ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΦΑΡΜΟΓΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΔΗΜΟΣΙΕΥΜΕΝΟΥ ΕΠΙΣΤΗΜΟΝΙΚΟΥ ΕΡΓΟΥ

ΚΑΦΑΛΗΣ ΠΑΝΑΓΙΩΤΗΣ

ΑΜ: 1370

Επιβλέπων Καθηγητής: ΓΚΟΓΚΟΣ ΧΡΗΣΤΟΣ

ΑΡΤΑ 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

DEPARTMENT: INFORMATICS AND TELECOMMUNICATIONS

THESIS

RESEARCH ACTIVITY MONITORING APPLICATION

KAFALIS PANAGIOTIS

ID:1370

SUPERVISOR: GOGOS CHRISTOS

ARTA 2022

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι είμαι ο συγγραφέας της παρούσας πτυχιακής εργασίας με τίτλο ΕΦΑΡΜΟΓΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΔΗΜΟΣΙΕΥΜΕΝΟΥ ΕΠΙΣΤΗΜΟΝΙΚΟΥ ΕΡΓΟΥ η οποία δεν αποτελεί προϊόν αντιγραφής ή ανάθεσης σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν για τη συγγραφή της παρούσας εργασίας περιλαμβάνονται τόσο στο κείμενο όσο και στη βιβλιογραφία.

ΚΑΦΑΛΗΣ ΠΑΝΑΓΙΩΤΗΣ

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία έχει ως θέμα την ανάπτυξη μίας εφαρμογής παρακολούθησης δημοσιευμένου επιστημονικού έργου μελών του τμήματος Πληροφορικής και Τηλεπικοινωνιών. Πιο συγκεκριμένα θα αναλυθεί η έννοια του επιστημονικού έργου, θα παρουσιαστεί ο τρόπος της αξιολόγησης του και θα αναφερθούν ιστότοποι και εφαρμογές που βοηθούν στην αξιολόγηση. Στη συνέχεια θα παρουσιαστεί η γλώσσα προγραμματισμού Python, μαζί με κάποιες βιβλιοθήκες και εργαλεία που συνέβαλαν στη δημιουργία της εφαρμογής. Τέλος, θα γίνει αναφορά στην υλοποίηση της εφαρμογής στην οποία ο εκάστοτε χρήστης έχει τη δυνατότητα να αποθηκεύει το δημοσιευμένο ερευνητικό έργο των μελών του Τμήματος και να μπορεί με ευκολία και αμεσότητα να το προβάλλει και να το επεξεργαστεί για να παράγει, μέσω αυτού, τα απαιτούμενα στατιστικά στοιχεία, τους πίνακες και τα γραφήματα που περιέχονται στις ετήσιες εκθέσεις αξιολόγησης του Τμήματος.

Λέξεις-κλειδιά: Επιστημονικές δημοσιεύσεις, ετήσια αξιολόγηση, Api, Python, web scraping, λογισμικά ανάλυσης.

ABSTRACT

The aim of this thesis is the development of an application for monitoring published scientific research projects of members of the Department of Computer Engineering. More specifically, the concept of the research project will be analysed, the manner of its evaluation will be presented and various websites and computer programs that help in the evaluation will be mentioned. In the following, the programming language Python will be introduced, along with some libraries and tools that assisted in creating the application. Finally, reference will be made to the implementation of the application, in which each user will be able to save the published research work of the members of the Department and to easily and directly access and edit it in order for it to be used in the production of the required statistics, tables and graphs contained in the annual evaluation reports of the Department.

Keywords: Scientific publications, annual evaluation Api, Python, web scraping, analysis software.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

1 Έρευνα και επιστημονικές δημοσιεύσεις

1.1 Εισαγωγή

Η ενημέρωση και η πληροφόρηση από το διαδίκτυο είναι μία ακμάζουσα διαδικασία στην εποχή μας. Όλο και περισσότεροι άνθρωποι εγκαταλείπουν τρόπους ενημέρωσης όπως η έντυπη πληροφόρηση (εφημερίδες και περιοδικά), το ραδιόφωνο αλλά ακόμη και την τηλεόραση με αποτέλεσμα να εμπιστεύονται κατά κύριο λόγο το διαδίκτυο. Αυτό σε συνδυασμό με την ελευθερία την οποία προσφέρει το διαδίκτυο και με τη τάση των ανθρώπων να ενδιαφέρονται κυρίως για την ποσότητα και όχι για την ποιότητα, μπορεί να είναι επικίνδυνο καθώς πολλές φορές υπάρχει παραπληροφόρηση που μπορεί να επηρεάσει τις επιλογές του ατόμου. Πολλοί άνθρωποι αναγνωρίζοντας το πρόβλημα της παραπληροφόρησης, που μπορεί να συναντάται σε διάφορα ενημερωτικά μέσα και όχι μόνο στο διαδίκτυο, προσπάθησαν να δημιουργήσουν χώρους που θα απαρτίζονται από ποιοτικότερες πληροφορίες και από πιο έγκυρες πηγές σε επίπεδο υγείας, κοινωνίας και τεχνολογίας. Έτσι διάφοροι επιστήμονες, ερευνητές και όχι μόνο, δημιούργησαν σελίδες και εφαρμογές για να δημοσιεύουν τις έρευνες τους. Στη συνέχεια θα γίνει αναφορά σε αυτές τις σελίδες και τις εφαρμογές αναφορικά με την προσφορά τους στην πληροφόρηση.

1.2 Επιστημονικές δημοσιεύσεις

Επιστημονικές δημοσιεύσεις είναι οι δημοσιεύσεις επιστημόνων σε ειδικές σελίδες και εφαρμογές στις οποίες παρουσιάζουν τις έρευνες και τα αποτελέσματα που πήραν από αυτές. Έτσι η ενημέρωση του κάθε ενός γίνεται ποιοτικότερη και βοηθούν άλλους επιστήμονες με θέματα που τους αφορούν. Μια επιστημονική δημοσίευση μπορεί να είναι:

- Ακαδημαϊκό δοκίμιο
- Διπλωματική εργασία
- Διατριβή
- Εγχειρίδιο
- Ανακοίνωση σε επιστημονικό συνέδριο
- Άρθρο

- Μονογραφία
- Πραγματεία
- Βιβλιοκρισία

1.3 Πλατφόρμες δημοσίευσης επιστημονικών δημοσιεύσεων

Στο διαδίκτυο υπάρχουν πολλές μηχανές αναζήτησης επιστημονικών δημοσιεύσεων, άλλες είναι εξιδεικευμένες πάνω σε έναν τομέα και άλλες στοχεύουν σε ένα ευρύ φάσμα κατηγοριών επιστημονικών δημοσιεύσεων. Επίσης υπάρχουν και άλλες μηχανές στις οποίες μπορεί κανείς να αναζητήσει όλες τις επιστημονικές δημοσιεύσεις που είναι σε ηλεκτρονική μορφή στο διαδίκτυο. Κάποιες δημοφιλείς μηχανές αναζήτησης είναι:

- Openarchives.gr
- Heal-link.gr (Σύνδεσμο Ελληνικών Ακαδημαϊκών Βιβλιοθηκών)
- unioncatalog.gr/ucportal/ (Συλλογικός κατάλογος ελληνικών ακαδημαϊκών βιβλιοθηκών) Library.panteion.gr (Ανθρωποκοινωνικά)
- Ipl.org (Internet Public Library)
- Doabooks.org (Directory of open access book)
- ncbi.nlm.nih.gov/pubmed (Εθνικής Ιατρικής Βιβλιοθήκης των Η.Π.Α)

(ΤΙΦΤΙΚΙΔΗΣ, 2020)

1.3.1 Google scholar

Το Google Scholar είναι μια δωρεάν προσβάσιμη μηχανή αναζήτησης που παρέχει έναν εύκολο τρόπο για εκτενή αναζήτηση επιστημονικής αρθρογραφίας. Δημοσιεύθηκε τον Νοέμβριο του 2004 και περιλαμβάνει επιστημονικά περιοδικά, βιβλία, πρακτικά συνεδρίων, διατριβές, προδημοσιεύσεις, περιλήψεις, τεχνικές εκθέσεις και άλλη επιστημονική βιβλιογραφία. Επίσης για τους ερευνητές, εκτός από την αναζήτηση, παρέχει έναν εύκολο τρόπο κατηγοριοποίησης των άρθρων και των επιστημονικών τους εργασιών, λαμβάνοντας υπόψη τον τίτλο, το όνομα του συγγραφέα, τη μορφή με την οποία δημοσιεύτηκε το έργο και τον αριθμό των παραπομπών που τα έχουν κάνει. Μια έρευνα που δημοσιεύτηκε το 2014 στο περιοδικό *Plos One*, χρησιμοποίησε μια ειδική μεθοδολογία και μπόρεσε να υπολογίσει περίπου ότι το 80-90% όλων αυτών των άρθρων είχαν ήδη δημοσιευθεί στην Αγγλική

γλώσσα, και εκτιμώνται περίπου στα 100 εκατομμύρια[1].Ενώ η Google δεν αποκαλύπτει το μέγεθος του Google Scholar, οι ερευνητές εκτιμούν ότι περιέχει περίπου 389 εκατομμύρια έγγραφα, καθιστώντας το τη μεγαλύτερη ακαδημαϊκή μηχανή αναζήτησης στον κόσμο έως τον Ιανουάριο του 2018[2].

1.3.2 Academia Edu

Το Academia.edu είναι ένας αμερικανικός κερδοσκοπικός ιστότοπος για ακαδημαϊκούς. Έκανε την εμφάνιση του το Σεπτέμβριο του 2008 και ξεκίνησε ως δωρεάν και ανοιχτή αποθήκη άρθρων ακαδημαϊκών περιοδικών και καταχώρησε ένα όνομα τομέα .edu όταν αυτό δεν περιοριζόταν σε εκπαιδευτικά ιδρύματα. Όπως αναφέρεται στην ιστοσελίδα του, το Academia διαθέτει ένα επιχειρηματικό μοντέλο "Freemium", το οποίο παρέχει δωρεάν πρόσβαση στην έρευνα για όλους και πληρωμένες premium δυνατότητες στους συνδρομητές. Η εγγραφή στο Academia Premium δίνει στους ερευνητές πρόσβαση σε προηγμένα εργαλεία ανακάλυψης της έρευνας και δίνει στους συγγραφείς βελτιωμένα εργαλεία ανάλυσης και παρακολούθησης αντικτύπου. Οι ακαδημαϊκοί έχουν ανεβάσει 37 εκατομμύρια εργασίες και 98 εκατομμύρια ακαδημαϊκοί, επαγγελματίες και φοιτητές διαβάζουν άρθρα για το Academia κάθε μήνα[3].

1.3.3 Microsoft Academic

Το Microsoft Academic ήταν μια δωρεάν δημόσια μηχανή αναζήτησης ιστού για ακαδημαϊκές δημοσιεύσεις και βιβλιογραφία, που αναπτύχθηκε από τη Microsoft Research. Είχε την δυνατότητα να μετρά πόσο συχνά αναφέρεται ένα συγκεκριμένο άρθρο, μετρώντας έτσι τον αντίκτυπο αυτής της εργασίας στην παγκόσμια ερευνητική κοινότητα. Η μηχανή αναζήτησης παρείχε όχι μόνο αποτελέσματα αναζήτησης και πρόσβαση σε πηγές, αλλά και πληροφορίες όπως τον αριθμό αναφορών, g -index και h -

[1] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0093949
[2] https://link.springer.com/article/10.1007/s11192-018-2958-5
[3] https://www.academia.edu/about

index. Εκτός από τις ακαδημαϊκές δημοσιεύσεις, χρησιμοποιούταν επίσης για την εύρεση ιστότοπων που περιέχουν κρατικά και τοπικά αρχεία. Οι κριτικές δήλωναν ότι η Microsoft Academic είναι ανταγωνιστής του Google scholar, του Web of Science και του Scopus για σκοπούς ακαδημαϊκής έρευνας καθώς και ανάλυση παραπομπών. Η Microsoft ανακοίνωσε[4] ότι ο ιστότοπος και τα API της Microsoft Academic αποσύρονται στις 31 Δεκεμβρίου 2021 και έτσι δεν είναι πλέον ενεργό.

1.3.4 ResearchGate

Το ResearchGate είναι ένας ευρωπαϊκός ιστότοπος κοινωνικής δικτύωσης για επιστήμονες και ερευνητές όπου μοιράζονται έγγραφα, κάνουν και απαντούν σε ερωτήσεις και βρίσκουν συνεργάτες. Ιδρύθηκε το 2008 και είναι το μεγαλύτερο ακαδημαϊκό κοινωνικό δίκτυο από άποψη ενεργών χρηστών, σύμφωνα με μελέτη του 2014 από το Nature[5]. Αν και δεν απαιτείται εγγραφή για την ανάγνωση άρθρων, τα άτομα που επιθυμούν να γίνουν μέλη του ιστότοπου πρέπει είτε να έχουν μια διεύθυνση email σε ένα διαπιστευμένο ίδρυμα είτε να προσδιορίζονται με μη αυτόματο τρόπο ως δημοσιευμένοι ερευνητές. Κάθε μέλος του ιστότοπου έχει ένα προφίλ χρήστη και μπορεί να ανεβάσει αποτελέσματα έρευνας, συμπεριλαμβανομένων εγγράφων, δεδομένων, κεφαλαίων, διπλωμάτων ευρεσιτεχνίας, ερευνητικών προτάσεων, μεθόδων, παρουσιάσεων και πηγαίου κώδικα λογισμικού. Οι χρήστες μπορούν επίσης να παρακολουθούν τις δραστηριότητες άλλων χρηστών και να έχουν συζητήσεις μαζί τους.

1.4 Αξιολόγηση επιστημονικού έργου

1.4.1 Αναφορές

Ως αναφορά ορίζεται κάθε πράξη δημόσιας αποκάλυψης ενός γεγονότος, προσώπου ή αντικειμένου και έχει πάντα σκοπό να ενημερώσει τον αντίστοιχο παραλήπτη σε τι αναφέρεται και σε τι βασίζεται το έργο ή ένα μέρος του. Ακόμα κι αν το ξένο δημιούργημα δεν παρουσιάζεται έτσι στο έργο, αλλά χρησιμοποιείται ως βάση για περαιτέρω ανάλυση, θα πρέπει να αναφέρεται για τη διατήρηση της επιστημονικής

[4] https://www.microsoft.com/en-us/research/project/academic/articles/microsoft-academic-to-expand-horizons-with-community-driven-approach/
[5] https://www.nature.com/news/online-collaboration-scientists-and-the-social-network-1.15711

ακεραιότητας. Επομένως, οποιαδήποτε αναφορά σε άλλο έργο θα πρέπει να σημειώνεται στο κείμενο καθώς και στο τέλος της εργασίας. Με αυτόν τον τρόπο, εκτός από την εγγύηση των πνευματικών δικαιωμάτων του δημιουργού, δίνεται ιδιαίτερη προσοχή και σεβασμός προς το πρόσωπο του. Οι αναφορές προστατεύουν τα πνευματικά δικαιώματα των δημιουργών και αποτελούν μέσο αξιολόγησης επιστημονικών εργασιών. Οι παραπομπές είναι απαραίτητες για να καταδειχθεί η ποιότητα και η αναγνώριση της εργασίας. Υπάρχουν εργασίες οι οποίες θεωρούνται πολύ σημαντικές για τον κλάδο στον οποίο ανήκουν. Η αναφορά σε τέτοιες εργασίες προσδίδει ένα κύρος στις εργασίες που τις περιέχουν.

Σε ένα επιστημονικό και μη έργο, οι αναφορές είναι ένα από τα βασικότερα πράγματα που πρέπει να περιέχονται. Με τις αναφορές το πρόσωπο που γράφει το έργο υποστηρίζει τις ιδέες του αλλά και τεκμηριώνει τα λεγόμενά του. Επίσης διασφαλίζουν τα πνευματικά δικαιώματα ενός συγγραφέα και αποτελούν ένα μέσο αξιολόγησης ενός επιστημονικού έργου.

(Γεώργιος Σελίμης, 2016)

1.4.2 Μετρικές

Η αναφορά έχει μια ποικιλία μετρήσεων που δείχνουν τον αντίκτυπο ενός έργου, ενός περιοδικού ή ακόμα και ολόκληρου του έργου ενός συγγραφέα. Ο βαθμός επιρροής (impact factor) υπολογίζεται διαιρώντας τον συνολικό αριθμό των αναφορών των εργασιών ενός περιοδικού τα τελευταία δύο χρόνια με τον συνολικό αριθμό εργασιών που δημοσιεύθηκαν σε αυτό το περιοδικό. Τα στοιχεία που προέρχονται από την παραπάνω διαδικασία είναι διαθέσιμα από το Journal Citation Reports (JCR). Η συχνότητα με την οποία χρησιμοποιείται ένα έργο από ερευνητές του πεδίου φαίνεται από τις αναφορές στα άρθρα του.

Τα στοιχεία αναφορών της υπηρεσίας JCR είναι διαθέσιμα κάθε χρόνο σε δύο διαφορετικές εκδόσεις:

• JCR Science Edition, με στοιχεία από περίπου 5900 περιοδικά στις φυσικές και εφαρμοσμένες επιστήμες, στην τεχνολογία και στις επιστήμες υγείας.

• JCR Social Sciences Edition, με στοιχεία από περίπου 1700 περιοδικά στις κοινωνικές, ανθρωπιστικές και οικονομικές επιστήμες.

Υπάρχουν και δύο άλλοι αριθμοί που σχετίζονται με μετρικούς δείκτες, ο SJR (SCImago Journal Ranking) και ο SNIP (Source Normalized Impact per Paper). Ο SJR υπολογίζει τη σημασία και τον αντίκτυπο ενός περιοδικού υπολογίζοντας τις αναφορές που λαμβάνει κάθε έργο του από αξιολογημένα περιοδικά. Ο SNIP είναι ένας δείκτης που υπολογίζει τον αντίκτυπο ενός περιοδικού (υπολογισμένος από τα προηγούμενα τρία χρόνια) με βάση την κατάσταση διαφορετικών ερευνητικών πεδίων (κάθε πεδίο έχει διαφορετική βαρύτητα και διαφορετικό δυναμικό αναφοράς)

(Γεώργιος Σελίμης, 2016)

1.4.3 Οι μετρικές h-index, i10 κ.α

Οι σημαντικότερες μετρικές που ενδιαφέρουν τους συγγράφεις, προκύπτουν από της συνολικές δημοσιεύσεις τους και των αριθμό των αναφορών που έχουν λάβει. Τέτοιες μετρικές διαθέτουν διαδικτυακά προγράμματα, όπως το Microsoft Academic Search, το Scopus, το Web of Science και το Google Scholar:

• **Δείκτης h (h-index) του Hirsch**, ίσως από τους σημαντικότερους δείκτες που μετρά την ποιότητα και την απήχηση των επιστημονικών δημοσιεύσεων ενός επιστήμονα. Δηλαδή κατά πόσο ποιοτικά είναι τα επιστημονικά άρθρα και πόσο παραγωγικός είναι ένας συγγραφέας. Το *h* -index συσχετίζεται με προφανείς δείκτες επιτυχίας, όπως η απόκτηση του Νόμπελ, η αποδοχή υποτροφιών έρευνας και η κατοχή θέσεων σε κορυφαία πανεπιστήμια. Ο δείκτης *h* ορίζεται ως η μέγιστη τιμή του *h* έτσι ώστε ο συγκεκριμένος συγγραφέας/περιοδικό να έχει δημοσιεύσει τουλάχιστον *h* εργασίες που έχουν αναφερθεί στο καθένα τουλάχιστον *h* φορές. Για παράδειγμα, ο δείκτης h=50 φανερώνει πως ο επιστήμονας έχει δημοσιεύσει τουλάχιστον 50 επιστημονικά άρθρα, τα οποία έχουν λάβει πάνω από 50 αναφορές το κάθε ένα από άλλους επιστήμονες.

• **Δείκτης g (g-index) του Egghe**, επιδιώκει να βελτιώσει τον δείκτη h, δίνοντας μεγαλύτερη βαρύτητα στα ιδιαίτερα αναφερόμενα άρθρα. Υπολογίζεται ταξινομώντας τα άρθρα σε φθίνουσα σειρά αναφορών και είναι ο μεγαλύτερος αριθμός έτσι ώστε g πρώτα άρθρα να έχουν λάβει συνολικά τουλάχιστον g^2 αναφορές.

• **Ο δείκτης i10** φανερώνει τον αριθμό των άρθρων που έχουν 10 παραθέσεις τουλάχιστον.

(Χρήστος, 2015)

2 Λογισμικά ανάλυσης επιστημονικών δημοσιεύσεων

2.1 Publish or Perish

Ο όρος Publish or Perish εμφανίστηκε στις Ηνωμένες Πολιτείες, κυρίως στον ακαδημαϊκό χώρο, για να αντικατοπτρίζει την πίεση που αντιμετωπίζουν οι ακαδημαϊκοί να δημοσιεύσουν επιστημονικά έργα ή να υποστηρίξουν τη σταδιοδρομία τους όσο το δυνατόν γρηγορότερα και αποτελεσματικότερα. Όταν ένας ερευνητής τείνει να δημοσιεύει την έρευνά του σε τακτική βάση, έχει την ευκαιρία να επιδείξει το ακαδημαϊκό ταλέντο και τραβήξει το ενδιαφέρον για αυτόν και το ίδρυμά του, ώστε να επωφελείται από τη συνεχή χρηματοδότηση του ερευνητικού έργου από ακαδημαϊκά ιδρύματα.

Το Publish or Perish δεν είναι απλώς μια φράση, αλλά το όνομα του προγράμματος λογισμικού που χρησιμοποιείται σε αυτήν την εργασία. Ο σκοπός αυτού του προγράμματος είναι να λαμβάνει ακατέργαστες αναφορές από το Google Scholar, να τις αναλύει και να παρέχει τις ακόλουθες μετρήσεις:

- Συνολικός αριθμός εγγράφων και συνολικές παραπομπές.
- Μέσος αριθμός αναφορών ανά άρθρο, ανά συγγραφέα και ανά χρόνο
- Ο δείκτης h του Hirsch και σχετικές παράμετροι
- Ο δείκτης g του Egghe
- Τρεις ξεχωριστές παραλλαγές h-index
- Μέσος ετήσιος ρυθμός αύξησης ενός μεμονωμένου δείκτη h
- Η ανάλογη αξία που παίρνει κάθε χρόνο η παραπομπή λόγω σημαντικότητας.
- Μια ανάλυση του αριθμού συντακτών ανά έγγραφο.

Επίσης, η ανάλυση του Publish or Perish μπορεί να μας δώσει ένα σημαντικό στοιχείο για τους ερευνητές: Όσο καλύτερος είναι ο δείκτης αναφοράς του, τόσο πιο πιθανό είναι να έχει αντίκτυπο και να είναι σημαντικός παράγοντας στην ανάπτυξη κάθε κλάδου. Το αντίθετο όμως δεν πρέπει να οδηγεί στο συμπέρασμα πως ο ερευνητής δεν είναι εξίσου σημαντικός, καθώς οι κακές μετρήσεις στις αναφορές των έργων του, μπορεί να οφείλονται στον τομέα που αφορά η δημοσίευση ή στη γλώσσα που δημοσιεύτηκε.

(Γεώργιος Σελίμης, 2016)

2.2 Researcher

Η εφαρμογή Researcher[6] είναι μια εφαρμογή αναζήτησης επιστημονικού έργου για κινητές συσκευές Android και iOS . Ιδρύθηκε το 2017 και πλέον μετράει πάνω από 2.000.000 εγγεγραμμένους χρήστες.

Οι χρήστες μπορούν να ακολουθήσουν τις τελευταίες εργασίες από περισσότερα 18.000 περιοδικά που καλύπτουν όλους τους κύριους κλάδους:

- Τέχνες και Ανθρωπιστικές Επιστήμες
- Διοίκηση Επιχειρήσεων
- Χημεία και Επιστήμη Υλικών
- Επιστήμες της Γης και Γεωγραφία
- Οικονομικά και Οικονομικά
- Μηχανική και Επιστήμη Υπολογιστών
- Επιστήμες Ζωής και Βιολογία
- Ιατρικές και Επιστήμες Υγείας
- Φυσική και Μαθηματικά
- Κοινωνικές Επιστήμες και Ψυχολογία

Επίσης, με την λειτουργία που ονομάζεται Researcher Live, επιτρέπει στους χρήστες να φιλοξενήσουν μια ζωντανή συνεδρία μέσα στην εφαρμογή και να παρουσιάσουν διαφάνειες ή εικόνες ως μέρος της ομιλίας τους. Χρήστες ή καλεσμένοι που διαθέτουν σύνδεσμο πρόσκλησης μπορούν να μπουν και να ακούσουν τη συνεδρία από το τηλέφωνο ή τον υπολογιστή τους. Οι ακροατές που συνδέονται δια ζώσης μπορούν να σηκώσουν τα

[6] https://www.researcher-app.com/

χέρια τους για να κάνουν μια ερώτηση, με τον οικοδεσπότη να μπορεί να επιλέξει ποιον θα καλέσει.

2.3 Go scholar or Google scholar

Το GoScholar[7] είναι μια εφαρμογή που δημιουργήθηκε με αναφορά στον επίσημο ιστότοπο https://scholar.google.com.pk/. Το συγκεκριμένο λογισμικό ανακτά πληροφορίες από τη μηχανή αναζήτησης Google Scholar. Σκοπός της εφαρμογής είναι να βοηθήσει τους χρήστες να έχουν πρόσβαση σε διάφορα άρθρα όπου κι αν βρίσκονται, αρκεί να συνδεθούν στο διαδίκτυο. Το GoScholar βρίσκεται μόνο σε APK μορφή καθώς δεν είναι επίσημη εφαρμογή της google και δεν κυκλοφορεί στα γνωστά App Stores.

3 Λογισμικά που απευθύνονται σε προγραμματιστές

3.1 Χρήση API

Το API είναι η συντομογραφία του Automatic Programming Interface, που μεταφράζεται ως **Διεπαφή Προγραμματισμού Εφαρμογών**, είναι ένα είδος ενδιάμεσου λογισμικού που επιτρέπει σε δύο εφαρμογές να επικοινωνούν μεταξύ τους. Τα API συνδέουν προγραμματιστές και επικοινωνούν χρησιμοποιώντας πρωτόκολλα. Το πρωτόκολλο συνδέει ολοκληρωμένο ή και ξεχωριστό λογισμικό, με τη διεπαφή(API) να είναι το μέσο με το οποίο τα δύο συστήματα μπορούν να επικοινωνούν. Τα περισσότερα λειτουργικά περιβάλλοντα παρέχουν API όπως και οι περισσότεροι μεγάλοι ιστότοποι.

[7] https://apkcombo.com/google-scholar-go-scholar/com.wGoogleScholar_7814448/

3.1.1 Πώς λειτουργεί ένα API

Τα API βρίσκονται μεταξύ μιας εφαρμογής και του διακομιστή ιστού, λειτουργώντας ως ενδιάμεσο επίπεδο που επεξεργάζεται τη μεταφορά δεδομένων μεταξύ συστημάτων.

Δείτε πώς λειτουργεί ένα API:

- Οι εφαρμογές πελατών εκκινούν κλήσεις API για ανάκτηση πληροφοριών, γνωστές και ως αιτήματα. Αυτό το αίτημα επεξεργάζεται η εφαρμογή στον διακομιστή ιστού μέσω του API, συμπεριλαμβανομένου του ρήματος αιτήματος, των κεφαλίδων και μερικές φορές του σώματος αιτήματος.
- Μετά τη λήψη ενός έγκυρου αιτήματος, το API καλεί ένα εξωτερικό πρόγραμμα ή διακομιστή web.
- Ο διακομιστής στέλνει μια απάντηση στο API ζητώντας πληροφορίες.
- Το API μεταφέρει τα δεδομένα στην αίτηση της αρχικής εφαρμογής.

Ενώ η μεταφορά δεδομένων θα ποικίλλει ανάλογα με την υπηρεσία ιστού που χρησιμοποιείται, αυτή η διαδικασία αιτήματος και απόκρισης πραγματοποιείται μέσω του API. Οι διεπαφές χρήστη έχουν σχεδιαστεί για ανθρώπινη χρήση, τα API έχουν σχεδιαστεί για χρήση υπολογιστή ή εφαρμογών.

3.1.2 Τύποι API

- **Ιδιωτικά API:** Τα ιδιωτικά API δεν είναι διαθέσιμα για χρήστες εκτός της εταιρείας και αντίθετα αποσκοπούν στη βελτίωση της παραγωγικότητας και της επικοινωνίας μεταξύ διαφορετικών εσωτερικών ομάδων ανάπτυξης.

- **API συνεργατών:** Τα API συνεργατών προωθούνται ανοιχτά αλλά μοιράζονται με επιχειρηματικούς συνεργάτες που έχουν υπογράψει συμφωνία με τον εκδότη.

- **Σύνθετα API:** Τα σύνθετα API συνδυάζουν πολλαπλά API δεδομένων ή υπηρεσιών. Αυτές οι υπηρεσίες επιτρέπουν στους προγραμματιστές να έχουν πρόσβαση σε πολλά τελικά σημεία σε μία μόνο κλήση.

- **Δημόσια API:** Τα ανοιχτά API είναι διεπαφές προγραμματισμού εφαρμογών ανοιχτού κώδικα στις οποίες έχεις πρόσβαση με το πρωτόκολλο HTTP.

3.1.3 Τα σύγχρονα API

Με την πάροδο των ετών, το τι είναι ένα "API" έχει περιγράψει συχνά κάθε είδους γενική διεπαφή συνδεσιμότητας σε μια εφαρμογή. Πιο πρόσφατα, ωστόσο, το σύγχρονο API έχει λάβει ορισμένα χαρακτηριστικά που τα καθιστούν εξαιρετικά πολύτιμα και χρήσιμα:

- Τα σύγχρονα API τηρούν πρότυπα (συνήθως HTTP και REST), που είναι φιλικά προς τους προγραμματιστές, εύκολα προσβάσιμα και ευρέως κατανοητά
- Αντιμετωπίζονται περισσότερο σαν προϊόντα παρά ως κώδικας. Είναι σχεδιασμένα για κατανάλωση για συγκεκριμένα είδη κοινού (π.χ. προγραμματιστές κινητών συσκευών), τεκμηριώνονται και εκδίδονται με τρόπο ώστε οι χρήστες να έχουν ορισμένες προσδοκίες για τη συντήρηση και τον κύκλο ζωής του.
- Επειδή είναι πολύ πιο τυποποιημένα, έχουν πολύ ισχυρότερη πειθαρχία για την ασφάλεια και τη διακυβέρνηση, καθώς και παρακολουθούνται και διαχειρίζονται για την απόδοση και την κλίμακα
- Όπως κάθε άλλο κομμάτι παραγωγικού λογισμικού, το σύγχρονο API έχει τον δικό του κύκλο ζωής ανάπτυξης λογισμικού (SDLC) σχεδιασμού, δοκιμής, κατασκευής, διαχείρισης και εκδόσεων. Επίσης, τα σύγχρονα API είναι καλά τεκμηριωμένα για κατανάλωση και έκδοση.
- Τα API κλιμακώνονται ώστε να μπορούν να εξυπηρετούν μεγάλο αριθμό χρηστών.
- Υπάρχει καλή τεκμηρίωση
- Οι χρήστες ενδέχεται να έχουν μεγάλες προσδοκίες για τη συντήρηση και τον κύκλο ζωής ενός API.

3.1.4 Επίπεδο ασφαλείας API

Οι κλήσεις API συχνά περιλαμβάνουν διαπιστευτήρια εξουσιοδότησης για τη μείωση του κινδύνου επιθέσεων διακομιστή και οι πύλες API μπορούν να περιορίσουν την πρόσβαση για να ελαχιστοποιήσουν τις απειλές ασφαλείας. Επιπλέον, οι κεφαλίδες TTP, τα cookies ή οι παράμετροι συμβολοσειράς ερωτήματος παρέχουν ένα επιπλέον επίπεδο

ασφάλειας για τα δεδομένα κατά την ανταλλαγή. Για παράδειγμα, σκεφτείτε ένα API που παρέχεται από μια υπηρεσία επεξεργασίας πληρωμών. Οι πελάτες μπορούν να εισάγουν τα στοιχεία της κάρτας τους στο προσκήνιο της εφαρμογής του καταστήματος. Ο επεξεργαστής πληρωμών δεν χρειάζεται πρόσβαση στον τραπεζικό λογαριασμό του χρήστη. Το API δημιουργεί ένα μοναδικό διακριτικό για αυτήν τη συναλλαγή και το περιλαμβάνει στην κλήση API προς τον διακομιστή. Αυτό εξασφαλίζει υψηλότερο επίπεδο ασφάλειας έναντι πιθανών απειλών hacking.

3.1.5 Πλατφόρμες επιστημονικού έργου με δυνατότητα API

Παρακάτω θα αναφερθούν κάποιες πλατφόρμες οι οποίες έχουν υλοποιήσει ένα σύστημα API για τη διευκόλυνση ανάκτησης πληροφοριών από τον προγραμματιστή χωρίς την ανάγκη ανάπτυξης web scraping(εκτενέστερη αναφορά θα γίνει στη συνέχεια) εργαλείου για την ανάκτηση πληροφοριών.

- Microsoft Academic
- Semantic Scholar
- Scopus
- PubMed
- Elsevier

Αυτές είναι κάποιες από τις πολλές πλατφόρμες που υποστηρίζουν επικοινωνία με API. Μέσα στη λίστα βλέπουμε πως δεν βάλαμε μια μεγάλη πλατφόρμα την Google Scholar και αυτό γιατί η Google δεν παρέχει κάποιο public API για την ανάκτηση των πληροφοριών από την πλατφόρμα. Με αφορμή την έλλειψη παροχής API κάνουμε τη συγκεκριμένη πτυχιακή δείχνοντας πως θα πάρουμε κάποια δεδομένα από αυτήν την ιστοσελίδα, αναλυτικότερα θα μιλήσουμε στο επόμενο κεφάλαιο.

(ΤΙΦΤΙΚΙΔΗΣ, 2020)

3.2 Web scraping

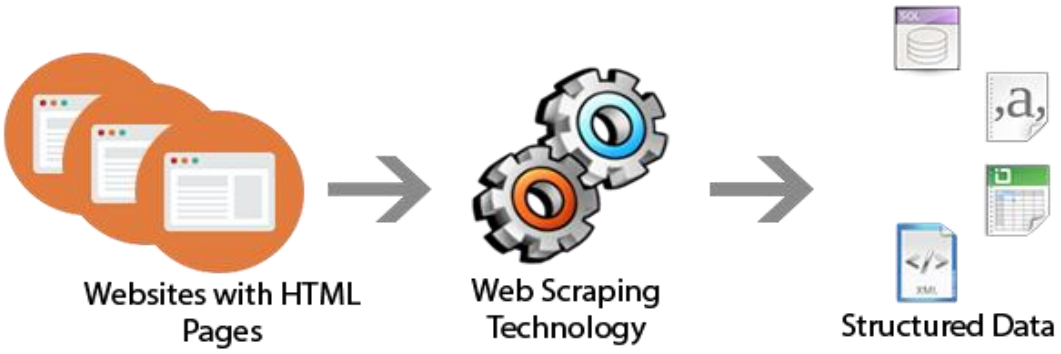
Web scraping ονομάζεται η διαδικασία κατά την οποία γίνεται εξαγωγή περιεχομένου και δεδομένων από έναν ιστότοπο με τη χρήση bots. Η χειροκίνητη αντιγραφή δεδομένων από μια ιστοσελίδα αντιγράφει μόνο εικονοστοιχεία που εμφανίζονται στην οθόνη ενώ στην περίπτωση του wed scraping εξάγετε ο κώδικας

HTML και δεδομένα από την βάση δεδομένων. Ο scraper στη συνέχεια μπορεί να αναπαράγει ολόκληρο το περιεχόμενο της ιστοσελίδας αλλού. Το wed scraping χρησιμοποιείται από πολλές ψηφιακές επιχειρήσεις που βασίζονται στη συλλογή δεδομένων.

Οι νόμιμες περιπτώσεις χρήσης περιλαμβάνουν:

- Bots μηχανών αναζήτησης που ανιχνεύουν έναν ιστότοπο, αναλύουν το περιεχόμενό του και στη συνέχεια τον κατατάσσουν.
- Ιστότοποι σύγκρισης τιμών που αναπτύσσουν bots για αυτόματη ανάκτηση τιμών και περιγραφών προϊόντων για ιστότοπους αντίπαλων επιχειρήσεων.
- Εταιρείες έρευνας αγοράς που χρησιμοποιούν scraping για να αντλήσουν δεδομένα από φόρουμ και κοινωνικά μέσα (π.χ. ανάλυση συναισθημάτων).

Το web scraping χρησιμοποιείται επίσης για παράνομους σκοπούς, συμπεριλαμβανομένης της μείωσης των τιμών και της κλοπής περιεχομένου που προστατεύεται από πνευματικά δικαιώματα. Μια διαδικτυακή οντότητα που στοχεύει ένας scraper μπορεί να υποστεί σοβαρές οικονομικές απώλειες, ειδικά αν πρόκειται για μια επιχείρηση που βασίζεται έντονα σε ανταγωνιστικά μοντέλα τιμολόγησης ή σε διανομές περιεχομένου.



Εικόνα 3.2.1 Web Parsing Nov 25, 2015

3.2.1 Εργαλεία scaping και bots

Τα εργαλεία scraping είναι λογισμικό (bots) προγραμματισμένο να κοσκινίζει βάσεις δεδομένων και να εξάγει πληροφορίες. Χρησιμοποιείται μια ποικιλία τύπων bot, πολλά από τα οποία είναι πλήρως προσαρμόσιμα σε:

- Αναγνωρίζει μοναδικές δομές ιστότοπου HTML
- Εξαγωγή και μετατροπή περιεχομένου
- Αποθήκευση scraped δεδομένων
- Εξαγωγή δεδομένων από API

Δεδομένου ότι όλα τα scraping bots έχουν τον ίδιο σκοπό, την πρόσβαση στα δεδομένα του ιστότοπου, μπορεί να είναι δύσκολο να γίνει διάκριση μεταξύ νόμιμων και κακόβουλων bots. Οι πόροι που απαιτούνται για την εκτέλεση τέτοιων bots είναι τόσο σημαντικοί που οι νόμιμοι χειριστές bot scraping επενδύουν σε μεγάλο βαθμό σε servers για την επεξεργασία του τεράστιου όγκου δεδομένων που εξάγονται.

Ένας δράστης που δεν διαθέτει τέτοιο προϋπολογισμό, καταφεύγει συχνά στη χρήση ενός botnet, γεωγραφικά διασκορπισμένους υπολογιστές, μολυσμένους με το ίδιο κακόβουλο λογισμικό και ελεγχόμενους από μια κεντρική τοποθεσία. Οι μεμονωμένοι κάτοχοι υπολογιστών botnet δεν γνωρίζουν τη συμμετοχή τους. Η συνδυασμένη ισχύς των μολυσμένων συστημάτων επιτρέπει την απόσυρση μεγάλης κλίμακας πολλών διαφορετικών ιστότοπων από τον δράστη.

3.2.2 Κακόβουλα παραδείγματα web scraping

Το web scraping θεωρείται κακόβουλο όταν τα δεδομένα εξάγονται χωρίς την άδεια των κατόχων του ιστότοπου. Οι δύο πιο συνηθισμένες περιπτώσεις χρήσης είναι το price scraping και η κλοπή περιεχομένου.

Σύμφωνα με μια πρόσφατη δίκη[8] που έφτασε μέχρι το εφετείο των ΗΠΑ, σε μια μακροχρόνια δικαστική διαμάχη που άσκησε το LinkedIn με στόχο να σταματήσει μια αντίπαλη εταιρεία από το να αφαιρεί προσωπικές πληροφορίες από τα δημόσια προφίλ των χρηστών, η απόφαση από το Ανώτατο Δικαστήριο ήταν ότι όταν οι πύλες ενός υπολογιστή ή ενός ιστότοπου είναι ανοιχτές, και επομένως οι πληροφορίες είναι δημόσια προσβάσιμες , δεν απαιτείται εξουσιοδότηση. Αυτό κάνει το web scrapping ‘’νόμιμο’’ στις ΗΠΑ, χωρίς βέβαια να υπάρχει ακόμη κάποια νόμος που να αναφέρεται σε αυτό.

[8] https://tcrn.ch/3NUEU2T

- **price scraping**

Όταν κλέβουν τιμές, οι εισβολείς συνήθως χρησιμοποιούν botnet για να εκκινήσουν scraper bots και να ελέγξουν τις ανταγωνιστικές βάσεις δεδομένων των επιχειρήσεων. Ο στόχος είναι η πρόσβαση στις πληροφορίες τιμολόγησης, η μείωση των ανταγωνιστών και η αύξηση των πωλήσεων.

Οι επιθέσεις συμβαίνουν συχνά σε βιομηχανίες όπου τα προϊόντα είναι εύκολο να συγκριθούν και η τιμή παίζει σημαντικό ρόλο στις αποφάσεις της αγοράς. Τα θύματα της απόσυρσης των τιμών μπορεί να περιλαμβάνουν ταξιδιωτικά γραφεία, πωλητές εισιτηρίων και ηλεκτρονικούς πωλητές ηλεκτρονικών ειδών. Για να αποκτήσει πλεονέκτημα, ένας προμηθευτής μπορεί να χρησιμοποιήσει ένα bot για να ξύνει συνεχώς τους ιστότοπους των ανταγωνιστών του και να ενημερώνει αμέσως τις δικές του τιμές ανάλογα.

- **Κλοπή περιεχομένου**

Η κλοπή περιεχομένου περιλαμβάνει περιεχόμενο μεγάλης κλίμακας από έναν ιστότοπο. Οι συνήθεις στόχοι περιέχουν διαδικτυακούς καταλόγους προϊόντων και ιστότοπους που στηρίζονται σε ψηφιακό περιεχόμενο για την προώθηση των επιχειρήσεων. Σε αυτές τις επιχειρήσεις, μια τέτοια επίθεση μπορεί να είναι καταστροφική.

3.2.3 Παράκαμψη των anti-scraping τεχνικών

Ορισμένοι ιστότοποι προστατεύονται με διάφορες τεχνικές από το Web Scraping με σκοπό να προστατέψουνε το περιεχόμενό τους. Για το λόγο αυτό θα κάνουμε αναφορά στις βασικότερες anti-scraping τεχνικές με σκοπό να παρακάμψουμε την οποιαδήποτε προστασία διαθέτει ένας ιστότοπος.

Οι τέσσερις τεχνικές είναι:

1. IP detection
2. IP rate limiting
3. Browser detection
4. Tracking user behavior

IP detection

Οι λόγοι οι οποίοι μπορεί ένας ιστότοπος να χρησιμοποιεί IP Detection είναι:

- Περιεχόμενο το οποίο απευθύνεται σε συγκεκριμένες χώρες.
- Περιεχόμενο το οποίο προστατεύεται από μη ανθρώπινη κίνηση, για παράδειγμα από ένα script το οποίο κάνει αυτόματα αιτήματα σε έναν ιστότοπο. Αυτή η τεχνική μπορεί πολύ εύκολα να παρακαμφθεί από έναν proxy server.

Proxy server είναι μια υπηρεσία η οποία της ζητάμε να επισκεφθεί έναν ιστότοπο και να μας επιστρέψει το αποτέλεσμα μέσου αυτού, με μια λέξη θα λέγαμε πως είναι ένας διαμεσολαβητής.

IP rate limiting

Η δεύτερη πιο διαδεδομένη τεχνική είναι η IP Rate Limiting η οποία περιορίζει την πρόσβασή βάση του πόσα αιτήματα έχουν υποβληθεί από μια IP σε ένα συγκεκριμένο χρονικό διάστημα.

Αυτός ο τύπος προστασίας είναι είτε χειροκίνητος δηλαδή ένας υπάλληλος βλέπει αυξημένη κίνηση στα αρχεία καταγραφής και μπλοκάρει την πρόσβαση ή γίνεται αυτοματοποιημένα.

Παραδείγματος χάριν, ένας ιστότοπος όταν έχει ένα όριο 200 αιτήματα και εμείς κάνουμε 201 πιθανών να δούμε Captcha αντί το αποτέλεσμα μιας αίτησης.

Τέτοια προστασία μπορεί να είναι είτε προσωρινή είτε μόνιμη όταν δεν γίνεται από φυσικό άνθρωπο.

Οι τεχνικές παράκαμψης στο συγκεκριμένο μέτρο προστασίας είναι δύο:

1. Εισαγωγή καθυστέρησης ανάμεσα στα αιτήματα.
2. Χρησιμοποίηση ενός proxy server με IP Rotation με σκοπό την αυτόματη εναλλαγή των IP ύστερα από συγκεκριμένο αριθμό αιτημάτων.

User agents

Η τρίτη προστασία που μπορεί να χρησιμοποιήσει είναι η ανίχνευση κεφαλίδων HTTP User-Agent με σκοπό τον αποκλεισμό της πρόσβασης βάση συγκεκριμένων συσκευών. Ωστόσο, εάν ένα λογισμικό εντοπίσει μια κακόβουλη συσκευή, κατευθείαν θα τη μπλοκάρει.

Για την παράκαμψη αυτής της προστασίας πρέπει :

- Κάθε αίτημα να γίνεται με μια τυχαία κεφαλίδα HTTP User-Agent.
- Αφαίρεση των Cookies πριν γίνει η επίσκεψη στην ιστοσελίδα.

Ο προγραμματισμός και η επίσκεψη της σελίδας μέσω του puppeteer έρχεται να μας δώσει λύση στο συγκεκριμένο τρόπο προστασίας.

Tracking user behavior

Η τελευταία προστασία στην οποία θα κάνουμε αναφορά είναι η παρακολούθηση της συμπεριφοράς του χρήστη σε έναν ιστότοπο, με σκοπό να ανιχνεύσει οτιδήποτε δεν γίνεται από τον άνθρωπο. Παραδείγματος χάρη το κλικ σε έναν σύνδεσμο χωρίς να μετακινηθεί ο κέρσορας.

Αυτός ο τύπος προστασίας συνήθως εφαρμόζεται με IP rate limiting και με τα αποτυπώματα που έχει αφήσει ένας browser (cookies), είναι πιο περίπλοκος από τις προηγούμενες τεχνικές προστασίας και εντοπίζεται βάση συγκεκριμένων μοτίβων.

(ΤΙΦΤΙΚΙΔΗΣ, 2020)

4 Python

4.1 Εισαγωγή στην Python

Η Python είναι μια δυναμική γλώσσα προγραμματισμού υψηλού επιπέδου. Ο κύριος στόχος της είναι να βελτιώσει την παραγωγικότητα των προγραμματιστών και την αναγνωσιμότητα του κώδικα. Η φιλοσοφία του επικεντρώνεται στην εύκολη κατανόηση κώδικα, ενώ η σύνταξή της επιτρέπει στους χρήστες να γράφουν λιγότερες γραμμές κώδικα για συγκεκριμένες λειτουργίες από άλλες δημοφιλείς γλώσσες όπως η Java και η C++. Διαθέτει ενσωματωμένες δομές δεδομένων όπως λίστες και πίνακες

κατακερματισμού που ονομάζονται λεξικά (dictionaries). Δανείζεται στοιχεία από τον προστακτικό, αντικειμενοστραφή αλλά και συναρτησιακό προγραμματισμό, χωρίς να επιβάλλει κάποιον ιδιαίτερο τρόπο προγραμματισμού. Χρησιμοποιεί δυναμικούς τύπους, δηλαδή το μεγαλύτερο μέρος του ελέγχου τύπων της εκτελείται κατά το χρόνο εκτέλεσης αντί για το χρόνο μεταγλώττισης. Τα χαρακτηριστικά της τη καθιστούν ιδανική για ποικίλες εφαρμογές όπου η απόδοση δεν είναι σημαντικός παράγοντας.

Η Python είναι εύκολη στην εκμάθηση της ενώ είναι μια ισχυρή γλώσσα προγραμματισμού. Οι διερμηνευτές της μπορούν να εγκατασταθούν σε μεγάλο αριθμό λειτουργικών συστημάτων, επιτρέποντας στην Python να εκτελείτε σε μια ποικιλία συστημάτων. Η ευχάριστη σύνταξη και οι δυναμικοί τύποι της, καθώς και η λειτουργία της ως διερμηνευόμενης (αντί μεταγλωττιζόμενης) γλώσσας, την καθιστούν ιδανική γλώσσα για την ταχεία ανάπτυξη εφαρμογών σε πολλούς τομείς και πλατφόρμες.

4.2 Ιστορικά στοιχεία

Δημιουργήθηκε από τον Ολλανδό Guido van Rossum και κυκλοφόρησε το 1990. Η Python ήταν αρχικά γλώσσα σεναρίων που χρησιμοποιήθηκε στο λειτουργικό σύστημα Amoeba. Η Python 2.0 κυκλοφόρησε στο κοινό τον Οκτώβριο του 2000, παρέχοντας σημαντικές δυνατότητες, όπως έναν συλλέκτη σκουπιδιών και υποστήριξη για Unicode. Τον Δεκέμβριο του 2008 κυκλοφόρησε η Python 3.0, η οποία αποτέλεσε τη βάση για αυτή την εργασία.

4.3 Δομή

Η Python έχει σχεδιαστεί για να είναι μια εύκολα κατανοητή γλώσσα. Σε πολλές περιπτώσεις, χρησιμοποιεί γνωστές λέξεις (στα αγγλικά) για τη λειτουργία της. Για παράδειγμα, η εντολή ”Εκτύπωσε” χρησιμοποιεί τη δεσμευμένη λέξη "print". Επίσης, σε αντίθεση με τις περισσότερες γλώσσες προγραμματισμού, η Python χρησιμοποιεί κενά για να διαχωρίσει τη συντακτική δομή ενός προγράμματος, αντί για τις κοινες αγκύλες. Επιπλέον, εκτός από μερικές γνωστές δεσμευμένες λέξεις που είναι κοινές μεταξύ γλωσσών, όπως η λέξη "if", η Python χρησιμοποιεί ορισμένες λέξεις που είναι μοναδικές για την γλώσσα, μερικές από τις οποίες είναι οι ακόλουθες:

- **elif** : Δηλώνεται η εναλλαγή σε βρόχο η οποία ξεκινάει με την εντολή if. Με τον ίδιο τρόπο λειτουργεί το else if των java/c++.
- **def**: Με τη λέξη def δηλώνουμε τις συναρτήσεις. Αντίθετα με άλλες γλώσσες προγραμματισμού, η Python δεν απαιτεί συγκεκριμένο τύπο επιστροφής για συναρτήσεις (π.χ int, string ή και void). Εάν ο χρήστης θέλει να επιστρέψει μια τιμή, μπορεί εύκολα να τη δηλώσει στο τέλος της συνάρτησης με την εντολή return.
- **pass**: Άλλη μια ιδιομορφία της Python είναι η χρησιμοποίηση της εντολής pass στη περίπτωση που ο χρήστης πρέπει να συμπληρώσει κάποιες γραμμές με κώδικα(π.χ σε μια if-elif) αλλά θέλει να τις γράψει αργότερα. Έτσι ο μεταγλωττιστής δεν εμφανίζει σφάλματα αν οι γραμμές μείνουν κενές.
- **with**: Το χρησιμοποιούμε σε περιπτώσεις που επεξεργαζόμαστε μη διαχειρίσιμους πόρους, όπως π.χ την επεξεργασία ενός αρχείου κειμένου. Επομένως διευκολύνει την επεξεργασία τέτοιων αρχείων, καθότι αναλαμβάνει να τα κλείσει αυτόματα, μετά το τέλος της επεξεργασίας τους.
- **yield**: Χρησιμοποιείται όταν μια συνάρτηση πρέπει να επιστρέψει μια ακολουθία τιμών, και παράλληλα να ελέγξει όλες τις τιμές(iterate). Το θετικό της yield σε αντίθεση με άλλες μεθόδους, είναι ότι ελαττώνει έτσι τη χρήση της μνήμης, καθώς δεν φορτώνει όλες τις τιμές της ακολουθίας.
- **assert**: Η χρήση του γίνεται για να ελέγξει μία συνθήκη (condition) και στις καταστάσεις που δεν είναι αληθείς αναφέρει σφάλμα (error). Το συναντάμε κατά το debugging ενός προγράμματος.
- **or**: Είναι το αντίστοιχο του || στην Java. Στην ουσία χρησιμοποιείται όπως ακριβώς στον φυσικό κόσμο.
- **not**: Αντίστοιχο του !=.
- **and**: Αντίστοιχο του &&.

Παρατηρείται ότι οι τρεις τελευταίες λέξεις έχουν ακριβώς την ίδια σημασία με την προφορική γλώσσα, γεγονός που καθιστά τον κώδικα που είναι γραμμένος σε Python πολύ εύκολο να κατανοηθεί από άτομα χωρίς αρκετή εμπειρία προγραμματισμού.

(ΜΟΣΧΟΣ, 2015)

4.4 Βιβλιοθήκες

Στην πληροφορική βιβλιοθήκη (library) ονομάζουμε μια συλλογή από υποπρογράμματα που βοηθούν στην ανάπτυξη λογισμικού. Οι βιβλιοθήκες περιέχουν βοηθητικό κώδικα και δεδομένα για την εξυπηρέτηση προγραμμάτων.σ Αυτό επιτρέπει τον διαμοιρασμό και τη χρήση του κώδικα και των δεδομένων με αρθρωτό τρόπο. Η έννοια των βιβλιοθηκών αποτελεί αναπόσπαστο μέρος του δομημένου προγραμματισμού και έχει εξελιχθεί παράλληλα με αυτόν. Ορισμένα εκτελέσιμα αρχεία (executables) είναι προγράμματα και βιβλιοθήκες ταυτόχρονα, αλλά οι περισσότερες βιβλιοθήκες δεν είναι εκτελέσιμες. Τα εκτελέσιμα αρχεία και οι βιβλιοθήκες παραπέμπουν το ένα στον κώδικα και τα δεδομένα του άλλου μέσω της σύνδεσης και την πραγματοποιεί ο συνδέτης (linker). Τα σύγχρονα λειτουργικά συστήματα παρέχουν βιβλιοθήκες που υλοποιούν τις περισσότερες υπηρεσίες συστήματος. Επομένως, το μεγαλύτερο μέρος του κώδικα που χρησιμοποιείται από τις σύγχρονες εφαρμογές παρέχεται από αυτές τις βιβλιοθήκες και δεν χρειάζεται να γράφεται από την αρχή για κάθε νέο πρόγραμμα.

(wikipedia, 2021)

4.4.1 Βιβλιοθήκες για την ανάλυση δεδομένων

- **Pandas**

Η Pandas είναι μια πολύ χρήσιμη και ευρέως χρησιμοποιούμενη βιβλιοθήκη που παρέχει δομές δεδομένων υψηλής απόδοσης για επεξεργασία, καθαρισμό και προετοιμασία δεδομένων για εισαγωγή σε διαδικασίες μηχανικής μάθησης. Η παροχή «καθαρών» δεδομένων στη σωστή μορφή είναι κρίσιμη για την πραγματοποίηση σωστών προβλέψεων στη μηχανική εκμάθηση και σίγουρα δεν πρέπει να αγνοηθεί.

Το Pandas μας επιτρέπει να μορφοποιούμε αποτελεσματικά τα δεδομένα μας όπως τα χρειαζόμαστε, αποφεύγοντας τις παγίδες της συμπερίληψης αμφισβητήσιμων ή ελλιπών δεδομένων στην ανάλυσή μας.

- **scikit-learn**

Μόλις τα Pandas καθαρίσουν σωστά τα δεδομένα, περνάμε στη φάση μηχανικής εκμάθησης, η οποία εκτελείται χρησιμοποιώντας τη βιβλιοθήκη scikit-learn. Ένα από τα πράγματα που κάνει το scikit-learn είναι ότι παρέχει μια μεγάλη συλλογή αλγορίθμων από διαφορετικούς τύπους μηχανικής μάθησης κάτω από ένα κοινό API.

Αυτό σημαίνει ότι πολλές τεχνικές μπορούν να χρησιμοποιούν την ίδια σύνταξη, επομένως μπορούμε εύκολα να εναλλάσσουμε τεχνικές ανάλογα με τον τύπο των δεδομένων που αναλύουμε και το πρόβλημα που λύνουμε. Το Scikit-learn παρέχει επίσης εργαλεία για την επικύρωση των αποτελεσμάτων και τη διασφάλιση της βέλτιστης επιλογής μοντέλων. Εάν χρησιμοποιούσαμε άλλη γλώσσα προγραμματισμού, θα χρειαζόμασταν πολλές βιβλιοθήκες, συχνά με διαφορετικές συνθέσεις.

- **NumPy**

Μάλλον η πιο σημαντική βιβλιοθήκη στο επιστημονικό υπολογιστικό οικοσύστημά της Python είναι η NumPy, μια βιβλιοθήκη αριθμητικών υπολογιστών υψηλής απόδοσης. Το NumPy παρέχει τη βάση για πολλές από τις επιστημονικές βιβλιοθήκες υπολογιστών της Python, συμπεριλαμβανομένων των Pandas και της scikit-learn. Εν ολίγοις, το NumPy είναι η πρώτη ύλη που δίνει στις βιβλιοθήκες εξαιρετική απόδοση. Χωρίς NumPy, πολλοί από τους υπολογισμούς στην επιστήμη δεδομένων θα ήταν αδύνατο να εφαρμοστούν στην Python.

(Pro, 2019)

5 Γραφικό περιβάλλον χρήστη (GUI)

Το Graphical User Interface (GUI) ή Γραφικό Περιβάλλον Χρήστη όπως ονομάζεται στα ελληνικά, περιγράφει τη διεπαφή μιας εφαρμογής χρησιμοποιώντας γραφικά, όπως εικονίδια, μενού, πλαίσια ελέγχου και άλλα που διευκολύνουν τον χρήστη να αλληλοεπιδράσει με την εφαρμογή. Το γραφικό περιβάλλον είναι μια **εξέλιξη** των παλαιότερων μορφών αλληλεπίδρασης σε έναν υπολογιστή οι οποίες βασίζονταν σε ένα περιβάλλον κειμένου με χρήση εντολών (όπως το DOS).

Ένα παράδειγμα γραφικής διεπαφής χρήστη είναι το λειτουργικό σύστημα Windows. Οι χρήστες αλληλεπιδρούν με την εφαρμογή μέσω των επιλογών που εμφανίζονται στο παράθυρο. Τα γραφικά περιβάλλοντα χρησιμοποιούνται συχνά σε συσκευές όπως κινητά τηλέφωνα, τηλεοράσεις, ηλεκτρονικοί υπολογιστές και άλλα

5.1 Python TKinter

Η Python έχει πολλά πλαίσια διεπαφών χρήστη (GUI), αλλά το TKinter είναι το μόνο πλαίσιο που είναι ενσωματωμένο στην τυπική βιβλιοθήκη της Python. Το Tkinter έχει πολλά πλεονεκτήματα. Είναι cross-platform, επομένως ο ίδιος κώδικας λειτουργεί σε Windows, macOS και Linux. Τα οπτικά στοιχεία αποδίδονται χρησιμοποιώντας στοιχεία του εγγενούς λειτουργικού συστήματος, επομένως οι εφαρμογές που δημιουργούνται με το Tkinter φαίνεται να ανήκουν στην πλατφόρμα στην οποία εκτελούνται.

Εάν θέλετε μια κομψή, μοντέρνα διεπαφή, τότε το Tkinter μπορεί να μην είναι αυτό που ψάχνετε. Ωστόσο, σε σύγκριση με άλλα πλαίσια, το Tkinter είναι ελαφρύ και σχετικά εύκολο στη χρήση. Αυτό το καθιστά μια συναρπαστική επιλογή για τη δημιουργία εφαρμογών GUI στην Python, ειδικά για εφαρμογές που δεν απαιτούν μοντέρνα στιλπνότητα και των οποίων η προτεραιότητα είναι η γρήγορη δημιουργία λειτουργικών και πολλαπλών πλατφόρμων εφαρμογών.Widgets ονομάζονται τα στοιχεία μέσω των οποίων ο χρήστης αλληλοεπιδρά με το πρόγραμμα. Κάθε widget ορίζεται από μια κλάση. Τα βασικά widget του TKinter είναι τα εξης:

- Label
- Button

- Entry
- Text
- Frame

Το Label είναι ένα γραφικό στοιχείο που χρησιμοποιείται για την εμφάνιση κειμένου στην οθόνη .

Το Button είναι ένα κουμπί που μπορεί να περιέχει κείμενο και να εκτελέσει μια ενέργεια όταν πατηθεί.

Το Entry είναι ένα widget εισαγωγής κειμένου που επιτρέπει μόνο μια γραμμή κειμένου.

Το Text είναι ένα widget εισαγωγής κειμένου που επιτρέπει την εισαγωγή πολλών γραμμών κειμένου.

Το Frame είναι μια ορθογώνια περιοχή που χρησιμοποιείται για την ομαδοποίηση σχετικών στοιχείων .

5.2 wxPython

Το wxPython είναι ένα cross-platform GUI (συχνά αναφέρεται ως " εργαλειοθήκη ") για τη γλώσσα προγραμματισμού Python . Επιτρέπει στους προγραμματιστές Python να δημιουργούν προγράμματα με ένα ισχυρό, εξαιρετικά λειτουργικό γραφικό περιβάλλον χρήστη, απλά και εύκολα. Υλοποιείται ως ένα σύνολο λειτουργικών μονάδων επέκτασης Python που τυλίγουν τα στοιχεία GUI της δημοφιλούς βιβλιοθήκης πολλαπλών πλατφόρμων wxWidgets , η οποία είναι γραμμένη σε C++.

Όπως η Python και το wxWidgets, έτσι και το wxPython είναι Ανοιχτού Κώδικα , που σημαίνει ότι είναι δωρεάν για οποιονδήποτε επιθυμεί να το χρησιμοποιήσει και ο πηγαίος κώδικας είναι διαθέσιμος για οποιονδήποτε να τον δει και να τον τροποποιήσει. Επιπλέον, οποιοσδήποτε έχει την δυνατότητα να συνεισφέρει διορθώσεις ή βελτιώσεις στο έργο.

Το wxPython είναι μια εργαλειοθήκη πολλαπλών πλατφορμών . Αυτό σημαίνει ότι το ίδιο πρόγραμμα θα εκτελείται σε πολλές πλατφόρμες χωρίς τροποποίηση. Επί του παρόντος, οι υποστηριζόμενες πλατφόρμες είναι τα Microsoft Windows, Mac OS X και macOS και Linux ή άλλα συστήματα τύπου unix με βιβλιοθήκες GTK2 ή GTK3. Στις περισσότερες

περιπτώσεις, τα εγγενή γραφικά στοιχεία χρησιμοποιούνται σε κάθε πλατφόρμα για να παρέχουν 100% εγγενή εμφάνιση και αίσθηση για την εφαρμογή.

Δεδομένου ότι η γλώσσα προγραμματισμού είναι η Python, τα προγράμματα wxPython είναι απλά, εύκολα στη γραφή και κατανοητά .

(Team, 2020)

5.3 PyQt5

Το **πακέτο PyQt** είναι χτισμένο γύρω από το **πλαίσιο Qt, το** οποίο είναι ένα πλαίσιο πολλαπλών πλατφορμών που χρησιμοποιείται για τη δημιουργία μιας πληθώρας εφαρμογών για διάφορες πλατφόρμες. Το πακέτο PyQt5 περιλαμβάνει ένα λεπτομερές σύνολο συνδέσεων για την Python με βάση την **τελευταία έκδοση v5** του πλαισίου εφαρμογής Qt.

Παρόμοια με το πλαίσιο Qt5, το PyQt5 είναι επίσης **πλήρως cross-platform** . Αξιοποιώντας τη δύναμη του PyQt5, οι προγραμματιστές μπορούν να δημιουργήσουν εφαρμογές για πλατφόρμες όπως Windows, Mac, Linux, iOS, Android και άλλα.

Όσον αφορά τη δημιουργία GUI, το οπλοστάσιο PyQt5 προσφέρει το εντυπωσιακό QtGui και τη μονάδα QtDesigner, τα οποία παρέχουν πολυάριθμα οπτικά στοιχεία που ο προγραμματιστής μπορεί να εφαρμόσει με ένα απλό drag and drop. Φυσικά, υπάρχει και η δυνατότητα δημιουργίας αυτών των στοιχείων μέσω κώδικα, επιτρέποντάς σας να δημιουργείτε εφαρμογές τόσο μικρής όσο και μεγάλης κλίμακας με ευκολία. Η αρθρωτότητα της Python φτάνει στο PyQt5 με τη μορφή επεκτάσεων, δίνοντάς σας πολύ περισσότερες δυνατότητες από την απλή δημιουργία GUI.

(Riv)

6 Βάσεις δεδομένων

6.1 SQlite

Το SQLite είναι ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων γραμμένο στη γλώσσα προγραμματισμού C και σε αντίθεση με πολλά συστήματα διαχείρισης

βάσεων δεδομένων, δεν εκτελείται στη λογική πελάτη-διακομιστή, αλλά κατά κάποιο τρόπο ενσωματώνεται στο τελικό πρόγραμμα/εφαρμογή που το χρησιμοποιεί.

Είναι μια από τις πιο δημοφιλείς τοπικά αποθηκευμένες βάσεις δεδομένων καθώς χρησιμοποιείται από πολλά λογισμικά όπως εφαρμογές ή προγράμματα περιήγησης Ιστού. Υποστηρίζει τις πιο κοινές γλώσσες προγραμματισμού, συνολικά περισσότερες από 30 γλώσσες. Είναι διαθέσιμο για διάφορα λειτουργικά συστήματα. Τα πιο αξιοσημείωτα από αυτά είναι τα Windows, Android, Blackberry OS, Symbian OS και αρκετές εκδόσεις του UNIX.

6.1.1 Χαρακτηριστικά

Το SQLite έχει τις περισσότερες δυνατότητες και λειτουργίες της SQL. Τα βασικά πλεονεκτήματα της SQL είναι διαθέσιμα και εδώ, όπως τα σύνθετα ερωτήματα αναζήτησης, αλλά δεν έχει τη δυνατότητα αλλαγής μορφών πινάκων με αλλαγή ή απόθεση στηλών. Επίσης, το σύστημα τύπου που χρησιμοποιεί είναι λίγο ασυνήθιστο. Αντί να εισάγει τύπους σε ολόκληρη τη στήλη, παραθέτει κάθε τιμή ξεχωριστά, δηλαδή χρησιμοποιεί ένα δυναμικό σύστημα τύπων. Αυτό το σύστημα τύπων θεωρείται επίσης αδύναμο επειδή, για παράδειγμα, μια ακέραια τιμή μπορεί να εκχωρηθεί σε ένα πεδίο που δέχεται συμβολοσειρές. Το SQLite επιτρέπει την ταυτόχρονη πρόσβαση από διαφορετικά νήματα ή διεργασίες. Πιο συγκεκριμένα, η πρόσβαση ανάγνωσης από πολλαπλά νήματα είναι δυνατή ταυτόχρονα και η πρόσβαση εγγραφής επιτρέπεται μόνο όταν δεν υπάρχει άλλη ενεργή πρόσβαση εκείνη τη στιγμή. Διαφορετικά, δεν χορηγείται άδεια πρόσβασης και η εγγραφή αποτυγχάνει.

6.2 MySQL

Η MySQL είναι ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDBMS) που αλληλεπιδρά με τη SQL (Structured Query Language), ένα εργαλείο για τον χειρισμό την πρόσβαση σε βάσεις δεδομένων. Είναι ένα πολύ γρήγορο και ισχυρό σύστημα διαχείρισης βάσεων δεδομένων. Οι βάσεις δεδομένων επιτρέπουν την αποθήκευση, αναζήτηση, ταξινόμηση και ανάκτηση δεδομένων. Ο διακομιστής MySQL ελέγχει την πρόσβαση στα δεδομένα, έτσι ώστε πολλοί χρήστες να μπορούν να λειτουργούν ταυτόχρονα, παρέχει γρήγορη πρόσβαση και διασφαλίζει ότι μόνο πιστοποιημένοι χρήστες

μπορούν να έχουν πρόσβαση σε αυτά. Επομένως, η MySQL είναι ένας διακομιστής πολλαπλών νημάτων και πολλών χρηστών. Ο διακομιστής MySQL είναι ο διαχειριστής συστήματος της βάσης δεδομένων. Επεξεργάζεται όλες τις εντολές της βάσης. Για να λάβει εντολές, ο διακομιστής MySQL πρέπει να εκτελείται και να περιμένει αιτήματα

Αυτός είναι ο λόγος για τον οποίο ο διακομιστής MySQL είναι συνήθως ρυθμισμένος να ξεκινά όταν ξεκινά ο κεντρικός υπολογιστής. Όλη η αλληλεπίδραση με τη βάση δεδομένων γίνεται με την αποστολή μηνυμάτων στον διακομιστή MySQL. Επικοινωνεί χρησιμοποιώντας SQL (Structured Query Language), μια γλώσσα που μπορεί να κατανοήσει πολλά συστήματα διαχείρισης βάσεων δεδομένων (RDBMS). Η PHP δεν καταλαβαίνει την SQL γιατί δεν χρειάζεται να την κατανοήσει. Η PHP δημιουργεί απλώς μια σύνδεση με τον διακομιστή MySQL για την αποστολή μηνυμάτων SQL. Ο διακομιστής MySQL ερμηνεύει τα μηνύματα SQL και ακολουθεί τις οδηγίες. Στη συνέχεια στέλνει μια απάντηση με ένα μήνυμα που εξηγεί την κατάστασή του και τι έκανε ή αναφέρει ένα σφάλμα σε περίπτωση που δεν μπορούσε να καταλάβει και να ακολουθήσει τις οδηγίες.

(ΖΑΧΑΡΙΑΣ, 2020)

7 Υλοποίηση εφαρμογής

7.1 Εισαγωγή

Η ιδέα για την δημιουργία της εφαρμογής προέκυψε από την εσωτερική αξιολόγηση του τμήματος. Η αξιολόγηση είναι μια έκθεση που συγγράφουν καθηγητές για να αναδείξουν τα θετικά και τα αρνητικά του τμήματος σε όλους τους τομείς. Με τη βοήθεια του φοιτητικού συλλόγου και του διοικητικού προσωπικού, αναλύουν όλες τις πλευρές του τμήματος και προσπαθούν να το αλλάξουν προς το καλύτερο. Ένα από τα κριτήρια αξιόλογης είναι οι ερευνητικές υποδομές που παρέχει το τμήμα και εξετάζονται οι

δημοσιεύσεις των μελών του διδακτικού προσωπικού του τμήματος . Έτσι δημιούργησα μια εφαρμογή η οποία μπορεί πολύ ευκολά να δείξει της δημοσιεύσεις όλων των μελών του τμήματος καθώς επίσης και να υπολογίσει κάποια χρήσιμα στατιστικά για την ετίσια αξιολόγηση.

7.2 Εισαγωγή βιβλιοθηκών

Το πρώτο πράγμα που κάνουμε όταν γράφουμε κώδικα για μια εφαρμογή είναι να εισάγουμε τις βιβλιοθήκες που θα χρειαστούμε. Για την εφαρμογή μας διαλέξαμε να χρησιμοποιήσουμε το GUI του Tkinter. Έτσι από την βιβλιοθήκη του Tkinter εισήγαμε τις κλάσεις που χρειαστήκαμε. Στη συνέχεια προσθέσαμε και τις υπόλοιπες βιβλιοθήκες οι οποίες μας βοήθησαν με την επεξεργασία των αρχείων, την βάση δεδομένων και τον υπολογισμό και την προβολή των αποτελεσμάτων όπως επιθυμούσαμε όπως είναι η numpy και η maplotlib. (Ο κώδικας της εφαρμογής υπάρχει αναρτημένος στο GitHub.

(https://github.com/panoskaf/ptixiaki.git)

```
from tkinter import ttk, Tk, Toplevel , Label , Text , Button, \
    OptionMenu, Frame , Scrollbar, Entry, Listbox
import tkinter as tk
from tkinter.constants import  END, INSERT
from tkinter.filedialog import askopenfilename
import json , sqlite3, csv
import numpy as np
import matplotlib.pyplot as plt
import clipboard
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from datetime import datetime
from collections import Counter
import locale
from PIL import ImageTk, Image
import getpass
```

Εικόνα 7.2.1 Βιβλιοθήκες εφαρμογής

7.3 Βασική διεπαφή εφαρμογής

Το επόμενο μας βήμα είναι η δημιουργία του κυρίου μέρους της εφαρμογής. Σχηματίζουμε το βασικό παράθυρο στο οποίο τοποθετούμε τα υπόλοιπα στοιχεία όπως κουμπιά, πλαίσια και εικόνες.

Το κύριο παράθυρο της εφαρμογής ονομάζεται “root” και έχει διαστάσεις 1280x680 pixels. Το μέγεθος του παράθυρου μπορεί να αλλαξει και μαζί με το παράθυρο θα προσαρμοστούν δυναμικά και όλα τα αντικείμενα της εφαρμογής.

```
#main window
root = Tk()
root.title('Research Activity Application')
root.geometry("1280x680+20+20")
root.iconbitmap('icon.ico')
root.configure(bg="#505251")
```

Εικόνα 7.3.1 Δημιουργία κύριου παράθυρου εφαρμογής

Αφού δημιουργήσαμε το αρχικό μας παράθυρο ξεκινάμε να δημιουργούμε τα στοιχεία που θα μας χρειαστούν. Ξεκινάμε τοποθετώντας μια εικόνα του λογοτύπου του τμήματος στην επάνω αριστερή θέση του παραθύρου. Έπειτα δημιουργούμε ένα πλαίσιο στο οποίο θα τοποθετήσουμε τα κουμπιά για την χρήση της εφαρμογής. Αυτό που έμεινε είναι να προσθέσουμε τα κουμπιά και τις λίστες μας, με τις κατάλληλες ρυθμίσεις σε μέγεθος και χρώμα.

```
#Logo τμήματος
img = ImageTk.PhotoImage(Image.open("photo_1.png"))
panel = Label(root, image = img)
panel.place(relx=.167, rely=.12, anchor="c")

#Πλαισιο κουμπιων
button_frame = Frame(root,bg="#2f929c")
button_frame.place(relx=0.66, rely=0.02, relwidth=0.64, relheight=0.2,
anchor='n')



#λιστα καθηγητων
clicked = tk.StringVar()
clicked.set("Επιλέξτε Καθηγητή")
drop = OptionMenu(button_frame, clicked , *list_of_entries, command=
return_key)
```

```
drop.config(bg="WHITE", fg="BLACK", activebackground='#32667d',
activeforeground='white',relief= "flat")
drop.place(relx=0.01, rely=0.08,relwidth=0.2, relheight=0.3)
drop["menu"].config(activebackground='#32667d')

#λιστα αποτελεσματων
clicked2 = tk.StringVar()
clicked2.set("Αποτελέσματα")
drop2 = OptionMenu(button_frame, clicked2 , *functions, command=results)
drop2.config(bg="WHITE", fg="BLACK", activebackground='#32667d',
activeforeground='white',relief= "flat")
drop2.place(relx=0.45, rely=0.08,relwidth=0.2, relheight=0.3)
drop2["menu"].config(activebackground='#32667d')
```

Εικόνα 7.3.2 Widgets εφαρμογής

Εδώ δημιουργούμε τα κουμπιά και τους δίνουμε τις κατάλληλες τιμές και εντολές που θα ακολουθούν.

```
#Δημιουργια κουμπιων

# κουμπι εμφανισης αρθρων
data_print_button = Button(button_frame, text="Εμφάνιση Δημοσιεύσεων",
font=("Helvetica", 10),bg="WHITE",relief= "flat", command=data_print)
data_print_button.place(relx=0.23, rely=0.08,relwidth=0.2, relheight=0.3)

# κουμπι εισαγωγης δεδομενων
import_jsonfile_button = Button(button_frame, text="Προσθήκη αρχείου JSON",
font=("Helvetica",10 ),relief= "flat", command=import_jsonfile)
import_jsonfile_button.place(relx=0.01, rely=0.6,relwidth=0.2, relheight=0.3)

# κουμπι καθαρισμου πινακα
clear_text_button = Button(button_frame, text="Καθαρισμός",relief= "flat",
command=persons)
clear_text_button.place(relx=0.67, rely=0.08,relwidth=0.2, relheight=0.3)

# κουμπι εισαγωγης καθηγητη
add_person_button = Button(button_frame, text="Προσθηκη νεου καθηγητη",
font=("Helvetica", 10),relief= "flat", command= add_person_creation)
add_person_button.place(relx=0.23, rely=0.6,relwidth=0.2, relheight=0.3)

# κουμπι διαγραφης καθηγητη
del_person_button = Button(button_frame, text="Διαγραφη Καθηγητή",
font=("Helvetica", 10),relief= "flat", command=delete_person_creation)
del_person_button.place(relx=0.45, rely=0.6,relwidth=0.2, relheight=0.3)

# κουμπι πληροφοριων καθηγητη
```

```
scholar_info_button = Button(button_frame, text="Scholar Info",
font=("Helvetica", 10),relief= "flat", command=scholar_info)
scholar_info_button.place(relx=0.67, rely=0.6,relwidth=0.2, relheight=0.3)

# κουμπι εξοδου εφαρμογης
quit_button = Button(button_frame, text=" Έξοδος", font=("Helvetica", 12),
bg="#d13636",relief= "solid", command= root.quit)
quit_button.place(relx=0.89, rely=0.26,relwidth=0.1, relheight=0.5)
```

Εικόνα 7.3.3 Δημιουργία πλήκτρων εφαρμογής

7.4 Πίνακας προβολής δημοσιεύσεων

Το τελευταίο κομμάτι της κύριας διεπαφής της εφαρμογής είναι ο πίνακας στον οποίο θα εμφανίζονται οι δημοσιεύσεις του κάθε καθηγητή. Δημιουργούμε ένα πίνακα Treeview και του δίνουμε το μέγεθος και το χρώμα που επιθυμούμε. Στη συνέχεια δηλώνουμε ποιες θα είναι οι στήλες από τις οποίες θα αποτελείται ο πίνακας και τις φτιάχνουμε μια - μια, με το όνομα και το μέγεθος της. Τέλος, ενεργοποιούμε και την συνάρτηση που θα καλείται όταν ο χρήστης κάνει διπλό κλικ πάνω σε μια δημοσίευση.

```
#Treeview
style = ttk.Style()

style.theme_use('default')

# Treeview Colors
style.configure("Treeview",
    background="#D3D3D3",
    foreground="black",
    fieldbackground="#D3D3D3")

# Change Selected Color
style.map('Treeview',
    background=[('selected', "#32667d")])

# create tree frame
tree_frame = Frame(root)
tree_frame.place(relx=0.5, rely=0.25, relwidth=0.98, relheight=0.73,
anchor='n')
```

```
# Create Vertical Scrollbar
tree_scroll = Scrollbar(tree_frame)
tree_scroll.pack(side=tk.RIGHT, fill=tk.Y)

my_tree = ttk.Treeview(tree_frame, yscrollcommand=tree_scroll.set,
selectmode="extended")
my_tree.place(relheight=1,relwidth=1)

# Configure Scrollbars
tree_scroll.config(command=my_tree.yview)
#hor_scroll.config(command=my_tree.xview)

my_tree['columns'] = ("ID", "TYPE", "TITLE", "AUTHORS", "YEAR", "SOURCE",
"CITES")

# Format Our Columns
my_tree.column("#0", width=0, stretch=tk.NO)
my_tree.column("ID", anchor=tk.W,minwidth=10, width=150,stretch=tk.NO )
my_tree.column("TYPE", anchor=tk.CENTER,minwidth=10, width=130,stretch=tk.NO )
my_tree.column("TITLE", anchor=tk.W,minwidth=10, width=420 )
my_tree.column("AUTHORS", anchor=tk.W,minwidth=10, width=270 )
my_tree.column("YEAR", anchor=tk.CENTER,minwidth=10, width=40,stretch=tk.NO)
my_tree.column("SOURCE", anchor=tk.W ,minwidth=10, width=200)
my_tree.column("CITES", anchor=tk.CENTER,minwidth=10, width=40,stretch=tk.NO)

# Create Headings
my_tree.heading("#0", text="", anchor=tk.W)
my_tree.heading("ID", text="ID", anchor=tk.CENTER)
my_tree.heading("TYPE", text="TYPE", anchor=tk.CENTER)
my_tree.heading("TITLE", text="TITLE", anchor=tk.CENTER)
my_tree.heading("AUTHORS", text="AUTHORS", anchor=tk.CENTER)
my_tree.heading("YEAR", text="YEAR", anchor=tk.CENTER)
my_tree.heading("SOURCE", text="SOURCE", anchor=tk.CENTER)
my_tree.heading("CITES", text="CITES", anchor=tk.CENTER)

my_tree.bind("<Double-1>",on_double_click)
```

Εικόνα 7.4.1 Πίνακας προβολής δημοσιεύσεων

7.5 Βάση δεδομένων

Στην αποθήκευση και την επεξεργασία των δεδομένων χρησιμοποιούμε μια τοπική βάση σε SQLite. Δημιουργούμε ένα αρχείο με όνομα jbase.db και εκεί προσθέτουμε τους 3 πίνακες που θα φυλάξουν τα δεδομένα μας.

Ο πρώτος πίνακας ονομάζεται person και αποτελείται από τα πεδία id και pname. Σε αυτό το πίνακα θα αποθηκεύσουμε τα στοιχεία των καθηγητών, δηλαδή το όνομα και το Google Scholar ID του καθενός. Ο επόμενος πίνακας ονομάζεται papers και θα περιέχει της δημοσιεύεις του κάθε καθηγητή. Ο πίνακας papers αποτελείται από τα παρακάτω πεδία:

- Uidd: το id της κάθε δημοσίευσης όπως αναγράφεται στο google scholar.
- Types: ο τύπος της δημοσίευσης.
- Title: ο τίτλος της δημοσίευσής
- Authors: οι συγγραφείς της δημοσίευσης
- Years: το έτος που αναρτήθηκε η δημοσίευση.
- Source: η πηγή της δημοσίευσης.
- Cites: ο αριθμός αναφορών της δημοσίευσης.

Τελευταίος είναι ο πίνακας ο οποίος θα φιλοξενήσει τα ξένα κλειδιά της βάσης μας. Ονομάζεται person-paper και αποτελείται από 3 πεδία. Το id, το person_id και το paper_id. Σε αυτό το πίνακα δημιουργούμε σχέσεις 1 προς πολλά μεταξύ των πινάκων person-person_paper και papers – person_paper. Έτσι ο πίνακας μας θα περιέχει τα ids των καθηγητών και τα ids των δημοσιεύσεων του κάθε καθηγητή.

```
#database
connection = sqlite3.connect('jbase.db')
c = connection.cursor()

#authors
c.execute("""CREATE TABLE IF NOT EXISTS  person(
      id  text  PRIMARY KEY,
      pname text

)""")


#papers
```

```
c.execute("""CREATE TABLE IF NOT EXISTS  papers(
      uidd  integer  PRIMARY KEY AUTOINCREMENT,
      types text,
      title text,
      authors text,
      years integer,
      source text,
      cites integer

)""")
c.execute("""CREATE TABLE IF NOT EXISTS person_paper(
    id integer PRIMARY KEY AUTOINCREMENT,
    person_id text,
    paper_id text,
    UNIQUE (person_id, paper_id)
    FOREIGN KEY (person_id)
    REFERENCES person (id),
    FOREIGN KEY (paper_id)
    REFERENCES papers(uidd)
)""")

connection.commit()
```

Εικόνα 7.5.1 Βάση δεδομένων εφαρμογής

7.6 Λειτουργείες εφαρμογής

Εδώ θα δούμε τις λειτουργείες της εφαρμογής μας, δηλαδή τις συναρτήσεις που γράψαμε για να κάνουμε την εφαρμογή λειτουργική και να πάρουμε τα αποτελέσματα που θέλουμε.

7.6.1 sort_csv()

Η συνάρτηση sort_csv είναι υπεύθυνη για την ταξινόμηση του csv αρχείου που περιέχει τα ονόματα και τα ids των καθηγητών. Η λειτουργεία της είναι πολύ απλή, με την βοήθεια του module « locale » θέτουμε την γλώσσα ταξινόμησης στα ελληνικά και χρησιμοποιώντας την μέθοδο sort ταξινομούμε το αρχείο μας εύκολα βάση των ελληνικών χαρακτήρων. Το επόμενο βήμα είναι να ξαναπεράσουμε τα ταξινομημένα πλέον δεδομένα στο csv. Τη συγκεκριμένη συνάρτηση την χρησιμοποιούμε μόνο μέσα στην συνάρτηση csv_open που θα αναφέρουμε στη συνέχεια.

```
# ταξινομιση csv αρχειου
def sort_csv(data):
    locale.setlocale(locale.LC_ALL, 'el_GR.UTF-8')
    data.sort(key= lambda x: locale.strxfrm(x[0]))
    print(data)
```

```
    with open('optnames.csv','w',encoding="utf-8 -sig",newline='') as file:
        writer=csv.writer(file)
        for row in data:
            writer.writerow(row)
```

Εικόνα 7.6.1 Συνάρτηση sort_csv()

7.6.2 csv_open()

Μέσω αυτής της συνάρτησης, διαβάζουμε το csv αρχείο και δημιουργούμε δυο λίστες. Η πρώτη λίστα είναι η λίστα Data, στην οποίο τοποθετούμε όλα τα δεδομένα έτσι όπως τα παίρνουμε από το αρχείο. Μόλις δημιουργήσουμε την λίστα Data, χρησιμοποιούμε την συνάρτηση sort_csv, για να ταξινομήσουμε αλφαβητικά την λίστα μας και να έχουμε ένα όμορφο και λειτουργικό αποτέλεσμα. Στην δεύτερη λίστα που ονομάσαμε list_of_entries, τοποθετούμε μόνο την πρώτη στήλη από τα δεδομένα, δηλαδή μόνο τα ονόματα των καθηγητών με ελληνικούς χαρακτήρες. Αυτή τη λίστα θα τη χρησιμοποιήσουμε στην βασική μας διεπαφή, στο κουμπί επιλογής καθηγητή. Με την εκκίνηση του προγράμματος η συγκεκριμένη συνάρτηση καλείται να διαβάσει τα δεδομένα, τα οποία χρειαζόμαστε για την δημιουργία της βάσης και του κουμπιού με τα ονόματα των καθηγητών. Επίσης ξανακαλείται σε συναρτήσεις που επεξεργάζονται την λίστα με τους καθηγητές, όπως η διαγραφή και η προσθήκη καθηγητών.

```
#csv με ονοματα και id καθηγητών
def csv_open():
    File = open('optnames.csv',encoding="utf-8 -sig")
    Reader = csv.reader(File,delimiter=',')
    Data = list(Reader)
    File.close()

    sort_csv(Data)

    list_of_entries = []
    for x in list(range(0,len(Data))):
     list_of_entries.append(Data[x][0])

    #print(list_of_entries)
    return Data,list_of_entries

#Data = csv_reader  |  list_of_entries = ονοματα καθηγητων στα ελληνικα
Data, list_of_entries = csv_open()
```

Εικόνα 7.6.2 Συνάρτηση csv_open()

7.6.3 person_csv_to_db()

Η συγκεκριμένη συνάρτηση έχει φτιαχτεί για να λειτουργεί μόνο σε ειδικές περιπτώσεις. Καλείται με την εκκίνηση της εφαρμογής και ελέγχει για το αν υπάρχουν όλα τα ονόματα από το αρχείο csv, μέσα στην βάση δεδομένων. Αν κάποιο από αυτά λείπει, το προσθέτει. Μια περίπτωση λειτουργείας της συνάρτησης είναι η διαγραφή ολόκληρης της βάσης.

```
#σε περιπτωση που εχουν χαθει δεδομενα απο την person , τν ξαναγεμιζει
def person_csv_to_db():
    persons= c.execute("""SELECT id FROM person""").fetchall()
    for i in range(0,len(persons)):
            persons[i]=persons[i][0]
    for data in Data:
        if data[2] not in persons:
            c.execute("INSERT INTO person(id, pname) VALUES(:id, :pname)",
                {
                    'id' : data[2],
                    'pname' :str(data[1]),
                }
            )
        connection.commit()


person_csv_to_db()
```

Εικόνα 7.6.3 Συνάρτηση person_csv_to_db()

7.6.4 return_key()

Αυτή η συνάρτηση επιστρέφει το id του καθηγητή που επιλέχθηκε και τρέχει κάθε φορά που επιλέγουμε ένα καθηγητή με το κουμπί επιλογής καθηγητή. Έτσι η εφαρμογή αναγνωρίζει για ποιον καθηγητή θα προβάλει ή θα αποθηκεύσει δεδομένα.

```
# επιστρεφει τον id του καθηγητη που επιλεχθηκε
def return_key(*args):
    for row in Data:
        if clicked.get() == row[0]:
            id_key = row[2]
            print(id_key)
            return id_key
```

Εικόνα 7.6.4 Συνάρτηση return_key()

7.6.5 import_jsonfile()

Η συνάρτηση import_jsonfile() προσθέτει το αρχείο json στην εφαρμογή και ενεργοποιείται με το πατήματα του κουμπιού «προσθήκη αρχείου json». Για να λειτουργήσει πρέπει πρώτα να έχουμε επιλέξει τον καθηγητή για τον οποίο θέλουμε να δώσουμε τα δεδομένα. Μόλις πατήσουμε το κουμπί εμφανίζεται ένα παράθυρο εξερεύνησης αρχείων χρήστη. Εκεί επιλέγουμε το αρχείο json που θέλουμε να προσθέσουμε και πατάμε άνοιγμα. Η συνάρτηση παίρνει τα δεδομένα από το αρχείο, ελέγχει αν τα δεδομένα είναι πλήρες και αν έχουν προσθεθεί ξανά και τα περνάει στην βάση με το όνομα του καθηγητή που επιλέξαμε.

```
# fortosi json file -koumpi jsonfile- epistrefei diadromi arxeioy
def import_jsonfile():
    REQUIRED_FIELDS= {"type", "title", "authors", "year", "source", "cites"}
    id_key = return_key()
    flag=0
    if clicked.get() != "Επιλέξτε Καθηγητή":
        json_name = askopenfilename(initialdir="C:/Users/%s/Documents" % user,
                                    filetypes =(("Json File", "*.json"), ("All
Files", "*.*")),
                                    title = "Choose a file."
                                    )

        try:
            with open(json_name ,'r',encoding="utf-8 -sig") as json_file:
                    jdata = json.load(json_file)

            # elegxos gia kena pedia sto arxeio json
            ids = c.execute("SELECT title FROM papers").fetchall()
            list_ids=[i[0] for i in ids]
            print(list_ids)
            for paper in jdata:
                if REQUIRED_FIELDS.issubset(set(paper.keys())) : # elegxos gia
kena pedia sto arxeio json
                    if paper['title'] in list_ids : # an t ar8ro uparxei idi,
enimerose t cites
                        c.execute("UPDATE papers SET cites = ? WHERE title= ?
",(paper['cites'],paper['title']))
                        flag = 1
```

```
                        connection.commit()
                        print("enimerose: ", paper['title'])

                        # paper_ids =  c.execute("SELECT paper_id FROM
person_paper WHERE person_id = ?",(id_key,)).fetchall()
                        # list_paper_ids=[i[0] for i in paper_ids]
                        # if paper['uid'] not in list_paper_ids: #an to ar8ro
uparxei idi alla m  allo ka8igiti
                        #     c.execute("INSERT INTO person_paper(person_id,
paper_id) VALUES(:person_id, :paper_id)",
                        #         {
                        #             'person_id' : str(id_key),
                        #             'paper_id' : str(paper['uid']),
                        #             }
                        #         )
                        #     connection.commit()
                        #     flag=0
                        #     print("pros8ese person_paper")

                    else: #den yparxei t ar8ro
                        c.execute("INSERT INTO papers(types, title, authors,
years, source, cites) VALUES \
                                    (:types, :title, :authors, :years, :source,
:cites)",
                                {
                                    'types' : str(paper['type']),
                                    'title' : paper['title'],
                                    'authors' : str(paper['authors']),
                                    'years' : int(paper['year']),
                                    'source' : paper['source'],
                                    'cites' : int(paper['cites'])
                                        }
                                    )
                        connection.commit()
                        print("pros8ese paper")


                #         c.execute("INSERT INTO person_paper(person_id,
paper_id) VALUES(:person_id, :paper_id)",
                #                 {
                #                     'person_id' : str(id_key),
                #                     'paper_id' : str(paper['uid']),
                #                     }
                #                 )
                #         connection.commit()
                #         print("pros8ese person_paper")
                # else:
                #     print( "Ελλειπή στοιχεια")
                #     continue
```

```
        except FileNotFoundError:
            tk.messagebox.showerror(title="ERROR", message="Δεν βρέθηκε
αρχείο!!!")
    else:
        tk.messagebox.showerror(title="ERROR", message="Επιλέξτε καθηγητή!!!")

    if flag ==1:
             tk.messagebox.showinfo(title="Ενημέρωση",  message="Οι  εγραφές
ενημερώθηκαν!!")
```

Εικόνα 7.6.5 Συνάρτηση import_jsonfile()

7.6.6 data_print()

Η συνάρτηση data_print() δέχεται το όνομα του καθηγητή που επιλέξαμε και μας εμφανίζει όλες τις δημοσιεύσεις που υπάρχουν στην βάση δεδομένων στο όνομα του συγκεκριμένου καθηγητή.

```
# provoli data apo base
def data_print():
    if clicked.get() != "Επιλέξτε Καθηγητή":
        for i in my_tree.get_children():     # καθαρισμος treeview
            my_tree.delete(i)

        # c.execute("""SELECT * FROM papers
        #             JOIN person_paper ON papers.uidd =
person_paper.paper_id
        #             WHERE person_paper.person_id = ?
        #             ORDER BY years""",(return_key(),))
        c.execute("""SELECT * FROM papers
                  ORDER BY years""")
        papers = c.fetchall()

        for row in (papers):
            my_tree.insert("", tk.END, values=row)
        connection.commit()
    else:
        tk.messagebox.showerror(title="ERROR", message="Επιλέξτε καθηγητή!!!")
```

Εικόνα 7.6.6 Συνάρτηση data_print()

7.6.7 results()

Η συνάρτηση results() μας δίνει τα στατιστικά και τα σχηματικά που προκύπτουν από τα δεδομένα που έχουμε δώσει στην βάση. Ενεργοποιείται από το κουμπί με τη λίστα αποτελεσμάτων και ανοίγει ένα ξεχωριστό παράθυρο για κάθε επιλογή και με την βοήθεια της βιβλιοθήκης matplotlib. Στην πρώτη επιλογή βλέπουμε το πλήθος όλων των δημοσιεύσεων από όλα τα μέλη του τμήματος την τελευταία πενταετία. Έτσι η συνάρτηση υπολογίζει το πλήθος για κάθε έτος και εμφανίζει ένα πίνακα Text και μια πίτα με τα αποτελέσματα.

```
#koympi leitourgeion
def results(*args):
    current_year = datetime.now().year
    function_number =functions.index(clicked2.get())
    zipped_data=list(zip(*Data))
    if function_number == 0:
        #πληθος αρθρων ανα ετος απο ολους τς καθηγητες μαζι
        c.execute("""SELECT years FROM papers
                  WHERE years >= ?
                  ORDER BY years""",(current_year-4,))

        paper_year = c.fetchall()

        papers_per_year=dict(Counter(elem[0] for elem in paper_year))
        #print(papers_per_year)

        pop = Toplevel(root)
        pop.title("Πλήθος δημοσιεύσεων ανα έτος από όλα τα μέλη του τμήματος
μαζί (5ετία)")
        pop.geometry("1000x550+150+90")
        pop.config(bg="#505251")

        text = Text(pop)
        text.place(relx=.02 , rely=.03 , relwidth=.12 , relheight=.4)
        for key in papers_per_year.keys():
            text.insert(INSERT, str(key) + ": " + str(papers_per_year[key]) +
'\n')


        ok = Button(pop, text="OK",bg="#3bdb51", command=lambda:
pop.destroy())
        ok.place(relx=.02 , rely=.85 , relwidth=.09 , relheight=.09)

        #matplotlib
        exp_vals = papers_per_year.values()
        exp_labels = papers_per_year.keys()
```

```
        figure1 = plt.figure()
        ax = figure1.subplots()
        ax.set_title('Πλήθος δημοσιεύσεων ανα έτος από όλα τα μέλη του
τμήματος (5ετία)')
        ax.pie(exp_vals,labels=exp_labels, shadow=True,autopct='%1.1f%%')
        pie = FigureCanvasTkAgg(figure1, pop)
        pie.get_tk_widget().place(relx=.2, rely=.03, relwidth=.7 ,
relheight=.9)
```

Εικόνα 7.6.7 Συνάρτηση results() (1)

Στην δεύτερη περίπτωση παίρνει τα δεδομένα από την βάση και με την βοήθεια των συναρτήσεων «h_index» και «i_10» που φτιάξαμε, υπολογίζει τις τιμές h-index και i-10 για κάθε καθηγητή. Στη συνέχεια τα προβάλει σε ξεχωριστό παράθυρο με την χρήση ραμβογράμματος.

```
elif function_number ==1 :
        #h-index και το i-10
        authors_names=list()
        h_index_per_author=list()
        i10_per_author=list()
        for id in zipped_data[2]: # pairno cites gia ka8e ka8igiti xorista
            c.execute("""SELECT cites FROM papers
                         JOIN person_paper ON papers.uidd =
person_paper.paper_id
                         WHERE person_paper.person_id = ? """,(id,))
            citations= c.fetchall()
            for i in range(0,len(citations)):
                citations[i]=citations[i][0] # t kano aplo list apo
tuple
            h_index_per_author.append(h_index(citations))
            i10_per_author.append(i_10(citations))
        for i in range(0,len(zipped_data[0])):
            if h_index_per_author[i] is None: #entopizo k allazo oti einai
None se 0 giati vgazei error sto 307
                h_index_per_author[i]=0
            authors_names.append(zipped_data[0][i])
            print(zipped_data[0][i] ," H-index:" ,h_index_per_author[i], ", I-
10:", i10_per_author[i])

        pop = Toplevel(root)
        pop.title("H-index και I10 κάθε καθηγητή")
        pop.geometry("1280x680+20+20")
        pop.config(bg="#505251")

        text = Text(pop)
```

```
        text.place(relx=.01 , rely=.03 , relwidth=.16 , relheight=.8)

        for i, item in enumerate(zipped_data[0]):
            text.insert(tk.INSERT, zipped_data[0][i] +  ":" "\n" + "H-index: "
+ str(h_index_per_author[i]) + " | I-10:" + str(i10_per_author[i]) +"\n"+"\n")

        ok = Button(pop, text="OK",bg="#3bdb51", command=lambda:
pop.destroy())
        ok.place(relx=.03, rely=.88, relwidth=.09 , relheight=.09)

        labels =authors_names
        data1 = h_index_per_author
        data2 = i10_per_author

        x = np.arange(len(labels))  # the label locations
        width = 0.35 # the width of the bars

        figure1, ax = plt.subplots(dpi=80)

        rects1=ax.barh(x + 0.2+width/2, data1, width, label="H_index")
        rects2=ax.barh(x - 0.2+width/2, data2, width, label="I_10")

        ax.set_title('H-index και I10 κάθε καθηγητή')
        ax.set_yticks(x)
        ax.set_yticklabels(labels)
        ax.legend()

        ax.bar_label(rects1, padding=3)
        ax.bar_label(rects2, padding=3)

        plot = FigureCanvasTkAgg(figure1, pop)
                plot.get_tk_widget().place(relx=.2,   rely=.03,relwidth=.88,
relheight=.95 ,anchor='nw')
```

Εικόνα 7.6.8 Συνάρτηση results() (2)

```
def h_index(citations):
    citations.sort(reverse = True)
    for indx , citation in enumerate(citations):
        if indx + 1 >= citation:
            return citation


def i_10(citations):
    citations.sort(reverse = True)
    count=0
    for indx  in citations:
        if indx >= 10:
            count= count + 1
```

```
    return count
```

Εικόνα 7.6.9 Συναρτήσεις h_index() και i_10()

Οι επόμενες επιλογές εμφανίζουν τον αριθμό συνολικών αναφορών ανά καθηγητή και τον αριθμό δημοσιεύσεων ανά καθηγητή αντίστοιχα, με τη χρήση ραβδογράμματος.

```
elif function_number ==2 :
        #αριθμος αναφορων ανα καθηγητη
        authors_names = list()
        cite=list()
        for i,id in enumerate(zipped_data[2]): # pairno cites gia ka8e
ka8igiti xorista
            authors_names.append(zipped_data[0][i])
            c.execute("""SELECT cites FROM papers
                        JOIN person_paper ON papers.uidd =
person_paper.paper_id
                        WHERE person_paper.person_id = ? """,(id,))
            citations= c.fetchall()
            for y in range(0,len(citations)):
                citations[y]=citations[y][0] # t kano aplo list apo tuple
            cite.append(sum(citations))
            #print(authors_names[i] , ':', cite[i])  # na all3o to id sto
onoma tou ka8igiti

        pop = Toplevel(root)
        pop.title("Αριθμός αναφορών ανά καθηγητή")
        pop.geometry("1520x750+10+10")
        pop.config(bg="#505251")

        text = Text(pop)
        text.place(relx=.01 , rely=.03 , relwidth=.16 , relheight=.8)

        for i, item in enumerate(cite):
            text.insert(tk.INSERT, authors_names[i] + ":" +" "+ str(cite[i]) +
"\n"+"\n")

        ok = Button(pop, text="OK",bg="#3bdb51", command=lambda:
pop.destroy())
        ok.place(relx=.03, rely=.88, relwidth=.09 , relheight=.09)

        exp_vals = cite
        exp_labels = authors_names

        figure1, ax = plt.subplots(dpi=90)
        ax.set_title('Αριθμός αναφορών ανά καθηγητή')
        ax.barh(authors_names,exp_vals)
```

```
        for Y, X in enumerate(exp_vals):
            ax.annotate(X, xy=(X,Y), va='center')

        plot = FigureCanvasTkAgg(figure1, pop)
        plot.get_tk_widget().place(relx=.2, rely=.03,relwidth=.88,
relheight=.95 ,anchor='nw')
```

Εικόνα 7.6.10 Συνάρτηση results() (3)

```
elif function_number ==3 :
        #Αριθμός δημοσιεύσεων ανά καθηγητή
        authors_names = list()
        count_papers = list()
        for i,id in enumerate(zipped_data[2]):
            authors_names.append(zipped_data[0][i])
            c.execute("""SELECT COUNT(person_id) FROM person_paper
                        WHERE person_id = ? """,(id,))
            count_papers.append(c.fetchone())
        for i in range(0,len(count_papers)):
            count_papers[i]=count_papers[i][0]
            #print(authors_names[i] ,":", count_papers[i])

        pop = Toplevel(root)
        pop.title("Αριθμός δημοσιεύσεων ανά καθηγητή")
        pop.geometry("1520x750+10+10")
        pop.config(bg="#505251")

        text = Text(pop)
        text.place(relx=.01 , rely=.03 , relwidth=.16 , relheight=.8)

        for i, item in enumerate(count_papers):
            text.insert(tk.INSERT, authors_names[i] + ":" + " " +
str(count_papers[i]) + "\n"+"\n")

        ok = Button(pop, text="OK",bg="#3bdb51", command=lambda:
pop.destroy())
        ok.place(relx=.03, rely=.88, relwidth=.09 , relheight=.09)

        exp_vals = count_papers
        exp_labels = authors_names

        figure1, ax = plt.subplots(dpi=90)
        ax.set_title('Αριθμός δημοσιεύσεων ανά καθηγητή')
        ax.barh(authors_names,exp_vals)

        for Y, X in enumerate(exp_vals):
```

```
            ax.annotate(X, xy=(X,Y), va='center')

        plot = FigureCanvasTkAgg(figure1, pop)
        plot.get_tk_widget().place(relx=.2, rely=.03,relwidth=.88,
relheight=.95 ,anchor='nw')
```

Εικόνα 7.6.11 Συνάρτηση results() (4)

Μια άλλη επιλογή είναι ο αριθμός συνολικών δημοσιεύσεων ανά καθηγητή ανά έτος για την τελευταία πενταετία.

```
elif function_number ==4 :
        #Αριθμός δημοσιεύσεων ανά καθηγητή ανά έτος (5ετία)
        authors_names = list()
        items_list = list()
        for i,id in enumerate(zipped_data[2]):
            authors_names.append(zipped_data[0][i])
            c.execute("""SELECT years FROM papers
                         JOIN person_paper ON papers.uidd =
person_paper.paper_id
                         WHERE person_paper.person_id = ? AND years >= ?
                         ORDER BY years""",(id,current_year-4))
            years= c.fetchall()
            papers_per_year_per_author=dict(Counter(elem[0] for elem in
years))
            items_list.append(papers_per_year_per_author)
            #print(authors_names[i] , items_list[i])

        pop = Toplevel(root)
        pop.title("Αριθμός δημοσιεύσεων ανά καθηγητή ανά έτος (5ετία)")
        pop.geometry("1520x750+0+0")
        pop.config(bg="#505251")

        text = Text(pop)
        text.place(relx=.01 , rely=.03 , relwidth=.16 , relheight=.8)

        for i, item in enumerate(items_list):
            text.insert(tk.INSERT, authors_names[i] + ":" + "\n")
            for key in item.keys():
                text.insert(tk.INSERT, "       "+ str(key) + " "+
str(item[key]) + "\n")

        ok = Button(pop, text="OK",bg="#3bdb51", command=lambda:
pop.destroy())
        ok.place(relx=.03, rely=.88, relwidth=.09 , relheight=.09)

        bar_data=[len(authors_names)*[0] for x in range(5)]
```

```
        for i,year in enumerate(range(current_year-4, current_year+1)):
            for item in range(0,len(items_list)):
                if year in items_list[item]:
                    temp = items_list[item]      # ta keys einai integers k
anagnorizontai os indexes st pinaka
                    bar_data[i][item]= temp[year]

        labels =authors_names
        data1 = bar_data[0]
        data2 = bar_data[1]
        data3 = bar_data[2]
        data4 = bar_data[3]
        data5 = bar_data[4]

        x = np.arange(len(labels))  # the label locations
        width = 0.12  # the width of the bars

        figure1, ax = plt.subplots(dpi=90)

        ax.barh(x + 1.1*width, data1, width, label=current_year-4)
        ax.barh(x + 1.1*width*2, data2, width, label=current_year-3)
        ax.barh(x + 1.1*width*3, data3, width, label=current_year-2)
        ax.barh(x + 1.1*width*4, data4, width, label=current_year-1)
        ax.barh(x + 1.1*width*5, data5, width, label=current_year)

        ax.set_title('Αριθμός δημοσιεύσεων ανά καθηγητή ανά έτος (5ετία)')
        ax.set_yticks(x+ 2.5*width+width/2)
        ax.set_yticklabels(labels)
        ax.legend()

        plot = FigureCanvasTkAgg(figure1, pop)
        plot.get_tk_widget().place(relx=.2, rely=.03,relwidth=.88,
relheight=.95 ,anchor='nw')
```

Εικόνα 7.6.12 Συνάρτηση results() (5)

Τελευταία επιλογή είναι η εμφάνιση των τύπων των δημοσιεύσεων των μελών του τμήματος την τελευταία πενταετία.

```
elif function_number ==5 :
        #Τύποι δημοσιεύσεων μελών του τμήματος (5ετία)
        authors_names = list()
        items_list = list()

        c.execute("""SELECT types FROM papers
```

```
                    WHERE years >= ?""",(current_year-4,))

        types= c.fetchall()

        for i in range(0,len(types)):
            types[i]=types[i][0] # t kano aplo list apo tuple
        items_list.append(types)
        print(items_list)

        counts = Counter(items_list[0])
        print(counts)

        pop = Toplevel(root)
        pop.title("Τύποι δημοσιεύσεων μελών του τμήματος (5ετία)")
        pop.geometry("970x550+150+90")
        pop.config(bg="#505251")

        text = Text(pop)
        text.place(relx=.02 , rely=.03 , relwidth=.2 , relheight=.4)

        for key in counts.keys():
            text.insert(tk.INSERT, str(key) + ": " + str(counts[key]) + '\n')


        ok = Button(pop, text="OK",bg="#3bdb51", command=lambda:
pop.destroy())
        ok.place(relx=.04, rely=.85, relwidth=.09 , relheight=.09)

        exp_vals = counts.values()
        exp_labels = counts.keys()

        figure1 = plt.figure()
        ax = figure1.subplots()
        ax.set_title('Τύποι δημοσιεύσεων μελών του τμήματος (5ετία)')
        ax.pie(exp_vals,labels=exp_labels, shadow=True,autopct='%1.1f%%')
        pie = FigureCanvasTkAgg(figure1, pop)
        pie.get_tk_widget().place(relx=.24, rely=.03, relwidth=.7 ,
relheight=.9)
```

Εικόνα 7.6.13 Συνάρτηση results() (6)

7.6.8 add_person_creation(), add_person()

Η συνάρτηση add_person_creation() δημιουργεί ένα παράθυρο στο οποίο ο χρήστης πρέπει να συμπληρώσει τα στοιχεία του καθηγητή που θέλει να προσθέσει. Μόλις πατήσει το κουμπί «OK» ενεργοποιείται η συνάρτηση add_person() η οποία προσθέτει τον καθηγητή στην βάση δεδομένων και στο αρχείο csv.

```
# παραθυρο προσθηκης νεου καθηγητη
def add_person_creation():
    pop = Toplevel(root)
    pop.title("Προσθήκη νέου καθηγητή")
    pop.config(bg="#c6e2ff")
    pop.geometry("650x150+400+300")

    pop_label = Label(pop, text="Όνομα με Ελληνικούς χαρακτήρες
:",  font=("helvetica", 12),bg="#c6e2ff")
    pop_label.grid(row=0, column=0, padx=10 ,pady=10)

    pop_label2 = Label(pop, text="Όνομα με Λατινικούς χαρακτήρες
:",  font=("helvetica", 12),bg="#c6e2ff")
    pop_label2.grid(row=1, column=0, padx=10,pady=10)

    pop_label3 = Label(pop, text="Κωδικός Google Scholar ID
:",  font=("helvetica", 12),bg="#c6e2ff")
    pop_label3.grid(row=2, column=0, padx=10,pady=10)

    elperson = Text(pop, height =" 1", width = "30" )
    elperson.grid(row=0, column=1, padx=10)

    enperson = Text(pop, height =" 1", width = "30" )
    enperson.grid(row=1, column=1, padx=10)

    scholar_id = Text(pop, height =" 1", width = "30" )
    scholar_id.grid(row=2, column=1, padx=10)

    ok = Button(pop, text="OK",height =" 3", width = "8",bg='#00ff00',relief=
"flat",
                command=lambda: add_person(elperson.get("1.0",tk.END),
enperson.get("1.0",tk.END) , scholar_id.get("1.0",tk.END) , pop))
    ok.grid(row=1, column=4, padx=10)
```

Εικόνα 7.6.14 Συνάρτηση add_person_creation()

```
#lambda προσθηκη νεου καθηγητη
def add_person(elperson, enperson, scholar_id, pop):
    global Data, list_of_entries
    el = elperson.strip()
    en = enperson.strip()
    sid = scholar_id.strip()

    sid_list=list(zip(*Data))[2]

    if el != "" and en != "" and sid != "":
            if sid not in sid_list:
                try:
```

```
                    print("uparxei t id")
                    error=c.execute("INSERT INTO person(id, pname) VALUES(:id,
:pname )", #  t error epistrefei t sql string i an uparxei error, t error
                                    {
                                        'pname' : str(en),
                                        'id' : str(sid)
                                        }
                                )
                    connection.commit()
                    with open(r'optnames.csv', 'a', encoding="utf-8 -sig") as
f:
                        f.write("{},{},{}\n".format(el, en, sid))

                except:
                    connection.rollback()
                    tk.messagebox.showerror(title="ERROR", message="Αποτυχία
προσθήκης")
                    print(error)
            else:
                tk.messagebox.showerror(title="ERROR", message="Ο καθηγητής
υπάρχει ήδη")

    else:
        tk.messagebox.showerror(title="ERROR", message="Κενά πεδία")

    Data, list_of_entries = csv_open()

    menu = drop["menu"]
    menu.delete(0, "end")
    for string in list_of_entries:
        menu.add_command(label=string,
                                command=lambda value=string:
clicked.set(value))

    pop.destroy()
```

Εικόνα 7.6.15 Συνάρτηση add_person()

7.6.9 delete_person_creation(), delete_person()

Η συνάρτηση delete_person_creation() ανοίγει ένα παράθυρο που περιέχει μια λίστα με όλους τους καθηγητές που υπάρχουν στην βάση. Εκεί ο χρήστης επιλεγεί αυτόν που θέλει να διαγράψει και πατάει το κουμπί «Διαγραφή». Το κουμπί ενεργοποιεί την συνάρτηση delete_person() η οποία συνδέεται στην βάση δεδομένων και διαγραφεί τον καθηγητή και

τα αρχεία που αναφέρονται στο όνομα του. Επίσης ο καθηγητής διαγράφεται και από το αρχείο csv

```
#παραθυρο διαγραφή καθηγητη
def delete_person_creation():
    pop = Toplevel(root)
    pop.title("Διαγραφή καθηγητή")
    pop.geometry("400x450+250+170")
    pop.config(bg="#c6e2ff")

    global mylist
    mylist = Listbox(pop, height =" 20", width = "35", selectmode =
"multiple",selectforeground='white',selectbackground='#32667d')
    mylist.grid(row=0, column=0, padx=20,pady=20)
    for item in list_of_entries:
        mylist.insert(END,item)

    dbutton = Button(pop, text="Διαγραφή",height =" 3", width =
"10",bg="#d13636",fg='white',relief= "flat",
                     command=lambda: delete_person(mylist.curselection() ,
pop))
    dbutton.grid(row=0, column=1, padx=10)
```

Εικόνα 7.6.16 Συνάρτηση delete_person_creation()

```
#lambda διαγραφη καθηγητη
def delete_person(indices_to_delete, pop):
    global Data, list_of_entries
    ids_to_delete =[]
    names_to_delete=[]

    try:
        connection.commit()
        for row in enumerate(Data):
            if row[0] in indices_to_delete:
                ids_to_delete.append(row[1][2])
                names_to_delete.append(row[1][0])

        print(ids_to_delete)
        sqlquery1 =c.execute("""SELECT papers.uidd FROM papers
                            JOIN person_paper ON papers.uidd =
person_paper.paper_id
                            WHERE person_paper.person_id IN (""" + ",
".join(["?"] * len(ids_to_delete)) + ")",ids_to_delete).fetchall()

        for i in range(0,len(sqlquery1)):
                sqlquery1[i]=sqlquery1[i][0]
```

```
        print(sqlquery1,"sql1")

        sqlquery2 =c.execute("""SELECT person_paper.paper_id FROM person_paper
                            WHERE person_paper.paper_id  IN ("""+ ",
".join(["?"] * len(sqlquery1))+
                            ") AND person_paper.person_id NOT IN ("+ ",
".join(["?"] * len(ids_to_delete))+ ")",sqlquery1+ids_to_delete).fetchall()

        for i in range(0,len(sqlquery2)):
                sqlquery2[i]=sqlquery2[i][0]
        print(sqlquery2,"sql2") # elegxos gia keno sqlquery output

        paper_set  = set(sqlquery1)
        person_paper_set = set(sqlquery2)  # ta kano set gia n afero t
duplicates

        not_duplicates = list(paper_set-person_paper_set)

        c.execute("""DELETE FROM papers
                  WHERE  papers.uidd IN (""" + ", ".join(["?"] *
len(not_duplicates)) + ")",not_duplicates)

        c.execute("""DELETE FROM person_paper
                WHERE person_id IN (""" + ", ".join(["?"] *
len(ids_to_delete)) + ")",ids_to_delete)

        c.execute("""DELETE FROM person
                WHERE id IN (""" + ", ".join(["?"] * len(ids_to_delete)) +
")",ids_to_delete)

        with open('optnames.csv','w',encoding="utf-8 -sig",newline='') as
file:
           writer=csv.writer(file)
           for row in enumerate(Data):
               if row[0] not in indices_to_delete:
                   writer.writerow(row[1])

        connection.commit()

        for x in names_to_delete:
            list_of_entries.remove(x)
        menu = drop["menu"]
        menu.delete(0, "end")
        for string in list_of_entries:
            menu.add_command(label=string,
                             command=lambda value=string: clicked.set(value))

    except:
        connection.rollback()
```

```
        tk.messagebox.showerror(title="ERROR", message="Αποτυχία
Διαγραφής")

    connection.commit()
    Data, list_of_entries = csv_open()
    pop.destroy()
```

Εικόνα 7.6.17 Συνάρτηση delete_person()

7.6.10 scholar_info(), el_copy(), en_copy(), id_copy()

Η συνάρτηση scholar_ info() χρησιμοποιήθηκε για να βοηθήσει τον χρήστη στην αναζήτηση του καθηγητή στο Google Scholar. Επιλέγοντας έναν καθηγητή από την λίστα καθηγητών και πατώντας το κουμπί «Scholar info» εμφανίζεται ένα παράθυρο στο οποίο περιχέονται τα στοιχεία του καθηγητή. Στο κάτω μέρος του παραθύρου υπάρχουν 3 κουμπιά τα οποία αντιγράφουν τα στοιχεία του καθηγητή στο πρόχειρο για την διευκόλυνση του χρήστη στην αντιγραφή των στοιχείων.

```
#ονομα κ id καθηγητη
def scholar_info():

    if clicked.get() != "Επιλέξτε Καθηγητή":
        pop = Toplevel(root)
        pop.title("Στοιχεία Google scholar")
        pop.geometry("530x140+400+300")
        pop.config(bg="#c6e2ff")

        for x in range(0,len(Data)):
            if Data[x][0] == clicked.get():
                row = Data[x]

        #print(row[0] + row[1] + row[2])
        text = Text(pop, height =" 2", width = "58",font=("Helvetica", 10))
        text.insert(tk.INSERT, row[0] +" | "+ row[1] +" | "+ row[2])
        text.place(relx=.03,rely=.63)

        el = Button(pop, text="Ελληνικά",height =" 2", width = "8" ,
bg='#32667d',fg="white",activeforeground="blue", command=el_copy)
        el.place(relx=.05,rely=.12)

        en = Button(pop, text="Λατινικά",height =" 2", width = "8" ,
bg='#32667d',fg="white",activeforeground="blue", command=en_copy)
        en.place(relx=.27,rely=.12)
```

```
        id = Button(pop, text="Scholar ID",height =" 2", width = "8" ,
bg='#32667d',fg="white",activeforeground="blue", command=id_copy)
        id.place(relx=.47,rely=.12)

        ok = Button(pop, text="OK",height =" 3", width =
"7",bg='#00ff00',relief= "flat", command=lambda: pop.destroy())
        ok.place(relx=.85,rely=.3)
    else:
        tk.messagebox.showerror(title="ERROR", message="Επιλέξτε καθηγητή!!!")
```

Εικόνα 7.6.18 Συνάρτηση scholar_info()

```
# συναρτησεις για κουμπι αντιγραφης στ συναρτηση scholar_info
def el_copy():
    for x in range(0,len(Data)):
            if Data[x][0] == clicked.get():
                text = Data[x]
    clipboard.copy(text[0])

def en_copy():
    for x in range(0,len(Data)):
            if Data[x][0] == clicked.get():
                text = Data[x]
    clipboard.copy(text[1])

def id_copy():
    for x in range(0,len(Data)):
            if Data[x][0] == clicked.get():
                text = Data[x]
    clipboard.copy(text[2])
```

Εικόνα 7.6.19 Συναρτήσεις el_copy(), en_copy και id_copy()

7.6.11 On_double_click(), update_record(), delete_record()

Η συνάρτηση on_double_click() συνδέεται με τον πίνακα προβολής δεδομένων. Εκεί ο χρήστης μπορεί να κάνει διπλό κλικ πάνω σε μια δημοσίευση και να την επεξεργαστεί. Κάνοντας διπλό κλικ η συνάρτηση ανοίγει ένα νέο παράθυρο όπου φαίνονται αναλυτικά όλα τα στοιχεία της δημοσίευ

σης τα οποία μπορούν να δεχθούν περαιτέρω επεξεργασία. Όταν ο χρήστης κάνει τις κατάλληλες αλλαγές μπορεί να πατήσει το πλήκτρο «Update» και να τις οριστικοποιήσει. Μπορεί επίσης να πατήσει το κουμπί «»Delete έτσι ώστε να διαγράψει τελείως την

δημοσίευση από την εφαρμογή. Οι δυο αυτές διεργασίες πραγματοποιούνται με την χρήση των αντίστοιχων συναρτήσεων «update_record()» και «delete_record()».

```
#συναρτηση χρησης διπλου κλικ στο μενουν των αρθρων
def on_double_click(event):
    item = my_tree.selection()
    values= my_tree.item(my_tree.selection())['values']

    pop = Toplevel(root)
    pop.title("Επεξεργασία δεδομένων")
    pop.geometry("770x440+400+100")
    pop.config(bg="#c6e2ff")

    #Labels
    id = Label(pop, text="ID",bg="#c6e2ff" )
    type = Label(pop, text="Type",bg="#c6e2ff")
    title = Label(pop, text="Title",bg="#c6e2ff")
    author = Label(pop, text="Authors",bg="#c6e2ff")
    year = Label(pop, text="Year",bg="#c6e2ff")
    source = Label(pop, text="Source",bg="#c6e2ff")
    cites = Label(pop, text="Cites",bg="#c6e2ff")

    id.grid(row=0, column=0, pady=5)
    type.grid(row=2, column=0, pady=5)
    title.grid(row=4, column=0, pady=5)
    author.grid(row=6, column=0, pady=5)
    year.grid(row=8, column=0, pady=5)
    source.grid(row=10, column=0, pady=5)
    cites.grid(row=12, column=0, pady=5)


    #Entry boxes
    id_box = Entry(pop,width=120)
    type_box = Entry(pop, width=120)
    title_box = Entry(pop, width=120)
    authors_box = Entry(pop, width=120)
    year_box = Entry(pop, width=120)
    source_box = Entry(pop, width=120)
    cites_box = Entry(pop, width=120 )

    id_box.insert(tk.INSERT,values[0])
    type_box.insert(tk.INSERT,values[1])
    title_box.insert(tk.INSERT,values[2])
    authors_box.insert(tk.INSERT,values[3])
    year_box.insert(tk.INSERT,values[4])
    source_box.insert(tk.INSERT,values[5])
```

```
    cites_box.insert(tk.INSERT,values[6])

    id_box.grid(row=1, column=0, padx=20)
    type_box.grid(row=3, column=0)
    title_box.grid(row=5, column=0)
    authors_box.grid(row=7, column=0)
    year_box.grid(row=9, column=0)
    source_box.grid(row=11, column=0)
    cites_box.grid(row=13, column=0)


    update_button = Button(pop, text="Update",  font="Helvetica",
bg='#00ff00', height =" 2", width = "10",relief="flat",
                           command=lambda
:update_record(values[0],id_box.get(), type_box.get(), title_box.get(),
                                                  authors_box.get(),
year_box.get(),
                                                  source_box.get(),
cites_box.get(), pop))

    update_button.grid(row=14,sticky="W", padx=150,pady= 15)

    delete_button = Button(pop, text="Delete",  font="Helvetica",
bg="#d13636",fg='white', height =" 2", width = "10",relief="flat",
                           command=lambda :delete_record(values[0],pop))

    delete_button.grid(row=14, sticky="E",padx=150, pady= 15)

    ok = Button(pop, text="OK",font="Helvetica",fg='#66ccff',  height =" 2",
width = "10", bg='#505251', command=lambda: pop.destroy())
    ok.grid(row=14,  padx=150, pady= 15)

    return item
```

Eik;ona 7.6.20 Syn;arthsh on_double_click()

```
# on_double_click update
def update_record(old_id, id_box, type_box, title_box, authors_box,
year_box,source_box, cites_box,pop):
    c.execute("""UPDATE papers SET
              uidd = :uidd,
              types = :types,
              title = :title,
              authors = :authors,
              years = :years,
              source = :source,
              cites = :cites
              WHERE uidd= :old_id """,
```

```
                {
                  'old_id' : old_id,
                  'uidd' : id_box,
                  'types' : type_box,
                  'title' : title_box,
                  'authors' : str(authors_box),
                  'years' : int(year_box),
                  'source' : source_box,
                  'cites' : int(cites_box)
                  })

    c.execute("""UPDATE person_paper SET paper_id = ? WHERE paper_id= ?""",
            (id_box, old_id,))
    connection.commit()

    data_print()
    pop.destroy()

# on_double_click  delete
def delete_record(id,pop):
    c.execute("""DELETE FROM papers
              WHERE papers.uidd = ? """,(id,))

    c.execute("""DELETE FROM person_paper
              WHERE person_paper.paper_id = ?
              AND person_paper.person_id = ? """,(id, return_key()) )

    x=c.execute("""SELECT person_paper.paper_id FROM person_paper
              WHERE person_paper.person_id = "YB4p5_8AAAAJ" """).fetchall()

    connection.commit()

    print(x)
    data_print()
    pop.destroy()
```

Εικόνα 7.6.21 Συναρτήσεις update_record() και delete_record()

8 Παρουσίαση εφαρμογής

8.1 Εκτελέσιμο αρχείο εφαρμογής

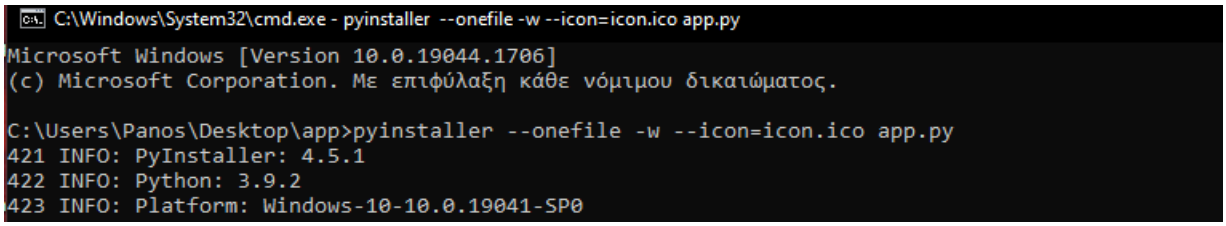
Για να δημιουργήσουμε το εκτελέσιμο αρχείο της εφαρμογής, ακολουθούμε μια σειρά από εντολές που θα αναφέρουμε στη συνέχεια.

Αρχικά πρέπει να εγκαταστήσουμε το PyInstaller, το οποίο είναι ένα λογισμικό που αναλύει των κώδικα και βρίσκει ποιες βιβλιοθήκες χρειάζεται για να εκτελεστεί τις οποίες και συλλέγει. Η εγκατάσταση γίνεται με την εντολή :

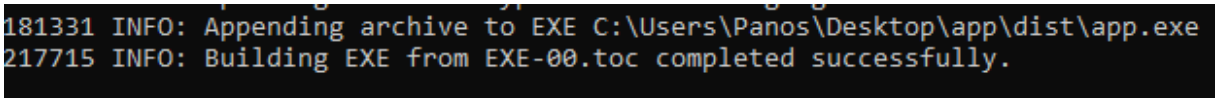
pip install pyinstaller

Για να τρέξουμε το PyInstaller ανοίγουμε μια γραμμή εντολών μέσα στο φακελο που βρίσκεται το πρόγραμμα και τρέχουμε την εξής εντολή:

pyinstaller --onefile -w --icon=icon.ico app.py



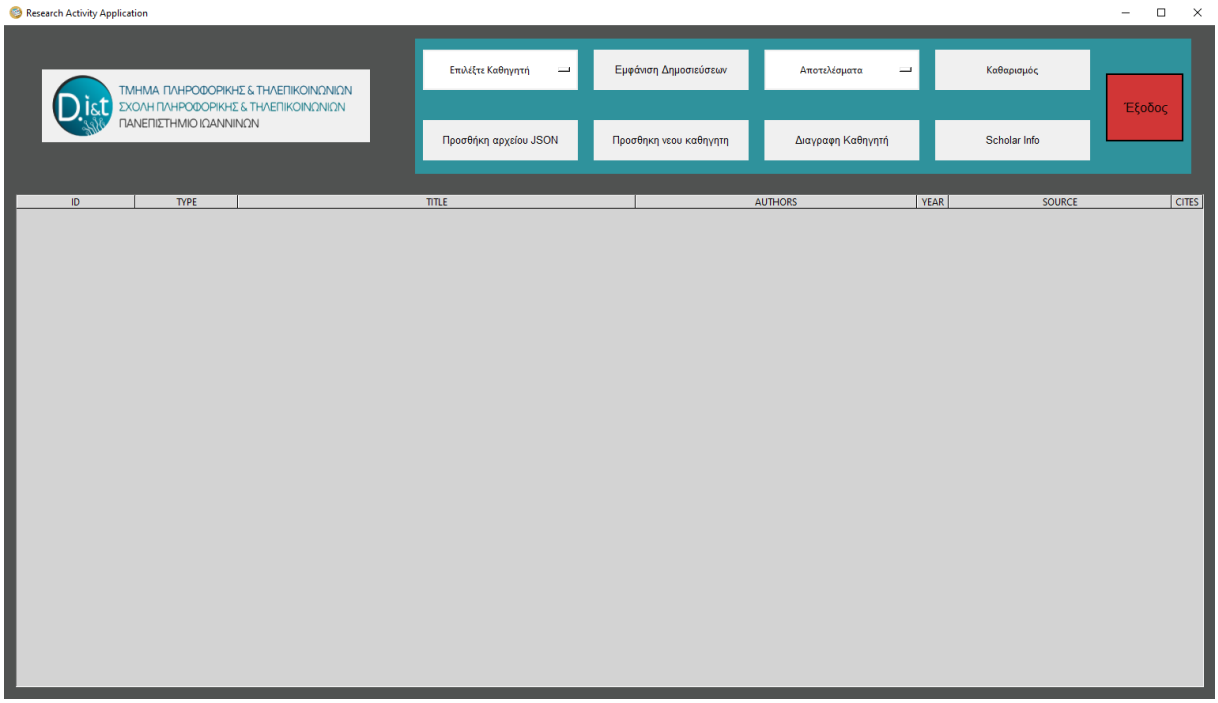
Η διαδικασία ολοκληρώνεται όταν εμφανιστεί το παρακάτω μήνυμα.



Τώρα μέσα στον αρχικό μας φακελο υπάρχει ένας υποφάκελος που ονομάζεται dist. Εκεί μέσα βρίσκεται το εκτελέσιμο αρχείο.

8.2 Λειτουργεία εφαρμογής

Ο τρόπος λειτουργείας της εφαρμογής είναι πολύ απλός, ώστε να μπορεί ο χρήστης γρήγορα και χωρίς δυσκολίες να επεξεργαστεί τα δεδομένα του και να πάρει τα αποτελέσματα που χρειάζεται.



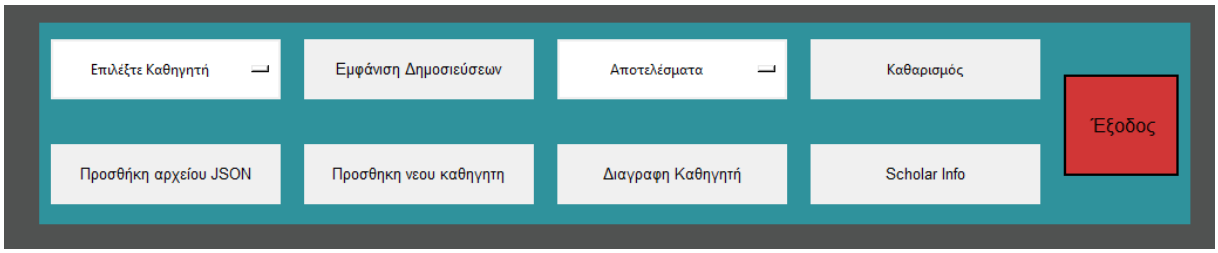
Εικόνα 8.2.1 Κύρια διεπαφή εφαρμογής

Η εφαρμογή δέχεται τα πρώτα της δεδομένα αυτόματα από την χρήση ενός csv αρχείου, το οποίο περιέχει τα ονόματα των καθηγητών με ελληνικούς και λατινικούς χαρακτήρες, καθώς επίσης και το google scholar id του καθενός. Το συγκεκριμένο αρχείο βρίσκεται από την αρχή στο φάκελο με τα υπόλοιπα αρχεία της εφαρμογής. Περιεχόμενο του αποτελούν τα στοιχεία των καθηγητών που είναι εγγεγραμμένοι στο τμήμα μέχρι και την περίοδο 2020-2021 για τους οποίους υπήρχε διαθέσιμο το προφίλ τους στην πλατφόρμα του Google Scholar. (Το συγκεκριμένο αρχείο μπορεί να δεχτεί αλλαγές με την χρήση της εφαρμογής, οπότε πιθανόν να διαφέρει μετά από την χρήση).

```
1  Αγγέλης Κωνσταντίνος,Constantinos T. Angelis,KjwXtK8AAAAJ
2  Αδάμ Σταύρος,Stavros P. Adam,KBd8e4kAAAAJ
3  Βαρτζιώτης Φώτιος,Vartziotis. Fotios,EvYhBKQAAAAJ
4  Γιαννακέας Νικόλαος,Nikos Giannakeas,0cawcn0AAAAJ
5  Γκόγκος Χρήστος,Christos Gogos,iNX_7IsAAAAJ
6  Γλαβάς Ευριπίδης,E. Glavas,lJfsZGkAAAAJ
7  Δουμένης Γρηγόριος,Gregory Doumenis,tx3_DZQAAAAJ
8  Καρβέλης Πέτρος,Petros Karvelis,z_ZpXjgAAAAJ
9  Λιάγκου Βασιλική,Vasiliki Liagkou,UMEG7NIAAAAJ
10 Μαργαρίτη Σπυριδούλα,Spiridoula V. Margariti,us9XwhYAAAAJ
11 Στεργίου Ελευθέριος,E. Stergiou,TBPgMkkAAAAJ
12 Στύλιος Χρυσόστομος,Chrysostomos Stylios,uhQpWk8AAAAJ
13 Τζάλλας Αλέξανδρος,Alexandros T. Tzallas,Jrpqo_QAAAAJ
14 Τσούλος Ιωάννης,Ioannis Tsoulos,jot0KgkAAAAJ
15 Φουτσιτζή Γεωργία,Georgia Foutsitzi,YofEfbAAAAAJ
16
```

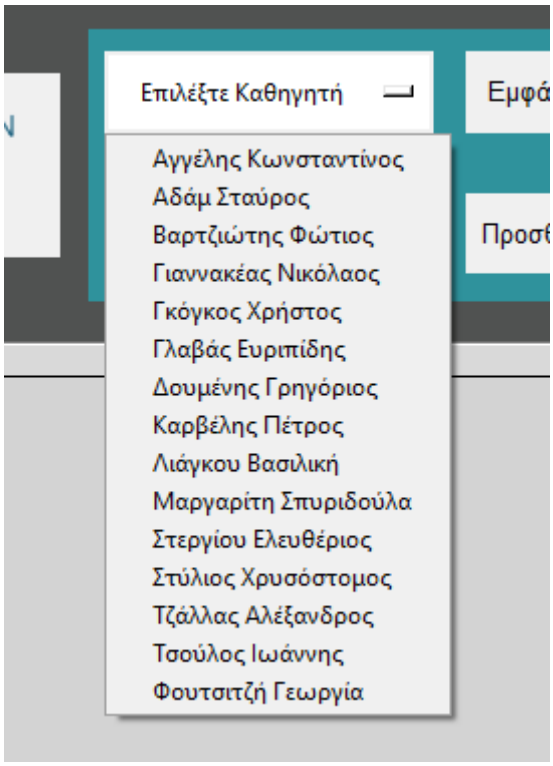
Εικόνα 8.2.2 Αρχείο csv

Ο χρήστης έχει την δυνατότητα να περιηγηθεί στην εφαρμογή και να πραγματοποιήσει τις λειτουργείες που επιθυμεί με την χρήση 9 πλήκτρων.



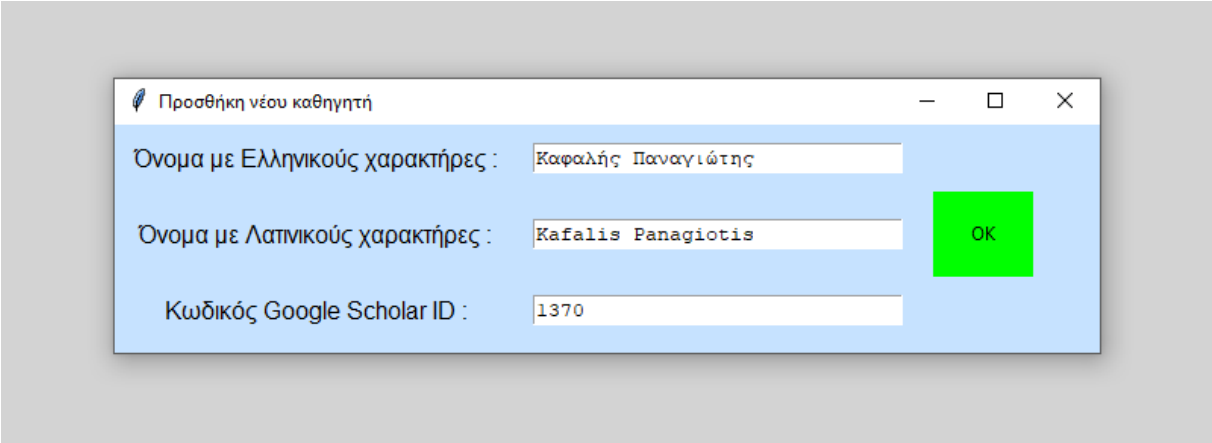
Εικόνα 8.2.3 Πλήκτρα εφαρμογής

Η πρώτη κίνηση κατά την έναρξη της εφαρμογής είναι η επιλογή του καθηγητή που θέλουμε είτε για να προσθέσουμε δεδομένα είτε για να εμφανίσουμε τα δεδομένα που υπάρχουν ήδη στην βάση μας.



Εικόνα 8.2.4 Λίστα καθηγητών

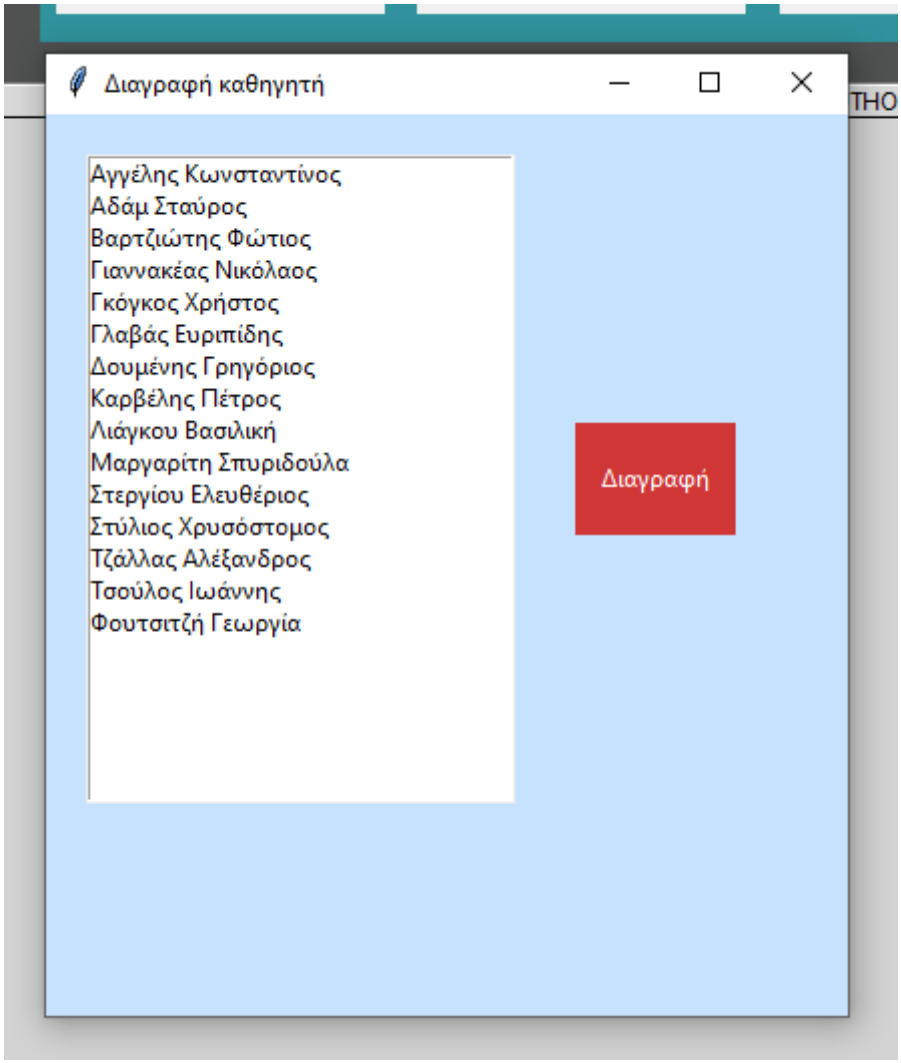
Για να προσθέσουμε ένα νέο καθηγητή στη εφαρμογή πατάμε το πλήκτρο «Προσθήκη νέου καθηγητή». συμπληρώνουμε το πινακάκι που εμφανίζεται και πατάμε το πλήκτρο «OK». Έτσι ο νέος καθηγητής έχει προστεθεί στην λίστα και στην βάση μας.



Εικόνα 8.2.5 Προσθήκη νέου καθηγητή

Αν θέλουμε να διαγράψουμε κάποιον καθηγητή από την εφαρμογή, πατώντας το πλήκτρο «Διαγραφή καθηγητή» μπορούμε να επιλέξουμε όσους καθηγητές θέλουμε από την λίστα

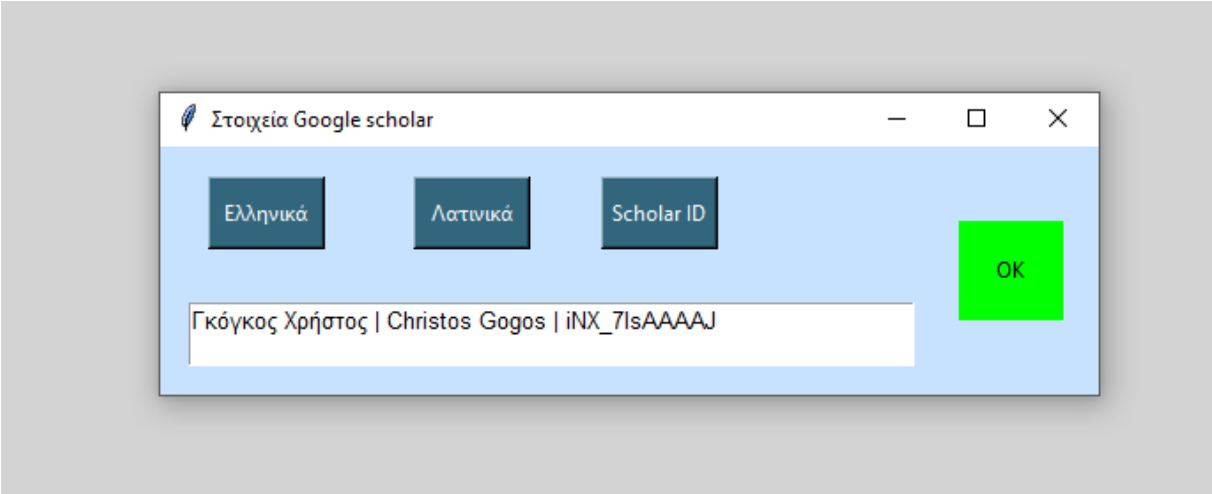
που εμφανίζεται και πατώντας το πλήκτρο «Διαγραφή», διαγράφονται μόνιμα από την εφαρμογή μαζί με τις δημοσιεύσεις που υπάρχουν αποθηκευμένες στο όνομα τους.



Εικόνα 8.2.6 Διαγραφή Καθηγητών

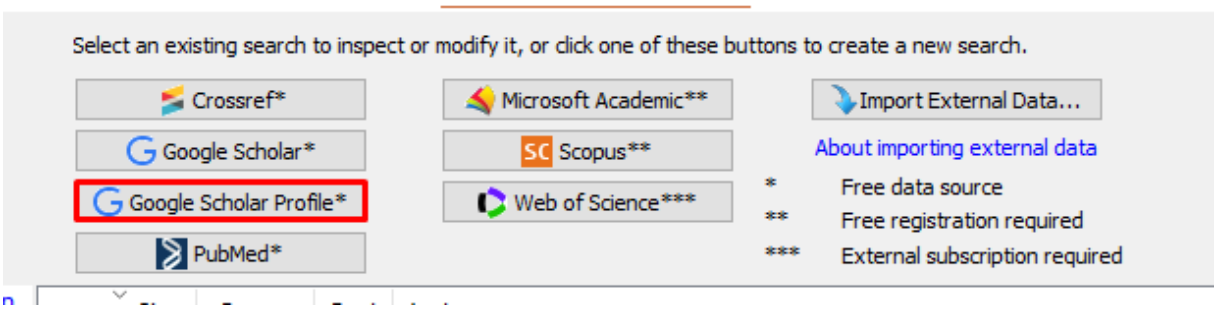
Η προσθήκη δεδομένων γίνεται με την χρήση αρχείων JSON. Ο χρήστης πρέπει να πάρει αυτά τα αρχεία με την βοήθεια ενός εξωτερικού προγράμματος που ονομάζεται Publish or Perish(συμβατό μέχρι την έκδοση 7). Εκεί θα αναζητήσει με τα στοιχεία Google Scholar του κάθε καθηγητή ξεχωριστά και θα αποθηκεύσει τα δεδομένα που βρέθηκαν, σε μορφή JSON. Για την ευκολία του χρήστη υπάρχει το πλήκτρο «Scholar Info» το οποίο εμφανίζει ένα παράθυρο με τα στοιχεία Google Scholar του καθηγητή που θα βοηθήσουν στην αναζήτησή του στην εφαρμογή Publish or Perish. Πατώντας στο αντίστοιχο πλήκτρο που

βρίσκεται πάνω από τα στοιχεία του καθηγητή, το κείμενο αντιγράφεται αυτόματα στο πρόχειρο.



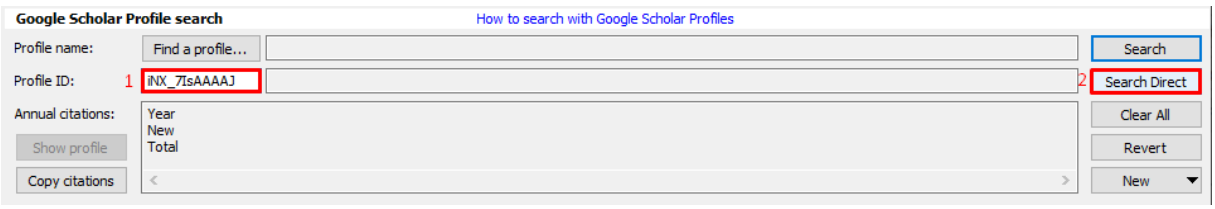
Εικόνα 8.2.7 Scholar Info

Έπειτα στην εφαρμογή Publish or Perish πατώντας στο κουμπί Google Scholar Profile , εμφανίζεται ένα πεδίο στο οποίο κάνουμε επικόλληση το id του καθηγητή.



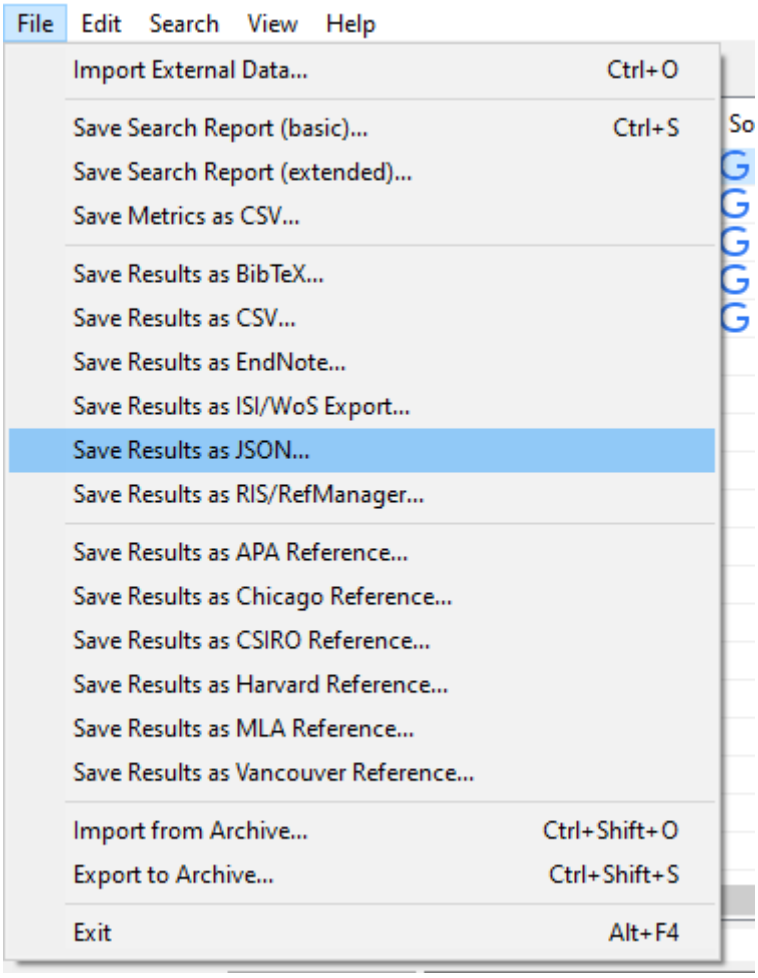
Εικόνα 8.2.8 Publish or Perish Google Scholar search

Και στη συνέχεια πατάμε το κουμπί Search Direct για αναζήτηση δημοσιεύσεων.



Εικόνα 8.2.9 Publish or Perish search direct

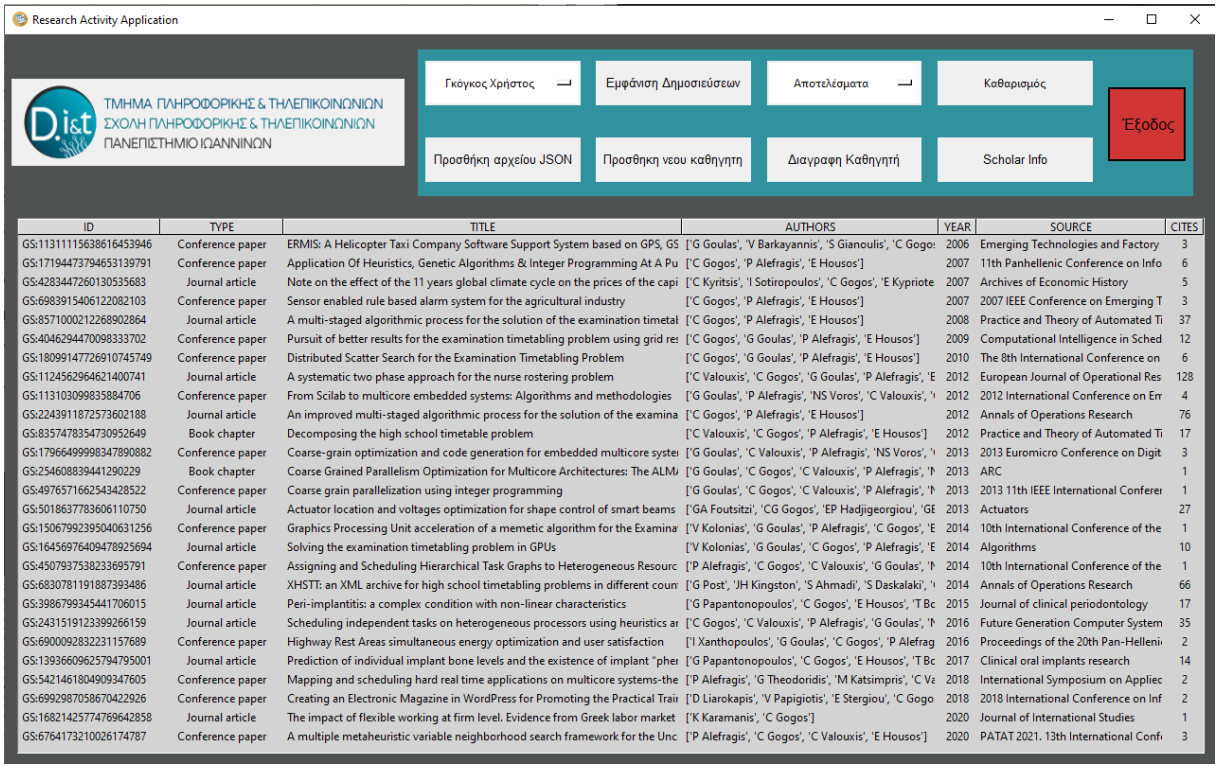
Μόλις ολοκληρωθεί η αναζήτηση, αποθηκεύουμε το αρχείο σε μορφή JSON.



Εικόνα 8.2.10 Αποθήκευση αρχείου σε μορφή JSON

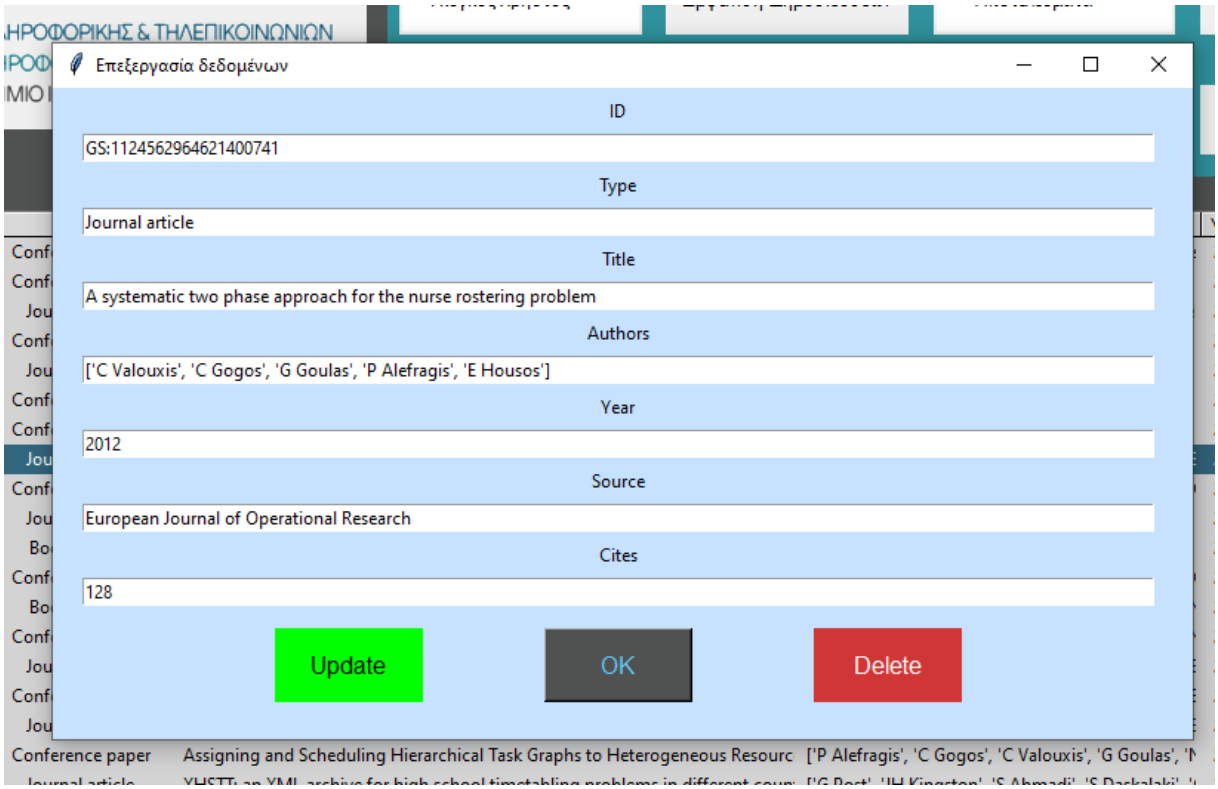
Να σημειωθεί πως όλη αυτή η διαδικασία θα μπορούσε να αποφευχθεί με την χρήση κάποιου API που θα περνάει αυτόματα τα δεδομένα από το Google Scholar στην εφαρμογή αλλά το μόνο διαθέσιμο API για την συγκεκριμένη δουλειά (scholarly API), μετά την ολοκλήρωση ενός συγκεκριμένου αριθμού αιτημάτων η πλατφόρμα μπλόκαρε την IP address του δικτύου από το οποίο γινόταν το Web Scraping με αποτέλεσμα να μην μπορούμε να εξάγουμε όλα τα δεδομένα που χρειάζονται.

Έτσι με την χρήση του κουμπιού «Προσθήκη αρχείου JSON», ο χρήστης ανεβάζει τα δεδομένα για τον κάθε καθηγητή. Μόλις τα δεδομένα προσθεθούν, μπορούμε πολύ εύκολα να τα εμφανίσουμε στο πίνακα με τη χρήση του κουμπιού «Εμφάνιση Δημοσιεύσεων Καθηγητή».



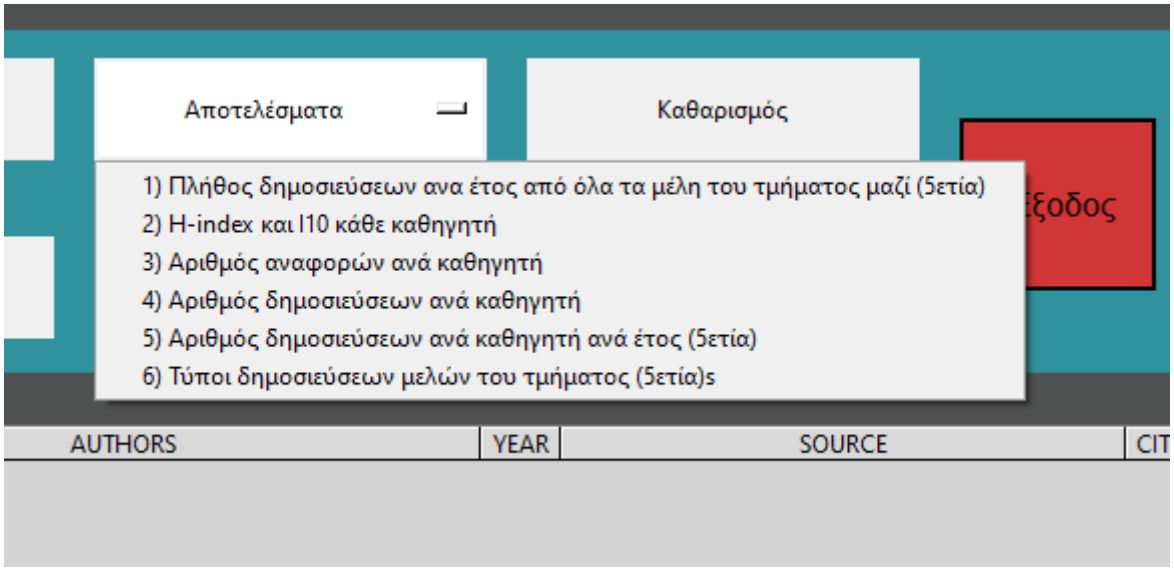
Εικόνα 8.2.11 Εμφάνιση δημοσιεύσεων καθηγητή με χρήση της εφαρμογής

Αν ο χρήστης επιθυμεί να επεξεργαστεί κάποια δημοσίευση, μπορεί απλά με τη χρήση διπλού κλικ πάνω στην δημοσίευση, να κάνει την οποιαδήποτε αλλαγή ή ακόμη και την οριστική διαγραφή της δημοσίευσης από την εφαρμογή.



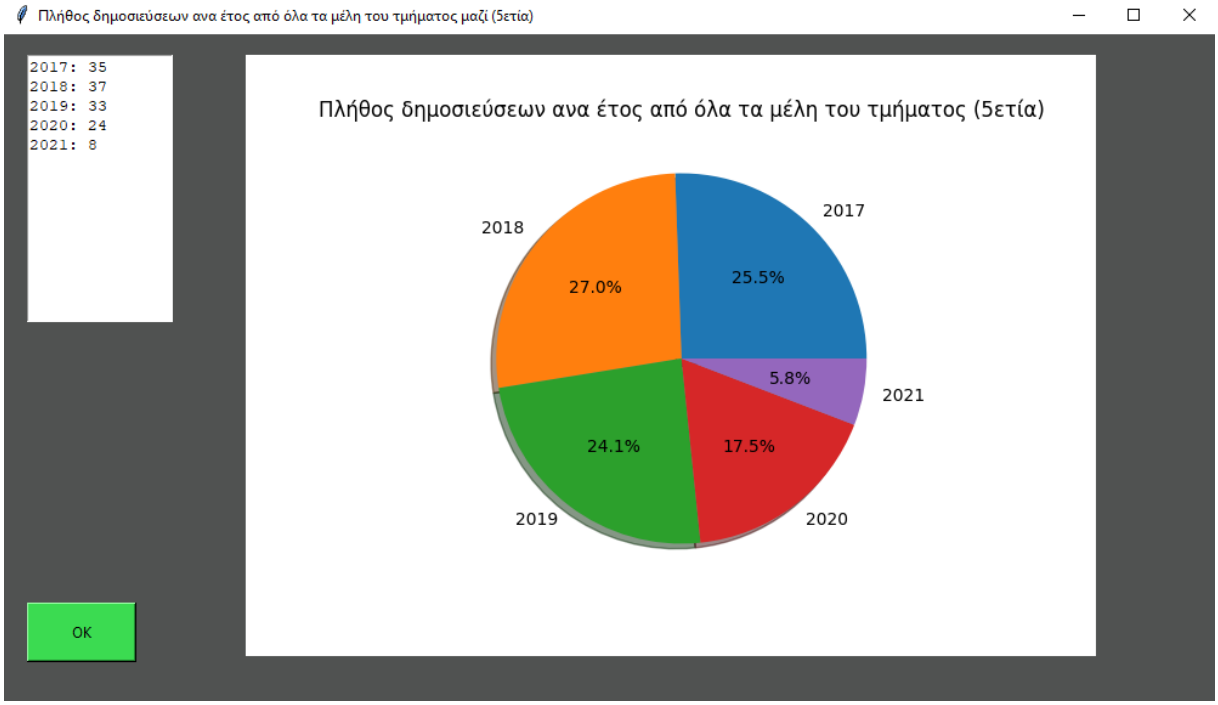
Εικόνα 8.2.12 Παράθυρο διπλού κλικ

Η λίστα με τα αποτελέσματα προσφέρει έξι επιλογές αποτελεσμάτων(Εικόνα 9.1.11).



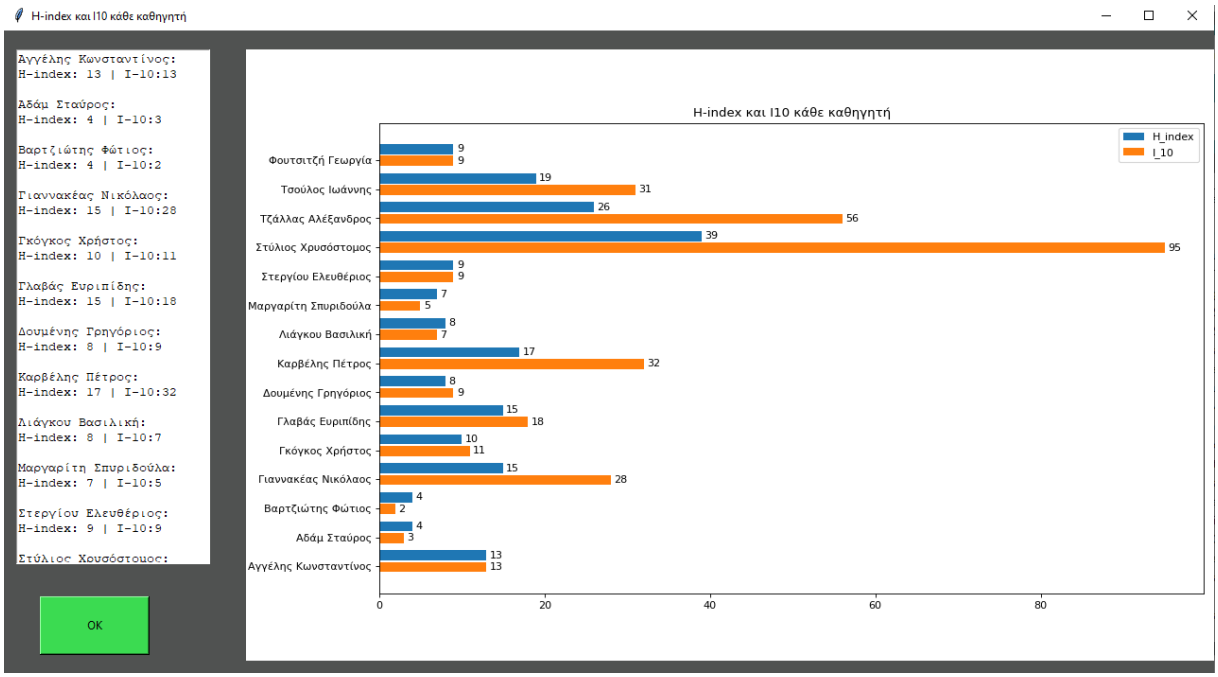
Εικόνα 8.2.13 Λίστα προβολής αποτελεσμάτων

Η πρώτη επιλογή μας εμφανίζει με μορφή πίτας το πλήθος δημοσιεύσεων ανά έτος από όλα τα μέλη του τμήματος για την τελευταία πενταετία.



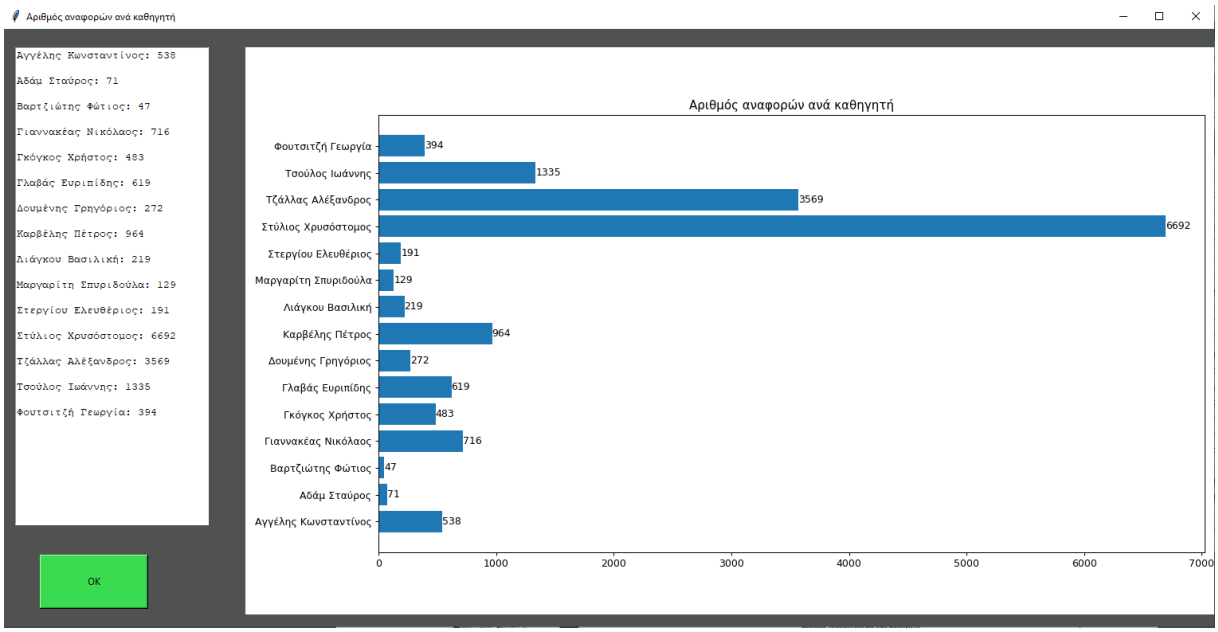
Εικόνα 8.2.14 Πλήθος δημοσιεύσεων ανά έτος από όλα τα μέλη του τμήματος(5ετία)

Η δεύτερη επιλογή εμφανίζει τις μετρικές H-index και i-10 κάθε καθηγητή όπως προκύπτουν από τις συναρτήσεις της εφαρμογής.



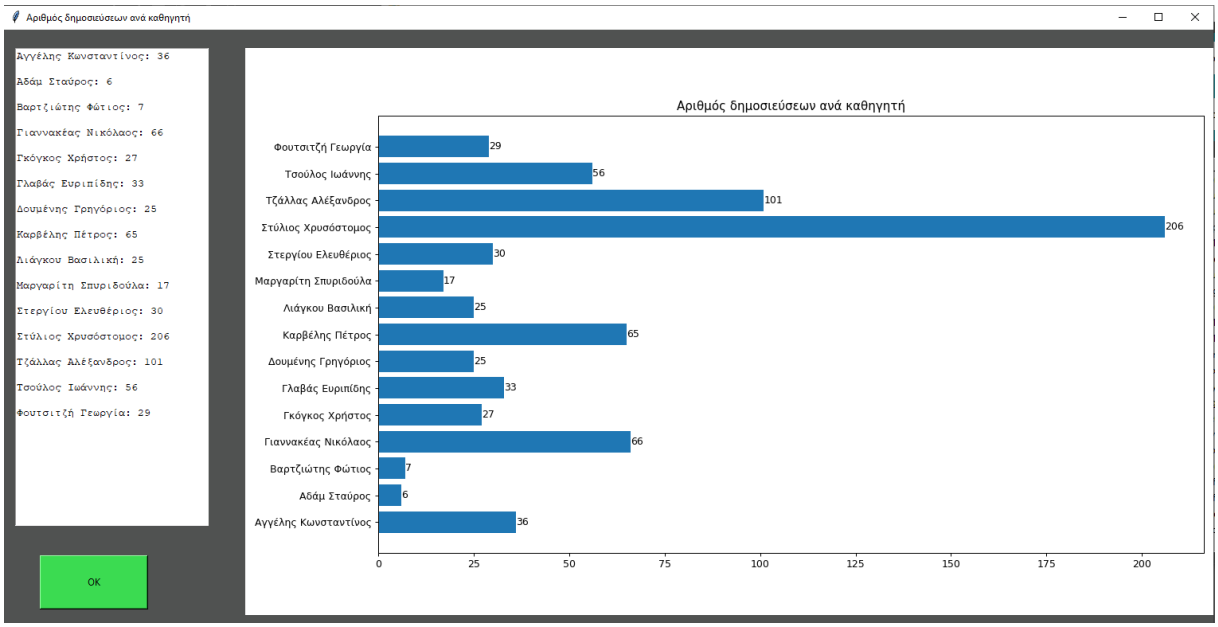
Εικόνα 8.2.15 h-index και i-10 καθηγητή

Τρίτη επιλογή είναι ο αριθμός αναφορών ανά καθηγητή.



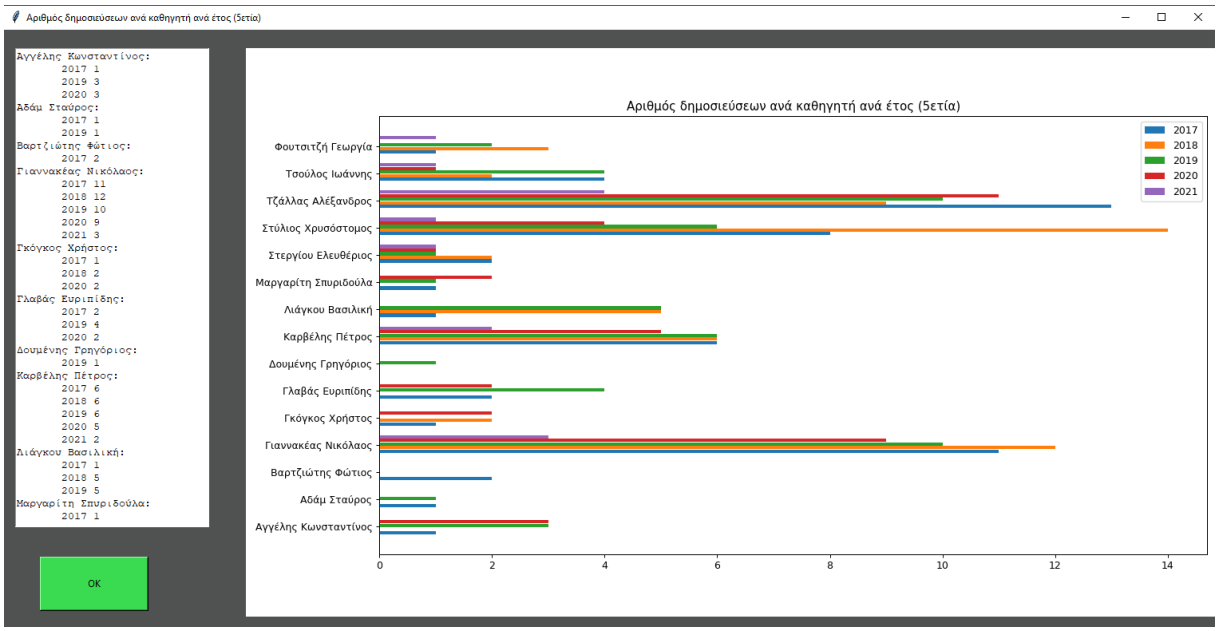
Εικόνα 8.2.16 Αριθμός αναφορών ανά καθηγητή

Τέταρτη επιλογή είναι ο αριθμός συνολικών δημοσιεύσεων ανά καθηγητή.



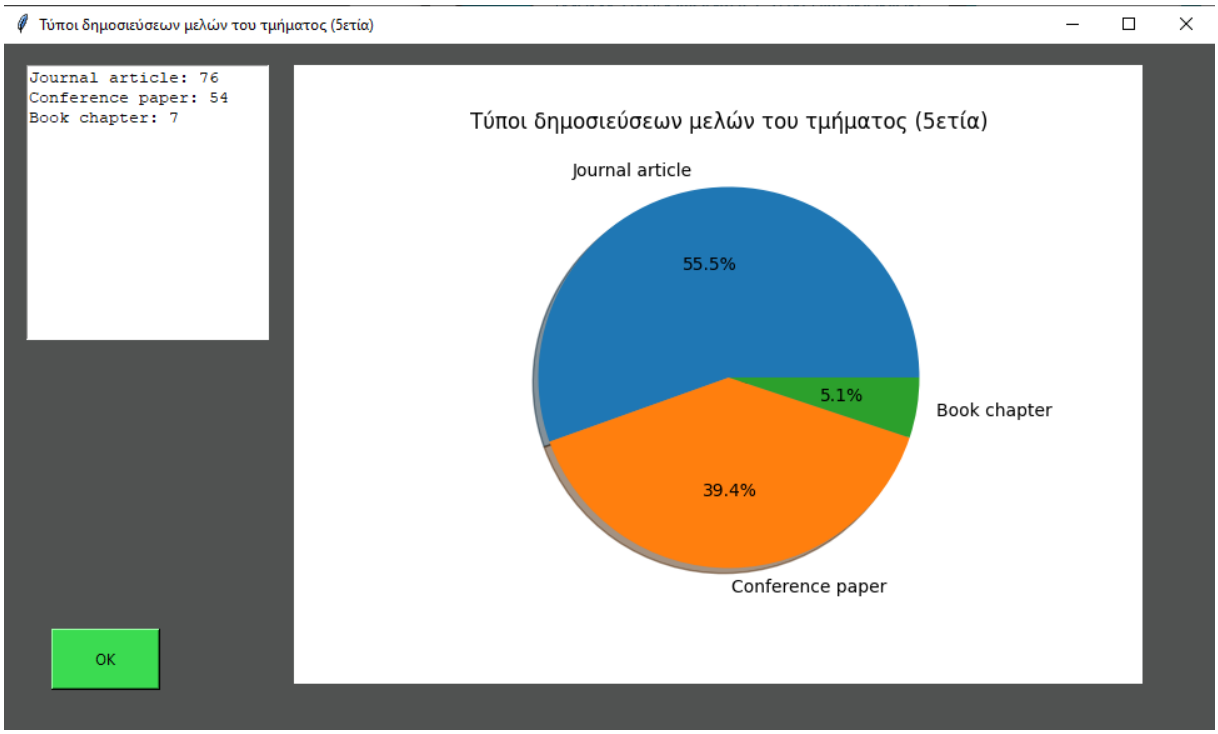
Εικόνα 8.2.17 Αριθμός δημοσιεύσεων ανά καθηγητή

Η πέμπτη επιλογή εμφανίζει τον αριθμό δημοσιεύσεων ανά καθηγητή ανά έτος για την τελευταία πενταετία.



Εικόνα 8.2.18 Αριθμός δημοσιεύσεων ανά καθηγητή ανά έτος (5ετία)

Τέλος, η έκτη επιλογή μας εμφανίζει τους τύπους των δημοσιεύσεων του κάθε καθηγητή που έχει αναρτήσει την τελευταία πενταετία.



Εικόνα 8.2.19 Τύποι δημοσιεύσεων μελών του τμήματος (5ετία)

Να σημειωθεί πως η εφαρμογή δέχεται μόνο τις δημοσιεύσεις που περιέχουν όλα τα στοιχεία που χρειάζονται για την σωστή λειτουργεία της (id, τύπο, τίτλο, συγγραφείς, έτος, πηγή, αναφορές), οπότε κάποιες δημοσιεύσεις έχουν εξαιρεθεί και από τα αποτελεσματα.

9 Αποτελέσματα

Σύμφωνα με τα αποτελέσματα που πήραμε από την εφαρμογή, προκύπτει ότι :

- Το μεγαλύτερο πλήθος δημοσιεύσεων από μέλη του τμήματος ήταν το 2018 με σύνολο 37 δημοσιεύσεις ενώ το χαμηλότερο παρατηρείται το 2021 με 19 δημοσιεύσεις καθώς το τρέχων έτος δεν έχει ολοκληρωθεί ακόμη.
- Επίσης το μεγαλύτερο h-index που εντοπίστηκε είναι 41 και το χαμηλότερο είναι 3.
- Το μεγαλύτερο I-10 βρέθηκε να είναι 102 ενώ το χαμηλότερο 2.
- Ο καθηγητής με τις περισσότερες αναφορές κατέχει 7229 συνολικές αναφορές.
- Οι περισσότερες συνολικές δημοσιεύσεις για ένα καθηγητή αριθμούν 223.
- Την τελευταία 5ετια ο πιο ενεργός καθηγητής έχει δημοσιεύσει συνολικά 38 .
- Τέλος, τα μέλη του τμήματος προτιμούν να δημοσιεύουν άρθρα σε περιοδικά με ποσοστό 55.5% σε σχέση με αλλά είδη.

Να σημειωθεί πως στις παραπάνω μετρήσεις λάβαμε υπόψη και το έτος 2022 που δεν έχει ολοκληρωθεί ακόμη.

9.1 Συμπεράσματα

Η εφαρμογή μπορεί να καλύψει τις βασικές ανάγκες ενός χρήστη και να εξάγει τα απαραίτητα αποτελέσματα ώστε να έχει μια ολοκληρωμένη εικόνα για τις δημοσιεύσεις των μελών του τμήματος. Η δημιουργία μιας τέτοια εφαρμογής δεν απαιτεί μεγάλη γνώση προγραμματισμού και η γλώσσα Python είναι κατάλληλη για απλές και γρήγορες εφαρμογές όπως η παρούσα που η απόδοση της εφαρμογής δεν είναι σημαντικός παράγοντας. Με την χρήση του tkinter έχουμε ένα καλό αποτέλεσμα για την δημιουργία ενός βασικού GUI. Αν χρειαζόμαστε περισσότερες λεπτομέρειες και τροποποιήσεις θα ήταν καλό να χρησιμοποιήσουμε κάποια άλλη βιβλιοθήκη.

9.2 Δυσκολίες και πιθανές επεκτάσεις

9.2.1 Δυσκολίες

Δυσκολίες προέκυψαν στην προσπάθεια χρήσης API για την αναζήτηση δημοσιεύσεων όπου το μόνο διαθέσιμο API για την τη χρήση (scholarly) , μετρά από μερικές προσπάθειες μπλόκαρε την IP και δεν επέτρεπε να συνεχιστεί η αναζητεί. Έτσι ο μονός τρόπος ήταν η χειροκίνητη αναζήτηση από τον χρήστη

Επίσης ένα άλλο σοβαρό πρόβλημα δημιουργήθηκε με το πρόγραμμα Publish or Perish. Μετά από την τελευταία του ενημέρωση στις 1 Νοέμβριου (Publish or Perish 8) , άλλαξε η δομή του αρχείου JSON που εξήγαμε και αφαιρέθηκε ένα σημαντικό πεδίο, το UID της δημοσίευσης. Έτσι η εφαρμογή δεν μπορούσε να αναγνωρίσει καμία δημοσίευση καθώς έλειπε ένα από τα βασικότερα πεδία. Για αυτό το λόγο η εφαρμογή στην τωρινή της μορφή υποστηρίζει μόνο την έκδοση του Publish or Perish 7.

9.2.2 Πιθανές επεκτάσεις

Η εφαρμογή θα μπορούσε να συνδέεται σε μια Online βάση δεδομένων έτσι ώστε να μπορεί να ενημερώνεται για όλους τους χρήστες.

Επίσης μια άλλη σημαντική επέκταση είναι η χρήση API για αναζήτηση δημοσιεύσεων και καθηγητών ώστε να μην χρειάζεται χειροκίνητα.

Θα μπορούσε να συμπεριλάβει και καθηγητές από αλλά τμήματα του πανεπιστήμιου.

Τέλος , ίσως η πιο βασική αλλαγή που πρέπει να γίνει στην εφαρμογή είναι η παραμετροποίηση της ώστε να μπορεί να αναγνωρίζει τις δημοσιεύσεις από το νέο Publish or Perish 8.

10 Βιβλιογραφία

Academia. 2021. Academia. [Ηλεκτρονικό] 2021. https://www.academia.edu/about.

Academic, Microsoft. Microsoft Academic. [Ηλεκτρονικό] https://academic.microsoft.com/FAQ.

C. Lee Giles, Madian Khabsa. 2014. PLOS ONE. *The Number of Scholarly Documents on the Public Web.* [Ηλεκτρονικό] 09 05 2014. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0093949.

Costa, Claire D. 2020. towardsdatascience. [Ηλεκτρονικό] 2020. https://towardsdatascience.com/top-10-python-gui-frameworks-for-developers-adca32fbe6fc.

Gusenbauer, Michael. 2019. Springer Link. *Google Scholar to overshadow them all? Comparing the sizes of 12 academic search engines and bibliographic databases.* [Ηλεκτρονικό] 2019. https://link.springer.com/article/10.1007/s11192-018-2958-5.

Hejlek, Jaroslav. help.apify.com/. [Ηλεκτρονικό] https://help.apify.com/en/articles/1961361-several-tips-on-how-to-bypass-website-anti-scraping-protections.

IBM. IBM. [Ηλεκτρονικό] https://www.ibm.com/cloud/learn/api.

imperva. imperva. [Ηλεκτρονικό] https://www.imperva.com/learn/application-security/web-scraping-attack/.

Kenny, Colm. zyte. [Ηλεκτρονικό] https://www.zyte.com/learn/what-is-web-scraping/.

Madian Khabsa, C. Lee Giles. PLOS ONE. *The Number of Scholarly Documents on the Public Web.* [Ηλεκτρονικό] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0093949.

Microsoft. 2021. Microsoft. *Next Steps for Microsoft Academic – Expanding into New Horizons.* [Ηλεκτρονικό] 04 05 2021. https://bit.ly/3zJ9Xue.

MuleSoft. MuleSoft. [Ηλεκτρονικό] https://www.mulesoft.com/resources/api/what-is-an-api.

Noorden, Richard Van. 2014. NATURE . *Online collaboration: Scientists and the social network.* [Ηλεκτρονικό] 13 08 2014. https://www.nature.com/news/online-collaboration-scientists-and-the-social-network-1.15711.

Pro, Golden Gate. 2019. [Ηλεκτρονικό] 2019. https://medium.com/@GoldenGatePro/python-libraries-data-science-bbc98c1bb148.

ResearchGate. ResearchGate. [Ηλεκτρονικό] https://www.researchgate.net/about.

RiverBank Computing. [Ηλεκτρονικό] https://riverbankcomputing.com/software/pyqt/intro.

sqlite. sqlite. [Ηλεκτρονικό] https://www.sqlite.org/index.html.

Staff, RapidAPI. 2021. rapidapi.com. [Ηλεκτρονικό] 2021. https://rapidapi.com/blog/how-to-use-an-api/.

Team, The wxPython. 2020. [Ηλεκτρονικό] 2020. https://wxpython.org.

Whittaker, Zack. 2022. techcrunch. *Web scraping is legal, US appeals court reaffirms.* [Ηλεκτρονικό] 18 04 2022. https://tcrn.ch/3NUEU2T.

wikipedia. Google Scholar. [Ηλεκτρονικό] https://el.wikipedia.org/wiki/Google_Scholar.

—. 2021. Library (computing). [Ηλεκτρονικό] 2021. https://en.wikipedia.org/wiki/Library_(computing).

Γεώργιος Σελίμης, Δημήτριος Σιμιακάκης. 2016. Αξιολόγηση της αξιοπιστίας του Google Scholar. [Ηλεκτρονικό] 22 ιουνίου 2016. http://gav.uop.gr/docs/thesis/BSc2016SelimisSimiakakis.pdf.

Διέτης, Νικόλας. 2017. 3 ΒΗΜΑΤΑ..μέχρι την πρώτη δημοσίευση. [Ηλεκτρονικό] 2017. https://www.ucy.ac.cy/ctl/documents/KEDIMA/TaxyrrythmaFall2017-18/3_steps_to_Publication.pdf.

ΖΑΧΑΡΙΑΣ, ΠΑΥΛΑΤΟΣ. 2020. Σχεδιασμός και ανάπτυξη συστήματος παραγγελιών για εστιατόριο. [Ηλεκτρονικό] 2020. https://bit.ly/3dp36tq.

ΜΟΣΧΟΣ, ΑΧΙΛΛΕΥΣ. 2015. Διαδραστική-πολυμεσική αναπαράσταση. [Ηλεκτρονικό] 2015. https://ikee.lib.auth.gr/record/280036/files/Moschos_2075.pdf.

ΤΑ ΕΙΔΗ ΤΗΣ ΕΠΙΣΤΗΜΟΝΙΚΗΣ ΕΡΓΑΣΙΑΣ. [Ηλεκτρονικό] https://www.nup.ac.cy/wp-content/uploads/2020/09/Type-Writings.pdf.

ΤΙΦΤΙΚΙΔΗΣ, ΠΑΥΛΟΣ. 2020. ΕΦΑΡΜΟΓΗ ΜΗΤΡΩΟΥ ΕΠΙΣΤΗΜΟΝΙΚΩΝ ΔΗΜΟΣΙΕΥΣΕΩΝ ΜΕΛΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ. [Ηλεκτρονικό] 2020. https://bit.ly/3dp2Ri0.

Χρήστος, Γκόγκος. 2015. eclass.teiep. *Αξιολόγηση ερευνητικού έργου με χρήση βιβλιομετρικών δεικτών.* [Ηλεκτρονικό] 07 10 2015. https://bit.ly/39tyngA.