Ατομική Διπλωματική Εργασία

"Υλοποίηση μίας Νοητής Βιβλιοθήκης Πληθοπορισμού με Έξυπνα Κινητά Τηλέφωνα"

Γαβριέλα Κουπεπίδου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

"Υλοποίηση μίας νοητής Βιβλιοθήκης Πληθοπορισμού με Έξυπνα Κινητά Τηλέφωνα"

Γαβριέλα Κουπεπίδου

Επιβλέπων Καθηγητής

Δημήτρης Ζεϊναλιπούρ

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

Μάιος 2012

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω θερμά τον επιβλέπον καθηγητή της παρούσας διπλωματικής εργασίας, κύριο Δημήτρη Ζεϊναλιπούρ, για την ευκαιρία που μου προσέφερε μιας και μέσα από την καθοδήγηση του αλλά και με τον εξοπλισμό που μου παρείχε κατάφερα να εργαστώ εποικοδομητικά και να φέρω εις πέρας την εν λόγω διπλωματική εργασία. Πρόκειται για έναν εξαίρετο καθηγητή με απεριόριστες γνώσεις σε πολλούς τομείς, και ιδιαίτερα στους τομείς που αφορούν την επιστήμη της Πληροφορικής, αλλά πάνω από όλα για έναν εξαίρετο άνθρωπο πάντα πρόθυμο και με κατανόηση.

Στη συνέχεια θα ήθελα να ευχαριστήσω τους φίλους και συναδέλφους μου Γιώργο Λάρκου και Κωνσταντίνο Κώστα για την πολύτιμη βοήθεια αλλά και για τις συμβουλές τους στις δυσκολίες που αντιμετώπισα κατά την υλοποίηση της διπλωματικής μου εργασίας.

Θα ήθελα επίσης να ευχαριστήσω το Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου για την αγορά και διάθεση όλων των συσκευών που ήταν απαραίτητα για την ολοκλήρωση της παρούσας διπλωματικής εργασίας, όπως επίσης και όλους τους καθηγητές του Τμήματος Πληροφορικής οι οποίοι δέχτηκαν να συμβάλουν και να ενταχθούν στην νοητή ηλεκτρονική βιβλιοθήκη που δημιουργήσαμε.

Τέλος, οφείλω ένα τεράστιο ευχαριστώ στην οικογένεια μου, και ιδιαίτερα στους γονείς μου Αχιλλέα και Ελένη που με στήριξαν ηθικά και οικονομικά καθ’ όλη τη διάρκεια των σπουδών μου, όπως επίσης και στους φίλους μου για όλη τη στήριξη και τη συμπαράσταση που μου παρείχαν όλο αυτό τον καιρό.

Περίληψη

Οι έξυπνες συσκευές (smartphones) στις μέρες μας είναι εξοπλισμένες με πολλών ειδών αισθητήρες, όπως για παράδειγμα κάμερα, γυροσκόπιο, επιταχυνσιόμετρο αλλά και δέκτες ασύρματου δικτύου (WiFi, Bluetooth), γεγονός που τους δίνει την ικανότητα να ενασχολούνται με υπηρεσίες επίλυσης αποστολών που εκτελούνται από ομάδες ατόμων οι οποίες οικειοθελώς απαντούν σε ερωτήματα. Με τον όρο CrowdSource εννοούμε την αξιοποίηση του πλήθους αυτού για ακριβή και πραγματικού χρόνου βοήθεια σε διάφορες εργασίες που ανατίθενται με κίνητρο κάποιο έπαθλο ή την ηθική ικανοποίηση.

Στόχος της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μίας νοητής ηλεκτρονικής βιβλιοθήκης της οποίας την συντήρηση και επέκταση αναλαμβάνουν εξολοκλήρου οι ίδιοι οι χρήστες του συστήματος (CrowdSourcing). Η νοητή ηλεκτρονική βιβλιοθήκη μπορεί να χρησιμοποιηθεί από ομάδες ατόμων για εύρεση αλλά και δανεισμό βιβλίων από άτομα τα οποία γνωρίζουν ή άτομα τα οποία δεν γνωρίζουν αλλά στα οποία μπορεί να έχουν εύκολα πρόσβαση αφού για παράδειγμα βρίσκονται στον ίδιο εργασιακό χώρο. Συγκεκριμένα, αφορά τη δημιουργία συλλογικών κοινόχρηστων βιβλιοθηκών με απώτερο σκοπό τη δημιουργία ενός συστήματος προτάσεων (recommendation system) υψηλής ποιότητας. Η νοητή αυτή βιβλιοθήκη αποτελείται από δύο μέρη: μία εφαρμογή που αφορά έξυπνη συσκευή, SmartScan, (συγκεκριμένα Android[6]) και μία ιστοσελίδα υλοποιημένη σε γλώσσα προγραμματισμού HTML[7] και PHP[8].

Οι χρήστες μπορούν σε πρώτο στάδιο να σαρώσουν τους κωδικούς που αναγράφονται στο πίσω μέρος των βιβλίων (ISBN), μέσω της εφαρμογής στην Android συσκευή, την οποία και μπορούν να αποκτήσουν μέσω της ιστοσελίδας που έχει δημιουργηθεί, έτσι ώστε να δημιουργήσουν την προσωπική τους βιβλιοθήκη. Για την υλοποίηση της εφαρμογής αυτής έγινε χρήση της βιβλιοθήκης Zxing[9] η οποία είναι ανοικτή για χρήση από τους προγραμματιστές και η οποία παρέχει τη δυνατότητα σάρωσης των κωδικών των βιβλίων μέσω της κάμερας του κινητού. Για τα πειράματα της εφαρμογής χρησιμοποιήθηκε μια Android έξυπνη κινητή συσκευή HTC[10] (HTC Desire) η οποία και χορηγήθηκε από τον ακαδημαϊκό σύμβουλο. Επίσης, οι χρήστες οι οποίοι έχουν λογαριασμό, τον οποίο και δημιουργούν μέσω της ιστοσελίδας, μπορούν να δουν τα

βιβλία όλων των χρηστών, να διαχειριστούν τα δικά τους προσωπικά βιβλία έχοντας τη δυνατότητα να διαγράψουν κάποια από αυτά, ενώ παράλληλα μπορούν να βαθμολογήσουν τα βιβλία με βάση τα δικά τους κριτήρια έτσι ώστε να δημιουργείται και μία λίστα με τα top rated βιβλία, η οποία αποτελεί και μια σύσταση προς όλους του χρήστες για να τα διαβάσουν εάν το επιθυμούν. Όπως προαναφέραμε η ιστοσελίδα δίνει ακόμη τη δυνατότητα σε κάθε χρήστη να δει τον ιδιοκτήτη του βιβλίου αλλά και να επικοινωνήσει μαζί του μέσω ηλεκτρονικού ταχυδρομείου. Έτσι μπορεί να διευθετηθεί και ο δανεισμός κάποιου βιβλίου μετά από συνομιλία μεταξύ των δύο χρηστών εφόσον το επιθυμούν.

Συνολικά για την αξιολόγηση και την εγκυρότητα του συγκεκριμένου συστήματος μας έχουν σαρωθεί εκατοντάδες βιβλία που ανήκουν στις βιβλιοθήκες καθηγητών του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου.

Εν κατακλείδι το παρών σύστημα βρίσκεται σε λειτουργία και οι φοιτητές αλλά και το προσωπικό του τμήματος Πληροφορικής μπορούν να το επισκεφτούν μέσω της ηλεκτρονικής διεύθυνσης http://www.cs.ucy.ac.cy/thesis/smartscan/.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Βασικό Υπόβαθρο

1.2 Υποκίνηση Εργασίας

1.3 Σχετικά Άρθρα και μελέτη

1.4 Περιγραφή Εργασίας

1.1 Βασικό Υπόβαθρο

Ο ορισμός του "Crowdsourcing"[25] (Σχήμα 1.1.2) αναφέρεται στην πράξη της ανάθεσης καθηκόντων (tasks) που ενώ κανονικά εκτελούνται από κάποιον υπάλληλο τώρα εκτελούνται από μία μεγάλη ομάδα εθελοντών ή από μία κοινότητα, μέσω μίας ανοικτής πρόσκλησης με αντάλλαγμα κάποιο μικρό έπαθλο. Δεδομένου ότι η πρόσκληση αναφέρεται σε άγνωστο πλήθος ατόμων θα πρέπει να συγκεντρώνεται η άποψη και οι απαντήσεις των πιο ικανών για να συνεισφέρουν και για να αναλάβουν τα συγκεκριμένα καθήκοντα. Ο όρος έχει γίνει δημοφιλής στις επιχειρήσεις, σε συγγραφείς και δημοσιογράφους, ως μια συντόμευση της τάσης για την ανάδειξη της μαζικής συνεργασίας που προσφέρουν οι τεχνολογίες του Web προκειμένου να επιτευχθούν συγκεκριμένοι επιχειρηματικοί στόχοι.

Επιπλέον, το Crowdsourcing αναφέρεται σε ένα κατανεμημένο μοντέλο επίλυσης προβλημάτων στο οποίο ένα πλήθος , απροσδιόριστου μεγέθους, ασχολείται με την επίλυση ενός πολύπλοκου προβλήματος μέσω μίας ανοικτής πρόσκλησης. Αν και το Crowdsourcing δεν έχει αναπτυχθεί και διεισδύσει ακόμη πλήρως στο κόσμο των έξυπνων κινητών συσκευών, γεγονός που θα συνέτεινε στην πλήρη ανάπτυξη αυτού του νέου μοντέλου επίλυσης προβλημάτων, τα έξυπνα τηλέφωνα μπορούν να αξιοποιήσουν πλήρως τις δυνατότητες του crowdsourcing επιτρέποντας έτσι στους χρήστες να συμβάλουν στην επίλυση καινοφανών και περίπλοκων προβλημάτων. Αυτό

ισχύει λόγω των μοναδικών χαρακτηριστικών των έξυπνων κινητών συσκευών, αφού η καθημερινή χρήση τους αλλά και η συνεχής σύνδεση τους στο διαδίκτυο είναι πλέον τόσο διαδεδομένη. Επίσης, οι έξυπνες κινητές συσκευές προσφέρουν μία εξαιρετική πλατφόρμα η οποία επεκτείνει τις υφιστάμενες crowdsourcing εφαρμογές σε μεγαλύτερο πλήθος, γεγονός που κάνει ευκολότερη και συνεχή την συμβολή τους στην επίλυση προβλημάτων. Επιπρόσθετα, οι πολλαπλές δυνατότητες έξυπνης αναζήτησης που παρέχονται από τα έξυπνα κινητά, όπως για παράδειγμα η γεωγραφική τοποθεσία και οι οπτικοί αισθητήρες, παρέχουν πολλαπλές δυνατότητες για νέες εφαρμογές crowdsourcing.

Για την επίτευξη τέτοιων καθηκόντων υπάρχουν αρκετές Crowdsource υπηρεσίες μεγάλης κλίμακας όπως είναι οι: *Amazon Mechanical Turk (AMT)* και το *oDesk* [27] οι οποίες έχουν ως σκοπό να επιλύουν εργασίες που είναι απλό να αναγνωριστούν από τους χρήστες αλλά εμφανώς δύσκολο να επιλυθούν από αλγορίθμους. Οι υπηρεσίες αυτές είναι υπεύθυνες για εργασίες που αφορούν εξόρυξη δεδομένων, επιμέλεια και διαχείριση επιχειρήσεων και υποστηρίζονται από χιλιάδες άτομα που δουλεύουν καθημερινά στις επιχειρήσεις αυτές. Περισσότερες πληροφορίες για τις υπηρεσίες αυτές θα παρουσιάσουμε στις επόμενες παραγράφους.

Το *Amazon Mechanical Turk (AMT)*[26] είναι μία δικτυακή αγορά πληθοπορισμού η οποία επιτρέπει στους προγραμματιστές υπολογιστών, γνωστοί και ως αιτητές, να συντονίσουν τη χρήση της ανθρώπινης νοημοσύνης έτσι ώστε να εκτελέσουν καθήκοντα που οι υπολογιστές δεν είναι σε θέση να εκτελέσουν. Πρόκειται για μία από τις υπηρεσίες του *Amazon Web Services* όπου οι αιτητές μπορούν να κοινοποιήσουν κάποια tasks, γνωστά και ως *HITs (Human Intelligence Tasks),*. Παράδειγμα κάποιου προβλήματος είναι να επιλεγεί η καλύτερη φωτογραφία μέσα από μία συλλογή η οποία αφορά την πρόσοψη κάποιου καταστήματος, να γραφτούν κάποιες περιγραφές ή να εντοπιστούν οι καλλιτέχνες σε ένα εξώφυλλο ενός δίσκου. Οι εργαζόμενοι του AMT μπορούν στη συνέχεια να περιηγηθούν στα διάφορα υπάρχον tasks και να συμπληρώσαν όποια αυτοί επιθυμούν κερδίζοντας κάποιο συμβολικό ποσό, το οποίο έχει καθοριστεί από τους αιτητές. Για να εισαχθεί κάποιο HIT υπάρχει ο περιορισμός ότι οι αιτητές θα πρέπει να βρίσκονται εντός των ΗΠΑ. Οι αιτητές μπορούν να ζητήσουν από τους εργαζόμενους να δηλώσουν τα προσόντα τους προτού εμπλακούν στην ολοκλήρωση μίας εργασίας, όπως επίσης και να προβούν σε μία δοκιμασία για

εξακρίβωση των συγκεκριμένων προσόντων. Ακόμη, οι αιτητές μπορούν να αποδεχθούν ή να απορρίψουν τα αποτελέσματα που έχουν παρθεί από τους εργαζόμενους.

Το *oDesk* (σύντομη έκδοση του "online desk" αφού επιτρέπει σε οποιονδήποτε να εργάζεται οπουδήποτε, οποτεδήποτε) είναι μία εταιρεία με παγκόσμια αγορά εργασίας η οποία περιλαμβάνει μία σειρά από εργαλεία που απευθύνονται σε επιχειρήσεις οι οποίες προτίθενται να προσλάβουν απομακρυσμένους εργαζόμενους. Η εταιρεία επιτρέπει στους πελάτες της να δημιουργήσουν online ομάδες εργαζομένων οι οποίοι συντονίζονται και πληρώνονται μέσω του ιδιόκτητου λογισμικού της εταιρείας και της ιστοσελίδας. Οι υποψήφιοι πελάτες μπορούν να δημοσιεύσουν θέσεις εργασίας δωρεάν, όπως επίσης και οι διάφοροι ανεξάρτητοι συνεργάτες που ενδιαφέρονται να εργαστούν μπορούν να δημιουργήσουν το δικό τους προφίλ και να δηλώσουν ενδιαφέρον για κάποια θέση εργασίας, επίσης δωρεάν.

Επιπρόσθετα, μία Crowdsourcing δικτυακή αγορά επιτρέπει στους προγραμματιστές Η/Υ να συντονίζουν την χρήση της ανθρώπινης νοημοσύνης έτσι ώστε ομάδες ανθρώπων να εκτελούν καθήκοντα που οι υπολογιστές δεν είναι ακόμη σε θέση να εκτελέσουν. Οι αιτητές, δηλαδή τα άτομα που ενδιαφέρονται να βγάλουν κάποια συμπεράσματα για θέματα που τα ενδιαφέρουν, μπορούν να υποβάλουν εργασίες (tasks) γνωστά ως HITs (Human Intelligence Tasks) όπως για παράδειγμα να ψηφιστεί η καλύτερη φωτογραφία για εξώφυλλο ενός βιβλίου ανάμεσα σε αρκετές φωτογραφίες ή να περιγραφούν αντικείμενα τα οποία παρουσιάζονται σε κάποια φωτογραφία. Αφότου ανατεθούν οι διάφορες εργασίες από τους αιτητές, οι εργαζόμενοι, δηλαδή οι ομάδες ανθρώπων που προαναφέραμε, καλούνται να περιηγηθούν ανάμεσα στις διάφορες εργασίες και να απαντήσουν επιλέγοντας όποιες θέλουν και να κερδίσουν για κάθε απάντηση τους ένα συμβολικό ποσό το οποίο κυμαίνεται μεταξύ $0.01-$0.02 (ποσό που έχει προκαθοριστεί από τον αιτείτε).

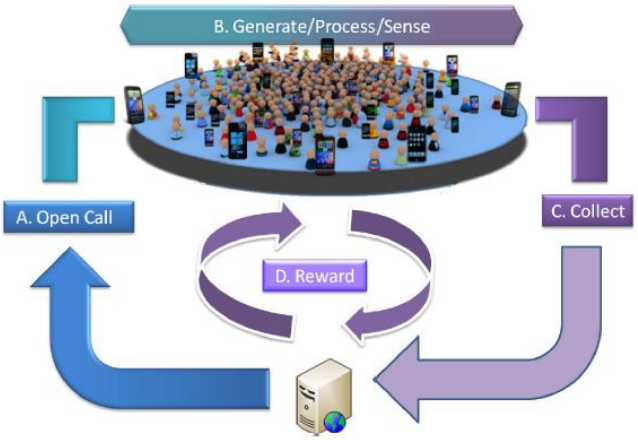
Κάποια παραδείγματα από συστήματα πληθοπορισμού σε κινητές συσκευές έχουν ήδη αναπτυχθεί από φοιτητές του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου. Συγκεκριμένα, το σύστημα SmartTrace εντοπίζει και μας αναφέρει τους χρήστες τους οποίους κινούνται παρόμοια προς το Q, όπου το Q είναι κάποια ερωτήματα ίχνους όπως είναι η τροχιά για παράδειγμα. Σε υπαίθριους χώρους η αναζήτηση γίνεται μέσω του GPS ενώ σε εσωτερικούς χώρους γίνεται με τη χρήση Wi-Fi, χωρίς βέβαια να γίνονται

γνωστά τα ίχνη των χρηστών που συμμετέχουν στην αναζήτηση. Το γεγονός αυτό έχει ως πλεονέκτημα την εξοικονόμηση ενέργειας και το σεβασμό απορρήτου προς τους χρήστες. Ένα άλλο σύστημα είναι το σύστημα SmartP2P και επιτρέπει αναζήτηση υψηλής ποιότητας και κοινοποίηση δεδομένων μεταξύ χρηστών οι οποίοι συμμετέχουν σε κάποιο κοινωνικό δίκτυο.

Σχήμα 1.1.1: SmartTrace[5], Το *SmartTrace* σύστημα δίνει τη δυνατότητα αναζήτησης όμοιων τροχιών ανάμεσα σε χρήστς έξυπνων συσκευών. Απαντάει σε ερωτήματα της μορφής: *"Ανάφερε τους χρήστες που κινούνται παρόμοια ως προς το* Q*"*, όπου το Q είναι ένα ερώτημα τροχιάς. Βελτιστοποιεί τέτοιου είδους ερωτήματα σε σχέση με το χρόνο απόκρισης και την κατανάλωση ενέργειας στις έξυπνες κινητές συσκευές, χωρίς να διαμοιράζονται οι προσωπικές τροχιές με τον επεξεργαστή ερωτημάτων. (a) Το μοντέλο του συστήματος. (b) Στιγμιότυπα οθόνης του SmartTrace+ χρήστη για εξωτερικούς χώρους με τη χρήση GPS και για εσωτερικούς χώρους με τη χρήση σημάτων RSS.

1.2 Υποκίνηση Εργασίας

Σήμερα, η χρήση των έξυπνων κινητών συσκευών[11] ολοένα και αυξάνεται καθώς συνεχώς η αγορά «βομβαρδίζεται» από νέα μοντέλα που χρησιμοποιούν το λειτουργικό σύστημα Android, με αποτέλεσμα οι πωλήσεις του εν λόγω λειτουργικού συστήματος να έχουν πολλαπλασιαστεί και πλέον να είναι το λειτουργικό για κινητές συσκευές με τις μεγαλύτερες πωλήσεις παγκόσμια. Επίσης, η διάδοση του διαδικτύου και η συνεχής και απρόσκοπτη πρόσβαση με σχεδόν αμελητέο κόστος έχει γίνει συνήθης φαινόμενο



Σχήμα 1.1.2: Πληθοπορισμός με έξυπνες συσκευές [5], Ένα πλήθος ατόμων που είναι κάτοχοι έξυπνων συσκευών κινούνται συνεχώς και αισθάνονται, γεγονός που τους παρέχει αρκετά ευκαιριακά δεδομένα τα οποία ενεργοποιούν νέες υπηρεσίες και εφαρμογές.

στα κινητά τηλέφωνα της εποχής μας και κάνει τη χρήση τους για αναζητήσεις αλλά και γενικότερα όσο αφορά τις υπόλοιπες διαδικτυακές υπηρεσίες που παρέχουν, πολύ δημοφιλή. Για τους λόγους αυτούς παρατηρείται μεγάλη αύξηση όσο αφορά τη δημιουργία εφαρμογών για τέτοιες συσκευές κι έτσι η Android "κοινότητα" αποτελείται από πολλούς προγραμματιστές οι οποίοι γράφουν διάφορες εφαρμογές που επεκτείνουν την λειτουργικότητα των εν λόγω συσκευών και συνάμα του λειτουργικού συστήματος. Συγκεκριμένα, υπάρχουν ήδη πολλές χιλιάδες τέτοιων εφαρμογών οι οποίες είναι διαθέσιμες στο κοινό από το Android Market[16], το οποίο είναι ένα διαδικτυακό αποθετήριο εφαρμογών το οποίο διαχειρίζεται η Google και εκεί μπορεί κάθε εγγεγραμμένος χρήστης να βρει όλες τις υπάρχουσες εφαρμογές δωρεάν ή με κάποια μικρή χρέωση.

Η εφαρμογή SmartScan παρέχει τη δυνατότητα σάρωσης κωδικών των βιβλίων και μπορεί να χρησιμοποιηθεί από οποιονδήποτε χρήστη κατέχει Android έξυπνη κινητή συσκευή ούτως ώστε να μπορεί να δημιουργήσει ένα ιστορικό, με όλα τα βιβλία που έχει διαβάσει ή απλά του ανήκουν, με τη βοήθεια της κινητής του συσκευής. Βασική προϋπόθεση, όπως και για τις πλείστες εφαρμογές που απευθύνονται σε έξυπνες κινητές συσκευές, είναι η πρόσβαση στο διαδίκτυο. Επίσης, η δυνατότητα που παρέχει η εφαρμογή στο χρήστη να προσφέρει τον κατάλογο των βιβλίων του σε όλους τους υπόλοιπους χρήστες της συγκεκριμένης εφαρμογής με σκοπό να δημιουργηθεί και να

επεκταθεί μία νοητή ηλεκτρονική βιβλιοθήκη μπορεί να χαρακτηριστεί και σαν ένα είδος εθελοντισμού.

1.3 Σχετικά άρθρα και Μελέτη

Το Crowdsearch είναι η αξιοποίηση πλήθους ατόμων για ακριβή και πραγματικού χρόνου αναζήτηση εικόνων σε κινητά τηλέφωνα [1]. Χιλιάδες άνθρωποι απαντούν σε απλά tasks εικόνων, που είναι αναρτημένα στο *Amazon Mechanical Turk*, για μικρά χρηματικά έπαθλα. Η εκτέλεση ερώτησης-απάντησης-επεξεργασίας γίνεται σε συνδυασμό με την τοπική επεξεργασία στην κινητή συσκευή και την έμμεση επεξεργασία σε απομακρυσμένους εξυπηρετητές. Τα αποτελέσματα βάση των απαντήσεων για κάθε task χαρακτηρίζονται ως έγκυρα ή ως άκυρα μέσω κάποιου αλγορίθμου. Αξιοσημείωτο είναι το γεγονός ότι οι αναζητήσεις από τα κινητά μας τηλέφωνα έχουν γίνει πλέον πολύ δημοφιλής λόγω της σχεδόν συνεχής παροχής του ασύρματου δικτύου, με μειονεκτήματα ότι ίσως οι ασύρματες συνδέσεις να είναι πιο αργές, να υπάρχει σπατάλη ενέργειας και λιγότερη ακρίβεια στην απεικόνιση εικόνων. Λύση στα συγκεκριμένα μειονεκτήματα θα μπορούσε να ήταν η προσφορά μεγαλύτερου χρηματικού επάθλου για πιο γρήγορη ανταπόκριση ή ακόμη και ο συνδυασμός παράλληλης και σειριακής ανάρτησης εικόνων, γεγονός που επιτυγχάνει μικρότερη καθυστέρηση και κόστος. Επίσης, για να παρθεί το επιθυμητό αποτέλεσμα για κάποιο task χρειάζονται πολλές απαντήσεις, με σκοπό να μαζευτούν πολλά διπλότυπα απαντήσεων. Γενικά το Crowdsearch περιλαμβάνει 3 συστατικά αρχιτεκτονικής (κινητό τηλέφωνο, δυνατό απομακρυσμένο server και ένα Crowdsourcing σύστημα το οποίο επικυρώνει τα αποτελέσματα αναζήτησης εικόνων), 3 συστατικά πριν την έναρξη αναζήτησης (εικόνα για εύρεση, χρονική προθεσμία εύρεσης εικόνας και μηχανισμό πληρωμής για όσους χρήστες επικυρωθούν). Οι εικόνες που ανατίθενται για εύρεση είναι κυρίως πρόσωπα ανθρώπων, λουλούδια, κτίρια και εξώφυλλα βιβλίων.

Η Crowdsourced ανίχνευση ομοιότητας των τροχιών σε έξυπνα τηλέφωνα [1] είναι μία εφαρμογή η οποία έχει ήδη αναπτυχθεί. Στις μέρες μας τα έξυπνα τηλέφωνα είναι εξοπλισμένα με πολλών ειδών αισθητήρες, όπως για παράδειγμα Wi-Fi και GPS, γεγονός που δίνει στους χρήστες τους την δυνατότητα να ενασχολούνται με υπηρεσίες επίλυσης αποστολών που εκτελούνται από πλήθος. Το πλήθος έχει ως κίνητρο κάποιο

συμβολικό χρηματικό έπαθλο ή την ηθική. Για την υλοποίηση της εφαρμογής αυτής χρησιμοποιήθηκαν 25 Smartphones μέσω του SmartLab, σύστημα που έχει ήδη αναπτυχθεί και παρέχει ενοικίαση κινητών συσκευών εξ' αποστάσεως. Συγκεκριμένα, η εφαρμογή αναζήτησης ομοιοτήτων ονομάζεται SmartTrace (Σχήμα 1.1.1) και μας αναφέρει τους χρήστες οι οποίοι κινούνται παρόμοια προς το Q, όπου το Q είναι κάποια ερωτήματα ίχνους (π.χ. τροχιά). Σε υπαίθριους χώρους η αναζήτηση γίνεται μέσω του GPS ενώ σε εσωτερικούς χώρους με τη χρήση Wi-Fi, χωρίς βέβαια να γίνονται γνωστά τα ίχνη των χρηστών που συμμετέχουν στην αναζήτηση. Το γεγονός αυτό έχει ως πλεονέκτημα την εξοικονόμηση ενέργειας και το σεβασμό απορρήτου προς τους χρήστες.

Crowdsourced βάσεις δεδομένων: ερωτήματα που τα επεξεργάζονται άνθρωποι [2]. Οι Crowdsourced βάσεις δεδομένων χρησιμοποιούνται ευρέως σε tasks τα οποία είτε είναι ευκολότερο να επιλυθούν από ανθρώπους παρά μηχανές, είτε δεν υπάρχουν ακόμα αποτελεσματικοί αλγόριθμοι τεχνητής νοημοσύνης για την επίλυση τους. Η υπηρεσία *Amazon Mechanical Turk* επιτρέπει στους χρήστες να δημοσιεύσουν μικρά tasks (HITs) τα οποία κάποιοι άλλοι χρήστες καλούνται να ολοκληρώσουν λαμβάνοντας ως βραβείο κάποιο συμβολικό χρηματικό ποσό ($.01-$.02). Συνήθως τα tasks αυτά αποτελούνται από 2 συλλογές εικόνων και οι χρήστες καλούνται να αναγνωρίσουν ποια εικόνα εμφανίζεται και στις 2 συλλογές. Ένα άλλο είδος task είναι τα συναισθήματα του χρήστη διαβάζοντας κάποιο συγκεκριμένο απόσπασμα κειμένου (σύγκριση 2 κειμένων). Ο χρόνος εκτέλεσης κάθε HIT εξαρτάται από το χρόνο εξέτασης του χρηματικού κόστους, το ποιος θα εισάγει κάποιο task, το ποιο HIT να επιλεγεί από το σύστημα ανάμεσα σε πολλά και το πόσα HITs να δημοσιευτούν για 1 συγκεκριμένο task. Για την επίλυση των πιο πάνω υπάρχει το Qurk (ένα crowdworker - aware σύστημα βάσης) που χρησιμοποιεί τη γλώσσα του MTurk. Στο μέλλον, για καλύτερα αποτελέσματα ένα ερώτημα Qurk θα μπορεί να προσδιοριστεί βάση διαφόρων παραγόντων όπως είναι το maxCost (το μέγιστο κόστος το οποίο επιθυμεί να προσφέρει αυτός που θέτει το ερώτημα), το minConfidence (ο ελάχιστος αριθμός εργαζομένων που πρέπει να συμφωνήσουν πριν την αποδοχή της απάντησης) και το maxLatency (το μέγιστο χρονικό διάστημα που θα περιμένει ο χρήστης έως ότου ολοκληρωθεί το HIT).

Απαντήσεις ερωτημάτων με την συνεισφορά ανθρώπων, αλγορίθμων και βάσεων δεδομένων [3]. Υπάρχουν υπηρεσίες μεγάλης κλίμακας όπως η Mechanical Turk,

oDesk και SamaSource οι οποίες έχουν ως σκοπό να επιλύουν tasks τα οποία είναι απλό να αναγνωριστούν από τους ανθρώπους χρήστες αλλά εμφανώς δύσκολο να επιλυθούν από αλγόριθμους. Οι υπηρεσίες αυτές είναι υπεύθυνες για tasks που αφορούν εξόρυξη δεδομένων, επιμέλεια και διαχείριση επιχειρήσεων κι έχουν μαζέψει χιλιάδες ατόμων γι' αυτή τη δουλειά. Για την δημιουργία και τη διαχείριση των tasks έχουν αναπτυχθεί βιβλιοθήκες από την προγραμματιστική κοινότητα. Οι μέθοδοι προγραμματισμού εξακολουθούν να μην μπορούν να ανταπεξέλθουν στην αυξανόμενη πολυπλοκότητα των επιχειρησιακών εφαρμογών και για το λόγο αυτό υποστηρίζεται ότι οι διάφορες υπηρεσίες πρέπει να συνδυάζουν πληροφορίες από βάσεις δεδομένων, ανθρώπινο λογισμό και αλγορίθμους. Συγκεκριμένα, στη βάση δεδομένων είναι αποθηκευμένες όλες οι πληροφορίες που χρειάζονται π.χ. εικόνες που θα χρησιμοποιηθούν σε κάποιο task, ο ανθρώπινος λογισμός περιλαμβάνει τις απαντήσεις στα tasks δίνονται από τους ανθρώπους χρήστες και με τη χρήση αλγορίθμων υποθέτουμε ότι έχουμε μια εκτίμηση σχετικά με την αξιοπιστία των απαντήσεων που έχουν δοθεί. Τέλος, τα κριτήρια για την ολοκλήρωση ενός task είναι το χρηματικό έπαθλο, το πλήθος των ανθρώπων που θα το επιλύσουν και τα κριτήρια που πρέπει να πληρεί ο χρήστης για να το επιλύσει.

Μία πλατφόρμα για την αξιολόγηση αλγορίθμων εντοπισμού της θέσης των δακτυλικών αποτυπωμάτων σε Android έξυπνα κινητά [4]. Στις μέρες μας, λόγω του γεγονότος ότι υπάρχει παντού το διαδίκτυο, κινούν το ενδιαφέρον οι εφαρμογές που απαιτούν τη θέση εντοπισμού, π.χ. GPS, οι οποίες δεν είναι διαθέσιμες σε εσωτερικούς χώρους. Για το λόγο αυτό αναπτύχθηκε μία εφαρμογή βασισμένη στο WLAN η οποία λαμβάνονται τα RSS (Received Signal Strength) δακτυλικά αποτυπώματα για τον προσδιορισμό θέσης. Στην υλοποίηση της συγκεκριμένης εφαρμογής υπάρχουν περιορισμοί όπως είναι η ακρίβεια, ο χρόνος εκτίμησης της θέσης και η χαμηλή κατανάλωση ενέργειας. Συγκεκριμένα, όταν κάποιος χρήστης εισέρχεται σε μια κλειστή περιοχή καλυμμένη από WLAN Access Points, π.χ. Πανεπιστημιούπολη, επικοινωνεί με ένα συγκεκριμένο server για να λάβει το RSS radiomap και τις παραμέτρους του αλγόριθμου (με σεβασμό στην ιδιωτικότητα).

Ο πληθοπορισμός σε συνδυασμό με τις έξυπνες κινητές συσκευές [5]:Οι έξυπνες συσκευές μπορούν να αναπτύξουν πλήρως το δυναμικό του πληθοπορισμού επιτρέποντας στους χρήστες τους να συμβάλουν με σαφήνεια στην επίλυση περίπλοκων προβλημάτων. Ήδη έχουν αναπτυχθεί αρκετές εφαρμογές σε έξυπνες συσκευές οι

οποίες υποστηρίζουν πληθοπορισμό, εφαρμογές που αναπτύσσονται παράλληλα με το αναδυόμενο πεδίο πληθοπορισμού στις έξυπνες συσκευές. Κάποιες από αυτές τις εφαρμογές τις έχουμε ήδη προαναφέρει, όπως για παράδειγμα η εφαρμογή SmartTrace και η εφαρμογή SmartP2P. Μία άλλη εφαρμογή είναι αυτή του Crowdcast η οποία επιτρέπει τον υπολογισμό των πλησιέστερων γειτόνων μας ανά πάση στιγμή με βάση την τοποθεσίας μας. Για το σκοπό εξακρίβωσης της εγκυρότητας εφαρμογών πληθοπορισμού έχει επίσης αναπτυχθεί μία πειραματική πλατφόρμα δοκιμών Android έξυπνων συσκευών, και βρίσκεται στο κτίριο όπου στεγάζεται το Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου. Πιο αναλυτικά, το SmartLab παρέχει μία δημόσια μόνιμη πλατφόρμα δοκιμών για την ανάπτυξη και δοκιμή εφαρμογών σε δίκτυα έξυπνων συσκευών και αποτελείται από περισσότερες των 40 Android έξυπνων κινητών συσκευών.

1.4 Περιγραφή Εργασίας

Στο πρώτο κεφάλαιο παρουσιάστηκε το βασικό υπόβαθρο για το Crowdsourcing αλλά και για το πως αυτός υποβοηθά στην διεκπεραίωση διάφορων εργασιών από ομάδες εθελοντών. Επίσης, αναφερθήκαμε στο πως το Crowdsourcing υιοθετήθηκε από την εφαρμογή την οποία προτείνουμε με στόχο να αναπτύξουμε μια όσο το δυνατό πιο πλήρη νοητή ηλεκτρονική βιβλιοθήκη η οποία σε πρώτο στάδιο θα μπορούσε να χρησιμοποιηθεί από ακαδημαϊκά ιδρύματα όπως το Πανεπιστήμιο Κύπρου. Η νοητή ηλεκτρονική βιβλιοθήκη εκτός από πλήρης θα μπορούσε να χαρακτηριστεί και αυτό-επεκτάσιμη, αυτό-συντηρήσιμη αφού το έργο της συντήρησης και επέκτασης της αναλαμβάνουν εξολοκλήρου οι χρήστες αλλά και να παρέχει κάποιου είδους αυτόματη αξιολόγηση για τα βιβλία τα οποία περιέχει. Το μακροπρόθεσμο όραμα μας περιλαμβάνει την επέκταση του συστήματος σε παγκόσμιο επίπεδο.

Στο δεύτερο κεφάλαιο της εργασίας αυτής θα παρουσιάσουμε σχετική δουλειά που έχει γίνει όπως επίσης και το τεχνολογικό υπόβαθρο που χρησιμοποιήθηκε για τη δημιουργία του Smartscan (εφαρμογή και ιστοσελίδα).

Στο τρίτο κεφάλαιο θα παρουσιάσουμε την αρχιτεκτονική τόσο της εφαρμογής Android αλλά και της νοητής ηλεκτρονική βιβλιοθήκης καθώς και τις διάφορες τεχνολογίες που έχουμε χρησιμοποιήσει.

Στο τέταρτο κεφάλαιο θα παρουσιάσουμε και θα επεξηγήσουμε το γραφικό περιβάλλον της εφαρμογής αλλά και της ιστοσελίδας. Θα γίνει χρήση στιγμιότυπων οθόνης και θα αναφερθούμε εκτενώς στις λειτουργίες που παρέχονται προς τους απλούς χρήστες αλλά και προς του χρήστες μέλη.

Στο πέμπτο κεφάλαιο θα παρουσιάσουμε την πειραματική διαδικασία που ακολουθήθηκε έτσι ώστε να μπορέσουμε να καταχωρήσουμε δεδομένα στη βάση δεδομένων όπως επίσης τις μετρήσεις που πάρθηκαν για να μπορεί να αξιολογηθεί η εγκυρότητα του Smartscan συστήματος.

Κεφάλαιο 2

Σχετική δουλειά και Τεχνολογικό Υπόβαθρο

Στο κεφάλαιο αυτό θα αναφερθούμε σε όλους τους τεχνικούς όρους που αφορούν το αντικείμενο που μελετάμε και θα δώσουμε ένα γενικό μοντέλο συστήματος που θα μας βοηθήσει στην κατανόηση της εφαρμογής και της ιστοσελίδας που αναπτύχθηκαν.

2.1 Τεχνολογικό Υπόβαθρο

Smartphones

Τα Smartphones, αποτελούν την φυσική εξέλιξη των κλασικών συσκευών κινητής τηλεφωνίας. Είναι μια συσκευή τηλεπικοινωνίας, η οποία έχει επιπρόσθετα την δυνατότητα να πραγματοποιήσει κάποιες από τις εργασίες που εκτελούν οι προσωπικοί υπολογιστές, όπως την λήψη και αποστολή emails, την επεξεργασία κειμένων κ.λ.π. Τα Smartphones είναι αποτέλεσμα της σύζευξης των κλασσικών κινητών τηλεφώνων με τα Personal Digital Assistants (PDA), τα οποία ήταν στην πράξη ηλεκτρονικές φορητές ατζέντες, που μπορούσαν να επικοινωνήσουν με τον υπολογιστή για ανταλλαγή στοιχείων. Ίσως το κυριότερο χαρακτηριστικό που ξεχωρίζει τα Smartphones, πέραν της εμφάνισης, είναι το λειτουργικό σύστημα που χρησιμοποιούν. Υπάρχει πληθώρα εφαρμογών για το λειτουργικό σύστημα, από παιχνίδια μέχρι εξειδικευμένες υπηρεσίες και πολλές εταιρίες ξεκίνησαν να δημιουργούν ηλεκτρονικά καταστήματα εφαρμογών ειδικά για τα Smartphones (π.χ. Android Market).

Barcodes

Ένα barcode είναι μια αναπαράσταση δεδομένων τα οποία αντιπροσωπεύουν πληροφορίες οι οποίες σχετίζονται με το αντικείμενο στο οποίο επισυνάπτεται το

συγκεκριμένο barcode. Είναι οπτικά αναγνώσιμο από μηχανήματα τα οποία αρχικά ήταν ειδικοί οπτικοί σαρωτές ενώ με την ανάπτυξη της τεχνολογίας τέτοιου είδους λογισμικό σάρωσης είναι διαθέσιμο και σε desktop printers όπως επίσης και σε Smartphones. Με τον όρο barcode καλύπτονται και οι δύο εκδοχές 1D και 2D όπου στην 1D εκδοχή τα barcodes αντιπροσωπεύονταν από δεδομένα βασισμένα στα πλάτη και τα κενά των παράλληλων γραμμών (Σχήμα 2.1.1) ενώ στην μετέπειτα έκδοση 2D τα barcodes εξελίχθηκαν σε ορθογώνια, τελείες και άλλα γεωμετρικά σχήματα σε δύο διαστάσεις (Σχήμα 2.1.2).

Σχήμα 2.1.1 : 1D barcode

Σχήμα 2.1.1 : 2D barcode

Τα barcodes χρησιμεύουν για το σύστημα παρακολούθησης των προϊόντων που αγοράζονται και πωλούνται εντός μίας επιχείρησης, δηλαδή πλέον είναι εφικτό να ελέγχεται το απόθεμα στις διάφορες επιχειρήσεις που διαθέτουν προϊόντα προς πώληση, όπως επίσης και ο έλεγχος για απώλειες προϊόντων π.χ. από κακόβουλη ενέργεια. Με τη χρήση των barcodes στα διάφορα προϊόντα επιτρέπεται να κρατάμε ένα κεντρικό αρχείο σε ένα σύστημα υπολογιστή το οποίο παρακολουθεί τα προϊόντα, τις τιμές και τα επίπεδα αποθεμάτων. Αξίζει να σημειωθεί ότι κάθε barcode χαρακτηρίζει μοναδικά κάθε αντικείμενο, έτσι όποιαδήποτε στιγμή μπορούμε να κρατάμε καταστατικό για το κάθε προϊόν όπως επίσης να αλλάζουμε και τα στοιχεία του π.χ. την τιμή του.

Ένα barcode σύστημα αποθεμάτων όπως αυτό που αναφέραμε πιο πάνω αποτελείται από τρία μέρη:

1. Ένα κεντρικό υπολογιστή ο οποίος λειτουργεί ως βάση δεδομένων (σύστημα καταγραφής) και κρατά ιστορικό για τα όσα προϊόντα είναι προς πώληση: την προέλευση τους, την τιμή τους, πόσα υπάρχουν σε stock κτλ.
2. Τα barcodes είναι εκτυπωμένα σε όλες τις συσκευασίες των προϊόντων.

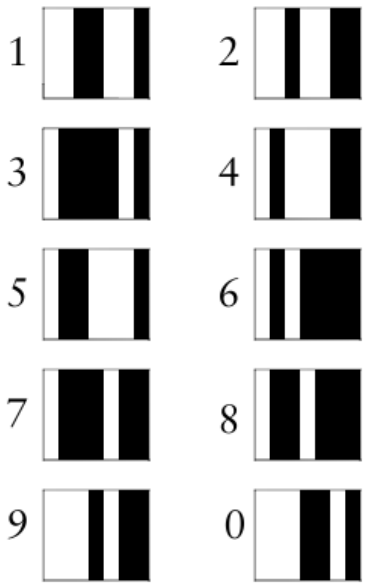
3. Υπάρχουν ένα ή περισσότεροι σαρωτές που διαβάζουν τα συγκεκριμένα barcodes που βρίσκονται στη συσκευασία των προϊόντων.

Ένα barcode είναι μία πολύ απλή ίδεα : δίνουμε ένα μοναδικό αριθμό σε κάθε αντικείμενο που θέλουμε να χαρακτηρίσουμε και μετά πολύ απλά εκτυπώνουμε τον αριθμό στο αντικείμενο κι έτσι μία ηλεκτρονική συσκευή σάρωσης μπορεί να τον διαβάσει. Θα μπορούσαμε πού απλά να εκτυπώνουμε τον αριθμό αυτόν καθεαυτόν στη θέση ενός κωδικοποιημένου αριθμού αλλά πιθανον να υπήρχς πρόβλημα ανάγνωσης από τιν υπολογιστή αν για παράδειγμα ένα ελλειπές τυπωμένο οκτώ έμοιαζε με τον αριθμό τρία ή το γεγονός ότι το έξι είναι πανομοιότυπο με το εννέα αν το αναποδογυρίσουμε. Αυτό που χρειαζόμαστε πραγματικά είναι είναι έναν απόλυτα αξιόπιστο τρόπο εκτύπωσης αριθμών, έτσι ώστε ν αμπορούν να διαβαστούν με μεγάλη ακρίβεια σε υψηλές ταχύτητες. Αυτό το πρόβλημα λοιπόν επιπλύουν τα barcodes.

Εάν κοιτάξουμε ένα barcode τότε μάλλον δεν μπορούμε να καταλάβουμε αν βλέπουμε την αρχή ή το τέλος του, ούτε το τι αντιπροσωπεύει αφού δεν γνωρίζουμε που αρχίζει και που τελειώνει. Για κάθε ψηφίο στον κωδικό ενός προϊόντος δίνεται το ίδιο ποσό οριζόντιου χώρου: ακριβώς 7 τμήματα. Στη συνέχεια, για να αναπαραστήσουμε οποιονδήποτε αριθμό από το μήδέν εώς το εννέαμ μπορούμε απλά να χρωματίσουμε αυτά τα επτά τμήματα με διαφορετικό πρότυπο από μάυρες και άσπρες γραμμές. Για παράδειγμα, ο αριθμός ένα αντιπροσωπεύεται από δύο άσπρες και δύο μαύρες γραμμές, ο αριθμός δύο από δύο άσπρες γραμμές, μία μαύρη γραμμή, δύο άσπρε γραμμές και τέλος μία μάυρη γραμή κ.ο.κ (η αναπαράσταση αυτή φαίνεται στο Σχήμα 2.1.3).

Τα barcodes είναι αρκετά μεγάλα διά τον λόγο ότι πρέπει να αντιπροωπεύουν τρεις διαφορετικούς τύπος πληροφοριών. Συγκεκριμένα, το πρώτο κομμάτι ενός barcode αντιπροσωπεύει την χώρα στην οποία εκδόθηκε το προϊόν, το δεύτερο μέρος

αποκαλύπτει τον κατασκευαστή του προϊόντος ενώ το τελευαίο μέρος ταυτοποιεί το συγκεκριμένο προϊόν.Διαφορετικοί τύποι του ίδιου βασικού προϊόντος μπορεί να έχουν εντελώς διαφορετικό αριθμό barcode. Για παράδειγμα, μία τετράδα από μπουκάλια από αναψυκτικά μάρκας Α με μία εξάδα από τενεκεδάκια από αναψυκτικά της ίδιας μάρκας Α.



Σχήμα 2.1.3: Παρουσίαση barcodes

Κάθε ψηφίο αναπαρηστάται από εφτά ισομεγέθη κάθετα μπλοκς. Τα μπλοκς αυτά είναι χρωματισμένα είτε σε άσπρο είτε σε μαύρο χρώμακαι αντιπροσωπεύουν τους αριθμούς 0-9. Κάθε μπλοκ έχει σχεδιαστεί έτσι ώστε αν γυρίσουμε ανάποδα το barcode δεν μπορεί να συνγχιστεί με κάποιο αλλό.

Τα περισσότερα προϊόντα που έχουν ένα από barcode, γνωστό και ως UPC (Universal Product Code) – μία γραμμή από κάθετες γραμμές που περιέχει ένα σύνολο αριθμών στο κάτω μέρος ούτως ώστε κάποιο να μπορεί να καταχωρήσει δια χειρός τον κωδικό αυτό στην περίπτωση που το barcode δεν είναι σωστά εκτυπωμένο ή είναι κατεστραμμένο και έτσι δεν μπορεί να διαβαστεί από κάποια ηλεκτρονική μηχανή σάρωσης. Υπάρχει και μία άλλου είδους μορφή barcode η οποία ολοένα και πιο κοινή αφού περιέχει πολύ περισσότερες πληροφορίες. Η μορφή αυτή ονομάζεται δισδιάστατη (2D barcode).

Δε θα ήταν καλό να έχουμε barcodes εάν δεν είχαμε την τεχνολογία με την οποία να μπορούμε να τα διαβάσουμε/σαρώσουμε. Τα barcode scanners πρέπει να είναι ικανά να διαβάζουν εξαιρετικά γρήγορα τα μαυρόασπρα zebra lines πάνω στα προϊόντα και να

τροφοδοτούν τις πληροφορίες αυτές σε computer ή σε σημεία ελέγχου τα οποία θα μπορούν να τα αναγνωρίσουν χρησιμοποιώντας μια βάση δεδομένων των προϊόντων. Τώρα θα εξηγήσουμε πως επιτυνγ΄χανεται αυτό:

Για χάριν αυτού του απλού παραδείγματος υποθέτουμε ότι τα barcodes είναι απλά on – off , binary patterns με κάθε μαύρη γραμμή να αντιστοιχεί σε ένα (1) και κάθε άσπρη γραμμή σε μηδέν (0). (Έχουμε ήδη δεί ότι τα πραγματικά barcodes είναι πιο εξελιγμένα από αυτό, αλλά ας κρατήσουμε τα πράγματα απλά) - Σχήμα 2.1.4

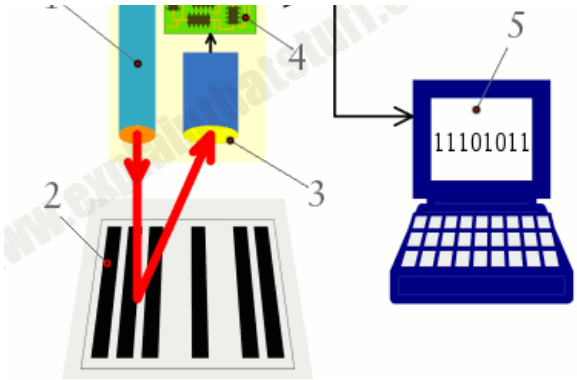
1. Ακτίνες σάρωσης LED ή laser light πάνω στο barcode.

2. Το φως αντικατοπτρίζει πίσω στο barcode σε ένα ηλεκτρονικό εξάρτημα εντοπισμού φωτός που λέγεται photoelectric cell. Οι άσπρες περιοχές του barcode αντανακλούν περισσότερο φως, οι μαύρες περιοχές αντανακλούν λιγότερο.

3. Όπως το scanner προχωρά πάνω στο barcode, το cell δημιουργεί ένα μοτίβο από on-off pulses παλμούς τα οποία αντιστοιχούν στις μαύρες και άσπρες ρίγες. Έτσι για το code που φαίνεται εδώ ("μαύρο μαύρο μαύρο άσπρο μαύρο άσπρο μαύρο μαύρο "), το cell θα είναι "off off off on off on off off."

4. Ένα ηλεκτρονικό κύκλωμα συνδεδεμένο στο scanner μετατρέπει αυτά τα on-off παλμούς σε δυαδικά ψηφία (μηδενικά και άσσους).

5. Τα δυαδικά ψηφία αποστέλλονται σε ένα υπολογιστή συνδεδεμένο με το scanner, ο οποίος ανιχνεύει το code σαν 11101011.



Σχήμα 2.1.4: Σαρωτής barcodes

Σε μερικά scanners, υπάρχει ένα απλό photoelectric cell και καθώς μετακινείς την κεφαλή του σαρωτή πάνω από το το προϊόν (ή το προϊόν πάνω σ την κεφαλή του σαρωτή), το cell ανιχνεύει κάθε κομμάτι του μαυρόασπρου barcode στη σειρά. Σε πιο εξελιγμένους σαρωτές υπάρχει μια ολόκληρη γραμμή από photoelectric cells και ολόκληρος ο κωδικός ανιχνεύεται με ένα πέρασμα.

Στην πραγματικότητα, οι σαρωτές δεν ανιχνεύουν μηδενικά και άσσους και παράγουν δυαδικά ψηφία αλλά παράγουν μαύρες και άσπρες ρίγες και τις μετατρέπουν κατευθείαν σε δεκαδικούς αριθμούς, δίνοντας δεκαδικούς αριθμούς σαν έξοδο.

Υπάρχουν διάφορα είδη σαρωτές barcode για όλα τα είδη των εφαρμογών. Σε μικρά περίπτερα θα βρείτε συνήθως ένα basic wand scanner. Τα πιο απλά μοιάζουν με ηλεκτρονικά στυλό ή με γιγάντια, υπερμεγέθη ξυράφια. Φωτίζουν με ένα κόκκινο φως LED πάνω στο μαύρο και άσπρο μοτίβο του barcode και στη συνέχεια, διαβάζουν το μοτίβο της αντανάκλασης του φωτός με ένα φωτοευαίσθητο CCD ή μια σειρά φωτοκυττάρων. Σε ένα πολυσύχναστο υπερκατάστημα, είναι πιο πιθανό να δούμε ένα πολύ εξελιγμένο laser scanner το οποίο θα είναι ενσωματωμένο κάτω από ένα κομμάτι γυαλί, μέσα από το οποίο θα αναπηδούν ακτίνες laser με μεγάλη ταχύτητα ώστε να διαβάζει τα προϊόντα με τη χρήση φλας.

Μια άλλη τεχνολογία χρησιμοποιεί μια μικρή video camera για να λάβει μια άμεση ψηφιακή φωτογραφία του barcode. Μετά, ένας υπολογιστής αναλύει τη φωτογραφία επιλέγοντας μόνο το μέρος του barcode του και μετατρέποντας μόνο το ασπρόμαυρο μοτίβο σε αριθμό. (Barcode-scanning εφαρμογές που τρέχουν σε κινητά τηλέφωνα λειτουργούν με αυτόν τον τρόπο, χρησιμοποιώντας την ενσωματωμένη κάμερα του τηλεφώνου για να φωτογραφίσουν τον κώδικα).Σαρωτές όπως αυτόν μπορούν να διαβάσουν με ακρίβεια δωδεκάδες προϊόντων που μπορεί να περάσουν από μπροστά τους κάθε λεπτό και είναι πολύ πιο ακριβοί από ότι τα παλαιού τύπου ταμεία (όπου θα έπρεπε η τιμή του κάθε προϊόντος να πληκτρολογηθεί με το χέρι).

Η τεχνολογία barcode scanning εμφανίστηκε από τις αρχές της δεκαετίας του 1970, αλλά πραγματικά εξελίχθηκε στη δεκαετία του 1980 και του 1990 μετά που τα καταστήματα άρχισαν να επενδύουν σε εξελιγμένα, με τη βοήθεια ηλεκτρονικού

υπολογιστή, σαρωτών. Τότε, τα ταμεία των καταστημάτων είχαν κοστίσει πολλές χιλιάδες δολάρια ενώ σήμερα, οι τιμές για σαρωτές είναι πολύ πιο προσιτές.

ZXing Βιβλιοθήκη

Η βιβλιοθήκη ZXing (η οποία και προφέρεται "zebra crossing") είναι μία open-source, multi-format 1D/2D barcode βιβλιοθήκη για επεξεργασία εικόνας η οποία είναι υλοποιημένη στη γλώσσα προγραμματισμού Java. Συγκεκριμένα, η βιβλιοθήκη αυτή εστιάζει στη χρήση της ενσωματωμένης κάμερας στα κινητά τηλέφωνα για να ανιχνεύσει και να αποκωδικοποιήσει τα barcodes στη συσκευή, χωρίς όμως να επικοινωνεί με κάποιο διακομιστή. Ωστόσο, το πρόγραμμα μπορεί να χρησιμοποιηθεί για την κωδικοποίηση και αποκωδικοποίηση barcodes σε επιτραπέζιους υπολογιστές και διακομιστές εξίσου. Προς το παρόν υποστηρίζονται οι ακόλουθες μορφές:

- UPC-A and UPC-E
- EAN-8 and EAN-13
- Code 39 ,Code 93,Code 128 ITF
- Codabar
- RSS-14 (all variants)
- QR Code

- Data Matrix
- Aztec ('beta' quality)
- PDF 417 ('alpha' quality)

Σχήμα 2.1.5 : Σχηματικά οι μορφές barcode

Η βιβλιοθήκη ZXing είναι χωρισμένη σε πολλά κύρια στοιχεία τα οποία και υποστηρίζονται ενεργά:

- core: The core image decoding library, and test code
- javase: J2SE-specific client code
- zxingorg: The source behind zxing.org/w
- android: Android client, called Barcode Scanner
- androidtest: Android test application
- android-integration: Supports integration with our Barcode Scanner app via Intent

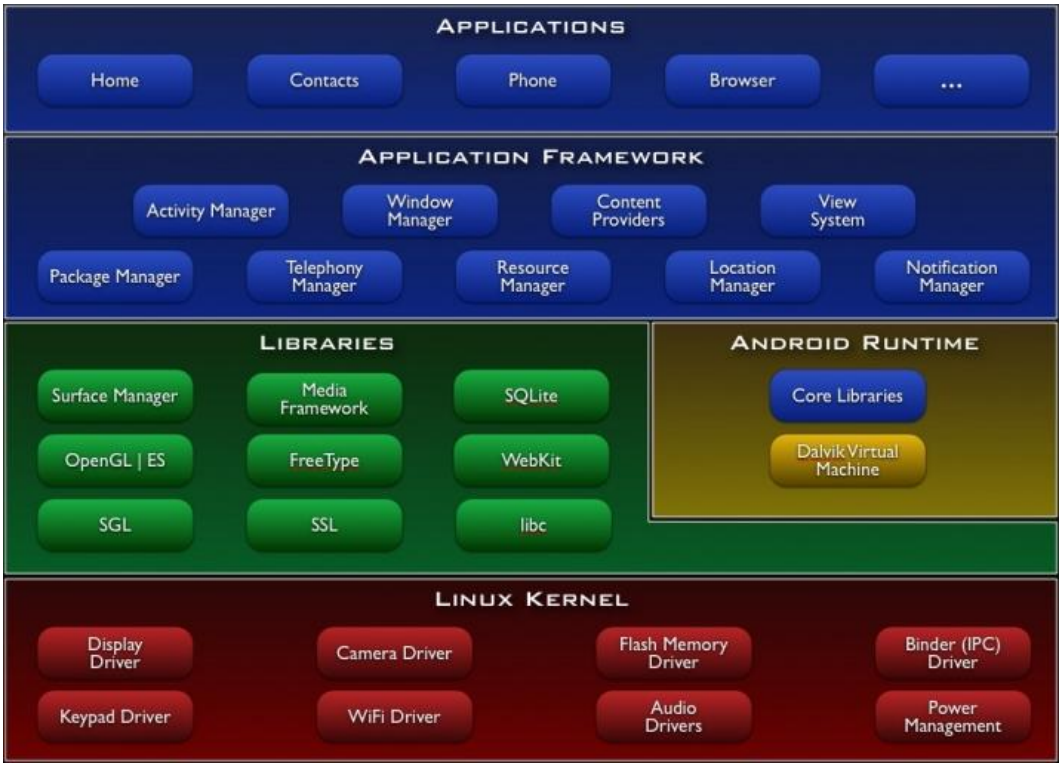
Barcode Scanner by ZXing Team

Η εφαρμογή Barcode Scanner έχει υλοποιηθεί από την ομάδα της ZXing και παρέχει στους χρήστες της τη δυνατότητα να σαρώσουν τα barcodes των προϊόντων (βιβλία, προϊόντα υπεραγοράς κτλ)και στη συνέχει να εξετάσουν τις τιμές τους αλλά και τις διάφορες κριτικές που υπάρχουν online για τα συγκεκριμένα προϊόντα. Είναι μία

εφαρμογή η οποία συνδέεται με την βάση δεδομένων : “Google Mobile Product Search Database” και για το λόγο αυτό μπορεί να παρέχει τις υπηρεσίες που αναφέραμε και πιο πάνω, όπως για παράδειγμα να συγκρίνει τιμές και να βρει αν υπάρχει κάποιο παρόμοιο προϊόν σε καλύτερη τιμή. Δουλεύει καλύτερα με μέσα όπως είναι τα βιβλία και τα dvds αλλά έχει τη δυνατότητα να βρει αποτελέσματα για μεγάλα φάσματα προϊόντων αρκεί να επιλεγεί από το μενού που προσφέρει. Η συγκεκριμένη εφαρμογή μπορεί επίσης να σαρώσει Data Matrix[17] και QR Codes[18] τα οποία μπορεί να περιέχουν διευθύνσεις, τηλέφωνα κτλ.

Android

Το Android είναι μία στοίβα λογισμικού για κινητές συσκευές η οποία περιλαμβάνει ένα λειτουργικό σύστημα, ενδιάμεσο λογισμικό και κάποιες βασικές εφαρμογές. Το Android SDK παρέχει τα εργαλεία και τις απαραίτητες εφαρμογές ανάπτυξης του API (Application Programming Interface) σε μία Android πλατφόρμα η οποία χρησιμοποιεί τη γλώσσα προγραμματισμού Java.



Σχήμα 2.1.6 : Αρχιτεκτονική Android

Το Android αποτελείται από πολλά αναγκαία και ανεξάρτητα μέρη συμπεριλαμβανομένων των ακόλουθων :

- Το σχέδιο ή υπόδειγμα αναφοράς υλικού που περιγράφει τις ικανότητες που απαιτούνται από μια κινητή συσκευή, ώστε να υποστηρίξει το λογισμικό στοίβα
- Ένα Linux πυρήνα του λειτουργικού συστήματος που παρέχει τη διασύνδεση χαμηλού επιπέδου με το υλικό, τη διαχείριση μνήμης, και ελέγχου διεργασιών, όλα βελτιστοποιημένα για φορητές συσκευές
- Βιβλιοθήκες ανοιχτού κώδικα για την ανάπτυξη εφαρμογών, συμπεριλαμβανομένων των SQLite, WebKit, OpenGL, και Media Manager
- Ο χρόνος εκτέλεσης είναι ο χρόνος που χρησιμοποιείται για να εκτελέσει και να φιλοξενήσει τις Android εφαρμογές, συμπεριλαμβανομένης της Dalvik εικονικής μηχανής και τις βιβλιοθήκες των βασικών ικανοτήτων που παρέχουν στο Android συγκεκριμένη λειτουργικότητα. Ο χρόνος εκτέλεσης είναι σχεδιασμένος για να είναι μικρός και αποδοτικός για να χρησιμοποιείται από σε κινητές συσκευές.
- Ένα πλαίσιο εφαρμογών που εκθέτει πραγματικά σύστημα υπηρεσιών για το επίπεδο εφαρμογών, συμπεριλαμβανομένου τον διαχειριστή παραθύρων, content providers, τηλεφωνία και peer-to-peer υπηρεσίες.
- Ένα πλαίσιο για διεπαφή του χρήστη που χρησιμοποιείται για να φιλοξενήσει και να εκκινήσετε εφαρμογές.
- Προεγκατεστημένες εφαρμογές που αποστέλλονται ως μέρος της στοίβας.
- Μια πακέτο ανάπτυξης λογισμικού που χρησιμοποιείται για τη δημιουργία εφαρμογών, συμπεριλαμβανομένων των εργαλείων, plug-ins, και τεκμηρίωσης.

Java Language

Η Java είναι μία γλώσσα προγραμματισμού που κυκλοφόρησε το 1995 και μεγάλο μέρος της σύνταξης της προέρχεται από τις γλώσσες προγραμματισμού C και C++ με τη διαφορά ότι έχει ένα απλούστερο μοντέλο και λιγότερες εντολές χαμηλού επιπέδου. Οι εφαρμογές της Java μπορούν να τρέξουν σε οποιαδήποτε JVM (Java Virtual

Machine) ανεξάρτητα από την αρχιτεκτονική του υπολογιστή. Είναι μία αντικειμενοστρεφής γλώσσα, από τις πιο δημοφιλείς γλώσσες προγραμματισμού σε χρήση - κυρίως για client-server εφαρμογές Web.

JavaScript

JavaScript[12] είναι μία αντικειμενοστρεφής scripting γλώσσα η οποία χρησιμοποιείται παραδοσιακά από τους web browsers έτσι ώστε να παρέχουν πιο πλούσιες διαπροσωπείες χρήστη όπως επίσης και δυναμικές ιστοσελίδες.

jQuery

Το jQuery[13] είναι μία βιβλιοθήκη JavaScript η οποία σχεδιάστηκε για να απλοποιήσει την διαδικασία προγραμματισμού της HTML στην πλευρά client. Σήμερα αποτελεί την πιο δημοφιλή βιβλιοθήκη της JavaScript. Το jQuery είναι δωρεάν λογισμικό ανοικτού πηγαίου κώδικα. Η σύνταξη της εν λόγω βιβλιοθήκης σχεδιάστηκε έτσι ώστε να κάνει πιο εύκολη την μετακίνηση σε ένα έγγραφο, την επιλογή στοιχείων DOM (Document Object Model),τη δημιουργία animation, τον χειρισμό συμβάντων, και την ανάπτυξη AJAX(Asynchronous JavaScript and XML)[14] εφαρμογών. Το jQuery προσφέρει επίσης τη δυνατότητα στους προγραμματιστές να δημιουργήσουν plugins πάνω από τη βιβλιοθήκη της JavaScript.

AJAX

Το AJAX[14] είναι ένα σύνολο αλληλένδετων τεχνικών ανάπτυξης ιστοσελίδων που χρησιμοποιούνται στην πλευρά του πελάτη για τη δημιουργία ασύγχρονων web εφαρμογών. Με τη χρήση AJAX οι web εφαρμογές μπορούν να στέλνουν δεδομένα και να λαμβάνουν δεδομένα από κάποιον διακομιστή ασύγχρονα, χωρίς να παρεμβαίνει με την εμφάνιση και τη συμπεριφορά της υπάρχουσας σελίδας. Τα δεδομένα συνήθως ανακτούνται χρησιμοποιώντας το αντικείμενο XML HttpRequest (χωρίς να σημαίνει απαραίτητα και τη χρήση XML, αλλά συνήθως έχουμε χρήση JSON (JavaScript Object Notation)). Είναι μία ομάδα τεχνολογιών όπου η HTML και τα css μπορούν να χρησιμοποιηθούν σε συνδυασμό με τη σήμανση και την ενημέρωση του στυλ της ιστοσελίδας.

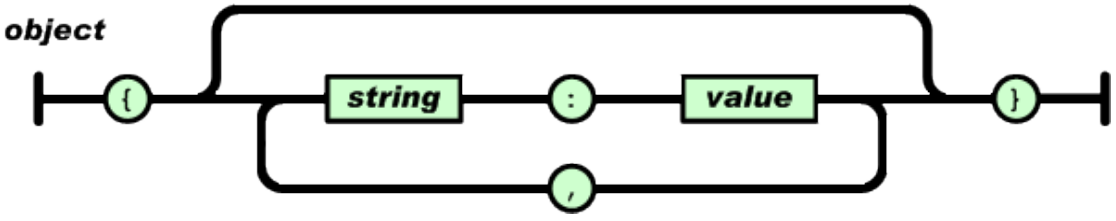
JSON

Το JSON[20] (η συντομογραφία του JavaScript Object Notation) είναι μία γλώσσα ανταλλαγής δεδομένων και χρησιμοποιεί JavaScript σύνταξη για την περιγραφή των αντικειμένων των δεδομένων. Η JSON μορφή συνήθως χρησιμοποιείται για σειριοποίηση και μετάδοση δομημένων δεδομένων μέσω μίας σύνδεσης δικτύου. Χρησιμοποιείται κυρίως για τη μετάδοση δεδομένων μεταξύ ενός διακομιστή και μίας διαδικτυακής εφαρμογής, που χρησιμεύει ως μία εναλλακτική λύση σε XML. Το JSON είναι κτισμένο σε δύο δομές:

Μία συλλογή από ζευγάρια όνομα/τιμή (π.χ. αντικείμενο, δομή δεδομένων κτλ).
Μία διατεταγμένη λίστα τιμών (π.χ. πίνακας).

Το JSON μπορούμε να το βρούμε στις εξής μορφές:

Αντικείμενο (Σχήμα 2.1.7 και Σχήμα 2.1.8) : ένα μη διατεταγμένο σύνολο από ζευγάρια όνομα/τιμή. Ξεκινάει με αριστερό άγκιστρο ({) και τελειώνει με δεξιό άγκιστρο (}). Κάθε όνομα ακολουθείται από άνω και κάτω τελεία (:) και κάθε ζευγάρι όνομα/τιμή διαχωρίζεται με κόμμα (,).
Πίνακας : Μία μη διατεταγένη συλλογή τιμών. Ξεκινάει με αριστερό άγκιστρο ({) και τελειώνει με δεξιό άγκιστρο (}), ενώ οι τιμές διαχωρίζονται με κόμμα (,).
Τιμή : Μπορεί να είναι κάποια συμβολοσειρά σε διπλά εισαγωγικά («»), κάποιος αριθμός,true/false/null, αντικείμενο είτε πίνακας. Αυτές οι δομές μπορούν να είναι φωλιασμένες.
Συμβολοσειρά : Είναι μια ακολουθία από μηδέν ή περισσότερους χαρακτήρες, εγκιβωτισμένοι σε διπλά εισαγωγικά («»).
Αριθμός : παρόμοια σύνταξη με τις υπόλοιπες γλώσσες προγραμματισμού.



Σχήμα 2.1.7 : JSON αντικείμενο

```
{
      "firstName": "John",
      "lastName" : "Smith",
      "age"      : 25,
      "address"  :
      {
          "streetAddress": "21 2nd Street",
          "city"         : "New York",
          "state"        : "NY",
          "postalCode"   : "10021"
      },
      "phoneNumber":
      [
          {
            "type"  : "home",
            "number": "212 555-1234"
          },
          {
            "type"  : "fax",
            "number": "646 555-4567"
          }
      ]
}
```

Σχήμα 2.1.8 : Παράδειγμα για JSON αντικείμενο

jqGrid

Το jqGrid[22] είναι ένα JavaScript το οποίο είναι Ajax-enabled και παρέχει λύσεις όσο αφορά την αναπαράσταση και το χειρισμό ενός πίνακα δεδομένων στον παγκόσμιο ιστό. Δεδομένου ότι το διαδίκτυο είναι μία client-side λύση η οποία φορτώνει δυναμικά μέσω Ajax, μπορεί να ενσωματωθεί με οποιανδήποτε server-side τεχνολογία εκ των οποίων είναι και οι PHP,ASP,Java Servlets, Perl κτλ. Το jqGrid χρησιμοποιεί μία jQuery JavaScript βιβλιοθήκη.

jCarousel

Το jCarousel[23] είναι μία ανοικτή προς το κοινό προσθήκη του jQuery που υλοποιεί ένα μπλοκ κύλισης το οποίο περιέχει εικόνες και κείμενο. Επιτρέπει πλοήγηση η οποία παρέχεται σε ένα κυκλικό τρόπο.

Books APIs

Το API στο Google Books API Family μας επιτρέπει να εκτελέσουμε πλήρης αναζητήσεις και να ανακτήσουμε και να εμφανίσουμε όλες τις πληροφορίες που παρέχονται για ένα βιβλίο μέσω του Google Books, στην ιστοσελίδα ή στην εφαρμογή μας. Μέσω της χρήσης του επιτρέπεται να εκτελέσουμε τις πλείστες από τις λειτουργίες τις οποίες θα μπορούσαμε να κάνουμε διαδραστικά μέσω της Google Books ιστοσελίδας.

Κάποιες από τις κύριες λειτουργίες του Google API είναι:

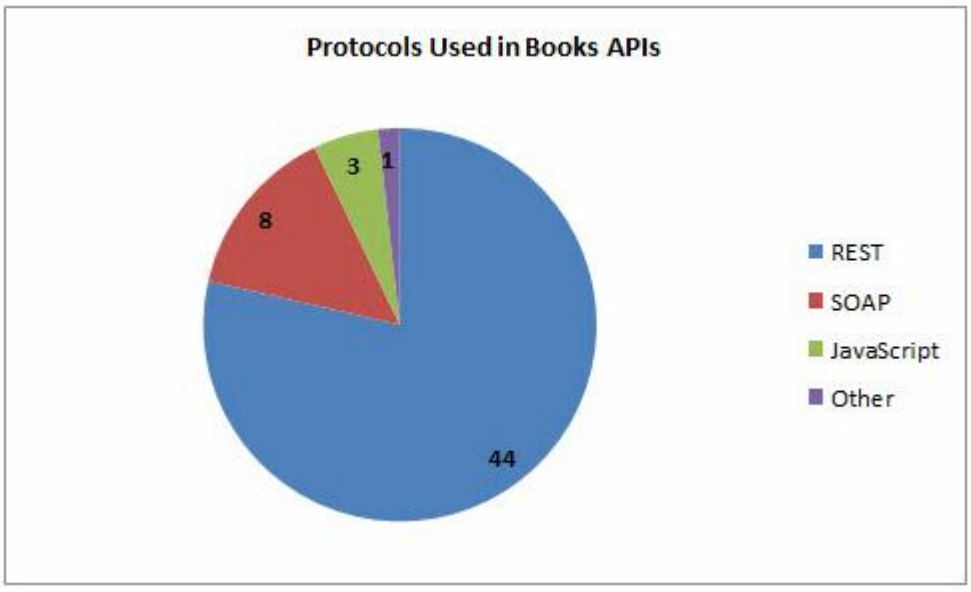
- Αναζήτηση και Πλοήγηση μέσα από μία λίστα βιβλίων με τη χρήση ενός συγκεκριμένου ερωτήματος.
- Εμφάνιση πληροφοριών ενός βιβλίου, συμπεριλαμβανομένου μεταπληροφοριών, διαθεσιμότητα και τιμή, όπως επίσης και προεπισκόπηση βιβλίου.
- Διαχείριση μίας προσωπικής βιβλιοθήκης του χρήστη.

Το Google Books API χρησιμοποιείται από διάφορες ιστοσελίδες και εφαρμογές με δημιουργικούς τρόπους, έτσι ώστε να παρέχονται προς το κοινό οι πληροφορίες οι οποίες μπορούν να ανακτηθούν από το Google Books. Ένα τέτοιο παράδειγμα είναι η πλέον διαδεδομένη ιστοσελίδα "*WorldCat*" που αποτελεί την μεγαλύτερη συνεταιρική βιβλιοθήκη του κόσμου με στόχο να συνδέει τους ανθρώπους με τις βιβλιοθήκες και τις συλλογές τους. Συγκεκριμένα, οι άνθρωποι χρησιμοποιούν τον συγκεκριμένο ιστοχώρο

για να εντοπίσουν, να εξετάσουν, να αξιολογήσουν και να σχολιάσουν τον αντικείμενα της βιβλιοθήκης.

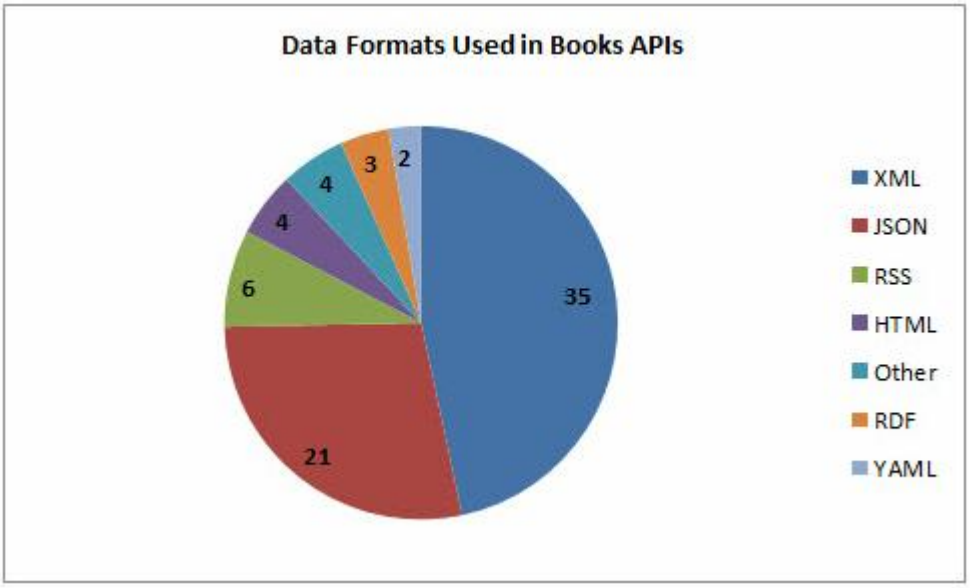
Ενώ υπάρχουν ήδη πολλές διεπαφές προγραμματισμού εφαρμογών που αναφέρονται σε βιβλία (API Books), όπως για παράδειγμα το AbeBooks[29] και το Bertram Books[30], το Google Books API είναι το δημοφιλέστερο όσο αφορά τις λειτουργίες του που προσφέρουν τον συνδιασμό, την απεικόνιση και την ομαδοποίηση των δεδομένων με σκοπό να γίνουν πιο χρήσιμα ως προς τον χρήστη για προσωπική ή και επαγγελματική χρήση. Κάποια παραδείγματα των λειτουργιών αυτών που κάνουν το Google Books API ξεχωριστό είναι:

- "bookfriend": είναι μία Android εφαρμογή η οποία δημιουργεί άμεσα συνοδευτικό οδηγό για κάθε βιβλίο ανεξάρτητα από το μέρος που βρίσκεται ο χρήστης.
- "Mash the Book": είναι ένας περιηγητής βιβλίων που επιτρέπει πλοήγηση ανάμεσα σε χιλιάδες βιβλία καθώς προβάλλονται οι εικόνες των εξωφύλλων τους. Γίνεται χρήση των Google Book Search Data και Book Viewer APIs.
- "ReadWhale": Μια κοινότητα για τους ανθρώπους για να ανακαλύψουν και να μοιραστούν το πάθος τους για τα βιβλία. Παρέχει ένα περιβάλλον όπου οι άνθρωποι μπορούν συλλογικά να συζητήσουν για βιβλία και να μοιραστούν τις δικές τους συλλογές με την κοινότητα.



Σχήμα 2.1.8: Πρωτόκολλα που χρησιμοποιούνται στις διεπαφές προγραμματισμού εφαρμογών που αναφέρονται σε βιβλία [31].

- "Tagbulb": Απλοποιεί την αναζήτηση ετικέτας με το να συσσωρεύει το περιεχόμενο από διάφορες πηγές, όπως είναι το *Flickr*, το *YouTube* και πολλά άλλα. Οι χρήστες μπορούν να περιηγηθούν ανάλογα με τον τύπο του περιεχομένου, όπως φωτογραφίες, videos, blogs, σελιδοδείκτες, podcasts, προϊόντα, βιβλία κλπ.
- " Book Stash": είναι ένα online κατάστημα βιβλίων, μία βάση δεδομένων που περιλαμβάνει κριτικές και περιλήψεις βιβλίων. Επιτρέπει αναζήτηση, διάβασμα και αγορά βιβλίων.



Σχήμα 2.1.9 Μορφότυποι Δεδομένων που χρησιμοποιούνται στις διεπαφές προγραμματισμού εφαρμογών που αναφέρονται σε βιβλία[31].

2.2 Το μοντέλο του πλαισίου Smartscan

Ας υποθέσουμε ότι το {U1,U2, …Um} είναι ένα σύνολο από τους m χρήστες της εφαρμογής μας, δηλαδή χρήστες που έχουν ήδη εγκαταστήσει την εφαρμογή Smartscan στο έξυπνο τηλέφωνο τους. Ο κάθε χρήστης, οποιαδήποτε στιγμή, μπορεί να σαρώσει βιβλία με τη χρήση της εφαρμογής, η οποία αυτόματα ενεργοποιεί την κάμερα της κινητής συσκευής, και μέσω της επιλογής αποθήκευσης μεταφέρει τα συγκεκριμένα βιβλία στην βάση δεδομένων - έτσι ταυτόχρονα επιτυγχάνει εμφάνιση των συγκεκριμένων βιβλίων στην online σελίδα μας.

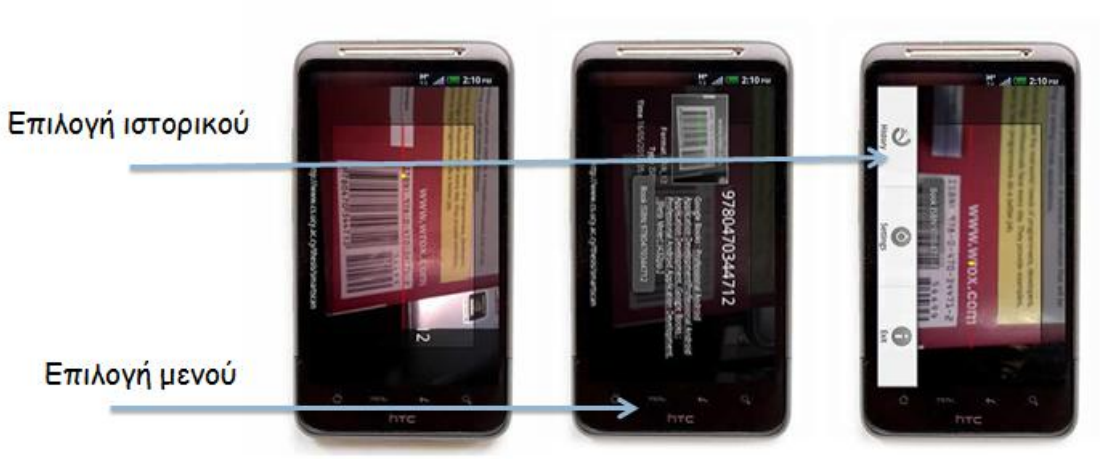
Το Smartscan είναι η δημιουργία μίας νοητής online βιβλιοθήκης η οποία αποτελείται από μία εφαρμογή σε Smartphone, βασισμένη στην ήδη υπάρχουσα "barcode scanner" εφαρμογή που διατίθεται στο Android market, και μία online ιστοσελίδα.

Σχήμα 2.2.1 : Smartscan εφαρμογή

Η Smartscan εφαρμογή παρέχει τη δυνατότητα στους χρήστες της να σαρώσουν τους μοναδικούς κωδικούς των βιβλίων οι οποίοι βρίσκονται στο πίσω μέρος του εξωφύλλου (ISBN).

Η Smartscan ιστοσελίδα παρουσιάζει όλους τους χρήστες-κατόχους των βιβλίων και όλες τις πληροφορίες για κάθε ένα από αυτά.

Αρχικά οι χρήστες καλούνται να κατεβάσουν την εφαρμογή από την Smartscan ιστοσελίδα και να τη εγκαταστήσουν στην Android έξυπνη συσκευή τους. Έπειτα, και αφού είναι ήδη εγγεγραμμένα μέλη, μπορούν να συνδεθούν στην εφαρμογή και να ξεκινήσουν τη σάρωση των βιβλίων που επιθυμούν (Σχήμα 2.2.1). Αφού γίνει η σάρωση ενός κωδικού του βιβλίου, υπό τη μορφή barcode, τότε αν το βιβλίο είναι καταχωρημένο στα Google Books γίνεται ανάκτηση των στοιχείων του από το συγκεκριμένο API (Σχήμα 2.2.2). Επίσης, κάθε βιβλίο που σαρώνεται συμπεριλαμβάνεται πλέον στο ιστορικό της εφαρμογής, στην συγκεκριμένη κινητή συσκευή. Ο χρήστης μπορεί να καταχωρήσει στη βάση δεδομένων Smartscan όσα βιβλία έχει σαρώσει και βρίσκονται πλέον στο ιστορικό της εφαρμογής. Τέλος, όσα βιβλία είναι καταχωρημένα στη βάση δεδομένων Smartscan εμφανίζονται στην ιστοσελίδα. Κάποιος χρήστης μπορεί να επισκεφτεί την ιστοσελίδα είτε ως απλός χρήστης είτε ως χρήστης μέλος, έχοντας τη δυνατότητα για περαιτέρω λειτουργίες.



Σχήμα 2.2.2 : Smartscan εφαρμογή

Κεφάλαιο 3

Η Αρχιτεκτονική του Smartscan

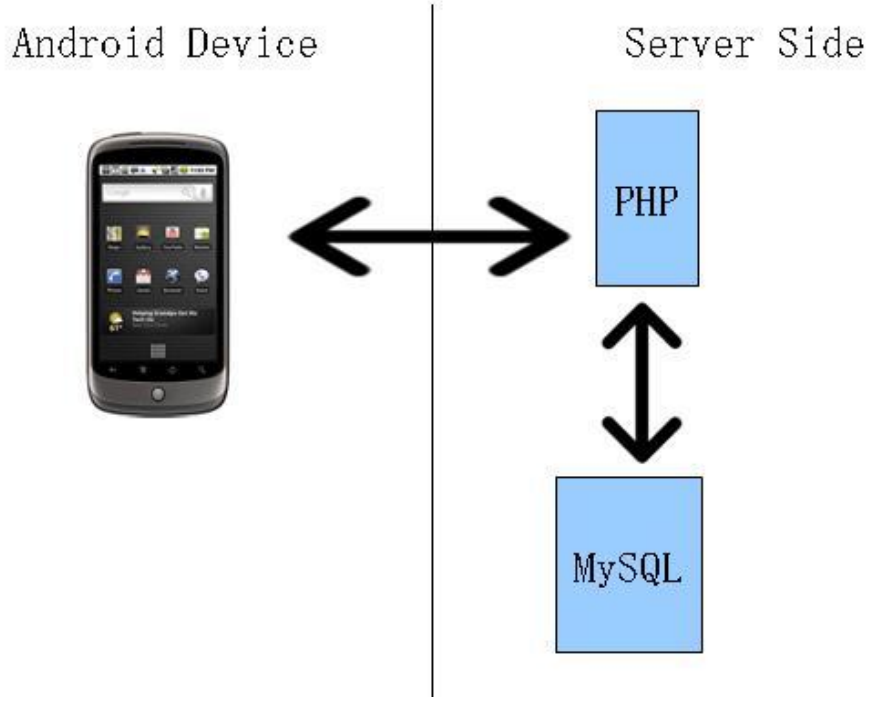
Στο κεφάλαιο αυτό αρχικά θα γίνει επισκόπηση της αρχιτεκτονικής του Smartscan με μία σύντομη περιγραφή για το ρόλο του κάθε συστατικού και πως επιτυγχάνεται η ανάκτηση και η αποθήκευση δεδομένων, από και προς την κινητή συσκευή/βάση δεδομένων. Θα αναφερθούμε στην ασφάλεια που προσφέρεται από το Smartscan σύστημα, και κατά πόσο τα δεδομένα του χρήστη είναι ασφαλή έτσι ώστε να μην παραβιάζεται η προσωπικότητα του. Τέλος, θα γίνει πιο λεπτομερής ανάλυση των επιμέρους συστατικών του συστήματος και ανάλυση των συστατικών της βάσης δεδομένων Smartscan.

3.1 Επισκόπηση Αρχιτεκτονικής Smartscan

Το Smartscan απαρτίζεται από μία Android έξυπνη κινητή συσκευή, έναν κεντρικό εξυπηρετητή και μία βάση δεδομένων MySQL.

Η πιο διαδεδομένη μέθοδος για να γίνει σύνδεση από μία Android συσκευή σε μια απομακρυσμένη βάση δεδομένων MySQL[15] είναι κάποιο είδος ενδιάμεσης υπηρεσίας. Χρησιμοποιούμε τη γλώσσα PHP δεδομένου ότι αλληλεπιδρά με τις βάσεις δεδομένων, για την ασφαλή ανάκτηση πληροφοριών από τον διακομηστή (server). Για την πραγματοποίηση σύνδεσης στο PHP script χρησιμοποιήσαμε το πρωτόκολλο HTTP[19] από το Android σύστημα. Γίνεται χρήση της αρχιτεκτονικής client-server όπου πελάτης είναι η Android έξυπνη συσκευή ενώ στην πλευρά του διακομιστή

υπάρχει ένας συνδυασμός PHP script με MySQL. Συνοψίζοντας η γλώσσα PHP βρίσκεται στο ενδιάμεσο για την επικοινωνία της Android συσκευής με τη βάση δεδομένων MySQL (Σχήμα 3.1.1) .



Σχήμα 3.1.2 : Android-PHP-MySQL

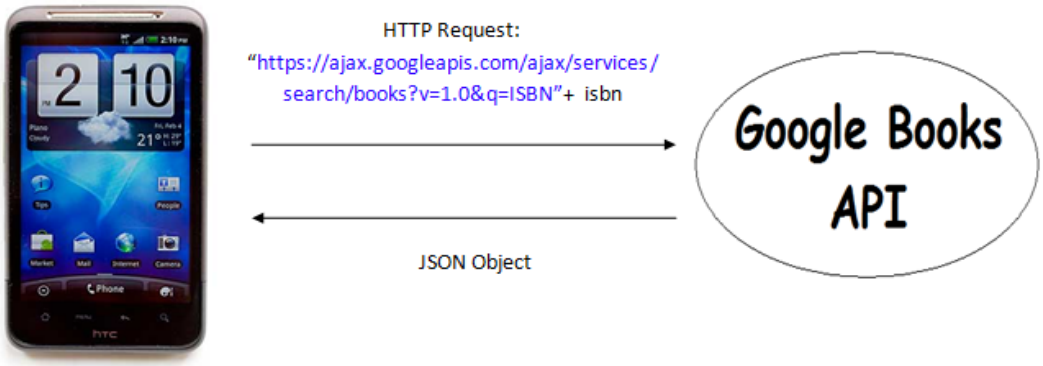
Συγκεκριμένα, η Android έξυπνη συσκευή συνδέεται με τη smartscan βάση δεδομένων μέσω PHP Script όταν:

Ο χρήστης επιχειρήσει να συνδεθεί στην smartscan εφαρμογή και χρειάζεται να γίνει επαλήθευση των στοιχείων του, ώστε να του επιτραπεί η είσοδος. Συγκεκριμένα, όταν ο χρήστης πληκτρολογήσει το συνθηματικό και τον κωδικό πρόσβασης του και επιλέξει να προχωρήσει σε σύνδεση στην εφαρμογή τότε γίνεται επαλήθευση του συγκεκριμένου συνδυασμού με όλους τους συνδυασμούς που είναι ήδη καταχωρημένοι στη βάση δεδομένων. (Η επαλήθευση αυτή εκτελείται και όταν ο χρήστης επιχειρήσει σύνδεση από την Smartscan ιστοσελίδα, με τη μόνη διαφορά ότι τα στοιχεία του χρήστη αποστέλλονται πλέον από την ιστοσελίδα και όχι από την εφαρμογή)

Ο χρήστης πλοηγείται στην Android εφαρμογή και θέλει να αποθηκεύσει τα δεδομένα που έχει συλλέξει μέσω της σάρωσης, έτσι ώστε να καταχωρηθούν τα συγκεκριμένα

δεδομένα. Συγκεκριμένα, όταν ο χρήστης επιλέξει την αποθήκευση του ιστορικού που υπάρχει αποθηκευμένο στην έξυπνη κινητή συσκευή του τότε τα στοιχεία των βιβλίων αποθηκεύονται στον πίνακα που αφορά τα βιβλία όπως επίσης και κάθε μοναδικός κωδικός για κάθε βιβλίο καταχωρείται στη λίστα με τα βιβλία του συγκεκριμένου χρήστη.

Γενικά, για την ανάκτηση των δεδομένων που αφορούν κάποιο βιβλίο που σαρώνεται μέσω της έξυπνης κινητής συσκευής γίνεται κάποια αναζήτηση στο Google Books API. Το Google Books API επιτρέπει στην εφαρμογή μας να εκτελέσει πλήρης αναζητήσεις και να ανακτήσει όλες τις πληροφορίες που αφορούν κάποιο βιβλίο. Η αναζήτηση γίνεται μέσω ενός HTTP Request αντικαθιστώντας κάθε φορά το αντίστοιχο ISBN κι επιστρέφεται ένα JSON Object (Σχήμα 3.1.3). Το JSON Object που επιστρέφεται μετά από την αίτηση για το συγκεκριμένο βιβλίο περιέχει όλες τις πληροφορίες που αφορούν σε αυτό. Αναλυτικά επιστρέφεται ένα κείμενο που περιέχει όλες τις χρήσιμες πληροφορίες όπως είναι ο τίτλος, οι συγγραφείς, το έτος δημοσίευσης, το πλήθος των σελίδων όπως επίσης και το URL (tbUrl) που παραπέμπει στη φωτογραφία του εξωφύλλου του συγκεκριμένου βιβλίου (Σχήμα 3.1.4).. Το συγκεκριμένο URL δεν



Σχήμα 3.1.3 : Ανάκτηση στοιχείων μέσω του Google Books API

Παραπέμπει αυτόματα στην εικόνα που θέλουμε, για το λόγο αυτό τυγχάνει επεξεργασίας μέσα από τον java κώδικα μας, όπου το αποκωδικοποιούμε και το μετατρέπουμε στο σωστό URL.

```
String Url = URLDecoder.decode(fields.getString("tbUrl"));
```

```
{"responseData":
{"results":[{"GsearchResultClass":"GbookSearch","unescapedUrl":"http://bo
oks.google.com/books?id\u003dOiGhQgAACAAJ\u0026dq\u003dISBN9780321295354\
u0026num\u003d4\u0026client\u003dinternal-
uds\u0026hl\u003del\u0026cd\u003d1\u0026source\u003duds","url":"http://bo
oks.google.com/books%3Fid%3DOiGhQgAACAAJ%26dq%3DISBN9780321295354%26num%3
D4%26client%3Dinternal-
uds%26hl%3Del%26cd%3D1%26source%3duds","title":"Algorithm
Design","titleNoFormatting":"Algorithm Design","authors":"Jon Kleinberg,
Éva
Tardos","bookId":"ISBN0321295358","publishedYear":"2006","tbUrl":"http://
bks6.books.google.com/books?id\u003dOiGhQgAACAAJ\u0026printsec\u003dfront
cover\u0026img\u003d1\u0026zoom\u003d5","tbHeight":"80","tbWidth":"60","p
ageCount":"838"},{"GsearchResultClass":"GbookSearch","unescapedUrl":"http
://books.google.com/books?id\u003dTnkZAQAAIAAJ\u0026q\u003dISBN9780321295
354\u0026dq\u003dISBN9780321295354\u0026num\u003d4\u0026client\u003dinter
nal-
uds\u0026hl\u003del\u0026cd\u003d2\u0026source\u003duds","url":"http://bo
oks.google.com/books%3Fid%3DTnkZAQAAIAAJ%26q%3DISBN9780321295354%26dq%3DI
SBN9780321295354%26num%3D4%26client%3Dinternal-
uds%26hl%3Del%26cd%3D2%26source%3duds","title":"Data structures and
algorithm analysis in Java","titleNoFormatting":"Data structures and
algorithm analysis in Java","authors":"Mark Allen
Weiss","bookId":"STANFORD36105114517480","publishedYear":"2007","tbUrl":"
http://bks9.books.google.com/books?id\u003dTnkZAQAAIAAJ\u0026printsec\u00
3dfrontcover\u0026img\u003d1\u0026zoom\u003d5","tbHeight":"80","tbWidth":
"60","pageCount":"555"},{"GsearchResultClass":"GbookSearch","unescapedUrl
":"http://books.google.com/books?id\u003dNLngYyWFl_YC\u0026printsec\u003d
frontcover\u0026dq\u003dISBN9780321295354\u0026num\u003d4\u0026client\u00
3dinternal-
uds\u0026hl\u003del\u0026cd\u003d3\u0026source\u003duds","url":"http://bo
oks.google.com/books%3Fid%3DNLngYyWFl_YC%26printsec%3Dfrontcover%26dq%3DI
SBN9780321295354%26num%3D4%26client%3Dinternal-
uds%26hl%3Del%26cd%3D3%26source%3duds","title":"Introduction To
Algorithms","titleNoFormatting":"Introduction To
Algorithms","authors":"Thomas H Cormen, Charles E Leiserson, Ronald L
Rivest, Clifford
Stein","bookId":"ISBN0262032937","publishedYear":"2001","tbUrl":"http://b
ks0.books.google.com/books?id\u003dNLngYyWFl_YC\u0026printsec\u003dfrontc
over\u0026img\u003d1\u0026zoom\u003d5\u0026edge\u003dcurl","tbHeight":"80
","tbWidth":"60","pageCount":"1180"}],"cursor":{"pages":[{"start":"0","la
bel":1}],"estimatedResultCount":"3","currentPageIndex":0,"moreResultsUrl"
:"http://books.google.com/books?source\u003duds\u0026start\u003d0\u0026hl
\u003del\u0026q\u003dISBN9780321295354"}}, "responseDetails": null,
"responseStatus": 200}
```


Σχήμα 3.1.4 : παράδειγμα JSON αντικείμενο

3.2 Ασφάλεια Δεδομένων του συστήματος Smartscan

Στο σύστημα Smartscan φροντίσαμε για την ασφάλεια των δεδομένων των χρηστών αλλά και του διαχειριστή του συστήματος. Συγκεκριμένα, με τη σύνδεση της Android εφαρμογής στην MySQL βάση δεδομένων μέσω του PHP (Σχήμα 3.2.1) εξασφαλίζουμε την ασφάλεια και την μη έκθεση των στοιχείων πρόσβασης στη βάση δεδομένων του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου (username και password που μας έχουν δοθεί για την πρόσβαση μας στη βάση δεδομένων smartscan που έχουμε αναπτύξει). Αυτή η απόκρυψη πληροφοριών δεν θα ήταν εφικτή στην περίπτωση που ενώναμε απευθείας την Android εφαρμογή με τη χρήση JDBC Driver στη βάση δεδομένων. Με την έκθεση των credentials (username, password) που αφορούν στη σύνδεση μας με τη βάση δεδομένων, έτσι ώστε να διαχειριζόμαστε όλες τις καταχωρημένες πληροφορίες, θα μπορούσε ο οποιοσδήποτε κακόβουλος να εισβάλει και να τροποποιήσει ή να διαγράψει δεδομένα από τη βάση δεδομένων. Άλλο ένα πιθανό σενάριο θα ήταν να κλαπούν οι πληροφορίες σύνδεσης των χρηστών στο σύστημα από επιτήδειους, γεγονός που θα παραβίαζε την προσωπικότητα τους.

Επίσης, ασφάλεια των δεδομένων των χρηστών εφαρμόζεται και στην smartscan ιστοσελίδα αφού κάποιος χρήστης για να μπορέσει να δει τα στοιχεία κάποιου άλλου χρήστη, ιδιοκτήτη ενός βιβλίου, θα πρέπει να είναι ήδη συνδεδεμένος. Για να μπορεί να συνδεθεί κάποιος χρήστης θα πρέπει αρχικά να δημιουργήσει λογαριασμό και να εγκριθεί από τον διαχειριστή του συστήματος ώστε να μπορεί να συνδεθεί.

Σχήμα 3.2.1 : Ασφάλεια Δεδομένων

3.3 Υλικό που χρησιμοποιήθηκε

Για την ανάπτυξη του smartscan συστήματος χρησιμοποιήθηκε ένας εξυπηρετητής, ο εξυπηρετητής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου. Μέσω αυτού μπορούσαμε να εκτελούμε και να δοκιμάζουμε την εφαρμογή και την ιστοσελίδα μας. Επίσης για την ανάπτυξη της εφαρμογής και την λήψη δειγμάτων χρησιμοποιήθηκε μία συσκευή HTC Desire (Σχήμα 3.3.1) , η οποία και παραχωρήθηκε από τον επιβλέπον καθηγητή της εν λόγω ατομικής διπλωματικής εργασίας.

HTC DESIRE	
	***Storage:*** **Internal phone storage:** 4 GB **RAM:** 768 MB **Expansion slot:** microSD™ memory card (SD 2.0 compatible) ***CPU Processing Speed:*** 1 GHz ***Platform:*** Android™ 2.2 (Froyo) with HTC Sense™ ***Wi-Fi®:*** IEEE 802.11 b/g/n

Σχήμα 3.3.1 : HTC Desire

3.4 Λειτουργίες Συστήματος

- Διαχείριση Βιβλίων

1. Σάρωση Βιβλίου

   Η διαδικασία όπου με τη βοήθεια της κάμερας του κινητού τηλεφώνου σαρώνεται το ISBN του βιβλίου και ακολούθως προσθέτεται στη βάση δεδομένων.

   - Δεδομένα Εισόδου: Οι πληροφορίες που χρειάζονται για τη σάρωση κάποιου βιβλίου είναι το ISBN του βιβλίου.
   - Επεξεργασία: Αφού σαρωθεί το βιβλίο γίνεται επεξεργασία του ISBN έτσι ώστε να διαπιστωθεί αν είναι όντως βιβλίο και αν υπάρχουν στο διαδίκτυο πληροφορίες για το συγκεκριμένο βιβλίο.

- Αποτελέσματα: Αν η σάρωση του βιβλίου ολοκληρωθεί με επιτυχία εμφανίζονται στην οθόνη του κινητού τα στοιχεία του βιβλίου, αλλιώς αν το αντικείμενο που σαρώθηκε δεν είναι βιβλίο εμφανίζεται ανάλογο μήνυμα στην οθόνη του κινητού ότι δεν είναι βιβλίο.

2. Αναζήτηση Βιβλίου

   Η διαδικασία όπου κάποιος χρήστης αναζητά κάποιο βιβλίο μέσω της ιστοσελίδας.

   - Δεδομένα Εισόδου: Η αναζήτηση κάποιου βιβλίου μπορεί να γίνει δίνοντας σαν είσοδο το ISBN του βιβλίου, ο τίτλος του βιβλίου, ο συγγραφέας του βιβλίου ή και το έτος έκδοσης του βιβλίου.
   - Επεξεργασία: Με την εισαγωγή ενός από τα στοιχεία που προαναφέραμε γίνεται η αναζήτηση και βρίσκονται τα βιβλία που ταιριάζουν με τα κριτήρια που δώσαμε.
   - Αποτελέσματα: Μετά την επεξεργασία των δεδομένων που δίνονται στην είσοδο, ανανεώνεται ο πίνακας της ιστοσελίδας και εμφανίζει τα βιβλία όπου τα στοιχεία τους ταιριάζουν με τα κριτήρια που δίνει ο χρήστης. Αν δεν υπάρχει κάποιο τέτοιο βιβλίο δεν εμφανίζεται τίποτα στον πίνακα βιβλίων.

3. Διαγραφή Βιβλίου

   Η διαδικασία όπου κάποιος χρήστης θέλει να διαγράψει κάποιο βιβλίο το οποίο όμως πρέπει να ανήκει στην δική του προσωπική βιβλιοθήκη.

   - Δεδομένα εισόδου: Ο χρήστης επιλέγει τη σειρά του πίνακα στην οποία εμφανίζεται το συγκεκριμένο βιβλίο.
   - Επεξεργασία: Ζητείται από τον χρήστη να επιβεβαιώσει τη διαγραφή και ακολούθως διαγράφεται το βιβλίο από τη Βάση Δεδομένων.
   - Αποτελέσματα: Ανανεώνεται ο πίνακας με τα βιβλία της προσωπικής βιβλιοθήκης του χρήστη και σε αυτήν δεν παρουσιάζεται το βιβλίο που έχει διαγραφεί αφού δεν υπάρχει πλέον στην βιβλιοθήκη.

4. Βαθμολόγηση Βιβλίου

Η διαδικασία όπου κάποιος χρήστης θέλει να βαθμολογήσει κάποιο βιβλίο το οποίο μπορεί να ανήκει σε οποιονδήποτε χρήστη.

- Δεδομένα Εισόδου: Μέσα από την ιστοσελίδα ο χρήστης επιλέγει πόσα αστεράκια θέλει να δώσει στο βιβλίο, όπου τα αστεράκια αντιπροσωπεύουν τη βαθμολογία του χρήστη για το βιβλίο.
- Επεξεργασία: Αφού ο χρήστης επιλέξει τη βαθμολογία που θέλει για το βιβλίο υπολογίζεται ξανά η νέα βαθμολογία του βιβλίου.
- Αποτελέσματα: Σύμφωνα με την επεξεργασία που γίνεται στη βαθμολογία του βιβλίου ανανεώνεται η οθόνη του υπολογιστή και μειώνονται ή αυξάνονται τα αστεράκια ανάλογα με τη νέα βαθμολογία του βιβλίου.

- Διαχείριση Χρηστών

1. Δημιουργία Λογαριασμού

Ο χρήστης μπορεί μέσω της ιστοσελίδας να δημιουργήσει λογαριασμό έτσι ώστε να γίνει μέλος.

- Δεδομένα Εισόδου: Για να δημιουργηθεί ο λογαριασμός του χρήστη πρέπει να δώσει σαν είσοδο το όνομά του, το επίθετό του, username, password και email. Το password δίνεται 2 φορές για επιβεβαίωση.
- Επεξεργασία: Αφού δοθούν τα στοιχεία από τον χρήστη ελέγχεται αν είναι έγκυρα και αν είναι δημιουργείται ο λογαριασμός.
- Αποτελέσματα: Με τη δημιουργία του λογαριασμού εμφανίζεται μήνυμα στην ιστοσελίδα ότι ο χρήστης είναι πλέον μέλος της ιστοσελίδας.

2. Σύνδεση στην Ιστοσελίδα

Ο χρήστης μπορεί να συνδεθεί στην ιστοσελίδα για να μπορεί να κάνει κάποιες πιο εξειδικευμένες λειτουργίες.

- Δεδομένα Εισόδου: Ο χρήστης εισάγει το username και το password του στην ιστοσελίδα και επιλέγει το κουμπί Login.

- Επεξεργασία: Ελέγχονται τα στοιχεία που δίνει ο χρήστης αν είναι σωστά και αν το password που δίνεται αντιστοιχεί στο username του χρήστης.
- Αποτελέσματα: Αν τα στοιχεία είναι σωστά τότε γίνεται η σύνδεση και εμφανίζεται στην ιστοσελίδα ο πίνακας με τα βιβλία του χρήστη. Αν δεν είναι σωστά τα στοιχεία τότε εμφανίζεται μήνυμα λάθους.

- Διαχείριση Εφαρμογής

1. Κατέβασμα εφαρμογής από ιστοσελίδα

- Διαχείρηση Ιστοσελίδας

Κάποιος απλός χρήστης μπορεί:

1. Να κατεβάσει την smartscan εφαρμογή.
2. Να δημιουργήσει λογαριασμό ώστε να γίνει μέλος.
3. Να αναζητήσει ένα βιβλίο μέσω της λειτουργίας αναζήτησης.

Κάποιος χρήστης μέλος μπορεί:

1. Να συνδεθεί στην εφαρμογή και να σαρώσει βιβλία.
2. Να συνδεθεί στην ιστοσελίδα για πιο εξειδικευμένες λειτουργίες:
3. Να βαθμολογήσει κάποιο βιβλίο.
4. Να δει αναλυτικές πληροφορίες για τον κάτοχο κάθε βιβλίου.
5. Να δει τα βιβλία που βρίσκονται στην προσωπική του βιβλιοθήκη.
6. Να διαγράψει βιβλία από την προσωπική του βιβλιοθήκη.

3.5 Η βάση δεδομένων του Smartscan

Στη βάση MySQL έχουμε υλοποιήσει τρεις πίνακες (Σχήμα 3.4.1):

Χρήστης (user) : αντιπροσωπεύει το χρήση που χειρίζεται τα βιβλία του, είτε από την εφαρμογή στην Android συσκευή είτε από την online ιστοσελίδα του. Ο χρήστης έχει τα εξής χαρακτηριστικά :

Username : συνθηματικό όνομα του χρήστη το οποίο επιλέγει ο χρήστης κατά τη διαδικασία δημιουργίας λογαριασμού του (Register). Τον χαρακτηρίζει μοναδικά (αφού γίνεται έλεγχος αν υπάρχει ήδη το συγκεκριμένο username πριν την δημιουργία λογαριασμού του χρήστη) και χρησιμοποιείται από τον χρήστη για τη σύνδεση του είτε στην εφαρμογή στην Android συσκευή, είτε στην ιστοσελίδα.

Password : κωδικός χρήστη τον οποίο επιλέγει ο χρήστης κατά τη διαδικασία δημιουργίας λογαριασμού του (Register) και χρησιμοποιείται από τον χρήστη για τη σύνδεση του είτε στην εφαρμογή στην Android συσκευή, είτε στην ιστοσελίδα.

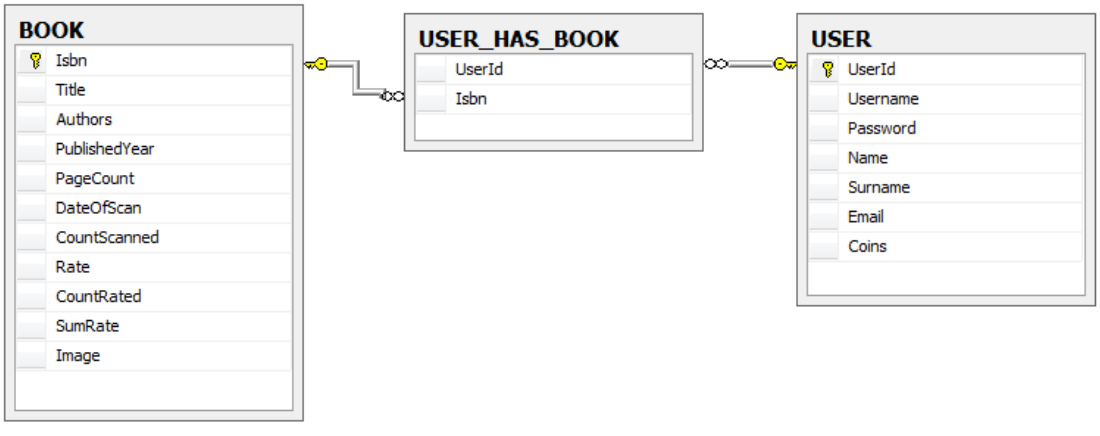
Name : το πραγματικό όνομα του χρήστη το οποίο καταχωρεί ο χρήστης κατά τη διαδικασία δημιουργίας λογαριασμού του (Register), μέσω της ιστοσελίδας.

Surname : το επίθετο του χρήστη το οποίο καταχωρεί ο χρήστης κατά τη διαδικασία δημιουργίας λογαριασμού του (Register) , μέσω της ιστοσελίδας.

User_id : η ταυτότητα του χρήστη η οποία αντιπροσωπεύει τον αριθμό εισδοχής του χρήστη (σε αύξουσα σειρά) στο σύστημα. Το χαρακτηριστικό αυτό ορίζεται αυτόματα από τη βάση δεδομένων, δηλαδή αυξάνεται με την εισδοχή κάθε νέου χρήστη.

Coins : το έπαθλο του χρήστη για κάθε βιβλίο που εισαγάγει στη βάση δεδομένων. Αποτελεί κίνητρο προς το χρήστη έτσι ώστε να εισάγει όσα περισσότερα βιβλία μπορεί με σκοπό να αυξάνονται τα βιβλία που υπάρχουν στη βάση δεδομένων και να είναι πιο εύκολο και εφικτό για κάποιο χρήστη να βρει κάποιο βιβλίο που ψάχνει.

Email : το επίθετο του χρήστη το οποίο καταχωρεί ο χρήστης κατά τη διαδικασία δημιουργίας λογαριασμού του (Register), μέσω της ιστοσελίδας.



Σχήμα 3.4.1 : Σχήμα Βάσης Δεδομένων

Βιβλίο (book) : αντιπροσωπεύει κάθε βιβλίο το οποίο έχει καταχωρηθεί στη βάση δεδομένων από οποιονδήποτε χρήστη του συστήματος. Το βιβλίο έχει τα εξής χαρακτηριστικά:

Isbn : ο κωδικός του βιβλίου που το χαρακτηρίζει μοναδικά και μας βοηθά να ανακτήσουμε τα υπόλοιπα στοιχεία του βιβλίου μέσω της χρήσης JSON.

Title : ο τίτλος του βιβλίου.

Authors : ο/οι συγγραφέας/συγγραφείς του βιβλίου.

Published Year : ο χρόνος δημοσίευσης του βιβλίου.

Page Count : το πλήθος των σελίδων του βιβλίου.

Date of Scan : η ημερομηνία που έγινε η σάρωση του βιβλίου από την Android εφαρμογή.

Count Scanned : το πλήθος των φορών που σαρώθηκε το συγκεκριμένο βιβλίο, δηλαδή από πόσους χρήστες.

Rate : η βαθμολογία ενός βιβλίου η οποία διαμορφώνεται από τις διάφορες ψηφοφορίες των χρηστών για κάθε βιβλίο ξεχωριστά. Η βαθμολογία αυτή εμφανίζεται δίπλα από το κάθε βιβλίο στην ιστοσελίδα, μόνο στην περίπτωση που ο χρήστης είναι συνδεδεμένος.

Image : για κάθε βιβλίο είναι αποθηκευμένο στη βάση δεδομένων ένα url το οποίο παραπέμπει στη φωτογραφία εξωφύλλου του συγκεκριμένου βιβλίου.

Count rated : με αυτή τη μεταβλητή κρατάμε το πλήθος τον χρηστών που έχουν ψηφίσει το συγκεκριμένο βιβλίο έτσι ώστε να μπορούμε να υπολογίσουμε τη συνολική βαθμολογία ενός βιβλίου.

Sum rate : με αυτή τη μεταβλητή κρατάμε το άθροισμα όλων των ψηφοφοριών που έγιναν για τη βαθμολόγηση κάποιου βιβλίου έτσι ώστε να μπορούμε να υπολογίσουμε τη συνολική βαθμολογία του.

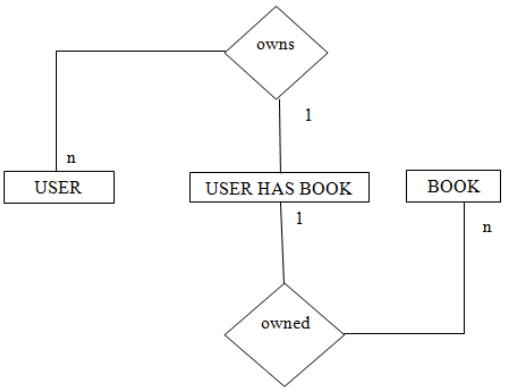
Βιβλία_Χρήστη (user_has_book) : αντιπροσωπεύει το ζεύγος χρήστης-βιβλίο δηλαδή τον χρήστη και τα βιβλία τα οποία του αντιστοιχούν, έτσι ώστε να μπορούμε να ξέρουμε ποια βιβλία έχει ο κάθε χρήστης. Η σχέση αυτή έχει τα εξής χαρακτηριστικά:

User Id : η ταυτότητα του κάθε χρήστη η οποία αντιπροσωπεύει τον αντιπροσωπεύει μοναδικά, και με αυτόν τον τρόπο μπορούμε να προσδιορίσουμε για ποιον χρήστη μιλάμε.

Isbn : ο κωδικός κάθε βιβλίου που το χαρακτηρίζει μοναδικά και ανήκει στο χρήστη με το πιο πάνω user id.

Κάθε εγγεγραμμένος χρήστης μπορεί να έχει από 0 έως πολλά βιβλία. Επίσης κάθε βιβλίο μπορεί να ανήκει σε ένα ή και περισσότερους χρήστες (η συγκεκριμένη πλειάδα υπάρχει μία φορά καταχωρημένη στη βάση δεδομένων) (Σχήμα 3.4.1).

Ο PHP κώδικας χρησιμοποιήθηκε για να ενωθούμε στη βάση δεδομένων MySQL, να εκτελέσουμε τα MySQL ερωτήματα όπου στο WHERE μπλοκ υπήρχαν συσχετίσεις σε δεδομένα που αφορούσαν τιμές POST ή GET όπως επίσης και στην έξοδο δεδομένων σε JSON μορφή.



Σχήμα 3.4.1 : Διάγραμμα Σχέσεων

Κεφάλαιο 4

Υλοποίηση του Smartscan

Στο κεφάλαιο αυτό θα παρουσιάσουμε το Graphical User Interface του συστήματος Smartscan. Πιο αναλυτικά, θα ακολουθήσουν στιγμιότυπα οθόμης που αφορούν στην εφαρμογή του Smartscan (barcode Scanner) όπως επίσης και την Smartscan ιστοσελίδα με σύνδεσμο www.cs.ucy.ac.cy.thesis/smartscan/ . Θα εξηγήσουμε με σαφήνεια πως γίνεται χρήση της εφαρμογής αλλά και της ιστοσελίδας, και επιπρόσθετα θα αναφερθούμε σε όλες τις λειτουργίες που παρέχονται από το σύστημα Smartscan για τους απλούς χρήστες αλλά και για τους χρήστες μέλη.

4.1 GUI εφαρμογής Smartscan

Η εφαρμογή Smartscan είναι μία παραλλαγή της Barcode Scanner εφαρμογής η οποία έχει αναπτυχθεί από την ομάδα ZXing, με τη χρήση της ομόνυμης βιβλιοθήκης που είναι ανοικτή προς το κοινό. Συγκεκριμένα η εφαρμογή έχει τροποποιηθεί έτσι ώστε να μπορεί να ανακτά τις πληροφορίες για τα βιβλία που σαρώνονται και να τις εισάγει στη βάση δεδομένων. Η συγκεκριμένη εφαρμογή λειτουργεί μόνο για τη σάρωση και ανάκτηση δεδομένων που αφορούν ένα βιβλίο και δεν μπορεί να χρησιμοποιηθεί για οποιαδήποτε άλλα προϊόντα, όπως συμβαίνει με την πρωτότυπη εφαρμογή. Βασική προϋπόθεση για να χρησιμοποιήσει κάποιος χρήστης την εφαρμογή είναι να έχει ήδη δημιουργήσει λογαριασμό (μέσω της ιστοσελίδας), κι έπειτα να κατεβάσει την εφαρμογή στην έξυπνη κινητή συσκευή του. Επίσης, για την σύνδεση ενός χρήστη στην εφαρμογή απαραίτητη είναι η ασύρματη σύνδεση διαδικτύου, έτσι ώστε να μπορεί να γίνει επαλήθευση των στοιχείων του από την Smartscan βάση δεδομένων .

Σχήμα 4.1.1 : Σύνδεση Χρήστη

Σχήμα 4.1.2 : Κυρίως Μενού Επιλογών

Ο χρήστης μπορεί να κάνει σύνδεση στην εφαρμογή (Σχήμα 4.1.1) εισάγοντας το συνθηματικό και τον κωδικό πρόσβασης του, τα οποία ο ίδιος έχει επιλέξει κατά την δημιουργία του λογαριασμού του μέσω της ιστοσελίδας.

Με την επιλογή "Login" η εφαρμογή μεταφέρεται στο κυρίως μενού (Σχήμα 4.1.2). Στην περίπτωση που δεν υπάρχει σύνδεση στο διαδίκτυο δεν μπορεί να γίνει σύνδεση αφού δεν μπορούν να ταυτοποιηθούν τα στοιχεία του χρήστη από τη βάση δεδομένων, κι έτσι η εφαρμογή δεν προχωρά.

Ο χρήστης καθώς βρίσκεται στο κυρίως μενού (Σχήμα 4.1.2) μπορεί να επιλέξει:

«Scan Book»: Με την επιλογή αυτή ο χρήστης μεταφέρεται στην οθόνη σάρωσης (Σχήμα 4.1.3) και είναι πλέον έτοιμος να σαρώσει τα barcodes των βιβλίων που επιθυμεί.

«Insert Book by ISBN»: Επιλέγοντας τη συγκεκριμένη επιλογή δίνεται στο χρήστη η δυνατότητα να καταχωρήσει από το πληκτρολόγιο της κινητής συσκευής του τον μοναδικό κωδικό βιβλίου (ISBN), ο οποίος αναγράφεται στο πίσω μέρος του εξωφύλλου του βιβλίου. Η επιλογή αυτή μπορεί να χρησιμοποιηθεί στις περιπτώσεις που ο κωδικός βιβλίου (ISBN) δεν μπορεί να σαρωθεί από την εφαρμογή (είτε επειδή δεν βρίσκεται στη μορφή 1D barcode είτε επειδή δεν είναι σε καλή ανάλυση ώστε να

μπορεί αν διαβαστεί) (Σχήμα 4.1.15). Και σε αυτή την ειδική περίπτωση όλες οι πληροφορίες που αφορούν το συγκεκριμένο βιβλίο αποθηκεύονται στη βάση δεδομένων, όπως θα γινόταν και με την επιλογή της προηγούμενης λειτουργίας (Scan Book).

«Logout»: Με την επιλογή αυτή ο χρήστης αποσυνδέεται από την εφαρμογή, και η εφαρμογή επιστρέφει στην οθόνη σύνδεσης (Σχήμα 4.1.1).

Σχήμα 4.1.3 : Οθόνη Σάρωσης

Σχήμα 4.1.4 : Ανάκτηση Δεδομένων μετά τη σάρωση του βιβλίου

Ο χρήστη επιλέγοντας την πρώτη λειτουργία (Scan Book) μεταφέρεται στην οθόνη σάρωσης και αφού τοποθετήσει το βιβλίο μπροστά από την κάμερα της κινητής συσκευής μπορεί αν σαρώσει το barcode του. Στην περίπτωση που το barcode που σαρώθηκε δεν αντιστοιχεί σε κάποιο βιβλίο τότε στην οθόνη εμφανίζεται ένα μήνυμα «This is not a book» και ο συγκεκριμένος κωδικός δεν καταχωρείται στο ιστορικό. Σε αντίθετη περίπτωση ο κωδικός και τα υπόλοιπα στοιχεία του βιβλίου εμφανίζονται στην οθόνη (Σχήμα 4.1.4) και το συγκεκριμένο βιβλίο καταχωρείται στο ιστορικό της εφαρμογής.

Ο χρήστης αφού έχει σαρώσει το βιβλίο με τη βοήθεια της κάμερας της κινητής συσκευής κι αφότου ο κωδικός αναφέρεται σε βιβλίο τότε τα στοιχεία του βιβλίου αναγράφονται στην οθόνη (ο τίτλος, ο συγγραφές και το πλήθος των σελίδων) τα οποία

ανακτώνται εκείνη τη δεδομένη χρονική στιγμή μέσω του Google Books API. Το βιβλίο καταχωρείται στο ιστορικό της εφαρμογής (Σχήμα 4.1.6).

Ο χρήστης σε αυτή τη φάση μπορεί να επιλέξει το κουμπί μενού που βρίσκεται στην έξυπνη κινητή συσκευή του κι έτσι να εμφανιστεί κάποιο υπομενού (Σχήμα 4.1.5) με τις επιλογές:

«History»: Με την επιλογή αυτή η εφαρμογή μεταφέρεται στο ιστορικό (Σχήμα 4.1.7) όπου υπάρχουν όλα τα βιβλία που έχουν σαρωθεί από τον χρήστη και δεν έχουν διαγραφεί από εκεί. Σημείωση: Αρχικά, μόλις η εφαρμογή εγκατασταθεί στην κινητή συσκευή έχει άδειο ιστορικό

«Settings»: Με την επιλογή αυτή η εφαρμογή μεταφέρεται στις ρυθμίσεις (Σχήμα 4.1.8).

«Exit»: Με την επιλογή αυτή η εφαρμογή εξέρχεται από την οθόνη σάρωσης κι επιστρέφει στο κυρίως μενού(Σχήμα 4.1.2).

Ο χρήστης μπορεί να δει το ιστορικό των όσων βιβλίων έχει διαβάσει μέχρι τώρα μέσω της barcode scanner εφαρμογής από την συγκεκριμένη κινητή συσκευή ιστορικό (Σχήμα 4.1.7). Είναι εφικτό να πλοηγηθούμε στο ιστορικό όπως επίσης και να επιλέξουμε κάποια εισαγωγή βιβλίου και να παραπεμφθούμε σε αυτό.

Ο χρήστης καθώς βρίσκεται στο ιστορικό (Σχήμα 4.1.6), μπορεί να επιλέξει:

«Save History»: Με την επιλογή αυτή το ιστορικό που υπάρχει στην συγκεκριμένη συσκευή καταχωρείται στη βάση δεδομένων smartscan κι επίσης αποθηκεύεται σε ένα αρχείο κειμένου στη συγκεκριμένη συσκευή στο μονοπάτι "/mnt/sdcard/media/books.txt".

«Clear History»: Με την επιλογή αυτή το ιστορικό που υπάρχει στην συγκεκριμένη εφαρμογή θα διαγραφεί. Η διαγραφή αυτή δεν θα επηρεάσει ούτε τα στοιχεία στη βάση δεδομένων αλλά ούτε και το αρχείο κειμένου στη συγκεκριμένη συσκευή.

Σχήμα 4.1.5 : Οθόνη Σάρωσης με Υπομενού

Σχήμα 4.1.6 : Αποθήκευση Ιστορικού

Η επιλογή «Save History» καταχωρεί το ιστορικό που υπάρχει στην συγκεκριμένη συσκευή στη βάση δεδομένων smartscan κι επίσης δημιουργεί ένα αρχείο κειμένου στη συγκεκριμένη συσκευή στο μονοπάτι "/mnt/sdcard/media/books.txt" που περιέχει όλα τα στοιχεία των βιβλίων που έχουν σαρωθεί και βρίσκονται στο ιστορικό. Τα στοιχεία αυτά είναι τα εξής: Book ISBN, title, authors, published year, number of pages και date of scan.

Η επιλογή «Clear History» διαγράφει το ιστορικό που υπάρχει στην συγκεκριμένη συσκευή αλλά δεν επηρεάζει ούτε τα στοιχεία που υπάρχουν στη βάση δεδομένων smartscan ούτε και το αρχείο κειμένου στη συγκεκριμένη συσκευή στο μονοπάτι "/mnt/sdcard/media/books.txt" που περιέχει όλα τα στοιχεία των βιβλίων που έχουν σαρωθεί και βρίσκονταν στο ιστορικό όταν έγινε η δημιουργία του. Πρωτού η εφαρμογή προχωρήσει στη διαγραφή επιβεβαιώνει την επιλογή του χρήστη (Σχήμα 4.1.8).

Εάν ο χρήστης επιβεβαιώσει τη διαγράφη του ιστορικού που υπάρχει στην συγκεκριμένη συσκευή τότε όλο το ιστορικό διαγράφεται (Σχήμα 4.1.9). Με την επιβεβαίωση το ιστορικό έχει πλέον διαγραφεί από τη συγκεκριμένη εφαρμογή και είναι άδειο. Μόλις ο χρήστης σαρώσει κάποιο νέο βιβλίο τότε θα αρχίσουν να προστίθενται νέες καταχωρήσεις στο ιστορικό.

Σχήμα 4.1.7 : Διαγραφή Ιστορικού

Σχήμα 4.1.8 : Επιβεβαίωση Διαγραφής Ιστορικού

Σχήμα 4.1.9 : Άδειο Ιστορικό

Μία επιπλέον λειτουργία που παρέχεται είναι το γεγονός ότι ο χρήστης μπορεί να καταχωρήσει από το πληκτρολόγιο της κινητής συσκευής του τον κωδικό βιβλίου (ISBN) ο οποίος αναγράφεται στο πίσω μέρος του βιβλίου (Σχήμα 4.1.10). Η επιλογή αυτή μπορεί να χρησιμοποιηθεί στις περιπτώσεις που ο κωδικός βιβλίου (ISBN) δεν μπορεί να σαρωθεί από την εφαρμογή (είτε επειδή δεν βρίσκεται στη μορφή 1D barcode είτε επειδή δεν είναι σε καλή ανάλυση ώστε να μπορεί αν διαβαστεί).

Το μήνυμα «Book Found» εμφανίζεται στην οθόνη αν το συγκεκριμένο ISBN αντιστοιχεί σε κάποιο βιβλίο. Αν το μήνυμα αυτό εμφανιστεί σημαίνει ότι η καταχώρηση του συγκεκριμένου βιβλίου στη βάση δεδομένων (από τον συνδεδεμένο χρήστη) έγινε με επιτυχία αλλιώς το συγκεκριμένο βιβλίο δεν έχει βρεθεί και δεν έχει καταχωρηθεί επιτυχώς στη βάση δεδομένων.

Ο χρήστης έχει τη δυνατότητα να επωφεληθεί των ρυθμίσεων που παρέχονται (Σχήμα 4.1.11). Οι ρυθμίσεις της εφαρμογής είναι οι εξής:

«1D barcodes» : Αποκωδικοποίηση barcodes σε 1D μορφή – μορφή στην οποία βρίσκονται και οι κωδικοί των βιβλίων (ISBN).
«Beep» : Όταν γίνει επιτυχής σάρωση ενός barcode τότε ένας ήχος «beep» ακούγεται.
«Vibrate» : Όταν γίνει επιτυχής σάρωση ενός barcode τότε η κινητή συσκευή δονείται.

«Copy to clipboard» : Ο κωδικός του βιβλίου (ISBN) αντιγράφεται και μπορεί να επικολληθεί όπου θέλει ο χρήστης.

«Remember duplicates» : Η ίδια καταχώρηση ενός βιβλίου στο ιστορικό μπορεί να αποθηκεφτεί πολλές φορές.

«Retrieve more info» : Ανάκτηση περισσότερων πληροφοριών όσο αφορά τα περιεχόμενα του συγκεκριμένου barcode.

«Custom search URL» : Μπορεί να γίνει αλλαγή του URL στο οποίο γίνεται αίτημα για την ανάκτηση των στοιχείων που αφορούν το barcode που έχει σαρωθεί.

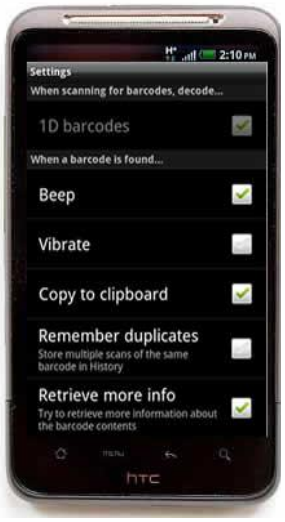
«Use front light» : Με την επιλογή αυτή βελτιώνεται η σάρωση σε κάποιες κινητές συσκευές, στην περίπτωση που ο φωτισμός είναι χαμηλός.

«Bulk scan mode» : Για «ανανέωση» της οθόνης σάρωσης κάθε 10 δευτερόλεπτα έτσι ο χρήστης να μην πρέπει να πατά συνεχώς το κουμπί «back» για να μπορεί να σαρώσει το επόμενο barcode.

«Reserved camera image» : Οι κινητές συσκευές που έχουν κάμερα στο μπροστινό τους μέρος μπορούν να σαρώσουν μέσω αυτής.

«Select country» : Επιλογή χώρας μέσω της οποίας θα γίνεται η αναζήτηση για την ανάκτηση στοιχείων για κάθε barcode που σαρώνεται.

Σχήμα 4.1.10 : Εισαγωγή βιβλίου με ISBN

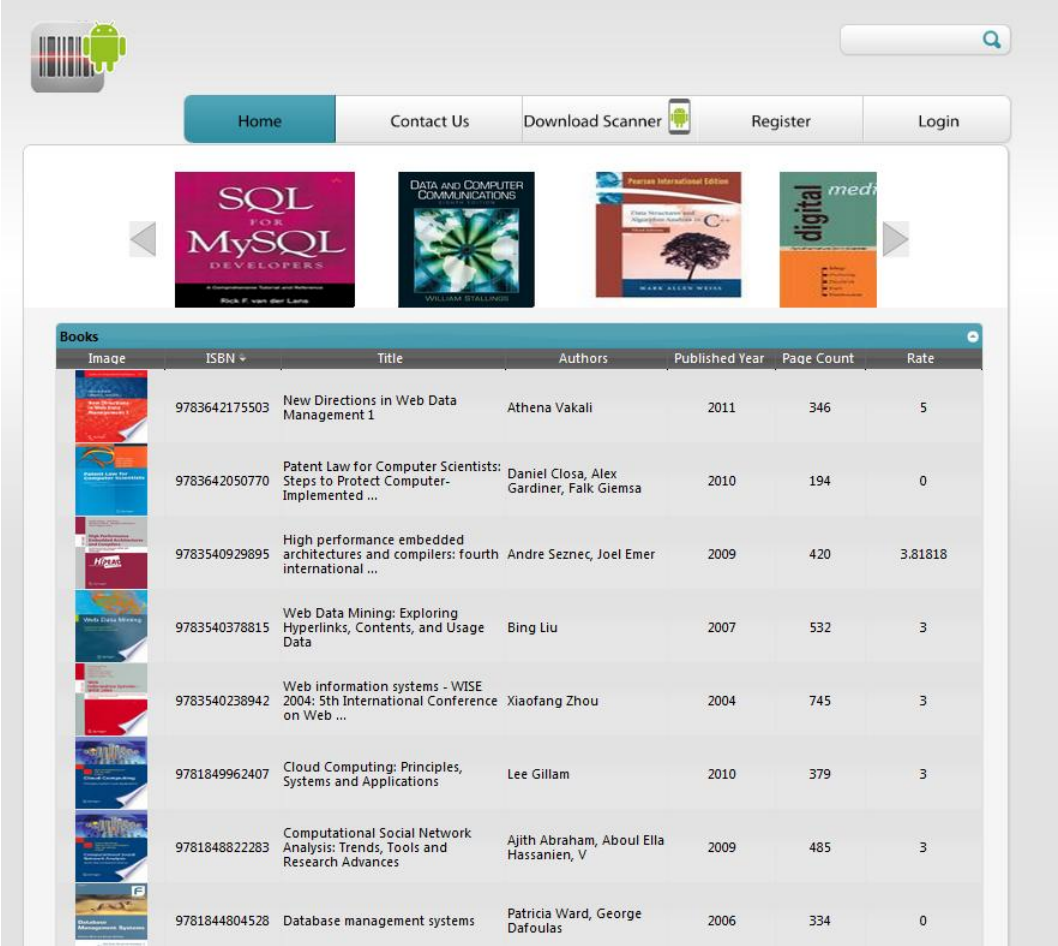


Σχήμα 4.1.11 : Ρυθμίσεις εφαρμογής

4.2 Ιστοσελίδα Smartscan

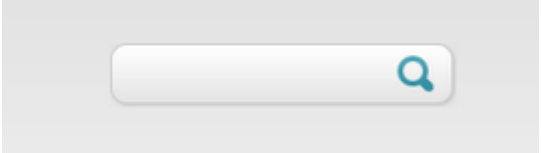
Η ιστοσελίδα Smartscan έχει σχεδιαστεί αποκλειστικά για την online παρουσίαση των βιβλίων που έχουν σαρωθεί από την smartscan (barcode scanner) εφαρμογή και βρίσκεται στον σύνδεσμο http://www.cs.ucy.ac.cy/thesis/smartscan/. Συγκεκριμένα, για όσα βιβλία έχουν σαρωθεί από την εφαρμογή παρουσιάζονται: η φωτογραφία εξωφύλλου, ο κωδικός (ISBN), ο τίτλος, ο/οι συγγραφείς, το έτος δημοσίευσης, το πλήθος των σελίδων όπως επίσης και η βαθμολογία τους (η οποία έχει παρθεί από τις ψηφοφορίες των χρηστών μελών). Στην ιστοσελίδα μπορεί κάποιος χρήστης να βρει πληροφορίες επικοινωνίας όπως επίσης και να κατεβάσει την smartscan εφαρμογή, η οποία και είναι το μέσο σάρωσης βιβλίων. Μέσω της ιστοσελίδας κάποιος χρήστης μπορεί να δημιουργήσει λογαριασμό και να συνδεθεί για πιο εξειδικευμένες λειτουργίες. Τέλος, κάποιος χρήστης μπορεί να αναζητήσει ένα βιβλίο μέσω της λειτουργίας αναζήτησης. Αξίζει να σημειωθεί ότι η συγκεκριμένη ιστοσελίδα έχει αναπτυχθεί έτσι ώστε να παρέχει στους χρήστες της τη δυνατότητα να την επισκέπτονται και να την χρησιμοποιούν από την πλειονότητα των web browsers αλλά και από browsers στις έξυπνες κινητές τους συσκευές (Cross-browser).

Κατά την είσοδο στην ιστοσελίδα παρουσιάζεται η αρχική σελίδα (Home) όπως φαίνεται και στο Σχήμα 4.2.1. Κάθε χρήστης μπορεί να επιλέξει να πλοηγηθεί στις πληροφορίες επικοινωνίας (Contact as - Σχήμα 4.2.9), να κατεβάσει την εφαρμογή σάρωσης των barcodes (Download Scanner) και να δημιουργήσει λογαριασμό (Register - Σχήμα 4.2.10). Οι εγγεγραμμένοι χρήστες, δηλαδή οι χρήστες που έχουν ήδη δημιουργήσει λογαριασμό (μέσω του Register) μπορούν να συνδεθούν μέσω της επιλογής login (Σχήμα 4.2.11) και να πλοηγηθούν στις πιο εξειδικευμένες λειτουργίες που παρέχονται από την ιστοσελίδα. Στην αρχική σελίδα υπάρχει ένας πίνακας (jQgrid [22]) ο οποίος εμφανίζει την εικόνα εξωφύλλου του βιβλίου, τον κωδικό (ISBN) του βιβλίου, τον τίτλο του βιβλίου, τον/τους συγγραφείς του βιβλίου, το έτος δημοσίευσης του βιβλίου, το πλήθος των σελίδων όπως επίσης και τη βαθμολογία του βιβλίου (η οποία έχει παρθεί κι έχει διαμορφωθεί από τις μέχρι στιγμής ψηφοφορίες των χρηστών μελών).



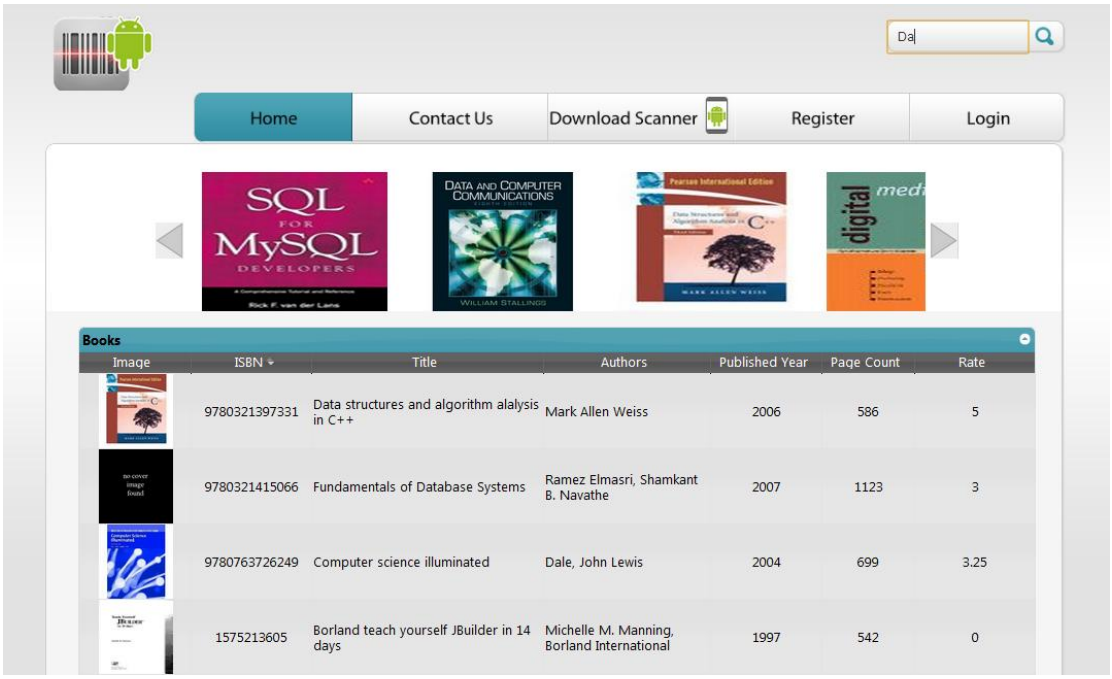
Σχήμα 4.2.1 : Αρχική σελίδα

Μέσω της λειτουργίας αναζήτησης που βρίσκεται στο πάνω μέρος της ιστοσελίδας (Σχήμα 4.2.2) κάθε χρήστης μπορεί να κάνει αναζήτηση σε όλα τα στοιχεία της βάσης δεδομένων που αφορούν τα πεδία που βρίσκονται στον πίνακα που εμφανίζεται. Καθώς ο χρήστης γράφει κάποια λέξη κλειδί για αναζήτηση τα δεδομένα που εκτυπώνονται στην οθόνη αναναιώνεται κάθε 500ms (εφόσον ο χρήστης συνεχίζει να γράφει) έτσι ώστε να φιλτράρει τα αποτελέσματα βάση της λέξης κλειδί μέχρι στιγμής (Σχήμα 4.2.3 και Σχήμα 4.2.4). Επίσης, αναζήτηση μπορεί να εκτελεστεί εφόσον ο χρήστης καταχωρήσει κάποια λέξη κλειδί και έπειτα πατήσει το πλήκτρο «Enter».

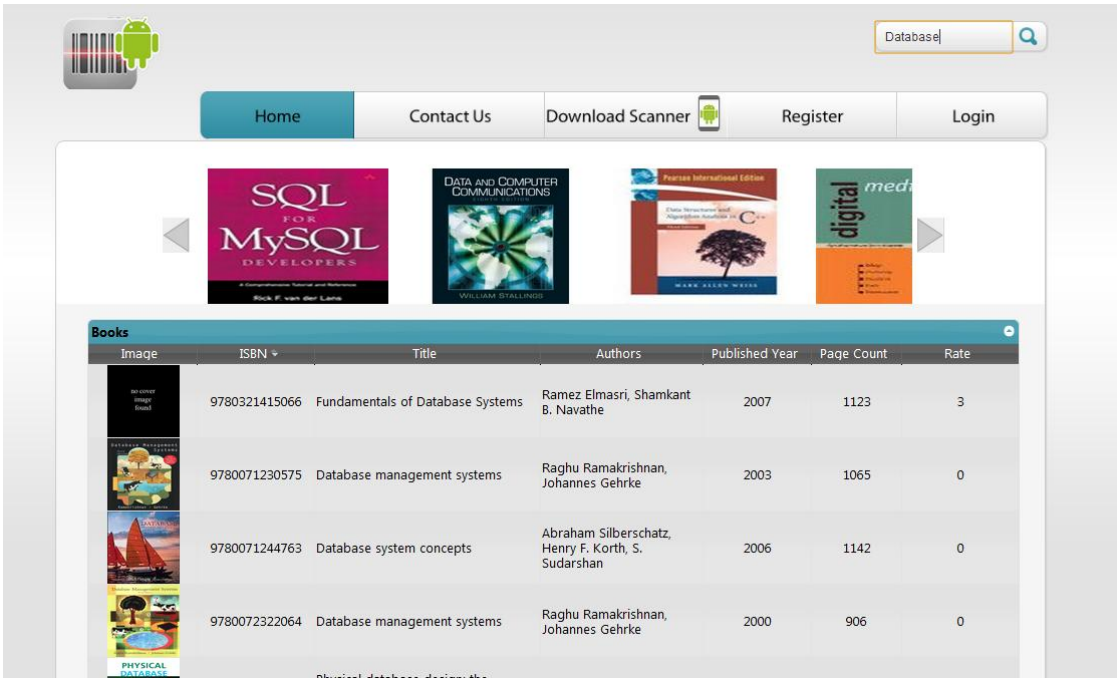


Σχήμα 4.2.2 : Αναζήτηση

Τέλος, κάτω από το μενού επιλογών εμφανίζεται ένα jQuery carousel[13][23] το οποίο παρουσιάζει τα πρώτα 20 βιβλία με την υψηλότερη βαθμολογία. Μπορούμε να πλοηγηθούμε στο συγκεκριμένο μενού κινώντας δεξιά/αριστερά τις φωτογραφίες με τα βιβλία που έχουν συλλέξει την υψηλότερη βαθμολογία (Top Rated) από τους χρήστες μέλη της ιστοσελίδας.

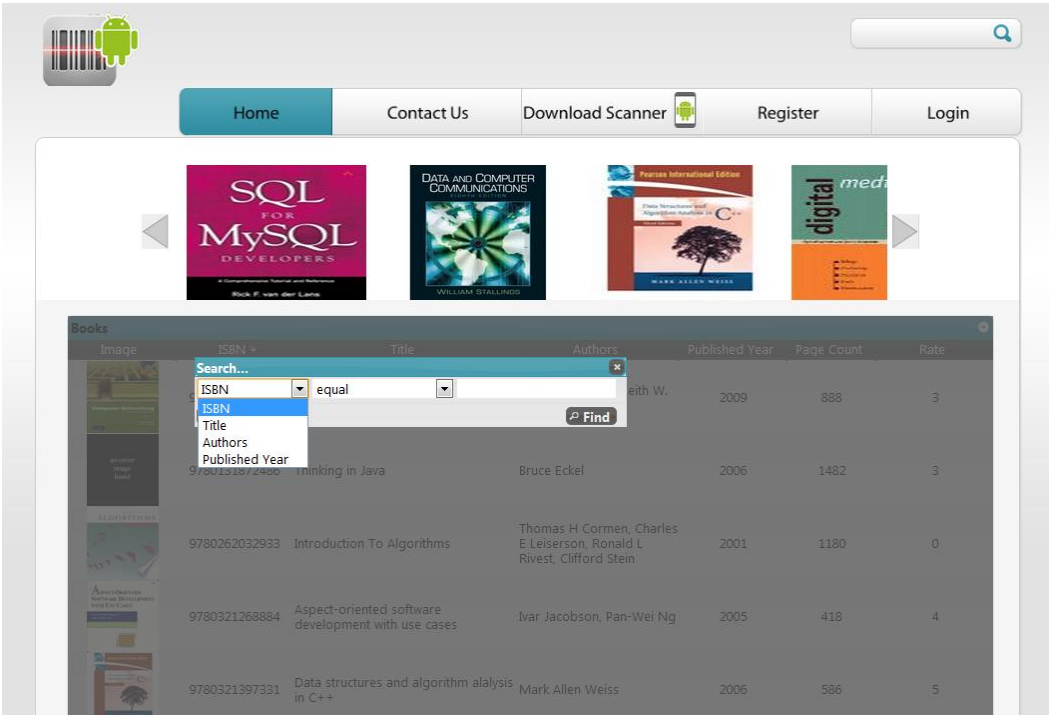


Σχήμα 4.2.3 : Αναζήτηση

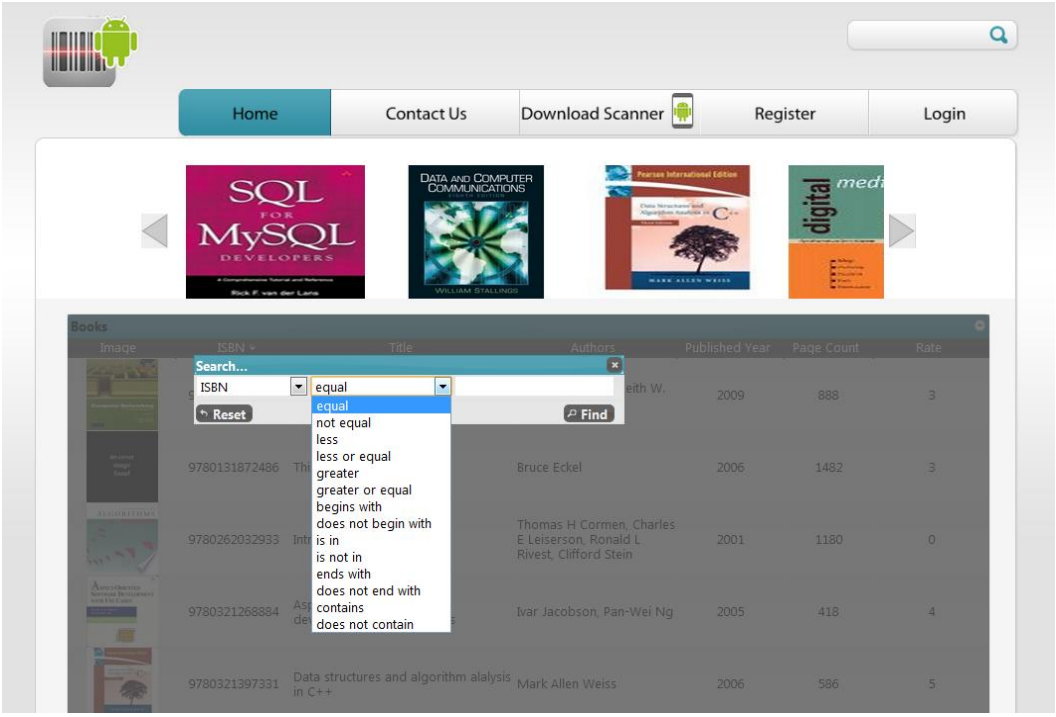


Σχήμα 4.2.4 : Αναζήτηση

Ο χρήστης έχει την δυνατότητα για μία πιο εξειδικευμένη αναζήτηση από αυτήν που παρουσιάσαμε πιο πάνω, Συγκεκριμένα, μπορεί να γίνει αναζήτηση στα πεδία του πίνακα που αφορούν ISBN, τίτλο, συγγραφείς και έτος έκδοσης (Σχήμα 4.2.5).

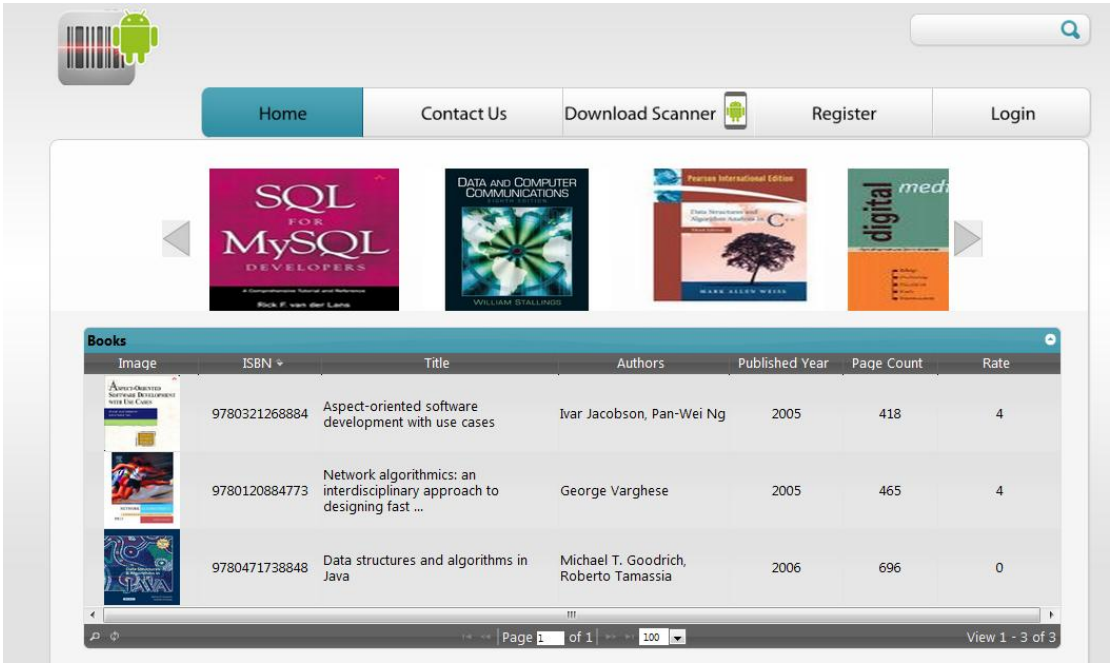


Σχήμα 4.2.5 : Εξειδικευμένη αναζήτηση

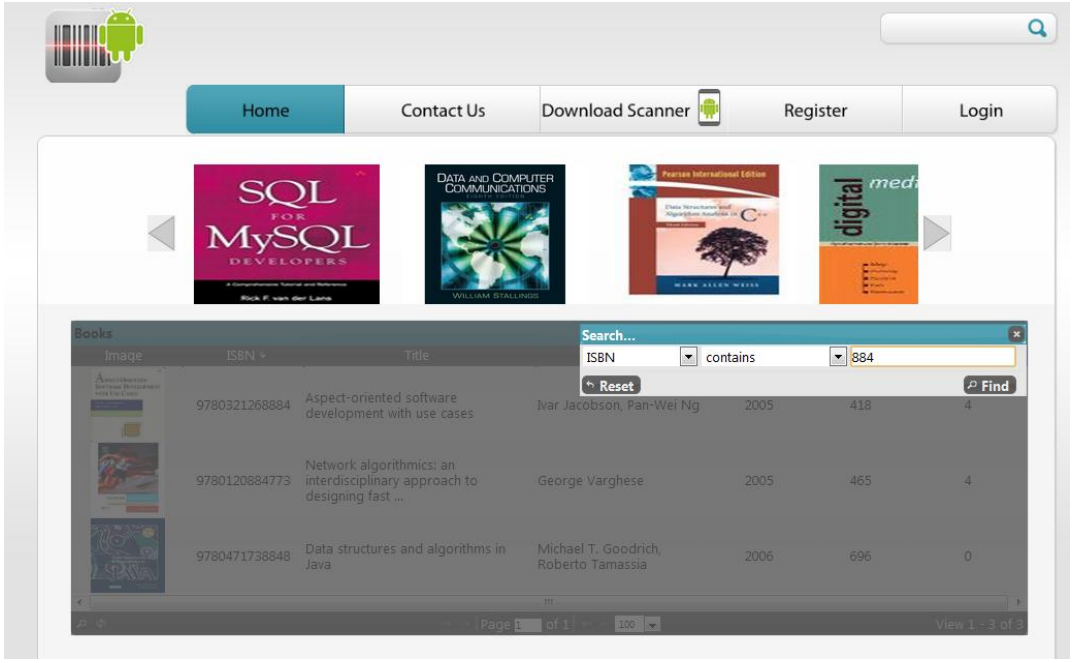


Σχήμα 4.2.6 : Εξειδικευμένη αναζήτηση

Ακόμη, για την αναζήτηση αυτή υπάρχουν πολλές επιλογές όσο αφορά το κάθε ένα από τα τέσσερα πεδία που προαναφέρθηκαν όπως είναι το «equal» δηλαδή η λέξη κλειδί που πληκτρολογείται να είναι ακριβώς αυτή καθεαυτή στο συγκεκριμένο πεδίο, «contains» δηλαδή η λέξη κλειδί που πληκτρολογείται να είναι υπολέξη στο συγκεκριμένο πεδίο κτλ (Σχήμα 4.2.6).

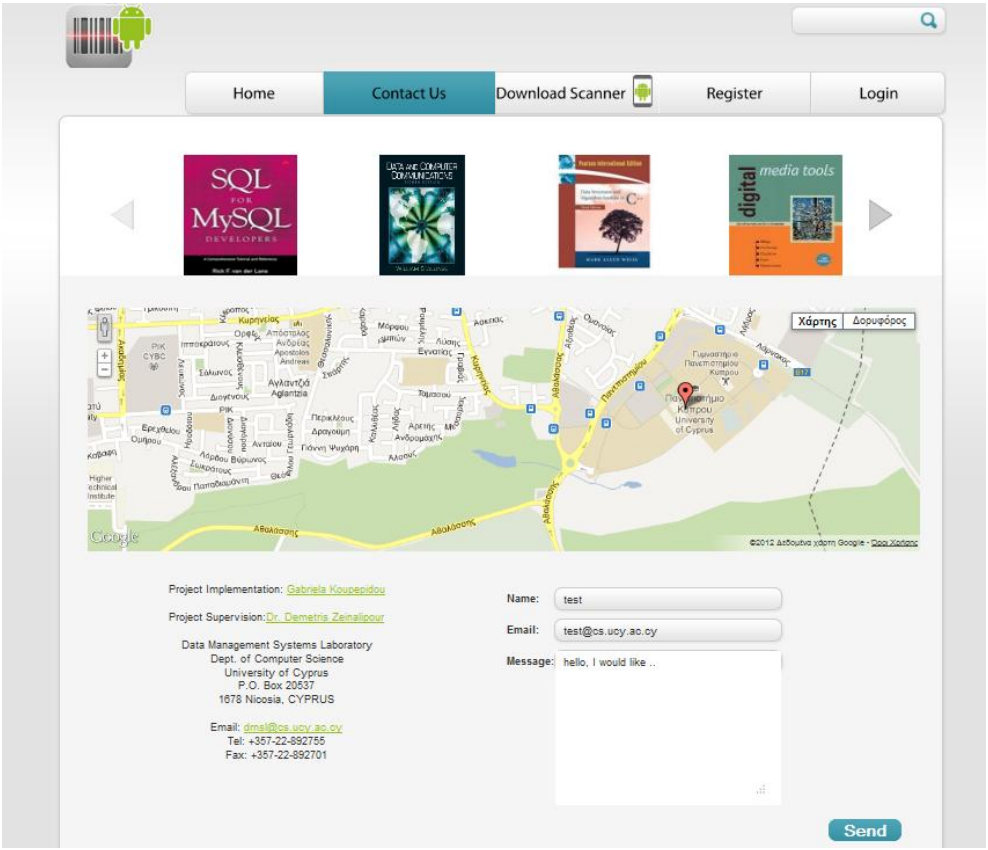


Σχήμα 4.2.7 : Εξειδικευμένη αναζήτηση βάση του ISBN



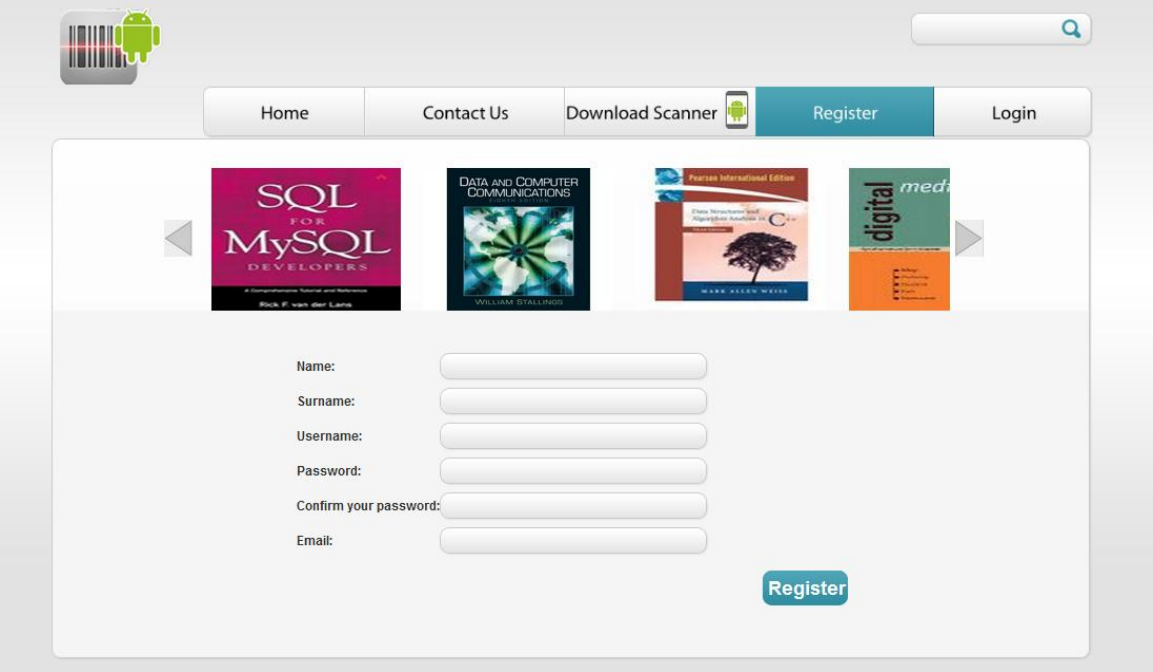
Σχήμα 4.2.8 : Εξειδικευμένη αναζήτηση βάση του ISBN

Ένα παράδειγμα εξειδικευμένης αναζήτησης είναι ο συνδυασμός: Search -> ISBN -> contains -> 884 το οποίο και μας επιστρέφει 3 βιβλία στα οποία εμπεριέχεται ο αριθμός «884» στο πεδίο κωδικού τους. (Σχήμα 4.2.7 και Σχήμα 4.2.8)



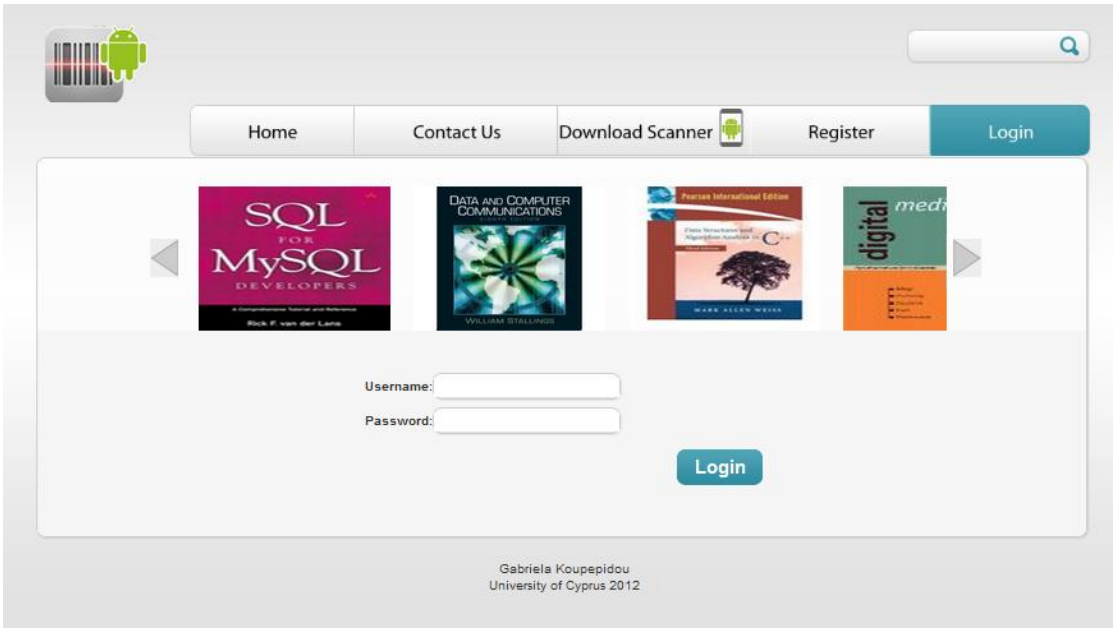
Σχήμα 4.2.9 : Πληροφορίες Επικοινωνίας

Ο χρήστης μέλος/επισκέπτης μπορεί να επικοινωνήσει μαζί μας μέσω email, fax ή τηλεφωνικώς στα στοιχεία που αναγράφονται στην επιλογή «Contact us» (Σχήμα 4.2.9). Ο χρήστης μέλος/επισκέπτης μπορεί να αποστείλει ηλεκτρονικό μήνυμα συμπληρώνοντας το όνομα του, το email του (αυτό στο οποίο επιθυμεί να σταλεί η απάντηση) όπως επίσης και το μήνυμα που επιθυμεί να αποστείλει. Το μήνυμα αυτό θα σταλεί αυτόματα στο διαχειριστή του συστήματος κι αφού τύχει επεξεργασίας θα σταλεί η ανάλογη απάντηση. Στην συγκεκριμένη επιλογή παρουσιάζεται και ένας χάρτης όπου ο χρήστης μπορεί να κατατοπισθεί πλήρως για το που ακριβώς βρισκόμαστε.



Σχήμα 4.2.10 : Δημιουργία Λογαριασμού

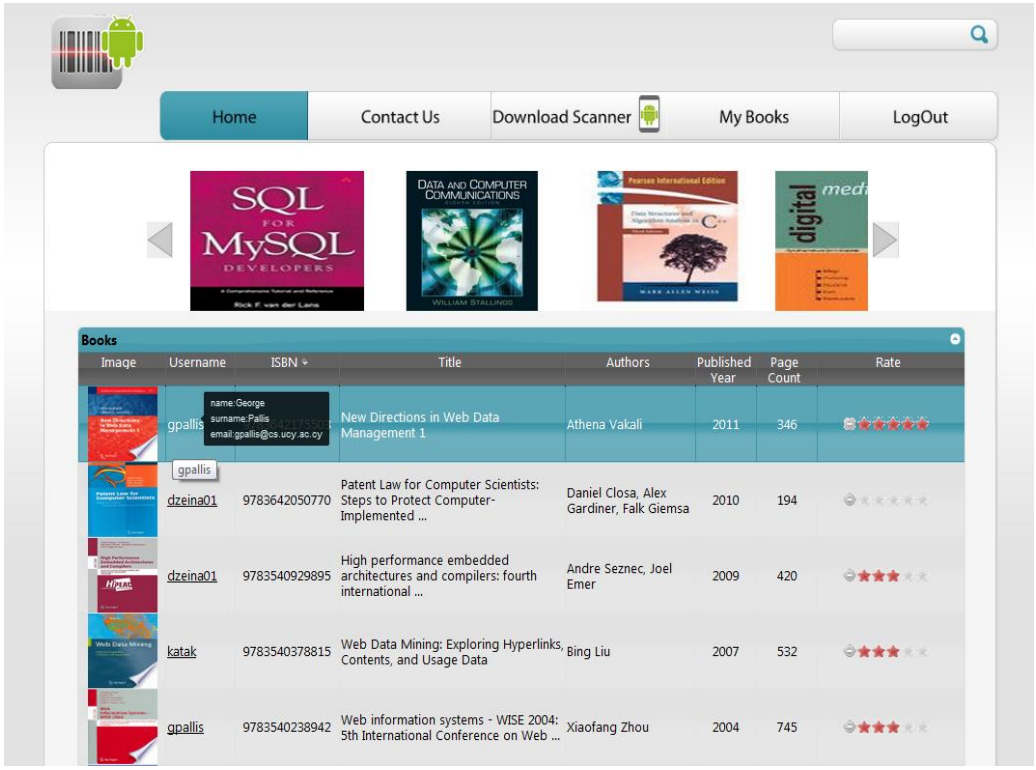
Ο χρήστης επισκέπτης μπορεί να δημιουργήσει λογαριασμό έτσι ώστε να γίνει χρήστης μέλος και να μπορεί να επωφεληθεί των περεταίρω λειτουργιών. Για την εγγραφή χρειάζονται το όνομα και το επίθετο του χρήστη, το συνθηματικό και ο κωδικός πρόσβασης όπως επίσης και το email του για ταυτοποίηση του. Ο χρήστης θα μπορεί να εγγραφεί μόνο όταν η αίτηση του γίνει αποδεκτή από τον διαχειριστή του συστήματος (Σχήμα 4.2.10).



Σχήμα 4.2.11 : Σύνδεση χρήστη

Ο χρήστης-μέλος μπορεί να συνδεθεί πλέον στο σύστημα πληκτρολογώντας το συνθηματικό και τον κωδικό πρόσβασης του (Σχήμα 4.2.11). Αν αυτά ταυτοποιηθούν από τα δεδομένα που υπάρχουν καταχωρημένα στη βάση δεδομένων τότε η σύνδεση του χρήστη είναι επιτυχής, αλλιώς εμφανίζονται μηνύματα λάθους στην οθόνη.

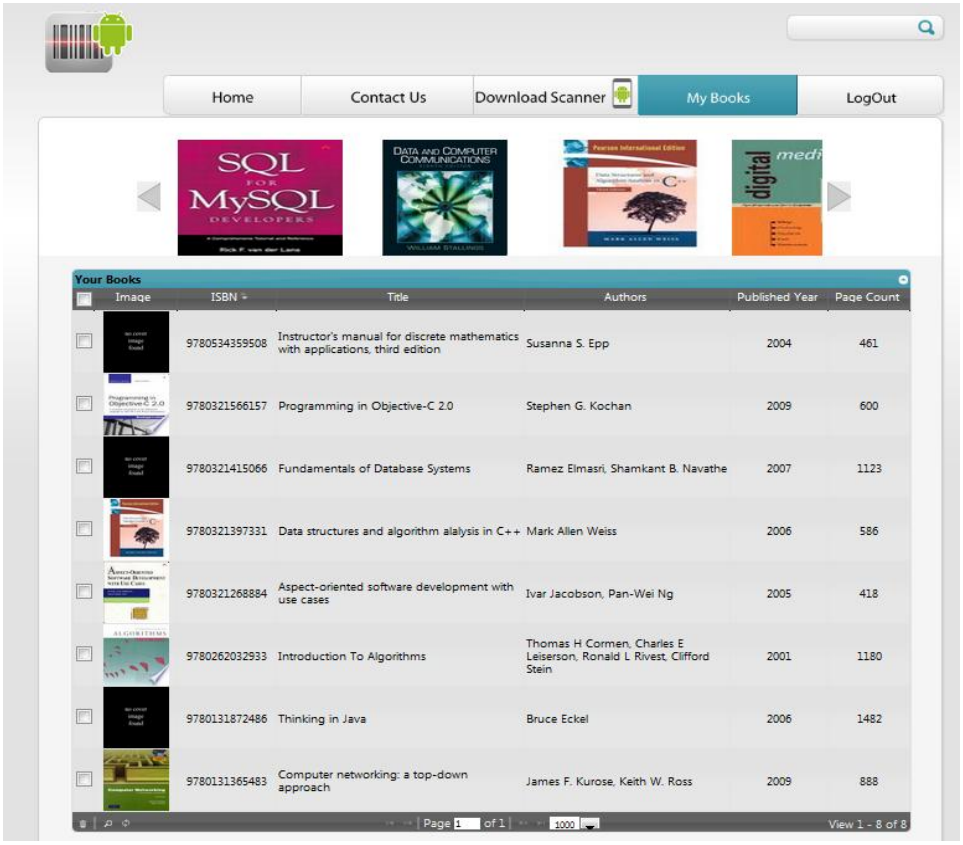
Ο συνδεδεμένος χρήστης-μέλος μπορεί πλέον να δει σε ποιον χρήστη ανήκει το κάθε βιβλίο που είναι καταχωρημένο στη βάση δεδομένων. Συγκεκριμένα κάθε μέλος μπορεί να δει το όνομα, το επίθετο και το email για κάθε ιδιοκτήτη κάθε βιβλίου. Επιπλέον, πατώντας στο username κάθε κάτοχου του βιβλίου το μέλος μπορεί να αποστείλει email σε αυτόν έτσι ώστε να αιτηθεί δανεισμό του συγκεκριμένου βιβλίου. Ακόμη, μία επιπλέον λειτουργία που προσφέρεται στα μέλη της ιστοσελίδας είναι η ικανότητα να βαθμολογήσουν/ψηφίσουν όποιο βιβλίο θέλουν με βαθμολογία από το 1-5 επιλέγοντας το αντίστοιχο πλήθος από αστεράκια που εμφανίζονται στην τελευταία στήλη. Η βαθμολογία για κάθε βιβλίο αναναιώνετα με βάση τις μέχρι στιγμής ψήφους που έχουν δοθεί σε κάθε βιβλίο ξεχωριστά(Σχήμα 4.2.12). Τέλος, επιλέγοντας οποιαδήποτε από τις οκτώ στήλες του πίνακα μπορεί να γίνει ταξινόμηση σε αύξουσα ή φθίνουσα σειρά ανάλογα για το συγκεκριμένο πεδίο.



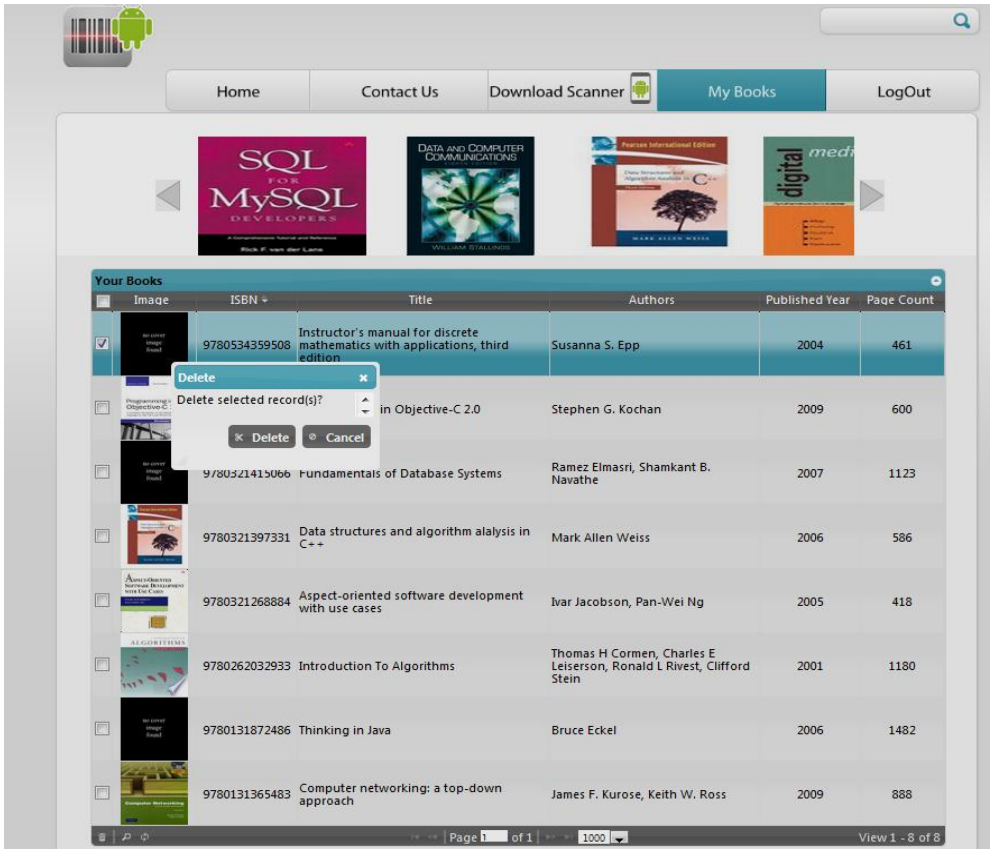
Σχήμα 4.2.12 : Αρχική σελίδα συνδεδεμένου χρήστη

Ο συνδεδεμένος χρήστης-μέλος μπορεί να δει την δική του προσωπική βιβλιοθήκη εάν μεταφερθεί στην επιλογή «My Books» (Σχήμα 4.2.14). Μέσω της επιλογής αυτής ο χρήστης έχει και πάλι όσες δυνατότητες είχε και στις προηγούμενες κατηγορίες, δηλαδή ταξινόμηση και αναζήτηση με την επιπλέον δυνατότητα διαγραφής βιβλίων σε περίπτωση που δεν επιθυμεί πλέον κάποια βιβλία να μην βρίσκονται στην online σελίδα. Η επιλογή διαγραφής βρίσκεται στο κάτω μέρος του πίνακα (Σχήμα 4.2.13) και γίνεται μόνο εφόσον ο χρήστης-μέλος επιλέξει τα βιβλία που θέλει να διαγράψει και εφόσον στην συνέχεια επιβεβαιώσει την απόφαση του (Σχήμα 4.2.15).

Σχήμα 4.2.13 : Διαγραφή Βιβλίων



Σχήμα 4.2.14 : Βιβλία συνδεδεμένου χρήστη



Σχήμα 4.2.15 : Διαγραφή Βιβλίων

Κεφάλαιο 5

Πειραματική Διαδικασία

5.1 Πειραματική Υποδομή

Σε αυτό το κεφάλαιο θα αναφέρουμε την διαδικασία που ακολουθήσαμε για να μπορούμε να πάρουμε κάποιες μετρήσεις. Στόχος είναι να εξετάσουμε την εγκυρότητα του smartscan (εφαρμογή και ιστοσελίδα) όπως επίσης και την απόδοση σε χρόνο. Επίσης θα εξεταστεί και το ποσοστό αποτυχίας της εφαρμογής να σαρώσει κάποια barcodes. Θα παρουσιάσουμε τις μετρήσεις μας μέσω γραφικών παραστάσεων.

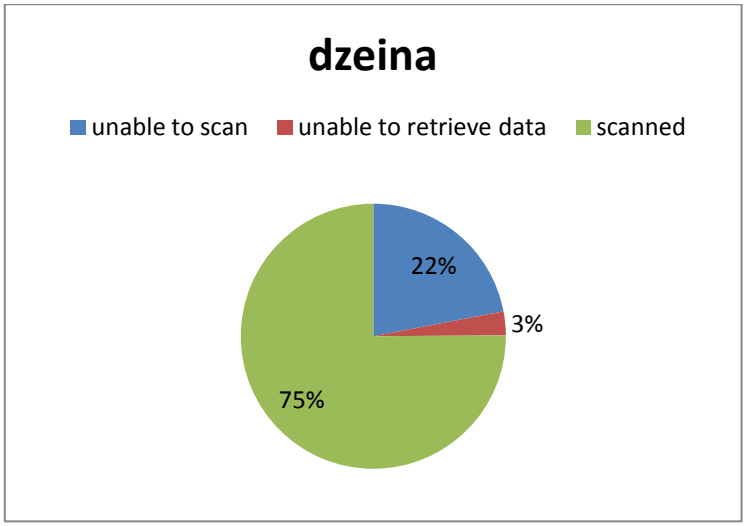
Για να μπορέσουμε να πάρουμε της αναγκαίες μετρήσεις για την αξιολόγηση των αποτελεσμάτων χρησιμοποιήσαμε την Android Smartphone κινητή συσκευή HTC Desire η οποία παραχωρήθηκε από τον επιβλέπον καθηγητή της εν λόγω ατομικής διπλωματικής εργασίας. Για να παρθούν οι μετρήσεις συνέβαλαν αρκετοί καθηγητές του τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου, οι οποίοι μας επέτρεψαν και μας παραχώρησαν τα βιβλία από τις προσωπικές τους βιβλιοθήκες έτσι ώστε να τα σαρώσουμε. Με αυτό τον τρόπο μπορέσαμε να καταλήξουμε σε διάφορα συμπεράσματα όπως επίσης και να επεκτείνουμε τη νοητή βιβλιοθήκη που δημιουργήσαμε. Ένα Crowdsourcing στοιχείο του Smartscan συστήματος είναι ο εθελοντισμός που αναδείχθηκε από τα πλέον εγγεγραμμένα μέλη έτσι ώστε νε εμπλουτιστεί η βάση δεδομένων που αποτελείται αποκλειστικά από βιβλία. Με τον τρόπο αυτό έχουμε διαμορφώσει μία πλούσια βιβλιοθήκη, κάνοντας την επιθυμία δανεισμού οποιουδήποτε βιβλίου πολύ εύκολη και προπάντων εφικτή.

5.2 Πειραματική Διαδικασία και Μετρήσεις

Μετρήσεις από τη σάρωση βιβλίων

κος. Δημήτρης Ζεϊναλιπούρ

Συνολικά στη βιβλιοθήκη του κύριου Δημήτρη Ζεϊναλιπούρ υπήρχαν 234 βιβλία με barcode στο πίσω μέρος τους εξωφύλλου εκ των οποίων τα 53 δεν ήταν δυνατόν να σαρωθούν λόγω του ότι το barcode είτε ήταν πολύ μικρό είτε δεν ήταν σε καλή ανάλυση. Επίσης 7 βιβλία κατέστη αδύνατον να ανακτηθούν τα στοιχεία τους από την βάση δεδομένων προϊόντων της Google (δηλαδή τα συγκεκριμένα βιβλία δεν ήταν καταχωρημένα στη βάση δεδομένων της Google Books). Για τη σάρωση των 181 βιβλίων αλλά και τη φωτογράφηση των υπόλοιπων 53(που δεν ήταν εφικτό να σαρωθούν) χρειάστηκαν συνολικά 106 λεπτά.

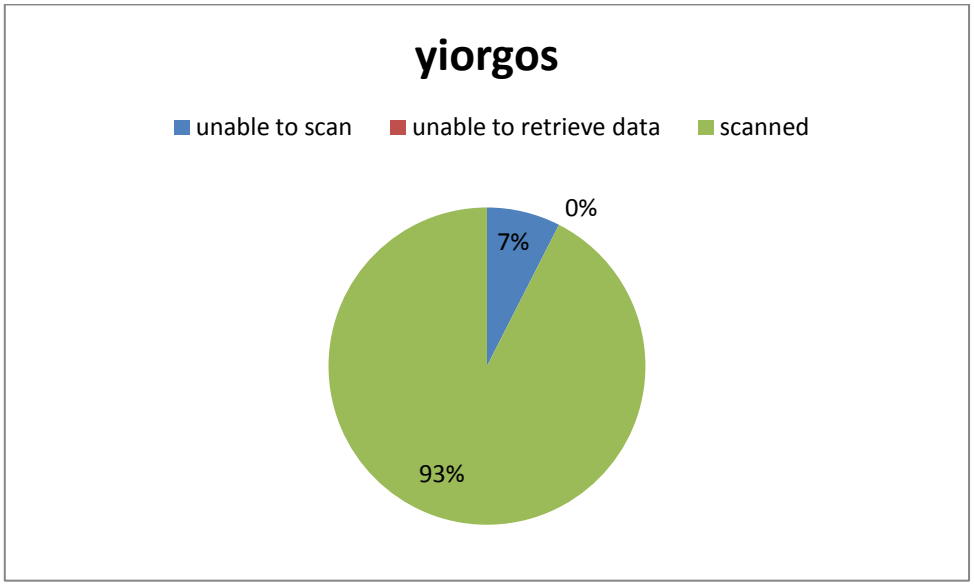


Παρατηρούμε ότι είχαμε μία απώλεια της τάξεως του 25%, λόγω του ότι 53 βιβλία δεν ήταν δυνατόν να σαρωθούν και για 7 βιβλία δεν ήταν δυνατόν να ανακτηθούν οι πληροφορίες τους από το Google Books API.

Επίσης, για τη σάρωση 181 βιβλίων και τη φωτογράφηση ακόμα 53 χρειάστηκαν 106 λεπτά. Αυτό μας δίνει ένα μέσο όρο 27 δευτερόλεπτα για το κάθε βιβλίο. Τα αποτελέσματα για τη σάρωση θα είναι πιο αντικειμενικά στις περιπτώσεις που έχουμε λιγότερες απώλειες.

κος. Γιώργος Χρυσάνθου

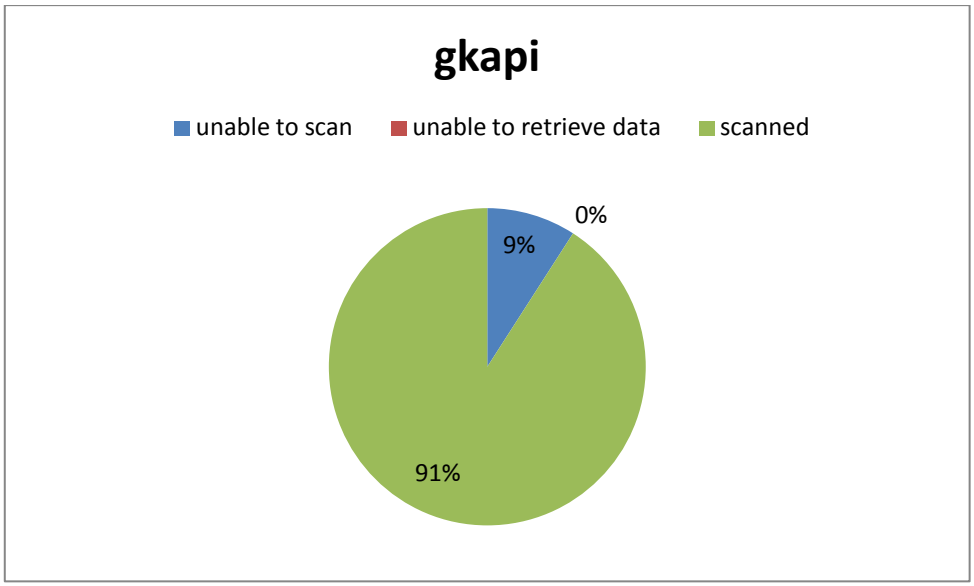
Συνολικά 120 βιβλία εκ των οποίων τα 9 δεν ήταν δυνατόν να σαρωθούν λόγω του ότι το barcode είτε ήταν πολύ μικρό είτε δεν ήταν σε καλή ανάλυση. Για τη σάρωση των 111 βιβλίων αλλά και τη φωτογράφηση των υπόλοιπων 9(που δεν ήταν εφικτό να σαρωθούν) χρειάστηκαν 60 λεπτά.



Παρατηρούμε ότι είχαμε μία απώλεια της τάξεως του 7%. Για τη σάρωση 111 βιβλίων και τη φωτογράφηση ακόμα 9 χρειάστηκαν 60 λεπτά. Αυτό μας δίνει ένα μέσο όρο 30 δευτερόλεπτα για το κάθε βιβλίο.

κα. Γεωργία Καπιτσάκη

Συνολικά 22 βιβλία εκ των οποίων τα 2 δεν ήταν δυνατόν να σαρωθούν λόγω του ότι το barcode είτε ήταν πολύ μικρό είτε δεν ήταν σε καλή ανάλυση. Για τη σάρωση των 20 βιβλίων αλλά και τη φωτογράφηση των υπόλοιπων 2 (που δεν ήταν εφικτό να σαρωθούν) χρειάστηκαν 8 λεπτά.

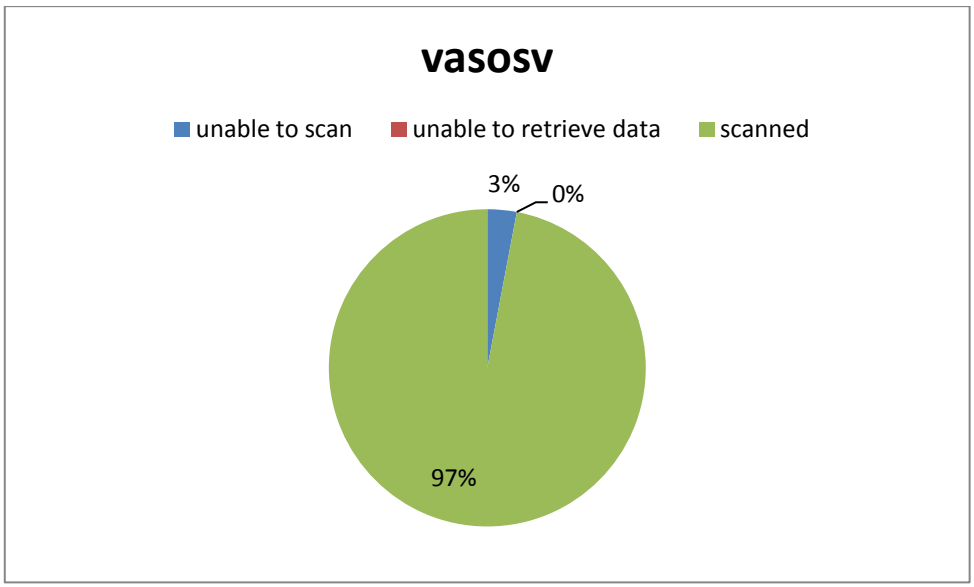


Παρατηρούμε ότι είχαμε μία απώλεια της τάξεως του 14%. Για τη σάρωση 20 βιβλίων και τη φωτογράφηση ακόμα 2 χρειάστηκαν 8 λεπτά. Αυτό μας δίνει ένα μέσο όρο 21 δευτερόλεπτα για το κάθε βιβλίο.

κος. Βάσος Βασσιλείου

Συνολικά 67 βιβλία εκ των οποίων τα 2 δεν ήταν δυνατόν να σαρωθούν λόγω του ότι το barcode είτε ήταν πολύ μικρό είτε δεν ήταν σε καλή ανάλυση.

Για τη σάρωση των 65 βιβλίων αλλά και τη φωτογράφηση των υπόλοιπων 2 (που δεν ήταν εφικτό να σαρωθούν) χρειάστηκαν 18 λεπτά.

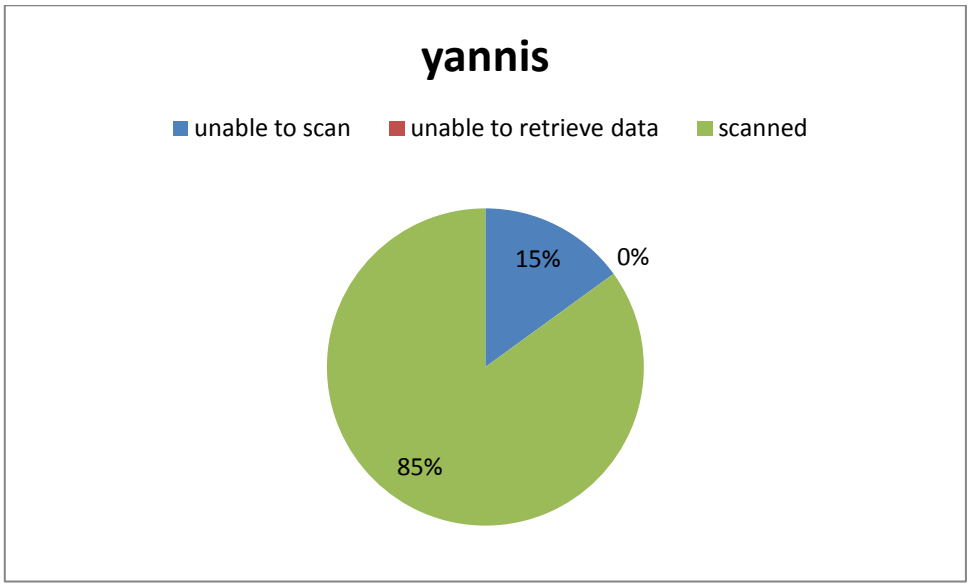


Παρατηρούμε ότι είχαμε μία απώλεια της τάξεως του 3%. Για τη σάρωση 65 βιβλίων και τη φωτογράφηση ακόμα 2 χρειάστηκαν 18 λεπτά. Αυτό μας δίνει ένα μέσο όρο 16 δευτερόλεπτα για το κάθε βιβλίο.

κος. Γιάννης Δημόπουλος

Συνολικά 40 βιβλία εκ των οποίων τα 6 δεν ήταν δυνατόν να σαρωθούν λόγω του ότι το barcode είτε ήταν πολύ μικρό είτε δεν ήταν σε καλή ανάλυση.

Για τη σάρωση των 34 βιβλίων αλλά και τη φωτογράφηση των υπόλοιπων 6 (που δεν ήταν εφικτό να σαρωθούν) χρειάστηκαν 12 λεπτά.

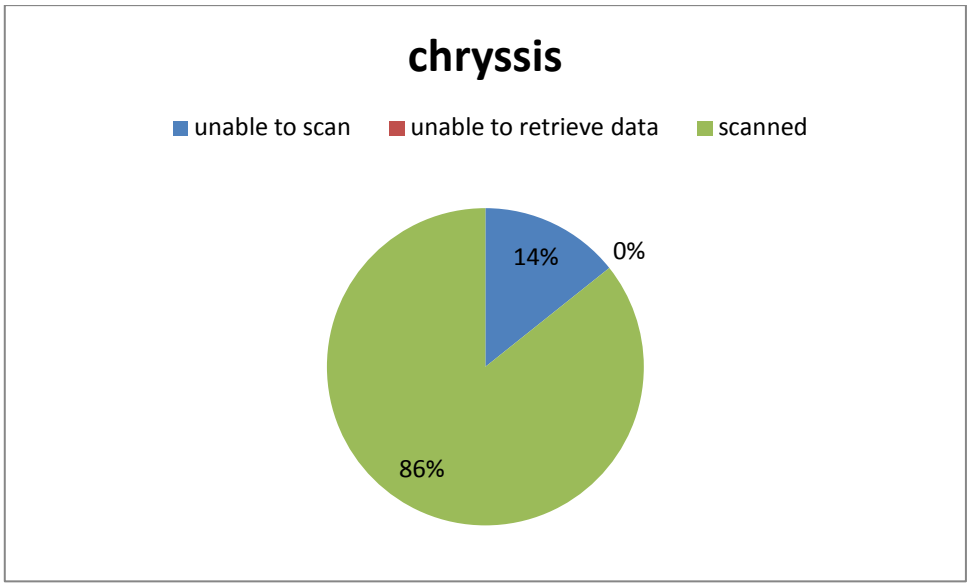


Παρατηρούμε ότι είχαμε μία απώλεια της τάξεως του 15%. Για τη σάρωση 34 βιβλίων και τη φωτογράφηση ακόμα 6 χρειάστηκαν 12 λεπτά. Αυτό μας δίνει ένα μέσο όρο 18 δευτερόλεπτα για το κάθε βιβλίο.

κος. Χρύσης Γεωργίου

Συνολικά 63 βιβλία εκ των οποίων τα 9 δεν ήταν δυνατόν να σαρωθούν λόγω του ότι το barcode είτε ήταν πολύ μικρό είτε δεν ήταν σε καλή ανάλυση.

Για τη σάρωση των 54 βιβλίων αλλά και τη φωτογράφηση των υπόλοιπων 9 (που δεν ήταν εφικτό να σαρωθούν) χρειάστηκαν 22 λεπτά.



Παρατηρούμε ότι είχαμε μία απώλεια της τάξεως του 14%. Για τη σάρωση 54 βιβλίων και τη φωτογράφηση ακόμα 9 χρειάστηκαν 22 λεπτά. Αυτό μας δίνει ένα μέσο όρο 20 δευτερόλεπτα για το κάθε βιβλίο.

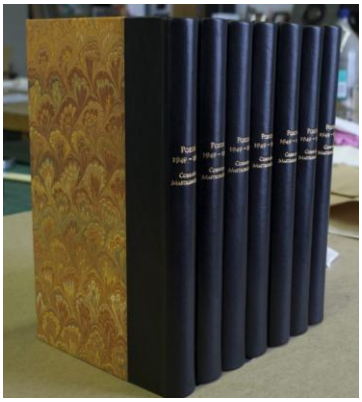
Γενικά, κάθε βιβλίο χρειάζεται γύρω στα 22 δευτερόλεπτα. Ο χρόνος αυτός θα ήτανε αρκετά πιο μικρός εάν δεν είχαμε δυσκολίες στην σάρωση κάποιων βιβλίων. Ήδη από τις μετρήσεις μας παρατηρούμε ότι στις περιπτώσεις που έχουμε μικρά ποσοστά απώλειας (unable to scan) τότε ο μέσος χρόνος για να σαρωθεί κάποιο βιβλίο είναι μικρότερος. Αυτό συμβαίνει γιατί η διαδικασία φωτογράφησης παίρνει χρόνο, αφού χρειάζεται να αποσυνδεθούμε από την smartscan εφαρμογή και να ανοίξουμε την κάμερα ώστε να φωτογραφίσουμε το συγκεκριμένο barcode με σκοπό να το εισάγουμε στη βάση δεδομένων αργότερα, μέσω της δεύτερης λειτουργίας που παρέχεται από την εφαρμογή μας.

Μία μελλοντική επέκταση θα ήταν η δυνατότητα φωτογράφησης των barcodes ενώ βρισκόμαστε στην Smartscan εφαρμογή. Για παράδειγμα, καθώς προσπαθούμε να σαρώσουμε κάποιο συγκεκριμένο βιβλίο κι αφού έχουμε ήδη αποτύχει στις πρώτες πέντε προσπάθειες τότε μία λύση θα ήταν να ενεργοποιείται η κάμερα της έξυπνης κινητής συσκευής έτσι ώστε να φωτογραφηθεί το συγκεκριμένο barcode.

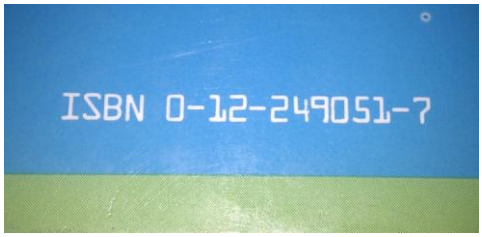
- Παρατήρηση: Δεν κατέστη εφικτό να σαρωθεί το 100% των βιβλίων που υπήρχαν στις προσωπικές βιβλιοθήκες των συμμετοχόντων.

  Κάποια βιβλία δεν έχουν barcode στο πίσω μέρος του εξωφύλλου τους λόγω του ότι δεν αποτελούν αντίτυπο βιβλίου, δηλαδή είναι για παράδειγμα έκδοση που αφορά σε κάποιο συνέδριο, κάποια εγκυκλοπαίδεια κτλ.

  Επίσης, κάποια βιβλία ενώ έχουν κωδικό ISBN αυτός ο κωδικός δεν βρίσκεται κρυπτογραφημένος σε μορφή barcode αλλά αναγράφεται υπό τη μορφή κανονικών αριθμών.
  Τέλος, σε κάποια βιβλία ήταν αδύνατο να σαρωθεί το barcode είτε επειδή το barcode ήταν πολύ μικρό είτε επειδή δεν ήταν σε καλή ανάλυση είτε επειδή κάτι άλλο το επισκίαζε.



Σχήμα 5.2.1 :Περίπτωση 1



Σχήμα 5.2.2 :Περίπτωση 2

Σχήμα 5.2.3 :Περίπτωση 3

Google Analytics

Το Google Analytics (GA)[24] είναι μία δωρεάν υπηρεσία που προσφέρεται από το Google και δημιουργεί λεπτομερή στατιστικά στοιχεία σχετικά με τους επισκέπτες σε ένα ιστοχώρο. Το GA μπορεί να παρακολουθεί τους επισκέπτες από όλες τις αναφορές, συμπεριλαμβανομένου των μηχανών αναζήτησης που χρησιμοποιούνται, τις διαφημίσεις που εμφανίζονται, τα pay-per-click δίκτυα, την αγορά ηλεκτρονικού ταχυδρομείου, ακόμα και την ψηφιακή ασφάλεια που υπάρχει.

Η υπηρεσία Google Analytics προσφέρει διάφορους τύπους αναφορών. Ανάμεσα σε αυτούς είναι:

- Αναφορά σε πραγματικό χρόνο: Μετρά την δραστηριότητα στην ιστοσελίδα καθώς συμβαίνει. Παρουσιάζει πόσοι άνθρωποι βρίσκονται τώρα στην ιστοσελίδα, από πού ήρθαν, και τι βλέπουν. Σε πραγματικό χρόνο μπορεί κάποιος να γνωρίζει αν το περιεχόμενο της ιστοσελίδας του είναι δημοφιλείς, εάν η προώθηση που γίνεται είναι καλή για την ιστοσελίδα, και μπορεί να δεί άμεσα αποτελέσματα από tweets και blog posts.
- Προσαρμοσμένες Αναφορές: Καθορίζονται οι πληροφορίες που θέλει κάποιος να αναλύσει. Μπορεί να φτιάξει τον δικό του πίνακα μετρήσεων και έχει άμεση πρόσβαση στις απαντήσεις που χρειάζεται.

Με τη χρήση της υπηρεσίας αυτής επιλέξαμε να δημιουργήσουμε στατιστικά στοιχεία σχετικά με τους επισκέπτες της Smartscan ιστοσελίδας μας. Με αυτό τον τρόπο θελήσαμε να καταγράψουμε με ποιο μέσο επισκέπτονται οι χρήστες την ιστοσελίδα μας (μέσω ποιας μηχανής αναζήτησης). Επίσης, με το Google Analytics θα μπορούσαμε να δούμε σε πραγματικό χρόνο πόσοι χρήστες είναι ανά πάσα στιγή στην Smartscan ιστοσελίδα (πόσο δημοφιλείς είναι η ιστοσελίδα) αλλά και πόσοι χρήστες έχουν επισκευθεί την ιστοσελίδα γενικότερα.

Κάτι τέτοιο δυστυχώς δεν κατέστη εφικτό αφού η ιστοσελίδα μας δεν είναι ακόμη διαδεδομένη προς το κοινό και χρησιμοποιείται κυρίως από τον διαχειριστή της.

Κεφάλαιο 6

Συμπεράσματα και Μελλοντικές Επεκτάσεις

Το crowdsourcing είναι ένα αναδυόμενο πεδίο στις Smartphone συσκευές και αναμένουμε ότι θα εξελιχθεί ακόμα πιο γρήγορα στο μέλλον. Τα Smartphones και οι εφαρμογές τους πλέον αποτελούν ένα νέο σύστημα υπολογισμού που αποτελεί τον συνδετικό κρίκο μεταξύ των υπολογιστών και των ανθρώπων.

Το μακροπρόθεσμο όραμα μας περιλαμβάνει την επέκταση του συστήματος σε παγκόσμιο επίπεδο ώστε το Smartscan σύστημα να αποτελεί μία online βιβλιοθήκη μέσω της οποίας κάποιος χρήστης θα μπορεί να εντοπίσει και να δανειστεί κάποιο βιβλίο. Κάποιος χρήστης θα έχει τη δυνατότητα να επιλέξει κάποιο βιβλίο είτε βάση της βαθμολογίας του, είτε βάση κάποιας συγκεκριμένης κατηγορίας π.χ. θέμα, συγγραφείς κτλ.

Κεφάλαιο 7

Βιβλιογραφία και επεξήγηση

[1]. Tingxin Yan, Vikas Kumar, Deepak Ganesan, "Crowdsearch:Exploiting Crowds for Accurate real-time Image Search on Mobile Phones" In MobilSys, 2010

[2]. Demetrios Zeinalipour-Yazti, Christos Laoudias, Constandinos Costa, Michalis Vlachos, Maria I. Andreou, Dimitrios Gunopulos, "Crowdsourced Trace Similarity with Smartphones"

[3]. Adam Marcus, Eugene Wu, David R. Karger, Samuel Madden, Robert C. Miller, "Crowdsourced Databases:Query Processing with People" In CIDR, 2011

[4]. Aditya Parameswaran, Neoklis Polyzoti, "Answering Queries using Humans, Algorithms and Databases" In CIDR, 2011

[5]. C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour-Yazti, C. G. Panayiotou, "A Platform for the Evaluation of Fingerprint Positioning Algorithms on Android Smartphones" In MobilSys, 2012

[6]. G. Chatzimilioudis, A. Konstantinidis, C. Laoudias, D. Zeinalipour-Yazti, "Crowdsourcing with Smartphones", In IC, 2012

[7]. Android, Ying Huang, "Begin Android Journey In Hours", 2009, http://developer.android.com/index.html

[8]. HTML, http://www.w3schools.com/html/default.asp

[9]. PHP, http://www.w3schools.com/php/default.asp

[10]. Zxing, http://code.google.com/p/zxing/

[11]. HTC, http://en.wikipedia.org/wiki/HTC

[12]. Smartphones, http://en.wikipedia.org/wiki/Smartphone

[13]. JavaScript, http://www.w3schools.com/js/default.asp,

[14]. jQuery, http://www.w3schools.com/jquery/default.asp

[15]. AJAX, http://www.w3schools.com/ajax/default.asp

[16]. MySQL, http://www.cs.ucy.ac.cy/~dzeina/courses/epl342/, http://el.wikipedia.org/wiki/MySQL

[17]. Android Market, https://play.google.com/store/apps/details?id=com.google.android.finsky&hl=el

[18]. Data Matrix, http://en.wikipedia.org/wiki/Data_Matrix

[19]. QR Codes, http://qrcode.kaywa.com/, http://en.wikipedia.org/wiki/QR_code

[20]. HTTP, http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol, http://en.wikipedia.org/wiki/HTTP_Secure

[21]. JSON, http://json.org/, http://en.wikipedia.org/wiki/JSON

[22]. PHP Script, http://www.freewebmasterhelp.com/tutorials/php

[23]. jQgrid,http://www.trirand.com/blog/jqgrid/jqgrid.html, http://www.trirand.com/blog/jqgrid/jqgrid.html

[24]. jCarousel, http://sorgalla.com/jcarousel/

[25]. Google Analytics, http://en.wikipedia.org/wiki/Google_Analytics, http://www.google.com/analytics/

[26]. Crowdsourcing, http://el.wikipedia.org/wiki/Crowdsourcing

[27]. Amazon Mechanical Turk, https://www.mturk.com/mturk/welcome, http://en.wikipedia.org/wiki/Amazon_Mechanical_Turk

[28]. oDesk, https://www.odesk.com/?_redirected, http://en.wikipedia.org/wiki/ODesk

[29]. AbeBooks API, http://www.programmableweb.com/api/abebooks

[30]. Bertram Books API, http://www.programmableweb.com/api/bertram-books

[31]. Books APIs, http://blog.programmableweb.com/2012/03/13/53-books-apis-google-books-goodreads-and-sharedbook

Παράρτημα Α

Πηγαίος Κώδικας

A.1 Smartscan Εφαρμογή

InsertBookByISBN.java

```
package com.google.zxing.client.android.smartscan;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLDecoder;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
```



```
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.zxing.client.android.R;

public class insertBookByISBN extends Activity {
	/** Called when the activity is first created. */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.insert_book);
		connectButtons();
	}

	void connectButtons() {
		Button insertButton = (Button)
findViewById(R.id.btn_insert);

		insertButton.setOnClickListener(new
Button.OnClickListener() {

			@Override
			public void onClick(View v) {

				TextView isbn = (TextView)
findViewById(R.id.et_bookISBN);

				String isbn1;

				isbn1 = (String) isbn.getText().toString();

				try {
					findBook(isbn1);
				} catch (IOException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				} catch (JSONException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}

			}
		});
```

```
	}

	void findBook(String isbn) throws IOException, JSONException {

/*************************************************************/
		try {
			BufferedWriter out = new BufferedWriter(new
FileWriter(
					"/mnt/sdcard/media/books2.txt", true));
			String link =
"http://www.cs.ucy.ac.cy/thesis/smartscan/connection/testingAddBook.ph
p";

			URL url = new URL(

	"https://ajax.googleapis.com/ajax/services/search/books?v=1.0&q=
ISBN"
						+ isbn
						+
"&key=ABQIAAAA94MJuF2gFa9KqeGTuI0EjBSkkyQUZK2-
HCU2v8Q1ngVNh8D3JBThgk7ik35T7gM0BpiNLjAu-mKyIg&userip=192.168.0.1");
			URLConnection connection = url.openConnection();
			connection
					.addRequestProperty("Referer",

	"https://ajax.googleapis.com/ajax/services/search/booksByIsbn");

			String line;
			// String isbn1;
			StringBuilder builder = new StringBuilder();
			BufferedReader reader = new BufferedReader(new
InputStreamReader(
					connection.getInputStream()));
			while ((line = reader.readLine()) != null) {
				builder.append(line);
			}
			// Creates a new InputStream object of the entity.
			try {
				JSONObject json = new
JSONObject(builder.toString());
				JSONObject res =
json.getJSONObject("responseData");
				JSONArray results =
res.getJSONArray("results");

				JSONObject fields = results.getJSONObject(0);

				out.append("book ISBN:" + isbn + "\n");
				out.append("title:" +
fields.getString("title") + "\n");
				out.append("authors:" +
fields.getString("authors") + "\n");
				out.append("published Year:"
						+
fields.getString("publishedYear") + "\n");
				out.append("pageCount:" +
fields.getString("pageCount") + "\n");

				String str=fields.getString("title");
				String newTitle=str.replace("'", "''");
```



```
				String
str1=URLDecoder.decode(fields.getString("tbUrl"));
				String
newUrl=str1.replace("/googlebooks/images/no_cover_thumb.gif",
"../smartscan/googlebooks/images/no_cover_thumb.gif");


				ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
				nameValuePairs.add(new
BasicNameValuePair("username",
						Connection.username));
				nameValuePairs.add(new
BasicNameValuePair("isbn", isbn + ""));
				nameValuePairs.add(new
BasicNameValuePair("title",/* fields.getString("title")*/newTitle));
				//nameValuePairs.add(new
BasicNameValuePair("title",fields.getString("title")));
				nameValuePairs.add(new
BasicNameValuePair("authors", fields
						.getString("authors")));
				nameValuePairs.add(new
BasicNameValuePair("publishedYear",

	fields.getString("publishedYear")));
				nameValuePairs.add(new
BasicNameValuePair("pageCount", fields
						.getString("pageCount") + ""));
				nameValuePairs.add(new
BasicNameValuePair("rate", "0.0" + ""));
				nameValuePairs.add(new
BasicNameValuePair("imageUrl",newUrl));

if (fields.getString("title") != null) {
	Toast.makeText(getApplicationContext(), "Book Found!",
	Toast.LENGTH_SHORT).show();
						}

				else
					Toast.makeText(getApplicationContext(),
							"Unable To Find Book!",
Toast.LENGTH_SHORT).show();

				String result = getResults(link,
nameValuePairs);

				try {

					JSONObject json_data = new
JSONObject(result);

					if (json_data.getString("isbn") != null)
{
						//
Toast.makeText(getApplicationContext(),
						// "Successful Insertion!",
Toast.LENGTH_SHORT).show();
					}

				} catch (JSONException e) {
```

```
                        System.out.println("ErrorG " +
e.getMessage());

                  }

            } catch (JSONException e1) {
                  // Toast.makeText(getApplicationContext(),
                  // "Error!\nYou may have already inserted the
same book!",
                  // Toast.LENGTH_SHORT).show();
            }

            out.close();
      } catch (IOException e) {
      }

/*****************************************************************/

}

public String getResults(String link, ArrayList<NameValuePair>
values) {

      String result = null;
      InputStream is = null;

      // http post
      try {
            HttpClient httpclient = new DefaultHttpClient();
            HttpPost httppost = new HttpPost(link);
            httppost.setEntity(new
UrlEncodedFormEntity(values));
            HttpResponse response =
httpclient.execute(httppost);
            HttpEntity entity = response.getEntity();
            is = entity.getContent();
      } catch (Exception e1) {
            Log.e("log_tag", "Error in http connection " +
e1.toString());
      }

      // convert response to string
      try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(
                        is, "iso-8859-1"), 8);
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                  sb.append(line + "\n");
            }
            is.close();

            result = sb.toString();
      } catch (Exception e) {
            Log.e("log_tag", "Error converting result " +
e.toString());
      }
      return result;
}
```



```
}
```

loginUser.java

```
package com.google.zxing.client.android.smartscan;

import com.google.zxing.client.android.R;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class loginUser extends Activity {
    /** Called when the activity is first created. */
        @Override
         public void onCreate(Bundle savedInstanceState) {
             super.onCreate(savedInstanceState);
             setContentView(R.layout.main);
           connectButtons();
        }

           void connectButtons() {
                 Button
scanButton=(Button)findViewById(R.id.btn_login);

                 scanButton.setOnClickListener(new
Button.OnClickListener() {

                       @Override
                       public void onClick(View v) {

                             TextView uname =
(TextView)findViewById(R.id.et_uname);
                             TextView pass =
(TextView)findViewById(R.id.et_pass);

                             String uname1, pass1;

                             uname1 =
(String)uname.getText().toString();
                             pass1 =
(String)pass.getText().toString();

                       }

                 });
         }

}
```

mainClass.java

```
package com.google.zxing.client.android.smartscan;

import com.google.zxing.client.android.CaptureActivity;
```

```
import com.google.zxing.client.android.R;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

/**
 * @author Gavriela
 *
 */
public class mainClass extends Activity {

      /** Called when the activity is first created. */
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            connectButtons();
      }

      void connectButtons() {
            Button scanButton = (Button) findViewById(R.id.btnScan);
            Button insertButton = (Button)
findViewById(R.id.btnInsertBook);
            Button logoutButton = (Button)
findViewById(R.id.btn_logout);

            scanButton.setOnClickListener(new Button.OnClickListener()
{

                  @Override
                  public void onClick(View v) {
                        Intent myIntent = new
Intent(mainClass.this,CaptureActivity.class);
                        startActivity(myIntent);
                  }
            });

            insertButton.setOnClickListener(new
Button.OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        Intent myIntent = new
Intent(mainClass.this,insertBookByISBN.class);
                        startActivity(myIntent);
                  }
            });

            logoutButton.setOnClickListener(new
Button.OnClickListener() {

                  @Override
                  public void onClick(View v) {
                         finish();
                     System.exit(0);
                  }
            });
```



```
      }

      @Override
      public void onBackPressed() {
            // TODO Auto-generated method stub
            super.onBackPressed();
            finish();
      }
}
```

captureActivity.java

```
package com.google.zxing.client.android;

/*********************************************************************
****/
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.text.DateFormat;
import java.util.Collection;
import java.util.Date;
import java.util.EnumSet;
import java.util.Map;
import java.util.Set;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Rect;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.preference.PreferenceManager;
import android.text.ClipboardManager;
import android.util.Log;
import android.util.TypedValue;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.zxing.BarcodeFormat;
```

```
import com.google.zxing.Result;
import com.google.zxing.ResultMetadataType;
import com.google.zxing.ResultPoint;
import com.google.zxing.client.android.camera.CameraManager;
import com.google.zxing.client.android.history.HistoryActivity;
import com.google.zxing.client.android.history.HistoryItem;
import com.google.zxing.client.android.history.HistoryManager;
import com.google.zxing.client.android.result.ResultButtonListener;
import com.google.zxing.client.android.result.ResultHandler;
import com.google.zxing.client.android.result.ResultHandlerFactory;
import
com.google.zxing.client.android.result.supplement.SupplementalInfoRetr
iever;

/**
 * This activity opens the camera and does the actual scanning on a
background
 * thread. It draws a viewfinder to help the user place the barcode
correctly,
 * shows feedback as the image processing is happening, and then
overlays the
 * results when a scan is successful.
 *
 * @author dswitkin@google.com (Daniel Switkin)
 * @author Sean Owen
 */
public class CaptureActivity extends Activity implements
            SurfaceHolder.Callback {

      private static final String TAG =
CaptureActivity.class.getSimpleName();

      /**********************************************************
***************/
        private static final int HISTORY_ID = Menu.FIRST;
        private static final int SETTINGS_ID = Menu.FIRST+1;
        private static final int EXIT_ID = Menu.FIRST+2;
      /**********************************************************
***************/

      private static final long DEFAULT_INTENT_RESULT_DURATION_MS =
1500L;
      /**********************************************************
***********/
      private static final long BULK_MODE_SCAN_DELAY_MS = 3000L;//
wait for 3sec
      /**********************************************************
***********/
      private static final String PACKAGE_NAME =
"com.google.zxing.client.android";
      private static final String PRODUCT_SEARCH_URL_PREFIX =
"http://www.google";
      private static final String PRODUCT_SEARCH_URL_SUFFIX =
"/m/products/scan";
      private static final String[] ZXING_URLS = {
                  "http://zxing.appspot.com/scan", "zxing://scan/" };
      private static final String RETURN_CODE_PLACEHOLDER = "{CODE}";
      private static final String RETURN_URL_PARAM = "ret";

      public static final int HISTORY_REQUEST_CODE = 0x0000bacc;
```



```
      private static final Set<ResultMetadataType>
DISPLAYABLE_METADATA_TYPES = EnumSet
                  .of(ResultMetadataType.ISSUE_NUMBER,
                              ResultMetadataType.SUGGESTED_PRICE,

      ResultMetadataType.ERROR_CORRECTION_LEVEL,
                              ResultMetadataType.POSSIBLE_COUNTRY);

      private CameraManager cameraManager;
      private CaptureActivityHandler handler;
      private Result savedResultToShow;
      private ViewfinderView viewfinderView;
      private TextView statusView;
      private View resultView;
      private Result lastResult;
      private boolean hasSurface;
      private boolean copyToClipboard;
      private IntentSource source;
      private String sourceUrl;
      private String returnUrlTemplate;
      private Collection<BarcodeFormat> decodeFormats;
      private String characterSet;
      private String versionName;
      private HistoryManager historyManager;
      private InactivityTimer inactivityTimer;
      private BeepManager beepManager;

      private final DialogInterface.OnClickListener aboutListener =
new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int
i) {
                  Intent intent = new Intent(Intent.ACTION_VIEW,

      Uri.parse(getString(R.string.zxing_url)));

      intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
                  startActivity(intent);
            }
      };

      ViewfinderView getViewfinderView() {
            return viewfinderView;
      }

      public Handler getHandler() {
            return handler;
      }

      CameraManager getCameraManager() {
            return cameraManager;
      }

      @Override
      public void onCreate(Bundle icicle) {
            super.onCreate(icicle);
            Window window = getWindow();

      window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
```

```
            setContentView(R.layout.capture);

            hasSurface = false;
            historyManager = new HistoryManager(this);
            historyManager.trimHistory();
            inactivityTimer = new InactivityTimer(this);
            beepManager = new BeepManager(this);

            PreferenceManager.setDefaultValues(this,
R.xml.preferences, false);

            //showHelpOnFirstLaunch();
      }

      @Override
      protected void onResume() {
            super.onResume();

            // CameraManager must be initialized here, not in
onCreate(). This is
            // necessary because we don't
            // want to open the camera driver and measure the screen
size if we're
            // going to show the help on
            // first launch. That led to bugs where the scanning
rectangle was the
            // wrong size and partially
            // off screen.
            cameraManager = new CameraManager(getApplication());

            viewfinderView = (ViewfinderView)
findViewById(R.id.viewfinder_view);
            viewfinderView.setCameraManager(cameraManager);

            resultView = findViewById(R.id.result_view);
            statusView = (TextView) findViewById(R.id.status_view);

            handler = null;
            lastResult = null;

            resetStatusView();

            SurfaceView surfaceView = (SurfaceView)
findViewById(R.id.preview_view);
            SurfaceHolder surfaceHolder = surfaceView.getHolder();
            if (hasSurface) {
                  // The activity was paused but not stopped, so the
surface still
                  // exists. Therefore
                  // surfaceCreated() won't be called, so init the
camera here.
                  initCamera(surfaceHolder);
            } else {
                  // Install the callback and wait for
surfaceCreated() to init the
                  // camera.
                  surfaceHolder.addCallback(this);

      surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
            }
```



```
            beepManager.updatePrefs();

            inactivityTimer.onResume();

            Intent intent = getIntent();

            SharedPreferences prefs = PreferenceManager
                        .getDefaultSharedPreferences(this);
            copyToClipboard = prefs.getBoolean(
                        PreferencesActivity.KEY_COPY_TO_CLIPBOARD,
true)
                        && (intent == null || intent.getBooleanExtra(
                                    Intents.Scan.SAVE_HISTORY, true));

            source = IntentSource.NONE;
            decodeFormats = null;
            characterSet = null;

            if (intent != null) {

                  String action = intent.getAction();
                  String dataString = intent.getDataString();

                  if (Intents.Scan.ACTION.equals(action)) {

                        // Scan the formats the intent requested, and
return the result
                        // to the calling activity.
                        source = IntentSource.NATIVE_APP_INTENT;
                        decodeFormats =
DecodeFormatManager.parseDecodeFormats(intent);

                        if (intent.hasExtra(Intents.Scan.WIDTH)
                                    &&
intent.hasExtra(Intents.Scan.HEIGHT)) {
                              int width =
intent.getIntExtra(Intents.Scan.WIDTH, 0);
                              int height =
intent.getIntExtra(Intents.Scan.HEIGHT, 0);
                              if (width > 0 && height > 0) {

      cameraManager.setManualFramingRect(width, height);
                              }
                        }

                        String customPromptMessage = intent

      .getStringExtra(Intents.Scan.PROMPT_MESSAGE);
                        if (customPromptMessage != null) {
                              statusView.setText(customPromptMessage);
                        }

                  } else if (dataString != null
                              &&
dataString.contains(PRODUCT_SEARCH_URL_PREFIX)
                              &&
dataString.contains(PRODUCT_SEARCH_URL_SUFFIX)) {
```

```
                        // Scan only products and send the result to
mobile Product
                        // Search.
                        source = IntentSource.PRODUCT_SEARCH_LINK;
                        sourceUrl = dataString;
                        decodeFormats =
DecodeFormatManager.PRODUCT_FORMATS;

                  } else if (isZXingURL(dataString)) {

                        // Scan formats requested in query string (all
formats if none
                        // specified).
                        // If a return URL is specified, send the
results there.
                        // Otherwise, handle it ourselves.
                        source = IntentSource.ZXING_LINK;
                        sourceUrl = dataString;
                        Uri inputUri = Uri.parse(sourceUrl);
                        returnUrlTemplate = inputUri

      .getQueryParameter(RETURN_URL_PARAM);
                        decodeFormats = DecodeFormatManager
                                    .parseDecodeFormats(inputUri);

                  }

                  characterSet =
intent.getStringExtra(Intents.Scan.CHARACTER_SET);

            }
      }

      private static boolean isZXingURL(String dataString) {
            if (dataString == null) {
                  return false;
            }
            for (String url : ZXING_URLS) {
                  if (dataString.startsWith(url)) {
                        return true;
                  }
            }
            return false;
      }

      @Override
      protected void onPause() {
            if (handler != null) {
                  handler.quitSynchronously();
                  handler = null;
            }
            inactivityTimer.onPause();
            cameraManager.closeDriver();
            if (!hasSurface) {
                  SurfaceView surfaceView = (SurfaceView)
findViewById(R.id.preview_view);
                  SurfaceHolder surfaceHolder =
surfaceView.getHolder();
                  surfaceHolder.removeCallback(this);
            }
```



```
            super.onPause();
      }

      @Override
      protected void onDestroy() {
            inactivityTimer.shutdown();
            super.onDestroy();
      }

      @Override
      public boolean onKeyDown(int keyCode, KeyEvent event) {
            if (keyCode == KeyEvent.KEYCODE_BACK) {
                  if (source == IntentSource.NATIVE_APP_INTENT) {
                        setResult(RESULT_CANCELED);
                        finish();
                        return true;
                  } else if ((source == IntentSource.NONE || source ==
IntentSource.ZXING_LINK)
                              && lastResult != null) {
                        restartPreviewAfterDelay(0L);
                        return true;
                  }
            } else if (keyCode == KeyEvent.KEYCODE_FOCUS
                        || keyCode == KeyEvent.KEYCODE_CAMERA) {
                  // Handle these events so they don't launch the
Camera app
                  return true;
            }
            return super.onKeyDown(keyCode, event);
      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
            super.onCreateOptionsMenu(menu);

      /*********************************************************************
**************/
            menu.add(Menu.NONE, HISTORY_ID, Menu.NONE,
R.string.menu_history)
            .setIcon(android.R.drawable.ic_menu_recent_history);

            menu.add(Menu.NONE, SETTINGS_ID, Menu.NONE,
R.string.menu_settings)
            .setIcon(android.R.drawable.ic_menu_preferences);

            menu.add(Menu.NONE, EXIT_ID, Menu.NONE,
R.string.menu_exit).setIcon(
                        android.R.drawable.ic_menu_info_details);

      /*********************************************************************
**************/
            return true;
      }

      /*********************************************************************
**************/
      // Don't display the share menu item if the result overlay is
showing.
/*    @Override
      public boolean onPrepareOptionsMenu(Menu menu) {
```

```
            super.onPrepareOptionsMenu(menu);
            menu.findItem(SHARE_ID).setVisible(lastResult == null);
            return true;
      }*/
      /*********************************************************************
**************/

      @Override
      public boolean onOptionsItemSelected(MenuItem item) {
            Intent intent = new Intent(Intent.ACTION_VIEW);

      intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
            switch (item.getItemId()) {

            case HISTORY_ID:
                  intent = new Intent(Intent.ACTION_VIEW);

      intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
                  intent.setClassName(this,
HistoryActivity.class.getName());
                  startActivityForResult(intent,
HISTORY_REQUEST_CODE);
                  break;
            case SETTINGS_ID:
                  intent.setClassName(this,
PreferencesActivity.class.getName());
                  startActivity(intent);
                  break;
      /*********************************************************************
***************/
                  case EXIT_ID:
                  this.finish();
              break;

/*********************************************************************
*********/
            default:
                  return super.onOptionsItemSelected(item);
            }
            return true;
      }

      @Override
      public void onActivityResult(int requestCode, int resultCode,
Intent intent) {
            if (resultCode == RESULT_OK) {
                  if (requestCode == HISTORY_REQUEST_CODE) {
                        int itemNumber = intent.getIntExtra(
                                    Intents.History.ITEM_NUMBER, -1);
                        if (itemNumber >= 0) {
                              HistoryItem historyItem = historyManager

      .buildHistoryItem(itemNumber);
                              decodeOrStoreSavedBitmap(null,
historyItem.getResult());
                        }
                  }
            }
      }
```



```
      private void decodeOrStoreSavedBitmap(Bitmap bitmap, Result
result) {
            // Bitmap isn't used yet -- will be used soon
            if (handler == null) {
                  savedResultToShow = result;
            } else {
                  if (result != null) {
                        savedResultToShow = result;
                  }
                  if (savedResultToShow != null) {
                        Message message = Message.obtain(handler,
                                    R.id.decode_succeeded,
savedResultToShow);
                        handler.sendMessage(message);
                  }
                  savedResultToShow = null;
            }
      }

      @Override
      public void surfaceCreated(SurfaceHolder holder) {
            if (holder == null) {
                  Log.e(TAG,
                              "*** WARNING *** surfaceCreated() gave
us a null surface!");
            }
            if (!hasSurface) {
                  hasSurface = true;
                  initCamera(holder);
            }
      }

      @Override
      public void surfaceDestroyed(SurfaceHolder holder) {
            hasSurface = false;
      }

      @Override
      public void surfaceChanged(SurfaceHolder holder, int format, int
width,
                  int height) {

      }

      /**
       * A valid barcode has been found, so give an indication of
success and show
       * the results.
       *
       * @param rawResult
       *            The contents of the barcode.
       * @param barcode
       *            A greyscale bitmap of the camera data which was
decoded.
       */
      public void handleDecode(Result rawResult, Bitmap barcode) {

            inactivityTimer.onActivity();
            lastResult = rawResult;
```

```
            ResultHandler resultHandler =
ResultHandlerFactory.makeResultHandler(
                        this, rawResult);
            historyManager.addHistoryItem(rawResult, resultHandler);

            if (barcode == null) {
                  // This is from history -- no saved barcode
                  handleDecodeInternally(rawResult, resultHandler,
null);
            } else {
                  beepManager.playBeepSoundAndVibrate();
                  drawResultPoints(barcode, rawResult);
                  switch (source) {
                  case NATIVE_APP_INTENT:
                  case PRODUCT_SEARCH_LINK:
                        handleDecodeExternally(rawResult,
resultHandler, barcode);
                        break;
                  case ZXING_LINK:
                        if (returnUrlTemplate == null) {
                              handleDecodeInternally(rawResult,
resultHandler, barcode);
                        } else {
                              handleDecodeExternally(rawResult,
resultHandler, barcode);
                        }
                        break;
                  case NONE:
                        SharedPreferences prefs = PreferenceManager

      .getDefaultSharedPreferences(this);
                        if
(prefs.getBoolean(PreferencesActivity.KEY_BULK_MODE, false)) {

      /*************************************************************
*******/
                              handleDecodeInternally(rawResult,
resultHandler, barcode);

                              //decide if its a book or not
                              String searchMe =
resultHandler.toString();
                              String substring = "ISBN";

                     if(searchMe.contains(substring))
                        {
                        Toast.makeText(getApplicationContext(),"Book
ISBN:" +rawResult.getText(), Toast.LENGTH_LONG).show();
                        }
                        else
                        {
                        Toast.makeText(getApplicationContext(),"This
is not a book!", Toast.LENGTH_LONG).show();
                        }
                              //
/*************************************************************
*/
                              // // Wait a moment or else it will scan
the same barcode
                              // // continuously about 3 times
```



```
                              resultView.setVisibility(View.VISIBLE);

      restartPreviewAfterDelay(BULK_MODE_SCAN_DELAY_MS);

                              // Toast
                              //
toast=Toast.makeText(getApplicationContext(),resultHandler.get
                              // , Toast.LENGTH_SHORT);
                              // toast.setView();
                              // toast.show();
                              //
resultView.setVisibility(View.VISIBLE);

                        } else {
                              handleDecodeInternally(rawResult,
resultHandler, barcode);
                        }
                        break;
                  }
            }
      }

      /**
       * Superimpose a line for 1D or dots for 2D to highlight the key
features of
       * the barcode.
       *
       * @param barcode
       *            A bitmap of the captured image.
       * @param rawResult
       *            The decoded results which contains the points to
draw.
       */
      private void drawResultPoints(Bitmap barcode, Result rawResult)
{
            ResultPoint[] points = rawResult.getResultPoints();
            if (points != null && points.length > 0) {
                  Canvas canvas = new Canvas(barcode);
                  Paint paint = new Paint();

      paint.setColor(getResources().getColor(R.color.result_image_bord
er));
                  paint.setStrokeWidth(3.0f);
                  paint.setStyle(Paint.Style.STROKE);
                  Rect border = new Rect(2, 2, barcode.getWidth() - 2,
                              barcode.getHeight() - 2);
                  canvas.drawRect(border, paint);


      paint.setColor(getResources().getColor(R.color.result_points));
                  if (points.length == 2) {
                        paint.setStrokeWidth(4.0f);
                        drawLine(canvas, paint, points[0], points[1]);
                  } else if (points.length == 4
                              && (rawResult.getBarcodeFormat() ==
BarcodeFormat.UPC_A || rawResult
                                          .getBarcodeFormat() ==
BarcodeFormat.EAN_13)) {
                        // Hacky special case -- draw two lines, for
the barcode and
```

```
                        // metadata
                        drawLine(canvas, paint, points[0], points[1]);
                        drawLine(canvas, paint, points[2], points[3]);
                  } else {
                        paint.setStrokeWidth(10.0f);
                        for (ResultPoint point : points) {
                              canvas.drawPoint(point.getX(),
point.getY(), paint);
                        }
                  }
            }
      }

      private static void drawLine(Canvas canvas, Paint paint,
ResultPoint a,
                  ResultPoint b) {
            canvas.drawLine(a.getX(), a.getY(), b.getX(), b.getY(),
paint);
      }

      // Put up our own UI for how to handle the decoded contents.
      private void handleDecodeInternally(Result rawResult,
                  ResultHandler resultHandler, Bitmap barcode) {
            statusView.setVisibility(View.GONE);
            viewfinderView.setVisibility(View.GONE);
            resultView.setVisibility(View.VISIBLE);

            ImageView barcodeImageView = (ImageView)
findViewById(R.id.barcode_image_view);
            if (barcode == null) {

      barcodeImageView.setImageBitmap(BitmapFactory.decodeResource(
                        getResources(),
R.drawable.launcher_icon));
            } else {
                  barcodeImageView.setImageBitmap(barcode);
            }

            TextView formatTextView = (TextView)
findViewById(R.id.format_text_view);

      formatTextView.setText(rawResult.getBarcodeFormat().toString());

            TextView typeTextView = (TextView)
findViewById(R.id.type_text_view);
            typeTextView.setText(resultHandler.getType().toString());

            DateFormat formatter =
DateFormat.getDateTimeInstance(DateFormat.SHORT,
                        DateFormat.SHORT);
            String formattedTime = formatter.format(new Date(rawResult
                        .getTimestamp()));
            TextView timeTextView = (TextView)
findViewById(R.id.time_text_view);
            timeTextView.setText(formattedTime);

            TextView metaTextView = (TextView)
findViewById(R.id.meta_text_view);
            View metaTextViewLabel =
findViewById(R.id.meta_text_view_label);
```



```
            metaTextView.setVisibility(View.GONE);
            metaTextViewLabel.setVisibility(View.GONE);
            Map<ResultMetadataType, Object> metadata = rawResult
                        .getResultMetadata();
            if (metadata != null) {
                  StringBuilder metadataText = new StringBuilder(20);
                  for (Map.Entry<ResultMetadataType, Object> entry :
metadata
                              .entrySet()) {
                        if
(DISPLAYABLE_METADATA_TYPES.contains(entry.getKey())) {

      metadataText.append(entry.getValue()).append('\n');
                        }
                  }
                  if (metadataText.length() > 0) {
                        metadataText.setLength(metadataText.length() -
1);
                        metaTextView.setText(metadataText);
                        metaTextView.setVisibility(View.VISIBLE);
                        metaTextViewLabel.setVisibility(View.VISIBLE);
                  }
            }

            TextView contentsTextView = (TextView)
findViewById(R.id.contents_text_view);
            CharSequence displayContents =
resultHandler.getDisplayContents();
            contentsTextView.setText(displayContents);
            // Crudely scale betweeen 22 and 32 -- bigger font for
shorter text
            int scaledSize = Math.max(22, 32 -
displayContents.length() / 4);
            contentsTextView.setTextSize(TypedValue.COMPLEX_UNIT_SP,
scaledSize);

            TextView supplementTextView = (TextView)
findViewById(R.id.contents_supplement_text_view);
            supplementTextView.setText("");
            supplementTextView.setOnClickListener(null);
            if
(PreferenceManager.getDefaultSharedPreferences(this).getBoolean(
                        PreferencesActivity.KEY_SUPPLEMENTAL, true)) {

      SupplementalInfoRetriever.maybeInvokeRetrieval(supplementTextVie
w,
                              resultHandler.getResult(), handler,
historyManager, this);
            }

            int buttonCount = resultHandler.getButtonCount();
            ViewGroup buttonView = (ViewGroup)
findViewById(R.id.result_button_view);
            buttonView.requestFocus();
            for (int x = 0; x < ResultHandler.MAX_BUTTON_COUNT; x++) {
                  TextView button = (TextView)
buttonView.getChildAt(x);
                  if (x < buttonCount) {
                        button.setVisibility(View.VISIBLE);
```

```
      button.setText(resultHandler.getButtonText(x));
                        button.setOnClickListener(new
ResultButtonListener(
                                    resultHandler, x));
                  } else {
                        button.setVisibility(View.GONE);
                  }
            }

            if (copyToClipboard && !resultHandler.areContentsSecure())
{
                  ClipboardManager clipboard = (ClipboardManager)
getSystemService(CLIPBOARD_SERVICE);
                  clipboard.setText(displayContents);
            }
      }

      // Briefly show the contents of the barcode, then handle the
result outside
      // Barcode Scanner.
      private void handleDecodeExternally(Result rawResult,
                  ResultHandler resultHandler, Bitmap barcode) {
            viewfinderView.drawResultBitmap(barcode);

            // Since this message will only be shown for a second,
just tell the
            // user what kind of
            // barcode was found (e.g. contact info) rather than the
full contents,
            // which they won't
            // have time to read.

      statusView.setText(getString(resultHandler.getDisplayTitle()));

            if (copyToClipboard && !resultHandler.areContentsSecure())
{
                  ClipboardManager clipboard = (ClipboardManager)
getSystemService(CLIPBOARD_SERVICE);

      clipboard.setText(resultHandler.getDisplayContents());
            }

            if (source == IntentSource.NATIVE_APP_INTENT) {

                  // Hand back whatever action they requested - this
can be changed to
                  // Intents.Scan.ACTION when
                  // the deprecated intent is retired.
                  Intent intent = new Intent(getIntent().getAction());

      intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
                  intent.putExtra(Intents.Scan.RESULT,
rawResult.toString());
                  intent.putExtra(Intents.Scan.RESULT_FORMAT,
rawResult
                              .getBarcodeFormat().toString());
                  byte[] rawBytes = rawResult.getRawBytes();
                  if (rawBytes != null && rawBytes.length > 0) {
```



```
                        intent.putExtra(Intents.Scan.RESULT_BYTES,
rawBytes);
                  }
                  Map<ResultMetadataType, ?> metadata =
rawResult.getResultMetadata();
                  if (metadata != null) {
                        Integer orientation = (Integer) metadata

      .get(ResultMetadataType.ORIENTATION);
                        if (orientation != null) {

      intent.putExtra(Intents.Scan.RESULT_ORIENTATION,
                                          orientation.intValue());
                        }
                        String ecLevel = (String) metadata

      .get(ResultMetadataType.ERROR_CORRECTION_LEVEL);
                        if (ecLevel != null) {

      intent.putExtra(Intents.Scan.RESULT_ERROR_CORRECTION_LEVEL,
                                          ecLevel);
                        }
                        Iterable<byte[]> byteSegments =
(Iterable<byte[]>) metadata

      .get(ResultMetadataType.BYTE_SEGMENTS);
                        if (byteSegments != null) {
                              int i = 0;
                              for (byte[] byteSegment : byteSegments)
{
                                    intent.putExtra(

      Intents.Scan.RESULT_BYTE_SEGMENTS_PREFIX + i,
                                                byteSegment);
                                    i++;
                              }
                        }
                  }
                  sendReplyMessage(R.id.return_scan_result, intent);

            } else if (source == IntentSource.PRODUCT_SEARCH_LINK) {

                  // Reformulate the URL which triggered us into a
query, so that the
                  // request goes to the same
                  // TLD as the scan URL.
                  int end = sourceUrl.lastIndexOf("/scan");
                  String replyURL = sourceUrl.substring(0, end) +
"?q="
                              + resultHandler.getDisplayContents() +
"&source=zxing";
                  sendReplyMessage(R.id.launch_product_query,
replyURL);

            } else if (source == IntentSource.ZXING_LINK) {

                  // Replace each occurrence of
RETURN_CODE_PLACEHOLDER in the
                  // returnUrlTemplate
```

```
                // with the scanned code. This allows both queries
and REST-style
                // URLs to work.
                if (returnUrlTemplate != null) {
                        String codeReplacement =
String.valueOf(resultHandler
                                        .getDisplayContents());
                        try {
                                codeReplacement =
URLEncoder.encode(codeReplacement,
                                                "UTF-8");
                        } catch (UnsupportedEncodingException e) {
                                // can't happen; UTF-8 is always
supported. Continue, I
                                // guess, without encoding
                        }
                        String replyURL = returnUrlTemplate.replace(
                                        RETURN_CODE_PLACEHOLDER,
codeReplacement);
                        sendReplyMessage(R.id.launch_product_query,
replyURL);
                }

        }
    }

    private void sendReplyMessage(int id, Object arg) {
        Message message = Message.obtain(handler, id, arg);
        long resultDurationMS = getIntent().getLongExtra(
                        Intents.Scan.RESULT_DISPLAY_DURATION_MS,
                        DEFAULT_INTENT_RESULT_DURATION_MS);
        if (resultDurationMS > 0L) {
            handler.sendMessageDelayed(message,
resultDurationMS);
        } else {
            handler.sendMessage(message);
        }
    }

    /**
     * We want the help screen to be shown automatically the first
time a new
     * version of the app is run. The easiest way to do this is to
check
     * android:versionCode from the manifest, and compare it to a
value stored
     * as a preference.
     */
    private boolean showHelpOnFirstLaunch() {
        try {
            PackageInfo info =
getPackageManager().getPackageInfo(PACKAGE_NAME,
                                0);
            int currentVersion = info.versionCode;
            // Since we're paying to talk to the PackageManager
anyway, it makes
            // sense to cache the app
            // version name here for display in the about box
later.
            this.versionName = info.versionName;
```



```
            SharedPreferences prefs = PreferenceManager
                            .getDefaultSharedPreferences(this);
            int lastVersion = prefs.getInt(

    PreferencesActivity.KEY_HELP_VERSION_SHOWN, 0);
            if (currentVersion > lastVersion) {
                prefs.edit()

    .putInt(PreferencesActivity.KEY_HELP_VERSION_SHOWN,

    currentVersion).commit();
                Intent intent = new Intent(this,
HelpActivity.class);

    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
                // Show the default page on a clean install,
and the what's new
                // page on an upgrade.
                String page = lastVersion == 0 ?
HelpActivity.DEFAULT_PAGE
                                : HelpActivity.WHATS_NEW_PAGE;

    intent.putExtra(HelpActivity.REQUESTED_PAGE_KEY, page);
                startActivity(intent);
                return true;
            }
        } catch (PackageManager.NameNotFoundException e) {
            Log.w(TAG, e);
        }
        return false;
    }

    private void initCamera(SurfaceHolder surfaceHolder) {
        try {
            cameraManager.openDriver(surfaceHolder);
            // Creating the handler starts the preview, which
can also throw a
            // RuntimeException.
            if (handler == null) {
                handler = new CaptureActivityHandler(this,
decodeFormats,
                                characterSet, cameraManager);
            }
            decodeOrStoreSavedBitmap(null, null);
        } catch (IOException ioe) {
            Log.w(TAG, ioe);
            displayFrameworkBugMessageAndExit();
        } catch (RuntimeException e) {
            // Barcode Scanner has seen crashes in the wild of
this variety:
            // java.?lang.?RuntimeException: Fail to connect to
camera service
            Log.w(TAG, "Unexpected error initializing camera",
e);
            displayFrameworkBugMessageAndExit();
        }
    }

    private void displayFrameworkBugMessageAndExit() {
```

```
            AlertDialog.Builder builder = new
AlertDialog.Builder(this);
            builder.setTitle(getString(R.string.app_name));

      builder.setMessage(getString(R.string.msg_camera_framework_bug))
;
            builder.setPositiveButton(R.string.button_ok, new
FinishListener(this));
            builder.setOnCancelListener(new FinishListener(this));
            builder.show();
      }

      public void restartPreviewAfterDelay(long delayMS) {
            if (handler != null) {

      handler.sendEmptyMessageDelayed(R.id.restart_preview, delayMS);

            }
            resetStatusView();
      }

      private void resetStatusView() {
            // resultView.setVisibility(View.GONE);
            statusView.setText(R.string.msg_default_status);
            statusView.setVisibility(View.VISIBLE);
            viewfinderView.setVisibility(View.VISIBLE);
            lastResult = null;
      }

      public void drawViewfinder() {
            viewfinderView.drawViewfinder();
      }

      /***************************************************/
      public View getResultView() {
            // TODO Auto-generated method stub
            return resultView;
      }
      /***************************************************/
}
```

captureActivityHandler.java

```
package com.google.zxing.client.android;

import com.google.zxing.BarcodeFormat;
import com.google.zxing.Result;
import com.google.zxing.client.android.camera.CameraManager;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.View;

import java.util.Collection;
```



```
/**
 * This class handles all the messaging which comprises the state
machine for capture.
 *
 * @author dswitkin@google.com (Daniel Switkin)
 */
public final class CaptureActivityHandler extends Handler {

  private static final String TAG =
CaptureActivityHandler.class.getSimpleName();

  private final CaptureActivity activity;
  private final DecodeThread decodeThread;
  private State state;
  private final CameraManager cameraManager;

  private enum State {
    PREVIEW,
    SUCCESS,
    DONE
  }

  CaptureActivityHandler(CaptureActivity activity,
                         Collection<BarcodeFormat> decodeFormats,
                         String characterSet,
                         CameraManager cameraManager) {
    this.activity = activity;
    decodeThread = new DecodeThread(activity, decodeFormats,
characterSet,
        new
ViewfinderResultPointCallback(activity.getViewfinderView()));
    decodeThread.start();
    state = State.SUCCESS;

    // Start ourselves capturing previews and decoding.
    this.cameraManager = cameraManager;
    cameraManager.startPreview();
    restartPreviewAndDecode();
  }

  @Override
  public void handleMessage(Message message) {
    switch (message.what) {
      case R.id.auto_focus:
        //Log.d(TAG, "Got auto-focus message");
        // When one auto focus pass finishes, start another. This is
the closest thing to
        // continuous AF. It does seem to hunt a bit, but I'm not sure
what else to do.
        if (state == State.PREVIEW) {
          cameraManager.requestAutoFocus(this, R.id.auto_focus);
        }
        break;
      case R.id.restart_preview:
        Log.d(TAG, "Got restart preview message");
        restartPreviewAndDecode();
        break;
      case R.id.decode_succeeded:
        Log.d(TAG, "Got decode succeeded message");
        state = State.SUCCESS;
```

```
        Bundle bundle = message.getData();
        Bitmap barcode = bundle == null ? null :
            (Bitmap)
bundle.getParcelable(DecodeThread.BARCODE_BITMAP);
        activity.handleDecode((Result) message.obj, barcode);
        break;
      case R.id.decode_failed:

        // We're decoding as fast as possible, so when one decode
fails, start another.
        state = State.PREVIEW;
        cameraManager.requestPreviewFrame(decodeThread.getHandler(),
R.id.decode);
        break;
      case R.id.return_scan_result:
        Log.d(TAG, "Got return scan result message");
        activity.setResult(Activity.RESULT_OK, (Intent) message.obj);
        activity.finish();
        break;
      case R.id.launch_product_query:
        Log.d(TAG, "Got product query message");
        String url = (String) message.obj;
        Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse(url));
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
        activity.startActivity(intent);
        break;
    }
  }

  public void quitSynchronously() {
    state = State.DONE;
    cameraManager.stopPreview();
    Message quit = Message.obtain(decodeThread.getHandler(),
R.id.quit);
    quit.sendToTarget();
    try {
      // Wait at most half a second; should be enough time, and
onPause() will timeout quickly
      decodeThread.join(500L);
    } catch (InterruptedException e) {
      // continue
    }

    // Be absolutely sure we don't send any queued up messages
    removeMessages(R.id.decode_succeeded);
    removeMessages(R.id.decode_failed);
  }

  private void restartPreviewAndDecode() {
    if (state == State.SUCCESS) {
      state = State.PREVIEW;
      cameraManager.requestPreviewFrame(decodeThread.getHandler(),
R.id.decode);
      cameraManager.requestAutoFocus(this, R.id.auto_focus);
      activity.drawViewfinder();
      /*********************************************/
      activity.getResultView().setVisibility(View.GONE);
      /*********************************************/
    }
```



```
  }

}
```

HistoryActivity.java

```
package com.google.zxing.client.android.history;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ListActivity;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import com.google.zxing.client.android.CaptureActivity;
import com.google.zxing.client.android.Intents;
import com.google.zxing.client.android.R;

import java.util.List;

public final class HistoryActivity extends ListActivity {

/*************************************************************************/
  private static final int SAVE_ID = Menu.FIRST;

/*************************************************************************/
  private static final int CLEAR_ID = Menu.FIRST + 1;

  private HistoryManager historyManager;
  private HistoryItemAdapter adapter;

  @Override
  protected void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    this.historyManager = new HistoryManager(this);
    adapter = new HistoryItemAdapter(this);
    setListAdapter(adapter);
    ListView listview = getListView();
    registerForContextMenu(listview);
  }

  @Override
  protected void onResume() {
    super.onResume();
    List<HistoryItem> items = historyManager.buildHistoryItems();
    adapter.clear();
    for (HistoryItem item : items) {
      adapter.add(item);
    }
    if (adapter.isEmpty()) {
      adapter.add(new HistoryItem(null, null, null));
    }
```

```
  }

  @Override
  protected void onListItemClick(ListView l, View v, int position,
long id) {
    if (adapter.getItem(position).getResult() != null) {
      Intent intent = new Intent(this, CaptureActivity.class);
      intent.putExtra(Intents.History.ITEM_NUMBER, position);
      setResult(Activity.RESULT_OK, intent);
      finish();
    }
  }

  @Override
  public void onCreateContextMenu(ContextMenu menu,
                                  View v,
                                  ContextMenu.ContextMenuInfo
menuInfo) {
    int position = ((AdapterView.AdapterContextMenuInfo)
menuInfo).position;
    menu.add(Menu.NONE, position, position,
R.string.history_clear_one_history_text);
  }

  @Override
  public boolean onContextItemSelected(MenuItem item) {
    int position = item.getItemId();
    historyManager.deleteHistoryItem(position);
    finish();
    return true;
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    if (historyManager.hasHistoryItems()) {

/*********************************************************************/
      menu.add(0, SAVE_ID, 0,
R.string.history_save).setIcon(android.R.drawable.ic_input_add);

/*********************************************************************/
      menu.add(0, CLEAR_ID, 0,
R.string.history_clear_text).setIcon(android.R.drawable.ic_menu_delete
);
      return true;
    }
    return false;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
 /*********************************************************************/
      case SAVE_ID:

/*********************************************************************/
        CharSequence history = historyManager.buildHistory();
        Uri historyFile =
HistoryManager.saveHistory(history.toString());
```



```
        if (historyFile == null) {
          AlertDialog.Builder builder = new AlertDialog.Builder(this);
          builder.setMessage(R.string.msg_unmount_usb);
          builder.setPositiveButton(R.string.button_ok, null);
          builder.show();

/*********************************************************************/
          //ean to historyFile den einai null
        }

/*********************************************************************/
        break;
      case CLEAR_ID:
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(R.string.msg_sure);
        builder.setCancelable(true);
        builder.setPositiveButton(R.string.button_ok, new
DialogInterface.OnClickListener() {
          @Override
          public void onClick(DialogInterface dialog, int i2) {
            historyManager.clearHistory();
            dialog.dismiss();
            finish();
          }
        });
        builder.setNegativeButton(R.string.button_cancel, null);
        builder.show();
        break;
      default:
        return super.onOptionsItemSelected(item);
    }
    return true;
  }

}
```

HistoryManager.java

```
package com.google.zxing.client.android.history;

/*********************************************************************/
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

```
/*****************************************************************/

import com.google.zxing.BarcodeFormat;
import com.google.zxing.Result;
import com.google.zxing.client.android.Intents;
import com.google.zxing.client.android.PreferencesActivity;
import com.google.zxing.client.android.result.ResultHandler;
import com.google.zxing.client.android.smartscan.Connection;

import android.app.Activity;
import android.content.ContentValues;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.net.Uri;
import android.os.Environment;
import android.preference.PreferenceManager;
import android.util.Log;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLDecoder;
import java.nio.charset.Charset;
import java.text.DateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * <p>
 * Manages functionality related to scan history.
 * </p>
 *
 * @author Sean Owen
 */
public  class HistoryManager{

/*****************************************************************/
      File file = new File("/mnt/sdcard/media/books.txt");
/*****************************************************************/

      private static final String TAG =
HistoryManager.class.getSimpleName();

      private static final int MAX_ITEMS = 500;

      private static final String[] COLUMNS = { DBHelper.TEXT_COL,
                  DBHelper.DISPLAY_COL, DBHelper.FORMAT_COL,
DBHelper.TIMESTAMP_COL,
                  DBHelper.DETAILS_COL, };

      private static final String[] COUNT_COLUMN = { "COUNT(1)" };
```



```
      private static final String[] ID_COL_PROJECTION = {
DBHelper.ID_COL };
      private static final String[] ID_DETAIL_COL_PROJECTION = {
DBHelper.ID_COL,
                  DBHelper.DETAILS_COL };
      private static final DateFormat EXPORT_DATE_TIME_FORMAT =
DateFormat
                  .getDateTimeInstance();

      private final Activity activity;

      public HistoryManager(Activity activity) {
            this.activity = activity;
      }

      public boolean hasHistoryItems() {
            SQLiteOpenHelper helper = new DBHelper(activity);
            SQLiteDatabase db = null;
            Cursor cursor = null;
            try {
                  db = helper.getReadableDatabase();
                  cursor = db.query(DBHelper.TABLE_NAME, COUNT_COLUMN,
null, null,
                              null, null, null);
                  cursor.moveToFirst();
                  return cursor.getInt(0) > 0;
            } finally {
                  close(cursor, db);
            }
      }

      public List<HistoryItem> buildHistoryItems() {
            SQLiteOpenHelper helper = new DBHelper(activity);
            List<HistoryItem> items = new ArrayList<HistoryItem>();
            SQLiteDatabase db = null;
            Cursor cursor = null;
            try {
                  db = helper.getReadableDatabase();
                  cursor = db.query(DBHelper.TABLE_NAME, COLUMNS,
null, null, null,
                              null, DBHelper.TIMESTAMP_COL + " DESC");
                  while (cursor.moveToNext()) {
                        String text = cursor.getString(0);
                        String display = cursor.getString(1);
                        String format = cursor.getString(2);
                        long timestamp = cursor.getLong(3);
                        String details = cursor.getString(4);
                        Result result = new Result(text, null, null,
                                    BarcodeFormat.valueOf(format),
timestamp);
//*** If its not a book do not add it in history
                        if(details!=null)
                        {
                        items.add(new HistoryItem(result, display,
details));
                        }
                  }
            } finally {
                  close(cursor, db);
            }
```

```
            return items;
      }

      public HistoryItem buildHistoryItem(int number) {
            SQLiteOpenHelper helper = new DBHelper(activity);
            SQLiteDatabase db = null;
            Cursor cursor = null;
            try {
                  db = helper.getReadableDatabase();
                  cursor = db.query(DBHelper.TABLE_NAME, COLUMNS,
null, null, null,
                              null, DBHelper.TIMESTAMP_COL + " DESC");
                  cursor.move(number + 1);
                  String text = cursor.getString(0);
                  String display = cursor.getString(1);
                  String format = cursor.getString(2);
                  long timestamp = cursor.getLong(3);
                  String details = cursor.getString(4);
                  Result result = new Result(text, null, null,
                              BarcodeFormat.valueOf(format),
timestamp);
                  return new HistoryItem(result, display, details);
            } finally {
                  close(cursor, db);
            }
      }

      public void deleteHistoryItem(int number) {
            SQLiteOpenHelper helper = new DBHelper(activity);
            SQLiteDatabase db = null;
            Cursor cursor = null;
            try {
                  db = helper.getWritableDatabase();
                  cursor = db.query(DBHelper.TABLE_NAME,
ID_COL_PROJECTION, null,
                              null, null, null, DBHelper.TIMESTAMP_COL
+ " DESC");
                  cursor.move(number + 1);
                  db.delete(DBHelper.TABLE_NAME,
                              DBHelper.ID_COL + '=' +
cursor.getString(0), null);
            } finally {
                  close(cursor, db);
            }
      }

      public void addHistoryItem(Result result, ResultHandler handler)
{
            // Do not save this item to the history if the preference
is turned off,
            // or the contents are
            // considered secure.
            if
(!activity.getIntent().getBooleanExtra(Intents.Scan.SAVE_HISTORY,
                        true)
                        || handler.areContentsSecure()) {
                  return;
            }

            SharedPreferences prefs = PreferenceManager
```



```
                        .getDefaultSharedPreferences(activity);
            if
(!prefs.getBoolean(PreferencesActivity.KEY_REMEMBER_DUPLICATES,
                        false)) {
                  deletePrevious(result.getText());
            }

            ContentValues values = new ContentValues();
            values.put(DBHelper.TEXT_COL, result.getText());
            values.put(DBHelper.FORMAT_COL,
result.getBarcodeFormat().toString());
            values.put(DBHelper.DISPLAY_COL,
handler.getDisplayContents()
                        .toString());
            values.put(DBHelper.TIMESTAMP_COL,
System.currentTimeMillis());

            SQLiteOpenHelper helper = new DBHelper(activity);
            SQLiteDatabase db = null;
            try {
                  db = helper.getWritableDatabase();
                  // Insert the new entry into the DB.
                  db.insert(DBHelper.TABLE_NAME,
DBHelper.TIMESTAMP_COL, values);
            } finally {
                  close(null, db);
            }
      }

      public void addHistoryItemDetails(String itemID, String
itemDetails) {
            // As we're going to do an update only we don't need need
to worry
            // about the preferences; if the item wasn't saved it
won't be udpated
            SQLiteOpenHelper helper = new DBHelper(activity);
            SQLiteDatabase db = null;
            Cursor cursor = null;
            try {
                  db = helper.getWritableDatabase();
                  cursor = db.query(DBHelper.TABLE_NAME,
ID_DETAIL_COL_PROJECTION,
                              DBHelper.TEXT_COL + "=?", new String[] {
itemID }, null,
                              null, DBHelper.TIMESTAMP_COL + " DESC",
"1");
                  String oldID = null;
                  String oldDetails = null;
                  if (cursor.moveToNext()) {
                        oldID = cursor.getString(0);
                        oldDetails = cursor.getString(1);
                  }

                  String newDetails = oldDetails == null ? itemDetails
: oldDetails
                              + " : " + itemDetails;
                  ContentValues values = new ContentValues();
                  values.put(DBHelper.DETAILS_COL, newDetails);
```

```
                  db.update(DBHelper.TABLE_NAME, values,
DBHelper.ID_COL + "=?",
                              new String[] { oldID });

            } finally {
                  close(cursor, db);
            }
      }

      private void deletePrevious(String text) {
            SQLiteOpenHelper helper = new DBHelper(activity);
            SQLiteDatabase db = null;
            try {
                  db = helper.getWritableDatabase();
                  db.delete(DBHelper.TABLE_NAME, DBHelper.TEXT_COL +
"=?",
                              new String[] { text });
            } finally {
                  close(null, db);
            }
      }

      public void trimHistory() {
            SQLiteOpenHelper helper = new DBHelper(activity);
            SQLiteDatabase db = null;
            Cursor cursor = null;
            try {
                  db = helper.getWritableDatabase();
                  cursor = db.query(DBHelper.TABLE_NAME,
ID_COL_PROJECTION, null,
                              null, null, null, DBHelper.TIMESTAMP_COL
+ " DESC");
                  cursor.move(MAX_ITEMS);
                  while (cursor.moveToNext()) {
                        db.delete(DBHelper.TABLE_NAME,
                                    DBHelper.ID_COL + '=' +
cursor.getString(0), null);
                  }
            } finally {
                  close(cursor, db);
            }
      }

      /**
       * <p>
       * Builds a text representation of the scanning history. Each
scan is
       * encoded on one line, terminated by a line break (\r\n). The
values in
       * each line are comma-separated, and double-quoted. Double-
quotes within
       * values are escaped with a sequence of two double-quotes. The
fields
       * output are:
       * </p>
       *
       * <ul>
       * <li>Raw text</li>
       * <li>Display text</li>
       * <li>Format (e.g. QR_CODE)</li>
```



```
       * <li>Timestamp</li>
       * <li>Formatted version of timestamp</li>
       * </ul>
       */
      CharSequence buildHistory() {
/*****************************************************************/
            // diagrafi tou file etsi wste na min vazei duplicates!
            file.delete();
/*****************************************************************/
            StringBuilder historyText = new StringBuilder(1000);
            SQLiteOpenHelper helper = new DBHelper(activity);
            SQLiteDatabase db = null;
            Cursor cursor = null;
            try {
                  db = helper.getWritableDatabase();
                  cursor = db.query(DBHelper.TABLE_NAME, COLUMNS,
null, null, null,
                              null, DBHelper.TIMESTAMP_COL + " DESC");

                  while (cursor.moveToNext()) {

                        historyText.append('"')

      .append(massageHistoryField(cursor.getString(0)))
                              .append("\",");
                        historyText.append('"')

      .append(massageHistoryField(cursor.getString(1)))
                              .append("\",");
                        historyText.append('"')

      .append(massageHistoryField(cursor.getString(2)))
                              .append("\",");
                        historyText.append('"')

      .append(massageHistoryField(cursor.getString(3)))
                              .append("\",");

                        // Add timestamp again, formatted
                        long timestamp = cursor.getLong(3);
                        historyText
                              .append('"')

      .append(massageHistoryField(EXPORT_DATE_TIME_FORMAT
                                          .format(new
Date(timestamp)))).append("\",");

                        // Above we're preserving the old ordering of
columns which had
                        // formatted data in position 5

                        historyText.append('"')

      .append(massageHistoryField(cursor.getString(4)))
                              .append("\"\r\n");

/*****************************************************************/
                        try {
                              String isbn = cursor.getString(0);
```

```
                              BufferedWriter out = new
BufferedWriter(new FileWriter(

      "/mnt/sdcard/media/books.txt", true));

                              URL url = new URL(

      "https://ajax.googleapis.com/ajax/services/search/books?v=1.0&q=
ISBN"
                                                      + isbn
                                                      +
"&key=ABQIAAAA94MJuF2gFa9KqeGTuI0EjBSkkyQUZK2-
HCU2v8Q1ngVNh8D3JBThgk7ik35T7gM0BpiNLjAu-mKyIg&userip=192.168.0.1");
                              URLConnection connection =
url.openConnection();
                              connection

      .addRequestProperty("Referer",

      "https://ajax.googleapis.com/ajax/services/search/books");

                              String line;
                              //String isbn1;
                              StringBuilder builder = new
StringBuilder();
                              BufferedReader reader = new
BufferedReader(
                                          new
InputStreamReader(connection.getInputStream()));
                              while ((line = reader.readLine()) !=
null) {
                                    builder.append(line);
                              }
                              // Creates a new InputStream object of
the entity.
                              try {
JSONObject json = new JSONObject(builder.toString());
JSONObject res = json.getJSONObject("responseData");
JSONArray results = res.getJSONArray("results");

JSONObject fields = results.getJSONObject(0);

out.append("book ISBN:" + isbn + "\n");
out.append("title:" + fields.getString("title") + "\n");
out.append("authors:" + fields.getString("authors")+ "\n")
out.append("published Year:"+ fields.getString("publishedYear")+\n");
out.append("pageCount:" + fields.getString("pageCount")+ "\n");
out.append("date of scan:"+ EXPORT_DATE_TIME_FORMAT.format(new
Date(timestamp))).append("\"\r\n\n");

String str=fields.getString("title");
String newTitle=str.replace("'", "''");
String str1=URLDecoder.decode(fields.getString("tbUrl"));
String newUrl=str1.replace("/googlebooks/images/no_cover_thumb.gif",
"../smartscan/googlebooks/images/no_cover_thumb.gif");


addBook(isbn, newTitle, fields.getString("authors"),
fields.getString("publishedYear") ,
fields.getString("pageCount"),"0.0", newUrl);
```



```
                              } catch (JSONException e1) {
                              }

                              out.close();
                        } catch (IOException e) {
                        }

/*********************************************************************/

                  }
                  return historyText;
            } finally {
                  close(cursor, db);
            }
      }

      void clearHistory() {
            SQLiteOpenHelper helper = new DBHelper(activity);
            SQLiteDatabase db = null;
            try {
                  db = helper.getWritableDatabase();
                  db.delete(DBHelper.TABLE_NAME, null, null);
            } finally {
                  close(null, db);
            }
      }

      static Uri saveHistory(String history) {
            File bsRoot = new
File(Environment.getExternalStorageDirectory(),
                        "BarcodeScanner");
            File historyRoot = new File(bsRoot, "History");
            if (!historyRoot.exists() && !historyRoot.mkdirs()) {
                  Log.w(TAG, "Couldn't make dir " + historyRoot);
                  return null;
            }
            File historyFile = new File(historyRoot, "history-"
                        + System.currentTimeMillis() + ".csv");
            OutputStreamWriter out = null;
            try {
                  out = new OutputStreamWriter(new
FileOutputStream(historyFile),
                              Charset.forName("UTF-8"));
                  out.write(history);
                  return Uri.parse("file://" +
historyFile.getAbsolutePath());
            } catch (IOException ioe) {
                  Log.w(TAG, "Couldn't access file " + historyFile + "
due to " + ioe);
                  return null;
            } finally {
                  if (out != null) {
                        try {
                              out.close();
                        } catch (IOException ioe) {
                              // do nothing
                        }
                  }
            }
```

```
	}

	private static String massageHistoryField(String value) {
		return value == null ? "" : value.replace("\"", "\"\"");
	}

	private static void close(Cursor cursor, SQLiteDatabase
database) {
		if (cursor != null) {
			cursor.close();
		}
		if (database != null) {
			database.close();
		}
	}

	public Boolean addBook(String isbn, String title, String
authors,
			String publishedYear, String pageCount, String rate,
String imageUrl) {

		//String link =
"http://www.cs.ucy.ac.cy/thesis/smartscan/connection/addBook.php";
		String link =
"http://www.cs.ucy.ac.cy/thesis/smartscan/connection/testingAddBook.ph
p";


		ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
		nameValuePairs.add(new
BasicNameValuePair("username",Connection.username));
		nameValuePairs.add(new BasicNameValuePair("isbn", isbn +
""));
		nameValuePairs.add(new BasicNameValuePair("title",
title));
		nameValuePairs.add(new BasicNameValuePair("authors",
authors));
		nameValuePairs.add(new
BasicNameValuePair("publishedYear",publishedYear + ""));
		nameValuePairs.add(new BasicNameValuePair("pageCount",
pageCount + ""));
		nameValuePairs.add(new BasicNameValuePair("rate", rate +
""));
		nameValuePairs.add(new BasicNameValuePair("imageUrl",
imageUrl));

		String result = getResults(link, nameValuePairs);

		try {

			JSONObject json_data = new JSONObject(result);


			if (json_data.getString("isbn") != null) {
				// Toast.makeText(get,
				// "Succesfully added: " + title,
Toast.LENGTH_SHORT)
				// .show();
			}
```



```
		} catch (JSONException e) {
			System.out.println("ErrorG " + e.getMessage());
			// Log.e("log_tag", "Error parsing data " +
e.toString());
		}

		return true;
	}

	public String getResults(String link, ArrayList<NameValuePair>
values) {

		String result = null;
		InputStream is = null;

		// http post
		try {
			HttpClient httpclient = new DefaultHttpClient();
			HttpPost httppost = new HttpPost(link);
			httppost.setEntity(new
UrlEncodedFormEntity(values));
			HttpResponse response =
httpclient.execute(httppost);
			HttpEntity entity = response.getEntity();
			is = entity.getContent();
		} catch (Exception e1) {
			Log.e("log_tag", "Error in http connection " +
e1.toString());
		}

		// convert response to string
		try {
			BufferedReader reader = new BufferedReader(new
InputStreamReader(
					is, "iso-8859-1"), 8);
			StringBuilder sb = new StringBuilder();
			String line = null;
			while ((line = reader.readLine()) != null) {
				sb.append(line + "\n");
			}
			is.close();

			result = sb.toString();
		} catch (Exception e) {
			Log.e("log_tag", "Error converting result " +
e.toString());
		}

		return result;
	}
}
```

A.2 Smartscan Ιστοσελίδα

site_login.php

```
<?php include 'header.php'; ?>
```

```
<script type="text/javascript">
    jQuery(document).ready(function(){
            $('#menu li').eq(4).attr('id', 'menu_active')
      });
</script>
<?php


//This displays your login form

function index(){

echo "<form id='ContactForm' action='?act=login'
method='post'><center><table id='login'>"

."<tr><td>Username:</td> <td><div class='bg'><input type='text'
name='username' size='30'></div></td></tr>"

."<tr><td>Password:</td> <td><div class='bg'><input type='password'
name='password' size='30'></div></td></tr>"

."<tr><td colspan=2 align ='right'><input type='submit' class='button'
value='Login'></td></tr>"

."</table></center>"

."</form>";

}

//This function will find and checks if your data is correct



function login(){

//Collect your info from login form

$username = $_REQUEST['username'];
$password = $_REQUEST['password'];

//Connecting to database
$connect =
mysql_connect("dbserver.cs.ucy.ac.cy:3306","smartscan","UZaXYqXBdKX6cr
mq");

if(!$connect)
{
die("Connection Error: " . mysql_error());
}

//Selecting database

$database="smartscan";

mysql_select_db($database) or die("Error conecting to db.");

//Find if entered data is correct
```



```
$result = mysql_query("SELECT user_id,username,password,email FROM
USER WHERE username='$username' AND password='$password'");

$row = mysql_fetch_array($result);

$id = $row['user_id'];

$select_user = mysql_query("SELECT user_id,username,password,email
FROM USER WHERE user_id='$id'");

$row2 = mysql_fetch_array($select_user);

$user = $row2['username'];

if($username == null){
  if($password==null)
      die("Please insert a username and  password!");
  else
      die("Please insert a username");
}



if($username != $user){
  if($password==null)
      die("Please insert a password");
  else
      die("Username is wrong!");
}

$pass_check = mysql_query("SELECT user_id,username,password,email FROM
USER WHERE username='$username' AND user_id='$id'");

$row3 = mysql_fetch_array($pass_check);

$email = $row3['email'];

$select_pass = mysql_query("SELECT user_id,username,password,email
FROM USER WHERE username='$username' AND user_id='$id' AND
email='$email'");

$row4 = mysql_fetch_array($select_pass);

$real_password = $row4['password'];

if($username!=null and $password==null){

die("Please insert a username and a password!");
}

if($password != $real_password){

die("Your password is wrong!");

}

//Now if everything is correct let's finish his/her/its login

$_SESSION['username'] = $username;
$_SESSION['password'] = $password;
```

```
   $ok=1;
   header("Location:
http://www.cs.ucy.ac.cy/thesis/smartscan/index.php");

   ?>
   <script type="text/javascript">

window.location.replace("http://www.cs.ucy.ac.cy/thesis/smartscan/inde
x.php");
   </script>
   <?
   exit();

}

$act = $_REQUEST['act'];


switch($act){


default;


index();

break;


case "login";



login();


break;

}


?>

<?php include 'footer.php'; ?>
```

logout.php

```
<?php include 'header.php'; ?>

<?php

session_start();

//This function will destroy your session
session_destroy();
```



```
      header("Location:
http://www.cs.ucy.ac.cy/thesis/smartscan/index.php");

   ?>
   <script type="text/javascript">

window.location.replace("http://www.cs.ucy.ac.cy/thesis/smartscan/inde
x.php");
   </script>
   <?
   exit();

?>

<?php include 'footer.php'; ?>
```

register.php

```
<?php include 'header.php'; ?>
<script type="text/javascript">
    jQuery(document).ready(function(){
            $('#menu li').eq(3).attr('id', 'menu_active')
     });
</script>
<?php

//This function will display the registration form

function register_form(){

echo "<form id='ContactForm' action='?act=register'
method='post'><center><table id='register' cellspacing='5px'>"

."<tr><td>Name:</td>                              <td><div
class='bg'><input type='text' class='input' name='name'
size='30'></div></td></tr>"

."<tr><td>Surname:</td>                   <td><div class='bg'><input
type='text' class='input' name='surname' size='30'></div></td></tr>"

."<tr><td>Username:</td>                  <td><div class='bg'><input
type='text' class='input' name='username' size='30'></div></td></tr>"

."<tr><td>Password:</td>                  <td><div class='bg'><input
type='password' class='input' name='password'
size='30'></div></td></tr>"

."<tr><td>Confirm your password:</td> <td><div class='bg'><input
type='password' class='input' name='password_conf'
size='30'></div></td></tr>"

."<tr><td>Email:</td>                     <td><div class='bg'><input
type='text' class='input' name='email' size='30'></div></td></tr>"

."<tr><td colspan='2' align='right'><input type='submit'
class='button' value='Register'></td></tr>"

."</table></center>"
```

```
."</form>";

}

//This function will register users data

function register(){

//Connecting to database

$connect =
mysql_connect("dbserver.cs.ucy.ac.cy:3306","smartscan","UZaXYqXBdKX6cr
mq");

if(!$connect){

die("Connection Error: " . mysql_error());

}

//Selecting database
$database="smartscan";
mysql_select_db($database) or die("Error conecting to db.");

//Collecting info

$name = $_REQUEST['name'];

$surname = $_REQUEST['surname'];

$username = $_REQUEST['username'];

$password = $_REQUEST['password'];

$pass_conf = $_REQUEST['password_conf'];

$email = $_REQUEST['email'];

//$date = $_REQUEST['date'];

//Here we will check do we have all inputs filled

if(empty($name)){

die("Please enter your name!<br>");

}

if(empty($surname)){

die("Please enter your surname!<br>");

}

if(empty($username)){
```



```
die("Please enter your username!<br>");

}

if(empty($password)){

die("Please enter your password!<br>");

}

if(empty($pass_conf)){

die("Please confirm your password!<br>");

}

if(empty($email)){

die("Please enter your email!");

}

//Let's check if this username is already in use

$user_check = mysql_query("SELECT username FROM USER WHERE
username='$username'");

$do_user_check = mysql_num_rows($user_check);

//Now if email is already in use

$email_check = mysql_query("SELECT email FROM USER WHERE
email='$email'");

$do_email_check = mysql_num_rows($email_check);

//Now display errors

if($do_user_check > 0){

die("Username is already in use!<br>");

}

if($do_email_check > 0){

die("Email is already in use!");

}
```

```
//Now let's check does passwords match


if($password != $pass_conf){

die("Passwords don't match!");

}

//If everything is okay let's register this user


$insert = mysql_query("INSERT INTO USER (name, surname, username,
password, email) VALUES ('$name', '$surname','$username', '$password',
'$email')");

if(!$insert){

die("There's a problem: ".mysql_error());

}


echo $username.", you are now registered. Thank you!<br><a
href=site_login.php>Login</a> | <a href=index.php>Index</a>";
}

$act = $_REQUEST['act'];

switch($act){


default;

register_form();

break;


case "register";

register();

break;

}
?>

<?php include 'footer.php'; ?>
```

my_search.php

```
<?php
$username1 = $_GET['username'];

      $page = $_GET['page']; // get the requested page
```



```
      $limit = $_GET['rows']; // get how many rows we want to have
into the grid

      //$sidx = $_GET['sidx']; // get index row - i.e. user click to
sort

      //$sord = $_GET['sord']; // get the direction

      //if(!$sidx) $sidx =1;


      $searchString = isset($_GET['searchString']) ?
$_GET['searchString'] : false;


      // connect to the database

      $username="smartscan";

      $password="UZaXYqXBdKX6crmq";

      $database="smartscan";


      $db =
mysql_connect("dbserver.cs.ucy.ac.cy:3306",$username,$password)

      or die("Connection Error: " . mysql_error());



      mysql_select_db($database) or die("Error conecting to db.");

      $result = mysql_query("Select COUNT(*) AS count FROM

                                          (SELECT
B.image,B.isbn,B.title,B.authors,B.publishedYear,B.pageCount

                                           FROM BOOK B,USER_HAS_BOOK
UHB,USER U

                                           WHERE U.username='$username1'
AND UHB.user_id=U.user_id AND UHB.isbn=B.isbn) test

            WHERE  isbn LIKE '%$searchString%' OR title LIKE
'%$searchString%' OR authors LIKE '%$searchString%' OR publishedYear
LIKE '%$searchString%' OR pageCount LIKE '%$searchString%'");



      $row = mysql_fetch_array($result,MYSQL_ASSOC);

      $count = $row['count'];

      //echo $row['count'];



      if( $count >0 ) {
```

```
            $total_pages = ceil($count/$limit);

      } else {

            $total_pages = 0;

      }

if ($page > $total_pages) $page=$total_pages;

$start = $limit*$page - $limit; // do not put $limit*($page - 1)


if ($start < 0 ) {$start = 0;}

$SQL = "SELECT * FROM

       (SELECT
B.image,B.isbn,B.title,B.authors,B.publishedYear,B.pageCount

        FROM BOOK B,USER_HAS_BOOK UHB,USER U

        WHERE U.username='$username1' AND UHB.user_id=U.user_id AND
UHB.isbn=B.isbn) test

        WHERE isbn LIKE '%$searchString%' OR title LIKE
'%$searchString%' OR authors LIKE '%$searchString%' OR publishedYear
LIKE '%$searchString%' OR pageCount LIKE '%$searchString%'

            LIMIT $start , $limit";


$result = mysql_query( $SQL ) or die("Couldn t execute
query.".mysql_error());



$responce->page = $page;

$responce->total = $total_pages;

$responce->records = $count;

$i=0;

while($row = mysql_fetch_array($result,MYSQL_ASSOC)) {

    $responce->rows[$i]['isbn']=$row[isbn];

 $responce-
>rows[$i]['cell']=array($row[image],$row[isbn],$row[title],$row[author
s],$row[publishedYear],$row[pageCount]);

    $i++;

}

echo json_encode($responce);
```



```
?>
```

star.php

```
<?php

//get the rating of user and the isbn of book tha is rated
$rating = $_REQUEST['rating'];
$book = $_REQUEST['book'];

//Connecting to database
$connect =
mysql_connect("dbserver.cs.ucy.ac.cy:3306","smartscan","UZaXYqXBdKX6cr
mq");

if(!$connect)
{
die("Connection Error: " . mysql_error());
}

//Selecting database
$database="smartscan";
mysql_select_db($database) or die("Error conecting to db.");

//update sumRate and Count

mysql_query("UPDATE BOOK
                SET
countRated=countRated+1,sumRate=sumRate+'$rating',rate=sumRate/countRa
ted
                WHERE isbn='$book'");

$newrating=mysql_query("SELECT countRated,ROUND(rate)
                                    FROM BOOK
                                    WHERE isbn='$book';");

?>
```

contact.php

```
<?php include 'header.php'; ?>
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript" src="js/jquery.gmap.min.js"></script>
<script type="text/javascript" src="js/custom.js"></script>


<script type="text/javascript">
    jQuery(document).ready(function(){
            $('#menu li').eq(1).attr('id', 'menu_active')
```

```
      });
</script>


<div id="map"></div>

<?php

      $name = $_REQUEST['name'];
      $email = $_REQUEST['email'];
      $message = $_REQUEST['message'];

      if (isset($_POST['name']) && isset($_POST['email']) &&
isset($_POST['message'])) {
            if ($name != '' && $email != '' & $message != '') {

                  $to = "gabi_gew@hotmail.com";
                  $subject = "Smartscan Email";
                  $body = "Name: ". $name . "\n Email: " . $email. "\n
Message: ". $message."\n";
                  if (mail($to, $subject, $body)) {
                        echo "<div class='success'>Succesfully Send
Email</div>";
                  } else {
                        echo "<div class='error'>Server error. Cannot
send email</div>";
                  }
            }
            else {
                  echo "<div class='error'>Please fill all the
required fields!</div>";
            }
      }
?>

<center><div id="contact">

<p>Project Implementation: <a
href="mailto:gkoupe01@cs.ucy.ac.cy">Gabriela Koupepidou</a></p>
<p>Project Supervision:<a href="http://www2.cs.ucy.ac.cy/~dzeina/"
target="_blank">Dr. Demetris Zeinalipour</a></p>
<p> Data Management Systems Laboratory <br>
      Dept. of Computer Science <br>
      University of Cyprus <br>
    P.O. Box 20537 <br>
      1678 Nicosia, CYPRUS <br>
</p>

<p>Email: <a href="mailto:dmsl@cs.ucy.ac.cy">dmsl@cs.ucy.ac.cy</a><br>
      Tel: +357-22-892755<br>
      Fax: +357-22-892701
</p>
</div>

<div id="sendEmail">
<form id="ContactForm" method="post">
<center>
<table id="register" cellspacing="5px" border="10">
```



```
<tr><td>Name:</td>                              <td><div
class="bg"><input type="text"class="input" name="name"
size="30"></div></td></tr>

<tr><td>Email:</td>                        <td><div class="bg"><input
type="text" class="input" name="email" size="30"></div></td></tr>

<tr><td>Message:</td>               <td height="200"><div
class="bg"><textarea name="message" cols="5"
rows="30"></textarea></div></td></tr>


<tr><td colspan="2" align="right"><input type="submit" class="button"
value="Send"></td></tr>

</table></center>

</form>;

</div>

</center>
<?php include 'footer.php'; ?>
```

addBook.php

```
<?php

//first create a connection to your database...
function connection(){
      $username="smartscan";
      $password="UZaXYqXBdKX6crmq";
      $database="smartscan";
      $host="dbserver.cs.ucy.ac.cy";

      $mysqli = new mysqli($host,$username, $password, $database);

            if (mysqli_connect_errno()) {
                  printf("Connect failed: %s\n",
mysqli_connect_error());
                  exit();
            }

            return $mysqli;

      }
//second create the function to call the stored procedure
function query($query){
//call the connection function
  if($result = connection()->query($query)){
                  while($row = $result->fetch_assoc()){
                              $data[] = $row;
                  }
                  return $data;
                  }else {
                  print('error on:'.$query.mysqli_error($this));
                      }
```

```
 }

      $username = $_REQUEST['username']; // get the requested page

      $isbn = $_REQUEST['isbn']; // get the requested page
      $title = $_REQUEST['title']; // get how many rows we want to
have into the grid
      $authors = $_REQUEST['authors']; // get the requested page
      $publishedYear = $_REQUEST['publishedYear']; // get how many
rows we want to have into the grid
      $pageCount = $_REQUEST['pageCount']; // get the requested page
      $rate = $_REQUEST['rate']; // get how many rows we want to have
into the grid
      $scanDate = date('M,d,Y');
      $countScanned = 1;
      $imageUrl = $_REQUEST['imageUrl']; // get how many rows we want
to have into the grid

//then call the function to execute stored procedure
      $result=query("CALL
sp_testingInput('".$username."','".$isbn."','".$title."','".$authors."
','".$publishedYear."','".$pageCount."','".$scanDate."','".$countScann
ed."','".$rate."','".$imageUrl. "');");

if($result!=NULL)
{     echo json_encode($result[0]);
    echo ("GABRIELA");
}
else {
      $ans = array(
      "Answer"=>"false"
      );
}
?>
```