ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη εφαρμογής Android για την χαρτογράφηση του Ε.Μ.Π. με τη χρήση Beacons

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Του

Ξενοφώντα Καρύδη

Επιβλέπων : Ιάκωβος Βενιέρης

Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη εφαρμογής Android για την χαρτογράφηση του Ε.Μ.Π. με τη χρήση Beacons

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Του

Ξενοφώντα Καρύδη

Επιβλέπων : Ιάκωβος Βενιέρης

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Τρίτη, 13 Μαρτίου 2018

.....	.....	.....
Ιάκωβος Βενιέρης	Δήμητρα-Θεοδώρα Κακλαμάνη	Αθανάσιος Παναγόπουλος
Καθηγητής ΕΜΠ	Καθηγήτρια ΕΜΠ	Αν. Καθηγητής ΕΜΠ

Αθήνα, Μάρτιος 2018

Ξενοφών Χ. Καρύδης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

2018

Περίληψη

Στην σημερινή εποχή τα κινητά τηλέφωνα, ακολουθώντας την ραγδαία ανάπτυξη της τεχνολογίας, έχουν γίνει αναπόσπαστο κομμάτι της ζωής μας καθώς εξυπηρετούν πλέον το μεγαλύτερο ποσοστό των αναγκών μας όσον αφορά στην πληροφόρηση, επικοινωνία, ενημέρωση, ψυχαγωγία κλπ.

Με βάση τη χρησιμότητά τους αυτή, στα πλαίσια της Σχολής, ο σκοπός της διπλωματικής είναι η σχεδίαση και ανάπτυξη δυο εφαρμογών, οι οποίες δίνουν στους μεν χρήστες/φοιτητές πληροφορίες για τοποθεσίες εντός της Σχολής καθώς και για λοιπές πληροφορίες που αφορούν τα μαθήματα, τις αίθουσες και τους καθηγητές και στους δε χρήστες/καθηγητές τη δυνατότητα να ενημερώνουν το φοιτητικό σύνολο για τυχόν αλλαγές ή ανακοινώσεις που αφορούν αυτούς ή τα μαθήματα τους.

Πιο συγκεκριμένα, από την πλευρά του φοιτητή, η εφαρμογή θα δίνει στο χρήστη τη δυνατότητα να κάνει αναζήτηση με βάση τα Μαθήματα, τους Καθηγητές και τους Τομείς και να βρίσκουν πληροφορίες πάνω σ' αυτά. Επίσης, η εφαρμογή δίνει τη δυνατότητα στο χρήστη να ενημερώνεται αυτόματα για διάφορα θέματα που αφορούν τη σχόλη ανάλογα με την τοποθεσία του (τρέχων μάθημα έξω από τη αίθουσα, στοιχεία καθηγητή έξω από το γραφείο του κ.α.), με τη χρήση Beacons, καθώς και πλοήγηση με χρήση *Google Maps* για την εύρεση επιθυμητών τοποθεσιών.

Από την πλευρά του καθηγητή, η εφαρμογή θα δίνει στον καθηγητή τη δυνατότητα να ακυρώνει κάποιο μάθημα για την επόμενη διάλεξή του, να ενημερώνει για την ολοκλήρωση των βαθμολογιών κάποιων μαθημάτων και να δημοσιεύει κάποια ανακοίνωση με οποιοδήποτε περιεχόμενο.

Η ανάπτυξη της εφαρμογής έγινε με τη βοήθεια ενός Lenovo Laptop σε περιβάλλον Windows 7. Για την δοκιμή της εφαρμογής χρησιμοποιήθηκε ένα κινητό Samsung Galaxy J3 6 το οποίο τρέχει την έκδοση 5.0.1 (API 21) του λειτουργικού συστήματος Android.

Λέξεις κλειδιά: Android, εφαρμογή, υπηρεσίες τοποθεσίας, χαρτογράφηση, Beacons, Φοιτητής, Καθηγητής

Abstract

Nowadays, smart devices and especially smarthones, are an intergral part of our everyday lives. They offer a huge amount of applications, that serve our biggest needs when it comes to information, communication, entertainment etc.

The main purpose of this thesis, is to design and develop two applications, which works as personal assistance to both Student and Professor. Students can get informed about Locations in the campus using Location Services and Professors can update news about the school schedule and inform Students about those.

More specifically, Student will be able to search from a database for information about the Professors, the Courses and the Sections of the school. Moreover he will get notified about several events based on his location and he will be able to navigate himself using GoogleMaps in order to find a place he wants.

On the other application, Professor will be able to inform students for a subject's grades, for an upcoming cancelation of a subject and publish an announcement to all the students.

The development of the application was performed using a Lenovo laptop, in an environment of Windows 7. Also, for testing the application, we used a Samsung Galaxy J3, which runs the Android 5.0.1 (API 21) version of Android operating system.

Keywords: Android, application, Location Services, charting, Beacons, Student, Professor

Ευχαριστίες

Για την εκπόνηση της παρούσας διπλωματικής εργασίας, και την ανάπτυξη της εφαρμογής αλλά και γενικότερα για την ολοκλήρωση του κύκλου των σπουδών μου, νιώθω την ανάγκη να ευχαριστήσω όλους τους ανθρώπους που με βοηθήσαν.

Αρχικά θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Ιάκωβο Βενιέρη, οποίος μου έδωσε την δυνατότητα να ασχοληθώ με ένα αντικείμενο που πραγματικά επιθυμούσα. Τον ευχαριστώ ιδιαίτερα για τις πολύτιμες γνώσεις που μου παρείχε κατά τη διάρκεια των σπουδών μου, αλλά κυρίως για την προσωπική ικανοποίηση που εισέπραξα με το πέρας της παρούσας διπλωματικής εργασίας και την αίσθηση του ότι συμπεριλαμβάνομαι πλέον και εγώ στους εν δυνάμει προγραμματιστές.

Ιδιαίτερες ευχαριστίες θα ήθελα να δώσω στους Υποψήφιους Διδάκτορες, κ. Εμμανουήλ Καραμανή και κ. Πέτρο Φ. Μπάκαλο, χωρίς τη βοήθεια των οποίων θα ήταν αδύνατη η ολοκλήρωση της διπλωματικής εργασίας. Η συμβολή τους στην ανάπτυξη της εφαρμογής υπήρξε καθοριστική.

Επίσης, θα ήθελα να ευχαριστήσω όλο το προσωπικό του Εργαστηρίου Ευφυών Επικοινωνιών και Δικτύων Ευρείας Ζώνης, που μου πρόσφερε τη βοήθειά του όποτε αυτή ζητήθηκε.

Τέλος, θα ήθελα να ευχαριστήσω από καρδίας τη μητέρα μου και τον πατέρα μου που με στήριξαν με κάθε τρόπο σε όλο το ταξίδι της φοιτητικής μου ζωής και γι' αυτό τους αφιερώνω την παρούσα διπλωματική εργασία.

Πίνακας Περιεχομένων

1 Εισαγωγή

1.1 Η εξέλιξη των κινητών τηλεφώνων

Η περιπέτεια της κινητής τηλεφωνίας ξεκίνησε αμέσως μετά τον Β' παγκόσμιο πόλεμο με τις πρώτες προσπάθειες Σουηδών , Φιλανδών και Αμερικανών να δημιουργήσουν μια συσκευή που δεν εξαρτάται από καλωδιακή σύνδεση με δίκτυο παροχής τηλεφωνίας, ούτε από κάποια τοπική ασύρματη συσκευή εκπομπής ραδιοφωνικού σήματος χαμηλής συχνότητας. Όμως ως ληξιαρχική πράξη γέννησής της θεωρείται η 3η Απριλίου 1973 με την κατασκευή και έναρξη της χρήσης του κινητού τηλεφώνου από το κοινό. Βέβαια η απογείωση των κινητών τηλεφώνων άρχισε την δεκαετία του 90' όπου με την ψηφιοποίηση δικτύων και συσκευών οι συσκευές έγιναν μικρότερες και ελαφρύτερες χωρώντας αντίθετα με παλαιότερα στην παλάμη ή στην τσέπη . Περάσαμε έτσι στα κινητά δεύτερης γενιάς (2G) όπου παρείχαν και άλλες λειτουργίες όπως την αποστολή σύντομων γραπτών μηνυμάτων και τη λήψη φωτογραφιών. Στις αρχές του 21ου αιώνα ήρθαν στην αγορά τα κινητά τρίτης γενιάς τα λεγόμενα 3G δηλαδή 3rd generation όπου είχαν απεριόριστες δυνατότητες και λειτουργίες!

Τα κινητά τηλέφωνα δεν είχαν πάντοτε οθόνη αφής, κάμερες πολλών Megapixels και την δυνατότητα να χρησιμοποιηθούν ως ένας μικροσκοπικός Η/Υ, λύνοντας μας τα χέρια. Για την ακρίβεια, δεν ήταν καν μικροσκοπικά αφού ειδικά στα πρώτα μοντέλα το μέγεθός τους δεν τα καθιστούσε και πολύ “κινητά”! Η πρώτη συσκευή κινητού τηλεφώνου, η οποία άλλαξε για πάντα την καθημερινότητα μας, εμφανίστηκε το 1973 και στη συνέχεια ακολούθησαν χιλιάδες μοντέλα που βοήθησαν στην εξέλιξη της μορφής αλλά και των

λειτουργιών αυτού του μικρού θαύματος της τεχνολογίας που έμελλε να γίνει απαραίτητο gadget για κάθε άνθρωπο.

1.2 Το λειτουργικό Android

Όπως φαίνεται λοιπόν, τα κινητά κατέκτησαν μια εξέχουσα θέση στην τσέπη των ανθρώπων και για αυτό το λόγο οι εταιρίες προγραμματισμού αφοσιώθηκαν να τα κάνουν όσο πιο λειτουργικά και ταυτόχρονα εύχρηστα γίνετε. Οι καινοτομίες εμφανίζονταν η μια μετά την άλλη με τη μεγαλύτερη να είναι τα έξυπνα κινητά, η αλλιώς "smartphones".

Τα smartphones χρησιμοποιούσαν διάφορα λειτουργικά συστήματα τα οποία τα καθιστούσαν ολοένα και πιο φιλικά στο χρήστη και με όλο και περισσότερες λειτουργίες και δυνατότητες. Οι δυο μεγαλύτεροι αντίπαλοι σ΄ αυτή την κατηγορία είναι η Apple με το λειτουργικό σύστημα *iOS* και η Google με το *Android*. Και οι δυο ανταποκρίθηκαν πολύ αποτελεσματικά στις επιτακτικές ανάγκες της ανθρωπότητας προσφέροντας συνεχώς τεράστιες καινοτομίες που επέτρεπαν στους κοινούς χρήστες να χρησιμοποιούν τα κινητά τηλέφωνα με τρόπους που δεν είχαν φανταστεί στο παρελθόν. Συγκεκριμένα στην παρούσα διπλωματική θα γίνει αναφορά στο λογισμικό **Android**.

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα smartphones και τα tablet, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες βιντεοπαιχνιδιών, συνηθισμένους Η/Υ (π.χ. το HP Slate 21) και σε άλλες ηλεκτρονικές συσκευές.

Το μεγαλύτερο μερίδιο της αγοράς των λειτουργικών συστημάτων, κατέχει το Android της Google, με το συντριπτικό ποσοστό των 86.2% των πωλήσεων το τελευταίο τρίμηνο της χρονιάς και ακολουθεί το iOS της Apple με 12.9%. Όπως είναι λογικό και αναμενόμενο από τη στιγμή που το Android κυριαρχεί στην αγορά, γίνετε αντικείμενο μελέτης και ανάπτυξης από πολλούς προγραμματιστές με στόχο την προσφορά τους στον τομέα των εφαρμογών αλλά και ως ένα επιπλέον εφόδιο στην ανταγωνιστική αγορά εργασίας.

Παρακάτω φαίνεται ένας δημοσιευμένος πίνακας πωλήσεων του 2016:

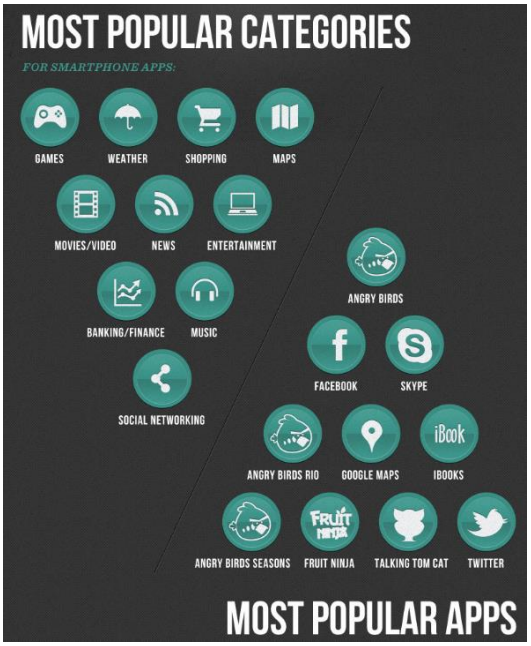
Worldwide Smartphone Sales to End Users by Operating System in 2Q16 (Thousands of Units)

Operating System	2Q16 Units	2Q16 Market Share (%)	2Q15 Units	2Q15 Market Share (%)
Android	296,912.8	86.2	271,647.0	82.2
iOS	44,395.0	12.9	48,085.5	14.6
Windows	1,971.0	0.6	8,198.2	2.5
Blackberry	400.4	0.1	1,153.2	0.3
Others	680.6	0.2	1,229.0	0.4
Total	**344,359.7**	**100.0**	**330,312.9**	**100.0**

Source: Gartner (August 2016)

Σήμερα, βασιζόμαστε στις εφαρμογές να κάνουν σχεδόν τα πάντα και όπως είναι επόμενο μετά από πολλά εκατομμύρια λήψεων αυτών των apps, να έχει διαμορφωθεί μια σχετική οικονομία γύρω από αυτά η οποία και είναι ισχυρή όσο ποτέ.

Το App Development, έχει δημιουργήσει περισσότερες από 466.000 θέσεις εργασίας σε όλες τις διαθέσιμες πλατφόρμες, σύμφωνα με μια έρευνα που πραγματοποιήθηκε από το TechNet. Σύμφωνα με αυτή την έρευνα η οικονομία αυτή έχει επεκταθεί και σε άλλους τομείς σε τοπικό επίπεδο, καθώς πολλοί είναι οι προγραμματιστές που ολοκληρώνουν τις δουλειές τους μέσα από Internet Café. Η ανάπτυξη των εφαρμογών για smartphones και ιδιαίτερα αυτές για Android, έχει δημιουργήσει μια ραγδαία ανάπτυξη στον χώρο των εφαρμογών, την οποία και βιώνουμε καθημερινά. Η δυνατότητα επικοινωνίας από οπουδήποτε, η άμεση πρόσβαση στο διαδίκτυο, η επεκτασιμότητα με τις εφαρμογές, άλλαξαν μια για πάντα το τοπίο, ανοίγοντας οριστικά τον δρόμο για τα κινητά και όπως όλα δείχνουν νικητές αναδεικνύονται τα smartphones, με κυρίαρχο το Android.Οι εφαρμογές που αναπτύσσονται και δημοσιεύονται κάθε μέρα στο Google Play Store είναι πολυάριθμες και αφορούν μια μεγάλη γκάμα των κοινωνικών και προσωπικών αναγκών του ανθρώπου. Μερικές κιόλας έχουν γίνει κατά κάποιο τρόπο απαραίτητες στη ζωή μας καθώς χωρίς αυτές θα είμασταν αποκομμένοι από μεγάλα κομμάτια πληροφόρησης και επικοινωνίας για την υπόλοιπη υφήλιο.



Κατηγορίες που καλύπτουν πλέον τα Mobile Apps.

Λογότυπο (Logo) του Android

Το logo του Android είναι ένα πράσινο ρομπότ ονόματι Droid, και έχει γίνει πλέον δημοφιλές και αναγνωρίσιμο παντού. Χαρακτηριστικό είναι επίσης το γεγονός ότι η Google επέλεξε να δίνει ξεχωριστές ονομασίες. Συγκεκριμένα, οι εκδόσεις του Android έχουν θέμα από την ζαχαροπλαστική στην κωδική ονομασία τους, και κυκλοφόρησαν σε αλφαβητική σειρά, εξαιρουμένων των εκδόσεων 1.0 και 1.1, που δεν τέθηκαν υπό συγκεκριμένα κωδικά ονόματα

Ας τις θυμηθούμε:

Κωδικό όνομα	Νούμερο έκδοσης	Ημερομηνία αρχικής κυκλοφορίας	Επίπεδο API
N/A	1.0	23 Σεπτεμβρίου 2008	1
	1.1	9 Φεβρουάριου 2009	2
Cupcake	1.5	27 Απριλίου 2009	3
Donut	1.6	15 Σεπτεμβρίου 2009	4
Eclair	2.0 – 2.1	26 Οκτωβρίου 2009	5–7
Froyo	2.2 – 2.2.3	20 Μαΐου 2010	8
Gingerbread	2.3 – 2.3.7	6 Δεκεμβρίου 2010	9–10
Honeycomb	3.0 – 3.2.6	22 Φεβρουάριου 2011	11–13
Ice Cream Sandwich	4.0 – 4.0.4	18 Οκτωβρίου 2011	14–15
Jelly Bean	4.1 – 4.3.1	9 Ιουλίου 2012	16–18
KitKat	4.4 – 4.4.4	31 Οκτωβρίου 2013	19–20
Lollipop	5.0 – 5.1.1	12 Νοεμβρίου 2014	21–22
Marshmallow	6.0 – 6.0.1	5 Οκτωβρίου 2015	23
Nougat	**7.0**	**22 Αυγούστου 2016**	**24**

Η κύρια πλατφόρμα υλικού για το Android είναι η αρχιτεκτονική ARM, ενώ επίσημα υποστηρίζονται επίσης οι x86 και MIPS αρχιτεκτονικές και σε 32bit αλλά και σε 64bit συστήματα. Οι ελάχιστες απαιτήσεις υλικού έχουν αναβαθμιστεί ανά τις περιόδους αρχίζοντας από 32MB μνήμης RAM αλλά λιγότερο από 128MB δεν συνίσταται. Από την έκδοση 4.4 και μετά, που είναι για συσκευές ARM-based που απαιτούν ARMv7 επεξεργαστή, η ελάχιστη ποσότητα RAM μνήμης είναι 512MB καθώς το λειτουργικό απαιτεί τουλάχιστον 340MB και συσκευές με λιγότερο από 512MB θεωρούνται συσκευές 'χαμηλής μνήμης'. Τέλος, για χρήση GPU το Android 4.0 χρησιμοποιεί OpenGL ES 2.0 και η επιτάχυνση υλικού έγινε υποχρεωτική, ανεξάρτητα αν οι εφαρμογές το χρησιμοποιούν άμεσα ή όχι. Αργότερα, το Android 4.3 πρόσθεσε υποστήριξη για το OpenGL ES 3.0 αν χρησιμοποιείται, με παράλληλη την υποστήριξη και για τις δύο παλαιότερες εκδόσεις (ES 2.0 και 1.0) που εξακολουθεί να είναι υποχρεωτική.

1.3 Gimbal

Η Gimbal είναι μια εταιρία στην Αμερική, η οποία έχει σαν αντικείμενο την ανάπτυξη mobile platforms των οποίων χρήση γίνεται από λιανοπωλητές, διαφημιστές, (OOH) out-of-home networks, κάθε ένας για τον δικό του σκοπό. Ένα από τα αντικείμενα ανάπτυξης τους ήταν τα Gimbal Beacons, τα οποία είναι μικρές (BLE) Bluetooth Low Energy συσκευές, τα οποία γίνονται ανιχνεύσιμα από κάποιον χρήστη μέσω Bluetooth από το κινητό, σε αποστάσεις από μερικά εκατοστά έως και 50 μέτρα.

Χρησιμοποιώντας την πλατφόρμα της Gimbal, μπορεί κανείς να αναπτύξει apps, με τα οποία λαμβάνει διαφόρων ειδών μετρικές, όπως απόσταση (RSSI), θερμοκρασία, Current Location κ.α., να συνδέεται σε απεριόριστο αριθμό Geofences και Beacons, indoor και outdoor και να αξιοποιεί με όποιο τρόπο θέλει τα δεδομένα αυτά. Επίσης παρέχει τη δυνατότητα ειδοποιήσεων στο χρήστη μέσω κάποιου Cloud Service. Η Gimbal Platform παρέχει όλα τα απαραίτητα εργαλεία όπως Android (και iOS) SDK, εγγραφή και ενεργοποίηση Beacons και

Geofences και διάφορα άλλα APIs που διευκολύνουν την εκμετάλλευση αυτών των τεχνολογιών.

Στην περίπτωσή μας, θα χρησιμοποιήσουμε αυτές τις υπηρεσίες Geofencing είτε για την καθοδήγηση του χρήστη μέσω Google Maps σε επιθυμητά σημεία, είτε για την ειδοποίησή τους όποτε βρίσκονται στην εμβέλεια ενός Beacon ή ενός Geofencing το οποίο έχει δηλωθεί στην πλατφόρμα της Gimbal και έχει συνδεθεί να στέλνει το κατάλληλο μήνυμα.

2 Τεχνολογίες

2.1.1 ***Android Studio IDE***

Το *Android Studio* είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android. Ανακοινώθηκε τον Μάιο του 2013 στο συνέδριο *Google I/O* από την *Google Product Manager : Katherine Chou*. Το *Android Studio* είναι διαθέσιμο ελεύθερα με την άδεια *Apache License 2.0*. Μπορεί να το κατεβάσει κάθε χρήστης του Internet, να το εγκαταστήσει και να φτιάξει τις δικές του εφαρμογές.

Είναι βασισμένο στο λογισμικό της JetBrains' IntelliJ IDEA, το *Android Studio* σχεδιάστηκε αποκλειστικά για προγραμματισμό Android. Είναι διαθέσιμο για όλα τα ευρέως διαδεδομένα λειτουργικά συστήματα (Windows, Mac OS X, Linux) και αντικατέστησε τα Eclipse Android Development Tools (ADT) ως το κύριο IDE της Google για ανάπτυξη εφαρμογών Android.

Το Android Software Development Kit (SDK) περιέχει όλα τα εργαλεία και τις διεπαφές που απαιτούνται για την ανάπτυξη, εγκατάσταση, λειτουργία και τη δοκιμή των εφαρμογών που εμφανίζονται στην οθόνη μιας Android συσκευής. Όλες οι εφαρμογές κατασκευάζονται χρησιμοποιώντας τα πλαίσια (frameworks) του συστήματος του Android καθώς και την αντικειμενοστραφή γλώσσα προγραμματισμού Java. Οι εφαρμογές εγκαθίστανται και τοποθετούνται δίπλα στις άλλες εγκατεστημένες εφαρμογές στην αρχική οθόνη της συσκευής και είναι πάντα διαθέσιμες στο χρήστη. Η κατανόηση των τεχνολογιών και των εργαλείων που συνθέτουν το Android SDK είναι απαραίτητη για να σχεδιασθεί και να υλοποιηθεί αποτελεσματικά μια εφαρμογή.

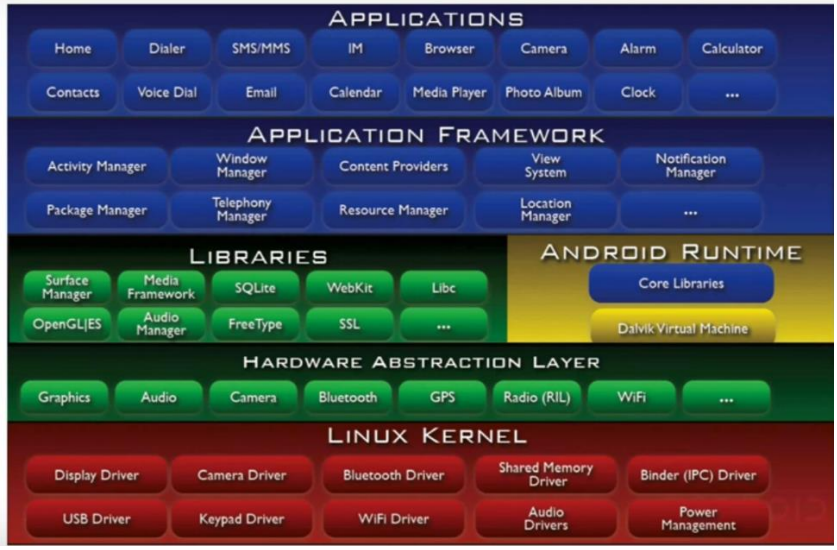
Στο σημείο αυτό πρέπει να τονιστεί ότι το Android Studio παρέχει έναν Default Simulator. Ο Simulator χρησιμοποιείται από τον προγραμματιστή για την προεπισκόπηση της εφαρμογής στην πράξη. Δίνεται δυνατότητα στο χρήστη να χτίσει μια Virtual συσκευή με δικές του προδιαγραφές και να συλλέγει δεδομένα κατά το τρέξιμο της εφαρμογής τα οποία βοηθούν τον χρήστη να αναλύσει, να διορθώσει και να ολοκληρώσει την εφαρμογή του. Ωστόσο, εκτός από αυτήν την επιλογή, υπάρχει και η δυνατότητα να γίνουν όλα αυτά σε πραγματικό κινητό με λογισμικό Android, το οποίο αναγνωρίζεται από το Android Studio και χρησιμοποιείται ακριβώς με τον ίδιο τρόπο όπως το Virtual κινητό από τον Simulator.Εν προκειμένω, χρησιμοποιείται ρεαλιστική συσκευή πάνω στην οποία χτίστηκε το project.

2.1.2 ***Αρχιτεκτονική του Android OS***

Η διαστρωμάτωση της αρχιτεκτονικής του Android, κατά βάση αποτελείται από 4 επίπεδα (layers), εκ των οποίων το καθένα έχει το δικό του ρόλο και όλα μαζί συντελούν στην αποτελεσματική και γρήγορη απόκριση στις προγραμματισμένες από τον Developer εφαρμογές. Τα επίπεδα αυτά είναι

i) Linux Kernel Layer
ii) Hardware Abstraction Layer – Libraries
iii) Application Framework Layer
iv) Applications-UI Layer

Παρακάτω δίνεται μια σύντομη περιγραφή του καθενός για την βασική και απλή κατανόηση του ρόλου τους.



i) **Kernel (πυρήνας)** ονομάζουμε το τμήμα ενός λειτουργικού, το οποίο αναλαμβάνει **την διασύνδεση των εφαρμογών με το hardware.**
Στον **Android Linux Kernel** μίας συσκευής, ενσωματώνονται όλοι οι drivers για τα διάφορα υποσυστήματά της, όπως για παράδειγμα οι drivers για το Wi-Fi, Bluetooth, GPS κλπ.

ii) Στο ακριβώς επόμενο layer, βρίσκεται ένα σύνολο από βιβλιοθήκες που περιλαμβάνουν open-source browser engine WebKit, SQLite database για την αποθήκευση και χρήση δεδομένων από εφαρμογές, βιβλιοθήκες για την αναπαραγωγή βίντεο και ήχου, SSL libraries για την ασφάλεια της σύνδεσης με το internet κλπ.

iii) Το Application Framework layer παρέχει πολλά υψηλού επιπέδου services στα applications με τη μορφή Java κλάσεων. Τα services αυτά, επιτρέπεται να τα χρησιμοποιήσει ο developer για την ανάπτυξη δικών του εφαρμογών.

iv) Στο Application layer βρίσκονται όλα τα applications που έχουν γίνει install είτε από το χρήστη είτε ήταν εγκατεστημένα πριν την αγορά της συσκευής. Παραδείγματα applications είναι Games, Contacts, Chrome κλπ.

2.2 ***Java***

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής *Sun Microsystems* σαν ένα μέρος έργου ανάπτυξης λογισμικού για ηλεκτρονικές συσκευές καταναλωτικού επιπέδου. Ο τρόπος αυτός ανάπτυξής της τη μετέτρεψε σε μια ιδανική γλώσσα για τη διανομή εκτελέσιμων προγραμμάτων μέσω του παγκόσμιου ιστού καθώς επίσης και σε μια γλώσσα γενικού σκοπού για την ανάπτυξη προγραμμάτων που θα μπορούν εύκολα να μεταφέρονται σε διαφορετικά λειτουργικά συστήματα.

Ένα από τα πολλά χαρακτηριστικά της γλώσσας αυτής έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος της πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουνε το ίδιο σε Windows, Unix, Linux, Macintosh χωρίς να χρειαστεί να ξανά γίνει μεταγλώττιση ή να αλλάξει ο πηγαίος κώδικας για κάθε λειτουργικό σύστημα. Για να επιτευχθεί αυτό χρειαζόταν κάποιος τρόπος ώστε τα προγράμματα γραμμένα σε Java να μπορούν να είναι «κατανοητά» από κάθε υπολογιστή, ανεξάρτητα από το είδος του επεξεργαστή ή του λειτουργικού συστήματος. Η λύση δόθηκε με την ανάπτυξη της εικονικής μηχανής.

2.2.1 ***Εικονική Μηχανή***

Αφού γραφεί κάποιο πρόγραμμα σε Java, στη συνέχεια μεταγλωττίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode). Ο κώδικας byte είναι η μορφή που παίρνει ο πηγαίος κώδικας της Java όταν μεταγλωττιστεί. Όταν πρόκειται να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, το Java Virtual Machine που πρέπει να είναι εγκατεστημένο σε αυτό θα αναλάβει να διαβάσει τα αρχεία .class. Στη συνέχεια τα μεταφράζει σε γλώσσα μηχανής που να υποστηρίζεται από το λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να εκτελεστεί. Πιο σύγχρονες εφαρμογές της εικονικής Μηχανής μπορούν και μεταγλωττίζουν εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (εγγενή κώδικα ή native code) με αποτέλεσμα να βελτιώνεται η ταχύτητα). Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java. Η JVM είναι λογισμικό που εξαρτάται από την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές κλπ.

Οτιδήποτε θέλει να κάνει ο προγραμματιστής (ή ο χρήστης) γίνεται μέσω της εικονικής μηχανής. Αυτό βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη - υπολογιστή. Ο προγραμματιστής δεν

μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Από την άλλη μεριά ούτε ο χρήστης μπορεί να κατεβάσει «κακό» κώδικα από το δίκτυο και να τον εκτελέσει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα όπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως.

2.2.2 *Ο συλλέκτης απορριμμάτων (Garbage Collector)*

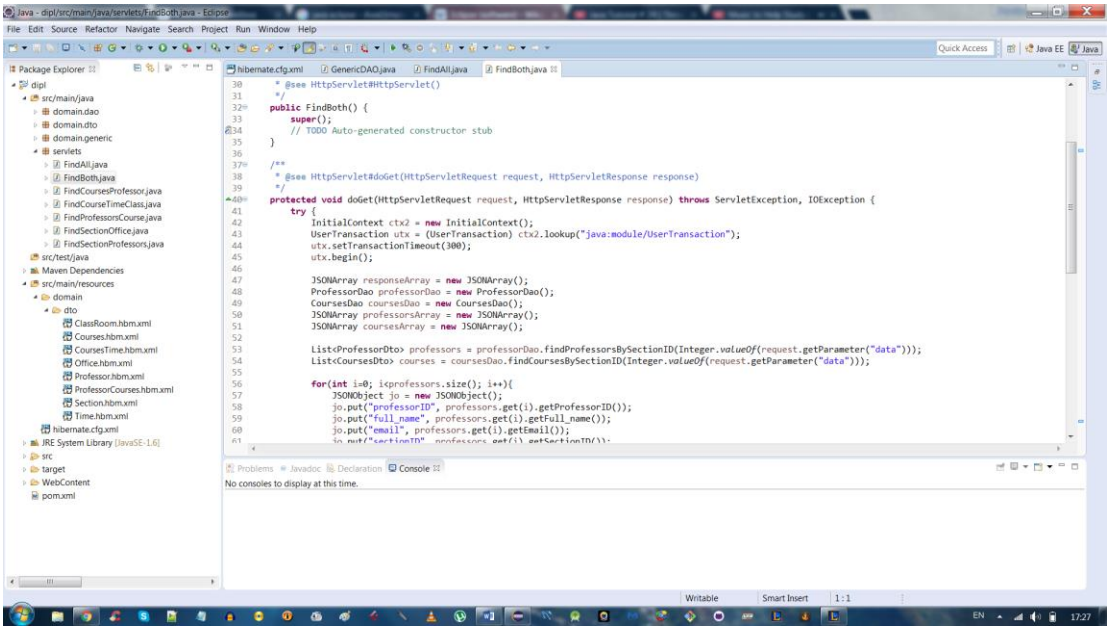
Ακόμα μία ιδέα που βρίσκεται πίσω από τη *Java* είναι η ύπαρξη του συλλέκτη απορριμμάτων (*Garbage Collector*). Συλλογή απορριμμάτων είναι μία κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο. Αυτή η απελευθέρωση μνήμης στη Java είναι αυτόματη και γίνεται μέσω του συλλέκτη απορριμμάτων. Υπεύθυνη για αυτό είναι και πάλι η εικονική μηχανή η οποία μόλις «καταλάβει» ότι ο σωρός (heap) της μνήμης (στη Java η συντριπτική πλειοψηφία των αντικειμένων αποθηκεύονται στο σωρό σε αντίθεση με τη C++ όπου αποθηκεύονται κυρίως στη στοίβα) κοντεύει να γεμίσει ενεργοποιεί το συλλέκτη απορριμμάτων. Έτσι ο προγραμματιστής δε χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για σφάλματα δεικτών. Αυτό είναι ιδιαίτερα σημαντικό γιατί είναι κοινά τα σφάλματα προγραμμάτων που οφείλονται σε λανθασμένο χειρισμό της μνήμης.

2.2.3 *Επιδόσεις*

Παρόλο που η εικονική μηχανή προσφέρει όλα αυτά (και όχι μόνο) τα πλεονεκτήματα, η *Java* αρχικά ήταν πιο αργή σε σχέση με άλλες προγραμματιστικές γλώσσες υψηλού επιπέδου (high-level) όπως η C και η C++. Εμπειρικές μετρήσεις στο παρελθόν είχαν δείξει ότι η C++ μπορούσε να είναι αρκετές φορές γρηγορότερη από την Java. Ωστόσο γίνονται προσπάθειες από τη Sun για τη βελτιστοποίηση της εικονικής μηχανής, ενώ υπάρχουν και άλλες υλοποιήσεις της εικονικής μηχανής από διάφορες εταιρίες (όπως της IBM), οι οποίες μπορεί σε κάποια σημεία να προσφέρουν καλύτερα και σε κάποια άλλα χειρότερα αποτελέσματα. Επιπλέον με την καθιέρωση των μεταγλωττιστών JIT (Just In Time), οι οποίοι μετατρέπουν τον κώδικα byte απευθείας σε γλώσσα μηχανής, η διαφορά ταχύτητας από τη C++ έχει μικρύνει κατά πολύ.

2.2.4 ***Το περιβάλλον ανάπτυξης Eclipse***

Το *Eclipse* είναι ένα περιβάλλον προγραμματισμού συμβατού με πολλές γλώσσες προγραμματισμού. Είναι γραμμένο κυρίως σε Java και μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών σε Java αλλά και σε άλλες γλώσσες μέσω διαφόρων plug-ins. Το Eclipse SDK έχει γραφτεί από προγραμματιστές Java και προορίζεται για αυτή τη γλώσσα προγραμματισμού. Είναι φυσικά ένα πολύ χρήσιμο εργαλείο, για αυτό και είναι τόσο διαδεδομένη η χρήση του. Ακόμα και αν η κύρια γλώσσα προγραμματισμού που χρησιμοποιεί κάποιος δεν είναι Java, δεν είναι λίγες οι περιστάσεις που το Eclipse μπορεί να «λύσει» τα χέρια του προγραμματιστή.



Το περιβάλλον Eclipse Mars, έκδοση 4.5

Το Eclipse υποστηρίζει development για πολλούς servers όπως ο Tomcat, ο Apache, ο GlassFish και πολλοί άλλοι. Τις περισσότερες φορές παρέχεται η δυνατότητα για εγκατάσταση του απαιτούμενου server απευθείας από το IDE του eclipse. Υποστηρίζει επίσης remote debugging, ώστε να μπορεί ο χρήστης να επιβλέπει τις τιμές των μεταβλητών και την πορεία του κώδικα ενός application που τρέχει στον servers μας.

2.3 JSON

Το JSON (Javascript Object Notation) είναι ένας τρόπος ανταλλαγής δεδομένων που στηρίζεται στην εύκολη ανάγνωση από τον άνθρωπο. Προέρχεται από την γλώσσα Javascript για την αναπαράσταση απλών δεδομένων και συσχετισμένων πινάκων, που ονομάζονται αντικείμενα. Παρά τη σχέση του με την Javascript , είναι ανεξάρτητη γλώσσα με προγράμματα ανάλυσης που διατίθενται σε πολλές γλώσσες προγραμματισμού. Το JSON χρησιμοποιείται συχνά για σειριοποίηση και διαβίβαση δεδομένων πάνω από μια σύνδεση δικτύου. Κατά κύριο λόγο χρησιμοποιείται για τη μετάδοση δεδομένων μεταξύ ενός server και μια web εφαρμογής την οποία μπορεί να εξυπηρετεί και λειτουργεί σαν εναλλακτική λύση στη θέση της XML, αν και πλέον η χρήση του υπερτερεί .

Το JSON είναι χτισμένο πάνω σε δυο δομές:

- Ένα σύνολο από ζεύγη ονόματος – τιμής. Στις διάφορες γλώσσες αυτό μεταφράζεται ως αντικείμενο, record, struct, dictionary, πίνακας ή λίστα.
- Μια ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού αυτό γίνεται αντιληπτό σαν πίνακας ή λίστα.

Αυτές είναι ευρέως αναγνωρίσιμες δομές δεδομένων. Όλες οι σύγχρονες γλώσσες προγραμματισμού μπορούν να τις υποστηρίξουν με τον έναν ή τον άλλον τρόπο. Αυτό φυσικά είναι και το μεγάλο πλεονέκτημα του JSON. Το γεγονός δηλαδή ότι μέσω του JSON το δεδομένα μπορούν να πάρουν μια μορφή που είναι αναγνώσιμη από πολλές διαφορετικές γλώσσες μπορεί να λύσει σημαντικά προβλήματα που αφορούν στη μετάδοση δεδομένων.

2.4 Hibernate

Το Hibernate Framework είναι λογισμικό ανοιχτού κώδικα (ελεύθερο λογισμικό) που σκοπό έχει να συνδέσει τα αντικείμενα που δημιουργούνται σε μια αντικειμενοστραφή γλώσσα προγραμματισμού (Java) με τους πίνακες μιας σχεσιακής βάσης δεδομένων. Η σύνδεση αυτή επιτυγχάνεται με την χρήση επιπρόσθετης πληροφορίας (metadata) που τοποθετείται κατάλληλα (μαζί με τον κώδικα Java ή σε ξεχωριστά xml αρχεία) και περιγράφει την αντιστοιχία μεταξύ των αντικειμένων και της βάσης δεδομένων. Γενικά το Hibernate προσφέρει την αυτόματη μετατροπή της μιας μορφής (αντικείμενα) στην άλλη (σχεσιακή βάση δεδομένων).

2.4.1 Χρήση

Το Hibernate συγκαταλέγεται στην κατηγορία του λογισμικού ORM (object/relational mapping). Το λογισμικό ORM στοχεύει στην δημιουργία μιας διεπαφής (interface) μεταξύ των διαδεδομένων σχεσιακών βάσεων δεδομένων και του αντικειμενοστραφούς προγραμματισμού. Με απλά λόγια, προσφέρει την χρησιμοποίηση μιας σχεσιακής βάσης δεδομένων σαν να ήταν αντικειμενοστραφής. Για να το επιτύχει αυτό δημιουργεί αντιστοιχίες μεταξύ των εννοιών του αντικειμενοστραφούς προγραμματισμού (συσχετίσεις, κληρονομικότητα, πολυμορφισμός) - που δεν υπάρχουν σε μια σχεσιακή βάση δεδομένων - και των πινάκων και σχέσεων μεταξύ των πινάκων μιας σχεσιακής βάσης. Με αυτό τον τρόπο ο προγραμματιστής βλέπει τελικά μια αντικειμενοστραφή βάση δεδομένων, παρ΄ όλο που στην ουσία χρησιμοποιεί μια σχεσιακή. Έτσι ο προγραμματιστής χρησιμοποιεί τα αντικείμενα της συγκεκριμένης εφαρμογής, τα τροποποιεί σχετικά με τη λογική της εφαρμογής που αναπτύσσει και τα αποθηκεύει (τροποποιεί, διαγράφει και αναζητά) στην βάση ως αντικείμενα, σκεπτόμενος δηλαδή με αντικειμενοστραφείς έννοιες και όχι με βάση το σχήμα της σχεσιακής βάσης δεδομένων. Σε αυτό το σημείο είναι το Hibernate που, γνωρίζοντας την αντιστοιχία μεταξύ βάσης και λογικής της εφαρμογής, αναλαμβάνει να κατασκευάσει την κατάλληλη εντολή της SQL η οποία και στέλνεται τελικά στην βάση δεδομένων. Έπειτα, τα αποτελέσματα που επιστρέφει η βάση το Hibernate τα επιστρέφει στον προγραμματιστή ως αντικείμενα της εφαρμογής. Είναι δηλαδή ένα ενδιάμεσο επίπεδο μεταξύ της εφαρμογής και της βάσης δεδομένων.

2.4.2 Πλεονεκτήματα Hibernate

Το Hibernate προσφέρει τα παρακάτω στον προγραμματιστή:

- **Παραγωγικότητα**: Στην ανάπτυξη λογισμικού ένα μεγάλο μέρος της προγραμματιστικής προσπάθειας αφιερώνεται στην διεπαφή της εφαρμογής με τη βάση δεδομένων. Το Hibernate αυτοματοποιώντας τις βασικές λειτουργίες Δημιουργία/Ανάγνωση/Τροποποίηση/Διαγραφή (CRUD – Create Read Update Delete) επιτρέπει αρχικά στον προγραμματιστή να επικεντρώνει την προσπάθειά του στη λογική της εφαρμογής (business logic). Επίσης, υπάρχει η δυνατότητα να ακολουθηθούν δύο στρατηγικές ανάπτυξης λογισμικού: είτε αρχίζοντας από το μοντέλο δεδομένων είτε από τη βάση δεδομένων. Αυτό μειώνει σε μεγάλο βαθμό το χρόνο ανάπτυξης.
- **Συντηρησιμότητα**: Με τη χρήση του Hibernate γράφονται σημαντικά λιγότερες γραμμές κώδικα και ο κώδικας είναι πιο κατανοητός και καλογραμμένος. Αυτό κάνει την συντήρηση της εφαρμογής ευκολότερη.

- **Ανεξαρτησία από τη βάση δεδομένων**: Με τη συμβατότητα του Hibernate με διαφορετικές βάσεις δεδομένων και τη δυνατότητα σύνδεσής του με τη βάση μέσω δηλώσεων οριζομένων σε ειδικό αρχείο η αναπτυσσόμενη εφαρμογή μπορεί με ελάχιστες τροποποιήσεις να χρησιμοποιηθεί με βάσεις δεδομένων διαφορετικών κατασκευαστών. Το γεγονός αυτό στερεί μεν από το Hibernate την εκμετάλλευση των ιδιαίτερων χαρακτηριστικών της χρησιμοποιούμενης βάσης, όμως, και σε αυτή την περίπτωση, δίνεται η δυνατότητα χρήσης πηγαίας SQL μέσα στο Hibernate που εκμεταλλεύεται τα ιδιαίτερα αυτά χαρακτηριστικά. Αυτό βέβαια μειώνει την ανεξαρτησία του Hibernate.

2.5 Βάση Δεδομένων και MySql

Με τον όρο *βάση δεδομένων* εννοείται μία συλλογή από *συστηματικά μορφοποιημένα* σχετιζόμενα δεδομένα στα οποία είναι δυνατή η ανάκτηση δεδομένων μέσω αναζήτησης κατ' απαίτηση. Ειδικότερα, στην επιστήμη της πληροφορικής και στην καθημερινή χρήση των ηλεκτρονικών υπολογιστών, με τον όρο *βάσεις δεδομένων* αναφερόμαστε σε οργανωμένες, διακριτές συλλογές σχετιζόμενων δεδομένων ηλεκτρονικά και ψηφιακά αποθηκευμένων, στο λογισμικό που χειρίζεται τέτοιες συλλογές (Σύστημα Διαχείρισης Βάσεων Δεδομένων, ή *DBMS*).

Για τη διαχείριση της βάσης δεδομένων μας, χρησιμοποιήθηκε το λογισμικό MySQL. Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που μετρά περισσότερες από 11 εκατομμύρια εγκαταστάσεις. Το πρόγραμμα τρέχει έναν εξυπηρετητή (server) παρέχοντας πρόσβαση πολλών χρηστών σε ένα σύνολο βάσεων δεδομένων. Η MySQL είναι δημοφιλής βάση δεδομένων για διαδικτυακά προγράμματα και ιστοσελίδες. Χρησιμοποιείται σε κάποιες από τις πιο διαδεδομένες διαδικτυακές υπηρεσίες, όπως το Flickr, το YouTube, η Wikipedia, το Google, το Facebook και το Twitter.

3 Ανάλυση

3.1 ***Απαιτήσεις εφαρμογής***

Όπως προαναφέραμε, ο ρυθμός της καθημερινότητας έχει ανέβει εκθετικά τα τελευταία χρόνια. Οι υποχρεώσεις από εργασιακές μέχρι υπηρεσιακές συνεχώς πληθαίνουν με αποτέλεσμα να είναι αδύνατο να ανταπεξέλθουμε στους ρυθμούς αυτούς χωρίς τη χρήση κάποιας βοήθειας. Στο σημείο αυτό εμφανίζονται τα «έξυπνα» κινητά και οι δυνατότητες που φέρουν μαζί τους. Ως εκ τούτου, όλοι οι καθημερινοί άνθρωποι αργά ή γρήγορα αποκτούν μια επαφή με την τεχνολογία και τις διευκολύνσεις που παρέχει, ώστε να είναι σε θέση να βελτιώσουν την καθημερινή τους απόδοση και να μειώσουν το φόρτο εργασίας που έχουν να διαχειριστούν.

Στην περίπτωσή μας, ένας σύγχρονος φοιτητής μπορεί να εκμεταλλευτεί τις δυνατότητες που του παρέχει η τεχνολογία, είτε για να μάθει πληροφορίες που αφορούν τη σχολή του είτε ακόμα και για να πλοηγηθεί μέσα σ' αυτή βρίσκοντας τα δικά του σημεία ενδιαφέροντος, όλα αυτά μέσα από τη χρήση μιας εφαρμογής κινητού.

Αντίστοιχα, ένας καθηγητής στις μέρες μας μπορεί να εκμεταλλευτεί επίσης την τεχνολογία για να διευθετήσει θέματα που αφορούν την σχολή. Τα θέματα αυτά κυρίως επικεντρώνονται στην άμεση για τον φοιτητή και εύκολη για τον καθηγητή ενημέρωση διαφόρων θεμάτων, με τα οποία ασχολούνται συχνότερα φοιτητές και καθηγητές.

Οι «έξυπνες» συσκευές, δίνουν τη δυνατότητα και στις δύο πλευρές να έχουν πρόσβαση σε μια τέτοια εφαρμογή η οποία είναι άμεση, αφού τις συσκευές αυτές τις φέρουμε συνεχώς μαζί μας, καθώς και γρήγορη αφού με λίγες μόνο κινήσεις αποκτούμε πρόσβαση σε αυτή, αν έχει σχεδιαστεί κατάλληλα.

Ανάλυση εφαρμογής φοιτητή (Ajenda)

Συνεπώς γεννιούνται αυτόματα κάποιες απαιτήσεις που πρέπει να πληροί η εφαρμογή ώστε να ανταποκρίνεται στις ανάγκες του χρήστη. Μερικές απ' αυτές είναι οι παρακάτω:

- Πρόσβαση σε πληροφορίες που αφορούν Καθηγητές της σχολής. Αυτές κυρίως είναι στοιχεία επικοινωνίας, τοποθεσία γραφείου στο χάρτη, μαθήματα που είναι υπεύθυνος, διάφορες ανακοινώσεις του κ.α. Συνεπώς χρειαζόμαστε μια πλήρη καταγραφή με όλα αυτά τα στοιχεία.

- Πρόσβαση σε πληροφορίες που αφορούν Μαθήματα. Αυτές οι πληροφορίες είναι ο τομέας στον οποίο ανήκει, οι ώρες και οι αίθουσες διδασκαλίας, οι καθηγητές που διδάσκουν αυτό το μάθημα καθώς και ειδοποιήσεις που αφορούν το μάθημα αυτό. Συνεπώς, χρειαζόμαστε μια πλήρη καταγραφή με όλα αυτά τα στοιχεία.
- Πλήρης χαρτογράφηση της Σχολής ΗΜΜΥ. Η εφαρμογή αυτή πρέπει να είναι σε θέση να κατατοπίζει τον χρήστη για το πού βρίσκονται τα γραφεία των καθηγητών οπότε θα πρέπει να έχουν καταχωρηθεί τα αντίστοιχα Locations.
- Ειδοποίηση του χρήστη για την άφιξη του σε ένα προκαθορισμένο Geofence. Κάνοντας χρήση των υπηρεσιών τοποθεσίας, η εφαρμογή θα πρέπει να ενημερώνει το χρήστη με κατάλληλο μήνυμα, ανάλογα με την τοποθεσία του.
- Αλληλεπίδραση με online υπηρεσίες. Η εφαρμογή θα πρέπει να μπορεί να καθοδηγεί το χρήστη για την πλοήγησή του σε ένα επιθυμητό σημείο μέσω Google Maps.
- Να είναι εύχρηστη, κατανοητή και φιλική προς το χρήστη.

Η εφαρμογή που θα αναπτύξουμε στην παρούσα διπλωματική εργασία, για την πλευρά του φοιτητή, ονομάστηκε ***Ajenda***. Είναι μια εφαρμογή που λειτουργεί σαν ένας προσωπικός βοηθός, ένα εργαλείο που μπορεί να χρησιμοποιηθεί στο δρόμο ή και μέσα στη σχολή, παρέχοντας στο χρήστη τη δυνατότητα να ενημερωθεί για πληροφορίες που αφορούν τη σχολή του εύκολα και γρήγορα. Χρειάζεται, βέβαια η εφαρμογή αυτή να προσφέρει κάποιες παραπάνω λειτουργίες από μια απλή αναζήτηση μια πληροφορίας, οι οποίες να την κάνουν να ξεχωρίζει από άλλες αντίστοιχες.

Αρχικά, η δομή της πρόσβασης του φοιτητή στις πληροφορίες χωρίστηκε σε 4 κατηγορίες. Αυτές είναι **Καθηγητές**, **Μαθήματα**, **Τομείς** και **Gimbal**. Σε κάθε μια από αυτές ο χρήστης θα μπορεί να περιορίζει το εύρος των πληροφοριών στο οποίο έχει πρόσβαση καταλήγοντας έτσι γρήγορα στο επιθυμητό αποτέλεσμα. Βέβαια είναι προφανές ότι επειδή υπάρχει σύνδεση μεταξύ αυτών των 4 κατηγοριών θα πρέπει με κάποιο άμεσο τρόπο ο χρήστης να μπορεί να μεταβεί από τη μια στην άλλη, είτε μεμονωμένα είτε συνδυάζοντας δεδομένα, αναλόγως με το τι ζητάει.

- Όταν ο φοιτητής/χρήστης επιλέξει την κατηγορία **Καθηγητές**, θα πρέπει να έχει πρόσβαση σε μια λίστα με όλους του καθηγητές και να μπορεί είτε να επιλέξει είτε να κάνει αναζήτηση για κάποιον συγκεκριμένο. Αυτό θα έχει ως αποτέλεσμα την προβολή μιας φόρμας με τα στοιχεία του επιλεγμένου καθηγητή, δηλαδή, email, αριθμός γραφείου, τομέας στον οποίο ανήκει και όλα τα μαθήματα τα οποία διδάσκει.
- Αντίστοιχα, αν επιλέξει την κατηγορία **Μαθήματα**, θα πρέπει να όμοια να έχει πρόσβαση σε μια λίστα με όλα τα μαθήματα και να όταν βρει αυτό που θέλει να μεταβεί σε μια φόρμα που περιέχει πληροφορίες για αυτό το μάθημα. Αυτές οι πληροφορίες είναι ο τομές που ανήκει το μάθημα, οι ώρες και η αίθουσα διδασκαλίας του μαθήματος καθώς και μια λίστα με όλους τους καθηγητές που διδάσκουν το μάθημα.

- Μια επιπλέον λειτουργία που παρέχεται είναι η κατηγοριοποίηση **Καθηγητών και Μαθημάτων** με βάση έναν **Τομέα** της σχολής. Με την επιλογή κάποιου **Τομέα**, θα μπορεί ο χρήστης να ενημερώνεται για όλους του καθηγητές και όλα τα μαθήματα που ανήκουν σε κάποιο τομέα επιλέγοντας πάλι με τον ίδιο τρόπο ένα από αυτά όπως παραπάνω.
- Τέλος, θα πρέπει να έχει τη δυνατότητα ο χρήστης να εντοπίζει πάνω στο χάρτη σε ποιο σημείο βρίσκονται κάποιες τοποθεσίες που τον ενδιαφέρουν, όπως οι αίθουσες διδασκαλίας και τα γραφεία των καθηγητών. Αυτή η λειτουργία θα υπάρχει στην κατηγορία **Gimbal**.

Επιπρόσθετα, πέρα από τις στάνταρ πληροφορίες που θα μπορούσε να μάθει κάποιος φοιτητής για τους καθηγητές του ή τα μαθήματα του, θα πρέπει να είναι σε θέση να ενημερώνεται άμεσα για τυχόν αλλαγές και ενημερώσεις της τελευταίας στιγμής. Αυτές είναι η ακύρωση ενός μαθήματος που έγινε από τον καθηγητή ή η ανακοίνωση των βαθμών ενός μαθήματος μετά την εξεταστική ή μια ανακοίνωση που έβγαλε ο καθηγητής γενικού σκοπού και ενδιαφέροντος. Η λειτουργία αυτή θα βρίσκεται ενσωματωμένη στις λειτουργίες αναζήτησης είτε μαθήματος είτε καθηγητή.

Φυσικά, ο χρήστης θα πρέπει να ειδοποιείται άμεσα από την εφαρμογή αναλόγως με την τοποθεσία του για διάφορα θέματα που πιθανόν να τον αφορούν. Έτσι θα χρειαστεί κάποιο *service* το οποίο θα αντιλαμβάνεται το *location* του χρήστη, και σε συνδυασμό με αυτό θα έρχεται ειδοποίηση στο κινητό του, ενημερώνοντας τον για τον μέρος από το οποίο περνά και κάποιες πληροφορίες που αφορούν αυτό το μέρος. Θα χρειαστεί, λοιπόν, να καταχωρήσουμε ένα σύνολο από σημεία πάνω στο χάρτη (**hotspots**), τα οποία θα λειτουργούν σαν *trigger points* κάποιων λειτουργιών μεταφοράς δεδομένων προς το χρήστη.

Τέλος, θα πρέπει ο χρήστης να έχει τη δυνατότητα να μεταβεί άμεσα σε κάποιο γραφείο ενός καθηγητή, του οποίου την τοποθεσία δεν γνωρίζει. Επομένως, θα πρέπει να γίνει καταχώρηση της τοποθεσίας του γραφείου για κάθε καθηγητή και να παρέχεται στο χρήστη η δυνατότητα πλοήγησης από την τρέχουσα τοποθεσία που βρίσκεται, προς την επιθυμητή τοποθεσία λαμβάνοντας ακριβείς οδηγίες.

Αλληλεπίδραση με server

Για να ταιριάζει η εφαρμογή μας με τη σύγχρονη εποχή, δίνουμε τη δυνατότητα στον χρήστη να έχει άμεσα πρόσβαση σε πληροφορίες τις οποίες μπορεί να αντλεί από το internet. Η πρόσβαση στις πληροφορίες αυτές δεν θα είναι άμεση, αλλά έμμεση, χρησιμοποιώντας έναν **server**. Ο χρήστης για να χρησιμοποιήσει την εφαρμογή θα πρέπει να έχει πρόσβαση στο διαδίκτυο ώστε να μπορεί να επικοινωνήσει με τον server και να ανταλλάξει δεδομένα που θα χρειάζεται ο χρήστης. Τέτοιου είδους πληροφορίες, κάθε χρήστης εφαρμογής στον κόσμο, τις αντλεί μετά από αιτήματα (request) προς τον/τους servers και σαν απάντηση (response) λαμβάνει κάποια δεδομένα τα οποία μετά από επεξεργασία τα δημιουργεί τα επιθυμητά αποτελέσματα στο χρήστη. Οι servers είναι προγράμματα τα οποία μετά την εγκατάστασή τους σε κάποιο υπολογιστή είναι προσβάσιμα από όλους τους χρήστες του διαδικτύου μέσω ενός link (URL) αναφοράς στο οποίο γίνεται το request. Συνήθως ο server λαμβάνει δεδομένα από κάποια βάση δεδομένων ή από κάποιον άλλον server και στέλνει πίσω τα αποτελέσματα στο χρήστη για υλοποιήσουν το σκοπό τους. Στην περίπτωσή μας θα πρέπει να δημιουργήσουμε έναν server και μέσω αυτού η εφαρμογή προμηθεύεται με ότι χρειάζεται για την εξυπηρέτηση του χρήστη. Το κάθε request που γίνεται από τον χρήστη της εφαρμογής ή από την ίδια εφαρμογή, εκτελεί ένα κομμάτι κώδικα στον server το οποίο ονομάζεται servlet. Το κάθε servlet παίρνει δεδομένα από το request και επικοινωνεί με την βάση δεδομένων από όπου αντλεί τα δεδομένα που έχει αιτηθεί η εφαρμογή.

Ajenda

Αρχικά, με την ενεργοποίηση της εφαρμογής, αποστέλλεται ένα αίτημα προς τον server για την ανάκτηση όλων των *καθηγητών*, *των μαθημάτων και των τομέων*. Η εφαρμογή έχει πλέον διαθέσιμα στη μνήμη της μια λίστα για κάθε ένα από τα παραπάνω αντικείμενα. Με τον τρόπο αυτόν ο χρήστης θα έχει τη δυνατότητα να πλοηγηθεί και να αναζητήσει τις όποιες πληροφορίες θέλει. Κατά τη διάρκεια της χρήσης της εφαρμογής από τον φοιτητή θα πρέπει γίνονται αιτήματα προς τον server, τροφοδοτώντας την με όποιες πληροφορίες έχει επιλέξει ο χρήστης. Συνεπώς θα πρέπει να δημιουργηθούν διάφορα *servlets*, κάθε ένα από τα οποία να εξυπηρετεί και μια συγκεκριμένη μεταφορά δεδομένων προς το κινητό.

Σύμφωνα με την παραπάνω ανάλυση, τα servlets θα χρησιμοποιούνται κυρίως για άντληση πληροφοριών που θα αφορούν κάποιον καθηγητή ή κάποιο μάθημα ή κάποιον τομέα, αφού ο χρήστης μπορεί να επιλέξει κάτι από αυτά κατά την εκκίνηση. Επίσης θα χρησιμοποιούνται για πληροφορίες που χρειάζεται ο χρήστης όταν θα βρίσκεται σε κάποια συγκεκριμένα *locations*, ώστε να ενημερωθεί κατάλληλα για ό,τι πιθανόν τον ενδιαφέρει σχετικά με την περιοχή που βρίσκεται.

Συνοπτικά

Οι λειτουργίες που θα πρέπει να ενσωματώνει η εφαρμογή είναι

- Προβολή όλων των **Καθηγητών, Μαθημάτων, Τομέων και Hotspots**
- Αναζήτηση και προβολή πληροφοριών για κάθε ένα συγκεκριμένο στοιχείο από αυτά
- Δυνατότητα πλοήγησης με χρήση Χάρτη από την τρέχουσα τοποθεσία προς κάποιο σημείο ενδιαφέροντος
- Δυνατότητα προβολής όλων των Hotspots στο χάρτη και μετάβαση σε κάποιο από αυτά
- Δυνατότητα αυτόματης ενημέρωσης του χρήστη για ανακοινώσεις όταν βρίσκεται σε ένα Hotspot

Ανάλυση εφαρμογής καθηγητή (ProfessorApp)

Παρόμοια την πλευρά του φοιτητή, γεννιούνται ανάλογες απαιτήσεις που πρέπει να πληροί και η εφαρμογή για την ικανοποίηση των αναγκών του καθηγητή. Αυτές είναι οι εξής:

- Πρόσβαση του Καθηγητή στα μαθήματα τα οποία διδάσκει. Αν ο Καθηγητής μπορεί να ενημερώσει για διάφορες αλλαγές στα πλαίσια των μαθημάτων θα πρέπει να υπάρχει μια διαθέσιμη λίστα με τα μαθήματα που αφορούν τον ίδιο.
- Δυνατότητα ενημέρωσης για βαθμολογίες μαθημάτων, για ακυρώσεις μαθημάτων καθώς και δημοσίευση ανακοινώσεων γενικού θέματος σε μορφή κειμένου.
- Αλληλεπίδραση με online υπηρεσίες για άντληση και προώθηση δεδομένων που αφορούν τις κινήσεις του χρήστη κατά την πλοήγηση του στην εφαρμογή.
- Να είναι εύχρηστη, κατανοητή και φιλική προς το χρήστη.

Η εφαρμογή που θα αναπτύξουμε στην παρούσα διπλωματική εργασία, για την πλευρά του καθηγητή ονομάστηκε ProfessorApp, επιλέγοντας σαν παράδειγμα χρήσης τον κ. Βενιέρη. Η υπόθεση αυτή γίνεται χωρίς βλάβη της γενικότητας, καθώς η εφαρμογή λειτουργεί με τον ίδιο τρόπο για κάθε καθηγητή. Είναι μια εφαρμογή που λειτουργεί σαν ένας προσωπικός βοηθός, ένα εργαλείο που μπορεί να χρησιμοποιηθεί στο δρόμο ή και μέσα στη σχολή, παρέχοντας στο χρήστη να ενημερώσει τους φοιτητές για διάφορα θέματα εύκολα και γρήγορα. Με βάση τις απαιτήσεις που περιγράψαμε παραπάνω, η δομή της εφαρμογής θα είναι ως εξής.

Με την εκκίνηση της εφαρμογής, ο Καθηγητής/χρήστης θα μπορεί να επιλέξει ανάμεσα σε τρεις λειτουργίες. Αυτές οι λειτουργίες είναι η **Ακύρωση, Βαθμολογία και Ανακοίνωση** μαθημάτων. Σε κάθε μια από αυτές τις λειτουργίες ο χρήστης θα πρέπει να αλληλοεπιδρά με διαφορετικό τρόπο με την εφαρμογή. Αναλυτικότερα:

- Όταν ο χρήστης επιλέξει την **Ακύρωση** ενός μαθήματος, θα πρέπει να έχει πρόσβαση σε μια λίστα με όλα τα μαθήματα που διδάσκει. Αυτό θα έχει σαν αποτέλεσμα την προβολή μιας φόρμας επιλογής μαθημάτων από μια λίστα από αυτά τα οποία θέλει να ακυρώσει για την επόμενη διάλεξη.
- Όταν ο χρήστης επιλέξει την **Βαθμολογία** ενός μαθήματος, θα πρέπει να έχει πρόσβαση σε μια λίστα με όλα τα μαθήματα που διδάσκει. Αυτό θα έχει σαν αποτέλεσμα την προβολή μιας φόρμας επιλογής μαθημάτων από μια λίστα από αυτά τα οποία έχει βαθμολογήσει.
- Όταν ο χρήστης επιλέξει την **Ανακοίνωση**, τότε θα πρέπει να μεταβαίνει σε μια φόρμα συμπλήρωσης ενός κειμένου. Το κείμενο αυτό θα συμπληρώνεται και θα αποστέλλεται για την ενημέρωση των φοιτητών.

Αλληλεπίδραση με server

Στην περίπτωση του Καθηγητή/χρήστη, η αλληλεπίδραση με έναν server καθίσταται αναγκαία καθώς ο χρήστης και λαμβάνει δεδομένα από πηγές εκτός κινητού αλλά και αποστέλλει δεδομένα με σκοπό οι φοιτητές να έχουν πρόσβαση σε αυτά όποτε χρειαστούν. Στην περίπτωσή μας, λοιπόν, θα πρέπει να δημιουργηθούν κάποια servlets εκ των οποίων άλλα θα στέλνουν δεδομένα στο κινητό και άλλα θα λαμβάνουν από αυτό. Σύμφωνα με την παραπάνω ανάλυση, η αλληλεπίδραση με τον server εξασφαλίζει την πρόσβαση του Καθηγητή στα μαθήματα τα οποία διδάσκει, αλλά και την ενημέρωση για τις αλλαγές που έχει κάνει ή ανακοινώσεις που έχει δημοσιεύσει.

Συνοπτικά

Οι λειτουργίες που θα πρέπει να ενσωματώνει η εφαρμογή είναι

- Πρόσβαση στα μαθήματα του Καθηγητή/χρήστη
- Δυνατότητα Ακύρωσης ενός μαθήματος
- Δυνατότητα ενημέρωσης Βαθμολογίων ενός μαθήματος
- Δυνατότητα δημοσίευσης μιας Ανακοίνωσης από τον Καθηγητή.

4 Σχεδίαση

4.1 Η εφαρμογή Ajenda

Οι οθόνες της εφαρμογής

Για την εφαρμογή που σχεδιάζουμε προκειμένου να καλυφθούν οι λειτουργίες που έχουν περιγραφεί στο κεφάλαιο της ανάλυσης, θα πρέπει να υπάρξουν κάποιες οθόνες που να υλοποιούν οι παραπάνω λειτουργίες. Οι οθόνες αυτές είναι οι εξής:

- ***Main Activity***

Η εφαρμογή ξεκινάει με την εμφάνιση μια οθόνης (Activity) που περιέχει 4 tabs, **Καθηγητές – Μαθήματα – Τομείς – Gimbal**. Στο κάθε Tab εμφανίζεται μια λίστα με όλα τα εκάστοτε στοιχεία. Ο χρήστης μπορεί είτε να επιλέξει ένα στοιχείο από τη λίστα είτε να κάνει αναζήτηση κάποιο στοιχείο που θέλει στο εκάστοτε Tab, ανάλογα με το τι ψάχνει, και να επιλέξει αυτό που θέλει.

- ***ProfessorsList***

Στο συγκεκριμένο Tab, το οποίο θα είναι και το προεπιλεγμένο σαν αρχικό, θα εμφανίζεται μια λίστα με τα ονόματα όλων των καθηγητών της σχολής. Από τη λίστα αυτή, ο χρήστης είτε επιλέγει ένα όνομα είτε κάνει μια αναζήτηση και, αν και εφόσον υπάρχει το προς αναζήτηση όνομα, επιλέγει το όνομα που θέλει.

- ***SubjectsList***

Στο συγκεκριμένο Tab, έχουμε τη λίστα με τα μαθήματα όλης της σχολής, από τα οποία ο χρήστης μπορεί να επιλέξει όποιο θέλει είτε από τη λίστα είτε από αναζήτηση με τον ίδιο τρόπο όπως παραπάνω.

- ***SectionsList***

Στο συγκεκριμένο Tab, έχουμε πάλι μια λίστα με όλους τους Τομείς της σχολής, εκ των οποίων ο χρήστης επιλέγει με τον ίδιο τρόπο όπως παραπάνω.

- ***Gimbal***

  Στο συγκεκριμένο Tab, φαίνονται στον χάρτη όλα τα hotspots τα οποία είναι είτε Γραφεία Καθηγητών είτε Αίθουσες Διδασκαλίας.

- ***ProfessorInfo***

Το Activity αυτό εμφανίζεται όταν ο χρήστης έχει επιλέξει ένα στοιχείο από τη λίστα με τα ονόματα των καθηγητών. Σε αυτή την οθόνη θα φαίνονται πληροφορίες, όπως το email του καθηγητή, πληροφορίες γραφείου και ο τομέας που ανήκει. Επίσης, ο χρήστης θα μπορεί να ενημερώνεται για κάποια ακύρωση μαθήματος, για ανακοίνωση βαθμολογιών ενός μαθήματος και για διάφορες ανακοινώσεις του επιλεγμένου καθηγητή.

- ***ProfessorSubjects***

Στην οθόνη αυτή εμφανίζονται σε μια λίστα, όλα τα μαθήματα τα οποία διδάσκει ο καθηγητής που έχει επιλεγεί από το χρήστη.

- ***SubjectInfo***

Το Activity αυτό εμφανίζεται όταν ο χρήστης έχει επιλέξει ένα στοιχείο από τη λίστα με τα ονόματα των μαθημάτων. Σε αυτή την οθόνη θα φαίνονται πληροφορίες, όπως ο τομέας που ανήκει το μάθημα, η αίθουσα και οι ώρες που γίνεται η διδασκαλία. Επίσης, ο χρήστης θα μπορεί να ενημερώνεται για αν το επερχόμενο μάθημα έχει ακυρωθεί καθώς και για το αν έχουν ανακοινωθεί βαθμολογίες για το εν λόγω μάθημα.

- ***SubjectProfessors***

Στην οθόνη αυτή εμφανίζονται σε μια λίστα, όλοι οι καθηγητές οι οποίοι έχουν αναλάβει τη διδασκαλία ενός συγκεκριμένου μαθήματος που έχει επιλέξει ο χρήστης.

- ***SectionInfo***

Το Activity αυτό εμφανίζεται όταν ο χρήστης έχει επιλέξει ένα στοιχείο από τη λίστα με τους τομείς που έχει η σχολή. Σε αυτή την οθόνη θα υπάρχουν δυο λίστες και ο χρήστης θα έχει τη δυνατότητα να επιλέξει ανάμεσα σε μια από τις δυο. Η μια λίστα περιέχει τους καθηγητές που ανήκουν στον επιλεγμένο τομέα και η άλλη περιέχει τα μαθήματα που ανήκουν στον επιλεγμένο τομέα. Σε κάθε περίπτωση, ο χρήστης μεταβαίνει σε ανάλογη οθόνη που έχουμε αναφέρει παραπάνω ανάλογα με το αν η επιλογή είναι μάθημα ή καθηγητής.

4.2 Η εφαρμογή ProfessorApp

Οι οθόνες της εφαρμογής

Για την εφαρμογή που σχεδιάζουμε προκειμένου να καλυφθούν οι λειτουργίες που έχουν περιγραφεί στο κεφάλαιο της ανάλυσης, θα πρέπει να υπάρξουν κάποιες οθόνες που να υλοποιούν οι παραπάνω λειτουργίες. Οι οθόνες αυτές είναι οι εξής:

- ***Main Activity***

Η εφαρμογή ξεκινάει με την εμφάνιση μιας οθόνης (Activity), που περιέχει τρεις επιλογές για τον χρήστη Καθηγητή. Με τις τρεις αυτές επιλογές έχει τη δυνατότητα να Ακυρώσει κάποιο μάθημα, να Ενημερώσει για τη βαθμολογία κάποιου μαθήματος καθώς και να βγάλει μια γενική Ανακοίνωση προς όλους τους χρήστες της εφαρμογής Ajenda.

- ***Cancel Activity***

Σε αυτό το Activity, θα εμφανίζεται στον χρήστη μια λίστα με όλα τα μαθήματα τα οποία διδάσκει ο καθηγητής (Βενιέρης) και τα οποία δεν έχει ακυρώσει την τρέχουσα ημέρα. Θα έχει τη δυνατότητα να επιλέξει ένα η περισσότερα από αυτά τα μαθήματα και να ενημερώσει τη Βάση Δεδομένων στο κατάλληλο πεδίο για την εγκυρότητα αυτού του μαθήματος.

- ***Grade Activity***

Σε αυτό το Activity, θα εμφανίζεται πάλι μια λίστα στον χρήστη με όλα τα μαθήματα που διδάσκει και θα μπορεί να επιλέγει ένα η περισσότερα από αυτά τα οποία θα ενημερώνονται στη Βάση Δεδομένων ότι έχουν πλέον βαθμολογηθεί και είναι διαθέσιμα προς το φοιτητή.

- ***Announce Activity***

Σε αυτό το Activity, θα εμφανίζεται στον χρήστη ένα πεδίο στο οποίο θα μπορεί να γράψει μια σύντομη ανακοίνωση η οποία θα είναι διαθέσιμη για όλο το φοιτητικό κοινό που χρησιμοποιεί την εφαρμογή φοιτητή Ajenda. Η ανακοίνωση μπορεί να ενημερώνεται κάθε φορά που ο καθηγητής (κ. Βενιέρης), προωθεί μια καινούργια, υπερκαλύπτοντας την προηγούμενη.

4.3 *Αλληλεπίδραση με τον Server*

Για να ταιριάζει η εφαρμογή μας με τη σύγχρονη εποχή, δίνουμε τη δυνατότητα στον χρήστη να έχει άμεσα πρόσβαση σε πληροφορίες τις οποίες μπορεί να αντλεί από το internet. Αυτές τις πληροφορίες, κάθε χρήστης εφαρμογής στον κόσμο, τις αντλεί μετά από αιτήματα (request) προς τον/τους servers και σαν απάντηση (response) λαμβάνει κάποια δεδομένα τα οποία μετά από επεξεργασία τα δημιουργεί τα επιθυμητά αποτελέσματα στο χρήστη. Οι servers είναι προγράμματα τα οποία μετά την εγκατάστασή τους σε κάποιο υπολογιστή είναι προσβάσιμα από όλους τους χρήστες του διαδικτύου μέσω ενός link (URL) αναφοράς στο οποίο γίνεται το request. Συνήθως ο server λαμβάνει δεδομένα από κάποια βάση δεδομένων ή από κάποιον άλλον server και στέλνει πίσω τα αποτελέσματα στο χρήστη για υλοποιήσουν το σκοπό τους. Στην περίπτωσή μας έχουμε δημιουργήσει ένα server και μέσω αυτού η εφαρμογή προμηθεύεται με ότι χρειάζεται για την εξυπηρέτηση του χρήστη. Το κάθε request που γίνεται από τον χρήστη της εφαρμογής ή από την ίδια εφαρμογή, εκτελεί ένα κομμάτι κώδικα στον server το οποίο ονομάζεται servlet. Το κάθε servlet παίρνει δεδομένα από το request και επικοινωνεί με την βάση δεδομένων από όπου αντλεί τα δεδομένα που έχει αιτηθεί η εφαρμογή. Τα servlets που χρησιμοποιούνται για τα requests είναι τα ακόλουθα.

- *FindAll*
  Αυτό το servlet τρέχει κάθε φορά που τρέχει η εφαρμογή. Καλείται και από τα 3 Tabs με διαφορετικό όρισμα από το καθένα. Το Professor Tab το καλεί με όρισμα “full_name”, το Subject Tab με όρισμα “courses” και το Section Tab με όρισμα “section”. Όταν γίνει η κλήση του κώδικα, ο server κάνει ένα query στη βάση δεδομένων παίρνοντας πίσω όλα τα δεδομένα που χρειάζεται σε μια λίστα που περιέχει όλα τα στοιχεία αντίστοιχα με το όρισμα. Έπειτα στέλνει πίσω στο κινητό το response και εκεί το κάθε Tab εμφανίζει τη λίστα με τα στοιχεία που της αντιστοιχούν.

- *FindBoth*
  Αυτό το servlet καλείται από το server κάθε φορά που ο χρήστης έχει επιλέξει από το SectionList Tab έναν τομέα. Ο server συνδέεται με τη βάση δεδομένων και παίρνει πίσω όλους τους *professor*s και όλα τα μαθήματα που έχουν σχέση με τον επιλεγμένο τομέα. Έπειτα επιστρέφει τα αποτελέσματα με το response στην εφαρμογή και ο χρήστης έχει πλέον πρόσβαση σ' αυτά.

- *FindCoursesProfessor*
  Αυτό το servlet τρέχει από το server κάθε φορά που ο χρήστης επιλέγει να δει ποια *courses* έχει υπό την διδασκαλία του ο επιλεγμένος *professor*. Με βάση το ID του *professor* στη βάση δεδομένων γίνετε το αντίστοιχο query το οποίο επιστρέφει στον server τη λίστα με τα μαθήματα που διδάσκει ο *professor*. Έπειτα η εφαρμογή λαμβάνει το response το οποίο περιέχει αυτή τη λίστα η οποία πλέον είναι έτοιμη να αποτυπωθεί στην οθόνη και να είναι προσβάσιμη από το χρήστη.

- *FindProfessorsCourse*
  Αυτό το servlet τρέχει καλείται από τον server σε αντίστοιχη περίπτωση με την προηγούμενη αλλά για ένα συγκεκριμένο *course*. Όταν ο χρήστης έχει επιλέξει δηλαδή να δει ποιοι *professors* έχουν υπό τη διδασκαλία τους ένα μάθημα, ο server με βάση το ID του μαθήματος κάνει μια αναζήτηση στη βάση δεδομένων και παίρνει όλους τους *professors*. Έπειτα ο server στέλνει το response στην εφαρμογή και εκεί αποτυπώνεται στην οθόνη η λίστα με τα επιθυμητά ονόματα.

- *FindCourseTimeClass*
  Αυτό το servlet καλείται από το server όταν ο χρήστης επιλέξει ένα *course* από την αρχική λίστα που του εμφανίζεται στην αρχική οθόνη της εφαρμογής. Ο server συνδέεται με τη βάση δεδομένων και με βάση το ID του επιλεγμένου μαθήματος κάνει JOINT τα κατάλληλα TABLES για να επιστρέψει στην εφαρμογή το response που περιέχει τη λίστα με τα επιθυμητά αποτελέσματα. Το response αυτό έχει τα *ονόματα όλων των professor που κάνουν το μάθημα* καθώς και πληροφορίες όπως τον Τομέα που ανήκει το μάθημα, την αίθουσα στην οποία γίνεται η διδασκαλία καθώς επίσης και τις μέρες και ώρες του ωρολόγιου προγράμματος της εβδομάδας για το συγκεκριμένο μάθημα. Τα παραπάνω είναι διαθέσιμα στο χρήστη αμέσως μόλις η εφαρμογή δεχτεί το response.

- *FindSectionOffice*
  Αυτό το servlet καλείται από τον server όταν ο χρήστης έχει επιλέξει στην αρχική οθόνη της εφαρμογής κάποιον *professor* και πρόκειται να ενημερωθεί σχετικά με πληροφορίες για αυτόν. Ο server με βάση το ID του *professor* κάνει ένα query και με τα κατάλληλα JOINT σε διάφορα TABLES επιστρέφει μια λίστα στην εφαρμογή. Το response αυτό περιέχει όλες τις απαραίτητες πληροφορίες για το χρήστη τις οποίες πρόκειται να δει στην αντίστοιχη οθόνη.

4.4 ***Αλληλεπίδραση με την Βάση Δεδομένων***

Η προχωρημένη πλέον τεχνολογία έχει εφαρμογή σε όλες τις πτυχές της ζωής μας με σκοπό να μας εξυπηρετεί δίνοντας μας πληροφορίες ανά πάσα στιγμή για οποιουδήποτε είδους θέμα επιθυμούμε. Για το λόγο αυτό, οι εν λόγω πληροφορίες πρέπει να είναι ασφαλείς, αποθηκευμένες, αξιόπιστες, άμεσα προσβάσιμες, ανανεώσιμες και εύκολα διαχειρίσιμες. Αυτός είναι και ο λόγος που η πλειοψηφία των εφαρμογών οι οποίες έχουν ως στόχο να δίνουν πληροφορίες στο χρήστη για οτιδήποτε ζητήσει, πρέπει να έχουν πρόσβαση σε κάποιου είδους βάση δεδομένων από όπου ανακτούν ή αποθηκεύουν πληροφορίες. Στην εφαρμογή μας προκύπτει επιτακτική ανάγκη να υπάρχει μια τέτοια υλοποίηση μιας βάσης δεδομένων για να μπορεί να γίνετε η συλλογή των πληροφοριών που θέλει εν τέλει να πάρει ο χρήστης. Η βάση δεδομένων μας έχει 8 πίνακες (Tables) στους οποίους έχουμε αποθηκευμένα δεδομένα που χρειάζεται η εφαρμογή για να είναι λειτουργική. Οι πίνακες είναι οι εξής :

1. ***Professor***
   Ο πίνακας *Professor* αντιπροσωπεύει τον κάθε καθηγητή και έχει τα παρακάτω columns :
   - ProfessorID -> Το ID κάθε καινούργιας καταχώρησης ενός καθηγητή. Λειτουργεί σαν PK (PrimaryKey).
   - full_name -> Το όνομα κάθε καθηγητή.
   - Email -> Το email κάθε καθηγητή.
   - sectionID -> Το ID από τον τομέα στον οποίο ανήκει ο κάθε καθηγητής.
   - officeID -> Το ID από το γραφείο που βρίσκεται ο κάθε καθηγητής.
   - announce -> Τιμή που δείχνει αν ο καθηγητής έχει βγάλει ανακοίνωση.
   - announceText-> Η ανακοίνωση που δημοσίευσε ο καθηγητής.

2. ***Courses***
   Ο πίνακας *Courses* αντιπροσωπεύει το κάθε μάθημα και έχει τα παρακάτω columns:
   - courseID -> Το ID κάθε καινούργιας καταχώρησης ενός μαθήματος. Λειτουργεί σαν PK (PrimaryKey).
   - course_name -> Το όνομα/τίτλος κάθε μαθήματος.
   - roi -> Η ροή στην οποία ανήκει το κάθε μάθημα.
   - sectionID -> Το ID του τομέα στον οποίο ανήκει το κάθε μάθημα.
   - classRoomID -> Το ID της αίθουσας που διδάσκεται το κάθε μάθημα.
   - valid -> Τιμή που δείχνει αν έχει ακυρωθεί το μάθημα.
   - grades -> Τιμή που δείχνει αν έχουν βγει βαθμολογίες για το μάθημα.

3. ***Section***

Ο πίνακας *Section* αντιπροσωπεύει τον κάθε τομέα και έχει τα παρακάτω columns :

- sectionID -> Το ID κάθε καινούργιας καταχώρησης ενός τομέα. Λειτουργεί σαν PK (PrimaryKey).
- section_name -> Το όνομα/τίτλος κάθε τομέα.

4. ***Office***

Ο πίνακας *Officeι* αντιπροσωπεύει το γραφείο του κάθε καθηγητή και έχει τα παρακάτω columns :

- officeID -> Το ID κάθε καινούργιας καταχώρησης ενός γραφείου. Λειτουργεί σαν PK (PrimaryKey).
- phone -> Το τηλέφωνο κάθε γραφείου.
- number -> Το νούμερο κάθε γραφείου.
- floor -> Ο όροφος που βρίσκεται κάθε γραφείο.
- building -> Το κτήριο στο οποίο στεγάζεται κάθε γραφείο.
- longtitude -> Οι γεωγραφικές συντεταγμένες του γραφείου.
- latitude -> » » »

5. ***Classroom***

Ο πίνακας *Classroom* αντιπροσωπεύει την κάθε αίθουσα που γίνονται τα μαθήματα και έχει τα παρακάτω columns :

- classroomID -> Το ID κάθε καινούργιας καταχώρησης μιας αίθουσας διδασκαλίας. Λειτουργεί σαν PK (PrimaryKey).
- building -> Το κτήριο στο οποίο γίνεται η διδασκαλία.
- number -> Το νούμερο της αίθουσας διδασκαλίας.
- longtitude -> Οι γεωγραφικές συντεταγμένες της αίθουσας.
- latitude -> » » »

6. ***Time***

Ο πίνακας *Time* αντιπροσωπεύει την ώρα που γίνεται κάποιο μάθημα μέσα στη βδομάδα και έχει τα παρακάτω columns :

- timeID -> Το ID κάθε καινούργιας καταχώρησης μιας ώρας μαθήματος. Λειτουργεί σαν PK (PrimaryKey).
- day -> Η μέρα διδασκαλίας ενός μαθήματος.
- start_hour -> Η ώρα έναρξης της διδασκαλίας.
- end_hour -> Η ώρα λήξης της διδασκαλίας.

7. ***Hotspot***

Ο πίνακας Hotspot αντιπροσωπεύει το σημείο πάνω στο χάρτη που έχει τοποθετηθεί ένα beacon ή ένα Geofence και έχει τα παρακάτω columns :

- hotspotID -> Το ID κάθε καινούργιας καταχώρησης μιας τοποθεσίας. Λειτουργεί σαν PK (Primary Key).
- spotname -> Η ονομασία της κάθε τοποθεσίας.
- gimbalID -> Το ID που είναι καταχωρημένο στην πλατφόρμα της Gimbal.
- type -> Ο τύπος της τοποθεσίας που έχει καταχωρηθεί.
- longtitude -> Οι γεωγραφικές συντεταγμένες της τοποθεσίας.
- latitude -> » » »

8. ***CoursesTime***

Ο πίνακας *CoursesTime* αντιπροσωπεύει τη σχέση μεταξύ των πινάκων Courses και Time. Με τη χρήση του μπορούμε να βρούμε για κάθε εγγραφή στον πίνακα Courses, ποια ή ποιες εγγραφές του πίνακα Time της αντιστοιχεί και αντίστροφα. Ο πίνακας μας έχει τα παρακάτω columns :

- coursesTimeID -> Το ID κάθε καινούργιας καταχώρησης για αυτόν τον πίνακα. Λειτουργεί σαν PK (PrimaryKey).
- courseID -> Το ID του μαθήματος.
- timeID -> Το ID της ώρας διδασκαλίας.

9. ***ProfessorCourses***

Ο πίνακας *ProfessorCourses* αντιπροσωπεύει τη σχέση μεταξύ των πινάκων Professor και Course. Με τη χρήση του μπορούμε να βρούμε για κάθε εγγραφή στον πίνακα Professor ποια ή ποιες εγγραφές στον πίνακα Course της αντιστοιχεί και αντίστροφα. Ο πίνακας μας έχει τα παρακάτω columns :

- professorCoursesID -> Το ID κάθε καινούργιας καταχώρησης για αυτόν τον πίνακα. Λειτουργεί σαν PK (PrimaryKey).
- professorID -> Το ID του καθηγητή.
- courseID -> Το ID του μαθήματος.

5 Υλοποίηση

Έπειτα από τον σχεδιασμό και την ανάλυση των βασικών επιμέρους κομματιών που χρειάζεται η εφαρμογή μας παραθέτουμε τον κώδικα από τις βασικές λειτουργίες της εφαρμογής και του server.

5.1 Η εφαρμογή Ajenda

Υλοποιώντας την εφαρμογή με βάση την ανάλυση που κάναμε στα προηγούμενα κεφάλαια, δημιουργήθηκαν κατά κύριο λόγο 3 packages και μια Main Class. Παρακάτω παραθέτουμε τις βασικές λειτουργίες από την κάθε κλάση μαζί με κομμάτια κώδικα από το Android Studio.

Main Activity

Με τον όρο Activity, εννοούμε μια ξεχωριστή οθόνη την οποία βλέπει ο χρήστης. Όπως και σε κάθε Activity, έχουμε την κλάση *onCreate()* η οποία τρέχει κάθε φορά που το σύστημα του Android χρειάζεται να δημιουργήσει ένα καινούργιο Activity. Όταν τρέχει αυτή η μέθοδος δημιουργείται το View το οποίο θα εμφανιστεί στην οθόνη του χρήστη και επιτελούνται όλες οι άλλες λειτουργίες που έχει προγραμματίσει ο χρήστης. Στην περίπτωσή μας, δημιουργούμε έναν ViewPager και έναν SectionPagerAdapter. Ο ViewPager είναι ένα container στο οποίο μπορεί κανείς να φορτώσει δυναμικά διάφορα fragments, δηλαδή αυτόνομα Views τα οποία μπορούν να φορτωθούν μέσα σε άλλα Views όποτε θέλει ο χρήστης. Ο SectionPagerAdapter είναι ένας μηχανισμός ο οποίος αντιστοιχεί το κάθε fragment στο αντίστοιχο Tab του ViewPager. Ξεκινώντας, λοιπόν, ο SectionPagerAdapter αντιστοιχεί τα **ProfessorTab**, **SubjectTab**, **SectionTab** και **GimbalTab** με τη σειρά που αναφέραμε σε παραπάνω κεφάλαιο. Η υλοποίησή του φαίνεται παρακάτω.

```
public class SectionsPagerAdapter extends FragmentPagerAdapter {

    public SectionsPagerAdapter(FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int position) {

        Fragment fragment = null;
        if (position == 0) {
            fragment = new ProfessorsList();
        }
        if (position == 1) {
            fragment = new SubjectList();
        }
        if (position == 2) {
            fragment = new SectionList();
        }
        if (position == 3) {
            fragment = new GimbalFragment();
        }
        return fragment;
    }

    @Override
    public int getCount() {
        // Show 3 total pages.
        return 4;
    }

    @Override
    public CharSequence getPageTitle(int position) {
        switch (position) {
            case 0:
                return "Professors";
            case 1:
                return "Subjects";
            case 2:
                return "Sections";
            case 3:
                return "Gimbal";
        }
        return null;
    }
}
```

onCreateOptionsMenu

Όπως αναφέραμε και στο κομμάτι της ανάλυσης, ο χρήστης έχει τη δυνατότητα να κάνει search για να βρει τον Professor ή το Course που επιθυμεί από την ήδη υπάρχουσα λίστα. Η υλοποίηση της λειτουργίας αυτής γίνεται με την δημιουργία ενός **SearchView**. Σ' αυτό το View ενσωματώνουμε τη λειτουργία φιλτραρίσματος του προς αναζήτηση κειμένου πάνω στη λίστα που υπάρχει στο αντίστοιχο Tab.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    searchItem = menu.findItem(R.id.menu_search);


    searchView = (SearchView) searchItem.getActionView();
    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) {
            return false;
        }

        @Override
        public boolean onQueryTextChange(String newText) {
            adapter.clear();
            adapter.getFilter().filter(newText);
            return true;
        }
    });

    return true;
}
```

DataFiltering

Η διαδικασία του Filtering γίνεται με την υλοποίηση μιας κλάσης *DataFiltering* η οποία δέχεται σαν ορίσματα την λίστα πάνω στην οποία θα γίνει το filter καθώς και το string το οποίο έχει πληκτρολογηθεί από τη χρήστη. Τα αποτελέσματα επιστρέφουν και εμφανίζονται στην οθόνη πάνω στην αρχική λίστα από την οποία πλέον ο χρήστης μπορεί να επιλέξει το στοιχείο που τον ενδιαφέρει. Στην όλη διαδικασία μεσολαβεί ένας ListViewAdapter, μια κλάση η οποία κάνει adapt μια λίστα πάνω στο ListView του κάθε Tab.

```
private class DataFilter extends Filter {

    @Override
    protected FilterResults performFiltering(CharSequence constraint) {
        FilterResults filterResults = new FilterResults();
        constraint = constraint.toString().toLowerCase();
        if (constraint != null && constraint.toString().length() > 0) {

            List<String> founded = new ArrayList<String>();
            for (String item : searchList) {
                if (item.toString().toLowerCase().contains(constraint)) {
                    founded.add(item);
                }
            }
            filterResults.values = founded;
            filterResults.count = founded.size();
        } else {
            synchronized (this) {
                filterResults.values = searchList;
                filterResults.count = searchList.size();
            }
        }
        return filterResults;
    }

    @Override
    protected void publishResults(CharSequence constraint, FilterResults
filterResults) {
        showUpList = (ArrayList<String>) filterResults.values;
        notifyDataSetChanged();
        clear();
        for (int i = 0; i < showUpList.size(); i++) {
            add(showUpList.get(i));
        }
        notifyDataSetInvalidated();
    }
}
```

Μετά την εκκίνηση της εφαρμογής αφότου τρέξει η *onCreate()* από τη **MainActivity** class καλούνται και δημιουργούνται τα **Fragments**. Αξίζει να σημειωθεί ότι η *onCreate()* μέθοδος από κάθε fragment τρέχει από το σύστημα χωρίς να πρόκειται να εμφανιστεί άμεσα στο χρήστη. Αυτό συμβαίνει για λόγους performance, έτσι ώστε σε ενδεχόμενο κάλεσμα του fragment από το χρήστη, το σύστημα θα είναι έτοιμο να αποκριθεί αμέσως εφόσον η onCreate() του fragment έχει ήδη τρέξει και έχει γίνει cache στο σύστημα.

Το project περιέχει 3 άλλα packages. Το **Fragment Package**, το **Model Package** και το **Utilities Package**.

Στο Fragments Package περιέχονται τα 4 επιμέρους packages με τις κλάσεις από τα Activities που αναφέρθηκαν στο κεφάλαιο 3. Ας τα δούμε ένα-ένα :

- # ProfessorTab Fragment Package

ProfessorList

Στο Package αυτό, βρίσκεται η κλάση ProfessorList η οποία καλείται από τον SectionsAdapterPager για να φορτωθεί το fragment που αντιστοιχεί στο ProfessorTab. Όπως και στα υπόλοιπα fragments, κατά τη δημιουργία τους τρέχουν η onCreate() και η onCreateView() methods. Στην πρώτη μέθοδο επιτελούνται διαδικασίες όπως επικοινωνία με το server και άλλα, ενώ η δεύτερη μέθοδος είναι υπεύθυνη για τη δημιουργία όλων των αντικειμένων που αφορούν το UI της εφαρμογής για το συγκεκριμένο fragment. Η μόνη λειτουργία που εκτελεί η onCreate() είναι ένα request που γίνεται προς το server για την ανάκτηση της λίστας όλων των *professors.*

```
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Make the Professors Request onCreate the Fragment //
    String string = "full_name";
    downloadProfessors(string);
}
```

Το request με όρισμα *fullname* παίρνει σαν response ένα JSONArray με όλα τα DTOs από τον πίνακα Professors της βάσης δεδομένων. Έπειτα δημιουργεί μια λίστα που περιέχει μόνο τα ονόματα από κάθε DTO. Για την υλοποίηση του κάθε request υπάρχει μια ξεχωριστή κλάση η οποία θα αναλυθεί παρακάτω.

```
private void downloadProfessors(String data) {

    requests.getSearchProfList(data, new
Requests.VolleyCallbackSearchProfList() {
                @Override
                public void onSuccess(JSONArray jsonArray) {

                    for (int i = 0; i < jsonArray.length(); i++) {
                        try {
                            ProfessorDto professorDto = new
ProfessorDto(jsonArray.getJSONObject(i));
                            professorDtoList.add(professorDto);
                            full_nameList.add(professorDto.getFull_name());
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }

                    if (activity.getCurrentTab() == 0)
                        setUpAdapter(full_nameList, professorDtoList);
                }
            }
    );
}
```

Τέλος η τελευταία μέθοδος που καλείται μέσα στο fragment αυτό είναι η setUpAdapter, η οποία παίρνει ως όρισμα δυο λίστες. Μια λίστα μόνο με τα ονόματα των professors, η οποία είναι και η λίστα η οποία γίνεται adapt πάνω στο ListView μέσω του ListViewAdapter που αναφέραμε παραπάνω. Αυτή η λίστα είναι η λίστα που εμφανίζεται στο χρήστη στο παρών Tab και του δίνει τη δυνατότητα να επιλέξει ή να αναζητήσει πάνω σ' αυτήν προχωρώντας στης πληροφορίες του επιλεγμένου professor.

```
private void setUpAdapter(List list, List<ProfessorDto> professorDtoList) {

    activity.setAdapter(new ListViewAdapter(activity, R.layout.item_listview,
list));
    professor_listView.setAdapter(activity.getAdapter());

    final List<ProfessorDto> myDto = professorDtoList;

    professor_listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {

            String selection = (String) parent.getItemAtPosition(position);
            for (int i = 0; i < myDto.size(); i++) {
                if (myDto.get(i).getFull_name().equals(selection)) {
                    myProfessorDto = myDto.get(i);
                    break;
                }
            }

            Intent intent = new Intent(activity, ProfessorInfo.class);
            Bundle bundle = new Bundle();
            bundle.putSerializable("myProfessorDto", myProfessorDto);
            intent.putExtras(bundle);
            getActivity().startActivityForResult(intent, PICK_VIEW_REQUEST);
        }
    });
}
```

Σε περίπτωση που ο χρήστης επιλέξει έναν καθηγητή, στο *professor_listView* είναι ενεργοποιημένος ένας Listener οποίος ενεργοποιεί ένα Intent. Η κλάση Intent είναι μια κλάση του συστήματος του Android, με την οποία γίνεται triggering η αλλαγή μιας τρέχουσας οθόνης και εμφάνισης μιας καινούργιας στο χρήστη μεταφέροντας δεδομένα από τη μια στη άλλη. Στην περίπτωσή μας, μεταφέρουμε όλο το DTO του επιλεγμένου από το χρήστη *professor,* σε μορφή Bundle. Η κλάση Bundle είναι μια κλάση με την οποία γίνεται mapping ένα String Key με ένα πακέτο δεδομένων.

ProfessorInfo

Όπως αναφέρθηκε στο *3 κεφάλαιο*, αυτό το Activity δείχνει πληροφορίες για κάποιον professor. Οι ενδιαφέρουσες μέθοδοι που πρέπει να δούμε εδώ είναι οι εξής.

Όταν φορτώνει αυτή η οθόνη εκτός από αντικείμενα που αφορούν το γραφικό περιβάλλον της εφαρμογής, γίνεται αυτόματα μια επικοινωνία με το server για την άντληση πληροφοριών που πρέπει να δοθούν στο χρήστη. Οι πληροφορίες αφορούν τα Office και Section Tables από τη βάση δεδομένων. Τα δυο αυτά στοιχεία έρχονται σε δυο ξεχωριστά JSONObjects.

```
private void downloadInfos(int professorID)     {
    requests.getProfessorSectionOffice(professorID, new
Requests.VolleyCallbackSectionOffice() {
        @Override
        public void onSuccess(JSONArray jsonArray) {
            try {
                JSONObject officeJson = jsonArray.getJSONObject(0);
                JSONObject sectionJson = jsonArray.getJSONObject(1);
                setUpTabs(officeJson, sectionJson);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });
}
```

Σ' αυτό το Activity δίνεται η δυνατότητα στο χρήστη να πλοηγηθεί μέσω GoogleMaps προς το γραφείο του επιλεγμένου *professor* μέσω ενός Button. Ξεκινώντας το Activity έχει δρομολογηθεί από την εφαρμογή η σύνδεση με το GoogleMaps API, υπηρεσία η οποία παρέχεται από την Google και μας δίνει τη δυνατότητα να πάρουμε το τρέχων Location μας σε μορφή συντεταγμένων LatLong (latitude, longitude).

```
if (mGoogleApiClient == null) {
    mGoogleApiClient = new
GoogleApiClient.Builder(AjendaContext.getContext())
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(API)
            .build();
}
```

Με το πάτημα του *professor_map* Button, γίνεται εκκίνηση μέσω της κλάσης Intent η οποία μας μεταφέρει στην ενσωματωμένη εφαρμογή του κινητού **Google Maps**, μεταφέροντας μαζί το Location του χρήστη και το Location του *professor.* Ο κώδικας που πραγματοποιεί

αυτή τη λειτουργία φαίνεται παρακάτω

```
professor_map.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        LocationRequest locationRequest = LocationRequest.create();
        locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        locationRequest.setInterval(30 * 1000);
        locationRequest.setFastestInterval(5 * 1000);
        LocationSettingsRequest.Builder builder = new
LocationSettingsRequest.Builder()
                .addLocationRequest(locationRequest);

        builder.setAlwaysShow(true);

        PendingResult<LocationSettingsResult> result =

LocationServices.SettingsApi.checkLocationSettings(mGoogleApiClient,
builder.build());
        result.setResultCallback(new ResultCallback<LocationSettingsResult>()
{
            @Override
            public void onResult(LocationSettingsResult result) {
                final Status status = result.getStatus();
                final LocationSettingsStates state =
result.getLocationSettingsStates();
                switch (status.getStatusCode()) {

                    case LocationSettingsStatusCodes.SUCCESS:
                        try {
                            mLastLocation = getmLastLocation();

                            if (mLastLocation == null)
                                Toast.makeText(ProfessorInfo.this, "Αναμονή
τοποθεσίας..Δοκιμάστε ξανά", Toast.LENGTH_SHORT).show();
                            else {
                                OfficeDto officeDto = new
OfficeDto(officeJson);
                                String uri = String.format(Locale.ENGLISH,
"http://maps.google.com/maps?saddr=%f,%f(%s)&daddr=%f,%f (%s)",
mLastLocation.getLatitude(), mLastLocation.getLongitude(), "Είστε εδώ!",
officeDto.getLatitude(), officeDto.getLongitude(), "Γραφείο Καθηγητή ");
                                Intent intent = new
Intent(Intent.ACTION_VIEW, Uri.parse(uri));

intent.setClassName("com.google.android.apps.maps",
"com.google.android.maps.MapsActivity");
                                startActivity(intent);
                            }
                        } catch (JSONException e) {
                            e.printStackTrace();
```

```
break;

                    case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:

                        try {
                            status.startResolutionForResult(
                                    ProfessorInfo.this, REQUEST_LOCATION);
                        } catch (IntentSender.SendIntentException e) {
                        }
                        break;
                    case
LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:
                        Toast.makeText(ProfessorInfo.this, "Unavailable
Location!", Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        });
    }
});
```

Μια άλλη κλάση που χρησιμοποιείται είναι η *onConnected()*, η οποία χρησιμεύει για να πάρει η εφαρμογή το τρέχων Location του χρήστη κάνοντας πρώτα τον κατάλληλο έλεγχο με τον οποίο επιβεβαιώνεται ότι ο χρήστης έχει δικαιώματα παροχής *Location_Services* από το κινητό.

```
public void onConnected(@Nullable Bundle bundle) {
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;
    }
    mLastLocation = FusedLocationApi.getLastLocation(
            mGoogleApiClient);
}
```

Τέλος, υπάρχει και η μέθοδος setUpTabs(), η οποία καλείται όταν έχει επιστρέψει το response από το αρχικό request του Activity στο server. Η μέθοδος αυτή εξυπηρετεί τις λειτουργίες τριών Buttons τα οποία εμφανίζουν τις πληροφορίες του *professor*, τα οποία είναι emailBtn, sectionBtn και officeBtn. Κάθε ένα όταν πατηθεί εμφανίζει και την αντίστοιχη πληροφορία σε ένα result_textView στο μέσο της οθόνης.

```
private void setUpTabs(final JSONObject officeJson, final JSONObject
sectionJson) {

    emailBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String string = "E-Mail :: " + myProfessorDto.getEmail();
            result_textView.setText(string);
        }
    });

    sectionBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            try {
                SectionDto sectionDto = new SectionDto(sectionJson);
                String string = "Τομεας :: " + sectionDto.getSection_name();
                result_textView.setText(string);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });

    officeBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            try {
                OfficeDto officeDto = new OfficeDto(officeJson);
                String string = "Τηλεφωνο :: " + officeDto.getPhone();
                result_textView.setText(string);
                result_textView.append("\nΓραφειο :: " +
officeDto.getNumber());
                result_textView.append("\nΚτηριο :: " +
officeDto.getBuilding());
                result_textView.append("\nΟροφος :: " +
officeDto.getFloor());
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });
```

ProfessorSubject

Στο παρών Activity, εμφανίζονται όλα τα μαθήματα που κάνει ένας επιλεγμένος καθηγητής. Έχει προηγηθεί ένα request στο οποίο με το πάτημα του Button από το προηγούμενο Activity ανακτούμε από το server όλα τα μαθήματα που κάνει ο professor με δεδομένο ID. Δημιουργούμε μια λίστα με τους τίτλους των μαθημάτων η οποία περνάει σαν παράμετρος στο παρών Activity μέσα από το Intent. Το request φαίνεται στο παρακάτω κομμάτι κώδικα.

```
subjectsBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        requests.getProfessorSubjectList(myProfessorDto.getProfessorID(), new
Requests.VolleyCallbackProfessorSubjectList() {
            @Override
            public void onSuccess(JSONArray jsonArray) {
                List<CoursesDto> coursesDtoList = new ArrayList<>();
                for (int i = 0; i < jsonArray.length(); i++) {
                    try {
                        JSONObject jo = jsonArray.getJSONObject(i);
                        CoursesDto coursesDto = new CoursesDto(jo);
                        coursesDtoList.add(coursesDto);
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }

                Intent intent = new Intent(ProfessorInfo.this,
ProfessorSubjects.class);
                Bundle bundle = new Bundle();
                bundle.putSerializable("CoursesDtoList", (Serializable)
coursesDtoList);
                bundle.putString("myProfessorName",
myProfessorDto.getFull_name());
                intent.putExtras(bundle);
                startActivity(intent);
            }
        });
    }
});
```

Η λίστα αυτή που δημιουργείται, γίνεται adapt στο ListView του Activity με την μέθοδο setUpAdater(). Σ' αυτή τη μέθοδο ενεργοποιούμε έναν Listener στο subjects_listView ώστε να είναι ο χρήστης σε θέση να επιλέξει ένα από τα μαθήματα που διδάσκει αυτός ο *professor.* Όταν επιλεγεί ένα από τα στοιχεία της λίστας τότε παίρνουμε το DTO του επιλεγμένου μαθήματος και μέσω Intent ο χρήστης μεταφέρεται σε νέο Activity μεταφέροντας ταυτόχρονα και το αντίστοιχο DTO. Το Activity αυτό βρίσκεται σε άλλο Package, το οποίο θα αναφερθεί παρακάτω.

```
private void setUpAdapter(List list,List<CoursesDto> coursesDtoList) {
    ListViewAdapter listViewAdapter = new ListViewAdapter(this,
R.layout.item_listview, list);
    subjects_listView.setAdapter(listViewAdapter);
    final List<CoursesDto> myDto = coursesDtoList;

    subjects_listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            String selection = (String) parent.getItemAtPosition(position);
            for (int i=0; i< myDto.size(); i++){
                if (myDto.get(i).getCourse_name().equals(selection)) {
                    myCourseDto = myDto.get(i);
                    break;
                }
            }

            Intent intent = new Intent(ProfessorSubjects.this,
SubjectInfo.class);
            Bundle bundle = new Bundle();
            bundle.putSerializable("myCourseDto", myCourseDto);
            intent.putExtras(bundle);
            startActivity(intent);
        }
    });

}
```

- **SubjectTab Fragment Package**

SubjectList

Στο Package αυτό, βρίσκεται η κλάση SubjectList η οποία καλείται από τον SectionsAdapterPager για να φορτωθεί το fragment που αντιστοιχεί στο SubjectTab. Όπως και στα υπόλοιπα fragments, κατά τη δημιουργία τους τρέχουν η onCreate() και η onCreateView() methods. Στην πρώτη μέθοδο επιτελούνται διαδικασίες όπως επικοινωνία με το server και άλλα, ενώ η δεύτερη μέθοδος είναι υπεύθυνη για τη δημιουργία όλων των αντικειμένων που αφορούν το UI της εφαρμογής για το συγκεκριμένο fragment. Η μόνη λειτουργία που εκτελεί η onCreate() είναι ένα request που γίνεται προς το server για την ανάκτηση της λίστας όλων των *courses.*

```
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    System.out.println("Subjects onCreate");
    // Make the Courses Request onCreate the Fragment //
    String string = "courses";
    downloadSubjects(string);
}
```

Το request με όρισμα *courses* παίρνει σαν response ένα JSONArray με όλα τα DTOs από τον πίνακα Courses της βάσης δεδομένων. Έπειτα δημιουργεί μια λίστα που περιέχει μόνο τους τίτλους από κάθε DTO. Για την υλοποίηση του κάθε request υπάρχει μια ξεχωριστή κλάση η οποία θα αναλυθεί παρακάτω.

```
private void downloadSubjects(String data) {
    requests.getSearchSubjList(data, new
Requests.VolleyCallbackSearchSubjList() {
        @Override
        public void onSuccess(JSONArray jsonArray) {
            coursesList = new ArrayList<>();
            coursesDtoList = new ArrayList<>();

            for (int i = 0; i < jsonArray.length(); i++) {
                try {
                    CoursesDto coursesDto = new
CoursesDto(jsonArray.getJSONObject(i));
                    coursesDtoList.add(coursesDto);
                    coursesList.add(coursesDto.getCourse_name());
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }
    });
}
```

Τέλος, η τελευταία μέθοδος που καλείται μέσα στο fragment αυτό είναι η setUpAdapter, η οποία παίρνει ως όρισμα δυο λίστες. Μια λίστα μόνο με τους τίτλους των μαθημάτων, η οποία είναι και η λίστα που γίνεται adapt πάνω στο ListView μέσω του ListViewAdapter που αναφέραμε παραπάνω. Αυτή η λίστα είναι η λίστα που εμφανίζεται στο χρήστη στο παρών Tab και του δίνει τη δυνατότητα να επιλέξει ή να αναζητήσει πάνω σ' αυτήν προχωρώντας στης πληροφορίες του επιλεγμένου course.

```
private void setUpAdapter(List list, List coursesDtoList) {
    activity.setAdapter(new ListViewAdapter(activity, R.layout.item_listview,
list));
    subject_listView.setAdapter(activity.getAdapter());
    final List<CoursesDto> myDto = coursesDtoList;

    subject_listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            String selection = (String) parent.getItemAtPosition(position);
            for (int i=0; i< myDto.size(); i++){
                if (myDto.get(i).getCourse_name().equals(selection)) {
                    myCourseDto = myDto.get(i);
                    break;
                }
            }

            Intent intent = new Intent(activity, SubjectInfo.class);
            Bundle bundle = new Bundle();
            bundle.putSerializable("myCourseDto", myCourseDto);
            intent.putExtras(bundle);
            getActivity().startActivityForResult(intent, PICK_VIEW_REQUEST);
        }
    });
}
```

Σε περίπτωση που ο χρήστης επιλέξει έναν μάθημα, στο *subject_listView* είναι ενεργοποιημένος ένας Listener οποίος ενεργοποιεί ένα Intent. Η κλάση Intent είναι μια κλάση του συστήματος του Android, με την οποία γίνεται triggering η αλλαγή μιας τρέχουσας οθόνης και εμφάνισης μιας καινούργιας στο χρήστη μεταφέροντας δεδομένα από τη μια στη άλλη. Στην περίπτωσή μας, μεταφέρουμε όλο το DTO του επιλεγμένου από το χρήστη *course,* σε μορφή Bundle. Η κλάση Bundle είναι μια κλάση με την οποία γίνεται mapping ένα String Key με ένα πακέτο δεδομένων.

SubjectInfo

Όπως αναφέρθηκε στο 3 κεφάλαιο, αυτό το Activity δείχνει πληροφορίες για κάποιο course. Οι ενδιαφέρουσες μέθοδοι που πρέπει να δούμε εδώ είναι οι εξής.

Όταν φορτώνει αυτή η οθόνη εκτός από αντικείμενα που αφορούν το γραφικό περιβάλλον της εφαρμογής, γίνεται αυτόματα μια επικοινωνία με το server για την άντληση πληροφοριών που πρέπει να δοθούν στο χρήστη. Οι πληροφορίες αφορούν τα Section, Time και ClassRoom Tables από τη βάση δεδομένων. Τα τρία αυτά στοιχεία έρχονται σε τρία ξεχωριστά JSONObjects ενσωματωμένα μέσα σε ένα JSONObject.

```
private void downloadInfos(final int courseID, int sectionID, int
classRoomID){
    requests.getSubjectInformation(courseID, sectionID, classRoomID, new
Requests.VolleyCallbackSubjectInfos() {
        @Override
        public void onSuccess(JSONArray jsonArray) {
            try {
                jsonObject = jsonArray.getJSONObject(0);
                setUpTabs(jsonObject);

            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });
}
```

Τέλος, υπάρχει και η μέθοδος setUpTabs(), η οποία καλείται όταν έχει επιστρέψει το response από το αρχικό request του Activity στο server. Η μέθοδος αυτή εξυπηρετεί τις λειτουργίες τριών Buttons τα οποία εμφανίζουν τις πληροφορίες του *course*, τα οποία είναι sectionBtn, classBtn και timeBtn. Κάθε ένα όταν πατηθεί εμφανίζει και την αντίστοιχη πληροφορία σε ένα result_textView στο μέσο της οθόνης.

```
private void setUpTabs(final JSONObject jsonObject){
    subjectName.setText(myCourseDto.getCourse_name());
    sectionBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            try {
                JSONObject jo = jsonObject.getJSONObject("section");
                SectionDto sectionDto = new SectionDto(jo);
                String string = "Τομεας :: " + sectionDto.getSection_name();
                resultView.setText(string);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });
    classBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            try {
                JSONObject jo = jsonObject.getJSONObject("classRoom");
                ClassRoomDto classRoomDto = new ClassRoomDto(jo);
                String string = "Αιθουσα :: " + classRoomDto.getNumber();
                resultView.setText(string);
                resultView.append("\nΚτηριο :: " +
classRoomDto.getBuilding());
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });
    timeBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            resultView.setText("");
            try {
                JSONArray jo = jsonObject.getJSONArray("times");
                for (int i = 0; i < jo.length(); i++) {
                    TimeDto timeDto = new TimeDto(jo.getJSONObject(i));
                    String string = "Ημερα :: " + timeDto.getDay();
                    resultView.setText(string);
                    resultView.append("\nΕναρξη :: " +
timeDto.getStart_hour());
                    resultView.append("\nΛηξη :: " + timeDto.getEnd_hour() +
"\n");
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });}
```

SubjectProfessor

Στο παρών Activity, εμφανίζονται όλοι οι *professors* που διδάσκουν ένα επιλεγμένο μάθημα. Έχει προηγηθεί ένα request στο οποίο με το πάτημα του Button από το προηγούμενο Activity ανακτούμε από το server όλους τους καθηγητές που διδάσκουν το μάθημα με δεδομένο ID. Δημιουργούμε μια λίστα με τα ονόματα όλων των καθηγητών, η οποία περνάει σαν παράμετρος στο παρών Activity μέσα από το Intent. Το request φαίνεται στο παρακάτω κομμάτι κώδικα.

```
professorsBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        requests.getSubjectProfessorList(myCourseDto.getCourseID(), new
Requests.VolleyCallbackSubjectProfessorList() {
            @Override
            public void onSuccess(JSONArray jsonArray) {
                List<ProfessorDto> professorDtoList = new ArrayList<>();
                for (int i = 0; i < jsonArray.length(); i++) {
                    try {
                        JSONObject jo = jsonArray.getJSONObject(i);
                        ProfessorDto professorDto = new ProfessorDto(jo);
                        professorDtoList.add(professorDto);
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }

                Intent intent = new Intent(SubjectInfo.this,
SubjectProfessors.class);
                Bundle bundle = new Bundle();
                bundle.putSerializable("ProfessorsDtoList", (Serializable)
professorDtoList);
                bundle.putString("myCourseName",
myCourseDto.getCourse_name());
                intent.putExtras(bundle);
                startActivity(intent);
            }
        });
    }
});
```

Η λίστα αυτή που δημιουργείται, γίνεται adapt στο ListView του Activity με την μέθοδο setUpAdater(). Σ' αυτή τη μέθοδο ενεργοποιούμε έναν Listener στο professor_listView ώστε να είναι ο χρήστης σε θέση να επιλέξει έναν από τους καθηγητές που διδάσκουν αυτό το course. Όταν επιλεγεί ένα από τα στοιχεία της λίστας τότε παίρνουμε το DTO του επιλεγμένου καθηγητή και μέσω Intent ο χρήστης μεταφέρεται σε νέο Activity μεταφέροντας ταυτόχρονα και το αντίστοιχο DTO. Το Activity αυτό βρίσκεται σε άλλο Package, το οποίο αναφέρθηκε παραπάνω.

```
private void setUpAdapter(List list, final List<ProfessorDto>
professorDtoList) {
    ListViewAdapter listViewAdapter = new ListViewAdapter(this,
R.layout.item_listview, list);
    professors_listView.setAdapter(listViewAdapter);
    final List<ProfessorDto> myDto = professorDtoList;

    professors_listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            String selection = (String) parent.getItemAtPosition(position);
            for (int i=0; i<professorDtoList.size(); i++){
                if (myDto.get(i).getFull_name().equals(selection)) {
                    myProfessorDto = myDto.get(i);
                    break;
                }
            }

            Intent intent = new Intent(SubjectProfessors.this,
ProfessorInfo.class);
            Bundle bundle = new Bundle();
            bundle.putSerializable("myProfessorDto", myProfessorDto);
            intent.putExtras(bundle);
            startActivity(intent);
        }
    });
}
```

- **SectionTab Fragment Package**

SectionList

Στο Package αυτό, βρίσκεται η κλάση SectionList η οποία καλείται από τον SectionsAdapterPager για να φορτωθεί το fragment που αντιστοιχεί στο SectionTab. Όπως και στα υπόλοιπα fragments, κατά τη δημιουργία τους τρέχουν η onCreate() και η onCreateView() methods. Στην πρώτη μέθοδο επιτελούνται διαδικασίες όπως επικοινωνία με το server και άλλα, ενώ η δεύτερη μέθοδος είναι υπεύθυνη για τη δημιουργία όλων των αντικειμένων που αφορούν το UI της εφαρμογής για το συγκεκριμένο fragment. Η μόνη λειτουργία που εκτελεί η onCreate() είναι ένα request που γίνεται προς το server για την ανάκτηση της λίστας όλων των *sections.*

```
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Make the Sections Request onCreate the Fragment //
    String string = "section";
    downloadSections(string);
}
```

Το request με όρισμα *section* παίρνει σαν response ένα JSONArray με όλα τα DTOs από τον πίνακα Section της βάσης δεδομένων. Έπειτα δημιουργεί μια λίστα που περιέχει μόνο τους τίτλους από κάθε DTO. Για την υλοποίηση του κάθε request υπάρχει μια ξεχωριστή κλάση η οποία θα αναλυθεί παρακάτω.

```
private void downloadSections(String data){
    requests.getSearchSecList(data, new
Requests.VolleyCallbackSearchSecList() {
        @Override
        public void onSuccess(JSONArray jsonArray) {
            sectionList = new ArrayList<>();
            sectionDtoList = new ArrayList<>();
            for (int i = 0; i < jsonArray.length(); i++) {
                try {
                    SectionDto sectionDto = new
SectionDto(jsonArray.getJSONObject(i));
                    sectionDtoList.add(sectionDto);
                    sectionList.add(sectionDto.getSection_name());
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }
    });
}
```

Τέλος, η τελευταία μέθοδος που καλείται μέσα στο fragment αυτό είναι η setUpAdapter, η οποία παίρνει ως όρισμα δυο λίστες. Μια λίστα μόνο με τους τίτλους των τομέων, η οποία είναι και η λίστα που γίνεται adapt πάνω στο ListView μέσω του ListViewAdapter που αναφέραμε παραπάνω. Αυτή η λίστα είναι η λίστα που εμφανίζεται στο χρήστη στο παρών Tab και του δίνει τη δυνατότητα να επιλέξει ή να αναζητήσει πάνω σ' αυτήν προχωρώντας στης πληροφορίες του επιλεγμένου section.

```
private void setUpAdapter(List list, List sectionDtoList){
    activity.setAdapter(new ListViewAdapter(activity, R.layout.item_listview,
list));
    section_listView.setAdapter(activity.getAdapter());

    final List<SectionDto> myDto = sectionDtoList;

    section_listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            String selection = (String) parent.getItemAtPosition(position);
            for (int i=0; i<myDto.size(); i++){
                if (myDto.get(i).getSection_name().equals(selection)) {
                    mySectionDto = myDto.get(i);
                    break;
                }
            }

            Intent intent = new Intent(activity, SectionInfo.class);
            Bundle bundle = new Bundle();
            bundle.putSerializable("mySectionDto", mySectionDto);
            intent.putExtras(bundle);
            getActivity().startActivityForResult(intent, PICK_VIEW_REQUEST);
        }
    });
}
```

SectionInfo

Όπως αναφέρθηκε στο 3 κεφάλαιο, αυτό το Activity δείχνει πληροφορίες για κάποιο section. Όταν φορτώνει αυτή η οθόνη εκτός από αντικείμενα που αφορούν το γραφικό περιβάλλον της εφαρμογής, γίνεται αυτόματα μια επικοινωνία με το server για την άντληση πληροφοριών που πρέπει να δοθούν στο χρήστη. Οι πληροφορίες αφορούν τα Professor και Courses Tables από τη βάση δεδομένων. Τα δυο αυτά στοιχεία έρχονται σε δυο JSONObjects ενσωματωμένα μέσα σε ένα JSONArray.

Μετά τη λήψη του response, φτιάχνονται αμέσως δύο λίστες από το κάθε JSONObject. Η κάθε λίστα περιέχει μόνα τα ονόματα και τους τίτλους από τους καθηγητές και τα μαθήματα αντίστοιχα.

```
private void downloadInfos(int sectionID){
    request.getSectionInformation(sectionID, new
Requests.VolleyCAllbackSectionInfos() {
        @Override
        public void onSuccess(JSONArray jsonArray) {
            try {

                JSONArray professorsJsonArray = jsonArray.getJSONArray(0);
                JSONArray coursesJsonArray = jsonArray.getJSONArray(1);

                List<ProfessorDto> professorDtoList = new ArrayList<>();
                List<CoursesDto> courseDtoList = new ArrayList<>();

                ArrayList<String> profNameList = new ArrayList<>();
                ArrayList<String> subNameList = new ArrayList<>();

                for (int i=0; i<professorsJsonArray.length(); i++){
                    ProfessorDto professorDto = new
ProfessorDto(professorsJsonArray.getJSONObject(i));
                    professorDtoList.add(professorDto);
                    profNameList.add(professorDto.getFull_name());
                }

                for (int i=0; i<coursesJsonArray.length(); i++){
                    CoursesDto coursesDto = new
CoursesDto(coursesJsonArray.getJSONObject(i));
                    courseDtoList.add(coursesDto);
                    subNameList.add(coursesDto.getCourse_name());
                }


setUpAdapter(profNameList,subNameList,professorDtoList,courseDtoList);

            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });
}
```

Σ’ αυτό το Activity υπάρχουν δυο διαθέσιμες λίστες που αφορούν το επιλεγμένο section του χρήστη. Η μια είναι η ProfNameList και η άλλη είναι η SubNameList, οι οποίες δημιουργούνται όπως αναφέραμε παραπάνω. Η πρώτη αντιστοιχεί σε όλους τους *professors* που ανήκουν στον επιλεγμένο τομέα και η δεύτερη σε όλα τα *courses* που ανήκουν στον επιλεγμένο τομέα. Από προεπιλογή εμφανίζεται η λίστα με τα ονόματα των καθηγητών. Ο χρήστης έχει τη δυνατότητα να εναλλάσσει το περιεχόμενο της λίστας με δυο *RadioButtons,* οπότε πατώντας κάθε φορά ένα από τα δύο γίνεται adapt στο ListView η αντίστοιχη λίστα.

Το ListView που περιέχει αυτό το Activity έχει ενεργοποιημένο έναν Listener. Ο χρήστης έτσι έχει τη δυνατότητα να πλοηγηθεί σε πληροφορίες για *professors* και *courses* ανάλογα με το ποια λίστα έχει επιλέξει να φαίνεται. Επιλέγοντας ένα στοιχείο από τη λίστα, γίνεται έλεγχος ποιο από τα δυο *RadioButtons* είναι ενεργοποιημένο και αντίστοιχα γίνεται μετάβαση σε αντίστοιχο Activity. Όταν είναι ενεργοποιημένο το *professorRadioBtn,* αποθηκεύεται το DTO του καθηγητή που έχει επιλεγεί και μέσω Intent ο χρήστης μεταβαίνει στο Activity *ProfessorInfo* το οποίο έχει αναλυθεί παραπάνω. Αντίστοιχα, όταν είναι ενεργοποιημένο το *coursesRadioBtn*, αποθηκεύεται το DTO του μαθήματος που έχει επιλεγεί και μέσω Intent ο χρήστης μεταβαίνει στο Activity *SubjectInfo*, το οποίο έχει αναλυθεί επίσης παραπάνω. Και στις δύο περιπτώσεις, ο χρήστης πατώντας το πίσω κουμπί του κινητού επανέρχεται στην οθόνη με τις λίστες που ήταν.

```
private void setUpAdapter(final List profNames, final List subNames, final
List<ProfessorDto> professorDtoList, final List<CoursesDto> coursesDtoList){
    final ListViewAdapter listViewAdapterProf = new ListViewAdapter(this,
R.layout.item_listview, profNames);
    final ListViewAdapter listViewAdapterSub = new ListViewAdapter(this,
R.layout.item_listview, subNames);
    listView.setAdapter(listViewAdapterProf);
    radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            if (checkedId == R.id.coursesRadioBtn){
                listView.setAdapter(listViewAdapterSub);
            } else if (checkedId == R.id.professorsRadioBtn){
                listView.setAdapter(listViewAdapterProf);
            }
        }
    });
```

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            int selectedId = radioGroup.getCheckedRadioButtonId();
            String selection = (String) parent.getItemAtPosition(position);
            if (selectedId == professorsRadioBtn.getId()){
                ProfessorDto myProfessorDto = null;
                for (int i=0; i<professorDtoList.size(); i++){
                    if
(professorDtoList.get(i).getFull_name().equals(selection)){
                        myProfessorDto = professorDtoList.get(i);
                        break;
                    }
                }
                Intent intent = new Intent(SectionInfo.this,
ProfessorInfo.class);
                Bundle bundle = new Bundle();
                bundle.putSerializable("myProfessorDto", myProfessorDto);
                intent.putExtras(bundle);
                startActivity(intent);

            } else if (selectedId == coursesRadioBtn.getId()){
                CoursesDto myCourseDto = null;
                for (int i=0; i<coursesDtoList.size(); i++){
                    if
(coursesDtoList.get(i).getCourse_name().equals(selection)){
                        myCourseDto = coursesDtoList.get(i);
                        break;
                    }
                }
                Intent intent = new Intent(SectionInfo.this,
SubjectInfo.class);
                Bundle bundle = new Bundle();
                bundle.putSerializable("myCourseDto", myCourseDto);
                intent.putExtras(bundle);
                startActivity(intent);
            }
        }
    });
}
```

- **GimbalTab Fragment Package**

GimbalFragment

Στο Package αυτό, βρίσκεται η κλάση GimbalFragment η οποία καλείται από τον SectionsAdapterPager για να φορτωθεί το fragment που αντιστοιχεί στο GimbalTab. Όπως και στα υπόλοιπα fragments, κατά τη δημιουργία τους τρέχουν η onCreate() και η onCreateView() methods. Στην πρώτη μέθοδο επιτελούνται διαδικασίες όπως επικοινωνία με το server και άλλα, ενώ η δεύτερη μέθοδος είναι υπεύθυνη για τη δημιουργία όλων των αντικειμένων που αφορούν το UI της εφαρμογής για το συγκεκριμένο fragment. Η μόνη λειτουργία που εκτελεί η onCreate() είναι ένα request που γίνεται προς το server για την ανάκτηση της λίστας όλων των *Hotspot* που απεικονίζονται πάνω σε ένα MapFragment.

```
private void downloadInfos() {
    try {
        requests.getAllHotspots(new Requests.VolleyCallbackHotspots() {
            @Override
            public void onSuccess(JSONArray jsonArray) {

                markers.clear();
                for (int i = 0; i < jsonArray.length(); i++) {
                    try {
                        HotspotDto hotspotDto = new
HotspotDto(jsonArray.getJSONObject(i));
                        hotspotDtoList.add(hotspotDto);
                        markers.add(hotspotDto.getSpotname());
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            }
        });
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

```
@Override
public void onMapReady(GoogleMap googleMap) {

    MapsInitializer.initialize(AjendaContext.getContext());
    googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
    mGoogleMap = googleMap;
    for (int i = 0; i < hotspotDtoList.size(); i++) {
        HotspotDto hotspotDto = hotspotDtoList.get(i);
        LatLng latLng = new LatLng(hotspotDto.getLatitude(),
hotspotDto.getLongitude());
        final MarkerOptions markerOptions = new
MarkerOptions().position(latLng).title(hotspotDto.getSpotname());
        googleMap.addMarker(markerOptions);
        googleMap.setOnInfoWindowClickListener(new
GoogleMap.OnInfoWindowClickListener() {
            @Override
            public void onInfoWindowClick(Marker marker) {
                selectedInfoWindow(marker.getTitle());
            }
        });
    }
    googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(center, 16.0f));
}
```

Υπάρχει και η δυνατότητα επίσης ο χρήστης να επιλέξει πάνω στο χάρτη ένα από τα πολλά *Hotspots* με τη χρήση των Markers. Με αυτόν τον τρόπο μπορεί να ενημερωθεί για την τοποθεσία του επιλεγμένου Hotspot καθώς και να πλοηγηθεί προς αυτό με τη χρήση του Google Maps. Επιπλέον, ανάλογα με τον τύπο της τοποθεσίας που αντιστοιχεί το κάθε marker, δηλαδή είτε Class είτε Office αντίστοιχα, ο χρήστης ενημερώνεται για τον τρέχων μάθημα που γίνεται στην αίθουσα αν πρόκειται για τύπο Class. Αντίστοιχα, αν πρόκειται για τύπο Office, η εφαρμογή οδηγεί τον χρήστη στο ProfessorInfo Activity, το οποίο δίνει πληροφορίες για τον καθηγητή του οποίου επιλέχθηκε το γραφείο.

Η λειτουργία αυτή επιτελείται με δυο ξεχωριστά request προς το Server, ανάλογα με τον τύπο του Hotspot. Ο κώδικας που υλοποιεί την παραπάνω λειτουργία φαίνεται παρακάτω.

```
private void selectedInfoWindow(String spotName) {
    HotspotDto hotspotDto = mapNameToHotSpotDto(spotName);
    if(hotspotDto != null) {
        try {
            if (hotspotDto.getType().equals("class")) {
                if (hotspotDto.getHotspotID() == 1) {
                    requests.getCurrentSubject(1, new
Requests.VolleyCallbackCurrentSubject() {
                        @Override
                        public void onSuccess(JSONArray jsonArray) {
                            if (jsonArray.length() == 0) {
                                upDate = "Δεν γίνεται Μάθημα αυτή τη στιγμή";
                            } else {
                                try {
                                    CoursesDto coursesDto = new
CoursesDto(jsonArray.getJSONObject(0));
                                    upDate = "Τρέχων Μάθημα :: " +
coursesDto.getCourse_name();
                                } catch (JSONException e) {
                                    e.printStackTrace();
                                }
                            }
                            resultMarker.setText(upDate);
                        }
                    });

                } else {
                    resultMarker.setText("Δεν γίνεται Μάθημα αυτή τη
στιγμή");
                }

            } else if (hotspotDto.getType().equals("office")) {
                requests.getVisitProfessor(1, new
Requests.VolleyCallbackVisitProfessor() {
                    @Override
                    public void onSuccess(JSONArray jsonArray) {
                        try {
                            ProfessorDto professorDto = new
ProfessorDto(jsonArray.getJSONObject(0));
                            Intent intent = new Intent(activity,
ProfessorInfo.class);
                            Bundle bundle = new Bundle();
                            bundle.putSerializable("myProfessorDto",
professorDto);
                            intent.putExtras(bundle);
                            startActivity(intent);
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
```

GimbalServiceFinder

Η κλάση αυτή καλείται αυτόματα κάθε φορά που η εφαρμογή ξεκινάει. Πρόκειται για ένα Service το οποίο τρέχει στο Background του κινητού. Σκοπός του είναι να ειδοποιεί τον χρήστη σε περίπτωση που το κινητό βρεθεί μέσα στην εμβέλεια ενός Beacon. Σε αυτή την περίπτωση τρέχει η Override method του Gimbal API *onBeaconSighting*, η οποία με βάση το beaconID ενημερώνει τον χρήστη με ένα Notification στο κινητό.

```
Gimbal.setApiKey(this.getApplication(), Gimbal_API_KEY);
Gimbal.start();

if (!Gimbal.isStarted()) {
    Toast toast = Toast.makeText(AjendaContext.getContext(), "Gimbal failed
to start", Toast.LENGTH_SHORT);
    toast.show();
} else {
    Toast toast = Toast.makeText(AjendaContext.getContext(), "Gimbal
started", Toast.LENGTH_SHORT);
    toast.show();

    //Listening for Places//
    placeEventListener = new PlaceEventListener() {
        @Override
        public void onBeaconSighting(BeaconSighting beaconSighting,
List<Visit> list) {
            if (!foundBefore(beaconSighting.getBeacon().getIdentifier())) {

                try {

requests.getHotspotID(beaconSighting.getBeacon().getIdentifier(), new
Requests.VolleyCallbackHotspotID() {
                        @Override
                        public void onSuccess(JSONArray jsonArray) {
                            try {
                                HotspotDto hotspotDto = new
HotspotDto(jsonArray.getJSONObject(0));
                                VisitNotify visitNotify = new
VisitNotify(hotspotDto);
                                if (hotspotDto.getType().equals("office")) {
                                    visitNotify.OfficeType();
                                } else if
(hotspotDto.getType().equals("class")) {
                                    visitNotify.ClassType();
                                }
                            } catch (JSONException e) {
                                e.printStackTrace();
                            }
                        }
                    });
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }
```

VisitNotify

Η κλάση *VisitNotify* καλείται από το Service του Gimbal για να δημιουργήσει το *notification*, ανάλογα με τον τύπο του VisitEvent που πραγματοποιείται. Στην εφαρμογή μας υπάρχουν δυο ειδών Visits. Το ένα είναι το OfficeType(), το οποίο ενημερώνει τον χρήστη ότι περνάει έξω από το γραφείο ενός Καθηγητή και τον πλοηγεί στη Activity *ProfessorInfo*. Το άλλο είδος είναι το ClassType(), το οποίο ενωμερώνει τον χρήστη ότι περνάει έξω από μια αίθουσα διδασκαλίας και ενημερώνεται για το ποιο μάθημα γίνεται αυτή τη στιγμή πλοηγώντας τον στο Activity *SubjectInfo*.

Και στις δυο περιπτώσεις γίνεται ένα request στον Server για να αντληθούν πληροφορίες στη μια περίπτωση για τον Καθηγητή και στην άλλη για το Μάθημα.

```
public void OfficeType() {
    try {
        requests.getVisitProfessor(1, new
Requests.VolleyCallbackVisitProfessor() {
            @Override
            public void onSuccess(JSONArray jsonArray) {
                try {

                    ProfessorDto professorDto = new
ProfessorDto(jsonArray.getJSONObject(0));
                    String upDate = "Δείτε πληροφορίες για τον Καθηγητή!";
                    NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(context);
                    mBuilder.setSmallIcon(R.drawable.notify_img);
                    mBuilder.setContentTitle("Γραφείο " +
professorDto.getFull_name());
                    mBuilder.setContentText(upDate);
                    mBuilder.setAutoCancel(true);

                    Intent resultIntent = new Intent(context,
ProfessorInfo.class);
                    resultIntent.putExtra("myProfessorDto", professorDto);
                    TaskStackBuilder stackBuilder =
TaskStackBuilder.create(AjendaContext.getContext());

                    stackBuilder.addParentStack(MainActivity.class);

                    stackBuilder.addNextIntent(resultIntent);
                    PendingIntent resultPendingIntent =
                            stackBuilder.getPendingIntent(
                                    0,
                                    PendingIntent.FLAG_CANCEL_CURRENT
                            );
                    mBuilder.setContentIntent(resultPendingIntent);

                    NotificationManager mNotificationManager =
                            (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);

                    mNotificationManager.notify(1, mBuilder.build());
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        });
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

```
public void ClassType() {
    try {
        requests.getCurrentSubject(getClassID(hotspotDto.getSpotname()), new
Requests.VolleyCallbackCurrentSubject() {
            @Override
            public void onSuccess(JSONArray jsonArray) {
                try {

                    if (jsonArray.length() == 0) {
                        String upDate = "Δεν γίνεται Μάθημα αυτή τη στιγμή";
                        NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(context);
                        mBuilder.setSmallIcon(R.drawable.notify_img);
                        mBuilder.setContentTitle(hotspotDto.getSpotname());
                        mBuilder.setContentText(upDate);
                        mBuilder.setAutoCancel(true);
                    } else {
                        CoursesDto coursesDto = new
CoursesDto(jsonArray.getJSONObject(0));
                        String upDate = "Δείτε ποιο Μάθημα γίνετε αυτή τη
στιγμή!";
                        NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(context);
                        mBuilder.setSmallIcon(R.drawable.notify_img);
                        mBuilder.setContentTitle(hotspotDto.getSpotname());
                        mBuilder.setContentText(upDate);
                        mBuilder.setAutoCancel(true);

                        Intent resultIntent = new Intent(context,
SubjectInfo.class);
                        resultIntent.putExtra("myCourseDto", coursesDto);
                        TaskStackBuilder stackBuilder =
TaskStackBuilder.create(AjendaContext.getContext());

                        stackBuilder.addParentStack(MainActivity.class);

                        stackBuilder.addNextIntent(resultIntent);
                        PendingIntent resultPendingIntent =
                                stackBuilder.getPendingIntent(
                                        0,
                                        PendingIntent.FLAG_CANCEL_CURRENT
                                );
                        mBuilder.setContentIntent(resultPendingIntent);

                        NotificationManager mNotificationManager =
                                (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);

                        mNotificationManager.notify(1, mBuilder.build());
```

Requests

Τέλος, μια άλλη χρήσιμη κλάση είναι η κλάση *Requests*, στην οποία βρίσκονται όλα τα requests που χτυπάει στον Server το κινητό κατά τη χρήση της εφαρμογής.

Τα requests αυτά είναι τα εξής :

- `getSearchProfList`
- `getSearchSubjList`
- `getSearchSecList`
- `getProfessorSectionOffice`
- `getProfessorSubjectList`
- `getSubjectProfessorList`
- `getSubjectInformation`
- `getSectionInformation`
- `getCurrentSubject`
- `getAllHotspots`
- `getHotspotID`
- `getVisitProfessor`

Ενδεικτικά, αναφέρεται ο κώδικας από το getSearchProfList request, το οποίο καλείται κατά του ProfessorTab, και φέρνει μια λίστα με όλους του καθηγητές της σχολής. Για τα requests χρησιμοποιήθηκε η βιβλιοθήκη Volley, αφού έγινε import στο Android Studio. Εφόσον τα δεδομένα που ανταλλάζουμε με τον Server είναι σε JSON Format, καλούμε την έτοιμη συνάρτηση JsonArrayRequest, περνώντας τα κατάλληλα ορίσματα κάθε φορά, και ορίζοντας το callBackInterface, δηλαδή την μέθοδο που θα καλεστεί όταν το request επιστρέψει επιτυχώς. Σε διαφορετική περίπτωση ενημερώνεται ο χρήστης με μήνυμα λάθους.

```
public void getSearchProfList(final String data, final
VolleyCallbackSearchProfList callbackSearchProfList) throws JSONException {
    String url = myUrl + "FindAll";
    final JSONObject jsonObject = new JSONObject();
    jsonObject.put("data", data);
    JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.POST, url, null,
            new Response.Listener<JSONArray>() {
                @Override
                public void onResponse(JSONArray response) {
                    callbackSearchProfList.onSuccess(response);
                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    System.out.println("Error Request........
"+error.toString());
                    ErrorToast errorToast = new ErrorToast();
                    errorToast.ShowToastMsg("getSearchProfList with data " +
data);
                }
            }) {
        @Override
        protected Map<String, String> getParams() {
            Log.v("checkLogin", "getParams");
            Map<String, String> params = new HashMap<>();
            params.put("data", data);
            return params;
        }

        @Override
        public byte[] getBody() {
            try {
                byte[] bytes = jsonObject.toString().getBytes("UTF-8");
                return bytes;
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
            return null;
        }

        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            Log.v("checkLogin", "getHeaders");
            Map<String, String> headers = new HashMap<String, String>();
            headers.put("Content-Type", "application/json;charset=utf-8");
            headers.put("Accept", "application/json;charset=utf-8");

            return headers;
```

5.2 Η εφαρμογή ProfessorApp

Στην εκκίνηση της εφαρμογής, όπως αναφέραμε στο κεφάλαιο της σχεδίασης, εμφανίζεται στον χρήστη μια οθόνη (Main Activity) με τρεις επιλογές. Αυτές οι επιλογές είναι πρακτικά τρία Buttons τα οποία το καθένα παραπέμπει σε αντίστοιχη οθόνη ανάλογα. Ας δούμε τον κώδικα από το κάθε Button ξεχωριστά.

- **Main Activity**

Με το *cancelBtn* γίνεται ένα request στον Server, φέρνοντάς μας όλα τα μαθήματα του Καθηγητή. Μετά το response γίνεται μετάβαση σε αντίστοιχη οθόνη. Ο κώδικας φαίνεται παρακάτω.

```
cancelBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            requests.getProfessorSubjectList(1, new
Requests.VolleyCallbackProfessorSubjectList() {
                @Override
                public void onSuccess(JSONArray jsonArray) {
                    List<CoursesDto> coursesDtoList = new ArrayList<>();
                    for (int i = 0; i < jsonArray.length(); i++) {
                        try {
                            JSONObject jo = jsonArray.getJSONObject(i);
                            CoursesDto coursesDto = new CoursesDto(jo);
                            if (coursesDto.getValid() != 0) {
                                coursesDtoList.add(coursesDto);
                            }
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                    Intent intent = new Intent(MainActivity.this,
Cancel.class);
                    Bundle bundle = new Bundle();
                    bundle.putSerializable("CoursesList", (Serializable)
coursesDtoList);
                    intent.putExtras(bundle);
                    startActivity(intent);
                }
            });
        } catch (JSONException e) {
            e.printStackTrace();
        }

    }
});
```

Με το *gradesBtn*, ομοίως γίνεται πρώτα ένα request στον Server, για να μας επιστρέψει τη λίστα με τα μαθήματα που δεν έχουν ενημερωθεί με βαθμολόγηση. Με την επιστροφή του request, ο χρήστης οδηγείται σε αντίστοιχη οθόνη.

```
gradeBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            requests.getProfessorSubjectList(1, new
Requests.VolleyCallbackProfessorSubjectList() {
                @Override
                public void onSuccess(JSONArray jsonArray) {
                    List<CoursesDto> coursesDtoList = new ArrayList<>();
                    for (int i = 0; i < jsonArray.length(); i++) {
                        try {
                            JSONObject jo = jsonArray.getJSONObject(i);
                            CoursesDto coursesDto = new CoursesDto(jo);
                            if (coursesDto.getGrades() != 1)
                                coursesDtoList.add(coursesDto);
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                    Intent intent = new Intent(MainActivity.this,
Grade.class);
                    Bundle bundle = new Bundle();
                    bundle.putSerializable("CoursesList", (Serializable)
coursesDtoList);
                    intent.putExtras(bundle);
                    startActivity(intent);
                }
            });
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
});
```

Τέλος, με το *announceBtn*, ο χρήστης παραπέμπεται αμέσως σε μια άλλη οθόνη για να ενημερώσει με μια Ανακοίνωση.

```
announceBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this, Announce.class);
        startActivity(intent);
    }
});
```

- **Cancel Activity**

Η κύρια συνάρτηση που τρέχει σε αυτό το Activity είναι η onCreate() η οποία δημιουργεί τη λίστα με τα ονόματα των μαθημάτων για να επιλέξει ο καθηγητής τα μαθήματα τα οποία θέλει να ακυρώσει. Σε περίπτωση που ο καθηγητής έχει ακυρώσει όλα τα μαθήματα, του έρχεται αντίστοιχη ειδοποίηση ότι τα έχει επιλέξει όλα. Με την ολοκλήρωση της επιλογής ο χρήστης πατάει το sendBtn, και τότε αφού γίνουν οι κατάλληλες τροποποιήσεις γίνεται το τελικό request στον Server για την ενημέρωση της Βάσης όσον αφορά την αλλαγή των ακυρωμένων μαθημάτων. Ο βασικός κώδικας φαίνεται παρακάτω.

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
R.layout.item_list, courseNames);
listView.setAdapter(adapter);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
        String selection = ((TextView) view).getText().toString();
        if (selectedItems.contains(selection)) {
            selectedItems.remove(selection);
        } else
            selectedItems.add(selection);
    }
});

sendBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (flag) {
            final JSONArray jsonArray = new JSONArray();
            for (int i = 0; i < selectedItems.size(); i++) {
                try {
                    JSONObject jsonObject = new JSONObject();
                    jsonObject.put("id", getID(selectedItems.get(i)));
                    jsonArray.put(jsonObject);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
            try {
                requests.sendCancelIDs(jsonArray, new
Requests.VolleyCallbackSendCancelIDs() {
                    @Override
                    public void onSuccess(String response) {
                        MyToast myToast = new
MyToast(MyAppContext.getContext());
                        myToast.showUp();
                        Intent intent = new Intent(Cancel.this,
MainActivity.class);
                        startActivity(intent);
                    }
                });
            } catch (JSONException e) {
                e.printStackTrace();
            }
        } else {
            Intent intent = new Intent(Cancel.this, MainActivity.class);
            startActivity(intent);
        }
```

- **Grade Activity**

Η κύρια συνάρτηση που τρέχει σε αυτό το Activity, ομοίως, είναι η onCreate() η οποία δημιουργεί τη λίστα με τα ονόματα των μαθημάτων για να επιλέξει ο καθηγητής τα μαθήματα τα οποία θέλει να ενημερώσει ότι βαθμολογηθήκαν. Σε περίπτωση που ο καθηγητής έχει ενημερώσει όλα τα μαθήματα, του έρχεται αντίστοιχη ειδοποίηση ότι τα έχει ενημερώσει όλα. Με την ολοκλήρωση της επιλογής ο χρήστης πατάει το sendBtn, και τότε αφού γίνουν οι κατάλληλες τροποποιήσεις γίνεται το τελικό request στον Server για την ενημέρωση της Βάσης όσον αφορά την αλλαγή των ενημερωμένων βαθμολογημένων μαθημάτων. Ο βασικός κώδικας φαίνεται παρακάτω.

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
R.layout.item_list, courseNames);
listView.setAdapter(adapter);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
        String selection = ((TextView) view).getText().toString();
        if (selectedItems.contains(selection)) {
            selectedItems.remove(selection);
        } else
            selectedItems.add(selection);
    }
});

sendBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (flag) {
            JSONArray jsonArray = new JSONArray();
            for (int i = 0; i < selectedItems.size(); i++) {
                try {
                    JSONObject jsonObject = new JSONObject();
                    jsonObject.put("id", getID(selectedItems.get(i)));
                    jsonArray.put(jsonObject);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }

            try {
                requests.sendGradeIDs(jsonArray, new
Requests.VolleyCallbackSendGradeIDs() {
                    @Override
                    public void onSuccess(String response) {
                        MyToast myToast = new
MyToast(MyAppContext.getContext());
                        myToast.showUp();
                        Intent intent = new Intent(Grade.this,
MainActivity.class);
                        startActivity(intent);
                    }
                });
            } catch (JSONException e) {
                e.printStackTrace();
            }
        } else {
            Intent intent = new Intent(Grade.this, MainActivity.class);
            startActivity(intent);
        }
```

- **Announce Activity**

Παρομοίως με τα παραπάνω, έτσι και αυτό το Activity, αποτελείται από μια κύρια συνάρτηση την onCreate(), στην οποία δημιουργείται ένα EditText view στο οποίο ο χρήστης καθηγητής μπορεί να γράψει την ανακοίνωση που θέλει. Πατώντας Αποστολή, γίνεται ένα request στον Server για να αποθηκευτεί στην Βάση Δεδομένων η ανακοίνωση. Ο κύριος κώδικας φαίνεται παρακάτω.

```
sendBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String announce = announceText.getText().toString();
        if (announce.length() > 1) {
            try {
                requests.sendAnnounce(announce, new
Requests.VolleyCallbackSendAnnounce() {
                    @Override
                    public void onSuccess(String response) {
                        MyToast myToast = new
MyToast(MyAppContext.getContext());
                        myToast.showUp();
                        Intent intent = new Intent(Announce.this,
MainActivity.class);
                        startActivity(intent);
                    }
                });
            } catch (JSONException e) {
                e.printStackTrace();
            }
        } else {
            Toast toast = Toast.makeText(MyAppContext.getContext(), "Γράψτε
μια Ανακοίνωση!", Toast.LENGTH_SHORT);

toast.setGravity(Gravity.CENTER_VERTICAL|Gravity.CENTER_HORIZONTAL, 0, 0);
            toast.show();
        }
    }
});
```

- **Requests**

Τέλος, μια άλλη χρήσιμη κλάση είναι η κλάση *Requests*, στην οποία βρίσκονται όλα τα requests που χτυπάει στον Server το κινητό κατά τη χρήση της εφαρμογής.

Τα requests αυτά είναι τα εξής :

- `getProfessorSubjectList`
- `sendCancelIDs`
- `sendGradeIDs`
- `sendAnnounce`

Η δομή του κώδικα από το καθένα είναι ακριβώς ίδια με της εφαρμογής του φοιτητή Ajenda.

- **MyAppContext**

Σε κάθε μία από τις εφαρμογές υπάρχει έτοιμη κλάση που μας δίνει κάθε φορά που χρειάζεται το Context της εφαρμογής. Ο κώδικας φαίνεται παρακάτω

```
public class MyAppContext extends Application{
    private static Context context;
    public static Context getContext() {
        return context;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        context = getApplicationContext();
    }
}
```

6 Σενάριο Χρήσης

Στο κεφάλαιο αυτό θα αναπτύξουμε ένα αρκετά πλήρες σενάριο χρήσης και των δυο εφαρμογών σε συνδυασμό. Στόχος είναι να φανεί και να εμπεδωθεί το τελικό αποτέλεσμα των σχεδιασμών και υλοποιήσεων που προβήκαμε καθώς και η λειτουργικότητα των εφαρμογών με πραγματικά δεδομένα. Το περιεχόμενο θα είναι ως επί το πλείστον οπτικό δείχνοντας screenShots των εφαρμογών για την κάθε μετάβαση από οθόνη σε οθόνη.

Το σενάριο αυτό θα χωριστεί σε δυο χρήσεις:

- Η μια θα είναι μια αλληλεπίδραση μεταξύ Καθηγητή και Φοιτητή, όπου ο Καθηγητής θα κάνει κάποιες αλλαγές με τη χρήση της εφαρμογής του και ο Φοιτητής θα μπορέσει εν τέλει να ενημερωθεί για αυτές με τη χρήση της δικής του.

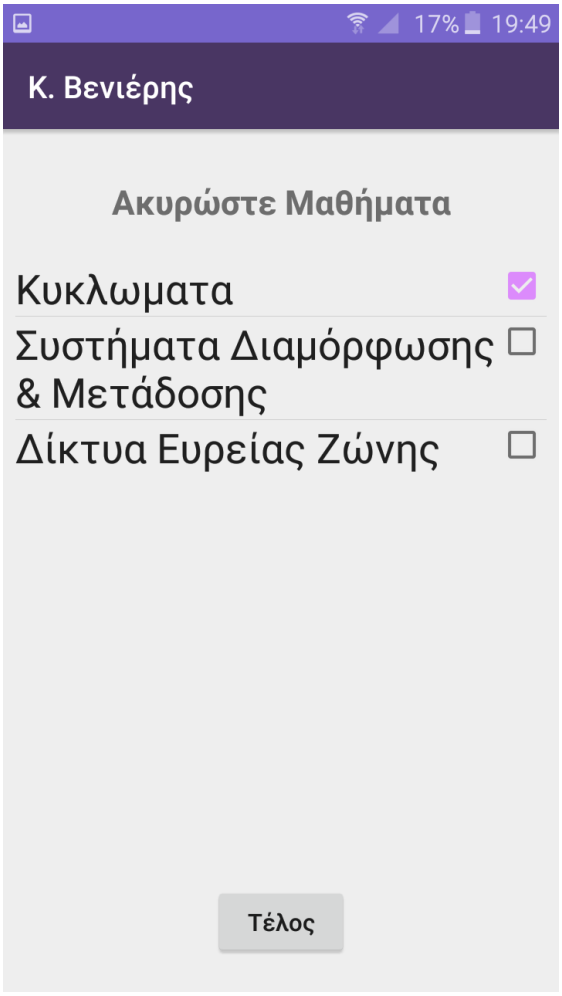
- Η άλλη χρήση θα είναι καθαρά από την πλευρά του Φοιτητή. Θα γίνει χρήση των beacons ώστε ο Φοιτητής να ενημερωθεί με βάση την τοποθεσία που βρίσκεται, πχ έξω από Αίθουσα Διδασκαλίας είτε έξω από το γραφείο ενός Καθηγητή, και θα ενημερωθεί ανάλογα για το τρέχων μάθημα της αίθουσας είτε για τον καθηγητή από του οποίου το γραφείο περνάει.

Ας ξεκινήσουμε θεωρώντας ότι ο κ. Βενιέρης θέλει:

1) Να ενημερώσει ότι η επόμενη διάλεξη από το μάθημα των «Κυκλωμάτων» θα ακυρωθεί.
2) Να ενημερώσει ότι οι βαθμολογίες του μαθήματος «Κυκλώματα» έχουν ανακοινωθεί.
3) Να ενημερώσει το φοιτητικό κοινό του πως θα δέχεται φοιτητές για επεξήγηση βαθμολογιών μόνο Τετάρτη 15:00 – 17:00.

Με την εκκίνηση της εφαρμογής ProfessorApp, θα εμφανιστεί η παρακάτω οθόνη στον χρήστη. Είναι η MainActivity της εφαρμογής όπου ο χρήστης μπορεί να πλοηγηθεί στο επιμέρους μενού που υποδηλώνεται από κάθε κουμπί ξεχωριστά.

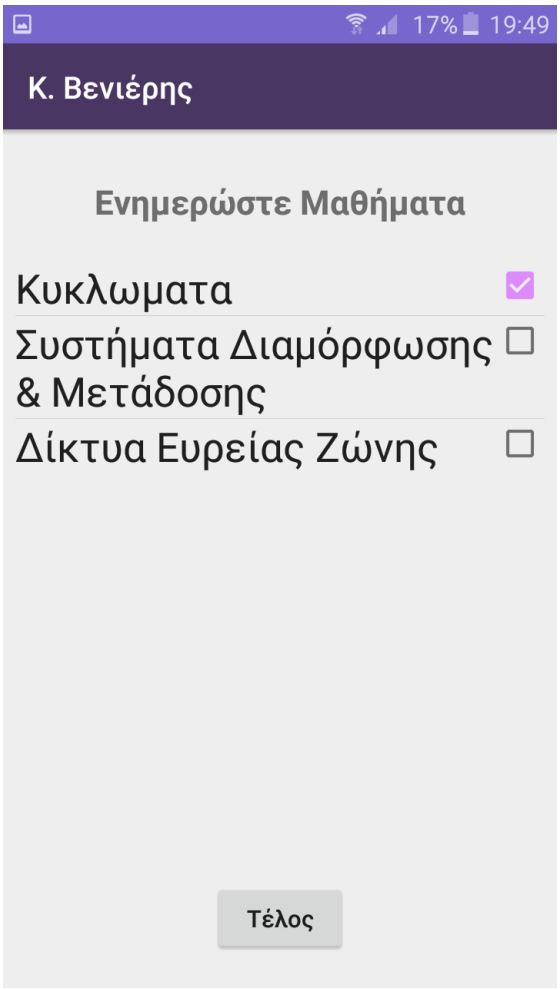
Για να ακυρώσει το μάθημα που θέλει επιλέγει «Ακύρωση Μαθήματος». Έτσι οδηγείται στην επόμενη οθόνη η οποία είναι η εξής.



Εδώ, βλέπει μια λίστα με τα μαθήματα του τα οποία δεν έχουν ακυρωθεί πιο πριν. Επιλέγοντας το μάθημα «Κυκλώματα» και πατώντας «Τέλος» θα ενημερώσει τη Βάση Δεδομένων ότι αυτό το μάθημα είναι ακυρωμένο.

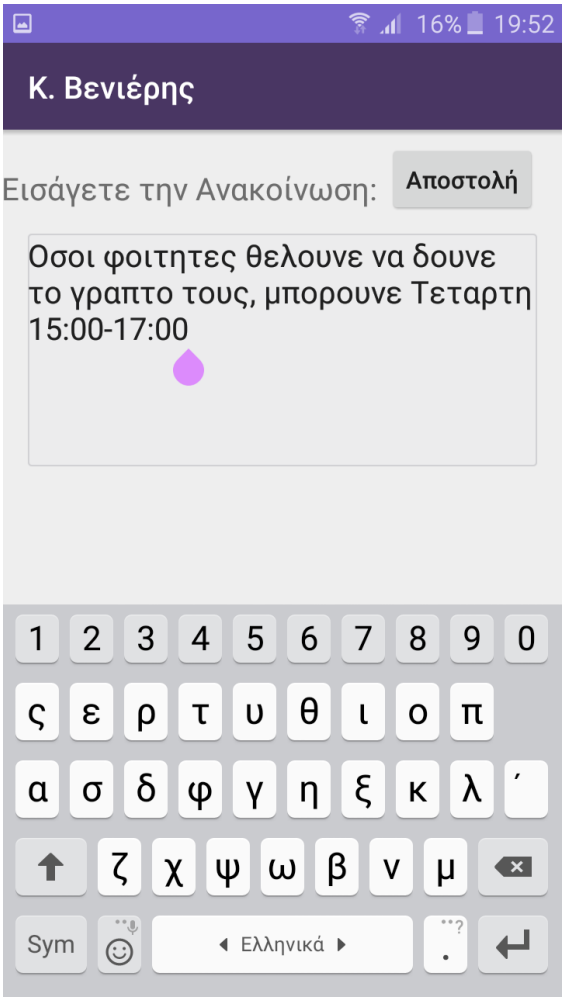
Μετά το τέλος αυτής της διαδικασίας ο καθηγητής βρίσκεται ξανά στην αρχική οθόνη και αυτή τη φορά επιλέγει «Ενημέρωση Βαθμών». Θα μεταβεί με τον ίδιο τρόπο σε μια λίστα με όλα τα μαθήματα που διδάσκει και για τα οποία δεν έχουν βγει βαθμολογίες.

Η οθόνη αυτή μοιάζει με την παραπάνω παρόλο που επιτελούν διαφορετική λειτουργία.



Επιλέγοντας ξανά το μάθημα της επιλογής του, στην περίπτωσή μας «Κυκλώματα», μετά το πάτημα του κουμπιού «Τέλος» αποστέλλεται ενημέρωση στον Server για να ενημερώσει με τη σειρά του τη Βάση Δεδομένων ότι αυτό το μάθημα έχει βαθμολογηθεί και έχουν ανακοινωθεί οι βαθμολογίες.

Μετά το τέλος αυτής της διαδικασίας ο χρήστης θα ξαναβρεθεί στην αρχική οθόνη και αυτή τη φορά θα επιλέξει «Προσθήκη Ανακοίνωσης» για να ενημερώσει τους φοιτητές ότι είναι διαθέσιμος για επεξήγηση γραπτών κάθε Τετάρτη 15:00 με 17:00. Πλοηγείται, λοιπόν, στην επόμενη οθόνη η οποία είναι η εξής:



Με το πάτημα του κουμπιού «Αποστολή», το κείμενο θα αποθηκευτεί στη Βάση Δεδομένων, ώστε να είναι διαθέσιμο σε όλους τους φοιτητές που μπορεί να θέλουν να «δουν το γραπτό τους».

Με αυτό τον πολύ απλό και γρήγορο τρόπο, ο κ Βενιέρης διευθέτησε τρία θέματα μόνο με μια διαδικασία λιγοστών λεπτών, χωρίς καν τη χρήση Η/Υ και χωρίς καν να χρειάζεται να είναι στο γραφείο του ή στο σπίτι του.

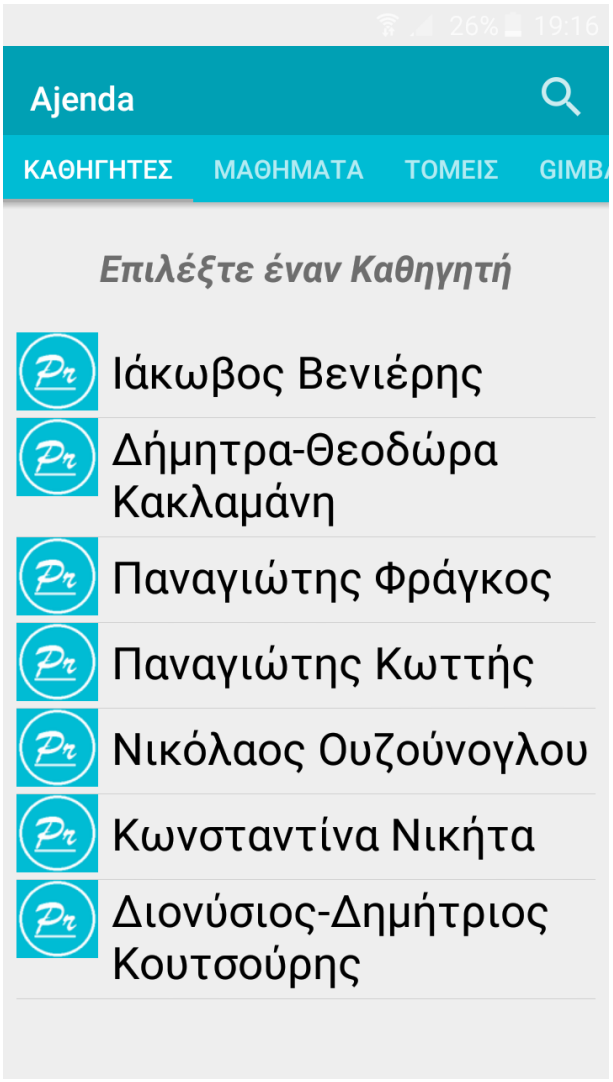
Ας δούμε τώρα, πως αυτές οι πληροφορίες μπορούν να αντληθούν από τον εκάστοτε Φοιτητή, ο οποίος χρησιμοποιεί την εφαρμογή Ajenda.

Στην περίπτωσή μας, ο Χ φοιτητής της σχολής υποθέτουμε πως θέλει να αντλήσει πληροφορίες για τα εξής:

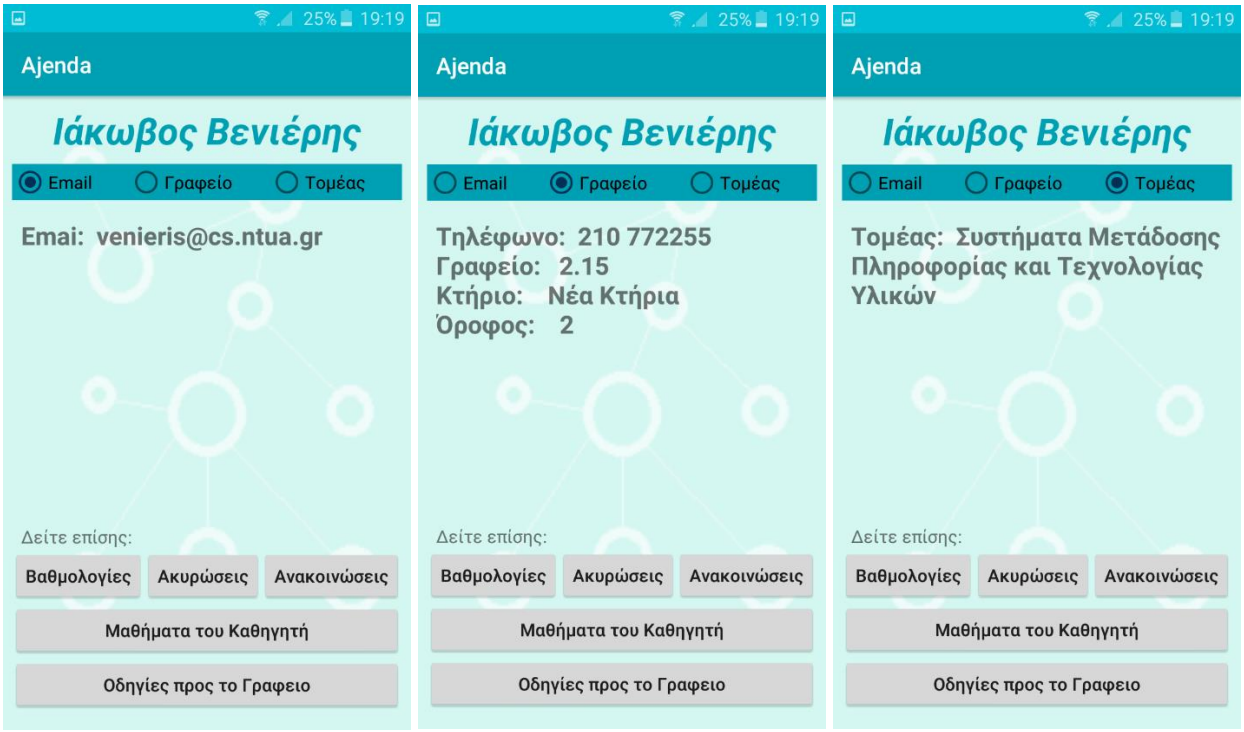
1) Να βρει το «προφίλ» του καθηγητή κ. Βενιέρη και να μάθει αν έχει ακυρώσει κάποιο μάθημα που μπορεί να τον ενδιαφέρει γιατί το παρακολουθεί
2) Να ενημερωθεί για το αν βγήκε η βαθμολογία από κάποιο απ' τα μαθήματα του κ. Βενιέρη που μπορεί να είχε εξεταστεί στις περασμένες εξετάσεις

3) Να ενημερωθεί για οποιαδήποτε ανακοίνωση μπορεί να έβγαλε ο κ Βενιέρης η οποία να του φανεί χρήσιμη και χρειάζεται να ξέρει.

Με τη εκκίνηση της εφαρμογής Ajenda ο φοιτητής θα αντικρίσει την παρακάτω οθόνη. Σε αυτή την οθόνη φορτώνονται από τον Server τα ονόματα όλων των καθηγητών σε μια λίστα. Ο χρήστης μπορεί (πάνω δεξιά) να κάνει search το όνομα του καθηγητή που ψάχνει για πιο γρήγορη πρόσβαση.

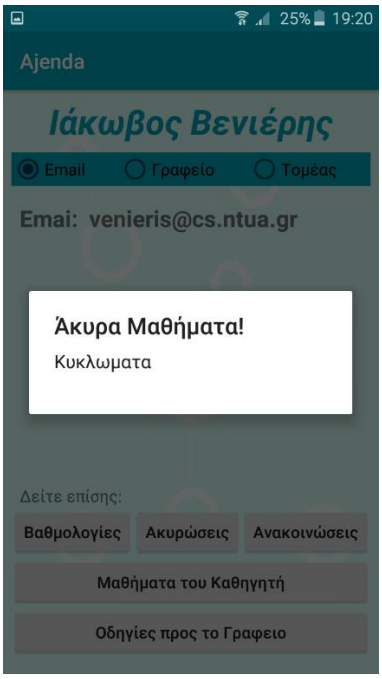


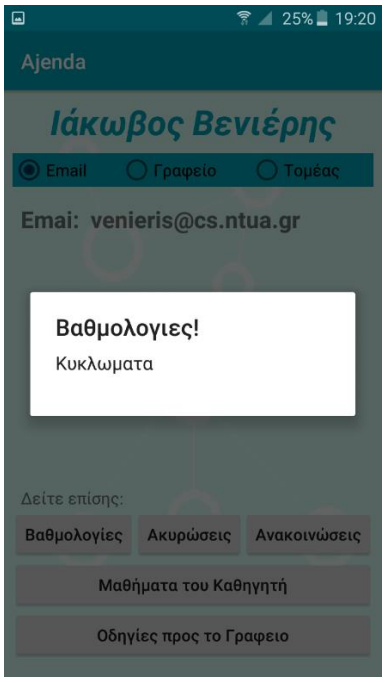
Δεδομένου ότι ο Φοιτητής ενδιαφέρεται για τον κ Βενιέρη θα επιλέξει το πρώτο όνομα στη λίστα. Με αυτή την επιλογή θα πλοηγηθεί στην επόμενη οθόνη η οποία περιέχει πληροφορίες για τον επιλεγμένο καθηγητή, στην περίπτωσή μας τον κ Βενιέρη. Παρακάτω φαίνεται αυτή η οθόνη.



Να σημειωθεί ότι η αριστερή οθόνη που φαίνεται παραπάνω, είναι και η οθόνη που βλέπει ο χρήστης μόλις επιλέξει κάποιον καθηγητή. Η διαφορά τους είναι ότι επιλέγοντας ένα από τα 3 RadioButtons μπορούμε να βλέπουμε κάθε φορά τις αντίστοιχες πληροφορίες (email-γραφείο-τομέας).

Δεδομένου ότι ο χρήστης ήθελε εξ αρχής να μάθει αν ο κ Βενιέρης έχει ακυρώσει κάποιο μάθημα, πατώντας το Button «Ακυρώσεις», βλέπει ποια μαθήματα από αυτά που διδάσκει ο καθηγητής έχουν ακυρωθεί οι προσεχείς τους διαλέξεις.

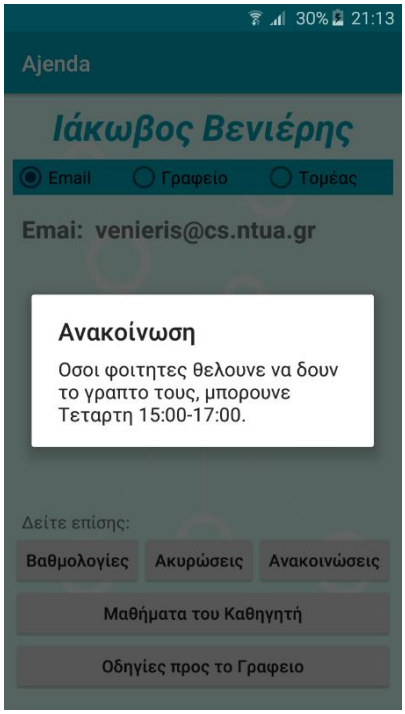




Όπως φαίνεται παραπάνω αριστερά, το μάθημα «Κυκλώματα» έχει ακυρωθεί από τον κ Βενιέρη και έτσι ο φοιτητής μπορεί να το γνωρίζει χωρίς ιδιαίτερο ψάξιμο.

Με παρόμοιο τρόπο πατώντας στο Button «Βαθμολογίες», μπορεί να ενημερωθεί για ποιο μάθημα έχουν ανακοινωθεί βαθμολογίες από μαθήματα που μπορεί να τον ενδιαφέρουν. Όπως φαίνεται στη δεξιά εικόνα, ο κ Βενιέρης έχει ανακοινώσει βαθμολογίες για το μάθημα «Κυκλώματα», μάθημα που ίσως να ενδιαφέρει το φοιτητή.

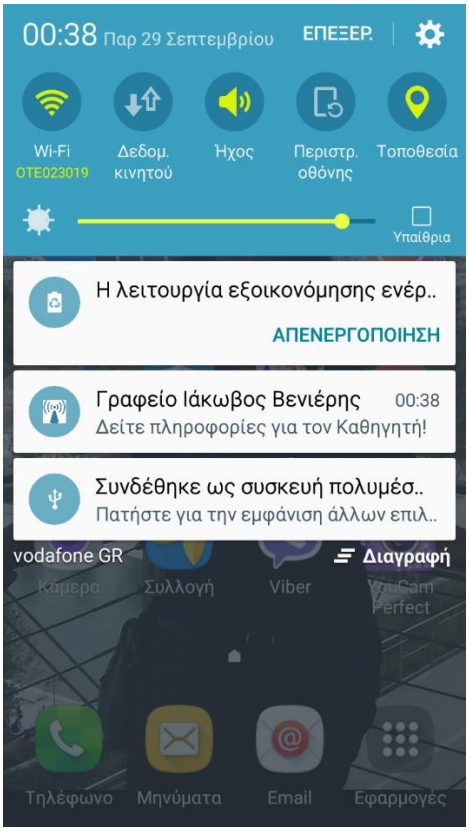
Τέλος, όπως είπαμε παραπάνω, ο φοιτητής θέλει να μάθει αν ο κ Βενιέρης έβγαλε κάποια γενική ανακοίνωση που να αφορά όλους τους φοιτητές ή έστω ένα μέρος απ' αυτούς. Πατώντας το Button «Ανακοινώσεις», ο φοιτητής βλέπει την οποιαδήποτε ανακοίνωση έχει βγάλει ο καθηγητής. Παρακάτω φαίνεται η ανακοίνωση που έχει αποθηκεύσει ο κ Βενιέρης στην εφαρμογή του.



Είδαμε, λοιπόν, πως γίνεται από την πλευρά του καθηγητή μέσα σε λίγα λεπτά και με γρήγορες διαδικασίες να ενημερώσει μέσω της εφαρμογής του για τρία ζητήματα τα οποία είναι αρκετά σημαντικά στη φοιτητική πορεία του φοιτητή και αντίστοιχα ο φοιτητής να ενημερωθεί πολύ γρήγορα και με πολύ απλές διαδικασίες για τα ζητήματα αυτά.

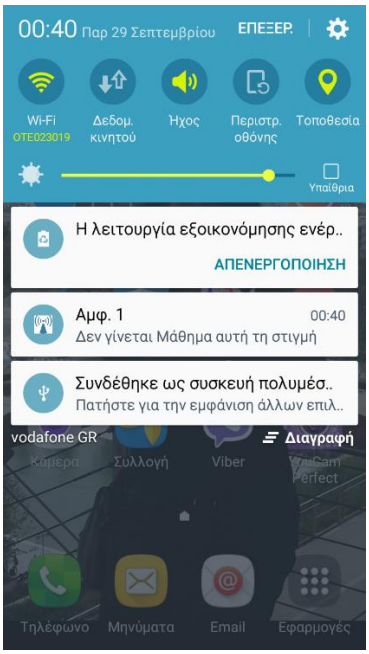
Στη δεύτερη περίπτωση του σεναρίου, θα δούμε πως ο φοιτητής μπορεί να ενημερωθεί αυτόματα από το κινητό του με βάση την τοποθεσία που βρίσκεται. Θεωρώντας ότι κυκλοφορεί μέσα στη σχολή θα δεχτεί ένα notification στο κινητό του σε δυο περιπτώσεις. Είτε περνώντας έξω από μια αίθουσα διδασκαλίας είτε περνώντας έξω από το γραφείο ενός καθηγητή. Στα συγκεκριμένα σημεία που δέχεται αυτή την ειδοποίηση έχουν τοποθετηθεί τα beacons τα οποία ενεργοποιούν τον listener από το Service που τρέχει στο background του κινητού και δημιουργούν την ειδοποίηση.

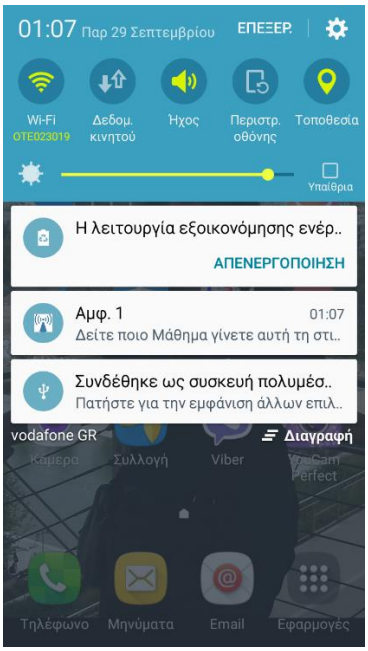
Στην περίπτωσή μας, λοιπόν, ο φοιτητής υποθέτουμε ότι κινείται στο χώρο των γραφείων των καθηγητών, αναζητώντας για παράδειγμα το γραφείο του κ. Βενιέρη. Όταν μπει στην εμβέλεια του beacon, τότε το Service που τρέχει στο κινητό, εντοπίζει το beacon. Με βάση το ID του beacon, ο server επιστρέφει πληροφορίες μετά από ένα request ταυτοποίησης του Τύπου του beacon. Να σημειωθεί ότι υπάρχουν δυο τύποι beacon με τον οποίον είναι αποθηκευμένα στη βάση μας. Ο τύπος “Class” που αναφέρεται σε beacons τα οποία έχουν τοποθετηθεί έξω από Τάξεις, και ο τύπος “Office” που αναφέρεται σε beacons τα οποία έχουν τοποθετηθεί έξω από γραφεία καθηγητών. Εν προκειμένω, ο server ταυτοποιεί τον τύπο “Office”. Για το ID του beacon κάνοντας την αντιστοιχία σε ποιον Καθηγητή ανήκει το beacon δημιουργείται Notification με τη χρήση της κλάσης **VisitNotify** το οποίο ενημερώνει το χρήστη ότι περνάει έξω από το γραφείο του Καθηγητή αυτού. Παρακάτω φαίνεται η ειδοποίηση που δημιουργείται όταν ο φοιτητής περνάει έξω από το γραφείο του κ. Βενιέρη.



Όταν ο χρήστης πατήσει την μπάρα ειδοποίησης, τότε η εφαρμογή Ajenda ανοίγει αμέσως στο Activity ProfessorInfo, δείχνοντας πληροφορίες για τον κ. Βενιέρη.

Στην περίπτωση που ο φοιτητής περάσει έξω από μια αίθουσα, τότε ο ίδιος μηχανισμός ενεργοποιείται και έρχεται ταυτοποίηση από το Server ότι το beacon είναι τύπου “Class”. Στο σημείο αυτό καλείται η κλάση **VisitNotify** για να εμφανιστεί το μάθημα το οποίο διδάσκεται αυτή την ώρα στην αίθουσα αυτή. Σε περίπτωση που δεν διδάσκεται κάποιο μάθημα θα εμφανιστεί ανάλογο μήνυμα στο Notification. Αντιθέτως, αν εντοπιστεί μάθημα που διδάσκεται εκείνη την ώρα θα εμφανιστεί μήνυμα το οποίο θα παραπέμπει στο Activity του μαθήματος αυτού. Παρακάτω φαίνονται οι δυο πιθανές εκδοχές του σεναρίου που μόλις αναλύσαμε.


00:40 Παρ 29 Σεπτεμβρίου
ΕΠΕΞΕΡ.
Wi-Fi
OTE023019
Δεδομ. κινητού
Ήχος
Περιστρ. οθόνης
Τοποθεσία
Υπαίθρια
Η λειτουργία εξοικονόμησης ενέρ..
ΑΠΕΝΕΡΓΟΠΟΙΗΣΗ
Αμφ. 1
00:40
Δεν γίνεται Μάθημα αυτή τη στιγμή
Συνδέθηκε ως συσκευή πολυμέσ..
Πατήστε για την εμφάνιση άλλων επιλ..
vodafone GR
Διαγραφή
Κάμερα
Συλλογή
Viber
YouCam Perfect
Τηλέφωνο
Μηνύματα
Email
Εφαρμογές


01:07 Παρ 29 Σεπτεμβρίου
ΕΠΕΞΕΡ.
Wi-Fi
OTE023019
Δεδομ. κινητού
Ήχος
Περιστρ. οθόνης
Τοποθεσία
Υπαίθρια
Η λειτουργία εξοικονόμησης ενέρ..
ΑΠΕΝΕΡΓΟΠΟΙΗΣΗ
Αμφ. 1
01:07
Δείτε ποιο Μάθημα γίνετε αυτή τη στι..
Συνδέθηκε ως συσκευή πολυμέσ..
Πατήστε για την εμφάνιση άλλων επιλ..
vodafone GR
Διαγραφή
Κάμερα
Συλλογή
Viber
YouCam Perfect
Τηλέφωνο
Μηνύματα
Email
Εφαρμογές


8% 01:08
Ajenda
Κυκλώματα
Τομέας
Αίθουσα
Ώρες
Τομέας: Κορμός
Δείτε επίσης:
Βαθμολογία
Ακυρωμένο
Καθηγητές που κάνουν το μάθημα

7 Παρατηρήσεις

Με το πέρας του σχεδιασμού των παραπάνω δύο εφαρμογών καταλήγουμε σε κάποιες παρατηρήσεις σχετικά με τις τεχνολογίες που χρησιμοποιήθηκαν. Οι παρατηρήσεις αυτές αφοράν στο κατά πόσο αυτές οι εφαρμογές που υλοποιήσαμε περιορίζονται από διάφορους παράγοντες και πως μελλοντικές διορθώσεις και ανασχεδιασμοί θα μπορούσαν να παρακάμψουν αυτά τα εμπόδια.

Αρχικά, παρατηρώντας τη λειτουργία των beacons, φαίνεται ότι πρόκειται για τεχνολογία υλικού που είναι αρκετά ξεπερασμένη και αρκετά δύσχρηστη. Πιο συγκεκριμένα, παρατηρούμε τρία σημαντικά μειονεκτήματα που έχουν να κάνουν με τη λειτουργία του.

- Το πρώτο είναι η σημαντική κατανάλωση ενέργειας που έχουν, καθώς η μπαταρία τους πέφτει σχετικά πολύ γρήγορα, αν αναλογιστούμε βέβαια ότι πρόκειται για μια συσκευή η οποία πρέπει να είναι συνεχώς σε κατάσταση λειτουργίας. Αυτό είναι ένα θέμα το οποίο καθιστά τη χρησιμότητά του ασύμφορη.
- Το δεύτερο μειονέκτημα είναι ότι ο χρόνος απόκρισης του Notification που δημιουργείται, θα μπορούσε να είναι μικρότερος, δηλαδή η επικοινωνία μεταξύ κινητού και beacon φαίνεται να λειτουργεί σχετικά «αργά». Ο πιθανότερος λόγος που συμβαίνει κάτι τέτοιο είναι διότι η αναζήτηση των beacons από το κινητό γίνεται μέσω Bluetooth. Το Bluetooth βασίζεται σε ένα σχετικά αργό πρωτόκολλο σε σχέση με άλλα τελευταίας τεχνολογίας και για αυτό δεν προτιμάται για λειτουργίες άμεσης απόκρισης. Επίσης, είναι προφανές ότι η μεγαλύτερη μερίδα των χρηστών κινητού δεν συνηθίζει να έχει ανοιχτή τη λειτουργία του Bluetooth και λόγω κατανάλωσης μπαταρίας αλλά και επειδή πλέον η ανταλλαγές δεδομένων και όλων των ειδών τα interactions αποφεύγουν τη χρήση Bluetooth. Αυτό το μειονέκτημα θα μπορούσε να οδηγήσει σε σενάριο κατά το οποίο ο χρήστης εισέρχεται μέσα στην εμβέλεια του beacon και λόγω καθυστέρησης του όλου συστήματος ο χρήστης θα εξέλθει από την περιοχή ειδοποιήσεων προτού προλάβει να λάβει Notification. Έτσι καταργείται τελείως η λειτουργικότητα της εφαρμογής.
- Το τρίτο μειονέκτημα είναι η εμβέλεια των beacons η οποία είναι περιορισμένη μέχρι περίπου 50+ μέτρα. Με αυτόν τον τρόπο, δε γίνεται η εφαρμογή αυτή να λειτουργήσει για ειδοποιήσεις απόστασης παραπάνω από 50+ μέτρων. Πλέον, έχουν αναπτυχθεί beacons τα οποία έχουν πολύ μεγαλύτερη εμβέλεια και παρέχουν παραπάνω πληροφορίες στον χρήστη, οπότε τα beacons που χρησιμοποιήθηκαν θεωρούνται ξεπερασμένα και παλιάς τεχνολογίας και περιορίζουν την εφαρμογή στα παραπάνω θέματα που αναφέραμε.

Όσον αφορά στην λειτουργία των εφαρμογών, παρατηρούμε ότι ο χρήστης/Φοιτητής δεν ενημερώνεται εγκαίρως για διάφορες αλλαγές που τυχόν να έχει κάνει ο χρήστης/Καθηγητής στη δική του εφαρμογή. Δεν υπάρχει δηλαδή real-time interaction μεταξύ των δύο εφαρμογών. Έτσι για παράδειγμα, θα μπορούσε να δημιουργηθεί το εξής προβληματικό σενάριο. Ο φοιτητής ελέγχει αν το μάθημα που θέλει να πάει να παρακολουθήσει πρόκειται να γίνει. Αμέσως μετά ο Καθηγητής ενημερώνει μέσω της εφαρμογής του ότι το εν λόγω μάθημα ακυρώνεται. Ωστόσο ο φοιτητής δε το μαθαίνει ποτέ ή όταν το μάθει έχει ήδη φτάσει στην αίθουσα. Η αδυναμία, λοιπόν, της real-time επικοινωνίας των δυο εφαρμογών μπορεί να δημιουργήσει τέτοιου είδους ανεπιθύμητα σενάρια.

8 Μελλοντικές Χρήσεις

Με αφορμή τις παρατηρήσεις που αναφέρθηκαν παραπάνω αλλά και την δυνατότητα επέκτασης του όλου σχεδιασμένου συστήματος που φτιάξαμε, μπορούμε να διατυπώσουμε κάποιες γενικές μελλοντικές χρήσεις των εφαρμογών καθώς και της χρησιμότητας των beacons.

Ακολουθώντας την εξέλιξη της τεχνολογίας, θα μπορούσαν τα beacons που χρησιμοποιήθηκαν να αντικατασταθούν με νέα πιο εξελιγμένα beacons τα οποία παρέχουν αφενός μεγαλύτερη αξιοπιστία σε σχέση με τα μειονεκτήματα που αναφέρθηκαν παραπάνω, αφετέρου περισσότερα δεδομένα τα οποία θα μπορούσαν με κατάλληλο σχεδιασμό και υλοποίηση να προσφέρουν και έξτρα πληροφορίες στο χρήστη. Έτσι, θα έδιναν τη δυνατότητα στην εφαρμογή για περισσότερες λειτουργίες. Κάποιες απ' αυτές τις λειτουργίες είναι :

- Η ταχύτερη και πιο ακριβής ενημέρωση του χρήστη σε μεγαλύτερη απόσταση. Ο περιορισμός της κατανάλωσης της ενέργειας που καταναλώνουν τα beacons.

Μελλοντικές χρήσεις της εφαρμογής εντοπίζονται κυρίως όσον αφορά λειτουργίες τοποθεσίας και Navigation. Θα μπορούσαν τα beacons να χρησιμοποιούνται σαν σημεία αναφοράς για λήψη δεδομένων κίνησης κρατώντας σε κάποιον server, κάποιου είδους ιστορικό. Έτσι, θα μπορούσαμε και βραχυπρόθεσμα αλλά και μακροπρόθεσμα να αντλήσουμε σημεία η περιοχές στα οποία υπάρχει υψηλή κινητικότητα ή χαμηλή αντίστοιχα. Από τα δεδομένα αυτά θα μπορούμε να δημιουργήσουμε αποφάσεις συμβάντων όπως για παράδειγμα τα εξής:

- Αναπάντεχη ξαφνική κίνηση σε περιοχή με συνήθως χαμηλή κινητικότητα θα μπορούσε να σηματοδοτεί κάποια ένδειξη κινδύνου ή την εξέλιξη κάποιου συμβάντος στο σημείο αυτό.

- Αυτόματος διαχωρισμός των ειδοποιήσεων που πρόκειται να έρθουν σε κάποιον χρήστη με βάση κάποιου είδους ιστορικό. Δηλαδή, όταν ο χρήστης περνάει σχεδόν καθημερινά από κάποιο σημείο δεν υπάρχει ιδιαίτερη σημασία να ενημερωθεί γι' αυτό όσο όταν περνάει από ένα σημείο για πρώτη φορά.

- Δυνατότητα εξάπλωσης μιας ενημέρωσης στο κοινό, επιλέγοντας σαν beacons τα σημεία στα οποία υπάρχει υψηλή κινητικότητα. Συνήθως, τέτοιες πολιτικές χρησιμοποιούνται σε πολυκαταστήματα ή σχετικούς πολυσύχναστους χώρους, για την διαφήμιση ενός προϊόντος.

- Υπολογισμός στατιστικών δεδομένων κίνησης π.χ. ταχύτητας οχημάτων μετρώντας τον χρόνο που έκανε το κινητό να εμφανιστεί από το ένα beacon στο άλλο.

- Εφαρμογές navigation για χρήστες κινητών οι οποίοι βρίσκονται εντός κτηρίου. Στην περίπτωση αυτή τα beacons θα μπορούσαν να λειτουργούν σαν σημεία αναφοράς όσον αφορά μια εσωτερική διαδρομή σε έναν εσωτερικό χώρο. Ο χρήστης, κινούμενος «ανάμεσα» στα beacons καθοδηγείται πάνω σε μια διαδρομή, η οποία φτιάχνεται real-time με επανυπολογισμούς κάθε φορά που προσεγγίζεται ένα beacon για νέες οδηγίες.

Παραπάνω είδαμε μερικές γενικές εφαρμογές που θα μπορούσε να έχει αυτή η τεχνολογία των beacons σε διάφορα επίπεδα. Είναι μια εφαρμογή που θα μπορούσε να αναπτυχθεί κάλλιστα και σε πλατφόρμα iOS για ευρεία χρήση από όλους τους χρήστες κινητών. Μπορεί να διατεθεί από τη σχολή στους φοιτητές και στους καθηγητές μέσω του Google Play Store. Ταυτόχρονα μπορεί να συνεχώς να δέχεται βελτίωση είτε προγραμματιστικά διορθώνοντας τυχόν bugs τα οποία ανακαλύπτονται κατά τη χρήση, είτε εμπειρικά προσθέτοντας νέα features τα οποία θα διευκολύνουν τη λειτουργία της εφαρμογής και την καθιστούν όλο και πιο χρηστική. Να σημειωθεί βέβαια, πως χρειάζεται να δοθεί περαιτέρω έμφαση στην ασφάλεια των δεδομένων και τη διαφύλαξη προσωπικών στοιχείων. Συνεπώς, θα πρέπει η Βάση Δεδομένων και ο Server, να σχεδιαστούν σωστά για αποφυγή κακόβουλων λογισμικών και προσπάθειες υποκλοπής δεδομένων. Με αυτόν τον τρόπο, ο χρήστης/Φοιτητής και ο χρήστης/Καθηγητής θα μπορούν να χρησιμοποιούν με ασφάλεια τις εφαρμογές διευκολύνοντας κατά πολύ την ενασχόλησή τους με την Σχολή.

9 Βιβλιογραφία-Αναφορές

[1] Android Developer portal, *https://developer.android.com/*

[2] Gimbal Manager portal, *https://manager.gimbal.com/*

[3] Stackoverflow Forum, *https://stackoverflow.com/*

[4] Eclipse IDE, *http://www.eclipse.org/*

[5] Java, *https://www.oracle.com/*

[6] Hibernate framework, *http://hibernate.org/*

[7] MySQL and Workbench framework, *https://dev.mysql.com/*

[8] Bryan Basham, Kathy Sierra and Bert Bates, *Head First Servlets and JSP*