Ατομική Διπλωματική Εργασία

ΥΛΟΠΟΙΗΣΗ WEB 2.0 ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΓΙΑ ΔΙΑΧΕΙΡΙΣΗ ΝΟΗΤΩΝ ΒΙΒΛΙΟΘΗΚΩΝ

Στέφανος Χριστοδουλίδης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΥΛΟΠΟΙΗΣΗ WEB 2.0 ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΓΙΑ ΔΙΑΧΕΙΡΙΣΗ ΝΟΗΤΩΝ ΒΙΒΛΙΟΘΗΚΩΝ

Στέφανος Χριστοδουλίδης

Επιβλέπων Καθηγητής

Δρ. Δημήτρης Ζεϊναλιπούρ

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω θερμά τον Δρ. Δημήτρη Ζεϊναλιπούρ, επιβλέπον καθηγητής της παρούσας διπλωματικής εργασίας, για την άψογη συνεργασία, την καθοδήγηση και τις γνώσεις που μου παρείχε καθ'όλη την διάρκεια της χρονιάς. Πρόκειται για ενάρετο άνθρωπο και αξιόλογο καθηγητή, με εκπληκτικές γνώσεις σε θέματα πληροφορικής, πάντα πρόθυμος να βοηθήσει και να κατανοήσει τυχόν προβλήματα.

Επίσης θα ήθελα να ευχαριστήσω τους φίλους και συναδέλφους, Κωνσταντάντίνο Κώστα, Γιώργο Λάρκου και Γιώργο Νικολαΐδη για την εξαιρετική συνεργασία και την βοήθεια που μου παρείχαν κατά τακτά χρονικά διαστήματα. Οι συμβουλές τους ήταν καθοδηγητικές στις διάφορες δυσκολίες που παρουσιάστηκαν καθ'όλη την διάρκεια μελέτης, ανάπτυξης και υλοποίησης της διπλωματικής μου εργασίας.

Θα ήθελα ακόμη να ευχαριστήσω το Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου για το ολοκληρωμένο πρόγραμμα εκμάθησης των διάφορων τεχνολογιών Πληροφορικής και τους καθηγητές του τμήματος για το σύνολο των γνώσεων που μου παρείχαν κατά την τετραετή φοίτηση στο τμήμα.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου, που με στηρίζει οικονομικά και ψυχολογικά καθ’ όλη την διάρκεια των σπουδών μου.

Περίληψη

Στην παρούσα διπλωματική εργασία θα παρουσιαστεί η σχεδίαση, ανάπτυξη και υλοποίηση του συστήματος SmartLib+. Αρχικά θα παρουσιάσουμε το τι μας παρακίνησε να προχωρήσουμε στην μελέτη του συστήματος, ενώ παράλληλα θα δούμε άρθρα με σχετικές εργασίες και συστήματα.

Επιπρόσθετα, θα παρουσιάσουμε την αρχιτεκτονική του συστήματος, η οποία αποτελείται από 3 διαφορετικά μέρη. Το κομμάτι της βάσης δεδομένων, το κομμάτι του διακομιστή που αναλαμβάνει την διαχείριση των δεδομένων που υπάρχουν, καθώς επίσης και αυτών που εισάγονται στο σύστημα, και το κομμάτι της διεπαφής μέσω της οποίας ο χρήστης μπορεί να διαχειριστεί τις βιβλιοθήκες του.

Παράλληλα, θα δούμε την βάση δεδομένων, όπου θα υπάρξει αναφορά στον τρόπο σχεδιασμού της, και επεξήγηση του κάθε πίνακα για το είδος των δεδομένων που αποθηκεύει και τον τύπο δεδομένων που χρησιμοποιήθηκε σε κάθε πεδίο του πίνακα.

Ταυτόχρονα θα αναφερθούμε στην υλοποίηση του SmartLib+, αρχικά με αναφορά στις διάφορες τεχνολογίες Διαδικτύου που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος. Επίσης θα παρουσιάσουμε εικόνες από την εφαρμογή, και θα υπάρξει εκτενής περιγραφή και επεξήγηση του τρόπου λειτουργίας του.

Τέλος θα παρουσιάσουμε την πειραματική διαδικασία, όπου θα αναφερθούμε στην διαδικασία που ακολουθήσαμε για τον πειραματικό έλεγχο και αποτίμηση του συστήματος. Υπάρχουν γραφικές παραστάσεις που απεικονίζουν τους αριθμούς που λάβαμε, κατά τι φάση αυτή, έτσι ώστε να γίνουν πιο κατανοητά τα αποτελέσματα. Συμπεράσματα και μελλοντικές επεκτάσεις θα παρουσιαστούν στο τελευταίο κεφάλαιο της παρούσας ατομικής διπλωματικής εργασίας.

Περιεχόμενα

Κεφάλαιο 1 **Εισαγωγή** …….. **5**

1.1 Υποκίνηση Εργασίας ..…… **8**

1.2 Περιγραφή Εργασίας ……... **9**

Κεφάλαιο 2 **Σχετική δουλειά** ……... **10**

2.1 Σχετικά Άρθρα και Μελέτη ……. **11**

2.2 Το μοντέλο του πλαισίου SmartLib ……. **16**

Κεφάλαιο 3 **Η αρχιτεκτονική του SmartLib** …….. **16**

3.1 Προδιαγραφές Συστήματος SmartLib ……. **18**

3.2 Επισκόπηση Αρχιτεκτονικής SmartLib …….. **19**

Κεφάλαιο 4 **Βάση Δεδομένων** ……..…… **25**

4.1 Σχεδίαση Βάσης Δεδομένων ……..……. **26**

Κεφάλαιο 5 **Υλοποίηση SmartLib** ……..……. **32**

5.1 Τεχνολογικό Υπόβαθρο ……..……. **32**

5.2 Υλοποίηση Διαδικτυακής Εφαρμογής ……. **38**

Κεφάλαιο 6 **Πειραματική Διαδικασία** ……..…….. **45**

6.1 Πειραματική Υποδομή ……..……... **46**

6.2 Πειραματική Διαδικασία και Μετρήσεις ……. **46**

Κεφάλαιο 7 **Συμπεράσματα και Μελλοντικές Επεκτάσεις** ……... **48**

Βιβλιογραφία ……..……..……... **55**

Παράρτημα Α ……..……..…….. **Α-2-1**

Κεφάλαιο 1

Εισαγωγή

1.1 Υποκίνηση Εργασίας 8
1.2 Περιγραφή Εργασίας 9

Στις μέρες μας, το διαδίκτυο είναι ένα αναπτυσσόμενο σύμπαν αλληλένδετων ιστοσελίδων και εφαρμογών ιστού, γεμάτες με βίντεο, φωτογραφίες και διαδραστικό περιεχόμενο. Αυτό που ο μέσος χρήστης δεν βλέπει είναι η αλληλεπίδραση των τεχνολογιών του διαδικτύου και των πληγών. Με την πάροδο του χρόνου οι τεχνολογίες Διαδικτύου έχουν εξελιχθεί για να δώσουν την δυνατότητα στους προγραμματιστές Ιστού, να δημιουργήσουν νέες γενιές των εμπειριών διαδικτύου. Η δυνατότητα που παρέχει το διαδίκτυο για ταυτόχρονη πρόσβαση μεγάλου αριθμού χρηστών, οδήγησε στην ανάπτυξη μεθόδων όπως το crowdsourcing.

Πληθοπορισμός (Crowdsourcing) είναι η διαδικασία της απόκτησης των αναγκαίων υπηρεσιών, ιδεών ή/και περιεχόμενο από την προσέλκυση μιας μεγάλης ομάδας ανθρώπων και ιδιαίτερα από μια διαδικτυακή κοινότητα. Αν και ο ορισμός αυτός είναι αποδεκτός από ένα μέρος ανθρώπων, ένας πιο συγκεκριμένος ορισμός είναι έντονα συζητήσιμος. Η διαδικασία του crowdsourcing, χρησιμοποιείται συχνά για να διασπάσει την κουραστική εργασία συλλογής δεδομένων, με την εμπλοκή μεγάλου αριθμού ατόμων. Το ενδιαφέρον μέρος της διαδικασίας αυτής είναι το γεγονός ότι η ομάδα των ατόμων που συμβάλλουν στην απόκτηση του επιθυμητού περιεχομένου είναι δημοσίως απροσδιόριστη. Μέσω της απροσδιοριστίας αυτής επιτυγχάνεται η απρόσωπη και πιο υποκειμενική συλλογή δεδομένων.

Στόχος της παρούσας διπλωματικής εργασίας, είναι η υλοποίηση και αποτίμηση υποσυστημάτων διαχείρισης νοητών βιβλιοθηκών. Η υλοποίηση των υποσυστημάτων αυτών έχουν στόχο την παροχή στους χρήστες την ευκολία δημιουργίας νοητών βιβλιοθηκών και τοποθέτηση τους στο χάρτη στην φυσική τους τοποθεσία, και καταγραφή των προσωπικών τους βιβλίων στις βιβλιοθήκες αυτές. Η νοητή ηλεκτρονική βιβλιοθήκη παρέχει τη δυνατότητα στους χρήστες να αποστέλλουν αιτήματα συνεργασίας . Με την αποστολή ενός

τέτοιου αιτήματος, και την αποδοχή του από τον παραλήπτη, δημιουργείται ένας σύνδεσμος μεταξύ των δύο χρηστών.

Επιπρόσθετα, παρέχεται η δυνατότητα αποστολής μηνυμάτων μεταξύ των χρηστών, που μπορεί να είναι και ο πιο άμεσος τρόπος επικοινωνίας. Μέσω των λειτουργιών αυτών, κάποιος χρήστης μπορεί να εντοπίσει στο χάρτη τις βιβλιοθήκες των συνεργατών του, και σε περίπτωση που επιθυμεί να έχει πρόσβαση σε κάποια πληροφορία, η οποία ανήκει σε ένα φυσικό αντίτυπο ενός βιβλίου, το οποίο ανήκει σε κάποιο από τους συνδέσμους του, και να ζητήσει, σε περίπτωση που το αντίτυπο είναι διαθέσιμο προς δανεισμό, να το δανειστεί και να το διαβάσει και να το επιστρέψει με την ολοκλήρωση της μελέτης του. Η νοητή αυτή βιβλιοθήκη αποτελείται από μια διαδικτυακή εφαρμογή η οποία παρέχει τις προαναφερθείσες λειτουργίες.

Σε πρώτο στάδιο οι χρήστες μπορούν να συνδεθούν στη διαδικτυακή εφαρμογή μέσω του λογαριασμού του στο Google. Μέσω του πίνακα εργαλείων ο χρήστης μπορεί να εισάγει μια νέα βιβλιοθήκη, η οποία τοποθετείται στο χάρτη στο σημείο όπου βρίσκεται ο χρήστης. Υπάρχει η δυνατότητα ο χρήστης να μετακινήσει την βιβλιοθήκη του στο χάρτη σε περίπτωση που ο ίδιος αλλάξει τη φυσική τοποθέτηση της συγκεκριμένης βιβλιοθήκης. Σε περίπτωση, που ο χρήστης απολέσει την κατοχή της φυσικής βιβλιοθήκης και επιθυμεί να αφαιρέσει την ύπαρξη της από το σύστημα, για να αποφευχθούν οποιεσδήποτε παρεξηγήσεις και παραπλανήσεις άλλων χρηστών, παρέχεται η δυνατότητα από το σύστημα, να αφαιρεθεί η συγκεκριμένη βιβλιοθήκη.

Μόνο με τις λειτουργίες αυτές, δεν θα ήταν δυνατό να προσελκύσουμε ομάδες ατόμων να γίνουν χρήστες του συστήματος αυτού. Έτσι υλοποιήθηκε το υποσύστημα συνεργατών, όπου κάποιος χρήστης μπορεί να αποστείλει αίτημα συνεργασίας μέσω ηλεκτρονικού ταχυδρομείου. Η αποστολή του ηλεκτρονικού μηνύματος για το αίτημα συνεργασίας αναλαμβάνεται εξολοκλήρου από το σύστημα. Το μόνο που χρειάζεται από το χρήστη είναι να παρέχει την διεύθυνση του ηλεκτρονικού ταχυδρομείου του ατόμου με το οποίο επιθυμεί να συνεργαστεί.

Για την εισαγωγή ενός νέου βιβλίου υπάρχει η δυνατότητα μέσω επιλογής του δείκτη της βιβλιοθήκης, η καταχώρηση του μοναδικού αριθμού ISBN που υπάρχει στο οπισθόφυλλο του κάθε βιβλίου. Με την επιλογή του δείκτη βιβλιοθήκης, παρουσιάζεται στο χρήστη μία λίστα με τα βιβλία της επιλεγμένης βιβλιοθήκης, καθώς επίσης και μια φόρμα εισόδου για τον κωδικό ISBN. Με την εισαγωγή του συγκεκριμένου κωδικού, γίνεται κλήση στο διεπαφή

προγράμματος εφαρμογής για βιβλία της Google η οποία βοηθά στην ανάκτηση πληροφοριών σχετικά με το βιβλίο του οποίου εισάχθηκε ο κωδικός ISBN. Με την κλήση στην διεπαφή επιστρέφεται ένας μεγάλος αριθμός από πληροφορίες οι οποίες μπορούν να χρησιμοποιηθούν κατά την αποθήκευση του βιβλίου.

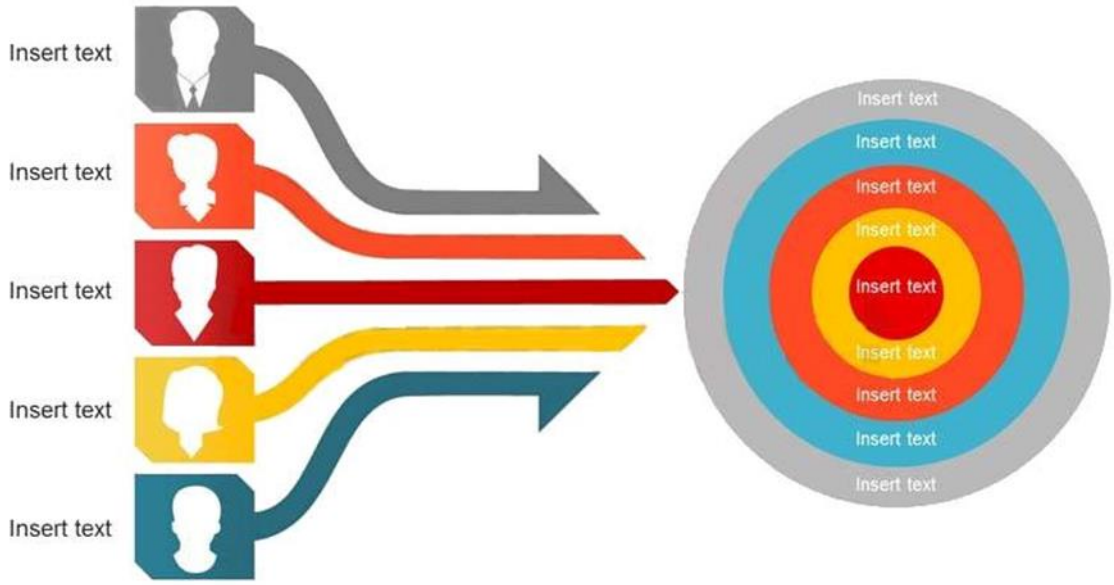
Τέλος, παρέχεται η δυνατότητα στους χρήστες, να αναζητήσουν κάποιο βιβλίο που μπορεί να υπάρχει σε κάποια από τις βιβλιοθήκες, είτε δική του είτε κάποιου συνεργάτη του. Με την διεκπεραίωση της αναζήτησης, οι δείκτες τον βιβλιοθηκών, οι οποίες δεν περιέχουν το αντικείμενο της συγκεκριμένης αναζήτησης, εμφανίζονται με μαυρόασπρο εικονίδιο ενώ οι βιβλιοθήκες που έχουν στο περιεχόμενο τους το αντικείμενο αναζήτησης εμφανίζονται με χρωματιστό εικονίδιο.

Στη παρούσα διπλωματική, οποιαδήποτε αναφορά σε ψηφιακή ή νοητή βιβλιοθήκη, εννοεί την ηλεκτρονική καταγραφή των πληροφοριών έντυπης μορφής βιβλίων, σε κάποιο σύστημα. Οι πληροφορίες αυτές μπορεί να περιλαμβάνουν τον τίτλο, περιγραφή, ημερομηνία έκδοσης, συγγραφής, ή/και ηλεκτρονική μορφή του βιβλίου.

Εν κατακλείδι, η πιο πάνω εφαρμογή, βρίσκεται σε λειτουργία στην ηλεκτρονική διεύθυνση https://smartlib.cs.ucy.ac.cy/ και είναι διαθέσιμη προς όλους τους ενδιαφερόμενους.

1.1 Υποκίνηση Εργασίας

Στης μέρες μας η ανάπτυξη του ιστού έχει λάβει τεράστιες διαστάσεις. Θα μπορούσε κανείς να πει ότι το διαδίκτυο αποτελεί το νευραλγικό κέντρο για την παγκόσμια οικονομία. Αρκετές μεγάλες εταιρίες και οργανισμοί έχουν καταλάβει ότι μόνο οι οργανισμοί που θα αντιληφτούν και θα κατανοήσουν τη σημασία του διαδικτύου θα ευδοκιμήσουν οικονομικά. Μια καλά λοιπόν σχεδιασμένη εφαρμογή αποτελεί ταυτότητα στο διαδίκτυο και αυτό είναι το ουσιαστικό μέρος της επιτυχίας.
Είναι ευρέως αποδεχτό το γεγονός ότι κάποιος για να σε εμπιστευτεί πρέπει να είναι εύκολο να σε βρει. Για παράδειγμα, όταν κάποιος είναι καταγεγραμμένος στο Χρυσό Οδηγό, είναι πιο πιθανόν νέα άτομα να τον εντοπίσουν και κυρίως να τον εμπιστευθούν, λόγο της εχεμύθειας και εμπιστοσύνης που εμπνέει ο Χρυσός Οδηγός. Έτσι λοιπόν και με το διαδίκτυο. Η ύπαρξη ενός διαδικτυακού χώρου υποδεικνύει την παρουσία και εμπνέει εμπιστοσύνη.
Το κυριότερο κομμάτι με τα πλεονεκτήματα του διαδικτύου είναι ότι είναι πολύ εύκολο να γίνεις οικουμενικά γνωστός. Για μια νέα εφαρμογή η οποία κάνει την πρώτη της εμφάνιση,



Σχήμα 1.1 : Διαγραμματική απεικόνιση του τρόπου λειτουργίας του crowdsourcing.

αυτό που χρειάζεται είναι να γίνει όσο το δυνατό πιο σύντομα γνωστή. Και αυτό γιατί, με τι χρήση του crowdsourcing επιτυγχάνονται ανεκτίμητα αποτελέσματα που θα χρειάζονταν χρόνια για να επιτευχθούν.

Επιπρόσθετα, η ευκολία πρόσβασης στη πληροφορία, μέσω των διάφορων μηχανών αναζήτησης, έχουν παρακινήσει τους προγραμματιστές διαδικτυακών εφαρμογών, στην ανάπτυξη μιας πληθώρας εφαρμογών με σκοπό την καταγραφή περαιτέρω πληροφοριών, καθώς επίσης και την διευκόλυνση των χρηστών με την παροχή διάφορων άλλων υπηρεσιών. Για όλους τους πιο πάνω λόγους, και για πολλούς άλλους, έχει παρουσιαστεί την τελευταία δεκαετία, μεγάλη ανάπτυξη στον ιστό και στις τεχνολογίες διαδικτύου. Έτσι λοιπόν στόχος μας είναι η δημιουργία μιας διαδικτυακής εφαρμογής που να παρέχει στους χρήστες την ευκολία διατήρησης και ενημέρωσης της κατάστασης της βιβλιοθήκης ή/και των βιβλιοθηκών τους. Η εφαρμογή μας παρέχει την δυνατότητα καταγραφής των βιβλίων ενός χρήστη και της ομαδικής εργασίας μεταξύ των χρηστών.

1.2 Περιγραφή Εργασίας

Κατά το πρώτο κεφάλαιο παρουσιάστηκε η υποκίνηση της παρούσας εργασίας. Το σημαντικό δηλαδή με το διαδίκτυο καθώς επίσης και το πως επηρεάζει την καθημερινή ζωή των ανθρώπων. Επιπρόσθετα, είχαμε μια μικρή αναφορά στο crowdsourcing και για το πως

βοηθά τις διάφορες διαδικτυακές εφαρμογές. Περισσότερες πληροφορίες για την μέθοδο αυτή θα δούμε στην συνέχεια.

Στο δεύτερο κεφάλαιο θα δούμε άρθρα με παρόμοια δουλεία και εφαρμογές σχετικές με το SmartLib, καθώς επίσης και το μοντέλο πλαισίου που ακολουθήσαμε για την ανάπτυξη της διαδικτυακής μας εφαρμογής, με σκοπό την καταγραφή βιβλίων σε βιβλιοθήκες.

Στο τρίτο κεφάλαιο θα παρουσιάσουμε τις προδιαγραφές του συστήματος καθώς επίσης και την αρχιτεκτονική στην οποία είναι βασισμένη η όλη ανάπτυξη του συστήματος. Θα παρουσιαστούν επίσης οι διάφορες λειτουργίες του συστήματος και θα περιγραφούν τα διάφορα υποσυστήματα και ο τρόπος με τον οποίο λειτουργούν.

Στο τέταρτο κεφάλαιο θα παρουσιάσουμε το κομμάτι της βάσης δεδομένων του συστήματος. Θα περιγραφεί ο τρόπος σχεδίασης της βάσης δεδομένων, και οι διάφοροι πίνακες που απαρτίζουν τη σύστημα της βάσης για την καταγραφή και αποθήκευση των στοιχείων που θα προέρχονται από το σύστημα. Στη συνέχεια θα μελετήσουμε τον τρόπο με τον οποίο έχει υλοποιηθεί το όλο σύστημα της βάσης δεδομένων.

Στο πέμπτο κεφάλαιο θα παρουσιάσουμε το τεχνολογικό υπόβαθρο της εφαρμογής μας και θα περιγράψουμε τον τρόπο υλοποίησης της διαδικτυακής εφαρμογής σε πιο εκτενή μορφή, όπου θα παρουσιαστούν εικόνες από το SmartLib.

Κεφάλαιο 2

Σχετική δουλειά

2.1 Σχετικά Άρθρα και Μελέτη 11
2.2 Το μοντέλο του πλαισίου SmartLib 16

2.1 Σχετικά Άρθρα και Μελέτη

Οι Robert A. Cochran, Loris D'Antoni, Benjamin Livshits, David Molnar, και Margus Veanes στο άρθρο τους "Program Boosting: Program Synthesis via Crowd-Sourcing"[1] παρουσιάζουν μια νέα τεχνική ονομαζόμενη programm boosting η οποία βασίζεται στη λύση του crowdsourcing. Η τεχνική αυτή στοχεύει στη συλλογή της "γνώσης του κοινού" για εξεύρεση μιας αν όχι τέλειας αλλά τουλάχιστο καλής λύσης στις εκ φύσεως δύσκολες προγραμματιστικά εργασίες, οι οποίες διαφεύγουν ακόμα και στους πιο έμπειρους προγραμματιστές.

Η τεχνική αυτή συνδυάζει το crowdsourcing με ένα δύσκολο προς επίλυση πρόβλημα με ένα τρόπο ο οποίος βελτιώνει τη ορθότητα. Στο σύστημα CROWDBOOST υλοποιείται η τεχνική αυτή και στα πειράματα που έχουν γίνει φαίνεται ότι σε αρκετά ενδιαφέρον και μη τετριμμένες εργασίες όπως η συγγραφή κανονικών εκφράσεων σε URL και διευθύνσεις ηλεκτρονικής αλληλογραφίας (email) μπορεί να εφαρμοστεί η τεχνική του crowdsourcing.

Παρουσιάζεται επίσης το γεγονός ότι με την προσεκτική ανάμειξη των αποτελεσμάτων από crowdsourcing προκύπτει κάποια ενίσχυση (Boost) και προκύπτουν αποτελέσματα τα οποία είναι βελτιωμένα σε σύγκριση με άλλα προγράμματα. Τα πειράματα τους σε 465 ζεύγη προγραμμάτων έδειξαν ότι το boosting μπορεί να εφαρμοστεί σε σχετικά μέτριο νομισματικό κόστος.

Οι Annika Hinze και ο David Bainbridge στο άρθρο τους "Tipple: location-triggered mobile access to a digital library for audio books"[2] αναφέρουν την διερεύνηση του ήχου ως μέσο για πρόσβαση σε βιβλία σε μια ψηφιακή βιβλιοθήκη, ενώ ο χρήστης βρίσκεται στη φυσική τοποθεσία στη οποία βρίσκονται τοποθετημένα τα βιβλία.

Τα βιβλία αυτά προέρχονται από ψηφιακή βιβλιοθήκη και συνοδεύονται από προ-ηχογραφημένο ήχο η με την χρήση μεθόδων όπως "κείμενο σε ομιλία" (text-to-speech). Στο άρθρο περιγράφονται λεπτομερώς οι λειτουργικές απαιτήσεις του συστήματος , ο σχεδιασμός και η υλοποίηση. Παρουσιάζονται επίσης οι διάφορες δοκιμές που έχουν γίνει στα πλαίσια της μελέτης.

Οι Dean B. Krafft, Aaron Birkland, και Ellen J. Cramer στο άρθρο τους με τίτλο "Ncore: architecture and implementation of a flexible, collaborative digital library"[4] αναφέρονται σε μια open-source αρχιτεκτονική και την πλατφόρμα λογισμικού για την δημιουργία ευέλικτων και συνεργατικών ψηφιακών βιβλιοθηκών, με το όνομα NCore.

Το NCore αναπτύχθηκε από το National Sciwence Digital Library (NSDL), και λειτουργεί ως το κέντρο υποδομής του. Το NCore αποτελείται από ένα κεντρικό, βασισμένο σε Fendora ψηφιακό αποθετήριο, ένα συγκεκριμένο μοντέλο δεδομένων, ένα API και ένα σύνολο υπηρεσιών, backend και frontend εργαλείων που δημιουργούν ένα νέο μοντέλο για την συνεργατική ψηφιακή βιβλιοθήκη.

Στο άρθρο αυτό , παρουσιάζονται και αναλύονται, η αρχιτεκτονική, τα εργαλεία και οι υπηρεσίες, καθώς επίσης και αναλυτική περιγραφή του NCore. Τέλος παρουσιάζονται αναφορές από τα πειράματα που έγιναν στο National Science Digital Library και αφορούν την κατασκευή και εκμετάλλευση ψηφιακών βιβλιοθηκών στη πλατφόρμα NCore.

Οι Ann Blandford, Suzette Keith, Iain Connell, και Helen Edwards στο άρθρο τους "Analytical usability evaluation for digital libraries: a case study"[3] παρουσιάζουν τα δύο είδη προσέγγισης για την εξέταση της χρησιμότητας της ύπαρξης οποιουδήποτε συστήματος. Τα δύο είδη προσεγγίσεων είναι η εμπειρικές προσεγγίσεις και οι αναλυτικές προσεγγίσεις.

Οι εμπειρικές τεχνικές περιλαμβάνουν έλεγχο των διαφόρων συστημάτων με την εμπλοκή των χρηστών ενώ οι αναλυτικές τεχνικές περιλαμβάνουν την χρηστικότητα του προσωπικού για την αξιολόγηση των διαφόρων συστημάτων εφαρμόζοντας καθιερωμένες θεωρίες και μεθόδους αποτίμησης.

Στο άρθρο αναφέρονται μια σειρά από μελέτες που έγιναν , στις οποίες 4 διαφορετικές τεχνικές εφαρμόστηκαν σε διάφορες ψηφιακές βιβλιοθήκες εστιάζοντας στα δυνατά σημεία, τους περιορισμούς και την έκταση της κάθε προσέγγισης.

Δύο τεχνικές, η αξιολόγηση Heuristic και η Cognitive Walkthrough εφαρμόστηκαν σε περιπτώσεις παραδοσιακών βιβλίων κειμένου για το λόγο ότι δεν υπήρχε προφανές τρόπος να ενταχθούν στο τομέα των ψηφιακών βιβλιοθηκών.

Για την τρίτη τεχνική, χρησιμοποιήθηκε το Claim Analysis, αναπτύχθηκαν μια σειρά από επαναχρησιμοποιήσιμα σενάρια και περσόνες που αφορούν την προσέγγιση της ανάπτυξης ψηφιακών βιβλιοθηκών.

Η τέταρτη τεχνική, CASSM, αναφέρεται ρητά στον τομέα των ψηφιακών βιβλιοθηκών συνδυάζοντας εμπειρικά δεδομένα με μια αναλυτική προσέγγιση. Οι ερευνητές έχουν διαπιστώσει ότι οι δύο μέθοδοι αξιολόγησης, η Heuristic και η Cognitive Walkthrough, έχουν αντιμετωπίσει μόνο επιφανειακές πτυχές και θέματα που έχουν να κάνουν με τον σχεδιασμό της διεπαφής.

Οι τεχνικές αυτές, παρόλο που διαπιστώθηκε πως δεν λειτουργούν και τόσο καλά στην αξιολόγηση της λειτουργικότητας, διαπιστώθηκε, ότι είναι πολύ καλές στην αξιολόγηση της διεπαφής. Παράλληλα, οι τεχνικές Claim Analysis και CASSM, μπορούν να βοηθήσουν στον εντοπισμό βαθύτερων εννοιολογικών δυσκολιών με το μόνο πρόβλημα που αντιμετωπίζουν να είναι η απαίτηση μεγάλης εμπειρικής ικανότητας από τον αναλυτή.

Παρόλα αυτά, καμία από τις τεχνικές αυτές, δεν φαίνεται να εφαρμόζεται τέλια με τις υπάρχουσες τεχνικές ανάπτυξης ψηφιακών βιβλιοθηκών, αναδεικνύοντας έτσι την

σημαντικότητα μιας περιοχής η οποία χρήζει συνέχιση των εργασιών για υποστήριξη της βέλτιστης χρηστικότητας.

Στο άρθρο "Use of multiple digital libraries: a case study"[5], οι Ann Blandford, Hanna Stelmaszewska, και Nick Bryan-Kinns αναφέρουν πως σκοπός της εργασίας τους ήταν η κατανόηση των ζητημάτων ευχρηστίας που προέκυψαν κατά την χρησιμοποίηση ψηφιακών βιβλιοθηκών σε φυσικές τοποθεσίες.

Η μέθοδος που χρησιμοποιήθηκε ήταν η ανάλυση πρωτοκόλλου των χρηστών που εργάζονται σε ένα έργο της επιλογής τους για να ανακτήσουν δεδομένα από δημόσια διαθέσιμες ψηφιακές βιβλιοθήκες. Όπως φαίνεται στο άρθρο, υπάρχουν διάφορα επίπεδα δυσκολίας στην ευχρηστία.

Συγκεκριμένα οι ερευνητές επικεντρώθηκαν στα συμφραζόμενα, δηλαδή τα προβλήματα στη χρηστικότητα που προέκυψαν από το γεγονός ότι οι βιβλιοθήκες είναι προσβάσιμες με συγκεκριμένους τρόπους, και υπάρχουν περιορισμοί όπως τεχνικοί καθώς επίσης και οργανωτικοί. Επίσης, η χρήση συγκεκριμένων πόρων είναι διακριτική.

Οι έννοιες από ένα πλαίσιο αλληλεπίδρασης, το οποίο παρέχει υποστήριξη για τα πρότυπα της αλληλεπίδρασης μεταξύ των χρηστών και των συστημάτων, εφαρμόζονται για την κατανόηση των θεμάτων της αλληλεπίδρασης.

Ταυτόχρονα οι Kim Grohs, Caroline Reed, και Nancy Allen στο άρθρο με τίτλο "Marketing the virtual library"[6] αναφέρουν ότι κατα την διάρκεια της τελευταίας δεκαετίας έχουν υπάρξει σημαντικές αλλαγές στην τριτοβάθμια εκπαίδευση ιδίως με την ανάδειξη τη εκπαίδευσης εξ αποστάσεως.

Αυτό με τη σειρά του είχε μια συνεχή επίδραση στις προσπάθειες στο να εννοιολογηθεί τί είναι μια ακαδημαϊκή βιβλιοθήκη και τι κάνει. Δεν αποτελεί μεγάλη έκπληξη το γεγονός ότι οι ακαδημαϊκές βιβλιοθήκες αντιμετωπίζουν μια σειρά από κρίσιμα ζητήματα, όπως η αύξηση του κόστους των πόρων, η επέκταση των παραδοσιακών υπηρεσιών, την αύξηση του ανταγωνισμού από άλλους παροχής πληροφοριών καθώς και των αντίκτυπο των νέων τεχνολογιών.

Παρά το γεγονός ότι τα ζητήματα αυτά εμφανίζονται ως απειλές, είναι ταυτόχρονα ευκαιρίες για τις βιβλιοθήκες για να σχεδιάσουν το μέλλον τους. Στο εγγύς μέλλον, οι ακαδημαϊκές

βιβλιοθήκες θα παραμείνουν ζωτικός πόρος για τους φοιτητές και το ακαδημαϊκό προσωπικό σχολών και πανεπιστημίων.

Ενώ είναι εύκολο για τις ακαδημαϊκές βιβλιοθήκες να εφησυχαστούν για την κατάσταση τους σε ένα πανεπιστήμιο επειδή δεν υπάρχει ανταγωνισμός, τα επιτυχή προγράμματα μάρκετινγκ μπορεί να ενισχύσουν την προβολή και να
κατασήσουν κατανοητή την αξία της διαμορφώνοντας την κοινή αντίληψη για το πεδίο εφαρμογών των πόρων και των υπηρεσιών που παρέχει.

Οι Patricia Pettijohn και Tina Neville στο άρθρο τους "Collection development for virtual libraries"[7] αναφέρουν ότι η εξέλιξη από το χαρτί στις ηλεκτρονικές πηγές αλλάζει τον τρόπο με τον οποίο οι πληροφορίες κατέχονται, διαμοιράζονται και είναι προσβάσιμες. Για τις βιβλιοθήκες, η εμπορευματοποίηση των ψηφιακών πληροφοριών έχει μακροπρόθεσμες συνέπειες για την απόκτηση και την ανάπτυξη των συλλογών μιας βιβλιοθήκης.

Όπως η αδειοδότηση αντικαθιστά την αγορά, και οι επιχειρηματικές πρακτικές των εταιριών λογισμικού αντικαθιστούν εκείνες των εκδοτών, η πρόσβαση σε πληροφορίες σχετικά με τη ζήτηση, αντικαθιστά τη συλλογή, και οι συνεταιριστικές εξαγορές συμπληρώνουν την τοπική ανάπτυξη συλλογών.

Η αυξανόμενη ζήτηση για το πλήρες κείμενο σε απευθείας σύνδεση με το περιεχόμενο που μπορούν να αναζητηθούν εύκολα και με απομακρυσμένη πρόσβαση, έχει οδηγήσει τις βιβλιοθήκες να εξαρτώνται από μια σειρά από μεσάζοντες και συνεταιρισμούς.

Μέσα σε αυτό το τοπίο των πολλαπλασιαζόμενων πληροφοριών και της μείωσης της αγοραστικής δύναμης, δεν είναι έκπληξη το γεγονός ότι όταν η Ομοσπονδία Ψηφιακών Βιβλιοθηκών ξεκίνησε μια ανεπίσημη έρευνα μεγάλων προκλήσεων που αντιμετωπίζει ερευνητικές βιβλιοθήκες, όπου οι ερωτηθέντες δήλωσαν ότι ο εντοπισμός ψηφιακών βιβλιοθηκών είναι η μεγαλύτερη τους πρόκληση.

2.2 Το μοντέλο του πλαισίου SmartLib

Ας υποθέσουμε πως κάθε σπίτι, και κάθε άτομο ξεχωριστά, κατέχει ένα σύνολο από βιβλία σε έντυπη μορφή. Έστω ότι το σύνολο το βιβλίων αυτών αποτελούν μια ξεχωριστή βιβλιοθήκη, ο κάθε τέτοιο σύνολο.

Υπάρχει μεγάλος αριθμός ατόμων όπου θα ήθελαν με το πάτημα ενός κουμπιού να μπορούν να διαχειριστούν θεωρητικά το σύνολο των βιβλίων τους. Να μπορούν δηλαδή να ανακτήσουν μια λίστα με τα βιβλία τους αποφεύγοντας το δυσάρεστο γεγονός επαναγοράς τους σε μελλοντικό στάδιο.

Με το SmartLib, επιτυγχάνει η δημιουργία νοητών βιβλιοθηκών, όπου ο χρήστης μπορεί να τοποθέτηση την βιβλιοθήκη του σε ψηφιακό χάρτη στο σημείο της πραγματικής του γεωφυσικής τοποθέτησης.

Το μόνο που χρειάζεται από το χρήστη είναι το πάτημα ενός κουμπιού. Με την ενέργεια αυτή του χρήστη, εμφανίζεται αυτόματα στον χάρτη, στο σημείο όπου βρίσκεται ο χρήστης μια νέα βιβλιοθήκη. Με την εμφάνιση της βιβλιοθήκης, δημιουργείται και μια νέα εγγραφή στην βάση δεδομένων του συστήματος και ανατίθεται στον χρήστη.

Με την εμφάνιση της βιβλιοθήκης, ενεργοποιείται αυτόματα και η δυνατότητα, της εισαγωγής ενός βιβλίου στην βιβλιοθήκη. Το μόνο που χρειάζεται είναι να επιλέξει την βιβλιοθήκη ο χρήστη, και να πατήσει το κουμπί προσθήκης ενός βιβλίου. Στη συνέχεια το μόνο που απομένει είναι η καταχώρηση του ISBN μοναδικού αναγνωριστικού αριθμού.

Με την εισαγωγή του αριθμού ISBN, γίνεται κλήση στο Google Books API, όπου και επιστρέφονται οι πληροφορίες που αφορούν το βιβλίο, όπως ο τίτλος, ο ISBN αριθμός, μια μικρή περιγραφή και διάφορες άλλες πληροφορίες.

Όταν επιστραφούν οι πληροφορίες αυτές, το σύστημα αναλαμβάνει την αποθήκευση τους στην βάση δεδομένων. Με την καταχώριση του βιβλίου στην βάση δεδομένων του συστήματος, δημιουργείται μια νέα εγγραφή για το βιβλίο, και παράλληλα ανατίθεται η εγγραφή του βιβλίου στην εγγραφή της βιβλιοθήκης, δημιουργώντας έτσι ένα σύνδεσμο μεταξύ της βιβλιοθήκης και του βιβλίου, που δείχνει ότι το συγκεκριμένο βιβλίο ανήκει στην συγκεκριμένη βιβλιοθήκη. Με τον τρόπο αυτό διευκολύνεται η ανάκτηση των δεδομένων σε μετέπειτα στάδιο, όπου θα επαναχρησιμοποιηθεί η εφαρμογή.

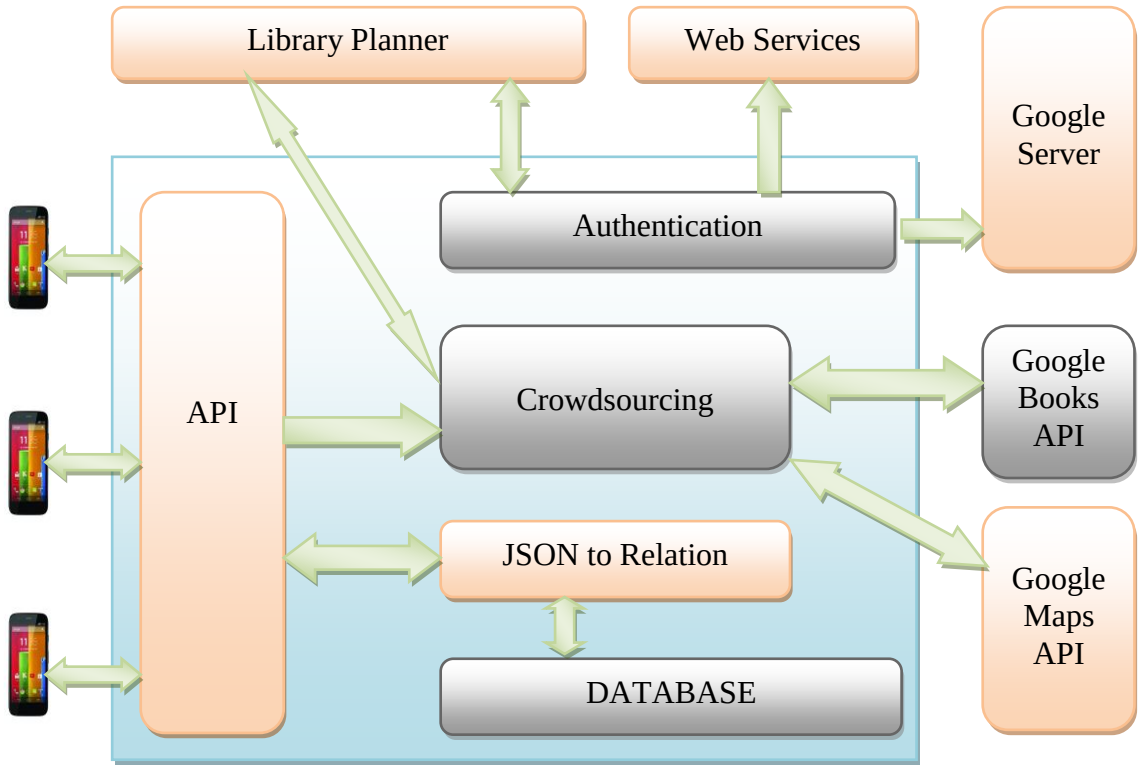
Κεφάλαιο 3

Η αρχιτεκτονική του SmartLib

3.1 Προδιαγραφές Συστήματος SmartLib+ 18

3.2 Επισκόπηση Αρχιτεκτονικής SmartLib+ 19

Κατά την ανάπτυξη του συστήματος, μελετήθηκε η υπάρχουσα αρχιτεκτονική του SmartLib, όπως παρουσιάζεται στην προηγούμενη ΑΔΕ [8]. Όπως παρουσιάζεται και στο διάγραμμα αρχιτεκτονικής SmartLib+ (Σχήμα 3.1 : Διάγραμμα αρχιτεκτονικής SmartLib+ τα έγχρωμα μέρη της αρχιτεκτονικής, είναι τα αυτά που μελετήθηκαν και αναπτύχτηκαν στη παρούσα διπλωματική εργασία, ενώ αυτά που είναι μαυρόασπρα είναι τα μέρη που υπάρχουν κοινά με την προηγούμενη αρχιτεκτονική, στο σύστημα SmartLib.



Σχήμα 3.1 : Διάγραμμα αρχιτεκτονικής SmartLib+

3.1 Προδιαγραφές Συστήματος SmartLib+

Κατά την μελέτη της παρούσας διπλωματικής εργασίας είχαμε να αντιμετωπίσουμε το πρόβλημα καταγραφής και αποτελεσματικής χρήσης των μεγάλων συλλογών βιβλίων έντυπης μορφής. Μετά από αρκετή μελέτη καταλήξαμε στις λειτουργικές προδιαγραφές που παρουσιάζονται πιο κάτω.

Λειτουργία Εισόδου/Σύνδεσης

Το σύστημα πρέπει να παρέχει τη λειτουργία στο σύστημα όπου θα γίνεται αναγνώριση του χρήστη, για αποφυγή απώλειας δεδομένων, καθώς επίσης και για προστασία των δεδομένων του χρήστη. Επιπρόσθετα, με την λειτουργία αυτή θα δημιουργηθεί προσωποποίηση των χρηστών, έτσι ώστε να μπορούν να δημιουργηθούν σχέσεις μεταξύ τους.

Λειτουργία Αποσύνδεσης

Με την λειτουργία αυτή ο χρήστης θα μπορεί να αποτρέψει την πρόσβαση σε οποιοδήποτε άτομο να έχει πρόσβαση στα δεδομένα του σε περίπτωση κοινόχρηστου υπολογιστή.

Λειτουργία Χάρτη

Είναι καλή ιδέα η γεωγραφική τοποθέτηση μιας βιβλιοθήκης, γιατί με τον τρόπο αυτό είναι εύκολο κάποιος χρήστης να εντοπίσει βιβλία που των ενδιαφέρουν και είναι κοντά στην δική του γεωγραφική θέση.

Λειτουργία Συνεργασίας

Με την λειτουργία αυτή θέλουμε να επιτρέψουμε στους χρήστες του συστήματος να δημιουργούν σχέσεις μεταξύ τους για την καλύτερη αναζήτηση βιβλίων καθώς επίσης και επιλογή κάποιου βιβλίου με βάση την γεωγραφική του θέση. Για παράδειγμα αν το ίδιο βιβλίο βρίσκεται σε δύο διαφορετικές βιβλιοθήκες όπου η κάθε μια ανήκει σε ξεχωριστό χρήστη, τότε ο χρήστης μπορεί να επιλέξει αυτό που βρίσκεται πιο κοντά του.

Λειτουργία Επικοινωνίας

Μια λειτουργία η οποία θα παρέχει την δυνατότητα στους χρήστες να επικοινωνήσουν μεταξύ με την αποστολή ηλεκτρονικών μηνυμάτων. Επιτρέπεται έτσι να ζητηθεί η ενοικίαση ενός βιβλίου από το ιδιοκτήτη του.

Λειτουργία Επικοινωνίας

Θα πρέπει να επιτρέπεται η δημιουργία βιβλιοθηκών καθώς επίσης και η εισαγωγή βιβλίων στη κάθε βιβλιοθήκη. Η εισαγωγή θα πρέπει να γίνεται αυτόματα μέσω του μοναδικού αναγνωριστικού αριθμού ISBN για αποφυγή λανθασμένων στοιχείων. Τέλος θα πρέπει να επιτρέπεται η διαγραφή μιας βιβλιοθήκης σε περίπτωση που τερματιστεί η ύπαρξη της.

3.2 Επισκόπηση Αρχιτεκτονικής SmartLib+

Η εφαρμογή του SmartLib+ αποτελείται από την διαδικτυακή διαπροσωπεία, ένα κεντρικό εξυπηρετητή και μια κεντρική βάση δεδομένων σε MySQL.

Για την αλληλεπίδραση μεταξύ της διαδικτυακής διαπροσωπίας και του εξυπηρετητή, χρησιμοποιήσαμε το ευρέως διαδεδομένο πρωτόκολλο HTTP, από το οποίο χρησιμοποιήσαμε μεθόδους όπως η get και η post. Για την διεκπεραίωση των αιτημάτων που θα χρησιμοποιούν τις μεθόδους αυτές, χρησιμοποιήθηκε η scripting γλώσσα προγραμματισμού JavaScript.

Εκτός από την επικοινωνία μεταξύ χρήστη και εξυπηρετητή, η JavaScript χρησιμοποιείται και για τις διάφορες διαδραστικές λειτουργίες που παρέχει η διαδικτυακή διαπροσωπεία. Χρησιμοποιείται για όλες τις λειτουργίες όπως η προσθήκη βιβλιοθήκης, η προσθήκη ενός νέου βιβλίου σε μια βιβλιοθήκη, η επικοινωνία μεταξύ χρηστών, καθώς επίσης και στις διάφορες λειτουργίες του χάρτη.

Σχήμα 3.2 : Τα τρία μέρη της βασικής αρχιτεκτονικής της εφαρμογής SmartLib+

Για της διάφορες λειτουργίες του εξυπηρετητή, χρησιμοποιήθηκε μια νέα τεχνολογία, η NodeJS. Η NodeJS, είναι βασισμένη σε JavaScript και είναι η πρώτη στο είδος της σε εφαρμογή JavaScript σε λειτουργίες εξυπηρετητή. Χρησιμοποιείται κυρίως για διαχείριση των HTTP αιτημάτων, με την χρήση δρομολογήσεων των διάφορων αιτημάτων σε συγκεκριμένα μονοπάτια. Όταν ένα αίτημα φτάσει σε κάποιο μονοπάτι, εκτελείται μια μέθοδος για να υλοποιηθεί μια συγκεκριμένη λειτουργία. Κάθε μονοπάτι εκτελεί μια

συγκεκριμένη και προκαθορισμένη μέθοδο. Έτσι γνωρίζουμε πως για να εκτελεστεί μια συγκεκριμένη λειτουργία πρέπει να εκτελέσουμε ένα HTTP αίτημα στο συγκεκριμένο μονοπάτι.

Για την επικοινωνία με την βάση δεδομένων του συστήματος, έχουν υλοποιηθεί μέθοδοι σε JavaScript, οι οποίες καλούνται μετά από αίτημα HTTP σε συγκεκριμένο μονοπάτι. Με την σύνδεση στη βάση εκτελούνται κανονικά τα επερωτήματα που χρειάζεται να εκτελεστούν έτσι ώστε να ενημερωθεί η βάση δεδομένων του συστήματος.

Υπάρχει συνεχής επικοινωνία με την βάση αφού, από την εκκίνηση της εφαρμογής, εκτελείται επερώτημα αρχικά για εισαγωγή του χρήστη στο σύστημα αν δεν υπάρχει, αν είναι η πρώτη φορά δηλαδή, και στη συνέχεια για φόρτωμα των βιβλιοθηκών και των βιβλίων τους στην διαδικτυακή εφαρμογή.

Το σύστημα SmartLib+, βασίζεται σε 3 βασικά υποσυστήματα. Το πρώτο είναι το front-end κομμάτι που αφορά την κυρίως διεπαφή με την οποία αλληλεπιδρά ο χρήστης. Το δεύτερο υποσύστημα αφορά το back-end κομμάτι του συστήματος και περιλαμβάνει όλες τις λειτουργίες που αφορούν την βάση δεδομένων, την αποστολή ηλεκτρονικών μηνυμάτων καθώς επίσης και τις διάφορες κλήσεις προς το Google Books API για την ανάκτηση των δεδομένων στην περίπτωση εισαγωγής ενός νέου βιβλίου.

Υποσύστημα 1: Διεπαφή (Front-End)

Το υποσύστημα αυτό περιλαμβάνει όλες τις λειτουργίες οι οποίες παρέχονται στο χρήστη για μια πλήρης και οργανωμένη εμπειρία κατά την χρήση του συστήματος. Η εμπειρία που πρέπει να παρέχεται στο χρήστη πρέπει να είναι όσο το δυνατό πιο κατανοητή καθώς επίσης και χωρίς σφάλματα. Κατά την χρήση του συστήματος ο χρήστης πρέπει να ενημερώνεται για τυχόν σφάλματα, κατά την έγερση των οποίων το σύστημα πρέπει να αντιδρά και να τα αντιμετωπίζει χωρίς να τερματίζει τις τυχόν ενέργειες του χρήστη. Η απόκριση του συστήματος πρέπει να είναι όσο το δυνατό πιο έγκαιρη και χωρίς δυσλειτουργίες. Στο υποσύστημα αυτό είναι υλοποιημένες όλες οι λειτουργίες που αφορούν την διαχείριση των βιβλιοθηκών του χρήστη. Πιο κάτω ακολουθεί αναλυτική περιγραφή των διάφορων λειτουργιών.

Είσοδος Χρήστη: Ο χρήστης με την χρήση του λογαριασμού του στο Google, μπορεί να εισέλθει στο σύστημα. Αυτό επιτυγχάνεται με την ενέργεια του να κάνει κλικ στο κουμπί που υπάρχει στην κύρια οθόνη. Με την ενέργεια του αυτή, καλείτε μια συνάρτηση στο front-end κομμάτι μέσω της οποίας το σύστημα επικοινωνεί με το back-end κομμάτι. Στη συνέχεια

γίνεται εξακρίβωση στον στοιχείων που παρείχε ο χρήστης και επιστρέφεται το μήνυμα μέσω του οποίου ο χρήστης μπορεί να συνδεθεί στο σύστημα, εάν και εφόσον τα στοιχεία που παρείχε είναι αληθή.

Πίνακας Ελέγχου

Προσθήκη Βιβλιοθήκης: Παρέχεται στο χρήστη η δυνατότητα να προσθέσει μια νέα βιβλιοθήκη. Με το πάτημα του συγκεκριμένου κουμπιού, εμφανίζεται ένα νέο σημείο στο χάρτη με την μορφή βιβλιοθήκης. Η νέα αυτή βιβλιοθήκη δεν περιέχει κανένα βιβλίο, τοποθετείται στο σημείο όπου βρίσκεται ο χρήστης και η ονομασία της είναι DefaultXX όπου XX είναι αύξον αριθμός.

Επεξεργασία Βιβλιοθήκης: Ο χρήστης έχει την δυνατότητα εφόσον δημιουργήσει κάποια βιβλιοθήκη να την επιλέξει από την λίστα βιβλιοθηκών. Η λίστα αυτή ενημερώνεται αυτόματα μετά την εισαγωγή της νέας βιβλιοθήκης. Με την επιλογή της βιβλιοθήκης, ο χρήστης έχει την δυνατότητα να εισάγει ένα νέο όνομα το οποίο είναι το επιθυμητό για την συγκεκριμένη βιβλιοθήκη και έτσι να μετονομάσει την βιβλιοθήκη της επιλογής του.

Διαγραφή Βιβλιοθήκης: Το σύστημα παρέχει επίσης την δυνατότητα διαγραφής μιας βιβλιοθήκης. Η διαδικασία η οποία πρέπει να ακολουθήσει κάποιος χρήστης για την διαγραφή μιας βιβλιοθήκης είναι η ακόλουθη. Ο χρήστης πρέπει καταρχάς να έχει δημιουργήσει κάποια νοητή βιβλιοθήκη. Αν δεν υπάρχει κάποια βιβλιοθήκη στο σύστημα, τότε η λίστα με τις βιβλιοθήκες είναι κενή. Για να διαγράψει κάποιος κάποια βιβλιοθήκη, πρέπει πρώτα να επιλέξει την βιβλιοθήκη την οποία επιθυμεί να διαγράψει, και μετά πατώντας το κουμπί διαγραφής, να αφαιρέσει την επιλογή του από το σύστημα. Ο χρήστης ενημερώνεται ότι η ενέργεια του θα είναι μόνιμη και μη αναστρέψιμη. Η ανακοίνωση αυτή χρησιμοποιείται για απαλλαγή των δημιουργών από τις ευθύνες. Μετά την διαγραφή της βιβλιοθήκης, η λίστα βιβλιοθηκών ενημερώνεται αυτόματα.

Αποστολή Αιτήματος Συνεργασίας: Με την λειτουργία αυτή, ο χρήστης μπορεί να εισάγει την ηλεκτρονική διεύθυνση ενός από τους φίλους/συναδέλφους του, με τον οποίο επιθυμεί να συνεργαστεί. Η συνεργασία αυτή περιλαμβάνει εμφάνιση των βιβλιοθηκών με τα βιβλία του συνεργάτη στο χάρτη του χρήστη. Η εμφάνιση αυτή είναι αμφίδρομη. Με την αποστολή του αιτήματος συνεργασίας, αποστέλλεται ένα ηλεκτρονικό μήνυμα στο χρήστη του οποίου εισάχθηκε η ηλεκτρονική διεύθυνση, με το οποίο ενημερώνεται ότι υπάρχει ένα αίτημα συνεργασίας στο SmartLib+. Στο μήνυμα υπάρχει επίσης μια διεύθυνση μέσω της οποίας ο

παραλήπτης, κάνοντας κλικ, αποδέχεται το αίτημα, και καταχωρείται στη βάση δεδομένων του συστήματος η σχέση συνεργασίας των δύο χρηστών.

Αποστολή Μηνύματος: Κατά την σύνδεση του χρήστη, το σύστημα ελέγχει κατά πόσο ο συγκεκριμένος χρήστης έχει καταχωρίσεις στη βάση δεδομένων που να δείχνουν συνδέσμους συνεργασίας. Στην περίπτωση που υπάρχουν τέτοιοι σύνδεσμοι, δημιουργείται μια λίστα με τις διευθύνσεις των χρηστών αυτών. Αν ο χρήστης κατά την χρήση του συστήματος διαπιστώσει ότι κάποιος από τους συνεργάτες του διαθέτει σε κάποια από τις βιβλιοθήκες του ένα από τα βιβλία τα οποία ενδιαφέρεται να διαβάσει είτε για λόγους έρευνας είτε για λόγους ψυχαγωγίας, είτε για οποιοδήποτε άλλο λόγο, έχει την δυνατότητα από το σύστημα, να στείλει ένα μήνυμα στο συγκεκριμένο συνεργάτη του, και να ζητήσει, αν και εφόσον το βιβλίο είναι διαθέσιμο προς δανεισμό, να δανειστεί το αντίτυπο, και με την ολοκλήρωση της χρήσης του να το επιστρέψει στον ιδιοκτήτη του.

Υποσύστημα 2: Διακομηστή (Back-End)

Το υποσύστημα διακομηστή είναι υλοποιημένος στην πλατφόρμα NodeJS. Μια πλατφόρμα που χρησιμοποιεί την scripting γλώσσα JavaScript. Το υποσύστημα διακομηστή υλοποιεί τις λειτουργίες που αφορούν την είσοδο και έξοδο δεδομένων από τη βάση δεδομένων, καθώς επίσης και την αποστολή ηλεκτρονικών μηνυμάτων στις περιπτώσεις αιτήματος συνεργασίας, καθώς επίσης και στην προσπάθεια επικοινωνίας μεταξύ των χρηστών. Πιο κάτω ακολουθεί λεπτομερής περιγραφή των διάφορων λειτουργιών του υποσυστήματος διακομηστή. Όλες οι λειτουργίες εκτελούνται μετά από αιτήματα HTTP get και post.

Προσθήκη Νέου Χρήστη: Η λειτουργία αυτή εκτελείται μετά από αίτημα post. Το αίτημα έχει σαν παραμέτρους το όνομα, το επίθετο και την ηλεκτρονική διεύθυνση του χρήστη. Αρχικά ελέγχεται αν ο χρήστης είναι καταχωρημένος στη βάση δεδομένων του συστήματος. Αν ο χρήστης είναι ήδη καταχωρημένος, επιστρέφεται ο κωδικός κατάστασης 200. Αν ο χρήστης δεν υπάρχει στην βάση δεδομένων, τότε το σύστημα επικοινωνεί με την βάση δεδομένων και εκτελείται ένα επερώτημα για την εισαγωγή του χρήστη. Αν η εισαγωγή του χρήστη ήταν επιτυχής επιστέφεται κωδικός κατάσταση 200 και μήνυμα που ενημερώνει για την επιτυχή εισαγωγή στο σύστημα. Αλλιώς ενημερώνεται ο χρήστης για την αδυναμία εισαγωγής του στη βάση δεδομένων του συστήματος.

Ανάκτηση Βιβλιοθηκών: Η εκτέλεση της λειτουργίας αυτής απαιτεί get αίτημα. Στο αίτημα αυτό είναι απαραίτητη η παράμετρος με την ηλεκτρονική διεύθυνση του χρήστη. Το υποσύστημα συνδέεται με την βάση δεδομένων. Αν υπάρχει οποιοδήποτε πρόβλημα κατά

την σύνδεση, ο χρήστης ενημερώνεται για το σφάλμα. Αν η σύνδεση είναι επιτυχής, εκτελείται επερώτημα επιλογής των βιβλιοθηκών που ανήκουν στο χρήστη, καθώς επίσης και επιλογής των βιβλίων που ανήκουν στης ανακτηθείσες βιβλιοθήκες. Στη συνέχεια γίνεται συνδυασμός των δεδομένων που ανακτήθηκαν για να επιστραφούν σε μορφή JSON την οποία έχει υλοποιηθεί το σύστημα να αναγνωρίζει και να διαχειρίζεται. Τέλος, με την ολοκλήρωση της προετοιμασίας των δεδομένων, επιστρέφονται στο υποσύστημα διεπαφής, όπου και φορτώνονται.

Ανάκτηση Συνεργατών: Ένα get αίτημα είναι απαραίτητο για την λειτουργία αυτή. Απαραίτητη παράμετρος είναι η ηλεκτρονική διεύθυνση του χρήστη. Με την λήψη του αιτήματος, το υποσύστημα προσπαθεί να συνδεθεί με την βάση δεδομένων. Σε περίπτωση αποτυχίας ενημερώνεται ο χρήστης. Σε περίπτωση επιτυχής σύνδεσης, εκτελείται επερώτημα επιλογής όλων των συνεργατών του χρήστη. Τα δεδομένα επιστρέφονται στο υποσύστημα διεπαφής σε μορφή JSON. Με την λήψη των δεδομένων το υποσύστημα διεπαφής φορτώνει τα δεδομένα καθιστώντας τα διαθέσιμα για τον χρήστη.

Αποστολή Μηνύματος: Το υποσύστημα διεπαφής διενεργεί ένα αίτημα post το οποίο αναλαμβάνει το υποσύστημα διακομηστή. Το αίτημα έχει ως παραμέτρους τις ηλεκτρονικές διευθύνσεις του αποστολέα και του παραλήπτη καθώς επίσης και το μήνυμα. Η δύο διευθύνσεις είναι απαραίτητες, του παραλήπτη για ευνόητους λόγους, αφού είναι ο προορισμός του μηνύματος. Του αποστολέα είναι απαραίτητη η ύπαρξη της ηλεκτρονικής του διεύθυνσης για να μπορέσει ο παραλήπτης να απαντήσει στο μήνυμα. Για την αποστολή του μηνύματος χρησιμοποιείται μια υποδομή υπηρεσιών αποστολής ηλεκτρονικών μηνυμάτων. Η υπηρεσία αυτή είναι η Mandrill και είναι ευρέως γνωστή και χρησιμοποιήσιμη. Παρέχει ένα εύκολο τρόπο δημιουργίας ρυθμίσεων για το πρωτόκολλο SMTP καθώς επίσης και μια διεπαφή προγράμματος εφαρμογής (API), για μια πιο ολοκληρωμένη χρήση της υπηρεσίας.

Αποστολή Αιτήματος: Για την χρήση της λειτουργίας αποστολής αιτήματος είναι απαραίτητο ένα αίτημα get. Στο αίτημα οι παράμετροι είναι οι δύο ηλεκτρονικές διευθύνσεις των ατόμων μεταξύ των οποίο θα δημιουργηθεί σύνδεσμος συνεργασίας. Στη περίπτωση μας, η μια ηλεκτρονική διεύθυνση είναι αυτή του χρήστη. Η άλλη είναι η ηλεκτρονική διεύθυνση του ατόμου με το οποίο ο χρήστης επιθυμεί να γίνουν συνεργάτες. Με την χρήση της υποδομής υπηρεσιών αποστολής ηλεκτρονικών μηνυμάτων Mandrill[1] αποστέλλεται μήνυμα στο άτομο που επιθυμεί ο χρήστης, με το οποίο ο παραλήπτης ενημερώνεται ότι υπάρχει ένα

[1] https://www.mandrill.com/

αίτημα συνεργασίας στο SmartLib+. Παράλληλα παρέχεται ένας προκαθορισμένος σύνδεσμος, μέσω του οποίου, ο παραλήπτης αποδέχεται το αίτημα με το να επιλέξει το σύνδεσμο αυτό. Με την αλληλεπίδραση του παραλήπτη με το σύνδεσμο, εκτελείται η λειτουργία αποδοχής αιτήματος, περιγραφή της οποίας ακολουθεί πιο κάτω.

Αποδοχή Αιτήματος: Μετά την αποστολή ενός αιτήματος συνεργασίας, είναι απαραίτητη η διαχείριση της αποδοχής του αιτήματος. Η λειτουργία αποδοχής αιτήματος συνεργασίας εκτελείται μετά από ένα αίτημα get το οποίο εγείρεται μέσω του συνδέσμου που αποστέλλεται στον παραλήπτη του αιτήματος. Το αίτημα αυτό περιέχει της δύο ηλεκτρονικές διευθύνσεις, του αποστολέα και του παραλήπτη του αιτήματος. Με την αποδοχή του αιτήματος και την εκτέλεση της λειτουργίας, το υποσύστημα διακομηστή, συνδέεται με την βάση δεδομένων του συστήματος. Εάν δεν υπάρχει σχέση συνεργασίας μεταξύ των δύο μελών, εισάγονται στη βάση δεδομένων οι ηλεκτρονικές διευθύνσεις των δύο με τέτοιο τρόπο ώστε να υπάρχει αμφίδρομη σχέση συνεργασίας. Με την αποδοχή του αιτήματος ο χρήστης αποδέχεται να μοιράζεται τα δεδομένα του με τον χρήστη που απέστειλε το αίτημα συνεργασίας. Ταυτόχρονα, ο αποστολέας αποδέχεται να μοιράζεται και αυτός με τη σειρά του τα δικά του δεδομένα.

Εισαγωγή Νέας Βιβλιοθήκης: Με ένα αίτημα post και με παραμέτρους το όνομα της βιβλιοθήκης, της συντεταγμένες της γεωγραφικής τοποθεσίας της βιβλιοθήκης, γεωγραφικό μήκος και γεωγραφικό πλάτος, και την ηλεκτρονική διεύθυνση του ιδιοκτήτη της βιβλιοθήκης γίνεται εισαγωγή της βιβλιοθήκης.

Ενημέρωση Συντεταγμένων Βιβλιοθήκης: Για την ενημέρωση των συντεταγμένων μιας βιβλιοθήκης, του γεωγραφικού μήκους και πλάτους, είναι αναγκαίο ένα post αίτημα με παραμέτρους τον μοναδικό αναγνωριστικό αριθμό – ταυτότητα της βιβλιοθήκης, και τις νέες συντεταγμένες της βιβλιοθήκης, γίνεται ενημέρωση της εν λόγω βιβλιοθήκης.

Διαγραφή Βιβλιοθήκης: Με ένα αίτημα post και παράμετρο τον μοναδικό αναγνωριστικό αριθμό – ταυτότητα της βιβλιοθήκης, μπορούμε να διαγράψουμε την βιβλιοθήκη με την ταυτότητα αυτή από την βάση δεδομένων του συστήματος. Αρχικά γίνεται έλεγχος αν η βιβλιοθήκη υπάρχει και στη συνέχεια διαγράφεται η βιβλιοθήκη, καθώς επίσης και ο σύνδεσμος που υπάρχει μεταξύ της βιβλιοθήκης και του ιδιοκτήτη της. Αν η βιβλιοθήκη δεν υπάρχει τότε δεν είναι απαραίτητη οποιαδήποτε ενέργεια.

Εισαγωγή Νέου Βιβλίου: Για την εισαγωγή ενός νέου βιβλίου είναι απαραίτητο να εκτελεστεί ένα αίτημα post προς το μέρος του διακομηστή της εφαρμογής. Για την εισαγωγή του βιβλίου είναι απαραίτητες οι παράμετροι με την ταυτότητα της βιβλιοθήκης στην οποία θα ανήκει το βιβλίο, καθώς επίσης και ο μοναδικός αναγνωριστικός αριθμός του βιβλίου ISBN. Με την χρήση του μοναδικού αριθμού του βιβλίου, γίνεται κλήση στη διεπαφή προγράμματος διεπαφής της Google για βιβλία, όπου ανακτώνται τα δεδομένα που αφορούν το εν λόγο βιβλίο. Με την ανάκτηση των δεδομένων, τα οποία επιστρέφονται σε μορφή JSON στο σύστημα, γίνεται εισαγωγή του νέου βιβλίου στη βάση δεδομένων, και δημιουργείται σύνδεσμος μεταξύ του βιβλίου και της βιβλιοθήκης της οποίας εισάχθηκε η ταυτότητα.

Μετονομασία Βιβλιοθήκης: Ένα αίτημα post είναι αρκετό για την μετονομασία μιας βιβλιοθήκης που υπάρχει στο σύστημα. Αφού επιλέξει την βιβλιοθήκη που επιθυμεί να μετονομάσει ο χρήστης, και αφού συμπληρώσει το πεδίο με το όνομα που θέλει να δώσει στη βιβλιοθήκη, εκτελείται το αίτημα με παραμέτρους την ταυτότητα της βιβλιοθήκης που θα μετονομαστεί και την νέα ονομασία. Ο διακομηστής του συστήματος λαμβάνει το αίτημα και εκτελεί μια λειτουργία ενημέρωσης της βάσης δεδομένων όπου αλλάζεται το όνομα της βιβλιοθήκης με το νέο που δόθηκε από το χρήστη.

Κεφάλαιο 4

Βάση Δεδομένων

4.1 Σχεδίαση Βάσης Δεδομένων 26

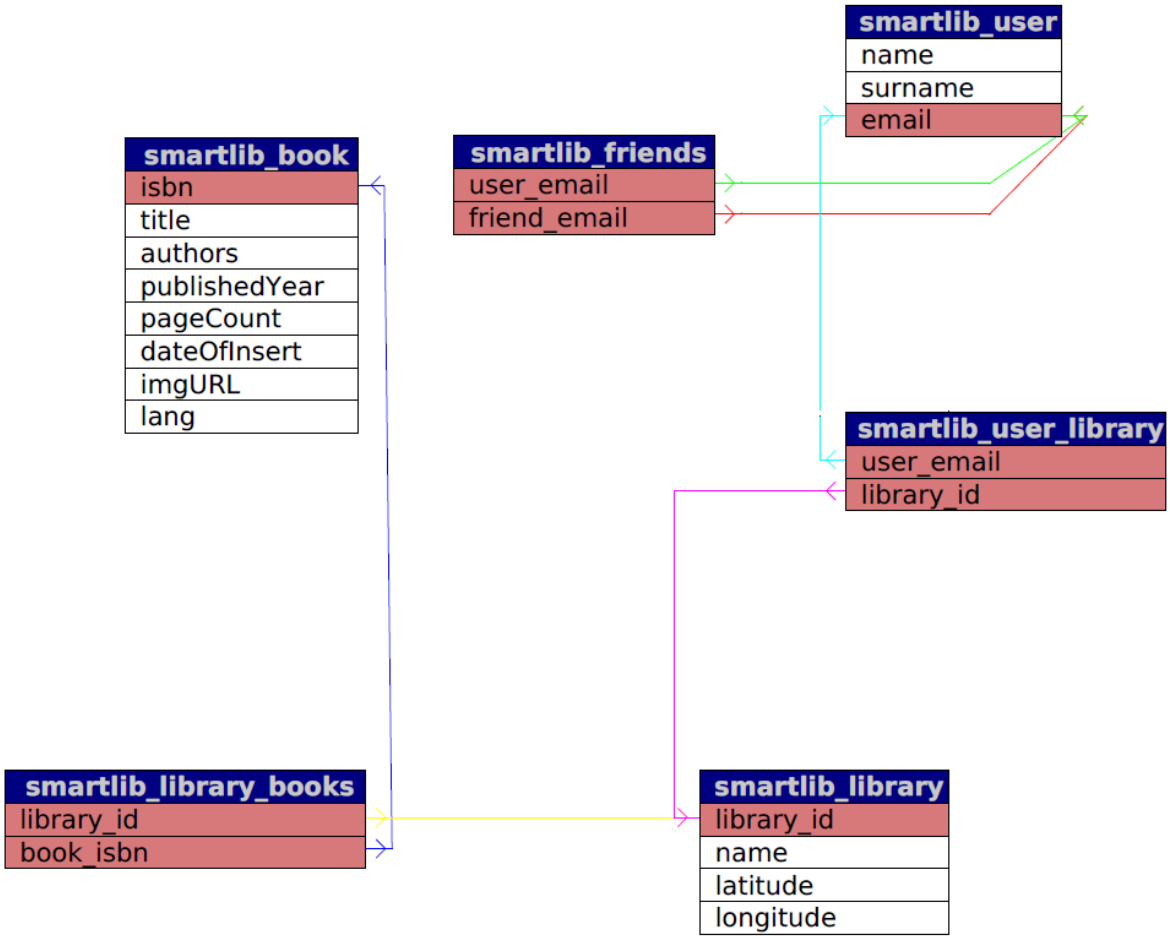
Για τις ανάγκες του συστήματος, για την διαχείριση των δεδομένων που θα συλλέγονται κατά διαστήματα, ήταν απαραίτητη η ανάπτυξη μιας βάσης δεδομένων η οποία θα μπορεί να αποθηκεύει σωστά όλα τα δεδομένα. Για την αποθήκευση των δεδομένων είναι απαραίτητη και η πιο αποδοτική χρήση του χώρου χωρίς την σπατάλη του με την αποθήκευση αχρείαστων πληροφοριών.

Μια αποδοτική βάση δεδομένων συμβάλει αρκετά στην πιο αποτελεσματική και αποδοτική λειτουργία του συστήματος, διατηρώντας την συνοχή, καθώς επίσης και βελτίωση της εμπειρικής χρήσης του συστήματος.

4.1 Σχεδίαση Βάσης Δεδομένων

Η βάση δεδομένων του συστήματος είναι υλοποιημένη σε MySQL. Αποθηκεύει όλες τις πληροφορίες που αφορούν τις βιβλιοθήκες των χρηστών, όπως τις συντεταγμένες, την ονομασία και τους αναγνωριστικούς κωδικούς αριθμούς των βιβλίων. που περιέχουν.

Επίσης αποθηκεύονται οι σχέσεις συνεργασίας μεταξύ των χρηστών, για διευκόλυνση παρουσίασης των βιβλιοθηκών των συνεργατών του κάθε χρήστη. Τέλος αποθηκεύονται οι διάφορες πληροφορίες που αφορούν τα διάφορα βιβλία, όπως ο τίτλος, περιγραφή του βιβλίου, αναγνωριστικός κωδικός αριθμός ISBN, συγγραφείς, και άλλες πληροφορίες.
Πιο κάτω ακολουθεί αναλυτική περιγραφεί του περιεχομένου της βάσης δεδομένων, και πιο συγκεκριμένα πληροφορίες για τους πίνακες που κρατούν/αποθηκεύουν τα διάφορα δεδομένα.



Σχήμα 4.1 : Σχεσιακό Διάγραμμα Βάσης Δεδομένων.

Πιο πάνω παρουσιάζεται το σχεσιακό διάγραμμα της βάσης δεδομένων του συστήματος (Σχήμα 4.1). Αποτελείται από του πίνακες στους οποίους αποθηκεύονται οι πληροφορίες που θα εισαχθούν και θα επεξεργαστούν από τους χρήστες του συστήματος.

Πιο κάτω ακολουθεί εκτενέστερη περιγραφή από τους πίνακες της βάσης δεδομένων, καθώς γίνεται περιγραφή στις πληροφορίες που κρατά το κάθε πεδίο. Ταυτόχρονα, γίνεται αναφορά και στους τύπους δεδομένων που αποθηκεύονται στο κάθε πεδίο.

Column	Type	Null	Default	Extra	Links to
isbn	varchar(13)	No			
title	varchar(200)	No			
authors	varchar(100)	No			
publishedYear	int(5)	No			
pageCount	int(5)	No			
dateOfInsert	timestamp	No	CURRENT_TIMESTAMP		
imgURL	varchar(200)	No	images/hocover.png		
lang	varchar(2)	No			

Πίνακας 4.1 : Πίνακας smartlib_book

• isbn: Αποθηκεύεται ο μοναδικός αναγνωριστικός αριθμός του βιβλίου. Όπως είναι γνωστό, ο συγκεκριμένος αναγνωριστικός αριθμός αποτελείται από 13 ψηφία.

• title: Αποθηκεύεται ο τίτλος του βιβλίου. έχει ανατεθεί σε συμβολοσειρά 200 χαρακτήρων λόγο του αβέβαιου μήκους του τίτλου.

Πίνακας 4.1 : Πίνακας smartlib_book

• authors: Συμβολοσειρά που αποθηκεύει τον συγγραφέα, ή/και συγγραφείς του συγκεκριμένου βιβλίου.

• publishedYear: Ακέραιος αριθμός μήκους 5 για την αποθήκευση της χρονολογίας έκδοσης του εν λόγο βιβλίου.

• pageCount: Στον ακέραιο αυτό μήκους 5 αποθηκεύεται ο αριθμός των σελίδων του βιβλίου.

• dateOfInsert: Χρονική σήμανση για την αποθήκευση της ημερομηνίας εισαγωγής στο σύστημα της ηλεκτρονικής βιβλιοθήκης.

• imgURL: Συμβολοσειρά μήκους 200 χαρακτήρων για την αποθήκευση της ηλεκτρονικής διεύθυνσης στην οποία μπορούμε να εντοπίσουμε την εικόνα του εξωφύλλου του βιβλίου.

• lang: Συμβολοσειρά μήκους δύο για την αποθήκευση της σήμανσης γλώσσας. Η σήμανση αποτελείται από δύο γράμματα τα οποία είναι συντομογραφία της γλώσσας στην αγγλική έκδοση. Για παράδειγμα, για την αγγλική γλώσσα έχουμε την συντομογραφία en για το English, ενώ για το Greek έχουμε την συντομογραφία gr. Με παρόμοιο τρόπο έχουμε συντομογραφίες για όλες τις γλώσσες.

Column	Type	Null	Default	Extra	Links to
user_email	varchar(50)	No			smartlib_user -> email
friend_email	varchar(50)	No			smartlib_user -> email

Πίνακας 4.2 : Πίνακας smartlib_friends

Column	Type	Null	Default	Extra	Links to
library_id	int(11)	No		auto_increment	
name	varchar(45)	No			
latitude	decimal(9,7)	No			
longitude	decimal(9,7)	No			

Πίνακας 4.3 : Πίνακας smartlib_library

Πίνακας 4.2 : Πίνακας smartlib_friends

• **user_email**: Συμβολοσειρά μήκους 50 για την αποθήκευση της ηλεκτρονικής διεύθυνσης του χρήστη του συστήματος.

Links to: Δημιουργείται ένας σύνδεσμος προς το πίνακα smartlib_user και συγκεκριμένα στη στήλη email, για να υπάρχει συσχέτιση μεταξύ των δύο πινάκων για ευκολία ανάκτησης δεδομένων.

• **friend_email**: Συμβολοσειρά μήκους 50 για την αποθήκευση της ηλεκτρονικής διεύθυνσης του συνεργάτη του χρήστη του συστήματος.

Links to: Δημιουργείται ένας σύνδεσμος προς το πίνακα smartlib_user και συγκεκριμένα στη στήλη email, για να υπάρχει συσχέτιση μεταξύ των δύο πινάκων για ευκολία ανάκτησης δεδομένων.

Πίνακας 4.3 : Πίνακας smartlib_library

• **library_id:** Ακέραιος αριθμός μήκους 11 για την αποθήκευση της ταυτότητας της βιβλιοθήκης. Η ταυτότητα ανατίθεται αυτόματα στην βιβλιοθήκη και αποτελείται από ένα αριθμό που είναι κατά ένα μεγαλύτερος από τον αριθμό ταυτότητας της βιβλιοθήκης που εισάχθηκε τελευταία.

• **name:** Στη συμβολοσειρά name μήκους 45 χαρακτήρων αποθηκεύεται το όνομα της βιβλιοθήκης.

• **latitude:** Δεκαδικός αριθμός μήκους 9 χαρακτήρων με 7 δεκαδικά ψηφία για την αποθήκευση του γεωγραφικού πλάτους.

• **longitude:** Δεκαδικός αριθμός μήκους 9 χαρακτήρων με 7 δεκαδικά ψηφία για την αποθήκευση του γεωγραφικού μήκους.

Column	Type	Null	Default	Extra	Links to
library_id	in(11)	No			smartlib_library -> library_id
book_isbn	varchar(13)	No			smartlib_book -> isbn

Πίνακας 4.4 : Πίνακας smartlib_library_books

Column	Type	Null	Default	Extra	Links to
name	varchar(20)	No			
surname	varchar(30)	No			
email	varchar(60)	No			

Πίνακας 4.5 : Πίνακας smartlib_user

Πίνακας 4.4 : Πίνακας smartlib_library_books

• **library_id**: Ακέραιος αριθμός μήκους 11 για την αποθήκευση της ταυτότητας της βιβλιοθήκης. Η ταυτότητα ανατίθεται αυτόματα στην βιβλιοθήκη και αποτελείται από ένα αριθμό που είναι κατά ένα μεγαλύτερος από τον αριθμό ταυτότητας της βιβλιοθήκης που εισάχθηκε τελευταία.

Links to: Δημιουργείται ένας σύνδεσμος προς το πίνακα smartlib_library και συγκεκριμένα στη στήλη library_id, για να υπάρχει συσχέτιση μεταξύ των δύο πινάκων για ευκολία ανάκτησης δεδομένων.

• **book_isbn**: Αποθηκεύεται ο μοναδικός αναγνωριστικός αριθμός του βιβλίου. Όπως είναι γνωστό, ο συγκεκριμένος αναγνωριστικός αριθμός αποτελείται από 13 ψηφία.

Links to: Δημιουργείται ένας σύνδεσμος προς το πίνακα smartlib_book και συγκεκριμένα στη στήλη isbn, για να υπάρχει συσχέτιση μεταξύ των δύο πινάκων για ευκολία ανάκτησης δεδομένων.

Πίνακας 4.5 : Πίνακας smartlib_user

• **name:** Στη συμβολοσειρά name μήκους 45 χαρακτήρων αποθηκεύεται το όνομα του χρήστη.

• **longitude:** Δεκαδικός αριθμός μήκους 9 χαρακτήρων με 7 δεκαδικά ψηφία για την αποθήκευση του γεωγραφικού μήκους.

Column	Type	Null	Default	Extra	Links to
user_email	varchar(50)	No			smartlib_user -> email
library_id	int(11)	No			smartlib_library -> library_id

Πίνακας 4.6 : Πίνακας smartlib_user_library

Πίνακας 4.6 : Πίνακας smartlib_user_library

• **user_email:** Συμβολοσειρά μήκους 50 για την αποθήκευση της ηλεκτρονικής διεύθυνσης του χρήστη του συστήματος.

Links to: Δημιουργείται ένας σύνδεσμος προς το πίνακα smartlib_user και συγκεκριμένα στη στήλη email, για να υπάρχει συσχέτιση μεταξύ των δύο πινάκων για ευκολία ανάκτησης δεδομένων.

• **library_id:** Ακέραιος αριθμός μήκους 11 για την αποθήκευση της ταυτότητας της βιβλιοθήκης. Η ταυτότητα ανατίθεται αυτόματα στην βιβλιοθήκη και αποτελείται από ένα αριθμό που είναι κατά ένα μεγαλύτερος από τον αριθμό ταυτότητας της βιβλιοθήκης που εισάχθηκε τελευταία.

Links to: Δημιουργείται ένας σύνδεσμος προς το πίνακα smartlib_library και συγκεκριμένα στη στήλη library_id, για να υπάρχει συσχέτιση μεταξύ των δύο πινάκων για ευκολία ανάκτησης δεδομένων

Κεφάλαιο 5

Υλοποίηση SmartLib+

5.1 Τεχνολογικό Υπόβαθρο 32
5.2 Υλοποίηση Διαδικτυακής Εφαρμογής 38

5.1 Τεχνολογικό Υπόβαθρο

HTML

Για την υλοποίηση του συστήματος χρησιμοποιήθηκαν διάφορες τεχνολογίας προγραμματισμού διαδικτυακών εφαρμογών. Για την κυρίως διεπαφή, χρησιμοποιήθηκε η γλώσσα σήμανσης HTML 5[2] . Η HTML 5[3] είναι η κύρια τεχνολογία που χρησιμοποιείται για την δόμηση και παρουσίαση του περιεχομένου στον παγκόσμιο ιστό. Κύριοι στόχοι της είναι η βελτίωση της ώστε να υποστηρίζει τις τελευταίες μορφές πολυμέσων ενώ παράλληλα να παραμένει κατανοητή από τους ανθρώπους καθώς επίσης και από τους υπολογιστές.

JavaScript

Για τις διάφορες λειτουργίες του συστήματος χρησιμοποιήθηκε η δυναμική γλώσσα προγραμματισμού JavaScript[4,5] . Χρησιμοποιείται συχνά σε διαδικτυακές εφαρμογές όπου η υλοποίηση επιτρέπει την χρήση διάφορων script για την αλληλεπίδραση με τον χρήστη. Η δυναμικότητα που παρέχεται από την γλώσσα προγραμματισμού, επιτρέπει στο προγραμματιστή ιστού την χρήση αντικειμενοστρέφειας κατά την ανάπτυξη της εφαρμογής. Κατά την υλοποίηση του συστήματος χρησιμοποιήθηκαν τρείς νέες τεχνολογίες πλαισίου.

Πρώτο, χρησιμοποιήθηκε το framework AngularJS. Το AngularJS είναι ένα ανοικτού πηγαίου κώδικα framework, το οποίο διατηρείται από την Google. Στόχος του συγκεκριμένου framework είναι η απλούστευση τόσο της υλοποίησης, όσο και στην δοκιμή

[2] http://en.wikipedia.org/wiki/HTML
[3] http://www.w3.org/wiki/HTML/Elements
[4] http://en.wikipedia.org/wiki/JavaScript
[5] http://www.w3schools.com/js/

των διάφορων εφαρμογών που υλοποιούνται με την χρήση του συγκεκριμένου framework. Κύρια αρχιτεκτονική που χρησιμοποιεί το AngularJS είναι το model-view-controller (MVC) μέσω της οποίας όταν υπάρξουν αλλαγές σε ένα από τα τρία μέρη της συγκεκριμένης αρχιτεκτονικής, χρειάζονται ελάχιστες μέχρι καθόλου αλλαγές στα υπόλοιπα δύο. Από την κατάληξη JS θα μπορούσε εύκολα κάποιος να καταλάβει ότι το framework AngularJS[6,7 8] βασίζεται στην scripting γλώσσα JavaScript. Το AngularJS χρησιμοποιήθηκε κατά την υλοποίηση του front-end μέρους της εφαρμογής και κατά κύριο μέρος για την υλοποίηση των διάφορων λειτουργιών της διεπαφής. Μερικές από της λειτουργίες που υλοποιήθηκαν με το συγκεκριμένο framework, είναι η τοποθέτηση των σημείων των βιβλιοθηκών στο χάρτη, ο ίδιος ο χάρτης είναι υλοποιημένος σε AngularJS η οποία φορτώνει τους χάρτες από την διεπαφή προγράμματος εφαρμογής (API) που παρέχει η Google για χάρτες, καθώς επίσης και το σύνολο των λειτουργιών του πίνακα ελέγχου, όπως η προσθήκη, αφαίρεση και επεξεργασία των βιβλιοθηκών, η αποστολή αιτημάτων συνεργασίας και των μηνυμάτων για επικοινωνία.

Κατά δεύτερον, χρησιμοποιήθηκε μια πλατφόρμα η οποία είναι επίσης βασισμένη σε JavaScript. Η πλατφόρμα αυτή ονομάζεται NodeJS[9,10] και είναι ανοικτού πηγαίου κώδικα κάτω από την άδεια MIT. Η συγκεκριμένη πλατφόρμα παρέχει μια αρχιτεκτονική η οποία είναι event-driven, δηλαδή οδηγείται από τα διάφορα συμβάντα. Παρέχει επίσης μια διεπαφή προγράμματος εφαρμογής (API) η οποία είναι non-blocking I/O. Αυτό σημαίνει ότι οι διάφορες λειτουργίες εισόδου/εξόδου γίνονται ασύγχρονα και εγείρουν κάποιο συμβάν όταν ολοκληρωθούν. Με τον τρόπο αυτό περιορίζεται ο χρόνος απόκρισης του συστήματος και κατά συνέπεια βελτιώνεται η εμπειρία του χρήστη. Η NodeJS περιέχει μια βιβλιοθήκη η οποία επιτρέπει στις διάφορες εφαρμογές που την χρησιμοποιούν να συμπεριφέρονται σαν διακομιστές (servers) χωρίς να υπάρχει η ανάγκη λογισμικού όπως Apache HTTP Server και IIS. Τα μόνα αρνητικά με σε σύγκριση με ένα Apache Server είναι ότι η NodeJS εκτελείται σε ένα μόνο νήμα (thread) και σε περίπτωση σφάλματος μπορεί να καταρρεύσει το σύστημα μας. Επίσης δεν υπάρχει τόσο καλή τεκμηρίωση της NodeJS όσο του Apache. Στην υλοποίηση της δικής μας εφαρμογής, η NodeJS χρησιμοποιήθηκε όπως προαναφέρθηκε για την υλοποίηση του back-end κομματιού της εφαρμογής, που υλοποιεί λειτουργίες που αφορούν την βάση δεδομένων καθώς επίσης και την αποστολή ηλεκτρονικών μηνυμάτων στις περιπτώσεις αιτημάτων συνεργασίας και διάφορων μηνυμάτων επικοινωνίας.

[6] http://en.wikipedia.org/wiki/AngularJS
[7] http://www.w3schools.com/angular/
[8] https://angularjs.org/
[9] http://en.wikipedia.org/wiki/Node.js
[10] https://nodejs.org/

Τέλος, χρησιμοποιήθηκε ένα άλλο framework το οποίο βασίζεται και αφορά την πλατφόρμα NodeJS.Το συγκεκριμένο framework είναι γραμμένο σε JavaScript και είναι ανοικτού πηγαίου κώδικα, κάτω από την άδεια MIT. Το συγκεκριμένο framework χρησιμοποιείται για διευκόλυνση τις συγγραφείς των λειτουργιών που αφορούν την NodeJS[11,12].

Barcode

Τα barcode [13] (Σχήμα 5.1) είναι μια αναπαράσταση των δεδομένων που αφορούν το αντικείμενο στο οποίο βρίσκεται προσκολλημένο το συγκεκριμένο barcode. Η αναπαράσταση αυτή είναι εύκολο να διαβαστεί από μηχανές με οπτική ικανότητα, συνήθως με την χρήση ακτινών laser. Η αναπαράσταση των δεδομένων γίνεται με την χρήση παράλληλων κάθετων γραμμών, όπου η κάθε γραμμή έχει διαφορετικό πλάτος, ανάλογα με τον αριθμό που αναπαριστά. Τα barcodes στις μέρες μας είναι ευρέως διαδεδομένα και χρησιμοποιήσιμα για τον λόγο ότι καθιστούν εύκολο τον έλεγχο των διάφορων προϊόντων που μπορεί να υπάρχουν σε ένα κατάστημα, καθώς επίσης και την καταγραφή του αποθέματος στο συγκεκριμένο κατάστημα ή ακόμη και σε μία αλυσίδα καταστημάτων.

Σχήμα 5.1 : Barcode

ISBN

Το ISBN [14] (Σχήμα) το οποίο αναφέρεται ως Διεθνείς Πρότυπο Αριθμού Βιβλίου (International Standard Book Number) είναι μία αριθμητική ταυτότητα αναγνώρισης βιβλίων. Ένα αναγνωριστικό ISBN ανατίθεται σε κάθε έκδοση και παραλλαγή ενός βιβλίου. Είναι ένας αριθμός ο οποίος έχει 13 ψηφία σε περίπτωση που έχει ανατεθεί στο βιβλίο μετά την 1η Ιανουαρίου 2007 και 10 ψηφία σε περίπτωση που ανατέθηκε πριν το 2007. Με την χρήση λοιπόν της προαναφερθείσας αναπαράστασης δεδομένων barcode, όπου κάθε αριθμός

[11] http://en.wikipedia.org/wiki/Express.js
[12] http://expressjs.com/
[13] http://en.wikipedia.org/wiki/Barcode
[14] http://en.wikipedia.org/wiki/International_Standard_Book_Number

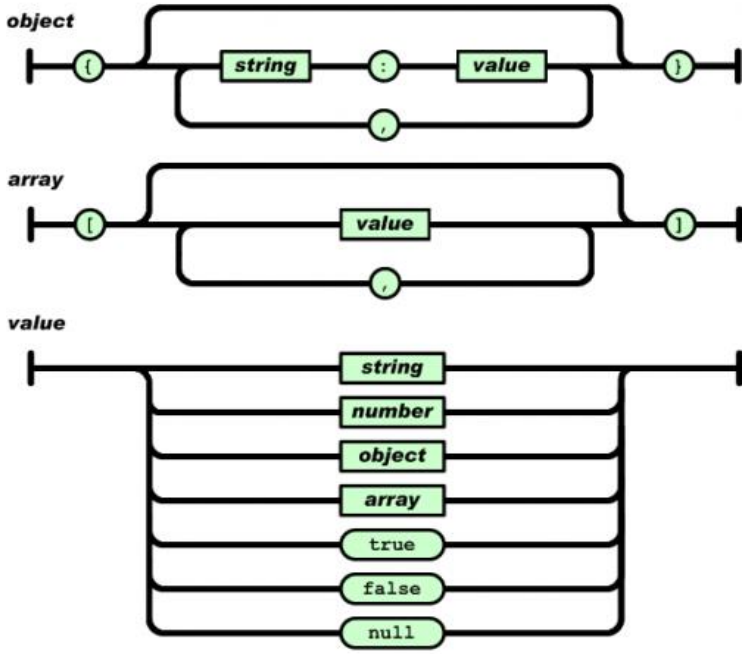
αναπαριστάτε με κάθετες γραμμές διαφορετικού πλάτους, δημιουργείται ένα barcode με βάση τον αναγνωριστικό αυτό αριθμό ISBN, το οποίο εκτυπώνεται συνήθως στο οπισθόφυλλο του κάθε βιβλίου.

Σχήμα 5.2 : International Standard Book Number

JSON

JSON[15,16,17] ή αλλιώς JavaScript Object Notation είναι ένα ανοικτό πρότυπο αναπαράστασης δεδομένων (Σχήμα 5.3,Σχήμα 5.4) η οποίο χρησιμοποιεί μια μορφή κειμένου κατανοητή από τον άνθρωπο, για την μετάδοση δεδομένων. Χρησιμοποιείται συνήθως για την μεταφορά δεδομένων από κάποιο διακομιστή σε κάποιο χρήστη ή και διαδικτυακή εφαρμογή. Η διάταξη του μπορεί να είναι είτε αντικείμενο, είτε πίνακας. Κα στις δύο περιπτώσεις, το περιεχόμενο αποτελείται από ζευγάρια ονομασίας/τιμής, οι οποίες διαχωρίζονται με άνω και κάτω τελεία (:). Η ύπαρξη περισσότερων από ένα ζευγών, διαχωρίζεται με κόμμα (,). Στη περίπτωση του αντικειμένου , οι τιμές περικλείονται σε αριστερό ({) και δεξιό άγκιστρο (}), ενώ στην περίπτωση του πίνακα, οι τιμές περικλείονται σε αριστερή τετράγωνη αγκύλη ([) και δεξιά τετράγωνη αγκύλη (]).

[15] http://en.wikipedia.org/wiki/JSON
[16] http://www.json.org/
[17] http://www.w3schools.com/json/



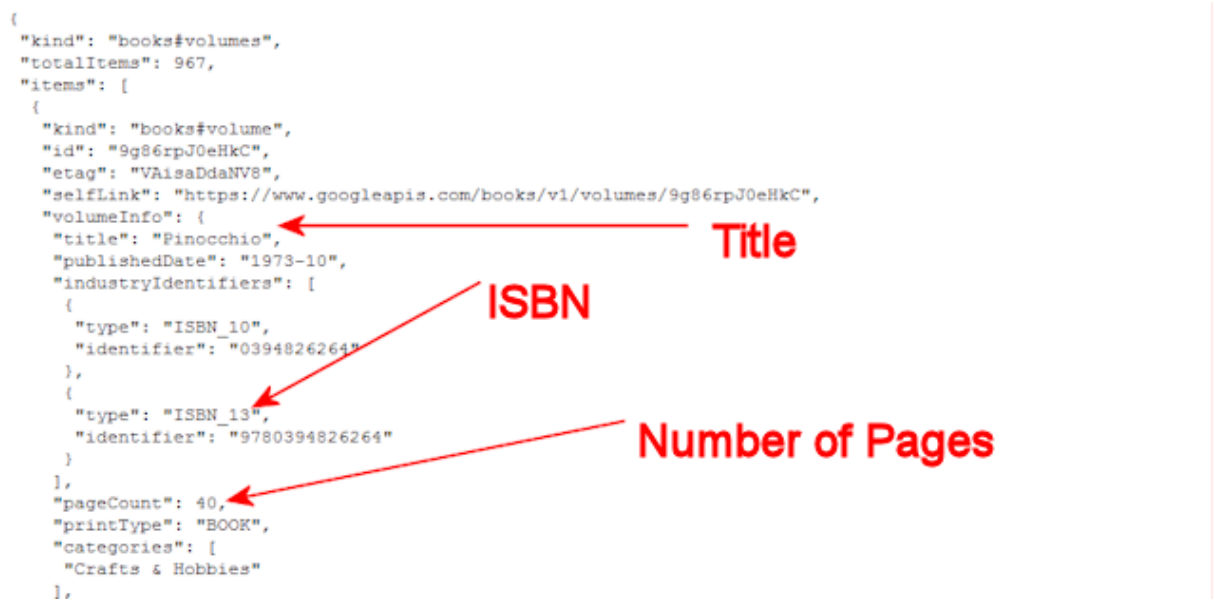
Σχήμα 5.3 : Τρόπος Σύνταξης JSON Αντικειμένου

```
{
  "arguments" : { "number" : 10 },
  "url" : "http://localhost:8080/restty-tester/collection",
  "method" : "POST",
  "header" : {
    "Content-Type" : "application/json"
  },
  "body" : [
    {
      "id" : 0,
      "name" : "name 0",
      "description" : "description 0"
    },
    {
      "id" : 1,
      "name" : "name 1",
      "description" : "description 1"
    }
  ],
  "output" : "json"
}
```

Σχήμα 5.4 : Παράδειγμα JSON αντικειμένου

Google Books API

Το Google Books API αποτελεί μια προσπάθεια της Google να καταστήσει το περιεχόμενο της σε βιβλία πιο εύκολο στην ανεύρεση. Με την χρήση του Google Books API, η εφαρμογή μπορεί να αναζητήσει περιεχόμενο με την χρήση πλήρους κειμένου και να ανακτήσει πληροφορίες για το βιβλίο (Σχήμα 5.5) για το οποίο έχει γίνει αναζήτηση, καθώς επίσης και την διαθεσιμότητα σου σε ηλεκτρονική μορφή. Για την χρήση του συγκεκριμένου API αρκεί μόνο ένα αίτημα GET σε συγκεκριμένη διεύθυνση. Η απάντηση που παίρνει ο χρήστης είναι



Σχήμα 5.5 : Απόσπασμα απάντησης Google Books API

σε μορφή JSON και περιέχει πληροφορίες όπως ο τίτλος του βιβλίου, τους συγγραφείς, την ημερομηνία έκδοσης, μια σύντομη περιγραφή για το βιβλίο, καθώς επίσης και τον αναγνωριστικό αριθμό ISBN, εικόνες εξωφύλλου (συνδέσμους για το εξώφυλλο) και διάφορες άλλες πληροφορίες.

OAuth

Είναι ένα ανοικτό πρότυπο για εξουσιοδότηση χρηστών. Είναι κοινός χρησιμοποιούμενο σε διαδικτυακές εφαρμογές, όπου απαιτείται η αναγνώριση και εξουσιοδότηση κάπου χρήστη. Στο SmartLib+ χρησιμοποιήθηκε για την εξουσιοδότηση στους χρήστης με την χρήση Google λογαριασμών.

Google Maps API

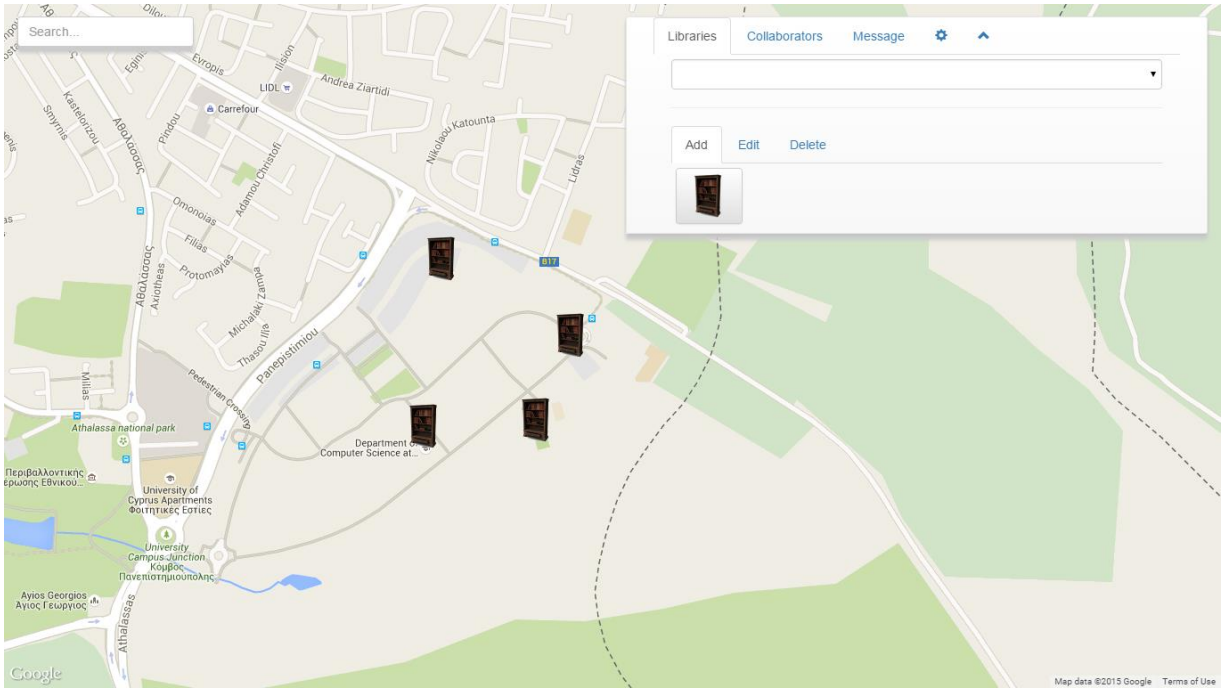
Είναι μια υπηρεσία χαρτογράφησης που παρέχεται από την Google. Παρέχει δορυφορική, εικονογραφημένη και προοπτική λειτουργία χαρτών. Επίσης υπάρχει η δυνατότητα σχεδίασης δρομολογίων στους χάρτες. Τέλος, παρέχεται ένα API μέσω του οποίου μπορούν οι υπηρεσίες αυτές να χρησιμοποιηθούν από προγραμματιστές για την ανάπτυξη διάφορων εφαρμογών.

5.2 Υλοποίηση Διαδικτυακής Εφαρμογής

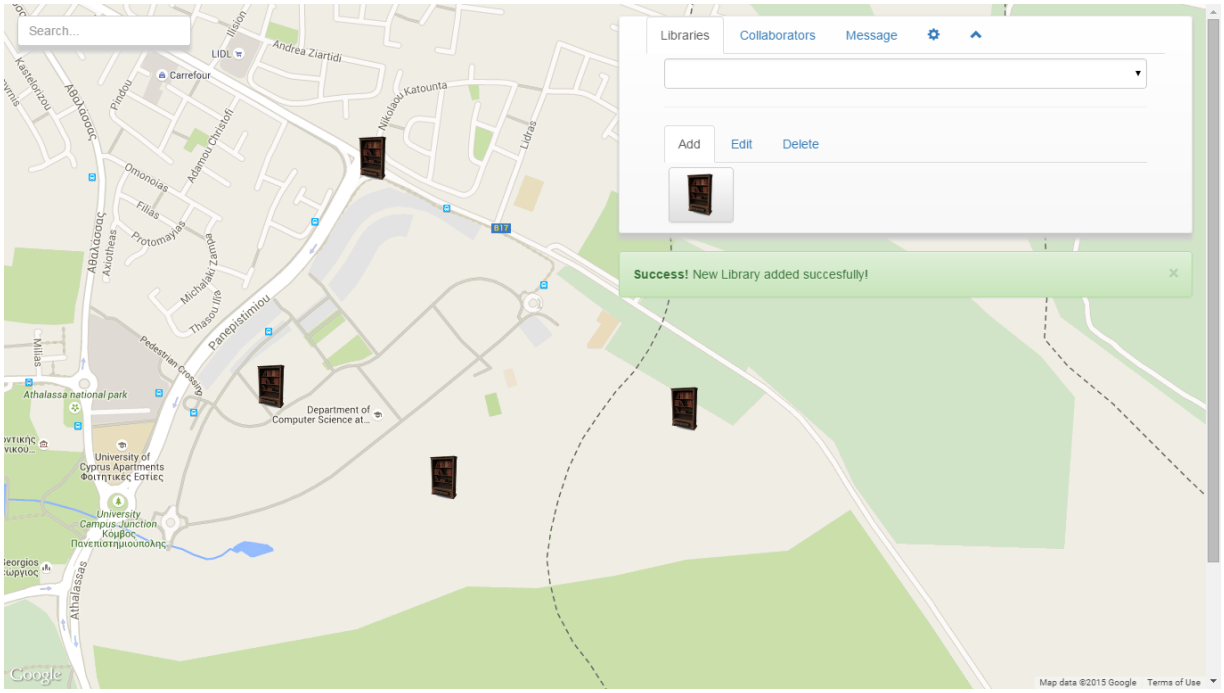
Η διαδικτυακή εφαρμογή έχει υλοποιηθεί για την καταγραφή των έντυπων βιβλίων που κατέχει ένας χρήστης, και η ταξινόμηση τους σε βιβλιοθήκες.

Αρχικά ο χρήστης πρέπει να εισέλθει στο σύστημα. Για την είσοδο του είναι απαραίτητη η χρήση λογαριασμού Google. Αν ο χρήστης δεν κατέχει λογαριασμό, είναι εύκολο να αποκτήσει σε λίγα μόλις λεπτά με το να επισκεφτεί την ιστοσελίδα της Google. Με την εισαγωγή των συνθηματικών του στην αρχική σελίδα, γίνεται επιβεβαίωση από την Google ότι τα εισαχθέντα συνθηματικά είναι ορθά. Αν επιστραφεί θετική απάντηση, τότε επιτρέπετε είσοδος του χρήστη στο σύστημα. Κατά την είσοδο του χρήστη, εάν είναι η πρώτη φορά που χρησιμοποίει την εφαρμογή ο συγκεκριμένος χρήστης, καταγράφεται αυτόματα στην βάση δεδομένων του συστήματος.

Με την είσοδο στο σύστημα, παρουσιάζεται στον χρήστη το κυρίως μενού (Σχήμα 5.6) , που αποτελείται από τον χάρτη της εφαρμογής, την φόρμα αναζήτησης και την φόρμα ελέγχου.

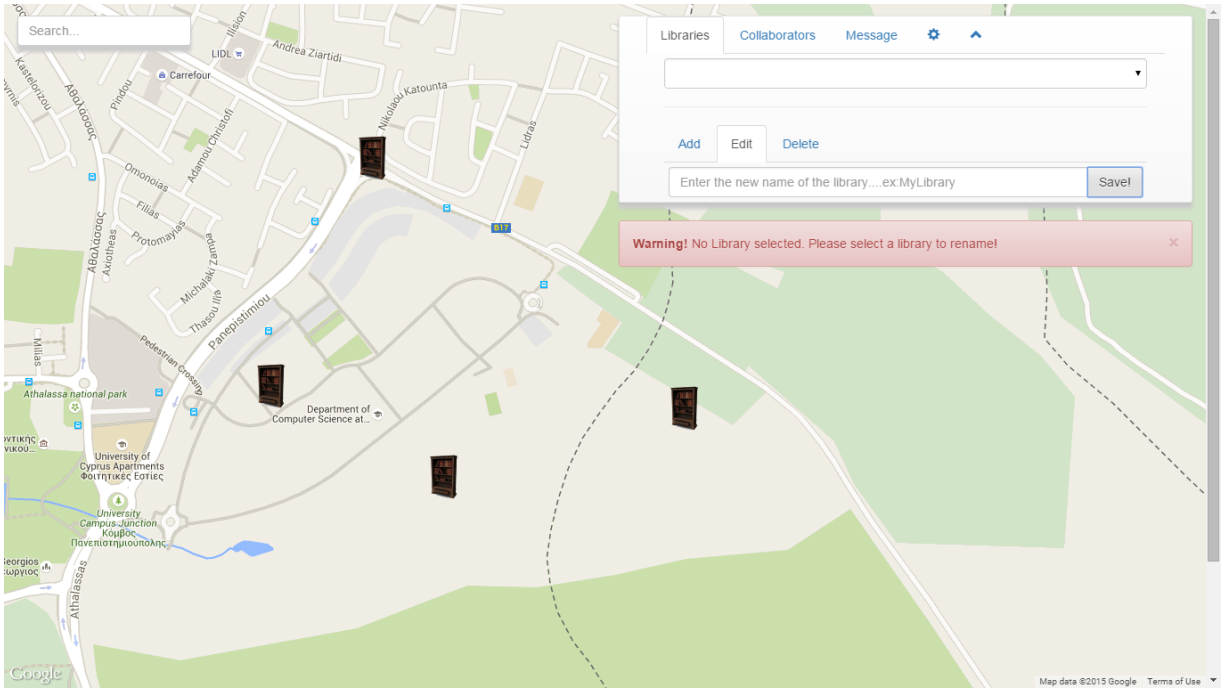


Σχήμα 5.6: Κυρίως Μενού της εφαρμογής SmartLib+



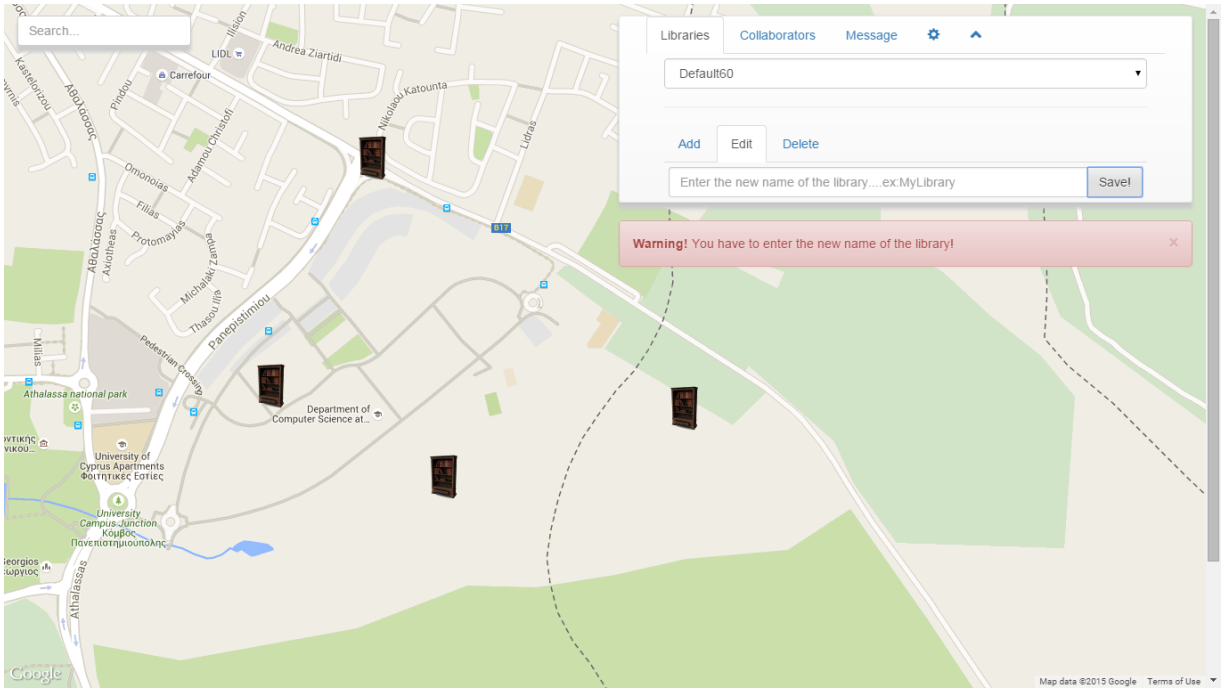
Σχήμα 5.7: Επιτυχής εισαγωγή Βιβλιοθήκης

Από το σημείο αυτό ο χρήστης μπορεί να διενεργήσει οποιανδήποτε ενέργεια επιθυμεί. Καταρχάς, επιτρέπετε στο χρήστη να προσθέσει μια νέα βιβλιοθήκη. Με την προσθήκη της βιβλιοθήκης, εμφανίζεται στον χάρτη ένα νέο σημείο με το σχήμα βιβλιοθήκης. καθώς επίσης και ένα μήνυμα με το οποίο ενημερώνεται ο χρήστης για την επιτυχή εισαγωγή της βιβλιοθήκης (Σχήμα 5.7). Παράλληλα, η εφαρμογή επικοινωνεί με το κομμάτι του διακομηστή, ο οποίος αναλαμβάνει την εισαγωγή της νέας βιβλιοθήκης στη βάση δεδομένων και την δημιουργία συσχέτισης της βιβλιοθήκης με τον χρήστη που την έχει εισάγει.

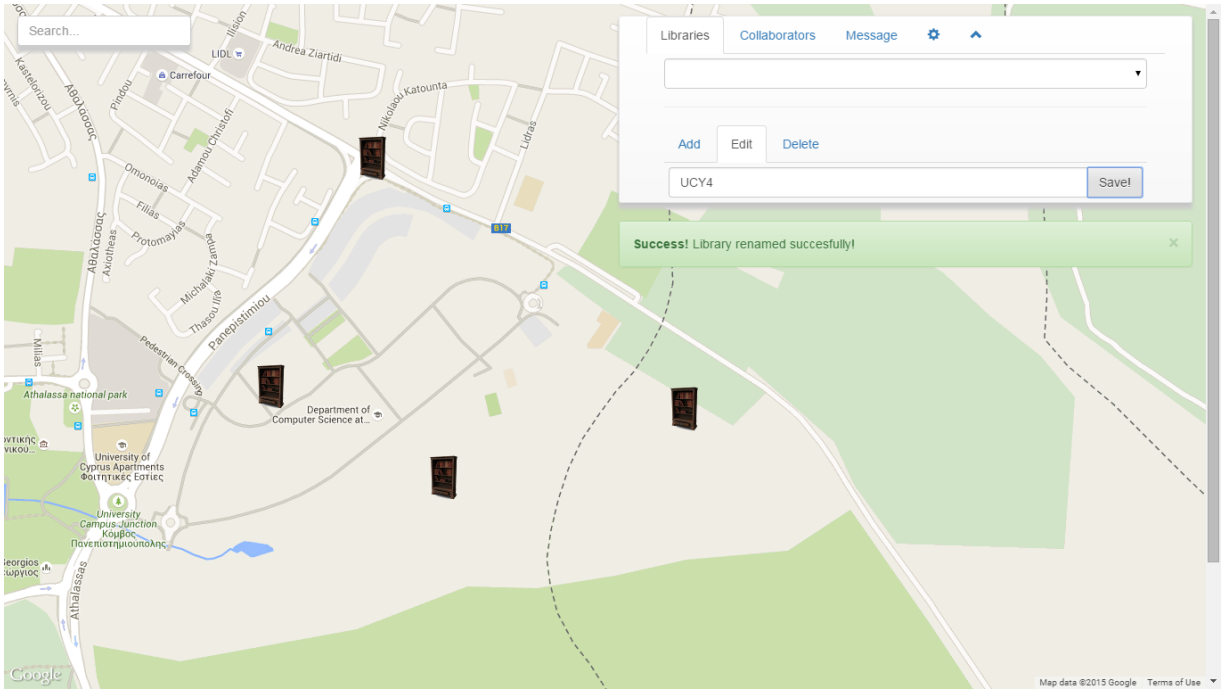


Σχήμα 5.8: Αποτυχία μετονομασίας λόγο μη επιλογής βιβλιοθήκης

Επιπλέον, ο χρήστης έχει την δυνατότητα να μετονομάσει μια από τις βιβλιοθήκες του, αφού μπορεί το όνομα προεπιλογής που έχει ανατεθεί στο σύστημα να μην αρέσει στο χρήστη και να μην είναι περιγραφικό. Έτσι, μπορεί ο χρήστης να επιλέξει την βιβλιοθήκη που θέλει να μετονομάσει από τη λίστα που δημιουργείται αυτόματα. Στη συνέχεια μπορεί να εισάγει την νέα ονομασία στο πεδίο ονομασίας, και να αποθηκεύσει τις αλλαγές. Αν ο χρήστης δεν επιλέξει κάποια βιβλιοθήκη και προσπαθήσει να αποθηκεύσει τις αλλαγές, το σύστημα ενημερώνει το χρήστη πως δεν έχει επιλέξει κάποια βιβλιοθήκη (Σχήμα 5.8). Επιπλέον, αν ο χρήστης επιλέξει βιβλιοθήκη και δεν εισάγει κάποιο όνομα, η εφαρμογή ενημερώνει το χρήστη ότι δεν έχει εισαχθεί η νέα ονομασία (Σχήμα 5.9).

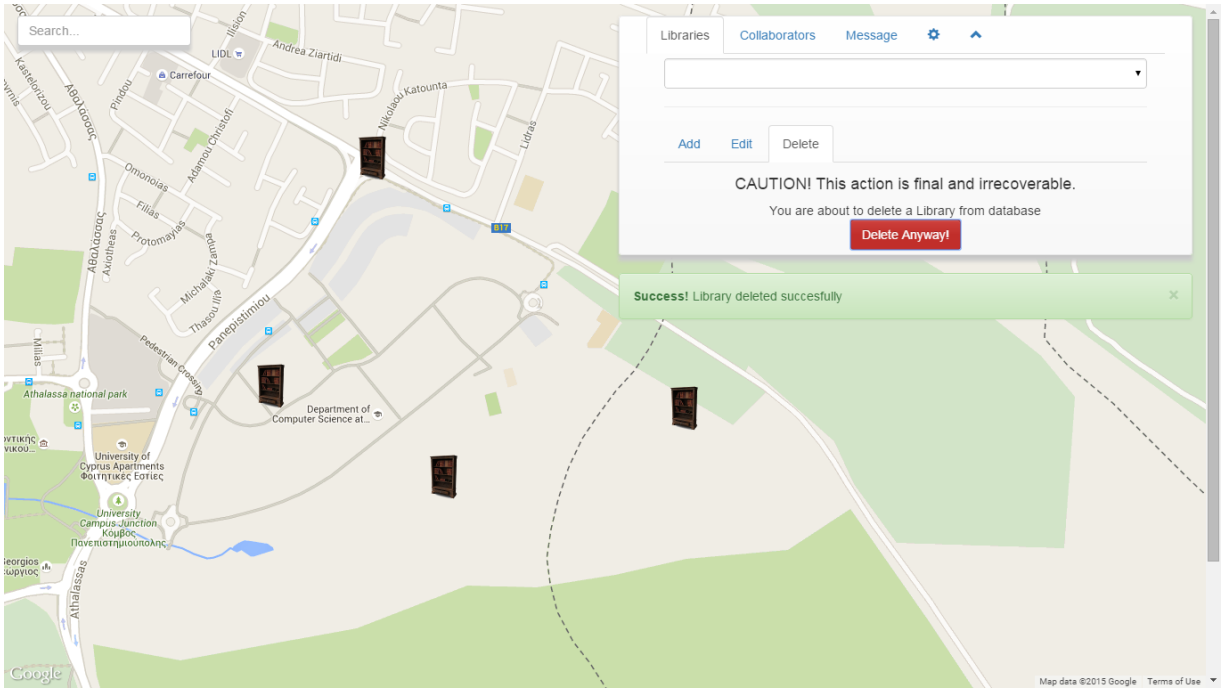


Σχήμα 5.9: Αποτυχία μετονομασίας λόγο μη εισαγωγής νέας ονομασίας.



Σχήμα 5.10: Επιτυχής Μετονομασία Βιβλιοθήκης.

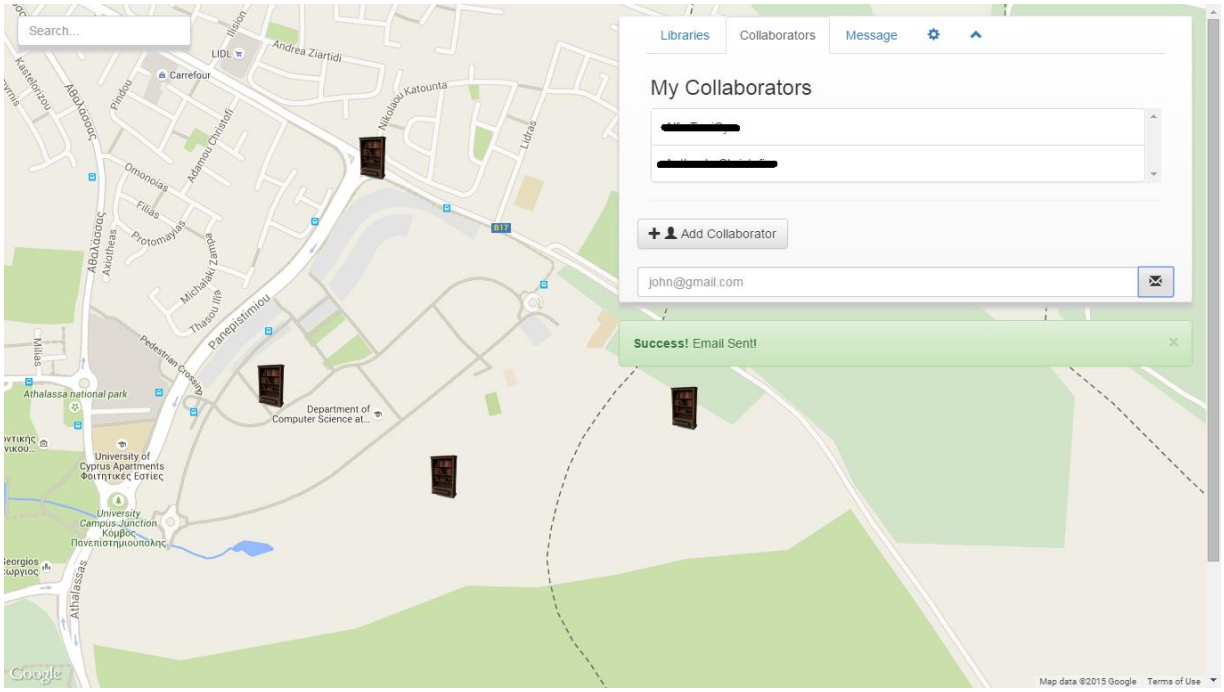
Σε περίπτωση επιτυχής επιλογής βιβλιοθήκης και εισαγωγής ονομασίας βιβλιοθήκης, τότε γίνεται ενημέρωση του εξυπηρετητή, ο οποίος με τη σειρά του ενημερώνει την βάση δεδομένων του συστήματος, αλλάζοντας την ονομασία της επιλεγμένης βιβλιοθήκης. Ταυτόχρονα, εμφανίζεται στο χρήστη μήνυμα επιτυχούς μετονομασίας της βιβλιοθήκης (Σχήμα 5.10).



Σχήμα 5.11: Επιτυχής Διαγραφή Βιβλιοθήκης

Επιπρόσθετα, μπορεί ο χρήστης σε κάποια φάση να απολέσει την ιδιοκτησία της βιβλιοθήκης, είτε λόγω φυσικής καταστροφής, είτε για διάφορους άλλους λόγους, και να επιθυμεί να διαγράψει την συγκεκριμένη βιβλιοθήκη. Μπορεί να επιλέξει την καρτέλα διαγραφής, η οποία θα του εμφανίσει το κουμπί διαγραφής. Η διαδικασία διαγραφής είναι απλή. Πρώτα πρέπει να επιλεγεί η βιβλιοθήκη που επιθυμεί ο χρήστης να διαγράψει και στη συνέχεια να πατηθεί το κουμπί διαγραφής. Κοντά στο κουμπί διαγραφής εμφανίζεται ένα μήνυμα το οποίο ενημερώνει το χρήστη ότι η ενέργεια στην οποία πρόκειται να προβεί, είναι τελική και μη αναστρέψιμη. Με το πάτημα του κουμπιού, ενημερώνεται ο εξυπηρετητής, ο οποίος και διενεργεί τις κατάλληλες ενέργειες για διαγραφή της βιβλιοθήκης από την βάση δεδομένων του συστήματος. Όταν γίνει η επιτυχής διαγραφή της βιβλιοθήκης από το σύστημα, εμφανίζεται στο χρήστη μήνυμα επιτυχούς διαγραφής (Σχήμα 5.11).

Κατά την χρήση του συστήματος, ο χρήστης μπορεί να επιθυμεί να συνεργαστεί με άλλα άτομα, όπου και θα ανταλλάξουν βιβλία, θα δανείσουν η θα δανειστούν κάποια βιβλία. Στόχος της εφαρμογής SmartLib+ είναι να παρέχει την δυνατότητα στους χρήστες, να δημιουργούν σχέσεις συνεργασίας μεταξύ τους,. Για την δημιουργία ενός συνδέσμου είναι απαραίτητο να αποσταλεί ένα αίτημα στο οποίο καλείται το άμεσα εμπλεκόμενο άτομο να απαντήσει. Με την λήψη της θετικής απάντησης του, ο εξυπηρετητής αναλαμβάνει την δημιουργία συσχέτισης των δύο μελών στη βάση δεδομένων του συστήματος. Για την αποστολή του αιτήματος συνεργασίας, αρκεί να επιλεγεί η καρτέλα συνεργατών, όπου εμφανίζεται μια λίστα με τους συνεργάτες του χρήστη και ένα πεδίο εισόδου ηλεκτρονικής

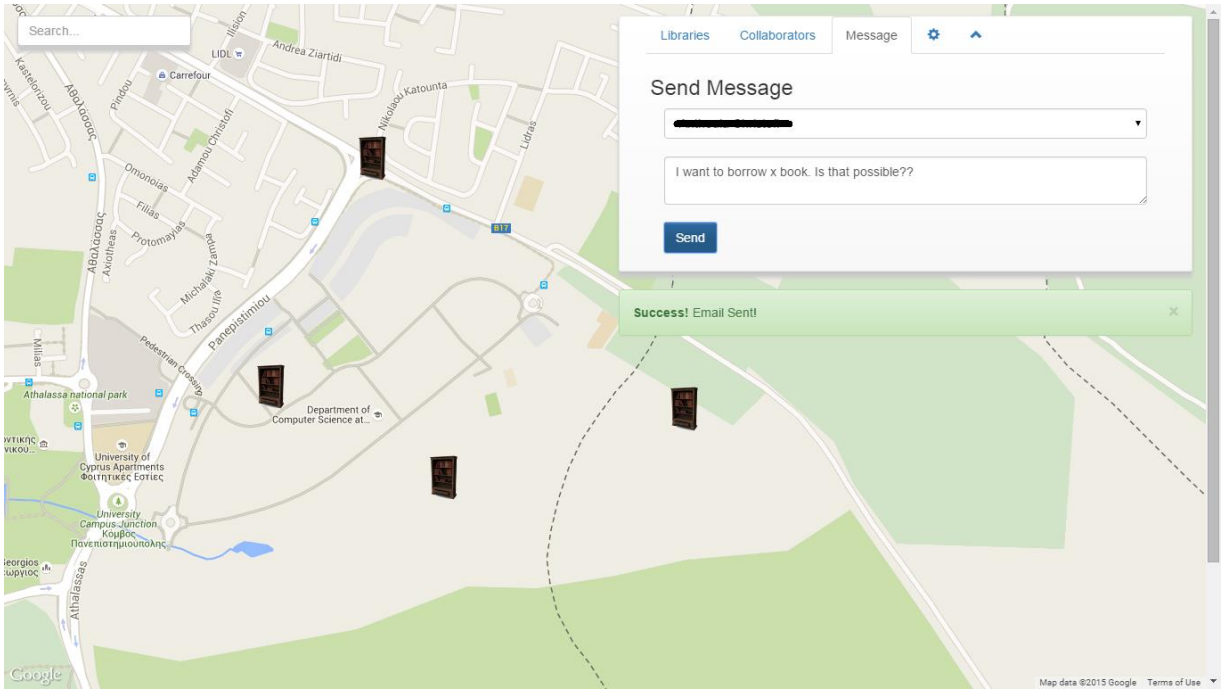


Σχήμα 5.12: Επιτυχής αποστολή αιτήματος συνεργασίας.

διεύθυνσης, όπου εισάγεται η ηλεκτρονική διεύθυνση του ατόμου με το οποίο επιθυμεί ο χρήστης να συνεργαστεί. Με το πάτημα του κουμπιού αποστολής, ο εξυπηρετητής αναλαμβάνει να αποστείλει ηλεκτρονικό μήνυμα στον παραλήπτη. Με την επιτυχή αποστολή του μηνύματος, ενημερώνεται ο χρήστης με το κατάλληλο μήνυμα (Σχήμα 5.12).

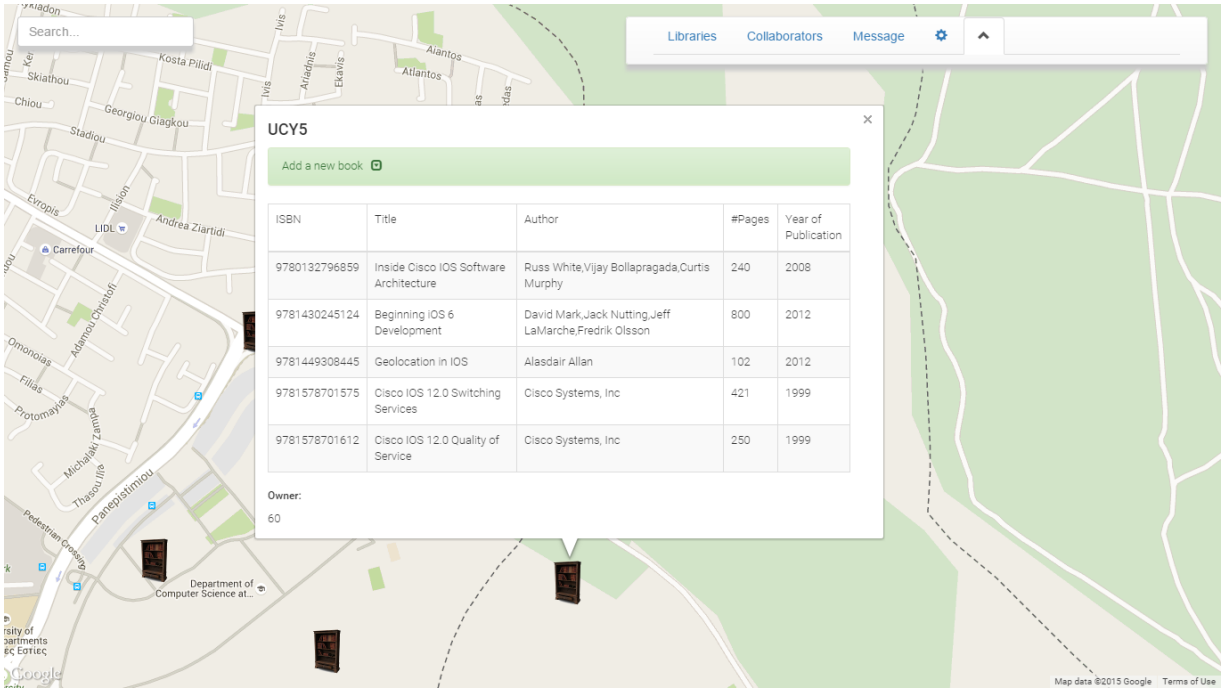
Παράλληλα, ο χρήστης μπορεί να επιθυμεί να δανειστεί κάποιο από τα βιβλία που ανήκει σε κάποιο από τους συνεργάτες του. Η εφαρμογή παρέχει την δυνατότητα αποστολής μηνύματος από το χρήστη προς τον ιδιοκτήτη του βιβλίου, και έτσι να έρθουν σε επαφή τα δύο μέρη για να συμφωνήσουν σε δανεισμό ή όχι του βιβλίου. Με τον τρόπο αυτό ο χρήστης μπορεί να δανειστεί κάποιο βιβλίο από κάποιο συνεργάτη του αντί να το αγοράσει ή να το ψάχνει σε δημόσιες βιβλιοθήκες. Έτσι καταλήγουμε σε εξοικονόμηση και χρόνου και χρημάτων. Το μόνο που χρειάζεται να κάνει ο χρήστη είναι να επιλέξει την καρτέλα μηνυμάτων, να επιλέξει τον συνεργάτη που επιθυμεί και να συμπληρώσει το πεδίο μηνύματος με το μήνυμα που επιθυμεί να αποστείλει. Με την αποστολή του μηνύματος, εμφανίζεται στο χρήστη ειδοποίηση για επιτυχή αποστολή του (Σχήμα 5.13).

Για την εισαγωγή ενός βιβλίου στη βιβλιοθήκη του ο χρήστης αρκεί να επιλέξει την βιβλιοθήκη στην οποία επιθυμεί να εισάγει το βιβλίο, μέσω του χάρτη. Με την επιλογή του σημείου βιβλιοθήκης, εμφανίζεται ένα παράθυρο πάνω από την βιβλιοθήκη, με μια λίστα που περιλαμβάνει όλα το βιβλία της (Σχήμα 5.14), και μια φόρμα για προσθήκη νέου βιβλίου.

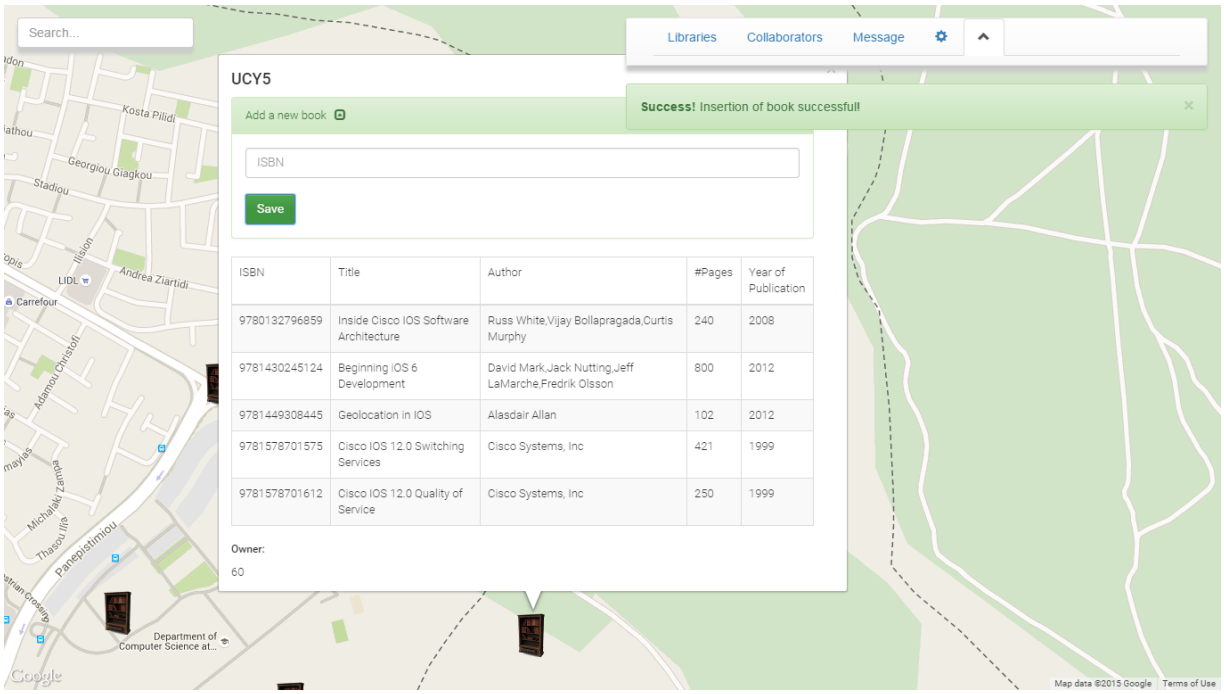


Σχήμα 5.13: Επιτυχής αποστολή μηνύματος

Στη φόρμα εισαγωγής ο χρήστης καλείται να εισάγει τον δεκατριψήφιο μοναδικό αριθμό ISBN ο οποίος βρίσκεται συνήθως στο πίσω μέρος της έντυπης μορφής του βιβλίου μαζί με το αναγνωριστικό barcode. Με την εισαγωγή του ISBN, ενημερώνεται ο εξυπηρετητής, ο οποίος εκτελεί αίτημα στο Google Books API, αποστέλλοντας τον αναγνωριστικό αριθμό. Στη συνέχεια λαμβάνεται η απάντηση η οποία περιέχει τις πληροφορίες για το βιβλίο. Σε περίπτωση αποτυχημένου αιτήματος, η απάντηση που λαμβάνεται περιέχει πληροφορίες που περιγράφουν το σφάλμα. Σε περίπτωση επιτυχούς απάντησης, ο εξυπηρετητής, αναλύει την απάντηση και καταχωρεί το νέο βιβλίο στη βάση δεδομένων. Ταυτόχρονα, ενημερώνεται ο χρήστης για την επιτυχημένη εισαγωγή του βιβλίου (Σχήμα 5.15), και ενημερώνεται η βιβλιοθήκη στο χάρτη.



Σχήμα 5.14: Βιβλιοθήκη με τα βιβλία της



Σχήμα 5.15: Επιτυχής Εισαγωγή Νέου Βιβλίου

Κεφάλαιο 6

Πειραματική Διαδικασία

6.1 Πειραματική Υποδομή 46
6.2 Πειραματική Διαδικασία και Μετρήσεις 46

6.1 Πειραματική Υποδομή

Στο κεφάλαιο αυτό θα παρουσιαστεί και θα αναλυθεί, η διαδικασία την οποία ακολουθήσαμε για να εξετάσουμε την εγκυρότητα και αποδοτικότητα της εφαρμογής SmartLib+. Κύριος στόχος μας ήταν να δούμε τον χρόνο απόκριση της εφαρμογής με σε ένα εύρους αριθμών βιβλιοθηκών, καθώς επίσης και τα ποσοστά αποτυχίας εύρεσης των δεδομένων για τα βιβλία από το Google Books API.

Για να μπορέσουμε να έχουμε τις απαραίτητες μετρήσεις χρειάστηκε να συμβάλουν διάφοροι συνάδελφοι, οι οποίοι προσφέρθηκαν να δοκιμάσουν και να χρησιμοποιήσουν το SmartLib+. Και στο σημείο αυτό φαίνεται η σημαντικότητα του crowdsourcing. Με την συμμετοχή των εθελοντών συναδέλφων έχει δημιουργηθεί ένα σύνολο δεδομένων το οποίο είναι αξιοσέβαστο.

Οι διάφοροι εθελοντές δημιούργησαν βιβλιοθήκες και εισήγαγαν σε αυτές τα διάφορα βιβλία τους τα οποία έχουν σε έντυπη μορφή, ενώ ταυτόχρονα, καταγράφοντα στοιχεία όπως ο αριθμός των εισαχθέντων στοιχείων, ο αριθμός τον επιτυχών εισαγωγών καθώς επίσης και ο αριθμός των αποτυχημένων εισαγωγών. Στη συνέχεια θα παρουσιαστούν τα δεδομένα αυτά σε γραφικές παραστάσεις όπου και θα γίνει λεπτομερής περιγραφή για το καθένα ξεχωριστά.

6.2 Πειραματική Διαδικασία και Μετρήσεις

Μετρήσεις από την εισαγωγή βιβλιοθηκών

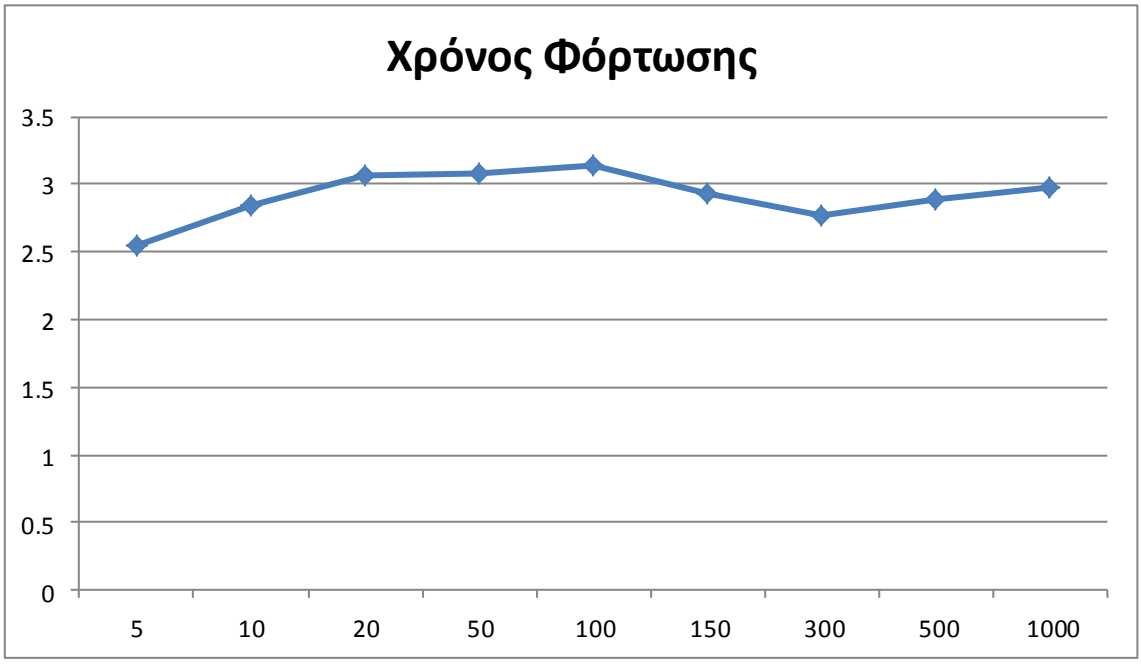
Κύριος στόχος κατά την εισαγωγή βιβλιοθηκών ήταν να ελέγξουμε την ταχύτητα και τον χρόνο απόκρισης της εφαρμογής με την ύπαρξη διαφορετικών αριθμών βιβλιοθηκών. Ο

λόγος που μας οδήγησε σε αυτές τις μετρήσεις ήταν το γεγονός ότι με την πάροδο του χρόνου, και λόγο της χρήσης της εφαρμογής του crowdsourcing, ο αριθμός των βιβλιοθηκών θα είναι πολύ μεγάλος. Έτσι στόχος μας είναι η όσο ταχύτερη φόρτωση των δεδομένων στο χάρτη, για δημιουργία καλύτερης εμπειρίας στο χρήστη.

Πιο κάτω παρουσιάζεται ένας πίνακας με τους χρόνους απόκρισης - φόρτωσης των βιβλιοθηκών στο χάρτη της εφαρμογής SmartLib+.

Αριθμός Βιβλιοθηκών	Χρόνος Φόρτωσης στο Χάρτη (s)
5	2.55
10	2.85
20	3.06
50	3.08
100	3.14
150	2.93
300	2.77
500	2.89
1000	2.97

Πιο κάτω παρουσιάζεται γραφική αναπαράσταση των χρόνων απόκρισης και φόρτωσης των βιβλιοθηκών στο χάρτη της εφαρμογής.



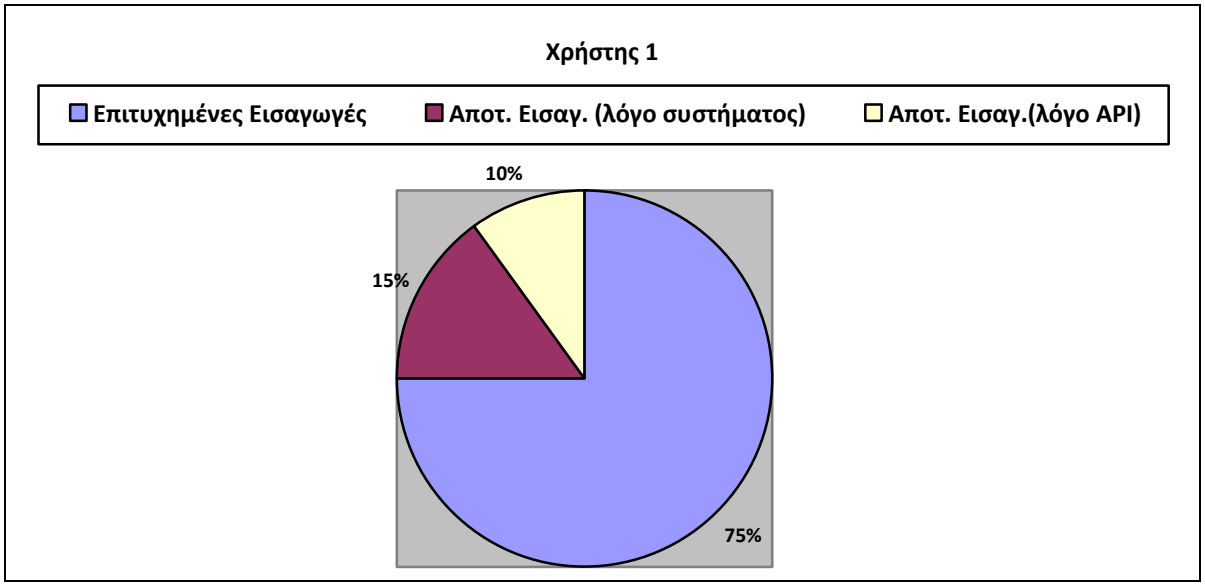
Στον άξονα x καταγράφεται ο αριθμός των βιβλιοθηκών που χρησιμοποιήθηκαν κατά την πειραματική φάση του φορτώματος της εφαρμογής. Στον άξονα y φαίνονται οι χρόνοι που χρειάστηκαν για την φόρτωση του κάθε αριθμού βιβλιοθηκών ξεχωριστά. Παρατηρείται ότι υπάρχει μια αυξομείωση στους χρόνους με μέσο όρο τα ~3s. Παρατηρούμε επίσης ότι στο πείραμα χρησιμοποιήθηκε εύρος τιμών για τον αριθμό των βιβλιοθηκών για καλύτερα αποτελέσματα.

Μετρήσεις από την εισαγωγή βιβλίων

Στόχος των πειραματικών μετρήσεων αυτών ήταν η καταγραφή των επιτυχημένων και αποτυχημένων προσπαθειών για εισαγωγή βιβλίων. Κατά την διαδικασία αυτή καταγράφηκαν ο αριθμός των βιβλίων που εισήχθησαν με επιτυχία, ο αριθμός των βιβλίων των οποίων η εισαγωγή απότυχε λόγο προβλήματος του συστήματος και ο αριθμός των βιβλίων που απέτυχε η εισαγωγή λόγο μη εύρεσης δεδομένων μέσω του Google Books API.

Χρήστης 1

Επιτυχημένες Εισαγωγές	15
Αποτυχημένες Εισαγωγές (λόγο συστήματος)	3
Αποτυχημένες Εισαγωγές (λόγο API)	2
Σύνολο	20

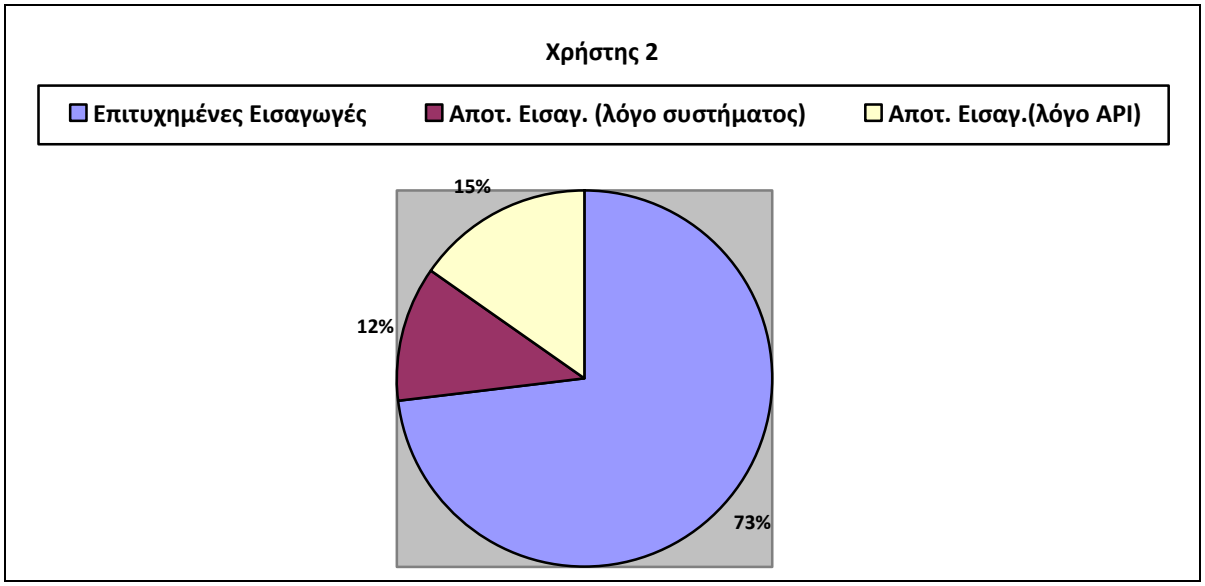


Γραφική Παράσταση 6.1

Σύμφωνα με τον χρήστη 1, ο χρόνος που χρειάστηκε για την δημιουργία βιβλιοθηκών και την εισαγωγή των βιβλίων ήταν 8 λεπτά ενώ ο μέσος χρόνος για την εισαγωγή του κάθε βιβλίου περίπου 57ms. Το ποσοστό επιτυχίας σύμφωνα με τις μετρήσεις του χρήστη και την γραφική παράσταση (Γραφική Παράσταση 6.1) ήταν 75% ενώ το υπόλοιπο ποσοστό διαμοιράζεται μεταξύ των αποτυχιών λόγο API 10% και λόγο του συστήματος 15%.

Χρήστης 2

Επιτυχημένες Εισαγωγές	19
Αποτυχημένες Εισαγωγές (λόγο συστήματος)	3
Αποτυχημένες Εισαγωγές (λόγο API)	4
Σύνολο	26

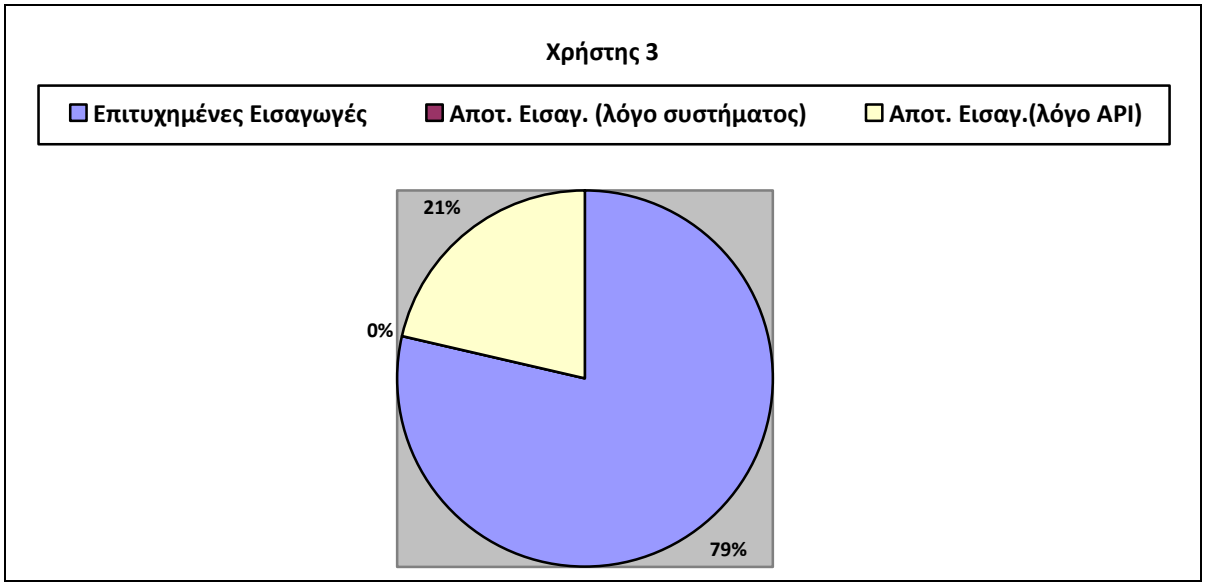


Γραφική Παράσταση 6.2

Ο χρήστης 2 έχει αναφέρει ότι ο χρόνος που χρειάστηκε για την δημιουργία βιβλιοθηκών και την εισαγωγή των βιβλίων ήταν 10 λεπτά ενώ ο μέσος χρόνος για την εισαγωγή του κάθε βιβλίου περίπου 62ms. Το ποσοστό επιτυχίας σύμφωνα με τις μετρήσεις του χρήστη και την γραφική παράσταση (Γραφική Παράσταση 6.2Γραφική Παράσταση 6.1) ήταν 73% ενώ το υπόλοιπο ποσοστό διαμοιράζεται μεταξύ των αποτυχιών λόγο API 15% και λόγο του συστήματος 12%.

Χρήστης 3

Επιτυχημένες Εισαγωγές	11
Αποτυχημένες Εισαγωγές (λόγο συστήματος)	0
Αποτυχημένες Εισαγωγές (λόγο API)	3
Σύνολο	14

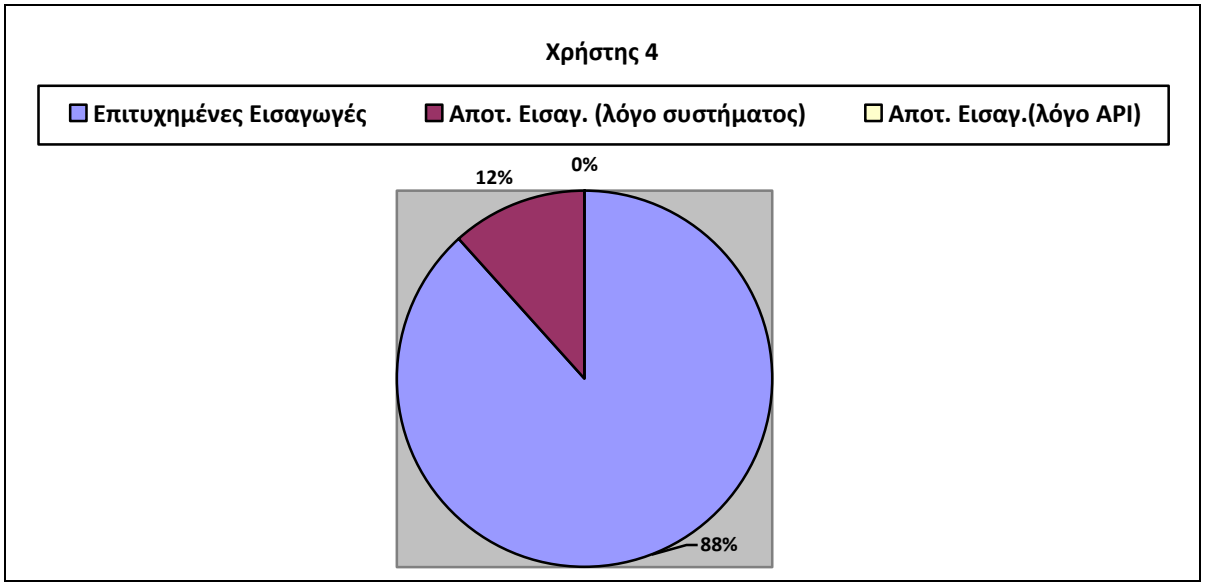


Γραφική Παράσταση 6.3

Για τον χρήστη 3, ο χρόνος που χρειάστηκε για την δημιουργία βιβλιοθηκών και την εισαγωγή των βιβλίων ήταν 8 λεπτά ενώ ο μέσος χρόνος για την εισαγωγή του κάθε βιβλίου περίπου 54ms. Το ποσοστό επιτυχίας σύμφωνα με τις μετρήσεις του χρήστη και την γραφική παράσταση (Γραφική Παράσταση 6.3) ήταν 79% ενώ το υπόλοιπο ποσοστό διαμοιράζεται μεταξύ των αποτυχιών λόγο API 21% και λόγο του συστήματος 0%.

Χρήστης 4

Επιτυχημένες Εισαγωγές	15
Αποτυχημένες Εισαγωγές (λόγο συστήματος)	2
Αποτυχημένες Εισαγωγές (λόγο API)	0
Σύνολο	17



Γραφική Παράσταση 6.4

Για τον χρήστη 3, ο χρόνος που χρειάστηκε για την δημιουργία βιβλιοθηκών και την εισαγωγή των βιβλίων ήταν 6 λεπτά ενώ ο μέσος χρόνος για την εισαγωγή του κάθε βιβλίου περίπου 50ms. Το ποσοστό επιτυχίας σύμφωνα με τις μετρήσεις του χρήστη και την γραφική παράσταση (Γραφική Παράσταση 6.3) ήταν 88% ενώ το υπόλοιπο ποσοστό διαμοιράζεται μεταξύ των αποτυχιών λόγο API 0% και λόγο του συστήματος 12%.

Σύνοψη

Συνοψίζοντας, καταλήξαμε να έχουμε μέσο χρόνο χρήσης του συστήματος για εισαγωγή βιβλίων τα 8 λεπτά. Ταυτόχρονα ο μέσος χρόνος που χρειάζεται για την αλληλεπίδραση με το Google Books API για την λήψη δεδομένων για το κάθε βιβλίο και η διαχείριση της απάντησης για την εισαγωγή του βιβλίου στο σύστημα ήταν περίπου 56 ms.

Όσο αφορά της εισαγωγές στο σύστημα είχαμε σύνολο 77 προσπάθειες εισαγωγής βιβλίων από 4 διαφορετικούς χρήστες. Στο σύνολο των προσπαθειών αυτών, τα 60 βιβλία εισήχθησαν επιτυχώς στο σύστημα.

Το σύνολο των αποτυχημένων προσπαθειών αποτελείται από 17 προσπάθειες εκ των οποίων 9 απέτυχαν για λόγους API, συνήθως μη εύρεσης δεδομένων για τα συγκεκριμένα βιβλία. Οι υπόλοιπες 8 προσπάθειες απότυχαν λόγο του συστήματος για το λόγο ότι το σύστημα έχει σχεδιαστεί να χειρίζεται μόνο τα 13-ψήφια ISBN. Λόγο ύπαρξης 10-ψήφιων ISBN, εμφανίστηκαν αποτυχημένες προσπάθειες. Η μη διαχείριση των 10-ψήφιων ISBN είναι ο λόγος ότι τείνουν να καταργηθούν.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές Επεκτάσεις

Ο Παγκόσμιος ιστός στις μέρες μας έχει εξελιχθεί δραματικά και αποτελεί αναπόσπαστο κομμάτι στην ανθρώπινη ζωή. Παρέχει μεγάλο όγκο πληροφοριών και δεδομένων στους χρήστες. Έχει καταφέρει με ένα έξυπνο τρόπο, μέσω του crowdsourcing, έχει πετύχει την ανάπτυξη μιας πληθώρας εφαρμογών. Οι εφαρμογές αυτές έχουν ως στόχο να προσπαθούν να συλλέξουν και το παραμικρό δεδομένο για την δημιουργία ενός χώρου όπου θα έχει συλλεχτεί όλη η γνώση που κατέχεται από τον άνθρωπο.

Μέσω του SmartLib+, παρέχεται η δυνατότητα να καταγράφονται όλα τα φυσικά βιβλία για να δημιουργηθεί μια τεράστια νοητή βιβλιοθήκη. Στην βιβλιοθήκη αυτή μπορούν να υπάρχουν όλα τα βιβλία διαθέσιμα στους ενδιαφερόμενους χρήστες. Με τον τρόπο αυτό, δια μειωθεί ο χρόνος αναζήτησης και θα αυξηθεί η αποδοτικότητα των ανθρώπων, αφού θα έχουν πρόσβαση σε μεγαλύτερο όγκο πληροφοριών και δεδομένων από πριν.

Στόχος μας είναι η επέκταση του συστήματος με την υλοποίηση της εφαρμογής για κινητές συσκευές, μέσω τον οποίων ο χρήστης θα έχει πρόσβαση σε ακριβώς τις ίδιες λειτουργίες. Είναι παγκοσμίως γνωστό ότι οι κινητές συσκευές αποτελούν αναπόσπαστο κομμάτι τις καθημερινής ζωής.

Βιβλιογραφία

[1] Robert A. Cochran, Loris D'Antoni, Benjamin Livshits, David Molnar, and Margus Veanes. 2015. Program Boosting: Program Synthesis via Crowd-Sourcing. In Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '15). ACM, New York, NY, USA, 677-688. DOI=10.1145/2676726.2676973 http://doi.acm.org/10.1145/2676726.2676973

[2] Annika Hinze and David Bainbridge. 2013. Tipple: location-triggered mobile access to a digital library for audio books. In Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries (JCDL '13). ACM, New York, NY, USA, 171-180. DOI=10.1145/2467696.2467724 http://doi.acm.org/10.1145/2467696.2467724

[3] Ann Blandford, Suzette Keith, Iain Connell, and Helen Edwards. 2004. Analytical usability evaluation for digital libraries: a case study. In Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries (JCDL '04). ACM, New York, NY, USA, 27-36. DOI=10.1145/996350.996360 http://doi.acm.org/10.1145/996350.996360

[4] Dean B. Krafft, Aaron Birkland, and Ellen J. Cramer. 2008. Ncore: architecture and implementation of a flexible, collaborative digital library. In Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries (JCDL '08). ACM, New York, NY, USA, 313-322. DOI=10.1145/1378889.1378943 http://doi.acm.org/10.1145/1378889.1378943

[5] Ann Blandford, Hanna Stelmaszewska, and Nick Bryan-Kinns. 2001. Use of multiple digital libraries: a case study. In Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries (JCDL '01). ACM, New York, NY, USA, 179-188. DOI=10.1145/379437.379479 http://doi.acm.org/10.1145/379437.379479

[6] Kim Grohs, Caroline Reed, and Nancy Allen. 2003. Marketing the virtual library. In Building a virtual library, Ardis Hanson and Bruce Lubotsky Levin (Eds.). IGI Publishing, Hershey, PA, USA 133-147.

[7] Patricia Pettijohn and Tina Neville. 2003. Collection development for virtual libraries. In Building a virtual library, Ardis Hanson and Bruce Lubotsky Levin (Eds.). IGI Global, Hershey, PA, USA 20-36.

[8] Γαβριέλα Κουπεπίδου 2012. "Υλοποίηση μίας Νοητής Βιβλιοθήκης Πληθοπορισμού με Έξυπνα Κινητά Τηλέφωνα" Κύπρος, Λευκωσία.

Παράρτημα Α

Index.hrml

```
<!DOCTYPE html>
<html ng-app="smartlib" xmlns="http://www.w3.org/1999/html">
<head lang="en">
    <meta charset="UTF-8">
    <title></title>

    <!--Bootstrap Styling CDN-->
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bo
otstrap.min.css">
    <!-- Optional theme -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bo
otstrap-theme.min.css">
    <!-- Latest compiled and minified JavaScript -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jqu
ery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/boot
strap.min.js"></script>

    <!--Personal Stylesheet-->
    <link rel="stylesheet" href="style/style.css">

</head>
<body ng-controller="mapController as mapCtrl">

<!--Search and dashboard-->
<div id="searchBox">
    <div class="col-md-2" style="z-index: 100">
        <input type="text" class="form-control"
id="searchContent" placeholder="Search..." ng-
model="filters.name"
               ng-change="filterLibraries()"/>
    </div>
    <div class="col-md-6 pull-right" style="z-index: 100"
ng-init="setDashboard(0)">
        <!--<span class="pull-right" id="signinButton"><span
class="g-signin" data-callback="signinCallback"-->
        <!--data-clientid="65378226875-
d9kcjd5g0pkuo0vr26rrpehe7phg0q1c.apps.googleusercontent.com"
-->
        <!--data-cookiepolicy="single_host_origin"-->
        <!--data-scope="profile"></span></span>-->
        <google-plus-signin id="google-plus-signin-button"
class="pull-right"
                            clientid="65378226875-
d9kcjd5g0pkuo0vr26rrpehe7phg0q1c.apps.googleusercontent.com"
                            ng-
show="isDashboard(0)"></google-plus-signin>
        <div>
            <div id="dashboard" ng-show="isDashboard(1)">
                <!-- Start Here -->
                <nav class="navbar navbar-default"
id="navbarr">
                    <div class="container-fluid">
                        <!-- Brand and toggle get grouped
for better mobile display -->
                        <div class="navbar-header">
                            <button type="button"
class="navbar-toggle collapsed" data-toggle="collapse"
                                    data-target="#bs-
example-navbar-collapse-1" id="buttonCollapse">
                                <span class="sr-only">Toggle
navigation</span>
                                <span class="icon-
```

```
bar"></span>
                                <span class="icon-
bar"></span>
                                <span class="icon-
bar"></span>
                            </button>
                        </div>

                        <!-- Collect the nav links, forms,
and other content for toggling -->
                        <div class="collapse navbar-
collapse" id="bs-example-navbar-collapse-1">
                            <ul class="nav nav-tabs">
                                <li class="active"><a data-
toggle="tab" href="#sectionA">Libraries</a></li>
                                <li><a data-toggle="tab"
href="#sectionB">Collaborators</a></li>
                                <li><a data-toggle="tab"
href="#sectionC">Message</a></li>
                                <li><a data-toggle="tab"
href="#sectionD"><span class="glyphicon glyphicon-
cog"></span></a>
                                </li>
                                <li><a data-toggle="tab"
href="#sectionE"><span
                                        class="glyphicon
glyphicon-chevron-up"></span></a></li>

                            </ul>
                            <div class="tab-content"
style="padding: 5px;">
                                <!--SECTION A-->
                                <div id="sectionA"
class="tab-pane fade in active">
                                    <div class="col-md-12">
                                        <select class="form-
control" id="library_dropdown"
                                                ng-
options="library.name for library in libraries track by
library.library_id"
                                                ng-
model="selected"></select>
                                        <hr>
                                    </div>

                                    <div class="col-md-12">
                                        <ul class="nav nav-
tabs">
                                            <li
class="active"><a data-toggle="tab"
href="#sectionAdd">Add</a></li>
                                            <li><a data-
toggle="tab" href="#sectionEdit">Edit</a></li>
                                            <li><a data-
toggle="tab" href="#sectionDelete">Delete</a></li>
                                        </ul>
                                        <div class="tab-
content" style="padding: 5px;">
                                            <!--Add new
library tab-->
                                            <div
id="sectionAdd" class="tab-pane fade in active">
                                                <div
class="col-md-2 float-left"

style="margin-bottom: 2%;  padding-left: 0px;">
                                                    <!--
Button that adds a new marker-->
                                                    <button
class="btn btn-default" ng-click="addMarker()">
                                                        <img
src="./images/marker100x100.png" id="addMarker"/>

</button>
                                                    <br>
                                                </div>
                                            </div>
                                            <!--Edit
```

```
selected library tab-->
                                        <div
id="sectionEdit" class="tab-pane fade">
                                            <div
class="input-group">
                                                <input
type="text" class="form-control"

placeholder="Enter the new name of the
library....ex:MyLibrary"

ng-model="library_new_name">
                                                <span
class="input-group-btn">

<button class="btn btn-default" type="button"

ng-click =
"renameLibrary(selected.library_id,library_new_name)"

>Save!</button>
                                                </span>
                                            </div>
                                            <!-- /input-
group -->
                                        </div>
                                        <!--Delete
selected library tab-->
                                        <div
id="sectionDelete" class="tab-pane fade">
                                            <div
class="row text-center">

<h4>CAUTION! This action is final and irrecoverable.</h4>You
are
                                                about to
delete a Library from database
                                            </div>
                                            <div
class="row text-center">
                                                <button
class="btn btn-danger"

ng-click="deleteLibrary(selected.library_id)">

Delete

Anyway!

</button>
                                                </div>
                                            </div>
                                        </div>
                                    </div>
                                    <!--SECTION B-->
                                    <div id="sectionB"
class="tab-pane fade">
                                        <h3>My
Collaborators</h3>
                                        <ul class="list-group"
id="friend-list">
                                            <li class="list-
group-item" ng-repeat="friend in friends"

value="{{friend.friend_email}}">{{friend.name}}
{{friend.surname}}
                                            </li>
                                        </ul>
                                        <hr>
                                        <div class="row">
                                            <button
type="button" class="btn btn-default" ng-
click="setAddFriend(1)">
                                                <span
class="glyphicon glyphicon-plus" aria-hidden="true"></span>
                                                <span
class="glyphicon glyphicon-user" aria-hidden="true"></span>
```

```
                                        Add Collaborator
                                    </button>
                                    <div ng-
show="isAddFriend(1)"><br>

                                        <div
class="input-group">
                                            <input
type="email" class="form-control"
placeholder="john@gmail.com"
                                                   ng-
model="friendemail" name="friendemail">
                                            <span
class="input-group-btn">
                                                <button
class="btn btn-default" type="button"

ng-click="addNewFriend(friendemail);friendemail=null;">

<span class="glyphicon glyphicon-envelope"></span>

</button>
                                            </span>
                                        </div>
                                        <!-- /input-
group -->
                                    </div>
                                </div>
                            </div>
                            <!--SECTION C-->
                            <div id="sectionC"
class="tab-pane fade">
                                <h3>Send Message</h3>

                                <div class="col-md-12">


                                    <select class="form-
control" ng-model="selected_friend">
                                                    <option ng-
repeat="friend in friends" value="{{friend.friend_email}}">

{{friend.name}} {{friend.surname}}
                                                    </option>
                                    </select><br>

                                    <textarea
class="form-control"

placeholder="Enter your message here..." ng-
model="contentMessage"></textarea><br>
                                    <button class="btn
btn-primary" ng-
click="emailFriend(selected_friend,contentMessage)">Send</bu
tton>
                                    <br><br>
                                </div>
                            </div>
                            <!--SECTION D-->
                            <div id="sectionD"
class="tab-pane fade">
                                <div class="row
vcenter">
                                    <div class="col-md-1
col-xs-2"><img id="dashboard-logo"/></div>
                                    <div class="col-md-7
col-xs-7">
                                        <div id="user-
name" class="text-center" style="padding-top:5px;">Users
                                                name
                                        </div>
                                    </div>
                                    <div class="col-md-1
col-xs-1" style="padding: 0px;"><img id="user-logo"

class="img-circle">
                                    </div>
                                    <div class="col-md-3
```

```
col-xs-2">
                                                    <button
class="btn btn-primary" style="width: 100%;">Sign
Out</button>
                                                </div>
                                            </div>
                                        </div>
                                        <!--SECTION E-->
                                        <div id="sectionE"
class="tab-pane fade"></div>
                                    </div>
                                </div>
                                <!-- /.navbar-collapse -->
                            </div>
                            <!-- /.container-fluid -->
                        </nav>
                        <!-- End here -->
                    </div>
                    <!--End of Dashboard-->
                </div>

                <!--Alert Box-->
                <!--class="alert alert-danger alert-dismissable"-->
                <div ng-class="{'alert':1===1 ,'alert-
dismissable':1===1, 'alert-danger':isAlertTypeCode(0),
'alert-success':isAlertTypeCode(1)}"  role="alert" ng-
show="isAlertOn(1)">
                    <button type="button" class="close" ng-
click="setAlert(0)">
                        <span aria-hidden="true">&times;</span>
                    </button>
                    <strong>{{alertType}}</strong> {{alertContent}}
                </div>

            </div>
        </div>

        <!--Map and Markers of the project-->
        <!--Map-->
        <gm-map gm-map-id="'myMap'" gm-center="center" gm-
        zoom="zoom" gm-bounds="bounds" gm-map-type-id="mapTypeId"
                gm-map-options="options.map" id="map-canvas">
            <gm-markers gm-objects="libraries"
                        gm-id="object.library_id"
                        gm-position="{lat: object.latitude, lng:
        object.longitude}"

                        gm-events="markerEvents"
                        gm-on-openinfowindow="selectedLibrary =
        object; infoWindow.open(marker.getMap(), marker);"
                        gm-on-click="triggerOpenInfoWindow(object)"


                        gm-marker-options="getMarkerOptions(object)"
                        gm-on-dragend="setLibraryLocation(object,
        marker)"
                    >
                <!--gm-marker-options="options.marker(object)"-->
            </gm-markers>
        </gm-map>
        <!--Infowindow-->
        <div gm-info-window="infoWindow" ng-init="setAddBook(0)">
            <h4>{{selectedLibrary.name}}</h4>
            <!--Add a New Book-->
            <div class="panel panel-success">
                <div class="panel-heading">
                    Add a new book
                    <button class="float-right" style="border:
        0;background-color: transparent">
                        <span class="glyphicon glyphicon-collapse-
        up" ng-show="isAddBook(1)" ng-click="setAddBook(0)"></span>
                        <span class="glyphicon glyphicon-collapse-
        down" ng-show="isAddBook(0)" ng-
        click="setAddBook(1)"></span>
                    </button>
                </div>
                <div class="panel-body" ng-show="isAddBook(1)">
                    <input placeholder="ISBN" class="form-control"
```

```
type="number" ng-model="isbnInput"/><BR>
            <button class="btn btn-success"
                    ng-
click="addNewBookRequest(isbnInput,selectedLibrary.library_i
d);isbnInput=null;">
                Save
            </button>
        </div>
    </div>
    <table class="table table-striped table-bordered">
        <thead>
        <td>ISBN</td>
        <td>Title</td>
        <td>Author</td>
        <td>#Pages</td>
        <td>Year of Publication</td>
        <!--<td>Status</td>-->
        </thead>
        <tr ng-repeat="book in selectedLibrary.books">
            <td>{{book.isbn}}</td>
            <td>{{book.title}}</td>
            <td>{{book.authors}}</td>
            <td>{{book.pageCount}}</td>
            <td>{{book.publishedYear}}</td>
            <!--<td>{{book.status === 0 ?
'Available':'Rent'}}</td>-->
        </tr>
    </table>



<span><h6>Owner:</h6>{{selectedLibrary.library_id}}</span>
</div>

<!--Scripts to include-->
<!--Scripts for map-->
<script
src="//maps.googleapis.com/maps/api/js?sensor=false"></scrip
t>
```

```
<!--<script
src="//ajax.googleapis.com/ajax/libs/angularjs/1.2.27/angula
r.min.js"></script>-->
<script src="scripts/angularjs/angular.min.js"></script>
<script src="//dylanfprice.github.io/angular-
gm/1.0.1/angular-gm.min.js"></script>
<!--/Scripts for map-->
<!--Scripts for login-->
<!--<script
src="https://apis.google.com/js/client:platform.js" async
defer></script>-->
<!--<script src="./scripts/signinCallback.js"></script>-->

<script src="scripts/google-plus-signin.js"></script>
<!--/Scripts for login-->
<script src="scripts/app.js"></script>
</body>
</html>
```

Server.js

```
/**
 * Created by Stefanos
 */
var mysql = require("mysql");
var express = require("express");
//var nodemailer = require('nodemailer');

var mandrill = require('mandrill-api/mandrill');
var mandrill_client = new
mandrill.Mandrill('hLxV7Qjf5KPJUDO5iD1Rfw');

var httprequest = require("request");
var app = express();



//Connection to database
var pool = mysql.createPool({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'smartlib2'
});

//To send e-mails you need a transporter object
//var transporter = nodemailer.createTransport();

app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Headers", "Origin, X-
Requested-With, Content-Type, Accept");
    next();
});

//Works OK! Add new user
app.post('/user/:name/:surname/:email', function (request,
response) {
    var name = request.params.name;
    var surname = request.params.surname;
    var email = request.params.email;
    //console.log(name + "\n" + surname + "\n" + email);
    var checkExistence = "SELECT * FROM  smartlib_user WHERE
email = '"+email+"'";
    //console.log(checkExistence);
    var insertQuery = "INSERT INTO smartlib_user
(name,surname,email) VALUES ('" + name + "','" + surname +
"','" + email + "')";
    pool.getConnection(function (err, connection) {
        connection.query(checkExistence, function (err,
rows) {
            if (err) {
                //console.error(err);
                connection.release();
                response.status(500).json("Internal Server
error!");
                response.end();
            } else {
                //console.log(rows);
                //console.log("Length:"+rows.length);
                if(rows.length!=0){
                    connection.release();
                    response.status(200).json("User already
registered!");
                    response.end();
                }else {
                    connection.query(insertQuery, function
(err, rows) {
                        if (err) {
                            console.error(err);
                        } else {
                            connection.release();
                            response.status(200).json("User
added successfully!");
                            response.end();
                        }
                    });
```

```
                }
            }
        });
    });
});

//Works OK! Retrieves all libraries of user and their books
app.get("/books/:email", function (request, response) {
    var email = request.params.email;
    var retrieveLibrariesQuery = "SELECT * from
SMARTLIB_LIBRARY " +
        "JOIN SMARTLIB_USER_LIBRARY " +
        "ON SMARTLIB_USER_LIBRARY.library_id =
smartlib_library.library_id " +
        "WHERE smartlib_user_library.user_email =
'"+email+"'";

    pool.getConnection(function (err, connection) {
        connection.query(retrieveLibrariesQuery, function
(err, rows) {
            if (err) {
                console.error(err);
                connection.release();
                response.status(500).json("Internal Server
Error!");
                response.end();
            }else {

                //Create library Objects
                var libraries = [];
                var librariesList ="(";
                for (var i = 0; i < rows.length; i++) {
                    libraries[rows[i].library_id] =
                    {
                        "library_id": rows[i].library_id,
                        "name": rows[i].name,
                        "latitude": rows[i].latitude,
                        "longitude": rows[i].longitude,
                        "books": []
                    };
                    librariesList+=rows[i].library_id+",";
                };
                librariesList = librariesList.substring(0,
librariesList.length - 1);
                librariesList+=")";

                var retrieveBooksQuery ="SELECT * FROM
smartlib_book " +
                    "JOIN smartlib_library_books " +
                    "ON smartlib_book.isbn =
smartlib_library_books.book_isbn " +
                    "WHERE smartlib_library_books.library_id
IN "+librariesList;

                connection.query(retrieveBooksQuery,
function (err, books) {
                    if (err) {
                        console.error(err);
                        connection.release();
                        response.status(500).json("Internal
Server Error!");
                        response.end();
                    } else {
                        var libraryBooks = [];
                        for (var i = 0; i < books.length;
i++) {

libraryBooks[libraryBooks.length] = {
                                    "library_id":
books[i].library_id,
                                    "isbn": books[i].isbn,
                                    "title": books[i].title,
                                    "authors": books[i].authors,
                                    "publishedYear":
books[i].publishedYear,
                                    "pageCount":
books[i].pageCount,
                                    "imgURL": books[i].imgURL,
```

```
                            "lang": books[i].lang
                        };
                    };

                    //console.log(libraryBooks);
                    for (var j = 0; j <
libraryBooks.length; j++) {

libraries[libraryBooks[j].library_id].books.push(libraryBook
s[j]);
                    };


                    for (var k = 0; k <
libraries.length; k++) {
                        if (!libraries[k]) {
                            libraries.splice(k, 1);
                            k--;
                        }
                    };

                    connection.release();
                    response.json(libraries);
                    response.end();
                }
            });
        }
    });
});

//Works OK! Retrieve user's friends
app.get("/friends/:email", function (request,response){
    var email = request.params.email;
    var retrieveFriendsQuery ="SELECT * FROM
smartlib_friends JOIN smartlib_user ON
smartlib_friends.friend_email = smartlib_user.email " +
    "WHERE smartlib_friends.user_email ='"+email+ "'";
    //"SELECT * FROM smartlib_friends WHERE
user_email='"+email+"'";

    pool.getConnection(function (err, connection) {
        connection.query(retrieveFriendsQuery, function
(err, rows) {
            if (err) {
                console.error(err);
                connection.release();
                response.status(500).json("Internal Server
Error!");
                response.end();
            } else {
                console.log(rows);
                response.status(200).json(rows);
                response.end();
                connection.release();
            }
        });
    });
});

//Works OK! Sents message from user to friend
app.post("/message/friends/:user_email/:friends_email/:messa
ge",function(request,response){
    var sender = request.params.user_email;
    var recipient = request.params.friends_email;
    var messageContent = request.params.message;

    var message = {
        "html": messageContent,
        "text": messageContent,
        "subject": "SmartLib Friend Message",
        "from_email": sender,
        "from_name": sender,
        "to": [{
            "email": recipient,
            "name": recipient,
            "type": "to"
        }]
```

```
    };

    var async = false;
    var ip_pool = "Main Pool";
    var send_at = null;
    mandrill_client.messages.send({"message": message,
"async": async, "ip_pool": ip_pool, "send_at": send_at},
function(result) {
        response.status(200).json("Email Sent!");
        response.end();
    }, function(e) {
        // Mandrill returns the error as an object with name
and message keys
        console.log('A mandrill error occurred: ' + e.name +
' - ' + e.message);
        response.status(500).json("Email wasn't sent!");
        response.end();
        // A mandrill error occurred: Unknown_Subaccount -
No subaccount exists with the id 'customer-123'
    });

});

//Works OK! Send email with link for friends request
acceptance
app.get("/request/friends/:user_id/:friends_id", function
(request, response) {
    var sender = request.params.user_id;
    var recipient = request.params.friends_id;

    //console.log("From:"+sender+" To:"+recipient);

    var link =
"http://localhost:8080/accept/friends/"+sender+"/"+recipient
;
    var message = {
        "html": "<h1>Welcome to SmartLib - The Virtual
Library</h1><br>" +
        "<p>You have a friend request from SmartLib.<br>" +
        " To accept the friend request you have to follow
the link below:</p>" +
        " <a href='"+link+"'>"+link+"</a><br><br>" +
        "By clicking the link above you accept to share your
data in SmartLib Virtual Library." +
        "You accept to share with the person sending the
request your libraries and their content." +
        "Meanwhile you are going to be able to view his
libraries and books also. <br><br>" +
        "Thank you for using SmartLib.",
        "text": "Welcome to SmartLib - The Virtual
Library\n" +
        "You have a friend request from SmartLib.\n" +
        " To accept the friend request you have to follow
the link below:\n" + link+
        "\n\nBy clicking the link above you accept to share
your data in SmartLib Virtual Library." +
        "You accept to share with the person sending the
request your libraries and their content." +
        "Meanwhile you are going to be able to view his
libraries and books also.\n\n" +
        "Thank you for using SmartLib.",
        "subject": "SmartLib Friend Request",
        "from_email": sender,
        "from_name": sender,
        "to": [{
            "email": recipient,
            "name": "Recipient Name",
            "type": "to"
        }]
    };

    var async = false;
    var ip_pool = "Main Pool";
    var send_at = null;
    mandrill_client.messages.send({"message": message,
"async": async, "ip_pool": ip_pool, "send_at": send_at},
function(result) {
        console.log("Message Sent");
```

```
        console.log(result);
        response.status(200).json("Email Sent!");
        response.end();
        /*
         [{
         "email": "recipient.email@example.com",
         "status": "sent",
         "reject_reason": "hard-bounce",
         "_id": "abc123abc123abc123abc123abc123"
         }]
         */
    }, function(e) {
        // Mandrill returns the error as an object with name
and message keys
        //console.log('A mandrill error occurred: ' + e.name
+ ' - ' + e.message);
        response.status(500).json(e.message);
        response.end();
        // A mandrill error occurred: Unknown_Subaccount -
No subaccount exists with the id 'customer-123'
    });

});

//Works OK! At friends to database
app.get("/accept/friends/:user_email/:friends_email",
function(request,response){
    var user = request.params.user_email;
    var friend = request.params.friends_email;
    var checkQuery = "SELECT * FROM smartlib_friends WHERE
user_email='"+user+"'";
    var insertQuery = "INSERT INTO
smartlib_friends(user_email, friend_email) VALUES
('"+user+"','"+friend+"')";
    var insertQueryReverse = "INSERT INTO
smartlib_friends(user_email, friend_email) VALUES
('"+friend+"','"+user+"')";
    pool.getConnection(function (err, connection) {
        connection.query(checkQuery, function (err,rows) {
            if (err) {
                console.error(err);
                connection.release();
                response.status(500).json("Server Internal
Error!");
                response.end();
            } else {
                if(rows.length>0){
                    connection.release();
                    //response.status(500).json("Server
Internal Error!");
                    response.end();
                }else{
                    connection.query(insertQuery, function
(err,rows) {
                        if (err) {
                            console.error(err);
                            connection.release();

response.status(500).json("Server Internal Error!");
                            response.end();
                        } else {

connection.query(insertQueryReverse, function (err,rows) {
                                if (err) {
                                    console.error(err);
                                    connection.release();

response.status(500).json("Server Internal Error!");
                                    response.end();
                                } else {
                                    connection.release();

//response.status(500).json("Server Internal Error!");
                                    response.end();
                                }
                            });
                        }
                    });
```

```
                }
            }
        });
    });

});

//Works OK! Inserts a new library
app.post("/library/:name/:lat/:lng/:email", function
(request, response) {
    var library_name = request.params.name;
    var library_lat = request.params.lat;
    var library_lng = request.params.lng;
    var email = request.params.email;
    var queryInsert = "INSERT INTO
smartlib_library(latitude,longitude,name)
VALUES('"+library_lat+"','"+library_lng+"','"+library_name+"
')";
    var queryRetrieve = "SELECT library_id FROM
smartlib_library WHERE
smartlib_library.name='"+library_name+"'";
    var queryInsertUserLibrary = "INSERT INTO
smartlib_user_library(user_email, library_id) VALUES
('"+email+"','";
    console.log(queryInsert);
    pool.getConnection(function (err, connection) {
        connection.query(queryInsert, function (err) {
            if (err) {
                console.error(err);
                connection.release();
                response.status(500).json("Server Internal
Error!");
                response.end();
            }else{
                connection.query(queryRetrieve, function
(err, rows) {
                    if (err) {
                        console.error(err);
                        connection.release();
                        response.status(500).json("Server
Internal Error!");
                        response.end();
                    } else {
                        var library_id = rows[rows.length-
1].library_id;
                        console.log(library_id);

queryInsertUserLibrary+=library_id+"')";
                        console.log(queryInsertUserLibrary);

connection.query(queryInsertUserLibrary, function (err) {
                            if (err) {
                                console.error(err);
                                connection.release();

response.status(500).json("Server Internal Error!");
                                response.end();
                            } else {
                                //console.log("User_Library
DONE!");
                                connection.release();

response.status(200).json("New Library added succesfully!");
                                response.end();
                            }
                        });
                    }
                });
            }
        });
    });
});

//Works OK!Update coordinates of a library
app.post("/update/library/:id/:lat/:lng",function(request,re
sponse){
    var id = request.params.id;
    var latitude = request.params.lat;
```

```
    var longitude = request.params.lng;
    var updateQuery = "UPDATE smartlib_library SET
latitude='"+latitude+"',longitude='"+longitude+
        "' WHERE library_id='"+id+"'";
    console.log("Here update");
    pool.getConnection(function (err, connection) {
        connection.query(updateQuery, function (err) {
            if (err) {
                console.error(err);
                connection.release();
                response.status(500).json("Server Internal
Error!");
                response.end();
            } else {
                connection.release();
                response.status(200).json("Library
coordinates changed succesfully!");
                response.end();
            }
        });
    });
});

//Works OK! Remove library
app.post("/remove/library/:library_id", function (request,
response) {
    var library_id = request.params.library_id;
    var deleteLibraryQuery = "DELETE FROM smartlib_library
WHERE library_id='"+library_id+"'";
    ////console.log(query2);
    //response.status(200);
    //response.end();
    pool.getConnection(function (err, connection) {
        connection.query(deleteLibraryQuery, function (err,
rows) {
            if (err) {
                console.error(err);
                connection.release();
                response.status(500).json("Server Iinternal
Error! "+err);
                response.end();
            }
            else{
                connection.release();
                response.status(200).json("Library deleted
succesfully");
                response.end();
            }
            //connection.release();
            //response.json(rows);
            //// response.end();
        });
    });
});

//Works OK! Insertion of new book
app.post("/library/:id/book/:isbn", function (request,
response) {
    var isbn = request.params.isbn;
    var library_id = request.params.id;

httprequest("https://www.googleapis.com/books/v1/volumes?q=i
sbn:" + isbn , function (error, httpresponse, body) {
        if (!error && httpresponse.statusCode == 200) {
            //console.log(body);
            var objectReturned = JSON.parse(body);
            //console.log("Total
items:"+objectReturned.totalItems);
            if (objectReturned.totalItems > 0) {
                var volumeInfo =
objectReturned.items[0].volumeInfo;

                var title = volumeInfo.title;
                var authors = volumeInfo.authors;
                var publishedYear =
volumeInfo.publishedDate;
                var pageCount = volumeInfo.pageCount;
                var imgURL =
```

```
volumeInfo.imageLinks.thumbnail;
                var lang = volumeInfo.language;

                //QUERIES
                var checkQuery = "SELECT * FROM
smartlib_book WHERE isbn='" + isbn + "'";
                var newBookQuery = "INSERT INTO
smartlib_book(isbn, title, authors, publishedYear,
pageCount, imgURL, lang)" +
                    " VALUES ('" + isbn + "','" + title +
"','" + authors + "','" + publishedYear + "','" + pageCount
+ "','" + imgURL + "','" + lang + "')";
                var newLibraryBookQuery = "INSERT INTO
smartlib_library_books(library_id, book_isbn) " +
                    "VALUES ('" + library_id + "','" + isbn
+ "')";
                pool.getConnection(function (err,
connection) {
                    connection.query(checkQuery, function
(err, rows) {
                        if (err) {
                            console.error(err);
                            connection.release();

response.status(500).json("Internal Server Error! " + err);
                            response.end();
                        } else {
                            if (rows.length>0) {

connection.query(newLibraryBookQuery, function (err) {
                                    if (err) {
                                        console.error(err);

connection.release();

response.status(500).json("Internal Server Error! " + err);
                                        response.end();
                                    } else {

connection.release();

response.status(200).json("Insertion of book successful!");
                                        response.end();
                                    }
                                });
                            } else {

connection.query(newBookQuery, function (err) {
                                    if (err) {
                                        console.error(err);

connection.release();

response.status(500).json("Internal Server Error! " + err);
                                        response.end();
                                    } else {

console.log("=====Insert book_info.....DONE!");

connection.query(newLibraryBookQuery, function (err) {
                                            if (err) {

console.error(err);

connection.release();

response.status(500).json("Internal Server Error! " + err);

response.end();
                                            } else {

connection.release();

response.status(200).json("Insertion of book successful!");

response.end();
                                            }
                                        });
```

```
                                    }
                                });
                            }
                        }
                    });
                });
            }else{
                connection.release();
                response.status(500).json("Server Internal
Error!");
                response.end();
            }
        }
    });
    //response.status(200);
});

//Works Ok!Rename library with id(id) to name(name)
app.post("/library/:id/:name", function (request, response)
{
    var id = request.params.id;
    var name = request.params.name;
    var updateQuery = "UPDATE smartlib_library SET name =
'"+name+"' WHERE library_id='"+id+"'";
    pool.getConnection(function (err, connection) {
        connection.query(updateQuery, function (err) {
            if (err) {
                console.error(err);
                connection.release();
                response.status(500).json("Server Internal
Error!");
                response.end();
            } else {
                connection.release();
                response.status(200).json("Library renamed
succesfully!");
                response.end();
            }
        });
    });
});

app.listen(8080, function () {
    console.log('Listening on port 8080...');
});
```

App.js

```
/**
 * Created by Stefanos
 */
(function(){
    var app = angular.module('smartlib',
['AngularGM','directive.g+signin']);

    app.controller('mapController',function($scope,$http){

        $scope.user_name = null;
        $scope.user_surname = null;
        $scope.user_email = null;


        //Scopes options for map and marker
        $scope.icons = {
            gray:'images/marker50x50_grsl.png',
            color:'images/marker50x50.png'
        };
        //Options for map and markers
        $scope.options = {
            map: {
                center: new google.maps.LatLng(35.1478913,
33.4143956),
                zoom: 16,
                mapTypeId: google.maps.MapTypeId.ROADMAP,
                disableDefaultUI: true
            },
```

```
            highlighted: {
                draggable: true,
                //title:library.name,
                icon: $scope.icons.color
            },
            unhighlighted: {
                icon: $scope.icons.gray,
                clickable:false
            }
            //marker: function(library) {
            //    return {
            //        clickable:false,
            //        draggable: true,
            //        //icon:$scope.icons.gray,
            //        title: library.name
            //    }
            //}
        };


        //=========================//
        //=======ALERT CONTROL=====//
        //=========================//
        // Value and functions for alerts
        // Warning:0 / Success:1
        $scope.alert = 0;
        $scope.alertContent = "Better check yourself, you're
not looking too good.";
        $scope.alertType = "Warning!";
        $scope.alertTypeCode = 0;
        $scope.isAlertTypeCode = function(alertCode){
            return alertCode === $scope.alertTypeCode
        };
        $scope.setAlertTypeCode = function(alertCode){
            $scope.alertTypeCode = alertCode;
        };
        $scope.isAlertOn = function(alertInput){
            return alertInput === $scope.alert;
        };
        $scope.setAlert = function(alertValue){
            $scope.alert = alertValue;
        };
        $scope.setAlertContent =
function(typeCode,alertContent){

            $scope.alertContent = alertContent;
            $scope.setAlertTypeCode(typeCode);
            if(typeCode == 0){
                $scope.alertType = "Warning!";
            }else if(typeCode == 1){
                $scope.alertType = "Success!";
            }
            $scope.alert = 1;
        };



        //=========================//
        //=======BOOK CONTROL======//
        //=========================//
        //Variable for add new book collapsed menu 0
collapsed / 1 non-collapsed
        $scope.addBook=0;
        //Set the variable for add new book collapsed menu
        $scope.setAddBook = function(value){
            $scope.addBook = value;
        };
        //Check if variable for add new book collapsed menu
is 0 or 1
        $scope.isAddBook = function(value){
            return $scope.addBook===value;
        };
        $scope.fetchBooks = function(){

$http.get('http://localhost:8080/books/'+$scope.user_email)
                .success(function (data) {
                    console.log(data);
                    $scope.libraries = data;
                    console.log($scope.libraries);
```

```
                })
                .error(function (data) {
                    console.log("Error");
                });
        };
        $scope.addNewBookRequest =
function(inputIsbn,inputLibrary){
            if(!inputIsbn){
                console.log("No input!");
            }else {
                console.log(inputIsbn,inputLibrary);
                $http.post('http://localhost:8080/library/'
+ inputLibrary+"/book/"+inputIsbn).
                    success(function (data, status) {
                        // this callback will be called
asynchronously
                        // when the response is available
                        console.log("Done!" + status);
                        $scope.setAlertContent(1,data);
                        $scope.fetchBooks();


//console.log($scope.selected.idLIBRARY);
                    }).
                    error(function (data, status) {
                        // called asynchronously if an error
occurs
                        // or server returns response with
an error status.
                        console.log("Status: "+status);
                        $scope.setAlertContent(0,data);
                    });
            }
        };


        //=========================//
        //=====LIBRARY CONTROL=====//
        //=========================//
        $scope.filters = {
            name: null
        };
        $scope.filterLibraries = function() {
            $scope.filteredLibraries = {};
            angular.forEach($scope.libraries,
function(library) {
                var nameMatch = ($scope.filters.name) ?
~library.name.indexOf($scope.filters.name) : true;
                if (nameMatch) {

$scope.filteredLibraries[library.library_id] = library;
                }
            });
            $scope.$broadcast('gmMarkersUpdate',
'libraries');
        };
        $scope.$watch('libraries', function() {
            $scope.filterLibraries();
        });
        //On drag of a marker change latitude and longitude
        $scope.setLibraryLocation = function(library,
marker) {
            var position = marker.getPosition();
            library.latitude = position.lat();
            library.longitude = position.lng();

            //console.log("Here:"+library.library_id);
            //HTTP REQUEST

$http.post("http://localhost:8080/update/library/"+library.l
ibrary_id+"/"+library.latitude+"/"+library.longitude).
                success(function (data, status, headers,
config) {
                    // this callback will be called
asynchronously
                    // when the response is available
                    console.log(data);
                    $scope.setAlertContent(1,data);
```

```
                $scope.fetchBooks();
            }).
            error(function (data, status, headers,
config) {
                // called asynchronously if an error
occurs
                // or server returns response with an
error status.
                console.log(data);
                $scope.setAlertContent(0,data);
                $scope.fetchBooks();
            });

        };
        //Projects libaries (it should be changed to [] and
filled up from http request
        $scope.libraries = [];
        function arrayObjectIndexOf(myArray, searchTerm,
property) {
            for(var i = 0, len = myArray.length; i < len;
i++) {
                if (myArray[i][property] === searchTerm)
return i;
            }
            return -1;
        }
        $scope.addNewLibrary = function(newLibrary){
            console.log(newLibrary);
            $http.post('http://localhost:8080/library/' +
newLibrary.name+"/"+newLibrary.latitude+"/"+newLibrary.longi
tude+"/"+$scope.user_email).
            success(function (data, status, headers,
config) {
                // this callback will be called
asynchronously
                // when the response is available
                console.log(data);
                $scope.setAlertContent(1,data);
                $scope.fetchBooks();
//console.log($scope.selected.library_id);
            }).
            error(function (data, status, headers,
config) {
                // called asynchronously if an error
occurs
                // or server returns response with an
error status.
                //console.log("Status: "+status);
                $scope.setAlertContent(0,data);
            });
        };
        $scope.renameLibrary =
function(libraryId,libraryNewName){
            $scope.setAlert(0);
            if(!libraryId){
                $scope.setAlertContent(0,"No Library
selected. Please select a library to rename!");
                console.log("Nothing selected!");
            }else {
                if(!libraryNewName){
                    $scope.setAlertContent(0,"You have to
enter the new name of the library!");
                    console.log("Nothing selected!");
                }else {
                    console.log("I should rename " +
libraryId + " to " + libraryNewName);
                    //HTTP REQUEST

$http.post("http://localhost:8080/library/"+libraryId+"/"+li
braryNewName).
                        success(function (data, status,
headers, config) {
                            // this callback will be called
asynchronously
                            // when the response is
available
                            console.log(data);
```

```
                        $scope.setAlertContent(1,data);
                        $scope.fetchBooks();
                    }).
                    error(function (data, status,
headers, config) {
                        // called asynchronously if an
error occurs
                        // or server returns response
with an error status.
                        console.log(data);
                        $scope.setAlertContent(0,data);
                        $scope.fetchBooks();
                    });
                }
            }
        };
        $scope.deleteLibrary = function(libraryId){
            console.log("Alert is: "+$scope.alert);
            if(!libraryId){
                $scope.setAlertContent(0,"No Library
selected. Please select a library to delete!");
                console.log("Nothing selected!");
            }
            else {
                var indexOfRemove =
arrayObjectIndexOf($scope.libraries,libraryId, "idLIBRARY");
                console.log($scope.libraries);
                console.log(indexOfRemove);
                $scope.libraries.splice(indexOfRemove,1);
                $scope.selected=null;

$http.post('http://localhost:8080/remove/library/' +
libraryId).
                    success(function (data, status, headers,
config) {
                        // this callback will be called
asynchronously
                        // when the response is available
                        //console.log("Delete!" + status);
                        $scope.setAlertContent(1,data);


//console.log($scope.selected.idLIBRARY);
                    }).
                    error(function (data, status, headers,
config) {
                        // called asynchronously if an error
occurs
                        // or server returns response with
an error status.
                        $scope.setAlertContent(0,data);
                        console.log("Status: "+status);
                    });
            }
        };


        //=========================//
        //======MARKER CONTROL=====//
        //=========================//
        //Add a new marker to the markers list
        $scope.addMarker = function(){
            var larray = $scope.libraries;
            var lastId = larray[larray.length-1].library_id;
            var center = $scope.options.map.center;
            var newLibrary ={
                library_id: lastId+1,
                latitude: center.k,
                longitude: center.D,
                name: "Default"+lastId
            };
            $scope.addNewLibrary(newLibrary);
            $scope.libraries.push(newLibrary);

//$scope.addNewLibrary($scope.libraries[$scope.libraries.len
gth-1]);
            $scope.filterLibraries();
            //$scope.$broadcast('gmMarkersUpdate',
```

```
'libraries');
            //console.log($scope.libraries);
        };
        //Remove marker and library
        $scope.removeMarker = function(){};
        //When gmMarkersUpdate is triggered write to console
        $scope.$on('gmMarkersUpdated', function(event,
objects) {
            console.log("Markers Updated");
            //$scope.$broadcast('gmMarkersRedraw',
'libraries');
        });
        //Trigger to open the InfoWindow of a marker
        $scope.triggerOpenInfoWindow = function(library) {
            $scope.markerEvents = [
                {
                    event: 'openinfowindow',
                    ids: [library.library_id]
                },
            ];
        };
        //Set marker's options
        $scope.getMarkerOptions = function(library) {
            var opts = {title: library.name};
            if (library.library_id in
$scope.filteredLibraries) {
                return angular.extend(opts,
$scope.options.highlighted);
            } else {
                return angular.extend(opts,
$scope.options.unhighlighted);
            }
        };


        //=========================//
        //====ADD FRIEND CONTROL===//
        //=========================//
        //List with user's friends
        $scope.friends=[
            //{name:'Navid Jakub'},
            //{name:'Matey Stan'},
            //{name:'Chidiebube Li'}
        ];
        //A variable for show/hide add new friend form
        $scope.addFriend=0;
        //Set the show/hide variable for new friend form
        $scope.setAddFriend = function(value){
            $scope.addFriend = value;
        };
        //Get true/false if variable is set
        $scope.isAddFriend = function(value){
            return $scope.addFriend === value;
        };
        //Function called to save the new friend
        $scope.addNewFriend = function(email){

$http.get('http://localhost:8080/request/friends/'+$scope.us
er_email+"/"+email)
                .success(function (data) {
                    $scope.setAlertContent(1,data);
                    $scope.fetchFriends();
                })
                .error(function (data) {
                    $scope.setAlertContent(0,data);
                });
        };
        //Function to retrieve friends
        $scope.fetchFriends = function(){

$http.get('http://localhost:8080/friends/'+$scope.user_email
)
                .success(function (data) {
                    console.log(data);
                    $scope.friends = data;
                    console.log($scope.friends);
                })
                .error(function (data) {
```

```
                console.log("Error");
            });
        };
        //Email friend
        $scope.emailFriend = function(email, content){
            if(!email){
                $scope.setAlertContent(0,"No friend
selected!");
            }else{
                if(!content){
                    $scope.setAlertContent(0,"No message!
Please enter your message above!");
                }else{

$http.post('http://localhost:8080/message/friends/'+$scope.u
ser_email+"/"+email+"/"+content)
                        .success(function (data) {
                            $scope.setAlertContent(1,data);
                        })
                        .error(function (data) {
                            $scope.setAlertContent(1,data);
                        });
                }
            }

        };

        //=========================//
        //====DASHBOARD CONTROL====//
        //=========================//
        //Variable for dashboard's visibility
        $scope.dashboardOn=0;
        //Set variable for dashboard's visibility
        $scope.setDashboard = function(value){
            $scope.dashboardOn = value;
        };
        //Check if dashboard is set to visible
        $scope.isDashboard = function(value){
          return $scope.dashboardOn === value;
        };


        //=========================//
        //=========SIGN IN=========//
        //=========================//
        //On succesfull login
        $scope.$on('event:google-plus-signin-success',
function (event,authResult) {
            // Send login to server or save into cookie
            $scope.setDashboard(1);
            //Load libraries with their books
            $scope.makeAPICall();
            $scope.$on('user:loaded',function(){
                $scope.saveUser();
            });
            $scope.$on('user:checked',function(){
                $scope.fetchBooks();
                $scope.fetchFriends();
                $scope.$apply();
            });
        });
        $scope.saveUser = function(){
          console.log("User loaded");
            console.log($scope.user_email+"
"+$scope.user_name+" "+$scope.user_surname);

$http.post('http://localhost:8080/user/'+$scope.user_name+"/
"+$scope.user_surname+"/"+$scope.user_email).
                success(function (data, status, headers,
config) {
                    // this callback will be called
asynchronously
                    // when the response is available
                    console.log("Done!" + status+" "+data);

//console.log($scope.selected.idLIBRARY);
                }).
                error(function (data, status, headers,
```

```
config) {
                    // called asynchronously if an error
occurs
                    // or server returns response with an
error status.
                    console.log("Status: "+status);
                    $scope.setAlertContent(0,"Could not
access server!");

                });
            $scope.$broadcast('user:checked');
        };
        //Load users data making an API call
        $scope.makeAPICall = function() {
            gapi.client.load('plus', 'v1', function () {
                var request = gapi.client.plus.people.get({
                    'userId': 'me'
                });
                request.execute(function (resp) {
                    console.log(resp);
                    if (resp.id) {
                        //console.log('ID: ' + resp.id);
                        $scope.user_ID = resp.id;
                        document.getElementById("user-
name").innerHTML = resp.id;
                    }
                    if (resp.displayName) {
                        //console.log('Display Name: ' +
resp.displayName);
                        $scope.user_name =
resp.name.givenName;
                        $scope.user_surname =
resp.name.familyName;
                        document.getElementById("user-
name").innerHTML = resp.displayName;
                    }
                    if (resp.image && resp.image.url) {
                        //console.log('Image URL: ' +
resp.image.url);
                        document.getElementById("user-
logo").src = resp.image.url;
                    }
                    if (resp.emails) {
                        //console.log('Profile URL: ' +
resp.url);
                        $scope.user_email =
resp.emails[0].value;
                    }
                    $scope.$broadcast('user:loaded');
                });
            });
        };



    });
})();
```

Style.css

```
html, body, #map-canvas {
    height: 100%;
    margin: 0;
    padding: 0;
}

#searchBox{
    position: absolute;
    width: 100%;
    padding-top: 1%;
}

#searchContent{
    box-shadow: 0px 7px 5px #ccc;
}

#dashboard{
    /*display: none;*/
    box-shadow: 0px 7px 5px #ccc;
```

```
}

#dashboard-logo{
    content: url("../images/smartlib_logo.png");
    width: 200%;
}

#addMarker{
    width: 100%;
}

#friend-list{
    max-height: 200px;
    overflow-y:scroll;
}
```