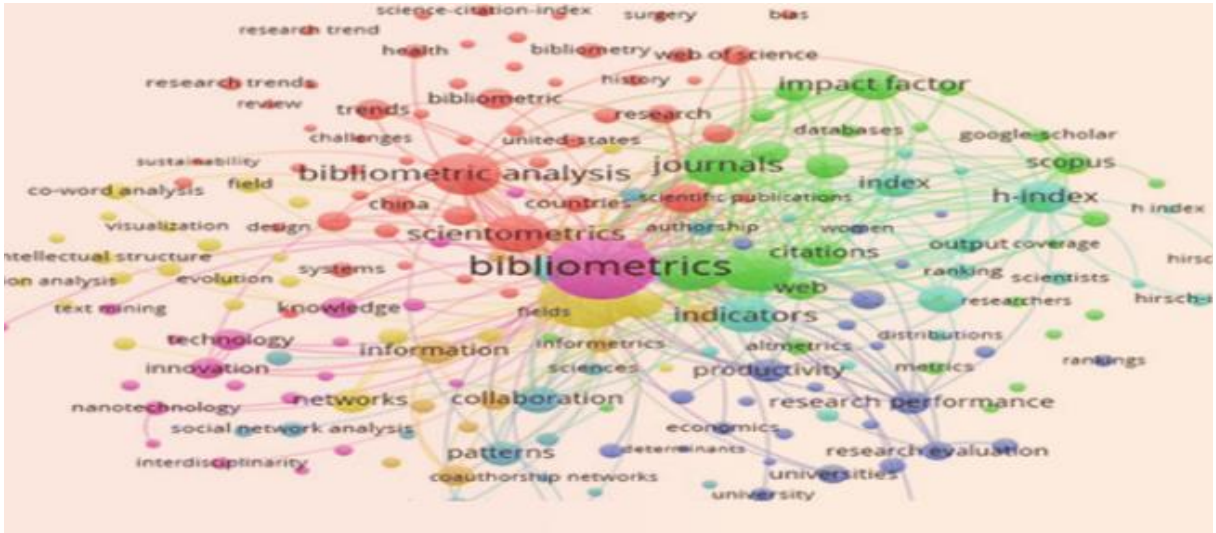
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αυτοματοποιημένο σύστημα συλλογής και οργάνωσης δεδομένων ερευνητών και δημοσιεύσεων από το Scopus/Elsevier για έναν πανεπιστημιακό οργανισμό.



Του φοιτητή
Κωνσταντίνου Μολοχίδη
Αρ. Μητρώου: 154496

Επιβλέπων
Αντώνης Σιδηρόπουλος
Επίκουρος Καθηγητής

Ημερομηνία 10/09/2021

Τίτλος Δ.Ε. Αυτοματοποιημένο σύστημα συλλογής και
οργάνωσης δεδομένων ερευνητών και δημοσιεύσεων
από το Scopus/Elsevier για έναν πανεπιστημιακό οργανισμό.
Κωδικός Δ.Ε. 21178
Ονοματεπώνυμο φοιτητή Κωνσταντίνος Μολοχίδης
Ονοματεπώνυμο εισηγητή Σιδηρόπουλος Αντώνης
Ημερομηνία ανάληψης Δ.Ε. 10 Μαρτίου 2021
Ημερομηνία περάτωσης Δ.Ε. 6 Φεβρουαρίου 2022

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Περίληψη

Η ανάπτυξη και η διάδοση του παγκόσμιου ιστού (World Wide Web, WWW) αντιπροσωπεύει μια επανάσταση στην ενημέρωση, με γρήγορη και πρακτική διανομή και αποθήκευση δεδομένων διαθέσιμων σε όλη την επιστημονική κοινότητα. Ένα από τα πιο σημαντικά παραδείγματα αποθήκευσης και διανομής σημαντικών πληροφοριών είναι η ανάπτυξη επιστημονικών βάσεων δεδομένων, η σημασία των οποίων αναγνωρίστηκε νωρίς.

Όμως, παρά τον μεγάλο όγκο δεδομένων που προσφέρουν οι πλέον διαδεδομένες βιβλιογραφικές βάσεις δεδομένων, δεν παρέχουν λειτουργίες σύγκρισης είτε μεταξύ των ερευνητών είτε μεταξύ ομάδων ερευνητών (ακαδημαϊκά ιδρύματα, τμήματα, εργαστήρια κλπ).

Στην παρούσα εργασία εξετάζεται η πλειονότητα των μετρικών που αφορούν τους ερευνητές και το ερευνητικό τους έργο. Στοιχεία που αφορούν τις δημοσιεύσεις τους είναι ο αριθμός h-index, ο αριθμός των αναφορών και δημοσιεύσεων κλπ.

Επικεντρωνόμαστε στην βάση δεδομένων της Scopus. Σχεδιάσαμε και αναπτύξαμε εργαλείο σε python για την αυτοματοποιημένη ανάκτηση των δεδομένων καθώς και την πλατφόρμα για την διεπαφή του χρήστη με το εργαλείο.

Automated system for collecting and organizing metrics and publications by Scopus / Elsevier for a university.

Konstantinos Molohidis

Abstract

The development and dissemination of the World Wide Web (WWW) represents a revolution in information, with the rapid and practical distribution and storage of data available throughout the scientific community. One of the most important examples of the storage and distribution of important data is the development of scientific databases, the importance of which was recognized early on.

However, despite the large volume of data offered by the most widespread bibliographic databases, they do not provide comparison functions either between researchers or between groups of researchers (academic institutions, departments, laboratories, etc.).

This paper examines the majority of metrics related to researchers and their research work. Data regarding their publications are the h-index number, the number of reports and publications, etc.

We focus on the Scopus database. We designed and developed a tool in python for the automated data recovery as well as the platform for the user interface with the tool.

Ευχαριστίες

Η εργασία αυτή δεν θα είχε περατωθεί χωρίς την συμπαράσταση και υποστήριξη του επιβλέποντα καθηγητή, κ. Αντώνη Σιδηρόπουλο. Όποτε χρειαζόταν εύρισκε τον χρόνο για δημιουργικές συζητήσεις που αποτελούσαν και έναν ουσιαστικό τρόπο εμψύχωσης. Η καθοδήγησή του και οι συμβουλές του στα κομβικά σημεία της εργασίας ήταν πολύ ωφέλιμες και η εμμονή του για την ποιότητα, καθοριστική. Ολόθερμα τον ευχαριστώ που πίστεψε στη δουλειά μου αλλά και στις ικανότητές μου. Ελπίζω το αποτέλεσμα να δικαιώνει και τις δικές του προσδοκίες.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Κώδικα

Συντομογραφίες

HTML	HyperText Markup Languange
CSS	Cascading Style Sheets
API	Application Programming Interface
REST	Representational State Transfer
DRF	Django Rest Framework
CRUD	Create Read Update Delete
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
CSR	Client Side Rendering
SSR	Server Side Rendering
SQL	Structured Query Language

Κεφάλαιο 1ο: Βιβλιογραφικές βάσεις δεδομένων

1.1 Εισαγωγή

Η ανάπτυξη και η διάδοση του παγκόσμιου ιστού (World Wide Web, WWW) αντιπροσωπεύει μια επανάσταση στην ενημέρωση, με γρήγορη και πρακτική διανομή και αποθήκευση δεδομένων διαθέσιμων σε όλη την επιστημονική κοινότητα. Ένα από τα πιο σημαντικά παραδείγματα αποθήκευσης και διανομής σημαντικών πληροφοριών είναι η ανάπτυξη **επιστημονικών βάσεων δεδομένων**, η σημασία των οποίων αναγνωρίστηκε νωρίς.

Στην έρευνα, επιστημονικές βάσεις δεδομένων θεωρούνται οι βάσεις δεδομένων που περιέχουν στοιχεία και πληροφορίες για δημοσιευμένα άρθρα σε περιοδικά STM[1], πολύτιμη πηγή πληροφοριών για έναν ερευνητή. Υπάρχουν διάφορες βιβλιογραφικές βάσεις δεδομένων.

Διεθνώς, οι πλέον καθιερωμένες που περιλαμβάνουν βιβλιογραφικές εγγραφές επιστημονικών δημοσιεύσεων σε παγκόσμιο επίπεδο και στοιχεία για τις αναφορές μεταξύ τους, είναι τα συστήματα Web of Science (της εταιρίας Thomson Reuters), Scopus (της Elsevier) και Google Scholar (της Google). Υπάρχουν επίσης πιο εξειδικευμένες βάσεις δεδομένων, που εστιάζονται σε θεματικές περιοχές όπως η PubMed, για θέματα συναφή με την ιατρική, η ERIC, για θέματα συναφή με την εκπαίδευση, δηλαδή καμία από τις βάσεις δεδομένων δεν μπορεί να θεωρηθεί ως «πλήρης», δηλαδή καμία δεν περιέχει απολύτως όλες τις δημοσιεύσεις, αλλά αλληλοσυμπληρώνονται. Έτσι ένας ερευνητής συνήθως αναζητά πληροφορίες σε περισσότερες από μία επιστημονικές βάσεις δεδομένων.

Το σύστημα Google Scholar, παρά τον τεράστιο αριθμό πηγών που περιλαμβάνει, δεν είναι κατάλληλο για βιβλιομετρικές αναλύσεις που αναφέρονται σε επίπεδο χωρών ή οργανισμών, λόγω της έλλειψης μεταδεδομένων που απαιτούνται για την ταυτοποίηση των δημοσιεύσεων και της απουσίας κριτηρίων που διασφαλίζουν την ποιότητα των δημοσιεύσεων που περιλαμβάνονται στο σύστημα. Όσον αφορά τα συστήματα Web of Science και Scopus, και τα δύο διασφαλίζουν τη διάθεση αναλυτικών μεταδεδομένων και την ποιότητα των δημοσιεύσεων που περιλαμβάνουν. Το σύστημα Web of Science (WoS) είναι η παλαιότερη βάση δεδομένων επιστημονικών δημοσιεύσεων με εγγραφές που ξεκινούν από το 1900. Αντλεί δεδομένα από περισσότερα από 12.000 περιοδικά τα οποία υπόκεινται σε αξιολόγηση κριτών (peer-review). Στο νεότερο σύστημα Scopus, ευρετηριάζονται πάνω από 18.500 τίτλοι επιστημονικών περιοδικών, οι οποίοι διευρύνονται συνεχώς, με στοιχεία όμως αναφορών σε δημοσιεύσεις που ξεκινούν μετά το 1996 [1].

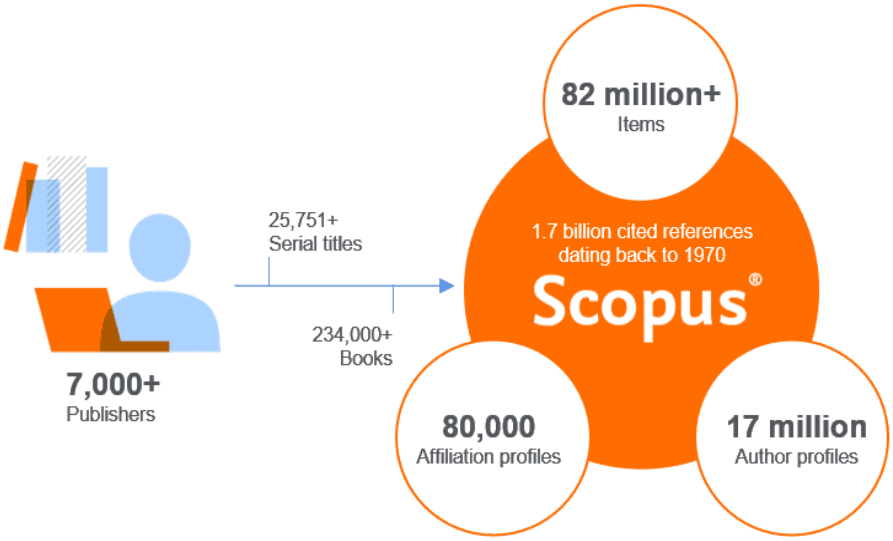
Η χρήση των δεδομένων από τις βιβλιογραφικές βάσεις πρέπει να υπόκεινται σε αυστηρό ποιοτικό έλεγχο, πχ. υπάρχει αδυναμία ποιοτικής διαφοροποίησης των αναφορών (εξομοιώνονται αρνητικές και θεωρητικές αναφορές), υπάχρει δυσκολία σύγκρισης μεταξύ διαφορετικών επιστημονικών πεδίων επειδή το κάθε επιστημονικό πεδίο έχει ιδιαίτερη δομή δημοσιεύσεων και αναφορών (publication and citation pattern) κλπ (Ζωντανός & Κατρανίδης, 2009)[2]. Επίσης δίνεται ιδιαίτερη βαρύτητα σε δημοσιεύσεις σε επιστημονικά περιοδικά με διεθνή απήχηση. Όμως στις κοινωνικές και στις ανθρωπιστικές επιστήμες βιβλία και κεφάλαια βιβλίων αποτελούν μια σημαντική πρακτική δημοσίευσης ερευνητικών εργασιών, που ακολουθούν συγκριτικά περισσότερο τις εθνικές τάσεις. Έτσι η συγγραφή των εργασιών στην εθνική γλώσσα και η δημοσίευση σε εθνικά περιοδικά να εμφανίζονται συχνότερα απ' ό,τι σε άλλες Επιστήμες. Για όλους τους παραπάνω λόγους οι περισσότεροι ερευνητές τονίζουν πως οι βιβλιομετρικοί δείκτες είναι ενδεικτικοί της απήχησης (impact) και της προβολής (visibility) των δημοσιεύσεων και σε καμία περίπτωση δεν μπορούν να αποτιμήσουν την ερευνητική ποιότητα και τάσσονται υπέρ του συνδυασμού της μεθόδου αξιολόγησης από ομότιμους με τη μέθοδο ανάλυσης παραθέσεων [2].

[1] STM: Scientific, technical and medical (επιστημονικά, τεχνολογικά και ιατρικά)

1.2 Scopus

Το Scopus είναι μια διεθνής βιβλιογραφική βάση δεδομένων. Ισχυρίζεται [3] ότι βιβλιογραφεί πάνω από 25.000 έγκριτα επιστημονικά περιοδικά και 234.000 βιβλία, θετικών, τεχνολογικών και κοινωνικών επιστημών, από 7000 εκδότες, αναφέροντας ότι είναι "η μεγαλύτερη βιβλιογραφική βάση δεδομένων που έχει υπάρξει», παρέχοντας περιλήψεις και παραπομπές για ακαδημαϊκά άρθρα, με 17 εκατομμύρια προφίλ συγγραφέων τα οποία καλύπτουν συνεργασίες, αριθμούς από εκδόσεις και δεδομένα της βιβλιογραφίας τους, καθώς και αναφορές και λεπτομέρειες στον αριθμό των παραπομπών που καθένα από τα δημοσιευμένα τεκμήρια έχει αποσπάσει.

Έχει ενημερωτικές ετικέτες που επιτρέπουν στους εγγεγραμμένους χρήστες να ακολουθήσουν της αλλαγές σε ένα προφίλ και την υποδομή, να αριθμήσει τον δείκτη απήχησης h index των συγγραφέων (wikipedia/scopus). Ο κατάλογος των βιβλιογραφούμενων τίτλων βασίζεται στην ζήτηση των χρηστών (user demand) και στην έρευνα αγοράς (market research). Περιλαμβάνει 27 εκατομμύρια περιλήψεις με αναφορές από το 1966, και καλύπτει Αμερικάνικη, Ευρωπαϊκή και Ασιατική (του Ειρηνικού) βιβλιογραφία στην αγγλική, γαλλική, ισπανική κλπ. γλώσσες (στην ιστοσελίδα https://en.wikipedia.org/wiki/Scopus αναφέρεται κάλυψη σε 40 γλώσσες)(για τα στοιχεία βλ. σχήμα 1.1 από την ιστοσελίδα της Scopus). Η πρόσβαση στα άρθρα περιοδικών και στις αναφορές που περιλαμβάνονται σε αυτά τα άρθρα, επιτρέπει στον χρήστη να αναζητήσει πληροφόρηση τόσο, χρονολογικά, προς τα πίσω, σε άρθρα που το κυρίως άρθρο αναφέρει (references), όσο και προς τα εμπρός (σε άρθρα που το κυρίως άρθρο βιβλιογραφείται (citations)). Έτσι παρέχεται στον χρήστη εκτενέστερη αλλά και πιο σύγχρονη πληροφόρηση, βοηθώντας τον στη έρευνα. Η πλήρης χρήση των παρεχόμενων υπηρεσιών είναι με συνδρομή (είτε ατομική είτε μέσω ακαδημαϊκών ή ερευνητικών φορέων) [3 - 4].



Σχήμα 1.1: Στατιστικά στοιχεία της Scopus database, όπως παρατίθενται στην ίδια την ιστοσελίδα https://www.elsevier.com/solutions/scopus/how-scopus-works (ανάκτηση 1/9/2021)

Η αναζήτηση στο scopus γίνεται με ποικίλους, φιλικούς προς τον χρήστη, τρόπους. Η βασική πλοήγηση γίνεται σε έγγραφα, σε συγγραφείς και σε ιδρύματα με τις κατάλληλες λέξεις κλειδιά που θέτει ο χρήστης (**Σχήμα. 1.2**).

Α) σε άρθρα: ο χρήστης έχει τη δυνατότητα επιλογής ανάμεσα σε πολλές κατηγορίες που προέρχονται:

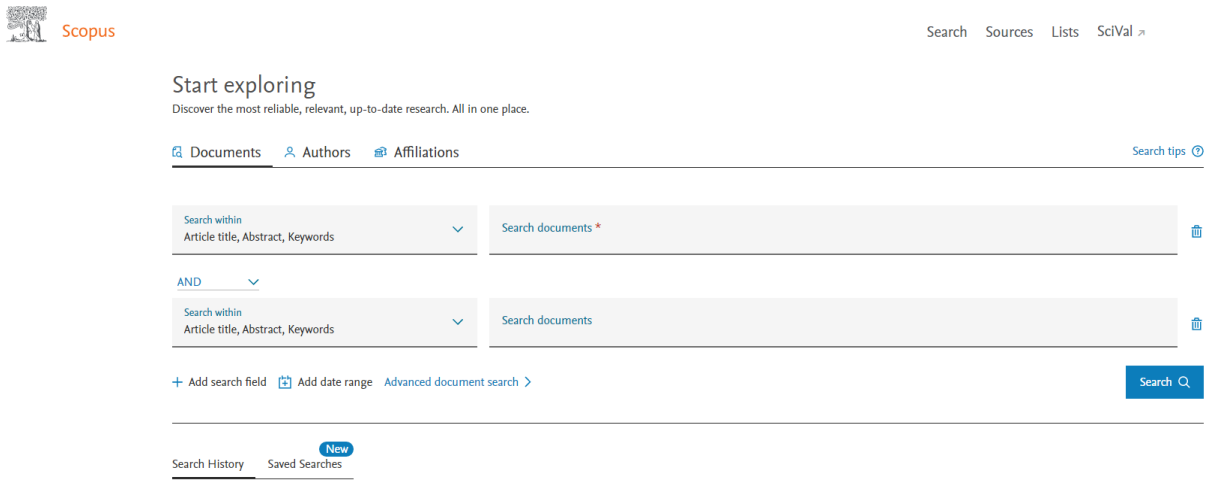
- από το κείμενο της εργασίας (τίτλος, περίληψη, λέξεις κλειδιά)
- από τους συγγραφείς και
- από τις λοιπές πληροφορίες που συνοδεύουν το άρθρο:
    - ίδρυμα (affiliation),

- ISSN (International Standard Serial Number - Διεθνής Μοναδικός Αριθμός Σειρών – που χρησιμοποιείται για την καταγραφή των περιοδικών εκδόσεων και είναι μοναδικό για κάποιο περιοδικό),
- DOI (Digital Object Identifier - ψηφιακό αναγνωριστικό αντικειμένου και είναι μοναδικό για κάθε έγγραφο) κλπ)

Β) αποκλειστικά σε συγγραφείς είτε με το επίθετο του συγγραφέα είτε με τον μοναδικό του αριθμό ORCID και

Γ) σε ιδρύματα

Υπάρχει η δυνατότητα, για πιο εστιασμένη αναζήτηση, να προστεθούν επίπεδα αναζήτησης (Σχήμα. 1.2)



Σχήμα 1.2: Δυνατότητες πλοήγησης στο Scopus

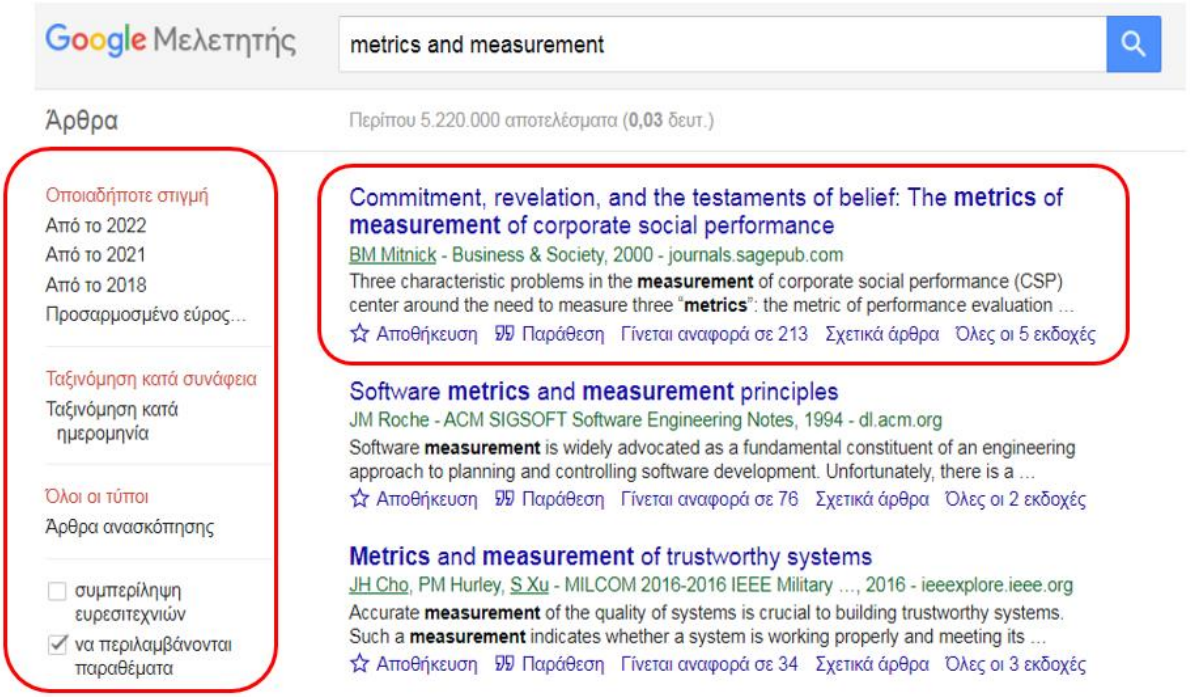
Στα μειονεκτήματα της βιβλιογραφικής βάσης δεδομένων Scopus είναι τα κενά που παρουσιάζει η κάλυψη των κοινωνικών επιστημών πριν το 1996, αν και υπάρχει βελτίωση με πολύ γρήγορους ρυθμούς. Τέλος, όπως στις περισσότερες διεθνείς βιβλιογραφικές βάσεις δεδομένων, τα αγγλόφωνα περιοδικά υπεραντιπροσωπεύονται με αποτέλεσμα να καλύπτονται λιγότερο οι κοινωνικές επιστήμες που είναι συγκριτικά περισσότερο ενταγμένες στο ιδιαίτερο εθνικό πλαίσιο μιας κοινωνίας και τα απορρέοντα ερευνητικά τους αποτελέσματα ενδιαφέρουν περισσότερο ένα εθνικό παρά ένα υπερεθνικό κοινό [2].

1.3 Google Scholar

Το Google Scholar είναι μια ελεύθερης πρόσβασης μηχανή αναζήτησης επιστημονικών κειμένων και δημοσιεύσεων, καθώς και μεταδεδομένων, που έχουν υλοποιηθεί σε μία εκτεταμένη σειρά επιστημονικών και τεχνικών εκδόσεων, σε παγκόσμια κλίμακα. Αφορά δημοσιεύσεις που έχουν γίνει αποκλειστικά με χρήση της αγγλικής γλώσσας.[2] Η beta έκδοσή του κυκλοφόρησε τον Νοέμβριο του 2004 και περιλαμβάνει τα περισσότερα διαδικτυακά περιοδικά και εργασίες συνεδρίων, ακαδημαϊκά βιβλία, διατριβές, τεχνικές αναφορές και λοιπή επιστημονική βιβλιογραφία, συμπεριλαμβανομένων ακόμα και των διπλωμάτων ευρεσιτεχνίας.[3] Παρέχει, μέσω δεικτών (metrics) έναν εύκολο τρόπο στους

[2] https://el.wikipedia.org/wiki/Google_Scholar
[3] https://scholar.google.com/intl/us/scholar/help.html#coverage

συγγραφείς να μετρήσουν γρήγορα την ορατότητα (visibility) και την επιρροή (influence) των πρόσφατων άρθρων σε επιστημονικές δημοσιεύσεις. Σημαντικοί δείκτες της είναι οι h-index για τις δημοσιεύσεις καιη διακύμανση h-median πενταετίας. Ο Δείκτης h μιας δημοσίευσης (δημιούργημα του J.E. Hirsch, στο άρθρο του με τίτλο "Ενας δείκτης για την ποσοτικοποίηση της επιστημονικής ερευνητικής παραγωγής ενός ατόμου", το 2005) είναι ο αριθμός των δημοσιεύσεων (h), που αποδίδονται στην υπό ανάλυση μονάδα, κατά τη διάρκεια του χρονικού διαστήματος που αναλύθηκαν, που έχει τουλάχιστον h αναφορές [4].



Σχήμα 1.3: Εμφάνιση αποτελεσμάτων, με αναζήτηση λέξεων – κλειδιών, στο Google Scholar

Στο Σχήμα 1.3 εμφανίζονται αποτελέσματα σε αναζήτηση τυχαίου συνδυασμού λέξεων – κλειδιών. Στην αριστερή οριοθετημένη περιοχή εμφανίζονται διάφορα συνήθη φίλτρα αναζήτησης (κάποια από αυτά έχουν πλήρη λειτουργικότητα μόνον στην αγγλική έκδοση). Στα στοιχεία ενός άρθρου, αποτέλεσμα της αναζήτησης (δεξιά οριοθετημένη περιοχή), παρέχονται στοιχεία που αντλούνται κατ' ευθείαν από το άρθρο, όπως ο τίτλος του άρθρου, οι συγγραφείς, με την σειρά που αναφέρονται στο άρθρο, το έτος δημοσίευσης και το περιοδικό ή ο εκδοτικός οίκος της δημοσίευσης. Αν ζητηθεί στα φίλτρα, παρέχεται σύντομο παράθεμα (όπως στο παράδειγμα του σχ. 1.2). Τέλος στο τέλος παρέχονται χρήσιμες πληροφορίες, όπως:

- ο τρόπος βιβλιογραφικής παράθεσης (citation), εμφανίζεται αναδυόμενο παράθυρο (pop up) επιλογής βιβλιογράφησης,
- ο αριθμός άλλων τεκμηρίων που αναφέρονται στο συγκεκριμένο άρθρο, δίνοντας την δυνατότητα στον χρήστη να βρει νεώτερες χρονολογικά σχετικές δημοσιεύσεις που βιβλιογραφούν το συγκεκριμένο άρθρο,
- παρόμοια άρθρα και
- εκδοχές βιβλιογράφησης του άρθρου.

Είναι φανερό από τα παραπάνω ότι από τα σημαντικά χαρακτηριστικά του Google Scholar είναι η βολική και εύκολη αναζήτηση άρθρων, πολλές φορές από ένα μόνο μέρος (πχ. μόνον τίτλο, μόνον συγγραφείς κλπ) καθώς και η εξερεύνηση σχετικών έργων, παραπομπών και συγγραφέων.

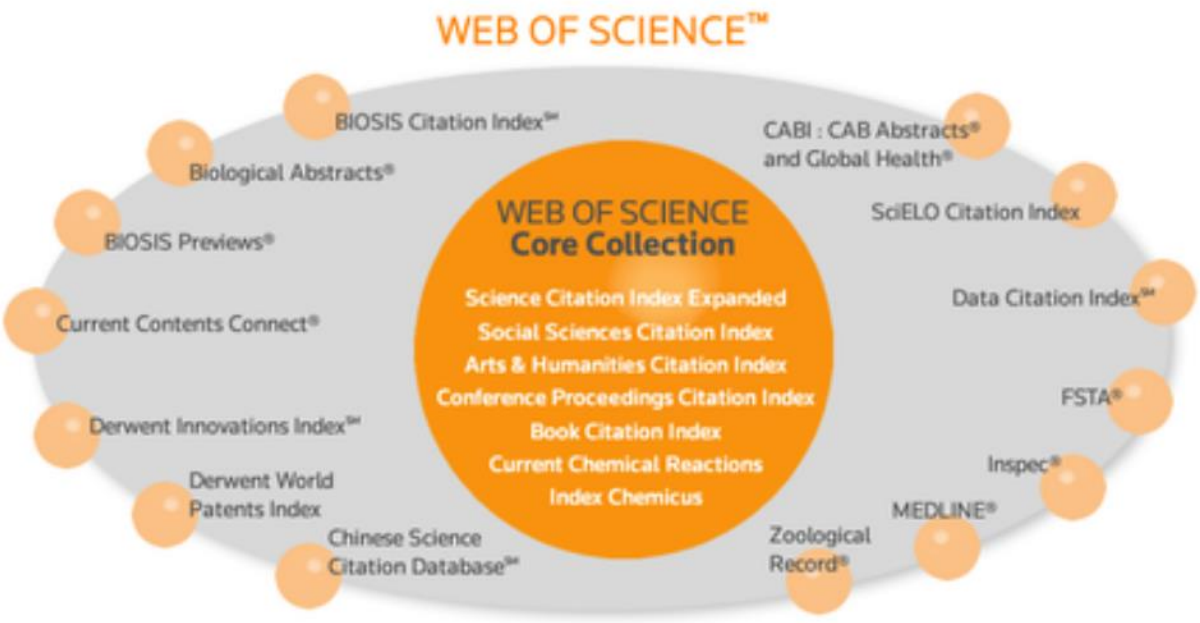
Η βασική κριτική που ασκείται στην αποτελεσματικότητα της μηχανής αναζήτησης Google Scholar είναι η εμφάνιση τεκμηρίων, που δεν πέρασαν απαραίτητα μέσα από διαδικασία κρίσης (peer review). Επίσης επισημαίνεται η ανάκτηση πολλαπλών εγγραφών για την ίδια δημοσίευση (ανακτά πολλές φορές ως ξεχωριστές δημοσιεύσεις το ίδιο δημοσιευμένο άρθρο, ένα ανάτυπο που φιλοξενείται π.χ. στο δικτυακό τόπο του συγγραφέα, την περίληψή του σε μια βάση δεδομένων κλπ.

Τέλος όπως αναφέρουν οι Ζωντανός & Κατρανίδης [2]:

> «Σύμφωνα με μια πρόσφατη έρευνα (Yang & Meho, 2006) στον κλάδο της πληροφορικής το 51,8% των αναφορών του έργου ενός επιστήμονα στη Google Scholar, προέρχεται από μη αξιολογημένα από ομότιμους τεκμήρια. Η Google Scholar είναι χρήσιμη για την ανακάλυψη αναφορών του έργου ενός νέου επιστήμονα ή για την ανάκτηση άλλου τύπου εργασιών, εκτός από άρθρα, ειδικότερα μάλιστα για τις επιστήμες στις οποίες τέτοιου είδους τεκμήρια παίζουν σημαντικό ρόλο στη δημοσίευση των ερευνητικών αποτελεσμάτων. Από την άλλη μεριά όμως δεν συνιστάται για την ανάλυση αναφορών μεγάλου αριθμού δημοσιεύσεων και θεωρείται γενικώς αναξιόπιστη και απρόβλεπτη στα αποτελέσματα που επιστρέφει». (Norris & Oppenheim, 2007).

1.4 Web of Science

Είναι μια ευρέως γνωστή υπηρεσία εντοπισμού βιβλιογραφικών αναφορών και αρθρογραφίας, που αποτελεί μέρος της πλατφόρμας Web of Knowledge της Thomson Reuters, η οποία καλύπτει περισσότερες 12 000 περιοδικά και 160 000 πρακτικά συνεδρίων στην επιστήμη, στις κοινωνικές και ανθρωπιστικές επιστήμες και στις τέχνες (https://en.wikipedia.org/wiki/Web_of_Science).



Σχήμα 1.4: Ο πυρήνας των βάσεων δεδομένων (εσωτερικός κύκλος) και τοπικές βάσεις δεδομένων (εξωτερική έλλειψη) που συνεισφέρουν στο Web of Science.

Μέχρι το 2004 οι μοναδικές πηγές άντλησης δεδομένων αναφορών ήταν οι βάσεις δεδομένων Science Citation Index (SCI), Social Sciences Citation Index (SSCI) και Art & Humanities Citation Index (A&HCI), που πλέον είναι προσβάσιμες στον παγκόσμιο ιστό μέσω της πλατφόρμας Web of Science

που συνεχώς διευρύνεται. Χρονολογικά η κάλυψη ξεκινάει από το 1900 για το SCI, το 1956 για το SSCI και το 1975 για το A&HCI [2].

Σημαντικό πλεονέκτημα του Web of Science θεωρείται η αξιοπιστία του, αποτέλεσμα της αυστηρής αξιολόγησης των εκδόσεων και κυρίως των περιοδικών που εισάγονται στο σύστημα βάσει συγκεκριμένων κριτηρίων, μεταξύ των οποίων και η επιστημονική τους απήχηση. Στα μειονεκτήματα αναφέρονται κυρίως η ανισομερής κάλυψη των επιστημονικών δημοσιεύσεων, τόσο γεωγραφικά καθώς η συντριπτική πλειοψηφία του υλικού προέρχεται από αγγλόφωνες χώρες, όσο και θεματικά καθώς μια αναζήτηση αναφορών στη Web of Science έχει τις δυσκολίες της, αφού δεν περιλαμβάνει στη βάση δεδομένων όλα τα περιοδικά, αναφορές από βιβλία, διατριβές, διπλώματα ευρεσιτεχνίας και τεχνικές εκθέσεις, ενώ επίσης δεν περιλαμβάνονται ισομερώς όλοι οι κλάδοι, αφού οι θετικές επιστήμες είναι καλύτερα καλυμμένες από άλλες, όπως οι θεωρητικές. Επίσης δεν καλύπτονται ομοιόμορφα όλες οι επιστήμες από χρονολογικής άποψης, τα επιστημονικά περιοδικά που καλύπτουν τις θετικές επιστήμες πάνε πολύ πιο πίσω στο χρόνο από ότι τα περιοδικά που καλύπτουν τις τέχνες, τις ανθρωπιστικές και κοινωνικές επιστήμες. Ορισμένες δε θεματικές περιοχές καλύπτονται ελάχιστα, όπως η οικονομία και η εκπαίδευση [6].

1.5 ORCID

Το ORCID (Open Researcher and Contributor ID) είναι αλφαριθμητικός κωδικός 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 (https://el.wikipedia.org/wiki/ORCID). Είναι μια προσπάθεια σε παγκόσμιο επίπεδο για:

- τη δημιουργία και τη συντήρηση ενός μητρώου μοναδικών αναγνωριστικών των ερευνητών / ερευνητριών
- τη διαφανή και απρόσκοπτη διασύνδεση των ερευνητικών δραστηριοτήτων και των αποτελεσμάτων της έρευνας.

Το ORCID στοχεύει να παράσχει έναν συγκροτημένο κώδικα για τους επιστήμονες, για να αντιμετωπίσει το πρόβλημα ότι η συμβολή ενός συγκεκριμένου συγγραφέα στην επιστημονική επικοινωνία μπορεί να είναι δύσκολο να αναγνωριστεί καθώς τα περισσότερα προσωπικά ονόματα δεν είναι μοναδικά και έτσι πολλοί ίδιοι άνθρωποι θα μπορούσαν να συνεισφέρουν στο ίδιο επιστημονικό πεδίο, ακόμη και από το ίδιο θεσμικό τμήμα. Επιπλέον, τα ονόματα μπορούν να αλλάξουν (όπως με το γάμο σε αρκετές χώρες). Υπάρχουν πολιτιστικές διαφορές στις συμβάσεις ονοματοδοσίας. Τα περιοδικά κάνουν ασυνεπή χρήση συντομογραφιών ονόματος, επιθημάτων ονόματος και μεσαίων αρχικών· και χρησιμοποιούν διαφορετικά συστήματα γραφής. Με το ORCID συνεργάζονται οι περισσότεροι εκδότες γιατί λύνει το πρόβλημα των πολλαπλών διαφορετικών ID συγκεντρώνοντας τα όλα σε ένα. Μάλιστα πολλοί από αυτούς πλέον θέτουν ως προαπαιτούμενο την ύπαρξη του ORCID iD από τους συγγραφείς για να προχωρήσουν σε δημοσίευση.

Το ORCID υποστηρίζει ότι μπορεί να βελτιώσει:

- την οργανωτική διαχείριση της ερευνητικής δραστηριότητας των ερευνητών,
- την παρακολούθηση και αξιολόγηση της επιστημονικής παραγωγής,
- τη διαλειτουργικότητα με άλλα συστήματα πληροφόρησης, εντός και εκτός ενός ιδρύματος.
- την ακρίβεια των δημόσιων αρχείων ενός ιδρύματος.
- την ανταλλαγή δεδομένων και συμμόρφωση με τις πολιτικές κατάθεσης των ερευνητικών αποτελεσμάτων και άρθρων.

Έτσι οι ερευνητές:

- Μπορούν να αποκτήσουν ελεύθερα ένα αναγνωριστικό ORCID (ORCID ID), το οποίο τους ξεχωρίζει από άλλους ερευνητές (πχ. με ίδιο όνομα) και τους επιτρέπει να διαχειρίζονται τις εγγραφές τις δημοσιεύσεις τους και να αναζητούν άλλους ερευνητές στο Μητρώο. Με αυτό τον τρόπο παίρνουν τον έλεγχο του τρόπου με τον οποίο το όνομά τους χρησιμοποιείται στην βιβλιογραφία, βελτιώνοντας τόσο τη δική τους αναγνώριση και ανιχνευσιμότητα όσο και των ερευνητικών τους αποτελεσμάτων.

- Μπορούν να συγκεντρώσουν πολλαπλά ID που δίνονται από διαφορετικές υπηρεσίες (πχ. SCOPUS ID, Researcher ID, IDs από διαφορετικούς εκδότες/παρόχους που ο καθένας προσφέρει στην δική του πλατφόρμα) κάτω από ένα μόνο ID, το ORCID ID.
- Συνδέονται εύκολα και αξιόπιστα με την ερευνητική τους δραστηριότητα (δημοσιεύσεις, συνεισφορές, συνέδρια, κτλ.) και τις εργασιακές/ερευνητικές τους σχέσεις.

Όπως γίνεται φανερό, διαφοροποιείται από τις άλλες βάσεις δεδομένων στο ότι συγκροτείται από τον ίδιο τον ερευνητή και από την ίδια την ερευνήτρια.

1.6 Σύγκριση των δημοφιλέστερων βάσεων δεδομένων

Στη συνέχεια επιχειρείται μια σύγκριση των δυνατοτήτων των τριών δημοφιλέστερων βάσεων δεδομένων [7]. Να επισημανθεί η σχετικότητα του Πίνακα, γιατί η πηγή είναι αρκετά παλιά (2008, ο έλεγχος των βάσεων δεδομένων από τον ερευνητή έγινε τον Νοέμβριο του 2004) για τέτοιου είδους επισκόπηση. Μόνον συγκριτικά μπορεί να μελετηθεί, πχ, ενδιαφέρον παρουσιάζει η αργή επικαιροποίηση της data base της Google Scholar.

Πίνακας 1: Βασικά χαρακτηριστικά των δημοφιλέστερων βάσεων δεδομένων

Χαρακτηριστικό	**Scopus**	**Web of Science**	**Google Scholar**
Περιεχόμενο			
Αριθμός περιοδικών	12,850 (500 ανοικτής πρόσβασης)	8700	Δεν παρέχονται δεδομένα (θεωρητικά, όλες οι ψηφιακές πηγές)
Γλώσσα	Αγγλικά (και επιπρόσθετα πάνω από 30 άλλες γλώσσες)	Αγγλικά (και επιπρόσθετα άλλες 45 γλώσσες)	Αγγλικά (και επιπρόσθετα κάθε γλώσσα, θεωρητικά)
Επιστημονικό πεδίο	Φυσικές Ειστήμες, Επιστήμες Υγείας, Επιστήμες Ζωής και Κοινωνικές Επιστήμες	Επιστήμη, Τεχνολογία, Κοινωνικές και Ανθρωπιστικές Επιστήμες, Τέχνες	Βιολογία, Επιστήμες Ζωής και Περιβαλλοντικές Επιστήμες, Επιχειρήσεις και Διοίκηση Επιχειρήσεων, Οικονομικά και Χρηματο-οικονομικά, Χημεία και Επιστήμη Υλικών, Πολυτεχνικές Επιστήμες, Φαρμακολογία, Κτηνιατρική, Κοινωνικές και Ανθρωπιστικές Επιστήμες, Τέχνες
Χρονολογία κάλυψης	1966-σήμερα	1900-σήμερα	Θεωρητικά όλα τα παρεχόμενα ψηφιακά και ψηφιοποιημένα έγγραφα
βάσεις δεδομένων που καλύπτονται	100% Medline, Embase, Compendex, World textile index, Fluidex, Geobase, Biobase	Science citation index expanded, social sciences citation index, arts and humanities citation index, index	PubMed, OCLC First Search

		chemistry, current chemical reactions	
Αριθμός επιτρεπόμενων λέξεων - κλειδιών	30	15	Θεωρητικά δεν υπάρχει όριο
Ιχνηλάτηση σε:			
Περιεχόμενα	(+)	(+)	(+)
Συγγραφείς	(+)	(+)	(+)
Αναφορές	(+)	(+)	(+)
Πατέντες	(+)	(+)	(−)
Χρήσεις	Διασυνδεδεμένο με πλήρες κείμενο των άρθρων και άλλων πηγών βιβλιοθηκών	Διασυνδεδεμένο με πλήρες κείμενο άρθρων και με σχετικά άρθρα	Διασυνδεδεμένο με πλήρες κείμενο άρθρων, με περιοδικά με σχετικά άρθρα, με βιβλιοθήκες
Αναθεώρηση κάθε	1–2 φορές την εβδομάδα	εβδομαδιαία	Κατά μέσον όρο μηνιαία
Ανάλυση Αναφορών	Όλα τα άρθρα που αναφέρονται σε ένα θέμα ή σε έναν ερευνητή	Όλα τα άρθρα που αναφέρονται σε ένα θέμα ή σε έναν ερευνητή	Δίπλα σε κάθε άρθρο υπάρχει σύνδεσμος με παραπομπή στην ανάλυση αναφορών

Κεφάλαιο 2ο: Γλώσσες και τεχνολογίες

2.1 Εισαγωγή

Ο προγραμματισμός Ιστού ορίζεται ως η γραφή και η κωδικοποίηση που εμπλέκεται στην ανάπτυξη Ιστού. Ο προγραμματισμός Ιστού διαφέρει από τον απλό προγραμματισμό, καθώς απαιτεί διεπιστημονική γνώση της περιοχής εφαρμογής, δέσμη ενεργειών πελάτη και διακομιστή και τεχνολογία βάσεων δεδομένων. Ο προγραμματισμός Ιστού διαχωρίζεται έτσι σε προγραμματισμό στην πλευρά του διακομιστή (backend) και στην πλευρά του πελάτη (frontend). Στο κεφάλαιο αυτό θα περιγράφουν και αναλυθούν οι τεχνολογίες που χρησιμοποιήθηκαν για την εκπόνηση της παρούσας πτυχιακής εργασίας.

2.1.1 Προγραμματισμός Backend και Frontend

Ο προγραμματισμός στην πλευρά του διακομιστή, γνωστός και ως προγραμματισμός back-end, περιλαμβάνει όλα τα είδη λειτουργιών που εκτελούνται στον διακομιστή. Όσον αφορά τις λειτουργίες, ο προγραμματισμός από την πλευρά του διακομιστή φροντίζει για την επεξεργασία της εισαγωγής δεδομένων του χρήστη, την εμφάνιση σελίδων, δόμηση διαδικτυακών εφαρμογών και αλληλεπίδραση με αρχεία. Ενώ ο πελάτης βλέπει μόνο μια έξοδο HTML από τον διακομιστή, ο πηγαίος κώδικας και η λογική του παραμένει κρυφός. Παραδείγματα γλωσσών που χρησιμοποιούνται στον προγραμματισμό από την πλευρά του διακομιστή είναι η PHP, η Python, η ASP.NET, η Java κα.

Ο προγραμματισμός στην πλευρά του πελάτη ή και αλλιώς προγραμματισμός front-end περιλαμβάνει τις λειτουργίες που εκτελούνται στον πελάτη (φυλλομετρητή). Περιλαμβάνει μια σειρά γλωσσών που καταλαβαίνει το πρόγραμμα περιήγησης. Από πλευράς λειτουργικότητας, οι γλώσσες αυτές χρησιμοποιούνται για την δημιουργία και σχεδίαση δυναμικών ιστοσελίδων, αποστολή αιτημάτων και ανάκτηση δεδομένων από τον διακομιστή. Οι κυρίαρχες γλώσσες που χρησιμοποιούνται είναι HTML, CSS και JavaScript.

2.2 Τεχνολογίες frontend

Οι ακόλουθες τεχνολογίες χρησιμοποιήθηκαν στο έργο για τον προγραμματισμό του front-end ή του client-side.

- HTML5 για την δομή του περιεχομένου
- CCS3 για το στιλ των σελίδων
    - CSS Frameworks
- JavaScript για την διαδραστική εμφάνιση
    - Javascript Frameworks

Κάθε μία από αυτές τις τεχνολογίες θα περιγραφεί στις ακόλουθες ενότητες.

2.2.1 HTML

Είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Η HTML5 είναι η πιο πρόσφατη εκδοχή της html και έχει σχεδιαστεί έτσι ώστε να παρέχει σχεδόν ό,τι χρειάζεται μια ιστοσελίδα χωρίς την χρήση κάποιου πρόσθετου λογισμικού (plugin).

Η HTML ακολουθεί την ίδια σύνταξη με την XML αλλά έχει πιο «χαλαρούς» κανόνες. Γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες λειτουργούν ανά ζεύγη με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης. Συχνά ένα στοιχείο HTML περιέχει με την σειρά του άλλα στοιχεία HTML δημιουργώντας έτσι ένα «δέντρο» στοιχείων που ονομάζεται μοντέλο αντικειμένων εγγράφων (DOM) όπως φαίνεται στο παρακάτω παράδειγμα.

Κώδικας 1: Σύνταξη html αρχείου

```
<!DOCTYPE html>
<html>
  <head>
    <title>My test page</title>
     <link> rel=”stylesheet” href=”styles/index.css”>
  </head>
  <body>
    <p> Στοιχείο παραγράφου. </p>
  </body>
</html>
```

- *<!DOCTYPE html>:* Είναι δήλωση προς τον φυλλομετρητή για το τύπου αρχείο να περιμένει. Το συγκεκριμένο δηλώνει το HTML5.
- Το *html* είναι το ριζικό στοιχείο και εμπεριέχει όλο το περιεχόμενο σε ολόκληρη την σελίδα.
- Μέσα στο head στοιχείο δηλώνονται οι πληροφορίες σχετικά με την ιστοσελίδα. Τέτοιο περιεχόμενο είναι η περιγραφή, ο τίτλος, οι λέξεις-κλειδιά της ιστοσελίδας αλλά και CSS περιεχόμενο για την μορφοποίηση της.
- Το *title* στοιχείο ορίζει τον τίτλο της ιστοσελίδας.
- Το *body* στοιχείο εμπεριέχει όλο το περιεχόμενο που θέλουμε να παρουσιάσουμε στους χρήστες.

Η HTML δε περιγράφει τη μορφοποίηση των εγγράφων, αυτό το επιτυγχάνουν καλύτερα τα διαδοχικά φύλλα στυλ (CSS).

2.2.2 CSS

Η CSS είναι επίσης μια γλώσσα που χρησιμοποιείται για την περιγραφή της παρουσίασης ενός εγγράφου γραμμένο σε γλώσσα σήμανσης. Με τη CSS καθορίζουμε τα χρώματα, την διάταξη και την γραμματοσειρά των στοιχείων HTML του εγγράφου. Μας δίνει επίσης την δυνατότητα να τροποποιούμε την εμφάνιση του εγγράφου ανάλογα με το τύπο και το μέγεθος της συσκευής-οθόνης. Η CSS είναι ανεξάρτητο από τη HTML και μπορεί να χρησιμοποιηθεί με οποιαδήποτε γλώσσα σήμανσης που βασίζεται σε XML. Ο χωρισμός των HTML από το CSS διευκολύνει τη συντήρηση των ιστοσελίδων, την κοινή χρήση CSS εγγράφων σε όλες τις σελίδες και την τροποποίηση σελίδων σε συγκεκριμένα περιβάλλοντα. Αυτό ονομάζεται διαχωρισμός της δομής από την παρουσίαση [8]

2.2.2.1 CSS Frameworks

Τα front-end frameworks είναι βιβλιοθήκες που περιέχουν εργαλεία που καθιστούν την ανάπτυξη CSS ευκολότερη. Το bootstrap αποτελεί ένα από αυτά. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης, πίνακες και άλλων στοιχείων του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις JavaScript. Το bootstrap για την διάταξη της σελίδας χρησιμοποιεί το πλέγμα (grid) με 12 στήλες, και περιέχει 5 σημεία διακοπής τα οποία ανάλογα με το πλάτος της οθόνης (extra small, small, medium, large, and extra large) το περιεχόμενο εμφανίζεται μέσα σε έναν αριθμό στηλών που επιλέγουμε, διατηρώντας έτσι την ανταποκρισημότητα (responsiveness) της σελίδας σε διάφορες συσκευές.

2.2.3 JavaScript

Σύμφωνα με την τεκμηρίωση της JavaScript, η JavaScript είναι μια αντικειμενοστραφής γλώσσα σεναρίου. Είναι μια μικρή και ελαφριά γλώσσα που εκτελείται μέσα σε ένα περιβάλλον υποδοχής (host), όπως έναν φυλλομετρητή [9].

Χαρακτηρίζεται από ασθενή ορισμό τύπων και αυτό σημαίνει ότι οι μεταβλητές μπορούν να μετατραπούν εύκολα από έναν τύπο δεδομένων σε άλλο κατά την εκτέλεση.

Αποτελείται από μια τυπική βιβλιοθήκη αντικειμένων, όπως Array, Date και Math, και ένα βασικό σύνολο στοιχείων γλώσσας όπως χειριστές, δομές ελέγχου και δηλώσεις. Η βασική JavaScript μπορεί να επεκταθεί για διάφορους σκοπούς συμπληρώνοντάς το με πρόσθετα αντικείμενα. Για παράδειγμα:

Η JavaScript από την πλευρά του πελάτη (client-side) επεκτείνει τη βασική γλώσσα παρέχοντας αντικείμενα που χρειάζονται για να ελεγχθεί το πρόγραμμα περιήγησης και το DOM. Έτσι ο φυλλομετρητής μπορεί να απαντήσει πιο γρήγορα σε συμβάντα χρηστών σε σχέση με μια αίτηση που υποβάλλεται σ ’έναν απομακρυσμένο διακομιστή, κάτι που βελτιώνει την εμπειρία του χρήστη.

Από την πλευρά του διακομιστή οι επεκτάσεις της JavaScript δίνουν την δυνατότητα να επικοινωνήσει μια εφαρμογή με την βάση δεδομένων ή ακόμα και να διαχειριστεί κάποια αρχεία του διακομιστή.

2.2.3.1 JavaScript Frameworks και βιβλιοθήκες

Για να είναι η εμπειρία του χρήστη σε μια δικτυακή τοποθεσία πιο διαδραστική, χρησιμοποιείται η AJAX. Με την AJAX, οι εφαρμογές Ιστού μπορούν να στέλνουν και να ανακτούν δεδομένα από έναν διακομιστή ασύγχρονα (στο παρασκήνιο) χωρίς να παρεμβαίνουν στην εμφάνιση και τη συμπεριφορά της υπάρχουσας σελίδας. Επειδή η ανάπτυξη τέτοιων διεπαφών μπορεί να έχει αρκετές απαιτήσεις για JavaScript, δημιουργήθηκαν αρκετές βιβλιοθήκες που μειώνουν τον όγκο του κώδικα που απαιτείται για την εκτέλεση τυπικών εργασιών AJAX. Μια από αυτές είναι η jQuery.

Η σύνταξη της jQuery έχει σχεδιαστεί για να απλοποιεί τη διέλευση και τον χειρισμό του δέντρου HTML DOM. Παρέχει δηλαδή ένα σύστημα που διαχειρίζεται τα events, διαχωρίζοντας και έτσι τον κώδικα της Javascript από αυτόν της HTML. Παράλληλα πάνω σε αυτήν μπορούν να αναπτυχθούν διάφορα plug-ins δημιουργώντας έτσι ισχυρές δυναμικές ιστοσελίδες με animations, εφέ, widgets κτλ [10].

Η βιβλιοθήκη chartJS που χρησιμοποιήθηκε για την ανάπτυξη αυτής της εφαρμογής αποτελεί ένα τέτοιο παράδειγμα. Είναι μια open source βιβλιοθήκη που μας δίνει την δυνατότητα να σχεδιάσουμε με ευκολία διαδραστικά γραφήματα βάση των δεδομένων που τα παρέχουμε [11].

Τέλος, ορισμένες βιβλιοθήκες JavaScript, όπως η Angular, η React, η Django κα, ταξινομούνται ως frameworks, καθώς παρουσιάζουν δυνατότητες πλήρους στοίβας (frontend-backend) και ιδιότητες που δεν βρίσκονται στις γενικές βιβλιοθήκες JavaScript.

2.3 Τεχνολογίες backend

Οι παρακάτω είναι κάποιες από τις τεχνολογίες που χρησιμοποιήθηκαν για τον προγραμματισμό της εφαρμογής από την πλευρά του διακομιστή. Οι τεχνολογίες αυτές έχουν να κάνουν με τη γλώσσα προγραμματισμού, το web framework, την βάση δεδομένων και τον διακομιστή ιστού και είναι αντίστοιχα:

- Python (έκδοση 3.8)
- Django Framework (έκδοση 3.2)
- MySQL
- Apache

Αναλυτικά θα περιγραφούν στις επόμενες ενότητες.

2.3.1 Python

Η python είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού γνωστή για τον «καθαρό» κώδικά της. Είναι μια διερμηνευόμενη γλώσσα προγραμματισμού που αναπτύχθηκε από τον Ολλανδό Guido van Rossum το 1990 και υποστηρίζει τόσο τον αντικειμενοστραφή όσο και τον διαδικαστικό προγραμματισμό. Κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της. Η μεγάλη πληθώρα βιβλιοθηκών καθώς και η μεγάλη κοινότητα προγραμματιστών που χρησιμοποιούν την γλώσσα αυτή, διευκολύνουν ιδιαίτερα στην ανάπτυξη σωστού και επαναχρησιμοποιήσιμου κώδικα. Χρησιμοποιείται σε διάφορα περιβάλλοντα προγραμματισμού για το web, όπως το Django και το Pyramid.

Το Django αποτελεί ένα μοντέρνο web framework βασισμένο στην python. Αναλυτικά για αυτό θα μιλήσουμε στο Κεφάλαιο 3.

2.3.2 MySQL

Μια σχεσιακή βάση δεδομένων οργανώνει δεδομένα σε έναν ή περισσότερους πίνακες δεδομένων στους οποίους οι τύποι δεδομένων μπορεί να σχετίζονται μεταξύ τους. Το βασικό πλεονέκτημα μιας σχεσιακής βάσης είναι η δυνατότητα απόκτησης πληροφορίας με σημασία με μια απλή σύνδεση πινάκων. Η σύνδεση πινάκων βοηθά στην καλύτερη κατανόηση των δεδομένων και στη καλύτερη διαχείρισή τους.

Η SQL είναι μια γλώσσα που ειδικεύεται σε έναν συγκεκριμένο τομέα εφαρμογών όπως την HTML, και βοηθάει στην αλληλεπίδραση με την σχεσιακή βάση δεδομένων. Επιπλέον, τα πλεονεκτήματα των σχεσιακών βάσεων δεδομένων σε σχέση με άλλες αρχιτεκτονικές είναι η ευελιξία, η συντηρησιμότητα, η ασφάλεια, η ευκολία ανάκτησης ενός backup μετά από καταστροφή του συστήματος.

Συνήθως στις δικτυακές τοποθεσίες οι σελίδες μοιράζονται κοινά στοιχεία διεπαφής αλλά κάθε σελίδα έχει διαφορετικό περιεχόμενο. Το περιεχόμενο αυτό «εισάγεται» από μια βάση δεδομένων συνήθως μετά από ερωτήματα GET ή POST. Με τον τρόπο αυτό επιτυγχάνεται και η δυναμικότητα της σελίδας [12].

2.3.3 Διακομιστής web: Apache

Ο διακομιστής είναι υπεύθυνος για να δώσει απαντήσεις σε όλες τις αιτήσεις πελατών. Ανεξάρτητα απ' το πόσο στατική ή πόσο απλή είναι η δικτυακή τοποθεσία, πρέπει να υπάρχει ένας διακομιστής web που ρυθμίζεται έτσι ώστε να δώσει απαντήσεις σε αιτήσεις για τον συγκεκριμένο τομέα.

Ένας διακομιστής web πέρα από το να απαντά μόνο σε αιτήσεις για αρχεία HTML, εκτελεί μια λίστα διεργασιών. Αυτή περιλαμβάνει τον χειρισμό συνδέσεων HTTP, την απόκριση σε αιτήσεις για στατικούς και δυναμικούς πόρους, τη διαχείριση των δικαιωμάτων και πρόσβασης για ορισμένους πόρους, τη διαχείριση συνδέσεων κα [13].

Το Apache εκτελείται ως daemon στον διακομιστή. Είναι μια διεργασία δηλαδή που δουλεύει στο παρασκήνιο και ενεργοποιείται όταν δεχθεί ένα συμβάν όπως ένα αίτημα HTTP.

Κάποια από τα ερωτήματα HTTP είναι [14]:

- GET - Παρέχει πρόσβαση μόνο για ανάγνωση σε έναν πόρο.
- Head - Το ίδιο με το GET, αλλά μεταφέρετε μόνο η γραμμή κατάστασης και η ενότητα κεφαλίδας
- PUT - Αντικατάσταση ενός πόρου
- DELETE - Διαγραφή ενός πόρου
- POST - Ανανέωση ή δημιουργία ενός πόρου
- OPTIONS - Χρησιμοποιείται για τη λήψη των υποστηριζόμενων λειτουργιών σε έναν πόρο.

2.4 Εργαλεία ανάπτυξης

2.4.1 Visual Studio Code

Ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) περιγράφεται ως ένα πρόγραμμα λογισμικού που περιέχει μια σειρά εργαλείων όπως πρόγραμμα επεξεργασίας κώδικα, μεταγλωττιστή, εντοπισμό σφαλμάτων και πολλά άλλα χρήσιμα εργαλεία που απαιτούνται για την αποτελεσματική εγγραφή και δοκιμή, αναμόρφωση κώδικα και δημιουργία διαγραμμάτων για μια εφαρμογή υπό ανάπτυξη . Παραδείγματα IDE είναι τα Emacs, Eclipse, Idea, JBuilder, Visual Studio/(Code), Netbeans κ.λπ.

Το Visual Studio Code είναι ένα λογισμικό ανοιχτού κώδικα και αναπτύχθηκε από την Microsoft. Είναι ένα πρόγραμμα επεξεργασίας πηγαίου κώδικα που μπορεί να χρησιμοποιηθεί με μια ποικιλία γλωσσών προγραμματισμού. Επιτρέπει στους χρήστες να ανοίγουν έναν ή περισσότερους καταλόγους, οι οποίοι στη συνέχεια μπορούν να αποθηκευτούν σε χώρους εργασίας για μελλοντική επαναχρησιμοποίηση.

Μια πληθώρα επεκτάσεων που είναι διαθέσιμες μέσω ενός κεντρικού αποθετηρίου, και η ύπαρξη ενσωματωμένης γραμμής εντολών διευκολύνουν υπερβολικά στην ταχεία ανάπτυξη του κώδικα.

Διαθέτει μια ειδική καρτέλα στη γραμμή μενού απ' όπου μπορούμε να αποκτήσουμε πρόσβαση στις ρυθμίσεις του συστήματος ελέγχου έκδοσης και να προβάλλονται οι αλλαγές που έγιναν στην τρέχον εφαρμογή.

2.4.2 Σύστημα Ελέγχου Έκδοσης

Στον σημερινό κόσμο του προγραμματισμού, η επιλογή ενός συστήματος ελέγχου έκδοσης έχει πολλά πλεονεκτήματα που κυμαίνονται από την επαναφορά και την παρακολούθηση αλλαγών που έγιναν σε ένα αρχείο και τη σύγκριση αλλαγών με την πάροδο του χρόνου. Χρησιμοποιήθηκε το Git για αυτήν την πτυχιακή.

Το Git είναι ένα open-source λογισμικό που αντιμετωπίζει τα δεδομένα του περισσότερο σαν μια ροή στιγμιότυπων. Οι στόχοι του περιλαμβάνουν την ταχύτητα, την ακεραιότητα των δεδομένων και την υποστήριξη κατανεμημένων, μη γραμμικών ροών εργασίας.

Το GitHub είναι ίσως το πιο γνωστό εργαλείο διαχείρισης και φιλοξενίας κώδικα με δυνατότητα ελέγχου έκδοσης της εφαρμογής και αποθήκευση του κώδικα στο cloud. Η δημοτικότητα του αυτή φαίνεται και από το γεγονός ότι πολυάριθμες εφαρμογές στο web development όπως host providers, servers, cloud databases έχουν ενσωματώσει το Github στις υπηρεσίες που προσφέρουν.

Το GitHub επιλέχθηκε για την αποθήκευση και διατήρηση του κώδικα λόγω της ευκολίας στη χρήση και της ασφάλειας που προσφέρει καθώς δεν υπάρχει κίνδυνος απώλειας του κώδικα αφού αυτός αποθηκεύεται στο cloud. Ακόμη το GitHub προσφέρει υποστήριξη σε περιβάλλοντα συγγραφής κώδικα όπως στην παρούσα περίπτωση είναι το VS Code με το οποίο το Github μέσω του αντίστοιχου plugin στο VS Code κάνει πιο εύκολη την διαχείριση του ελέγχου του κώδικα

Ένα από τα ποιο σημαντικά κομμάτια που μας προσφέρει, είναι η δυνατότητα να συνεργαζόμαστε με άλλους προγραμματιστές. Μπορούμε να δουλεύουμε ταυτόχρονα πάνω στο ίδιο project σε διαφορετικά branches ανεξάρτητα ο ένας από τον άλλον. Φυσικά όταν συμβαίνει αυτό υπάρχουν και κάποια βήματα που πρέπει να ακολουθούμε για την ομαλή συνένωση του κώδικα και για την αποφυγή επιπλοκών.

Η βασική ροή εργασίας Git γίνεται κάπως έτσι:

- Τροποποίηση των αρχείων στο δέντρο εργασίας.
- Εκτέλεση μόνο εκείνων των αλλαγών που θέλουμε να πραγματοποιηθούν στο επόμενο commit

Το commit, παίρνει τα αρχεία ως έχουν, και αποθηκεύει αυτό το στιγμιότυπο μόνιμα στον κατάλογό Git [15].

(Το αποθετήριο του κώδικα της εφαρμογής βρίσκεται στο: https://github.com/BscThesis/2021_RPDC)

Κεφάλαιο 3ο: Django Framework

Το Django Framework είναι web framework ανοιχτού κώδικα βασισμένο στη γλώσσα προγραμματισμού python και ακολουθεί το «MVC» μοντέλο αρχιτεκτονικής λογισμικού. Ενθαρρύνει την γρήγορη και ασφαλή ανάπτυξη εφαρμογών ιστού καθώς παρέχει ένα πλήθος βιβλιοθηκών για πρόσβαση στην βάση δεδομένων, αυθεντικοποιήση του χρήστη αλλά και ασφάλεια από συνηθισμένες επιθέσεις όπως XSS, SQL Injection και clickjacking. Αποτελεί ένα full stack framework εφόσον διαχειρίζεται και το backend και το frontend μέρος της εφαρμογής ιστού. Με τις προκαθορισμένες ρυθμίσεις, το Django κάνει render τα δεδομένα από την πλευρά του server, αλλά με την χρήση της βιβλιοθήκης django rest framework API μπορεί να κάνει render τα δεδομένα από την πλευρά του client [16].

3.1 Γενικές Φιλοσοφίες Σχεδιασμού της Django

Όπως κάθε ουσιαστικό σχέδιο, οι προγραμματιστές του Django ακολούθησαν κάποιες θεμελιώδης φιλοσοφίες για την ανάπτυξή και σχεδίασή του. Οι βασικές αυτές ιδέες είναι [17]:

- Χαλαρή σύζευξη
  Το πλαίσιο του Django κατασκευάστηκε με τέτοιο τρόπο ώστε κάθε στοιχείο του framework να είναι ανεξάρτητο το ένα από το άλλο. Τα επίπεδα του πλαισίου του Django δεν γνωρίζουν το ένα το άλλο. Για παράδειγμα, το σύστημα των templates δεν γνωρίζει τίποτα για τα αιτήματα που γίνονται στον διακομιστή, το επίπεδο βάσης δεδομένων δεν γνωρίζει τίποτα για την εμφάνιση δεδομένων και το σύστημα προβολής δεν ενδιαφέρεται ποιο template σύστημα χρησιμοποιεί ένας προγραμματιστής. Κατά συνέπεια, το πλαίσιο Django δημιουργεί διατηρήσιμες και ευέλικτες διαδικτυακές εφαρμογές.

- Λιγότερος κώδικας
  Όσο περισσότερος κώδικας τόσο περισσότερα σφάλματα. Το πλαίσιο Django διασφαλίζει ότι ελαχιστοποιείται η ποσότητα του κώδικα που χρησιμοποιείται στις εφαρμογές. Αυτό με τη σειρά του επιταχύνει και την ανάπτυξη της εφαρμογής.

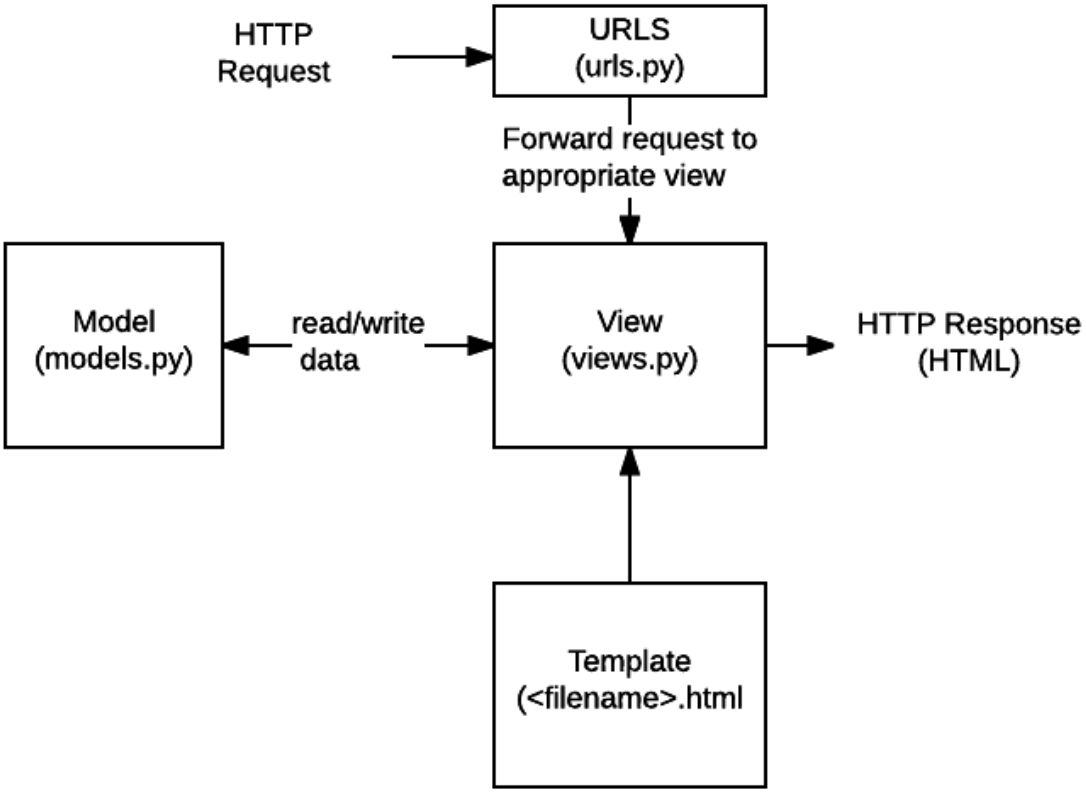
- Γρήγορη ανάπτυξη
  Ο κύριος λόγος πίσω από την εξέλιξη των framework είναι να γίνουν πιο γρήγορες οι κουραστικές και επαναλαμβανόμενες πτυχές της ανάπτυξης ιστού.

- DRY (Don’t Repeat Yourself)
  Το Django ακολουθεί την αρχή DRY που εξαλείφει όλες τις μορφές πλεονασμού. Στο Django, η λειτουργικότητα κάθε εφαρμογής βρίσκεται σε ένα μέρος. Αυτό όχι μόνο μειώνει την ποσότητα του κώδικα, αλλά συμβάλλει επίσης στην απλότητα ολόκληρης της εφαρμογής.

3.2 Αρχιτεκτονική σχεδιασμού MVC

Ο κύριος στόχος του σχεδιασμού αρχιτεκτονικής «MVC» είναι να προωθήσει την επαναχρησιμοποίηση κώδικα. Για αυτόν τον λόγο, το MVC διαιρεί την εφαρμογή ιστού σε τρία κύρια στοιχεία, δηλαδή Μοντέλο, Προβολή και Ελεγκτή (Model,View,Controller). Ενώ αυτή είναι η γενική προσέγγιση, διαφορετικά frameworks επιλέγουν διαφορετικό τρόπο εφαρμογής του. Στο Django το στοιχείο του ελεγκτή (controller) διαχειρίζεται από το ίδιο το framework. Έτσι κάθε αίτημα κατευθύνεται στις κλάσεις «view» οι οποίες έχουν τη λογική της εφαρμογής. Τα «μοντέλα» αποτελούν οντότητες που αντικατοπτρίζουν τους πίνακες που βρίσκονται στη βάση δεδομένων. Οι κλάσεις «view» θα χρησιμοποιήσουν τα αντικείμενα των «μοντέλων» ώστε να εκτελέσουν μια CRUD εντολή στα

δεδομένα. Στη συνέχεια οι κλάσεις θα χρησιμοποιήσουν τα «templates» για να αποδώσουν γραφικά τα δεδομένα. Άρα για την ακρίβεια το Django ακολουθεί την αρχιτεκτονική MTV (Model,Template,View).



Σχήμα 3.1: MTV Αρχιτεκτονική του Django Framework

3.3 Models

Τα Django **models** είναι Python αντικείμενα, που περιέχουν τα πεδία και τη συμπεριφορά των δεδομένων που θέλουμε να αποθηκεύσουμε στη βάση δεδομένων. Περιέχουν δηλαδή όλες τις βασικές πληροφορίες που χρειαζόμαστε για να καθορίσουμε τα δεδομένα ενός application. Προσφέρουν μηχανισμούς διαχείρισής, όπως προσθήκη, διαγραφή και τροποποίηση και συνήθως ένα model είναι καθορισμένο σε έναν πίνακα της βάσης. Η Django παρέχει κάποια βασικά μοντέλα. Αυτά είναι της **αυθεντικοποίησης**, του **διαχειριστή**, το **ContentType** που περιγράφει τα εγκατεστημένα μοντέλα, της **συνεδρίας** και το μοντέλο **site** που αποθηκεύει το domain και τα χαρακτηριστικά του ιστοτόπου.

Κώδικας 2: Παράδειγμα ενός πίνακα που ονομάζεται «User» και το μοντέλο που το αναπαριστά ως μια κλάση Python

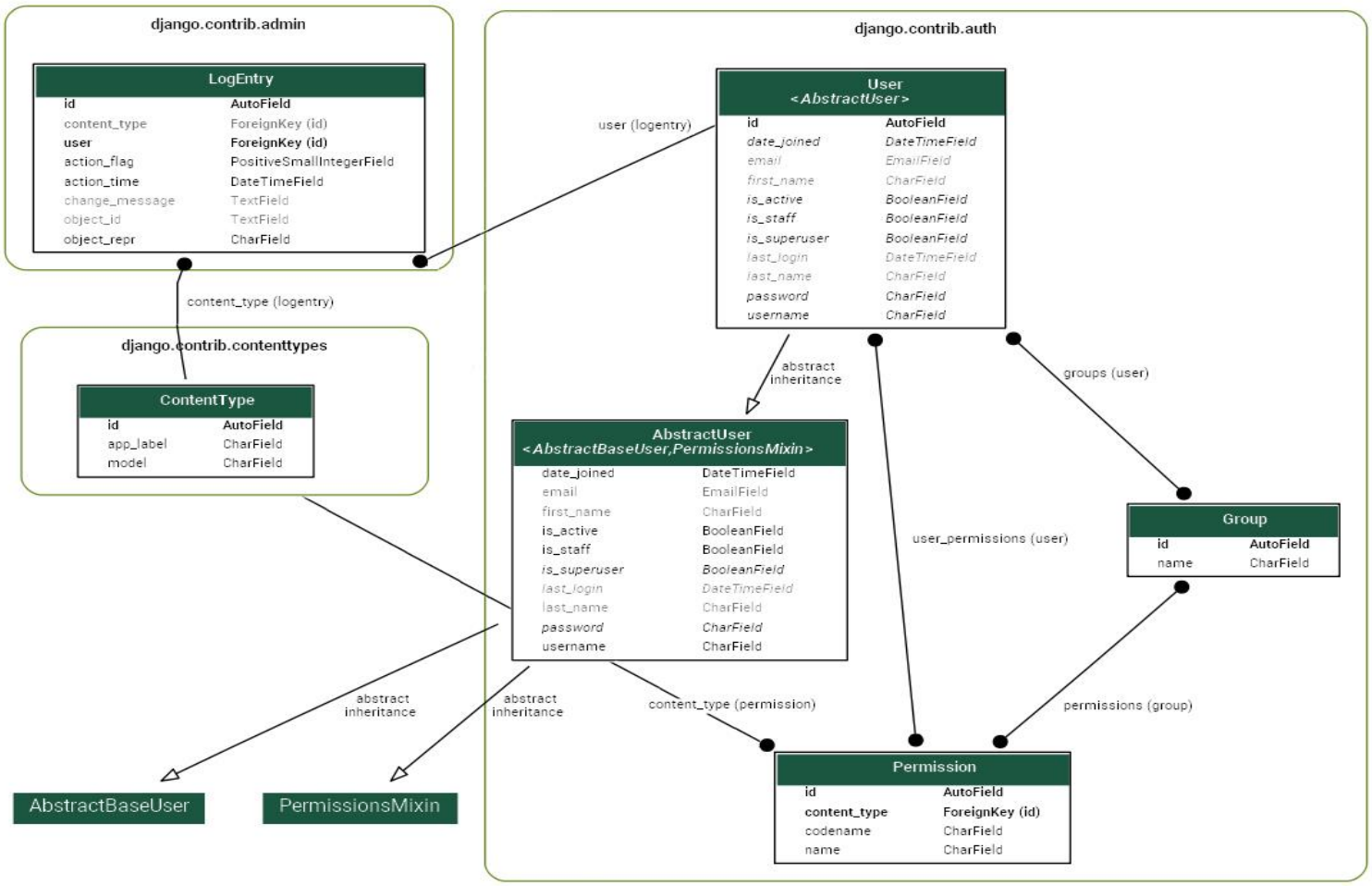
```
from django.db import models
from django.contrib.auth.models import User

class AppUser(models.Model):
    user =
models.OneToOneField(User,on_delete=models.CASCADE,primary_key=Tru
e)
    user_email = models.EmailField(max_length=255, blank=True,
null=True)
    user_firstname = models.CharField(max_length=255, blank=True,
null=True)
```

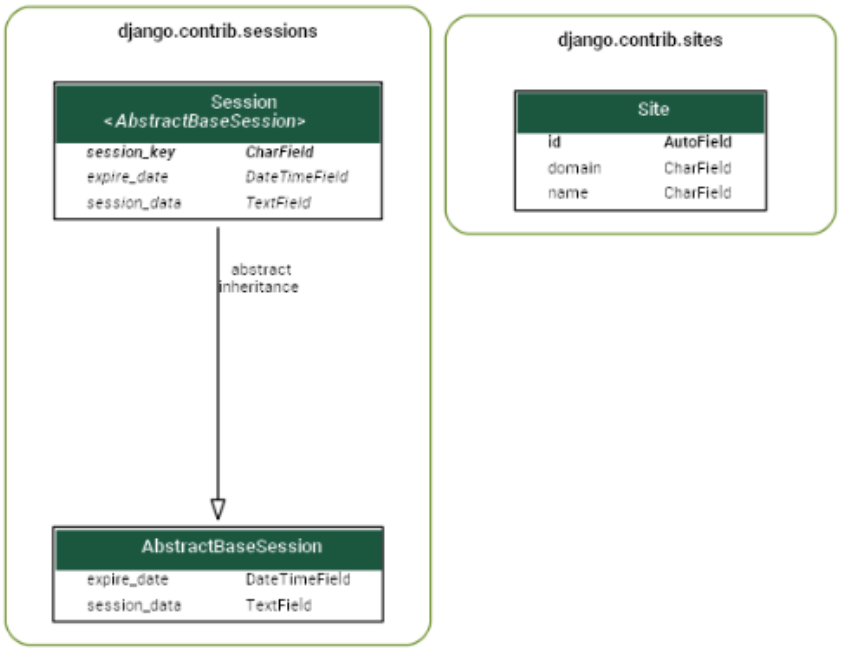
Το προηγούμενο μοντέλο ισοδυναμεί με το παρακάτω SQL script:

```
CREATE TABLE IF NOT EXISTS `RPDC`.`main_appuser` (
  `user_id` INT(11) NOT NULL, PRIMARY KEY,
  `user_email` VARCHAR(255) NULL DEFAULT NULL,
  `user_firstname` VARCHAR(255) NULL DEFAULT NULL,)
```

Για να αναγνωρίσει η Django τα μοντέλα της εφαρμογής θα πρέπει να προσθέσουμε το όνομά της στην λίστα «installed_apps» στο αρχείο ρυθμίσεων settings.py. Έπειτα, μετά από κάθε αλλαγή στο αρχείο των μοντέλων μας, με την εντολή «python manage.py migrate» το Django συγχρονίζει την βάση δεδομένων με την νέα κατάσταση τους.



Σχήμα 3.2: Σχέση μοντέλων (admin,auth,contentTypes)



Σχήμα 3.3: Μοντέλα συνεδρίας και ιστοσελίδων

3.4 Views

Τα **views** είναι συναρτήσεις, οι οποίες διαχειρίζονται αιτήματα (requests) του χρήστη και επιστρέφουν δεδομένα. Ο τύπος των δεδομένων μπορεί να είναι HTML περιεχόμενο, σφάλμα HTTP, εικόνες, ανακατεύθυνση σε άλλη διαδικτυακή τοποθεσία κα. Το αποτέλεσμα των δεδομένων παρουσιάζεται μέσω του template συστήματος [18].

Πιο περιληπτικά, όταν ένας χρήστης ζητά μια σελίδα από τον ιστότοπό μας, το Django :

- προσδιορίζει το python module root URLconf που θα χρησιμοποιήσει από την αντίστοιχη τιμή της ρύθμισης στο settings.py
- φορτώνει αυτό το module και αναζητά τη μεταβλητή urlpatterns
- διατρέχει κάθε μοτίβο διεύθυνσης URL, με τη σειρά, και σταματά στο πρώτο που ταιριάζει με το URL που ζητήθηκε
- μόλις ένα από τα μοτίβα URL ταιριάζει εισάγει και καλεί το view που ορίστηκε
- Εάν κανένα δεν ταιριάξει, καλεί ένα κατάλληλο error-handling view

Κώδικας 3: Χρήση συνάρτησης view που επιστρέφει, τα δεδομένα του χρήστη και της ημερομηνίας στην index.html σελίδα

```
from django.contrib.auth import logout, login, authenticate
from django.http import HttpResponseRedirect, HttpResponse
from .models import *

def index(request):
        app_user = AppUser.objects.get(user=request.user) # object χρήστη
        now = datetime.datetime.now() #ημερομηνία
        context = {"AppUser": app_user, "time": now}
        return render(request, 'index.html', context)
```

Το Django για να υποστηρίξει διάφορες λειτουργίες HTTP στις συναρτήσεις των views παρέχει τα decorators. Ένα βασικό decorator είναι το “@login_required”. Για να εκτελεστεί η συνάρτηση με αυτό το decorator, ο χρήστης θα πρέπει να έχει αυθεντικοποιηθεί στο σύστημα, σε άλλη περίπτωση να τον ανακατευθύνει στην σελίδα που έχουμε ορίσει για να συνδεθεί σε αυτό. Με τον τρόπο αυτόν δεν μπορεί κάποιος μη εξουσιοδοτημένος χρήστης να έχει πρόσβαση στην εφαρμογή από το URL [19].

Κώδικας 4: Σύνταξη decorator που ελέγχει εάν ο χρήστης έχει αυθεντικοποιηθεί

```
from django.contrib.auth.decorators import login_required

@login_required #decorator
def my_view(request):
```

3.6 Templates

Για το σύστημα των template, η Django παρέχει την Django Template Language (DTL).Τα Django templates είναι αρχεία κειμένου που καθορίζουν τη δομή ή το σχέδιο ενός αρχείου, όπως για παράδειγμα μιας HTML σελίδας. Ένα template περιέχει μεταβλητές, οι οποίες αντικαθίστανται με τιμές και ετικέτες, οι οποίες ελέγχουν τη λογική του template. Ένα view μπορεί να δημιουργήσει δυναμικά μία HTML σελίδα χρησιμοποιώντας HTML template, δημοσιεύοντάς το με τα δεδομένα ενός model.

Κώδικας 5: Παράδειγμα σύνταξης του Django template με τα δεδομένα του παραδείγματος 3

```
<html>
   <body>
     {{AppUser}}, η ώρα είναι {{time}}
   </body>
</html>
```

Η DTL παρέχει ενσωματωμένες ετικέτες και φίλτρα που έχουν σχεδιαστεί για να καλύπτουν τις ανάγκες της λογικής παρουσίασης της εφαρμογής. Παρ 'όλα αυτά μπορούμε να αναπτύξουμε και εξατομικευμένα templatetags προσαρμοσμένα στις ανάγκες μας.

3.5 URLS

Όπως αναφέραμε και στην ενότητα των views, τα python αρχεία ρυθμίσεων «url» χρησιμοποιούνται για να ανακατευθύνουν τα HTTP requests στο σωστό view. Ανάλογα με τη URL τοποθεσία που βρίσκεται ο χρήστης, το Django αναζητάει στα URL μοτίβα που έχουμε ορίσει και εκτελεί την αντίστοιχη συνάρτηση «view» του URL που ταιριάζει.

Κώδικας 6: Σύνταξη url αρχείου

app/urls.py

```
from django.urls import path,include
from . import views
urlpatterns = [
    path('', views.index),#εκτελεί την συνάρτηση index
    path('dashboard/', views.dashboard), #εκτελεί την συνάρτηση dashboard
    path('profile/', views.profile),
    path('documents/<int:sid>/<str:title>', views.paper),
]
```

Ένα ακόμα σημαντικό χαρακτηριστικό των url-μοτίβων είναι ότι μπορούν να δέχονται σαν όρισμα ένα πρότυπο αριθμών ή χαρακτήρων, με αποτέλεσμα οι συναρτήσεις view να επιστρέφουν διαφορετικό αποτέλεσμα ανάλογα με το αντικείμενο.

3.7 Φόρμες

Σε μια εφαρμογή που θέλουμε ο χρήστης να έχει διαδραστικότητα με αυτήν χρειαζόμαστε ένα σύστημα φορμών.

Στην HTML, μια φόρμα είναι μια συλλογή των στοιχείων στο εσωτερικό <form> ... </ form> που επιτρέπουν σε ένα χρήστη εισάγει κείμενο, να χειριστεί αντικείμενα ή ελέγχους, και ούτω καθεξής, και στη συνέχεια να στείλει αυτές τις πληροφορίες πίσω στο διακομιστή. Το Django παρέχει μια σειρά εργαλείων και βιβλιοθηκών που αυτοματοποιούν την διαδικασία αυτή σε μεγάλο βαθμό, καθώς επίσης παρέχουν και ένα επίπεδο ασφάλειας για επιθέσεις Cross site request forgery (CSRF) αλλά και SQL injection.

Το Django χειρίζεται τρία διαφορετικά μέρη για την διαδικασία της φόρμας:

- προετοιμασία και αναδιάρθρωση δεδομένων για να είναι έτοιμα για γραφική απόδοση,
- δημιουργία φορμών HTML για τα δεδομένα,
- λήψη και επεξεργασία υποβαλλόμενων εντύπων και δεδομένων από τον πελάτη

Η επεξεργασία του profile του χρήστη είναι ένα χαρακτηριστικό παράδειγμα της χρήσης του Django forms που γίνεται στην σελίδα αυτή. Τα πεδία της φόρμας δημιουργούνται μέσα στο αρχείο forms.py και αποδίδονται δυναμικά στη σελίδα μέσα από το views.py. Σε εκείνο το σημείο γίνεται και η επεξεργασία των πληροφοριών καθώς και η αποθήκευσή τους.

3.8 Διαχειριστής

Τέλος ένα από τα πιο σημαντικά εργαλεία που παρέχει το Django framework είναι η αυτόματη διεπαφή διαχειριστή. Μέσω αυτού του εργαλείου διαχείρισης, έμπιστοι χρήστες, μπορούνε εύκολα να επεξεργαστούν τα δεδομένα των μοντέλων μας. Κάτι που βοηθάει αρκετά στην ανάπτυξη της εφαρμογής, εφόσον μπορούμε με ευκολία να δοκιμάσουμε την σωστή λειτουργία της βάσης δεδομένων και των μοντέλων έχουμε δημιουργήσει. Η διεπαφή του διαχειριστή είναι προσπελάσιμη από το «localhost:8000/admin», όπου μετά από την είσοδό μας, εμφανίζονται κάποιοι αρχικοί πίνακες.

Σχήμα 3.4: Πίνακες Groups και Users στην διεπαφή του διαχειριστή

Οι πίνακες Groups και Users δημιουργούνται αυτόματα από το Django και χειρίζονται τις αυθεντικοποιήσεις των χρηστών.

3.9 Εγκατάσταση

Αρχικά για να εγκαταστήσουμε το Django Framework χρειαζόμαστε να έχουμε εγκατεστημένη την python. Δημιουργούμε ένα εικονικό περιβάλλον "venv" με την εντολή «python -m venv venv» και το ενεργοποιούμε εκτελώντας το «venv/Scripts/activate». Πάνω σε αυτό θα αποθηκεύονται όλες τις απαραίτητες βιβλιοθήκες που θα χρειαστούμε για την υλοποίηση της εφαρμογής. Με την εντολή «python -m pip install Django» κατεβάζουμε το django framework. Για την δημιουργία του project γράφουμε «django-admin startproject "ονομα_project" . ». Έτσι έχουμε μια δομή όπως φαίνεται παρακάτω:

```
root/
    manage.py
    venv/
    project/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

- Το αρχείο **manage.py** δημιουργείται αυτόματα σε κάθε django project και δείχνει στο αρχείο django-admin.py που χρησιμοποιήσαμε πιο μπροστά για την δημιουργία του project. Από εδώ και πέρα θα χρησιμοποιούμε αυτό το script για να αλληλεπιδράμε με το project.
- Το αρχείο **settings.py** περιέχει όλες τις ρυθμίσεις παραμέτρων της εγκατάστασης του Django.
- Το αρχείο **urls.py** περιέχει την αντιστοίχιση μεταξύ των διευθύνσεων URL
- Τέλος το αρχείο **wsgi.py** περιγράφει πως θα επικοινωνήσει ο web server (πχ Apache) με την εφαρμογή

Κάθε Django project μπορεί να αποτελείται από μια ή περισσότερες εφαρμογές (app), και κάθε app εκτελεί συγκεκριμένες λειτουργίες. Το Django παρέχει κάποια ενσωματωμένα apps όπου εκτελούν τις βασικές λειτουργίες όπως αυτές τις συνεδρίας, της αυθεντικοποίησης, της διαχείρισης, έλεγχο σφαλμάτων κ.α. Για να ξεκινήσουμε ένα app γράφουμε την εντολή «python manage.py startapp my_app». Για κάθε app το framework παράγει τρία σημαντικά αρχεία:

- Models.py: Περιέχει τις οντότητες των μοντέλων της βάσης δεδομένων
- Views.py: Περιέχει όλες τις συναρτήσεις, απαραίτητες για την λογική της εφαρμογής
- Ursl.py: Περιέχει την αντιστοίχιση μεταξύ των διευθύνσεων URL για την συγκεκριμένη εφαρμογή

Κεφάλαιο 4ο: Υλοποίηση της εφαρμογής

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει αναλυτική περιγραφή του αρχικού σχεδιασμού αλλά και λειτουργικότητας της εφαρμογής.

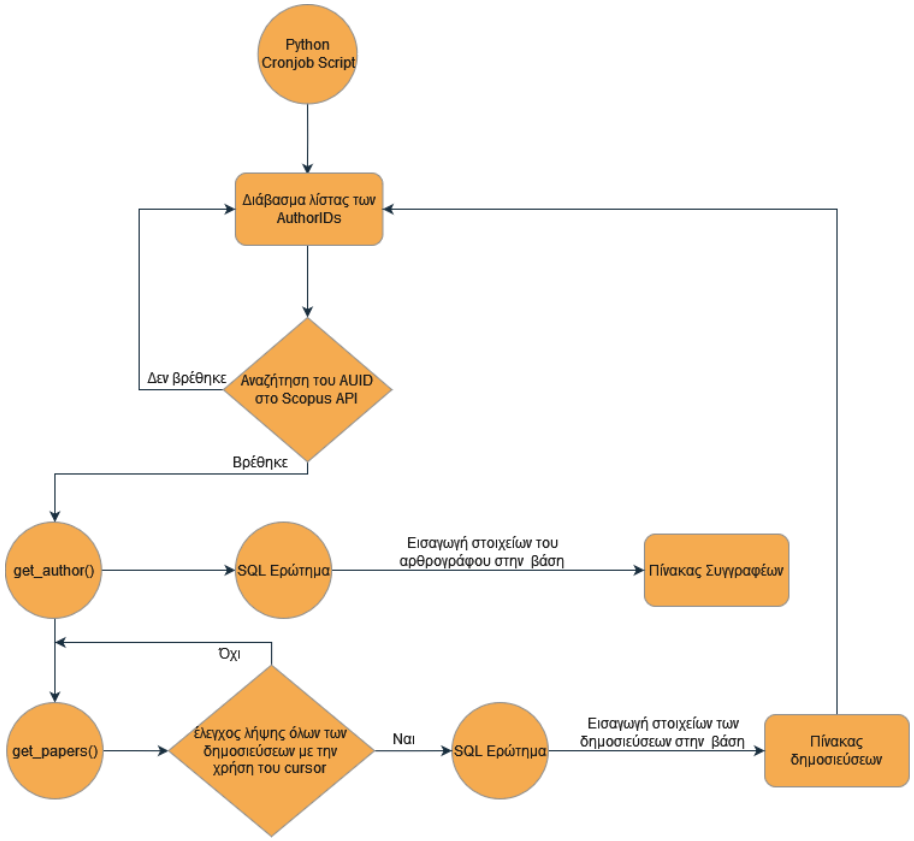
4.2 Αρχικός Σχεδιασμός

Για την σωστή ανάπτυξη μιας εφαρμογής, απαραίτητο είναι να σχεδιάσουμε

- Διαγράμματα ροής
- Διάγραμμα σεναρίων χρήσης
- Πρωτότυπο εφαρμογής

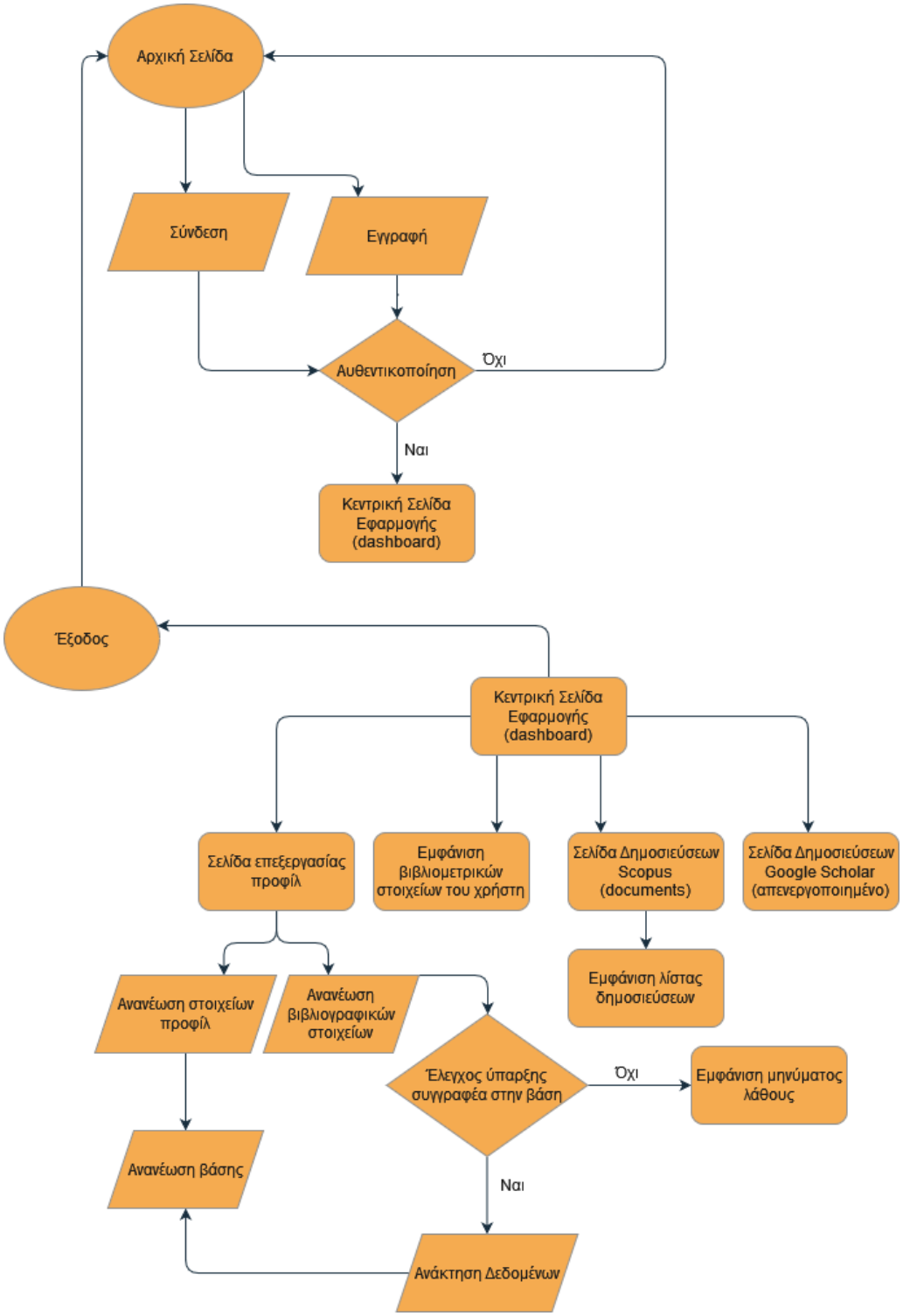
Πάνω σε αυτά θα συνοψίζουμε τα δεδομένα των χρηστών και τις πιθανές ενέργειές τους στο σύστημά μας. Με μια καλή οργάνωση και έναν σωστό σχεδιασμό, ο προγραμματισμός της εφαρμογής γίνεται πολύ πιο εύκολος. Τα διαγράμματα ροής είναι τύπος διαγραμμάτων που αποδίδουν την ροή μιας εργασίας. Η υλοποίηση του python script και της web εφαρμογής φαίνεται στο σχήμα **4.1** και **4.2** αντίστοιχα.

Ο αλγόριθμος του cronjob ξεκινάει διαβάζοντας τα id των ερευνητών από ένα στατικό αρχείο. Έπειτα αναζητάει το καθένα στο API του Scopus και σε περίπτωση που το βρει αποθηκεύει, στην βάση μας, πρώτα τις βασικές μετρικές του ερευνητή και έπειτα τις δημοσιεύσεις του. Όταν τελειώσει η τελευταία διαδικασία προχωράει στο επόμενο αντικείμενο της λίστας. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να τελειώσει η λίστα.



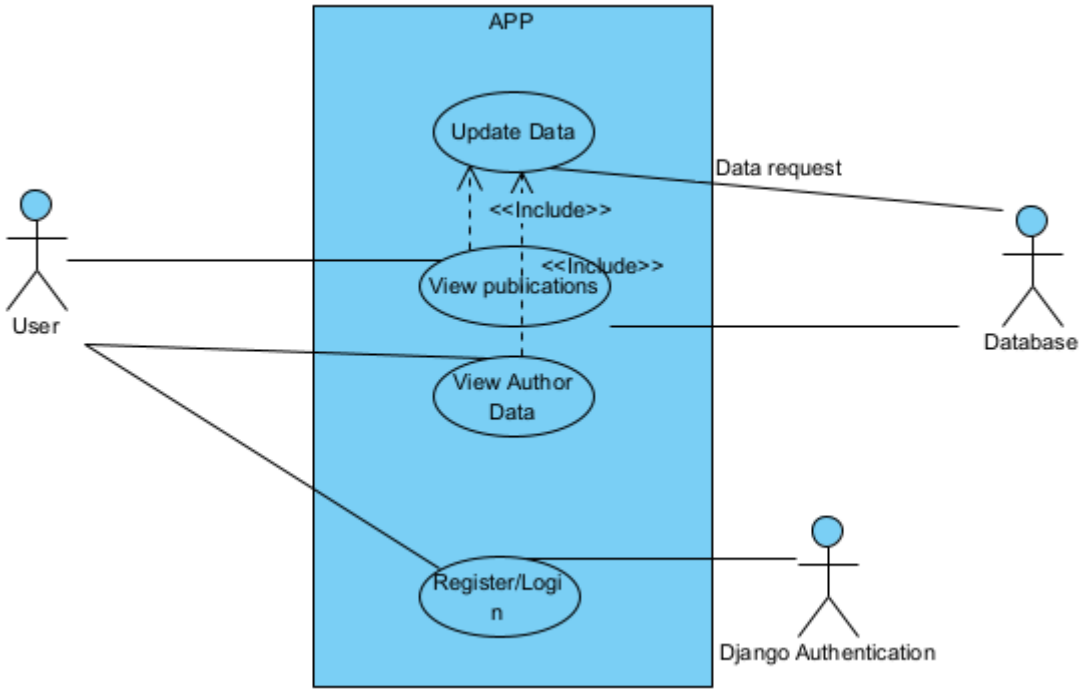
Σχήμα 4.1: Διάγραμμα ροής του Python cronjob script

Στο παρακάτω σχήμα παρατηρούμε ότι ο χρήστης για να μεταφερθεί στην κεντρική σελίδα της εφαρμογής θα πρέπει πρώτα να αυθεντικοποιηθεί. Εφόσον γίνει με επιτυχία ο χρήστης μεταφέρεται στην κεντρική σελίδα απ' όπου μπορεί να επιλέξει να επεξεργαστεί το προφίλ του, να εμφανίσει τα βιβλιομετρικά του στοιχεία ή τις δημοσιεύσεις του αλλά και να αποσυνδεθεί από την εφαρμογή, μεταφέροντας τον έτσι στην αρχική σελίδα.

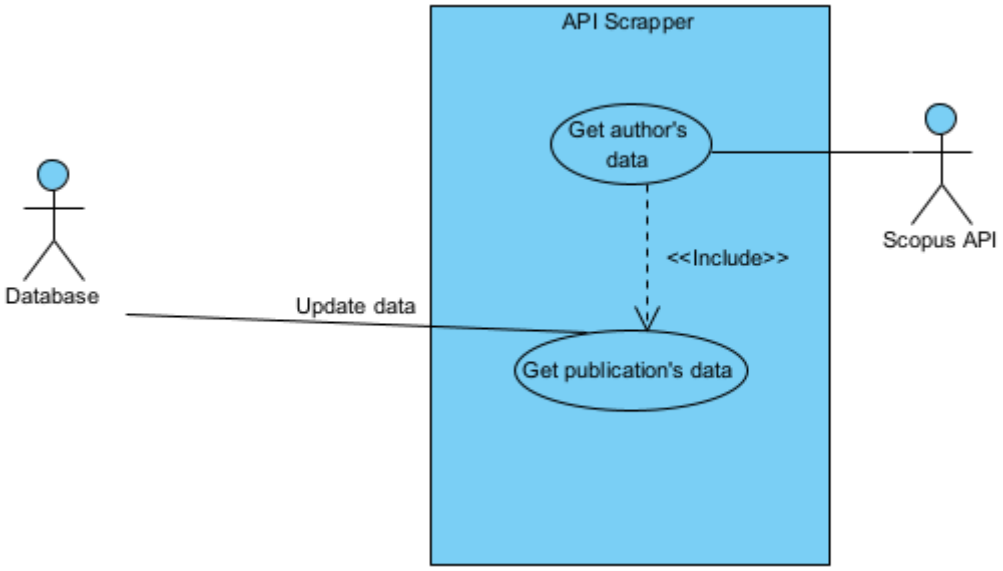


Σχήμα 4.2: Διάγραμμα ροής της web εφαρμογής

Ένα διάγραμμα σεναρίων χρήσης διαφέρει από αυτό της ροής στο ότι παρουσιάζει την «επικοινωνία» των ενεργειών του χρήστη με την εφαρμογή. Παρατηρούμε ότι οι “actors” μπορεί να είναι ένας χρήστης ή ένα σύστημα όπως μια βάση δεδομένων ή αυτό της αυθεντικοποίησης. Τα σενάρια εκπροσωπούνται συνήθως με ένα ρήμα και η σύνδεση με τους “actors” γίνεται με γραμμές.

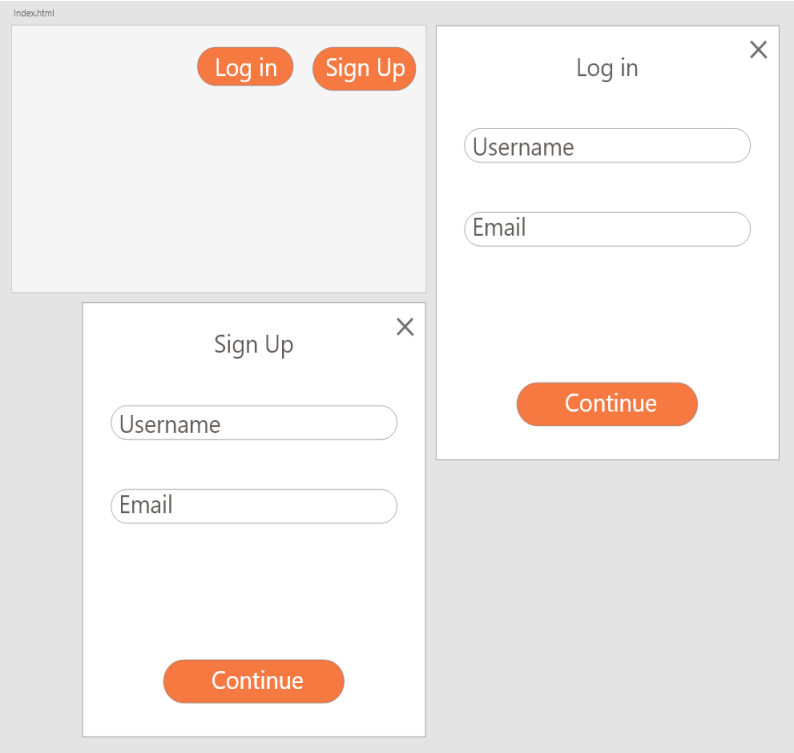


Σχήμα 4.3: Διάγραμμα σεναρίου χρήσης για την αυθεντικοποίηση του χρήστη στην εφαρμογή

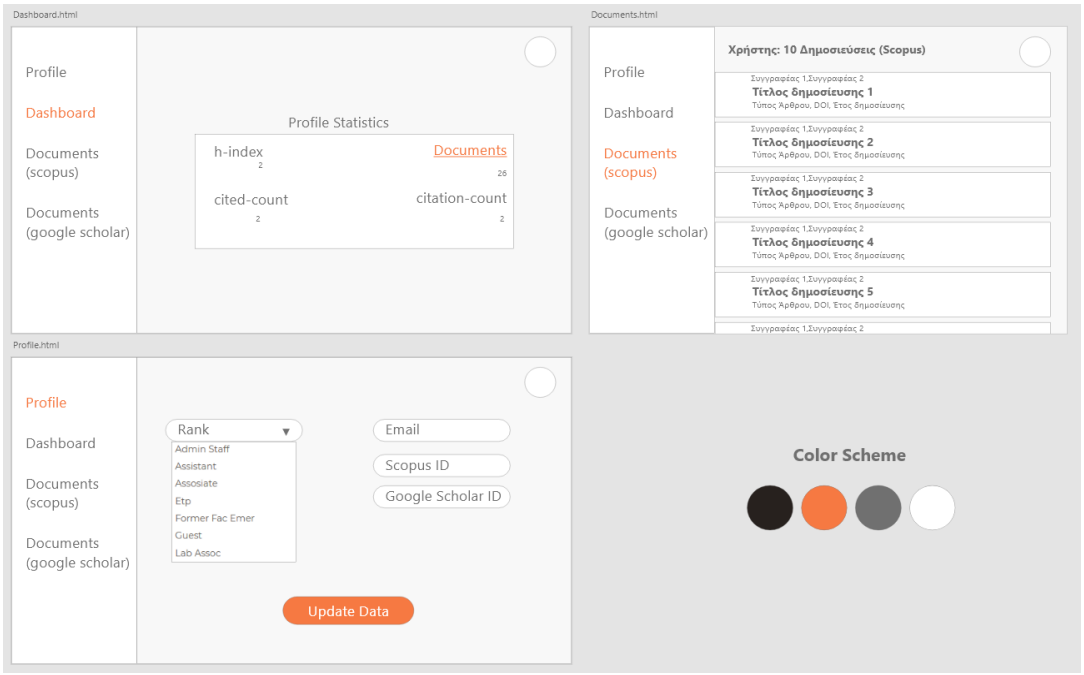


Σχήμα 4.4 Διάγραμμα σεναρίου χρήσης για την ανάκτηση των μετρικών του χρήστη από την βάση

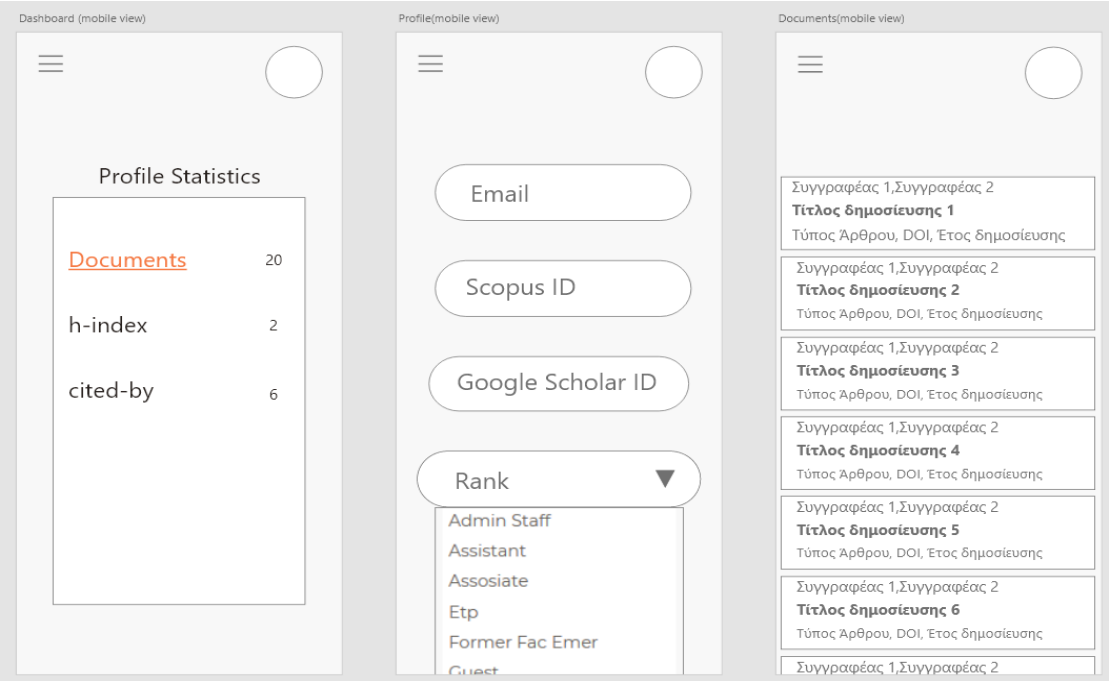
Τέλος με τον σχεδιασμό πρωτότυπων μπορούμε με ευκολία να αποδώσουμε γραφικά τις διεπαφές και τον σκελετό της εφαρμογής που θα βλέπει ο χρήστης, πριν να αρχίσουμε να γράφουμε κώδικα.



Σχήμα 4.5: Πρωτότυπο διεπαφών εγγραφής και σύνδεσης του χρήστη



Σχήμα 4.6: Πρωτότυπο των κυρίων διεπαφών της εφαρμογής



Σχήμα 4.7: Πρωτότυπο διεπαφών σε συσκευή κινητού

4.3 MTV

Όπως αναφέρθηκε στο κεφάλαιο 3, το Django Framework χρησιμοποιεί την αρχιτεκτονική MTV (model,template,view), τα μέρη της οποίας θα αναλυθούν, στις επόμενες ενότητες.

4.3.1 Models

Πέρα από τα μοντέλα που μας παρέχει το Django που αναφέραμε στο 3ο κεφάλαιο, όπως της αυθεντικοποίησης, της συνεδρίας κλπ, δημιουργήσαμε τρία επιπλέον βασικά μοντέλα για τη σωστή λειτουργικότητα της εφαρμογής.

Μοντέλο **AppUser**: Το μοντέλο αυτό περιέχει τα βασικά στοιχεία του χρήστη στην εφαρμογή και επεκτείνει το auth_user. Τα πεδία του είναι:

- user_id: Μοναδικός αριθμός για την συσχέτηση της εγγραφής αυτής με τον χρήστη από το auth_user
- user_email: Το email του χρήστη
- user_firstname: Το όνομα του χρήστη
- user_lastname: Το επίθετο του χρήστη
- user_rank: Την βαθμίδα του χρήστη
- apps_id: Το id του χρήστη στο apps
- user_scopus_id: Το id του χρήστη στο scopus
- user_orcid_id: Το id του χρήστη στο orcid
- user_scholar_id: Το id του χρήστη στο scholar
- user_researcher_id: Το id του χρήστη στο researcher
- hindex: Το hindex του χρήστη βάση του scopus
- documentcount: Το πλήθος των δημοσιεύσεων του χρήστη βάση του scopus
- coauthorcount:
- citedbycount

- citationcount: Το πλήθος αναφορών του χρήστη βάση του scopus
- last_update: Η ημερομηνία που ανανεώθηκαν, τελευταία, τα δεδομένα από το scopus στην βάση δεδομένων

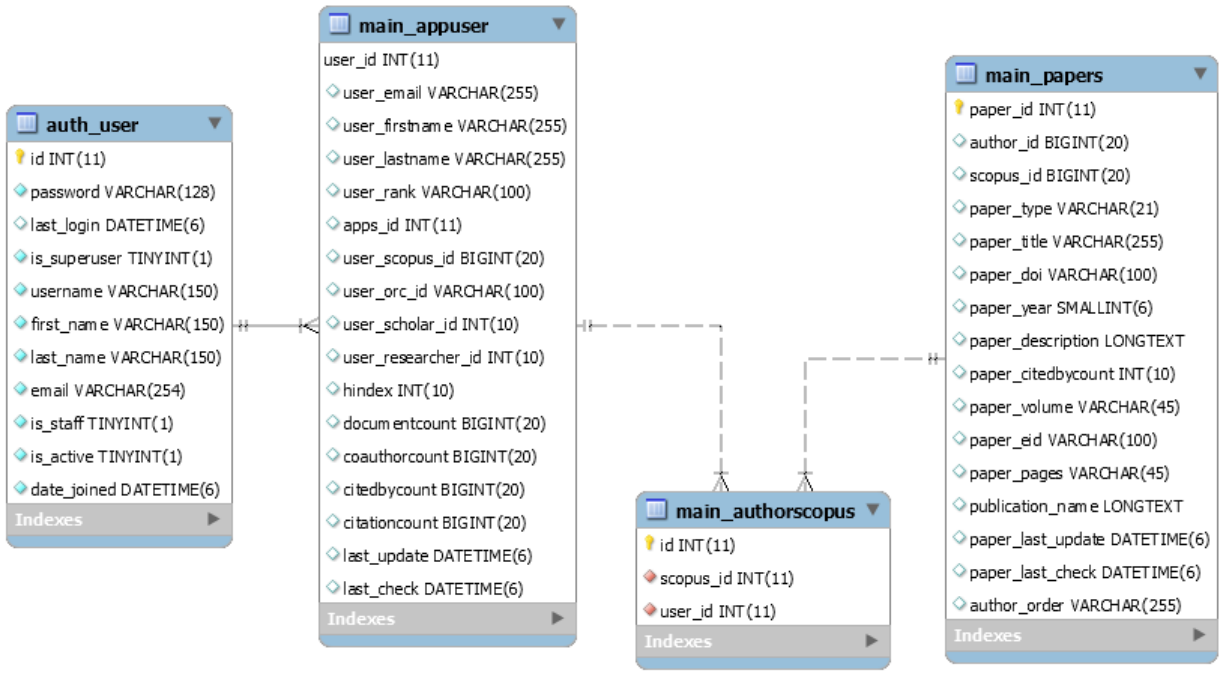
Μοντέλο **Papers**: Το μοντέλο αυτό περιέχει τα δεδομένα των δημοσιεύσεων του χρήστη (από το scopus).

- paper_id: Ο μοναδικός αριθμός της δημοσίευσης στην εφαρμογή
- scopus_id:Ο μοναδικός αριθμός της δημοσίευσης στο scopus
- paper_type: Ο τύπος της δημοσίευσης, για παράδειγμα Journal, ConferenceProceeding, BookSeries, Book
- paper_title: Ο τίτλος του εγγράφου
- paper_doi: Ο μοναδικός κωδικός «doi» της δημοσίευσης
- paper_year: Το έτος δημοσίευσης
- paper_citedbycount: Το πλήθος αναφορών της δημοσίευσης
- paper_volume: Το volume της δημοσίευσης
- paper_pages: Το εύρος των σελίδων της δημοσίευσης
- publication_name: Ο τίτλος της δημοσίευσης
- author_order: Ένα string που δείχνει με την σειρά τα ονόματα των συγγραφέων της δημοσίευσης
- paper_last_update: Η ημερομηνία που ανανεώθηκαν, τελευταία, τα δεδομένα από το scopus στην βάση δεδομένων

Μοντέλο **AuthorScopus**: Το μοντέλο αυτό συσχετίζει τον μοναδικό αριθμό της δημοσίευσης με τον αντίστοιχο του χρήστη συγγραφέα.

- id: Μοναδικός αριθμός της εγγραφής
- scopus_id: Μοναδικός αριθμός της δημοσίευσης

user_id: Μοναδικός αριθμός του χρήστη



Σχήμα 4.8: Διάγραμμα ER κύριων μοντέλων

4.3.2 Templates

Η γραφική απόδοση των αποτελεσμάτων των «view» συναρτήσεων γίνεται φυσικά με τα templates, τα οποία, για να αποφευχθεί η επαναληψιμότητα, υλοποιήθηκαν με κληρονομικότητα.

Σαν «σκελετός» του περιγράμματος της εφαρμογής αναπτύχθηκε ένα βασικό html (base.html). Έπειτα τα υπόλοιπα html αρχεία θα επεκτείνουν αυτό το βασικό και στην θέση των «block» του, θα φορτώνουν τα αντίστοιχα «block» των υπόλοιπων αρχείων. Πέρα από τα βασικό template, επεκτείνουν και κάποια html αρχεία που λειτουργούν σαν components τέτοια είναι τα:

- **navbar** που εμφανίζει το οριζόντιο μενού πλοήγησης
- και **sidebar** που εμφανίζει το πλαϊνό μενού πλοήγησης

Επίσης για καλύτερη οργάνωση, με το «load static» φορτώνουμε τα στατικά css/js ή και media αρχεία που βρίσκονται στην τοποθεσία που έχουμε ορίσει στο settings.py.

Κώδικας 7: Σύνταξη βασικού template αρχείου

app/main/templates/base.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    # Φόρτωση αρχείων css/js από τον φάκελο 'static'
    <link     rel="stylesheet"     type="text/css"     href="{%     static
'./css/bootstrap.min.css' %}" rel="stylesheet">
    <script src="{% static 'js/bootstrap.min.js' %}" defer></script>
    #Τίτλος που αλλάζει ανάλογα με την σελίδα
    <title>{% block title %}{% endblock %}</title>
    # Δυναμική φόρτωση CSS εάν χρειαστεί σε κάποια σελίδα
    {%block page_css%}{% endblock %}
</head>
<body>
    <div class="wrapper ">
        <!-- Start Sidebar - ->
        {% block aside %} #θέση block που θα φορτωθεί το sidebar
        {% endblock %}
        <!-- End Sidebar -->
        <div class="main-panel" style="height: 100vh;">
            <!-- Start Navbar -->
            {% block navbar %} # θέση block που θα φορτωθεί το navbar
            {% endblock %}
            <!-- End Navbar -->
            {% block content %}# θέση block που θα φορτωθεί το περιεχόμενο
της σελίδας{% endblock %}
            <!-- End Content -->
        </div>
    </div>
</body>
</html>
```

Κώδικας 8: Σύνταξη template του dashboard

app/main/templates/dashboard.html

```
{% extends './base.html' %}
{% load static %} # φόρτωση static αρχείων

{% block page_css%}
<link rel="stylesheet" type="text/css" href="{% static
'./css/dashboard.css' %}" rel="stylesheet">{%endblock%}
{% block title %}Dashboard{% endblock title %}

{% block aside %} # θέση block sidebar
{% include './sidebar.html' %} # φόρτωση sidebar.html
{% endblock aside %}

{% block navbar %} # θέση block navbar
{% include './navbar.html' %} # φόρτωση navbar.html
{% endblock navbar %}

{% block content %} #περιεχόμενο dashboard
<section class="content">
    # Εμφάνιση απλού πίνακα με τους τίτλους των δημοσιεύσεων του χρήστη
    <b>{{User}} has {{Papers|length}} papers in Scopus Database</b>
    <table>
        <thead><th>Title</th></thead>
        <tbody>
            {% for paper in papers %}
                <tr><td>{{paper.title}}</td></tr>
            {% endfor %}
        </tbody>
    </table>
</section>
{% endblock content %}
```

Κώδικας 9: Παράδειγμα κώδικα για το navbar component

app/main/templates/navbar.html

```
<nav>
    <div>
        <a
href="https://www.scopus.com/authid/detail.uri?authorId={{AppUser.u
ser_scopus_id}}">{{AppUser}}</a>
    </div>
    <div>
        <a href={% url 'logout' %}>Logout</a>
```

```
    </div>
</nav>
```

4.3.3 Views

Για την σωστή λειτουργικότητα της εφαρμογής αναπτύχθηκαν κάποιες «view» συναρτήσεις. Οι σημαντικότερες περιγράφονται παρακάτω:

- **Index**: Η συνάρτηση αυτή δέχεται σαν όρισμα τα στοιχεία που δίνει ο χρήστης στις φόρμες που του εμφανίζονται. Η μία δημιουργεί τον χρήστη ενώ η άλλη τον συνδέει (εάν υπάρχει) στην εφαρμογή. Εάν κάποια από της φόρμες είναι έγκυρη ή υπάρχει ήδη ενεργή συνεδρία, ο χρήστης ανακατευθύνετε στο κεντρικό μενού της εφαρμογής.
- **Dashboard**: Σε αυτό το view στέλνουμε ένα ερώτημα στην βάση δεδομένων που να επιστρέφει τις δημοσιεύσεις του χρήστη ανά έτος ώστε να παρουσιαστούν στο γράφημα στο dashboard template.
- **Profile**: Η εισαγωγή και η ανανέωση των στοιχείων του χρήστη πραγματοποιείται από την συνάρτηση της profile. Ανάλογα με την εγκυρότητα της κάθε φόρμας η συνάρτηση αποθηκεύει στην βάση τα στοιχεία που έχει δώσει ο χρήστης και έπειτα ανακατευθύνεται στο dashboard.
- **Documents**: Αυτή η συνάρτηση επιστρέφει σαν αποτέλεσμα όλες τις δημοσιεύσεις του χρήστη.

Κώδικας 10: View του dashboard.py

```
@login_required(login_url="index")#decorator για τον έλεγχο αυθεντικοποί-ησης του χρήστη, αλλίως να τον ανακατευθύνει στην αρχική σελίδα index.html
def dashboard(request):
    # get user data from database
    app_user = AppUser.objects.get(user=request.user)
    # get papers where author_id = user's scopups_id from db
    paper_list = Papers.objects.filter(author_id =
app_user.user_scopus_id)
    # «get» returns single row of data
    # «filter» can return many
    context = {'User': app_user, 'Papers':paper_list}
    return render(request,'dashboard.html',context)
```

4.3.3.1 URLS

Είναι πολύ σημαντικό να έχουμε καθαρά και ευανάγνωστα url. Το django μας δίνει τη δυνατότητα να δημιουργήσουμε τα url μας όπως μας αρέσει και με καθαρό τρόπο. Το αρχείο urls.py κρατά όλα τα url για το συγκεκριμένο app.
Τα βασικά urls που καθιστούν τα views λειτουργικά ανάλογα με την «τοποθεσία» του χρήστη είναι τα εξής:

- dashboard/ : Ανακατευθήνει τον χρήστη στην κεντρική σελίδα του χρήστη
- profile/ : Ανακατευθήνει τον χρήστη στην επεξεργασία του προφίλ του
- documents/ : Ανακατευθήνει τον χρήστη στην σελίδα που παρουσιάζονται αναλυτικά οι δημοσιεύσεις του.
- logout/ : Αποσυνδέει τον χρήστη

Κώδικας 11: Σύνταξη κώδικα που πραγματοποιεί την «χαρτογράφηση» των URL

app/main/urls.py

```
from django.urls import path,include
from . import views
urlpatterns = [
    path('', views.index, name='index'),
    path('dashboard/', views.dashboard, name='dashboard'),
    path('profile/', views.profile, name='profile'),
    path('documents/', views.documents, name='documents'),
    path('documents/<int:sid>/<str:title>', views.paper, name='paper'),
    path('logout/',views.userLogout,name='logout')]
```

4.4 Φόρμες

Οι φόρμες που χρησιμοποιούνται στο προφιλ view, από όπου ο χρήστης εισάγει τα στοιχεία του είναι η ProfileForm και η MetricsForm και ο κώδικας περιγράφεται παρακάτω.

Κώδικας 12: Σύνταξη της φόρμας του Profile και του Metrics

app/main/forms.py

```
class ProfileForm(ModelForm):
    class Meta:
        model = AppUser

        fields = ('user_email', 'user_firstname',
                  'user_lastname', 'user_rank', 'apps_id')

        labels = {
            'user_email': 'Email address',
            'user_firstname': 'First Name',
            'user_lastname': 'Last Name',
            'user_rank': 'Rank',
            'apps_id': 'Apps ID',}

class MetricsForm(ModelForm):
    class Meta:
        model = AppUser

        fields = ('user_scopus_id', 'user_orc_id',
                  'user_scholar_id', 'user_researcher_id')

        labels = {
            'user_scopus_id': 'Scopus ID',
```

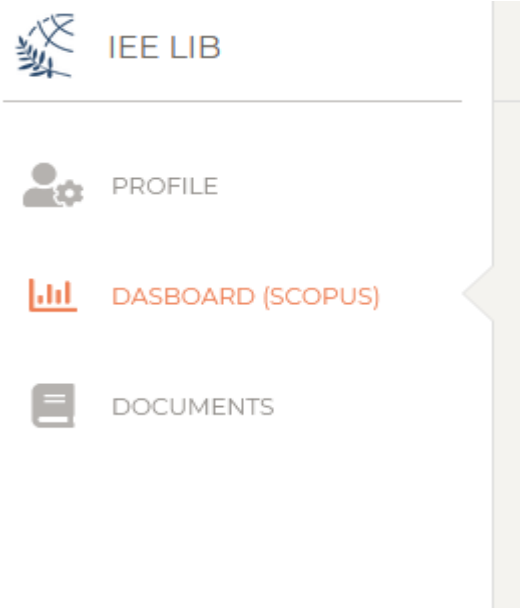
```
            'user_orc_id': 'ORC ID',
            'user_scholar_id': 'Scholar ID',
            'user_researcher_id': 'Researcher ID',}
```

4.5 Σενάρια χρήσης

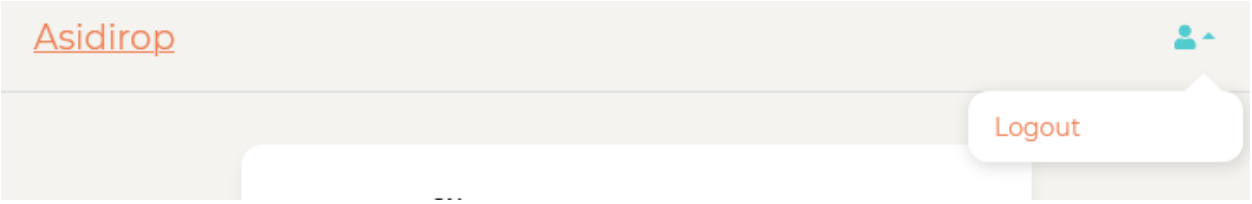
Στην ενότητα αυτήν θα δείξουμε τις πιθανές οθόνες που θα εμφανίζονται στον χρήστη μετά την επιτυχή σύνδεσή του με την εφαρμογή και αυτές της διεπαφής του διαχειριστή.

4.5.1 Μενού Πλοήγησης

Η πλοήγηση του χρήστη, εφόσον συνδεθεί στην εφαρμογή, γίνεται με ένα κάθετο μενού που βρίσκεται στα αριστερά της οθόνης (**Σχήμα 4.9**). Ενώ στο οριζόντιο μενού εμφανίζεται το όνομα του χρήστη που έβαλε στην εγγραφή του και το κουμπί «Logout» που αποσυνδέει τον χρήστη (**Σχήμα 4.10**).



Σχήμα 4.9: Κάθετο μενού πλοήγησης



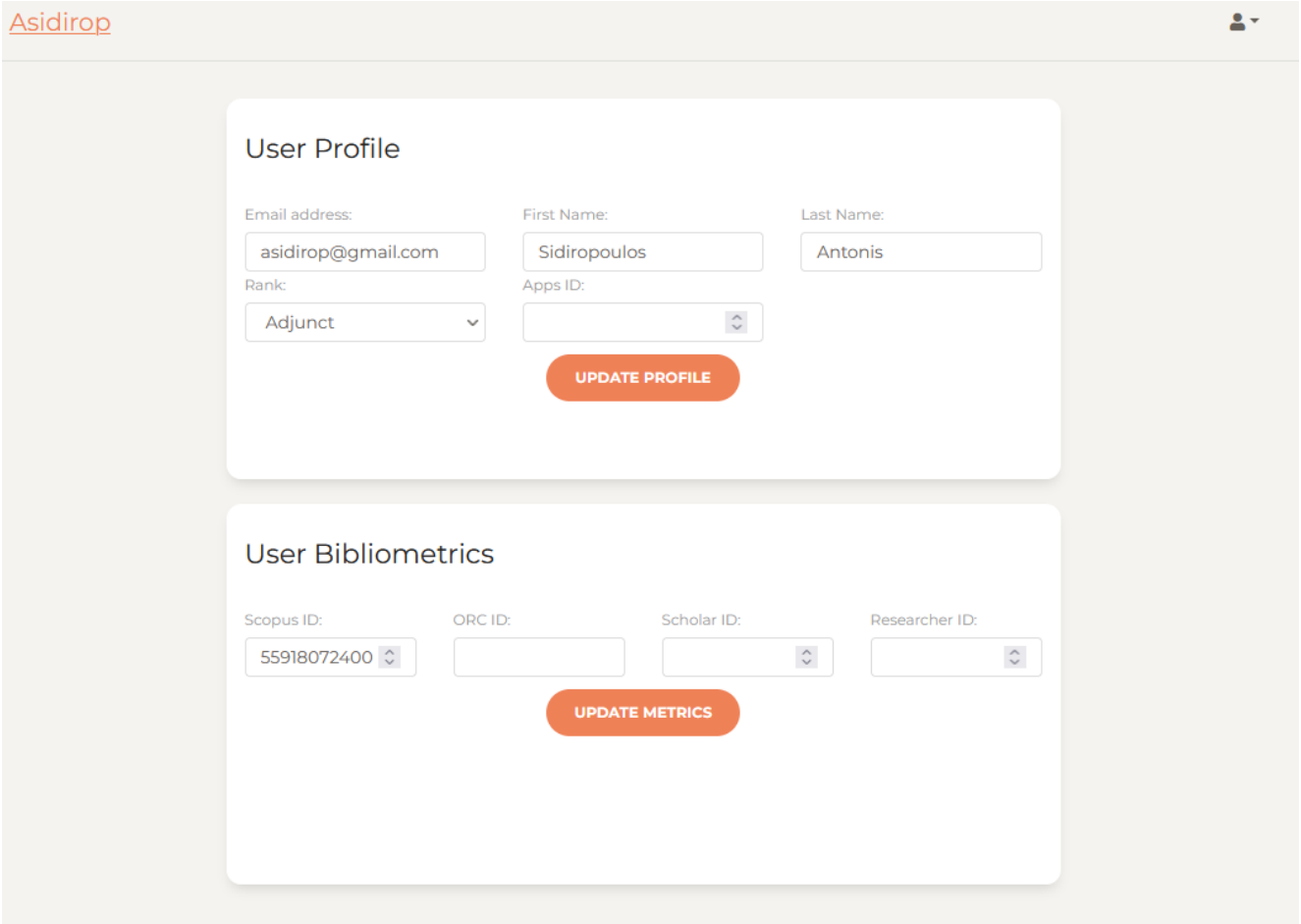
Σχήμα 4.10: Οριζόντιο μενού πλοήγησης

4.5.2 Προφίλ Χρήστη

Στην καρτέλα «profile» υπάρχουν πεδία όπου δίνεται η δυνατότητα στον χρήστη να βάλει κάποια επιπλέον στοιχεία όπως φαίνεται στο **Σχήμα 4.11**.

Με το κουμπί «update profile», ο χρήστης ανανεώνει τα βασικά του στοιχεία στην βάση δεδομένων. Στοιχεία όπως ηλ. ταχυδρομείο, όνομα, επίθετο, βαθμίδα, και το μοναδικό του ID στο APPS.

Με το κουμπί «update metrics», ανάλογα με τα στοιχεία που βάζει ο χρήστης στα πεδία (Scopus ID, ORC ID, Scholar ID, Researcher ID), ελέγχει η εφαρμογή εάν υπάρχουν στην cache βάση τα δεδομένα που αντιστοιχούν σε αυτά. Στην περίπτωση που υπάρχουν, φορτώνονται στην βάση της εφαρμογής και ο χρήστης ανακατευθύνεται στην κεντρική σελίδα της εφαρμογής (dashboard).



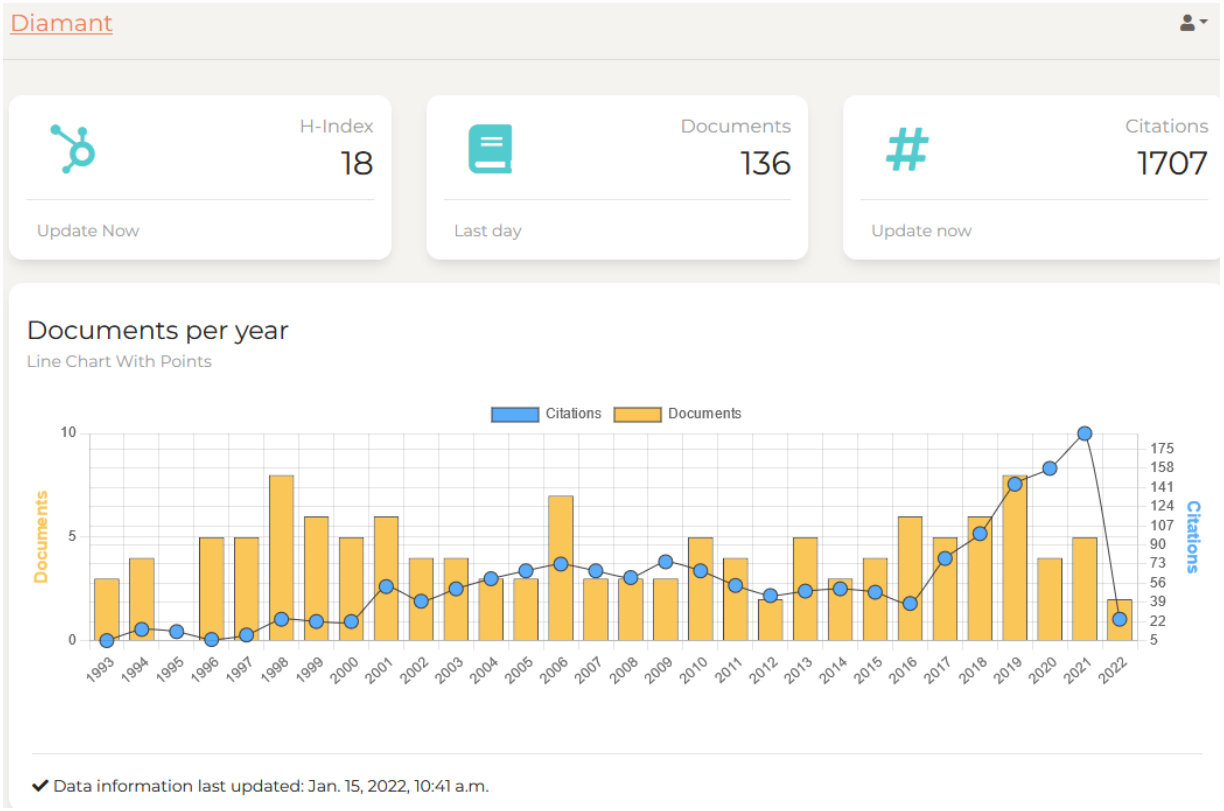
Σχήμα 4.11: Καρτέλα επεξεργασίας προφίλ (profile)

4.5.3 Dashboard

Η σελίδα του dashboard αποτελεί την βασική σελίδα της εφαρμογής καθώς εμφανίζονται τα στοιχεία του χρήστη που αντλήσαμε από την βιβλιοθήκη του scopus. Τα βασικά στοιχεία που εμφανίζονται είναι τα h-index και το συνολικό πλήθος των δημοσιεύσεων και των αναφορών.

Επίσης το γράφημα που δημιουργείται αυτόματα εμφανίζει τον αριθμό των δημοσιεύσεων και αναφορών για κάθε χρόνο. Το γράφημα είναι διαδραστικό και ο χρήστης μπορεί να επιλέξει να βλέπει

μόνο τα citations ή μόνο τα documents ή και τα δύο. Στον κάτω άξονα φαίνεται το έτος, στον αριστερό το πλήθος των δημοσιεύσεων και στον δεξί άξονα το πλήθος των αναφορών. Ο χρήστης μετακινώντας τον κέρσορα πάνω στο γράφημα μπορεί να ελέγξει αναλυτικά και τις τιμές του γραφήματος όπως φαίνεται στο παρακάτω σχήμα. Τέλος κάτω από το γράφημα εμφανίζεται και η ημερομηνία που έγινε τελευταία φορά η ανανέωση των δεδομένων.



Σχήμα 4.12: Καρτέλα εμφάνισης στοιχείων του χρήστη (dashboard)

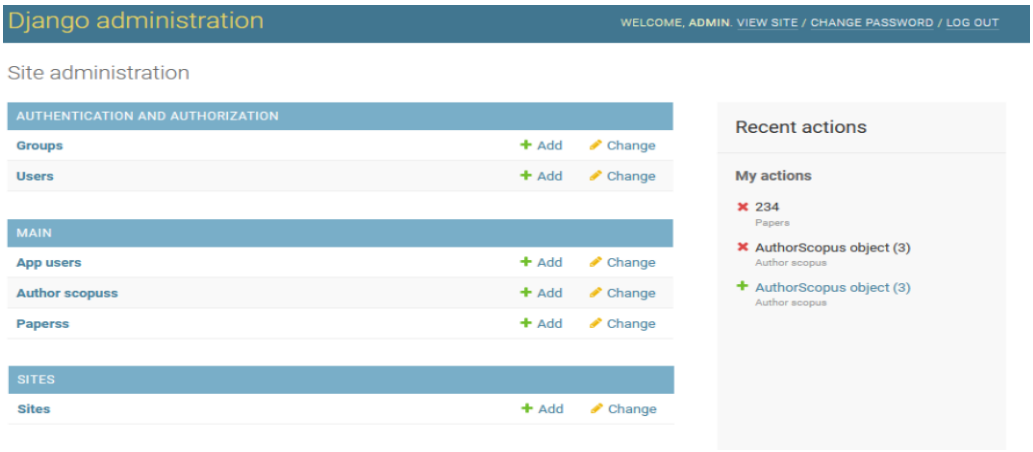
4.5.4 Δημοσιεύσεις

Τέλος στην καρτέλα «documents» (**Σχήμα 4.13**) εμφανίζονται σε σειρές ο τίτλος, οι συγγραφείς, ο τύπος, το doi, οι σελίδες και η χρονιά έκδοσης της δημοσίευσης. Οι τίτλοι των δημοσιεύσεων είναι υπερ-σύνδεσμοι που ανακατευθύνουν τον χρήστη στη σελίδα του scopus όπου παρουσιάζεται αναλυτικά το αντίστοιχο άρθρο.

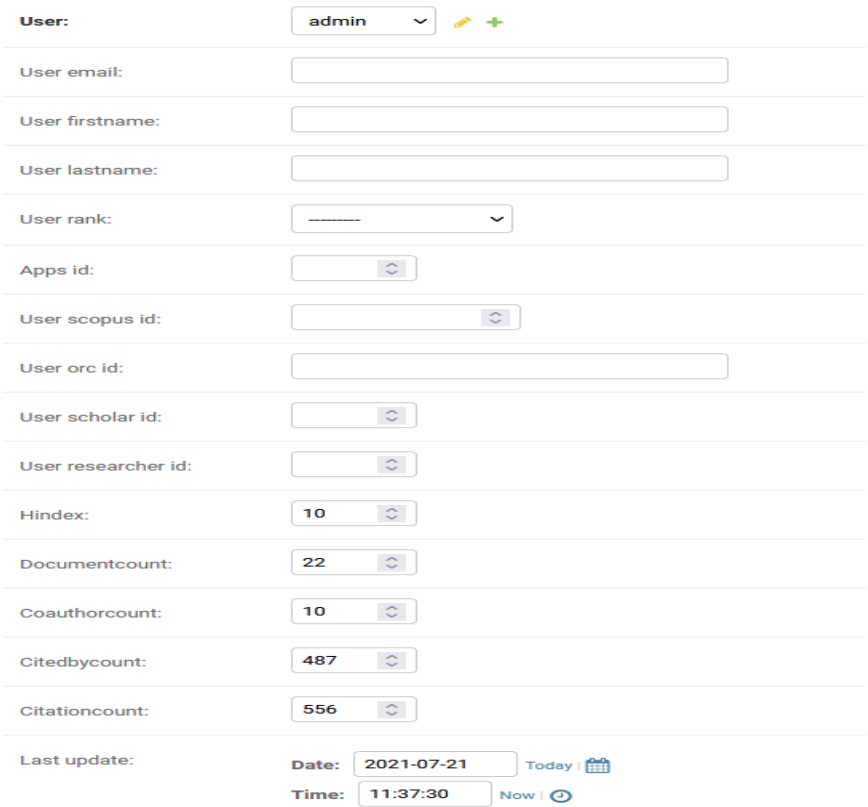
Σχήμα 4.13: Καρτέλα εμφάνισης των δημοσιεύσεων

4.5.5 Διεπαφή Διαχειριστή

Όπως αναφέραμε και στο κεφάλαιο 3, η Django παρέχει μια σημαντική διεπαφή, αυτή του διαχειριστή. Παρακάτω δείχνουμε πως μπορεί αυτός να δει και να επεξεργαστεί διάφορα δεδομένα των μοντέλων και της βάσης μας. Για παράδειγμα στην καρτέλα «Main» - «App Users» φαίνονται οι χρήστες που έχουν κάνει εγγραφή και τα στοιχεία του scopus τους (**Σχήμα 4.14**). Επιπλέον μέσω φόρμας, ο διαχειριστής μπορεί χειροκίνητα να ανανεώσει κάποια στοιχεία. (**Σχήμα 4.15**)



Σχήμα 4.14: Διεπαφή διαχείρισης του Django Framework



Σχήμα 4.15: Φόρμα που εμφανίζονται τα στοιχεία του χρήστη

4.6 Συλλογή δεδομένων από το Scopus API

Για να μπορούμε να ανακτήσουμε τα δεδομένα που χρειαζόμαστε από μια εφαρμογή Β σε μια δικιά μας εφαρμογή Α, χρειάζεται η Β να παρέχει API (Διεπαφή προγραμματισμού εφαρμογών).

Αρκετές μεγάλες πλατφόρμες επιτρέπουν σε τρίτους προγραμματιστές να αναπτύξουν εφαρμογές που μπορούν να βελτιώσουν τη χρήση και τη λειτουργία της κεντρικής πλατφόρμας.

Μια τέτοια πλατφόρμα αποτελεί και η Scopus, όπου μας δίνει την δυνατότητα να χρησιμοποιήσουμε το API της, ώστε να αντλήσουμε τα δεδομένα των αρθρογράφων και των δημοσιεύσεών τους από την βιβλιοθήκη της. Η άδεια μας δίνετε εφόσον συνδεθούμε με ακαδημαϊκό λογαριασμό.

Τα δεδομένα που επιστρέφουν από το Scopus API μετά από τα αιτήματα GET στο *https://api.elsevier.com/content* είναι σε μορφή JSON. Το JSON πρόκειται για ένα πολύ κοινό μορφότυπο δεδομένων που χρησιμοποιείται για την ασύγχρονη επικοινωνία μεταξύ φυλλομετρητή και διακομιστή. Για την ανάκτηση των δεδομένων του συγγραφέα από το API, το GET ερώτημα γίνεται στο "*/author/author_id/authorID?view=ENHANCED*"

όπου χρησιμοποιούνται οι παράμετροι **authorID** (ο μοναδικός αριθμός scopus του συγγραφέα) και **enhanced** για τα πιο αναλυτικά στοιχεία του. Στοιχεία όπως οι μετρικές του στο scopus, δηλαδή το πλήθος εγγράφων, hindex, πλήθος αναφορών και παραπομπών.

Κώδικας 13: Απόκρισης JSON από το API, για τα βασικά στοιχεία του συγγραφέα

```
{
"search-results": {
"opensearch:totalResults": "22",
"opensearch:itemsPerPage": "22",
"entry": [{"citedby-count": "0",
"dc:identifier": "SCOPUS_ID:85102392880",
"eid": "2-s2.0-85102392880",
"dc:title": "Quantifying the Difficulty of Academic Course Modules",
"dc:creator": "Kelesidis K.",
"prism:publicationName": "ACM International Conference",
"prism:pageRange": "245-249",
"prism:coverDate": "2020-11-20",
"citedby-count": "0",
"prism:doi": "10.1145/3437120.3437317",}]}
}
```

Κώδικας 14: Ανάλυση των στοιχείων του συγγραφέα και η εισαγωγή αυτών στην βάση

cronjob/sco.py

```
def get_author_by_auid(auid):
    par = {"apikey": apikey, "httpAccept": content, "view": "enhanced"}
    r = requests.get("{}/{}".format(AUTHOR, auid), params=par)
    js = r.json()
    try:
        data = queries.authors(js, auid)
        queries.insertAuthor(data)
        # print(bcolors.OKGREEN + auid + " inserted" + bcolors.ENDC)
        print(auid + " inserted")
        get_documents(auid)
    except Exception as e:
        print("{}, on get author: {}".format(e, auid))
```

Οι λίστα των κωδικών (authorID) των ερευνητών υπάρχουν σε ένα στατικό αρχείο όπου διαβάζονται από την εφαρμογή.

Στη συνέχεια για την αναλυτική ανάκτηση των εγγράφων του συγγραφέα το GET ερώτημα γίνεται στο:

/search/scopus?query=auid(authorID)&cursor=&count=25&view=COMPLETE.*

Επειδή όμως η απόκριση του ερωτήματος είναι περιορισμένη, από το API, μέχρι και 25 έγγραφα την φορά, χρησιμοποιήθηκε η παράμετρος cursor. Πιο αναλυτικά, για την πρώτη αίτηση, το cursor έχει την τιμή ”*” όπου σαν απάντηση λαμβάνουμε την τιμή @current που είναι ίση με την τελευταία τιμή του cursor και την τιμή @next που η οποία με την σειρά της χρησιμοποιείται για το επόμενο ερώτημα στο API. Όταν οι τιμές @current και @next είναι ίδιες τότε γνωρίζουμε ότι βρισκόμαστε στην τελευταία “σελίδα” των αποτελεσμάτων.

Εφόσον έχουν ληφθεί όλα τα δεδομένα από τις αιτήσεις μας, καλούνται κάποιες συναρτήσεις που δέχονται, αυτά, σαν παράμετρο. Οι συναρτήσεις αυτές πραγματοποιούν σύνδεση με τον απομακρυσμένο διακομιστή, χρησιμοποιώντας την βιβλιοθήκη mysql connector, και στέλνουν δυναμικά τα SQL ερωτήματα με τα δεδομένα που ανακτήσαμε από το scopus API.

Κώδικας 15: Απόκριση JSON του cursor

```
{
"cursor": {
        "@current": "*",
        "@next": "AoNVi75MCEGTlyYxMi1zMi4wLTc1NDQyMzMxOTk="
      },
}
```

Κώδικας 16: Συνάρτησης για SQL ερώτημα προς την βάση

cronjob/queries.py

```
def insertCoAuthor(data):
    statement = (
        """insert into scopus_author(scopus_auid,orcID,lastname,firstname)
        values (%s,%s,%s,%s)
        ON DUPLICATE KEY UPDATE
        orcID = values(orcID),
        lastname = values(lastname),
        firstname = values(firstname)""")
    try:
        q_execute(statement,data)
    except mysql.connector.Error as err:
        pass
```

Κώδικας 17: Συνάρτηση για την σύνδεση και αποστολή των SQL ερωτημάτων

cronjob/queries.py

```
def q_execute(statement,data):
        connection = mysql.connector.connect(
                user=sshUsername, password=sshPassword,
                host='omea.iee.ihu.gr', port="3306",
                database='RPDC', autocommit=True)
        cursor = connection.cursor()
        if len(data) == 1:
           cursor.execute(statement,data[0])
        else:
            cursor.executemany(statement,data)
```

Επειδή το scopus δεν μας επιτρέπει να αντλήσουμε τα δεδομένα των «citations» από το API με την συγκεκριμένη άδεια, χρησιμοποιήσαμε μια άλλη μέθοδο, αυτή του web scraping.
Web scraping ορίζεται ως μια αυτοματοποιημένη διαδικασία που αναλύει το περιεχόμενο DOM των html ιστοσελίδων και εξάγει δεδομένα που χρειαζόμαστε. Στην περίπτωσή μας χρησιμοποιήσαμε την python βιβλιοθήκη selenium. Με αυτήν, μπορεί ο αλγόριθμός μας, μέσω είτε αιτημάτων http είτε ενεργειών στην ιστοσελίδα του scopus, να ψάξει τα στοιχεία που θέλουμε, στην περίπτωσή μας το πλήθος των αναφορών. Έπειτα αυτά αποθηκεύονται σε αρχεία csv από ‘που μπορούν εύκολα να αναλυθούν περεταίρω με μια βιβλιοθήκη όπως την pandas.

Κώδικας 18: Σύνταξη python για την χρήση της βιβλιοθήκης selenium

```
def scrap(scopus_id):
    scopus_authorLink=
"https://www.scopus.com/authid/detail.uri?authorId="+scopus_id
    browser.get(scopus_authorLink)

browser.find_element(By.XPATH,button_citations).send_keys(Keys.RETURN)
# εύρεση του DOM πλήκτρου που μας δείχνει τις αναφορές και ενεργοποίησή
του
    content = response.content
    csv_file = open("./citationCSVs/"+scopus_id+".csv", "wb")
    csv_file.write(content)
    csv_file.close()
```

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

5.1 Συμπεράσματα

Στόχος αυτής της εργασίας ήταν η ανάπτυξη μιας εφαρμογής ιστού για να βοηθήσει στην πιο εύκολη και λιγότερο χρονοβόρα σύγκριση των δημοσιεύσεων των ερευνητών από διάφορες βιβλιογραφικές βάσεις δεδομένων. Για αρχή δίνει την δυνατότητα στον χρήστη να συγκρίνει όλα τα στοιχεία που δίνει το scopus και αργότερα μπορεί να επεκταθεί και σε περεταίρω βιβλιογραφικές βάσεις δεδομένων. Η εφαρμογή δεν αντικαθιστά ούτε το Scopus ούτε τις μελλοντικές βιβλιογραφικές βάσεις δεδομένων αλλά παρουσιάζει γραφικά, με μια πρώτη ματιά, τις σημαντικές μετρικές του χρήστη όπως τις δημοσιεύσεις, την μετρική h-index, το πλήθος αναφορών των ερευνητών και των δημοσιεύσεών τους κα.

Λόγω της χρονοβόρας άντλησης των δεδομένων από το Scopus, η διαδικασία αυτή εκτελείται περιοδικά κάθε εβδομάδα και τα δεδομένα αποθηκεύονται σε μια βάση δεδομένων που λειτουργεί σαν cache για την πλατφόρμα μας.

5.2 Προτάσεις βελτίωσης

Η πλατφόρμα που αναπτύχθηκε αποτελεί μια βάση που καλύπτει προς το παρόν την άντληση δεδομένων από το Scopus. Μια πρώτη, λοιπόν, μελλοντική επέκταση θα ήταν να εφαρμοστούν και άλλες βιβλιογραφικές βάσεις δεδομένων όπως google scholar, dblp κ.α.

Εφόσον οι κωδικοί (authorID) των ερευνητών διαβάζονται από ένα στατικό αρχείο μια, χρήσιμη ακόμη, βελτίωση θα ήταν ο χρήστης να συνδέεται στην εφαρμογή με τα στοιχεία του ιδρυματικού του λογαριασμού. Έτσι τα στοιχεία που υπάρχουν στον τελευταίο να εισάγονται απευθείας στην εφαρμογή μας.

Τέλος μια σημαντική βελτίωση θα ήταν η βελτιστοποίηση των χρόνων εκτέλεσης που έχουν να κάνουν με τις διάφορες εντολές από και προς την βάση δεδομένων. Τέτοιες αφορούν, από την ανάκτηση πληροφοριών των βιβλιογραφικών API έως και την εμφάνισή τους στην πλατφόρμα που αναπτύχθηκε. Αυτό θα έχει και σαν αποτέλεσμα την μείωση των ανενεργών χρόνων του χρήστη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Σαχίνη Ε., Μάλλιου Ν., Χούσος Ν., Καραϊσκος Δ., (2013), Ελληνικές Επιστημονικές Δημοσιεύσεις 1996-2010: Βιβλιομετρική Ανάλυση Ελληνικών Δημοσιεύσεων σε Διεθνή Επιστημονικά Περιοδικά - Scopus, Εθνικό Κέντρο Τεκμηρίωσης, ανακτήθηκε από το https://metrics.ekt.gr/publications/47 την 1η/9/2021

[2] Ζωντανός Κώστας & Κατρανίδης Στέλιος, 2009. Συγκριτική Αξιολόγηση Ερευνητικού Έργου Τμημάτων Οικονομικής Επιστήμης Πανεπιστημίων Ελλάδας και Κύπρου. Θεσσαλονίκη, Πανεπιστήμιο Μακεδονίας

[3] Burnham J. F. (2006). Scopus database: a review. Biomedical digital libraries, 3, 1. https://doi.org/10.1186/1742-5581-3-1

[4] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1420322/?tool=pmcentrez

[5] https://www.elsevier.com/ solutions/scopus/how-scopus-works/content

[6] Παπαβλασόπουλος Σώζων, 2015. Πρακτικός οδηγός κατασκευής και χρήσης Βιβλιομετρικών Δεικτών και Νόμων. Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών, Κάλλιπος. http://hdl.handle.net/11419/4757

[7] M. E. Falagas, E. I. Pitsouni, G. A. Malietzis, and G. Pappas, "Comparison of PubMed, Scopus, Web of Science, and Google Scholar: strengths and weaknesses," *The FASEB Journal*, vol. 22, no. 2, 2008, doi: 10.1096/fj.07-9492lsf.

[8] https://www.w3.org/standards/webdesign/htmlcss

[9] https://developer.mozilla.org/en-US/docs/Web/JavaScript

[10] https://jquery.com/

[11] https://www.chartjs.org/

[12] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems Sixth Edition*. 2016.

[13] R. Connolly and R. Hoar, *Fundamentals of Web Development*. 2014. doi: 10.1007/s13398-014-0173-7.2.

[14] R. T. Fielding and J. F. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," *Internet Engineering Task Force (IETF)*, 2014.

[15] S. Chacon and B. Straub, *Pro Git*. 2014. doi: 10.1007/978-1-4842-0076-6.

[16] https://www.django-rest-framework.org/

[17] https://docs.djangoproject.com/en/3.2/misc/design-philosophies

[18] https://docs.djangoproject.com/en/4.0/topics/http/views/

[19] https://docs.djangoproject.com/en/3.2/topics/http/decorators/

ΠΑΡΑΡΤΗΜΑ Α : Views

```
def userLogout(request):
      logout(request)
      return redirect('index')

def index(request):
      # If user authenticated redirect him to dashboard page
      if request.user.is_authenticated:
      return redirect('dashboard')
      Regform = NewUserForm()
      err = None
      if request.method == "POST":
      # Form for the user registration
      if request.POST.get('submit') == "register":
            form = NewUserForm(request.POST)
            if form.is_valid():
                  username = form.cleaned_data.get('username')
                  password = form.cleaned_data.get('password1')
                  print(User.objects.filter(username=username))
                  # If user exists redirect him else create him
                  try:
                  User.objects.get(username=username)
                  except User.DoesNotExist:
                  user  =  User.objects.create_user(username=username,
password=password)
                  # create user
                  AppUser.objects.create(user=user)
                  # create user's session with login()
                  login(request, user)
                  return redirect('dashboard')
            else:
                  err = form.error_messages
                  print(err)

      # Form for the user log in
      if request.POST.get('submit') == "login":
            username = request.POST.get('login_username')
```

```
            password = request.POST.get('login_password')

      # check if given credentials are valid and redirect user to
dashboard page
       User=authenticate(request,username=username,password=password)
            if user is not None:
                  # create user's session with login()
                  login(request, user)
                  return redirect('dashboard')
            else:
                  err = "Username or Password is wrong."
                  print(err)

      context = {'Regform': Regform, 'err': err}
      return render(request, 'index.html', context)

@login_required(login_url='index')
def dashboard(request):
      # get user
      app_user = AppUser.objects.get(user=request.user)

      try:
      # get papers per year (where author id = app user's scopus id)
      paper=(Papers.objects.filter(author_id = user.user_scopus_id)
                  .values('paper_year')
                  .annotate(Documents = Count('paper_title'))
                  .order_by('paper_year'))
      print(paper)
      labels = [] # year axis
      documents = [] # document axis
      for i in paper:
            print(i)
            labels.append(i['paper_year'])
            documents.append(i['Documents'])

      json = {"documents":documents,"labels":labels}

      except ObjectDoesNotExist:
```

```
    paper = None
    json = {""}

    context = {'AppUser': app_user ,'paper':paper,"json":json}
    return render(request, 'authed/dashboard.html', context)

@login_required(login_url='index')
def profile(request):
    app_user = AppUser.objects.get(user=request.user)

    print(request.POST)

    if "profile" in request.POST:
    pform = ProfileForm(request.POST)
    print("p")

    else:
    pform = ProfileForm()
    mform = MetricsForm()
    context = {"AppUser": app_user, "form_1":pform, "form_2":mform}

    return render(request, 'authed/profile.html', context)


@login_required(login_url='index')
def documents(request):
    app_user = AppUser.objects.get(user=request.user)

    # get user's papers
    Paper=Papers.objects.filter(author_id=app_user.user_scopus_id)
    print(paper)
    # order = AuthorScopus.objects.filter(scopus_auid= app_user)
    # papers = app_user.papers_set.all()

    # papers = zip(scopus, order)
    context = {"AppUser": app_user, "papers":paper}
    return render(request, 'authed/documents.html', context)
```

```
@login_required(login_url='index')
def paper(request, sid, title):
      app_user = AppUser.objects.get(user=request.user)

      paper = Papers.objects.get(paper_id=sid)
      context = {"AppUser": app_user, "paper": paper}

      return render(request, 'authed/paper.html', context)
```