίνακα "scopusList" είναι μεγαλύτερο από 0
- Εάν είναι αληθές, αφαίρεση των διπλοεγγραφών από τον πίνακα "scopusList" χρησιμοποιώντας τη συνάρτηση "removeDuplicatesDB" και αποθήκευση του αποτελέσματος στο "scopusRemovedDuplicates"
- Χρήση της συνάρτησης "mergeFunc" για τη συγχώνευση του "papersList" και του "scopusRemovedDuplicates" και αποθήκευση του αποτελέσματος στο "merge1"
- Εάν το μήκος του "merge1" είναι μεγαλύτερο από 0, το "merge1" προστίθεται στο "papersList" και το "toAdd"
- Έλεγχος εάν το μήκος του "orcidList" είναι μεγαλύτερο από 0
- Εάν είναι αληθές, αφαιρούνται οι διπλές εγγραφές από το "orcidList" χρησιμοποιώντας τη συνάρτηση "removeDuplicatesDB" και το αποτέλεσμα αποθηκεύεται στο "orcidRemoveDuplicates"
- Χρήση της συνάρτησης "mergeFunc" για τη συγχώνευση του "papersList" με το "orcidRemoveDuplicates" και αποθήκευση του αποτελέσματος στο "merge2"
- Εάν το μήκος του "merge2" είναι μεγαλύτερο από 0, το "merge2" προστίθεται στο "toAdd"
- Αφαίρεση των διπλοτύπων από το "toAdd" χρησιμοποιώντας τη συνάρτηση "removeDuplicatesDB"
- Επιστροφή "toAdd"

Η συνάρτηση removeDuplicates καλείται από τη συνάρτηση itemsToAdd με σκοπό να καθαρίσει από διπλοεγγραφές τις δύο λίστες που δέχεται ως ορίσματα

```
def removeDuplicatesDB(arr):
    indecesToRemove = []
    newArr = []
    for index in range(len(arr)):
        tempArr = copy.deepcopy(arr)
        for index2 in range(len(tempArr)):
            if (index != index2 and not(index in indecesToRemove)):
                if ((tempArr[index2]['doi'] == arr[index]['doi']) and
(arr[index]['doi'] != None)):
                    if (("source" not in tempArr[index2] and "source" not in
arr[index]) or (tempArr[index2]['source'] == "Mixed" and arr[in-
dex]['source'] == "Mixed") or (tempArr[index2]['source'] != "Mixed" and
arr[index]['source'] == "Mixed")):
                        indecesToRemove.append(index2)
                if ((textSimirality(titleClean(tempArr[index2]['title']),
titleClean(
                        arr[index]['title']))) and (checkDate(tempArr[in-
dex2]['publishedDate'], arr[index]['publishedDate']))):
                    if (("source" not in tempArr[index2] and "source" not in
arr[index]) or (tempArr[index2]['source'] == "Mixed" and arr[in-
dex]['source'] == "Mixed") or (tempArr[index2]['source'] != "Mixed" and
arr[index]['source'] == "Mixed")):
                        indecesToRemove.append(index2)
    for idx in range(len(arr)):
        if (not(idx in indecesToRemove)):
            newArr.append(arr[idx])
    return newArr
```

Συνάρτηση 10 Βοηθητική συνάρτηση αφαίρεσης διπλοτύπων

Όρισμα:

- arr: πίνακας που πρέπει να εντοπιστούν και να αφαιρεθούν οι διπλοεγγραφές του.

Επιστροφή:

- newArr: πίνακας καθαρισμένος από διπλοεγγραφές.

Αλγόριθμος:

- Αρχικοποίηση κενών λιστών που ονομάζονται "indecesToRemove" και "newArr"
- Δημιουργία αντίγραφου του "arr" και αποθήκευσή του το σε μια μεταβλητή "tempArr"
- Επανάληψη στα στοιχεία στο "tempArr" χρησιμοποιώντας μια μεταβλητή ευρετηρίου "index2"
- Έλεγχος εάν το "index" δεν είναι ίσο με το "index2" και το "index" δεν είναι στο "indecesToRemove"
- Ελέγχος εάν το "tempArr[index2]['doi']" είναι ίσο με "arr[index]['doi']" και το "arr[index]['doi']" δεν είναι None.
- Έλεγχος εάν είτε η "πηγή" δεν βρίσκεται στο "tempArr[index2]" και η "πηγή" δεν είναι στο "arr[index]" είτε η "tempArr[index2]['source']" είναι "Mixed" και "arr[index] To ['source']" είναι "Mixed" ή το "tempArr[index2]['source']" δεν είναι "Mixed" και το "arr[index]['source']" είναι "Mixed"
- Εάν ισχύουν οι παραπάνω συνθήκες, "index2" προστίθεται στο "indecesToRemove"

- Έλεγχος αν το textSimirality(titleClean(tempArr[index2]['title']), titleClean(arr[index]['title'])) είναι αληθές και checkDate(tempArr[index2]['publishedDate'], arr[index]['Ημερομηνία δημοσίευσης']) είναι αληθής
- Έλεγχος εάν είτε η "πηγή" δεν βρίσκεται στο "tempArr[index2]" και η "πηγή" δεν είναι στο "arr[index]" είτε η "tempArr[index2]['source']" είναι "Mixed" και "arr[index] To ['source']" είναι "Mixed" ή το "tempArr[index2]['source']" δεν είναι "Mixed" και το "arr[index]['source']" είναι "Mixed"
- Εάν ισχύουν οι παραπάνω συνθήκες, το "index2" προστίθεται στο "indecesToRemove"
- Επανάληψη στα στοιχεία στο "arr" χρησιμοποιώντας μια μεταβλητή ευρετηρίου "idx"
- Έλεγχος εάν το "idx" δεν βρίσκεται στο "indecesToRemove"
- Εάν η παραπάνω συνθήκη ισχύει, το "arr[idx]" προστίθεται στο "newArr"
- Επιστροφή του "newArr"

Η συνάρτηση mergeFunc καλείται από τη συνάρτηση itemsToAdd και στην ουσία πραγματοποιεί όλους τους απαραίτητους ελέγχους για να μπορέσει να πραγματοποιηθεί η συγχώνευση δύο λιστών με δημοσιεύσεις. Στην ουσία συγκρίνονται η λίστα δημοσιεύσεων της βάσης δεδομένων (papersDB), με τη λίστα δημοσιεύσεων όπως συλλέχθηκε από το Scopus ή το Orcid και προκύπτει ο τελικός πίνακας papersDB που ενημερώνει τη βάση δεδομένων με τα στοιχεία που έπρεπε να ενημερωθούν ή να προστεθούν.

```
def mergeFunc(mergedDB, toBeMergedDB, source):
    newAdditionDB = []
    if len(mergedDB) == 0:
        newAdditionDB = toBeMergedDB
        for item in newAdditionDB:
            item['source'] = source
        return newAdditionDB
    else:
        for index, publication in enumerate(toBeMergedDB):
            if (publication['doi'] != None):
                if any(pub['doi'] == publication['doi'] for pub in
mergedDB):
                    mergedIndex = next((i for i, item in enumerate(
                        mergedDB) if item['doi'] == publication['doi']), -1)
                    for key, value in mergedDB[mergedIndex].items():
                        if key in publication:
                            if ((value == None and publication[key] != None)
or (value != None and publication[key] != None and (isinstance(value, int)
== False and isinstance(publication[key], int) == False) and len(value) <
len(publication[key]))):
                                mergedDB[mergedIndex][key] = publica-
tion[key]
                                mergedDB[mergedIndex]['source'] = 'Mixed'
                                newAdditionDB.append(mergedDB[mergedIndex])
                else:
                    if any((textSimirality(titleClean(pub['title']), title-
Clean(publication['title']))) and (checkDate(pub['publishedDate'], publica-
tion['publishedDate'])) for pub in mergedDB):
                        mergedIndex = next((i for i, item in enumerate(
                            mergedDB) if ((textSimirality(title-
Clean(item['title']), titleClean(publication['title']))) and (check-
Date(item['publishedDate'], publication['publishedDate'])))), -1)
                    for key, value in mergedDB[mergedIndex].items():
                        if key in publication:
                            if ((value == None and publication[key] != None)
or (value != None and publication[key] != None and (isinstance(value, int)
== False and isinstance(publication[key], int) == False) and len(value) <
len(publication[key]))):
                                mergedDB[mergedIndex][key] = publica-
tion[key]
                                mergedDB[mergedIndex]['source'] = 'Mixed'
                                newAdditionDB.append(mergedDB[mergedIndex])
```

Συνάρτηση 11 Βοηθητική συνάρτηση συγχώνευσης

```
                else:
                    # Ean den vrethei allo publication me idio onoma
                    temp = publication
                    temp['source'] = source
                    newAdditionDB.append(temp)
        return newAdditionDB
                        for key, value in mergedDB[mergedIndex].items():

                             if key in publication:
                                 if ((value == None and publication[key] !=
None) or (value != None and publication[key] != None and (isinstance(value,
int) == False and isinstance(publication[key], int) == False) and len(value)
< len(publication[key]))):
                                     mergedDB[mergedIndex][key] = publica-
tion[key]
                                     mergedDB[mergedIndex]['source'] =
'Mixed'
                                     newAdditionDB.append(mergedDB[mergedIn-
dex])
                    else:
                        # Ean den vrethei allo publication me idio onoma
                        temp = publication
                        temp['source'] = source
                        newAdditionDB.append(temp)
            else:
                # Ean vrethei allo publication me idio onoma prospername kai
+-2 publication year
                if any((textSimirality(titleClean(pub['title']), title-
Clean(publication['title'])) and (checkDate(pub['publishedDate'], publica-
tion['publishedDate'])) for pub in mergedDB):
                    mergedIndex = next((i for i, item in enumerate(
                        mergedDB) if ((textSimirality(titleClean(item['ti-
tle']), titleClean(publication['title'])) and (checkDate(item['pub-
lishedDate'], publication['publishedDate'])))), -1)
                    for key, value in mergedDB[mergedIndex].items():
                        if key in publication:
                            if ((value == None and publication[key] != None)
or (value != None and publication[key] != None and (isinstance(value, int)
== False and isinstance(publication[key], int) == False) and len(value) <
len(publication[key]))):
```

Συνάρτηση 12 Βοηθητική συνάρτηση συγχώνευσης

Ορίσματα:

- mergedDB: πίνακας δημοσιεύσεων, αντίγραφο του πίνακα papers της βάσης που αποθηκεύεται πληροφορία αφού έχουν συγχωνευθεί οι πίνακες orcid και scopusID της βάσης,
- toBeMergedDB: πίνακας με τις δημοσιεύσεις προς συγχώνευση, source= string που δείχνει αν ο πίνακας προς συγχώνευση έρχεται από το Scopus ή το Orcid.

Επιστροφή:

- newAdditionDB: πίνακας με δημοσιεύσεις οι οποίες πρέπει να εισαχθούν στον πίνακα papers της βάσης.

Αλγόριθμος:

- Αρχικοποίηση μιας κενής λίστας που ονομάζεται "newAdditionDB"
- Εάν το μήκος του "mergedDB" είναι 0, ορίζεται το "newAdditionDB" σε "toBeMergedDB" και για κάθε στοιχείο στο "newAdditionDB" και επιστρέφεται το "newAdditionDB"
- Διαφορετικά, τρέχει επανάληψη κάθε "δημοσίευση" στο "toBeMergedDB"
- Εάν το κλειδί "doi" της "δημοσίευσης" δεν είναι None, γίνεται έλεγχος εάν οποιοδήποτε "pub" στο "mergedDB" έχει το ίδιο "doi"
- Εάν κάποια "pub" στο "mergedDB" έχει το ίδιο "doi", εντοπίζεται το index αυτής της "pub" και ενημερώνονται τα κλειδιά του αντίστοιχου στοιχείου "mergedDB" με τις τιμές από "publication" όπου η "publication" έχει μεγαλύτερη τιμή ή μια μη μηδενική τιμή, ορίζεται το κλειδί "source" σε "Mixed" και προστίθεται το ενημερωμένο στοιχείο "mergedDB" στο "newAdditionDB"
- Εάν καμία "pub" στο "mergedDB" δεν έχει το ίδιο "doi", γίνεται έλεγχος εάν κάποια "pub" στο "mergedDB" έχει τον ίδιο τίτλο και έτος δημοσίευσης +- 2 χρόνια με τη "δημοσίευση" χρησιμοποιώντας τις συναρτήσεις "textSimilarity" και "checkDate".
- Εάν κάποια "pub" στο "mergedDB" έχει τον ίδιο τίτλο και έτος δημοσίευσης +- 2 χρόνια, εντοπίζεται ο index αυτής της "pub" και ενημερώνονται τα κλειδιά του αντίστοιχου στοιχείου "mergedDB" με τις τιμές από "publication" όπου "publication" " έχει μεγαλύτερη τιμή ή μη μηδενική τιμή, ορίζεται το κλειδί "source" σε "Mixed" και προστίθεται το ενημερωμένο στοιχείο "mergedDB" στο "newAdditionDB"
- Εάν καμία "pub" στο "mergedDB" δεν έχει τον ίδιο τίτλο και έτος δημοσίευσης +- 2 χρόνια, προστίθεται η "δημοσίευση" με την πηγή της (όχι Mixed) στο "newAdditionDB"
- Εάν το κλειδί "doi" της "δημοσίευσης" είναι None, ελέγχεται εάν οποιοδήποτε "pub" στο "mergedDB" έχει τον ίδιο τίτλο και έτος δημοσίευσης +- 2 χρόνια με το "publication" χρησιμοποιώντας τις συναρτήσεις "textSimilarity" και "checkDate"
- Εάν κάποια "pub" στο "mergedDB" έχει τον ίδιο τίτλο και έτος δημοσίευσης +- 2 χρόνια, εντοπίζεται ο index αυτής της "pub" και ενημερώνονται τα κλειδιά του αντίστοιχου στοιχείου "mergedDB" με τις τιμές από "publication" όπου "publication" " έχει μεγαλύτερη τιμή ή μη μηδενική τιμή, ορίζεται το κλειδί "source" σε "Mixed" και προστίθεται το ενημερωμένο στοιχείο "mergedDB" στο "newAdditionDB"
- Εάν καμία "pub" στο "mergedDB" δεν έχει τον ίδιο τίτλο και έτος δημοσίευσης +- 2 χρόνια, προστίθεται η "δημοσίευση" με την πηγή της (όχι Mixed) στο "newAdditionDB".
- Επιστροφή του "newAdditionDB"

Η συνάρτηση textSimilarity καλείται από τη συνάρτηση mergeFunc έτσι ώστε να ελεγχθούν εάν δύο τίτλοι δημοσιεύσεων είναι παρόμοιοι ή όχι. Στην πραγματικότητα ελέγχεται η απόσταση από δύο strings με τη μέθοδο hamming και επιστρέφεται ένα ποσοστό ομοιότητας. Εφόσον αυτό το ποσοστό ξεπερνά ένα κατώφλι (0.7) τότε οι δύο τίτλοι θεωρούνται όμοιοι.

```
def textSimirality(title1, title2):
    if (textdistance.hamming.normalized_similarity(title1, title2) > 0.7):
        return True
    else:
        return False
```

Συνάρτηση 13 Βοηθητική συνάρτηση υπολογισμού ομοιότητας κειμένου

Ορίσματα:

- title1: string προς εξέταση ομοιότητας,
- title2: string προς εξέταση ομοιότητας.

Επιστροφή:

- True εάν οι 2 τίτλοι είναι όμοιοι,
- False αν δεν είναι.

Αλγόριθμος:

- Χρησιμοποιείται η βιβλιοθήκη απόστασης κειμένου για να υπολογίσει την απόσταση Hamming μεταξύ των δύο τίτλων εισόδου και στη συνέχεια κανονικοποιεί την ομοιότητα μεταξύ των δύο τίτλων εισόδου καλώντας τη συνάρτηση "textdistance.hamming.normalized_similarity".
- Στη συνέχεια, συγκρίνει τη βαθμολογία ομοιότητας με 0,7, Αν η βαθμολογία ομοιότητας είναι μεγαλύτερη από 0,7, η συνάρτηση επιστρέφει "True" διαφορετικά επιστρέφει "False".

Η συνάρτηση titleClean καλείται από τη συνάρτηση mergeFunc έτσι ώστε να “καθαριστεί” ένα συγκεκριμένο string. Αυτό που συμβαίνει είναι να αφαιρούνται όλοι οι ειδικοί χαρακτήρες από το δοθέν string, να αφαιρούνται τυχόν κενά και όλοι οι χαρακτήρες για γίνονται lowercase.

```
def titleClean(text):
    text = " ".join(re.findall(r"[a-zA-Z0-9]+", text)).lower()
    return text
```

Συνάρτηση 14 Βοηθητική συνάρτησης καθαρισμού κειμένου

Όρισμα:

- temp: string προς μορφοποίηση.

Επιστροφή:

- το κείμενο μορφοποιημένο.

Αλγόριθμος:

- Αφαίρεση ειδικών χαρακτήρων και κενά από τη μεταβλητή "text" χρησιμοποιώντας regex και μετατροπή όλων τους χαρακτήρων σε πεζούς
- Επιστροφή της μορφοποιημένη μεταβλητή "text".

Η συνάρτηση checkDate καλείται από τη συνάρτηση mergeFunc και υπολογίζει εάν δύο δοθείσες ημερομηνίες απέχουν μεταξύ τους περισσότερο ή λιγότερο από 2 χρόνια.

```
def checkDate(date1, date2):
    if (date1 == None or date2 == None):
        return True
    elif (isinstance(date1, int) == True and isinstance(date2, int) ==
True):
        if ((date1 <= date2+2) and (date1 >= date2-2)):
            return True
        else:
            return False
    elif (isinstance(date1, int) == True and isinstance(date2, int) ==
False):
        if ((date1 <= int(date2[0: 4])+2) and (date1 >= int(date2[0: 4])-
2)):
            return True
        else:
            return False
    elif (isinstance(date1, int) == False and isinstance(date2, int) ==
True):
        if ((int(date1[0: 4]) <= date2+2) and (date1 >= date2-2)):
            return True
        else:
            return False
    else:
        if ((int(date1[0: 4]) <= int(date2[0: 4])+2) and (int(date1[0: 4])
>= int(date2[0: 4])-2)):
            return True
        else:
            return False
```

Συνάρτηση 15 Βοηθητική συνάρτηση ελέγχου ημερομηνιών

Ορίσματα:

- date1: ημερομηνίες προς εξέταση εγγύτητας +-2 ετών.
- date2: ημερομηνίες προς εξέταση εγγύτητας +-2 ετών.

Επιστροφή:

- True εάν οι 2 ημερομηνίες έχουν διαφορά μεγαλύτερη των 2 ετών,
- False αν δεν είναι.

Αλγόριθμος:

- Εάν η ημερομηνία1 ή η ημερομηνία2 είναι None, επιστρέφει αμέσως True.
- Εάν και η ημερομηνία1 και η ημερομηνία2 είναι ακέραιοι, ελέγχει εάν η ημερομηνία1 είναι εντός 2 ετών από την ημερομηνία2.
- Εάν η ημερομηνία1 είναι ακέραιος και η ημερομηνία2 δεν είναι ακέραιος, ελέγχει εάν η ημερομηνία1 είναι εντός 2 ετών από το έτος της ημερομηνίας2.
- Εάν η ημερομηνία1 δεν είναι ακέραιος και η ημερομηνία2 είναι ακέραιος, ελέγχει εάν το έτος της ημερομηνίας1 είναι εντός 2 ετών από την ημερομηνία2.
- Εάν και η ημερομηνία1 και η ημερομηνία2 δεν είναι ακέραιοι, ελέγχει εάν το έτος της ημερομηνίας1 είναι εντός 2 ετών από το έτος της ημερομηνίας2.

- Εάν πληρούνται κάποια από αυτές τις προϋποθέσεις, επιστρέφει True. Διαφορετικά, επιστρέφει False.

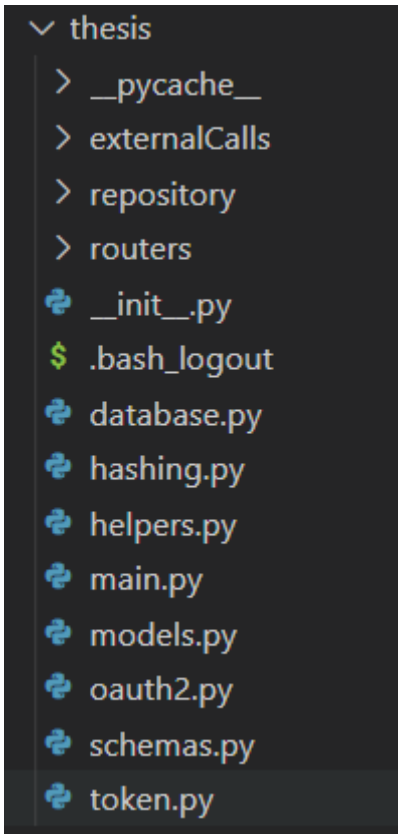
7

Backend - API

Για την υλοποίηση του Backend, χρησιμοποιήθηκε το FastAPI, ένα framework της Python. Παρακάτω, καταγράφεται η οργάνωση των φακέλων του Backend, και περιγράφονται όλα τα endpoints που δημιουργήθηκαν.

7.1.1 Οργάνωση φακέλων Backend

Μέσα στο φάκελο BE του project μας, βρίσκεται όλο το υλικό που έχει παραχθεί και αποτελεί το Backend της εφαρμογής. Στον φάκελο BE, βρίσκονται ο φάκελος thesis όπου ζει όλο το Backend, ο φάκελος thesis_env όπου είναι εγκατεστημένες όλες οι βιβλιοθήκες που χρειαζόμαστε δημιουργώντας ένα περιβάλλον αποκλειστικά για την εφαρμογή, και το αρχείο requirements, στο οποίο είναι καταγεγραμμένες όλες οι εξωτερικές βιβλιοθήκες που χρησιμοποιούνται και απαιτούνται για τα εκτελεστεί το Backend. Τα πιο σημαντικά αρχεία και φάκελοι βρίσκονται στο φάκελο thesis και φαίνονται στην παρακάτω εικόνα.



Εικόνα 5 Οργάνωση φακέλων Backend

Παρακάτω αναλύεται τι υπάρχει στον κάθε φάκελο/αρχείο που φαίνεται στην παραπάνω εικόνα:

- externalCalls, εδώ βρίσκονται 2 αρχεία, ένα που πραγματοποιεί τις συνδέσεις με τα εξωτερικά APIs του Scopus και του Orcid και ένα που αφού συλλέξει τις πληροφορίες αυτές τις συγχωνεύει σε έναν πίνακα.
- Repository, εδώ βρίσκονται οι συναρτήσεις για τις δημοσιεύσεις και τους χρήστες που εκτελούνται όταν πραγματοποιούνται κλήσεις στο Backend
- Routers, εδώ κάθε συνάρτηση που βρίσκεται στο φάκελο repository συνδέεται με ένα συγκεκριμένο route-κλήση έτσι ώστε να εκτελεστεί όταν πραγματοποιηθεί η κλήση στο συγκεκριμένο route.
- Database.py, εδώ αρχικοποιείται η βάση δεδομένων,
- hashing.py, εδώ βρίσκονται οι συναρτήσεις που πραγματοποιούν encrypt και decrypt σε περίπτωση που χρησιμοποιηθεί password για τους χρήστες στο μέλλον.
- Helpers.py εδώ βρίσκονται βοηθητικές συναρτήσεις όπως πχ έλεγχος αν ένα email είναι έγκυρο.

- Main.py εδώ είναι το αρχείο εισαγωγής από όπου αρχίζει να εκτελείται η Python.
- Models.py εδώ βρίσκονται τα μοντέλα τα οποία χρησιμοποιούνται από την Python για να δημιουργήσουν τους πίνακες της βάσης δεδομένων.
- Oauth2.py εδώ βρίσκονται οι συναρτήσεις που βοηθούν στην επαλήθευση του token,
- schemas.py, εδώ βρίσκονται οδηγίες για το τι πληροφορία πρέπει να επιστρέφει κάθε API που έχει δημιουργηθεί αλλά και τι πληροφορία-όρισμα απαιτείται κάθε φορά,
- token.py, εδώ βρίσκονται οι συναρτήσεις που δημιουργούν το token.

7.1.2 API Routes

Στο BE έχει αναπτυχθεί ένα WebAPI το οποίο διαθέτει στο FE όλες τις απαραίτητες πληροφορίες που υλοποιούν την εφαρμογή μας.

Υπάρχει λεπτομερέστατη καταγραφή (documentation) της υλοποίηση των API κλήσεων στη διεύθυνση https://mypubs.iee.ihu.gr/api/v1/docs. Εκεί υπάρχουν πληροφορίες σχετικές με όλα τα endpoints που έχουν δημιουργηθεί, τις απαιτήσεις του καθενός, καθώς επίσης μπορεί κάποιος να δοκιμάσει να καλέσει απευθείας κάποιο από τα endpoints. Η καταγραφή των λειτουργιών των endpoints έχει γίνει με τη βοήθεια του swagger, εργαλείου το οποίο είναι ενσωματωμένο στο FastAPI, το οποίο χρησιμοποιήσαμε.

Τα endpoints που έχουν δημιουργηθεί χωρίζονται σε 3 κατηγορίες:

7.1.2.1 User,

Έχουν δημιουργηθεί 2 endpoints σχετικά με τους χρήστες της εφαρμογής.

Μέθοδος	**Path**
POST	/user/
GET	/user/{uid}
PUT	/user/{uid}
PATCH	/user/updateAll

Το πρώτο endpoint:

- είναι ένα Post Request,
- βοηθά στη δημιουργία νέου χρήστη στην εφαρμογή.

Το δεύτερο endpoint:

- είναι ένα Get Request,
- απαιτεί τον χρήστη που πραγματοποιεί την κλήση να είναι αυθεντικοποιημένος,
- απαιτεί ως είσοδο το uid ενός χρήστη,
- επιστρέφει όλες τις πληροφορίες του χρήστη με το δοθέν uid.

Το τρίτο endpoint:

- είναι ένα Put Request,
- απαιτεί τον χρήστη που πραγματοποιεί την κλήση να είναι αυθεντικοποιημένος,
- απαιτεί ως είσοδο το uid ενός χρήστη,
- ενημερώνει όλες τις πληροφορίες του χρήστη με το δοθέν uid.

Το τέταρτο endpoint:

- είναι ένα Patch Request,
- απαιτεί τον χρήστη που πραγματοποιεί την κλήση να είναι αυθεντικοποιημένος,
- ενημερώνει όλες τις πληροφορίες όλων των χρηστών της βάσης δεδομένων.

7.1.2.2 Paper,

Έχουν δημιουργηθεί 5 endpoints σχετικά με τις δημοσιεύσεις της εφαρμογής.

Μέθοδος	**Path**
GET	/paper/
POST	/paper/
GET	/paper/{id}
PUT	/paper/{id}
DELETE	/paper/{id}

Το πρώτο endpoint:

- είναι ένα Get Request,
- απαιτεί τον χρήστη που πραγματοποιεί την κλήση να είναι αυθεντικοποιημένος,
- επιστρέφει όλες τις δημοσιεύσεις που έχουν εισαχθεί στην εφαρμογή.

Το δεύτερο endpoint:

- είναι ένα Post Request,

- απαιτεί τον χρήστη που πραγματοποιεί την κλήση να είναι αυθεντικοποιημένος,
- βοηθά στη δημιουργία και την αποθήκευση μιας δημοσίευσης στην εφαρμογή

Το τρίτο endpoint:

- είναι ένα Get Request,
- απαιτεί τον χρήστη που πραγματοποιεί την κλήση να είναι αυθεντικοποιημένος,
- απαιτεί ως είσοδο το id μιας δημοσίευσης,
- επιστρέφει όλες τις πληροφορίες της δημοσίευσης με το δοθέν id.

Το τέταρτο endpoint:

- είναι ένα Put Request,
- απαιτεί τον χρήστη που πραγματοποιεί την κλήση να είναι αυθεντικοποιημένος,
- απαιτεί ως είσοδο το id μιας δημοσίευσης,
- ενημερώνει όλες τις πληροφορίες της δημοσίευσης με το δοθέν id.

Το πέμπτο endpoint:

- είναι ένα Delete Request,
- απαιτεί τον χρήστη που πραγματοποιεί την κλήση να είναι αυθεντικοποιημένος,
- απαιτεί ως είσοδο το id μιας δημοσίευσης,
- διαγράφει τη δημοσίευση με το δοθέν id.

7.1.2.3 Authentication,

Έχει δημιουργηθεί 1 endpoint σχετικά με την αυθεντικοποίηση της εφαρμογής.

Μέθοδος	**Path**
POST	/login

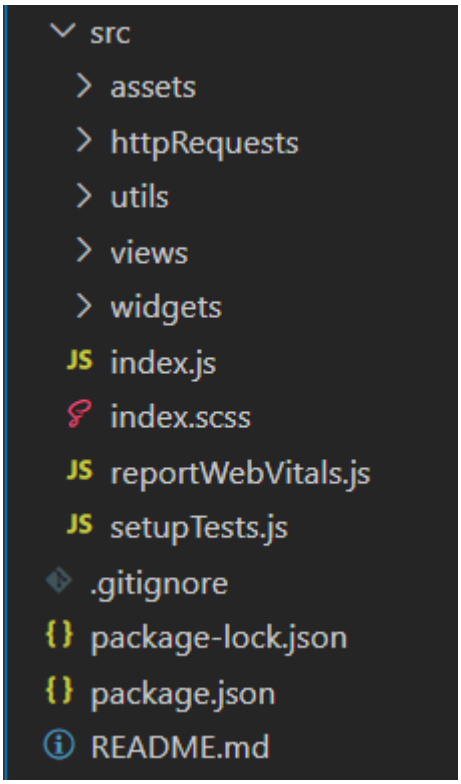
Το endpoint:

- είναι ένα Post Request,
- βοηθά στη είσοδο του χρήστη στην εφαρμογή.

8

Frontend

Όπως έχει ήδη αναφερθεί για την υλοποίηση του Frontend, χρησιμοποιήθηκε η React, μια βιβλιοθήκη της Javascript που έχει δημιουργηθεί από το Facebook. Παρακάτω, καταγράφεται η οργάνωση των φακέλων του Frontend, οι συναρτήσεις που καλούν το BE καθώς και κάποιες επιπρόσθετες βοηθητικές συναρτήσεις που υλοποιήθηκαν.



Εικόνα 6 Οργάνωση φακέλων Frontend

8.1.1 Οργάνωση φακέλων Frontend

Μέσα στο φάκελο FE του project μας, βρίσκεται όλο το υλικό που έχει παραχθεί και αποτελεί το Frontend της εφαρμογής. Τα πιο σημαντικά αρχεία και φάκελοι φαίνονται στην παρακάτω εικόνα. Στον φάκελο src (source) ζει όλο το Frontend. Στον φάκελο src, εντοπίζονται κάποιοι επιπρόσθετοι φάκελοι, οι οποίοι είναι:

- assets, με φωτογραφίες που χρησιμοποιούνται στην εφαρμογή,
- httpRequests, με συναρτήσεις οι οποίες καλούν το Backend,
- utils, με βοηθητικές συναρτήσεις για το Frontend,
- views, με όλα τα components του Frontend που συνιστούν διαφορετικά layouts-routes της εφαρμογής, όπως πχ η αρχική σελίδα, η σελίδα προφίλ χρήστη, η σελίδα δημοσιεύσεων χρήστη κτλ.
- Widgets, με τα βοηθητικά components που χρησιμοποιούνται σε διαφορετικά layouts-routes, όπως πχ η λίστα των δημοσιεύσεων ενός χρήστη που χρησιμοποιείται στη σελίδα των δημοσιεύσεων του χρήστη.
- Index.js, που δείχνει το αρχείο εισαγωγής της React,
- index.scss το αρχείο styling του index.js

Εκτός του φακέλου src βρίσκεται επίσης το σημαντικό αρχείο:

- package.json, που καταγράφει όλες τις εξωτερικές βιβλιοθήκες που χρησιμοποιούνται στο frontend καθώς και τις εκδόσεις τους, αλλά και διάφορα scripts για να εκτελεστεί το Frontend κτλ.,

8.1.2 Συναρτήσεις κλήσης του Backend (από το Frontend)

Στο Frontend έχουν υλοποιηθεί 3 συναρτήσεις που πραγματοποιούν API requests στο Backend της εφαρμογής. Όπως φαίνεται και παραπάνω οι συναρτήσεις αυτές βρίσκονται μέσα στο φάκελο httpRequests ενώ η συνολική διαδρομή είναι FE>src>httpRequets.Για όλες αυτές τις συναρτήσεις χρησιμοποιείται η βιβλιοθήκη axios για να πραγματοποιηθούν οι κλήσεις καθώς και η συνάρτηση baseURL, η οποία ανάλογα με το περιβάλλον στο οποίο βρισκόμαστε, δηλαδή σε παραγωγή (production) ή ανάπτυξη, χρησιμοποιεί το σωστό URL για να καλέσει το Backend.

8.1.2.1 Συνάρτηση getToken

Η παρακάτω συνάρτηση βοηθά στην λήψη του εσωτερικού token της εφαρμογής, έτσι ώστε να μπορούν να πραγματοποιηθούν περαιτέρω κλήσεις από τον επαληθευμένο χρήστη που έχει ολοκληρώσει τη διαδικασία της σύνδεσης (login - SSO) στην εφαρμογή. Όταν ο χρήστης προσπαθεί να συνδεθεί στην εφαρμογή μας μέσω SSO, παράγεται ένας κωδικός και ένα αναγνωριστικό του Client, τα οποία όταν αποσταλούν στο Backend ελέγχονται και αν είναι όλα εντάξει ο χρήστης επαληθεύεται, και του δίνεται ένας μοναδικός κωδικός (token), έτσι ώστε να μπορούν να πραγματοποιηθούν νέες κλήσεις, ως επαληθευμένος πια χρήστης.

```
import axios from "axios";
import baseURL from "../utils/baseURL";

const getToken = (code, clientID) => {
  const config = {
    client_id: clientID,
    code: code,
  };

  return axios.post(`${baseURL()}login`, config);
};

export default getToken;
```

Συνάρτηση 16 Συνάρτηση απόκτησης token

Ορίσματα συνάρτησης:

- code, κωδικός ο οποίος λήφθηκε μέσω του SSO,
- clientID, αναγνωριστικό client που λήφθηκε μέσω του SSO,

Επιστροφή συνάρτησης:

- Σε επιτυχημένη κλήση, μοναδικό token επαληθευμένου χρήστη,
- Σε αποτυχημένη κλήση, μήνυμα σφάλματος

8.1.2.2 Συνάρτηση showUser

Η παρακάτω συνάρτηση επιστρέφει πληροφορίες για έναν συγκεκριμένο χρήστη, όπως το ονοματεπώνυμο, το orcid, το scopus id, το email κτλ. Όπως φαίνεται και παρακάτω, το token εδώ χρησιμοποιείται στην κεφαλίδα της κλήσης (headers), έτσι ώστε ο χρήστης που πραγματοποιεί τη κλήση να επαληθευτεί και να πάρει τις πληροφορίες που ζητά εφόσον έχει την άδεια.

```
import axios from "axios";
import baseURL from "../utils/baseURL";

const showUser = (token, uid) => {
  const config = {
    headers: { Authorization: `Bearer ${token}` },
  };

  return axios.get(`${baseURL()}user/${uid}`, config);
};

export default showUser;
```

Συνάρτηση 17 Απόκτηση πληροφοριών χρήστη

Ορίσματα συνάρτησης:

- token, μοναδικός κωδικός για τον επαληθευμένο χρήστη,
- uid, μοναδικός κωδικός που αντιστοιχεί σε έναν μόνο χρήστη,

Επιστροφή συνάρτησης:

- Σε επιτυχημένη κλήση, όλα τα στοιχεία του χρήστη με το συγκεκριμένο uid
- Σε αποτυχημένη κλήση, μήνυμα σφάλματος

8.1.2.3 Συνάρτηση syncUser

Η παρακάτω συνάρτηση ενημερώνει τις δημοσιεύσεις για έναν συγκεκριμένο χρήστη. Όπως φαίνεται και παρακάτω, το token εδώ χρησιμοποιείται στην κεφαλίδα της κλήσης (headers),

έτσι ώστε ο χρήστης που πραγματοποιεί τη κλήση να επαληθευτεί και να πάρει τις πληροφορίες που ζητά εφόσον έχει την άδεια.

```
import axios from "axios";
import baseURL from "../utils/baseURL";

const syncUser = (token, uid) => {
  const config = {
    headers: { Authorization: `Bearer ${token}` },
  };

  return axios.put(`${baseURL()}user/${uid}`, null, config);
};

export default syncUser;
```

Συνάρτηση 18 Συγχρονισμός πληροφοριών χρήστη

Ορίσματα συνάρτησης:

- token, μοναδικός κωδικός για τον επαληθευμένο χρήστη,
- uid, μοναδικός κωδικός που αντιστοιχεί σε έναν μόνο χρήστη,

Επιστροφή συνάρτησης:

- Σε επιτυχημένη κλήση, όλα τα στοιχεία του χρήστη ενημερωμένα με το συγκεκριμένο uid
- Σε αποτυχημένη κλήση, μήνυμα σφάλματος

8.1.2.4 Συνάρτηση updatePublication

Η συγκεκριμένη συνάρτηση έχει σκοπό να ενημερώσει μια συγκεκριμένη δημοσίευση και συγκεκριμένα να την εντάξει στη λίστα με τις εμφανείς δημοσιεύσεις ή στη λίστα με τις κρυμμένες δημοσιεύσεις. Όπως φαίνεται και παρακάτω, το token εδώ χρησιμοποιείται στην κεφαλίδα της κλίσης (headers), έτσι ώστε ο χρήστης που πραγματοποιεί τη κλήση να επαληθευτεί και να μπορεί να τροποποιήσει τις πληροφορίες που επιθυμεί εφόσον έχει την άδεια.

```
import axios from "axios";
import baseURL from "../utils/baseURL";

const updatePublication = (token, publication) => {
  const config = {
    headers: { Authorization: `Bearer ${token}` },
  };

  return axios.put(`${baseURL()}paper/${publication.id}`, publica-
tion, config);
};

export default updatePublication;
```

Συνάρτηση 19 Ενημέρωση δημοσιεύσεων χρήστη

Ορίσματα συνάρτησης:

- token, μοναδικός κωδικός για τον επαληθευμένο χρήστη,
- publication, η δημοσίευση που πρέπει να τροποποιηθεί,

Επιστροφή συνάρτησης:

- Σε επιτυχημένη κλήση, μήνυμα επιτυχίας της τροποποίησης
- Σε αποτυχημένη κλήση, μήνυμα σφάλματος

8.1.3 Βοηθητικές συναρτήσεις στο Frontend

Στο Frontend έχουν υλοποιηθεί 3 συναρτήσεις που εξυπηρετούν διάφορες λειτουργίες στο Frontend της εφαρμογής. Όπως φαίνεται και παραπάνω οι συναρτήσεις αυτές βρίσκονται μέσα στο φάκελο utils ενώ η συνολική διαδρομή είναι FE>src>utils.

8.1.3.1 Συνάρτηση baseURL

Η συνάρτηση baseURL, ανάλογα με το περιβάλλον στο οποίο βρισκόμαστε, δηλαδή σε παραγωγή (production) ή τοπικά, χρησιμοποιεί το σωστό URL για να καλέσει το Backend.

```
const baseURL = () => {
  if (process.env.NODE_ENV === "development") {
    return "http://127.0.0.1:8000/";
  } else return "https://mypubs.iee.ihu.gr/api/v1/";
};

export default baseURL;
```

Συνάρτηση 20 Βοηθητική συνάρτηση επιλογής σωστής διεύθυνσης ανάλογα με το περιβάλλον της εφαρμογής

Ορίσματα συνάρτησης:

- χωρίς ορίσματα

Επιστροφή συνάρτησης:

- το σωστό URL για να πραγματοποιηθούν οι κλήσεις ανάλογα με το περιβάλλον στο οποίο εκτελείται η εφαρμογή.

8.1.3.2 Αρχείο colorCodes

Στο αρχείο αυτό ορίζονται βασικές μεταβλητές του SCSS, οι οποίες χρωματίζουν όλη την εφαρμογή. Οι συγκεκριμένες μεταβλητές χρησιμοποιούνται στα επιμέρους components με αποτέλεσμα όλη η χρωματική παλέτα της εφαρμογής να μπορεί να τροποποιηθεί από ένα και μόνο σημείο. Με άλλα λόγια είναι μια τεχνική, η οποία βοηθά στο να διαμορφώσουμε ένα χρωματικό θέμα σε όλη την εφαρμογή.

```
$headerBackground: #009688;
$headerFontColor: #ffffff;

$mainBackground: #eeeeee;
$secondaryBackground: #03a9f4;
$shadow: 0 8px 17px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0,
0.19);

$listItemBorder: #03a9f4;
$fontColorGrey: #838383;

$buttonBackground: #03a9f4;
$buttonFont: #ffffff;

$buttonBackgroundHover: #ffffff;
$buttonFontHover: #03a9f4;

$borderGrey: #838383;
```

Συνάρτηση 21 Χρωματικοί κωδικοί

8.1.3.3 Συνάρτηση isEmailValid

Η συνάρτηση αυτή, με τη βοήθεια μιας regular expression, ελέγχει αν η δοθείσα διεύθυνση email είναι έγκυρη ή όχι.

```
const isEmailValid = (email) => {
  if (/^\w+([\\.-]?\w+)*@\w+([\\.-]?\w+)*(\.\w{2,3})+$/.test(email))
{
    return true;
  }

  return false;
};

export default isEmailValid;
```

Συνάρτηση 22 Βοηθητική συνάρτηση εντοπισμού σωστής διεύθυνσης ταχυδρομείου

Ορίσματα συνάρτησης:

- email,

Επιστροφή συνάρτησης:

- true, αν η διεύθυνση email είναι έγκυρη,
- false, αν η διεύθυνση email είναι άκυρη,

9

Ροή Χρήστη στην Εφαρμογή (user flow)

Στο κεφάλαιο αυτό, παρουσιάζονται όλα τα σενάρια χρήσης της εφαρμογής, σε περιβάλλον παραγωγής (production), και φαίνεται ο τρόπος με τον οποίο αντιδρά η εφαρμογή σε κάθε περίπτωση, αλλά και την αλληλεπίδραση του Frontend με το Backend.

Παρακάτω καταγράφονται βήμα-βήμα τα σενάρια χρήσης της εφαρμογής, συνοδευόμενα με στιγμιότυπα από την εφαρμογή.

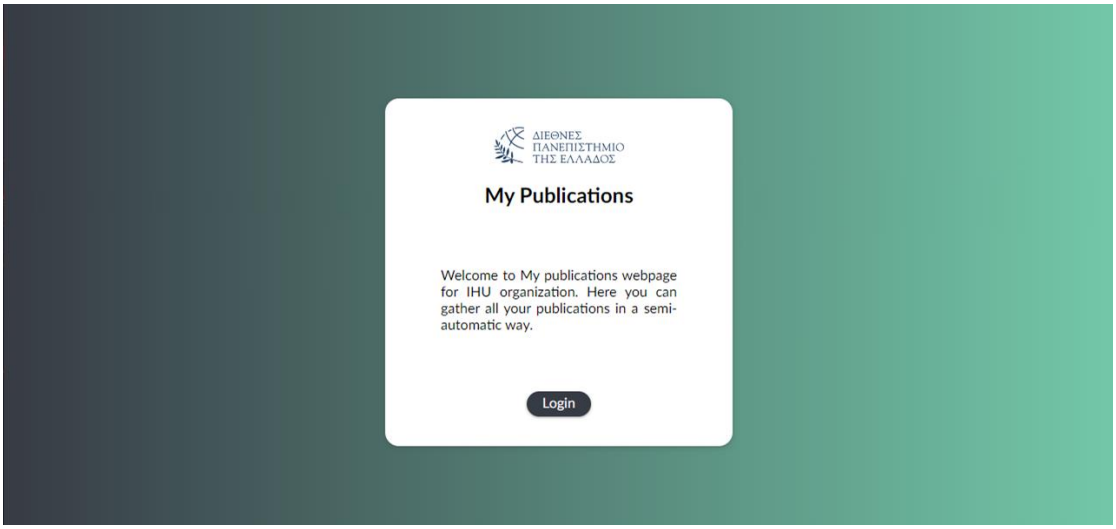
9.1 Σύνδεση - SSO

Σε αυτό το βήμα ο χρήστης καλείται να συνδεθεί στην εφαρμογή. Δεν προσφέρεται η δυνατότητα δημιουργίας λογαριασμού από τον χρήστη. Αντί αυτού έχει ρυθμιστεί ένα μηχανισμός αυτόματης σύνδεσης μέσω του πανεπιστημίου, με τεχνολογία Single Sign On. Αυτό έχει ως αποτέλεσμα ο χρήστης να μην συμπληρώνει χειροκίνητα τα στοιχεία του προφίλ/λογαριασμού του στην εφαρμογή, αλλά να αντλούνται από το πανεπιστήμιο, όπως τα έχει εισάγει ο χρήστης, στο προσωπικό του προφίλ στην σελίδα Apps (https://apps.iee.ihu.gr/), του πανεπιστημίου.

Επιπρόσθετα, κατά την πρώτη σύνδεση του χρήστη στην εφαρμογή, δημιουργείται αυτόματα λογαριασμός με όλα τα βασικά στοιχεία του χρήστη, όπως έχουν εισαχθεί στην σελίδα Apps του πανεπιστημίου. Τα στοιχεία αυτά είναι το ονοματεπώνυμο του χρήστη, το email,το Orcid και το Scopus id αν υπάρχουν. Τη στιγμή της πρώτης αυτής σύνδεσης, όταν ανακτώνται τα Orcid και Scopus id (εφόσον ο χρήστης τα έχει εισάγει στη σελίδα Apps του πανεπιστημίου), τρέχει και ο εσωτερικός μηχανισμός δημιουργίας του συνόλου δημοσιεύσεων του χρήστη. Όλες τις μετέπειτα φορές που ο χρήστης εισέρχεται στην εφαρμογή μας, όλες οι πληροφορίες που τον αφορούν υπάρχουν ήδη και δεν πραγματοποιείται καμία άλλη σύνδεση με τρίτες πήγες

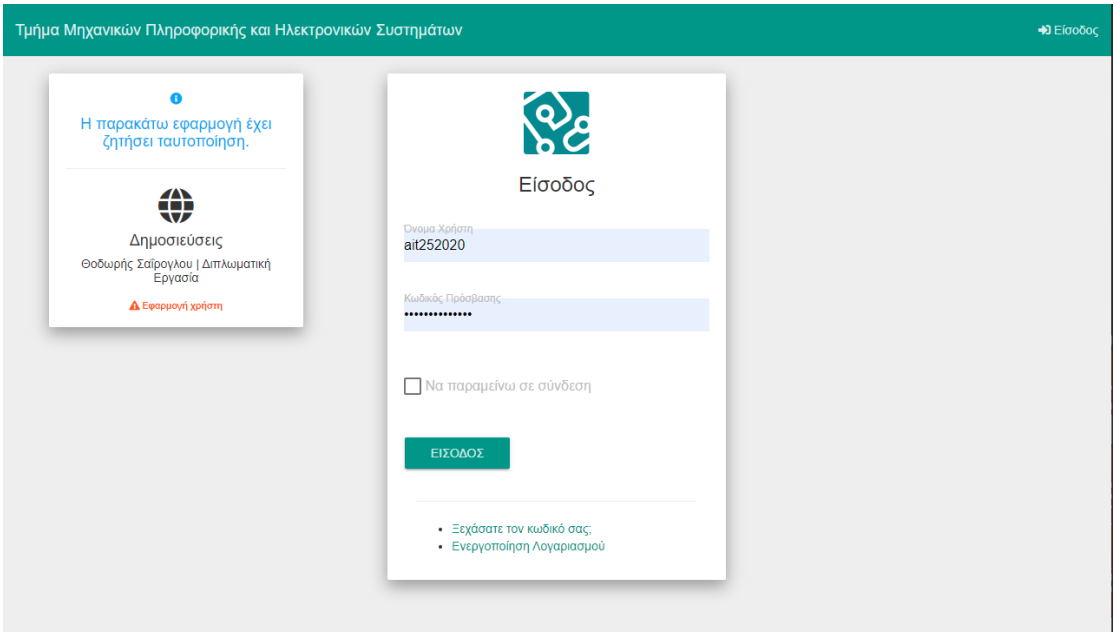
(Orcid, Scopus).

Όταν λοιπόν, ο χρήστης επισκεφθεί την εφαρμογή στη διεύθυνση https://mypubs.iee.ihu.gr/, αντικρίζει τη σελίδα σύνδεσης (login), όπως φαίνεται παρακάτω:



Εικόνα 7 Σελίδα σύνδεσης (login)

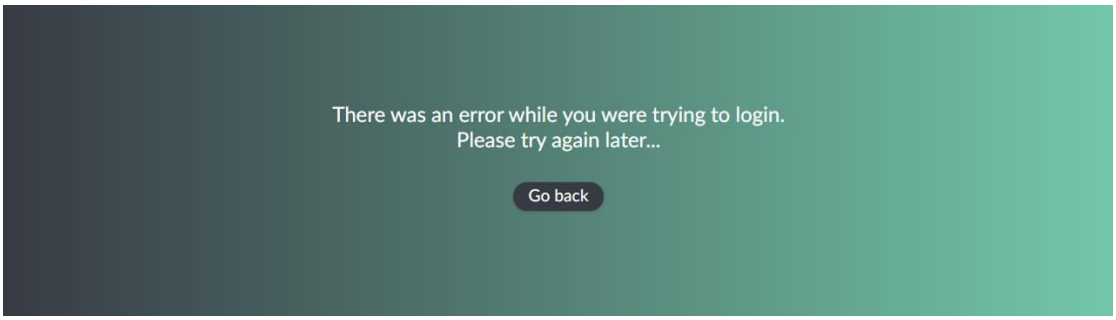
Έπειτα το πάτημα του κουμπιού "Login" ο χρήστης ανακατευθύνεται σε σελίδα του πανεπιστημίου έτσι ώστε να συνδεθεί, όπως φαίνεται παρακάτω:



Εικόνα 8 Σελίδα σύνδεσης ΔΙΠΑΕ (SSO)

Αφού ο χρήστης, πληκτρολογήσει τα στοιχεία του και συνδεθεί με το σύστημα του πανεπιστημίου, εφόσον όλα είναι σωστά, ανακατευθύνεται και πάλι πίσω στην εφαρμογή και συγκεκριμένα στην Αρχική σελίδα.

Σε περίπτωση που η σύνδεση με το πανεπιστήμιο γίνει επιτυχώς, αλλά προκύψει πρόβλημα κατά την ανακατεύθυνση πίσω στην εφαρμογή, το λάθος εντοπίζονται και προβάλλεται η παρακάτω εικόνα:

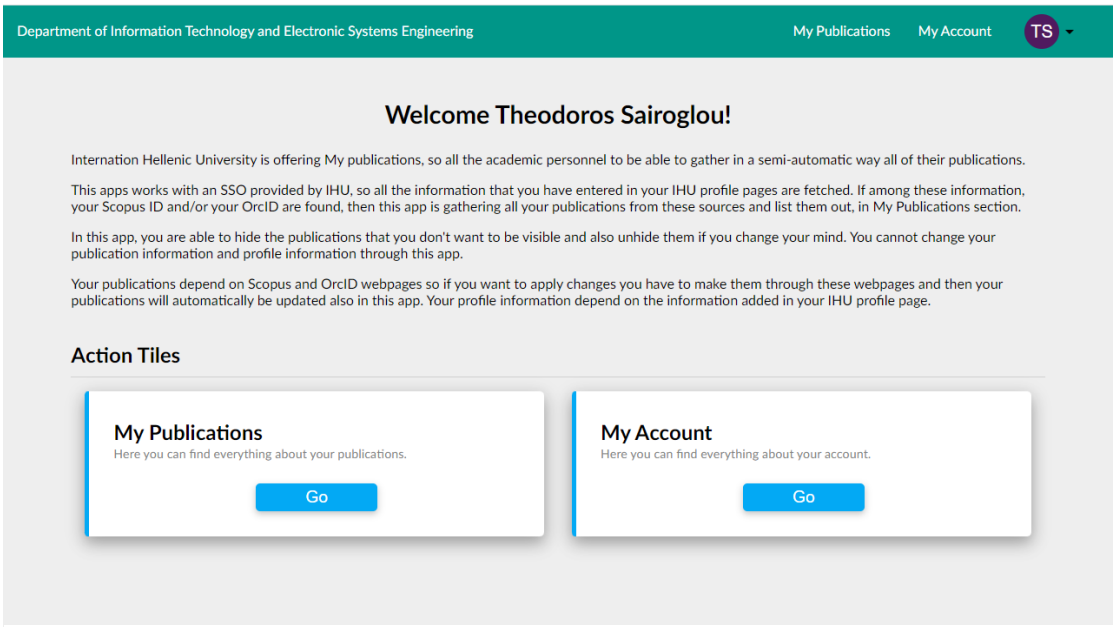


Εικόνα 9 Σελίδα αποτυχημένης σύνδεσης χρήστη

Το κουμπί “Go back” ανακατευθύνει το χρήστη πίσω στη σελίδα σύνδεσης (login) της εφαρμογής.

9.2 *Αρχική Σελίδα*

Αμέσως μετά την επιτυχημένη σύνδεση (login) του χρήστη στην εφαρμογή, ο χρήστης ανακατευθύνεται στην αρχική σελίδα, όπως φαίνεται παρακάτω:



Εικόνα 10 Αρχική σελίδα εφαρμογής έπειτα από επιτυχημένη σύνδεση χρήστη

Στην αρχική σελίδα, υπάρχει η κεφαλίδα η οποία περιέχει:

- τον τίτλο της εφαρμογής,
- το σύνδεσμο “My Publications” που ανακατευθύνει στη σελίδα με τη λίστα των δημοσιεύσεων του χρήστη,
- το σύνδεσμο “My Account” που ανακατευθύνει στη σελίδα με τα στοιχεία του χρήστη, όπως αντλήθηκαν από τη σελίδα Apps του πανεπιστημίου,
- Τα αρχικά του χρήστη που συνδέθηκε, τα οποία κρύβουν ένα μενού το οποίο εμφανίζεται όταν ο χρήστης πατήσει πάνω στα αρχικά του, και του επιτρέπει να αποσυνδεθεί.

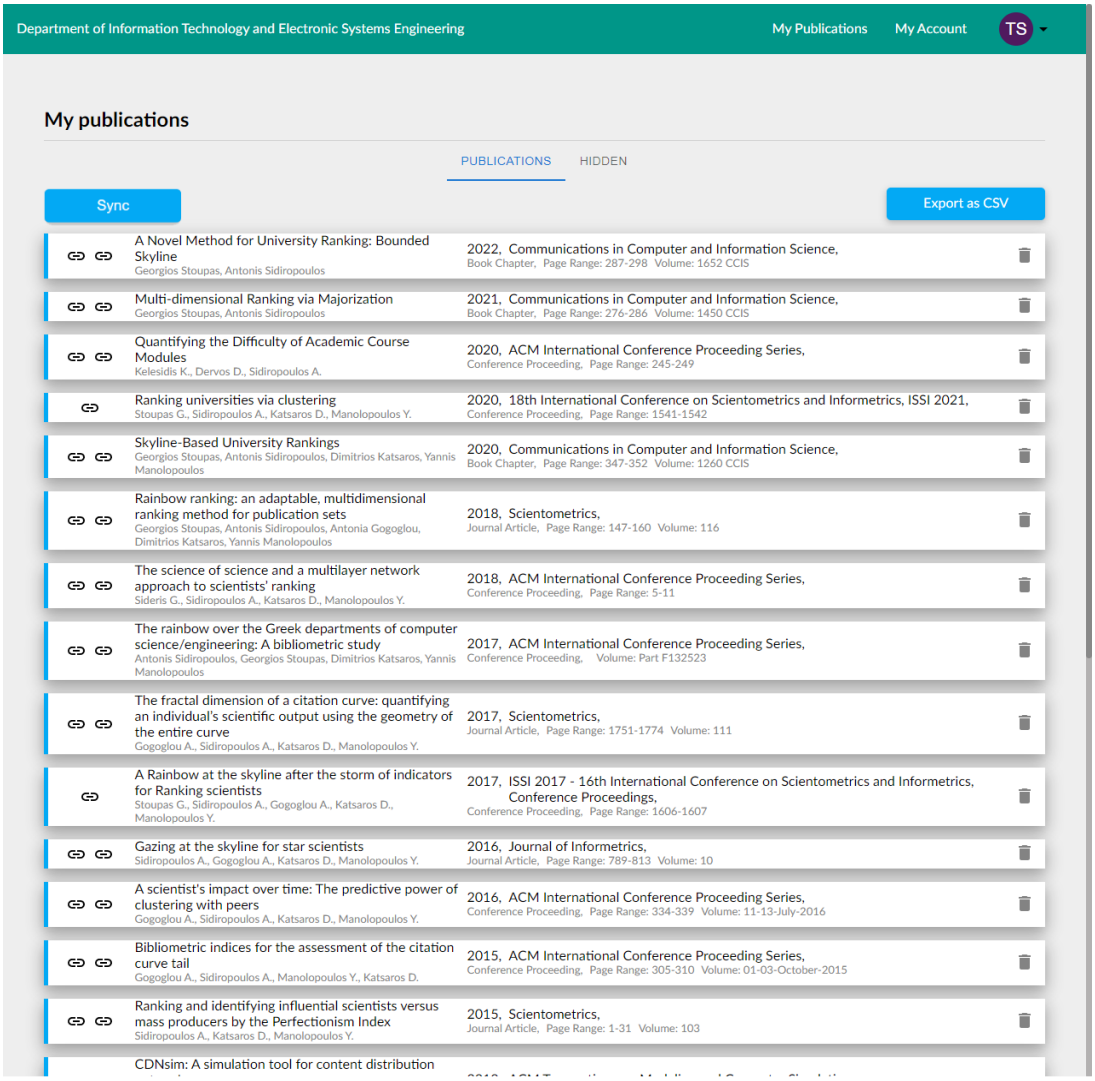
Το βασικό μέρος της αρχικής σελίδα περιλαμβάνει:

- κείμενο καλωσορίσματος του χρήστη που συνδέθηκε,
- περιγραφή της σελίδας και τι μπορεί να κάνει ο χρήστης εντός της εφαρμογής,
- το σύνδεσμο “My Publications” που ανακατευθύνει στη σελίδα με τη λίστα των δημοσιεύσεων του χρήστη,

- το σύνδεσμο “My Account” που ανακατευθύνει στη σελίδα με τα στοιχεία του χρήστη, όπως αντλήθηκαν από τη σελίδα Apps του πανεπιστημίου.

9.3 *Σελίδα Δημοσιεύσεων Χρήστη*

Πατώντας το κουμπί “My Publications” είτε από την κεφαλίδα, είτε από το βασικό μέρος της εφαρμογής ο χρήστης ανακατευθύνεται στη σελίδα δημοσιεύσεων του χρήστη. Η σελίδα αυτή περιλαμβάνει όλο το ακαδημαϊκό έργο του χρήστη όπως αντλήθηκε από το Scopus και το Orcid και συγχωνεύθηκε, σύμφωνα με τον αλγόριθμο της εφαρμογής.



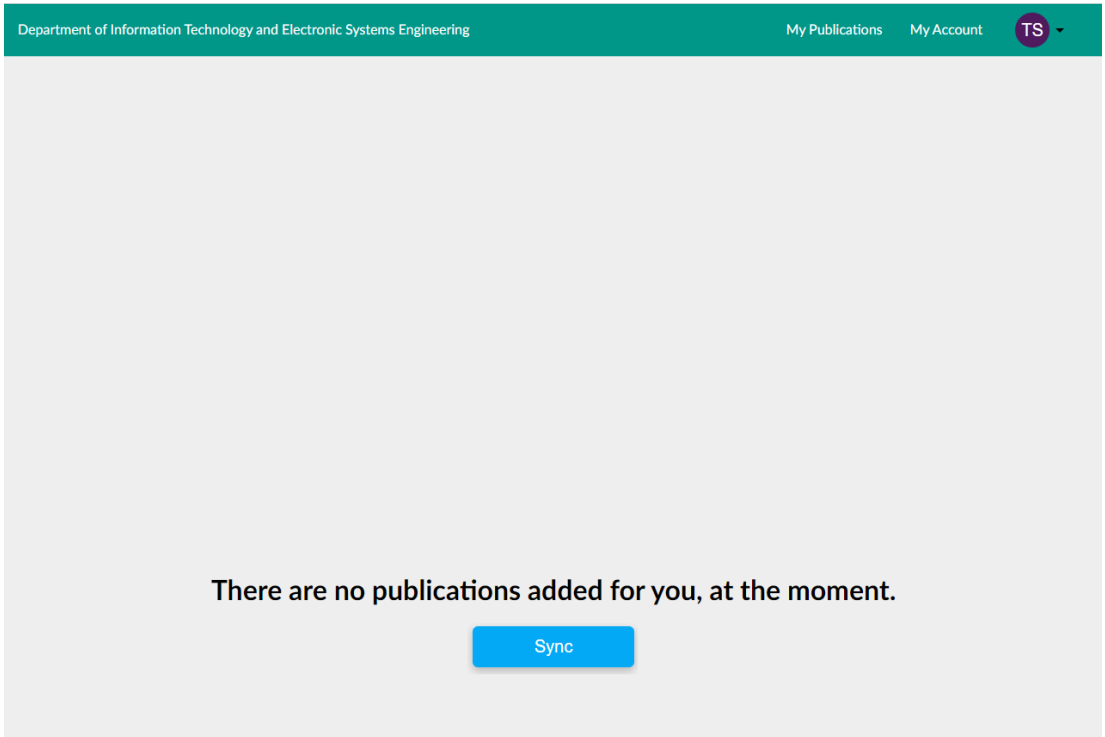
Εικόνα 11 Σελίδα δημοσιεύσεων χρήστη

Στην περίπτωση που ο χρήστης είχε εισάγει το Orcid ή/και το Scopus id στη σελίδα Apps του πανεπιστημίου, στη σελίδα των δημοσιεύσεων εμφανίζεται μια λίστα όπως στην φωτογραφία

παραπάνω. Σε αυτή τη σελίδα ο χρήστης μπορεί να ενημερώσει τις δημοσιεύσεις του πατώντας το κουμπί “Sync”. Πατώντας το συγκεκριμένο κουμπί ξεκινάει όλη η διαδικασία συγκέντρωσης του ακαδημαϊκού έργου του χρήστη από το Scopus και το Orcid, εφόσον έχουν εισαχθεί τα εν λόγω αναγνωριστικά στο σύστημα. Έπειτα πραγματοποιείται όλη η διαδικασία συγχώνευσης των δημοσιεύσεων, όπως έχει περιγραφεί, και εάν εντοπιστούν αλλαγές ενημερώνεται η βάση δεδομένων και κατά επέκταση και το UI του χρήστη.

Πέρα από την ενημέρωση των δημοσιεύσεών του, ο χρήστης μπορεί να κατεβάσει σε αρχείο csv τη λίστα των δημοσιεύσεων του. Επιπλέον δίνεται η δυνατότητα στο χρήστη να “κρύψει” τις δημοσιεύσεις που θέλει πατώντας στο εικονίδιο του κάδου. Αυτό που συμβαίνει πατώντας αυτό το κουμπί, είναι να μεταφερθεί η συγκεκριμένη δημοσίευση στο tab “Hidden” και να φύγει από το tab “Publications”. Όπως είναι φυσικό, ο χρήστης μπορεί να πραγματοποιήσει και την αντίθετη διαδικασία, δηλαδή να κάνει πάλι “ορατή” μια δημοσίευση που έχει μεταφέρει στο tab “Hidden”.

Στην περίπτωση που ο χρήστης δεν είχε εισάγει το Orcid και το Scopus id στη σελίδα Apps του πανεπιστημίου, όπως είναι φυσιολογικό δεν υπάρχουν δημοσιεύσεις να προβληθούν στη σελίδα των δημοσιεύσεων. Έτσι αυτό που βλέπει ο χρήστης σε αυτή την περίπτωση φαίνεται παρακάτω:

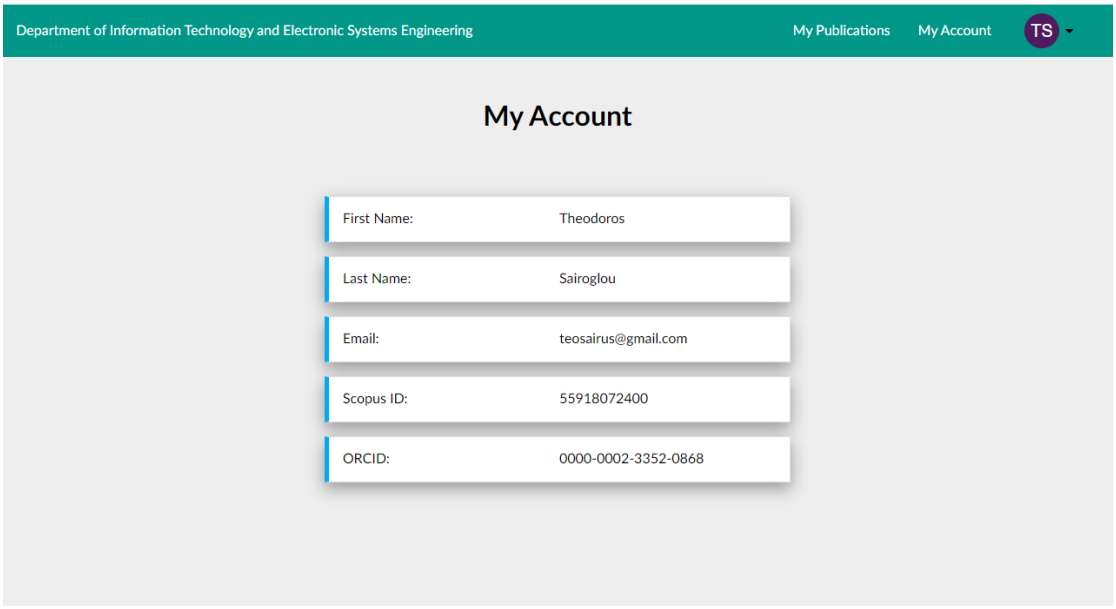


Εικόνα 12 Σελίδα δημοσιεύσεων χρήστη χωρίς ακαδημαϊκό έργο

9.4 Σελίδα Προφίλ Χρήστη

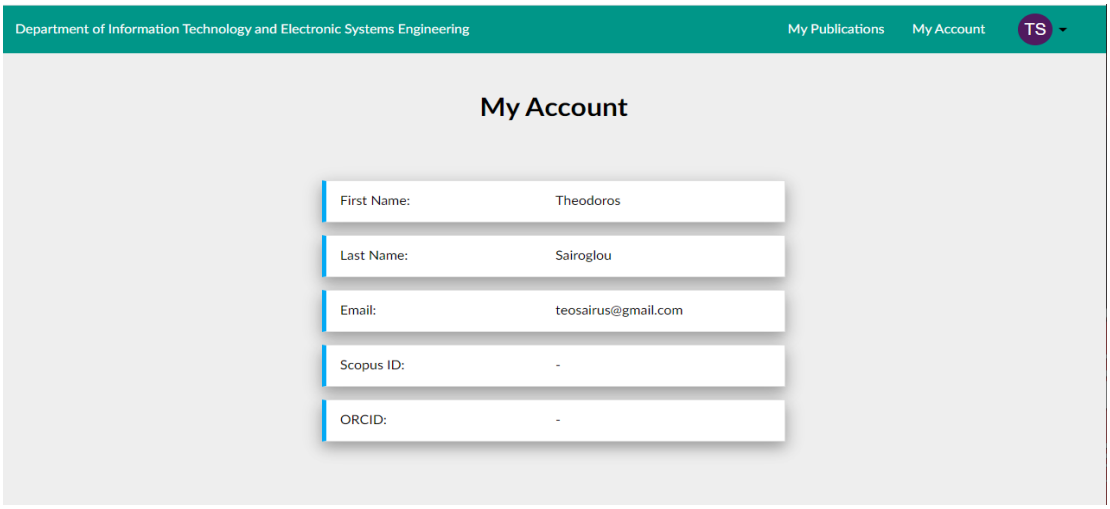
Πατώντας το κουμπί “My Account” είτε από την κεφαλίδα, είτε από το βασικό μέρος της εφαρμογής ο χρήστης ανακατευθύνεται στη σελίδα προφίλ του χρήστη. Η σελίδα αυτή περιλαμβάνει τις βασικές πληροφορίες του χρήστη, όπως έχουν εισαχθεί στην σελίδα Apps του πανεπιστημίου.

Σε περίπτωση που ο χρήστης έχει εισάγει όλες τις βασικές πληροφορίες στην σελίδα Apps του πανεπιστημίου, η σελίδα χρήστη της εφαρμογής εμφανίζεται ως εξής:



Εικόνα 13 Σελίδα προφίλ χρήστη

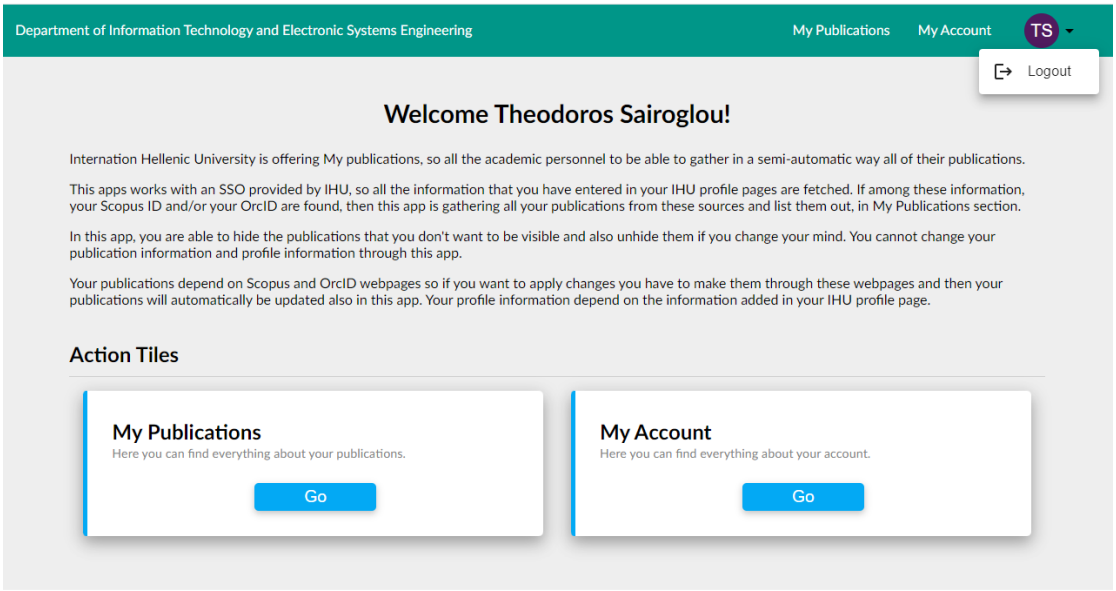
Σε περίπτωση που ο χρήστης δεν έχει εισάγει όλες τις βασικές πληροφορίες στην σελίδα Apps του πανεπιστημίου, η σελίδα χρήστη της εφαρμογής εμφανίζεται ως εξής:



Εικόνα 14 Σελίδα προφίλ χρήστη

9.5 *Αποσύνδεση*

Για να μπορέσει ο χρήστης να αποσυνδεθεί, πρέπει να πατήσει επάνω στα αρχικά του γράμματα στην κεφαλίδα της εφαρμογής. Πατώντας πάνω στα αρχικά, ένα κρυφό μενού εμφανίζεται, όπου υπάρχει η επιλογή αποσύνδεσης (Logout) του χρήστη, όπως φαίνεται στην φωτογραφία παρακάτω. Πατώντας αποσύνδεση (logout), ο χρήστης οδηγείται εκτός της εφαρμογής και ανακατευθύνεται στη σελίδα σύνδεσης (login).



Εικόνα 15 Μενού αποσύνδεσης χρήστη

10

Επίλογος

Στο κεφάλαιο αυτό γίνεται μια σύνοψη της υλοποίησης όλης της εφαρμογής, εξάγονται κάποια συμπεράσματα και προτείνονται μελλοντικές επεκτάσεις.

10.1 Σύνοψη και συμπεράσματα

Η υλοποίηση της εφαρμογής έρχεται να λύσει το πρόβλημα των πολλαπλών πηγών με διαφορετική ποιότητα πληροφορίας για το ακαδημαϊκό έργο των επιστημόνων, με έναν ημιαυτόματο τρόπο. Σίγουρα η αρχική υλοποίηση αφορά μόνο το επιστημονικό προσωπικό του ΔΙ.ΠΑ.Ε μιας που η σύνδεση γίνεται μέσω του μηχανισμού SingleSignOn της σχολής,. Βέβαια, η υλοποίηση είναι παραμετροποιήσιμη με αποτέλεσμα να μπορεί να δεχτεί διάφορους μηχανισμούς SSO από οποιονδήποτε πάροχο.

Η εφαρμογή που δημιουργήθηκε, έχει τη δυνατότητα να συλλέξει και να συγχωνεύσει το ακαδημαϊκό έργο οποιουδήποτε επιστήμονα και μπορεί να αποτελέσει την μοναδική "πηγή αλήθειας" (source of truth) για το επιστημονικό έργο που έχει κάποιος να επιδείξει. Είναι μια πρώτη έκδοση και θέτει τις βάσεις για να αποτελέσει τον πυρήνα ενός συστήματος που μπορεί να χρησιμοποιηθεί σε οποιαδήποτε εκπαιδευτική δομή ανώτατης εκπαίδευσης, ερευνητικό κέντρο, επιχειρήσεις και οργανισμούς που ασχολούνται με την έρευνα.

Τη δεδομένη στιγμή, η εφαρμογή συγκεντρώνει δεδομένα από δύο πηγές, το Scopus και το Orcid. Αφού συλλεχθεί η πληροφορία από τις εξωτερικές αυτές πηγές μετασχηματίζεται και συγχωνεύεται μετά δεδομένα της βάσης δεδομένων. Αυτό και μόνο αρκεί για να μπορέσει ο χρήστης να συλλέξει όλο το ακαδημαϊκό του έργο με το πιο πλήρη τρόπο, μετά από 2 clicks στην εφαρμογή, και να μπορεί να το κατεβάσει με τη μορφή αρχείου CSV.,

10.2 Μελλοντικές επεκτάσεις

Όπως έχει ήδη αναφερθεί, η εφαρμογή αυτή μπορεί να αποτελέσει τη βάση ενός πιο πολύπλοκου συστήματος. Παρακάτω προτείνονται μελλοντικές επεκτάσεις που μπορούν να πλαισιώσουν την ήδη υπάρχουσα υλοποίηση και να δημιουργήσουν ένα πολυσύνθετο σύστημα.

10.2.1 Σύνδεση χρηστών και SSO

Αυτή τη στιγμή η εφαρμογή δέχεται χρήστες που ανήκουν μόνο στο ΔΙ.ΠΑ.Ε λόγω της σύνδεσης των χρηστών διαμέσου του μηχανισμού SSO του πανεπιστημίου. Πέρα από το ΔΙ.ΠΑ.Ε, η εφαρμογή μπορεί να "κουμπώσει" με οποιοδήποτε άλλο πάροχο SSO μηχανισμού είτε παράλληλα με το μηχανισμό SSO του ΔΙ.ΠΑ.Ε και να δέχεται πολλαπλούς τρόπους σύνδεσης (πχ Google, Apple), είτε ανεξάρτητα τρέχοντας ως instance σε μια ανεξάρτητη δομή άλλου εκπαιδευτικού ιδρύματος.

Συμπληρωματικά σε οποιοδήποτε SSO μηχανισμό, η εφαρμογή μπορεί να υποστηρίξει και τη δημιουργία χρηστών και τη σύνδεση απευθείας από την εφαρμογή. Η υλοποίηση υποστηρίζει ήδη αυθεντικοποίηση μέσω του Oauth2 πρωτοκόλλου, η οποία μπορεί να χρησιμοποιηθεί απροβλημάτιστα και σε νέο μηχανισμό δημιουργίας και σύνδεσης νέων χρηστών.

10.2.2 Εξωτερικές πηγές δεδομένων

Η εφαρμογή αυτή τη στιγμή έχει τη δυνατότητα να συλλέξει δεδομένα από δύο εξωτερικές πηγές, το Scopus και το Orcid. Οι δομές που έχουν δημιουργηθεί μπορούν να υποστηρίξουν τη συγχώνευση δεδομένων από οποιαδήποτε πηγή.

Μελλοντικά μπορούν να προστεθούν και άλλες πηγές, όπως πχ Google Scholar, Aminer κτλ. έτσι ώστε η εφαρμογή να μπορεί να παρέχει ακόμη πληρέστερες πληροφορίες στον χρήστη.

10.2.3 Βελτιστοποίηση κώδικα και υποδομής

Η υπάρχουσα υλοποίηση είναι η πρώτη έκδοση ενός συστήματος που μπορεί να χρησιμοποιηθεί από πληθώρα χρηστών. Ενώ λοιπόν έχουν χρησιμοποιηθεί "best practices" κατά τη δημιουργία της εφαρμογής, πάντοτε έχοντας κατά νου η υλοποίηση να είναι επεκτάσιμη και μακριά από bugs, όταν η εφαρμογή θα υποδεχτεί αρκετούς χρήστες ενδεχομένως θα πρέπει να αλλάξει ο server και να βελτιωθούν οι υποδομές που εκτελούν την εφαρμογή.

Σε επίπεδο κώδικα, ενώ δεν έχουν παρατηρηθεί προβλήματα απόδοσης, πάντα υπάρχει χώρος για βελτίωση. Θα πρέπει λοιπόν τόσο το Frontend όσο και το Backend να ελεγχθούν και να υλοποιηθούν βελτιώσεις ώστε ο κώδικας γίνει ακόμη αποδοτικότερος και με γρηγορότερες επιδόσεις σε θέματα χρόνου εκτέλεσης.

Επιπρόσθετα ο μηχανισμός συγχώνευσης έχει τη δυνατότητα να βελτιωθεί και να γίνει ακόμη πιο έξυπνος. Στην παρούσα υλοποίηση η συγχώνευση και ενημέρωση εγγραφών της βάσης γίνεται απλοϊκά συγκρίνοντας συγκεκριμένα πεδία, ενώ ως προς την ενημέρωση συγκρίνεται το πλήθος των πεδίων μιας εγγραφής με το πλήθος των πεδίων της άλλης εγγραφής. Έτσι μια εγγραφή με περισσότερα πεδία από την άλλη, θεωρείται πιο πλήρης. Η σύγκριση αυτή θα μπορούσε να μεταφερθεί σε επίπεδο περιεχομένου των πεδίων και όχι ως προς το πλήθος των πεδίων που συμβαίνει αυτή τη στιγμή.

11

Βιβλιογραφία

[1] “React – A JavaScript library for building user interfaces.” https://reactjs.org/ (accessed Feb. 17, 2023).

[2] “Home v6.8.1.” https://reactrouter.com/en/main (accessed Feb. 17, 2023).

[3] “MUI Core: Ready to use components, free forever.” https://mui.com/core/ (accessed Feb. 17, 2023).

[4] “IT Login.” https://login.iee.ihu.gr/ (accessed Feb. 17, 2023).

[5] “FastAPI.” https://fastapi.tiangolo.com/ (accessed Feb. 17, 2023).

[6] “Django,” *Django Project*. https://www.djangoproject.com/ (accessed Feb. 17, 2023).

[7] “SQLAlchemy - The Database Toolkit for Python.” https://www.sqlalchemy.org/ (accessed Feb. 17, 2023).

[8] “Scopus preview - Scopus - Welcome to Scopus.” https://www.scopus.com/home.uri (accessed Feb. 17, 2023).

[9] “ORCID.” https://orcid.org/ (accessed Feb. 17, 2023).