ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία barcode scanner με Cordova

Του φοιτητή

Δημητρίου Βάντσου

Αρ. Μητρώου: 134107

Επιβλέπων καθηγητής

Κωνσταντίνος Διαμαντάρας

Θεσσαλονίκη 2018

ΠΡΟΛΟΓΟΣ

Η ανάπτυξη εφαρμογών σε mobile συσκευές βλέπει μεγάλη άνθηση στην εποχή μας, καθώς αυτές έχουν γίνει πλέον αναπόσπαστο κομμάτι της κοινωνίας μας. Ο κάθε άνθρωπος κουβαλάει μαζί του τουλάχιστον μια mobile συσκευή, συνήθως κάποιο κινητό τηλέφωνο, ενώ είναι πολύ πιθανό να είναι επίσης κάτοχος ενός tablet.

Αυτό έχει επιφέρει ένα μεγάλο ενδιαφέρον σε προγράμματα κατασκευασμένα για να τρέχουν σε αυτές τις συσκευές. Ο όρος που χρησιμοποιείται για την περιγραφή των προγραμμάτων αυτών είναι ***εφαρμογές***.

Η συγκεκριμένη πτυχιακή εργασία αποτελεί ένα ορόσημο στην καριέρα μου, καθώς αποτελεί την πρώτη εφαρμογή που υλοποίησα και διέθεσα σε κάποιον άλλο για την εξυπηρέτησή του. Επιπλέον αποτελεί το ξεκίνημα της εμπειρίας μου σε cross-platform λογική και σηματοδοτεί την αρχή της καριέρας μου ως mobile developer, μιας καριέρας που έχω ήδη ξεκινήσει και ασκώ σε επαγγελματικό επίπεδο.

ΠΕΡΙΛΗΨΗ

Σε αυτή την εργασία γίνεται περιγραφή της υλοποίησης μιας mobile εφαρμογής ανάγνωσης Barcode με τη χρήση του framework Cordova.

Στα κεφάλαια που ακολουθούν γίνεται μια αναφορά στην τεχνολογία του barcode scanning και αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν για την πραγματοποίηση της εφαρμογής, όπως το jQuery Mobile, το AJAX, η SQLite και βεβαίως το Cordova.

Επιπλέον παρουσιάζονται πολλά κομμάτια του γραμμένου κώδικα και γίνεται αναφορά στη λογική πίσω από την εφαρμογή, στη χρήση και στο συνδυασμό των επιμέρους τεχνολογιών ώστε να επιτευχθεί μια εφαρμογή που μπορεί να λειτουργήσει στο περιβάλλον μιας επιχείρησης..

Τέλος, περιγράφονται πλεονεκτήματα και μειονεκτήματα που επιφέρει η χρήση της εφαρμογής, όπως επίσης και προτάσεις για τη μελλοντική βελτίωσή της.

ΕΥΧΑΡΙΣΤΙΕΣ

Πρωτίστως θα ήθελα να ευχαριστήσω τον πατέρα μου Βάντσο Κωνσταντίνο, τη μητέρα μου, Χατζούδη Αγγελική-Άννα και την αδελφή μου, Βάντσου Βασιλική-Ιωάννα για την ηθική και πρακτική υποστήριξη καθ' όλη τη σταδιοδρομία μου στη σχολή. Επιπλέον θέλω να ευχαριστήσω τους συναδέλφους μου κατά τη διάρκεια της πρακτικής μου στη HELEXPO ΔΕΘ. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον Καπετανάκη Κωσταντίνο για την καθοδήγησή του και για τις συμβουλές του στον τρόπο συγγραφής HTML και JavaScript και τους υπόλοιπους συναδέλφους για την επεξήγησή τους στην ανάγνωση Barcodes μέσω Barcode Reader και κάθε άλλη βοήθεια κατά τη διάρκεια της πρακτικής.

Επίσης πολλές ευχαριστίες στους τωρινούς μου συναδέλφους, Παρασκευόπουλο Σάκη, Δαρατζίκη Ιωάννη, Σωτήρχο Σταύρο, Μαρινόπουλο Παναγιώτη και στον υπεύθυνό μου Καρατζογιάννη Στέφανο, που με βοήθησαν και με βοηθούν ακόμα να μαθαίνω μεθόδους ορθής συγγραφής εφαρμογών.

Τέλος θα ήθελα να ευχαριστήσω πολύ τον επιβλέποντα καθηγητή μου, Διαμαντάρα Κωνσταντίνο, ο οποίος με βοήθησε όχι μόνο να πραγματοποιήσω αυτή την πτυχιακή εργασία αλλά και να συνεχίσω την καριέρα μου στην αγορά εργασίας.

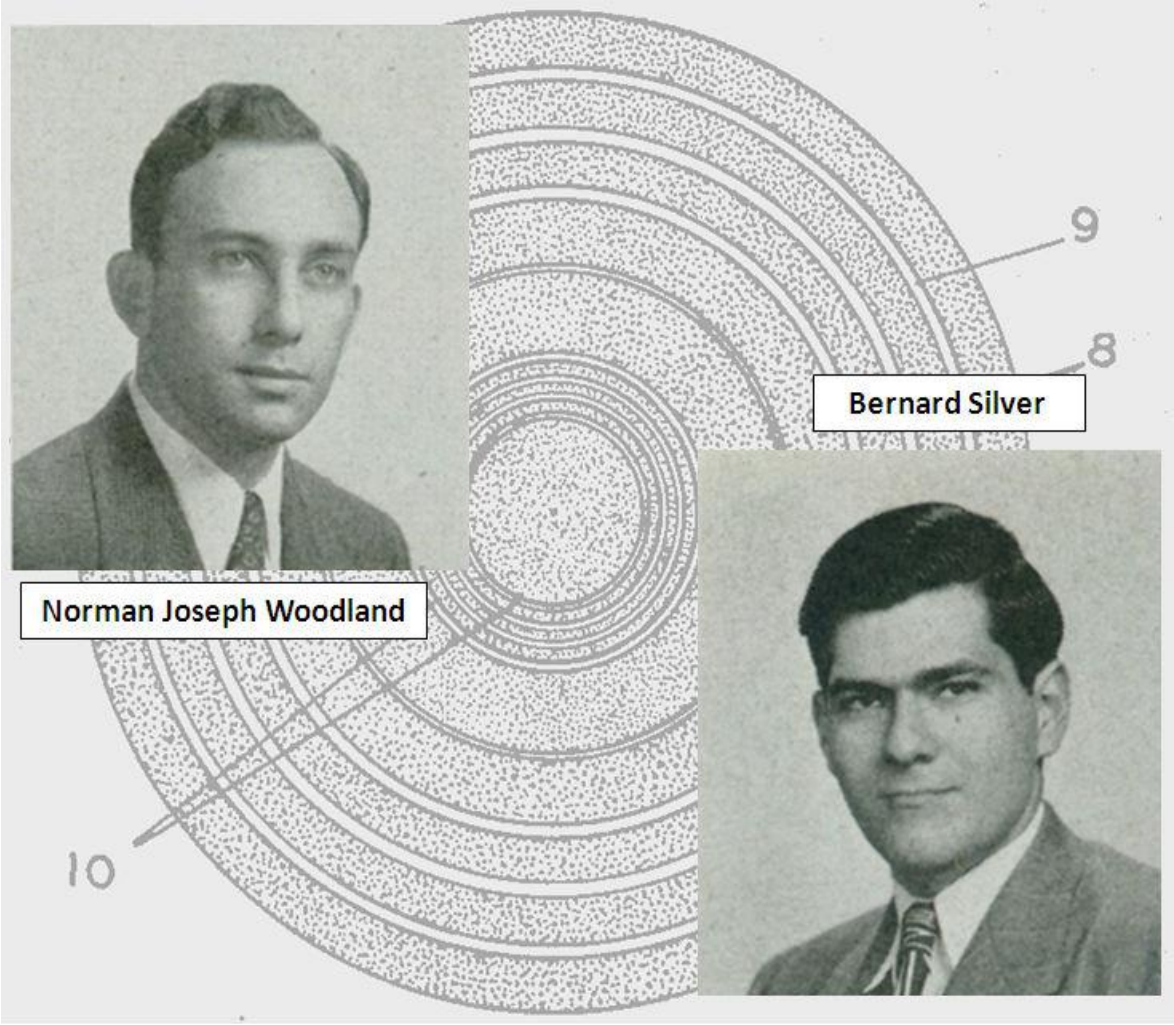
ΚΕΦΑΛΑΙΟ 1 - BARCODES

ΕΙΣΑΓΩΓΗ

Καθώς τα barcodes αποτελούν το κεντρικό κομμάτι της εφαρμογής, όπως επίσης και το σκοπό ύπαρξής της, σε αυτό το κεφάλαιο περιγράφεται η ιστορία και η λειτουργία των barcodes, καθώς και οι τρόποι με τους οποίους γίνεται η ανάγνωσή τους.

Η ΙΣΤΟΡΙΑ ΤΩΝ BARCODES

Η πρώτη μορφή Barcodes εφευρέθηκε το 1932 από τον Wallace Flint, ο οποίος επινόησε διάτρητες κάρτες που μπορούσαν να διαβαστούν από κάποια υπολογιστικά συστήματα σε ταμεία παντοπωλείων. Βέβαια η χρήση τους είχε περιορισμένες δυνατότητες, καθώς οι κάρτες αυτές δε μπορούσαν να εμπεριέχουν πολλά δεδομένα.



Εικόνα 1: Norman Joseph Woodland, Bernard Silver και Bull's Eye

12 χρόνια μετά, το 1948, ο Bernard Silver και ο φίλος του Norman Joseph Woodland, βελτίωσαν αυτήν την ιδέα χρησιμοποιώντας υπεριώδες φως για να ανιχνεύσουν γραμμές από μελάνη πάνω σε μια επιφάνια. Το σύστημα αυτό είχε επίσης πολλά μειονεκτήματα, καθώς κόστιζε αρκετά και δε μπορούσε να διαβάζει τη μελάνη με ακρίβεια.

Τον επόμενο χρόνο, το 1949, κατατέθηκε μια αίτηση ευρεσιτεχνίας ενός παρόμοιου συστήματος, το οποίο θα περιλάμβανε τη χρήση ομόκεντρων κύκλων το οποίο τελικά υλοποιήθηκε 3 χρόνια αργότερα ως το σύμβολο “Bull’s Eye”. Το σύστημα αυτό περιέγραφε επίσης την αρχή κωδικοποίησης με βάση το πάχος της γραμμής, πάνω στο όποιο βασίστηκαν αργότερα τα πρώτα barcodes.

Εικόνα 2: David J. Collins

Το πρώτο Barcode Scanner της μορφής που συναντάμε παντού αναπτύχθηκε από τον David J. Collins και την εταιρία Computer Identics Corporation. Το σύστημα αποτελούνταν από ετικέτες με άσπρες και μαύρες ράβδους διαφορετικού πλάτους σε συνδυασμό ενός laser reader. Οι πρώτες εφαρμογές του συστήματος αυτού έγιναν στη ναυτιλία.

Στην αγορά, τα Barcode Scanners κυκλοφορήσαν το 1969. Το Barcodes που μπορούσαν να διαβάσουν είχαν τη μορφή κωδικοποίησης, γνωστής και ως UGPIC (Universal Grocery Products Identification Code) που δημιούργησε η εταιρία Logicon Inc.

Η καθιερωμένη μορφή Barcode που χρησιμοποιείται ακόμη και σήμερα αποτελεί εξέλιξη της UGPIC και είναι η Universal Product Code (ή αλλιώς U.P.C.) από τον George J. Laurer το 1973.

Η ΛΕΙΤΟΥΡΓΙΑ ΤΩΝ BARCODES

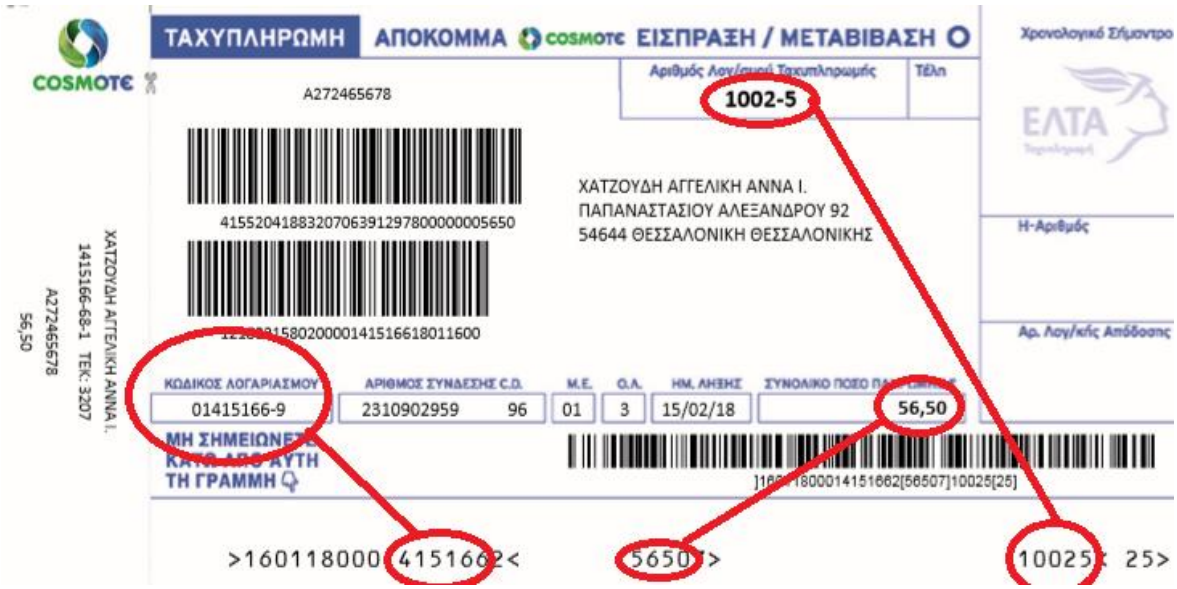
Εικόνα 3: Barcode

Τα barcodes είναι οπτικές αναπαραστάσεις δεδομένων. Αποτελούνται δηλαδή από πληροφορίες τροποποιημένες με κάποια γραφική αναπαράσταση. Τα πρώτα barcodes περιέγραφαν τα δεδομένα τους με τη χρήση μαύρων κάθετων γραμμών. Οι γραμμές αυτές τοποθετούνταν η μία δίπλα στην άλλη και είχαν διαφορετικά πλάτη και αποστάσεις μεταξύ τους. Αυτά τα δύο χαρακτηριστικά αναπαριστούσαν τα δεδομένα των barcode που είναι γνωστά ως 1D, όπως απεικονίζεται στην παραπάνω εικόνα.

Εικόνα 4: QR Code

Ενώ σήμερα κυκλοφορούν διάφοροι άλλοι τύποι barcodes, όπως οι 2D ή ακόμα και 3D codes, στην εργασία αυτή λαμβάνεται ως κύριος στόχος η ανάγνωση των 1D Barcodes, καθώς αυτοί είναι οι πιο διαδεδομένοι στη χρήση, ιδιαίτερα επάνω σε εισιτήρια.

Τα Barcodes περιγράφουν συνήθως κάποια πληροφορία για τα αντικείμενα στα οποία βρίσκονται. Για παράδειγμα, το barcode που βρίσκεται σε ένα εισιτήριο περιγράφει το μοναδικό ID του εισιτηρίου, τα barcodes σε λογαριασμούς εμπεριέχουν μέσα διάφορες πληροφορίες όπως το ID του λογαριασμού και το ποσό που αναλογεί σε αυτόν κ.ο.κ.



Εικόνα 5: Λογαριασμός του ΟΤΕ

Ας πάρουμε για παράδειγμα ένα λογαριασμό σταθερού τηλεφώνου της Cosmote. Εάν παρατηρήσουμε προσεκτικά τη συμβολοσειρά που αναπαριστά το Barcode μπορούμε να διακρίνουμε ορισμένα από τα χαρακτηριστικά του Barcode, όπως τον κωδικό του λογαριασμού (δίχως το ψηφίο ελέγχου), το οφειλόμενο ποσό και τον αριθμό λογαριασμού ταχυπληρωμής. Με την ανάγνωση αυτού του barcode, μπορεί κάποιος εύκολα να περάσει αυτά τα στοιχεία σε κάποιο μηχάνημα, δίχως να τα πληκτρολογήσει χειροκίνητα. Έτσι υπάρχει τόσο κέρδος σε χρόνο όσο και σε έλεγχο, καθώς ελαχιστοποιείται η πιθανότητα κάποιας λάθος εγγραφής.

Η ΑΝΑΓΝΩΣΗ ΜΕΣΩ BARCODE READER

Κατά παράδοση, η επικύρωση των διαφορετικών εισιτηρίων γίνεται μέσω ειδικών μηχανημάτων Barcode Reader. Τα Barcode Readers είναι μηχανήματα τα οποία συνδέονται σε υπολογιστές μέσω ενός ενσωματωμένου καλωδίου USB, διαβάζουν τα Barcodes και μετατρέπουν την αναγραφόμενη πληροφορία σε μια συμβολοσειρά (string).

Εικόνα 6: Barcode Reader με laser

Εικόνα 7: Imager

Υπάρχουν διάφοροι τρόποι λειτουργίας των Barcodes Readers. Οι δύο πιο κοινοί τύποι που υπάρχουν είναι οι Barcode Scanners με Laser και οι γραμμικοί σαρωτές εικονοληψίας, γνωστοί και ως Imagers. Ενώ οι πρώτοι χρησιμοποιούν μια δίοδο laser, το οποίο ανακλάται στον ασπρόμαυρο χώρο των Barcodes για την ανάγνωση, οι δεύτεροι στοχεύουν σε μια ολόκληρη περιοχή, παίρνοντας μεγαλύτερη περιοχή για ανάγνωση. Αυτό επιτρέπει στους Imagers να μπορούν να διαβάζουν τόσο 1D όσο και 2D codes.

Εικόνα 8: Barcode Reader με δύο λειτουργείες

Αξίζει επίσης να σημειωθεί ότι νεότερα μοντέλα Barcode Reader έρχονται με τη δυνατότητα να εναλλάσσονται μεταξύ λειτουργίας laser και imager.

BARCODES ΣΕ ΕΙΣΗΤΗΡΙΑ

Εικόνα 9: Εισιτήριο της ΔΕΘ 2017

Μετά την ανάγνωση ενός Barcode υπάρχει συνήθως μια εφαρμογή που διαχειρίζεται τη συμβολοσειρά που έχει παραχθεί.

Στη περίπτωση ανάγνωσης του Barcode ενός εισιτηρίου, η συμβολοσειρά στέλνεται συνήθως σε κάποιο server, ο οποίος αναζητά τη συμβολοσειρά μέσα σε κάποια βάση δεδομένων του και αναλόγως των κριτηρίων που θέτει αυτός (όπως για παράδειγμα εάν το εισιτήριο έχει λήξει, εάν έχει ήδη επικυρωθεί κ.ο.κ.) θα επικυρώσει ή θα απορρίψει το εισιτήριο.

Σε κάθε περίπτωση, ο server θα στείλει ένα μήνυμα πίσω στη συσκευή που έστειλε το Barcode, αναγράφοντας την επιτυχία ή αποτυχία της επικύρωσης του εισιτηρίου.

Τέλος η συσκευή είναι υπεύθυνη να διαμορφώσει το μήνυμα που έλαβε και να ενημερώσει τους εμπλεκόμενους για το αποτέλεσμα, είτε μέσω ενός μηνύματος σε μια οθόνη είτε αναπαράγοντας κάποιο ήχο δηλώνοντας επιτυχία ή αποτυχία.

ΤΟ ΠΡΟΒΛΗΜΑ

Ένα από τα προβλήματα που υπάρχουν σε αυτή τη διαδικασία είναι η καθολική εξάρτηση όλου του συστήματος από τη συσκευή που αναλαμβάνει να διαχειριστεί το Barcode, που σχεδόν πάντα είναι ένας υπολογιστής. Μια βλάβη σε αυτόν μπορεί να καθυστερήσει ή ακόμα και να διακόψει τελείως τη διαδικασία, ενώ η διόρθωση των προβλημάτων που ενδέχεται να παρουσιαστούν σε αυτόν, απαιτούν συνήθως κάποιον επαγγελματία μηχανικό,

εφοδιασμένο με αρκετό χρόνο. Επιπλέον, η αντικατάσταση ενός προβληματικού υπολογιστή δεν είναι εύκολη απόφαση, καθώς υπάρχουν οι παράγοντες του μεγάλου όγκου και του κόστους.

Η σοβαρότητα του προβλήματος αυτού είναι μεγάλη. Καθώς η ανάγνωση Barcode γίνεται συνήθως σε ουρές με πολλούς πελάτες, μια βλάβη στην όλη διαδικασία προκαλεί καθυστερήσεις, οι οποίες επιφέρουν λιγότερες και μεγαλύτερες ουρές, πολύ αναστάτωση και δυσαρέσκεια στους εμπλεκόμενους, ενώ συνήθως αφήνει κάποιο εργαζόμενο δίχως τα απαραίτητα εργαλεία για να βοηθήσει την κατάσταση.

ΕΠΙΛΟΓΟΣ

Τα Barcodes αποτελούν την αναπαράσταση κάποιων πληροφορίων, κρυπτογραφημένη με κάποια γραφική αναπαράσταση επάνω σε κάποια επιφάνεια.

Η αποκρυπτογράφησή τους γίνεται με ειδικά μηχανήματα που διαβάζουν και αποκρυπτογραφούν αυτή τη πληροφορία και την επιστρέφουν ως μια συμβολοσειρά.

Τη συμβολοσειρά αυτή τη διαχειρίζεται ένα πρόγραμμα που συνήθως συνεργάζεται με κάποιο server, ο οποίος θα επικυρώσει ή θα απορρίψει το barcode αυτό.

Σε αυτή την εργασία επιχειρείται να δοθεί μια εναλλακτική υλοποίηση αυτής της διαδικασίας, καθώς η ολική εξάρτηση από τον υπολογιστή μπορεί να επιφέρει προβλήματα εάν παρουσιαστούν σφάλματα σε αυτόν.

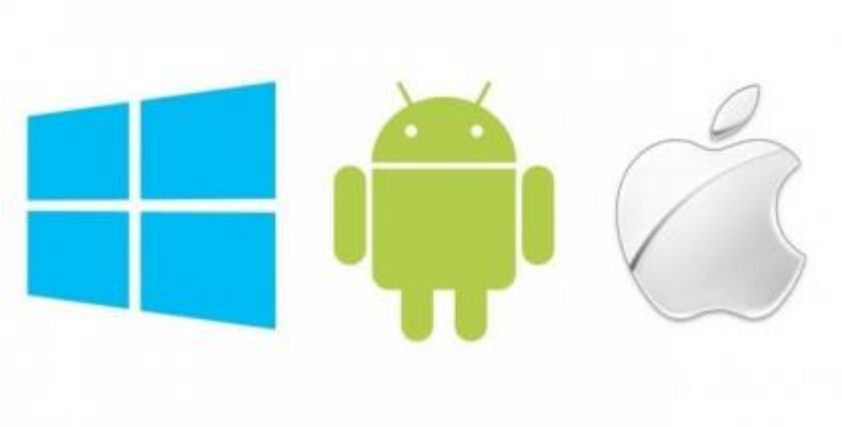
ΚΕΦΑΛΑΙΟ 2 - CORDOVA

ΕΙΣΑΓΩΓΗ

Η εναλλακτική λύση που περιγράφεται στο προηγούμενο κεφάλαιο αφορά την κατάργηση του υπολογιστή από τη διαδικασία της ανάγνωσης Barcode. Καθώς υπάρχει κίνδυνος βλάβης στη συσκευή που θα αντικαταστήσει τον υπολογιστή, πρέπει αυτή να είναι εύκολα αντικαταστάσιμη, ακόμα και σε ώρα αιχμής. Βασικά χαρακτηριστικά που πρέπει λοιπόν να έχει η νέα λύση είναι χαμηλό κόστος συντήρησης ή/και αντικατάστασης, μικρό όγκο και βάρος και πρέπει να μπορεί να αντικατασταθεί διακριτικά και σε μικρό χρόνο.

Εικόνα 10: Mobile Devices

Οι συσκευές που επιλέχθηκαν ως εναλλακτικό μέσω ανάγνωσης και επεξεργασίας των Barcode δεν είναι άλλες από τις διάφορες mobile συσκευές. ***Mobile συσκευές*** μπορούν να θεωρηθούν όλες οι συσκευές οι οποίες είναι σχετικά μικρές, εύκολα μετακινήσιμες και έχουν τη δυνατότητα να τρέχουν τα διάφορα προγράμματα που είναι σχεδιασμένα για αυτές, τις λεγόμενες εφαρμογές.



Εικόνα 11: Mobile Platforms

Υπάρχουν διαφορετικά λειτουργικά συστήματα για mobile συσκευές. Τα πιο διαδεδομένα από αυτά είναι το Android της Google, το iOS της Apple και το Windows Phone της Microsoft.

Για κάθε μια από αυτές τις πλατφόρμες πρέπει να χρησιμοποιείται διαφορετική γλώσσα προγραμματισμού:

- Για Android: Java και XML,
- Για iOS: Swift
- Για Windows Phone: Visual C++, C#, Visual Basic ή JavaScript.

Για να τρέξει το κάθε πρόγραμμα σε ξεχωριστή πλατφόρμα πρέπει να ξαναγραφεί από την αρχή με την κατάλληλη γλώσσα. Αυτό είναι προφανώς χρονοβόρο για ένα προγραμματιστή και απαιτεί καλές γνώσεις όλων των γλωσσών, καθώς η κάθε γλώσσα έχει τις δικιές της ιδιομορφίες.

Ευτυχώς υπάρχουν Frameworks, με τα οποία είναι δυνατή η δημιουργία εφαρμογών για πολλαπλές πλατφόρμες, συγγράφοντας όμως μόνο ένα κώδικα, μέσα από τον οποίο θα δημιουργηθεί η εφαρμογή στη κάθε πλατφόρμα ξεχωριστά.

Το δεύτερο συστατικό κομμάτι της εφαρμογής, που είναι και η αιτία συγγραφής αυτής της πτυχιακής εργασίας είναι η μελέτη και η χρήση του Cordova, ενός τέτοιου Framework, για την κατασκευή εφαρμογών σε mobile συσκευές. Σε αυτό το κεφάλαιο αναλύεται ποια είναι η χρησιμότητα του Cordova, που και πώς χρησιμοποιείται.

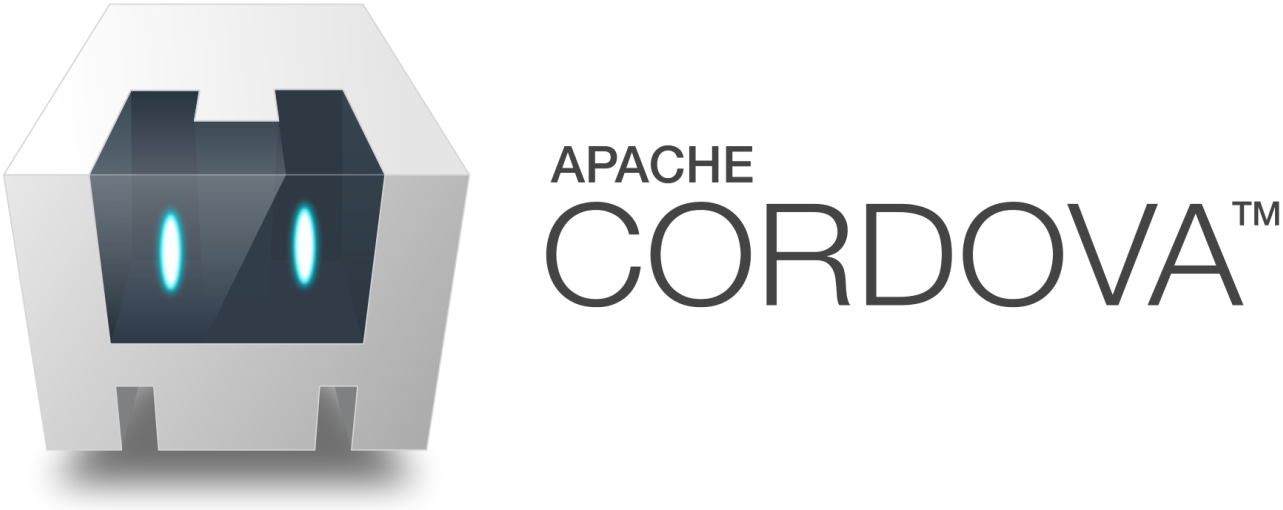
ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Εικόνα 12: PhoneGap logo

Το Cordova δεν είχε πάντα το ίδιο όνομα. Αναπτύχθηκε αρχικά από την εταιρία Nitobi σε ένα “iPhoneDevCamp” event με το όνομα PhoneGap. Κέρδισε μάλιστα το “People’s Choice Award” στο Web 2.0 Conference που διοργάνωσε η εταιρία O’Reilly Media το 2009. Το PhoneGap χρησιμοποιήθηκε για τη συγγραφή ενός μεγάλου πλήθους εφαρμογών, ενώ αποτέλεσε τη ραχοκοκαλιά ορισμένων mobile application πλατφόρμων όπως είναι το Monaca, το appMobi, το Convertigo, το ViziApps και το Worklight.

Στις 4 Οκτωβρίου του 2011 η Nitobi εξαγοράστηκε από την Adobe, η οποία διέθεσε το PhoneGap στην Apache Software Foundation για την υλοποίηση ενός νέου project το οποίο ξεκίνησε αρχικά ως Apache Callback και στη συνέχεια μετονομάστηκε σε Apache Cordova.

ΜΙΑ ΕΙΣΑΓΩΓΗ ΣΤΟ FRAMEWORK



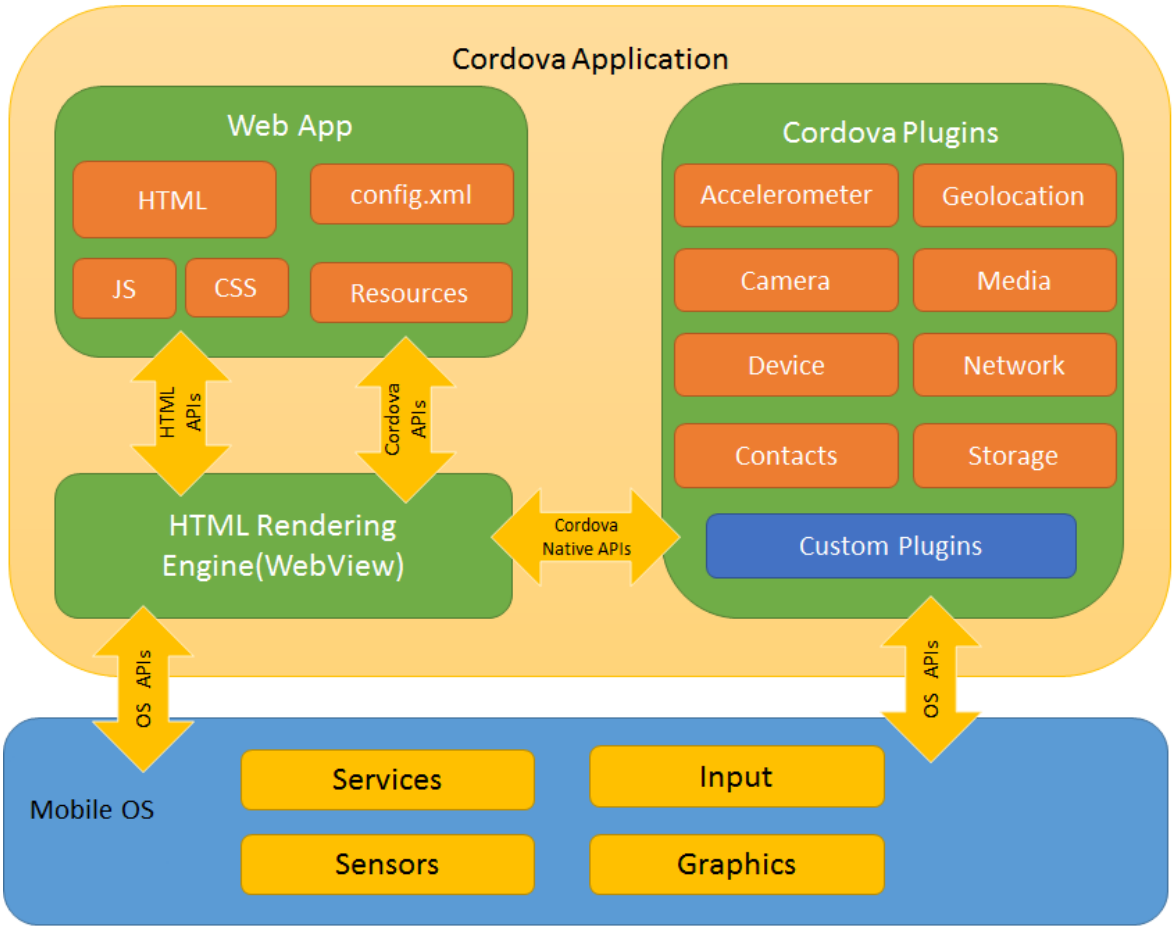
Εικόνα 13: Apache Cordova logo

Η συγγραφή της εφαρμογής ανάγνωσης των barcodes έγινε χρησιμοποιώντας το Cordova, ένα open-source Framework για τη δημιουργία εφαρμογών cross-platform σε κινητές συσκευές. Μερικές πλατφόρμες που υποστηρίζει το Cordova είναι το Android, το iOS και το Windows Phone.

Εικόνα 14: HTML, JavaScript και CSS logos

Η συγγραφή του κώδικα γίνεται χρησιμοποιώντας HTML5 για να δημιουργηθεί το front-end, CSS3 για τη διαμορφοποίηση του UI και JavaScript για το back-end κομμάτι της εφαρμογής.

Η ΛΕΙΤΟΥΡΓΕΙΑ ΤΟΥ CORDOVA



Εικόνα 15: Λειτουργικότητα του Cordova

Το Cordova αποτελείται από τα εξής στοιχεία:

- ***WebView***
  Το WebView κάνει το rendering του κώδικα που έχει γραφτεί σε HTML5, CSS3 και JavaScript. Σε ορισμένες πλατφόρμες μπορεί να αναμειχθεί με native κώδικα.

- ***Web App***

  Το Web App εμπεριέχει όλο τον κώδικα, τα αρχεία, τις εικόνες και όλα τα υπόλοιπα μέσα, ο συνδυασμός των οποίων διαμορφώνουν μια ιστοσελίδα. Συνήθως, το interface της ιστοσελίδας υλοποιείται σε ένα αρχείο το οποίο ονομάζεται index.html. Στο Web App υπάρχει και το αρχείο config.xml, το οποίο εμπεριέχει πληροφορίες για την εφαρμογή, όπως επίσης και παραμέτρους που ρυθμίζουν τον τρόπο λειτουργίας της.

- ***Plugins***
  Τα Plugins είναι ένα βασικό κομμάτι του Cordova, καθώς προσκομίζουν τις διεπαφές για την επικοινωνία με το native κομμάτι της εφαρμογής και

τα bindings με τα APIs της συσκευής. Μέσω των Plugins, το Cordova μπορεί και εκτελεί native κώδικα μέσω JavaScript. Το Cordova έχει ενσωματωμένο τα Core Plugins, τα οποία το επιτρέπουν να έχει πρόσβαση στις διάφορες δυνατότητες της συσκευής, όπως είναι η κάμερά της, η μπαταρία της κ.α. Επιπλέον υπάρχουν διάφορα Plugins φτιαγμένα από τρίτους, τα οποία επιτρέπουν διάφορες δυνατότητες. Το μειονέκτημα χρήσης αυτών είναι ότι πολλές φορές δεν υποστηρίζονται όλες οι πλατφόρμες.

Το Cordova διαθέτει δύο ειδών ***workflows***:

- ***Cross-platform workflow (CLI)***

  Το προτεινόμενο workflow για εφαρμογές που στοχεύουν πολλαπλές πλατφόρμες. Το CLI είναι ένα εργαλείο που επιτρέπει τη συγγραφή του κώδικα για πολλαπλές πλατφόρμες ταυτόχρονα. Αντιγράφει τα συστατικά του Web App σε διαφορετικούς υποφακέλους για κάθε πλατφόρμα, πραγματοποιεί τις απαραίτητες ρυθμίσεις για τη κάθε μία και δημιουργεί τα scripts που «χτίζουν» την εφαρμογή. Δίνει επίσης το περιβάλλον για την προσθήκη των Plugins, ανεξαρτήτως της πλατφόρμας.

- ***Platform Centered Workflow***

  Η χρήση αυτού του workflow γίνεται όταν η εφαρμογή απευθύνεται σε μια πλατφόρμα και πρέπει να τροποποιηθεί σε χαμηλότερο επίπεδο. Ένα παράδειγμα χρήσης είναι εάν χρειάζεται η προσθήκη native controls σε συνδυασμό με το κομμάτι του Cordova στην εφαρμογή. Το workflow αυτό βασίζεται στη λειτουργεία χαμηλού επιπέδου shell scripts, τροποποιημένα για κάθε πλατφόρμα ξεχωριστά, όπως επίσης και ένα utility, το Plugman για την προσθήκη των Plugins. Ενώ μπορούν να γραφούν εφαρμογές για πολλαπλές πλατφόρμες με αυτό το workflow, δεν συνιστάται η χρήση του, καθώς λείπουν τα εργαλεία που προσφέρει το CLI. Αυτό σημαίνει ότι για κάθε πλατφόρμα θα χρειαστεί να γίνει χειροκίνητη τροποποίηση των ρυθμίσεων και η ενσωμάτωση των Plugins ξεχωριστά.

Ενώ είναι δυνατή η αλλαγή από CLI σε Platform Centered Workflow, το ανάποδο δεν είναι εφικτό. Αυτό συμβαίνει επειδή στο CLI, η εφαρμογή χρησιμοποιεί τον ενιαίο Cross-Platform πηγαίο κώδικα για τη δημιουργία του κώδικα που θα εκτελέσει η κάθε πλατφόρμα. Μπορεί έτσι αυτός να περιοριστεί σε μόνο μία από αυτές για τη μετατροπή σε Platform Centered Workflow. Το ανάποδο όμως είναι αδύνατο, καθώς στο Platform Centered Workflow υπάρχουν πολλές ιδιαιτερότητες της πλατφόρμας που υλοποιεί για να δημιουργηθεί ένας ενιαίος Cross-Platform πηγαίος κώδικας.

ΧΡΗΣΗ ΤΟΥ CORDOVA

Για τη χρήση του Cordova πρέπει πρώτα να γίνει εγκατάσταση το NodeJS. Μέσω αυτού μπορεί να δοθεί η εντολή “npm install -g cordova”. Αυτό προκαλεί τη λήψη του Cordova στον υπολογιστή.

Το Cordova μπορεί να χρησιμοποιηθεί μέσω του CMD στα Windows και μέσω Terminal σε iOS και Linux.

```
Synopsis

    cordova command [options]

Global Commands
    create ............................. Create a project
    help ............................... Get help for a command
    telemetry .......................... Turn telemetry collection on or off
    config ............................. Set, get, delete, edit, and list global cordova options

Project Commands
    info ............................... Generate project information
    requirements ....................... Checks and print out all the requirements
                                         for platforms specified

    platform ........................... Manage project platforms
    plugin ............................. Manage project plugins

    prepare ............................ Copy files into platform(s) for building
    compile ............................ Build platform(s)
    clean .............................. Cleanup project from build artifacts

    run ................................ Run project
                                         (including prepare && compile)
    serve .............................. Run project with a local webserver
                                         (including prepare)

Learn more about command options using 'cordova help <command>'

Aliases
    build -> cordova prepare && cordova compile
    emulate -> cordova run --emulator

Options
    -v, --version ...................... prints out this utility's version
    -d, --verbose ...................... debug mode produces verbose log output for all activity,
    --no-update-notifier ............... disables check for CLI updates
    --nohooks .......................... suppress executing hooks
                                        (taking RegExp hook patterns as parameters)
```

Εικόνα 16: Εντολές του Cordova

Μέσα από τις υπάρχουσες εντολές, είναι δυνατή η δημιουργία ενός project με ή χωρίς τη χρήση προεπιλεγμένων templates (create), το χτίσιμο (build) ή το τρέξιμο (run) της εφαρμογής, καθώς και άλλες δυνατότητες που θα επεξηγηθούν πιο αναλυτικά στη συνέχεια.

```
Synopsis

    cordova create <PATH> [ID [NAME [CONFIG]]] [options]

Create a Cordova project

    PATH ......................... Where to create the project
    ID ........................... Reverse-domain-style package name - used in <widget id>
    NAME ......................... Human readable name
    CONFIG ....................... json string whose key/values will be included in
                                   [PATH]/.cordova/config.json

Options

    --template=<PATH|NPM PACKAGE|GIT URL> ... use a custom template located locally, in NPM, or GitHub.
    --copy-from|src=<PATH> ................... deprecated, use --template instead.
    --link-to=<PATH> ......................... symlink to custom www assets without creating a copy.

Example
    cordova create myapp com.mycompany.myteam.myapp MyApp
```

Εικόνα 17: Εντολές Cordova create

Για τη δημιουργία ενός νέου project, όπως φαίνεται στην παραπάνω εικόνα, πρέπει να ορίσουμε:

- Τη διαδρομή στην οποία θα αποθηκεύσουμε το project.
- Το ID της εφαρμογής. Συνήθως παίρνει τη μορφή: **“domain”.“publisher”. “applicationName”.**
- Το όνομα του project.

Προαιρετικά μπορούμε να προσθέσουμε:

- Το config, στο οποίο περνάμε μια συμβολοσειρά ένα JSON αρχείο, στο οποίο μπορούμε να συμπεριλάβουμε μεταβλητές και τις τιμές τους για την εισαγωγή τους στο config.json
- Άλλες επιλογές, όπως ένα έτοιμο template που θα χρησιμοποιηθεί για τη δημιουργία του project.

```
Examples
    cordova platform add android ios
    cordova platform add android@^5.0.0
    cordova platform add https://github.com/myfork/cordova-android.git#4.0.0
    cordova platform add ../android
    cordova platform add ../cordova-android.tgz
    cordova platform rm android --nosave
    cordova platform ls
```

Εικόνα 18: Εντολές Cordova platform

Μετά τη δημιουργία του project, θα πρέπει να κάνουμε προσθήκη (add) της κάθε πλατφόρμας, στην οποία θα υλοποιηθεί το project. Μπορούμε να ορίσουμε μία ή περισσότερες πλατφόρμες και συγκεκριμένες εκδόσεις για αυτές, χωρίς αυτό να είναι υποχρεωτικό.

Εδώ αξίζει να σημειωθεί ότι για να συμπεριληφθεί μια πλατφόρμα, αυτή πρέπει να υπάρχει στον υπολογιστή. Διαφορετικά μπορούμε να ορίσουμε τη πλατφόρμα μέσω GitHub.

Τέλος, υπάρχει η δυνατότητα αφαίρεσης μιας ή περισσοτέρων πλατφορμών (rm), ενώ μπορούμε πάντα να ζητήσουμε να εμφανιστούν όλες οι πλατφόρμες που έχουν συμπεριληφθεί μέσα στο project (ls)

```
Aliases
    plugins -> plugin
    rm -> remove
    ls -> list

Examples
    cordova plugin add cordova-plugin-camera cordova-plugin-file --nosave --searchpath ~/plugins
    cordova plugin add cordova-plugin-camera@^2.0.0 --nosave
    cordova plugin add https://github.com/myfork/cordova-plugin-camera.git#2.1.0 --nosave
    cordova plugin add ../cordova-plugin-camera --nosave
    cordova plugin add ../cordova-plugin-camera.tgz --nosave
    cordova plugin rm camera --nosave
    cordova plugin ls
```

Εικόνα 19: Εντολές Cordova plugin

Με το Cordova plugin μπορούμε να τροποποιήσουμε τα plugins του project. Υπάρχει η δυνατότητα προσθήκης των plugins (add) μέσω του internet ή από αρχεία αποθηκευμένα τοπικά στον υπολογιστή, αφαίρεσης ήδη υπαρχόντων plugins (rm) και η εμφάνιση των εγκατεστημένων plugins στο project (ls)

```
      MODE:        --list|--debug|--release
      BUILDOPTS:   --noprepare --nobuild
      TARGET:      DEVICECLASS|--target=FOO
      PLATS:       PLATFORM [...]
      BUILDCONFIG: --buildConfig=CONFIGFILE
      POPTS:       platformopts
      DEVICECLASS: --device|--emulator

Deploys app on specified platform devices / emulators

    --list ............................. Lists available targets
                                         Will display both device and emulator
                                         unless DEVICECLASS option is provided

    --debug ............................ Deploy a debug build
    --release .......................... Deploy a release build

    --noprepare ........................ Don't prepare
    --nobuild .......................... Don't build

    --device ........................... Deploy to a device
    --emulator ......................... Deploy to an emulator
    --target ........................... Deploy to a specific target

    --buildConfig....................... Use the specified build configuration
                                         instead of default build.json

    --browserify ....................... Compile plugin JS at build time using
                                         browserify instead of runtime.
```

Εικόνα 20: Εντολές Cordova run

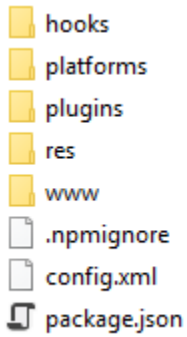
Τέλος, υπάρχουν δύο εντολές με πολύ παρόμοια σύνταξη, η build και η run. Η build προετοιμάζει και χτίζει το project, δίχως να το τρέξει. Εδώ φαίνεται αν

υπάρχει κάποιο πρόβλημα με κάποιο plugin, με κάποιο platform ή γενικότερα κάποιο πρόβλημα που αποτρέπει τη δημιουργία και εκτέλεση της εφαρμογής.

Η run είναι υπεύθυνη για το τρέξιμο της εφαρμογής με το αποτέλεσμα που παρήγαγε η build. Εάν δεν έχει δοθεί η εντολή build ή εάν έχει αλλάξει ο κώδικας από τη τελευταία φορά που δόθηκε εκείνη, το run θα εκτελέσει πρώτα τη build, εκτός αν οριστεί συγκεκριμένα να προσπεράσει αυτό το βήμα.

Και στις δύο περιπτώσεις μπορεί οριστεί η εκτέλεση της εφαρμογής σε debug ή σε release. Το debug χρησιμοποιείται από προγραμματιστές για να μπορέσουν να παρατηρήσουν τις εντολές που εκτελούνται σε συγκεκριμένα σημεία, τις τιμές των μεταβλητών που έχουν σε αυτά και για να εντοπίσουν τα λάθη που εμφανίζονται στην εφαρμογή. Το release χρησιμοποιείται όταν η εφαρμογή είναι έτοιμη για τη διάθεσή της στους χρήστες, καθώς τρέχει πιο γρήγορα και δεν εμφανίζει μηνύματα σφαλμάτων που δεν τους ενδιαφέρουν.

Επιπλέον δίνεται η δυνατότητα επιλογής της συσκευής στην οποία θα γίνει το build και το run. Οι επιλογές εμπεριέχουν τη χρήση emulator, ανάλογα με τις πλατφόρμες στις οποίες τρέχει η εφαρμογή, τη χρήση κάποιας συσκευής και την επιλογή μεταξύ αυτών, σε περίπτωση που υπάρχουν πάνω από μία επιλογές.



Εικόνα 21: Αρχική μορφή project στο Cordova

Με τη δημιουργία ενός νέου project, δημιουργείται αυτόματα ένα σύνολο φακέλων και αρχείων. Τα πιο σημαντικά από αυτά είναι:

- Ο φάκελος platforms, στον οποίο θα μπουν οι διαφορετικές πλατφόρμες στις οποίες θα τρέξει η εφαρμογή. Όταν δημιουργείται το project, ο φάκελος αυτός είναι κενός.
- Ο φάκελος plugins, στον οποίο θα μπουν τα διαφορετικά plugins που θα περιέχει η εφαρμογή. Όταν δημιουργείται το project, ο φάκελος αυτός είναι κενός.
- Ο φάκελος res, στον οποίο θα μπουν οι πόροι (resources) της εφαρμογής. Πιο συγκεκριμένα, εδώ αποθηκεύονται τα εικονίδια της εφαρμογής όπως επίσης και τα splash screen που θα εμφανιστούν κατά το άνοιγμα της εφαρμογής. Η κατηγοριοποίηση γίνεται σε φακέλους για τις διαφορετικές πλατφόρμες.

- Ο φάκελος www, στον οποίο μπαίνουν όλα τα αρχεία HTML, JavaScript και CSS. Όταν δημιουργείται το project, εδώ υπάρχουν 3 αρχεία: Το index.html, το index.js και το index.css. Είναι τα αρχεία που θα ανοίξουν όταν θα τρέξει για πρώτη φορά η εφαρμογή.
- Το config.xml, το οποίο εμπεριέχει όλες τις ρυθμίσεις της εφαρμογής. Εδώ ορίζεται και το HTML αρχείο που θα τρέξει κατά την εκκίνηση της εφαρμογής. Επιπλέον, καθώς εισάγουμε πλατφόρμες και plugins στο project, το αρχείο αυτό θα ενημερώνεται με το όνομα αυτών, καθώς επίσης και με την έκδοσή τους.
- Το package.json, το οποίο περιέχει ορισμένα στοιχεία της εφαρμογής. Σε αυτά περιλαμβάνονται το όνομα του πακέτου της εφαρμογής και το προβαλλόμενο όνομά της όπως επίσης και ο αριθμός της έκδοσής της και ο συντάκτης της.

```
<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="js/main.js"></script>
```

Εικόνα 22: Δήλωση του Cordova σε HTML

Στο HTML αρχείο που δημιουργήθηκε το Cordova ορίζεται όπως ένα οποιοδήποτε άλλο JavaScript αρχείο. Δηλώνεται ως ένα script ενός JavaScript αρχείου και συνήθως δηλώνεται πριν δηλωθεί το JavaScript αρχείο με τις εντολές που θα εκτελέσει η εφαρμογή.

ΕΠΙΛΟΓΟΣ

Το Cordova είναι ένα open-source framework που επιτρέπει την δημιουργία εφαρμογών για συσκευές διαφόρων mobile λειτουργικών συστημάτων χρησιμοποιώντας ένα κοινό κώδικα. Αυτό επιτυγχάνεται με την δημιουργία ενός WebView μέσα στη mobile συσκευή και τον συνδυασμό native κώδικα και διαφόρων Plugins σε αυτό.

Για τη διαχείριση του Cordova απαιτείται η χρήση του CMD ή του terminal. Μέσα από αυτά γίνονται οι περισσότερες ενέργειες του Cordova, όπως η δημιουργία ενός project, το χτίσιμο και τρέξιμό του και η διαχείριση των διαφόρων plugin του.

ΚΕΦΑΛΑΙΟ 3 – JQUERY MOBILE

ΕΙΣΑΓΩΓΗ

Η συγγραφή των δύο σελίδων της εφαρμογής γράφηκε σε HTML και JavaScript κώδικα. Η χρήση μόνο αυτών των δύο όμως δεν είναι αρκετή για τη δημιουργία των διαφορετικών σελίδων, την πλοήγηση μεταξύ αυτών και διαφόρων άλλων δυνατοτήτων που θα επεξηγηθούν σε επόμενα κεφάλαια. Για την υλοποίηση αυτών έγινε χρήση του jQuery Mobile.

ΠΕΡΙΓΡΑΦΗ

Εικόνα 23: jQuery Mobile logo

Το jQuery Mobile είναι μια open-source βιβλιοθήκη του JavaScript, το οποίο επιτρέπει τη δημιουργία των interfaces των σελίδων και τη τροποποίησή τους, έτσι ώστε να είναι λειτουργικές και προσβάσιμες σε κινητές και επιτραπέζιες συσκευές. Το jQuery Mobile αποτελεί επέκταση του jQuery, το οποίο χρησιμοποιείται πλέον στο back-end πολλών ιστοσελίδων.

Μέσω του jQuery Mobile μπορεί να γίνει πλοήγηση με AJAX με εναλλαγές των σελίδων, διαχείριση events για οθόνες αφής και πολλά ακόμη πράγματα. Άλλα πλεονεκτήματα χρήσης του, περιλαμβάνουν το χαμηλό του όγκο και την ευελιξία στη τροποποίηση των οπτικών του ιδιοτήτων.

Για την εισαγωγή του jQuery Mobile στην εφαρμογή χρειάζεται να γίνει πρώτα η δήλωσή του στο head του HTML αρχείου. Το jQuery Mobile μπορεί είτε να κατέβει σε αρχεία και να γίνει η εισαγωγή του μαζί με τα υπόλοιπα στοιχεία μέσα στο project είτε μπορεί να γίνει εισαγωγή με αναφορά στη CDN υπηρεσία. Σημειώνεται ότι η εφαρμογή υλοποιεί τη δεύτερη μέθοδο, καθώς είναι απαραίτητη η σύνδεση της συσκευής με το internet καθόλη τη χρήση της.

ΣΥΝΤΑΞΗ

Η σύνταξη του jQuery Mobile είναι παρόμοια με αυτή του jQuery και γράφεται ως εξής:

$(*element*).*action*(args)

Στο παραπάνω παράδειγμα δηλώνονται τα εξής:

- Το $ είναι το σύμβολο που δηλώνει την έναρξη χρήσης του jQuery Mobile.
- Το element περιγράφει το στοιχείο το οποίο θα επηρεάσει το jQuery Mobile. Αυτό το στοιχείο αναγνωρίζεται, όπως στη JavaScript, μέσω “id” ή “class”.

- Το action ορίζει την ενέργεια που θα προκαλέσει το jQuery Mobile επάνω στο αντικείμενο. Το action μπορεί να είναι είτε μία function, όπως για παράδειγμα “hide()” που θα κρύψει το επιλεγμένο element, ή ένα event, όπως για παράδειγμα το “click()” που ορίζει το γεγονός του “click” πάνω σε κάποιο element.
- Τα args χρησιμοποιούνται όταν το action που προηγήθηκε απαιτεί να υπάρχουν κάποια ορίσματα για τη πραγματοποίηση της ενέργειας

Για παράδειγμα εάν θέλουμε να ορίσουμε τις ενέργειες που θα πραγματοποιήσει το element με ID “element”, όταν πραγματοποιηθεί “click” επάνω του, θα γράψουμε τον εξής κώδικα:

```
$("#element").click(function(){
  // Lines of code
});
```

Όπως φαίνεται στο παραπάνω παράδειγμα, το όρισμα μπορεί να είναι ακόμα και function ορισμένο ως argument, δίχως να χρειάζεται να του δοθεί κάποιο όνομα.

PAGE TRANSITION

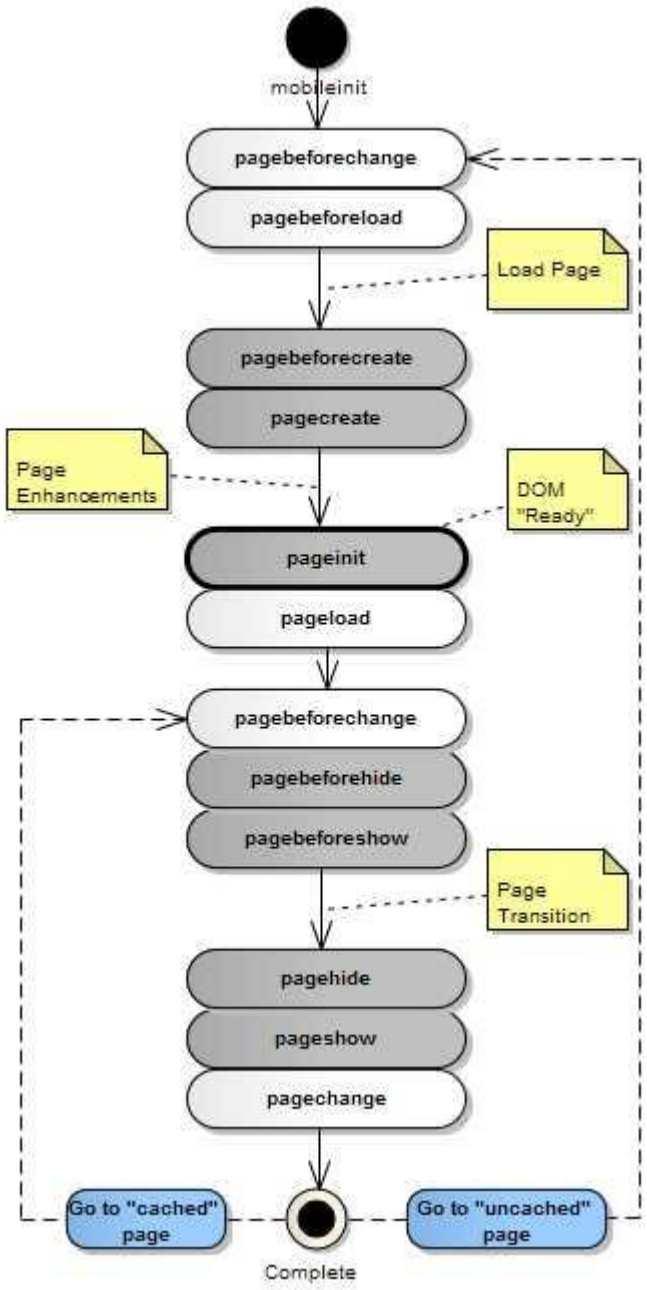
Μια από τις σημαντικότερες προσθήκες που προσφέρει το jQuery Mobile έναντι του jQuery είναι η εναλλαγή μεταξύ των διαφορετικών σελίδων και διαχείριση των events κατά τη φόρτωσή τους στην εφαρμογή. Δίνεται δηλαδή η δυνατότητα εκτέλεσης κώδικα πριν ή μετά από οποιαδήποτε αλλαγή στις σελίδες.

Η εναλλαγή των σελίδων γίνεται με την εξής εντολή:

```
$.mobile.changePage("#target_page");
```

Με την εντολή αυτή δηλώνεται ότι θα υλοποιηθεί η αλλαγή της σελίδας με τη χρήση του jQuery Mobile στη σελίδα που έχει id “target_page”.

Στη συνέχεια ας εξετάσουμε τα events που συμβαίνουν κατά την αλλαγή.



Εικόνα 24: Page transition events (< version 1.4)

Όπως φαίνεται στο παραπάνω state event diagram, κατά την αλλαγή δύο σελίδων γίνεται εκτέλεση μιας πληθώρας από events, τόσο κατά την αλλαγή όσο και κατά την εμφάνιση και εξαφάνιση της κάθε σελίδας.

Κατά την εκκίνηση της εφαρμογής ή κατά την πλοήγηση σε μια σελίδα που δε βρίσκεται στη cache, γίνεται φόρτωση σελίδας στο DOM. Στη συνέχεια θα κρυφτεί η παλιά σελίδα, θα εκτελεστεί το transition και θα εμφανιστεί η νέα σελίδα.

Από τα events αυτά ξεχωρίζουν τα pagebeforeshow, pagebeforehide, pageshow και pagehide. Η εκτέλεση αυτών των events γίνεται πριν και μετά πριν και μετά την εμφάνιση και εξαφάνιση της κάθε οθόνης.

Με τη χρήση του ***pagebeforeshow*** μπορεί να γίνει κάποια τροποποίηση στη νέα σελίδα προτού αυτή εμφανιστεί, επιτρέποντας τη δυναμική διαχείριση των στοιχείων της.

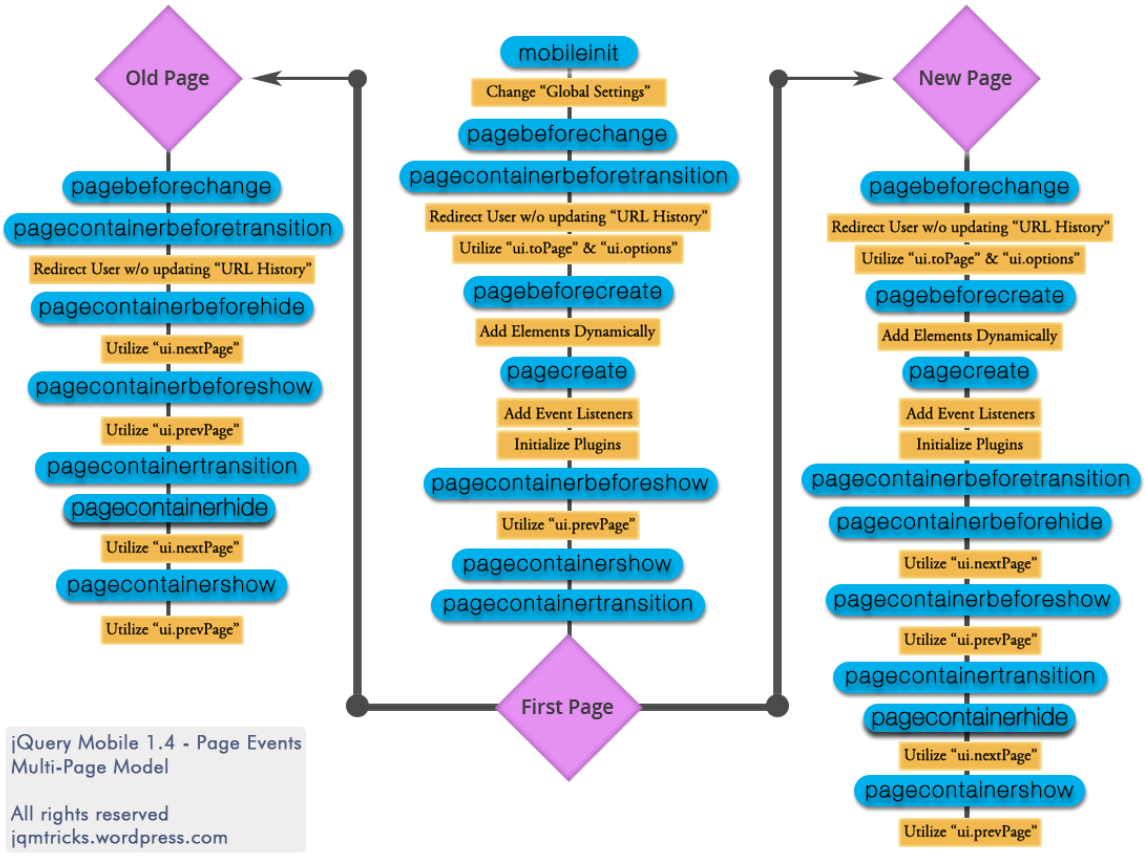
Στο ***pageshow***, μπορεί να γίνουν αλλαγές ενώ καθώς η σελίδα έχει εμφανιστεί, επιτρέπει στο χρήστη να παρατηρήσει τις αλλαγές που γίνονται σε αυτή.

Στα ***pagebeforehide*** και ***pagehide*** υπάρχει η δυνατότητα διαχείρισης της παλιάς σελίδας προτού και αφού κρυφτεί. Η χρήση αυτών των δύο είναι ιδιαίτερη. Ένα παράδειγμα χρήσης αυτών, είναι η αλλαγή στοιχείων μιας παλιάς σελίδας, στην οποία ο χρήστης μπορεί να πλοηγηθεί ξανά πατώντας το back button της συσκευής του.

Η σύνταξη των event αυτών είναι η εξής:

```
$(page_identification).on(event, function(event){
  // Lines of code
});
```

Παρόμοια με τις προηγούμενες εντολές, υπάρχει ένα είδος ταυτοποίησης για τη σελίδα στην οποία θα εκτελεστεί το event. Επιπλέον ορίζεται πάλι και κάποιο function για τη διαχείριση του event, με τη διαφορά ότι θα περαστεί και το event ως όρισμα σε αυτό. Για τη σύνδεση των προαναφερόμενων δύο, γίνεται η χρήση του “on”. Το όνομα του event εισάγεται ως όρισμα σε μορφή string, μαζί με το function που θα υλοποιηθεί.



Εικόνα 25: New age transition events

Τέλος, αξίζει να σημειωθεί ότι τα events κατά την αλλαγή δύο σελίδων έχουν αλλάξει στην έκδοση 1.4, με πολλά να έχουν προστεθεί και τα υπόλοιπα να έχουν αλλάξει όνομα. Παρά το γεγονός αυτό, τα προαναφερόμενα events υποστηρίζονται σε όλες τις εκδόσεις που κυκλοφορήσαν πριν την 1.5.

ΕΠΙΛΟΓΟΣ

Το jQuery Mobile, που βασίζεται στο jQuery, προσφέρει στην εφαρμογή δυνατότητες που δεν είναι εφικτές μόνο με τη χρήση JavaScript. Στοχεύει στην υλοποίηση "***περισσότερων πραγμάτων με τη συγγραφή λιγότερου κώδικα***". Ιδιαίτερα στη περίπτωση του jQuery Mobile, υπάρχει η δυνατότητα διαχείρισης διαφόρων events κατά το navigation.

ΚΕΦΑΛΑΙΟ 3 - AJAX

ΕΙΣΑΓΩΓΗ

Η συγγραφή κώδικα σε HTML έχει ένα μεγάλο μειονέκτημα: για την προσθήκη, την κατάργηση και την τροποποίηση στοιχείων σε μια σελίδα, απαιτείται η ανανέωσή της. Αυτό μπορεί να προκαλέσει μια δυσάρεστη εμπειρία για το χρήστη, καθώς οι native εφαρμογές δεν έχουν τέτοιο περιορισμό.

Επιπλέον οι διαφορετικές σελίδες της εφαρμογής είναι γραμμένες στο ίδιο HTML αρχείο. Η κάθε μία από αυτές πρέπει να μπορεί να τροποποιεί και να διαμορφώνει το χώρο της οθόνης, ώστε να δίνεται η ψευδαίσθηση μιας native εφαρμογής. Συνεπώς η χρήση μόνο της HTML δεν αρκεί στη δημιουργία μιας εφαρμογής.

Για την αποφυγή των προαναφερόμενων προβλημάτων, είναι αναγκαίο να συμπεριληφθεί AJAX στην εφαρμογή.

ΠΕΡΙΓΡΑΦΗ

Εικόνα 26: AJAX logo

AJAX είναι συντομογραφία του ονόματος **Asynchronous JavaScript And XML.** Όπως υποδηλώνει το όνομά του, είναι ο συνδυασμός JavaScript και

HTML DOM μαζί με ένα XMLHttpRequest αντικείμενο, ενσωματωμένο στο browser.

Ο συνδυασμός αυτών των δύο επιτρέπουν την ασύγχρονη ανταλλαγή δεδομένων μεταξύ Web Page και Web Server. Αυτό επιτρέπει την ενημέρωση στοιχείων της σελίδας, χωρίς να απαιτείται η ανανέωσή της.

Το AJAX είναι ενσωματωμένο μέσα στο jQuery Mobile και χρησιμοποιείται μέσα από αυτό. Η χρήση του επιτρέπει την εναλλαγή των διαφορετικών σελίδων μέσα στο ίδιο HTML αρχείο, χωρίς να απαιτείται η ανανέωση αυτού για κάθε αλλαγή. Η χρήση του είναι μεγάλης σημασίας για τη συγγραφή εφαρμογών, καθώς εξαλείφεται η καθυστέρηση μεταξύ των αλλαγών των σελίδων και δημιουργείται η αίσθηση μιας native πλοήγησης μεταξύ των σελίδων.

ΣΥΝΤΑΞΗ ΑΙΤΗΜΑΤΟΣ AJAX

Ένα AJAX αίτημα έχει την εξής σύνταξη:

```
var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var response = this.responseText;
       //Διαχείριση του “response”
    }
  };
  xhttp.open (method, URL, isAsync, username, password));
  xhttp.send();
```

Σε αυτές τις σειρές κώδικα, δημιουργείται ένα νέο XMLHttpRequest (xhttp), το οποίο θα σταλεί κάπου προκειμένου να πάρει πρόσβαση σε ένα αρχείο που βρίσκεται στο δοσμένο URL.

Το xhttp μπορεί να σταλεί με δύο μεθόδους, είτε GET είτε POST. Ενώ η GET είναι πιο γρήγορη, πιο απλή και μπορεί να χρησιμοποιηθεί στις περισσότερες περιπτώσεις, η POST χρησιμοποιείται όταν:

- Δε μπορεί να χρησιμοποιηθεί αρχείο cache.

- Γίνεται αποστολή μεγάλου όγκου αρχείων, καθώς η GET επιτρέπει συνήθως 2kb – 8kb, ανάλογα με τους περιορισμούς του παραλήπτη, ενώ η POST έχει αρκετά μεγαλύτερο επιτρεπτό όριο, συνήθως 2gb.

- Υπάρχουν στοιχεία που έχει εισάγει ο χρήστης. Καθώς το GET περνάει τα δεδομένα που έχει μέσω του URL, τα στοιχεία αυτά είναι εμφανή σε αυτό. Συνεπώς απόρρητα δεδομένα όπως το password είναι εμφανή. Επιπλέον πρόβλημα δημιουργεί επίσης ότι, καθώς ο χρήστης μπορεί να εισάγει διάφορους χαρακτήρες, υπάρχει πιθανότητα κάποιοι από αυτούς να επηρεάσουν το URL, όπως τα “<”, “>”, “#”, “%” και άλλα.

Το URL μπορεί να δείχνει ένα οποιοδήποτε είδος αρχείου (π.χ. “.txt”). Συνήθως επιλέγονται “.php” και “.asp”, καθώς έχουν τη δυνατότητα να εκτελέσουν υπολογισμούς και πράξεις προτού επιστρέψουν κάποιο αποτέλεσμα.

Το isAsync ορίζει αν το JavaScript αρχείο που στέλνει το Request θα περιμένει την απάντηση του server προτού συνεχίσει την εκτέλεσή του. Ορίζει δηλαδή αν το Request γίνεται ασύγχρονα ή όχι. Συνήθως επιλέγεται η ασύγχρονη αποστολή, καθώς η αργή ανταπόκριση του παραλήπτη μπορεί να προκαλέσει το πάγωμα ή ακόμα και το σταμάτημα της εφαρμογής.

Τα τελευταία δύο πεδία είναι προαιρετικά και χρησιμοποιούνται όταν ο παραλήπτης απαιτεί κάποια ταυτοποίηση από πλευράς του χρήστη για την ταυτότητά του, μέσω username και password.

Μετά την αρχικοποίηση ενός XMLHttpRequest και πριν τεθούν οι προαναφερόμενες ρυθμίσεις, τίθεται το function που θα εκτελεστεί όταν αλλάξει το state και το status του αιτήματος (onreadystatechanged). Καθώς υπάρχουν αρκετές αλλαγές σε αυτές τις δύο μεταβλητές και θα γίνουν πολλές κλήσεις της function, εισάγεται συνήθως στην αρχή μια εντολή “if”. Σε αυτή ελέγχεται αν το state γίνει 4 και το status είναι 200. Αυτά σημαίνουν αντιστοίχως ότι έχει ληφθεί η απάντηση στο αίτημα που στάλθηκε και ότι η κατάσταση είναι “OK”.

Η απάντηση έρχεται σε μορφή συμβολοσειράς (string) μέσω της μεταβλητής “responseText” που εμπεριέχεται μέσα στο Request. Έτσι διαχειριζόμαστε την απάντηση στο Request.

AJAX ΚΑΙ JQUERY

Όπως προαναφέρθηκε, το AJAX είναι ενσωματωμένο στο jQuery Mobile. Η δημιουργία ενός Request με GET ή POST, η οποία γίνεται παρόμοια στο jQuery, μπορεί να γίνει ως εξής:

```
$.ajax({
      type: ("GET"/"POST"),
      url: URL,
      data: ({
                  var1: value1,
                  var2: value2
            }),
      success: successFunction
});
```

Στο είδος (type) της μεθόδου που θα εκτελέσει το Request, εισάγεται είτε GET είτε POST. Στο URL προσδιορίζεται το αρχείο που θα κληθεί. Στα data ορίζονται τα ορίσματα που θα περαστούν. Στο success ορίζεται η function που θα κληθεί μόλις ληφθεί η απάντηση του παραλήπτη.

Πρέπει να σημειωθεί ότι μπορούν να συμπληρωθούν πολλά παραπάνω ορίσματα όπως το datatype, το οποίο δηλώνει τον τύπο της απάντησης που θα επιστρέψει ο server και το crossDomain για τη δήλωση της αποστολής του Request σε διαφορετικό domain.

ΠΛΟΗΓΗΣΗ ΜΕ AJAX

Συγκεκριμένα για το jQuery Mobile, το AJAX χρησιμοποιείται για την πλοήγηση του χρήστη σε διάφορες “υποσέλιδες» που υπάρχουν μέσα σε ένα HTML αρχείο.

```
<div data-role="page" id="barcode_page">
```

Εικόνα 27: Page declaration

Οι “υποσέλιδες” αυτές είναι μια σειρά από div, στα οποία το όρισμα ‘data-role’ έχει την τιμή ”page”’. Το jQuery Mobile, κατά το άνοιγμα του HTML αρχείου, θα εμφανίσει αρχικά το div που έχει αυτή την ένδειξη και βρίσκεται πιο πάνω από τα υπόλοιπα.

```
$.mobile.changePage("#barcode_page");
```

Εικόνα 28: Εντολή transition σε page

Το AJAX που εμπεριέχεται στο jQuery Mobile είναι υπεύθυνο για την πλοήγηση της εφαρμογής από το ένα div στο επόμενο. Με την εντολή που εμφανίζεται παραπάνω γίνεται ένα AJAX αίτημα, το οποίο θα ανανεώσει τη σελίδα, προκαλώντας την αλλαγή της τρέχουσας σελίδας να φορτώσει.

Πρέπει να επισημανθεί ότι εκτός της σελίδας στην οποία θα γίνει πλοήγηση, μπορούν να οριστούν και διάφορες επιλογές. Δύο αξιοσημείωτες επιλογές αποτελούν τη ***transition*** και τη ***changeHash***. Η transition ορίζει το animation με το οποίο θα γίνει η αλλαγή των δύο σελίδων, ενώ η changeHash μπορεί να αντικαταστήσει τη παλιά σελίδα από τη νέα στο browser history. Αυτό θα προκαλέσει την παλιά σελίδα να μη φορτώσει, όταν πατηθεί το back button της συσκευής.

ΕΠΙΛΟΓΟΣ

Η χρήση της HTML επιφέρει περιορισμούς στην τροποποίηση των στοιχείων που εμφανίζονται στην οθόνη. Με τη χρήση του AJAX, που είναι ενσωματωμένο μέσα στο jQuery Mobile, επιτυγχάνεται η καλύτερη πλοήγηση των στοιχείων μέσα στη σελίδα, δίχως να χρειάζεται η ανανέωση της για κάθε αλλαγή. Έτσι, μέσω του AJAX δημιουργείται μια περισσότερο ευχάριστη πλοήγηση για το χρήστη.

ΚΕΦΑΛΑΙΟ 4 - JSONP

ΕΙΣΑΓΩΓΗ

Ένα πρόβλημα που προκύπτει από την ανταλλαγή των αντικειμένων JSON εφαρμογής – server είναι ότι ο server βρίσκεται σε διαφορετικό domain από ότι η εφαρμογή. Αυτό προκαλεί πρόβλημα, καθώς η ανταλλαγή των αντικειμένων JSON και άλλων αρχείων δεν είναι εφικτή.

Τα μόνα δεδομένα που επιτρέπεται να περάσουν σε διαφορετικό domain είναι τα scripts. Βασιζόμενοι σε αυτό, μπορούμε να δεχτούμε δεδομένα τύπου JSON από το server τα οποία εμπεριέχονται μέσα σε ένα JSONP.

ΠΕΡΙΓΡΑΦΗ

Εικόνα 29: JSONP logo

Τα JSONP, ή αλλιώς JSON with padding, είναι μία μέθοδος αποστολής των JSON δεδομένων προσπερνώντας τον προαναφερόμενο περιορισμό. Η μέθοδος με την οποία γίνεται αυτό, είναι η αποστολή των δεδομένων μέσω scripts και όχι μέσω αντικειμένων τύπου XMLHttpRequest, καθώς τα scripts έχουν την δυνατότητα να περάσουν από το ένα domain στο άλλο, δίχως την ανάγκη ταυτοποίησης του client.

ΣΥΝΤΑΞΗ ΤΟΥ ΑΙΤΗΜΑΤΟΣ JSONP

Τα αιτήματα JSONP γίνονται με τη χρήση ενός αιτήματος AJAX. Υπενθυμίζεται ότι το AJAX βρίσκεται ενσωματωμένο μέσα στο jQuery Mobile.

```
$.ajax({
  type: "GET",
  url: url_string,
  crossDomain: true,
  data: {
    barCode: document.getElementById("bc").value,
    ekd: ekd_var,
    gate: gate_var,
    name: name_var
  },
  dataType: 'jsonp'
});
```

Εικόνα 30: AJAX Request για JSONP

Το αίτημα πρέπει να είναι “GET”, καθώς το JSONP αίτημα δεν μπορεί να είναι “POST”. Πρέπει επίσης να ορίσουμε μια διεύθυνση URL στην οποία θα σταλεί το αίτημα και να θέσουμε στη μεταβλητή που ορίζει ότι το αίτημα θα σταλεί σε άλλο domain (crossDomain), τιμή true. Το πεδίο που ορίζει τον τύπο του μηνύματος που θα επιστρέψει ο server (dataType) θέτεται “jsonp”. Εδώ υπάρχει προαιρετικά η δυνατότητα να περάσουμε επιπλέον δεδομένα μέσα στο αίτημα (data).

```
callback({"json_var1" : "value1", "json_var2" : "value2"})
```

Εικόνα 31: Η απάντηση του server

Η απάντηση που επιστρέφει ο server είναι στη πραγματικότητα η κλήση ενός function, το οποίο εμπεριέχει ως όρισμα ένα JSON.

```
function callback (json)
{
  //Handle "json"
}
```

Εικόνα 32: Δήλωση του callback σε JavaScript

Στο JavaScript αρχείο, γνωρίζοντας το όνομα του function που θα κληθεί, έχει αρχικοποιηθεί ένα function με το ίδιο όνομα. Έτσι, μόλις φτάσει η απάντηση του server, θα εκτελεστούν οι εντολές στο εσωτερικό της σαν να είχαν κληθεί από το JavaScript αρχείο. Έτσι, στο παράδειγμα που εμφανίζεται παραπάνω, θα εκτελεστεί το function callback και το JSON είναι πλέον προσβάσιμο μέσω αυτού.

ΕΠΙΛΟΓΟΣ

Τα αρχεία τύπου JSON δεν μπορούν να περάσουν αυτούσια σε διαφορετικό domain. Αντιθέτως, τα scripts δεν έχουν αυτό τον περιορισμό. Το JSONP βασίζεται σε αυτή τη λογική και ενθυλακώνει το JSON μέσα σε ένα script. Έτσι, μπορεί να δημιουργηθεί ένα AJAX αίτημα για ένα JSONP από το server. Μόλις αυτό επιστραφεί, η εφαρμογή θα εκτελέσει το script, αποκτώντας έτσι πρόσβαση στο ενθυλακωμένο JSON που υπάρχει.

ΚΕΦΑΛΑΙΟ 5 - SQLite

ΕΙΣΑΓΩΓΗ

Η εφαρμογή λειτουργεί βάση ορισμένων ρυθμίσεων, όπως είναι το όνομα της τωρινής έκθεσης και η διεύθυνση του Server. Φυσικά είναι άβολο για τον χρήστη να πρέπει να θέτει αυτές τις ρυθμίσεις με κάθε εκκίνηση της εφαρμογής. Πρέπει λοιπόν να υπάρχει η δυνατότητα αποθήκευσής τους κάπου, όπου να μπορούν να ανακτώνται εύκολα και να τίθενται άμεσα σε λειτουργεία. Αυτό επιτυγχάνεται με τη χρήση του SQLite.

ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Εικόνα 33: Dwayne Richard Hipp

Το SQLite δημιουργήθηκε την Παρασκευή, 9 Μαΐου 2000 από τον Dwayne Richard Hipp, ο οποίος δούλευε για την εταιρία General Dynamics πάνω σε πλοία με αυτοκαθοδηγούμενους πυραύλους για το αμερικάνικο ναυτικό. Ο στόχος ήταν η ενσωμάτωση μιας βάσης δεδομένων που δε θα χρειαζόταν ούτε κάποιο σύστημα διαχείρισής της, ούτε κάποιον διαχειριστή. Το σύστημα βασίστηκε πάνω στη PostgreSQL 6.5.

Η πρώτη έκδοση κυκλοφόρησε τρεις μήνες μετά, Πέμπτη 17 Αυγούστου τους 2000 και συνεχίζει να δέχεται αναβαθμίσεις και επιδιορθώσεις μέχρι και σήμερα.

ΠΕΡΙΓΡΑΦΗ

Εικόνα 34: SQLite logo

Το SQLite είναι μία βιβλιοθήκη η οποία υλοποιεί μια αυτοσυντηρούμενη τοπική βάση δεδομένων, η οποία δεν απαιτεί ούτε server ούτε ρυθμίσεις. Υπάρχει μέσα στη συσκευή παράλληλα με την εφαρμογή και μπορεί να επικοινωνεί μαζί της, χωρίς να απαιτείται η σύνδεση στο internet. Η ύπαρξή της είναι απαραίτητη για εφαρμογές που πρέπει να λειτουργούν offline, όπως επίσης και για εφαρμογές που βασίζονται σε ρυθμίσεις.

Το SQLite προσφέρει κάθε δυνατότητα που προσφέρει μια κανονική SQL βάση. Αποθηκεύει τα δεδομένα της απευθείας σε αρχεία μέσα στη μνήμη με ένα συγκεκριμένο format, πράγμα που επιτρέπει την αντιγραφή της βάσης σε πολλαπλές πλατφόρμες.

Το μέγεθος της βάσης, με όλες τις δυνατότητες ενεργοποιημένες, κυμαίνεται περίπου στα 500 kB. Υπάρχει επιπλέον η δυνατότητα απενεργοποίησης πολλών από αυτά, μειώνοντας το μέγεθός της σε λιγότερο από 300 kB. Το μέγεθός της, όπως και η ταχύτητα που προσφέρει στην ανάγνωση και εγγραφή των δεδομένων της, την έχουν κάνει δημοφιλή επιλογή για ενσωμάτωση σε mobile εφαρμογές.

ΛΕΙΤΟΥΡΓΕΙΑ ΤΗΣ SQLITE

Στον κώδικα σε Cordova, η χρήση της SQLite γίνεται με τη χρήση ενός plugin. Η διαχείρισή του γίνεται ακολουθώντας τα εξής βήματα:

```
document.addEventListener('deviceready', function() {
  db = window.sqlitePlugin.openDatabase({
    name: 'myDB.db',
    location: 'default'
  });
  db.transaction(runSettingsTransaction, error, success);
});
```

Εικόνα 35: Εντολή openDatabase

1. Αρχικοποίηση της βάσης, άνοιγμα αυτής με την “window.sqlitePlugin.openDatabase” και η αποθήκευσή της σε μια global μεταβλητή. Ως ορίσματα, δίνονται το όνομα της βάσης και το μονοπάτι της τοποθεσίας της. Μπορεί επιπλέον σαν τοποθεσία να δοθεί το “default”. Στην περίπτωση που δε βρεθεί η συγκεκριμένη βάση στη δοσμένη τοποθεσία, δημιουργείται αυτόματα από το plugin.

2. Έχοντας καταχωρήσει τη βάση σε μια μεταβλητή είναι δυνατή μέσω αυτής η εκτέλεση συναλλαγών με τη βάση. Υπάρχουν δύο ειδών συναλλαγές που μπορούν να πραγματοποιηθούν, η “transaction” και η “readTransaction”. Η “transaction” χρησιμοποιείται όταν επιχειρείται η δημιουργία, η προσθήκη, η τροποποίηση και γενικότερα οποιαδήποτε άλλη ενέργεια θα επηρέαζε κάποιο στοιχείο της βάσης. Σε αντίθεση με αυτή, η “readTransaction” χρησιμοποιείται όταν επιχειρείται η ανάγνωση ενός ή περισσοτέρων στοιχείων της βάσης, χωρίς τη δυνατότητα αλλαγής των στοιχείων της.

3. Με τη δημιουργία της βάσης χρειάζεται πρώτα να δημιουργηθεί ένας τουλάχιστον πίνακας, μέσα στον οποίο θα καταχωρηθούν τα δεδομένα που θα εισαχθούν. Οι πίνακες έχουν την ίδια δομή με τους πίνακες σε μια κοινή βάση SQL. Τα δεδομένα τους πρέπει να είναι ορισμένα με στατική ονομασία και τύπο δεδομένων, ενώ πρέπει να οριστεί τουλάχιστον ένα “primary key” που θα χρειαστεί για την διαφοροποίηση των διαφορετικών εγγραφών μέσα στη βάση.

4. Εφόσον δημιουργηθεί κάποιος πίνακας, είναι εφικτή η ανάγνωσή του και η εγγραφή σε αυτόν.

   Η ανάγνωση των στοιχείων της SQLite γίνεται ως εξής:

```
function executeTransaction(transaction, valueX, valueY) {
  transaction.executeSql(
    "SELECT * FROM table WHERE x = ? AND y = ?", [valueX, valueY],
    function(transaction, result) {
      //Success function:
```

```
      //διαχείριση του result
    },
    function(error) {
      //Failure function:
      //διαχείριση του error
    }
  );
}
```

Μέσα σε ένα function περνάμε ως όρισμα ένα transaction που θα γίνει με τη SQLite. Στη συνέχεια ορίζουμε ένα SELECT αίτημα σε SQL μέσα σε ένα string. Σε περίπτωση που υπάρχουν περισσότερες παράμετροι, θα πρέπει να τους ορίσουμε μέσα στο string ως αγγλικά question marks (“?”), ενώ στη συνέχεια θα περάσουμε την κάθε μεταβλητή μέσα σε ένα πίνακα, με τη κάθε μια από αυτές να αντιστοιχεί στο κάθε question mark. Στη συνέχεια δηλώνεται η function που θα εκτελεστεί σε περίπτωση επιτυχίας, η οποία δέχεται ως όρισμα το αποτέλεσμα του αιτήματος. Ύστερα από εκείνη, μπορούμε να ορίσουμε τη function που θα εκτελεστεί σε περίπτωση αποτυχίας του SQL αιτήματος. Η function αυτή παίρνει ως όρισμα το error. Μέσω αυτής μπορούμε να ορίσουμε τι θα εκτελεστεί σε περίπτωση που κάτι πάει λάθος.

ΕΠΙΛΟΓΟΣ

Η SQLite είναι ένας εναλλακτικός τρόπος διαχείρισης και αποθήκευσης δεδομένων τοπικά μέσα σε μια εφαρμογή, δίχως αυτά να χάνονται κατά τον τερματισμό της.

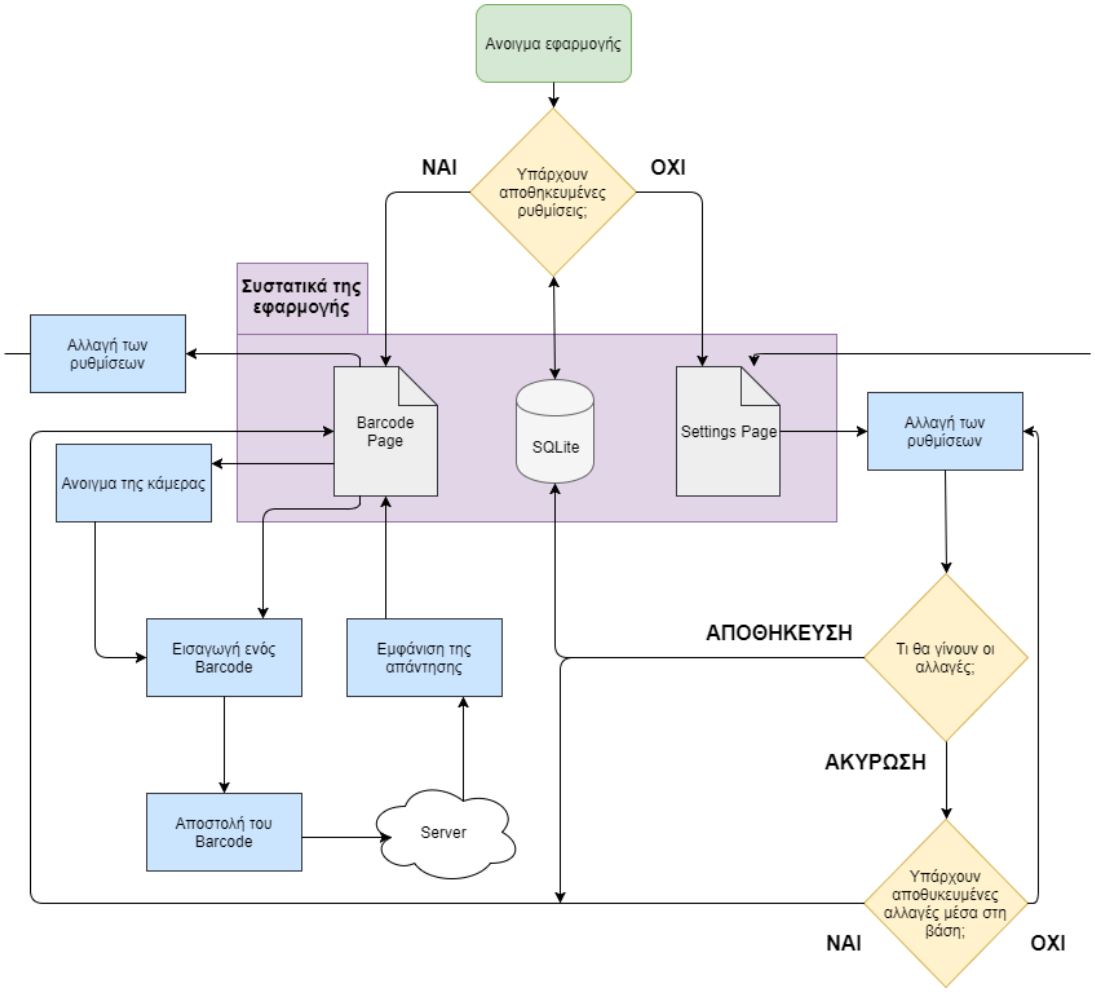
Έχει δημιουργηθεί με γνώμονα τις παραδοσιακές βάσεις δεδομένων. Παρά το γεγονός ότι δεν καταλαμβάνει μεγάλο όγκο, μπορεί να εκτελέσει τις περισσότερες εντολές SQL.

ΚΕΦΑΛΑΙΟ 6 - WORKFLOW

ΕΙΣΑΓΩΓΗ

Το βασικό κομμάτι της εφαρμογής αποτελείται από δύο HTML αρχεία, index.html και main.html, σε συνδυασμό με τα αντίστοιχα JavaScript αρχεία, index.js και main.js. Σε αυτό το κεφάλαιο αναγράφεται ο κύκλος ζωής της εφαρμογής, από την πρώτη εκκίνησή της μέχρι και την πρώτη αποστολή ενός barcode στον server. Κατά τη διάρκεια αυτού του κεφαλαίου παρουσιάζονται εικόνες από τον κώδικα των προαναφερόμενων αρχείων, με σχόλια και επεξηγήσεις για κάθε μία από αυτές.

Η ΡΟΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ



Εικόνα 36: Διάγραμμα ροής της εφαρμογής

Στο παραπάνω διάγραμμα φαίνεται η ροή της εφαρμογής από την εκκίνησή της. Με το ξεκίνημα της εφαρμογής, η εφαρμογή αναζητά υπάρχουσες ρυθμίσεις στη SQLite που εμπεριέχεται σε αυτή. Εάν δε βρεθούν ρυθμίσεις, η εφαρμογή ανακατευθύνει το χρήστη στη σελίδα με τις ρυθμίσεις για να τις διαμορφώσει όπως πρέπει, ειδάλλως γίνεται ανακατεύθυνση στη σελίδα στην οποία ο χρήστης μπορεί να διαβάζει barcodes.

Στη σελίδα των ρυθμίσεων, αφότου ο χρήστης κάνει τις επιθυμητές αλλαγές του, υπάρχει η δυνατότητα αποθήκευσης των αλλαγών που έχει κάνει στην SQLite και η δυνατότητα ακύρωσης των αλλαγών, ώστε να διατηρηθούν οι προηγούμενες ρυθμίσεις που έκανε. Στη περίπτωση που δεν είχαν βρεθεί προυπάρχουσες ρυθμίσεις, η εφαρμογή αποτρέπει τον χρήστη από την ακύρωση όσων έχει κάνει, καθώς η εφαρμογή λειτουργεί βάση αυτών. Σε κάθε περίπτωση, μετά από μια επιτυχή αποθήκευση ή ακύρωση, η εφαρμογή ανακατευθύνει το χρήστη στη σελίδα όπου μπορεί αυτός να διαβάσει Barcodes.

Στη δεύτερη αυτή σελίδα, ο χρήστης έχει τη δυνατότητα να εισάγει ένα Barcode, να ανοίξει την κάμερα της συσκευής και να αλλάξει τις ρυθμίσεις της εφαρμογής. Στη περίπτωση που ο χρήστης εισάγει ένα Barcode είτε γράφοντας το με το χέρι είτε διαβάζοντας το μέσω Barcode Reader ή κάμερας, αυτό στέλνεται άμεσα στο

server που έχει οριστεί από τις ρυθμίσεις. Η εφαρμογή έπειτα περιμένει να πάρει και να διαχειριστεί την απάντηση που θα λάβει από το server. Τέλος, η εφαρμογή προβάλει τα αποτελέσματα στο χρήστη.

INDEX.HTML ΚΑΙ JAVASCRIPT

```
        <link rel="stylesheet" type="text/css" href="css/index.css">
        <title></title>
    </head>
    <body>
        <div class="app">
            <h1>ΔΕΘ-HELEXPO</h1>
            <div id="deviceready" class="blink">
                <p class="event listening">Welcome</p>
                <p class="event received">Device is Ready</p>
            </div>
        </div>
        <script type="text/javascript" src="cordova.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
    </body>
</html>
```

Εικόνα 37: index.html

Η εφαρμογή ξεκινάει με την κλήση της index.html, η οποία δημιουργήθηκε αυτόματα από το Cordova. Η χρησιμότητα αυτής της σελίδας είναι η υλοποίηση του Splash Screen, της οθόνης που εμφανίζεται μέχρι να γίνει η φόρτωση της εφαρμογής, όπως επίσης και η φόρτωση της index.js, η οποία θα ανοίξει το επόμενο HTML αρχείο. Η σελίδα αυτή υλοποιεί ένα CSS, το οποίο δημιουργήθηκε επίσης αυτόματα από το Cordova.

```
var app = {
    initialize: function() {
        document.addEventListener('deviceready', this.onDeviceReady.bind(this), false);
    },
    onDeviceReady: function() {
        this.receivedEvent('deviceready');
    },
    receivedEvent: function(id) {
        var parentElement = document.getElementById(id);
        var listeningElement = parentElement.querySelector('.listening');
        var receivedElement = parentElement.querySelector('.received');

        listeningElement.setAttribute('style', 'display:none;');
        receivedElement.setAttribute('style', 'display:none;');
        location="main.html";
    }
};

app.initialize();
```

Εικόνα 38: index.js

Η index.html, η οποία δημιουργήθηκε επίσης αυτόματα από το Cordova, αποτελείται από την μεταβλητή “app”, τον constructor της, στον οποίο γίνονται bind τα events που μπορεί να δεχτεί και η κλήση ενός από αυτά.

Η “initialize” προσθέτει έναν EventListener για το div με id “deviceready”, ώστε μόλις γίνει η προετοιμασία της εφαρμογής, να καλεστεί η “onDeviceReady”.

Η “onDeviceReady” περιμένει την κλήση της από την “initialize” και έπειτα καλεί την “recievedEvent”.

Η “recievedEvent” εξαφανίζει τα στοιχεία που υπάρχουν μέσα στο “deviceready” και καλεί το επόμενο HTML αρχείο που θα φορτωθεί, το main.html.

MAIN.HTML

Το main.html είναι ένα HTML αρχείο το οποίο εμπεριέχει τις δύο διαφορετικές σελίδες της εφαρμογής, την “barcode_page” και την “settings_page”, όπως επίσης και τις δηλώσεις του jQuery Mobile, του Cordova και του main.js, το οποίο υλοποιεί όλες τις διεργασίες της εφαρμογής.

```
<div data-role="page" id="barcode_page" data-cache="false">
  <div id="barcode_div" data-role="ui-content">
    <table width="100%">
      <tr>
        <td width="100%" align="center">
          <table width="100%">
            <tr>
              <td height=40 align=center valign=middle bgcolor="#f0f0f0">
                ΔΕΘ 2107 - ΕΓΚΑΙΝΙΑ - ΕΛΕΓΧΟΣ ΠΡΟΣΚΛΗΣΕΩΝ
              </td>
            </tr>
          </table>
          <div id="msg">
          </div>
          <br>
          <input type="text" name="bc" id="bc" style="width:400px;height:100px" onchange="sendBC()">
          <br>
          <table style="width=100%">
            <tr>
              <td><button class="settings_button" href="#settings_page">SETTINGS</button></td>
              <td></td>
              <td><button id="camera_button">CAMERA</button></td>
            </tr>
          </table>
        </td>
      </tr>
    </table>
  </div>
</div>
```

Εικόνα 39: Δήλωση της barcode_page

Η “barcode_page” αποτελεί την κεντρική σελίδα της εφαρμογής. Εδώ γίνεται η εισαγωγή των barcodes, η αποστολή τους στον server και η προβολή των απαντήσεων που δέχεται από αυτόν. Η σελίδα περιέχει ένα input με όνομα “bc”, το οποίο χρησιμοποιείται για την εισαγωγή των barcode. Περιέχει επίσης ένα div με όνομα “msg”, στο οποίο εμφανίζονται μηνύματα προς τους κατόχους των εισιτηρίων και τους ελεγκτές. Τέλος υπάρχουν δύο κουμπιά στο κάτω μέρος της σελίδας, ένα για τις ρυθμίσεις και ένα για τη κάμερα.

```
<div data-role="page" id="settings_page" data-theme="a" data-cache="false">
  <div id="settings_div" data-role="ui-content">
      <table width="100%">
        <tr>
          <td width="100%" align="center">
            <table id="settings_table" style="width: 100%">
```

Εικόνα 40: Δήλωση της settings_page και του εμφολευμένου πίνακα

Η “settings_page” αποτελεί τη σελίδα όπου γίνονται αλλαγές στις ρυθμίσεις της εφαρμογής. Η σελίδα αυτή εμπεριέχει ένα table στο οποίο βρίσκονται όλες οι ρυθμίσεις. Στον πίνακα μέσα υπάρχουν τα εξής πεδία:

```
<tr>
  <td><label for="ekd_input">EXHIBITION: </label></td>
  <td colspan="2"><input id="ekd_input"
    type="text"
    placeholder="Set Exhibition..."
    onchange="enableSave()"
    onkeyup="enableSave()"
    style="font-size: 30px;">
  </td>
</tr>
```

Εικόνα 41: Exhibition input

Ένα πεδίο για την εισαγωγή του ονόματος της έκθεσης.

```
<tr>
  <td><label for="gate_input">GATE: </label></td>
  <td colspan="2"><input id="gate_input"
    type="text"
    placeholder="Set Gate..."
    onchange="enableSave()"
    onkeyup="enableSave()"
    style="font-size: 30px;">
  </td>
</tr>
```

Εικόνα 42: Gate input

Ένα πεδίο που περιγράφει την πύλη στην οποία βρίσκεται το tablet.

```
<tr>
  <td><label for="name_input">NAME: </label></td>
  <td colspan="2"><input id="name_input"
    type="text"
    placeholder="Set Name..."
    onchange="enableSave()"
    onkeyup="enableSave()"
    style="font-size: 30px;">
  </td>
</tr>
```

Εικόνα 43: Name input

Ένα πεδίο για την εισαγωγή του ονόματος

```
<tr>
  <td><label for="url_input">URL: </label></td>
  <td colspan="2"><input id="url_input"
    type="text"
    placeholder="Set URL..."
    onchange="enableSave()"
    onkeyup="enableSave()"
    style="font-size: 30px;">
  </td>
</tr>
```

Εικόνα 44: URL input

Ένα πεδίο για την εισαγωγή του URL του server που θα δέχεται τα barcodes και θα επιστρέφει τα JSONP.

```
<tr>
  <td><label for="callback_input">FUNCTION: </label></td>
  <td colspan="2"><input id="callback_input"
    type="text"
    placeholder="Default name: cb"
    style="font-size: 30px;">
  </br>
  </td>
</tr>
```

Εικόνα 45: Callback function name input

Ένα πεδίο που καθορίζει το όνομα του function που επιστρέφει ο server στο JSONP. Σε περίπτωση που δε δοθεί κάποια τιμή, χρησιμοποιείται το όνομα “cb”.

```
<tr>
  <td></td>
  <td style="width:30%;float:right">
    <button id="json_button">SHOW JSON</button>
    </br>
  </td>
</tr>
```

Εικόνα 46: Show JSON button

Ένα κουμπί το οποίο πραγματοποιεί ένα AJAX αίτημα με τα παραπάνω στοιχεία, ώστε να μπορεί να δοκιμαστεί κατά πόσο τα παραπάνω στοιχεία είναι έγκυρα και για να φανούν τα πεδία του JSONP που επιστρέφονται.

Στη συνέχεια υπάρχουν τρία πεδία τα οποία περιγράφουν την απάντηση του server.

```
<tr>
  <td><label for="barcode_input">BARCODE: </label></td>
  <td colspan="2"><input id="barcode_input"
    type="text"
    placeholder="Default var: barCode"
    style="font-size: 30px;">
  </td>
</tr>
```

Εικόνα 47: Barcode field input

Ένα πεδίο για την εισαγωγή του ονόματος της μεταβλητής που εμπεριέχει το barcode που στάλθηκε. Σε περίπτωση που δε δοθεί κάποια τιμή, χρησιμοποιείται το όνομα “barCode”.

```
<tr>
  <td><label for="validation_input">VALIDATION: </label></td>
  <td colspan="2"><input id="validation_input"
    type="text"
    placeholder="Default var: valid"
    style="font-size: 30px;">
  </td>
</tr>
```

Εικόνα 48: Validation field input

Ένα πεδίο για την εισαγωγή του ονόματος της μεταβλητής που εμπεριέχει την απάντηση του server για την εγκυρότητα του barcode. Σε περίπτωση που δε δοθεί κάποια τιμή, χρησιμοποιείται το όνομα “valid”.

```
<tr>
  <td><label for="reason_input">REASON: </label></td>
  <td colspan="2"><input id="reason_input"
    type="text"
    placeholder="Default var: reason"
    style="font-size: 30px;">
  </td>
</tr>
```

Εικόνα 49: Reason field input

Ένα πεδίο για την εισαγωγή του ονόματος της μεταβλητής που εμπεριέχει την εξήγηση της απάντησης του server, στη περίπτωση που αυτός δεν

επικυροποίησε το barcode. Σε περίπτωση που δε δοθεί κάποια τιμή, χρησιμοποιείται το όνομα “reason”.

```
<table style="width: 75%">
    <td style="width: 33%">
      <button class="save_button" style="float:left">SAVE</button>
    </td>
    <td style="width: 33%"></td>
    <td style="width: 33%">
      <button class="cancel_button" style="float:right">CANCEL</button>
    </td>
</table>
```

Εικόνα 50: Save και Cancel buttons

Τέλος υπάρχουν δύο κουμπιά, ένα για την αποθήκευση των αλλαγών και ένα για την ακύρωσή τους.

MAIN.JS

Έχοντας τελειώσει με τη παρουσίαση των δομικών στοιχείων του main.html, θα αρχίσει η παρουσίαση των διεργασιών μέσα στο main.js, με τη σειρά που αυτές εκτελούνται όταν αρχίζει η εφαρμογή.

```
document.addEventListener('deviceready', function() {
  db = window.sqlitePlugin.openDatabase({
    name: 'myDB.db',
    location: 'default'
  });
  db.transaction(runSettingsTransaction, error, success);
});
```

Εικόνα 51: Εκκίνηση της SQLite

Κατά την εκκίνηση της εφαρμογής, αρχικοποιήται και ανοίγει η SQLite που υπάρχει μέσα στην εφαρμογή, με το όνομά της να είναι “myDB.db” και η τοποθεσία της να είναι η default. Μόλις αρχηκοποιηθεί η SQLite, αρχίζει μια νέα συναλλαγή μαζί της.

```
function runSettingsTransaction(t) {
  t.executeSql(
    'CREATE TABLE IF NOT EXISTS settings(`ekd` TEXT,' +
                                        ``gate` TEXT,' +
                                        ``name` TEXT,' +
                                        ``url` TEXT,' +
                                        ``cb` TEXT,' +
                                        ``barcode_v` TEXT,' +
                                        ``valid_v` TEXT,' +
                                        ``reason_v` TEXT);', [],
    //success callback
    function() {
      db.readTransaction(runSettingsCheckTransaction, error, success);
    },
    //error callback
    error_alert("create table settings", error)
  );
}
```

Εικόνα 52: Δημιουργία πίνακα settings

Η “runSettingsTransaction” δημιουργεί ένα νέο πίνακα SQL με όνομα “settings”, εάν δεν υπάρχει ήδη, στον οποίο αποθηκεύονται όλες οι ρυθμίσεις από τη “settings_page”. Με το τελείωμα αυτής της συναλλαγής, αρχίζει μια νέα συναλλαγή.

```
function runSettingsCheckTransaction(tx) {
  tx.executeSql(
    'select * FROM settings', [],
    function(tx, result) {
      //if no entry exists go to the settings page
      if (result.rows.length === 0) {
        areFieldsCompleted = false;
        $.mobile.changePage("#settings_page");
      }
      //else check the stored permissions
      else {
        ekd_var = result.rows.item(0).ekd;
        name_var = result.rows.item(0).name;
        gate_var = result.rows.item(0).gate;
        url_string = result.rows.item(0).url;
        cb_string = result.rows.item(0).cb;
        barcode_var = result.rows.item(0).barcode_v;
        valid_var = result.rows.item(0).valid_v;
        reason_var = result.rows.item(0).reason_v;
        defineCallback();
        $.mobile.changePage("#barcode_page");
      }
    },
    error_alert("'select * FROM settings'", error)
  );
}
```

Εικόνα 53: Ανάκτηση ρυθμίσεων

Η “runSettingsCheckTransaction” διαβάζει όλες τις ρυθμίσεις που υπάρχουν στον πίνακα “settings” της SQLite. Στη περίπτωση που υπήρχε μια εγγραφή, οι ρυθμίσεις όλες θα αποθηκεύονταν σε μεταβλητές και η εφαρμογή θα άνοιγε τη σελίδα για την εισαγωγή των barcodes. Αφού όμως η εφαρμογή μόλις έτρεξε για πρώτη φορά και ο πίνακας δεν εμπεριέχει καμία εγγραφή, η εφαρμογή ανοίγει τη σελίδα “settings_page”, ώστε ο χρήστης να εισάγει τις απαραίτητες ρυθμίσεις.

```
$(document).on('pagebeforeshow', '#settings_page', function(e) {
  page_flag = false;
  document.getElementById("ekd_input").value = ekd_var;
  document.getElementById("gate_input").value = gate_var;
  document.getElementById("name_input").value = name_var;
  document.getElementById("url_input").value = url_string;
  document.getElementById("callback_input").value = cb_string;
  document.getElementById("barcode_input").value = barcode_var;
  document.getElementById("validation_input").value = valid_var;
  document.getElementById("reason_input").value = reason_var;
  if (!areFieldsCompleted) {
    $('.save_button').addClass('ui-disabled');
    $('.cancel_button').addClass('ui-disabled');
  }
});
```

Εικόνα 54: Συμπλήρωση πεδίων

Πρωτού εμφανιστεί η σελίδα, τα πεδία της παίρνουν τιμές ίδιες με αυτές που βρέθηκαν από την SQLite. Κατά την πρώτη εκκίνηση, εφόσον δεν υπάρχουν τιμές στις μεταβλητές, τα πεδία παραμένουν κενά.

Επιπλέον, τα κουμπιά αποθήκευσης και ακύρωσης απενεργοποιούνται μέχρι όλα τα πεδία που δεν έχουν κάποια σταθερή τιμή για όταν παραμένουν κενά να πάρουν κάποια τιμή. Στην περίπτωση που η μεταβλητή “areFieldsCompleted” είναι true, υποδεικνύεται ότι έχουν βρεθεί προηγούμενες ρυθμίσεις μέσα στην SQLite και άρα δεν υπάρχει λόγος να απενεργοποιηθούν τα κουμπιά.

```
function enableSave() {
  if (!areFieldsCompleted &&
      document.getElementById("ekd_input").value!="" &&
      document.getElementById("gate_input").value!="" &&
      document.getElementById("name_input").value!="" &&
      document.getElementById("url_input").value!="" ) {
        $('.save_button').removeClass('ui-disabled');
        areFieldsCompleted = true;
  }
  else if (areFieldsCompleted) {
    $('.save_button').addClass('ui-disabled');
    areFieldsCompleted = false;
  }
}
```

Εικόνα 55: Έλεγχος πληρότητας υποχρεωτικών πεδίων

Οι αλλαγές στα κουμπιά που δεν έχουν κάποια default τιμή προκαλούν την κλήση της function “enableSave”. Η function αυτή ελέγχει τη τιμή των πεδίων

που δεν έχουν τιμή και στην περίπτωση που όλες έχουν κάποια τιμή, ενεργοποιούν το κουμπί της αποθήκευσης. Το κουμπί της ακύρωσης παραμένει απενεργοποιημένο, καθώς στη πρώτη εκκίνηση είναι υποχρεωτική η αποθήκευση των ρυθμίσεων.

```
$(document).on('tap', '#json_button', function(e) {
  if (document.getElementById("url_input").value!="" &&
      document.getElementById("name_input").value!="" &&
      document.getElementById("gate_input").value!="" &&
      document.getElementById("ekd_input").value!=""){
    createTemporaryCallback(document.getElementById("callback_input").value);
    $.ajax({
      type: "GET",
      url: document.getElementById("url_input").value,
      crossDomain: true,
      data: {
        barCode: "test_barcode",
        ekd: document.getElementById("ekd_input").value,
        gate: document.getElementById("gate_input").value,
        name: document.getElementById("name_input").value
      },
      dataType: 'jsonp'
    });
  }
});
```

Εικόνα 56: Δοκιμή επικοινωνίας με server

Με την εισαγωγή των υποχρεωτικών πεδίων, ο χρήστης μπορεί να πραγματοποιήσει μια δοκιμαστική κλήση με τις επιλογές που εισήγαγε για να ελέγξει αν οι πληροφορίες που έδωσε είναι σωστές. Πρώτα όμως δημιουργείται μια δοκιμαστική function με όνομα αυτό που έδωσε ο χρήστης.

```
function createTemporaryCallback(callback){
  eval("window." + callback + " = " +
  "function(barcode){"+
    "alert('Responce JSON: ' + JSON.stringify(barcode));" +
    "window."+ callback +"='';" +
  "};");
}
```

Εικόνα 57: Δημιουργία callback function

Η “createTemporaryCallback” δημιουργεί μια νέα function με βάση το όνομα που εισήγαγε ο χρήστης. Μόλις κληθεί η function, πετάγεται ένα alert με το JSON που επιστράφηκε από το server, προβάλλοντας τα πεδία που έχει το

JSON και το μήνυμα λάθους. Αυτό επιτρέπει στο χρήστη να παρατηρήσει τα πεδία που δέχεται ο server και να τα εισάγει στα αντίστοιχα παρακάτω πεδία.

```
$(document).on('tap', '.save_button', function(e) {
  if (document.getElementById("url_input").value!="" &&
      document.getElementById("name_input").value!="" &&
      document.getElementById("gate_input").value!="" &&
      document.getElementById("ekd_input").value!=""){

        $('.cancel_button').removeClass('ui-disabled');
        ekd_var = document.getElementById("ekd_input").value;
        gate_var = document.getElementById("gate_input").value;
        name_var = document.getElementById("name_input").value;
        url_string = document.getElementById("url_input").value;
        cb_string = document.getElementById("callback_input").value;
        barcode_var = document.getElementById("barcode_input").value;
        valid_var = document.getElementById("validation_input").value;
        reason_var = document.getElementById("reason_input").value;
        if (cb_string === "") cb_string = "cb";
        if (barcode_var === "") barcode_var = "barCode";
        if (valid_var === "") valid_var = "valid";
        if (reason_var === "") reason_var = "reason";
        db.transaction(deleteSettings, error, success);
      }
    });
```

Εικόνα 58: Έναρξη αποθήκευσης

Με την ολοκλήρωση των ρυθμίσεων, ο χρήστης μπορεί να πατήσει το κουμπί αποθήκευσης για να οριστικοποιήσει τις αλλαγές του. Οι τιμές όλων των πεδίων καταχωρούνται σε μεταβλητές και γίνεται έλεγχος στα πεδία που μπορούν να πάρουν default τιμές. Έπειτα καλείται η συναλλαγή “deleteSettings”.

```
function deleteSettings(tx) {
  tx.executeSql(
    "DELETE FROM settings;", [],
    function(tx) {
      db.transaction(saveSettings, error, success);
    },
    error_alert("delete settings", error)
  );
}
```

Εικόνα 59: Διαγραφή προηγούμενων ρυθμίσεων

Η “deleteSettings” διαγράφει όλα τα αποθηκευμένα στοιχεία της λίστας, εάν υπάρχουν, ώστε να μπορέσει να γίνει εισαγωγή των νέων στοιχείων. Εδώ

πρέπει να σημειωθεί ότι, ενώ υπήρχε η δυνατότητα να γίνει χρήση της “UPDATE”, προτιμήθηκε η “DELETE” ώστε να καλύπτεται και το σενάριο όπου η βάση δεν έχει καμιά εγγραφή, όπως συμβαίνει με την πρώτη εκκίνηση της εφαρμογής. Μόλις ολοκληρωθεί η “DELETE”, καλείται η συναλλαγή “saveSettings”.

```
function saveSettings(tx) {
  tx.executeSql(
    "INSERT INTO settings(ekd,"+
                         "gate,"+
                         "name,"+
                         "url,"+
                         "cb,"+
                         "barcode_v,"+
                         "valid_v,"+
                         "reason_v) VALUES (?,?,?,?,?,?,?,?);",
    [ekd_var, gate_var, name_var, url_string, cb_string, barcode_var, valid_var, reason_var],
    function() {
      defineCallback();
      $.mobile.changePage("#barcode_page");
    },
    error_alert("Insert settings error: ", error)
  );
}
```

Εικόνα 60: Δημιουργία νέων ρυθμίσεων.

Η “saveSettings” κάνει την εισαγωγή των νέων στοιχείων μέσα στη SQLite. Με την ολοκλήρωσή της, θα δημιουργηθεί η function που θα καλεστεί από τα επερχόμενα JSONP και θα ανοίξει η “barcode_page”.

```
var callback_function = function(barcode) {
  var BC = eval("barcode."+barcode_var+";");
  var PE = eval("barcode."+valid_var+";");
  var RS = eval("barcode."+reason_var+";");
  var HTML = "";
  if (BC == '0')
    HTML = '<font face=tahoma size=6 color=red><b>ΑΝΥΠΑΡΚΤΟΣ</b></font><br>';
  else if (RE === '')
    HTML = '<font face=tahoma size=6 color=green><b>' + PE + '</b></font><br>';
  else
    HTML = '<font face=tahoma size=6 color=red><b>' + PE + '</b></br></br>' + RS + '</font>';
  document.getElementById("msg").innerHTML = HTML;
  document.getElementById("bc").value = "";
};
```

Εικόνα 61: Προβολή απάντησης του server

Το function που καλείται με το JSONP παίρνει ως όρισμα το αρχείο JSON που εμπεριέχεται στην απάντηση του server. Εδώ αποσπώνται τα πεδία που

διευκρινίστηκαν στις ρυθμίσεις από το JSON και γίνονται οι κατάλληλοι έλεγχοι για την εγκυρότητα του barcode. Σε κάθε περίπτωση, το μήνυμα αναμονής που υπήρχε στο “msg” αντικαθίσταται με νέο κατάλληλο μήνυμα. Το μήνυμα επιτυχία είναι πράσινο σε περίπτωση που το barcode έγινε αποδεκτό, κόκκινο με τη λέξη “ΑΝΥΠΑΡΚΤΟΣ” στη περίπτωση που δε βρέθηκε το barcode στη βάση δεδομένων του server. Επιπλέον, στη περίπτωση που το barcode βρέθηκε στη βάση αλλά δεν είναι έγκυρο, επιστρέφεται μήνυμα με κόκκινο φόντο που προβάλει το λόγο που το barcode απορρίφθηκε. Τέλος το πεδίο με το barcode γίνεται κενό, ώστε να μπορέσει να γίνει εισαγωγή ενός καινούργιου barcode.

```
$(document).on('pageshow', '#barcode_page', function(e) {
  page_flag = true;
  document.getElementById("bc").focus();
});
```

Εικόνα 62: Προβολή της barcode_page

Μόλις γίνει εμφάνιση της “barcode_page” τίθεται το page_flag να έχει τιμή “true”. Το page_flag καθορίζει εάν η ενεργή σελίδα είναι η “barcodes_page”. Επίσης το πεδίο που δέχεται τα barcodes παίρνει αμέσως το focus. Αυτό θα είναι το σενάριο που θα ισχύει για κάθε διαδοχική κλήση από εδώ και πέρα με το άνοιγμα της εφαρμογής.

```
$('#bc').blur(function() {
  if (page_flag)
    document.getElementById("bc").focus();
});
```

Εικόνα 63: Διαχείριση focus για barcodes

Για την ευκολία των υπαλλήλων που θα διαβάζουν τα barcodes, σε περίπτωση που πατηθεί η οθόνη κάπου εκτός του πεδίου και χαθεί το focus, έχει ρυθμιστεί αυτό να ξαναπηγαίνει στο πεδίο. Στην περίπτωση που η ενεργή σελίδα πάψει να είναι η “barcodes_page”, το page_flag θα αποτρέψει την εκτέλεση αυτής της εντολής.

```
function sendBC() {
  document.getElementById("msg").innerHTML =
    '<font face=tahoma size=6><b>ΑΝΑΜΟΝΗ ΑΠΑΝΤΗΣΗΣ...</b></font><br>';
  $.ajax({
    type: "GET",
    url: url_string,
    crossDomain: true,
    data: {
      barCode: document.getElementById("bc").value,
      ekd: ekd_var,
      gate: gate_var,
      name: name_var
    },
    dataType: 'jsonp'
  });
}
```

Εικόνα 64: Αποστολή barcode στο server

Μόλις γίνει η εισαγωγή ενός barcode στο πεδίο, αρχίζει η διαδικασία για την αποστολή του στον server. Το div “msg” αλλάζει με κατάλληλο μήνυμα που υποδηλώνει την αναμονή της απάντησης από τον server και στη συνέχεια δημιουργείται το κατάλληλο AJAX αίτημα για την αποστολή του barcode σε αυτόν.

Στη συνέχεια η εφαρμογή είναι έτοιμη να επαναλάβει τη διαδικασία για το επόμενο barcode.

```
$(document).on('tap', '#camera_button', function(e) {
        cordova.plugins.barcodeScanner.scan(
            function (result) {
                document.getElementById("bc").value = result.text;
                sendBC();
            },
            function (error) {
                alert("Scanning failed: " + error);
            },
            {
                preferFrontCamera: false,
                showFlipCameraButton: false,
                showTorchButton: true,
                torchOn: false,
                prompt: "Place a barcode ALONG the red line",
                resultDisplayDuration: 0,
                formats : "CODE_128",
                orientation: "landscape"
            }
        );
    });
```

Εικόνα 65: Ενεργοποίηση κάμερας

Ως επιπλέον δυνατότητα της εφαρμογής, δίνεται η δυνατότητα ανάγνωσης του barcode μέσω της κάμερας της συσκευής. Η δυνατότητα αυτή είναι ιδιαίτερα χρήσιμη στη περίπτωση που ο πελάτης έχει φέρει μόνο φωτογραφία του εισιτηρίου και άρα δεν μπορεί να γίνει η ανάγνωση του barcode μέσω barcode reader. Το plugin που χρησιμοποιήθηκε ανοίγει αρχικά την πίσω κάμερα (preferFrontCamera: false) και χρησιμοποιεί αποκλειστικά αυτή (showFlipCameraButton: false).

Επιπλέον έχει ρυθμιστεί να μην είναι ανοιχτός ο φακός όταν ανοίγει η κάμερα (torchOn: false), αλλά παρέχεται η δυνατότητα ενεργοποίησης και απενεργοποίησής του (showTorchButton: true). Η κάμερα παραμένει οριζόντια για όσο είναι ανοιχτή (orientation: “landscape”), εμφανίζει μια κόκκινη γραμμή στο κέντρο κατά μήκος της οθόνης και εμφανίζει μήνυμα στον χρήστη που του υποδεικνύει πώς να χρησιμοποιήσει την κάμερα (prompt: “Place a barcode ALONG the red line”). Το format των barcode που περιμένει η εφαρμογή έχει τεθεί σε CODE_128 (formats: “CODE_128”). Μόλις αναγνωριστεί και διαβαστεί κάποιο barcode, θέτει το πεδίο εισαγωγής των barcodes με τη τιμή που διαβάστηκε, ξεκινά αμέσως τη διαδικασία αποστολής του αιτήματος AJAX στο server χωρίς να περιμένει καθόλου (resultDisplayDuration: 0).

PLUGINS ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Το πρώτο plugin της εφαρμογής, το ***“cordova-plugin-compat”***. Χρησιμοποιείται για τη διαχείριση android permissions από άλλα plugins.

Το επόμενο plugin, το ***"cordova-plugin-whitelist"***, είναι υπεύθυνο για τη πρόσβαση της εφαρμογής σε διαφορετικά domain από την εφαρμογή.

Για να χρησιμοποιηθεί η SQLite με το Cordova απαιτείται η χρήση του ***"Cordova-SQLite-Storage"***. Αυτό το plugin επέτρεψε την δημιουργία και χρήση του SQLite στην εφαρμογή. Το plugin αυτό μπορεί να λειτουργήσει τόσο σε Android και iOS όσο και σε Windows 10 (UWP).

Για την ανάγνωση των barcodes χρησιμοποιήθηκε επίσης ένα άλλο plugin, το ***"Phonegap-Plugin-Barcode-Scanner"***, ένα plugin για την ανάγνωση barcodes μέσω της κάμερας του τηλεφώνου. Υποστηρίζει πλατφόρμες iOS, Android, UWP, BlackBerry 10 και Browser.

Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί το plugin της βιβλιοθήκης ***"ZXing" (Zebra Crossing)***, η οποία επιτρέπει την ανάγνωση και δημιουργία πολλών ειδών barcodes. Ενώ το interface που παρέχει και η ταχύτητα στην ανάγνωση των barcodes το έκαναν ιδανικότερο στη χρήση, λόγο προβλημάτων στις εκδόσεις του κατά τη διάρκεια δημιουργίας της εφαρμογής, προτιμήθηκε το προαναφερόμενο.

ΕΠΙΛΟΓΟΣ

Η εφαρμογή υλοποιεί δύο HTML αρχεία, το index.html και το main.html, μαζί με τα αντίστοιχα JavaScript αρχεία τους. Αρχίζοντας από το index.html, ο χρήστης εκτρέπεται στη main.html, όπου και θα μείνει για το υπόλοιπο της διάρκειας ζωής της εφαρμογής.

Κατά τη πρώτη εκκίνηση της εφαρμογής, η εφαρμογή θα πλοηγηθεί στη "settings_page", ώστε ο χρήστης να εισάγει τις ρυθμίσεις της εφαρμογής. Αφού γίνει αυτό, ο χρήστης μπορεί να αποθηκεύσει τις ρυθμίσεις του στην SQLite και μετά να πλοηγηθεί στη "barcode_page", τη σελίδα που θα φαίνεται από εδώ και πέρα στην αρχή.

Εδώ υπάρχει η δυνατότητα εισαγωγής barcodes, τα οποία θα στέλνονται στο server και θα εμφανίζεται η απάντηση που αυτός στέλνει στην οθόνη.

ΚΕΦΑΛΑΙΟ 7 – ΣΥΓΚΡΗΣΕΙΣ ΚΑΙ ΒΕΛΤΙΩΣΕΙΣ

ΕΙΣΑΓΩΓΗ

Εφόσον η κάθε υλοποίηση κάνει χρήση διαφορετικών γλωσσών προγραμματισμού, διαφορετικών συσκευών και διαφορετικών τεχνολογιών γενικότερα, είναι λογικό να υπάρχουν διαφοροποιήσεις μεταξύ τους.

Σε αυτό το κεφάλαιο εξετάζονται τα πλεονεκτήματα και τα μειονεκτήματα της κάθε υλοποίησης, τα προβλήματα που συναντήθηκαν εκτός του περιβάλλοντος προγραμματισμού καθώς και μερικές προτάσεις για μελλοντικές βελτιώσεις που θα μπορούσαν να γίνουν στην εφαρμογή. Επίσης αναφέρονται και μερικοί εναλλακτικοί τρόποι υλοποίησής της.

ΠΛΕΟΝΕΚΤΙΜΑΤΑ ΤΗΣ ΚΑΘΕ ΥΛΟΠΟΙΗΣΗΣ

Η κάθε υλοποίηση έχει τα πλεονεκτήματα και τα μειονεκτήματά της:

- ***Πλεονεκτήματα υπολογιστή:***

1. Ο υπολογιστής είναι ευκολότερος στη χρήση, καθώς οι περισσότεροι άνθρωποι χρησιμοποιούν τον υπολογιστή περισσότερο από τα tablet.

2. Ο υπολογιστής είναι πιο ανθεκτικός από ότι τα tablet και μπορεί να παρουσιάσει λιγότερα προβλήματα.

3. Τα προβλήματα που παρουσιάζονται σε υπολογιστές είναι πιο εύκολο να επιλυθούν, καθώς όχι μόνο είναι ευκολότερο το άνοιγμα του για την εξέτασή του αλλά υπάρχει επίσης περισσότερη γνώση για την επιδιόρθωση υπολογιστών από ότι για τα tablet.

4. Οι υπολογιστές μπορούν να εκτελέσουν παράλληλες διεργασίες, όπου εμφανίζεται ανάγκη. Αυτό μπορεί να γίνει και μέσω tablet, αλλά η διαχείριση πολλαπλών εργασιών είναι πολύ ευκολότερη σε υπολογιστές.

5. Οι υπολογιστές έχουν περισσότερες θύρες για περιφερειακές συσκευές και επιτρέπουν την διαχείριση πολλών τέτοιων συσκευών ταυτόχρονα. Ένα αξιόλογο παράδειγμα για αυτό το πλεονέκτημα είναι ο εκτυπωτής.

- ***Μειονεκτήματα υπολογιστή:***

1. Όπως προαναφέρθηκε, ο υπολογιστής είναι πιο δύσκολος να αντικατασταθεί από τις υπόλοιπες συσκευές και άρα η διαδικασία εξαρτάται κυρίως από τη λειτουργία του. Σε περίπτωση σφάλματος λοιπόν, η λειτουργία είναι δύσκολη και χρονοβόρα στην επιδιόρθωση.

2. Ο υπολογιστής αξιοποιείται για την ανάγνωση και επικύρωση barcodes ενώ θα μπορούσε να αξιοποιείται σε εργασίες που απαιτούν περισσότερη υπολογιστική ισχύ.

3. Η μετακίνηση, η εγκατάσταση και η απεγκατάσταση όλων των υπολογιστών σε κάποιο χώρο είναι δύσκολη και χρονοβόρα λόγω του όγκου που διαθέτουν.

4. Σε περίπτωση που απουσιάζει το barcode reader, ο υπολογιστής δε μπορεί να διαβάσει τα barcodes.

5. Η αντικατάσταση ή αγορά ενός νέου υπολογιστή κοστίζει περισσότερο.

- ***Πλεονεκτήματα tablet:***

1. Τα tablet δεν χρειάζονται εγκατάσταση ή απεγκατάσταση στο χώρο. Μπορούν να μετακινηθούν ή να αντικατασταθούν σε πολύ λίγο χρόνο και διακριτικά. Επιπλέον πλεονέκτημα είναι και η απουσία των καλωδίων, καθώς το μόνο που χρειάζεται είναι το καλώδιο του φορτιστή και αυτό μόνο κατά τη φόρτιση του tablet.

2. Τα tablet είναι πιο φθηνά από ότι οι υπολογιστές. Η αντικατάσταση και η αγορά τους συμφέρει περισσότερο για αυτή την εργασία.

3. Η αντικατάσταση των υπολογιστών από tablet σημαίνει ότι οι υπολογιστές που αποσύρθηκαν μπορούν να αξιοποιηθούν κάπου καλύτερα.

4. Για την ανάγνωση των barcodes, τα tablet όχι μόνο αποδίδουν το ίδιο καλά με τους υπολογιστές αλλά σε περίπτωση που απουσιάζει το barcode reader, μπορούν να διαβάσουν τα barcodes μέσω της ενσωματωμένης κάμεράς τους.

5. Χρησιμοποιώντας την ενσωματωμένη κάμερά τους, τα tablet μπορούν να αναγνωρίσουν και να διαβάσουν barcodes μέσα από οθόνες άλλων ηλεκτρονικών συσκευών όπως κινητών, πράγμα που το barcode reader δεν μπορεί να κάνει.

- ***Μειονεκτήματα tablet:***

1. Δεν είναι εύκολη η χρήση του για άλλες πιο πολύπλοκες διεργασίες σε σχέση με τον υπολογιστή.

2. Τα tablet εξαρτώνται από τη μπαταρία τους και χρειάζονται φόρτιση κατά τακτά χρονικά διαστήματα.

3. Ένα tablet, ανάλογα πάντα και με την ποιότητά του, είναι λιγότερο ανθεκτικό από ότι ο υπολογιστής και είναι πιο δύσκολο να επιδιορθωθεί σε περίπτωση βλάβης.

4. Η χρήση των tablet με άλλες περιφερειακές συσκευές περιορίζεται όχι μόνο από το πλήθος των θυρών που μπορεί να δεχτεί -συνήθως μία-, αλλά και από το είδος της θύρας που διαθέτει.

ΕΞΩΤΕΡΙΚΑ ΠΡΟΒΛΗΜΑΤΑ ΚΑΙ ΛΥΣΕΙΣ

Εκτός από την υλοποίηση της εφαρμογής στο Cordova, προέκυψαν μερικά εξωτερικά προβλήματα κατά την πρώτη χρήση της εφαρμογής, που έγινε μέσω tablet στη ΔΕΘ 2017. Τα προβλήματα αυτά ήταν:

- Τα tablets λειτουργούν με επαναφορτιζόμενη μπαταρία και όχι με ρεύμα. Αυτό σημαίνει ότι σε περίπτωση που δεν υπάρχει κάποια πηγή ρεύματος κοντά, τα tablet έχουν περιορισμένο χρόνο ζωής προτού χρειαστεί να αποσυρθούν για την επαναφόρτισή τους. Η λύση σε αυτό το πρόβλημα δόθηκε φέρνοντας επιπλέον tablet, τα οποία ήταν έτοιμα και φορτισμένα στη περίπτωση που χρειαζόταν η επαναφόρτιση ενός άλλου. Στη συνέχεια, το πρώτο tablet συνδέονταν με τον φορτιστή και έπαιρνε τη θέση του δευτέρου στην αναμονή.

- Υπάρχει κίνδυνος, λόγω ασθενούς σήματος ή λόγω του μεγάλου αριθμού πελατών που χρησιμοποιούν ηλεκτρονικές συσκευές, η σύνδεση με το internet να πέσει. Αυτό σημαίνει ότι η συσκευή δεν μπορεί να επικυροποιήσει το barcode μέσω του server. Η λύση που δόθηκε ήταν η δημιουργία ξεχωριστού δικτύου στο οποίο θα συνδέονταν αποκλειστικά τα tablets που θα πραγματοποιούσαν την ανάγνωση των Barcodes. Έτσι αποτράπηκαν οι θόρυβοι και οι καθυστερήσεις που θα προκαλούσαν άλλες συσκευές στο δίκτυο.

ΒΕΛΤΙΩΣΕΙΣ ΠΟΥ ΕΠΙΔΕΧΕΤΑΙ Η ΤΡΕΧΟΥΣΑ ΕΦΑΡΜΟΓΗ

Πιθανές βελτιώσεις που θα μπορούσαν να προστεθούν στην εφαρμογή είναι:

- ***Περισσότερος καθαρισμός του κώδικα:*** Καθώς αφιερώνουμε χρόνο σε μια καριέρα, αποκτάμε ολοένα και περισσότερες γνώσεις επάνω στο τομέα απασχόλησής μας. Για τους προγραμματιστές αυτό σημαίνει ότι μπορούμε να εφαρμόσουμε περισσότερες δυνατότητες στην εφαρμογή, κρατώντας παράλληλα έναν πιο ευανάγνωστο (και πιθανώς μικρότερο σε όγκο) κώδικα. Αν μου δινόταν η ευκαιρία να εκτελέσω κάποια αλλαγή στην εφαρμογή, η πρώτη ενέργειά μου θα ήταν να καθαρογράψω ό,τι είχα γράψει από την αρχή.

- ***Καλύτερο και πιο όμορφο interface:*** Όσο λειτουργική και αν είναι μια εφαρμογή, ένας χρήστης ή μια εταιρία θα προτιμήσει να χρησιμοποιήσει πιο εύκολα μια εφαρμογή που έχει καλύτερη παρουσίαση από άλλες όμοιές της. Αυτό που θα έκανα στη συνέχεια, μετά τον καθαρισμό του κώδικα, θα ήταν να επικεντρωθώ περισσότερο στο CSS της εφαρμογής και να δώσω μια καλύτερη παρουσίαση στην εφαρμογή.

- ***Προσθήκη περισσότερων γλωσσών:*** Μια εφαρμογή που μπορεί να λειτουργήσει για κάθε Barcode που υπάρχει, θα πρέπει να μπορεί να υποστηρίξει τουλάχιστον τα Αγγλικά. Γι' αυτό θα πρόσθετα την επιλογή υποστήριξης πολλαπλών γλωσσών στην εφαρμογή.

- ***Διαχείριση διαφορετικών ειδών Barcodes:*** Ανεξάρτητα από το γεγονότος ότι η εταιρία που χρησιμοποιεί αυτή τη στιγμή την εφαρμογή, η HELEXPO ΔΕΘ, χρησιμοποιεί μόνο Barcodes CODE_128, θα πρέπει να υπάρχει η επιλογή να αλλάζει ο στόχος της ανάγνωσης για την κάμερα, καθώς είναι πιθανό κάποια στιγμή να γίνουν αλλαγές στα εισιτήρια. Εφόσον υπάρχει η δυνατότητα αποθήκευσης των ρυθμίσεων της εφαρμογής, θα ήταν εύκολη η προσθήκη μιας τέτοια επιλογής.

- ***Διαχείριση Barcodes τοπικά / offline:*** Καθώς τα Barcodes αναγράφουν όλες τις απαραίτητες πληροφορίες μέσα τους, θα μπορούσε θεωρητικά να γίνει η ταυτοποίηση μέσω της εφαρμογής τοπικά και χωρίς της ανάγκης σύνδεσης σε κάποιο δίκτυο. Το πρόβλημα που δημιουργείται βέβαια στην πράξη είναι η ανάγνωση του ίδιου Barcode από δύο διαφορετικές συσκευές οι οποίες, εφόσον δεν επικοινωνούν, δεν μπορούν να ξέρουν εάν αυτό ακυρώθηκε νωρίτερα ή όχι.

ΔΙΑΦΟΡΕΤΙΚΟΙ ΤΡΟΠΟΙ ΥΛΟΠΟΙΗΣΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Αξίζει επίσης να σημειωθεί ότι υπάρχουν και άλλοι τρόποι υλοποίησης της εφαρμογής. Η εφαρμογή θα μπορούσε να είχε γίνει είτε με Java, ως native Android εφαρμογή είτε με τη χρήση του Xamarin, ενός άλλου Framework για τη συγγραφή cross-platform εφαρμογών, με τη χρήση C# για το code-behind και XAML για την υλοποίηση του UI.

Αν και οι δύο αυτοί τρόποι θα υλοποιούσαν μια πιο stable εφαρμογή, το Cordova προτιμήθηκε καθώς υπάρχει περισσότερη εξοικείωση προγραμματιστών με HTML, JavaScript και CSS παρά με τα υπόλοιπα και άρα η εφαρμογή μπορεί να συντηρείται ευκολότερα από τους προγραμματιστές που θα δουλέψουν επάνω στην εφαρμογή.

Επιπλέον θα μπορούσε να γίνει η χρήση άλλων Framework παράλληλα με το Cordova όπως το Ionic, το οποίο χρησιμοποιεί AngularJS. Το AngularJS είναι ένα Framework για JavaScript και μπορεί να κάνει databind δεδομένα μέσα από HTML.

ΕΠΙΛΟΓΟΣ ΤΗΣ ΕΡΓΑΣΙΑΣ

ΣΥΜΠΕΡΑΣΜΑΤΑ

Με τη χρήση των tablet υπάρχει κέρδος από διάφορες απόψεις. Από χρηματική άποψη, τα tablet κοστίζουν λιγότερο για την ίδια απόδοση. Από ζήτημα χρόνου, εγκατάστασης και μετακίνησης, όπως επίσης και φόρτου, τα tablet συμφέρουν λόγω του μικρού τους όγκου. Από θέμα λειτουργιών, τα tablet έρχονται

ενσωματωμένα με μια κάμερα που επεκτείνει τη λειτουργεία όσων έκαναν οι υπολογιστές σε συνδυασμό με τα barcode readers.

Αυτό φυσικά δε σημαίνει την ολική αντικατάσταση των υπολογιστών από τα tablet. Με τη χρήση των tablets, οι υπολογιστές μπορούν να διατίθενται για πιο πολύπλοκες εργασίες ή διεργασίες που τα tablet δεν μπορούν να υλοποιήσουν, όπως η έκδοση εισιτηρίων μέσω ειδικών εκτυπωτών.

ΔΑΠΑΝΩΜΕΝΟΣ ΧΡΟΝΟΣ ΤΗΣ ΕΡΓΑΣΙΑΣ

Το project εκπονήθηκε τον προτελευταίο μήνα της πρακτικής μου άσκησης στη HELEXPO ΔΕΘ, τον Αύγουστο του 2017. Αξίζει βέβαια να σημειωθεί ότι εφαρμόστηκαν πολλές από τις γνώσεις που απέκτησα κατά τη διάρκεια της πρακτικής.

Όσον αφορά το γραπτό κομμάτι της εργασίας, η διάρκεια εκπόνησής του υπήρξε μεγαλύτερη, από το τέλος Νοεμβρίου του 2017 μέχρι και τον Ιανουάριο του 2018.

ΠΡΟΣΩΠΙΚΕΣ ΒΕΛΤΙΩΣΕΙΣ

Όπως προαναφέρθηκε, αυτή η εργασία αποτελεί την πρώτη εφαρμογή που συνέγραψα και διέθεσα για τη χρήση κάποιας εταιρίας. Εδώ ξεκίνησε η εισαγωγή μου στο χώρο των επιχειρήσεων και στη λογική πίσω από το τρόπο συγγραφής μιας εφαρμογής, όπως επίσης και για τη συνέπεια που χρειάζεται να δείχνεται.

Ένα ακόμα σημαντικό μάθημα που πήρα είναι η προσαρμογή της εφαρμογής που απαιτείται στις ανάγκες του εργοδότη, όπως επίσης και της ανάγκης τροποποίησής της, όπου παρουσιάζεται ανάγκη.

Επιπλέον απέκτησα εμπειρίες στην εργασία πάνω σε mobile συστήματα (κυρίως στο Android) και πιο συγκεκριμένα στις ιδιομορφίες που απαιτεί το καθένα από αυτά.

Τέλος, δουλεύοντας στο γραπτό κομμάτι της εργασίας, έκανα μια παραπάνω έρευνα στις τεχνολογίες που χρησιμοποίησα και επανεξέτασα τον κώδικα που έγραψα. Μέσα από αυτό έμαθα δυνατότητες τις οποίες δε γνώριζα κατά τη συγγραφή της εφαρμογής. Η έρευνα αυτή με ενέπνευσε να σκεφτώ περισσότερες δυνατότητες που θα μπορούσα να προσθέσω στην εφαρμογή, ενώ αυτό με προκάλεσε να αναθεωρήσω μερικές αντιλήψεις που είχα για την συγγραφή του κώδικα και της λογικής πίσω από αυτόν. Έτσι, μπορώ να πω με βεβαιότητα ότι η συγγραφή αυτής της εργασίας με βελτίωσε στο τομέα του προγραμματισμού.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΒΙΒΛΙΟΓΡΑΦΙΑ

Chris Woodford (2017) Barcodes and Barcode Scanners
http://www.explainthatstuff.com/barcodescanners.html

Cordova Website https://cordova.apache.org/

Gajotres (2014) Page events order in jQuery Mobile – Version 1.4 update
https://www.gajotres.net/page-events-order-in-jquery-mobile-version-1-4-update/

GitHub Website https://github.com/

jQuery Mobile Website https://jquerymobile.com

jQuery Website https://jquery.com/

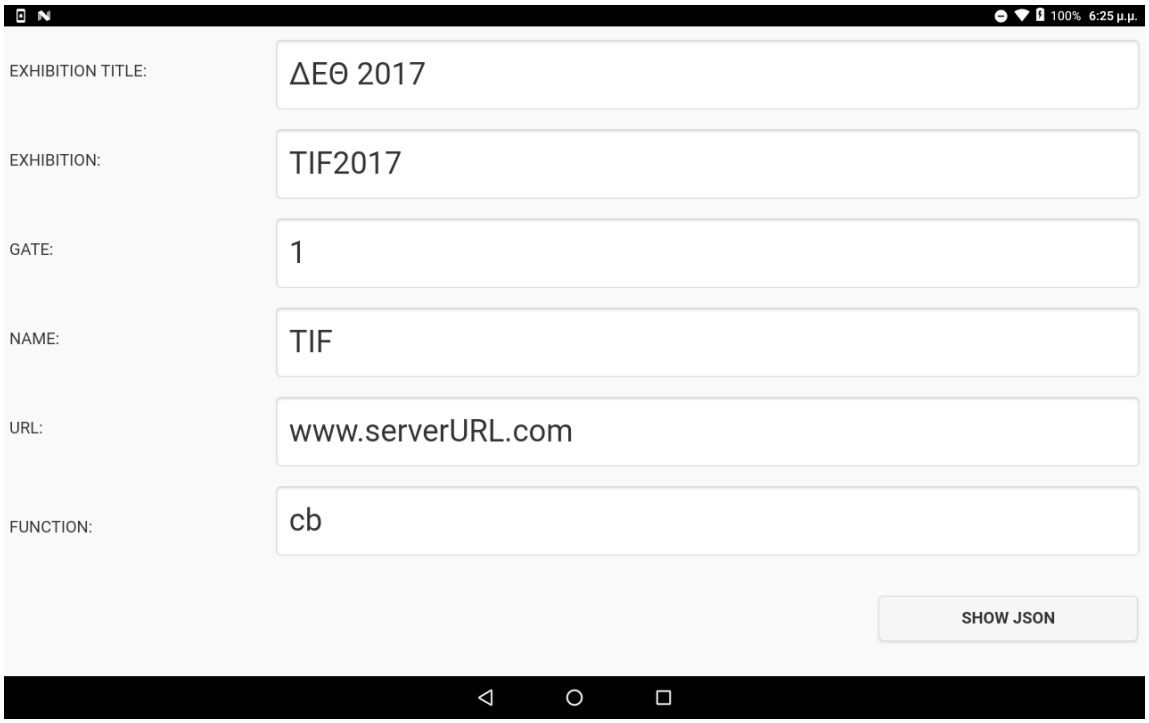
SQLite Website https://www.sqlite.org/

Stack Overflow Website https://stackoverflow.com/

W3schools Website https://www.w3schools.com/

Wikipedia https://en.wikipedia.org

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ



Εικόνα 66: Settings_page 1

Κατά τη πρώτη εκκίνηση ανοίγει η σελίδα των ρυθμίσεων. Εδώ ο χρήστης πρέπει να γεμίσει τουλάχιστον τα απαραίτητα πεδία των ρυθμίσεων της εφαρμογής.

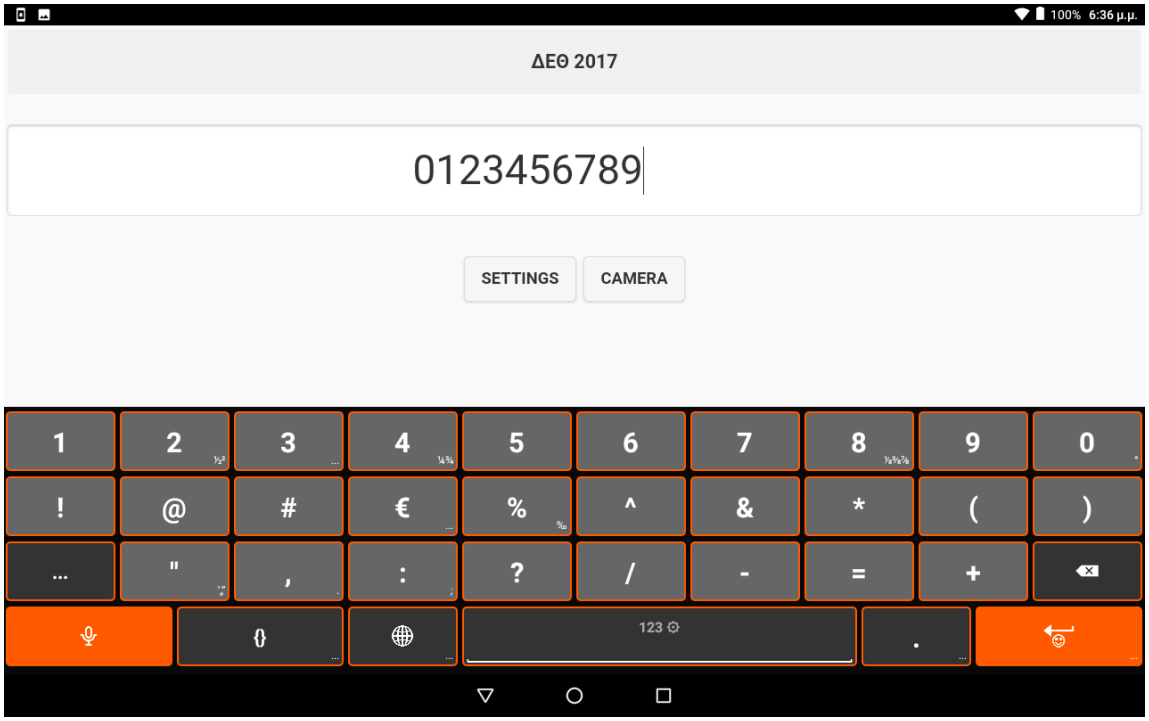
Σημειώνεται ότι κατά την πρώτη εκκίνηση της εφαρμογής απαιτήται η σύνδεση της συσκευής με το internet. Σε περίπτωση που δεν υπάρξει σύνδεση, δε θα κληθεί η CDN υπηρεσία του jQuery Mobile και η εφαρμογή δε θα λειτουργήσει σωστά.

Αφού ο χρήστης εισάγει τα απαραίτητα πεδία για την επικοινωνία με τον server, μπορεί να πατήσει το κουμπί εμφάνισης της απάντησης του server για να εκτελέσει μια δοκιμαστική αποστολή. Έτσι μπορεί να παρατηρήσει εάν τα στοιχεία που εισήγαγε είναι σωστά.

Εικόνα 67: Settings_page 2

Επιπλέον ενεργοποιείται το κουμπί αποθήκευσης. Με το πάτημα του κουμπιού, οι ρυθμίσεις αποθηκεύονται και ανοίγει η σελίδα των Barcodes.

Στη περίπτωση που υπάρχει κάποιο διαθέσιμο Barcode Reader, συνιστάται η σύνδεση του τώρα στη συσκευή. Απαιτείται επιπλέον η ύπαρξη ενός θηλυκού USB καλωδίου που καταλήγει σε αρσενικό καλώδιο που μπορεί να δεχτεί η συσκευή.



Εικόνα 68: Barcode_page πριν αποστολής

Εδώ μπορεί να γίνει εισαγωγή ενός Barcode τη φορά μέσα στο πλαίσιο στο κέντρο της σελίδας. Η αποστολή του Barcode στο server γίνεται αυτόματα στην περίπτωση που το Barcode διαβαστεί μέσω ειδικού μηχανήματος ή της κάμερας, αλλιώς χρειάζεται να πατηθεί το “ENTER” στο αναδυόμενο πληκτρολόγιο.

Εικόνα 69: Barcode_page μετά αποστολής

Με την επιστροφή της απάντησης από τον server, εμφανίζεται πάνω από το πλαίσιο, το μήνυμα που επέστρεψε ο server. Σε περίπτωση επικύρωσης του Barcode, εμφανίζεται το μήνυμα σε πράσινο, αλλιώς το μήνυμα εμφανίζεται σε κόκκινο.

Εικόνα 70: Scan με Camera

Εάν ο χρήστης επιθυμεί να ανοίξει την κάμερα της εφαρμογής ή να αλλάξει κάποιες από τις ρυθμίσεις που έχει ήδη δώσει, πρέπει μόνο να πατήσει το κουμπί της κάμερας ή των ρυθμίσεων αντίστοιχα για κάθε περίπτωση.

Στη περίπτωση αλλαγής των ρυθμίσεων, ενεργοποιείται το κουμπί της ακύρωσης, το οποίο δίνει τη δυνατότητα διαγραφής όλων των αλλαγών, ώστε να επιστρέψει η εφαρμογή στις προηγούμενες ρυθμίσεις της.