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

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Η χρήση ανοιχτών δεδομένων και η ανάπτυξη εφαρμογής ειδοποίησης για λεωφορεία ΟΑΣΘ σε κινητό τηλέφωνο.»

Του φοιτητή

Καλούδη Αναστάσιου

Αρ. Μητρώου: 05/2802

Επιβλέπων καθηγητής

Κ. Χρήστος Κουρουπέτρογλου

Θεσσαλονίκη Σεπτέμβριος 2013

ΠΡΟΛΟΓΟΣ

Μία από τις δυνατότητες που σήμερα έχει βοηθήσει στην ραγδαία εξέλιξη των εφαρμογών σε κινητά αλλά και στο web είναι η δυνατότητα που προσφέρουν ορισμένοι οργανισμοί και δικτυακοί τόποι να χρησιμοποιηθούν δεδομένα από τις εφαρμογές τους. Αυτό δίνει την δυνατότητα ανάπτυξης επιπλέον εφαρμογών από τρίτους που εξυπηρετούν επιπλέον ανάγκες.

Η παρούσα πτυχιακή εργασία έχει ως θέμα την ανάπτυξη εφαρμογής για έξυπνα κινητά τηλέφωνα (Smartphones) με λειτουργικό σύστημα Android, που θα αξιοποιεί δεδομένα του ΟΑΣΘ σε σχέση με την απόσταση που βρίσκονται λεωφορεία από συγκεκριμένη στάση. Η εφαρμογή αυτή με όνομα OasthAlarm έχει σκοπό την ειδοποίηση του χρήστη όταν ένα λεωφορείο πλησιάζει σε μία στάση προκειμένου να προσέλθει έγκαιρα σε αυτήν.

Η διάθεσή των δεδομένων του ΟΑΣΘ αποδείχτηκε εξαιρετικά δύσκολη υπόθεση, έτσι οι εναλλακτικές λύσεις ήταν μονόδρομος. Η ανάκτηση των ζητούμενων δεδομένων έγινε μέσω της ιστοσελίδας m.oasth.gr και με τη βοήθεια της βάσης των δρομολογίων και των στάσεων από τον ανοιχτό κώδικα μίας εφαρμογής του ΟΑΣΘ η οποία δημιουργήθηκε από ανεξάρτητο προγραμματιστή.

Η ανάπτυξη της εφαρμογής αναπτύχθηκε από τον φοιτητή Καλούδη Αναστάσιο στα πλαίσια ολοκλήρωσης των σπουδών του στο τμήμα Πληροφορικής της σχολής Τεχνολογικών Εφαρμογών του Αλεξάνδρειου Τεχνολογικού Ιδρύματος Θεσσαλονίκης, υπό την εποπτεία του καθηγητή κ. Χρίστου Κουρουπέτρογλου.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία εστιάζει στην μελέτη του λειτουργικού συστήματος Android, για την ανάπτυξη εφαρμογών σε φορητές συσκευές . Στα πλαίσια της εργασίας αναπτύχθηκε μία εφαρμογή για την ειδοποίηση των επιβατών των λεωφορείων του ΟΑΣΘ προκειμένου να τους ενημερώνει για την έγκαιρη προσέλευση τους στη στάση πριν την άφιξη του λεωφορείου.

Στο πρώτο κεφάλαιο γίνεται μία ιστορική αναδρόμη της δημιουργίας του Android και των εκδόσεων του μέσα στα χρόνια που ακολούθησαν. Στη συνεχεια αναλύεται η αρχιτεκτονική του λειτουργικού Android. Τέλος γίνεται αναφορά στα βασικά στοιχεία μιας εφαρμογής και τεχνολογιών που ενσωματώνει όπως είναι οι βάσεις δεδομένων.

Στο δεύτερο κεφάλαιο αναλύονται εκτεταμένα τα Ανοιχτά Δεδομένα , οι αρχές τους και παρουσιάζονται τα πλεονεκτήματα όταν είναι ανοικτά. Επίσης περιγράφονται ο Σημασιολογικός Ιστός και τα Συνδεδεμένα Δεδομένα και η σημαντικότητα τους για την ανάπτυξη εφαρμογών από τρίτους. Τέλος παρουσιάζεται η ενδεχόμενη αξιοποίηση τους συγκεκριμένα, για εφαρμογές που ασχολούνται με τα ΜΜΜ.

Στο τρίτο κεφάλαιο αναφέρονται οι απαιτήσεις της εφαρμογής και τα προβλήματα που αντιμετωπίστηκαν κατά τη διάρκεια της δημιουργίας της. Στη συνέχεια του κεφαλαίου παρουσιάζεται η εφαρμογή μέσω σεναρίων και εικόνων της.

Στο τελευταίο κεφάλαιο γίνεται πλήρης ανάλυση του κώδικα Java που αναπτύχθηκε για την υλοποίηση των λειτουργιών της εφαρμογής και των αρχείων XML που χρησιμοποιούνται για το γραφικό περιβάλλον χρήστη.

ABSTRACT

The present paper focuses on the study of the operating system Android and how it promotes the development of applications on portable equipment. For the completion of this research, an application was developed, which would notify in time the passengers of OASTH for the upcoming arrivals of buses at specific station.

In the first chapter we make a historical flashback on the development of Android and the numerous versions generated throughout time. Next we analyze the architecture of this operating system. Finally, we make a report of the basic elements of an application and the technology it incorporates like the databases.

In the second chapter we make a thorough analysis of the Open Data and their principles while presenting their advantages when they are open. Furthermore, the Semantic Web and the Linked Data are being discussed along with their efficiency when constructing applications by third parties. Finally, we present the probability to utilize them from applications dealing with the public transportation.

In the third chapter, we mention the requirements of the application and the difficulties faced during its creation. Then, the application is presented through different scenarios and images.

Finally, in the last chapter we make a full analysis of the Java code developed for the implementation of the application's functions and the files XML used for the Graphic User Environment (GUI).

ΕΥΧΑΡΙΣΤΙΕΣ

Καταρχάς θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της πτυχιακής μου εργασίας κ. Χρήστο Κουρουπέτρογλου για την καθοδήγηση του σε όλες τις φάσεις της δημιουργίας της. Στη συνέχεια τον κ. Παρασκευαιδη Ιάκωβο για το σχεδιασμό του εικονιδίου και του background της εφαρμογής. Τέλος θα ήθελα να εκφράσω την ευγνωμοσύνη μου για την οικογένεια μου που όλα αυτά τα χρόνια μου συμπαραστέκονται ηθικά και οικονομικά και διαμορφώνουν γύρω μου ένα άνετο περιβάλλον μέσα στο οποίο μπορώ να εργαστώ και να επεκτείνω τις γνώσεις μου.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ANDROID

1.1 Ιστορική Αναδρομή

Η Ιστορία του Android ξεκίνησε στο Palo Alto της Καλιφόρνια των Η.Π.Α τον Οκτώβριο του 2003 . Ο Andy Rubin , ο Rich Miner , ο Nick Sears και ο Chris White δημιουργούν την Android Inc. με σκοπό να αναπτύξουν ένα ανεπτυγμένο λειτουργικό σύστημα για ψηφιακές μηχανές . Συνειδητοποίησαν γρήγορα όμως ότι η συγκεκριμένη αγορά ήταν περιορισμένη εκείνη τη στιγμή και επαναπροσδιόρισαν το σκοπό τους ,έτσι ξεκίνησαν να αναπτύσσουν ένα λειτουργικό σύστημα για smartphone για να χτυπήσουν τους ανταγωνιστές τους εκείνη τη στιγμή το Symbian και το Windows Mobile (το iphone της Apple δεν είχε κυκλοφορήσει ακόμα).

Στις 17 Αυγούστου του 2005 η Google εξαγοράζει την άσημη ακόμα Android Inc , κρατώντας τα βασικά στελέχη της Android Inc (ανάμεσα τους ήταν ο Rubin , ο Miner κι ο White), κάνοντας έτσι την είσοδο της στην Αγορά των κινητών τηλεφώνων . Αμέσως μετά μια ομάδα της Google με αρχηγό τον Rubin δημιουργεί μια πλατφόρμα λογισμικού για κινητά υποστηριζόμενη με τον Linux Kernel.

Στις 5 Νοεμβρίου του 2007 ιδρύεται η OHA (OpenHandsetAlliance) μια σύμπραξη 34 εταιριών λογισμικού κατασκευής hardware και τηλεπικοινωνιών αφοσιωμένες στην ανάπτυξη καινούργιων τεχνολογιών για κινητές συσκευές ,με τον ηγετικό ρόλο της Google. Βασική προϋπόθεση συμμετοχής μια εταιρείας στην OHA είναι να παράγει μόνο κινητά τηλέφωνα συμβατά με τις εκδόσεις του Android. Την ίδια μέρα αποκαλύπτεται για πρώτη φορά στο ευρύτερο κοινό η Android και το πρώτο της προϊόν μια πλατφόρμα λογισμικού για κινητά κατασκευασμένη σε Linux Kernel 2.6. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της ApacheLicense, μιας ελεύθερης άδειας λογισμικού.

Το πρώτο κινητό που «έτρεχε» με Android ήταν το HTC Dream , το οποίο κυκλοφόρησε στις 22 Οκτωβρίου του 2008. Η επιτυχία του Android ως πλατφόρμα κινητών τηλεφώνων θα εξαρτηθεί κατά ένα μεγάλο μέρος από την επιτυχία των συνεργατών της OHA στην κυκλοφορία επιθυμητών κινητών τηλεφώνων και κινητών υπηρεσιών που θα ενθαρρύνουν την υιοθέτηση των Android τηλεφώνων Οι προγραμματιστές έχουν την ευκαιρία να δημιουργήσουν καινοτόμες, νέες εφαρμογές κινητών για Android ώστε να ενθαρρυνθούν περισσότερες επιχειρήσεις κινητής τεχνολογίας να γίνουν μέλη της OHA

1.2 Εκδόσεις

Οι εκδόσεις του λειτουργικού συστήματος Android ξεκίνησαν με την κυκλοφορία της πρώτης beta έκδοσης το Νοέμβριο του 2007. Η πρώτη εμπορική έκδοση (Android 1.0) κυκλοφόρησε τον Σεπτέμβριο του 2008. Το λειτουργικό σύστημα του Android που δημιουργήθηκε από την Google και την HandsetAlliance έχει δει βασικές ενημερώσεις από τη στιγμή της πρώτης κυκλοφορίας του. Οι ενημερώσεις αυτές διορθώνουν τυχόν προηγούμενα προβλήματα και προσθέτουν νέα χαρακτηριστικά. Από τον Απρίλιο του 2009 κάθε καινούργια έκδοση είχε μια κωδική ονομασία βασισμένη σε γλυκό – επιδόρπιο με αλφαβητική σειρά: Cupcake, Donut, Éclair, Froyo (Frozenyogurt), Gingerbread, Honeycomb, IceCreamSandwich και προσφάτως JellyBean. Οι δύο πρώτες εκδόσεις ονομάστηκαν μεταγενέστερα Astro και Bender αντίστοιχα αλλά δεν μπορούν να χρησιμοποιηθούν επίσημα για λόγους πνευματικής ιδιοκτησίας. Η τελευταία ενημέρωση του λειτουργικού ήταν η JellyBean v4.3 η οποία κυκλοφόρησε τον Ιούλιο του 2013.

ΕΚΔΟΣΗ	**API**	**ΟΝΟΜΑΣΙΑ**
1.5	3	Cupcake
1.6	4	Donut
2.0-2.1	5-7	Éclair
2.2	8	Froyo
2.3	9,10	Gingerbread
3.0-3.2	11-13	Honeycomb
4.0	14,15	IceCreamSandwich
4.1-4.3	16-18	Jelly Bean

Εικόνα 1.1 Πίνακας Εκδόσεων Android

Βασικές αναβαθμίσεις

- Η **Éclair** είναι η έκδοση 2.0/2.1, έφερε αλλαγές στο περιβάλλον και έφερε υποστήριξη HTML5 στον περιηγητή (browser).

- Η **FroYo (FrozenYogurt)** είναι η έκδοση 2.2, έδωσε μεγαλύτερη ταχύτητα και έφερε υποστήριξη flash μαζί με δυνατότητα Wi-Fihotspot.

- Η **Gingerbread** είναι η έκδοση 2.3 και βελτίωσε το περιβάλλον όπως και δυνατότητες για πιο "βαριές" εφαρμογές, επιπλέον πρόσθεσε υποστήριξη NFC (NearFieldCommunication).

- Η **Honeycomb** είναι η έκδοση 3.0/3.2 αποκλειστικά για ταμπλέτες, έφερε αλλαγές κυρίως στο γραφικό περιβάλλον και πρόσθεσε υποστήριξη πολλαπλών πυρήνων μαζί με βελτιωμένα γραφικά.

- Η **IceCreamSandwich**δηλαδή η έκδοση 4.0 έχει σκοπό να "ενώσει" τις εκδόσεις για ταμπλέτες και κινητά και να προσθέσει υποστήριξη για την Google TV.

- Η**JellyBean** είναι η τελευταία έκδοση του Android (4.1-4.3) και εκτός από ταχύτατη πλοήγηση υπόσχεται και μεγάλες αλλαγές στον τομέα της τεχνητής νοημοσύνης.

Η πρώτη έκδοση του Android κυκλοφόρησε χωρίς πρακτικά να υπάρχει κινητή συσκευή που να το χρησιμοποιεί. Ο σκοπός ήταν να αποδειχθεί ότι δεν χρειάζεται συσκευή για να ξεκινήσει κάποιος την ανάπτυξη εφαρμογών. Υπάρχουν μερικοί περιορισμοί όσον αφορά το υλικό (πχ αισθητήρας βαρύτητας) αλλά κατά το μεγαλύτερο κομμάτι περιλαμβάνει ότι χρειάζεται ο προγραμματιστής για να αναπτύξει εφαρμογές σε αυτό.

1.3 Βασικά χαρακτηριστικά και δυνατότητες

Το Android είναι ένα πακέτο λογισμικού για κινητές συσκευές που περιλαμβάνει ένα λειτουργικό σύστημα, εφαρμογές διαχείρισης hardware και μερικά πολύ σημαντικά προγράμματα για τον χρήστη. Ο όρος Android είναι ελληνικής προέλευσης καθώς προέρχεται από τη λέξη Andro-«ανθρώπινη» και eides-«μορφή, σχήμα». Η έννοια που δίνεται στη λέξη Android είναι το «Ανδροειδές» και συμβολίζεται σαν ένα ρομπότ με ανθρώπινη μορφή, το οποίο είναι και το λογότυπο του λειτουργικού. Το πακέτο ανάπτυξης λογισμικού για το Android (AndroidSDK) προσφέρεται δωρεάν και παρέχει στον προγραμματιστή πρόσβαση στις διεπαφές προγραμματισμού εφαρμογών (APIs) και τα απαραίτητα εργαλεία ώστε να μπορεί να ξεκινήσει την ανάπτυξη εφαρμογών για την πλατφόρμα του Android χρησιμοποιώντας την γλώσσα προγραμματισμού Java.

Οι συσκευές στις οποίες είναι εγκαταστημένο λογισμικό android είναι αρκετά ισχυρές από άποψη λογισμικού (επεξεργαστές , μνήμες κλπ) και κατά συνέπεια προσφέρουν πολλές δυνατότητες. Μερικές από αυτές τις δυνατότητες είναι :

- **Ενσωματωμένος Internet browser** βασισμένος στη μηχανή WebKit.
- **Σχεσιακή βάση SQLite** για δομημένη αποθήκευση δεδομένων
- **Πλαίσιο εφαρμογών που επιτρέπει την επαναχρησιμοποίηση και αντικατάσταση** στοιχείων/εξαρτημάτων
- **Πλούσιο προγραμματιστικό περιβάλλον συμπεριλαμβανομένου ενός προσομοιωτή συσκευής και εικονικής μηχανής**, εργαλεία αποσφαλμάτωσης, προφίλ μνήμης και απόδοσης και ένα πρόσθετο για το EclipseIDE.
- **Βελτιστοποιημένα 3D γραφικά** βασισμένα στην OpenGLES 1.0 (προαιρετική υποστήριξη επιτάχυνσης υλικού)
- **Υποστήριξη πολυμέσων για** κοινές μορφές ήχου, βίντεο και εικόνων (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **Υποστήριξη τηλεφωνικού δικτύου GSM και 3G ή 4G** ανάλογα το hardware της συσκευής.
- **Υποστήριξη τεχνολογιών συνδεσιμότητας Bluetooth, EDGE, 3G, NFC και Wi-Fi**
- **Κάμερα, GPS, πυξίδα και επιταχυνσιόμετρο** ανάλογα το hardware της συσκευής.

1.4 Αρχιτεκτονική του λειτουργικού συστήματος

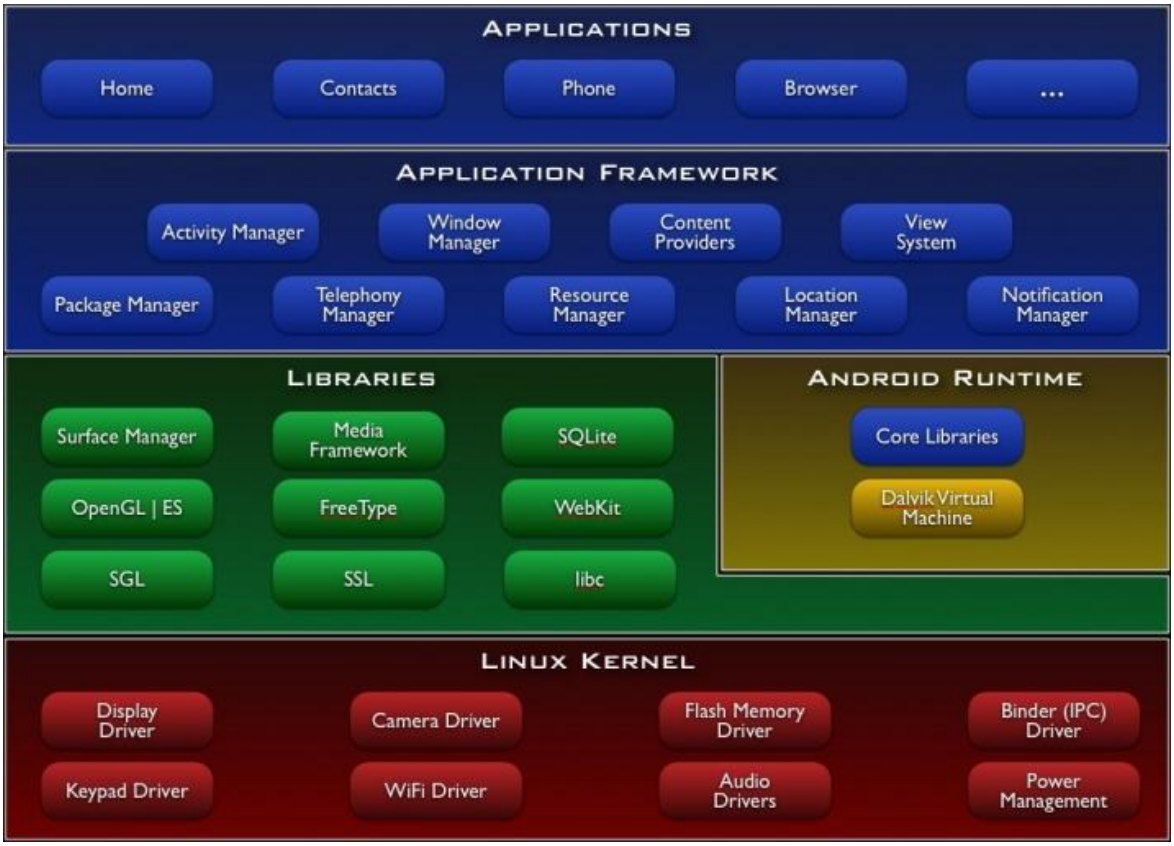
Το λειτουργικό σύστημα του Android, μπορεί να αναφερθεί και ως μια στοίβα λογισμικού από διαφορετικές στιβάδες, όπου η κάθε στιβάδα διαθέτει μία ή δύο ομάδες προγραμμάτων για τη σωστή του λειτουργία. Όλες μαζί αποτελούν το λειτουργικό σύστημα, το ενδιάμεσο λογισμικό και τις σημαντικές εφαρμογές. Κάθε στιβάδα της αρχιτεκτονικής (Εικόνα) παρέχει διαφορετικές υπηρεσίες στην ακριβώς επάνω της. Επιγραμματικά, οι στιβάδες του Android από κάτω προς τα πάνω είναι οι παρακάτω:

1. **Ο πυρήνα της Linux**, που περιλαμβάνει οδηγούς (drivers) για να τρέξει το σύστημα

2. **Οι εγγενής βιβλιοθήκες και η ομάδα του χρόνου εκτέλεσης**

3. **Το πλαίσιο εφαρμογών**, και στο τελευταίο επίπεδο,

4. **Οι Εφαρμογές**, οι οποίες είναι γραμμένες σε Java και εκτελούνται στην εικονική μηχανή Dalvik



Εικόνα 1.2 Αρχιτεκτονική Android

Linux Kernel

Η βάση της στοίβας λογισμικού είναι ο Kernel (πυρήνας) της Linux .Η πρώτη έκδοση που χρησιμοποιήθηκε ήταν η 2.6 και τώρα στο jellybean έχει φτάσει στην έκδοση 3.4. Περιέχει τους Drivers τους οποίους χρειάζεται για να τρέξει το σύστημα, όπως της οθόνης, της κάμερας κ.α. Το Android χρησιμοποιεί τον πυρήνα Linux για τη διαχείριση μνήμης, τη διαχείριση διεργασιών, τη δικτύωση και άλλες υπηρεσίες του λειτουργικού συστήματος. Ο πυρήνας του Android μπορεί να βασίζεται στον πυρήνα του Linux, αλλά διαφέρει αρκετά από αυτόν. Ο λόγος είναι οι αλλαγές στην αρχιτεκτονική που έχει κάνει η Google για να είναι ελαφρύτερος και βελτιστοποιημένος για χρήση σε κινητές συσκευές.

Native Libraries

Οι Εγγενείς Βιβλιοθήκες βρίσκονται στο αμέσως υψηλότερο επίπεδο και περιλαμβάνουν όλο τον κώδικα που περιέχει το Android OS. Είναι γραμμένες στις γλώσσες C ή C++ και δε μπορούν να τρέξουν ξεχωριστά μόνες τους σαν εφαρμογές, αλλά μπορούν να κληθούν μέσω ενός κατάλληλου interface της java. Μερικές από τις πιο σημαντικές είναι : η SQLite που είναι υπεύθυνη για την αποθήκευση δεδομένων εφαρμογών, η Webkit που παρέχει λειτουργίες για το διαδικτυακή περιήγηση και η Media Framework που περιέχει αποκωδικοποιητές (codecs) για την αναπαραγωγή πολυμέσων. Από την έκδοση Donut (1.6) και μετά, οι προγραμματιστές έχουν τη δυνατότητα να γράφουν τις δικές τους τέτοιες βιβλιοθήκες με τη χρήση της εργαλειοθήκης NDK (NativeDevelopmentKit).

Android Runtime

Στο ίδιο επίπεδο με τις βιβλιοθήκες, το Android Runtime (Χρόνος Εκτέλεσης) παρέχει ένα σύνολο βασικών βιβλιοθηκών που επιτρέπουν στους προγραμματιστές να γράψουν εφαρμογές χρησιμοποιώντας τη JAVA. Όπως στη Java υπάρχει η λεγόμενη Java Virtual Machine στην οποία εκτελείτε ο κώδικας bytecode των εφαρμογών, στο Android υπάρχει κάτι παρόμοιο και δεν είναι άλλο από την εικονική μηχανή Dalvik. Η Dalvik είναι μια εξειδικευμένη εικονική μηχανή, ειδικά διαμορφωμένη για κινητές συσκευές που έχουν περιορισμένη μνήμη και ισχύ.

Η DalvikVM χρησιμοποιεί τον πυρήνα Linux της συσκευής για να χειριστεί τις χαμηλού επιπέδου λειτουργίες που περιλαμβάνουν την ασφάλεια, τον πολυνηματισμό και τη διαχείριση διαδικασιών και μνήμης. Είναι επίσης δυνατό να γραφτούν εφαρμογές C/C++ που τρέχουν άμεσα στο εσωτερικό του λειτουργικού. Αν και μπορεί να γίνει αυτό, στις περισσότερες περιπτώσεις δεν υπάρχει κανένας λόγος.

Μέσω της Dalvik επιτυγχάνεται η ρύθμιση της πρόσβασης στο υλικό και στις υπηρεσίες του συστήματος. Με τη χρησιμοποίηση αυτής της εικονικής μηχανής

στην εκτέλεση εφαρμογής, η οποία προσφέρει ένα αφαιρετικό στρώμα, οι κατασκευαστές δεν χρειάζεται να ανησυχήσουν για κάποια υλοποίηση υλικού (hardwareimplementation).

Η Dalvik εκτελεί τα Dalvik εκτελέσιμα αρχεία των οποίων η μορφή είναι βελτιστοποιημένη ώστε να καταλαμβάνουν την ελάχιστη δυνατή μνήμη. Τα .dex εκτελέσιμα δημιουργούνται μετασχηματίζοντας κλάσεις που έχουν μεταγλωττιστεί από την Java χρησιμοποιώντας εργαλεία που παρέχονται μέσα στο SDK.

Application Framework

Το Android παρέχει στους developers μια ανοιχτού κώδικα πλατφόρμα ανάπτυξης και τη δυνατότητα να αναπτύξουν με αυτή ιδιαίτερα καινοτόμες και πλούσιες σε υλικό εφαρμογές. Οι developers έχουν στην διάθεση τους τη δυνατότητα ελέγχου του υλικού της συσκευής και μέσω αυτής μπορούν να αποκτήσουν πρόσβαση σε εκτέλεση διεργασιών παρασκηνίου, και πάρα πολλές ακόμη δυνατότητες οι οποίες βασίζονται στα APIs που είναι διαθέσιμα. Στο επόμενο επίπεδο της αρχιτεκτονικής του Android λοιπόν, συναντάμε το πλαίσιο των εφαρμογών. Οι developers έχουν πρόσβαση σε όλα τα APIs μεταξύ αυτών και στα κύρια APIs που χρησιμοποιούν οι ενσωματωμένες εφαρμογές. Η δομή των εφαρμογών είναι τέτοια που ευνοείται η επαναχρησιμοποίηση δομικών συστατικών, και επίσης επιτρέπεται η χρήση των δυνατοτήτων τις μίας εφαρμογής από άλλες εφαρμογές, βέβαια κάτω από τις προδιαγραφές ασφάλειας του Android. Τα σημαντικότερα δομικά στοιχεία του πλαισίου εφαρμογών είναι:

- *Διαχειριστής Δραστηριοτήτων – ActivityManager :* Υπεύθυνος για τον έλεγχο του χρόνου ζωής των εφαρμογών και για την διατήρηση μιας στοίβας που επιτρέπει την πλοήγηση του χρήστη σε προηγούμενες οθόνες.
- *Παροχείς Περιεχομένου – ContentProviders :* Αυτά τα αντικείμενα περιέχουν δεδομένα που μπορούν να διαμοιραστούν μεταξύ εφαρμογών.
- *Διαχειριστής Πόρων – ResourceManager :* Οι πόροι είναι οτιδήποτε υπάρχει σε ένα πρόγραμμα και δεν είναι κώδικας. Για παράδειγμα μπορεί να είναι κωδικοί χρωμάτων, αλφαριθμητικοί χαρακτήρες ή ακόμα και έτοιμα σχεδιαγράμματα οθονών φτιαγμένα σε XML, τα οποία μπορεί το πρόγραμμα να καλεί.
- *Διαχειριστής Τοποθεσίας – LocationManager :* Χρησιμοποιείται για να μπορεί να ξέρει το τηλέφωνο πού βρίσκεται ανά πάσα στιγμή.
- *Διαχειριστής Κοινοποιήσεων - NotificationManager:* Ιδανικός τρόπος για την ενημέρωση του χρήστη για γεγονότα που συμβαίνουν, διακριτικά χωρίς να διακόπτεται η εργασία του.
- *Διαχειριστής Ειδοποιήσεων - Notification Manager :* δίνει στις εφαρμογές πρόσβαση στις υπηρεσίες ειδοποιήσεων χρήστη. Τέτοιες είναι οι ειδοποιήσεις στη notification bar, τα toast μηνύματα στο κάτω μέρος της οθόνης, η δόνηση του κινητού και η ενεργοποίηση της οθόνης, κλπ

Applications

Στο πιο υψηλό επίπεδο υπάρχουν οι εφαρμογές που είναι εργοστασιακά εγκατεστημένες στις Android συσκευές (όπως τηλέφωνο,browser , SMS , μουσική κλπ), όπως επίσης και όλες οι εφαρμογές τρίτων που μπορούν να εγκατασταθούν στη συσκευή. Όλες οι εφαρμογές είναι γραμμένες σε Java και μπορούν να τρέχουν παράλληλα χωρίς η μία να επηρεάζει την άλλη.

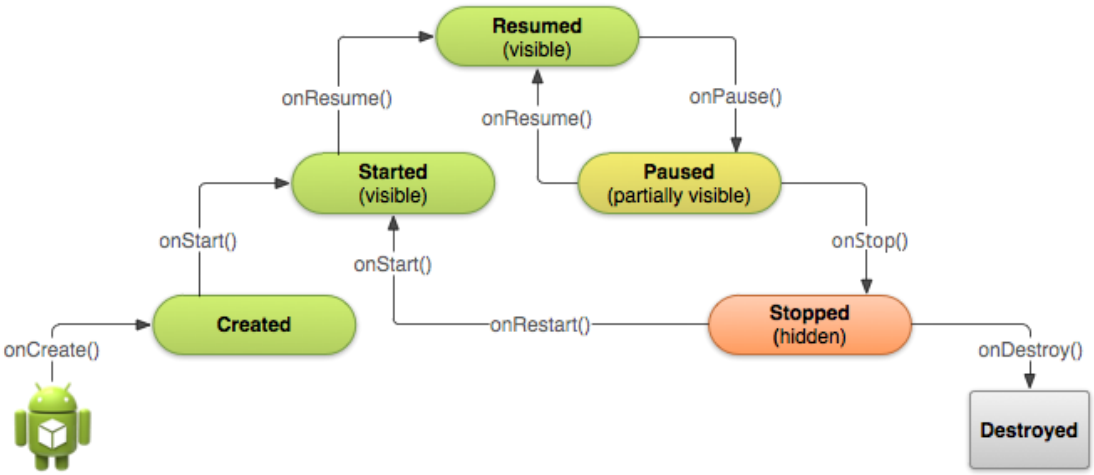
1.5 Activities

Είναι το επίπεδο παρουσίασης (presentation layer) της εφαρμογής. Μια δραστηριότητα είναι στην ουσία μια Φόρμα (Form) όπως έχουμε συνηθίσει να βλέπουμε σε εφαρμογές Η/Υ . Ένας χρήστης μπορεί να αλληλεπιδράσει με σχεδόν οποιαδήποτε Activity . Κάθε Activity υλοποιείται σαν μια κλάση που κληρονομεί την βασική κλάση android.app.Activity . Προβάλει μια διεπαφή χρήστη (userinterface) αποτελούμενη από Όψεις (Views) και ανταποκρίνεται σε Συμβάντα (Events). Μια εφαρμογή αποτελείται από πολλά Activities και ο χρήστης μπορεί να πηγαινοέρχεται σε αυτά ανάλογα με τις επιλογές που κάνει. Τα Activities είναι υπεύθυνα για κάθε οθόνη που βλέπει ο χρήστης **.**

Κύκλος ζωής μίας Activity

Όταν δημιουργείται μια Activity συμβαίνει μια καινούργια Linux διεργασία δεσμεύεται μνήμη για όλα τα αντικείμενα του γραφικού περιβάλλοντος και είναι σπατάλη να κλείνει αμέσως μόλις σταματήσει ο χρήστης να το χρησιμοποιεί. Έτσι το σύστημα αναλαμβάνει να χειριστεί τον κύκλο ζωής της, μέσω ενός «διαχειριστή δραστηριοτήτων» (Activity Manager) που αναλαμβάνει να χειριστεί τον κύκλο ζωής της.

Ο Activity Manager λοιπόν διαχειρίζεται όλες τις Activities σαν μία στοίβα και είναι υπεύθυνος για την δημιουργία, καταστροφή και τον χειρισμό τους. Μόλις ο χρήστης ανοίξει για πρώτη φορά μια νέα οθόνη, ο διαχειριστής θα δημιουργήσει το Activity και θα το προβάλει στην οθόνη. Η Δραστηριότητα μας αυτή τη στιγμή βρίσκεται στη κορυφή της στοίβας και περνάει στην εκτελέσιμη κατάσταση. Όσο ο χρήστης αλλάζει οθόνες τα προηγούμενα Activities μπαίνουν σε κατάσταση αναμονής και βρίσκονται πιο χαμηλά στη στοίβα. Η προηγούμενη Activity μένει πάντα αμέσως πιο κάτω από τη κορυφή Με τον τρόπο αυτό σε κάθε επαναφορά σε προηγούμενο Activity η διαδικασία θα γίνεται πολύ πιο γρήγορα. Οι παλαιότερες διεργασίες μπορούν να καταστραφούν από τον Activity Manager λόγω χαμηλών πόρων όσο ο χρήστης δεν τις χρησιμοποιεί για αρκετή ώρα . Ο μηχανισμός αυτός βελτιώνει την ταχύτητα του γραφικού περιβάλλοντος προσφέροντας έτσι το ζητούμενο αποτέλεσμα στο χρήστη.



Εικόνα 1.3 Κύκλος Ζωής μίας Activity

Κατάσταση εκκίνησης

Όταν ένα Activity δεν καταλαμβάνει ακόμα χώρο στη μνήμη λέγεται ότι είναι σε κατάσταση εκκίνησης. Κατά τη διάρκεια της εκκίνησης θα καλεστούν όλες οι μέθοδοι που ο προγραμματιστής θέλει να τρέξουν στην προκειμένη φάση ώστε να μεταβούμε στην κατάσταση λειτουργίας. Η μετάβαση αυτή είναι η πλέον δαπανηρή υπολογιστικά διαδικασία η οποία επηρεάζει άμεσα και την αυτονομία της μπαταρίας. Επειδή είναι πιθανό μια διεργασία να ξαναζητηθεί από το χρήστη λίγο αργότερα και έτσι «συμφέρει» να μείνει ενεργή για κάποιο διάστημα. Αυτός ακριβώς είναι ο λόγος για τον οποίο η φιλοσοφία του Android δεν επιτρέπει την καταστροφή κάθε διεργασίας μόλις εξαφανίζεται από την οθόνη.

Κατάσταση λειτουργίας

Στην κατάσταση λειτουργίας το Activity είναι εμφανισμένο στην οθόνη και αλληλεπιδρά με τον χρήστη. Σε αυτή τη κατάσταση όλες οι αλληλεπιδράσεις (όπως πληκτρολόγηση) διαχειρίζονται από το τρέχον Activity και γι’ αυτό έχει την πρώτη προτεραιότητα όσον αφορά τους πόρους που θα χρειαστεί για να δουλέψει όσο το δυνατόν γρηγορότερα. Αυτό γίνεται για να είναι το τρέχον Activity πάντα υπάκουο στις εντολές του χρήστη σε οποιαδήποτε χρονική στιγμή.

Κατάσταση παύσης

Η κατάσταση παύσης δεν είναι συνηθισμένο σενάριο καθώς αποτελεί την περίπτωση που ένα Activity είναι μεν εμφανισμένο στην οθόνη αλλά δεν αλληλεπιδρά με τον χρήστη. Αυτό συμβαίνει σπάνια γιατί, λόγω του μικρού μεγέθους οθόνης, είναι δύσκολο να καταλαμβάνει μια δευτερεύουσα δραστηριότητα ένα μικρό μέρος της οθόνης και όχι όλο το εύρος της. Στην

κατάσταση αυτή μπορούμε να οδηγηθούμε αν έχουμε κάποιο παράθυρο διαλόγου μπροστά από το προηγουμένως runningActivity. Και σε αυτή την κατάσταση δίνεται αρκετά μεγάλη προτεραιότητα καθώς, λόγω της εμφάνισής του, είναι πολύ περίεργο θέαμα να απομακρυνθεί από την οθόνη.

Κατάσταση διακοπής

Σε κατάσταση διακοπής είναι το Activity όταν δεν εμφανίζεται στην οθόνη αλλά εξακολουθεί να καταλαμβάνει μνήμη. Από αυτή την κατάσταση μπορούμε να το ξαναεμφανίσουμε στην οθόνη και έτσι να γίνει ενεργό (σε κατάσταση λειτουργίας) και πάλι, όπως επίσης και να το «καταστρέψουμε» και να διαγραφεί από τη μνήμη. Η λογική της διατήρησης δραστηριοτήτων σε αυτή την κατάσταση είναι ότι ο χρήστης είναι αρκετά πιθανό να τις ξαναζητήσει σύντομα και η επανεκκίνηση μιας σταματημένης διεργασίας είναι κατά πολύ προτιμότερο από το να την ξεκινήσουμε από την αρχή καθώς όλα τα αντικείμενα είναι ήδη φορτωμένα στη μνήμη και το μόνο που χρειάζεται είναι να καλεστούν ώστε να έρθουν στο προσκήνιο.

Κατάσταση καταστροφής

Ένα Activity σε κατάσταση διακοπής δεν είναι πλέον στη μνήμη. Ο ActivityManager αποφάσισε ότι δεν είναι ανάγκη να καταλαμβάνει πλέον χώρο στη μνήμη κι έτσι το απομάκρυνε. Πριν καταστραφεί τελείως ένα Activity μπορεί να εκτελέσει κάποιες λειτουργίες, όπως για παράδειγμα να σώσει πληροφορίες που δεν έχουν αποθηκευτεί μέχρι τότε. Είναι πιθανό να καταστραφεί κι ένα Activity σε κατάσταση παύσης επίσης γι’ αυτό και πρέπει να έχει προβλέψει ο προγραμματιστής να γίνονται οι σημαντικές εργασίες όπως το σώσιμο πληροφοριών μόλις ένα Activity φεύγει από την κατάσταση λειτουργίας και όχι λίγο πριν καταστραφεί.

Μέθοδοι που χρησιμοποιούνται στα Activities

- **onCreate(Bundle):** Αυτή η μέθοδος καλείται, όταν ξεκινάει το activity για πρώτη φορά και μπορούμε να τη χρησιμοποιήσουμε για να προβάλλουμε το userinterface. Παίρνει μία παράμετρο, είτε κενό (null), είτε την προηγούμενη κατάσταση στην οποία βρισκόταν.
- **onStart():** Αυτή η μέθοδος υποδεικνύει ότι το activity πρόκειται να εμφανιστεί στον χρήστη.
- **onResume():** Αυτή η μέθοδος καλείται, όταν το activity είναι έτοιμο για αλληλεπίδραση με το χρήστη**.**

- **onPause():** Αυτή η μέθοδος καλείται, όταν το activity περάσει σε αναστολή. Σε αυτό το σημείο καλό θα ήταν να αποθηκεύουμε τις πληροφορίες που θέλουμε, γιατί μπορεί να είναι και η τελευταία κατάσταση στην οποία θα βρεθεί το activity.

- **onStop():** Αυτή η μέθοδος καλείται, όταν το activity περάσει σε διακοπή. Αν οι πόροι του συστήματος είναι χαμηλοί τότε το σύστημα μπορεί να καταστρέψει το activity χωρίς να την εκτελέσει.

1.6 Services

Όπως αναφέραμε τα services είναι κώδικας που τρέχει στο background χωρίς να υπάρχει κάποιο εμφανές userinterface. Τα services τρέχουν ακόμα και αν ανοίξουμε άλλη εφαρμογή την ώρα που εκτελούνται. Για παράδειγμα, ένα service μπορεί να χειρίζεται μια διαδικτυακή συναλλαγή, να παίζει μουσική να «γράφει» σε κάποιο αρχείο ή να συνεργάζεται με έναν contentprovider, πάντα στο παρασκήνιο.

Ένα service μπορεί να πάρει δύο μορφές:

- **Started:** Ένα service λέμε ότι είναι started (ξεκινημένο) όταν ένα κομμάτι μιας εφαρμογής (ένα activity για παράδειγμα) το έχει ξεκινήσει καλώντας την μέθοδο startService(). Μόλις ξεκινήσει ένα service μπορεί να τρέχει στο background επ' αόριστον ακόμα κι αν το activity που το κάλεσε καταστραφεί κάποια στιγμή. Συνήθως ένα service εκτελεί μια συγκεκριμένη ενέργεια και δεν επιστρέφει τίποτα εκεί από όπου καλέστηκε. Για παράδειγμα μπορεί να κατεβάζει ή να ανεβάζει ένα αρχείο στο δίκτυο. Όταν τελειώσει η ενέργεια, το service θα σταματήσει μόνο του.

- **Bound:** Ένα service λέμε ότι είναι bound (περιορισμένο) όταν ένα κομμάτι μιας εφαρμογής το καλέσει με τη μέθοδο bindService(). Ένα boundservice προσφέρει μια διασύνδεση τύπου client-server που επιτρέπει οποιαδήποτε εφαρμογή να αλληλεπιδρά με αυτό, να στέλνει αιτήσεις, να παίρνει αποτελέσματα και να κάνει όλα τα παραπάνω μεταξύ διαφορετικών διεργασιών με διαδιεργασιακή επικοινωνία (InterprocessCommunication-IPC). Ένα boundservice μπορεί να συνεχίσει να τρέχει μόνο όσο τρέχει το κομμάτι εφαρμογής που το κάλεσε (με το οποίο είναι συνδεμένο).

Ανεξάρτητα από το ποια από τις δύο μορφές έχει το service μας, οποιαδήποτε εφαρμογή (ή κομμάτι οποιασδήποτε εφαρμογής) μπορεί να το χειριστεί καλώντας το με ένα Intent. Ωστόσο, μπορούμε να το δηλώσουμε ως private στο manifest και να μπλοκάρουμε την πρόσβαση από άλλες εφαρμογές.

Σημείωση: Τα services τρέχουν στο κύριο thread (νήμα) της διεργασίας που τα φιλοξενεί και δεν δημιουργούν καινούργιο thread ούτε τρέχουν σε ξεχωριστή διεργασία (εκτός και αν το καθορίσουμε διαφορετικά). Αυτό σημαίνει ότι αν το service μας πρόκειται να κάνει κάποια «κουραστική» για τον επεξεργαστή ενέργεια, όπως αναπαραγωγή MP3, θα πρέπει να δημιουργήσουμε καινούργιο thread μέσα στο οποίο θα τρέχει το service. Με τη χρήση ενός ξεχωριστού thread, μειώνεται το ρίσκο των σφαλμάτων τύπου ApplicationNotResponding (ANR) και το κύριο thread της εφαρμογής μας μπορεί να μείνει αφιερωμένο στην αλληλεπίδραση με τα activities.

Βασικές μέθοδοι

Για να δημιουργήσουμε ένα service πρέπει αρχικά να δημιουργήσουμε μια υποκλάση της κλάσης Service. Στην υλοποίησή της θα πρέπει να κάνουμε override κάποιες μεθόδους οι οποίες χειρίζονται βασικά χαρακτηριστικά του κύκλου ζωής των services και παρέχουν ένα μηχανισμό που επιτρέπει κομμάτια εφαρμογών να συνδέονται αν χρειαστεί με τα services. Οι πιο σημαντικές μέθοδοι που πρέπει να παρακαμφθούν είναι οι εξής:

- **onStartCommand():** Το σύστημα καλεί αυτή τη μέθοδο όταν ένα κομμάτι μιας εφαρμογής (πχ ένα activity) ζητάει την εκκίνηση του service καλώντας τη μέθοδο startService(). Μόλις εκτελεστεί αυτή η μέθοδος, το service έχει ξεκινήσει και μπορεί να τρέχει στο background επ' αόριστον. Αν υλοποιηθεί έτσι είναι ευθύνη μας να σταματήσουμε το service όταν τελειώσει η δουλειά του, καλώντας την stopSelf() ή την stopService().

- **onBind():** Το σύστημα καλεί αυτή τη μέθοδο όταν μια εφαρμογή θέλει να συνδεθεί με το service, καλώντας τη bindService(). Σε μια τέτοιου είδους υλοποίηση θα πρέπει να παρέχουμε μια διεπαφή που θα επιτρέπει στο χρήστη να επικοινωνεί με το service, επιστρέφοντας έναν IBinder. Πρέπει πάντα να υλοποιούμε αυτή τη μέθοδο, εκτός αν δεν θέλουμε να επιτρέψουμε σύνδεση, οπότε και θα επιστρέψουμε null.

- **onCreate():** Το σύστημα καλεί αυτή τη μέθοδο μόλις δημιουργηθεί το service ώστε να εκτελέσει διαδικασίες που πρέπει να ρυθμιστούν (πριν ακόμα καλέσει την onStartCommand() ή την onBind()). Αν το service εκτελείται ήδη, αυτή η μέθοδος δεν καλείται.

- **onDestroy():** Το σύστημα καλεί αυτή τη μέθοδο όταν το service δεν χρησιμοποιείται πλέον και πρέπει να καταστραφεί. Το service μας θα πρέπει να την υλοποιήσει ώστε να καθαριστούν τυχόν πόροι, όπως threads, listeners, receivers κλπ. Αυτή είναι και η τελευταία κλήση που λαμβάνει ένα service.

Αν ένα service ξεκινήσει από κάλεσμα της startService() (μέσω της οποίας καλείται η onStartCommand()) τότε το service συνεχίζει να τρέχει μέχρι να σταματήσει με την stopSelf() ή μέχρι να το σταματήσει κάποια εφαρμογή καλώντας την stopService().

Αν ένα service ξεκινήσει από κάλεσμα της bindService() (και η onStartCommand() *δεν* καλεστεί), τότε το service τρέχει μόνο για όσο η εφαρμογή που το κάλεσε είναι συνδεδεμένη με αυτό. Μόλις το service αποσυνδέεται από όλους τους clients που τυχόν είναι συνδεμένοι πάνω του, το σύστημα το καταστρέφει.

Το σύστημα του Android θα κλείσει αναγκαστικά ένα service μόνο όταν η μνήμη που απομένει είναι λίγη και θα πρέπει να ανακτηθούν επαρκείς πόροι συστήματος ώστε να λειτουργεί απρόσκοπτα το activity στο οποίο δουλεύει ο χρήστης. Αν το service είναι συνδεμένο με το ενεργό activity τότε είναι λιγότερο πιθανό να καταστραφεί ενώ στην περίπτωση που το service είναι φτιαγμένο ώστε να τρέχει στο προσκήνιο είναι σχεδόν απίθανο να καταστραφεί. Αντιθέτως, στην περίπτωση που το service έχει ξεκινήσει και ήδη τρέχει για ώρα, το σύστημα θα χαμηλώσει την προτεραιότητά του στη λίστα των διεργασιών παρασκηνίου, οπότε και η πιθανότητα να «σκοτωθεί» κάποια στιγμή από το σύστημα αυξάνεται (το service θα επανεκκινηθεί μόλις οι πόροι γίνουν διαθέσιμοι και πάλι). Για το λόγο αυτό πρέπει να είμαστε προσεκτικοί στη δημιουργία services και να χειριζόμαστε ομαλά την τυχόν επανεκκίνησή τους από το σύστημα.

1.7 Views

Το Android παρέχει μια εργαλειοθήκη με πρότυπα views για να μας βοηθήσει να δημιουργήσουμε απλά userinterfaces. Μερικά από τα πιο γνωστά views είναι:

- **TextView:** Ένα πρότυπο view που είναι μόνο για ανάγνωση κειμένου ετικέτας, υποστηρίζει πολλαπλές γραμμές κειμένου, μορφοποίηση κειμένου και αυτόματη αναδίπλωση λέξεων.

- **EditText:** Ένα επεξεργάσιμο πλαίσιο εισαγωγής κειμένου που δέχεται πολλαπλές εγγραφές, αναδίπλωση λέξεων και υπόδειξη κειμένου.

- **Spinner:** Είναι ένα σύνθετο στοιχείο ελέγχου που εμφανίζει ένα TextView και ένα ListView (σχετική προβολή λίστας) η οποία μας επιτρέπει να επιλέξουμε ένα στοιχείο από μια λίστα για να εμφανιστεί στο πλαίσιο κειμένου.

- **Button:** Είναι ένα πρότυπο κουμπί.

- **CheckBox:** Είναι ένα κουμπί δύο καταστάσεων με δυνατότητα επιλογής.

- **RadioButton:** Είναι ένα κουμπί δύο καταστάσεων που αντιπροσωπεύει μια ομάδα επιλογών εκ των οποίων μόνο μια επιλογή μπορεί να είναι ενεργή κάθε φορά.

1.8 Layouts

Τα layouts είναι επεκτάσεις της κλάσης ViewGroup και χρησιμοποιούνται για να τοποθετήσουν κατάλληλα στην οθόνη τα στοιχεία που περιέχουν. Κάθε layout χρησιμοποιεί τον δικό του τρόπο τοποθέτησης των στοιχείων και μπορεί να περιέχει άλλα layouts ώστε να επιτευχθεί το επιθυμητό αποτέλεσμα. Το Android SDK περιλαμβάνει ορισμένα απλά layouts για να μας βοηθήσει στην κατασκευή του UI. Μερικά από αυτά είναι:

- **FrameLayout:** Το απλούστερο από τα layouts, απλά «καρφιτσώνει» κάθε view που περιέχει στην πάνω αριστερή γωνία.

- **LinearLayout:** Αυτό το layout στοιχίζει κάθε view, είτε σε κάθετη είτε σε οριζόντια γραμμή. Ένα layout με κάθετη διάταξη έχει μια στήλη από views, ενώ παράλληλα ένα layout με οριζόντια διάταξη έχει μια σειρά από views. Ο LinearLayoutManager (διαχειριστής διάταξης) μας επιτρέπει να ορίσουμε ένα «βάρος» για κάθε view, δηλαδή το σχετικό μέγεθος του καθενός εντός του διαθέσιμου χώρου. Για παράδειγμα σε ένα LinearLayout με οριζόντια διάταξη τοποθετούμε ένα κουμπί με βάρος 0.5 και συνολικό βάρος 1. Αυτό σημαίνει ότι το κουμπί θα

καταλάβει το μισό χώρο της γραμμής και θα αφήσει τον υπόλοιπο χώρο για τα υπόλοιπα κουμπιά που θα προσθέσουμε. Για τιμή ίση με 1 απλά θα καταλάβει όλο το χώρο μη δίνοντας σημασία για το χώρο που χρειάζονται τα υπόλοιπα views.

- **TableLayout:** Αυτό το layout μας επιτρέπει να τοποθετήσουμε τα views χρησιμοποιώντας ένα πλέγμα γραμμών και στηλών. Οι πίνακες μπορούν να εκτείνονται σε πολλαπλές γραμμές και στήλες και οι στήλες μπορούν να ρυθμιστούν ώστε να συρρικνώνονται ή να αναπτύσσονται.

- **Gallery:** Ένα layout αυτού του τύπου εμφανίζει μια απλή σειρά στοιχείων σε μία οριζόντια κυλιόμενη λίστα.

Τα layouts που μας προσφέρει το SDK μπορεί να μην ικανοποιούν τις ανάγκες μας με αποτέλεσμα να καταφύγουμε σε ένα προσαρμοσμένο layout ή αλλιώς εργαλείο. Για παράδειγμα, αν το layout που θέλουμε να χρησιμοποιήσουμε με επιπλέον μεθόδους είναι ένα LinearLayout, τότε απλά το επεκτείνουμε και προσθέτουμε τις μεθόδους που χρειαζόμαστε.

Αν δεν μας ικανοποιεί κανένα από τα υπάρχοντα layouts μπορούμε να επεκτείνουμε την κλάση ViewGroup ή την View ώστε να φτιάξουμε κάτι προσαρμοσμένο για την εφαρμογή μας. Αυτό συμβαίνει συχνά στις εφαρμογές παιχνιδιών και ειδικά σε αυτές που δεν απαιτούν μεγάλο ρυθμό επεξεργασίας πλαισίων.

1.9 SQLite

Το Android χρησιμοποιεί βάσεις δεδομένων Sqlite για να αποθηκεύει σημαντικές πληροφορίες οι οποίες δεν πρέπει να χαθούν ακόμα και αν κλείσει η εφαρμογή ή ακόμα και μετά από επανεκκίνηση της συσκευής. Τα δεδομένα αυτά περιλαμβάνουν επαφές, ρυθμίσεις συστήματος, σελιδοδείκτες κλπ.

Η SQLite είναι μια βιβλιοθήκη λογισμικού που υλοποιεί μια αυτοσυντηρούμενη, συναλλακτική SQL μηχανή βάσεων δεδομένων η οποία δεν χρειάζεται server ούτε καμία ρύθμιση παραμέτρων για να λειτουργήσει. Είναι η πιο διαδομένη παγκοσμίως και ο πηγαίος κώδικάς της είναι ελεύθερος να χρησιμοποιηθεί για οποιαδήποτε χρήση. Αντίθετα με άλλες βάσεις η SQLite δεν απαιτεί τη χρήση ξεχωριστού server αλλά «διαβάζει» και «γράφει» απευθείας σε απλά αρχεία που

είναι αποθηκευμένα στο δίσκο. Μια ολοκληρωμένη βάση με τους πίνακές της, τα εναύσματα, τα views κλπ μπορεί να αποθηκευτεί σε ένα και μοναδικό αρχείο. Η μορφή του αρχείου αυτού δουλεύει σε οποιαδήποτε πλατφόρμα. Το υπερβολικά μικρό μέγεθος της βάσης που προκύπτει (350KB με όλες τις λειτουργίες ενεργές) κάνουν την SQLite πολύ δημοφιλή σε μικρές συσκευές (περιορισμένης μνήμης) όπως κινητά τηλέφωνα, PDA και MP3 players.

Χαρακτηριστικά

- Οι συναλλαγές είναι ατομικές, συνεπείς, απομονωμένες και ανθεκτικές ακόμα και μετά από κατάρρευση του συστήματος ή διακοπές ρεύματος.
- Δεν απαιτείται κανενός είδους διαχείριση ή ρύθμιση παραμέτρων.
- Μια ολοκληρωμένη βάση είναι αποθηκευμένη σε ένα μοναδικό αρχείο.
- Είναι δυνατή η υποστήριξη μέχρι και τεράστιων βάσεων δεδομένων (της τάξης των μερικών terabyte).
- Πολύ μικρό μέγεθος βιβλιοθήκης (λιγότερο από 350KB με όλες τις λειτουργίες ενεργές και λιγότερο από 200KB με παράλειψη των προαιρετικών χαρακτηριστικών).
- Γρηγορότερη στις περισσότερες κοινές λειτουργίες από τις διάσημες μηχανές που χρησιμοποιούν τη λογική client/server.
- Απλή και εύκολη στη χρήση διεπαφή προγραμματισμού (API).
- Προσιτή σαν ένα ANSI-C αρχείο το οποίο μπορεί εύκολα να χρησιμοποιηθεί σε ένα άλλο project.
- Αυτοσυντηρούμενη (χωρίς εξωτερικές εξαρτήσεις).
- Πολλέςυποστηριζόμενεςπλατφόρμες: Linux, Mac OS-X, Android, iOSκαι Windows.
- Ελεύθερος πηγαίος κώδικας για κάθε χρήση (εμπορική ή προσωπική).
- Υπάρχει αυτόνομος client για τη διαχείριση των SQLite βάσεων.

Η SQLite είναι διαθέσιμη σε κάθε Android συσκευή. Η χρήση της δεν απαιτεί κάποια ρύθμιση εκ των προτέρων. Είναι απαραίτητο μόνο να οριστούν οι δηλώσεις για την δημιουργία και την ενημέρωση της βάσης. Από εκεί και έπειτα η βάση διαχειρίζεται από την πλατφόρμα του Android.

Επειδή η πρόσβαση σε μια SQLite βάση δεδομένων περιλαμβάνει πρόσβαση στο σύστημα αρχείων η διαδικασία γραφής/ανάγνωσης μπορεί να είναι αργή. Για το

λόγο αυτό προτείνεται η εκτέλεση των εργασιών στη βάση ασύγχρονα, για παράδειγμα με τη χρήση της AsyncTask κλάσης.

Μια SQLite βάση που δημιουργήθηκε από μια Android εφαρμογή αποθηκεύεται από προεπιλογή στη διαδρομή DATA/data/APP_NAME/databases/FILENAME (με DATA τη διαδρομή όπου επιστρέφει η μέθοδος Environment.getDataDirectory(), APP_NAME το όνομα της εφαρμογής μας και FILENAME το καθορισμένο όνομα της βάσης).

Το framework του Android προσφέρει πολλούς τρόπους για τη χρήση της SQLite εύκολα και αποτελεσματικά. Μεγάλο πλεονέκτημα για τον προγραμματιστή είναι ότι παρόλο που η SQLite χρησιμοποιεί SQL, το Android παρέχει μια υψηλότερου επιπέδου βιβλιοθήκη με ένα περιβάλλον που είναι πολύ ευκολότερο να ενσωματωθεί σε μια εφαρμογή.

2. ΑΝΟΙΧΤΑ ΔΕΔΟΜΕΝΑ

Τα δεδομένα που παράγουν όλες οι θεσμοθετημένες πήγες εξουσίας καθώς και οι Δημόσιες Επιχειρήσεις Κοινής Ωφέλειας (ΔΕΚΟ), αποτελούν κτήμα όλων μας καθότι η πηγή κάθε εξουσίας είναι ο λαός. Η πρόσβαση σε αυτά αποτελεί στην Ελλάδα συνταγματικά κατοχυρωμένο δικαίωμα. Η ανοιχτή και ελεύθερη πρόσβαση στα δημόσια δεδομένα αποτελεί επίσης συστατικό στοιχείο για μια ουσιαστική συμμετοχή στην κοινωνική, οικονομική και πολιτική ζωή και στην κοινωνία της πληροφορίας. Η τεχνολογία της πληροφορικής μας δίνει τα εργαλεία για να διαθέτουμε την πληροφορία δημόσια με έναν εύκολο και οικονομικό τρόπο μέσω του διαδικτύου. Η πρόκληση όμως υπερβαίνει τις τεχνολογίες διάθεσης των δεδομένων από τις πηγές τους. Τα εμπόδια ουσιαστικής πρόσβασης στα δημόσια δεδομένα μπορεί να είναι ταυτόχρονα , θεσμικής , οργανωτικής και τεχνολογικής φύσης. Μέχρι σήμερα , λίγοι είναι οι δημόσιοι φορείς που καθιστούν τα δεδομένα που έχουν στην κατοχή τους διαθέσιμα στους πολίτες μέσω του διαδικτύου. Συχνά θεωρούν ότι τα δεδομένα είναι «δικά τους», «δεδομένα των πολιτών» , πάντως σε κάθε περίπτωση όχι δεδομένα που μπορούν να διατεθούν εκτός του οργανισμού. Σε άλλες περιπτώσεις , οι δημόσιοι οργανισμοί διαθέτουν τα δεδομένα αλλά με περιοριστικούς όρους χρήσης , οι οποίοι δημιουργούν εμπόδια στη περαιτέρω επεξεργασία και την επαναχρησιμοποίηση τους. Ακόμη όμως κι όταν τα θεσμικά ή οργανωτικά προβλήματα έχουν επιλυθεί τα δεδομένα είναι δυνατόν να μην είναι ουσιαστικά διαθέσιμα εξαιτίας προβλημάτων που σχετίζονται με τις τεχνολογικές

επιλογές. Η εκρηκτική αύξηση του όγκου της ψηφιακής πληροφορίας (ψηφιακή υπερφόρτωση) κατά τα τελευταία χρόνια , δυσχεραίνει τις δυνατότητες των πολιτών να εντοπίσουν την πληροφορία , ειδικά όταν αυτή δε βρίσκεται σε σταθερά και γνωστά σημεία. Επιπλέον η διάθεση με πρότυπα που δεν είναι μηχανικά αναγνώσιμα δεν επιτρέπει μετασχηματισμένες χρήσεις των δεδομένων.

2.1 Ανοιχτά Δημόσια Δεδομένα

Ο όρος **Ανοιχτά Δημόσια Δεδομένα** (Open Public Data) χρησιμοποιείται για να αναφερθούμε σε *δεδομένα ή σύνολα δεδομένων που αφορούν το συλλογικό γίγνεσθαι και για τα οποία υφίσταται μια συνειδητή και συνεπής πολιτική η οποία επιτρέπει την ελεύθερη διάθεση και επαναχρησιμοποίηση τους.* Ο παραπάνω ορισμός δεν είναι ούτε πλήρης ούτε καθολικός καθότι τόσο η έννοια *Ανοιχτά* όσο και ο προσδιορισμός *Δημόσια* , διαφοροποιούνται ανάλογα με το γενικότερο πλαίσιο περιγραφής και το νομικό σύστημα εντός του οποίου εξετάζονται. Στη συνέχεια θα προσπαθήσουμε να προσεγγίσουμε καλύτερα τους δύο αυτούς όρους , ξεκινώντας από τον προσδιορισμό της κλάσης των δημοσίων δεδομένων και συνεχίζοντας με τις προϋποθέσεις που εξασφαλίζουν την ανοιχτότητα τους.

Δημόσια Δεδομένα

Με τον όρο “**Δημόσια Δεδομένα**” εννοούμε δεδομένα και πληροφορίες που σχετίζονται με την δημόσια σφαίρα ανεξάρτητα του εάν παράγονται από δημόσιους ή ιδιωτικούς φορείς. Ο όρος δημόσια δεδομένα λοιπόν χρησιμοποιείται με την διασταλτική έννοια του όρου και μπορεί να περιλαμβάνει δεδομένα που :
Βρίσκονται στην κατοχή ενός οργανισμού του στενού ή ευρύτερου δημόσιου τομέα.

- Έχουν παραχθεί με δημόσιους πόρους.
- Παράγονται από τον ιδιωτικό τομέα αλλά αφορούν γενικότερα το κοινωνικό σύνολο.

Κυβερνητικά Δεδομένα

Σύμφωνα με τα παραπάνω, στα δημόσια δεδομένα δεν περιλαμβάνονται αποκλειστικά δεδομένα που σχετίζονται με κρατικές , διοικητικές ή πολιτειακές αρμοδιότητες. Μπορεί να συμπεριλαμβάνονται ακόμα και δεδομένα που παράγονται από ιδιωτικούς φορείς και επιχειρήσεις , οι οποίοι έχουν χρηματοδοτηθεί για την παραγωγή τους από κρατικά κονδύλια. Ακόμα όμως και στις περιπτώσεις που δεν υφίσταται κρατική χρηματοδότηση , τα δεδομένα που παράγει ο Ιδιωτικός τομέας μπορεί να θεωρηθούν δημόσια αρκεί να σχετίζονται με

την δημόσια σφαίρα. Ποιοτικά, δημόσια δεδομένα παράγει δυνητικά ο κάθε πολίτης. Πρώτα από όλα, όμως, το κάνει το κράτος στο όνομά τους. *Το υποσύνολο των δημοσίων δεδομένων που παράγονται από το κράτος αποκαλούνται* ***Κυβερνητικά Δεδομένα***.

Παράδειγμα δημοσίων δεδομένων είναι τα δεδομένα που αφορούν τα σήματα του μηχανισμού της αγοράς ή τα δεδομένα σχετικά με τον καιρό. Στα δημόσια δεδομένα όμως συμπεριλαμβάνονται όλα τα κυβερνητικά δεδομένα , γιατί αφορούν την σφαίρα του συλλογικού. Μάλιστα πολλές φορές το κράτος αναλαμβάνει την παραγωγή δεδομένων που είναι καθαυτά δημόσια και δεν σχετίζονται με κυβερνητικές αρμοδιότητες. Υπάρχουν αρκετοί λόγοι για τους οποίους το Κράτος παράγει δημόσια –μη κυβερνητικά- δεδομένα. Ο βασικός είναι γιατί η αγορά δεν μπορεί να αναλάβει τη παραγωγή τους. Είτε γιατί το κόστος παραγωγής είναι πολύ υψηλό, είτε γιατί τα συγκεκριμένα δεδομένα δεν μπορούν να διατεθούν ως προϊόν από την αγορά. Τέτοια παραδείγματα αποτελούν οι δορυφορικές απεικονίσεις ή οι αεροφωτογραφίες, το κόστος παραγωγής των οποίων είναι ιδιαίτερα υψηλό καθώς και τα στοιχεία σχετικά με τη ποιότητα των υδάτων και τους ατμοσφαιρικούς ρύπους τα οποία δεν θεωρούνται αξιοποιήσιμα από την αγορά.

«Ανοιχτά» δεδομένα

Σε γενικές γραμμές για να μπορούν τα δημόσια δεδομένα να θεωρηθούν ανοιχτά κα πρέπει να είναι δυνατή η **περαιτέρω χρήση** και **επαναδιάθεσή** τους. Ως περαιτέρω χρήση νοείται η επαναχρησιμοποίηση δεδομένων και πληροφοριών για σκοπό άλλο από αυτόν για τον οποίο παρήχθησαν, είτε αυτός είναι εμπορικός είτε όχι. Ο βαθμός της ανοιχτότητας των δεδομένων προσδιορίζεται από την ύπαρξη ή όχι περιορισμών στις παραπάνω δυνατότητες. Σύμφωνα με τον ορισμό της Ανοιχτότητας που δίνει το *Open Knowledge Foundation3* (OKFN) :

«Ανοιχτό είναι ένα κομμάτι περιεχομένου ή δεδομένων εφόσον οποιοσδήποτε είναι ελεύθερος να το χρησιμοποιήσει , επαναχρησιμοποιήσει και επαναδιαθέσει , ***χωρίς περιορισμούς ή με μόνο περιορισμό την αναφορά στο δημιουργό και την παρόμοια χρήση****» .*

Στον παραπάνω ορισμό:

(α) Ορίζονται οι **ελάχιστες δυνατότητες** που έχει οποιοσδήποτε χρήστης επί των δεδομένων, δηλαδή: η δυνατότητα χρησιμοποίησης, επαναχρησιμοποίησης και επαναδιάθεσης και

(β) Θέτονται τα **όρια των περιορισμών** που επιτρέπεται να ασκηθούν επί αυτών των δυνατοτήτων , δηλαδή : η υποχρέωση αναφοράς του δημιουργό ή παρόμοια χρήση.

Οι δυνατότητες που αναφέρονται στο πρώτο κομμάτι του ορισμό θα πρέπει να διασφαλίζονται με δύο τρόπους.

- Κατά πρώτον νομικά : η άδεια που συνοδέψει τα δεδομένα θα πρέπει να επιτρέπει όλες τις παραπάνω χρήσεις.
- Κατά δεύτερον τεχνολογικά : θα πρέπει δηλαδή οι τεχνολογίες και τα τεχνολογικά μέσα που χρησιμοποιούνται , να επιτρέπουν και να διευκολύνουν τόσο την επαναχρησιμοποίηση όσο και την επαναδιάθεσή τους.

Εικόνα 2.1 Το οικοσύστημα των Ανοιχτών Δεδομένων

Ένας πιο πλήρης ορισμός της έννοιας των ανοιχτών δεδομένων ο οποίος εστιάζει και στις τεχνικές απαιτήσεις της ανοιχτότητας είναι ο εξής : *Ανοιχτά θεωρούνται τα δεδομένα όταν δημοσιεύονται και ανανεώνονται μέσω του παγκόσμιου ιστού όσο το δυνατόν συντομότερα και συχνότερα , με τρόπο που επιτρέπει την νόμιμη ,δωρεάν επαναχρησιμοποίηση τους για οποιονδήποτε σκοπό και διευκολύνει την γρήγορη και αυτοματοποιημένη επεξεργασία τους με δωρεάν διαθέσιμο λογισμικό. Πρακτικά αυτό σημαίνει ότι τα δεδομένα είναι «ανοιχτά» όταν συνοδεύονται από ανοιχτή άδεια χρήσης που επιτρέπει όσα επισημάνθηκαν στη προηγούμενη πρόταση και δημοσιεύονται είτε με κάποιο ανοιχτό πρότυπο αρχείου είτε μέσω*

άμεσης πρόσβασης με χρήση ανοιχτών διαδικτυακών πρωτοκόλλων. Τα χαρακτηριστικά που θα πρέπει να διέπουν την διάθεση των δημοσίων δεδομένων για να θεωρούνται ανοιχτά περιλαμβάνουν τόσο τεχνολογικές όσο και νομικές / οργανωτικές απαιτήσεις και παρουσιάζονται αναλυτικά στην επόμενη ενότητα ως ένα σύνολο αρχών.

2.2 Οι αρχές των Ανοικτών Δημόσιων Δεδομένων

Τα δημόσια δεδομένα θεωρούνται ανοιχτά , εφόσον διατίθενται μέσω του διαδικτύου με τρόπο που είναι συμβατός με τις ακόλουθες βασικές αρχές :

1. Πληρότητα

Τα σύνολα δεδομένων που δημοσιεύονται κα πρέπει να είναι όσο το δυνατόν πλήρη, αντικατοπτρίζοντας την πληρότητα της καταγεγραμμένης πληροφορίας για ένα συγκεκριμένο θέμα. Όπου η πρωτογενής πληροφορία για ένα σύνολο δεδομένων πρέπει να είναι διαθέσιμο στο κοινό, με την εξαίρεση θεμάτων που αφορούν στην προστασία προσωπικών δεδομένων. Μεταδομένα που ορίζουν και εξάγουν τα πρωτογενή δεδομένα πρέπει να περιλαμβάνονται, μαζί με μαθηματικούς τύπους και εξηγήσεις σχετικά με το πώς τα παραγόμενα δεδομένα έχουν υπολογιστεί. Με τον τρόπο αυτό οι χρήστες θα μπορούν να κατανοήσουν το εύρος της διαθέσιμης πληροφορίας και να εξετάσουν τα διαθέσιμα δεδομένα στο μέγιστο δυνατό βαθμό λεπτομέρειας.

2. Πρωτότυπα

Τα δεδομένα που δημοσιεύονται πρέπει να είναι κυρίως πρωτογενή δεδομένα. Στην έννοια αυτή, περιλαμβάνεται η πρωτογενής πληροφορία που συλλέχτηκε, λεπτομέρειες σχετικά με το πώς τα δεδομένα έχουν συλλεχτεί, καθώς και τα πρωτότυπα κείμενα που καταγράφουν τη συλλογή των δεδομένων. Η δημόσια διάχυση τους Θα επιτρέπει στους χρήστες να επιβεβαιώσουν πως η πληροφορία συλλέχτηκε και καταγράφθηκε ορθά.

3. Αμεσότητα

Τα δεδομένα που δημοσιεύονται από την κυβέρνηση πρέπει να είναι διαθέσιμα στο κοινό όσο το δυνατόν πιο σύντομα. Όποτε είναι δυνατό, η πληροφορία που συλλέγεται πρέπει να δημοσιεύεται αμέσως μετά τη συγκέντρωση και συλλογή της. Προτεραιότητα πρέπει να δίνεται σε δεδομένα των οποίων η χρήση είναι χρονικά ευαίσθητη. Οι ενημερώσεις της πληροφορίας σε πραγματικό χρόνο θα μεγιστοποιήσουν την αξία που μπορεί να εξάγει το κοινό από την πληροφορία.

4. Ευκολία στην ηλεκτρονική πρόσβαση

Τα δεδομένα που δημοσιεύονται πρέπει να είναι όσο το δυνατόν περισσότερο προσβάσιμα, με την προσβασιμότητα τους να ορίζεται ως την ευκολία με την

οποία η πληροφορία μπορεί να ανακτηθεί με ηλεκτρονικά μέσα. Οι φραγμοί στη πρόσβαση περιλαμβάνουν τη διάθεση των δεδομένων μέσω της συμπλήρωσης φορμών ή συστήματα που απαιτούν ειδικές τεχνολογίες για τους browsers (π.χ. Flash, cookies, Java applets). Σε αντίθεση, προσφέροντας μια διεπαφή για τη μεταφόρτωση όλων των πληροφοριών που είναι αποθηκευμένες σε μια βάση συνολικά («μαζική» πρόσβαση) και τα μέσα για την εκτέλεση συγκεκριμένων αιτήσεων για δεδομένα μέσω μιας προγραμματιστικής διεπαφής (API), τα δεδομένα καθίστανται περισσότερο προσβάσιμα.

5. Αναγνωσιμότητα από υπολογιστές Οι υπολογιστές μπορούν να χειριστούν ορισμένα είδη εισόδου καλύτερα από άλλα. Για παράδειγμα, χειρόγραφες σημειώσεις σε ένα χαρτί είναι πολύ δύσκολο να επεξεργαστούν . Η ψηφιοποίηση κειμένου μέσω λογισμικού OCR (οπτικής αναγνώρισης χαρακτήρων) οδηγεί σε πολλά σφάλματα ταυτοποίησης και μορφοποίησης. Για παράδειγμα, πληροφορία που μοιράζεται στον ευρέως διαδεδομένο PDF, είναι δύσκολο να αναλυθεί από υπολογιστές. Συνεπώς, η πληροφορία πρέπει να είναι αποθηκευμένη σε ευρέως χρησιμοποιούμενους τύπους δεδομένων που είναι εύκολο να επεξεργαστούν από μηχανές. Τα εν λόγω αρχεία πρέπει να συνοδεύονται από τεκμηρίωση σχετικά με το μορφότυπο τους και πως αυτός μπορεί να χρησιμοποιηθεί σε σχέση με τα δεδομένα.

6. Όχι διακρίσεις

Η έννοια της «όχι-διάκρισης» αναφέρεται στο ποιοι μπορεί να έχουν πρόσβαση στα δεδομένα και πως αυτοί πρέπει να πράξουν ώστε να έχουν πρόσβαση στα δεδομένα. Φραγμοί στη χρήση των δεδομένων μπορούν να περιλαμβάνουν απαιτήσεις εγγραφής, ή με τη χρήση «φραγμένων κήπων» (walled gardens) όπου μόνο συγκεκριμένες εφαρμογής μπορούν να έχουν πρόσβαση στα δεδομένα. Γενικότερα, η πρόσβαση στα δεδομένα χωρίς διακρίσεις σημαίνει πως κάθε άτομο μπορεί να έχει πρόσβαση στα δεδομένα, σε κάθε χρονική στιγμή, χωρίς να οφείλει να ταυτοποιηθεί ι να αιτιολογήσει το σκοπό πρόσβασης.

7. Χρήση ανοικτών προτύπων

Η έννοια «ανοικτό» πρότυπο αναφέρεται στην ιδιοκτησία του μορφότυπου με τον οποίο είναι αποθηκευμένα τα δεδομένα. Για παράδειγμα, αν μόνο μία εταιρεία κατασκευάζει ένα λογισμικό που μπορεί να αναγνώσει το αρχείο στο οποίο είναι αποθηκευμένα τα δεδομένα, η πρόσβαση στην πληροφορία εξαρτάται από τη χρήση του λογισμικού που κατασκευάζει αυτή η μοναδική εταιρεία. Μερικές φορές το πρόγραμμα αυτό δεν είναι καν διαθέσιμο στο κοινό, ή μπορεί να είναι διαθέσιμο, αλλά να απαιτείται η αγορά του. Ελεύθερα διαθέσιμοι εναλλακτικοί μορφότυποι συχνά υπάρχουν, ώστε τα αποθηκευμένα δεδομένα να είναι προσβάσιμα χωρίς να υπάρχει απαίτηση για την αγορά άδειας χρήσης λογισμικού. Αφαιρώντας αυτό το κόστος, κάνουμε τα δεδομένα διαθέσιμα σε ένα μεγαλύτερο πλήθος δυνητικών χριστών.

8. Αδειοδότηση

Η επιβολή «Όρων Χρήσης», απαιτήσεων απόδοσης, περιορισμών στη διάχυση, κτλ, λειτουργούν ως εμπόδια στη δημόσια χρήση των δεδομένων. Το μέγιστο άνοιγμα των δεδομένων περιλαμβάνει την ξεκάθαρη σήμανση της δημόσιας πληροφορίας ως έργου της κυβέρνησης και της διάθεσης της χωρίς περιορισμούς σε δημόσιο καθεστώς.

9. Διαχρονικότητα

Η δυνατότητα εύρεσης πληροφορίας στο πέρασμα του χρόνου αναφέρεται ως διαχρονικότητα. Η πληροφορία που δημοσιεύεται από την κυβέρνηση στο διαδίκτυο πρέπει να είναι σταθερής αναφοράς, δηλαδή πρέπει να είναι διαθέσιμη στο διηνεκές. Πολύ συχνά, η πληροφορία ενημερώνεται, αλλάζει ή διαγράφεται, χωρίς καμία ένδειξη για την τέλεση αυτής της μεταβολής. Σε άλλες περιπτώσεις, είναι διαθέσιμη ως ρεύμα δεδομένων (stream of data) αλλά δεν αρχειοθετείται. Με σκοπό την καλύτερη χρήση από το κοινό, η πληροφορία που δημοσιεύεται στο διαδίκτυο πρέπει να παραμένει δημοσιευμένη στο διαδίκτυο, με κατάλληλες διατάξεις τήρησης των διαφορετικών της εκδόσεων και αρχειοθέτησής της.

10. Κόστος χρήσης

Ένας από τους μεγαλύτερους φραγμός στη δημοσίευση δημόσιων δεδομένων είναι το κόστος που επιβάλλεται στο κοινό για την πρόσβαση, ακόμη και αν αυτό είναι το ελάχιστο δυνατό (de minimus). Οι κυβερνήσεις χρησιμοποιούν ένα σύνολο επιχειρημάτων για τη χρέωση του κοινού με σκοπό την πρόσβαση στα δεδομένα τους: το κόστος δημιουργίας τους, η απόσβεση της επένδυσης, το κόστος ανάκτησης της πληροφορίας, το κόστος επεξεργασίας, κτλ.

Η πλειοψηφία της κυβερνητικής πληροφορίας συλλέγεται για κυβερνητικούς σκοπούς, οπότε η ύπαρξη αυτών των τελών έχει ελάχιστη ζωή μηδενικής επίδρασης στο αν τελικά η κυβέρνηση θα παρήγαγε εξαρχής την πληροφορία. Η επιβολή τελών για την πρόσβαση αλλοιώνει τη σύνθεση του κοινού που επιθυμεί (ή μπορεί) να έχει πρόσβαση στην πληροφορία. Επιπλέον, μπορεί να αποκλείει μετασχηματιζόμενες χρήσεις των δεδομένων που στη συνέχεια οδηγούν σε επιχειρηματική άνθηση και άρα φορολογικά έσοδα.

2.3 Γιατί τα Δημόσια Δεδομένα να είναι Ανοιχτά ;

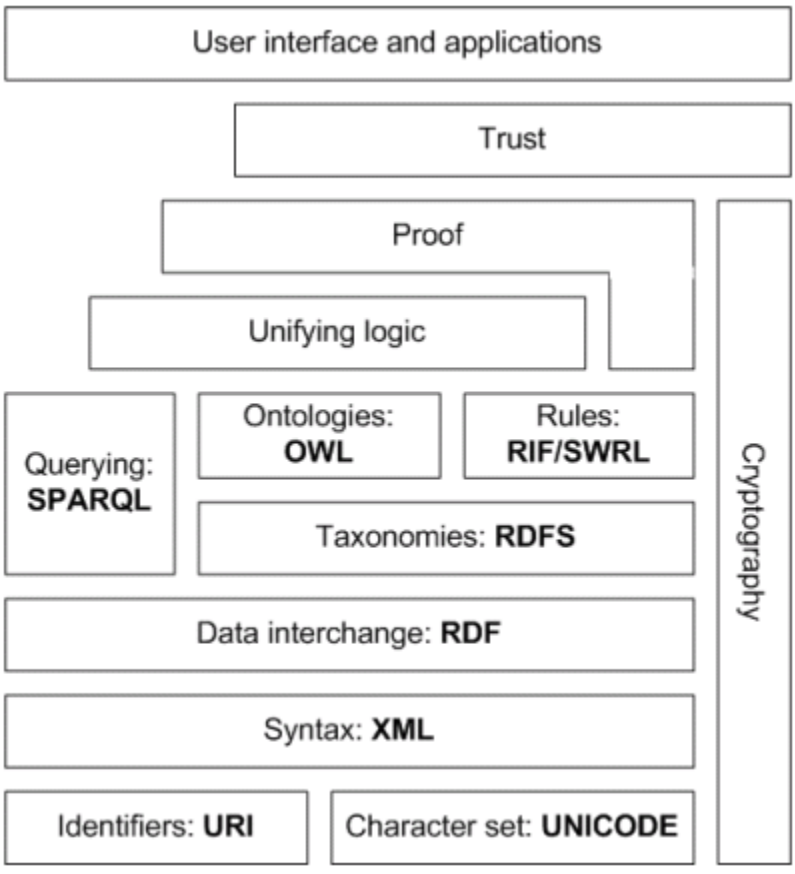
Υπάρχουν πολλοί λόγοι για τους οποίους τα δημόσια δεδομένα θα πρέπει να είναι ανοιχτά προς την κοινωνία. Καταρχήν επειδή τα δεδομένα αυτά παράγονται με χρήματα των φορολογούμενων πολιτών που χρηματοδοτούν έρευνες και υπηρεσίες της κυβέρνησης και συνεπώς αποτελεί αναφαίρετο δικαίωμα των πολιτών να έχουν δυνατότητα πρόσβασης στα αποτελέσματα του έργου αυτού. Δεύτερον γιατί τα δεδομένα αυτά αν συνδυαστούν και μετασχηματιστούν κατάλληλα εμπεριέχουν πολύτιμη γνώση, για την κοινωνική και οικονομική πραγματικότητα, που θα πρέπει να καθίσταται διαθέσιμη και να υποστηρίζει τη διαδικασία των συλλογικών αποφάσεων. Τρίτον γιατί μέρος αυτών των δεδομένων αποτυπώνει και καταγράφει την αποτελεσματικότητα της δημόσιας ασκούμενης πολιτικής και άρα αποτελεί παράγοντα αξιολόγησης του έργου της κυβέρνησης. Σε μία απώτερη αλλά ορατή αναγωγή , τα ανοιχτά κυβερνητικά δεδομένα μειώνουν την απόσταση μεταξύ πολίτη και εξουσίας , αφού αν αξιοποιηθούν κατάλληλα οδηγών σε πληροφορίες και γνώση, βάσει της οποία οι πολίτες τροφοδοτούν με σήματα το πολιτικό σύστημα μέσω της ψήφου τους. Επιπροσθέτως , τα ανοιχτά και ελευθέρα διαθέσιμα δημόσια δεδομένα ενισχύουν την αποτελεσματικότητα του δημόσιου τομέα μεταφέροντας κάποιες από τις αναλυτικές απαιτήσεις της διοίκησης σε τρίτα μέρη όπως κοινότητες πολιτών , μη κυβερνητικές οργανώσεις, ερευνητικά ιδρύματα και Μ.Μ.Ε. , τα οποία συνδυάζουν δεδομένα από διαφορετικές πήγες με πρωτότυπο και εφευρετικό τρόπο.

2.4 Σημασιολογικός Ιστός και Συνδεδεμένα Δεδομένα

Η ανάπτυξη του παγκόσμιου ιστού έκανε το διαδίκτυο προσβάσιμο σε εκατομμύρια χρήστες , επιτρέποντας την απρόσκοπτη δημοσιοποίηση και πρόσβαση σε πληροφορίες. Η εκρηκτική ανάπτυξη του παγκόσμιου ιστού

δημιούργησε όμως προβλήματα «πληροφοριακής υπερφόρτωσης» και προέκυψε το αίτημα καλύτερης οργάνωσης των πληροφοριών. Ως απάντηση , η παγκόσμια ερευνητική κοινότητα έχει στραφεί ήδη και λίγα χρόνια σε μια νέα κατεύθυνση εξέλιξης του παγκόσμιου ιστού , η οποία ονομάζεται **Σημασιολογικός Ιστός** (Semantic Web). Ο όρος Σημασιολογικός Ιστός αναφέρεται στο εγχείρημα εμπλουτισμού του υπάρχοντος ιστού με σημασιολογική πληροφορία, δηλαδή με πληροφορία που περιγράφει τα ίδια τα δεδομένα και τις σχέσεις μεταξύ τους. Στο όραμα του σημασιολογικού ιστού η σημασία και οι έννοιες στις οποίες αντιστοιχούν τα δεδομένα θα είναι κατανοητά από τις μηχανές , επιτρέποντας στις εφαρμογές του διαδικτύου να προσφέρουν καλύτερες υπηρεσίες στο χρήστη.

Σε αντίθεση με την σημερινή μορφή του παγκόσμιου ιστού όπου η γλώσσα HTML χρησιμοποιείται για να περιγράψει έγγραφα , οι γλώσσες του σημασιολογικού ιστού περιγράφουν αφηρημένα αντικείμενα όπως ανθρώπους , έργα , και βιβλία. Για το σκοπό αυτό κάθε πληροφορία εκφράζετε με έναν πλούσιο σημασιολογικά τρόπο , ο οποίος είναι όχι μόνο μηχανικά αναγνώσιμος αλλά και μηχανικά κατανοήσιμος και επεξεργάσιμος. Η παρούσα μορφή του παγκόσμιου ιστού μπορεί να γίνει αντιληπτή ως ένα παγκόσμιο εξυπηρετητή αρχείων , ενώ ο επερχόμενος σημασιολογικός ιστός ως μια παγκόσμια βάση δεδομένων. Για το λόγο αυτό συχνά ο σημασιολογικός ιστός αποκαλείται και Ιστός Δεδομένων σε αντίθεση με τον Ιστό Εγγράφων που γνωρίζαμε μέχρι σήμερα.



Εικόνα 2.2 Η Αρχιτεκτονική του Σημασιολογικού Ιστού

Στη βάση της αρχιτεκτονικής του Σημασιολογικού Ιστού βρίσκονται τα **URIs** (Uniform Resource Identifiers). Αυτά προέρχονται από τα γνωστά URLs (Uniform Resource Locators) τα οποία χρησιμοποιούνται για την διευθυνσιοδότηση πόρων προσπελάσιμων από το διαδίκτυο. Τα URI όμως επεκτείνουν τη λογική των URL και προσφέρουν αναγνωριστικά τόσο για δικτυακούς πόρους προσπελάσιμους από το διαδίκτυο (ιστοσελίδες, αρχεία , routers, printers) όσο και για αφηρημένες έννοιες ι αντικείμενα μη προσπελάσιμα από αυτό όπως είναι οι άνθρωποι, τα αυτοκίνητα και οι εταιρείες.

Στο επόμενο επίπεδο της αρχιτεκτονικής συναντάμε τις **γλώσσες για την ανταλλαγή δεδομένων**. Το βασικό μοντέλο αυτήν των γλωσσών βασίζεται στο RDF (Resource Description Framework). Το RDF είναι το βασικό μοντέλο που χρησιμοποιεί ο σημασιολογικός ιστός για την ανταλλαγή δεδομένων και χρησιμοποιείται για την αναπαράσταση των διαδικτυακών πόρων. Οι πόροι εκφράζονται πάντα με την χρήση URIs. Η αναπαράσταση των σχέσεων μεταξύ των πόρων πραγματοποιείται μέσω ενός συνόλου δηλώσεων (statements) οι οποίες αποτελούνται από τρία μέρη: (α) το υποκείμενο, (β) το κατηγορούμενο και (γ) το αντικείμενο. Λόγω της δομής τους, οι RDF δηλώσεις ονομάζονται και triples. Οι δηλώσεις RDF επιτραπούν τον μηχανικό τρόπο κατανόησης σχέσεων αντικειμένων που περιγράφονται από δεδομένα. Έτσι ενώ μια πρόταση της μορφής « Η Αθήνα είναι πρωτεύουσα της Ελλάδας» δεν σημαίνει απολύτως τίποτα για έναν υπολογιστή , αν η παραπάνω πρόταση αποδοθεί με κατάλληλες δηλώσεις RDF , μπορεί να εκφράσει ότι : 1) η Αθήνα είναι μια πόλη , 2) η Ελλάδα είναι μια χώρα και 3) ότι πρωτεύουσα της τελευταίας είναι η Αθήνα.

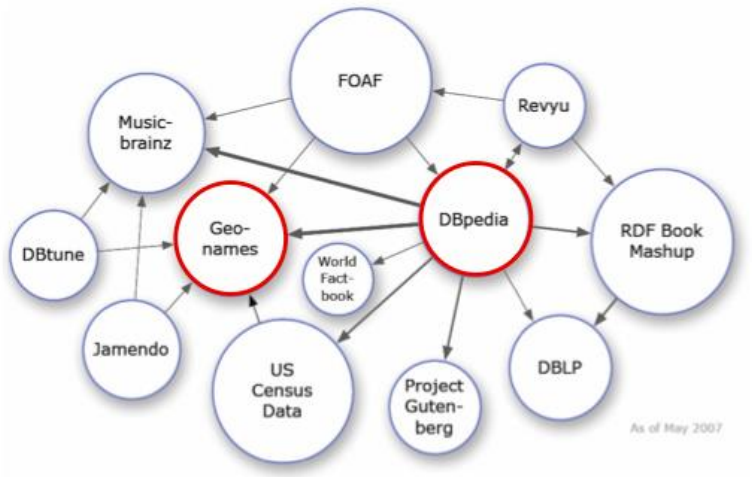
Στο ανώτερο επίπεδο της αρχιτεκτονικής του Σημασιολογικού Ιστού , βρίσκονται οι **γλώσσες περιγραφής οντολογιών**. Ο σημασιολογικός ιστός απαιτεί την δημοσίευση δεδομένων με χρήση γλωσσών ειδικά σχεδιασμένων για περιγραφή οντοτήτων (OWL , RDFa , RDFS). Αυτές οι γλώσσες βασίζονται στο μοντέλο δεδομένων RDF (Resource Description Framework) ενώ το συντακτικό τους στην γλώσσα XML. Υπάρχουν επίσης **γλώσσες διατύπωσης ερωτημάτων** όπως η SPARQL , μέσω των οποίων συνδεδεμένα σύνολα δεδομένων που βρίσκονται σε διαφορετικά μέρη του παγκόσμιου ιστού μπορούν να συνδυαστούν και να διατυπώνουν πολύπλοκα ερωτήματα σε πραγματικό χρόνο.

Ανοιχτά συνδεδεμένα Δεδομένα

Ο Συνδεδεμένος Ιστός είναι ήδη μια πραγματικότητα και σύμφωνα με πολλούς ειδικούς αυτό που απομένει είναι η καθολική επικράτηση του, η οποία θα οδηγήσει στην πλήρη κατάργηση του υπάρχοντος μοντέλου. Υπάρχουν όμως αρκετοί που αμφισβητούν αυτήν την βεβαιότητα και επισημαίνουν ότι τέτοιες θεωρήσεις αγνοούν τα κίνητρα και τις επιθυμίες των χρηστών οι οποίοι θα κληθούν να περιγράψουν οντότητες και χαρακτηριστικά του πραγματικού κόσμου στο πολύπλοκο σχήμα της αρχιτεκτονικής του σημασιολογικού ιστού. Επιπλέον κριτικές , επισημαίνουν ότι οι γλώσσες οντολογιών είναι ιδιαίτερα δύσκολες και

πολύπλοκες ενώ και η σήμανση και προετοιμασία των δεδομένων απαιτεί τεχνική κατάρτιση.

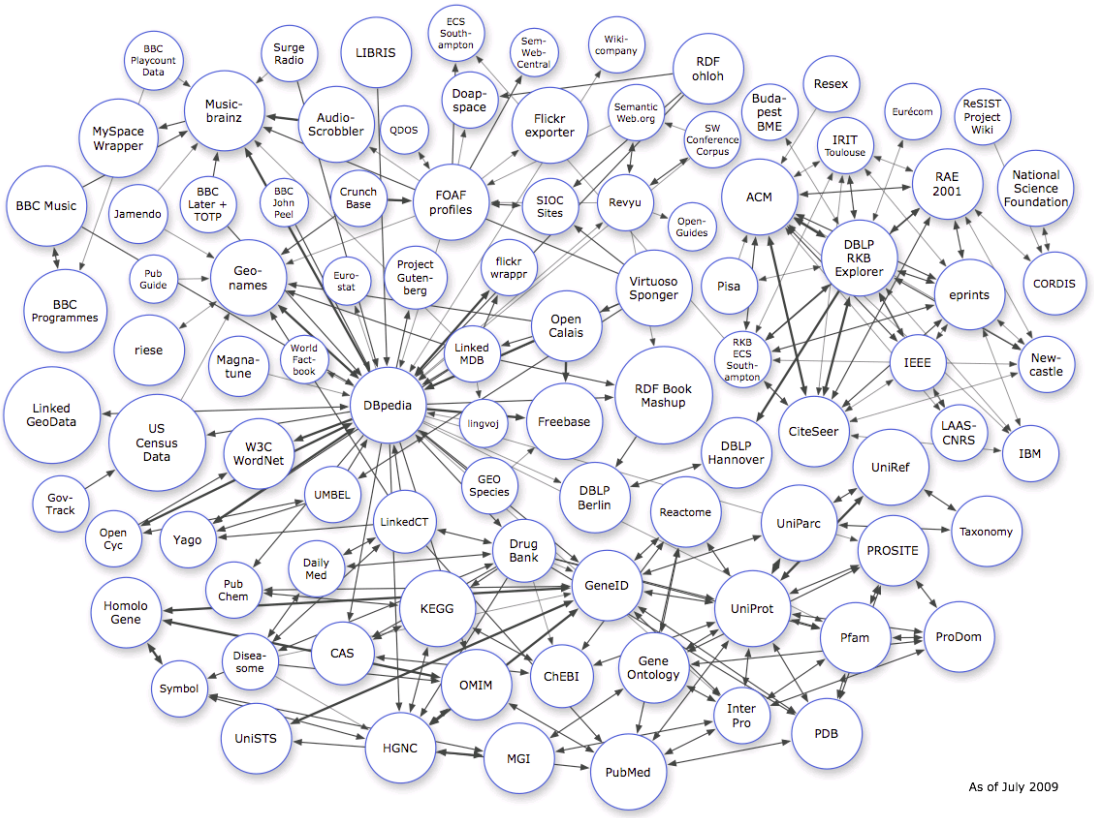
Ανεξάρτητα πάντως από τις κριτικές , είναι πράγματι γεγονός ότι μέχρι πρόσφατα ο ρυθμός υλοποίησης του σημασιολογικού ιστού δεν ήταν γενικά ικανοποιητικός. Σε μια προσπάθεια να δοθεί περαιτέρω ώθηση στην διασύνδεση των δεδομένων έχει ξεκινήσει ένα project υπό τον τίτλο **Ανοιχτά Συνδεδεμένα Δεδομένα** , μία κοινοτική προσπάθεια που θεμελιώθηκε τον Ιανουάριο του 2007 και υποστηρίχθηκε από την W3C . Η βασική ιδέα του εγχειρήματος αυτού είναι το «χτίσιμο» του Ιστού των Δεδομένων, εντοπίζοντας υπάρχοντα σύνολα δεδομένων διαθέσιμα κάτω από ανοιχτές άδειες, μετατρέποντάς τα σε RDF δεδομένα σύμφωνα με τις αρχές των Συνδεδεμένων Δεδομένων και, τέλος, δημοσιεύοντάς τα στον Ιστό . Έτσι επικεντρώνει τις απαιτήσεις στη δημοσίευση δομημένων δεδομένων με χρήση RDF και URIs και εστιάζει λιγότερο στο θέμα των οντολογιών. Η απλοποίηση αυτή στοχεύει στη μείωση των εμποδίων συμμετοχής για τους παρόχους δεδομένων. Η βασική υπόθεση είναι ότι πρέπει να δοθεί έμφαση στην διασύνδεση δεδομένων καθώς έτσι αυξάνεται η αξία και η αναζητησιμότητα τους. Όπως ακριβώς το παραδοσιακό διαδίκτυο των HTML εγγράφων μπορεί να σαρωθεί από μηχανής αναζήτησης ακλουθώντας τους υπερσυνδέσμους ανάμεσα στα έγγραφα, έτσι γίνεται και με τα Συνδεδεμένα Δεδομένα ακλουθώντας όμως RDF συνδέσμους. Κατ' αυτό το τρόπο οι μηχανής αναζήτησης μπορούν να παρέχουν εκλεπτυσμένες δυνατότητες εντοπισμού υποσυνόλων δεδομένων , παρόμοιες αυτήν που παρέχονται από σχεσιακής βάσεις δεδομένων.



Εικόνα 2.3 : Το σύννεφο των Συνδεδεμένων Ανοιχτών Δεδομένων – Μάιος 2007

Παρατηρούμε ότι στο σύννεφο των Συνδεδεμένων Ανοιχτών Δεδομένων υπάρχουν κάποιοι κεντρικοί κόμβοι, με τους οποίους συνδέονται αρκετά από τα υπόλοιπα σύνολα δεδομένων. Ο κεντρικότερος από αυτούς είναι η DBpedia, ένα από τα πρώτα σύνολα δεδομένων που εμφανίστηκαν στον Ιστό των Συνδεδεμένων Ανοιχτών Δεδομένων. Η DBpedia εξάγει πληροφορίες από τα

infoboxes της Wikipedia και τις δημοσιεύει ως Συνδεδεμένα Δεδομένα. Infoboxes είναι τα πλαίσια που βρίσκονται στην πάνω δεξιά πλευρά των σελίδων της Wikipedia και περιέχουν μερικές δομημένες πληροφορίες σχετικά με το αντικείμενο του άρθρου. Ένα άλλο βασικό σύνολο δεδομένων στο σύννεφο των Συνδεδεμένων Ανοιχτών Δεδομένων είναι το Geonames. Το Geonames δημοσιεύει περιγραφές εκατομμυρίων γεωγραφικών τοποθεσιών από όλο τον κόσμο σε μορφή Συνδεδεμένων Ανοιχτών Δεδομένων. Η DBpedia και το Geonames προσφέρουν URIs και πληροφορίες για πολλά από τα αντικείμενα του κόσμου μας στα οποία συχνά θέλουμε να αναφερθούμε. Αυτός είναι και ο λόγος που εξελίχθηκαν σε δύο κεντρικούς κόμβους στο σύννεφο των Συνδεδεμένων Ανοιχτών Δεδομένων.



Εικόνα 2.4 Το σύννεφο των Συνδεδεμένων Ανοιχτών Δεδομένων – Ιούλιος 2009

Η δημοσίευση σε μορφή Συνδεδεμένων Δεδομένων των δεδομένων που περιέχονται σε ένα σύνολο δεδομένων συνδέεται άμεσα με το dereferencing των URIs των πόρων που περιέχονται στο σύνολο αυτό. Εκτός από τη διαδικασία του dereferencing, ένα σύνολο δεδομένων προσφέρει στους πελάτες τη δυνατότητα να εξάγουν πληροφορίες για τους πόρους που τους ενδιαφέρουν και με άλλους τρόπους. Σε αυτούς περιλαμβάνεται η προσφορά ολόκληρης της αποθήκης (dump) και η δυνατότητα σύνδεσης με κάποιο SPARQL ακροδέκτη (SPARQL endpoint), ο οποίος δέχεται από τους πελάτες SPARQL ερωτήματα σχετικά με τα δεδομένα του συγκεκριμένου συνόλου δεδομένων, τα εκτελεί και επιστρέφει σε

αυτούς τις απαντήσεις. Ένα σύνολο δεδομένων μπορεί να προσφέρει τα δεδομένα του με έναν ή περισσότερους από τους προαναφερθέντες τρόπους.

Για να προχωρήσει γρηγορότερα η διασύνδεση των δεδομένων , η πρωτοβουλία για τα Ανοιχτά Συνδεδεμένα Δεδομένα , προτείνει τα ακόλουθα στους παρόχους δεδομένων

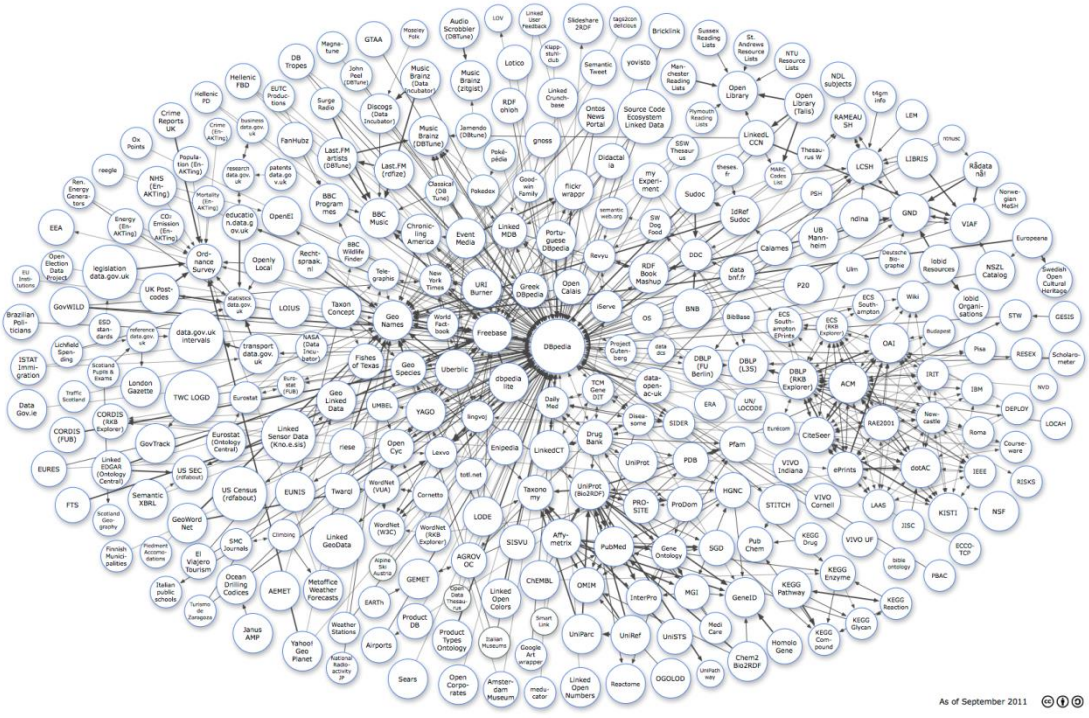
1. Χρήση URI's ως ονόματα αντικειμένων.

2. Χρήση HTTP URI's ώστε οι άνθρωποι να μπορών να ψάξουν αυτά τα ονόματα.

3. Χρήση προτύπων (RDF,SPARQL) για την παροχή πληροφοριών κατά την αναζήτηση ενός URI.

4. Αναφορά σε άλλα URI έτσι ώστε να ανακαλύπτονται περισσότερα αντικείμενα.

Η απλοποίηση που επιτεύχθηκε μέσω των κανόνων που έθεσε η πρωτοβουλία για τα Ανοιχτά Συνδεδεμένα Δεδομένα έφερε πράγματι αποτελέσματα. Τον Οκτώβριο του 2007, τα σύνολα συνδεδεμένων δεδομένων αριθμούσαν πάνω από 2 δισεκατομμύρια RDF τριπλές, συσχετισμένες μεταξύ τους με πάνω από 2 εκατομμύρια συνδέσμους RDF. Μέχρι τον Σεπτέμβριο του 2011, είχαν αυξηθεί σε 31 δισεκατομμύρια RDF τριπλέτες, συσχετισμένες μεταξύ τους με περίπου 504 εκατομμύρια συνδέσμους RDF.



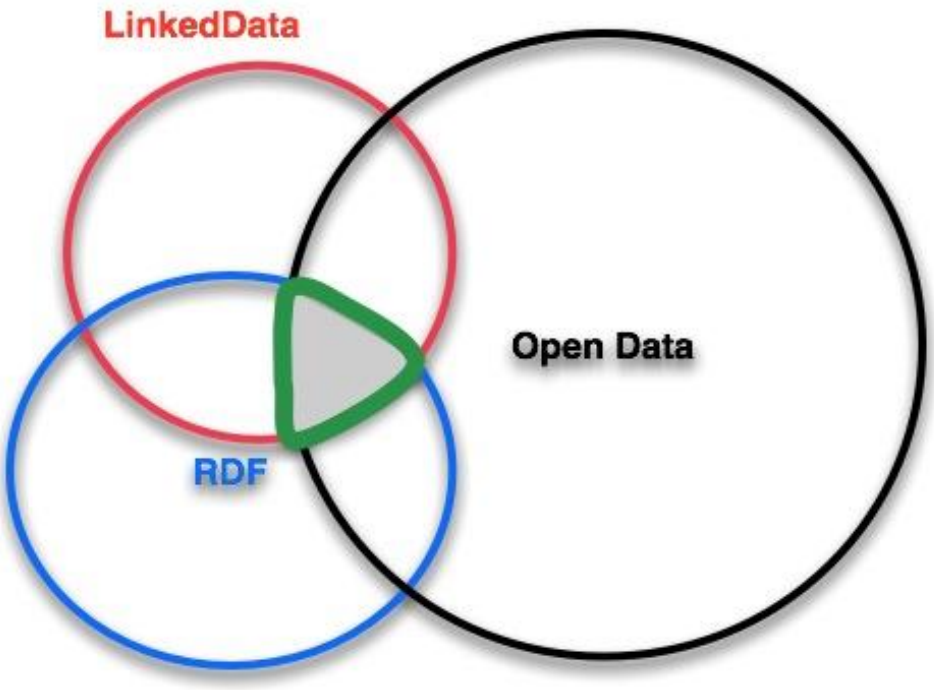
Εικόνα 2.5 Το σύννεφο των Συνδεδεμένων Ανοιχτών Δεδομένων – Σεπτέμβριος 2011

Ολοκληρώνοντας , είναι σημαντικό να διευκρινιστεί ότι οι όροι Συνδεδεμένα Δεδομένα και Ανοιχτά Δεδομένα δεν ταυτίζονται ούτε αποτελούν προϋπόθεση ο ένας του άλλου.

1. Τα δεδομένα μπορεί να είναι ανοιχτά, ενώ δεν είναι συνδεδεμένα

2. Τα δεδομένα μπορεί να είναι συνδεδεμένα, ενώ δεν είναι ανοιχτά

3. Τα δεδομένα όταν είναι ανοιχτά και συνδεδεμένα αποκτούν μεγαλύτερη βιωσιμότητα και αξία αυξάνοντας την δυνατότητα εντοπισμού τους.



Εικόνα 2.6 Σχέση Ανοιχτών Δεδομένων, Συνδεδεμένων και RDF

2.5 Τα Ανοιχτά Συνδεδεμένα Δεδομένα στα ΜΜΜ

Τα τελευταία χρόνια ολοένα και περισσότεροι άνθρωποι δεν χρησιμοποιούν δικό τους μέσο μεταφοράς και εμπιστεύονται τα Μέσα Μαζικής Μεταφοράς για τις μετακινήσεις τους. Αυτό συμβαίνει είτε για οικονομικούς λόγους (οικονομική κρίση, αύξηση τιμών καυσίμων , συντήρηση μέσου) , είτε για πρακτικούς λογούς (parking). Συνεπώς υπάρχει ολοένα κι αυξανόμενη ζήτηση που έχει οδηγήσει στην

ανάπτυξη όλων των ΜΜΜ. Με τη σειρά της η ανάπτυξη αυτή δημιουργεί νέα προβλήματα στους επιβάτες , ειδικότερα στους τουρίστες και εκείνους γενικότερα που τα χρησιμοποιούν λιγότερο.

Μερικά από τα σημαντικότερα προβλήματα ενός μέσου επιβάτη ενός ΜΜΜ είναι :

- Ποια είναι η ποιο γρήγορη διαδρομή από το σημείο Α μιας πόλης σε ένα σημείο Β ;
- Ποιο ΜΜΜ είναι ποιο οικονομικό για μία διαδρομή από το σημείο Α στο σημείο Β ;
- Με ποιο μέσο είναι καλύτερα να κάνω μία περιήγηση στην πόλη , βλέποντας τα αξιοθέατα
- Υπάρχουν χώροι στάθμευσης για το αυτοκίνητό μου στη συγκεκριμένη στάση του μετρό ;
- Μπορώ να ενημερώνομαι για τη πορεία του ΜΜΜ κι αν πλησιάζει ;

Πριν τη ραγδαία ανάπτυξη της τεχνολογίας τα τελευταία χρόνια φάνταζε ουτοπικό να μπορέσουμε να λύσουμε κάποια από αυτά τα προβλήματα. Το έντυπο υλικό που είναι διαθέσιμο, χάρτες, φυλλάδια, ακόμη και τα αναρτημένα δρομολόγια στις στάσεις πολλές φορές μπορούν να οδηγήσουν σε αναξιόπιστα αποτελέσματα και να μπερδέψουν ακόμα περισσότερο τον επιβάτη. Αν αναλογιστούμε και το κόστος των εντύπων αυτών που χρειάζονται αλλαγή μετά από οποιαδήποτε αναβάθμιση σε δρομολόγια , στάσεις κλπ. τότε η στροφή στην ηλεκτρονική του μορφή είναι μονόδρομος.

Πάνω σε αυτό το πλαίσιο οι περισσότεροι οργανισμοί ΜΜΜ έχουν εξελίξει τις ιστοσελίδες τους και έχουν αναπτύξει αντίστοιχες εφαρμογές κινητών προσφέροντας ακόμα μεγαλύτερη ευχέρεια στου χρηστές τους. Μια τέτοια ιστοσελίδα είναι και η ιστοσελίδα του ΟΑΣΘ (Οργανισμός Αστικών Συγκοινωνιών Θεσσαλονίκης), το www.oasth.gr από όπου μπόρεσα να ανακτήσω τα δεδομένα για την εφαρμογή μου.

Η «στροφή» αυτή είναι αλήθεια έχει προσφέρει λύσεις σε μερικά από τα προβλήματα που αναφέραμε. Οι οργανισμοί ΜΜΜ για να συνεχίσουν όμως προς τη κατεύθυνση αυτή είναι αναπόφευκτο πως τα δεδομένα τους πρέπει να γίνουν ανοικτό . Αναλύσαμε παραπάνω εμπεριστατώμενα πως τα ανοικτά δεδομένα και ειδικότερα τα συνδεδεμένα οδηγούν σε άνθηση εφαρμογών ικανών να αντιμετωπίσουν κάθε ειδών προβλήματα σε διαφόρους τομείς> Ένας τέτοιος τομέας είναι σίγουρα ο τομέας των ΜΜΜ.

Ο ΟΑΣΑ (Οργανισμός Αστικών Συγκοινωνιών Αθήνας) από τα μέσα του 2012 έχει αναπτύξει με τη βοήθεια του Phil Stubbings υπηρεσία πολυμεσικής δρομολόγησης με τα ανοικτά δεδομένα του. Η εφαρμογή βρίσκεται στην ιστοσελίδα www.zee.gr . Τα δεδομένα που χρησιμοποιούσε αρχικά η εφαρμογή προέρχονταν από το

ελάχιστο ηλεκτρονικό και κυρίως το έντυπο υλικό που ήταν διαθέσιμο .το οποίο ψηφιοποιήθηκε με το χέρι, καθώς δεν υπήρχε διαθέσιμη η πρωτογενής πληροφορία σε ηλεκτρονική μορφή και οργανώθηκαν από το μηδέν σε μια βάση δεδομένων.

Η ελεύθερη διάθεση των δρομολογίων από τον ΟΑΣΑ όμως ήταν αυτή έδωσε νέα πνοή στην εφαρμογή και επέτρεψε την ενασχόληση με αυτό που πραγματικά μετράει: την ανάπτυξη μιας χρήσιμης υπηρεσίας και όχι τη χειρωνακτική προσπάθεια για τη συλλογή της πληροφορίας. Τα δεδομένα δρομολόγησης του ΟΑΣΑ ακόμη δεν είναι τέλεια και βελτιώνονται διαρκώς. Ο κ. Stubbings προχώρησε στη διόρθωση αρκετών λαθών και στην ελεύθερη διάθεση της δουλειάς του, ώστε να μπορούν να αξιοποιηθούν από κάθε άλλον.

Δυστυχώς αντίστοιχη υπηρεσία και διάθεση ανοικτών δεδομένων του ΟΑΣΘ δεν υφίστανται αυτή τη στιγμή καθιστώντας εξαιρετικά δύσκολο το έργο των developers εφαρμογών για τα δρομολόγια του ΟΑΣΘ. Με αυτή την ευκαιρία καλώ όλους τους φορείς αστικών και δημοτικών συγκοινωνιών της χώρας να προσφέρουν τα δεδομένα τους ελεύθερα σε όλους τους πολίτες.

3. ΑΠΑΙΤΗΣΕΙΣ ΚΑΙ ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

3.1 Απαιτήσεις

Στο κεφάλαιο αυτό θα αναλύσουμε τις απαιτήσεις, τα προβλήματα καθώς και τρόπο επίλυσής τους στην εφαρμογή OasthAlarm η οποία αναπτύχθηκε για smartphones με λειτουργικό Android.

Ο κύριος στόχος της εφαρμογής OasthAlarm είναι να ειδοποιεί το χρήστη της για την άφιξη του λεωφορείου του στη στάση. Η εφαρμογή μπορεί και ειδοποιεί πρωτύτερα το χρήστη της ανάλογα με το χρόνο που θα ορίσει αυτός. Επίσης δώσαμε τη δυνατότητα στο χρήστη να μπορεί να αποθηκεύει τις ειδοποιήσεις αυτές με σκοπό να τις χρησιμοποιήσει άμεσα στο μέλλον.

Βασική προϋπόθεση για την ανάπτυξη της εφαρμογής ήταν η χρήση ανοικτών δεδομένων του ΟΑΣΘ. Ο ΟΑΣΘ έχει θέσει σε λειτουργία εδώ και κάποια χρόνια, ένα σύστημα τηλεματικής με σκοπό τον εντοπισμό λεωφορείων, την διαχείριση της κυκλοφορίας και την πληροφόρηση των επιβατών σε «έξυπνες» στάσεις.

Ο ΟΑΣΘ έχει δημιουργήσει και δύο ιστοσελίδες που εμπεριέχουν το σύστημα αυτό. Η πρώτη είναι η www.oasth.gr, η οποία είναι και το επίσημο site του οργανισμού, στην οποία λειτουργούν υπηρεσίες όπως «Πληροφορίες Γραμμής», «Βέλτιστη Διαδρομή» κ.λπ. Η δεύτερη, το m.oasth.gr είναι μια ιστοσελίδα πιο φιλική για χρήση παρόμοιων υπηρεσιών μέσω smartphones.

Στο πρώτο στάδιο, λοιπόν, ο καθηγητής μου κι εγώ προσπαθήσαμε να έρθουμε σε επικοινωνία με αρμόδιους του οργανισμού και των ιστοσελίδων του. Σκοπός μας ήταν να αποκτήσουμε πρόσβαση στα δεδομένα, πράγμα ζωτικής σημασίας για την κατασκευή της εφαρμογής.Το αποτέλεσμα όμως ήταν αρνητικό για λόγους που ακόμα δεν έχουμε καταλάβει. Στο κεφάλαιο 3 προσπάθησα να εξηγήσω αναλυτικά τα Ανοιχτά Δεδομένα και τα πλεονεκτήματα που προσφέρουν σ' αυτούς που τα χρησιμοποιούν κι ακόμα περισσότερο σε αυτούς που τα διαθέτουν. Ο ΟΑΣΘ θα πρέπει να ανοίξει τα δεδομένα επίσης, για το λόγο ότι είναι δικαίωμα των πολιτών ως Οργανισμός Κοινής Ωφέλειας που είναι.

Η λύση στο πρόβλημα αυτό δόθηκε μέσω ενός άλλου τομέα των Ανοιχτών Δεδομένων και πιο συγκεκριμμένα του Λογισμικού Ανοιχτού Κώδικα. Μετά από αρκετή αναζήτηση στο Internet βρέθηκε μία βάση δεδομένων με τις γραμμές, τις στάσεις και τα δρομολόγια του ΟΑΣΘ. Το αρχείο oasth.db που χρησιμοποιείται στην εφαρμογή ήταν η βάση δεδομένων για μία άλλη εφαρμογή για τον ΟΑΣΘ, ο κώδικας της οποίας ήταν ανοιχτός και αναρτήμένος στον ιστότοπο code.google. Στη συνέχεια με τα κατάλληλα ερωτήμα Sql στην oasth.db μπορούμε να δημιουργήσουμε το URI από όπου μπορούμε να ανακτήσουμε συγκεγκριμμενα δεδομένα που αφορουν σε κάθε όχημα του ΟΑΣΘ και την άφιξή του σε κάθε στάση. Περισσότερα για την διαδικάσια αυτήν μπόρει κάποιος να δει στο κεφάλαιο 4 και πιο συγκεκριμμένα στα υποκεφάλαια 4.7 GetMethodEx.java και 4.8 ArivalTimes.java.

Μια τεχνική απαίτηση που μπορεί κάποιος εύκολα να διακρίνει είναι ότι η συσκευή πρέπει να είναι συνδεδεμένη στο διαδίκτυο, τουλάχιστον μέχρι να παραλάβει τα απαραίτητα δεδομένα από την m.oasth.gr. Ένα smartphone μπορεί να συνδέεται στο internet είτε μέσω ενός WiFi είτε αν έχει ενεργοποιημένο το 3G ή 4G του (ανάλογα με τις δυνατότητες της συσκευής).

Η εφαρμογή έπρεπε να είναι φιλική προς το χρήστη χωρίς δυσκολίες στο χειρισμό της. Τα κουμπιά, οι λίστες , τα textboxes και γενικά όλο το User Interface δημιουργήθηκε έτσι ώστε να συμβάλει τα μέγιστα ως προς τη κατεύθυνση αυτή. Η ταχύτητα είναι άλλη μια βασική απαίτηση της εφαρμογής. Ο χρήστης θα έπρεπε να τη δημιουργεί όσο το δυνατόν πιο γρήγορα την ειδοποίηση για την γραμμή και την στάση που ενδιαφέρεται. Για να δημιουργήσει κάποιος μία άμεση ειδοποίηση

χρειάζεται λιγότερο από 5 δευτερόλεπτα ακόμα και αν δεν είναι εξοικειωμένος με την εφαρμογή. Η δυνατότητα χρησιμοποίησης αποθηκευμένης ειδοποίησης έχει πολλά πλεονεκτήματα σ' αυτό το κομμάτι. Χρησιμοποιώντας κάποιος μία αποθηκευμένη ειδοποίηση μπορεί να την ενεργοποιήσει ακόμα πιο γρήγορα με δύο κινήσεις. Σχεδόν σε κάθε οθόνη της εφαρμογής υπάρχουν κείμενα που αλλάζουν δυναμικά ανάλογα με τις επιλογές του χρήστη με σκοπό να ενημερώνουν το χρήστη για τις επιλογές του και να τον αποτρέπουν από λάθη που μπορούν να συμβούν λόγω κεκτημένης ταχύτητας.

Η εφαρμογή αναπτύχτηκε σε android έκδοση 2.2 διότι πρόκειται ίσως για την πιο διαδεδομένη έκδοση που χρησιμοποιείται στην Ελλάδα. Διαλέγοντας αυτήν την έκδοση καλύπτουμε παραπάνω από το 90% του ποσοστού των συσκευών στα οποία η εφαρμογή είναι διαθέσιμη.

3.2 Περιγραφή

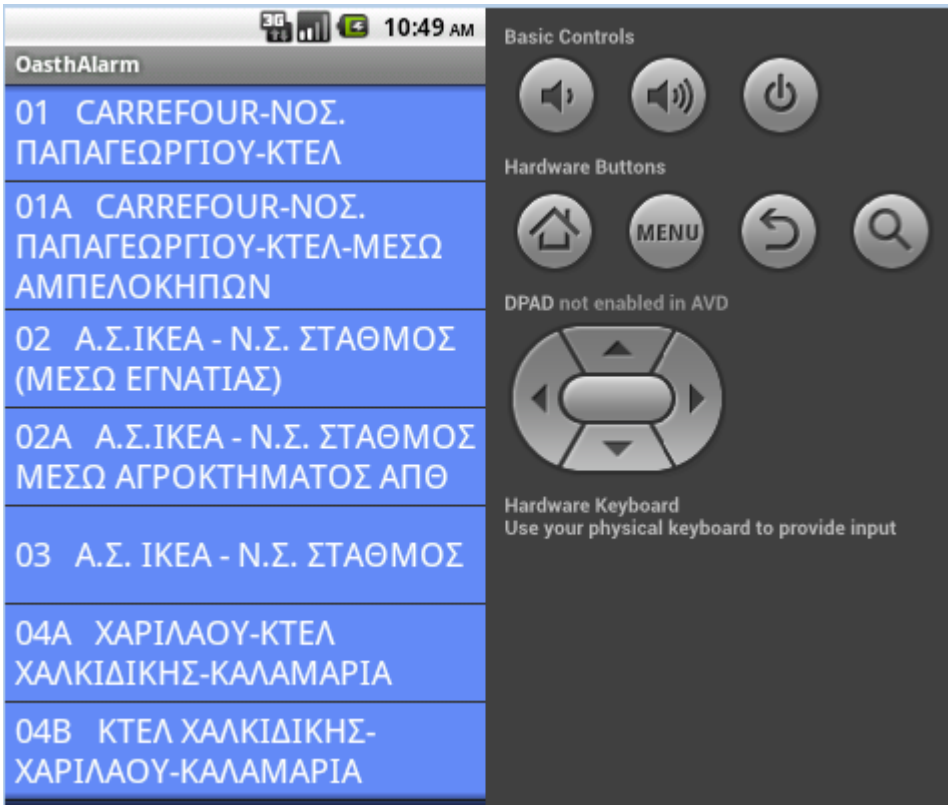
Στο υποκεφάλαιο αυτό θα προσπαθήσουμε να γνωρίσουμε την OasthAlarm και να διακρίνουμε την αναγκαιότητά της σε ορισμένα σενάρια χρησιμοποίησής της. Κάποιος θα μπορούσε να το χαρακτηρίσει και σαν ένα εγχειρίδιο της εφαρμογής.

Με την εκκίνηση της OasthAlarm εμφανίζεται στο κινητό μας η παρακάτω οθόνη (Εικόνα 3.1). Η απουσία splashscreen στην εφαρμογή είναι συνειδητή γιατί όπως προανέφερα βασική απαίτηση της εφαρμογής είναι η ταχύτητα της χρήσης της.

Εικόνα 3.1 Αρχική Οθόνη OasthAlarm

Στην πρώτη οθόνη δίνονται δύο επιλογές, η «Άμεση Δημιουργία Ειδοποίησης» και η «Αποθηκευμένες Ειδοποιήσεις». Για κάποιον που χρησιμοποιεί την εφαρμογή για πρώτη φορά η πρώτη επιλογή είναι δεδομένη καθώς δεν υπάρχουν αποθηκευμένες ειδοποιήσεις στη βάση.

Πατώντας λοιπόν κάποιος στο κουμπί «Άμεση Δημιουργία Ειδοποίησης» του εμφανίζεται μία λίστα με όλες τις γραμμές του ΟΑΣΘ. Κάθε στοιχείο της λίστας περιέχει τον αριθμό της γραμμής και ολοκληρωμένο το όνομά της. Ο χρήστης εδώ, πατώντας απλά πάνω σε κάποιο στοιχείο επιλέγει την γραμμή που τον ενδιαφέρει (Εικόνα 3.2).



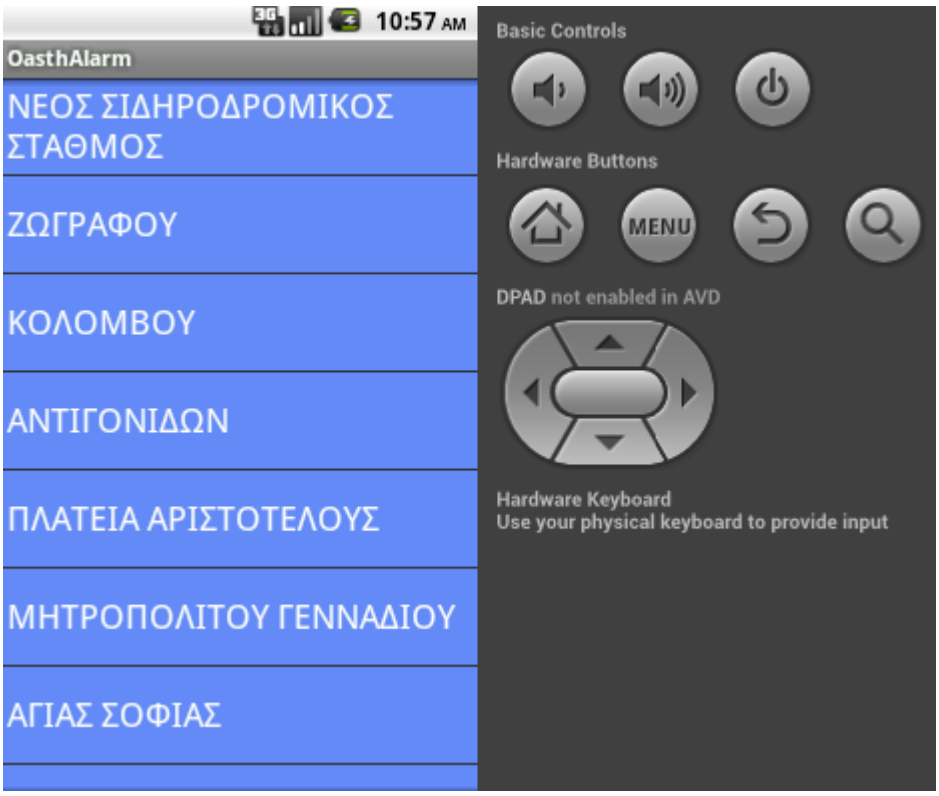
Εικόνα 3.2 Επιλογή Γραμμής ΟΑΣΘ

Στη συνέχεια η εφαρμογή ζητάει από τον χρήστη να επιλέξει τον προορισμό του με την παρουσία των κουμπιών «ΜΕΤΑΒΑΣΗ» και «ΕΠΙΣΤΡΟΦΗ». Στη συγκεκριμένη περίπτωση (Εικόνα 3.3) επιλέξαμε την γραμμή ΧΑΡΙΛΑΟΥ – Ν.Σ. ΣΤΑΘΜΟΣ. Αν ο χρήστης θέλει να ακολουθήσει τη διαδρομή από Χαριλάου προς Ν.Σ. Σταθμό θα πρέπει να επιλέξει το «Μετάβαση». Ενώ για την αντίθετη διαδρομή από Ν.Σ. Σταθμό προς Χαριλάου θα πρέπει να επιλέξει το «Επιστροφή».

Εικόνα 3.3 Επιλογή Προορισμού

Έχοντας επιλέξει κάποιος το «Μετάβαση» είτε το «Επιστροφή» στην επόμενη οθόνη θα εμφανιστεί μία νέα λίστα με όλες τις στάσεις της γραμμής που επιλέχθηκε. Οι στάσεις εμφανίζονται με τη σειρά που περνάει από αυτές το λεωφορείο και ανάλογα με την κατεύθυνση που έχει. Π.χ. στην Εικόνα 3.4 βλέπουμε τις στάσεις της γραμμής 10 με τη σειρά από την Τ.Σ. ΧΑΡΙΛΑΟΥ έως και το Ν.Σ. Σταθμό. Αυτό συμβαίνει αν ο χρήστης είχε επιλέξει το «ΜΕΤΑΒΑΣΗ» στη συγκεκριμένη γραμμή. Την αντίθετη περίπτωση μπορούμε να τη δούμε στην Εικόνα 3.5. Είναι η περίπτωση όπου ο χρήστης έχει επιλέξει το κουμπί «Επιστροφή» και έτσι οι στάσεις ταξινομούνται αντίστροφα με την πρώτη περίπτωση.

Εικόνα 3.4 Εμφάνιση και Επιλογή Στάσεων (ΜΕΤΑΒΑΣΗ)



Εικόνα 3.5 Εμφάνιση και Επιλογή Στάσεων (ΕΠΙΣΤΡΟΦΗ)

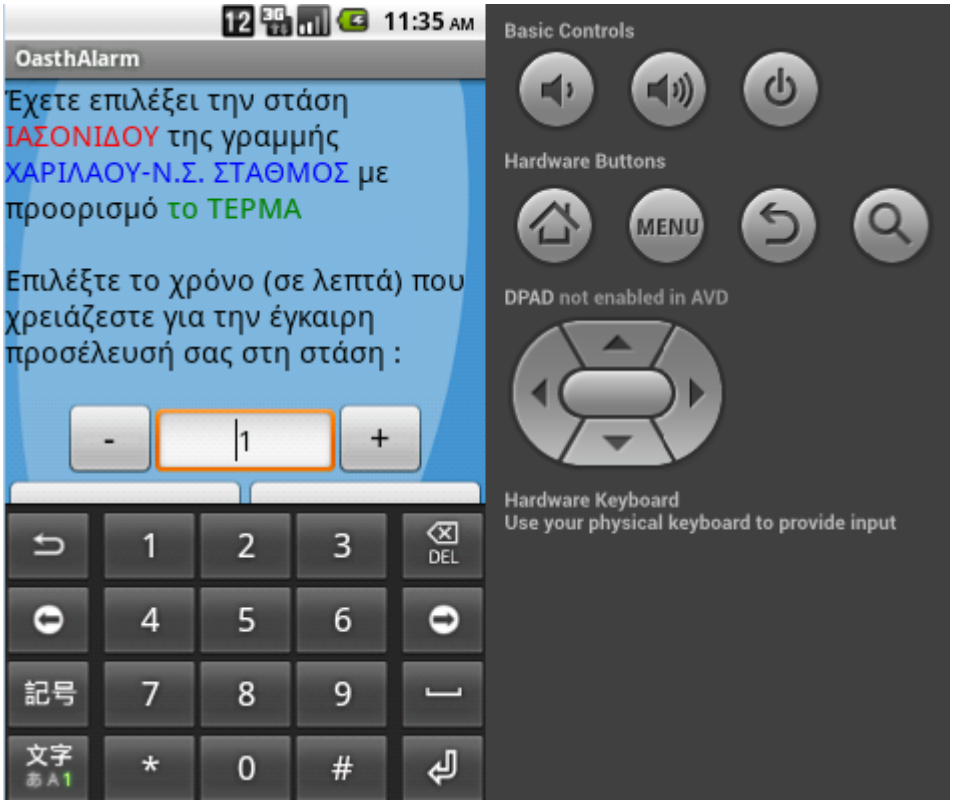
Όταν ο χρήστης κάνει κλικ πάνω σε μία στάση εμφανίζεται μία νέα οθόνη. Στην οθόνη αυτή η εφαρμογή ενημερώνει το χρήστη με ένα κείμενο για τις επιλογές που έχει κάνει ως τώρα. Στην Εικόνα 3.6 έχουμε επιλέξει τη στάση ΙΑΣΟΝΙΔΟΥ της γραμμής ΧΑΡΙΛΑΟΥ – Ν.Σ. ΣΤΑΘΜΟΣ με προορισμό το ΤΕΡΜΑ, δηλαδή τον σταθμό.

Εικόνα 3.6 Επιλογή Χρόνου Ειδοποίησης

Παρακάτω η εφαρμογή προτρέπει το χρήστη να επιλέξει ακριβώς το χρόνο σε λεπτά που χρειάζεται για να βρίσκεται έγκαιρα στη στάση. Αυτό μπορεί να το κάνει με δύο τρόπους, είτε χρησιμοποιώντας τα κουμπιά «+» και «-», είτε πληκτρολογώντας μέσα στο κουτί.

Αν ο χρήστης πατήσει το κουμπί «+» αυξάνεται κατά ένα η τιμή μέσα στο textbox, ενώ αν πατήσει «-» μειώνεται κατά ένα. Αξίζει να σημειωθεί πως η προεπιλεγμένη τιμή που είναι το ένα λεπτό, είναι και η μικρότερη δυνατή, οπότε αν κάποιος πατήσει το «-» στην περίπτωση της εικόνας δεν θα μειωθεί η τιμή από ένα σε μηδέν.

Ο χρήστης έχει και τη δυνατότητα να πληκτρολογήσει απευθείας το χρόνο που θέλει να επιλέξει. Πατώντας λοιπόν πάνω στο textbox εμφανίζεται το πληκτρολόγιο του android (Εικόνα 3.7) και εκεί πληκτρολογεί την επιλογή του.



Εικόνα 3.7 Επιλογή Χρόνου Ειδοποίησης (Πληκτρολόγιο)

Στη συνέχεια ο χρήστης μπορεί είτε να πατήσει κλικ στο «Επόμενο» και να συνεχίσει με την ειδοποίηση, είτε να πατήσει το κουμπί «Αποθήκευση» ούτως ώστε να κρατήσει τις ρυθμίσεις της ειδοποίησης για μελλοντική χρήση. Τις συνέπειες αυτής της επιλογής θα τις δούμε πιο κάτω σε αυτό το κεφάλαιο.

Αν ο χρήστης κάνει κλικ στο «Επόμενο» θα οδηγηθεί στην επόμενη activity. Εδώ όμως υπάρχουν αρκετοί περιορισμοί που θα τους παρουσιάσουμε έναν προς ένα με διαφορετικές περιπτώσεις – σενάρια.

Σενάριο 1: Η συσκευή του χρήστη δεν είναι συνδεδεμένη στο διαδίκτυο.
Βασική προϋπόθεση της λειτουργίας της εφαρμογής όπως προαναφέραμε στις απαιτήσεις, είναι η σύνδεση με την ιστοσελίδα m.oasth.gr ώστε να ανακτήσουμε τα δεδομένα που αφορούν στους χρόνους άφιξης των οχημάτων.

Σε περίπτωση λοιπόν που η συσκευή δεν είναι συνδεδεμένη εμφανίζεται ο διάλογος που βλέπουμε στην Εικόνα 3.8. Ο διάλογος δίνει δύο επιλογές στο χρήστη, είτε να επιλέξει το OK και να μεταβεί στις ρυθμίσεις δικτύου του ώστε να ενεργοποιήσει το WiFi του (αν του δίνεται η δυνατότητα) ή την «Ενεργοποίηση Δεδομένων», είτε να πατήσει το cancel και να επιστρέψει στην οθόνη «Επιλογή Χρόνου».

Εικόνα 3.8 Έλεγχος σύνδεσης δεδομένων.

<u>Σενάριο 2:</u> Ενεργοποιημένη σύνδεση της συσκευής, δεν υπάρχουν όμως οχήματα αυτή τη στιγμή που να βρίσκεται σε εξέλιξη το δρομολόγιό τους.

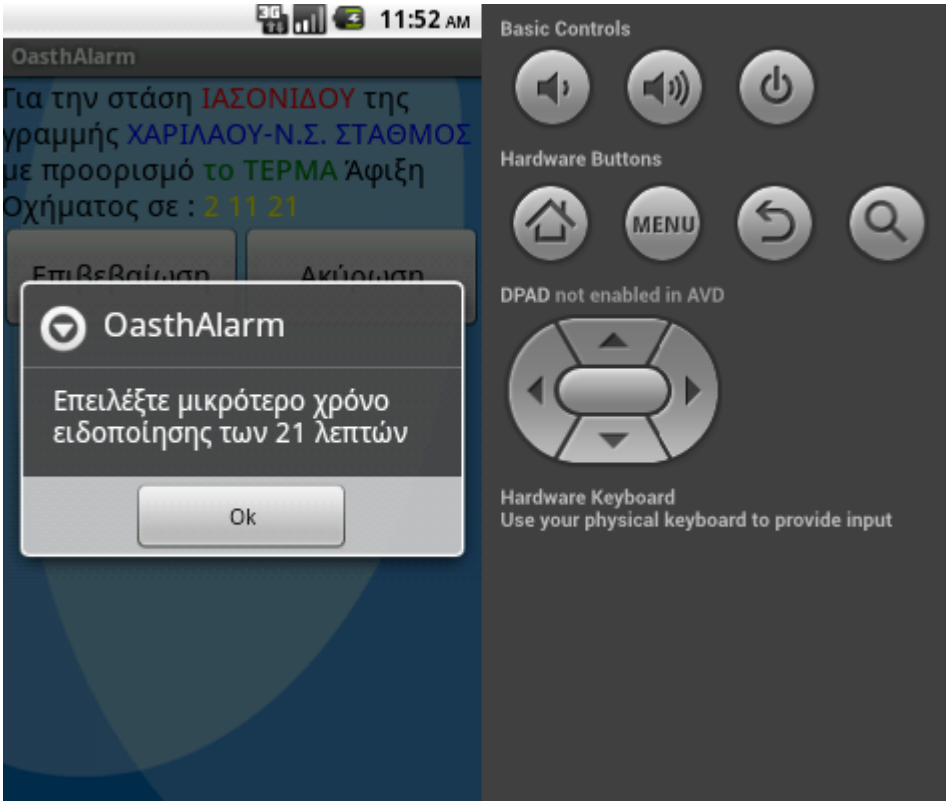
Στην περίπτωση αυτή ένας άλλος διάλογος εμφανίζεται στο χρήστη που τον ενημερώνει πως δεν υπάρχουν οχήματα αυτή τη στιγμή και τον προτρέπει να προσπαθήσει αργότερα. Όταν ο χρήστης πατήσει το κουμπί OK επιστρέφει στην προηγούμενη οθόνη (Εικόνα 3.9).

Εικόνα 3.9 Έλεγχος Αφίξεων (Χωρίς οχήματα)

<u>Σενάριο 3:</u> Υπάρχει ενεργή σύνδεση δεδομένων, όμως ο χρήστης έχει επιλέξει για χρόνο ειδοποίησης, πριν το όχημα φτάσει στη στάση, μεγαλύτερο χρόνο από το χρόνο άφιξης του λεωφορείου που θα έρθει αργότερα.

Για να το κατανοήσει κάποιος καλύτερα αυτό θα χρησιμοποιήσουμε ένα παράδειγμα. Έστω ότι ο χρήστης χρειάζεται 22 λεπτά για την έγκαιρη προσέλευση του στη στάση και ρυθμίζει την ειδοποίηση του ανάλογα στην προηγούμενη οθόνη. Οι αφίξεις των οχημάτων στις στάσεις όμως, αναμένεται να είναι σε 2,11 και 21 λεπτά. Εδώ μπορεί κάποιος εύκολα να καταλάβει πως δεν έχει νόημα να ειδοποιηθεί ο χρήστης σε 22 λεπτά εφόσον το αργότερο λεωφορείο θα περάσει σε 21 λεπτά.

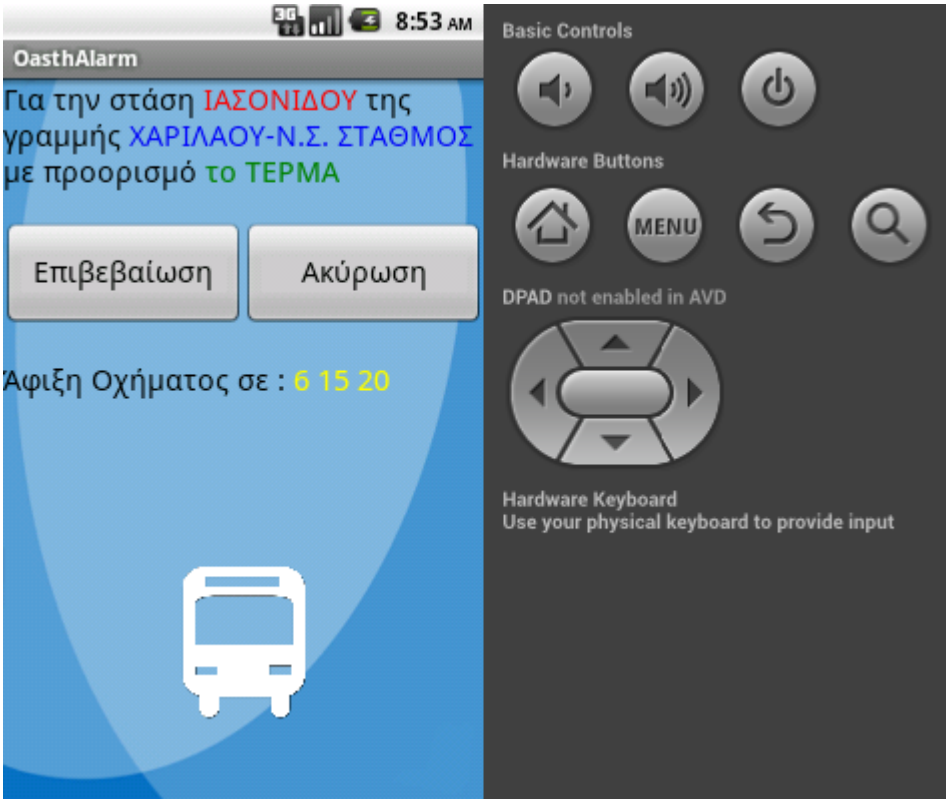
Έτσι για την αποφυγή αυτού του προβλήματος η εφαρμογή εμφανίζει ένα διάλογο (Εικόνα 3.10) που προτείνει στον χρήστη να επιλέξει χρόνο μικρότερο από τον μεγαλύτερο χρόνο άφιξης. Με κλικ στο OK ο χρήστης επιστρέφει στην προηγούμενη οθόνη.



Εικόνα 3.10 Έλεγχος χρόνου άφιξης σε σχέση με το χρόνο ειδοποίησης

Σενάριο 4: Στο σενάριο αυτό ισχύουν όλες οι προϋποθέσεις που δεν ισχύουν στα προηγούμενα σενάρια. 1. Η συσκευή είναι συνδεδεμένη στο internet, 2. υπάρχουν οχήματα που εκτελούν το δρομολόγιό τους προς τη στάση που επιλέχτηκε και 3. ο χρήστης στην προηγούμενη οθόνη επέλεξε χρόνο μικρότερο του χρόνου άφιξης ενός τουλάχιστον οχήματος.

Αυτό το σενάριο είναι και το απαιτούμενο σενάριο και μπορούμε να δούμε πώς εξελίσσεται στη συνέχεια η εφαρμογή. Έτσι αν ο χρήστης πατήσει «Επόμενο» στην προηγούμενη οθόνη εμφανίζεται μία νέα. Εδώ υπάρχει πάλι ένα κείμενο που ενημερώνει το χρήστη για τις ως τώρα επιλογές του. Επιπλέον εμφανίζονται και οι χρόνοι άφιξης των οχημάτων που εκτελούν το δρομολόγιο αυτή τη στιγμή έτσι ώστε ο χρήστης να αποκτήσει μια εικόνα για την πορεία τους (Εικόνα 3.11).



Εικόνα 3.11 Οθόνη για την δημιουργία ειδοποίησης

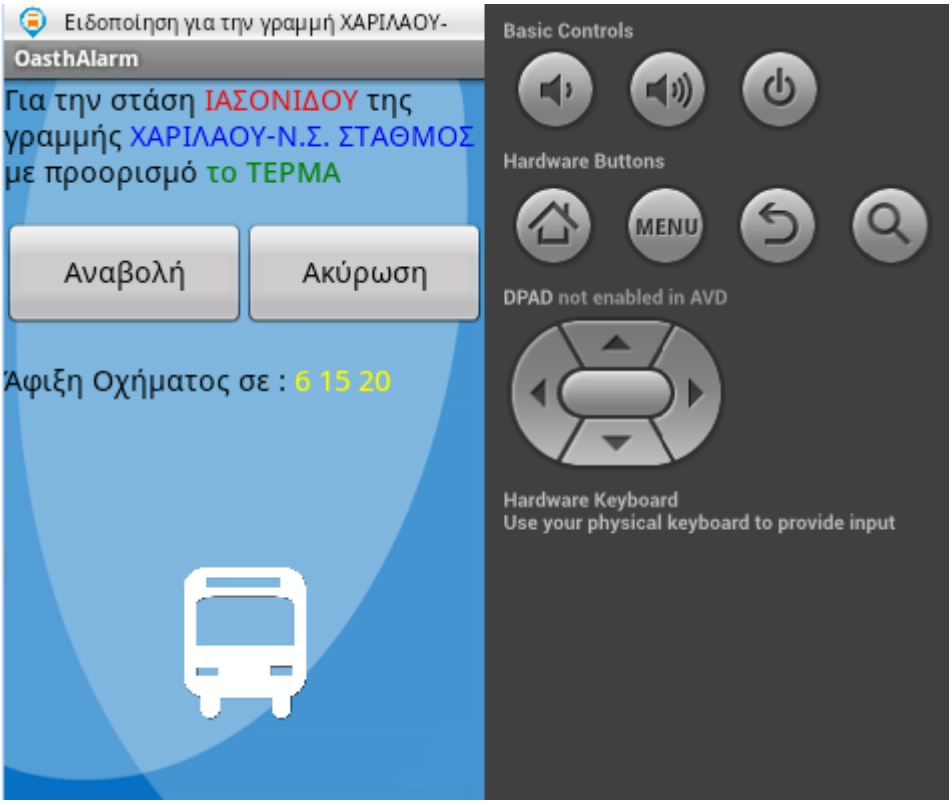
Τελικά, όταν ο χρήστης πατήσει «Επιβεβαίωση» δημιουργείται η ειδοποίηση και εμφανίζεται το εικονίδιο της ειδοποίησης επάνω αριστερά στο χώρο των ειδοποιήσεων, όπως φαίνεται και στην εικόνα 3.12.

Ο χρόνος της ειδοποίησης θα εξαρτηθεί σε σχέση με ένα μοναδικό χρόνο της άφιξης του οχήματος και την επιλογή του χρόνου της έγκαιρης προσέλευσης από το χρήστη στην προηγούμενη οθόνη. Για να κατανοήσουμε καλύτερα τη λειτουργία της εφαρμογής θα χρησιμοποιήσουμε το παράδειγμα που φαίνεται στην Εικόνα 3.12. Οι χρόνοι άφιξης των λεωφορείων όμως εδώ είναι τρείς και είναι ταξινομημένοι με αύξουσα σειρά 6,15 και 20 λεπτά (είναι τα νούμερα με κίτρινο χρώμα στην οθόνη). Η εφαρμογή θα επιλέξει εκείνο το λεωφορείο που επαληθεύει τις απαιτήσεις:

1. ο χρόνος άφιξής του είναι μεγαλύτερος του χρόνου έγκαιρης προσέλευσης του χρήστη (που επέλεξε νωρίτερα).
2. το νωρίτερο χρόνο άφιξης αν είναι παραπάνω από ένα τα λεωφορεία που επαληθεύουν την πρώτη απαίτηση.

Έτσι αν ο χρήστης εδώ επιλέξει τα 6 λεπτά σαν χρόνο έγκαιρης ειδοποίησης, σύμφωνα με τη πρώτη απαίτηση η εφαρμογή θα κληθεί να επιλέξει μεταξύ των οχημάτων που φτάνουν σε 15 και 20 λεπτά. Ο πρώτος χρόνος άφιξης απορρίπτεται γιατί είναι ίσος με τον χρόνο έγκαιρης προσέλευσης και δεν έχει νόημα η ειδοποίηση να χτυπήσει αμέσως μετά την ενεργοποίηση της. Τελικά η εφαρμογή θα επιλέξει το λεωφορείο που φτάνει σε 15 λεπτά και νωρίτερα από

αυτό που φτάνει σε 20 λεπτά, σύμφωνα με την δεύτερη απαίτηση. Συμπεραίνουμε ότι η ειδοποίηση θα χτυπήσει ακριβώς 15-6=9 μετά την ενεργοποίηση της.

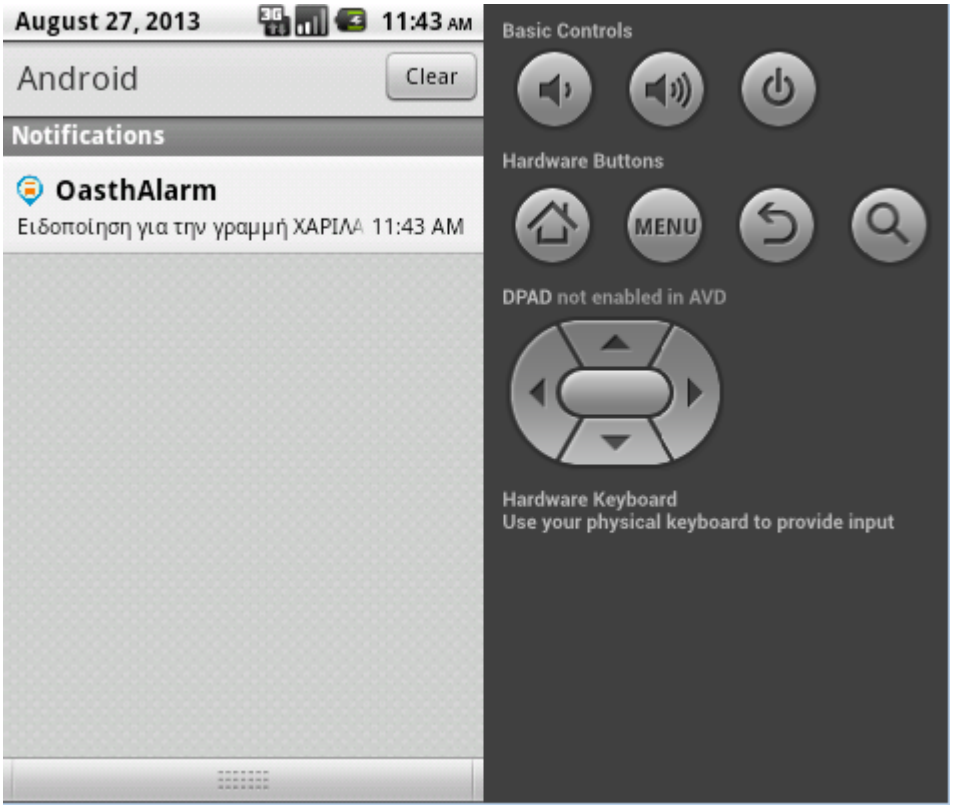


Εικόνα 3.12 Δημιουργία Ειδοποίησης

Μια ειδοποίηση είναι μια υπηρεσία του Android. Οι Services τρέχουν πίσω από την δραστηριότητα που εκτελεί το Android εκείνη τη στιγμή και ανεξάρτητα με την κατάσταση του Activity που δημιουργήθηκε. Έτσι μπορούμε να βγούμε από την εφαρμογή OasthAlarm και η ειδοποίηση να υπάρχει στο background, το εικονίδιο της ειδοποίησης πάνω αριστερά αυτό ακριβώς υποδηλώνει (Εικόνα 3.13).

Εικόνα 3.13 Ενεργή Ειδοποίηση (Εικονίδιο πάνω αριστερά).

Ο χρήστης, λοιπόν, σέρνοντας προς τα κάτω το μενού ειδοποιήσεων μπορεί να δει και την ειδοποίηση του OasthAlarm.Στο κείμενο της ειδοποίησης αναφέρεται η γραμμή του ΟΑΣΘ για την οποία δημιουργήθηκε η ειδοποίηση. Αν πατήσει σε αυτή εμφανίζεται η δραστηριότητα από όπου ενεργοποίησε την ειδοποίηση. Στη θέση του button «Επιβεβαίωση» υπάρχει πλέον το button «Αναβολή». Αυτό συμβαίνει από τη στιγμή που κάποιος θα πατήσει την «Επιβεβαίωση» και θα δημιουργηθεί η Ειδοποίηση (Εικόνα 3.14).



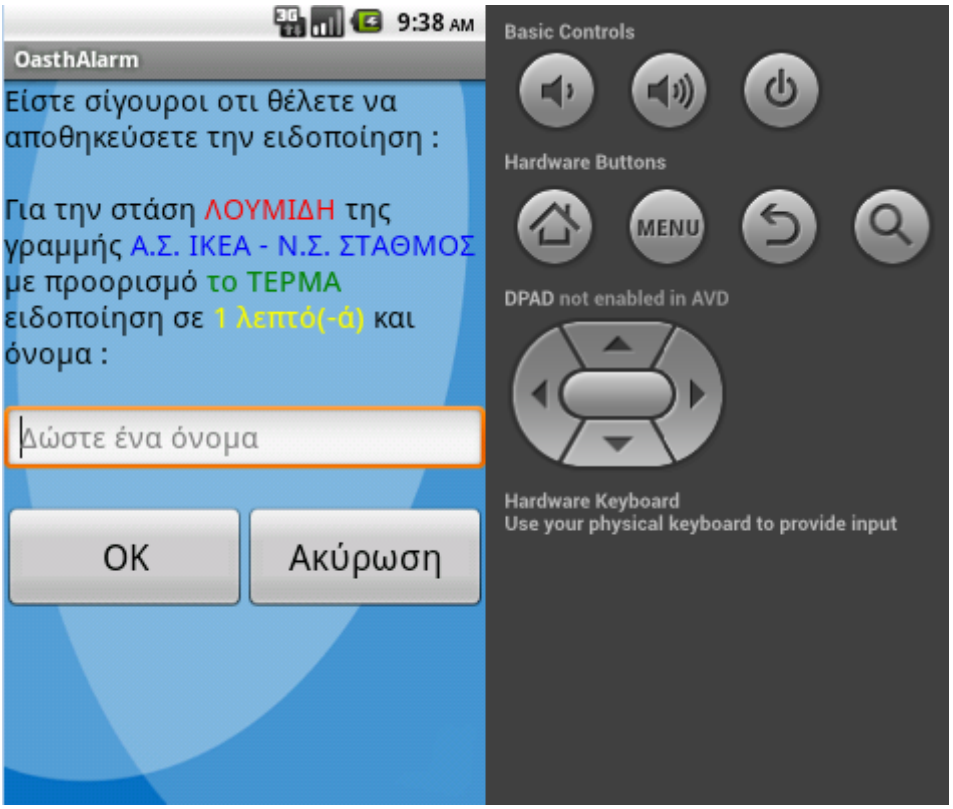
Εικόνα 3.14 Άνοιγμα Ειδοποίησης

Αν ο χρήστης πατήσει «Αναβολή» διαγράφει την υπάρχουσα ειδοποίηση και δημιουργεί μία νέα. Η νέα ειδοποίηση δημιουργείται ακριβώς με τις ίδιες προϋποθέσεις της παλιάς. Έτσι αφορά την ίδια γραμμή, τον προορισμό, τη στάση και το χρόνο έγκαιρης ειδοποίησης. Το μόνο που αλλάζει είναι οι χρόνοι άφιξης των οχημάτων που ανανεώνονται. Η νέα ειδοποίηση λοιπόν θα ενημερώσει το χρήστη ανάλογα με τους νέους χρόνους. Για παράδειγμα αν ο χρήστης είχε επιλέξει για χρόνο ειδοποίησης τα 3 λεπτά και οι χρόνοι άφιξης των οχημάτων ήταν σε 5 και 8 λεπτά η πρώτη ειδοποίηση που θα δημιουργούσε θα τον ενημέρωνε 2 λεπτά μετά τη δημιουργία της. Αν μέσα στο χρονικό όριο των 2 λεπτών πριν η ειδοποίηση «χτυπήσει», ο χρήστης πατήσει «Αναβολή» θα δημιουργηθεί μια νέα ειδοποίηση στη θέση της προηγούμενης. Το ερώτημα τώρα είναι για ποιο από τα δύο λεωφορεία θα χτυπήσει η ειδοποίηση. Είδαμε παραπάνω πως η ειδοποίηση επιλέγει τη πιο σύντομη άφιξη οχήματος σε στάση.

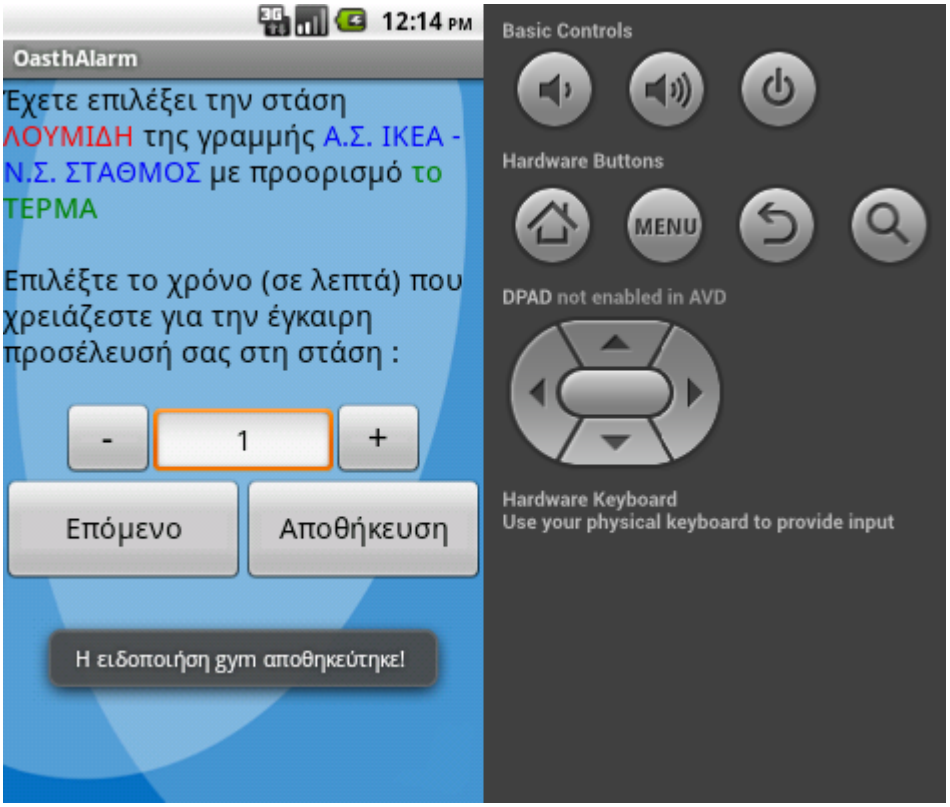
- Αν ο χρήστης αναβάλει την ειδοποίηση σε χρόνο μικρότερο του ενός λεπτού μετά την πρώτη ενεργοποίηση, οι αφίξεις των οχημάτων λογικά θα έχουν διαμορφωθεί σε 4 και 7 λεπτά. Έτσι η ειδοποίηση θα ισχύσει για το πρώτο λεωφορείο και θα χτυπήσει σε 1 λεπτό μετά τη δημιουργία της, αφού 4-3=1.
- Ενώ αν ο χρήστης αναβάλει την ειδοποίηση σε χρόνο μεγαλύτερο ή ίσο του ενός λεπτού μετά την πρώτη ενεργοποίηση, η ειδοποίηση θα ισχύσει για το δεύτερο λεωφορείο και θα χτυπήσει σε 3 λεπτά, εφόσον οι νέοι χρόνοι άφιξης θα είναι 3 και 6 λεπτά αντίστοιχα αφού 6-3=3 ενώ 3-3=0.

Για να ακυρώσει κάποιος την ειδοποίηση εντελώς, αρκεί να πατήσει το κουμπί «Ακύρωση». Αν υπάρχει ενεργοποιημένη ειδοποίηση εκείνη τη στιγμή θα διαγραφεί, αν παίζει ο ήχος της ειδοποίησης θα σταματήσει και ο χρήστης θα επιστρέψει στην προηγούμενη οθόνη. Αν δεν έχει ενεργοποιηθεί η ειδοποίηση πατώντας το «Ακύρωση» απλά θα επιστρέψει στην προηγούμενη οθόνη.

Αναφερθήκαμε παραπάνω στη δυνατότητα αποθήκευσης των ειδοποιήσεων. Αυτό συμβαίνει αν ο χρήστης πατήσει το κουμπί «Αποθήκευση» στην οθόνη επιλογής χρόνου ειδοποίησης (Εικόνα 3.6) . Εμφανίζεται μία νέα οθόνη λοιπόν (Εικόνα 3.15), μ' ένα κείμενο που αναφέρει τις επιλογές του χρήστη. Για να αποθηκεύσει την ειδοποίηση αρκεί να δώσει ένα όνομα σ' αυτήν, π.χ. Σπίτι – Σχολή, έτσι ώστε να ξέρει για ποια ειδοποίηση ενεργεί και να πατήσει το κουμπί OK. Ενώ αν πατήσει το «Ακύρωση» επιστρέφει στην προηγούμενη εικόνα. Τέλος εμφανίζεται ένα μήνυμα στο κάτω μέρος της οθόνης ότι η επαφή αποθηκεύτηκε και γίνεται η εγγραφή στην τοπική βάση των ειδοποιήσεων(Εικόνα 3.16).

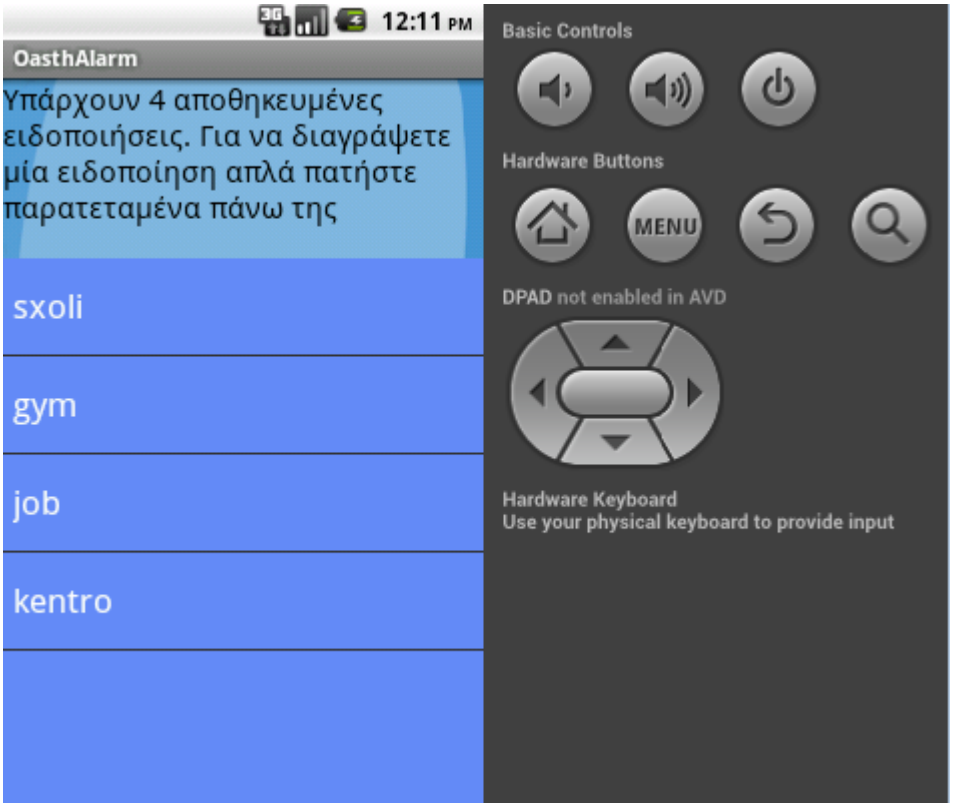


Εικόνα 3.15 Αποθήκευση Ειδοποίησης



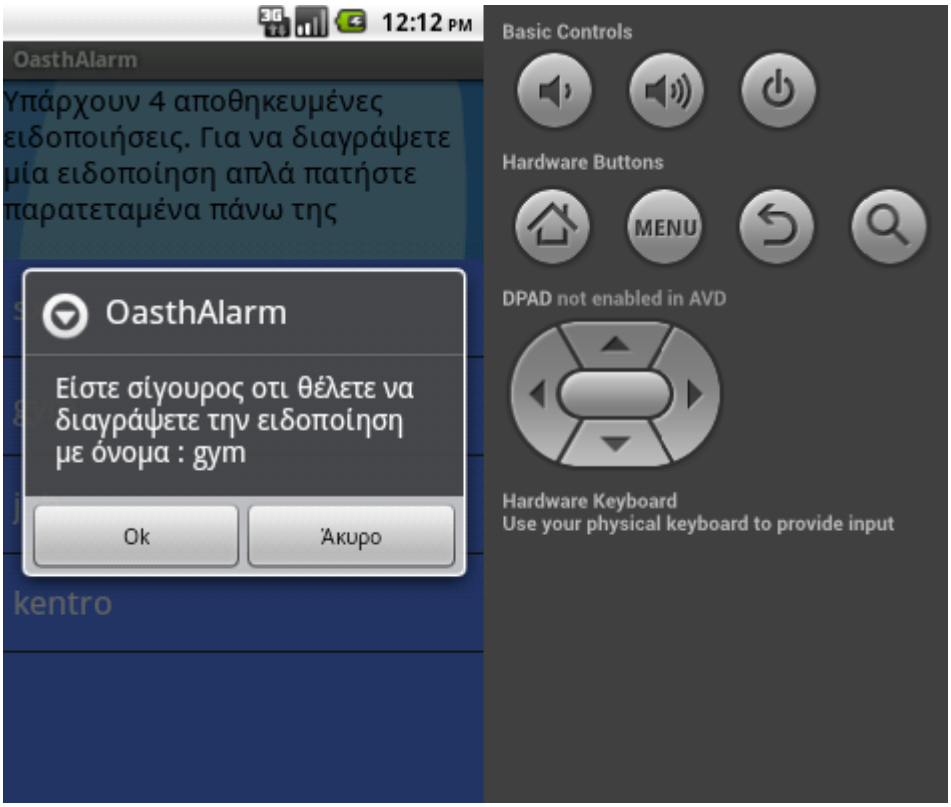
Εικόνα 3.15 Μήνυμα Αποθήκευσης Ειδοποίησης

Για να δει κάποιος τις αποθηκευμένες ειδοποιήσεις του αρκεί να πατήσει το ομώνυμο κουμπί στην πρώτη οθόνη της εφαρμογής(Εικόνα 3.1). Μεταφέρεται έτσι σε μία νέα οθόνη, εκεί υπάρχει κείμενο που ενημερώνει το χρήστη πόσες είναι οι αποθηκευμένες ειδοποιήσεις και πώς μπορεί να διαγράψει κάποια απ' αυτές (Εικόνα 3.17).

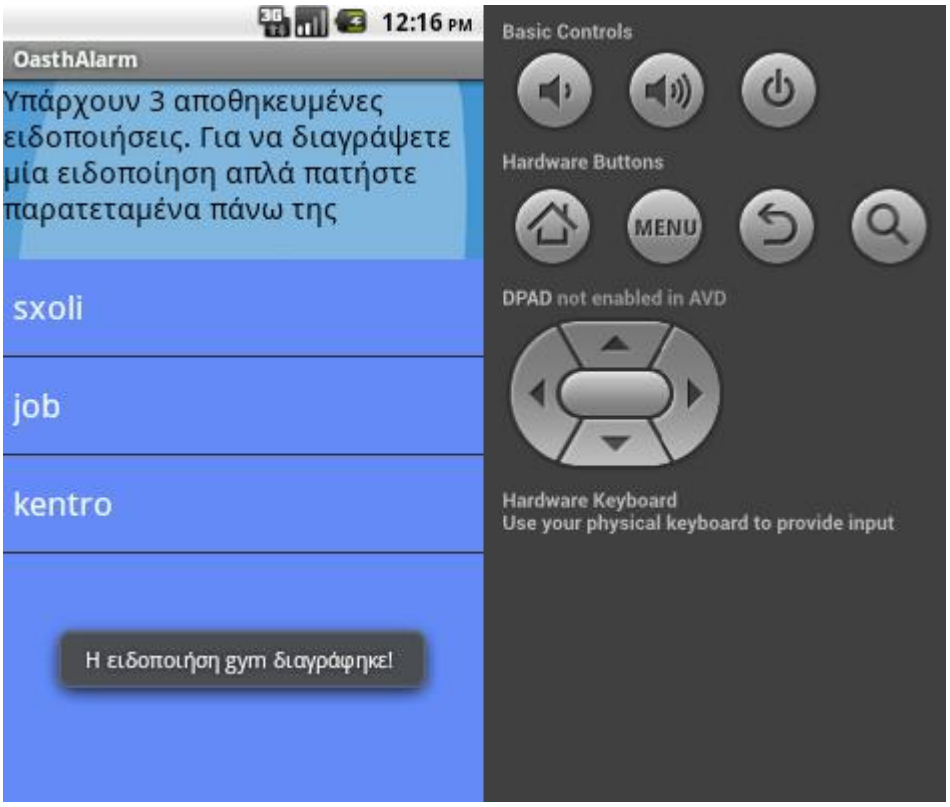


Εικόνα 3.17 Αποθηκευμένες Ειδοποιήσεις

Αν πατήσει παρατεταμένα λοιπόν σε μια ειδοποίηση εμφανίζεται ένας διάλογος με την εφαρμογή. Η εφαρμογή ρωτάει τον χρήστη αν είναι σίγουρος για τη διαγραφή της ειδοποίησης με όνομα π.χ. gym. Με το «OK» την διαγράφει και με το «Ακύρωση» εξαφανίζεται ο διάλογος. Μετά τη διαγραφή εμφανίζεται ένα μήνυμα toast που ενημερώνει τον χρήστη ότι η ειδοποίηση διαγράφηκε (Εικόνες 3.18 και 3.19).



Εικόνα 3.18 Διαγραφή Ειδοποίησης



Εικόνα 3.19 Μήνυμα Διαγραφής Ειδοποίησης

Αν ο χρήστης πατήσει απλά πάνω σε μία ειδοποίηση, μεταβαίνει στην οθόνη «Επιβεβαίωση της Ειδοποίησης» ούτως ώστε να την ενεργοποιήσει (βλέπε Εικόνα

3.11). Με τη χρήση των αποθηκευμένων ειδοποιήσεων ο χρήστης γλυτώνει σε χρόνο αφού μπορεί να προσπεράσει αρκετές οθόνες της εφαρμογής.

Ολοκληρώνοντας πιστεύω ότι η OasthAlarm είναι μία έξυπνη εφαρμογή για έξυπνα κινητά και έξυπνους ανθρώπους. Όσοι από εμάς χρησιμοποιούν τον ΟΑΣΘ έχουν τεθεί αντιμέτωποι με το πρόβλημα να συνωστίζονται και να περιμένουν αρκετή ώρα το λεωφορείο τους σε μια στάση. Αν λάβουμε υπόψη μας και τις καιρικές συνθήκες που επικρατούν ορισμένες μέρες μπορούμε να διαπιστώσουμε πιο εύκολα τη χρησιμότητα αυτής της εφαρμογής.

4. ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ ΕΦΑΡΜΟΓΗΣ (JAVA)

Στο κεφάλαιο αυτό θα γίνει μία σχετική ανάλυση του κώδικα που αναπτύχθηκε. Θα επεξηγηθούν τα σημαντικά κομμάτια κώδικα καθώς και οι μέθοδοι που καλούνται για την επίτευξη των διαφόρων λειτουργιών της εφαρμογής. Η ανάλυση θα γίνει σε δύο μέρη. Στο πρώτο μέρος θα αναλυθούν τα αρχεία που περιέχουν τον κώδικά σε Java τα οποία είναι αυτά που ουσιαστικά εκτελούν όλες τις λειτουργίες ενώ στο δεύτερο κομμάτι θα παραθέσουμε και θα αναλύσουμε τα αρχεία XML που χρησιμοποιήθηκαν για την δημιουργία του γραφικού περιβάλλοντος της εφαρμογής.

4.1 MainActivity.java

Το πρώτο αρχείο που θα αναλύσουμε είναι το αρχείο που αναλαμβάνει την λειτουργία της πρώτης οθόνης που βλέπουμε ανοίγοντας την εφαρμογή. Το αρχείο αυτό μπορεί να αποτελέσει μια βάση θα λέγαμε για τα περισσότερα αρχεία που θα δούμε παρακάτω , καθώς επεκτείνει μια δραστηριότητα (Activity).

Στην πρώτη γραμμή του δηλώνεται ότι το αρχείο αυτό ανήκει στο πακέτο της εφαρμογής με όνομα com.tasos.oasthalarm. Η γραμμή αυτή υπάρχει σε κάθε αρχείο java της εφαρμογής. Παρακάτω βλέπουμε τα διάφορα imports διάφορων βιβλιοθηκών του πακέτου του Android που πρέπει να κάνουμε έτσι ώστε να μπορέσουμε να χρησιμοποιήσουμε ενσωματωμένες μεθόδους και ιδιότητες των αντικειμένων των κλάσεων αυτών. Στο συγκεκριμένο παράδειγμα και σε οποιοδήποτε Activity γίνεται import των κλάσεων Bundle και Activity. Γίνεται επίσης import και για τις Intent,View,OnClickListener και Button. Όλες αυτές τις κλάσεις θα τις χρησιμοποιήσουμε όπως θα δούμε παρακάτω στο κώδικα.

Σε κάθε κλάση που θα χρησιμοποιηθεί σαν Δραστηριότητα πρέπει να επεκτείνουμε(extends) την Activity, αυτό μας δίνει τη δυνατότητα να χρησιμοποιούμε μεθόδους της κλάσης αυτής, μία εκ των οποίων είναι η onCreate που λαμβάνει ως όρισμα ένα Bundle και καλείται όταν το Activity μας πάει να εκτελεστεί. Επίσης η MainActivity υλοποιεί(implements) την OnClickListener, δίνοντας μας την δυνατότητα υπέρβασης των μεθόδων της OnClickListener. Ενώ η επέκταση και η υλοποίηση μιας κλάσης φαίνεται να μας προσφέρουν τις ίδιες δυνατότητες κάτι τέτοιο στη πραγματικότητα δε συμβαίνει. Μια κλάση μπορεί να επεκτείνει μονό μια άλλη κλάση ενώ μπορεί να υλοποιήσει άπειρες. Αλλά η βασικότερη διάφορα τους είναι πως από μια επεκτάσιμη κλάση μπορούμε να χρησιμοποιήσουμε και να υπερβούμε (override) οποιοδήποτε αριθμό των μεθόδων της, ενώ υλοποιώντας μια κλάση πρέπει να υπερβούμε όλες τις μεθόδους της.

Παρακάτω ορίζουμε δυο κουμπιά τα bLines και bNotView ,έχοντας κάνει import την κατάλληλη κλάση (Button) και υπερβαίνουμε τις μεθόδους onCreate της Activity και την onClick της OnClickListener.

Μέσα στην μέθοδο onCreate βλέπουμε δύο γραμμές που θα τις συναντάμε σε κάθε δρατηριότητα-κλάση μας, είναι η super.onCreate(savedInstanceState); η οποία καλεί τον super δομητή της κλάσης Activity με όρισμα το Bundle που δώσαμε ως παράμετρο στην onCreate της δικής μας κλάσης.Η επόμενη γραμμή χρησιμοποιεί μια πολύ χρήσιμη μέθοδο, την setContentView που παίρνει ως όρισμα ένα αρχείο layout από τα resources της εφαρμογής,στη συγκεκριμένη περίπτωση το R.layout.activity_main που αντιστοιχεί στο αρχείο activity_main.xml από τον φάκελο layout. Έτσι λοιπόν με τη χρήση του απλού αυτού κομματιού κώδικα, δημιουργείται ένα νέο Activity, με γραφικό της περιβάλλον τα περιεχόμενα ενός αρχείου xml της επιλογής του προγραμματιστή. Κάθε φορά που θέλουμε ένα View ή ένα Widget να το προγραμματίσουμε μέσω του κώδικα της java πρέπει να αντιστοιχήσουμε το id του από τα resources της εφαρμογής με την μέθοδο findViewById(). Στο layout activity_main.xml για παράδειγμα, υπάρχουν δύο κουμπιά με id bCreate και bNotView ενώ στο MainActivity έχουμε ορίσει αλλά δύο τα bLines και bNotView. Η συνδεσή τους γίνεται με τις γραμμές

```
bLines = (Button) findViewById(R.id.bCreate);
```

```
bNotView = (Button) findViewById(R.id.bNotView);
```

Τέλος με την μέθοδο setOnClickListener ενεργοποιούμε το γεγονός OnClick για τα buttons.

Η OnClick δέχεται ως όρισμα το View στο οποίο κάποιος θα κάνει κλικ , έτσι με μία switch ελέγχουμε το id του. Ανάλογα λοιπόν, το κουμπί που πατήθηκε θα εκτελεστεί κι ο αναλόγως κώδικας. Για το bLines θα δημιουργήσει ένα καινούριο Intent ,μια πρόθεση για το που θα πλοηγηθούμε αργότερα μεταξύ των activities, και θα ξεκινήσει με τη startActivity την Activity με το συγκεκριμένο intent. Στη περίπτωση μας αυτή θα είναι η ShowLines που έχει οριστεί στο AndroidManifest.xml ως activity και παρατηρούμε ότι πρέπει να δώσουμε σαν όρισμα στο intent όλη τη διαδρομή με το όνομα com.tasos.oasthalarm.SHOWLINES. Κάνουμε ακριβώς το ίδιο και για το άλλο μας κουμπί (bNotView) με τη διάφορα ότι αυτή τη φορά θα ξεκινήσουμε την CreatedNot activity.

4.2 ShowLines.java

Η ShowLines λοιπόν είναι μια δραστηριότητα-κλάση όπως η MainActivity με μια σημαντική διαφορά όμως. Η ShowLines δεν επεκτείνει την Activity , επεκτείνει την ListActivity με το ανάλογο import. Η ListActivity μας δίνει τη δυνατότητα να δημιουργήσουμε μια λίστα (ListView) από ένα πίνακα και να χρησιμοποιήσουμε μεθόδους για τα στοιχεία αυτής της λίστας καθώς επίσης και οποιαδήποτε μέθοδο της Activity.

Οι μεταβλητές που δηλώνονται είναι τρεις, ένας πίνακας linesTable[ ] και δύο global μεταβλητές τύπου String οι lineName και lineId που θα τις χρειαστούμε και μετέπειτα στην εφαρμογή μας. Στην onCreate δημιουργείται ένα αντικείμενο τύπου Line και ανοίγει η βάση μας. Γεμίζουμε τον linesTable με τον αριθμό και το όνομα κάθε γραμμής που διαθέτουμε στη βάση μας με τη βοήθεια της getColumnValue της κλάσης Utility. Στη συνέχεια ο linesTable περνάει σαν όρισμα στην μέθοδο setListAdapter, η οποία είναι υπεύθυνη για το «γέμισμα» της λίστας μας. Τέλος κλείνουμε τη βάση μας.

Η onListItemClick είναι μέθοδος της ListActivity και εδώ την υπερβαίνουμε. Είναι η μέθοδος εκείνη που διαχειρίζεται το γεγονός όταν κάνουμε κλικ σε ένα στοιχείο της λίστας. Δημιουργούμε την String μεταβλητή listline στην οποία δίνουμε το περιεχόμενο του στοιχείου της λίστας που πατήθηκε με την getItemAtPosition. Με τη βοήθεια των regular expressions της java και των αντικειμένων Pattern και Matcher παίρνουμε μόνο το όνομα της γραμμής και το εκχωρούμε στην global μεταβλητή lineName. Εκχωρούμε και το ανάλογο id στην lineId, δημιουργώντας ξανά ένα αντικείμενο Line και χρησιμοποιώντας την Utility.getColumnValue. Στο τέλος ξεκινάμε μια καινούρια Activity την ShowRoute.

Οι μέθοδοι του αντικειμένου Line και γενικά ο αντίστοιχος κώδικας που δεν αναλύονται εδώ θα αναλυθούν πιο διεξοδικά στα αρχεία που διαχειρίζονται τη βάση δεδομένων, όπως η DataBaseHelper. Όπως επίσης και τα regular expressions αναλύονται εκτενέστερα στην ArivalTimes που χρησιμοποιούνται και κατά κόρον.

4.3 ShowRoute.java

Η ShowRoute είναι και αυτή μία κλάση που επεκτείνει την Activity και υλοποιεί την OnClickListener όπως ακριβώς κι η MainActivity. Στη ShowRoute ορίζουμε δύο κουμπιά (Button) , ένα προβαλλόμενο κείμενο (TextView) και δύο global μεταβλητές τύπου String.

Στην onCreate τα buttons μας αρχικοποιούνται χρησιμοποιώντας την findViewById και ενεργοποιούμε το γεγονός onClick με τη setOnClickListener για το καθένα τους. Ενώ στο TextView εκχωρούμε τη global μεταβλητή lineName της ShowLines με σκοπό ο χρήστης να γνωρίζει ποια γραμμή έχει επιλέξει.

Στην OnClick αρχικοποιούμε τις global μεταβλητές broute και routeName ανάλογα με την επιλογή του χρήστη και ξεκινάει η επόμενη Activity , η ShowStops.

4.4 ShowStops.java

Η ShowStops είναι πανομοιότυπη με την ShowLines όσον αφορά τις λειτουργίες της. Είναι κι αυτή με τη σειρά της μία ListActivity και χρησιμοποιεί τις ίδιες μεθόδους , την onCreate και τη onListItemClick. Με τη διαφορά όμως ότι η ShowStops χρησιμοποιείται για την ανάκτηση πληροφοριών των στάσεων της συγκεκριμένης γραμμής και προορισμού που επιλέχτηκε από τον χρήστη κι όχι των γραμμών.

Οι μεταβλητές που δηλώνονται είναι τέσσερεις , ένας πίνακας stopsTable[ ] και τρεις global μεταβλητές τύπου String οι stopName, stopId και stopOrder που θα τις χρειαστούμε και μετέπειτα στην εφαρμογή μας. Στην onCreate δημιουργείται ένα αντικείμενο τύπου Stop και ανοίγει η βάση μας. Γεμίζουμε τον stopsTable με τον αριθμό και το όνομα κάθε γραμμής που διαθέτουμε στη βάση μας με τη βοήθεια της getColumnValue της κλάσης Utility. Στη συνέχεια ο stopsTable περνάει σαν όρισμα στην μέθοδο setListAdapter, η οποία είναι υπεύθυνη για το «γέμισμα» της λίστας μας.

Στην onListItemClick εκχωρούμε στις global μεταβλητές μας τις τιμές ανάλογα με το στοιχείο που πάτησε ο χρήστης στην λίστα, χρησιμοποιώντας τις κατάλληλες μεθόδους και δημιουργώντας τα κατάλληλα αντικείμενα , όπως ακριβώς στην ShowLines. Και τέλος ανοίγει η SelectTime που είναι η επόμενη Activity.

4.5 SelectTime.java

Στη συνέχεια θα ζητηθεί από το χρήστη να επιλέξει το κατάλληλο χρόνο για αυτόν, ανάλογα με τις ιδιαίτερες συνθήκες του καθενός, έτσι ώστε η εφαρμογή να τον ειδοποιήσει νωρίτερα από την άφιξη του λεωφορείου στη στάση. Για να το κατανοήσει κάποιος καλύτερα αυτό συνιστάται να έχει διαβάσει το προηγούμενο κεφάλαιο που αφορά την περιγραφή της εφαρμογής.

Η SelectTime λοιπόν, είναι κι αυτή μία Activity class με ότι κι αν αυτό συνεπάγεται. Σε αυτήν υπάρχουν έξι widgets όπου δηλώνονται ακριβώς με τον ίδιο τρόπο που κάναμε και στις προηγούμενες κλάσεις . Αυτά είναι ένα EditText , ένα TextView και τέσσερα Buttons . Εκτός από αυτά υπάρχουν μία global μεταβλητή και μία τοπική μεταβλητή τύπου Integer και οι δύο. Η global μεταβλητή είναι η timeNot που κρατάει ακριβώς αυτή τη τιμή του χρόνο που αναφέραμε παραπάνω, ενώ η τοπική μεταβλητή είναι ένας counter.

Στη μέθοδο onCreate μετά την κλήση της super καλούμε την setContentView που παίρνει ως όρισμα ένα αρχείο layout από τα resources της εφαρμογής, όπως ακριβώς κάναμε και για την MainActivity. Αυτή τη φορά όμως θα χρησιμοποιήσουμε ένα διαφορετικό xml αρχείο με σκοπό να δημιουργήσει η Activity διαφορετικό γραφικό περιβάλλον. Θα περάσουμε ως όρισμα στη setContentView το R.layout.selecttime που αντιστοιχεί στο αρχείο selecttime.xml . Μετέπειτα αρχικοποιούμε όλα τα widgets με την findViewById ενεργοποιούμε το γεγονός onClick στα Buttons καλώντας την setOnClickListener, με τον ίδιο ακριβώς τόπο που έχουμε δει και σε προηγούμενες κλάσεις. Επιπροσθέτως εισάγουμε κείμενο στο TextView tvST με την μέθοδο setText που δέχεται ως όρισμα οποιαδήποτε σειρά χαρακτήρων και στη προκειμένη περίπτωση μία τοπική String μεταβλητή την text. Εκχωρούμε στη text βοηθητικό κείμενο για το χρήστη προκειμένου να γνωρίζει ακριβώς τι έχει επιλέξει στις προηγούμενες Activities. Αυτό το καταφέρνουμε πολύ εύκολα με τις global μεταβλητές που έχουμε ορίσει ως τώρα σε άλλες Activities . Εδώ χρησιμοποιούμε την ShowStops.stopName, την ShowLines.lineName και την ShowRoute.routeName.
Τέλος αρχικοποιούμε το counter δίνοντας του το περιεχόμενο του mins με τη κλήση της getText που by default είναι η μικρότερη δυνατή τιμή που έχουμε ορίσει να υπάρχει στο TextView μας , το ένα. Με την ίδια τιμή έχει αρχικοποιηθεί και η timeNot από την δήλωσή της ακόμα.

Άξιο αναφοράς εδώ θα ήταν να εξηγήσουμε τον τρόπο που έχει στη διάθεση του ένας προγραμματιστής Android για χρωματίσει με διαφορετικό χρώμα συγκεκριμένα κομμάτια ενός κειμένου. Παρατηρούμε ότι μέσα στο text υπάρχουν εντολές που συναντώνται σε Html κώδικα όπως στη συγκεκριμένη περίπτωση `<font color='red'>"+ShowStops.stopName+"</font>"` . Αυτό συμβαίνει γιατί το Android μέσα από τις βιβλιοθήκες της java και συγκεκριμένα την android.text.Html που έχουμε κάνει import μας δίνει τη δυνατότητα να μορφοποιήσουμε το κείμενο μας πολύ εύκολα με Html εντολές. Τελικά αυτό το καταφέρνουμε παίρνοντας ως ορίσματα στην setText την κλήση της Html.fromHtml και το TextView.BufferType.SPANNABLE .

4.6 AlarmNot.java

Η AlarmNot είναι η τελική δραστηριότητα που οδηγείται ο χρήστης. Και αυτή με τη σειρά της δηλώνει τις βιβλιοθήκες που χρησιμοποιεί , επεκτείνει την Activity και υλοποιεί την OnClickListener με το τρόπο που έχουμε δει προηγουμένως. Στη java η μονάδα μέτρησης του χρόνου που χρησιμοποιείται είναι τα milliseconds, έτσι για να δουλεύουμε καλύτερα με δευτερόλεπτα ή λεπτά έχουμε ορίσει δύο final μεταβλητές τις ONE_SECOND και ONE_MINUTE. Ορίσαμε τα κουμπιά μας confirm και cancel ούτος ώστε να είναι λειτουργικά. Εδώ συναντάμε και κάποια αντικείμενα που δεν τα έχουμε δει και είναι απαραίτητα για την δημιουργία ειδοποιήσεων, αυτά είναι ένα PendingIntent, ένα BroadcastReceiver, ένα AlarmManager και ένα NotificationManager όπως επίσης και μία σταθερή και μοναδιαία μεταβλητή για το id της ειδοποίησης. Για όλα τα παραπάνω θα δούμε αργότερα πως χρησιμοποιούνται και πως αλληλεπιδρούν δημιουργώντας την ειδοποίηση αυτής της εφαρμογής. Στη συνέχεια ορίζουμε κάποιες βοηθητικές μεταβλητές για το χρόνο άφιξης των λεωφορείων (exactMin,notMin) και κάποιες Boolean που χρειάζονται σε παρακάτω ελέγχους (NotExists, isRinging). Τέλος δηλώνουμε ένα αντικείμενο ar τύπου ArrivalTimes όπου κρατάει τους χρόνους άφιξης των λεωφορείων και θα το δούμε αναλυτικότερα παρακάτω.

Στην OnCreate της AlarmNot αντιστοιχούμε το κατάλληλο layout οπού εδώ είναι το alarm.xml. Στα buttons και τα textViews αντιστοιχούμε τα id ενώ για τα buttons ενεργοποιούμε το onClick με το ίδιο τρόπο που κάναμε σε ανάλογες περιπτώσεις. Αρχικοποιούμε το isRinging ως false και δημιουργούμε ένα αντικείμενο AlertDialog.Builder το dlgAlert που είναι ένα κουτί διαλόγου μεταξύ χρήστη και εφαρμογής. Πριν τους αρκετούς ελέγχους που θα ακολουθήσουν αρχικοποιούμε το nm που δηλώσαμε νωρίτερα. Το nm είναι τύπου NotificationManager είναι δηλαδή μια service του android που μας ειδοποιεί για κάτι που συμβαίνει στο background. Ορίζουμε λοιπόν το nm σαν μια υπηρεσία του συστήματος.
Στη συνέχεια και ενώ βρισκόμαστε ακόμα στη κατάσταση OnCreate της εφαρμογής υπάρχουν τέσσερις έλεγχοι που σκοπός τους είναι να βοηθήσουν το χρήστη και να τον αποτρέψουν ακόμα από κάποια λάθη. Στην αρχή ελέγχουμε αν η συσκευή είναι συνδεδεμένη στο διαδίκτυο με τη βοήθεια της isOnLine() (θα την αναλύσουμε παρακάτω).Αν είναι συνδεδεμένη θα δημιουργήσει ένα νέο αντικείμενο ArivalTimes και θα εμφανίσει στο πρώτο textView τις επιλογές του χρήστη textView μας και στο δεύτερο τους χρόνους άφιξης των λεωφορείων για τα δεδομένα που έχουμε επιλέξει και θα συνεχίσει με τους όλους ελέγχους.

Ο επόμενος έλεγχος έχει να κάνει με το αν ο πίνακας ar.arTimes που περιέχει τις τιμές των χρονών σε λεπτά των αφίξεων των λεωφορείων είναι κενός. Αυτό μονό ένα πράγμα μπορεί να σημαίνει ότι δεν υπάρχουν λεωφορεία ως εκ τούτου ενημερώνουμε το χρήστη με ανάλογο μήνυμα στο dialog box με τη μέθοδο dlgAlert.setMessage(“ κείμενο “) και χρησιμοποιούμε και την setTitle της dlgAlert για να δώσουμε και όνομα στο dialogBox. Με την setPositiveButton μπορούμε να δημιουργήσουμε ένα button μέσα στο κουτί και να ενεργοποιήσουμε το γεγονός OnClick και να το προγραμματίσουμε μέσα στη μέθοδο. Αν ο χρηστής λοιπόν πατήσει το κουμπί οκ καλείται η finish() η οποία τερματίζει την συγκεκριμένη activity και μας επιστρέφει στη προηγούμενη (SelectTime). Θέτοντας την setCancelable true παρέχουμε τη δυνατότητα στην εφαρμογή να μπορεί να ακυρωθεί και τέλος με την create.show εμφανίζουμε το dialog μας.

Αν ο πίνακας δεν είναι κενός ελέγχουμε το SelectTime.*timeNot* ,ο χρόνος δηλαδή που διάλεξε ο χρήστης να ειδοποιηθεί πριν το λεωφορείο φτάσει στη στάση, αν είναι μικρότερος από τον μοναδικό χρόνο άφιξης της γραμμής σε περίπτωση δηλαδή που βρίσκεται μόνο ένα λεωφορείο καθοδόν. Μετέπειτα κάνουμε ακριβώς τις ίδιες ενέργειες για το dialog box που θέλουμε να εμφανιστεί όπως κάναμε και παραπάνω αλλάζοντας μόνο το κείμενο του.

Το τελευταίο ζήτημα που πρέπει να ελέγξουμε είναι αν ο χρόνος που επέλεξε ο χρήστης ξεπερνάει τον χρόνο άφιξης του λεωφορείου που θα έρθει αργότερα και βρίσκεται στη τελευταία θέση του πίνακα ar.arTimes[ar.arTimes.length – 1].

Σε περίπτωση όμως που ο αρχικός έλεγχος για τη σύνδεση στο Ιντερνέτ είναι αρνητικός τότε θα εμφανιστεί ένα καινούριο dialogBox με τη διαφορά ότι θα έχει ορισμένο ένα επιπλέον κουμπί το cancel. Αυτό το επιτυγχάνουμε με την setNegativeButton και προγραμματίζουμε το κουμπί όταν πατηθεί να επιστρέφει στην προηγούμενη activity με το finish, ενώ αν ο χρήστης επιλέξει ok τότε τον οδηγούμε στα settings/Wireless and Networks του android για να ενεργοποιήσει τη σύνδεση του. Για να το καταφέρουμε αυτό πρέπει πρώτα να διακόψουμε την activity στην οποία βρισκόμαστε me to finish και στη συνέχει να ξεκινήσουμε μια καινούρια activity του android startActivityForResult(**new** Intent(android.provider.Settings.*ACTION_WIRELESS_SETTINGS*),0);

Ανάλογα με τη κατάσταση της ειδοποίησης μας , αν είναι ενεργή ή όχι , το κουμπί confirm κάνει τις αντίστοιχες ενέργειες, σε περίπτωση που η ειδοποίηση είναι ενεργή λειτούργει σαν αναβολή. Αν το NotExists είναι true τότε διαγράφουμε το registerReiceiver και ακυρώνουμε την notification και σε περίπτωση που εκείνη τη στιγμή χτυπάει ο ήχος της ειδοποίησης (isRinging) τον σταματάμε με την μέθοδο release, έχοντας αλλάξει το NotExists σε κατάσταση false πιο πριν. Αλλιώς, αν δεν έχει δημιουργηθεί η ειδοποίηση, δημιουργούμε μία καινούρια με την createAlarm() (βλέπε πιο κάτω) και αλλάζουμε το κείμενο του κουμπιού σε Αναβολή με το την setText.

Αν ο χρήστης πατήσει το cancel τότε πρέπει να γίνουν ακριβώς οι ίδιοι έλεγχοι με παραπάνω. Σε περίπτωση ενεργής ειδοποίησης με τι ίδιες ακριβώς μεθόδους την απενεργοποιούμε την nm και επιπλέον απενεργοποιούμε και την pi με την cancel() της AlarmManager. Επιστρέφουμε στην προηγούμενη η activity με το finish και τέλος αλλάζουμε το κείμενο του confirm σε «Επιβεβαίωση» αν δεν είναι ενεργή η ειδοποίηση.

Η createAlarm καλεί την createNotification που θα δούμε παρακάτω και δημιουργεί μια ειδοποίηση. Με σκοπό να εξελίξουμε την κλασσική ειδοποίηση που μας δίνει by default το android ούτος ώστε να μας ειδοποιεί με συγκεκριμένο ήχο τη στιγμή ακριβώς που θέλουμε, πρέπει να δημιουργήσουμε ένα alarm. Πριν όμως από αυτό δημιουργούμε ένα νέο BroadcastReceiver. Τώρα πρέπει να υπερβούμε τη μέθοδο onReceive και εκεί να προγραμματίσουμε τις ενέργειες όταν θα την καλέσουμε. Ορίζουμε τον ήχο με την create μέθοδο της MediaPlayer βάζοντας στις παράμερους της την activity που συμβαίνει και το μονοπάτι που βρίσκεται ο ήχος μας στην εφαρμογή. Ξεκινάει να παίζει ο ήχος με την start και θέτουμε την Boolean μεταβλητή isRinging true. Στη συνέχεια εγγράφουμε το broadcast Receiver μας, το br, έτσι ώστε να τρέξει στο κύριο νήμα της εφαρμογής μας όταν

αυτή το παραλάβει. Ορίζουμε το PendingIntent να λάβει το broadcast από το παραπάνω και ορίζουμε το am να είναι μια υπηρεσία συστήματος ALARM_SERVICE, κάτι αντίστοιχο κάναμε και πιο πριν με την ειδοποίηση. Καλώντας την getExactMin παίρνουμε το χρόνο άφιξης που είναι πλησιέστερος στις απαιτήσεις μας, μετά αφαιρούμε το χρόνο που χρειάζεται ο χρήστης να φτάσει στη στάση και εκχωρούμε το αποτέλεσμα στην notMin που είναι ο χρόνος που θα πρέπει να ξεκινήσει ο ήχος από τη στιγμή που πατήσαμε το confirm. Τέλος πρέπει να σετάρουμε το alarm μας με την set. Σε αυτήν πρέπει να ορίσουμε τι τύπου θα είναι το alarm, πότε θα ενεργοποιήθει και ποια pending intent θα ενεργοποιήσει.
Στην getExactMin με τη βοήθεια ενός βρόχου for βρίσκουμε τον πλησιέστερο από τους χρόνους που ικανοποιούν τις απαιτήσεις του χρήστη. Π.χ. Αν ο χρήστης θέλει να ειδοποιηθεί 5 λεπτά πριν το λεωφορείο φτάσει στη στάση και οι χρόνοι άφιξης των λεωφορείων είναι 2, 4, 5, 7, 9 λεπτά, η getExactMin θα επιστρέψει το 7 και χρήστης θα ειδοποιηθεί σε 2 λεπτά.

Η createNotification όπως προδίδει το όνομά της δημιουργεί notifications. Για να δημιουργήσουμε μία ειδοποίηση στο android πρέπει πρώτα να δημιουργήσουμε την πρόθεση της και να ορίσουμε σε αυτήν τα ανάλογα flags της με την setFlag που έχουν να κάνουν με τη θέση του εικονιδίου της. Στην συνεχεία περνάμε σαν παράμετρο αυτή την πρόθεση στην pending Intent μας. Δημιουργούμε τελικά την notification βάζοντας σε παραμέτρους το εικονίδιο που θα χρησιμοποιεί, το σώμα-κείμενό της και το χρόνο που θα ενεργοποιηθεί. Μέτα με την setLatestEventInfo τη συνδέουμε με την pi και της δίνουμε και τίτλο. Εκχωρούμε στην n.defaults την τιμή Notification.*DEFAULT_ALL* ούτος ώστε να έχουμε τις default παραμέτρους μιας ειδοποίησης android. Ολοκληρώνοντας θέτουμε την NotExists true και με την βοήθεια του διαχειριστή Ειδοποιήσεων και την μέθοδο του notify ενεργοποιούμε την n και τις αντιστοιχούμε το μοναδικό id που ορίσαμε παραπάνω.

Τελευταία μέθοδο που συναντάμε στην AlarmNot είναι η isOnLine που επιστρέφει true ή false ανάλογα με τον αν είναι συνδεδεμένη η συσκευή μας ή όχι. Χρησιμοποιούμε την υπηρεσία ConnectivityManager του android και από αυτό παίρνουμε πληροφορίες σχετικά με τη σύνδεση μας. Το netInfo αντικείμενο που είναι τύπου NetworkInfo έχει την μέθοδο isConnectedOrConnecting που επιστρέφει true αν η συσκευή μας συνδέεται ή είναι συνδεδεμένη και κάνει όλη τη δουλειά εδώ για εμάς.

4.7 GetMethodEx.java

Η GetMethodEx είναι μία βοηθητική κλάση και ο χρήστης δε μπορεί να την δει, δεν είναι μία Activity όπου οι προηγούμενες που είδαμε ως τώρα. Τη χρησιμοποιούμε για να κάνουμε parse μια σελίδα Html και να ανακτήσουμε όλο τον Html κώδικά της. Συγκεκριμένα η GetMethodEx έχει μία μέθοδο την getInternetData η οποία δέχεται ως όρισμα ένα String που είναι το Url της σελίδας και μας επιστρέφει όλο το κώδικα της σε ένα String.

Στην getInternetData δηλώνουμε ένα BufferedReader in και ένα String data στο οποίο θα εγγράψουμε τους χαρακτήρες. Μέσα σένα μπλοκ try δημιουργούμε ένα client και μία website ανάλογα με το όρισμα της getInternetData . Στη συνέχεια

κάνουμε ένα αίτημα (request) στη σελίδα και παίρνουμε μία απάντηση(response) με την client.execute. Από την απάντηση ανακτούμε όλο το περιεχόμενο της στο BufferedReader in με ένα inputStreamReader και τις μεθόδους getEntity και getContent της response. Στη συνέχεια περνάμε γραμμή προς γραμμή όλο το κείμενο που υπάρχει στο in σε ένα StringBuffer το sb. Κλείνουμε το in και εκχωρούμε το sb αφού πρώτα το μετατρέψουμε σε String στο data και επιστρέφουμε το αποτέλεσμα. Τέλος σε περίπτωση λάθους πρέπει να βεβαιωθούμε ότι έχει κλείσει το BufferedReader κι έχουμε επιστρέψει το αποτέλεσμα διαφορετικά να πετάει ένα exception.

4.8 ArivalTimes.java

Στην AlarmNot είδαμε τη χρήση του αντικειμένου ArivalTimes αρκετές φορές και εξηγήσαμε πως είναι οι χρόνοι άφιξης των αστικών μιας γραμμής σε μία στάση εδώ θα εξηγήσουμε πως διαχειριζόμαστε αυτά τα δεδομένα που ανακτούμε από την html σελίδα http://m.oasth.gr/ με τη βοήθεια της GetMethodEx.getInternetData.
Το Url της σελίδας που παρουσιάζει τους χρόνους άφιξης μίας γραμμής σε μία στάση είναι της μορφής : http://m.oasth.gr/#index.php?md=3&sn=3&start=885&sorder=16&line=63&dir=1

Το start είναι η στάση , το sorder είναι η θέση της στάσης στη σειρά της διαδρομής, το line είναι η γραμμή και το dir είναι η κατεύθυνση. Π.χ. το παραπάνω Url είναι για τη γραμμή 10 με lineId στη βάση ίσο με 63, για τη στάση Ιασονίδου με stopId ίσο με 885, με κατεύθυνση από Χαριλάου για Σταθμό και είναι η 16η στη σειρά.

Χρειαζόμαστε λοιπόν 4 μεταβλητές τύπου String στις ποίες εκχωρούμε τις επιλογές του χρήστη που κρατήσαμε από προηγούμενα activities Αυτό συμβαίνει στο δομητή του ArivalTimes, εκεί δημιουργούμε και ένα αντικείμενο GetMethodEx για να κάνουμε parsing την ιστοσελίδα. Ότι κώδικα ανακτήσουμε από την Html σελίδα τον εκχωρούμε στην data με την κλήση της getDataFromInternet.
Μετέπειτα με τη χρήση της VehicleExists ελέγχουμε αν όντως υπάρχουν δεδομένα, στην περίπτωσή μας χρόνοι άφιξης και εκχωρούμε τον πίνακα των χρόνων στην arTimes με τη βοήθεια της getSpecialData και ένα string αυτών στην sdata. Αν δεν υπάρχουν δεδομένα κάνουμε mull τον πίνακα arTimes και εκχωρούμε ανάλογο μήνυμα στο sdata.

Η getDataFromInternet δέχεται τις 4 μεταβλητές που προαναφέραμε σαν ορίσματα. Με την κλήση της getInternetData της GetMethodEx και προσθέτοντας στον ορισμό της τις προηγούμενες μεταβλητές, δημιουργούμε το κατάλληλο URL. Έτσι το αποτέλεσμά της το εκχωρούμε στην data και επιστρέφουμε το αποτέλεσμα.

Η VexhicleExists ελέγχει αν υπάρχουν λεωφορεία, συνεπώς χρόνοι, εν κινήσει. Αυτό το καταφέρνουμε με τη χρήσηRegularExpressions της dara. Δημιουργούμε ένα αντικείμενο Pattern που είναι το μοτίβο που ψάχνουμε σ' ένα κείμενο και ένα Matcher που είναι το αποτέλεσμα της RegEx. Το Pattern στην προκειμένη

περίπτωση είναι το «ΑΦΙΞΗ ΣΕ». Το Matcher είναι το αποτέλεσμα της μεθόδου match του pattern μέσα στο κείμενο της data. Αν βρεθεί αυτό matcher.find ( ) η μέθοδός μας επιστρέφει true αλλιώς false.

Η getSpecialTimes χρησιμοποιεί κατά κόρον τα αντικείμενα Pattern και Match και με τη χρήση της compile του Pattern συνεχώς το μειώνει μέχρι να φτάσουμε στο επιθυμητό αποτέλεσμα. Αυτό είναι σύμβολα, μια σειρά χαρακτήρων της μορφής π.χ. 12’ 3’ 5’, που συνεχίζει να μην είναι λειτουργικό ακόμα για τις απαιτήσεις μας.

Η getArivalTimes είναι υπεύθυνη γι’ αυτό, συνεπώς. Δέχεται ως όρισμα το προηγούμενο αποτέλεσμα της getSpecialData. Για να κρατήσουμε τους πραγματικούς χρόνους, ξεχωρίζουμε κάθε χαρακτήρα της SpecialData και τους εκχωρούμε σε μία λίστα. Χρησιμοποιούμε λίστα γιατί δεν ξέρουμε ακριβώς πόσοι είναι ακριβώς οι χρόνοι και γιατί δε μπορούμε να ορίσουμε πίνακα στη Java αν δεν ξέρουμε το μέγεθός του. Με ένα βρόχο for ελέγχουμε αν κάθε χαρακτήρας είναι ψηφίο ή ο χαρακτήρας «’» και τους εκχωρούμε στον String πίνακα times1. Στην συνέχεια μετατρέπουμε αυτόν τον πίνακα σε integer στο πίνακα times2. Τέλος και με τη βοήθεια της Arrays.sort τον ταξινομούμε.

Η getToStringArivalTimes απλά εκχωρεί όλο τον πίνακα ArTimes σ’ ένα string για μελλοντική χρήση σε μηνύματα προς το χρήστη

4.9 DataBaseHelper.java

Η DataBaseHelper είναι μία βοηθητική κλάση όπως υποδηλώνει το όνομά της η οποία κληρονομεί την SQLiteOpenHelper. Παρέχει διάφορες μεθόδους που μπορούν να προσπελάσουν και να διαχειριστούν την έκδοση, το όνομα, τις στήλες και τις εγγραφές μιας SQL βάσης.
Εδώ δηλώνουμε 3 static μεταβλητές, το TAG που είναι μόνο για το LogCat, τη DB_PATH που είναι το μονοπάτι που βρίσκεται η βάση και την DB_NAME που είναι το όνομα της βάσης. Δηλώνουμε επίσης και τη SQLite βάση όπως επίσης και το Context της.

Με την εντολή `private static String DB_NAME = "oasth.db";` ορίζουμε το όνομα της βάσης να είναι το αρχείο oasth.db που είναι αποθηκευμένο στο φάκελο assets της εφαρμογής και έτσι γίνεται η σύνδεση με τη βάση μας. Στην βάση αυτή δεν μπορούμε να προσθέσουμε, να διαγράψουμε ή να τροποποιήσουμε εγγραφές , μπορούμε μόνο να ανακτήσουμε πληροφορίες από αυτήν.

HCreateDataBase δημιουργεί την βάση μας σε περίπτωση που δεν έχει ξαναδημιουργηθεί. Το αν έχει ξαναδημιουργηθεί το ελέγχει η checkDataBase.Στην ουσία την αντιγράφει με την copyDataBase που καλείται στην createDataBase και

είναι υπεύθυνη για την αντιγραφή της βάσης. Οι openDatabase και close ( ) είναι υπεύθυνες για το άνοιγμα και το κλείσιμο της βάσης αντίστοιχα

4.10 PinakasAdapter.java

Η κλάση PinakasAdapter είναι ο Adapter μεταξύ των πινάκων της βάσης μας και της Java. Οποιαδήποτε ενέργεια θέλουμε να κάνουμε στη βάση μας πρέπει πρώτα να περάσει από εδώ. Χρησιμοποιεί πλήθος μεθόδων για τη διαχείριση της βάσης, η πιο σημαντική μέθοδος όμως είναι η getLineNumber.

Η getLineNumber επιστρέφει ένα αντικείμενο τύπου Cursor. Σ' αυτό εμπεριέχονται όλες οι εγγραφές από ένα SQL ερώτημα. Για να κάνουμε ερωτήματα στη βάση μας χρησιμοποιούμε το rawquery. Οποιαδήποτε χρήση τηςrawquery επιβάλλεται να την περικλύσουμε σε μπλοκ try.

4.11 Line.java

Η Linejava είναι μία κλάση που επεκτείνει την PinakasAdapter. Αυτή η κλάση πραγματεύεται οτιδήποτε έχει να κάνει με μια λεωφορειακή γραμμή. Όλες οι μέθοδοί της είναι αντίστοιχες με την getLineNumber του PinakaAdapter που είδαμε νωρίτερα και έχουν να κάνουν με SQL ερωτήματα που αφορούν στοιχεία των γραμμών. Οι μέθοδοι αυτές είναι: η getLinesData, getLineId,

4.12 Stop.java

Η αντίστοιχη κλάση για τις στάσεις είναι η Stop. Είναι άλλη μια κλάση που επεκτείνει τον PinakaAdapter. Οι μέθοδοί της χρησιμοποιούνται για Sql ερωτήματα που έχουν να κάνουν με στοιχεία στάσεων όπως οι getStopsData, getStopId, getSOrder.

4.13 Utility.java

Η Utility περιλαμβάνει βοηθητικές μεθόδους για να χειριστούμε καλύτερα τα αποτελέσματα που είναι τύπου Cursor από τις μεθόδους των προηγούμενων κλάσεων παραπάνω. Οι πιο σημαντικές είναι οι GetColumnValue που μετατρέπει το Cursor σε String και η GetColumnIntValue που το μετατρέπει σε integer.

4.14 SaveNot.java

Η SaveNot είναι μία Activity κλάση. Σε αυτήν οδηγούμαστε αν στη SelectTimeactivityπατήσουμε το κουμπί «Αποθήκευση». Όπως είδαμε αρκετές

φορές πριν δηλώνουμε τα widgets στην αρχή και στην Oncreate κάνουμε τις όποιες εκχωρήσεις ή ενεργοποιήσεις γεγονότων.

Στην OnClick αν ο χρήστης πατήσει Ok δημιουργεί μια νέα καταχώρηση στην βάση δεδομένων για τις αποθηκευμένες ειδοποιήσεις. Πρώτα δημιουργούμε ένα αντικείμενο NotificationsDatabaseentry. Στη συνέχεια το ανοίγουμε και με την createEntry δημιουργούμε την εγγραφή. Τέλος κλείνουμε την βάση με το close ( ).

4.15 NotificationsDatabase.java

Η NotoficationsDatabase είναι η κλάση που δημιουργούμε βάση για τις αποθηκευμένες ειδοποιήσεις. Το Όνομα και έκδοση της βάσης, καθώς και τα ονόματα των πινάκων της πρέπει να είναι private, static, και final. Επίσης κάθε όνομα στήλης της βάσης πρέπει να είναι κι αυτό static και final. Σε αντίθεση με την προηγούμενη βάση που συνατήσαμε σε αυτήν εχουμε το δικαίωμα να δημιουργήσουμε και να διαγράψουμε μία εγγραφή.

Στην createEntry δίνουμε ως ορίσματα τα στοιχεία για μια εγγραφή. Αυτό το επιτυγχάνουμε όμως μόνο με την χρήση της ContentValue. Η ContentValue χρησιμοποιείται για να αποθηκεύσει ένα σύνολο τιμών. Στη συνέχεια με τη χρήση της put μπορούμε να αντιστοιχίσουμε αυτές τις τιμές στις αντίστοιχες μεταβλητές. Στην deleteEntry με τη χρήση της delete της SQLiteDatabase μπορούμε να διαγράψουμε μια γραμμή του πίνακα.

4.16 CreatedNot.java

Η CreatedNot είναι η κλάση – activity στην οποία μπορούμε να δούμε τις αποθηκευμένες μας ειδοποιήσεις. Εδώ έχουμε τον πίνακα Stringcomments, μια listview, ένα LongdlRow (για διαγραφή γραμμών), ένα AlertDialog dlgAlert και το TextView tvsn.

Στην oncreate κάνουμε τις απαραίτητες ενέργειες για την αντιστοίχιση του layout και των διευθύνσεων των views. Δημιουργούμε μία εικόνα της βάσης μας τύπου Notifications – Database, την ανοίγουμε και παίρνουμε τα δεδομένα της σ’ ένα στιγμιότυπο Cursor. Στη συνέχεια ελέγχουμε αν υπάρχει έστω και μία εγγραφή σ’ αυτόν με την getCount.

Αν υπάρχουν εγγραφές εκχωρούμε το κατάλληλο κείμενο στο tvsn. Με έναν βρόχο for κάνουμε προσπέλαση του cursor και παίρνουμε όλα τα comments – Ονόματα ειδοποιήσεων. Μετέπειτα μέσω ενός ListAdapter φορτώνουμε όλα τα ονόματα στη ListView με τρόπο που είδαμε σε προηγούμενα activities. Στην περίπτωση που δεν υπάρχουν εγγραφές τοποθετούμε το ανάλογο κείμενο στο textView. Τέλος ενεργοποιούμε το γεγονός onLongClick με τη setLongClickable και με την setOnLongClickListener την προγραμματίζουμε.

Με το παρατεταμένο πάτημα πάνω σ' ένα στοιχείο της λίστας, δημιουργούμε ένα κουτί επικοινωνίας με ένα θετικό κουμπί OK και ένα αρνητικό Cancel. Αν ο χρήστης πατήσει το θετικό, δημιουργούμε ένα στιγμιότυπο της βάσης, την ανοίγουμε, διαγράφουμε την γραμμή και μετά την κλείνουμε με τις κατάλληλες μεθόδους. Εμφανίζουμε και ένα toast που ενημερώνει τον χρήστη για τη διαγραφή και η δραστηριότητα συνεχίζεται με το OnResume ( ). Αν ο χρήστης πατήσει το Cancel απλά κάνουμε dismiss ( ) τον διάλογο.

Ενεργοποιούμε και το απλό κλικ σε στοιχείο της λίστας με την setOnItemClickListener. Εδώ εκχωρούμε τα κατάλληλα πεδία της εγγραφής στις ανάλογες global μεταβλητές που έχουμε ορίσει από προηγούμενα activities και με τη startActivity ξεκινάμε την AlarmNot.

4.17 Αρχεία XML Layouts

Τα αρχεία αυτά και οι δυνατότητες που μας προσφέρει η XML είναι υπεύθυνα για όλο το Interface της εφαρμογής. Μπορούμε να χρησιμοποιήσουμε πλήθος από widgets και με τις ανάλογες ρυθμίσεις να τα δώσουμε τη μορφή που θέλουμε. Η OasthAlarm περιέχει διάφορα buttons για την πλοήγηση μέσα στην εφαρμογή, textViews ως προς την βοήθεια και ενημέρωσή του, τρία ListViews και δύο EditTexts.

4.18 Αρχείο AndroidManifest.xml

Το αρχείο αυτό υπάρχει σε κάθε εφαρμογή και περιέχει σημαντικές πληροφορίες σχετικά με αυτή. Αρχικά ορίζεται σε αυτό το όνομα πακέτου της εφαρμογής, η έκδοση και το όνομα της έκδοσης όπως επίσης και μερικά ακόμα χαρακτηριστικά. Για παράδειγμα το path που θα εγκαθίσταται η εφαρμογή. Στο αρχείο αυτό ορίζονται οι άδειες που θα πρέπει να πάρουμε από τον χρήστη πριν εγκαταστήσει την εφαρμογή, για την εκτέλεση ορισμένων βασικών λειτουργιών του τηλεφώνου που δύναται να γίνουν μέσω της εφαρμογής. Στο αρχείο της εφαρμογής μας το AndroidManifest.xml οι άδειες που έχουμε ορίσει είναι οι εξής :

- `<uses-permission android:name="android.permission.INTERNET"/>`
- `<uses-permission android:name="android.permission.VIBRATE"/>`
- `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>`

Η πρώτη αφόρα την άδεια σύνδεσης στο Internet, η δεύτερη την άδεια χρησιμοποίσης της δόνησης της συσκευής και η τελευταία την κατάσταση δικτύου.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στη συγκεκριμένη εργασία επιχειρήθηκε η ανάπτυξη ενός λογισμικού για φορητές συσκευές με σκοπό την ειδοποίηση του χρήστη όταν ένα λεωφορείο πλησιάζει σε μία στάση προκειμένου να προσέλθει έγκαιρα σε αυτήν. Για την υλοποίηση της, απαιτήθηκαν γνώσεις για τον προγραμματισμό στο λειτουργικό Android. Συγκεκριμένα κρίθηκε απαραίτητη η γνώση της αντικειμενοστραφούς γλώσσας προγραμματισμού Java με την οποία έγινε ο προγραμματισμός ολόκληρης της εφαρμογής. Για την δημιουργία βάσης δεδομένων και συγγραφή ερωτημάτων προς αυτή απαιτήθηκε η γνώση της γλώσσας ερωτημάτων βάσεων δεδομένων SQL. Εξίσου ίδιας σημασίας ήταν η γνώση των αρχείων XML που δημιουργήθηκαν για την δημιουργία του γραφικού περιβάλλοντος της εφαρμογής.

Όλη η εργασία έγινε με την χρήση του λογισμικού Eclipse στο οποίο είχε εγκατασταθεί το AndroidDeveloperTool και φυσικά με την ύπαρξη του AndroidSDK.

Τέλος, με την ανάλυση της εφαρμογής, δημιουργήθηκε ένα εγχειρίδιο χρήσης της εφαρμογής για τον απλό χρήστη που εξασφαλίζει σε αυτόν μια πλήρη εικόνα των δυνατοτήτων και των λειτουργιών της καθώς και ένα σημαντικό βοήθημα για τον προγραμματιστή εφαρμογών Android ώστε να κατανοήσει και να αξιοποιήσει τον αναπτυχθέντα κώδικα.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΙΣΤΟΤΟΠΟΙ

ΒΙΒΛΙΟΓΡΑΦΙΑ:

Shane Conder and Lauren Darcey, (2011) , Android™ Wireless Application Development, Second Edition

Matt Egan - Rosie Hattersley, (2011), The Complete Guide to Google Android, ΕκδόσειςIDG Communications

ΙΣΤΟΤΟΠΟΙ :

thenewboston.org

mybringback.com

developer.android.com

stackoverflow.com

vogela.com

wikipedia.org

techrepublic.com

techradar.com

opendatahandbook.org

opendefinition.org