



ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΠΛΗΡΟΦΟΡΙΑΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΗ ΕΞΑΓΩΓΗ ΣΤΟΙΧΕΙΩΝ
ΔΗΜΟΣΙΕΥΣΕΩΝ ΑΠΟ ΤΗΝ ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΒΑΣΗ
SCOPUS

Πτυχιακή Εργασία

Του

Ζήση Δημητρίου

Θεσσαλονίκη, Ιούνιος 2020

Περίληψη

Σκοπός της παρούσας εργασίας είναι η παροχή αυτοματοποιημένης σάρωσης της βιβλιογραφικής βάσης Scopus, με αφορμή την εισήγηση της Επιτροπής Ερευνών και Διαχείρισης του ΕΛΚΕ στις 12 Ιουνίου 2019 για την ενεργοποίηση του θεσμού των ερευνητικών βραβείων. Μέσω του συστήματος αυτού, θα γίνεται ο (αυτόματος) εντοπισμός των δημοσιεύσεων των ερευνητών του Πανεπιστημίου Μακεδονίας. Από εκπαιδευτική οπτική, στόχος είναι η όσο το δυνατόν μεγαλύτερη εξοικείωση με τεχνολογίες, όπως η Python, Selenium, SQLite που χρησιμοποιήθηκαν.

Κατά την βιβλιογραφική ανασκόπηση, θα παρουσιαστούν διάφοροι βιβλιομετρικοί δείκτες, οι οποίοι χρησιμοποιούνται, γενικά, για την αξιολόγηση της ερευνητικής απόδοσης.

Θα παρουσιαστούν, επίσης, οι χρησιμοποιούμενες τεχνολογίες του συστήματος, καθώς και η δομή και αρχιτεκτονική του. Για να γίνει πιο κατανοητή η δομή, θα παρουσιαστούν, επιπλέον, κομμάτια του κώδικα του συστήματος.

Τέλος, θα γίνει και μία παρουσίαση της λειτουργικότητας της εφαρμογής, χρησιμοποιώντας, κυρίως, εικόνες (screenshots) από το ίδιο την ίδια την εφαρμογή.

Λέξεις Κλειδιά : Βιβλιομετρία, Scopus, Bibliometrics, Python, Selenium, PyQt5, SQLite

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα καθηγητή μου κ. Χατζηγεωργίου Αλέξανδρο για την στήριξη και εμπιστοσύνη που έδειξε, καθώς και για την άριστη συνεργασία που είχα μαζί του σε όλη την διάρκεια των σπουδών μου στο τμήμα Εφαρμοσμένης Πληροφορικής.

Επίσης, θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Κωνσταντίνο Μαργαρίτη, για την αρωγή του καθ' όλη την διάρκεια εκπόνησης της εργασίας.

Τέλος, θέλω να ευχαριστήσω θερμά την οικογένειά μου, η οποία με βοηθά και με στηρίζει σε όλα τα χρόνια της ζωής μου, κάνοντας αμέτρητες θυσίες.

Περιεχόμενα

Περίληψη.....	
Ευχαριστίες.....	
1 Εισαγωγή.....	1
1.1 Αντικείμενο της πτυχιακής εργασίας	1
1.2 Διάρθρωση της μελέτης.....	1
2 Βιβλιογραφική Ανασκόπηση	2
2.1 Βιβλιομετρία.....	2
2.2 Δείκτες μέτρησης ποιότητας επιστημονικών περιοδικών	2
2.2.1 Ποσοτικοί βιβλιομετρικοί δείκτες.....	2
2.2.2 Ποιοτικοί βιβλιομετρικοί δείκτες	5
3 Τεχνολογίες Ανάπτυξης.....	6
3.1 Python.....	6
3.1.1 Γενικά για την Python	6
3.1.2 Ιστορία.....	7
3.1.3 Μερικά Χαρακτηριστικά.....	7
3.2 Selenium Framework.....	9
3.2.1 Γενικά	9
3.2.2 Ιστορία.....	9
3.2.3 Χαρακτηριστικά του Selenium.....	10
3.3 Qt Framework.....	10
3.3.1 Ιστορική Αναδρομή.....	11
3.3.2 PyQt.....	11
3.3.3 Qt Designer.....	12
3.4 Βιβλιοθήκη Pandas.....	12
3.5 Βιβλιοθήκη Matplotlib	13
3.6 SQL	13
3.6.1 SQLite	13
3.6.2 Python-SocketIO	14
4 Σχεδιασμός και Υλοποίηση Συστήματος.....	15
4.1 Αρχιτεκτονική Συστήματος.....	15
4.1.1 Εφαρμογή Πελάτη	16
4.1.2 Τοπική Βάση Δεδομένων της Εφαρμογής Πελάτη	16
4.1.3 Σύνδεση Εφαρμογής Πελάτη με Βάση Δεδομένων.....	19
4.1.4 Διακομιστής Συστήματος.....	23

4.1.5	Selenium WebDriver	23
4.2	Λειτουργίες Εφαρμογής Πελάτη (ScopusAnalyzer)	24
4.2.1	Εκτέλεση Παραμετροποιημένης Αναζήτησης στην βιβλιογραφική βάση Scopus	24
4.2.2	Εξαγωγή Στατιστικών από Υπάρχοντα Αρχεία Excel.....	25
4.2.3	Εξαγωγή Στατιστικών από την Βάση Δεδομένων της Εφαρμογής	25
4.2.4	Εισαγωγή στοιχείων στη Βάση Δεδομένων από Εξωτερικά Αρχεία.....	25
4.2.5	Διαγραφή στοιχείων από την Βάση Δεδομένων.....	26
4.3	Κώδικας.....	26
4.3.1	Κώδικας Εφαρμογής Πελάτη (ScopusAnalyzer)	26
4.3.2	Κώδικας Διακομιστή (Server).....	41
5	Επίδειξη Λειτουργικότητας Συστήματος	53
5.1	Παραμετροποιημένη Αναζήτηση	53
5.1.1	Εκτέλεση Παραμετροποιημένης Αναζήτησης.....	53
5.1.2	Αποτελέσματα Παραμετροποιημένης αναζήτησης.....	55
5.2	Εξαγωγή Στατιστικών από Εξωτερικά Αρχεία.....	58
5.3	Βάση Δεδομένων.....	69
5.3.1	Διαγραφή Στοιχείων	69
5.3.2	Εισαγωγή στοιχείων από εξωτερικά αρχεία.....	70
5.3.3	Εξαγωγή Στατιστικών από Στοιχεία της Βάσης Δεδομένων	70
6	Επίλογος	72
6.1	Συμπεράσματα.....	72
6.2	Μελλοντικές Επεκτάσεις.....	72
	Βιβλιογραφία	74

Περιεχόμενα Εικόνων

Εικόνα 3-1:	Λογότυπο Python.....	7
Εικόνα 3-2:	Μεταφερσιμότητα Python	9
Εικόνα 3-3:	Λογότυπο Selenium.....	9
Εικόνα 3-4:	Λογότυπο Qt.....	11
Εικόνα 3-5:	Λογότυπο pandas.....	12
Εικόνα 3-6:	Λογότυπο matplotlib.....	13
Εικόνα 3-7:	Λογότυπο SQLite	14
Εικόνα 4-1:	Αρχιτεκτονική του Συστήματος	15
Εικόνα 4-2:	Διάγραμμα Οντοτήτων Συσχετίσεων της Τοπικής Βάσης Δεδομένων	18
Εικόνα 5-1:	Μενού Εφαρμογής Πελάτη.....	53
Εικόνα 5-2:	Σάρωση εγγράφων.....	54
Εικόνα 5-3:	Ολοκλήρωση της Αναζήτησης.....	54
Εικόνα 5-4:	Αποτελέσματα Αναζήτησης	55

Εικόνα 5-5: Αποτελέσματα Αναζήτησης στο Excel	55
Εικόνα 5-6: Αριθμός Δημοσιεύσεων Ανά Κατηγορία	56
Εικόνα 5-7: Στατιστικά Καθηγητών Ανά Τμήμα	56
Εικόνα 5-8: Καρτέλα Βάσης Δεδομένων	57
Εικόνα 5-9: Εισαγωγή Αρχείων 1	58
Εικόνα 5-10: Εισαγωγή Αρχείων 2	59
Εικόνα 5-11: Αρχεία που Εισήχθησαν	59
Εικόνα 5-12: Ρυθμίσεις Εξαγωγής Στατιστικών	60
Εικόνα 5-13: Επιλογή Καθηγητών για Εξαγωγή Στατιστικών τους	61
Εικόνα 5-14: Συγκεντρωτικά Δεδομένα.....	62
Εικόνα 5-15: Αριθμός Δημοσιεύσεων Ανά Χρονιά	62
Εικόνα 5-16: Αριθμός Top-10 Περιοδικών Ανά χρονιά	63
Εικόνα 5-17: Μέσος όρος της Τιμής του CiteScore Ανά Χρονιά	64
Εικόνα 5-18: Μέσο Average Percentile Ανά Χρονιά.....	65
Εικόνα 5-19: Μέγιστη τιμή του Δείκτη CiteScore Ανά Χρονιά	66
Εικόνα 5-20: Μέσος Όρος Average Percentile Καθηγητή Ανά Χρονιά 1	67
Εικόνα 5-21: Μέσος Όρος Average Percentile Καθηγητή Ανά Χρονιά 2	67
Εικόνα 5-22: Μέσος Όρος Average Percentile Καθηγητή Ανά Χρονιά 3	68
Εικόνα 5-23: Στατιστικά Καθηγητών Ανά Τμήμα	68
Εικόνα 5-24: Διαγραφή Στοιχείων από την Βάση Δεδομένων.....	69
Εικόνα 5-25: Εισαγωγή Στοιχείων στην Βάση Δεδομένων από Εξωτερικά Αρχεία	70
Εικόνα 5-26: Ρυθμίσεις Εξαγωγής Στατιστικών	71

1 Εισαγωγή

Στο παρόν κεφάλαιο γίνεται μια μικρή παρουσίαση του αντικειμένου της εργασίας, της δομής αυτού και των σκοπών του. Ακόμη, παρουσιάζεται η σύνοψη των επόμενων κεφαλαίων.

1.1 Αντικείμενο της πτυχιακής εργασίας

Αντικείμενο της συγκεκριμένης πτυχιακής αποτελεί η ανάπτυξη λογισμικού αυτόματης σάρωσης της βιβλιογραφικής βάσης Scopus, σύμφωνα με τη μεθοδολογία που προκύπτει από την εισήγηση της Επιτροπής Ερευνών και Διαχείρισης του ΕΛΚΕ στις **12 Ιουνίου 2019** για την ενεργοποίηση του θεσμού των ερευνητικών βραβείων. Στόχος της εργασίας είναι η διευκόλυνση του εντοπισμού των δημοσιεύσεων μελών του Πανεπιστημίου Μακεδονίας σε επιστημονικά περιοδικά, για ένα ημερολογιακό έτος στα θεματικά πεδία της βάσης Scopus.

Για την δημιουργία της συγκεκριμένης εφαρμογής έγινε χρήση της προγραμματιστικής γλώσσας Python (έκδοση 3.7), σε συνδυασμό με διάφορες βιβλιοθήκες ανοιχτού κώδικα.

1.2 Διάρθρωση της μελέτης

Η παρούσα εργασία περιλαμβάνει έξι κεφάλαια. Στο πρώτο κεφάλαιο γίνεται μια περιγραφή του αντικειμένου της εργασίας, καθώς και ο σκοπός εκπόνησής της. Στο δεύτερο κεφάλαιο πραγματοποιείται μια βιβλιογραφική ανασκόπηση στην Βιβλιομετρία, καθώς και στους διάφορους αριθμητικούς δείκτες, οι οποίοι χρησιμοποιούνται για την αξιολόγηση της ποιότητας των ακαδημαϊκών δημοσιεύσεων επιστημόνων. Στο τρίτο κεφάλαιο παρουσιάζονται οι τεχνολογίες, οι οποίες χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής (ScopusAnalyzer) αυτόματης σάρωσης της πλατφόρμας του Scopus, καθώς και του διακομιστή που εξυπηρετεί τα αιτήματα της εφαρμογής. Στο τέταρτο κεφάλαιο γίνεται μια περιγραφή της υλοποίησης του συστήματος. Στο πέμπτο κεφάλαιο γίνεται μια επίδειξη της λειτουργικότητας της εφαρμογής. Τέλος, στο έκτο κεφάλαιο εξάγονται και καταγράφονται τα συμπεράσματα της συγκεκριμένης πτυχιακής εργασίας, όπως και πιθανές μελλοντικές επεκτάσεις.

2 Βιβλιογραφική Ανασκόπηση

Στο παρόν κεφάλαιο πραγματοποιείται μια βιβλιογραφική ανασκόπηση επί του θεωρητικού υποβάθρου. Παρουσιάζεται ο όρος της [Βιβλιομετρίας](#), καθώς και ορισμένοι [ποσοτικοί](#) και [ποιοτικοί](#) βιβλιομετρικοί δείκτες (μέτρησης ποιότητας ερευνητικού έργου).

2.1 Βιβλιομετρία

Ως **Βιβλιομετρία** (Bibliometrics) ορίζεται ένα σύνολο μεθόδων, οι οποίες δημιουργήθηκαν προκειμένου να αξιολογείται η ερευνητική ποιότητα και επίδοση των επιστημόνων. Οι μέθοδοι αυτές, βασίζονται σε αντικειμενικά δεδομένα δημοσιεύσεων και στις βιβλιογραφικές αναφορές. Η ποσοτικοποίηση της ερευνητικής ποιότητας γίνεται με την χρήση βιβλιομετρικών δεικτών. Βιβλιομετρικοί δείκτες είναι τα αποτελέσματα που εξάγονται μέσω μιας βιβλιομετρικής ανάλυσης και σήμερα χρησιμοποιούνται ευρέως από τις διάφορες βάσεις επιστημονικών περιοδικών.

2.2 Δείκτες μέτρησης ποιότητας επιστημονικών περιοδικών

Όπως αναφέρθηκε και παραπάνω, οι βιβλιομετρικοί δείκτες χρησιμοποιούνται ευρέως για την αξιολόγηση της ερευνητικής απόδοσης. Το γεγονός αυτό έχει αυξήσει την σημαντικότητά τους τα τελευταία χρόνια και γι' αυτό γίνεται συνεχώς προσπάθεια ανάπτυξης νέων δεικτών, οι οποίοι υπόσχονται υψηλότερα επίπεδα μεροληψίας. Αυτοί είναι οι λεγόμενοι βιβλιομετρικοί δείκτες δεύτερης γενιάς ή ποιοτικοί δείκτες, ενώ οι παλιότεροι πιο βασικοί δείκτες ονομάζονται ποσοτικοί.

2.2.1 Ποσοτικοί βιβλιομετρικοί δείκτες

2.2.1.1 Συντελεστής Απήχησης (*Impact Factor*)

Ο συγκεκριμένος δείκτης είναι από τους πιο ευρέως γνωστούς δείκτες μέτρησης επιστημονικών άρθρων. Δοσμένου ενός χρόνου, ο Impact Factor (IF) υπολογίζεται ως ο λόγος των συνολικών αναφορών που έλαβε ο επιστήμονας στο δοσμένο έτος, τα άρθρα της προηγούμενης διετίας δια του συνολικού αριθμού των άρθρων αυτών.

Αναλυτικά ο τύπος:

$$Impact\ Factor_y = \frac{Citations_{y-1} + Citations_{y-2}}{Publications_{y-1} + Publications_{y-2}}$$

όπου y ο δοσμένος χρόνος.

Ο συγκεκριμένος δείκτης, γενικά, δεν θεωρείται αντικειμενικός, καθώς όπως μπορούμε να διακρίνουμε, λαμβάνει υπόψη του μόνο τον αριθμό των αναφορών. Ένας μεγάλος αριθμός αναφορών δεν εγγυάται (υποχρεωτικά) ποιότητα έργου.

2.2.1.2 *h-Index*

Ο δείκτης h ή αλλιώς δείκτης Hirsch, προτάθηκε το 2005 στις ΗΠΑ από τον καθηγητή Jorge Hirsch. Ο δείκτης h είναι ίσος με τον αριθμό των άρθρων που έχει δημοσιεύσει ένας επιστήμονας, τα οποία το καθένα έχει τουλάχιστον h αναφορές από άλλους ερευνητές. Ο συγκεκριμένος δείκτης, σε αντίθεση με τον Impact Factor, συνδυάζει παραγωγικότητα και δημοφιλία.

Στα μειονεκτήματα του h -index θα μπορούσαμε να αναφέρουμε ότι συνδέεται στενά με τον χρόνο που τα άρθρα δημοσιεύονται, με αποτέλεσμα να αδικεί τα προσφάτως δημοσιευμένα άρθρα. Επίσης, η τιμή του δείκτη h , περιορίζεται από τον αριθμό των δημοσιεύσεων. Για παράδειγμα, ένας επιστήμονας με λίγες, αλλά ποιοτικές δημοσιεύσεις θα είχε πολύ χαμηλό δείκτη h .

2.2.1.3 *m-index*

Ο δείκτης m ή αλλιώς m -quotient, είναι μια παραλλαγή του h -index που αναφέρθηκε παραπάνω. Ένα από τα μειονεκτήματα που αναφέραμε στον h -index είναι το γεγονός ότι μεροληπτεί απέναντι σε προσφάτως δημοσιευμένα άρθρα. Αυτό έρχεται να το διορθώσει ο m -index, ο οποίος υπολογίζεται ως εξής:

$$m - index = \frac{h}{n}$$

όπου h είναι ο h -index και n είναι ο αριθμός των χρόνων που πέρασαν από την πρώτη δημοσίευση του επιστήμονα.

Στα μειονεκτήματα του συγκεκριμένου δείκτη, μπορούμε να αναφέρουμε το γεγονός ότι η τιμή του μπορεί να αλλάξει σε μεγάλο βαθμό διαχρονικά και ότι ο ίδιος ο δείκτης μεροληπτεί απέναντι σε επιστήμονες που διακόπτουν προσωρινά την ακαδημαϊκή τους καριέρα.

2.2.1.4 Μέσος Αριθμός Αναφορών ανά Δημοσίευση (Citations Per Paper)

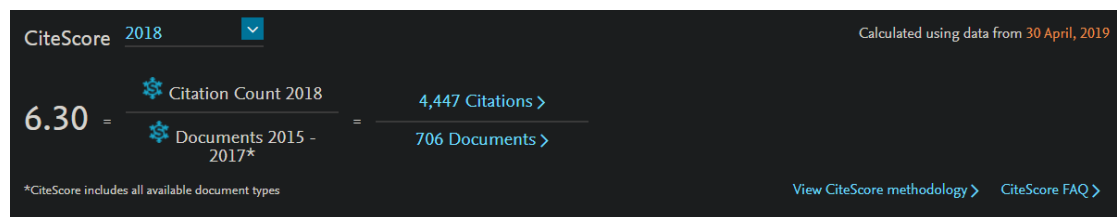
Ο CPP είναι μια μετρική, η οποία αντικατοπτρίζει την απήχηση των δημοσιεύσεων στην επιστημονική κοινότητα, μέσω των αναφορών που λαμβάνουν. Υπολογίζεται ως ο λόγος του συνολικού αριθμού των αναφορών (σε μια ορισμένη χρονική περίοδο) που έλαβαν τα άρθρα προς τον συνολικό αριθμό των άρθρων.

$$CPP = \frac{\text{Total \# of Citations}}{\text{Total \# of Papers}}$$

Είναι πολύ απλός στον υπολογισμό (ένα πηλίκο), αλλά έχει πολλά μειονεκτήματα, καθώς δεν υπάρχει κάποια μέθοδος στάθμισης (πχ διαγραφή των αυτοαναφορών, διάκριση του κύρους των περιοδικών απ' όπου προέρχονται οι αναφορές κλπ).

2.2.1.5 Cite Score

Ο συγκεκριμένος δείκτης παρέχεται μόνο από την βιβλιογραφική βάση του Scopus, την οποία χρησιμοποιήσαμε για τον εντοπισμό των δημοσιεύσεων. Ο συγκεκριμένος δείκτης υπολογίζει τον μέσο αριθμό αναφορών που έχουν ληφθεί σε ένα ημερολογιακό έτος της επιλογής μας, από όλα τα άρθρα του περιοδικού που έχουν δημοσιευτεί κατά τα προηγούμενα τρία έτη.



Υπολογισμός του CiteScore

Πηγή: <https://www.scopus.com/sources>, 23/11/2019

2.2.2 Ποιοτικοί βιβλιομετρικοί δείκτες

2.2.2.1 *CImago Journal Rank (SJR)*

Ο δείκτης SJR είναι ένας ποιοτικός βιβλιομετρικός δείκτης, ο οποίος παρέχεται αποκλειστικά από την βιβλιογραφική βάση Scopus. Λαμβάνει υπόψη τον αριθμό των αναφορών, τις οποίες δέχεται ένα περιοδικό, όπως επίσης και το κύρος και την θεματική ενότητα των περιοδικών από όπου προέρχονται οι αναφορές αυτές.

Ο υπολογισμός του SJR δεν είναι τόσο απλός, όσο των δεικτών που αναφέρθηκαν ως τώρα, καθώς δεν αποτελεί έναν μαθηματικό τύπο. Για την ακρίβεια, είναι ένας επαναληπτικός αλγόριθμος ο οποίος ολοκληρώνεται όταν υπάρξει σταθερή λύση. Αξίζει να αναφερθεί ότι μοιάζει πολύ με τον αλγόριθμο PageRank (στην βασική του μορφή) που χρησιμοποιεί η Google, ώστε να ξεχωρίζει την σημαντικότητα των ιστοσελίδων ιστού.

2.2.2.2 *Source Normalized Impact per Paper (SNIP)*

Ο δείκτης SNIP είναι μια μετρική, η οποία έχει ως στόχο να διευκολύνει την σύγκριση πηγών, που ανήκουν όμως σε διαφορετικά θεματικά πεδία. Ορίζεται ως ο λόγος της ακατέργαστης επίδρασης ανά δημοσίευση που δημοσιεύεται στο περιοδικό (RIP, Raw Impact per Publication) και της συγκριτικής δυναμικής παραπομπών της βάσης δεδομένων (RDCP, Relative Database Citation Potential):

$$SNIP = \frac{RIP}{RDCP}$$

Ο δείκτης SNIP, μεταξύ άλλων, δίνει και την δυνατότητα να προσδιοριστούν τα περιοδικά, τα οποία έχουν μεγαλύτερη απόδοση σε ένα συγκεκριμένο θεματικό πεδίο.

Ο SNIP αποτελεί, επίσης, άλλη μια μετρική, η οποία χρησιμοποιείται ευρέως από την βιβλιογραφική βάση Scopus.

2.2.2.3 *Eigenfactor Score*

Ο συγκεκριμένος δείκτης χρησιμοποιεί στατιστικά δεδομένα από βιβλιογραφικές αναφορές, ώστε να είναι σε θέση να αξιολογήσει την επιρροή μιας πηγής. Λαμβάνει υπόψη τα άρθρα, τα οποία είναι δημοσιευμένα την τελευταία πενταετία και τον αριθμό των

αναφορών που έγιναν σε ένα έτος επιλογής. Βασικό πλεονέκτημα της μετρικής αυτής είναι ότι αποκλείει τις αυτοαναφορές, που αυξάνει το επίπεδο αμεροληψίας. Επίσης, στον συγκεκριμένο δείκτη, τα περιοδικά που αναφέρονται περισσότερες φορές, επηρεάζουν σε μεγαλύτερο βαθμό την τιμή του δείκτη, απ' ό,τι τα υπόλοιπα.

2.2.2.4 Δείκτης Επιρροής Άρθρου (Article Influence Score)

Ο Δείκτης Επιρροής Άρθρου είναι μετρική, η οποία ορίζεται ως ο λόγος του γινομένου της τιμής του Eigenfactor Score (αναφέρθηκε παραπάνω) του περιοδικού επί 0.01 και του λόγου του αριθμού των άρθρων ενός περιοδικού σε διάστημα πέντε ετών με τον συνολικό αριθμό των άρθρων του ίδιο χρονικού διαστήματος.

$$\text{Article Influence Score} = 0.01 * \frac{\text{EigenFactor Score}}{x}$$

Όπου x είναι ο λόγος του αριθμού των άρθρων ενός περιοδικού σε διάστημα πέντε ετών με τον συνολικό αριθμό των άρθρων του ίδιο χρονικού διαστήματος.

3 Τεχνολογίες Ανάπτυξης

Στο παρόν κεφάλαιο πραγματοποιείται μια μικρή παρουσίαση των τεχνολογιών ανάπτυξης της εφαρμογής που χρησιμοποιήθηκαν. Παρουσιάζεται η γλώσσα προγραμματισμού Python, η βιβλιοθήκη Selenium, η βιβλιοθήκη γραφικής διασύνδεσης PyQt και η βιβλιοθήκη παραγωγής στατιστικών της Python, Pandas.

3.1 Python

3.1.1 Γενικά για την Python

Η Python αποτελεί μια υψηλού επιπέδου γλώσσα προγραμματισμού γενικού σκοπού, η οποία δημιουργήθηκε από τον Ολλανδό προγραμματιστή Guido van Rossum την δεκαετία του 1990. Είναι γνωστή για την αποδοτικότητα, αλλά και την αναγνωσιμότητα του κώδικα, όπως και για το γεγονός ότι επιτρέπει την χρήση διαφόρων ειδών προγραμματισμού, όπως αντικειμενοστραφή και συναρτησιακό. Η Python διακρίνεται για τον υψηλό αριθμό συμβατών βιβλιοθηκών, οι οποίες συμβάλλουν στην διευκόλυνση διαφόρων εργασιών. Σημαντικό χαρακτηριστικό της συγκεκριμένης γλώσσα προγραμματισμού, είναι η δυνατότητα εκτέλεσης του κώδικα σε διαφορετικά λειτουργικά συστήματα.



Εικόνα 3-1: Λογότυπο Python

Πηγή: python.org

3.1.2 Ιστορία

Η Python κυκλοφόρησε, αρχικά, το 1990, σχεδιασμένη για να χρησιμοποιείται σε συγκεκριμένο λειτουργικό σύστημα (Amoeba). Η έκδοση 2.0 της Python κυκλοφόρησε το ημερολογιακό έτος 2000, ενώ η 3^η έκδοση το 2008. Η καινοτομία που εισήγαγε η Python, είναι ότι έσπασε την προς τα πίσω συμβατότητα (ο κώδικας γραμμένος σε Python3 **δεν** είναι συμβατός με διερμηνευτή της Python2), ώστε να διορθώσει μια μεγάλη γκάμα λαθών και να εισάγει νέες απλουστεύσεις στον κώδικα.

3.1.3 Μερικά Χαρακτηριστικά

3.1.3.1 Απλότητα

Η Python θεωρείται μια μινιμαλιστική γλώσσα. Θα μπορούσαμε να πούμε ότι έχει πολλά κοινά χαρακτηριστικά με ψευδοκώδικα (pseudocode). Σε αρκετές περιπτώσεις χρησιμοποιεί γνωστές λέξεις της Αγγλικής γλώσσας. Χαρακτηριστικό παράδειγμα η εντολή εκτύπωσης *print*.

Επίσης, σε αντίθεση με τις περισσότερες γλώσσες υψηλού επιπέδου, η Python δεν χρησιμοποιεί παρενθέσεις ή άλλα σύμβολα (όπως αγκύλες, άγκιστρα κλπ), αλλά συνηθίζεται ο διαχωρισμός των συντακτικών δομών με κενά. Συνεπώς, τα whitespaces είναι κάτι το οποίο ο προγραμματιστής πρέπει να προσέχει.

3.1.3.2 Αναγνωσιμότητα Κώδικα

Μια άλλη καινοτομία της Python είναι το γεγονός ότι δεν αγνοεί την στοίχιση. Για παράδειγμα, όταν ο προγραμματιστής θελήσει να γράψει μια δομή επιλογής (if), θα πρέπει ο κώδικας που θα γράψει να είναι στοιχισμένος ανάλογα. Παράδειγμα:

```
if statement:  
    execute_some_code()
```

Στοιχισμένος Κώδικας Python

Αν δεν είναι στοιχισμένος:

```
if statement:  
execute_some_code()
```

Μη στοιχισμένος Κώδικας Python

Δε μπορεί να εκτελεστεί, ο διερμηνευτής το αναγνωρίζει ως λάθος. Γι' αυτό, όπως ήδη αναφέραμε, δεν υπάρχουν σύμβολα που συναντάμε σε άλλες γλώσσες. Ουσιαστικά, η Python «αναγκάζει» τον προγραμματιστή να παράγει αναγνώσιμο κώδικα.

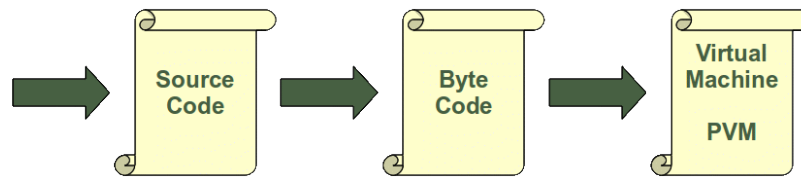
3.1.3.3 Ανοιχτού Κώδικα

Η Python είναι μια γλώσσα ανοιχτού κώδικα, το οποίο σημαίνει ότι οποιοσδήποτε επιθυμεί, έχει την δυνατότητα να διαβάσει τον πηγαίο κώδικα της, να κάνει αλλαγές σε αυτόν και να χρησιμοποιηθεί αυτούσιος σε άλλο project. Η Python είναι μια γλώσσα, η οποία βελτιώνεται και επεκτείνεται συνεχώς μέσα από μια κοινότητα.

3.1.3.4 Μεταφερσιμότητα

Όλα τα προγράμματα γραμμένα σε Python μπορούν να εκτελεστούν σε διαφορετικά λειτουργικά συστήματα (πχ Windows, Linux). Ίσως, βέβαια, να χρειαστούν μερικές αλλαγές σε παραμέτρους βιβλιοθηκών, αλλά γενικά η Python δεν εξαρτάται από συγκεκριμένο λειτουργικό σύστημα.

Τα αρχεία πηγαίου κώδικα Python μεταγλωττίζονται σε αρχεία bytecode, τα οποία είναι ανεξάρτητα μηχανής και μπορούν να εκτελεστούν από μία εικονική μηχανή.



Εικόνα 3-2: Μεταφερσιμότητα Python

Πηγή Εικόνας: python-course.eu

3.2 Selenium Framework

3.2.1 Γενικά

Η Selenium είναι ένα framework, το οποίο περιέχει ένα σύνολο από εργαλεία λογισμικού, τα οποία υποστηρίζουν την αυτοματοποιημένη δοκιμή. Το συγκεκριμένο framework, κατά κύριο λόγο, χρησιμοποιείται για τις ανάγκες των δοκιμών web εφαρμογών. Η γλώσσα προγραμματισμού, με την οποία δημιουργήθηκε είναι η Java.

Με το Selenium, έχουμε την δυνατότητα να εντοπίζουμε στοιχεία της διεπαφής χρήστη (κουμπιά, ετικέτες κλπ), να παίρνουμε τις τιμές τους, να εκτελούμε ενέργειες (πχ πάτημα του κουμπιού). Έχοντας αυτές τις δυνατότητες, μπορούμε (σε επίπεδο testing) να αντιληφθούμε αν σε πολλές περιπτώσεις οι εφαρμογή αντιδρά όπως θα έπρεπε ή αν υπάρχει κάποια δυσλειτουργία.



Εικόνα 3-3: Λογότυπο Selenium

Πηγή: selenium.dev

3.2.2 Ιστορία

Το Selenium Project δημιουργήθηκε από τον Jason Huggins το 2004, ενώ εκείνος δούλεψε για την εταιρία ThoughtWorks και αρχικά χρησιμοποιήθηκε από την εταιρία για εσωτερική χρήση. Το Selenium έγινε ένα open-source project, για το

οποίο έδειξαν μεγάλο ενδιαφέρον οι developers και γρήγορα επεκτάθηκε και βελτιώθηκε.

Σημαντική συνεισφορά υπήρξε και από την Google, η οποία από το 2007 μαζί με τον Huggins δημιούργησαν μία ομάδα υποστήριξης του Selenium.

3.2.3 Χαρακτηριστικά του Selenium

3.2.3.1 *Ανεξάρτητο από γλώσσα προγραμματισμού*

Βασικό πλεονέκτημα αποτελεί το γεγονός ότι υποστηρίζεται από μία γκάμα γλωσσών προγραμματισμού υψηλού επιπέδου, όπως η Java, Python, C#, Ruby, Scala, PHP.

3.2.3.2 *Ανεξάρτητο από φυλλομετρητή*

Πλεονέκτημα αποτελεί και το γεγονός ότι τα tests μπορούν να εκτελεστούν και σε διαφορετικούς web browsers (Chrome, Firefox κλπ).

3.2.3.3 *Ανεξάρτητο από λειτουργικό σύστημα*

Δεν περιορίζεται σε ένα συγκεκριμένο λειτουργικό σύστημα, αλλά μπορεί να χρησιμοποιηθεί σε Windows, Linux και macOS.

3.2.3.4 *Open Source*

Το Selenium είναι ένα open source project. Κάθε developer έχει την δυνατότητα, εάν αυτός θελήσει, να συμβάλει στην προσπάθεια βελτίωσης και επέκτασης του Selenium.

3.3 Qt Framework

Η Qt είναι μια δωρεάν, ανοιχτού κώδικα βιβλιοθήκη, με την οποία υπάρχει η δυνατότητα δημιουργίας γραφικής διεπαφής χρήστη. Η βιβλιοθήκη αυτή δεν εξαρτάται από κάποιο λειτουργικό σύστημα, αφού υπάρχει η δυνατότητα να χρησιμοποιηθεί σε Linux, Windows, macOS και σε Android με ελάχιστες διαφορές στον κώδικα. Η Qt έχει διπλή άδεια, εμπορική και ανοιχτού κώδικα.



Εικόνα 3-4: Λογότυπο Qt

Πηγή: tiparmentcon.gr

3.3.1 Ιστορική Αναδρομή

Η δημιουργία της βιβλιοθήκης ξεκίνησε το 1991 από τους Haavard Nord και Eirik Chambe-Eng. Αρχικά η Qt ήταν διαθέσιμη για την γλώσσα C++ και σε λειτουργικά Windows (Qt/Win) και Unix (Qt/X11). Συγκεκριμένα, στην πλατφόρμα των Windows, η άδεια χρήσης ήταν ιδιωτική.

Το 2009 η εμπορική άδεια αγοράστηκε από την Nokia, η οποία χρησιμοποίησε την συγκεκριμένη βιβλιοθήκη και στο λειτουργικό σύστημα Symbian, το οποίο χρησιμοποιούσαν τότε τα κινητά της. Το 2011, η Nokia αποφάσισε να πουλήσει την εμπορική άδεια της Qt στην Digia, αφού απέσυρε τις τεχνολογίες Symbian.

Από το 2016, η άδεια ανήκει στην Qt Company, η οποία μέχρι πρότινος τότε, ήταν θυγατρική της Digia.

Τελευταία έκδοση της Qt, είναι η Qt5, η οποία χρησιμοποιήθηκε και στην παρούσα εργασία.

3.3.2 PyQt

Η PyQt είναι ουσιαστικά η βιβλιοθήκη Qt, συμβατή με Python v2 και v3 και λειτουργεί σε όλες τις πλατφόρμες που υποστηρίζονται από το Qt, συμπεριλαμβανομένων των Windows, OS X, Linux, iOS και Android. Δημιουργήθηκε από την βρετανική εταιρία River Bank Computing.

Η τελευταία έκδοση του PyQt είναι η PyQt5. Η PyQt4 και Qt v4 δεν υποστηρίζονται πλέον και δεν θα υπάρξουν νέες ενημερώσεις για τις εκδόσεις αυτές.

3.3.3 Qt Designer

Βασικό εργαλείο (tool) της PyQt, αλλά και γενικότερα της Qt είναι ο Qt Designer. Ο Qt Designer αποτελεί ένα drag & drop εργαλείο για την κατασκευή γραφικού περιβάλλοντος. Σου επιτρέπει να σχεδιάσεις widgets, dialogs ή ολόκληρη τη βασική φόρμα καθώς μέσα από το Qt ευκολά σύρεις από τον Editor όποιο widget θες να συμπεριλάβεις στην φόρμα που δημιουργείς. Το Qt Designer χρησιμοποιεί αρχεία XML για την προβολή του γραφικού περιβάλλοντος, αλλά και για την εξοικονόμηση ενός μεγάλου μέρους του κώδικα που απαιτείτε για την κατασκευή μιας εφαρμογής.

Ο Qt Designer παράγει αρχεία .ui, τα οποία μπορούν να μετατραπούν σε κώδικα python με το εργαλείο uic Python, το οποίο κάνει την μετατροπή αυτή. Συγκριτικά με άλλους window builders που υπάρχουν, ο Qt Designer παράγει σχετικά καλό κώδικα και εύκολα τροποποιήσιμο.

Τα αρχεία που παράγονται από τον Qt Designer πλαισιώνονται με κώδικα του developer, κώδικας που θα δώσει λειτουργικότητα στα widgets.

Τα αρχεία που παράγονται από τον Qt Designer, μπορούν να χρησιμοποιηθούν και από Qt σε C++, αλλά και από PyQt.

3.4 Βιβλιοθήκη Pandas

Η Pandas αποτελεί την βιβλιοθήκη ανάλυσης δεδομένων της γλώσσας προγραμματισμού Python. Γενικά, η Pandas δημιουργήθηκε με την χρήση άλλων βιβλιοθηκών της Python, όπως είναι η NumPy, η οποία είναι βιβλιοθήκη που χρησιμοποιείται για θεμελιώδεις επιστημονικούς υπολογισμούς. Χρησιμοποιεί ευέλικτες, υψηλού επιπέδου δομές δεδομένων, όπως είναι τα Series και Dataframes, με τα οποία μπορεί κάποιος να διαχειριστεί δεδομένα (που μπορεί να εισάγονται από αρχεία που παράγουν άλλες γνωστές εφαρμογές, πχ Excel).



Εικόνα 3-5: Λογότυπο pandas

Πηγή: numfocus.org

3.5 Βιβλιοθήκη Matplotlib

Η Matplotlib είναι βιβλιοθήκη, η οποία συνήθως χρησιμοποιείται για την κατασκευή διαφόρων γραφημάτων. Η χρήση της και η κλήση των διαφόρων συναρτήσεων, θυμίζουν πολύ αυτές των αντίστοιχων στο MATLAB.



Εικόνα 3-6: Λογότυπο matplotlib

Πηγή: matplotlib.org

3.6 SQL

Η SQL (αγγλ. αρκτ. από το Structured Query Language) είναι μία γλώσσα προγραμματισμού τέταρτης γενιάς για βάσεις δεδομένων, που σχεδιάστηκε για τη διαχείριση δεδομένων, σε ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System, RDBMS) και η οποία, αρχικά, βασίστηκε στη σχεσιακή άλγεβρα. Η γλώσσα περιλαμβάνει δυνατότητες ανάκτησης και ενημέρωσης δεδομένων, δημιουργίας και τροποποίησης σχημάτων και σχεσιακών πινάκων, αλλά και ελέγχου πρόσβασης στα δεδομένα. Η SQL ήταν μία από τις πρώτες γλώσσες για το σχεσιακό μοντέλο του Edgar F. Codd, στο σημαντικό άρθρο του το 1970, και έγινε η πιο ευρέως χρησιμοποιούμενη γλώσσα για τις σχεσιακές βάσεις δεδομένων. Η SQL αναπτύχθηκε στην IBM από τους Andrew Richardson, Donald C. Messerly και Raymond F. Boyce, στις αρχές της δεκαετίας του 1970. Αυτή η έκδοση, αποκαλούμενη αρχικά SEQUEL, είχε ως σκοπό να χειριστεί και να ανακτήσει τα στοιχεία που αποθηκεύτηκαν στο πρώτο RDBMS της IBM, το System R.

3.6.1 SQLite

Η SQLite είναι ένα σύστημα διαχείρισης βάσεων δεδομένων, η οποία παρέχεται σε μία προγραμματιστική βιβλιοθήκη C. Η διάθεση της SQLite είναι ελεύθερη και υποστηρίζει τα χαρακτηριστικά των κλασικών σχεσιακών βάσεων δεδομένων, με την ίδια σύνταξη SQL.

3.6.1.1 Υποστηριζόμενοι Τύποι Δεδομένων

Οι τύποι δεδομένων, οι οποίοι υποστηρίζονται από την SQLite είναι οι εξής:

- **NULL.** Η κλασσική τιμή NULL που υπάρχει και στις κλασσικές γλώσσες προγραμματισμού
- **INTEGER.** Προσημασμένος ακέραιος αριθμός (int)
- **REAL.** Δεκαδικός αριθμός (float)
- **TEXT.** Αλφαριθμητικό με κωδικοποίηση UTF-8, UTF-16BE ή UTF-16LE
- **BLOB.** Οποιαδήποτε δυαδική πληροφορία (κομμάτι δεδομένων ακριβώς όπως εισήχθη)

3.6.1.2 Εφαρμογές της SQLite

Σήμερα η SQLite χρησιμοποιείται ευρέως από πολλές εφαρμογές πελάτη ως ενσωματωμένο λογισμικό βάσεων δεδομένων για τοπική αποθήκευση. Κύρια παραδείγματα εφαρμογών είναι οι φυλλομετρητές (web browsers), εφαρμογές διαχείρισης επαφών στα κινητά, στα ίδια τα λειτουργικά των κινητών (iOS, Android).

Ωστόσο, υπάρχουν πολλοί περιορισμοί σε σχέση με παραδοσιακά DBMS, όπως είναι η MySQL και η Oracle, οι οποίες, σε αντίθεση με την SQLite, κάνουν χρήση του μοντέλου πελάτη/εξυπηρετητή (client/server). Η SQLite, δίνει έμφαση στην απλότητα, την ανεξαρτησία, την ταχύτητα στην πρόσβαση δεδομένων, αλλά και την οικονομία από άποψη χωρητικότητας. Γι' αυτό χρησιμοποιείται, κυρίως, ως τοπική/προσωρινή βάση δεδομένων.



Εικόνα 3-7: Λογότυπο SQLite

Πηγή: <https://www.sqlite.org/index.html>

3.6.2 Python-SocketIO

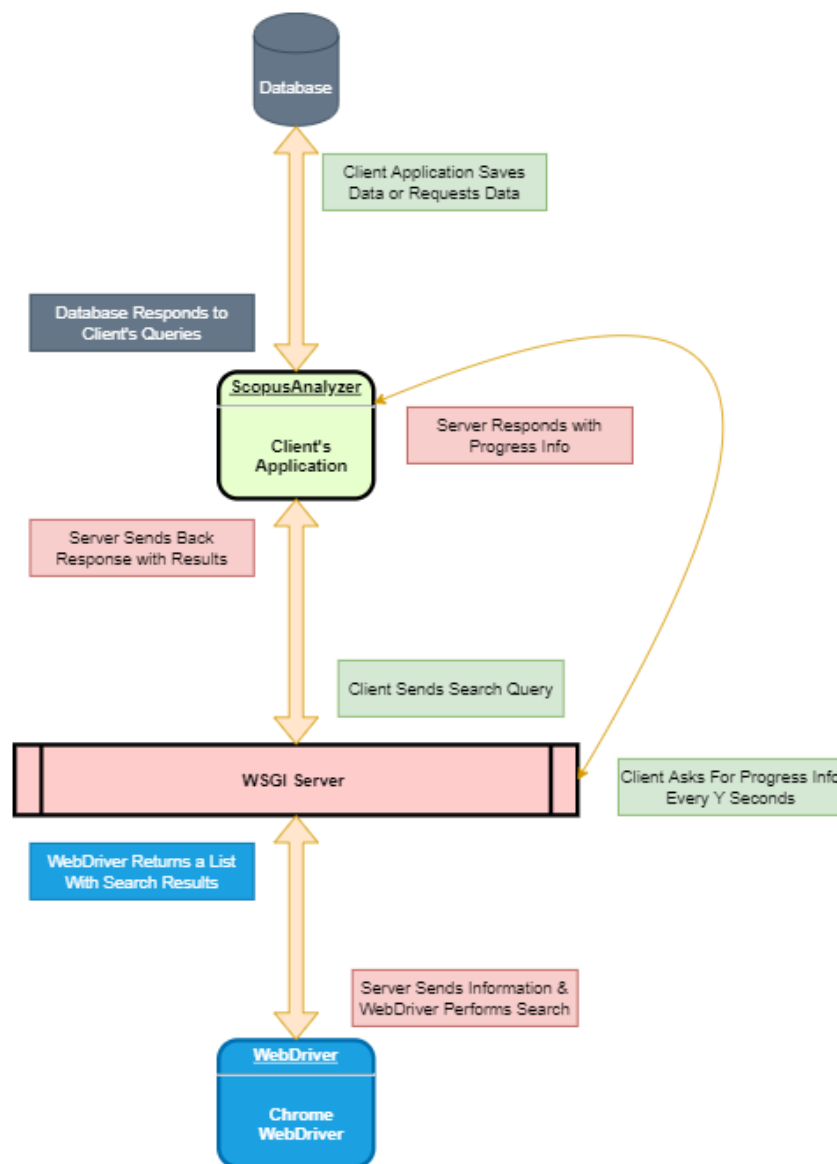
Το Socket.IO αποτελεί ένα πρωτόκολλο μεταφοράς, το οποίο επιτρέπει σε πραγματικό χρόνο αμφίδρομη επικοινωνία που βασίζεται σε συμβάντα μεταξύ πελατών και διακομιστή. Οι επίσημες υλοποιήσεις των στοιχείων πελάτη και διακομιστή γράφονται σε JavaScript. Το Python-SocketIO παρέχει υλοποιήσεις Python και των δύο (πελάτη, διακομιστή), το καθένα με τυπικές και ασύγχρονες

παραλλαγές. Στην παρούσα εργασία, χρησιμοποιούμε το συγκεκριμένο πρωτόκολλο τόσο στον πελάτη, όσο και στον διακομιστή για την ανταλλαγή μηνυμάτων.

4 Σχεδιασμός και Υλοποίηση Συστήματος

Στο παρόν κεφάλαιο πραγματοποιείται μια εκτενής αναφορά στην αρχιτεκτονική του συστήματος (διακομιστή και εφαρμογής πελάτη), καθώς και στον τρόπο κατασκευής και σύνδεσης των δομικών μερών της.

4.1 Αρχιτεκτονική Συστήματος



Εικόνα 4-1: Αρχιτεκτονική του Συστήματος

Όπως βλέπουμε και στην παραπάνω εικόνα, αρχιτεκτονική του συστήματος χωρίζεται σε τέσσερα μέρη:

1. Την εφαρμογή του πελάτη (Client Application - ScopusAnalyzer, Desktop Εφαρμογή)
2. Την βάση δεδομένων του πελάτη (SQLite, τοπική βάση)
3. Τον διακομιστή (Server, WSGI Application)
4. Τον WebDriver (ChromeDriver στην περίπτωσή μας)

4.1.1 Εφαρμογή Πελάτη

Η Client-Side εφαρμογή, η οποία ονομάζεται **ScopusAnalyzer**, είναι μία εφαρμογή γραφείου (desktop), η οποία είναι γραμμένη στην γλώσσα προγραμματισμού Python, με την βοήθεια της βιβλιοθήκης PyQt (έκδοση 5). Με την εφαρμογή αυτή, ο πελάτης έχει την δυνατότητα να εκτελέσει τις παρακάτω λειτουργίες (επιγραμματικά):

1. Να εκτελέσει μια παραμετροποιημένη αναζήτηση στην βιβλιογραφική βάση Scopus,
2. Να εξάγει διάφορα περιγραφικά στατιστικά από ήδη υπάρχοντα αρχεία Excel, τα οποία έχουν παραχθεί από την ίδια την εφαρμογή,
3. Να εξάγει στατιστικά από δεδομένα που βρίσκονται εσωτερικά στην τοπική βάση δεδομένων του πελάτη,
4. Να εισάγει δεδομένα από αρχεία Excel στην βάση δεδομένων του πελάτη (τα αρχεία πρέπει να έχουν εξαχθεί από την ίδια την εφαρμογή)

4.1.2 Τοπική Βάση Δεδομένων της Εφαρμογής Πελάτη

Στην ενότητα αυτή παρουσιάζεται η βάση δεδομένων, η οποία σχεδιάστηκε και υλοποιήθηκε, αποτελούμενη από τρεις πίνακες. Τον πίνακα *documents*, στον οποίο αποθηκεύονται τα δεδομένα που αφορούν τα έγγραφα (Documents), τον πίνακα *sources*, στον οποίο αποθηκεύονται τα δεδομένα που αφορούν τα περιοδικά (Sources) και τον πίνακα *uom_professors*, ο οποίος περιέχει τα που αφορούν τους καθηγητές του Πανεπιστημίου Μακεδονίας.

Στον Πίνακα 1 παραθέτουμε τον πίνακα *documents* της βάσης δεδομένων με τα πεδία και τον τύπο τους. Κλειδί του πίνακα είναι το *doc_id*. Στον πίνακα

αποθηκεύουμε το όνομα του εγγράφου, τους συγγραφείς και τον αριθμό τους και την χρονιά δημοσίευσης. Τέλος, υπάρχει και το πεδίο με το ξένο κλειδί του πίνακα, το *source_id*, το οποίο αναφέρεται στο πρωτεύον κλειδί του πίνακα sources. Ένα Document μπορεί να έχει αποκλειστικά ένα Source.

documents	
doc_id	INTEGER
doc_name	TEXT
authors_num	INTEGER
authors	TEXT
year	INTEGER
source_id	INTEGER

Πίνακας 1

Στον Πίνακα 2 παραθέτουμε τον πίνακα sources της βάσης δεδομένων με τα πεδία και τον τύπο τους. Κλειδί του πίνακα είναι το *source_id*. Στον πίνακα αποθηκεύουμε το όνομα του περιοδικού, τις μετρικές CiteScore, SJR, SNIP και τον μέσο όρο των Percentile του περιοδικού συγγραφείς και τον αριθμό τους και την χρονιά δημοσίευσης. Ένα Source μπορεί να συνδέεται με πολλά Documents.

sources	
source_id	INTEGER
source_name	TEXT
citescore	REAL
sjr	REAL
snip	REAL
avg_percentile	REAL

Πίνακας 2

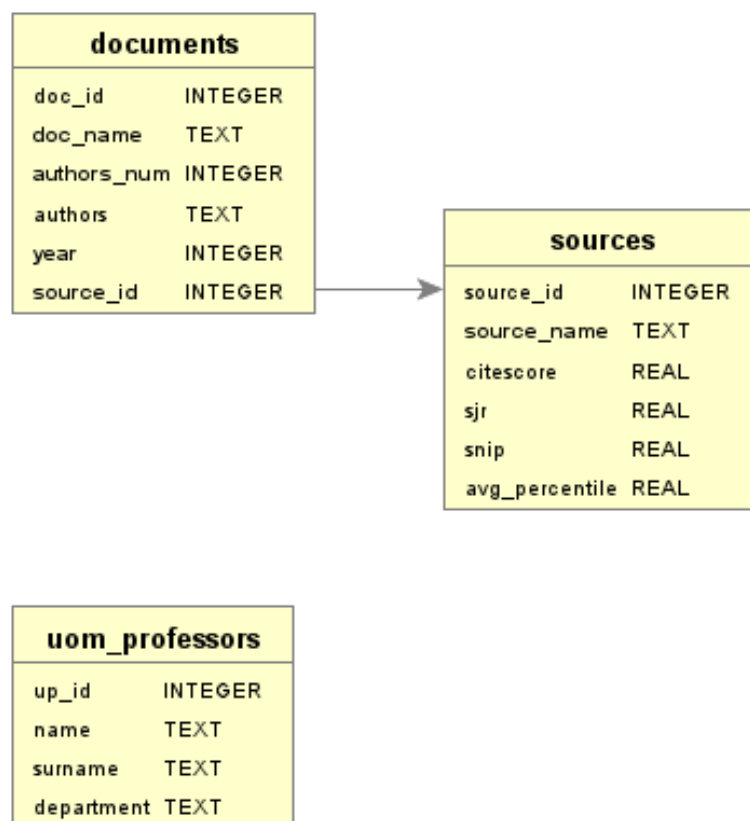
Στον Πίνακα 3 παραθέτουμε τον πίνακα uom_professors της βάσης δεδομένων με τα πεδία και τον τύπο τους. Κλειδί του πίνακα είναι το *up_id*. Στον πίνακα αποθηκεύουμε το όνομα και το επώνυμο του καθηγητή, καθώς και το όνομα του τμήματος στο οποίο ανήκει. Ο πίνακας δεν συνδέεται με κάποιον άλλο πίνακα,

απλά κρατείται, ώστε να παραχθούν τα κατάλληλα στατιστικά (μπορεί να αφορούν το τμήμα ή/και καθηγητή/τες), αν ζητηθούν από τον πελάτη.

uni_professors	
up_id	INTEGER
name	TEXT
surname	TEXT
department	TEXT

Πίνακας 3

Στην *Εικόνα 1* φαίνεται το ER (οντοτήτων – συσχετίσεων) διάγραμμα των πινάκων της βάσης δεδομένων στην παραδοσιακή μορφή όπου εμφανίζονται οι συσχετίσεις των πινάκων.



Εικόνα 4-2: Διάγραμμα Οντοτήτων Συσχετίσεων της Τοπικής Βάσης Δεδομένων

4.1.3 Σύνδεση Εφαρμογής Πελάτη με Βάση Δεδομένων

Παρακάτω θα παρουσιαστούν οι συναρτήσεις, με τις οποίες γίνεται η σύνδεση της βάσης δεδομένων (SQLite) με την εφαρμογή πελάτη (ScopusAnalyzer).

Χρησιμοποιούμε ένα ξεχωριστό module για την σύνδεση αυτή, το db.py, το οποίο περιέχεται στον ίδιο κατάλογο με την βάση της εφαρμογής. Το module db.py διαθέτει όλες τις βοηθητικές συναρτήσεις εκείνες, προκειμένου να εκτελούνται εύκολα και με διαφάνεια οι απαραίτητες ενέργειες από την γραφική διεπαφή (δεν γράφουμε κανένα «ωμό» query στον κώδικα του GUI).

```
1. def get_all_records():
2.     '''
3.     Gets all records from Database (if exist)
4.     in order to create a list of tuples, which later
5.     can be converted to classic dataframe
6.     '''
7.     query = """SELECT doc_name, authors_num, authors, year, source_name
8.         , avg_percentile, citescore, sjr, snip FROM documents NATURAL JOIN
9.         sources ORDER BY avg_percentile DESC"""
10.    cursor.execute(query)
11.    all_recs = cursor.fetchall()
12.
13.    return all_recs
```

Η παραπάνω συνάρτηση, η “get_all_records”, ουσιαστικά ζητάει από την βάση τις εγγραφές, οι οποίες περιέχουν όλα τα έγγραφα και τα περιοδικά, έτσι ώστε στην συνέχεια να εμφανιστούν/εξαχθούν κατάλληλα.

```
1. def get_df_by_year(years):
2.     '''
3.     Gets all records from the database
4.     whose year matches with the years/year passed as
5.     parameter to function
6.     Returns a dataframe with all the records & column names
7.     renamed with appropriate formation (to be exported directly to
8.     excel file)
9.     '''
10.    query = """SELECT doc_name, authors_num, authors, year, source_
11.        name, avg_percentile, citescore, sjr, snip FROM documents NATURAL J
12.        OIN sources WHERE year BETWEEN ? AND ? ORDER BY avg_percentile DESC
13.        """
14.    df = pd.read_sql_query(query, conn, params=years)
15.    df = df.rename(columns={"authors_num": "# Authors", "authors":
16.        "Authors", "avg_percentile": "Average Percentile", "citescore":
17.        "CiteScore", "doc_name": "Document
18.        Name", "sjr": "SJR", "snip": "SNIP", "source_name": "Source Name",
19.        "year": "Year"})
20.    return df
```

Η παραπάνω συνάρτηση, η “get_df_by_year”, ουσιαστικά ζητάει από την βάση όλες τις εγγραφές, των οποίων τα έγγραφα έχουν συγκεκριμένη χρονολογία/χρονολογίες. Οι πληροφορίες που επιστρέφει είναι τα έγγραφα και τα περιοδικά τους, έτσι ώστε στην συνέχεια να εμφανιστούν/εξαχθούν κατάλληλα. Η συνάρτηση επιστρέφει τις πληροφορίες, αφού πρώτα τις μετατρέψει σε ένα Pandas Dataframe.

```
1. def get_distinct_years(order='DESC'):  
2.     '''  
3.     Finds all different years that exist in database (no duplicates  
4.     )  
5.     Returns the list of tuples with the years (if there are any)  
6.     '''  
7.     query = f"""SELECT DISTINCT year FROM documents ORDER BY year {  
8.         order}"""  
9.     cursor.execute(query)  
10.    years = cursor.fetchall()  
11.    return years
```

Η παραπάνω συνάρτηση, η “get_distinct_years”, ουσιαστικά ζητάει από την βάση όλες τις χρονολογίες που υπάρχουν σε αυτή, χωρίς διπλότυπα. Για παράδειγμα, αν υπάρχουν έγγραφα χρονολογιών 2017 και 2018, θα επιστρέψει [(2017,), (2018,)]. Αυτή η λειτουργία είναι χρήσιμη, όταν ο χρήστης θέλει να διαγράψει εγγραφές από την βάση, καθώς η διαγραφή μπορεί να γίνει **μόνο ανά χρονολογία**.

```
1. def perform_deletion(year):  
2.     '''  
3.     Based on a specific year, performs deletion  
4.     to all records matching this year  
5.     Returns True if deletion is completed  
6.     '''  
7.     queries = [f''' DELETE FROM documents WHERE year = {year}''', f  
8.         f''' DELETE FROM sources WHERE source_id IN (SELECT source_id FROM  
9.             documents WHERE year = {year})''']  
10.    for query in queries:  
11.        cursor.execute(query)  
12.    conn.commit()  
13.    return True
```

Η παραπάνω συνάρτηση, η “perform_deletion”, ουσιαστικά διαγράφει από την βάση όλες τις εγγραφές (έγγραφα και περιοδικά), των οποίων η χρονολογία ταιριάζει με την παράμετρο “year” της συνάρτησης. Αυτή η λειτουργία είναι χρήσιμη, όταν ο χρήστης θέλει να διαγράψει εγγραφές από την βάση, καθώς η διαγραφή μπορεί να γίνει **μόνο ανά χρονολογία**.

```

1. def is_db_empty():
2.     '''
3.     Checks if database is empty
4.     Returns True if it is
5.     Returns False if it is not
6.     '''
7.     query = """SELECT doc_name, authors_num, authors, year, source_
      name, avg_percentile, citescore, sjr, snip FROM documents NATURAL J
      OIN sources ORDER BY avg_percentile DESC"""
8.     cursor.execute(query)
9.     row = cursor.fetchone()
10.    return row is None

```

Η παραπάνω συνάρτηση, η “is_db_empty”, ελέγχει αν υπάρχει οποιαδήποτε εγγραφή στην βάση (από τους πίνακες documents, sources).

```

1. def save_to_db(lst):
2.     '''
3.     Gets a list of records and saves to
4.     database appropriately
5.     If save is successful, returns True,
6.     Else returns False
7.     '''
8.     results_lst = lst[0]
9.     df = pd.DataFrame(results_lst)
10.
11.    for index, row in df.iterrows():
12.        try:
13.            cursor.execute('SELECT source_id FROM sources WHERE sou
      rce_name = ?', (row['Source Name'],))
14.            source = cursor.fetchone()
15.            if not source:
16.                cursor.execute('INSERT INTO sources VALUES (null,?,
      ?,?,?,?)', row[['Source Name', 'CiteScore', 'SJR', 'SNIP', 'Average
      Percentile']])
17.                tmp_lst = list(row[['Document Name', '# Authors', '
      Authors', 'Year']])
18.                tmp_lst.append(cursor.lastrowid)
19.            else:
20.                tmp_lst = list(row[['Document Name', '# Authors', '
      Authors', 'Year']])
21.                tmp_lst.append(source[0])
22.                cursor.execute('INSERT OR IGNORE INTO documents VALUES
      (null,?,?,?,?,?)', tmp_lst)
23.        except Exception as e:
24.            print(e)
25.            return False
26.
27.    conn.commit()
28.    return True

```

Η παραπάνω συνάρτηση, η “save_to_db”, αποθηκεύει τις εγγραφές που έλαβε η εφαρμογή του πελάτη από τον διακομιστή (λειτουργία της παραμετροποιημένης αναζήτησης). Αποθηκεύει σωστά τα έγγραφα και τα περιοδικά, δημιουργώντας, ταυτόχρονα, και τις κατάλληλες συσχετίσεις μεταξύ των δύο.

```

1. def insert_from_excel(df):
2.     '''
3.     Gets a dataframe (created from excel file)
4.     and saves all the records to database appropriately
5.     If save is successful, returns True,
6.     Else returns False
7.     '''
8.     for index, row in df.iterrows():
9.         try:
10.            cursor.execute('SELECT source_id FROM sources WHERE sou
11.                           rce_name = ?', (row['source_name'],))
12.            source = cursor.fetchone()
13.            if not source:
14.                cursor.execute('INSERT INTO sources VALUES (null,?,
15.                           ?,?,?,?)', row[['source_name', 'citescore', 'sjr', 'snip', 'avg_per
16.                           centile']])
17.                tmp_lst = list(row[['doc_name', 'authors_num', 'aut
18.                           hors', 'year']])
19.                tmp_lst.append(cursor.lastrowid)
20.            else:
21.                tmp_lst = list(row[['doc_name', 'authors_num', 'aut
22.                           hors', 'year']])
23.                tmp_lst.append(source[0])
24.            cursor.execute('INSERT OR IGNORE INTO documents VALUES
25.                           (null,?,?,?,?,?)', tmp_lst)
26.        except Exception as e:
27.            print(e)
28.            return False
29.    conn.commit()
30.    return True

```

Η παραπάνω συνάρτηση, η “insert_from_excel” αποθηκεύει τις εγγραφές που έλαβε η εφαρμογή του πελάτη από τον διακομιστή (λειτουργία της παραμετροποιημένης αναζήτησης). Αποθηκεύει σωστά τα έγγραφα και τα περιοδικά, δημιουργώντας, ταυτόχρονα, και τις κατάλληλες συσχετίσεις μεταξύ των δύο.

```

1. def fetch_professors_from_db():
2.     '''
3.     Fetches professors from database
4.     and returns a list of tuples with all the
5.     necessary information of each professor
6.     '''
7.     query = """SELECT name, surname, department FROM uom_professors
8.               """
9.     cursor.execute(query)
10.    professors = cursor.fetchall()
11.    return professors

```

Η παραπάνω συνάρτηση, η “fetch_professors_from_db”, ζητά όλες τις εγγραφές των καθηγητών του Πανεπιστημίου Μακεδονίας, και συγκεκριμένα το όνομα, το επώνυμο και το τμήμα στο οποίο ανήκουν. Η συνάρτηση πάντα επιστρέφει δεδομένα, αφού ο πίνακας των καθηγητών του πανεπιστημίου δεν μπορεί να μεταβληθεί από τον χρήστη.

4.1.4 Διακομιστής Συστήματος

Ο διακομιστής (Server) του συγκεκριμένου συστήματος έχει υλοποιηθεί, όπως και η εφαρμογή του πελάτη στην γλώσσα προγραμματισμού Python. Η βιβλιοθήκη, η οποία χρησιμοποιήθηκε για την υποστήριξη του διακομιστή, αλλά και την διαδικασία επικοινωνίας πελάτη-διακομιστή είναι η Python-SocketIO.

Ουσιαστικά ο Server έχει τις εξής αρμοδιότητες:

1. Παραλαβή αιτημάτων από τον πελάτη και μεταβίβασή τους στον Selenium WebDriver (ChromeDriver),
2. Διαχείριση του Selenium WebDriver, δίνοντάς του κατάλληλες εντολές κάθε φορά,
3. Ενημέρωση του πελάτη για την πρόοδο του αιτήματός του (στέλνει πόσο έγγραφα έχουν ελεγχθεί μέχρι την στιγμή αιτήματος),
4. Παραλαβή των αποτελεσμάτων μέσω του Selenium WebDriver και αποστολή τους (ως λίστα) στον πελάτη

4.1.5 Selenium WebDriver

Ουσιαστικά, ο WebDriver αποτελεί το τελευταίο και πιο «βαθύ» συστατικό του συστήματος. Ο WebDriver (στην περίπτωση του συστήματος είναι ο ChromeDriver), ουσιαστικά έχει την αρμοδιότητα να εκτελεί τις εντολές του διακομιστή προς αυτόν. Έχοντας την ιδιότητα του εντολοδόχου, αλλά και τα απαραίτητα δεδομένα, κάνει parsing ιστοσελίδων, αντλώντας από αυτές όλες τις απαραίτητες πληροφορίες.

Ο WebDriver αποτελεί μια ξεχωριστή διεργασία, η οποία εκτελείται στο λειτουργικό σύστημα του διακομιστή. Οι πληροφορίες που κρατάει ο ίδιος είναι ίδιες με αυτές που κρατάει ένας κοινός φυλλομετρητής (cache, cookies κλπ.). Ο κύκλος ζωής της διεργασίας αυτής εξαρτάται άμεσα από τον κύκλο ζωής του διακομιστή. Όσο ο διακομιστής εκτελείται, θα εκτελείται και ο WebDriver. Όταν κλείσει ο διακομιστής, ό,τι πληροφορίες κρατούσε ο WebDriver, χάνονται μαζί με αυτές του διακομιστή.

4.2 Λειτουργίες Εφαρμογής Πελάτη (ScopusAnalyzer)

4.2.1 Εκτέλεση Παραμετροποιημένης Αναζήτησης στην βιβλιογραφική βάση Scopus

1. Αν ο χρήστης επιθυμεί **να εκτελέσει νέα παραμετροποιημένη αναζήτηση**:
 - i. Επιλέγει τις παραμέτρους που του προσφέρονται από την γραφική διασύνδεση, όπως και φάκελο εξαγωγής του αρχείου Excel ή και το ίδιο το όνομα του Excel (αν επιθυμεί να γίνει εξαγωγή σε εξωτερικό αρχείο). Επίσης, επιλέγει το αν επιθυμεί να αποθηκευτούν τα αποτελέσματα στην τοπική βάση δεδομένων του προγράμματος. Μόλις ο χρήστης πατήσει το κουμπί “Proceed” δημιουργείται το query string (αλφαριθμητικό αιτήματος) και στέλνει το αίτημα στον διακομιστή.
 - ii. Από την στιγμή που ληφθεί το αίτημα από τον διακομιστή, στέλνεται στον WebDriver, ώστε να πραγματοποιήσει την αναζήτηση. Μόλις πραγματοποιηθεί η αναζήτηση και εμφανιστούν τα αποτελέσματα στο Document Page του Scopus, ο WebDriver ειδοποιεί τον διακομιστή και δέχεται νέο αίτημα από εκείνον, ώστε να αρχίσει να επισκέπτεται διαδοχικά τα sources και να επιστρέφει τις τιμές των μετρικών, το όνομα κλπ. Μόλις ελεγχθεί και το τελευταίο περιοδικό, η ανάλυση σταματά.
 - iii. Όταν σταματήσει η ανάλυση από τον WebDriver, ο διακομιστής της εφαρμογής τα επεξεργάζεται, και τα τοποθετεί σε κατάλληλη δομή, ώστε να είναι σε θέση να τα στείλει πίσω στην εφαρμογή πελάτη.
 - iv. Μόλις ολοκληρωθεί η ανάλυση, εμφανίζεται στον χρήστη ένα παράθυρο ειδοποίησης ότι ολοκληρώθηκε η αναζήτηση και του δίνεται η επιλογή να δει δομημένα τα αποτελέσματα της αναζήτησης μέσω της φόρμας Results.
- Στην περίπτωση που ο χρήστης επιλέξει να δει τα αποτελέσματα της αναζήτησης, του εμφανίζεται η φόρμα Results.
- Ο χρήστης έχει την δυνατότητα να μεταβεί πίσω στην φόρμα Menu ή να προχωρήσει να δει τα στατιστικά. Αν επιλέξει να δει τα στατιστικά διαγράμματα, το πρόγραμμα τα παράγει με την βοήθεια της βιβλιοθήκης matplotlib και τα εμφανίζει σε ξεχωριστό παράθυρο, από το οποίο ο χρήστης έχει την δυνατότητα αποθήκευσης της εικόνας.

4.2.2 Εξαγωγή Στατιστικών από Υπάρχοντα Αρχεία Excel

Αν ο χρήστης επιθυμεί **να εξάγει στατιστικά από πολλαπλά αρχεία Excel**, θα επιλέξει την καρτέλα Import Excels, και θα προσθέσει ή θα σύρει (drag & drop) τα αρχεία που ήδη κατέχει, ώστε το πρόγραμμα να παράγει στατιστικά διαγράμματα με βάση τα αρχεία αυτά. Ο χρήστης επιλέγει το κουμπί «Export» και εν συνεχεία, θα εμφανιστεί ένα νέο παράθυρο, το οποίο θα επιτρέπει στον χρήστη να ρυθμίσει τις λεπτομέρειες της εξαγωγής (π.χ. το εύρος των ετών που επιθυμεί να εξεταστούν), τι είδους στατιστικά θα εξαχθούν (αν θα εξαχθούν τα συγκεντρωτικά δεδομένα της βάσης δεδομένων, τα διαγράμματα, percentile ανά τμήμα, percentiles ανά καθηγητή), το όνομα του αρχείου εξαγωγής, καθώς και τον κατάλογο εξαγωγής. Τα διαγράμματα αυτά, θα δείχνουν διαχρονικά την πορεία των αναφορών των καθηγητών του Πανεπιστημίου Μακεδονίας. Εμφανίζονται, επίσης, και διαγράμματα που έχουν να κάνουν με την πορεία των διαφόρων δεικτών (κατά μέσο όρο). Αν ο χρήστης έχει προσθέσει μόνο ένα αρχείο, τότε, προφανώς, τα διαγράμματα που αφορούν την διαχρονική πορεία των διαφόρων δεικτών δε θα εμφανίζονται.

4.2.3 Εξαγωγή Στατιστικών από την Βάση Δεδομένων της Εφαρμογής

Αν ο χρήστης επιθυμεί **να εξάγει στατιστικά από τα ήδη υπάρχοντα στοιχεία της βάσης δεδομένων**, θα επιλέξει την καρτέλα Database, και θα επιλέξει το κουμπί «Export Statistics». Στη συνέχεια, θα εμφανιστεί ένα νέο παράθυρο, το οποίο θα επιτρέπει στον χρήστη να ρυθμίσει τις λεπτομέρειες της εξαγωγής (π.χ. το εύρος των ετών που επιθυμεί να εξεταστούν), τι είδους στατιστικά θα εξαχθούν (αν θα εξαχθούν τα συγκεντρωτικά δεδομένα της βάσης δεδομένων, τα διαγράμματα, percentile ανά τμήμα, percentiles ανά καθηγητή), το όνομα του αρχείου εξαγωγής, καθώς και τον κατάλογο εξαγωγής. Τα διαγράμματα αυτά, θα δείχνουν διαχρονικά την πορεία των αναφορών των καθηγητών του Πανεπιστημίου Μακεδονίας.

4.2.4 Εισαγωγή στοιχείων στη Βάση Δεδομένων από Εξωτερικά Αρχεία

Αν ο χρήστης επιθυμεί **να εισάγει στοιχεία από εξωτερικά αρχεία Excel στην βάση δεδομένων της εφαρμογής**, θα επιλέξει την καρτέλα Database και θα επιλέξει το κουμπί «Insert from Excel». Στη συνέχεια, θα εμφανιστεί ένα νέο παράθυρο, το οποίο θα επιτρέπει στον χρήστη να προσθέσει ή να σύρει (drag & drop) τα αρχεία που ήδη κατέχει, ώστε το πρόγραμμα να τα εισάγει στην τοπική βάση δεδομένων του.

Σημείωση: Σε περίπτωση διπλοτύπων, το πρόγραμμα θα διατηρήσει αυτά που υπάρχουν ήδη στην βάση.

4.2.5 Διαγραφή στοιχείων από την Βάση Δεδομένων

Αν ο χρήστης επιθυμεί να διαγράψει στοιχεία από την βάση δεδομένων, θα επιλέξει την καρτέλα Database και θα επιλέξει το κουμπί «Delete». Στη συνέχεια, θα εμφανιστεί ένα παράθυρο, το οποίο θα επιτρέπει στον χρήστη να επιλέξει χρονιά. Πατώντας «Ok», θα διαγραφούν όλα τα έγγραφα (documents), μαζί με τα περιοδικά (sources) της χρονιάς εκείνης.

Σε περίπτωση που δεν υφίστανται στοιχεία γενικά στην βάση δεδομένων της εφαρμογής, το παράθυρο δε θα εμφανιστεί. Αντ' αυτού, θα εμφανιστεί ενημερωτικό μήνυμα στον χρήστη ότι η βάση δεδομένων είναι κενή.

Σημείωση: Επιτρέπεται μόνο η διαγραφή ανά χρονιά, για να μην υπάρξουν πιθανές ασυνέπειες στην εξαγωγή των στατιστικών (από μία λαθεμένη διαγραφή).

4.3 Κώδικας

4.3.1 Κώδικας Εφαρμογής Πελάτη (ScopusAnalyzer)

4.3.1.1 Παραμετροποιήσεις Εφαρμογής

Πριν ξεκινήσουμε την ανάλυση του κώδικα της εφαρμογής αυτής καθ' αυτής, πρέπει να επισημάνουμε ότι η εφαρμογή έχει παραμετροποιηθεί, ώστε ορισμένες σημαντικές αλλαγές να μπορούν να γίνονται χωρίς επέμβαση στον κώδικα, ακόμα και από τον ίδιο τον χρήστη.

Συγκεκριμένα, έχει προστεθεί ένα αρχείο με ρυθμίσεις (configuration file), από το οποίο η εφαρμογή λαμβάνει συγκεκριμένες ρυθμίσεις, κατά την ώρα της εκτέλεσής της. Αυτό το αρχείο ονομάζεται «settings.ini» και βρίσκεται στον κεντρικό φάκελο, μαζί με το module main.py. Το αρχείο έχει την εξής μορφή:

[SYSTEM_SETTINGS]

CLIENT_NAMESPACE=/desktop_client // Το namespace που αναγνωρίζεται από τον διακομιστή

HOST=http://localhost:5000 // Η διεύθυνση IP του διακομιστή

STOP_EVENT=stop // Η τιμή που θα πρέπει να λάβει ο διακομιστής για να σταματήσει

UPDATE_EVENT=update // Η τιμή που θα πρέπει να λάβει ο διακομιστής για να ενημερώσει

GET_LIST_EVENT=get_final_lst // Η τιμή που θα πρέπει να λάβει ο διακομιστής για να στείλει πίσω τα αποτελέσματα

SEARCH_EVENT=search // Η τιμή που θα πρέπει να λάβει ο διακομιστής για να αρχίσει την παραμετροποιημένη αναζήτηση

GET_TOTAL_DOCS_EVENT=get_total_docs // Η τιμή που θα πρέπει να λάβει ο διακομιστής για να στείλει τα συνολικά έγγραφα που θα σαρώσει (χρήσιμο για την progress bar)

DATABASE=data // Το όνομα του αρχείου της τοπικής βάσης δεδομένων (.db), μπορεί να γραφεί και με την επέκταση του αρχείου (DATABASE=data.db)

[PUBLICATION_DATA]

FIRST_C_LOWER=95.0 // Η ελάχιστη τιμή της 1^{ης} κατηγορίας (percentile, άνω 5%)

SECOND_C_LOWER=80.0 // Η ελάχιστη τιμή της 2^{ης} κατηγορίας (percentile, άνω 20%)

THIRD_C_LOWER=50.0 // Η ελάχιστη τιμή της 3^{ης} κατηγορίας (percentile, άνω 50%)

Παραπάνω, βλέπουμε να υπάρχουν ονόματα σταθερών, και δεξιά του ίσον (=) οι τιμές τους.

Στην πρώτη κατηγορία (SYSTEM_SETTINGS), υπάρχουν οι σταθερές, οι οποίες αφορούν την λειτουργία του συστήματος. **Οι σταθερές της συγκεκριμένης κατηγορίας δεν συνίσταται να μεταβάλλονται από τον ίδιο τον χρήστη.** Παρόλα αυτά, είναι ιδιαίτερα σημαντικές καθώς μπορούμε εύκολα, για παράδειγμα, **να αλλάξουμε την διεύθυνση (IP) του host** (του διακομιστή), σε περίπτωση που αλλάξει ή **να αλλάξουμε το όνομα της βάσης δεδομένων.**

4.3.1.2 Αρχικοποίηση εφαρμογής

```
1. def open_qss(path):
2.     """
3.     Opens a Qt stylesheet with a path relative to the project
4.     Note: it changes the urls in the Qt stylesheet (in memory), and
5.     makes these urls relative to the project
6.     Warning: the urls in the Qt stylesheet should have the forward
7.     slash ('/') as the pathname separator
8.     """
9.     with open(path) as f:
10.         qss = f.read()
11.         pattern = r'url\((.*?)\)';
12.         for url in sorted(set(re.findall(pattern, qss)), key=len, r
13.                             everse=True):
14.             directory, basename = os.path.split(path)
15.             new_url = os.path.join(directory, *url.split('/'))
16.             new_url = os.path.normpath(new_url)
17.             new_url = new_url.replace(os.path.sep, '/')
18.             qss = qss.replace(url, new_url)
19.         return qss
```

Ο κώδικας, ο οποίος βρίσκεται στο αρχείο `main.py` και παρουσιάζεται παραπάνω, είναι αυτός που εκκινεί την εφαρμογή. Όπως βλέπουμε παραπάνω, υπάρχει η συνάρτηση `open_qss`, η οποία σχετίζεται με τα γραφικά όλων των φορμών (Menu, Results), τα οποία βρίσκονται σε ξεχωριστό αρχείο, το οποίο φορτώνεται κατά την και κατά την δημιουργία των φορμών φορτώνονται και οι όποιες εικόνες, εικονίδια χρησιμοποιούνται (από τον φάκελο `style/images`). Συγκεκριμένα η `open_qss`:

- Αναλαμβάνει να ανοίξει ένα αρχείο `qss`, πιο συγκεκριμένα το `stylesheet.qss` που βρίσκεται στον φάκελο `ui/style` και περιέχει τα γραφικά (χρώματα, σχήματα) της γραφικής διασύνδεσης χρήστη (GUI). Η φόρτωση γίνεται μια φορά (στην `main`) και έχει ισχύ για **κάθε** φόρμα της εφαρμογής.

4.3.1.3 Εκκίνηση της εφαρμογής

Μετά το `if __name__ == '__main__':`, το οποίο είναι ο τρόπος που χρησιμοποιεί η Python για να δηλώνει την `Main` (όπως είναι γνωστή και από την Java) δημιουργείται και εμφανίζεται η αρχική φόρμα της εφαρμογής, η φόρμα το μενού.

```
1. if __name__ == '__main__':
2.     app = QtWidgets.QApplication(sys.argv)
3.     qss = open_qss(os.path.dirname(os.path.abspath(__file__))+'ui/
4.         style/stylesheet.qss')
5.     app.setStyleSheet(qss)
6.     MainWindow = QtWidgets.QMainWindow()
7.     menu = Ui_MainWindow()
8.     menu.setupUi(MainWindow)
9.     MainWindow.show()
10.    sys.exit(app.exec_())
```

4.3.1.4 Αιτήματα προς τον Διακομιστή

Όπως έχουμε υλοποιήσει την εφαρμογή πελάτη, υπάρχουν 8 διαφορετικά είδη αιτήματος προς τον διακομιστή (server). Τα αιτήματα αυτά περιλαμβάνονται στο `module client.py`, το οποίο βρίσκεται στον κύριο φάκελο, μαζί με το `main.py`. Οι συναρτήσεις που υπάρχουν στην κλάση του `DesktopClientNamespace` (η κλάση ονοματοδοσίας της εφαρμογής μας) καλούνται από τα [Threads](#), τα οποία χρησιμοποιούνται από την γραφική διασύνδεση ([SearchThread](#), [AnalysisThread](#)) για να γίνουν τα αιτήματα (μέσω του [Python-SocketIO](#)).

4.3.1.4.1 Αίτημα Σύνδεσης στον Διακομιστή

Το αίτημα αυτό δημιουργείται μόλις ο χρήστης πατήσει το κουμπί της αναζήτησης. Αν ο διακομιστής δεν είναι διαθέσιμος, εμφανίζεται κατάλληλο ενημερωτικό μήνυμα στον χρήστη.

```

1. def connect_to_server(self):
2.     try:
3.         sio.connect(self.HOST)
4.     except Exception as e:
5.         return False, str(e)
6.     return True, True

```

4.3.1.4.2 Αίτημα για Παραμετροποιημένη Αναζήτηση

Στέλνεται το query string (αλφαριθμητικό αιτήματος) στον διακομιστή και εκείνος απαντά, εάν η αναζήτηση ήταν επιτυχής.

```

1. def make_search_request(self, query):
2.     response = sio.call(event=self.EVENTS['search_event'], data
    =query)
3.     return response

```

4.3.1.4.3 Αίτημα Αποστολής Συνολικού Αριθμού Εγγράφων

Το συγκεκριμένο αίτημα, γίνεται προκειμένου να γνωρίζει η εφαρμογή πόσα είναι τα συνολικά έγγραφα που πρόκειται να σαρωθούν. Με αυτόν τον τρόπο θα μπορεί να ανανεώνει την μπάρα προόδου και να ενημερώνει τον χρήστη.

```

1. def get_total_docs(self):
2.     response = sio.call(event=self.EVENTS['get_total_docs_event
    '])
3.     return response

```

4.3.1.4.4 Αίτημα Ανάλυσης Εγγράφων

Αν η απάντηση του διακομιστή ήταν επιτυχής, στέλνεται εκ νέου αίτημα για ανάλυση και σάρωση των εγγράφων της αναζήτησης από τον διακομιστή.

```

1. def start_analyzing(self):
2.     print('emitted analyze event')
3.     sio.emit(event=self.EVENTS['analyze_event'])

```

4.3.1.4.5 Αίτημα Ενημέρωσης Προόδου

Το αίτημα αυτό δημιουργείται κάθε X δευτερόλεπτα (προεπιλογή 5), ώστε να υπάρχει δυνατότητα ενημέρωσης της progress bar του πελάτη. Ο διακομιστής απαντά με τον αριθμό των εγγράφων που έχει σαρώσει μέχρι εκείνη τη στιγμή.

```
1. def update_process(self):
2.     response = sio.call(event=self.EVENTS['update_event'])
3.     return response
```

4.3.1.4.6 Αίτημα Διακοπής της Αναζήτησης

Σε περίπτωση που ο χρήστης, θελήσει για οποιονδήποτε λόγο να διακόψει την σάρωση των εγγράφων (πατώντας το κουμπί «Cancel» ή απλά κλείνοντας το παράθυρο), στέλνεται αίτημα διακοπής στον server, ο οποίος διακόπτει με ασφαλή τρόπο την αναζήτηση.

```
1. def stop_operation(self):
2.     sio.emit(event=self.EVENTS['stop_event'])
3.     sio.disconnect()
```

4.3.1.4.7 Αίτημα για την Τελική Λίστα Αποτελεσμάτων

Μόλις ο πελάτης αντιληφθεί ότι ο αριθμός των εγγράφων που έχουν ελεγχθεί (μέσω του [Αιτήματος Ενημέρωσης Προόδου](#)) ισούται με τον αριθμό των συνολικών εγγράφων της αναζήτησης (που το γνωρίζει μέσω του [Αιτήματος Αποστολής Συνολικού Αριθμού Εγγράφων](#)), στέλνει το συγκεκριμένο αίτημα, και περιμένει ως απάντηση την λίστα των αποτελεσμάτων από τον διακομιστή (λίστα από dictionaries της Python).

```
1. def get_response_lst(self):
2.     response = sio.call(event=self.EVENTS['get_lst_event'])
3.     return response
```

4.3.1.4.8 Αίτημα Αποσύνδεσης από τον Διακομιστή

Όταν η διαδικασία της παραμετροποιημένης αναζήτησης ολοκληρωθεί (επιτυχώς ή ανεπιτυχώς), η εφαρμογή αποσυνδέεται από τον διακομιστή στέλνοντάς του το αίτημα αυτό. Το αίτημα αυτό στέλνεται, επίσης, και μαζί με το [Αίτημα Διακοπής της Αναζήτησης](#).

```

1. def disconnect(self):
2.     sio.disconnect()
3.     print('disconnected from server')

```

4.3.1.5 Νήματα

Η εκτέλεση των [Αιτημάτων Πελάτη προς τον Διακομιστή](#), επιτυγχάνεται μέσω των νημάτων της εφαρμογής. Ο αριθμός των νημάτων στο σύνολο είναι 2. Καθένα νήμα είναι μια ξεχωριστή κλάση, η οποία κληρονομεί από την υπερκλάση QThread. Έχουμε:

1. Το νήμα της Αναζήτησης (SearchThread)
2. Το νήμα της Ανάλυσης (AnalysisThread)

4.3.1.5.1 Νήμα Αναζήτησης

Το συγκεκριμένο νήμα αναλαμβάνει να στείλει το query string (αλφαριθμητικό αιτήματος), το οποίο το λαμβάνει έτοιμο από την γραφική διασύνδεση χρήστη (Menu UI), στον διακομιστή και να λάβει απάντηση από εκείνον, στην περίπτωση που όλα πήγαν καλά. Αν το νήμα δε λάβει απάντηση, δεν προχωρά η διαδικασία στην φάση της ανάλυσης.

```

1. class SearchThread(QtCore.QThread):
2.     '''
3.     Search thread
4.     Performs the document search in separated thread
5.     Query is generated according to user's selections (menu UI)
6.     '''
7.     stop_operation = False
8.     def __init__(self, parent=None, query='', client=None):
9.         super(SearchThread, self).__init__(parent)
10.        self.query = query
11.        self.client = client
12.        self.response = None
13.
14.        # run method gets called when we start the thread
15.        def run(self):
16.            try:
17.                self.response = self.client.make_search_request(self.query)
18.            except:
19.                return

```

4.3.1.5.2 Νήμα Ανάλυσης

Το νήμα της ανάλυσης έχει τις περισσότερες αρμοδιότητες:

- Αρχικά, είναι εκείνο που θα λάβει τον συνολικό αριθμό εγγράφων που πρόκειται να σαρωθούν, στέλνοντας [Αίτημα Αποστολής Συνολικού Αριθμού Εγγράφων](#).
- Μόλις λάβει τον αριθμό των συνολικών εγγράφων, στέλνει [Αίτημα Ανάλυσης Εγγράφων](#).
- Όσο διαρκεί η ανάλυση, το νήμα είναι υπεύθυνο για την ενημέρωση της μπάρας προόδου (progress bar) της εφαρμογής. Για να το πετύχει αυτό, θα στέλνει [Αίτημα Ενημέρωσης Προόδου](#) κάθε X δευτερόλεπτα (προεπιλογή 5) και με την τιμή που λαμβάνει κάθε φορά, ενημερώνει την μπάρα.
- Όταν η σάρωση ολοκληρωθεί, στέλνει [Αίτημα για την Τελική Λίστα των Αποτελεσμάτων](#). Την τελική λίστα αυτή, την μεταβιβάζει στην συνέχεια στο module statistics.py, για την εξαγωγή των στατιστικών.

```
1. class AnalysisThread(QtCore.QThread):
2.     '''
3.     Analysis thread
4.     Performs the document analysis in separated thread
5.     '''
6.     total_docs_update = QtCore.pyqtSignal(int)
7.     update_progress_bar = QtCore.pyqtSignal(int)
8.
9.     thread_finished = QtCore.pyqtSignal(list)
10.
11.     def __init__(self, parent=None, excel_path=None, client=None, progressBar=None):
12.         super(AnalysisThread, self).__init__(parent)
13.         self.excel_path = excel_path
14.         self.client = client
15.         self.progressBar = progressBar
16.         self.stop = False
```

```

17.     # run method gets called when we start the thread
18.     def run(self):
19.         total_docs = self.client.get_total_docs()
20.         self.total_docs_update.emit(total_docs)
21.         self.client.start_analyzing()
22.
23.         import time
24.
25.         while True:
26.             if self.stop:
27.                 return
28.             try:
29.                 response = self.client.update_process()
30.                 self.progressBar.setValue(response)
31.                 if total_docs == response:
32.                     results_lst = self.client.get_response_lst()
33.                     self.client.disconnect()
34.                     break
35.                 time.sleep(SECONDS)
36.             except:
37.                 return
38.
39.         self.thread_finished.emit([results_lst, self.excel_path])
40.
41.     def stop_analysis(self):
42.         self.stop = True

```

4.3.1.6 Στατιστικά

Για την εξαγωγή στατιστικών από συγκεκριμένα δεδομένα (είτε από την βάση δεδομένων, είτε από ξεχωριστά αρχεία) έχει δημιουργηθεί η κλάση «StatisticsExportation», η οποία δημιουργεί όλα τα επιθυμητά στατιστικά διαγράμματα. Επίσης, αναλαμβάνει και την εξαγωγή του ίδιου του αρχείου Excel, μαζί με τα συγκεντρωτικά δεδομένα. Η StatisticsExportation καλείται:

1. Όταν ολοκληρωθεί μία παραμετροποιημένη αναζήτηση (και ο χρήστης έχει επιλέξει την εξαγωγή των αποτελεσμάτων σε αρχείο)
2. Όταν έχουν επιλεγεί πολλαπλά αρχεία Excel ως είσοδοι (από την καρτέλα Import Excels)
3. Όταν επιλεγεί η εξαγωγή στατιστικών από στοιχεία της βάσης δεδομένων

```

1. class StatisticsExportation:
2.
3.     def __init__(self, from_year, to_year, agg_data=False, stat_diagrams=False, department_stats=False, outpath='C:\\export.xlsx', df=None):
4.         self.first_cat, self.second_cat, self.third_cat = import_pu
           b_data()
5.         self.from_year = from_year
6.         self.to_year = to_year
7.         self.worksheet = None
8.         self.outpath = outpath
9.         self.export = {'agg_data': agg_data, 'stat_diagrams': stat_diagrams, 'department_stats': department_stats}
10.
11.         if not df.empty:
12.             self.df = db.get_df_by_year([from_year, to_year])

```

Όπως βλέπουμε παραπάνω στον κώδικα, η κλάση παίρνει ως ορίσματα το εύρος των χρόνων, για τα οποία θα γίνει εξαγωγή στατιστικών, μια σειρά από Boolean παραμέτρους (**True** = Να εξαχθούν, **False** = Να μην εξαχθούν):

- **Agg_data:** Συγκεντρωτικά δεδομένα
- **Stat_diagrams:** Στατιστικά διαγράμματα
- **Department_stats:** Στατιστικά ανά τμήμα

Και άλλες 2 παραμέτρους:

- **Outpath:** Κατάλογος και όνομα αρχείου εξόδου
- **Df:** Το dataframe των δεδομένων. Αν τα δεδομένα, πάνω στα οποία θα βασιστούν τα στατιστικά, προέρχονται από την βάση δεδομένων, η παράμετρος αυτή θα είναι None, οπότε θα χρειαστεί να δημιουργήσει το dataframe η ίδια η κλάση, κάνοντας κατάλληλη κλήση στην βάση δεδομένων. Σε διαφορετική περίπτωση, το dataframe θα είναι έτοιμο.

4.3.1.6.1 Διάγραμμα Αριθμού Εγγράφων Ανά Χρονιά

```
1. def create_num_of_documents_per_year_plot(self):
2.     '''
3.         Creates a plot which shows the process
4.         of the number of documents over the years
5.     '''
6.     num_of_docs_by_year = self.df.groupby(['Year']).size() #
    get number of documents by year
7.
8.     years = self.df['Year'].unique() # get unique values of years from data frame
9.
10.    plt.clf()
11.
12.    plt.xlabel('Year')
13.    plt.ylabel('Number of Docs')
14.
15.    plt.gca().xaxis.set_major_locator(mticker.MultipleLocator(1
16.    )) # only integers in year axis
17.
18.    plot = plt.plot(years, num_of_docs_by_year) # create the
    plot
19.
20.    imgdata = io.BytesIO()
21.    plt.savefig(imgdata)
22.
23.    return imgdata
```

4.3.1.6.2 Διάγραμμα του Αριθμού των Top-10 Εγγράφων Ανά Χρονιά

```
1. def create_num_of_documents_per_year_plot_top_ten(self):
2.     '''
3.         Creates a plot which shows the evolution
4.         of the number of documents, whose average percentile is over
5.         90, over the years
6.     '''
7.     top_ten_rows = self.df.loc[self.df['Average Percentile'] >=
8.                                90.0]
9.     num_of_docs_per_year_top_ten = top_ten_rows.groupby(['Year'
10.                                                         ]).size() # get number of documents in top ten percentile by year
11.     years = self.df['Year'].unique() # get unique values of years from data frame
12.
13.     plt.clf()
14.
15.     plt.xlabel('Year')
16.     plt.ylabel('Number of Docs (Top 10)')
17.
18.     plt.gca().yaxis.set_major_locator(mticker.MultipleLocator(1
19.                                                         )))
20.     plt.gca().xaxis.set_major_locator(mticker.MultipleLocator(1
21.                                                         ))) # only integers in year axis
22.     plot = plt.plot(years, num_of_docs_per_year_top_ten) # create the plot
23.
24.     imgdata = io.BytesIO()
25.     plt.savefig(imgdata)
26.
27.     return imgdata
```

4.3.1.6.3 Διάγραμμα Μέσου Όρου του δείκτη CiteScore Ανά Χρονιά

```
1. def create_citescore_mean_per_year_plot(self):
2.     '''
3.     Creates a plot which shows the evolution
4.     of citescore mean over the years
5.     '''
6.
7.     citescores = self.df.groupby('Year')['CiteScore'].mean()
8.
9.     years = self.df['Year'].unique()
10.
11.     plt.clf()
12.
13.     plt.xlabel('Year')
14.     plt.ylabel('CiteScore (Mean)')
15.
16.     plt.gca().xaxis.set_major_locator(mticker.MultipleLocator(1
17. )) # only integers in year axis
18.
19.     plot = plt.plot(years, citescores) # create the plot
20.
21.     imgdata = io.BytesIO()
22.     plt.savefig(imgdata)
23.
24.     return imgdata
```

4.3.1.6.4 Διάγραμμα Μέσου Όρου Percentile Ανά Χρονιά

```
1. def create_avg_percentile_per_year_plot(self):
2.     '''
3.     Creates a plot which shows the evolution
4.     of average percentile mean over the years
5.     '''
6.
7.     citescores = self.df.groupby('Year')['Average Percentile'].
8.     mean() # find mean by year
9.
10.     years = self.df['Year'].unique()
11.
12.     plt.clf()
13.
14.     plt.xlabel('Year')
15.     plt.ylabel('Average Percentile (Mean)')
16.
17.     plt.gca().xaxis.set_major_locator(mticker.MultipleLocator(1
18. )) # only integers in year axis
19.
20.     plot = plt.plot(years, citescores) # create the plot
21.
22.     imgdata = io.BytesIO()
23.     plt.savefig(imgdata)
24.
25.     return imgdata
```

4.3.1.6.5 Διάγραμμα Μέγιστης Τιμής του Δείκτη CiteScore Ανά Χρονιά

```
1. def create_citescore_max_per_year_barplot(self):
2.     '''
3.     Creates a plot which shows the evolution
4.     of maximum CiteScore over the years
5.     '''
6.
7.     years = self.df['Year'].unique()
8.     citescore_max = self.df.groupby('Year')['CiteScore'].max()
9.
10.    plt.clf()
11.
12.    plt.xlabel('Year')
13.    plt.ylabel('Maximum CiteScore')
14.
15.    plt.gca().xaxis.set_major_locator(mticker.MultipleLocator(1
16.    )) # only integers in year axis
17.    plot = plt.bar(years, citescore_max) # create the barplot
18.
19.    imgdata = io.BytesIO()
20.    plt.savefig(imgdata)
21.
22.    return imgdata
```

4.3.1.6.6 Διάγραμμα Ανά Κατηγορία Percentile (Για Μία Χρονιά)

```
1. def create_percentile_rank_barplot(self):
2.     '''
3.         Creates a bar plot, which shows how many
4.         journals exist in each category (Average Percentile >= 95%
5.         etc.)
6.     '''
7.     categories = ['<= 50%', '50% - 79,9%', '80% - 94,99%', '>=
8.         95%']
9.     first_category = self.df.loc[self.df['Average Percentile']
10.         >= float(self.first_cat)][['Average Percentile']] # find all the j
11.         ournals with average percentile >= 95%
12.     second_category = self.df.loc[(self.df['Average Percentile']
13.         ] >= float(self.second_cat)) & (self.df['Average Percentile'] < flo
14.         at(self.first_cat))][['Average Percentile']]
15.     third_category = self.df.loc[(self.df['Average Percentile']
16.         >= float(self.third_cat)) & (self.df['Average Percentile'] < float
17.         (self.second_cat))][['Average Percentile']]
18.     fourth_category = self.df.loc[(self.df['Average Percentile']
19.         ] <= float(self.third_cat))][['Average Percentile']]
20.     values = [fourth_category.size, third_category.size, second
21.         _category.size, first_category.size]
22.     plt.clf()
23.     plt.xlabel('Average Percentile')
24.     plt.ylabel('Number of Docs')
25.     barplot = plt.bar(categories, values) # create the barpl
26.     ot
27.     imgdata = io.BytesIO()
28.     plt.savefig(imgdata)
29.     return imgdata
```

4.3.1.6.7 Στατιστικός Πίνακας Καθηγητών Πανεπιστημίου

```
1. def create_authors_overall_ranking_excel(self):
2.
3.     professors = db.fetch_professors_from_db()
4.
5.     author_lst = self.create_author_list()
6.
7.     return pd.DataFrame(self.create_final_list(professors))
```

4.3.1.6.8 Διάγραμμα Μέσου Average Percentile Καθηγητή

```
1. def create_professor_rank_by_year(self, professor):
2.     prof_name = professor['Surname']+', '+professor['Name'][:1]
3.     indexes = list(self.df['Authors'].str.find(prof_name))
4.     indexes = [i for i in range(len(indexes)) if indexes[i] !=
5.               -1]
6.     values = self.df[['Average Percentile', 'Year']].iloc[index
7.                     es]
8.     values['Name'] = prof_name
9.     tmp_df = values.groupby(['Name', 'Year'], as_index=False).a
10.    gg({'Average Percentile' : 'mean'})
11.
12.    plt.clf()
13.    plt.xlabel('Year')
14.    plt.ylabel('Average of Average Percentile')
15.
16.    plt.gca().xaxis.set_major_locator(mticker.MultipleLocator(1
17.    ))
18.
19.    plt.title(prof_name)
20.
21.    plot = plt.bar(tmp_df['Year'], tmp_df['Average Percentile']
22.    ) # create the barplot
23.
24.    imgdata = io.BytesIO()
25.    plt.savefig(imgdata)
26.
27.    return imgdata
```

4.3.2 Κώδικας Διακομιστή (Server)

4.3.2.1 Παραμετροποιήσεις Εφαρμογής

Προτού ξεκινήσουμε την ανάλυση του κώδικα του διακομιστή, αξίζει να επισημάνουμε ότι ο διακομιστής έχει παραμετροποιηθεί, με τρόπο αντίστοιχο της [Εφαρμογής Πελάτη](#), ώστε ορισμένες σημαντικές αλλαγές να μπορούν να γίνονται δίχως επέμβαση στον κώδικα.

Συγκεκριμένα, έχει προστεθεί ένα αρχείο με ρυθμίσεις (configuration file), από το οποίο η εφαρμογή λαμβάνει συγκεκριμένες ρυθμίσεις, κατά την ώρα της εκτέλεσής της. Αυτό το αρχείο ονομάζεται «settings.ini» και βρίσκεται στον κεντρικό φάκελο, μαζί με το module main.py. Το αρχείο έχει την εξής μορφή:

```
[SYSTEM_SETTINGS]

DELAY_TIME=60

SEARCH_URL=https://www.scopus.com/search/form.uri?display=advanced
    // Ο σύνδεσμος που χρειάζεται ο driver για να πραγματοποιήσει
    αναζήτηση

WEBDRIVER_PATH=\driver\chromedriver.exe    // Ο υποκατάλογος, στον
    οποίο βρίσκεται το εκτελέσιμο του WebDriver (ChromeDriver)

IP=localhost    // Η IP στην οποία θα τρέχει ο διακομιστής

PORT=5000    // Η πόρτα στην οποία θα «ακούει» ο διακομιστής

PROXY=TRUE    // Σε περίπτωση που βρισκόμαστε εκτός δικτύου του
    πανεπιστημίου, πρέπει να έχει τιμή true

HEADLESS=FALSE // Αν ο WebDriver θα λειτουργεί ως απλή διεργασία στο
    background
```

Σημείωση: Μπορούμε να τρέξουμε χωρίς κέλυφος τον browser (ως διεργασία στο background) ΜΟΝΟ αν βρισκόμαστε εντός του δικτύου του πανεπιστημίου. Δεν υπάρχει δυνατότητα ταυτόχρονης χρήσης της δυνατότητας headless μαζί με proxy.

4.3.2.2 Αρχικοποίηση εφαρμογής

Προτού αρχίσει η διαδικασία εκτέλεσης του διακομιστή, εκτελούνται ορισμένες συναρτήσεις, η οποίες βρίσκονται στο module «init.py», το οποίο βρίσκεται στον κεντρικό φάκελο. Εκεί, γίνονται αρχικοποιήσεις, οι οποίες αφορούν κυρίως τον Selenium WebDriver. Συγκεκριμένα:

- Αρχικοποιούνται διάφορες τιμές μεταβλητών, όπως αυτή της *DELAY_TIME*, *search_url*, το path του WebDriver
- Σε περίπτωση που βρισκόμαστε εκτός δικτύου του πανεπιστημίου, δημιουργείται σύνδεση με proxy, ώστε να έχουμε πλήρη πρόσβαση στα δεδομένα της βιβλιογραφικής βάσης Scopus
- Εκτελείται ο Selenium WebDriver, ώστε να είναι σε πλήρη ετοιμότητα, όταν ο server θα «ακούει» για αιτήματα
- Για να λειτουργεί ο WebDriver σε headless mode, δηλαδή να λειτουργεί χωρίς παράθυρο, μόνο ως διεργασία στο background, πρέπει κάτι το οποίο δεν δέχονται οι περισσότερες ιστοσελίδες στο Web, για λόγους ασφάλειας. Εμείς εκτελούμε μια φορά τον WebDriver για να πάρουμε σε string τον user-agent* και εν συνεχεία τον επανεκτελούμε, αλλάζοντας τον user-agent, ώστε να μην εμφανίζεται ότι τρέχουμε browser σε headless mode.

(*) **User Agent:** *Ο User Agent είναι ένα string, ουσιαστικά, το οποίο γνωστοποιεί σε κάθε website που επισκεπτόμαστε διάφορες λεπτομέρειες σχετικά με το σύστημά μας (από τι browser επισκεπτόμαστε το website, ποιο είναι το λειτουργικό μας σύστημα). Αυτό υπάρχει για να γνωρίζουν οι ιστοσελίδες πώς να προσαρμόσουν το περιεχόμενό τους, ώστε να παρουσιάζεται καλύτερα σε εμάς (για παράδειγμα αλλιώς θα παρουσιαστεί ένα website σε λειτουργικό Windows, Linux και αλλιώς σε κινητή συσκευή με λειτουργικό σύστημα Android).*

```

1. def init_browser():
2.     options = Options()
3.     with_proxy = import_settings()[3]
4.     headless = import_settings()[4]
5.     webdriver_path = import_settings()[2]
6.
7.     if with_proxy.lower() == 'true':
8.         options.add_extension(proxy.create_plugin())
9.         options.add_argument('--no-proxy-server')
10.
11.    elif headless.lower() == 'true':
12.        browser = webdriver.Chrome(options=options, executable_path=webdriver_path)
13.        user_agent = str(browser.execute_script("return navigator.userAgent;")).replace('Headless','') # remove headless from user-agent (if headless is detected by site, our requests will be denied)
14.        browser.close() # close browser with old user-agent
15.        options.add_argument('user-agent={0}'.format(user_agent)) # update the user agent
16.        options.headless = True
17.        browser = webdriver.Chrome(options=options, executable_path=webdriver_path) # open browser with new user agent
18.        browser.get(search_url)
19.    return browser

```


4.3.2.3 Εκκίνηση διακομιστή

Η εκκίνηση του διακομιστή γίνεται από την `main`, του module «`server.py`». Σε αυτό το module, εκτός από την εκκίνηση του διακομιστή, υπάρχουν και όλοι οι υποδοχείς των αιτημάτων που θα δέχεται από τον πελάτη.

```
1. if __name__ == '__main__':  
2.     ip, port = import_settings()  
3.     eventlet.wsgi.server(eventlet.listen((ip, port)), app)
```

Από την στιγμή που ο διακομιστής δεχτεί αίτημα σύνδεσης από τον πελάτη, τότε εισέρχεται σε μια σειρά από κρίσιμα τμήματα:

- [Αναζήτηση των Εγγράφων](#)
- [Ανάλυση των Εγγράφων](#)

Για να διασφαλιστεί ότι δεν θα υπάρχει κάποια παρεμβολή κατά την εκτέλεση των παραπάνω ή ότι μία από τις δύο δε θα πειράξει δεδομένα της άλλης (π.χ. την κατάσταση του Selenium WebDriver), χρησιμοποιούμε ένα Lock (από την βιβλιοθήκη `threading` της Python), το οποίο διασφαλίζει ότι οι δύο παραπάνω λειτουργίες θα εκτελεστούν ως ατομικές ενέργειες (χωρίς διακοπή ή παρεμβολή).

Όσες συναρτήσεις έχουν το notation `@sio.event`, είναι συναρτήσεις, οι οποίες διαχειρίζονται αιτήματα από τον πελάτη για λογαριασμό του διακομιστή.

4.3.2.3.1 Σύνδεση με τον Πελάτη

```
1. @sio.event  
2. def connect(sid, environ):  
3.     print('connect ', sid)
```

4.3.2.3.2 Διαχείριση Αιτήματος Αναζήτησης

```
1. @sio.event  
2. def search(sid, data):  
3.     global search_page  
4.     mutex.acquire()  
5.     search_page = SearchPage()  
6.     response = search_page.search(data, sio, browser)  
7.     mutex.release()  
8.     return response
```

4.3.2.3 Διαχείριση Αιτήματος Αποστολής Συνολικού Αριθμού Εγγράφων

```
1. @sio.event
2. def get_total_docs(sid):
3.     global doc_page
4.     mutex.acquire()
5.     doc_page = DocumentPage()
6.     response = doc_page.get_total_number_of_docs(browser)
7.     mutex.release()
8.     return response
```

4.3.2.4 Διαχείριση Αιτήματος Ανάλυσης Εγγράφων

```
1. @sio.event
2. def analyze(sid):
3.     global doc_page
4.     mutex.acquire()
5.     doc_page.analyze_documents(sio, browser, final_lst)
6.     mutex.release()
```

[Παρακάτω](#) μπορείτε να δείτε και την συνάρτηση `analyze_documents`.

4.3.2.5 Διαχείριση Αιτήματος Αποστολής της Τελικής Λίστας Αποτελεσμάτων

```
1. @sio.event
2. def get_final_lst(sid):
3.     global final_lst
4.     final_lst = sorted(final_lst, key = lambda i: i['Average Percen
5.         tile'], reverse=True)
6.     final_lst = add_id(final_lst)
7.     return final_lst
```

Επιστρέφεται ταξινομημένη, ως προς το Average Percentile

4.3.2.6 Διαχείριση Αιτήματος Ενημέρωσης Προόδου

```
1. @sio.event
2. def update(sid):
3.     global final_lst
4.     return len(final_lst)
```

Επιστρέφεται το μέγεθος της τελικής λίστας εκείνη τη στιγμή

4.3.2.7 Διαχείριση Αιτήματος Αποσύνδεσης Πελάτη

Ο πελάτης όταν αποσυνδέεται από τον διακομιστή, είτε τελειώσει η αναζήτηση και ανάλυση (επιτυχώς/ανεπιτυχώς) ή σταμάτησε την ανάλυση συνειδητά. Μετά την αποσύνδεση, επαναφέρουμε τον WebDriver στην αρχική του κατάσταση (πριν συνδεθεί με τον πελάτη) μέσω της συνάρτησης `reset()`. Η `reset` αυτό που επιτελεί συγκεκριμένα, είναι να αδειάζει την λίστα (ώστε να χρησιμοποιηθεί για επόμενες αναζητήσεις και να επαναφέρει τον Selenium WebDriver στην αρχική σελίδα αναζήτησης (`search_url`)).

```
1. @sio.event
2. def disconnect(sid):
3.     global doc_page
4.     print('disconnect ', sid)
5.     try:
6.         doc_page.stop_analysis()
7.     except:
8.         pass
9.     finally:
10.        reset()
```

```
1. def reset():
2.     global doc_page, search_page, final_lst, browser
3.     mutex.acquire()
4.     final_lst = list()
5.     browser.delete_all_cookies()
6.     browser.get(init.search_url)
7.     mutex.release()
```

4.3.2.8 Λειτουργίες Διακομιστή

4.3.2.8.1 Αναζήτηση στο Scopus

Η παρακάτω κλάση αναφέρεται στην σελίδα αναζήτησης του Scopus. Πάλι στα attributes βλέπουμε κάποια IDs και link texts, ώστε να είμαστε σε θέση να εντοπίσουμε τα απαραίτητα στοιχεία για να κάνουμε την αναζήτηση.

```
1. class SearchPage():
2.
3.     def __init__(self):
4.         self.advanced_ref_link_text = 'Advanced'
5.         self.search_field_id = 'searchfield'
6.         self.search_btn_id = 'advSearch'
7.         self.close_window_id = '_pendo-close-guide_'
8.         self.results_lst_id = 'srchResultsList'
```

Η βασική συνάρτηση της συγκεκριμένης κλάσης είναι η `search`, η οποία δέχεται ως όρισμα ένα `query` (string), το οποίο παράγεται από τον χρήστη μέσω

της γραφικής διασύνδεσης και είναι αυτό που χρησιμοποιείται για την παραμετροποιημένη αναζήτηση.

Η λογική, ξανά, είναι απλή: Βρίσκουμε το αντικείμενο του input στην ιστοσελίδα, αποστέλλουμε το query ως string και πατάμε το κουμπί “Search”, το οποίο το εντοπίζουμε μέσω του id του. Μετά την πραγματοποίηση της αναζήτησης, κατευθυνόμαστε στην σελίδα με τα έγγραφα (Document Page), την οποία αναφέρουμε παρακάτω.

```

1. def search(self, query, sio, browser):
2.     '''
3.         Searches for sources, with given query
4.         in Scopus
5.     '''
6.     window_showed = False
7.
8.     try:
9.         # pop up window that appears in some occasions
10.        close_window_btn = WebDriverWait(browser, 10).until(EC.
11.            visibility_of_element_located((By.ID, self.close_window_id)))
12.        close_window_btn.click()
13.        window_showed = True
14.    except:
15.        print('No window showed.')
16.
17.    advanced_ref = WebDriverWait(browser, init.DELAY_TIME).until(
18.        # when page is loaded, click Advanced Search
19.        EC.presence_of_element_located((By.LINK_TEXT, self.advanced_ref_link_text)))
20.    advanced_ref.click()
21.
22.    search_field = WebDriverWait(browser, init.DELAY_TIME).until(
23.        # when page is loaded, click query text box & send our query
24.        EC.visibility_of_element_located((By.ID, self.search_field_id)))
25.    search_field.clear()
26.    search_field.send_keys(query)
27.
28.    search_btn = browser.find_element_by_id(self.search_btn_id)
29.    # when query is sent, find & press the search button
30.    search_btn.click()
31.
32.    if not window_showed:
33.        try:
34.            # pop up window that appears in some occasions
35.            close_window_btn = WebDriverWait(browser, 10).until(
36.                EC.visibility_of_element_located((By.ID, self.close_window_id)))
37.            close_window_btn.click()
38.        except:
39.            print('No window showed.')
40.
41.        try:
42.            results_lst = WebDriverWait(browser, init.DELAY_TIME).until(
43.                # when page is loaded, click query text box & send our query
44.                EC.visibility_of_element_located((By.ID, self.results_list_id)))
45.            print('emitted ok')
46.            return 'ok'
47.        except Exception:
48.            print('error')
49.        sio.emit(event='search_response', data='error', namespace='/desktop client')

```

Σημείωση: Ο έλεγχος για το `window_showed`, αναφέρεται για ένα παράθυρο (pop-up), το οποίο εμφανίζεται, συνήθως, σε μία από τις σελίδες του Scopus (είτε στην σελίδα αναζήτησης, είτε στην σελίδα αποτελεσμάτων). Επειδή κάθε τέτοιο παράθυρο της ιστοσελίδας του Scopus διαθέτει ίδιο ID για το κουμπί του κλεισίματός του (disposal button), μπορούμε εύκολα να το αποφύγουμε με σιγουριά.

4.3.2.8.2 Ανάλυση Εγγράφων και Περιοδικών

Στον παρακάτω κώδικα, υπάρχει η κλάση `DocumentPage`, η οποία αντιπροσωπεύει την σελίδα με τα αποτελέσματα της αναζήτησης που έχει προηγηθεί, δηλαδή όλα τα έγγραφα με τις πηγές τους.

Η κλάση `DocumentPage` περιέχει ίσως την σημαντικότερη συνάρτηση του προγράμματος, την `analyze_documents`. Παρακάτω εξηγείται η λογική της συνάρτησης, η οποία εν τέλει θα συλλέξει όλα τα απαραίτητα δεδομένα που θέλουμε να εξάγουμε και να παρουσιάσουμε στον χρήστη:

- i. Αρχικά, η συνάρτηση εντοπίζει πόσες σελίδες (με τον προεπιλεγμένο αριθμό εγγράφων ανά σελίδα που ορίζει το `Scopus`) με έγγραφα υπάρχουν με βάση την αναζήτηση που προηγήθηκε.
- ii. Από την σελίδα που βρισκόμαστε, αποθηκεύει όλα τα δεδομένα που είναι χρήσιμα (όπως όνομα εγγράφου, έτος, συγγραφείς κλπ) σε διαφορετικές ουρές (μία ουρά για τους συγγραφείς, μία ουρά για τα ονόματα των εγγράφων κλπ), οι οποίες ουρές είναι παράλληλες (στο πρώτο στοιχείο της ουράς εγγράφων, αντιστοιχεί το πρώτο στοιχείο της ουράς των συγγραφέων και ομοίως με τα υπόλοιπα).
- iii. Αφού αποθηκεύσει τα παραπάνω, ξεκινά να επισκέπτεται ένα προς ένα τις πηγές των εγγράφων, προκειμένου να καταγράψει για κάθε μία τις μετρικές (`SJR`, `SNIP`, `CiteScore`). Δημιουργείται ένα `dictionary` (δομή δεδομένων της `Python`) για κάθε `document` και όλα τα `dictionaries` αυτά, αποθηκεύονται σε μια λίστα. Η λίστα αυτή, είναι εκείνη που τελικώς θα επιστραφεί από την συνάρτηση. Αν τη σελίδα ενός περιοδικού την έχει επισκεφτεί ξανά κατά τη διάρκεια της ανάλυσης (άρα ξέρει τους δείκτες του), δεν το ξαναεπισκέπτεται, απλά κάνει `fetch` τους δείκτες από ένα `bitmap` (αποθηκεύεται: `{<όνομα_εγγράφου>, <δείκτες>}` με κλειδί αναζήτησης το όνομα (δεν υπάρχουν δύο περιοδικά με ίδιο όνομα).
- iv. Αυτό το κάνει για όλα τα έγγραφα της σελίδας. Μόλις δεν υπάρχουν άλλα έγγραφα ή αλλιώς δεν υπάρχουν άλλα στοιχεία στις ουρές που δημιουργήσαμε, αλλάζει σελίδα και κάνει τις αρχικοποιήσεις που περιεγράφηκαν στο βήμα ii).
- v. Αν δεν υπάρχει άλλη σελίδα να επισκεφτεί, η ανάλυση σταματά και προχωράμε, πλέον, στην παρουσίαση των δεδομένων στον χρήστη, όπως και των διαφόρων στατιστικών διαγραμμάτων.

Σημείωση 1: Στον παρακάτω κώδικα παρατηρούμε στην αρχή του `while loop` έναν έλεγχο (`self.stop`). Αυτός ο έλεγχος γίνεται, καθώς παρέχεται η δυνατότητα στον χρήστη να ακυρώσει την διαδικασία πριν αυτή ολοκληρωθεί. Για να γίνει αυτό, πρέπει να σταματήσουμε το `Thread` (νήμα) που εκτελείται εκείνη την στιγμή με τρόπο ομαλό, ώστε

να μην προκληθεί ζημιά στο πρόγραμμα και να ικανοποιηθεί το αίτημα του χρήστη. Είναι ενδιαφέρον το γεγονός ότι η PyQt προσφέρει συνάρτηση `terminate()` για τον άτακτο τερματισμό των `QThreads`. Η προαναφερθείσα συνάρτηση χρησιμοποιήθηκε και στο παρόν πρόγραμμα αρχικά, αλλά δημιουργούσε πολλά προβλήματα μετέπειτα, με αποτέλεσμα να καθιστά το πρόγραμμα, εν τέλει, μη λειτουργικό.

Σημείωση 2: Στον παρακάτω κώδικα υπάρχουν πολλές βοηθητικές συναρτήσεις, οι οποίες παραλείπονται.

```
1. class DocumentPage():
2.
3.     def __init__(self):
4.         self.percentile_categories_class_name = 'treeLineContainer'
5.
6.         self.percentiles_xpath = '//*[@contains(@class, "pull-
           left paddingLeftQuarter")]'
7.         self.metric_values_xpath = "//*[@contains(@class, 'value fon
           tMedLarge lineHeight2 blockDisplay')]"
8.         self.paging_ul_class_name = 'pagination'
9.         self.doc_source_class_name = 'ddmDocSource'
10.        self.search_results_table_id = 'srchResultsList'
11.        self.doc_title_class_name = 'ddmDocTitle'
12.        self.authors_list_class_name = 'ddmAuthorList'
13.        self.pub_year_class_name = 'ddmPubYr'
14.        self.docs_total_number = 'resultsCount'
15.        self.stop = False
16.        self.count = 0
```

```
1. def analyze_documents(self, sio, browser, final_lst):
2.     '''
3.     Scans every source & gets its rating
4.     Saves all percentiles, average of percentiles and name of s
   ource
5.     in a dictionary, which is appended in a list of dictionarie
   s (all sources)
6.     '''
7.     curr_page = 1    # begin with page 1
8.
9.     no_of_pages = self.get_number_of_pages(browser) # get total
   number of pages
10.    document_rows = self.get_document_rows(browser) # get all d
   ocument names
11.    author_rows = self.get_author_rows(browser) # get all autho
   r names
12.    source_rows = self.get_source_rows(browser)    # get all so
   urce names
13.    year_rows = self.get_year_rows(browser) # get all years
14.    total_docs = self.get_total_number_of_docs(browser)    # ge
   t the number of total docs
15.    # sio.emit(event='total_docs', data=total_docs, namespace='
   /desktop_client')
16.    self.count=1
17.    while True:
18.
19.        if self.stop == True:
20.            break
21.    --
```

```

22.         try:
23.             document_name = document_rows.popleft() # pop the f
                irst one in list
24.             author_list = author_rows.popleft()
25.             source_name = source_rows.popleft()
26.             year = year_rows.popleft()
27.
28.             doc_dict = next((src for src in final_lst if src['S
                ource Name'] == source_name), None)
29.
30.             if doc_dict is not None:
31.                 document_dict = self.create_dict(self.count, do
                    cument_name, doc_dict['Source Name'], year, author_list,
32.                 self.get_number_of_authors(
                    author_list), doc_dict['Average Percentile'], zip(['CiteScore', 'SJ
                    R', 'SNIP'], [doc_dict['CiteScore'], doc_dict['SJR'], doc_dict['SNI
                    P']]))
33.                 final_lst.append(document_dict)
34.                 self.count+=1
35.                 continue
36.
37.             if source_name['clickable']:
38.                 source = WebDriverWait(browser, init.DELAY_TIME
                    ).until(
39.                     EC.presence_of_element_located((By.LINK_TEX
                        T, source_name['name']))) # go in document's page
40.                 browser.execute_script("arguments[0].click();",
                    source) # javascript click
41.                 try:
42.                     categories = WebDriverWait(browser, init.DE
                        LAY_TIME).until(
43.                         EC.presence_of_all_elements_located((By
                            .CLASS_NAME, self.percentile_categories_class_name))) # find cat
                            egories names
44.                     categories = self.convert_to_txt(categories
                        ) # convert categories from web element to string
45.                     try:
46.                         percentiles = WebDriverWait(browser, in
                            it.DELAY_TIME).until(
47.                             EC.presence_of_all_elements_located
                                ((By.XPATH, self.percentiles_xpath))) # find percentiles
48.                         percentiles = self.percentiles_to_num(s
                            elf.convert_to_txt(percentiles)) # convert percentiles to number
                                (int)
49.                         print(percentiles)
50.                     except:
51.                         print('no percentiles found')

```



```

52. try:
53.             try:
54.                 citescore_element = WebDriverWait(browser, init.DELAY_TIME).until(
55.                     EC.presence_of_element_located((
56.                         By.XPATH, '//*[@id="rpCard"]/h2/span'))
57.                     citescore = citescore_element.text
58.             except:
59.                 citescore = 0
60.             try:
61.                 sjr_element = WebDriverWait(browser,
62.                     init.DELAY_TIME).until(
63.                         EC.presence_of_element_located((
64.                             By.XPATH, '//*[@id="sjrCard"]/h2/span'))
65.                         sjr = sjr_element.text
66.             except:
67.                 sjr = 0
68.             try:
69.                 snip_element = WebDriverWait(browser,
70.                     init.DELAY_TIME).until(
71.                         EC.presence_of_element_located((
72.                             By.XPATH, '//*[@id="snipCard"]/h2/span'))
73.                         snip = snip_element.text
74.             except:
75.                 snip = 0
76.                 metric_values = [citescore, sjr, snip]
77.
78.             except:
79.                 print('no metric values found')
80.                 document_dict = self.create_dict(self.count,
81.                     document_name, source_name['name'], year, author_list, self.get_number_of_authors(author_list), self.get_average_percentile(percentiles), zip(['CiteScore', 'SJR', 'SNIP'], metric_values))
82.                 final_lst.append(document_dict)
83.                 print(document_dict)
84.                 self.count+=1
85.                 browser.execute_script("window.history.go(-1)") # go to the previous page

```

```

79. except:
80.         document_dict = self.create_dict(self.count
      , document_name, source_name['name'], year, author_list, self.get_n
      umber_of_authors(author_list), 0, zip(['CiteScore', 'SJR', 'SNIP'],
      [0, 0, 0]))
81.         final_lst.append(document_dict)
82.         self.count+=1
83.         print(document_dict)
84.         browser.execute_script("window.history.go(-
      1)") # go to the previous page
85.         else:
86.         document_dict = self.create_dict(self.count, do
      cument_name, source_name['name'], year, author_list, self.get_numbe
      r_of_authors(author_list), 0, zip(['CiteScore', 'SJR', 'SNIP'], [0,
      0, 0]))
87.         final_lst.append(document_dict)
88.         self.count+=1
89.         print(document_dict)
90.         except:
91.         try:
92.             if curr_page < no_of_pages:
93.                 curr_page = self.change_page(curr_page, bro
      wser) # change page
94.                 document_rows = self.get_document_rows(brow
      ser) # get all document names
95.                 author_rows = self.get_author_rows(browser)
      # get all author names
96.                 source_rows = self.get_source_rows(browser)
      # get all source names
97.                 year_rows = self.get_year_rows(browser) # g
      et all years
98.         else:
99.             break
100.        except:
101.            break

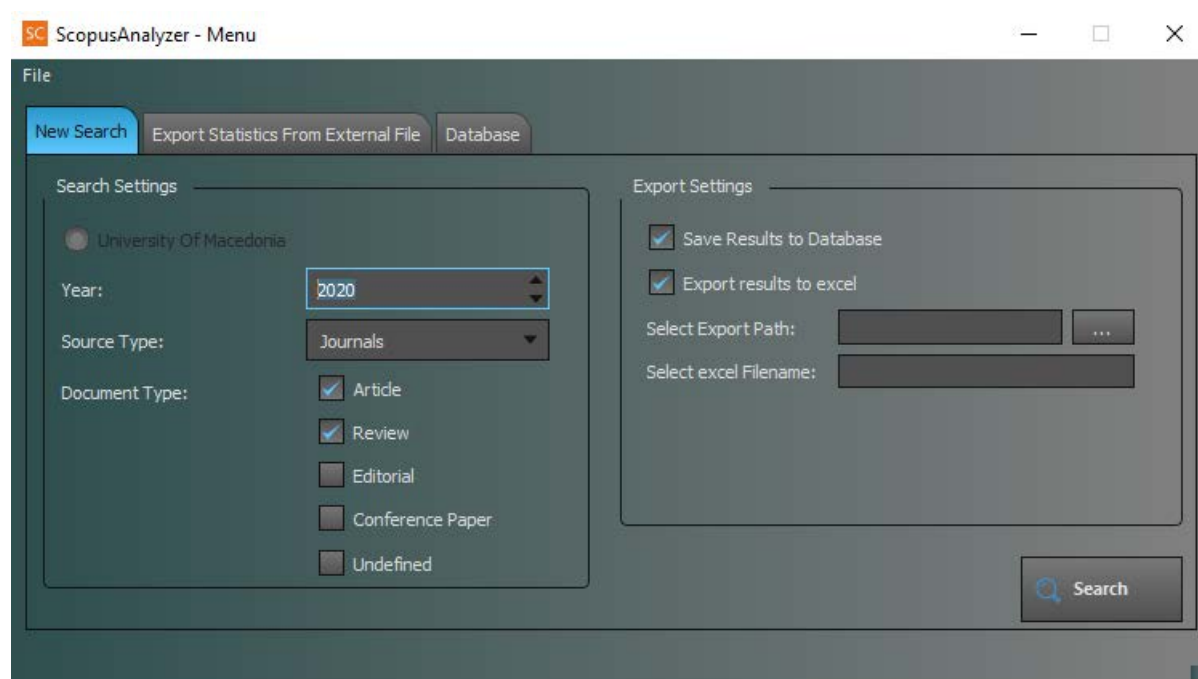
```

5 Επίδειξη Λειτουργικότητας Συστήματος

Στο συγκεκριμένο κεφάλαιο θα γίνει παρουσίαση της λειτουργικότητας του συστήματος, και συγκεκριμένα της εφαρμογής πελάτη ScopusAnalyzer.

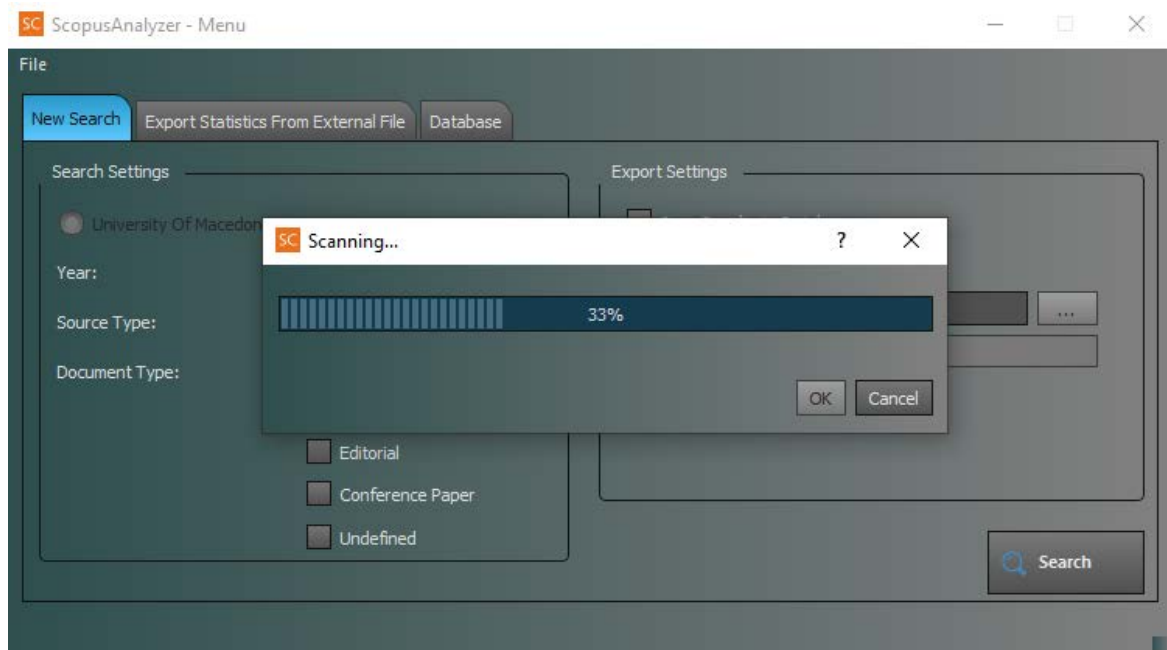
5.1 Παραμετροποιημένη Αναζήτηση

5.1.1 Εκτέλεση Παραμετροποιημένης Αναζήτησης



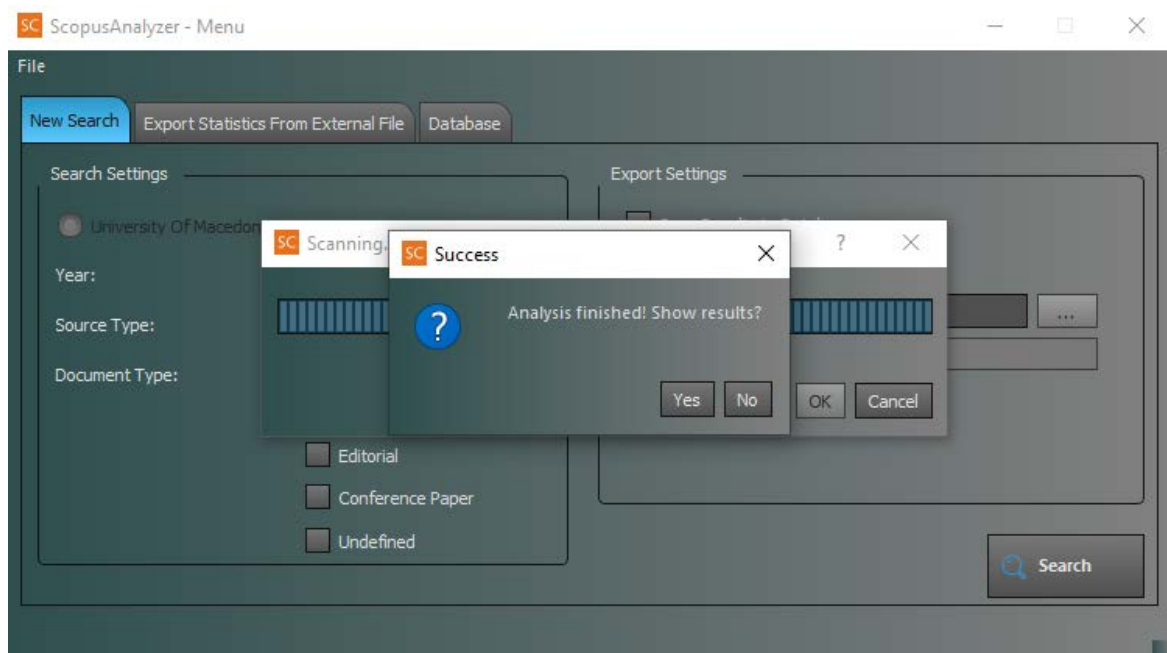
Εικόνα 5-1: Μενού Εφαρμογής Πελάτη

Όπως φαίνεται παραπάνω, η αρχική οθόνη της εφαρμογής είναι η οθόνη του μενού. Ξεκινάμε με την καρτέλα «New Search», από την οποία μπορούμε να καθορίσουμε τις ρυθμίσεις για μια παραμετροποιημένη αναζήτηση στην βιβλιογραφική βάση Scopus. Επίσης, μπορούμε να επιλέξουμε αν θα αποθηκευτούν τα αποτελέσματα της αναζήτησης αυτής σε κάποιο αρχείο Excel, ορίζοντας το όνομα του αρχείου και την διαδρομή (path) στην οποία θα εξαχθεί. Τέλος, ο χρήστης μπορεί να επιλέξει αν θα αποθηκεύσει τα αποτελέσματα και στην τοπική βάση δεδομένων της εφαρμογής για μελλοντική χρήση.



Εικόνα 5-2: Σάρωση εγγράφων

Εφόσον ο χρήστης επιλέξει τις ρυθμίσεις και πατήσει το κουμπί «Search», η εφαρμογή θα στείλει αίτημα με το αλφαριθμητικό αναζήτησης που θα έχει δημιουργήσει (με βάση τις επιλογές του χρήστη) στον διακομιστή, ο οποίος με τη σειρά του θα αναλάβει την επικοινωνία με την βιβλιογραφική βάση του Scopus, μέσω του WebDriver. Η μπάρα προόδου ανανεώνεται κάθε X δευτερόλεπτα (προεπιλογή τα 5 sec).



Εικόνα 5-3: Ολοκλήρωση της Αναζήτησης

Μόλις ολοκληρωθεί η αναζήτηση, εμφανίζεται κατάλληλο μήνυμα στον χρήστη, δίνοντάς του την επιλογή να εμφανίσει τα αποτελέσματα εντός της εφαρμογής.

5.1.2 Αποτελέσματα Παραμετροποιημένης αναζήτησης

SC Analysis Results

	#	# Authors	Authors	Average Percentile	CiteScore	Document Name	SJR	SNIP
1	1	2	Tampakoudis, I., Anagnostopoul...	98.0	7.93	The effect of mergers and ...	2.166	2.488
2	2	3	Kottas, A.T., Bozoudis, M.N., ...	97.0	7.54	Turbofan aero-engine efficienc...	3.292	3.009
3	3	3	Karakostas, P., Sifaleras, A., ...	94.67	6.36	Adaptive variable neighborhood ...	1.190	2.696
4	4	3	Metallidou, C.K., Psannis, K.E., ...	92.67	4.96	Energy Efficiency in Smart ...	0.609	1.718
5	5	3	Souravlas, S., Sifaleras, A., ...	92.67	4.96	Hybrid CPU-GPU Community ...	0.609	1.718
6	6	5	Bibi, S., Zozas, I., Ampatzoglou, A., ...	92.67	4.96	Crowdsourcing in Software ...	0.609	1.718

Back

Εικόνα 5-4: Αποτελέσματα Αναζήτησης

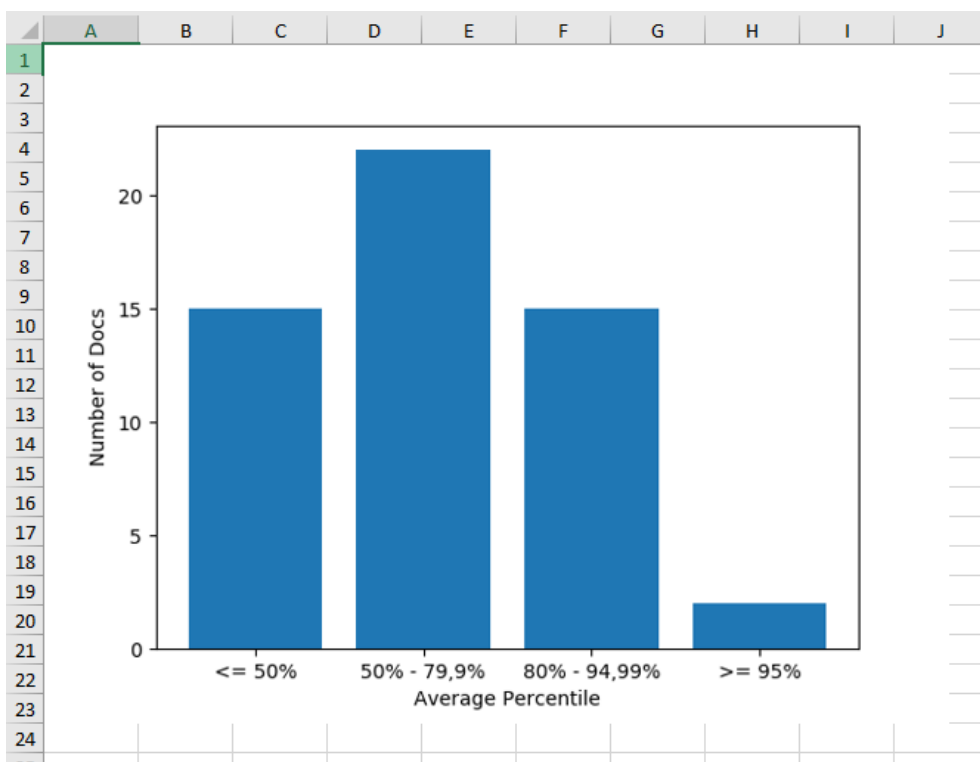
Τα αποτελέσματα εμφανίζονται σε νέο παράθυρο, όπου ο χρήστης έχει την δυνατότητα να περιηγηθεί σε αυτά, ταξινομώντας τα ανά στήλη.

Αν ο χρήστης επέλεξε την εξαγωγή σε αρχείο μπορεί να το ανοίξει με οποιαδήποτε εφαρμογή υποστηρίζει το διάβασμα και την επεξεργασία λογιστικών φύλλων (η μορφή στην οποία αποθηκεύεται είναι .xlsx).

	A	C	D	E	F	G	H	I	J	K
		# Authors	Authors	Average Percentile	CiteScore	Document Name	SJR	SNIP	Source Name	Year
1	0	2	Tampakoudis, I., Anagnostopoulou, E.	98	7.93	mental, social and governance perform	2.166	2.488	ness Strategy and the Environn	2020
2	1	3	Kottas, A.T., Bozoudis, M.N., Madas, M.A.	97	7.54	valuation: An integrated approach us	3.292	3.009	Omega (United Kingdom)	2020
3	2	3	Karakostas, P., Sifaleras, A., Georgiadis, M.C.	94,67	6,36	on methods for the fleet size and mix	1,190	2,696	xpert Systems with Application	2020
4	3	3	Metallidou, C.K., Psannis, K.E., Egyptiadou, E.A.	92,67	4,96	Efficiency in Smart Buildings: IoT App	0,609	1,718	IEEE Access	2020
5	4	3	Souravlas, S., Sifaleras, A., Katsavounis, S.	92,67	4,96	GPU Community Detection in Weight	0,609	1,718	IEEE Access	2020
6	5	5	ozas, I., Ampatzoglou, A., (...), Kalampokis, G., St	92,67	4,96	are Development: Empirical Support	0,609	1,718	IEEE Access	2020
7	6	3	Skaperas, S., Mamatas, L., Chorti, A.	92,67	4,96	Detection of Changes in the Variance	0,609	1,718	IEEE Access	2020
8	7	3	droutsopoulos, K.N., Manousakis, E.G., Madas, M	92,25	4,98	olving a bi-objective airport slot sche	2,205	2,455	ean Journal of Operational Res	2020

Εικόνα 5-5: Αποτελέσματα Αναζήτησης στο Excel

Το πρώτο φύλλο (Aggregated Data) έχει όλα τα δεδομένα που άντλησε το σύστημα από την βιβλιογραφική βάση του Scopus. Τα αποτελέσματα στέλνονται από τον διακομιστή ταξινομημένα ως προς το Average Percentile.



Εικόνα 5-6: Αριθμός Δημοσιεύσεων Ανά Κατηγορία

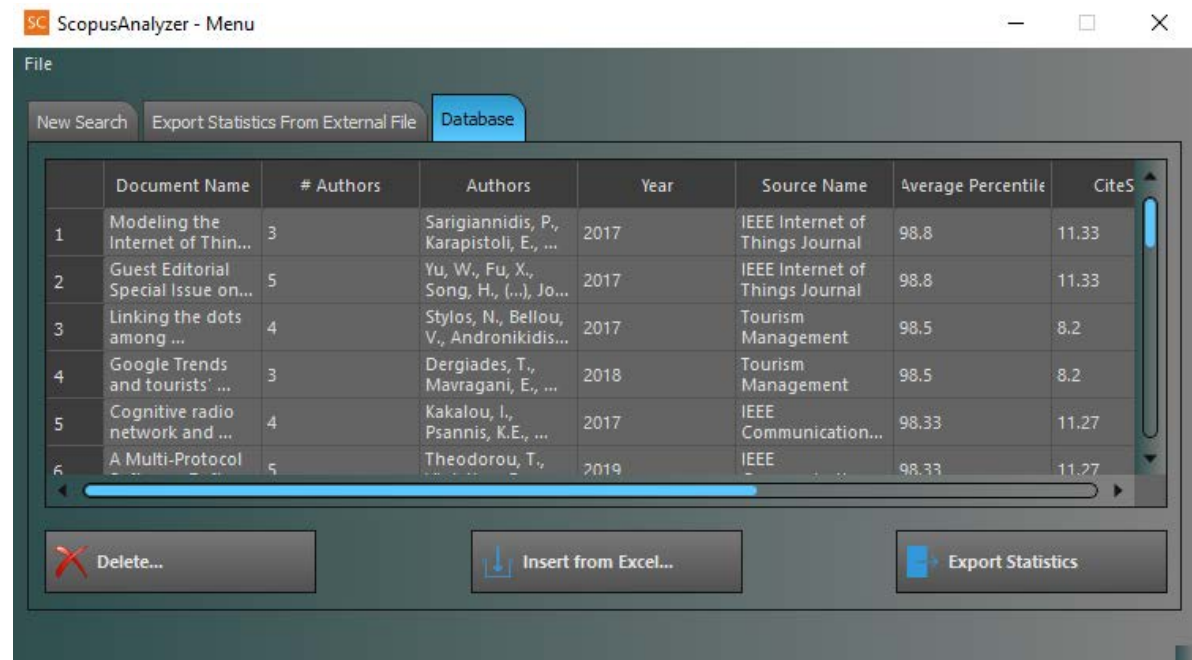
Το δεύτερο φύλλο (Average Percentile Rank), περιέχει ένα διάγραμμα (barplot), το οποίο παρουσιάζει το πλήθος των περιοδικών ανά κατηγορία (οι κατηγορίες φαίνονται στον άξονα των X).

	A	B	C	D	E
1		Name	Department	Ranking	Years
2	0	Alexander Chatzige	Applied Informatics	0	2020
3	1	Georgios Evangelidi	Applied Informatics	53	2020
4	2	Christos Georgiadis	Applied Informatics	0	2020
5	3	Dimitrios Hristu-Var	Applied Informatics	0	2020
6	4	Konstantinos Marga	Applied Informatics	83	2020
7	5	Ioannis Mavridis	Applied Informatics	0	2020
8	6	Ioannis Refanidis	Applied Informatics	0	2020
9	7	Manos Roumeliotis	Applied Informatics	0	2020
10	8	Nikolaos Samaras	Applied Informatics	0	2020

Εικόνα 5-7: Στατιστικά Καθηγητών Ανά Τμήμα

Το τρίτο και τελευταίο φύλλο, περιέχει τα rankings των καθηγητών ανά τμήμα (μέσο όρο των Average Percentiles της χρονιάς που έγινε η αναζήτηση). Τα ονόματα των καθηγητών υπάρχουν αποθηκευμένα στην τοπική βάση δεδομένων της εφαρμογής και γίνεται αντιστοίχιση με τους συγγραφείς που υπάρχουν στα

αποτελέσματα (aggregated data). Αυτή τη στιγμή υποστηρίζεται η εξαγωγή των συγκεκριμένων στατιστικών μόνο για το τμήμα Εφαρμοσμένης Πληροφορικής.



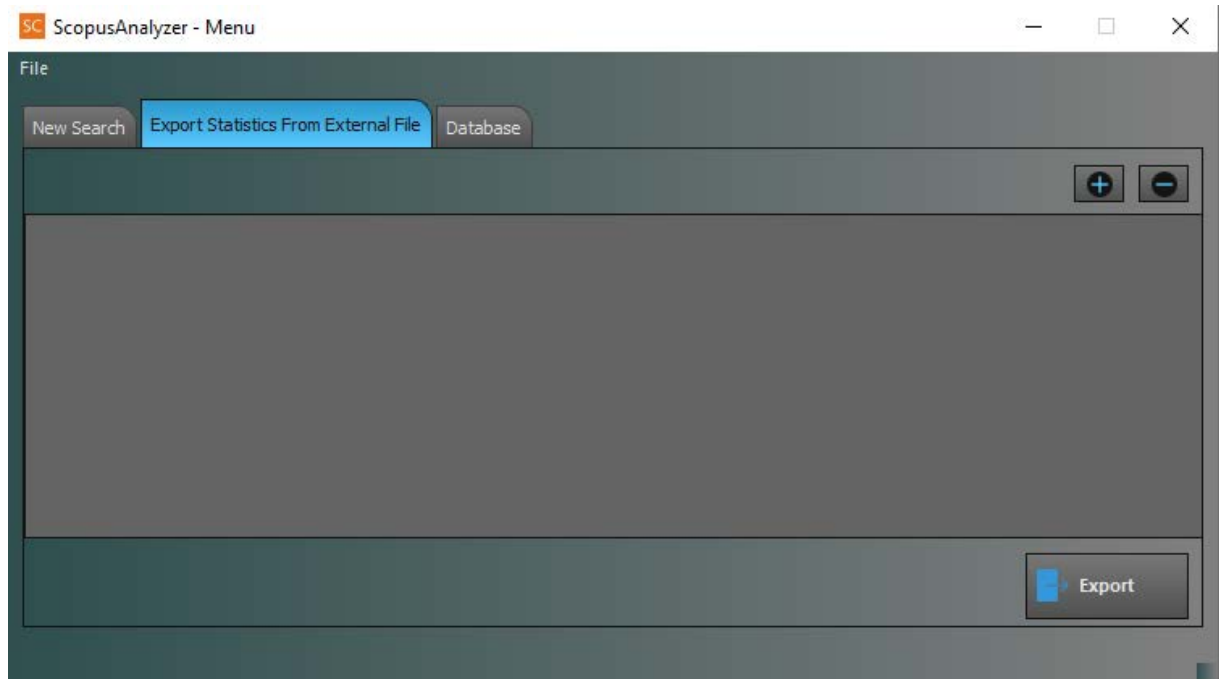
The screenshot shows the 'ScopusAnalyzer - Menu' application window. The 'Database' tab is selected, displaying a table with 8 columns: Document Name, # Authors, Authors, Year, Source Name, Average Percentile, and CiteS. The table contains 6 rows of data. Below the table are three buttons: 'Delete...', 'Insert from Excel...', and 'Export Statistics'.

	Document Name	# Authors	Authors	Year	Source Name	Average Percentile	CiteS
1	Modeling the Internet of Thin...	3	Sarigiannidis, P., Karapistoli, E., ...	2017	IEEE Internet of Things Journal	98.8	11.33
2	Guest Editorial Special Issue on...	5	Yu, W., Fu, X., Song, H., (...), Jo...	2017	IEEE Internet of Things Journal	98.8	11.33
3	Linking the dots among ...	4	Stylos, N., Bellou, V., Andronikidis...	2017	Tourism Management	98.5	8.2
4	Google Trends and tourists' ...	3	Dergiades, T., Mavragani, E., ...	2018	Tourism Management	98.5	8.2
5	Cognitive radio network and ...	4	Kakalou, I., Psannis, K.E., ...	2017	IEEE Communication...	98.33	11.27
6	A Multi-Protocol	5	Theodorou, T., ...	2019	IEEE ...	98.33	11.27

Εικόνα 5-8: Καρτέλα Βάσης Δεδομένων

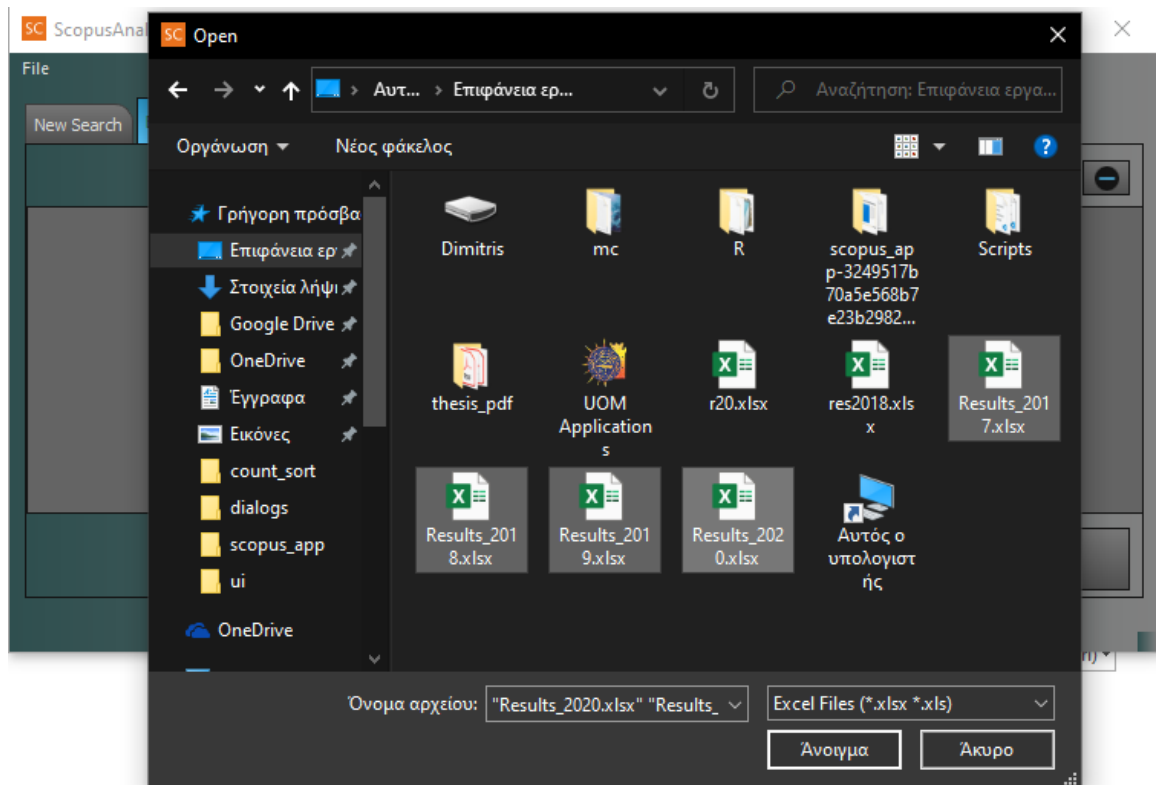
Σε περίπτωση που ο χρήστης επέλεξε να αποθηκευτούν τα αποτελέσματα της παραμετροποιημένης αναζήτησης και στην τοπική βάση δεδομένων, μπορεί να μεταβεί στην καρτέλα «Database» και να περιηγηθεί στα δεδομένα, ταξινομώντας τα με βάση την στήλη της αρεσκειάς του.

5.2 Εξαγωγή Στατιστικών από Εξωτερικά Αρχεία



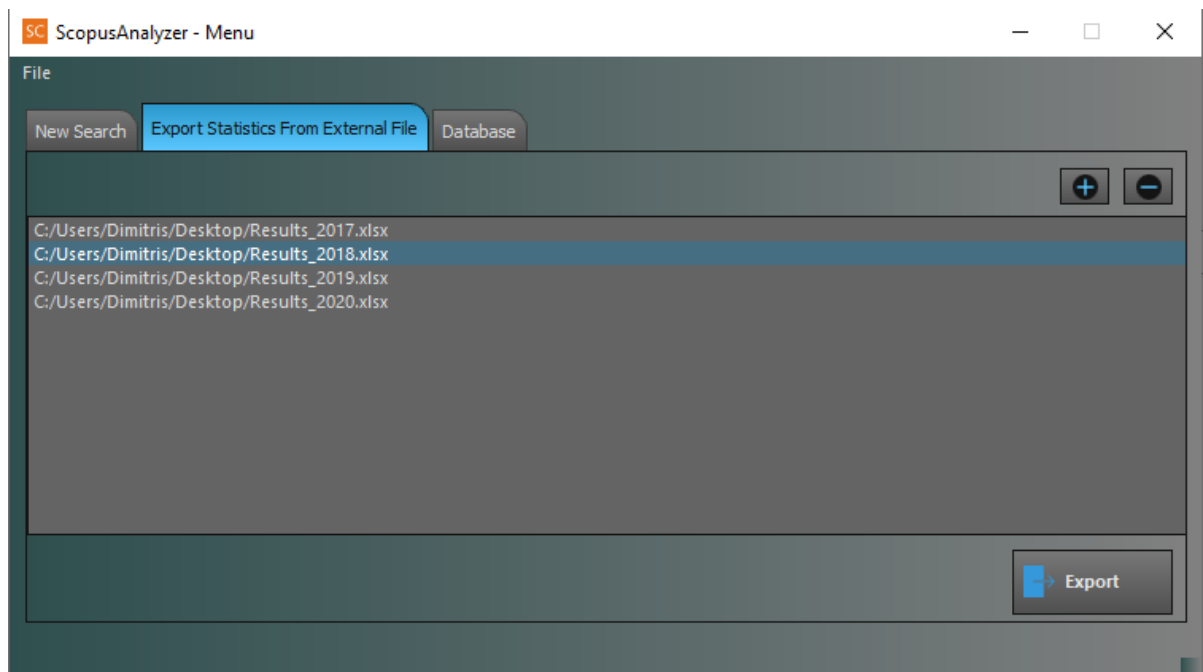
Εικόνα 5-9: Εισαγωγή Αρχείων 1

Ο χρήστης, μέσω της καρτέλας «Export Statistics From External File», έχει την δυνατότητα να εξάγει στατιστικά από εξωτερικά αρχεία Excel, τα οποία έχουν δημιουργηθεί από την ίδια εφαρμογή (από παραμετροποιημένες αναζητήσεις).



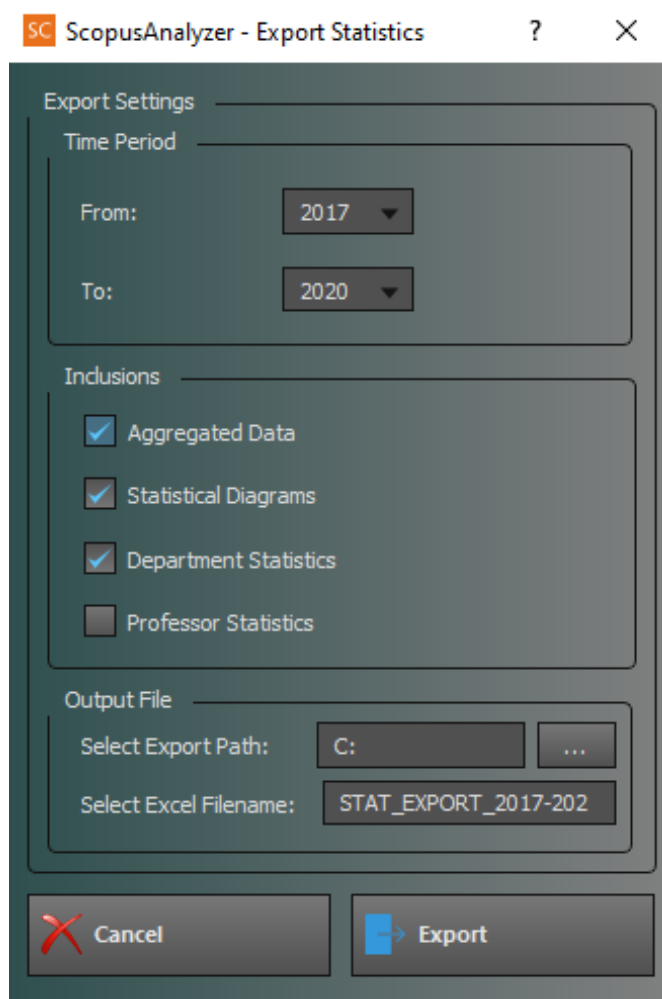
Εικόνα 5-10: Εισαγωγή Αρχείων 2

Προκειμένου να προσθέσει αρχεία, μπορεί να πατήσει το κουμπί προσθήκης «+» ή να σύρει απευθείας στο κενό που βρίσκεται κάτω από τα κουμπιά προσθήκης και αφαίρεσης.



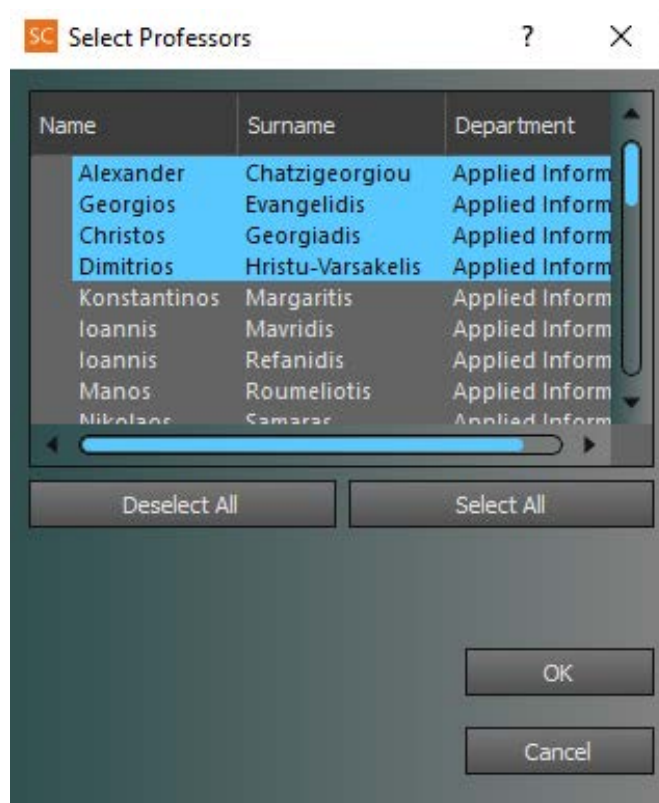
Εικόνα 5-11: Αρχεία που Εισήχθησαν

Μόλις προστεθούν τα αρχεία στην λίστα, ο χρήστης μπορεί να πατήσει το κουμπί Export, ώστε να του εμφανιστεί το παράθυρο επιλογής ρυθμίσεων εξαγωγής.



Εικόνα 5-12: Ρυθμίσεις Εξαγωγής Στατιστικών

Από το παράθυρο ρυθμίσεων, ο χρήστης μπορεί να επιλέξει συγκεκριμένες χρονιές, για τις οποίες θα εξαγάγει τα στατιστικά, όπως επίσης και το τι θα συμπεριληφθεί στο νέο αρχείο που θα προκύψει. Τα aggregated data είναι τα δεδομένα, με τα οποία θα παραχθούν τα στατιστικά, τα statistical diagrams είναι όλα τα διαγράμματα, τα οποία θα δείχνουν μεταβολή διαφόρων δεικτών και μετρικών στον χρόνο. Καθένα από τα παραπάνω θα αποθηκευτεί σε διαφορετικό φύλλο (sheet) στο νέο αρχείο που θα εξαχθεί.



Εικόνα 5-13: Επιλογή Καθηγητών για Εξαγωγή Στατιστικών τους

Αν ο χρήστης επιλέξει να εξάγει στατιστικά καθηγητών του Πανεπιστημίου Μακεδονίας, θα του εμφανιστεί ένα νέο παράθυρο. Στο πάνω μέρος του παραθύρου, υπάρχει μια λίστα με τους καθηγητές του Πανεπιστημίου Μακεδονίας (για την ώρα μόνο του τμήματος Εφαρμοσμένης Πληροφορικής). Από αυτούς, ο χρήστης μπορεί να επιλέξει όποιους θέλει, προκειμένου να εξάγει διάγραμμα, το οποίο θα απεικονίζει τον μέσο όρο του Average Percentile των περιοδικών των εγγράφων που έχει συμμετάσχει ως συγγραφέας ανά χρόνο για τον συγκεκριμένο καθηγητή.

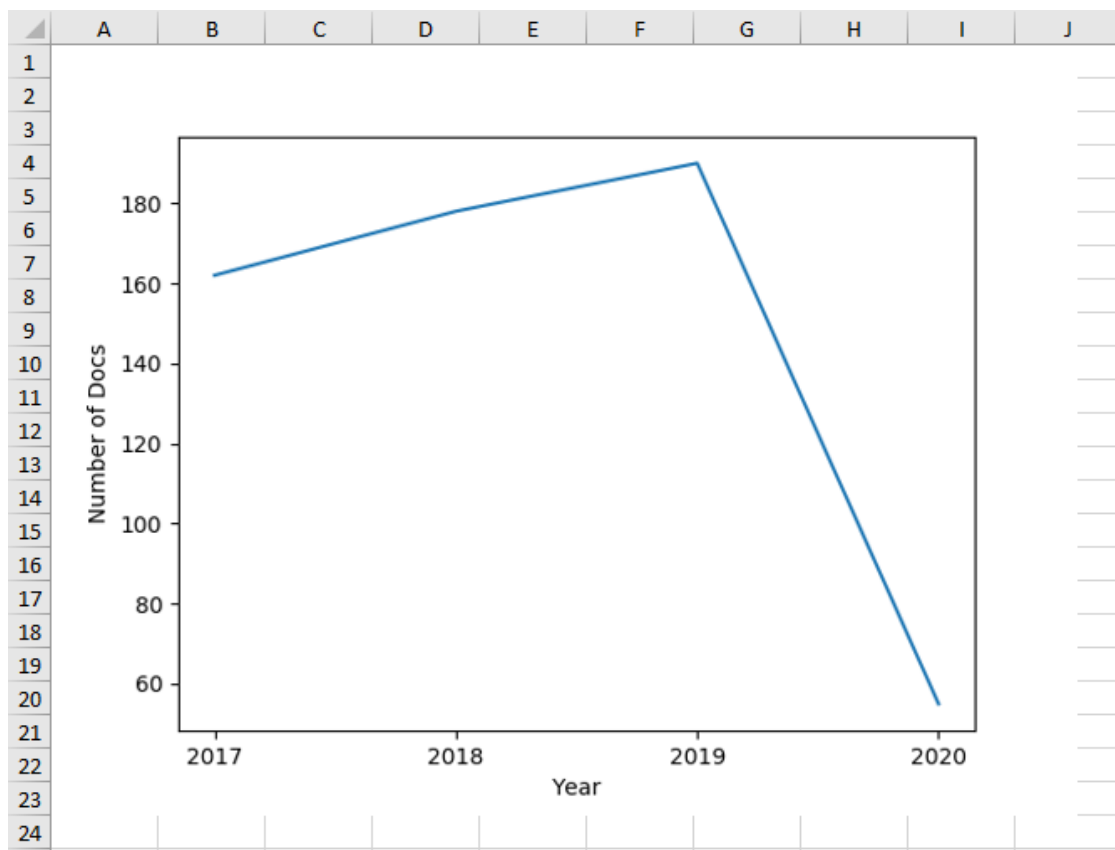
Ο χρήστης, όταν επιστρέψει στο προηγούμενο παράθυρο, μπορεί να πατήσει το κουμπί «Export», ώστε να εξάγει σε αρχείο τα στατιστικά, αφού πρώτα μπορεί να επιλέξει το όνομα του αρχείου και την διαδρομή, στην οποία θα αποθηκευτεί.

Εμείς για το παράδειγμά μας, θα επιλέξουμε την εξαγωγή όλων των διαθέσιμων στατιστικών, για όλα τα χρόνια.

	A	B	C	D	E	F	G	H	I	J	
1		Document Name	# Authors	Authors	Year	Source Name	Page	Perce	CiteScore	SJR	SNIP
2	0	Modeling the Inte	3	Sarigianni	2017	IEEE Intern	98,8	11,33	1,396	3,874	
3	1	Guest Editorial Spe	5	Yu, W., Fu	2017	IEEE Intern	98,8	11,33	1,396	3,874	
4	2	Google Trends and	3	Dergiades	2018	Tourism M	98,5	8,2	2,924	3,374	
5	3	Linking the dots ar	4	Stylos, N.,	2017	Tourism M	98,5	8,2	2,924	3,374	
6	4	Cognitive radio ne	4	Kakalou, I	2017	IEEE Comr	98,33	11,27	2,373	4,681	
7	5	A Multi-Protocol S	5	Theodoro	2019	IEEE Comr	98,33	11,27	2,373	4,681	
8	6	The effect of merg	2	Tampakou	2020	Business S	98	7,93	2,166	2,488	
9	7	Mobile-based asse	2	Nikou, S.A	2018	Computer	98	7,72	2,323	3,797	

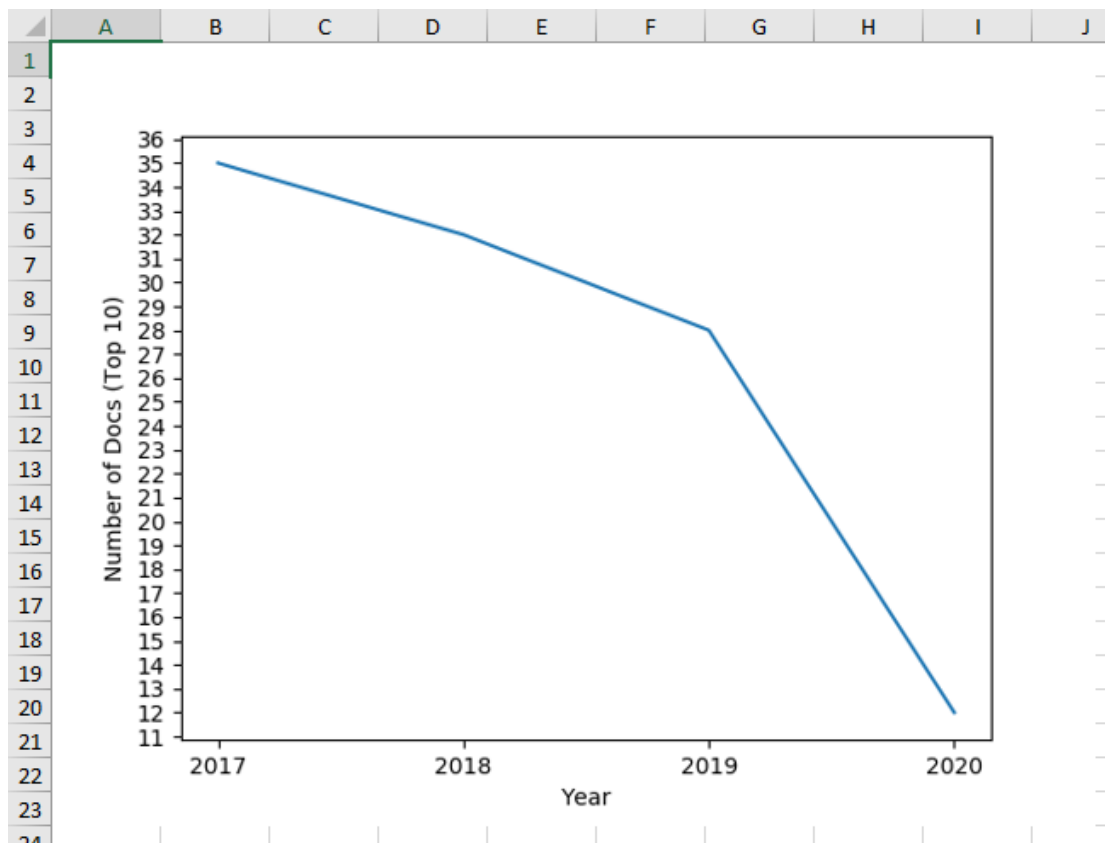
Εικόνα 5-14: Συγκεντρωτικά Δεδομένα

Αν ανοίξουμε το αρχείο, στο αρχικό φύλλο θα δούμε τα συγκεντρωτικά δεδομένα (aggregated data).



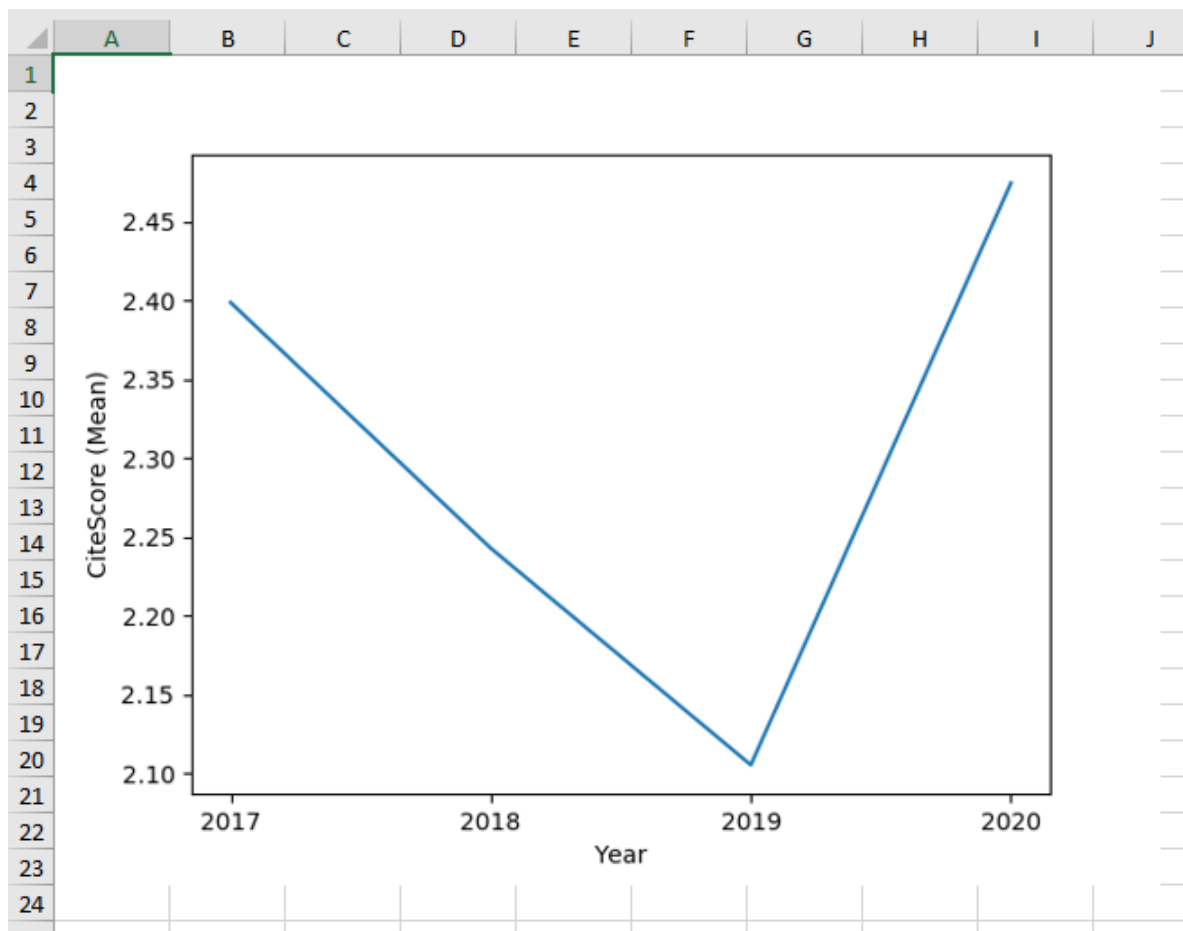
Εικόνα 5-15: Αριθμός Δημοσιεύσεων Ανά Χρονιά

Στο δεύτερο φύλλο «Documents Per Year», θα δούμε ένα διάγραμμα (line chart), το οποίο δείχνει τον συνολικό αριθμό εγγράφων ανά χρόνο.



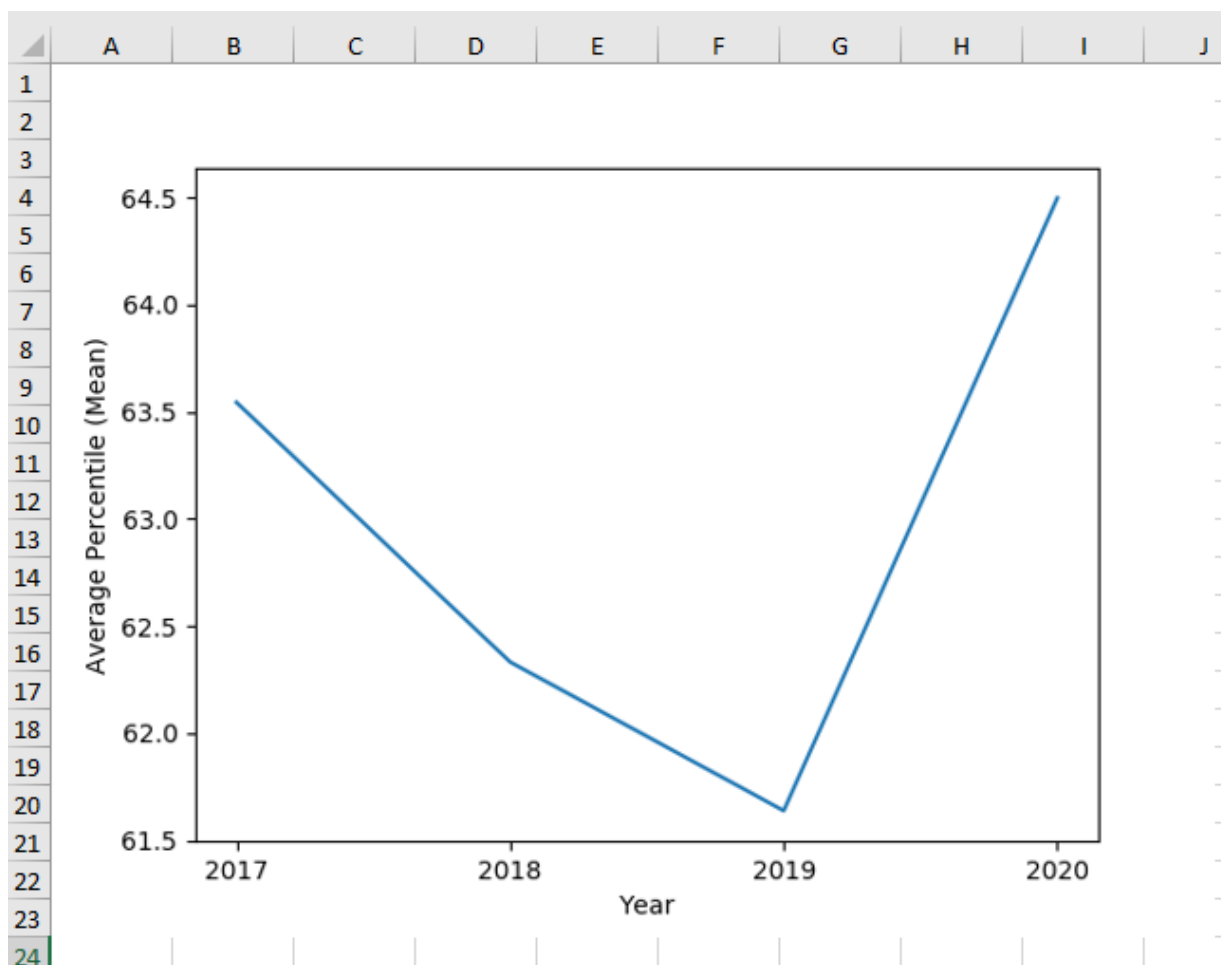
Εικόνα 5-16: Αριθμός Top-10 Περιοδικών Ανά χρονιά

Στο τρίτο φύλλο «Top Ten Per Year», έχουμε πάλι ένα line chart, το οποίο απεικονίζει τον συνολικό αριθμό περιοδικών στο άνω 10% ανά χρόνο.



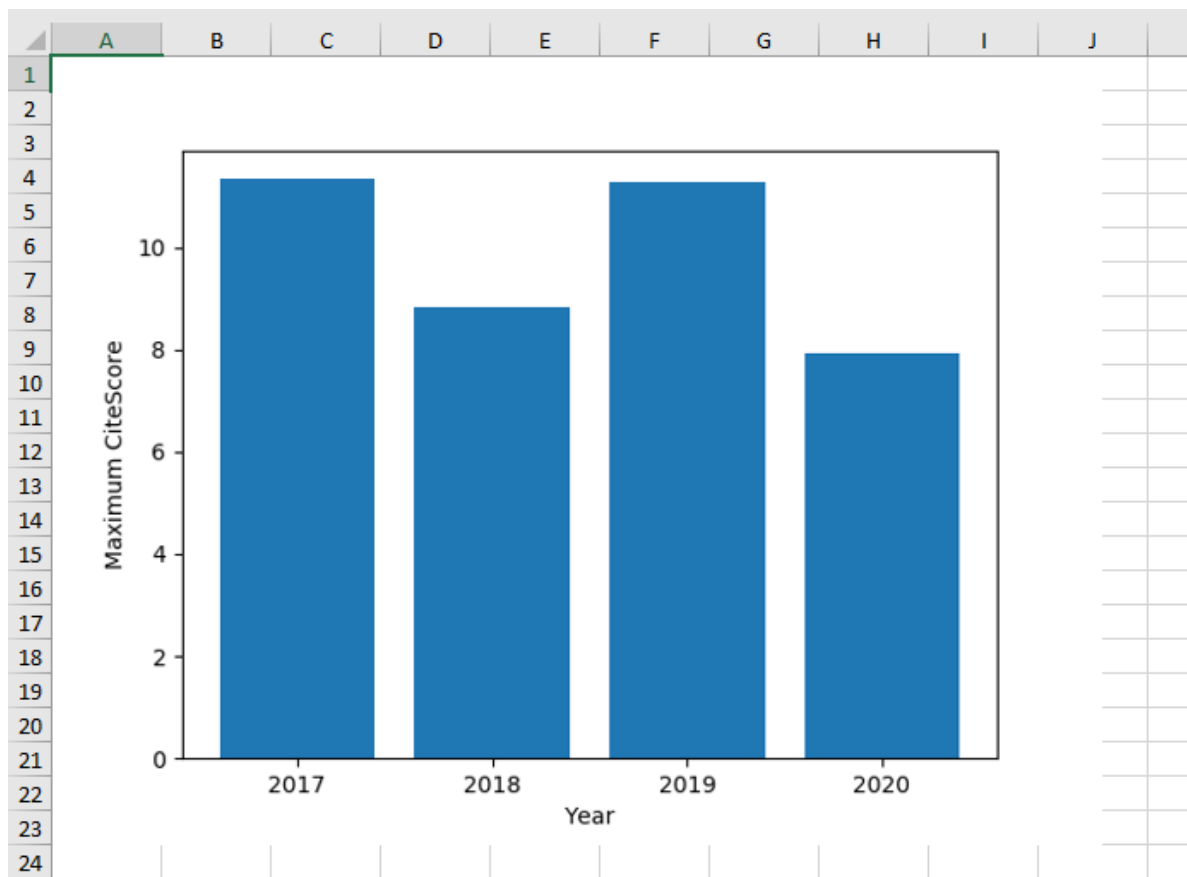
Εικόνα 5-17: Μέσος όρος της Τιμής του CiteScore Ανά Χρονιά

Στο τέταρτο φύλλο «CiteScore Mean Per Year», βρίσκεται ένα line chart, το οποίο δείχνει το μέσο CiteScore ανά χρόνο.



Εικόνα 5-18: Μέσο Average Percentile Ανά Χρονιά

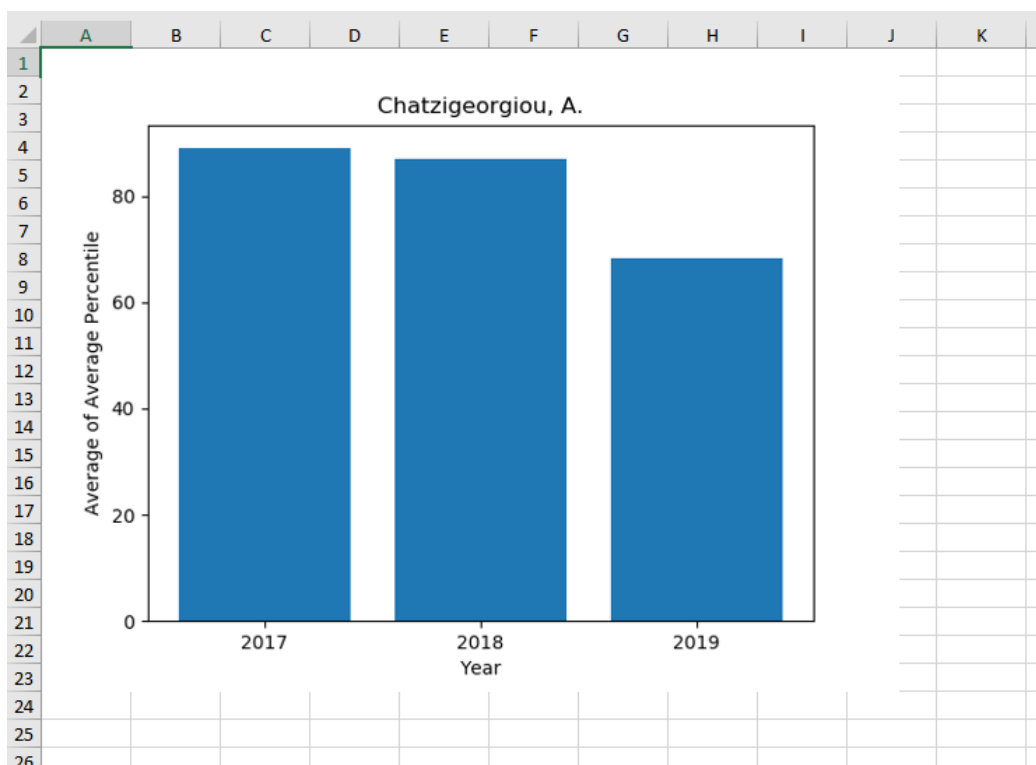
Στο πέμπτο φύλλο «Avg Percentile Mean Per Year», βρίσκεται ένα line chart, το οποίο δείχνει το μέσο Average Percentile ανά χρόνο.



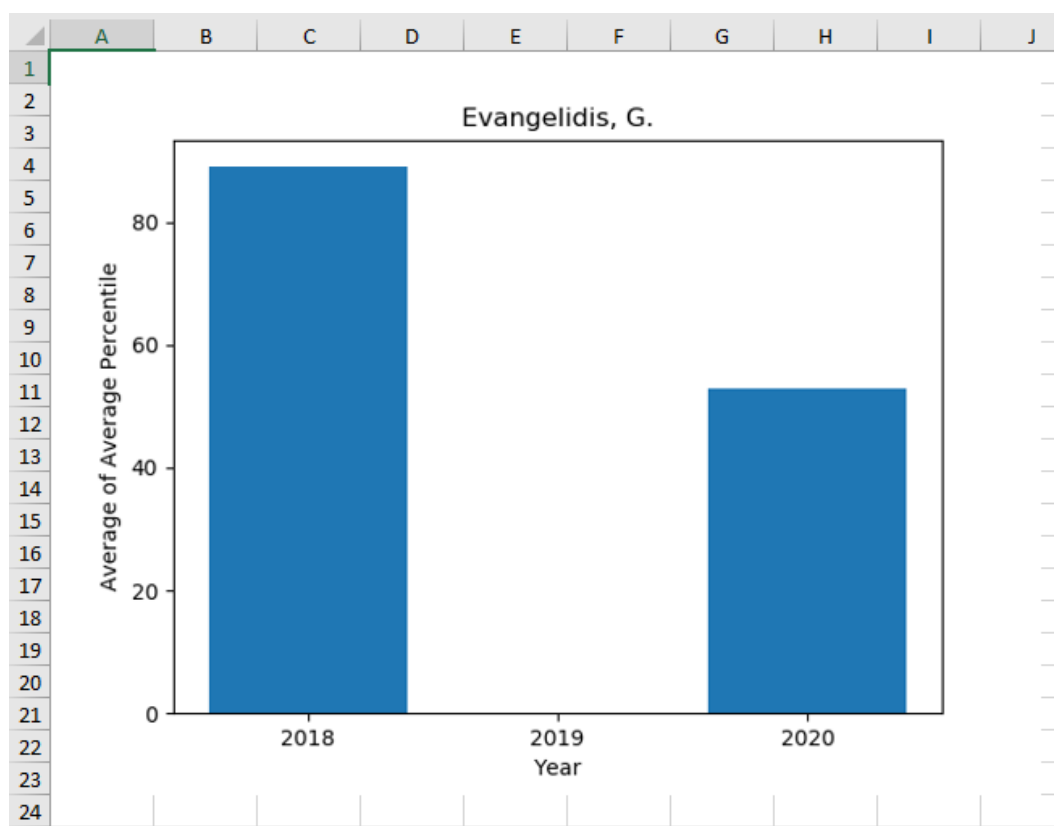
Εικόνα 5-19: Μέγιστη τιμή του Δείκτη CiteScore Ανά Χρονιά

Στο έκτο φύλλο «Max CiteScore Per Year», βρίσκεται ένα bar chart, το οποίο απεικονίζει την μέγιστη τιμή του δείκτη CiteScore ανά χρόνο.

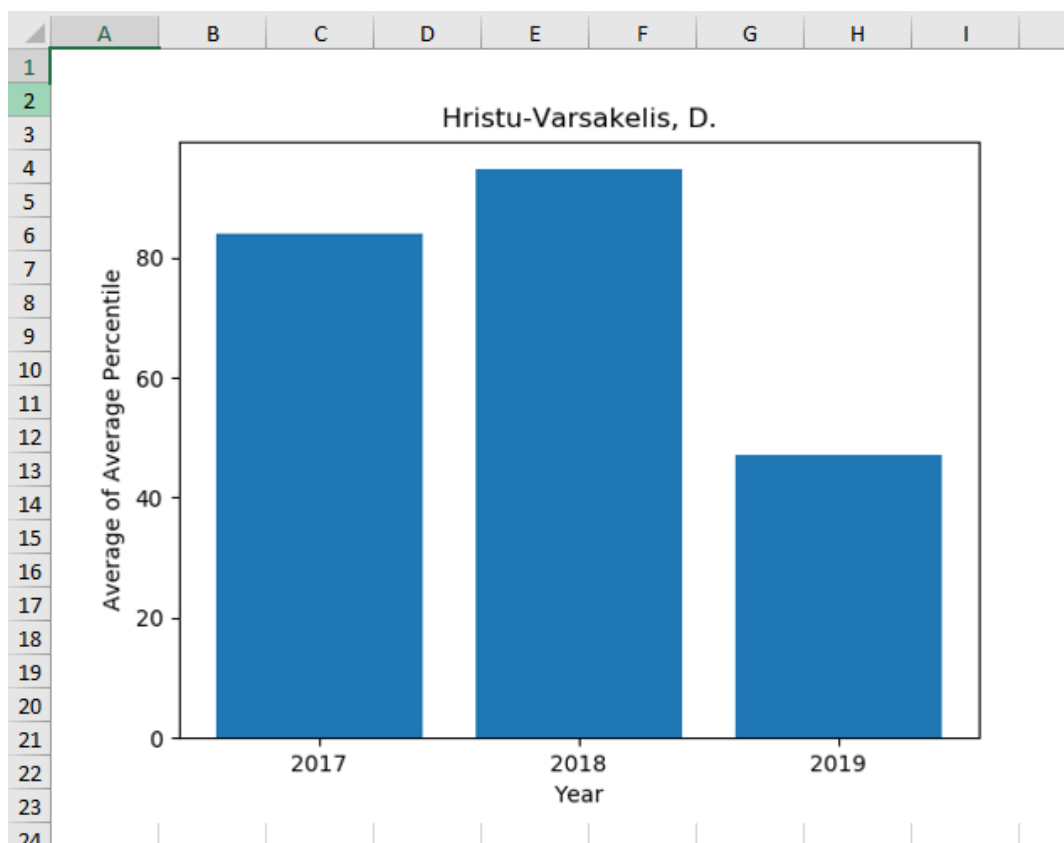
Θα δημιουργηθεί ένα φύλλο για κάθε καθηγητή που επιλέξαμε από το παράθυρο «Professor Statistics», το οποίο θα περιέχει ένα bar chart, το οποίο θα απεικονίζει τον μέσο όρο του Average Percentile των περιοδικών των εγγράφων που έχει συμμετάσχει ως συγγραφέας ανά χρόνο για τον συγκεκριμένο καθηγητή. Το κάθε νέο φύλλο, θα έχει το επώνυμο του καθηγητή και το αρχικό γράμμα του ονόματός του.



Εικόνα 5-20: Μέσος Όρος Average Percentile Καθηγητή Ανά Χρονιά 1



Εικόνα 5-21: Μέσος Όρος Average Percentile Καθηγητή Ανά Χρονιά 2



Εικόνα 5-22: Μέσος Όρος Average Percentile Καθηγητή Ανά Χρονιά 3

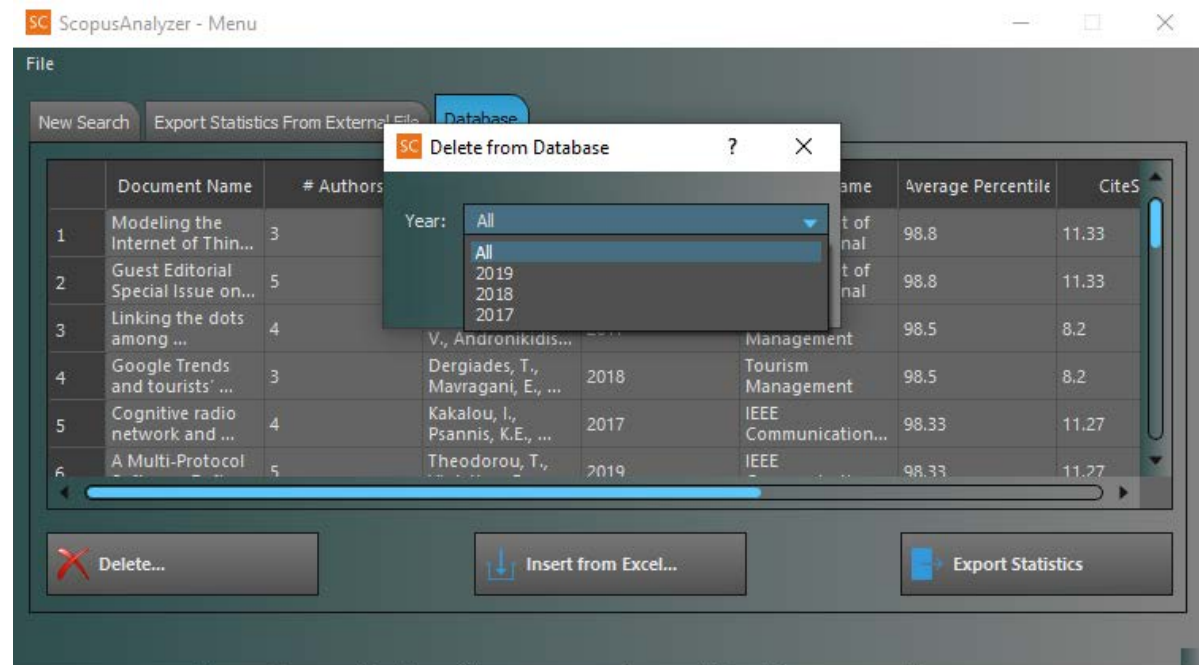
	A	B	C	D	E
1		Name	Department	Ranking	Years
2	0	Alexander Chatzigeorgidis	Applied Informatics	78,465	2017 - 2020
3	1	Georgios Evangelidis	Applied Informatics	71	2017 - 2020
4	2	Christos Georgiadis	Applied Informatics	68,64	2017 - 2020
5	3	Dimitrios Hristu-Varsakelis	Applied Informatics	75,223	2017 - 2020
6	4	Konstantinos Maroulis	Applied Informatics	79,179	2017 - 2020
7	5	Ioannis Mavridis	Applied Informatics	69	2017 - 2020

Εικόνα 5-23: Στατιστικά Καθηγητών Ανά Τμήμα

Τέλος, υπάρχει και το φύλλο «Department Statistics», το οποίο περιέχει τα rankings των καθηγητών ανά τμήμα (μέσο όρο των Average Percentiles της χρονιάς που έγινε η αναζήτηση). Είναι το ίδιο που εξάγεται και στο αρχείο της παραμετροποιημένης αναζήτησης.

5.3 Βάση Δεδομένων

5.3.1 Διαγραφή Στοιχείων

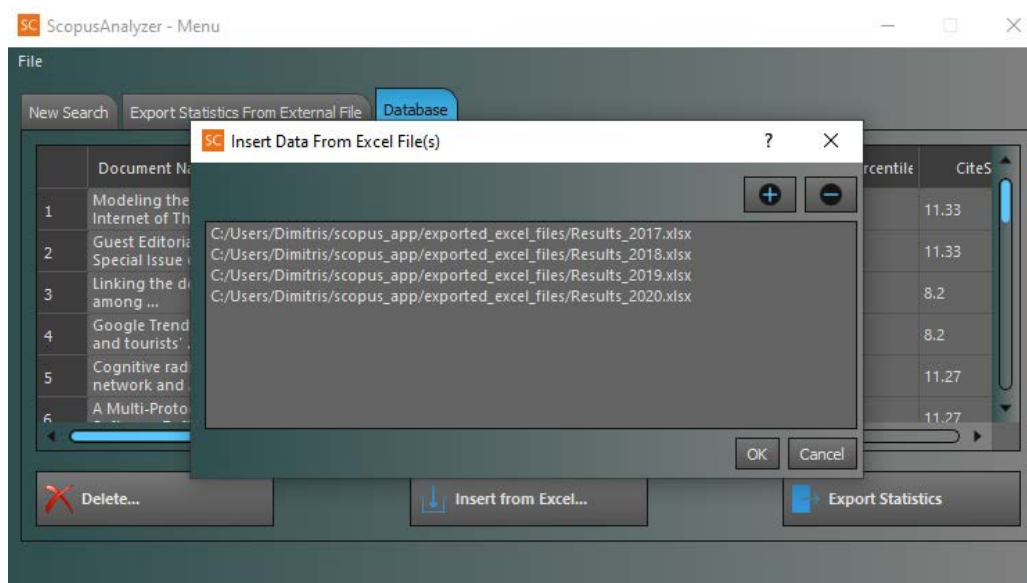


Εικόνα 5-24: Διαγραφή Στοιχείων από την Βάση Δεδομένων

Ο χρήστης έχει τη δυνατότητα μέσω της εφαρμογής, και συγκεκριμένα μέσω της καρτέλας «Database» να διαγράψει στοιχεία της τοπικής βάσης δεδομένων. Αυτό γίνεται, εάν επιλέξει το κουμπί «Delete», όπου θα του εμφανιστεί ένα νέο παράθυρο. Από το νέο παράθυρο αυτό, ο χρήστης μπορεί να επιλέξει την χρονιά, της οποίας τα στοιχεία επιθυμεί να διαγράψει. Πατώντας το ok, τα στοιχεία διαγράφονται από την βάση δεδομένων.

Σημείωση: Η διαγραφή στοιχείων απευθείας από την βάση, είναι δυνατή μόνο ανά χρονιά, ώστε να διατηρηθεί η απαραίτητη συνέπεια των στατιστικών που ενδεχομένως να εξαχθούν με βάση τα στοιχεία της βάσης μας.

5.3.2 Εισαγωγή στοιχείων από εξωτερικά αρχεία

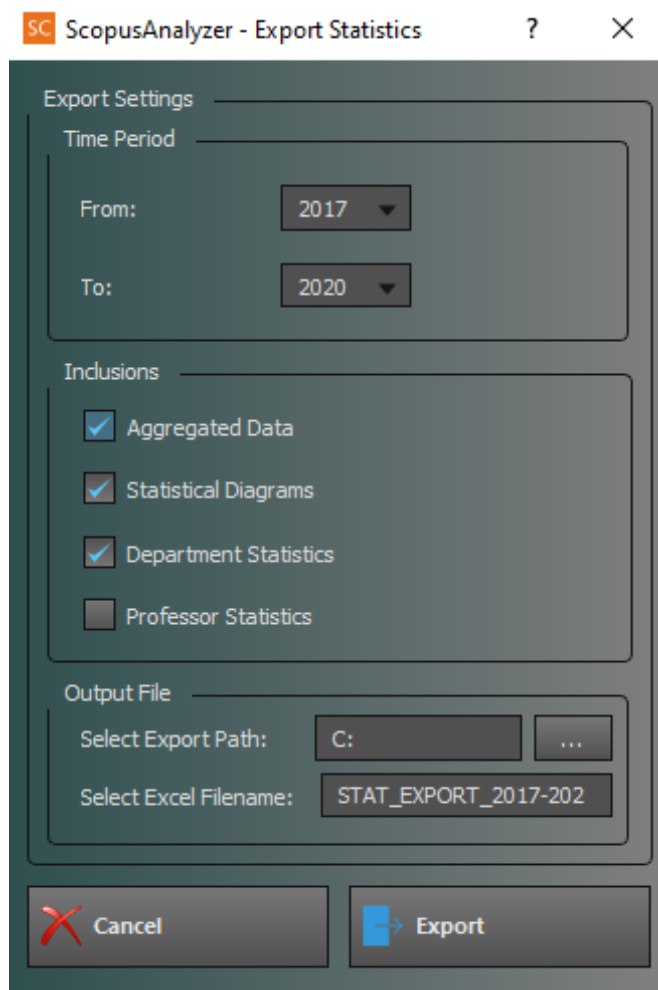


Εικόνα 5-25: Εισαγωγή Στοιχείων στην Βάση Δεδομένων από Εξωτερικά Αρχεία

Ο χρήστης, έχει την δυνατότητα μέσω της εφαρμογής, να μπορέσει να εισάγει στην τοπική βάση δεδομένων (για μόνιμη αποθήκευση) στοιχεία, τα οποία υφίστανται σε εξωτερικά αρχεία Excel, τα οποία έχουν δημιουργηθεί από την ίδια την εφαρμογή (μέσω της διαδικασίας της παραμετροποιημένης αναζήτησης). Για να το κάνει αυτό, ο χρήστης θα πρέπει να πατήσει το κουμπί «Insert from Excel», όπου θα του εμφανιστεί ένα νέο παράθυρο, στο οποίο μπορεί να προσθέσει νέα στοιχεία, είτε από το κουμπί προσθήκης, είτε σέρνοντάς τα στη λίστα (drag and drop). Μόλις ο χρήστης πατήσει το κουμπί «OK», τα στοιχεία εισάγονται στην βάση.

5.3.3 Εξαγωγή Στατιστικών από Στοιχεία της Βάσης Δεδομένων

Ο χρήστης έχει την δυνατότητα να εξάγει περιγραφικά στατιστικά από την βάση δεδομένων, πατώντας το κουμπί «Export Statistics». Μόλις πατήσει το συγκεκριμένο κουμπί, του εμφανίζεται ένα νέο παράθυρο, το παράθυρο επιλογής ρυθμίσεων εξαγωγής. Όπως βλέπουμε, το παράθυρο είναι ακριβώς το ίδιο με το παράθυρο ρυθμίσεων εξαγωγής που είδαμε παραπάνω στην [Εξαγωγή Στατιστικών από Εξωτερικά Αρχεία](#).



Εικόνα 5-26: Ρυθμίσεις Εξαγωγής Στατιστικών

Οι επιλογές του συγκεκριμένου παραθύρου έχουν αναλυθεί σε παραπάνω [κεφάλαιο](#), όπως επίσης έχουν παρουσιαστεί και τα διαγράμματα τα οποία μπορεί να παράξει.

6 Επίλογος

Στο παρόν κεφάλαιο, πραγματοποιείται η εξαγωγή συμπερασμάτων της παρούσας εργασίας, όπως επίσης προτείνονται μελλοντικές επεκτάσεις και αναδομήσεις του παρόντος συστήματος.

6.1 Συμπεράσματα

Αρχικά, έγινε αναφορά στον σκοπό για τον οποίο δημιουργήσαμε το συγκεκριμένο σύστημα και την ανάγκη δημιουργίας του συστήματος, με το οποίο δίνεται η δυνατότητα σε έναν υποψήφιο χρήστη να εκτελέσει μία αυτοματοποιημένη και παραμετροποιημένη αναζήτηση στην βιβλιογραφική βάση Scopus, ενώ παράλληλα δίνεται και η δυνατότητα εξαγωγής χρήσιμων και εύλωπτων στατιστικών στοιχείων.

Στη συνέχεια, αναλύσαμε το θεωρητικό υπόβαθρο του θέματος της εργασίας, το οποίο είναι η βιβλιομετρία, αναφέροντας σημαντικούς βιβλιομετρικούς δείκτες, αναλύοντάς τους έναν προς ένα.

Έπειτα, έγινε αναφορά στις τεχνολογίες, οι οποίες χρησιμοποιήθηκαν για την υλοποίηση του συστήματος, καθώς και ορισμένες πληροφορίες και λεπτομέρειες για την κάθε τεχνολογία.

Κατόπιν, έγινε εκτενής αναφορά στην αρχιτεκτονική του συστήματος (διακομιστή και εφαρμογής πελάτη), καθώς και στον τρόπο κατασκευής των δομικών μερών της.

Τέλος, έγινε επίδειξη της λειτουργικότητας της εφαρμογής ScopusAnalyzer, όπου παρουσιάστηκαν όλες οι δυνατές λειτουργίες που μπορεί να επιτελέσει ένας χρήστης με αυτή.

Συνοψίζοντας, μπορούμε να ισχυριστούμε ότι η εφαρμογή καλύπτει επαρκώς την παραμετροποιημένη αναζήτηση στην βιβλιογραφική βάση του Scopus, αφού ο χρήστης έχει τόσο τη δυνατότητα να εκτελέσει μία αναζήτηση και να παράξει τα αποτελέσματα αυτής σε κάποιο αρχείο, όσο και να παράξει σημαντικά στατιστικά, τα οποία μπορεί να οδηγήσουν σε συμπεράσματα, συνδυάζοντας στοιχεία πολλών χρόνων.

6.2 Μελλοντικές Επεκτάσεις

Η εφαρμογή που σχεδιάστηκε και υλοποιήθηκε είναι μία πλήρως λειτουργική εφαρμογή, η οποία έχει ως στόχο να διευκολύνει την διαδικασία επιβράβευσης ερευνητών του Πανεπιστημίου Μακεδονίας, και συγκεκριμένα για το τμήμα Εφαρμοσμένης Πληροφορικής.

Οι επεκτάσεις θα μπορούσαν να γίνουν σε 3 διαστάσεις:

- **Υποστήριξη Άλλων Ιδρυμάτων:** Μία επέκταση θα μπορούσε να ήταν η υποστήριξη και άλλων ακαδημαϊκών ιδρυμάτων και τμημάτων από όλη την Ελλάδα.
- **Προσθήκη Νέων Λειτουργιών:** Οι νέες λειτουργίες θα μπορούσαν να αφορούν σε νέα στατιστικά διαγράμματα, καθώς όπως είναι λογικό δεν καλύπτει όλο το εύρος των περιγραφικών στατιστικών διαγραμμάτων.
- **Γραφική Διεπαφή Χρήστη:** Πεδίο μελλοντικής επέκτασης, μπορεί να αποτελέσει και η αναβάθμιση της γραφικής διεπαφής του χρήστη, κάνοντάς την πιο φιλική και με περισσότερες συντομεύσεις.

Βιβλιογραφία

- The Qt Company (2018). Qt Designer Manual. Διαθέσιμο: <http://doc.qt.io/qt-5/qtdesigner-manual.html>
[Προσπελάστηκε 08/05/2020]
- Riverbank Computing (2017). Using Qt Designer. Διαθέσιμο: <http://pyqt.sourceforge.net/Docs/PyQt5/designer.html>
[Προσπελάστηκε 08/05/2020]
- Αγγελιδάκης, Ν. Α. (2015) Εισαγωγή στον προγραμματισμό με την Python. Διαθέσιμο: http://aggelid.mysch.gr/pythonbook/INTRODUCTION_TO_COMPUTER_PROGRAMMING_WITH_PYTHON.pdf
[Προσπελάστηκε 08/05/2020]
- Κιορπέ, Π. (2013) MPSMaker: Ανάπτυξη διεπαφής για την επεξεργασία μετραπογραμμάτων και σύνδεση τους με γνωστούς λύτες. Διαθέσιμο: <https://dspace.lib.uom.gr/bitstream/2159/16154/6/KiorpesPeriklisMsc2013.pdf>
[Προσπελάστηκε 08/05/2020]
- Μπαγκέρη, Ε. Σχεσιακές Βάσεις Δεδομένων: Η γλώσσα SQL και Αποθηκευμένες Διαδικασίες. Διαθέσιμο: <http://nefeli.lib.teicrete.gr/browse/stef/epp/2010/MpagkeriEvangelia/attached-document-1271831186-641028-6701/Mpaggeri2010.pdf>
[Προσπελάστηκε 08/05/2020]
- Ξένος, Μ., και Χριστοδουλάκης, Δ. (2000) Βάσεις Δεδομένων. Πάτρα: ΤΥΠΟΡΑΜΑ. Διαθέσιμο: <http://www.jimkava.com/wp-content/uploads/2017/08/xristodoulakis.pdf>
[Προσπελάστηκε 08/05/2020]
- Distinctive Features of SQLite, Διαθέσιμο: <https://www.sqlite.org/different.html>
[Προσπελάστηκε 08/05/2020]

- Simon Stewart, Selenium WebDriver. Διαθέσιμο: <http://www.aosabook.org/en/selenium.html>
[Προσπελάστηκε 08/05/2020]
- Malhotra, P. (n.d.). QTP vs Selenium. Διαθέσιμο: <http://www.aspiresys.com/WhitePapers/QTPvsSelenium.pdf>
[Προσπελάστηκε 08/05/2020]
- Bensman, S. J. (2007). GARFIELD AND THE IMPACT FACTOR: THE CREATION, UTILIZATION, AND VALIDATION OF A CITATION MEASURE Part 2, The Probabilistic, Statistical, and Sociological Bases of the Measure By. Διαθέσιμο: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.160.6965>
[Προσπελάστηκε 08/05/2020]
- Παπαβλασόπουλος Σώζων (2015), Βιβλιομετρία
- Σαχίνη, Ε., Μάλλιου Ν., Χούσος Ν. (2014). Βιβλιομετρική Ανάλυση: Μεθοδολογική Προσέγγιση ΕΚΤ, Εθνικό Κέντρο Τεκμηρίωσης 2η Έκδοση.
- Scimago Research Group. (2007). Description of SCImago Journal Rank indicator. Διαθέσιμο στο: <http://www.scimagojr.com/SCImagoJournalRank.pdf>.
[Προσπελάστηκε 08/05/2020]
- Iglesias, J., & Pecharromán, C. (2007). Scaling the h-index for different scientific ISI fields. Scientometrics
- Educba, Selenium vs QTP. Διαθέσιμο: <https://www.educba.com/selenium-vs-qtp/>
[Προσπελάστηκε 08/05/2020]
- Selenium History. Διαθέσιμο: <https://selenium.dev/history/>
[Προσπελάστηκε 08/05/2020]

- About pandas. Διαθέσιμο: <https://pandas.pydata.org/about/>
[Προσπελάστηκε 08/05/2020]
- Matplotlib History. Διαθέσιμο: <https://matplotlib.org/users/history.html>
[Προσπελάστηκε 08/05/2020]
- Getting Started: What is Socket.IO? Διαθέσιμο: <https://python-socketio.readthedocs.io/en/latest/intro.html>
[Προσπελάστηκε 08/05/2020]