ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη εφαρμογής προσωπικού προγραμματιστή δραστηριοτήτων σε πλατφόρμα iOS

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

Εμμανουήλ Γ. Καραμανή
Πέτρου Φλώριου Ν. Μπάκαλου

Επιβλέπων : Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη εφαρμογής προσωπικού προγραμματιστή δραστηριοτήτων σε πλατφόρμα iOS

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Των

Εμμανουήλ Γ. Καραμανή
Πέτρου Φλώριου Ν. Μπάκαλου

Επιβλέπων : Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Πέμπτη, 20 Ιουνίου 2013

……..	……..	……..
Ιάκωβος Βενιέρης Καθηγητής Ε.Μ.Π.	Δήμητρα Θεοδώρα Κακλαμάνη Καθηγήτρια Ε.Μ.Π.	Νικόλαος Ουζούνογλου Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2013

……

Εμμανουήλ Γ. Καραμανής

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

……

Πέτρος Φλώριος Ν. Μπάκαλος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Οι «έξυπνες» συσκευές και συγκεκριμένα τα «έξυπνα» τηλέφωνα (smartphones), αποτελούν πλέον αναπόσπαστο κομμάτι της καθημερινότητας μας. Σε αυτά, προσφέρεται μια πληθώρα εφαρμογών, που μας βοηθούν σε πολλές πτυχές της ζωής μας.

Ο σκοπός της διπλωματικής εργασίας είναι η σχεδίαση και η ανάπτυξη μίας εφαρμογής, η οποία θα λειτουργεί σαν προσωπικός βοηθός για το χρήστη της, βοηθώντας τον να καταγράψει και να εκτελέσει το πρόγραμμα της καθημερινότητας του. Η εφαρμογή προορίζεται για τις συσκευές που χρησιμοποιούν το λειτουργικό σύστημα iOS, οι οποίες είναι το iPhone, το iPod και το iPad.

Η εφαρμογή θα παρέχει στο χρήστη τη δυνατότητα να δημιουργεί δραστηριότητες και να τις οργανώνει σε ένα πρόγραμμα. Για κάθε δραστηριότητα ο χρήστης θα μπορεί να θέτει μία τοποθεσία για τη πραγματοποίηση της. Οι δραστηριότητες θα χωρίζονται σε δύο κατηγορίες, τις προγραμματισμένες και τις μη προγραμματισμένες. Για τις προγραμματισμένες ο χρήστης θα λαμβάνει ειδοποιήσεις χρονικά, κάποια στιγμή πριν από τη δραστηριότητα και τη στιγμή της έναρξης της. Για τις μη προγραμματισμένες θα χρησιμοποιηθούν υπηρεσίες τοποθεσίας και ο χρήστης θα ειδοποιείται όταν βρίσκεται σε κοντινή απόσταση από την επιθυμητή τοποθεσία.

Η χρήστης θα μπορεί να προσθέτει νέες τοποθεσίες από τον χάρτη αλλά θα μπορεί να χρησιμοποιεί και το λογαριασμό του στο Facebook για να αναζητά κοντινές τοποθεσίες. Αυτές τις τοποθεσίες θα είναι δυνατό να τις διαθέτει και στους άλλους χρήστες της εφαρμογής μέσω ενός server.

Η ανάπτυξη της εφαρμογής έγινε με τη βοήθεια δύο υπολογιστών της Apple, ενός Mac mini και ενός Macbook Pro, σε περιβάλλον Mac OS X 10.8.4. Επίσης για τη δοκιμή της εφαρμογής χρησιμοποιήθηκαν δύο iOS συσκευές, ένα iPhone 5 και ένα iPod touch τέταρτης γενιάς, τα οποία χρησιμοποιούν της έκτη έκδοση του λειτουργικού συστήματος, το iOS 6. Τέλος, για την ανάπτυξη της εφαρμογής, χρειάστηκε να εγγραφούμε ως μέλη στο πρόγραμμα ανάπτυξης εφαρμογών για iOS της Apple (iOS Developer Program).

Λέξεις κλειδιά: iOS, iPhone, εφαρμογή, υπηρεσίες τοποθεσίας, υπηρεσίες ειδοποίησης, σχεσιακή βάση δεδομένων.

Abstract

Smart devices and especially smartphones, are an integral part of our everyday lives. They offer a huge amount of applications, that could help us in almost every aspect of our lives.

The purpose of this thesis is to design and develop an application, which works as a personal assistant for the user, helping him to record and perform the program of his everyday life. The application is intended for devices that use the operating system iOS, which are iPhone, iPod and iPad.

The application will provide the user the ability to create activities and to organize them in a program. For each activity, the user may specify a location for its accomplishment. Activities will be divided into two categories, scheduled and not scheduled. For the scheduled activities, the user will receive notifications, sometime prior to the activity and at the time of accomplishment. For the not scheduled ones, location based services will be used and the user will be notified when he is within walking distance from the desired location.

The user will not only be able to add new locations on the map, but also use his account on Facebook to search for nearby locations. These locations will be possible to be disposed to other users of the application via a server.

The development of the application was performed using two computers of Apple, a Mac mini and a Macbook Pro, in an environment of Mac OS X 10.8.4. Also, for testing the application we used two iOS devices, an iPhone 5 and an iPod touch fourth generation, which use iOS 6. Finally, for the application development, we had to register as members of application development program for iOS of Apple (iOS Developer Program).

Keywords: iOS, iPhone, application, location-based services, notification services, relational database.

Ευχαριστίες

Για την εκπόνηση της παρούσας διπλωματικής εργασίας, και την ανάπτυξη της εφαρμογής αλλά και γενικότερα για την ολοκλήρωση του κύκλου των σπουδών μας, νιώθουμε την ανάγκη να ευχαριστήσουμε όλους τους ανθρώπους που μας βοήθησαν.

Αρχικά, θα θέλαμε να ευχαριστήσουμε θερμά τον επιβλέποντα της διπλωματικής εργασίας μας, καθηγητή Ιάκωβο Βενιέρη, ο οποίος μας έδωσε τη δυνατότητα να ασχοληθούμε με ένα αντικείμενο που πραγματικά επιθυμούσαμε και οι δύο. Τον ευχαριστούμε ιδιαίτερα για τις πολύτιμες γνώσεις και συμβουλές που μας παρείχε κατά την εκπόνηση της εργασίας και καθ' όλη τη διάρκεια των σπουδών μας, αλλά κυρίως για την προσωπική ικανοποίηση που εισπράξαμε με το πέρας της παρούσας διπλωματικής εργασίας και την αίσθηση του ότι συμπεριλαμβανόμαστε πλέον κι εμείς στους εν δυνάμει προγραμματιστές.

Στη συνέχεια, θα θέλαμε να ευχαριστήσουμε ξεχωριστά τα μέλη της επιτροπής, την καθηγήτρια Δήμητρα Θεοδώρα Κακλαμάνη για την σημαντική συμβολή της στην περαίωση της εργασίας καθώς και τον καθηγητή Νικόλαο Ουζούνογλου για την υποστήριξή του καθ' όλη τη διάρκεια της εκπόνησης της.

Ιδιαίτερες ευχαριστίες θα θέλαμε να απευθύνουμε στον υποψήφιο Διδάκτορα Αζίζ Μούσα, χωρίς τη βοήθεια του οποίου θα ήταν αδύνατη η ολοκλήρωση της διπλωματικής εργασίας. Η συμβολή του στην ανάπτυξη της εφαρμογής και στη συγγραφή της εργασίας υπήρξε καθοριστική.

Επίσης, θα θέλαμε να ευχαριστήσουμε όλο το προσωπικό του Εργαστηρίου Ευφυών Επικοινωνιών και Δικτύων Ευρείας Ζώνης που μας πρόσφερε τη βοήθεια του, όποτε αυτή ζητήθηκε.

Θα θέλαμε και οι δυο μας να ευχαριστήσουμε την Αικατερίνη Μπακάλου, η οποία μας βοήθησε σε θέματα που αφορούσαν τη σχεδίαση της διεπαφής χρήστη της εφαρμογής.

Τέλος, ο καθένας μας προσωπικά, θα θέλαμε να ευχαριστήσουμε τις οικογένειες μας.

Εγώ, ο Πέτρος Φλώριος Μπάκαλος, θα ήθελα να ευχαριστήσω την μητέρα μου για τη στήριξη που μου προσφέρει όλα αυτά τα χρόνια και αφιερώνω την διπλωματική μου εργασία στη μνήμη του πατέρα μου.

Εγώ, ο Εμμανουήλ Καραμανής, θα ήθελα να ευχαριστήσω τους γονείς και την αδερφή μου, που με στήριξαν σε όλη τη διάρκεια των σπουδών μου.

Πίνακας Περιεχομένων

1

Εισαγωγή

1.1. Η εξέλιξη των κινητών τηλεπικοινωνιών και οι «έξυπνες» συσκευές

Όταν ο Μάρτιν Κούπερ, μηχανικός της Motorola, στις 3 Απριλίου του 1973, πραγματοποιούσε την πρώτη κλήση από κινητό τηλέφωνο, σίγουρα δεν μπορούσε να φανταστεί την εξέλιξη που έγινε στο χώρο της κινητής τηλεφωνίας και των τηλεπικοινωνιών. Η εξέλιξη της κινητής τηλεφωνίας ήταν τόσο ραγδαία, που μόλις σε 21 χρόνια, από το 1990 μέχρι το 2011, κατάφερε να διεισδύσει στο 87% του παγκόσμιου πληθυσμού, με 6 δισεκατομμύρια συνδέσεις.

Στις μέρες μας, 40 χρόνια μετά την πρώτη κλήση, η επικοινωνία έχει αλλάξει δραματικά. Με την εξέλιξη της τεχνολογίας και του διαδικτύου, η ανάγκη των ανθρώπων για επικοινωνία αλλάζει, παίρνει διάφορες μορφές. Στο διαδίκτυο κυριαρχούν ιστοσελίδες κοινωνικής δικτύωσης (όπως Facebook, Twitter, Google+), ιστοσελίδες προορισμένες για συζήτηση μεταξύ των χρηστών, ιστοσελίδες αναπαραγωγής πολυμέσων, οι οποίες διοικούνται από εταιρίες κολοσσούς όπως είναι η Facebook , η Google, η Microsoft, η Yahoo.

Όπως είναι λογικό, η εξέλιξη αυτή της επικοινωνίας ώθησε τις κατασκευάστριες εταιρίες κινητών τηλεφώνων, να δημιουργούν ολοένα και πιο «έξυπνες» συσκευές, ώστε να ικανοποιήσουν τις νέες αυτές ανάγκες επικοινωνίας. Η δυνατότητα σύνδεσης των συσκευών αυτών στο διαδίκτυο ήταν το πρώτο βήμα που έγινε προς αυτήν την κατεύθυνση, με την ενσωμάτωση των δικτύων τρίτης γενιάς (3G), τα οποία πρόσφεραν στο χρήστη τη δυνατότητα να περιηγηθεί στο διαδίκτυο. Οι συσκευές όμως, αν και είχαν κάποιες δυνατότητες για λήψη διαφόρων ειδών υλικού, όπως πολυμέσα, είχαν και πάλι σαν κύριο χαρακτηριστικό τους την πραγματοποίηση τηλεφωνικών κλήσεων. Η εμπειρία περιήγησης στο διαδίκτυο και όλες οι επιπρόσθετες λειτουργίες δεν είχαν την απαιτούμενη εμπειρία χρήσης.

Εικόνα 1 Η εξέλιξη των συσκευών. Αριστερά το Qualcomm QCP-2700 (μέσα δεκαετίας του 1990) και δεξιά το iPhone 5, τελευταίο μοντέλο της Apple.

Το 2007 η Apple παρουσίασε το πρώτο της «έξυπνο» κινητό τηλέφωνο, το iPhone. Η συσκευή αυτή, έδινε στο χρήστη τη δυνατότητα να χρησιμοποιήσει εφαρμογές, για να καλύψει όχι μόνο τις ανάγκες επικοινωνίας του, αλλά να ψυχαγωγηθεί, να ενημερωθεί, να μοιραστεί και να δημιουργήσει, με εμπειρία χρήσης ανάλογη ενός υπολογιστή. Ενώ καινοτόμες ιδέες είχαν υλοποιηθεί τα προηγούμενα χρόνια, καμία από αυτές δεν είχε την ανταπόκριση που είχε η κίνηση της Apple. Το παράδειγμα της ακολούθησαν οι μεγάλες εταιρίες του χώρου, όπως η Samsung, η Sony, η Nokia, η HTC καταλήγοντας να έχουμε τη δυνατότητα να κρατάμε στα χέρια μας συσκευές με υπολογιστική ισχύ μεγαλύτερη από πολλούς υπολογιστές.

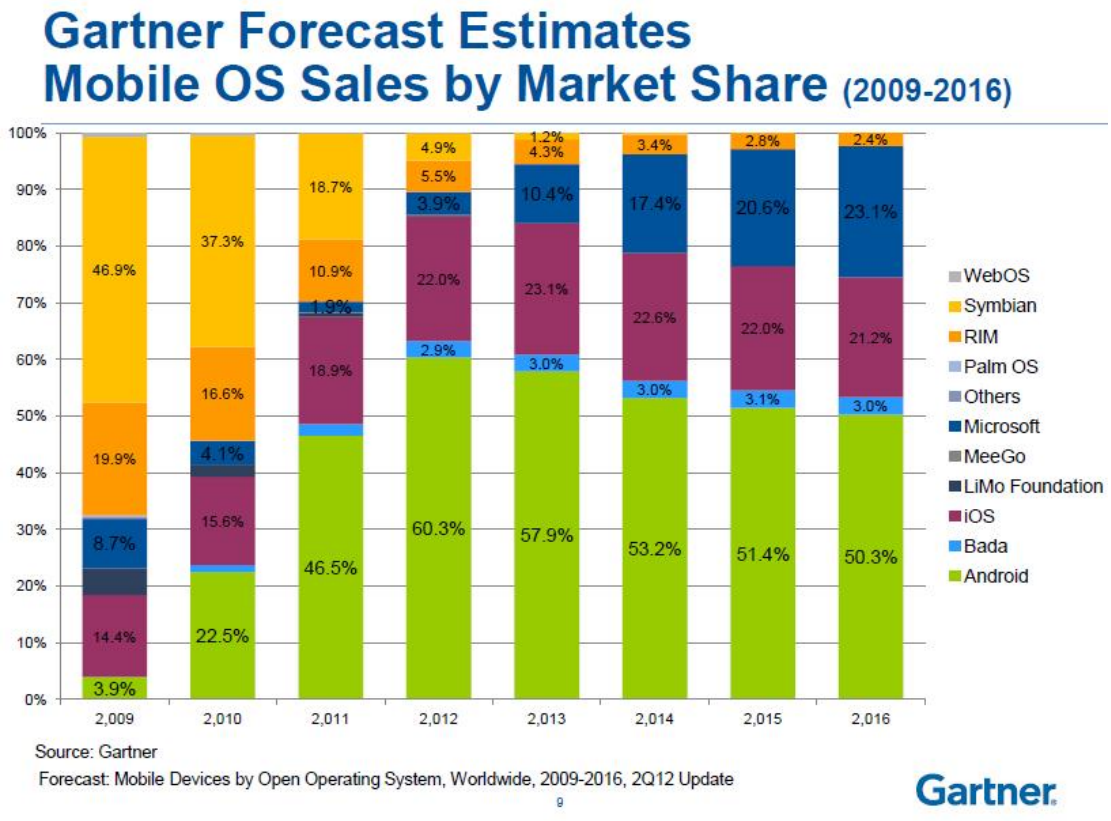
Με όλη αυτήν την εξέλιξη η ανάγκη για γρηγορότερα δίκτυα οδήγησε στη δημιουργία των δικτύων τέταρτης γενιάς (4G ή LTE). Τα δίκτυα αυτά, που έγιναν πρόσφατα διαθέσιμα και στην Ελλάδα, προσφέρουν απίστευτα μεγάλες ταχύτητες 20-50 MBps, ταχύτητες μεγαλύτερες από αυτές που μας προσφέρουν οι ευρυζωνικές συνδέσεις που έχουμε στα σπίτια μας.

1.2. Τα λειτουργικά συστήματα και οι εφαρμογές

Οι «έξυπνες» αυτές συσκευές λειτουργούν χρησιμοποιώντας κάποιο από τα τρία δημοφιλέστερα λειτουργικά συστήματα, που είναι το iOS (Apple), το Android (Google) και το Windows Phone (Microsoft). Οι συσκευές της Apple έχουν το iOS, η Google έχει κατασκευάσει μερικές συσκευές που έχουν το Android, ενώ η Samsung, η Sony, η Nokia και η HTC έχουν μερικές συσκευές που «τρέχουν» Android και άλλες που έχουν Windows Phone.

Τα λειτουργικά αυτά συστήματα προσφέρουν μια πληθώρα εφαρμογών, που είναι σε θέση να καλύψουν τις ανάγκες ακόμα και του πιο απαιτητικού χρήστη. Οι πιο διαδεδομένες και κοινές στα τρία λειτουργικά είναι η εφαρμογή τηλεφώνου, μηνυμάτων, ηλεκτρονικού ταχυδρομείου, περιήγησης ιστού, προβολής καιρού, αναπαραγωγής πολυμέσων, λήψης φωτογραφιών, αριθμομηχανής και πολλών ακόμα.

Το μεγαλύτερο μερίδιο της αγοράς των λειτουργικών συστημάτων, κατέχει το Android της Google, με ποσοστό που κυμαίνεται μεταξύ 50% και 60% τα τελευταία χρόνια. Ακολουθεί το iOS της Apple με ποσοστό περίπου στο 23%. Τον τελευταίο καιρό τα Windows Phone της Microsoft έχουν φτάσει στο 10% της αγοράς. Τα καθαρά νούμερα ενεργοποιήσεων των συσκευών είναι τεράστια. Τον περασμένο Δεκέμβριο μόνο, ενεργοποιήθηκαν 50 εκατομμύρια συσκευές iOS και Android.



Εικόνα 2 Το μερίδιο της αγοράς των λειτουργικών συστημάτων από το 2009 και μια πρόβλεψη μέχρι το 2016.

Τα λειτουργικά όμως προσφέρουν και τη δυνατότητα στους προγραμματιστές να δημιουργήσουν τις δικές τους εφαρμογές. Έτσι, τα ηλεκτρονικά καταστήματα εφαρμογών, περιέχουν χιλιάδες εφαρμογές, από παιχνίδια και εφαρμογές ενημέρωσης, μέχρι σύνθετες εφαρμογές επεξεργασίας φωτογραφιών και πολυμέσων. Υπάρχουν εφαρμογές για όλα τα κοινωνικά δίκτυα, στα οποία ο χρήστης μπορεί να συνδεθεί ανά πάσα στιγμή, ενώ πολλά έχουν δημιουργηθεί μόνο για αυτές τις συσκευές. Η συνδεσιμότητα με τους ηλεκτρονικούς υπολογιστές και με άλλες παρόμοιες συσκευές, όπως τα tablets, είναι επίσης μια σημαντική λειτουργία που προσφέρεται. Τέλος, οι χρήστες έχουν τη δυνατότητα να χρησιμοποιούν cloud υπηρεσίες. Μπορούν δηλαδή, να μην αποθηκεύουν τα δεδομένα τους στη συσκευή, αλλά να τα μεταφέρουν σε ένα διαδικτυακό «νέφος», και να έχουν πρόσβαση σε αυτά μέσω του διαδικτύου, από οποιαδήποτε συσκευή τους.

Στα λειτουργικά συστήματα υπάρχουν ηλεκτρονικά καταστήματα από τα οποία ο χρήστης μπορεί να περιηγηθεί στις διαθέσιμες, για τη συσκευή του, εφαρμογές και να επιλέξει να «κατεβάσει» όποια επιθυμεί. Οι περισσότερες εφαρμογές έχουν ως αντίτιμο ένα μικρό ποσό, της τάξης του ενός ευρώ ή διατίθενται δωρεάν, αλλά υπάρχουν και εφαρμογές που στοιχίζουν μέχρι και 10 ευρώ, μπορεί και περισσότερο. Στο κατάστημα της Apple υπάρχουν περίπου 900.000 εφαρμογές και ο συνολικός αριθμός των λήψεων υπολογίζεται σε 50 δισεκατομμύρια, ενώ τα κέρδη των προγραμματιστών έχουν ξεπεράσει τα 10 δισεκατομμύρια δολάρια. Πρόκειται λοιπόν για ένα οικοσύστημα που απασχολεί χιλιάδες προγραμματιστές ανά τον κόσμο, συνδέονται καθημερινά εκατομμύρια χρήστες οι οποίοι, χρησιμοποιώντας τις συσκευές τους επικοινωνούν και αλληλεπιδρούν μεταξύ τους.

Η χρήση των «έξυπνων» συσκευών και των εφαρμογών, έχει διεισδύσει βαθειά μέσα στην καθημερινότητα του ανθρώπου, με αποτέλεσμα να επηρεάζει πολλές πτυχές της ζωής του. Από την επικοινωνία με τους συνανθρώπους του και τη δουλειά του, τη διασκέδαση, την ενημέρωση και τη μόρφωση του, μέχρι και για την υγεία του, ο άνθρωπος χρησιμοποιεί τη συσκευή του. Έτσι λοιπόν, ο χρήστης απαιτεί ακόμα καλύτερες και ταχύτερες συσκευές, εφαρμογές πιο χρηστικές, που να ικανοποιούν ολοένα και περισσότερες ανάγκες του, όσο σύνθετες κι αν είναι αυτές. Όλο και περισσότερες εταιρίες επενδύουν προς αυτήν την κατεύθυνση, με αποτέλεσμα ο ανταγωνισμός να είναι πολύ μεγάλος και τελικά κερδισμένος να είναι ο καταναλωτής.

Εικόνα 3 Οι τυπικές αρχικές οθόνες των δύο δημοφιλέστερων λειτουργικών συστημάτων. Αριστερά είναι το iOS 6 στο iPhone 5 και δεξιά το Android 4.2 στο Nexus 4.

1.3. Υπηρεσίες βάσει της τοποθεσίας

Από τις υπηρεσίες που έχει ανάγκη ο χρήστης, δε θα μπορούσαν να λείπουν αυτές που είναι βασισμένες στην τρέχουσα τοποθεσία του χρήστη (Location Based Services, LBS) και γενικά στην τοποθεσία ενός σημείου ενδιαφέροντος. Ανάγκες όπως πλοήγηση σε μια συγκεκριμένη τοποθεσία, αναζήτηση κάποιας τοποθεσίας που να βρίσκεται κοντά στο χρήστη και να ικανοποιεί κάποια συγκεκριμένη επιθυμία του (εστιατόριο, μουσείο, μηχάνημα ανάληψης), είναι πολύ συνηθισμένες στις μέρες μας. Έχει ακόμα πολλές χρήσεις, όπως στην κοινωνική δικτύωση αλλά και σε υπηρεσίες αναψυχής.

Το 2000, συστήθηκε από την Ericsson, τη Motorola και τη Nokia το Location Interoperability Forum (LIF) το οποίο κάνει τα πρώτα βήματα ώστε να καθιερώσει τα πρωτόκολλα που είναι απαραίτητα για τη λειτουργία των συγκεκριμένων υπηρεσιών. Ένα χρόνο νωρίτερα είχε κυκλοφορήσει συσκευή Palm VII, η πρώτη φορητή συσκευή, που ήταν συμβατή με υπηρεσίες τοποθεσίας.

Οι σύγχρονες συσκευές, είναι εξοπλισμένες με GPS (Global Positioning System). Οι εφαρμογές κάνοντας χρήση του GPS, μπορούν να γνωρίζουν την τρέχουσα τοποθεσία της συσκευής και να ανταποκρίνονται αναλόγως. Το GPS είναι ένα σύστημα δορυφορικής

πλοήγησης που παρέχει πληροφορίες τοποθεσίας και χρόνου, σε οποιοδήποτε σημείο της Γης «βλέπει» τουλάχιστον 4 δορυφόρους. Είναι ιδιαιτέρως χρήσιμο σε στρατιωτικές και πολιτικές εφαρμογές ανά τον κόσμο. Αναπτύχθηκε το 1973 από τις Ηνωμένες Πολιτείες της Αμερικής και είναι χωρίς χρέωση σε όσους διαθέτουν τον κατάλληλο GPS λήπτη. Η ακρίβεια είναι αρκετά καλή και μπορεί να φτάσει και τα 4 μέτρα, όμως εξαρτάται πολύ από την ορατότητα που έχει η συσκευή στους δορυφόρους

Στις σύγχρονες συσκευές, στις υπηρεσίας συμμετέχουν και τα δίκτυα κινητών τηλεπικοινωνιών, το GSM αλλά και τα δίκτυα τρίτης (3G) και τέταρτης (4G) γενιάς. Κάνοντας χρήση των ισχυρών αυτών δικτύων, και των πληροφοριών από τους σταθμούς βάσης, βελτιώνεται πάρα πολύ η ακρίβεια καθώς και η ταχύτητα με την οποία εντοπίζεται η εκάστοτε τοποθεσία. Αυτό μπορεί να αποδειχθεί πολύ χρήσιμο σε πολλές εφαρμογές που απαιτούν ακρίβεια αλλά και ταχύτητα.

Κάποιες από τις πιο συνηθισμένες υπηρεσίες τοποθεσίας είναι:

- Να προτείνει κοινωνικές εκδηλώσεις σε μία περιοχή ή πόλη.
- Να προτείνει μία κοντινή επιχείρηση ή υπηρεσία, όπως φαρμακείο, εστιατόριο, καφετέρια.
- Εύρεση κάποιου μέσου μεταφοράς σε κοντινή απόσταση.
- Εύρεση νοσοκομείου ή φαρμακείου σε κοντινή απόσταση.
- Πλοήγηση σε οποιαδήποτε τοποθεσία.
- Ξενάγηση σε εσωτερικούς χώρους, όπως μουσεία.
- Εμφάνιση προειδοποιήσεων, όπως προειδοποίηση για κίνηση σε κάποια οδό.
- Προβολή διαφημίσεων βάσει της τοποθεσίας.
- Πρόγνωση του καιρού στην τοποθεσία που βρισκόμαστε.
- Εντοπισμός ανθρώπων μέσα από τα κοινωνικά δίκτυα.
- Εντοπισμός χαμένης συσκευής.

Οι υπηρεσίες τοποθεσίας έχουν γίνει αναπόσπαστο κομμάτι της καθημερινότητας μας. Μας βοηθούν να μετακινούμαστε ευκολότερα στην περιοχή μας, αλλά και αν βρεθούμε σε μία άλλη πόλη η χώρα. Μας προτείνουν μέρη και εκδηλώσεις να επισκεφτούμε, μας προειδοποιούν, μας ενημερώνουν. Οι υπηρεσίες αυτές είναι πολύ χρήσιμες και θα γίνονται συνεχώς και πιο απαραίτητες.

Εικόνα 4 Η συσκευή Palm VII, η πρώτη φορητή συσκευή που ήταν συμβατή με υπηρεσίες τοποθεσίας

1.4. Crowdsourcing

Crowdsourcing, σύμφωνα με το Merriam-Webster Dictionary, είναι η πρακτική της απόκτησης αναγκαίων υπηρεσιών, ιδεών ή περιεχομένου, προσελκύοντας συνεισφορές από μία μεγάλη ομάδα ανθρώπων, και ιδιαίτερα από μια online κοινότητα παρά από τους συνηθισμένους εργαζόμενους ή προμηθευτές. Συχνά, χρησιμοποιείται για την υποδιαίρεση κάποιας κουραστικής εργασίας, ή για να χρηματοδοτήσει νεοσύστατες εταιρίες και φιλανθρωπικές οργανώσεις, τόσο online, όσο και offline. Συνδυάζει την προσπάθεια του πλήθους των αναπροσδιοριζόμενων εθελοντών, όπου ο καθένας προσθέτει ένα μικρό τμήμα συνδυάζοντας ένα μεγαλύτερο αποτέλεσμα.

Το crowdsourcing στις μέρες μας, έχει μεταφερθεί κυρίως στο διαδίκτυο. Το διαδίκτυο παρέχει ένα καλό μέρος για crowdsourcing δεδομένου ότι οι άνθρωποι τείνουν να είναι πιο ανοιχτοί σε web-based έργα, όπου δεν κρίνονται ή ελέγχονται για τη σωματική τους συνεισφορά, και έτσι αισθάνονται πιο άνετα να βοηθούν. Αυτό επιτρέπει τη δημιουργία καλών έργων, επειδή τα άτομα που συμμετέχουν σε ένα online περιβάλλον, θα δίνουν περισσότερη προσοχή στο έργο, παρά στην επικοινωνία με άλλα άτομα.

Το crowdsourcing μπορεί να είναι είτε ρητό, είτε σιωπηρό. Ρητό crowdsourcing είναι όταν οι χρήστες δουλεύουν από κοινού, για την κατασκευή διαφόρων έργων, ενώ σιωπηρό, όταν οι χρήστες λύνουν ένα πρόβλημα ως αποτέλεσμα κάποιας άλλης ενέργειας που κάνουν.

Το crowdsourcing χρησιμοποιείται πολύ και σε εφαρμογές των «έξυπνων» κινητών τηλεφώνων αλλά και σε ιστοσελίδες. Για παράδειγμα, μια ιστοσελίδα ή μια εφαρμογή που έχει περιγραφές θεατρικών παραστάσεων και ζητάει από το κοινό που την χρησιμοποιεί να κρίνει και να βαθμολογήσει τις παραστάσεις, είναι ένα απλό παράδειγμα ρητού crowdsourcing. Όλοι οι χρήστες που βαθμολογούν μια παράσταση, κάνουν αυτό ακριβώς που τους έχει ζητηθεί, να αξιολογήσουν τις παραστάσεις. Αυτό έχει σαν αποτέλεσμα ένας νέος χρήστης να μπορέσει να δει τις αξιολογήσεις των παραστάσεων, από ένα μεγάλο αριθμό ατόμων. Αυτό το είδος έχει εφαρμογή και σε αξιολογήσεις κινηματογραφικών έργων, βιβλίων, αλλά ακόμη και εστιατορίων και άλλων επιχειρήσεων.

Ένα παράδειγμα σιωπηρού crowdsourcing είναι στο κοινωνικό δίκτυο Facebook, όπου δίνεται η επιλογή στους χρήστες να δηλώσουν ότι τους αρέσει (like) μια δημοσίευση ή να την αναφέρουν ως υβριστική (report as abuse). Με αυτόν τον τρόπο λύνουν ένα πρόβλημα των υπαλλήλων του Facebook, οι οποίοι είναι αδύνατο να παρακολουθούν τις δημοσιεύσεις των 1.1 δισεκατομμυρίων χρηστών. Έτσι, αντί ο έλεγχος να πραγματοποιείται από υπαλλήλους της εταιρίας γίνεται μέσω των ίδιων των χρηστών.

Καταλαβαίνουμε ότι το crowdsourcing μπορεί να παίξει καθοριστικό ρόλο σε μία εφαρμογή, διευκολύνοντας όχι μόνο τη δουλειά των δημιουργών της, αλλά παρέχοντας και πλουσιότερο υλικό στους άλλους χρήστες. Είναι λογικό λοιπόν που χρησιμοποιούνται τέτοιους είδους τεχνικές.

1.5. Αντικείμενο διπλωματικής εργασίας

Το αντικείμενο της διπλωματικής εργασίας είναι ο σχεδιασμός, η ανάπτυξη και η υλοποίηση εφαρμογής στην πλατφόρμα του iOS της Apple, που θα χρησιμοποιηθεί ως προσωπικός βοηθός του χρήστη, οργανώνοντας το πρόγραμμα του. Η εφαρμογή θα μπορεί να οργανώνει και να προγραμματίζει τις δραστηριότητες του χρήστη ενώ αναλαμβάνει και το ρόλο να του υπενθυμίζει τις δραστηριότητες που έχει προγραμματίσει, είτε χρονικά είτε λαμβάνοντας υπ' όψιν την τρέχουσα τοποθεσία του. Επίσης, θα υπάρχει κάποιος τρόπος έμμεσης επικοινωνίας μεταξύ των χρηστών, αφού, μέσω server θα μπορούν να χρησιμοποιήσουν μία τοποθεσία που έχει προσθέσει κάποιος άλλος χρήστης.

Για την περάτωση της εργασίας αυτής, αρχικά θα πρέπει να εξοικειωθούμε με τη χρήση του λειτουργικού συστήματος iOS. Γνωρίζοντας το πως λειτουργούν οι πιο δημοφιλείς εφαρμογές και ποια είναι η επιθυμητή εμπειρία χρήσης, θα μπορέσουμε να σχεδιάσουμε την εφαρμογή ώστε να ταιριάζει με τη λογική των εφαρμογών αυτών και η εμπειρία να είναι ανάλογη αυτής που έχουν συνηθίσει οι χρήστες.

Αφού η ιδέα υπάρχει και έχει γίνει η σχεδίαση της εφαρμογής, θα πρέπει να έρθουμε σε επαφή με τα εργαλεία ανάπτυξης των εφαρμογών. Τα εργαλεία αυτά είναι το Xcode και το iOS SDK. Το Xcode είναι το εργαλείο ανάπτυξης εφαρμογών για όλες τις πλατφόρμες της Apple, το iOS και το OS X, το λειτουργικό σύστημα των υπολογιστών της, ενώ το iOS SDK περιλαμβάνει όλα τα απαραίτητα πλαίσια (frameworks) για την ανάπτυξη ακόμη και της πιο σύνθετης εφαρμογής. Επίσης, θα χρησιμοποιηθούν εργαλεία ελέγχου της λειτουργίας της εφαρμογής, όπως είναι το iOS Simulator και το Instruments. Ο server θα αναπτυχθεί στο περιβάλλον του Eclipse.

Φυσικά, απαραίτητη είναι η γνώση αντικειμενοστραφούς προγραμματισμού. Τα frameworks του iOS SDK απαιτούν καλή γνώση της γλώσσας Objective C 2.0. Απαιτείται επίσης γνώση της γλώσσας προγραμματισμού Java αφού με τη βοήθεια αυτής θα αναπτυχθεί ο server.

1.6. Διάρθρωση της εργασίας

Στα πλαίσια της διπλωματικής εργασίας, προσπαθήσαμε να προσεγγίσουμε τη δημιουργία της εφαρμογής ακολουθώντας με σωστό τρόπο, τα βήματα που προβλέπονται. Τα βήματα αυτά παρουσιάζονται στα κεφάλαια της διπλωματικής. Στο δεύτερο κεφάλαιο, παρουσιάζονται οι τεχνολογίες τις οποίες πρέπει να γνωρίζει κάποιος, προκειμένου να υλοποιήσει μία τέτοιου είδους εφαρμογή. Στο τρίτο κεφάλαιο, γίνεται η περιγραφή των απαιτήσεων που πρέπει να ικανοποιεί η εφαρμογή και η ανάλυση των λειτουργιών και των σεναρίων χρήσης. Στη συνέχεια, το τέταρτο κεφάλαιο αναφέρεται στη σχεδίαση της εφαρμογής και του server. Έπειτα, το πέμπτο κεφάλαιο αφορά την υλοποίηση της εφαρμογής και παρουσιάζεται ένα πλήρες σενάριο χρήσης της. Τέλος, στο έκτο κεφάλαιο, καταγράφονται τα συμπεράσματα που προέκυψαν κατά την περάτωση της διπλωματικής εργασίας και προτείνονται πιθανές μελλοντικές επεκτάσεις.

2

Τεχνολογίες

2.1. Apple iOS

Το iOS είναι το λειτουργικό σύστημα (Operating System, OS) που τρέχει στις φορητές συσκευές της Apple, το iPhone, το iPod και το iPad. Το λειτουργικό σύστημα διαχειρίζεται το υλικό (hardware) της συσκευής και παρέχει τις τεχνολογίες που είναι απαραίτητες για τη λειτουργία των εγκατεστημένων εφαρμογών. Η συσκευή πωλείται έχοντας προεγκατεστημένες στο λειτουργικό της σύστημα κάποιες βασικές εφαρμογές, όπως το Τηλέφωνο, το Mail (εφαρμογή ηλεκτρονικού ταχυδρομείου) και το Safari (περιηγητής ιστού) που παρέχουν τις τυπικές υπηρεσίες του συστήματος στο χρήστη.

Το iOS Software Development Kit (SDK) περιέχει τα εργαλεία και τις διεπαφές που απαιτούνται για την ανάπτυξη, εγκατάσταση, λειτουργία και τη δοκιμή των εφαρμογών που εμφανίζονται στην αρχική οθόνη μιας iOS συσκευής. Όλες οι εφαρμογές κατασκευάζονται χρησιμοποιώντας τα πλαίσια (frameworks) του συστήματος του iOS και την αντικειμενοστραφή γλώσσα προγραμματισμού Objective-C. 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 iOS SDK είναι απαραίτητη για να σχεδιασθεί και να υλοποιηθεί αποτελεσματικά μία εφαρμογή.

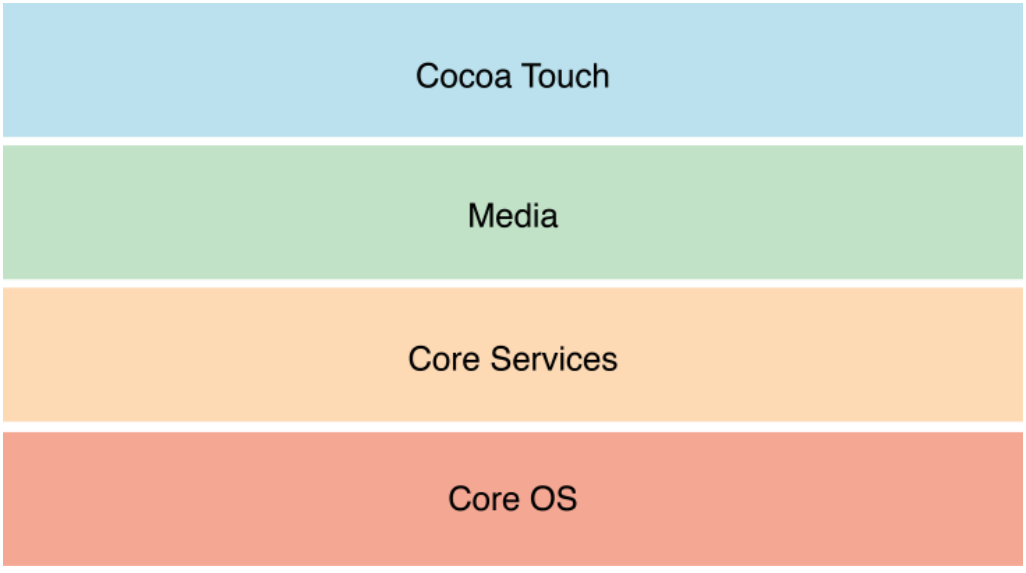
Τη στιγμή που γράφεται η εργασία το λειτουργικό σύστημα iOS βρίσκεται στην έκτη του έκδοση (iOS 6.1.4) με την έβδομη έκδοση (iOS 7) να βρίσκεται σε μορφή beta. Η πλατφόρμα θα αναλυθεί σύμφωνα με αυτά που ισχύουν στην έκτη έκδοση του λειτουργικού συστήματος.

2.1.1. Η διαστρωματωμένη αρχιτεκτονική του iOS

Στο υψηλότερο επίπεδο, το iOS ενεργεί ως μεσάζων μεταξύ του υποκείμενου hardware και των εφαρμογών που εμφανίζονται στην οθόνη. Οι εφαρμογές σπάνια επικοινωνούν άμεσα με το hardware της συσκευής. Αντιθέτως, η επικοινωνία γίνεται μέσω μιας σειράς καλά καθορισμένων διεπαφών του συστήματος, γεγονός που προστατεύει την εφαρμογή από αλλαγές στο hardware. Αυτό σημαίνει ότι οι εφαρμογές μπορούν να λειτουργούν σε συσκευές με διαφορετικό hardware. Οι τεχνολογίες του iOS μπορούν να συγκεντρωθούν σε τέσσερα στρώματα:

1. Το στρώμα Cocoa Touch
2. Το στρώμα Media
3. Το στρώμα Core Services
4. Το στρώμα Core OS

Στα κατώτερα στρώματα του συστήματος βρίσκονται οι θεμελιώδεις υπηρεσίες και τεχνολογίες, στις οποίες βασίζονται όλες οι εφαρμογές, ενώ τα στρώματα υψηλότερου επιπέδου περιέχουν πιο εξελιγμένες υπηρεσίες και τεχνολογίες.



Εικόνα 5 Η Διαστρωματωμένη Αρχιτεκτονική του iOS.

Η Apple προσφέρει τις περισσότερες από τις διεπαφές του συστήματος, οργανωμένες σε ειδικά πακέτα που ονομάζονται πλαίσια (frameworks). Ένα πλαίσιο είναι ένας κατάλογος που περιέχει μια δυναμική κοινή βιβλιοθήκη και τους πόρους (όπως αρχεία επικεφαλίδας, εικόνες) που χρειάζονται για την υποστήριξη της. Προκειμένου να χρησιμοποιηθεί κάποιο πλαίσιο απαιτείται η σύνδεση της εφαρμογής με αυτό, όπως γίνεται με οποιαδήποτε κοινή βιβλιοθήκη. Η σύνδεση αυτή δίνει την απαραίτητη πρόσβαση στα χαρακτηριστικά του πλαισίου.

Πολλές από τις τεχνολογίες του iOS είναι κοινές με αυτές του OS X, του λειτουργικού συστήματος των υπολογιστών της Apple. Οι μεγαλύτερες διαφορές βρίσκονται στο επίπεδο της διεπαφής χρήστη, αλλά ακόμη και εκεί υπάρχουν ομοιότητες.

Τέλος, για την ανάπτυξη των εφαρμογών, χρησιμοποιείται το εργαλείο Xcode. Το Xcode είναι ένα περιβάλλον ανάπτυξης που χρησιμοποιείται για τη δημιουργία, τη δοκιμή και την αποσφαλμάτωση των εφαρμογών. Το Xcode, περιλαμβάνει ακόμα δύο εργαλεία, το Instruments και το iOS Simulator. Το Xcode χρησιμοποιείται για τη συγγραφή του κώδικα της εφαρμογής, η οποία μπορεί να εκτελεστεί στο iOS Simulator ή απευθείας σε μία συνδεδεμένη iOS συσκευή. Το iOS Simulator είναι ένας προσομοιωτής των iOS συσκευών. Το εργαλείο Instruments χρησιμοποιείται για τον έλεγχο της απόδοσης της εφαρμογής. Τέλος, παρέχεται στον προγραμματιστή ένα ακόμη εργαλείο, η Developer Library. Η Developer Library αποτελεί ένα πολύ χρήσιμο εργαλείο, αφού περιέχει όλες τις απαραίτητες πληροφορίες των τεχνολογιών και των πλαισίων.

Το στρώμα Cocoa Touch

Το στρώμα Cocoa Touch περιέχει τα βασικά πλαίσια που είναι απαραίτητα για τη δημιουργία εφαρμογών στο iOS. Αυτό το στρώμα καθορίζει τη βασική υποδομή της εφαρμογής και την υποστήριξη των βασικών τεχνολογιών, όπως η πολυδιεργασία (multitasking), η τεχνολογία της οθόνης αφής, οι γνωστοποιήσεις τύπου push και πολλές άλλες υπηρεσίες υψηλού επιπέδου. Κατά το σχεδιασμό της εφαρμογής, θα πρέπει να διερευνηθεί αν οι τεχνολογίες του στρώματος αυτού ανταποκρίνονται στις ανάγκες της εφαρμογής.

Κάποιες από τις βασικές τεχνολογίες που είναι διαθέσιμες στο συγκεκριμένο στρώμα είναι:

- Η αυτόματη διάταξη των στοιχείων της διεπαφής χρήστη.
- Η υποστήριξη των Storyboards, που βοηθούν στη σχεδίαση ολόκληρης της διεπαφής χρήστη σε ένα μέρος όπου μπορούμε να ελέγξουμε όλες τις οθόνες και τις συνδέσεις μεταξύ τους.
- Η υποστήριξη εγγράφων.
- Η πολυδιεργασία (multitasking).
- Η εκτύπωση, που επιτρέπει την αποστολή περιεχομένου για εκτύπωση σε κοντινούς εκτυπωτές.
- Η υπηρεσία γνωστοποιήσεων.
- Η αναγνώριση χειρονομιών.
- Οι υπηρεσίες μεταξύ ομότιμων χρηστών (Peer to Peer).

- Η υποστήριξη πρότυπων οθονών του συστήματος, όπως η οθόνη σύνθεση μηνύματος ή η οθόνη προβολής των πληροφοριών μιας επαφής.

Τα πλαίσια που περιέχονται στο στρώμα αυτό είναι τα:

- *Address Book UI Framework*
- *Event Kit UI Framework*
- *Game Kit Framework*
- *iAd Framework*
- *Map Kit Framework*
- *Message UI Framework*
- *Twitter Framework*
- *UI Kit Framework*

Το στρώμα Media

Το στρώμα Media περιέχει τις τεχνολογίες γραφικών, ήχου και βίντεο και είναι προσανατολισμένο στη δημιουργία εφαρμογών πολυμέσων με πολύ καλή εμπειρία. Οι τεχνολογίες αυτές είναι πολύ εύκολες στη χρήση. Τα βασικά πλαίσια που περιέχονται στη στρώμα αυτό είναι τα:

- *Core Audio Framework*
- *Core Graphics Framework*
- *Core Image Framework*
- *Core Text Framework*
- *Core Video Framework*
- *GLKit Framework*
- *Media Player Framework*
- *OpenGL ES Framework*
- *Quartz Core Framework*

Το στρώμα Core Services

Το στρώμα Core Services περιέχει τις θεμελιώδεις υπηρεσίες συστήματος τις οποίες χρησιμοποιούν όλες οι εφαρμογές. Ακόμα και αν δε χρησιμοποιούνται άμεσα από την εφαρμογή, πολλά μέρη του συστήματος είναι φτιαγμένα «πάνω» σε αυτές τις υπηρεσίες. Κάποιες από τις βασικές τεχνολογίες είναι:

- Η υπηρεσία iCloud. Η Apple προσφέρει στους χρήστες της μία cloud υπηρεσία, το iCloud. Η υπηρεσία αυτή δίνει τη δυνατότητα στο χρήστη να αποθηκεύει τα

δεδομένα του σε μία κεντρική τοποθεσία, στην οποία μπορεί να έχει πρόσβαση από όλες τις iOS συσκευές του και από τον υπολογιστή του.

- Η υποστήριξη block αντικειμένων.
- Η προστασία ευαίσθητων δεδομένων του χρήστη.
- Το Grand Central Dispatch που επιτρέπει τη διαχείριση της εκτέλεσης των εργασιών της εφαρμογής.
- Η υποστήριξη της βιβλιοθήκης SQLite.

Τα βασικά πλαίσια που βρίσκονται στο στρώμα αυτό είναι τα:

- *Accounts Framework*
- *Ad Support Framework*
- *Core Data Framework*
- *Core Foundation Framework*
- *Core Location Framework*
- *Core Media Framework*
- *Core Motion Framework*
- *Core Telephony Framework*
- *Foundation Framework*
- *Social Framework*

Το στρώμα Core OS

Το στρώμα Core OS περιέχει όλα τα χαρακτηριστικά στα οποία βασίζονται οι περισσότερες άλλες τεχνολογίες. Ακόμα κι αν δε χρησιμοποιούνται άμεσα από την εφαρμογή, χρησιμοποιούνται από άλλα πλαίσια. Στην περίπτωση που χρειάζεται ασφάλεια ή επικοινωνία με κάποιο εξωτερικό hardware, χρησιμοποιούνται τα πλαίσια αυτού του στρώματος. Τα βασικά πλαίσια που υπάρχουν σε αυτό το στρώμα είναι τα:

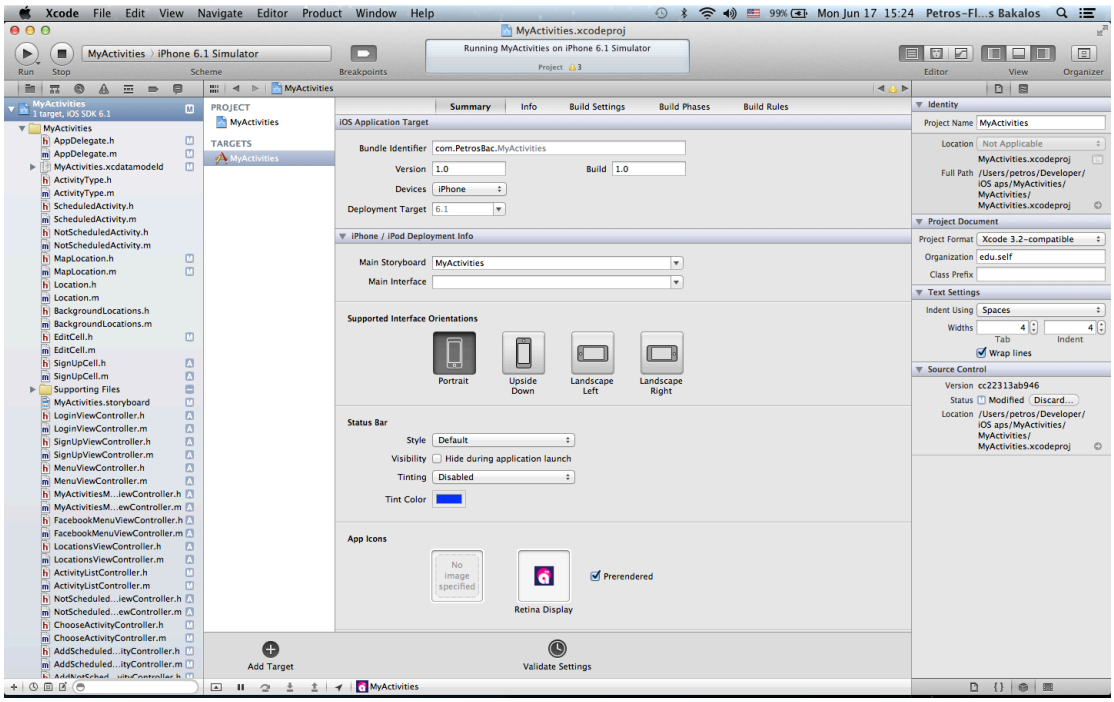
- *Accelerate Framework*
- *Core Bluetooth Framework*
- *Generic Security Framework*
- *Security Framework*
- *System*

2.1.2. Τα εργαλεία ανάπτυξης εφαρμογών

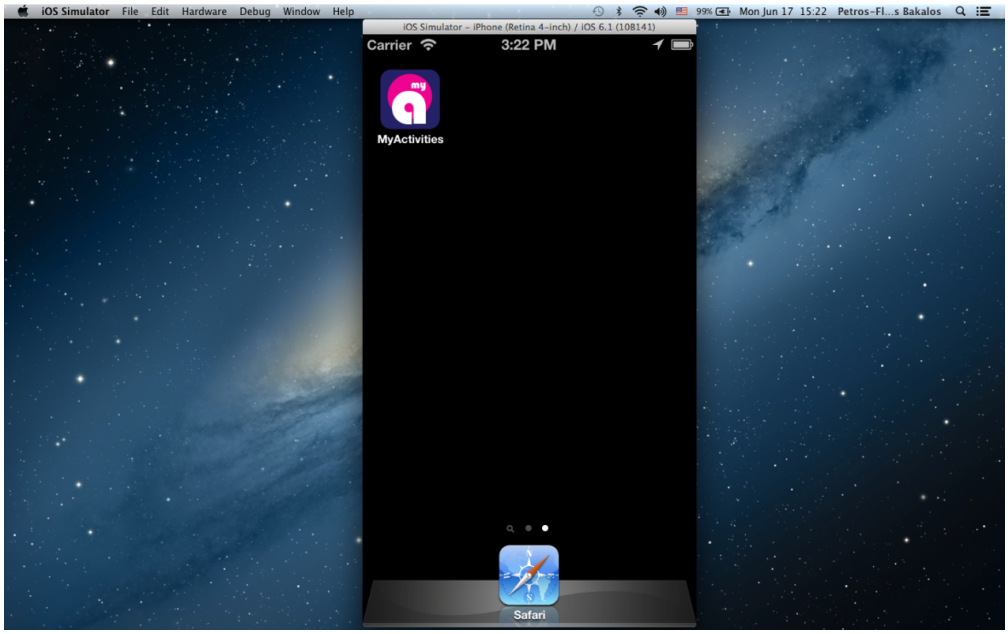
Για την ανάπτυξη εφαρμογών για τις iOS συσκευές, ο προγραμματιστής πρέπει να έχει έναν Intel-based Macintosh υπολογιστή και τα εργαλεία του Xcode. Το Xcode είναι μία σουίτα εργαλείων ανάπτυξης της Apple που προσφέρει στον προγραμματιστή υποστήριξη για τη διαχείριση των εφαρμογών, την επεξεργασία κώδικα, την αποσφαλμάτωση, τη ρύθμιση της απόδοσης της εφαρμογής και πολλά άλλα. Στο κέντρο της σουίτας βρίσκεται το Xcode, το οποίο προσφέρει το βασικό περιβάλλον ανάπτυξης κώδικα.

Το Xcode

Το Xcode είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment, IDE) που παρέχει όλα τα εργαλεία που χρειάζονται για τη δημιουργία και διαχείριση των εφαρμογών, τη σχεδίαση και την υλοποίηση της διεπαφής χρήστη, την αποσφαλμάτωση του κώδικα και πολλά άλλα. Όταν ο προγραμματιστής επιλέγει να «τρέξει» την εφαρμογή του, έχει δύο επιλογές, να χρησιμοποιήσει το iOS Simulator ή να χρησιμοποιήσει μία iOS συσκευή. Ο iOS Simulator είναι ένας προσομοιωτής των iOS συσκευών και παρέχει στον προγραμματιστή τη δυνατότητα να βεβαιωθεί ότι η εφαρμογή του συμπεριφέρεται όπως θα ήθελε. Αν η εφαρμογή λειτουργεί ικανοποιητικά, μπορεί να δοκιμαστεί σε μία iOS συσκευή που είναι συνδεδεμένη με τον υπολογιστή. Η δοκιμή στη συσκευή αποτελεί και το καλύτερο περιβάλλον δοκιμής.



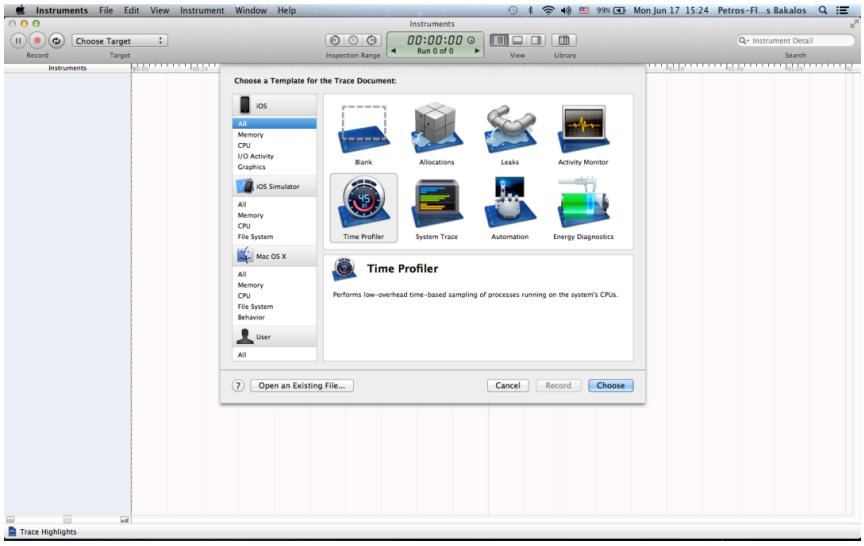
Εικόνα 6 Το περιβάλλον του Xcode.



Εικόνα 7 Χρήση του iOS Simulator για τον έλεγχο της συμπεριφοράς της εφαρμογής.

To Instruments

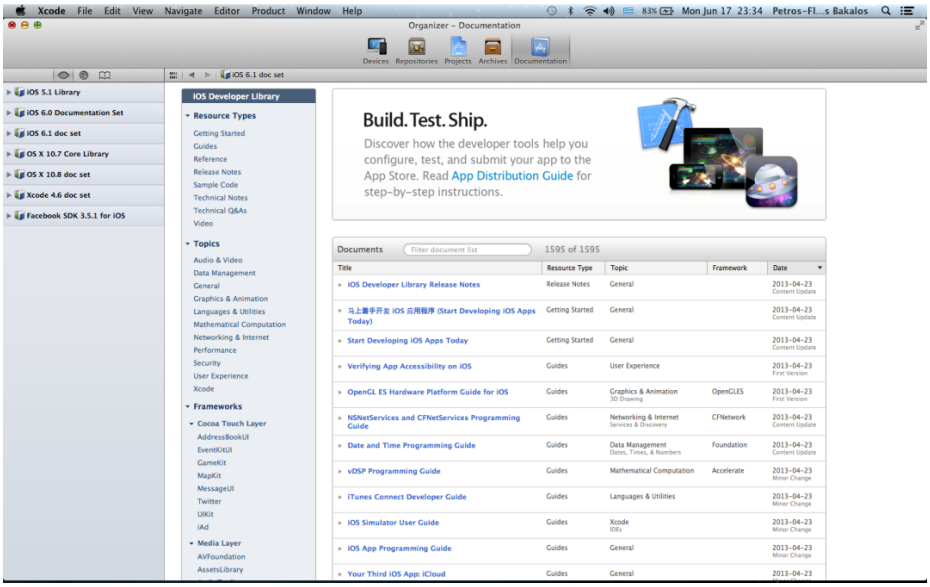
Για να εξασφαλιστεί η καλύτερη δυνατή εμπειρία χρήσης για την εφαρμογή, το περιβάλλον του Instruments επιτρέπει στον προγραμματιστή να αναλύσει την απόδοση της εφαρμογής του, όταν αυτή λειτουργεί στο iOS Simulator ή στη συσκευή που είναι συνδεδεμένη στον υπολογιστή. Το Instruments συγκεντρώνει δεδομένα από τη λειτουργία της εφαρμογής και τα παρουσιάζει σε ένα χρονοδιάγραμμα. Τα δεδομένα που συλλέγονται είναι σχετικά με τη χρήση της μνήμης της εφαρμογής, τη δραστηριότητα του δίσκου, τη δραστηριότητα του δικτύου καθώς και την απόδοση των γραφικών. Το χρονοδιάγραμμα μπορεί να εμφανίζει όλα τα είδη πληροφορίας, επιτρέποντας στον προγραμματιστή να συσχετίσει τη συνολική συμπεριφορά της εφαρμογής και όχι μόνο σε ένα συγκεκριμένο σημείο. Τέλος, το Instruments προσφέρει τη δυνατότητα αποθήκευσης των δεδομένων, ώστε να μπορεί να αναλυθεί η συμπεριφορά της εφαρμογής με την πάροδο του χρόνου.



Εικόνα 8 Η αρχική οθόνη του Instruments.

Η Developer Library

Η iOS Developer Library περιέχει όλα τα έγγραφα (documentation), παραδείγματα κώδικα, βοηθήματα (tutorials) και άλλες πληροφορίες που χρειάζονται για τη συγγραφή iOS εφαρμογών. Επειδή η Developer Library περιέχει τεράστιο αριθμό σελίδων, χρησιμοποιούνται πολλές τεχνικές οργάνωσης τους, με αποτέλεσμα να γίνεται ευκολότερα η αναζήτηση. Η πρόσβαση στην Developer Library μπορεί να γίνει είτε από την ιστοσελίδα του Apple Developer, είτε από το Xcode. Όταν εγκαθίστανται το iOS SDK, το Xcode κάνει αυτόματα την iOS Developer Library διαθέσιμη στον προγραμματιστή καθώς και την ενημερώνει αυτόματα.



Εικόνα 9 Το περιβάλλον της iOS Developer Library.

2.2. Java

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

Οι δημιουργοί της Java την έχουν χαρακτηρίσει ως μία γλώσσα "απλή, αντικειμενοστραφή, διαμοιραζόμενη, εύρωστη, ασφαλή, με ουδέτερη αρχιτεκτονική, εύκολη στη μεταφορά, υψηλής απόδοσης και πολυνηματική".

Ένα από τα βασικά χαρακτηριστικά της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος της πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix, Macintosh, (σύντομα θα τρέχουν και σε Playstation και σε άλλες κονσόλες παιχνιδιών) χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε λειτουργικό σύστημα. Για να επιτευχθεί αυτό χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα γραμμένα σε Java να μπορούν να είναι «κατανοητά» από κάθε υπολογιστή ανεξάρτητα από το είδος του επεξεργαστή και 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 , δηλαδή σε κώδικα 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, η εικονική μηχανή (Java Virtual Machine) που πρέπει να είναι εγκατεστημένη 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. Πρέπει να σημειωθεί ότι η JVM (εικονική μηχανή) 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. 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) 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 πολύ συχνά τα σφάλματα προγραμμάτων που οφείλονται σε λανθασμένο χειρισμό της μνήμης.

2.2.3. Επιδόσεις

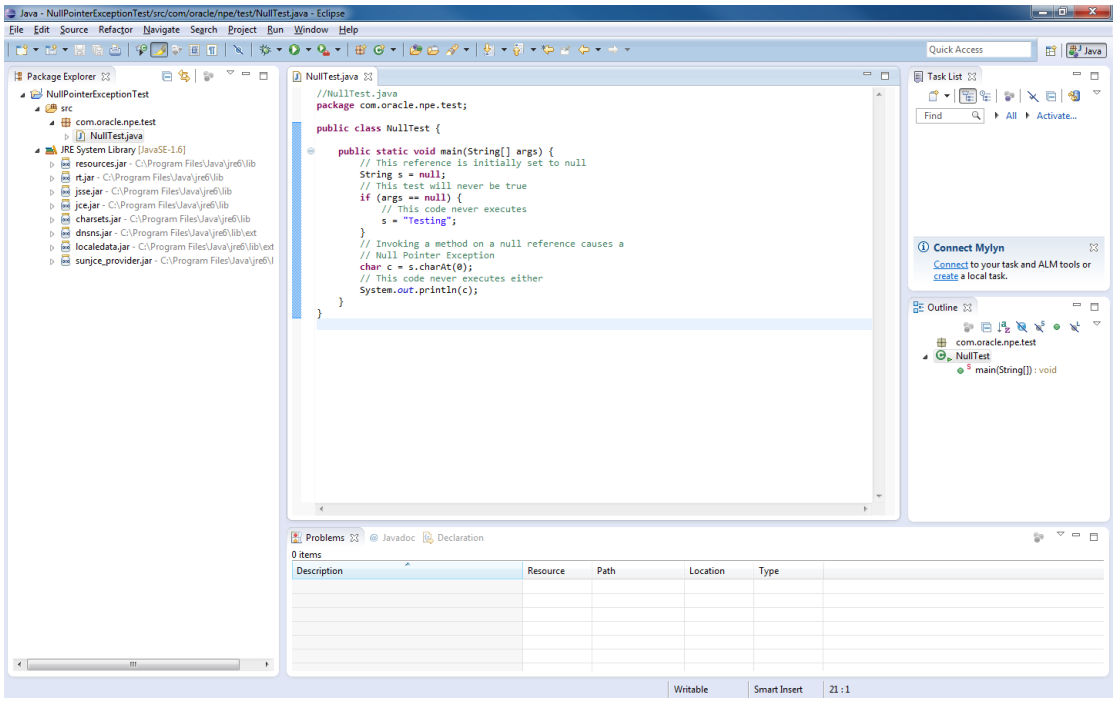
Παρόλο που η εικονική μηχανή προσφέρει πολλά πλεονεκτήματα, η Java αρχικά ήταν πιο αργή σε σχέση με άλλες προγραμματιστικές γλώσσες υψηλού επιπέδου όπως η C και η C++. Εμπειρικές μετρήσεις στο παρελθόν είχαν δείξει ότι η C++ μπορούσε να είναι αρκετές φορές γρηγορότερη από την Java. Ωστόσο, γίνονται προσπάθειες 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.

Οι τελευταίες εκδόσεις του javac με τη χρήση της τεχνολογίας Hot Spot έχουν καταφέρει αξιόλογες επιδόσεις που πλησιάζουν ή και ξεπερνούν σε μερικές περιπτώσεις τον εγγενή κώδικα.

2.2.4. Το περιβάλλον ανάπτυξης εφαρμογών Eclipse

Το Eclipse είναι ένα περιβάλλον προγραμματισμού συμβατό 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 γραφεί από προγραμματιστές Java και προορίζεται για αυτή τη γλώσσα προγραμματισμού. Είναι φυσικά ένα πολύ χρήσιμο εργαλείο, γι' αυτό και είναι τόσο διαδεδομένη η χρήση του. Ακόμα και αν η κύρια γλώσσα προγραμματισμού που χρησιμοποιεί κάποιος δεν είναι η Java, δεν είναι λίγες οι περιπτώσεις που το Eclipse μπορεί να «λύσει» τα χέρια του προγραμματιστή.



Εικόνα 10 Το περιβάλλον ανάπτυξης εφαρμογών Eclipse.

2.3. JSON

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

Το JSON είναι χτισμένο πάνω σε δύο δομές:

- Ένα σύνολο από ζεύγη ονόματος-τιμής. Στις διάφορες γλώσσες αυτό μεταφράζεται ως αντικείμενο, record, struct, dictionary, πίνακας ή λίστα.
- Μία 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 ένα τρόπο ή τον άλλον. Αυτό φυσικά είναι και το μεγάλο πλεονέκτημα του JSON. Το γεγονός δηλαδή ότι μέσω του JSON τα δεδομένα μπορούν να πάρουν μία μορφή που είναι αναγνώσιμη από πολλές διαφορετικές γλώσσες μπορεί να λύσει σημαντικά προβλήματα που αφορούν στη μεταφορά δεδομένων.

2.4. Hibernate

Το Hibernate Framework είναι λογισμικό ανοιχτού κώδικα που σκοπό έχει να συνδέσει τα αντικείμενα που δημιουργούνται στη 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 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, παρόλο 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.

3

Ανάλυση

3.1. Απαιτήσεις εφαρμογής προσωπικού βοηθού

Ο ρυθμός της καθημερινότητας του ανθρώπου έχει ανέβει εκθετικά τα τελευταία χρόνια. Οι υποχρεώσεις, από εργασιακές μέχρι και κοινωνικές, συνεχώς πληθαίνουν με αποτέλεσμα, να είναι αδύνατο να ανταπεξέλθουμε στους ρυθμούς αυτούς. Ο άνθρωπος της εποχής μας είναι καθολικός, δηλαδή μπορεί και θέλει να ασχολείται με όλες τις πτυχές τις ζωής του. Για να το πετύχει αυτό πρέπει να «στριμώξει» στην καθημερινότητα του αμέτρητες υποχρεώσεις οι οποίες μπορεί να αφορούν επαγγελματικές ή προσωπικές ανάγκες. Ο σύγχρονος άνθρωπος όμως, πρέπει να διασκεδάσει και να αναψυχθεί, μέσω κοινωνικών εκδηλώσεων ή κάποιας άλλης δραστηριότητας. Δημιουργείται λοιπόν η ανάγκη για εργαλεία, τα οποία να βοηθούν στην οργάνωση του προγράμματος μας.

Οι «έξυπνες» συσκευές μας, έχουν τη δυνατότητα να διαθέτουν στο χρήστη τέτοια εργαλεία, μέσω των εφαρμογών τους. Τις συσκευές τις έχουμε συνεχώς μαζί μας, οπότε μπορούμε να έχουμε πρόσβαση, με λίγες μόνο κινήσεις στις εφαρμογές αυτές. Γίνεται λοιπόν αντιληπτό, ότι μία εφαρμογή τέτοιου είδους μπορεί να φανεί ιδιαιτέρως χρήσιμη στους χρήστες της. Αυτό το είδος των εφαρμογών μπορεί να χαρακτηριστεί σαν ένα είδος προσωπικού βοηθού.

Για να είναι χρήσιμο ένα τέτοιο εργαλείο, θα πρέπει να πληροί κάποιες βασικές προδιαγραφές. Μερικές από αυτές μπορεί να είναι:

- Υπενθύμιση μιας δραστηριότητας βάσει του χρόνου. Θα πρέπει ο χρήστης να μπορεί να προγραμματίσει τις υποχρεώσεις του και η εφαρμογή να φροντίσει αυτό να τηρηθεί.
- Υπενθύμιση μιας δραστηριότητας βάσει της τοποθεσίας. Κάνοντας χρήση των υπηρεσιών τοποθεσίας, ένας προσωπικός βοηθός θα πρέπει να ενημερώνει το χρήστη για τις δραστηριότητες που μπορεί να πραγματοποιήσει στην κοντινή του περιοχή.

- Μια τέτοια εφαρμογή θα πρέπει να έχει και κάποια αλληλεπίδραση με άλλους χρήστες, άμεσα ή έμμεσα μέσω κάποιου server.
- Να είναι εύχρηστη και ευχάριστη κατά τη χρήση.

3.2. Ο προσωπικός βοηθός MyActivities

Η εφαρμογή που αναπτύσσουμε στη παρούσα διπλωματική εργασία ονομάστηκε MyActivities. Είναι μία εφαρμογή που θα λειτουργεί σαν προσωπικός βοηθός παρέχοντας στο χρήστη τη δυνατότητα να οργανώσει το πρόγραμμα του και να το εκτελέσει. Θα πρέπει να ικανοποιεί κάποιες βασικές λειτουργίες, όπως αυτές που αναπτύχθηκαν παραπάνω στις απαιτήσεις ενός προσωπικού βοηθού. Χρειάζεται επίσης να προσφέρει κάποιες παραπάνω δυνατότητες στο χρήστη που θα κάνουν την εφαρμογή να ξεχωρίζει από άλλες αντίστοιχες. Σκοπός μας είναι η εφαρμογή να είναι εύχρηστη, να βοηθά το χρήστη στην καθημερινότητα του και να προσφέρει κάποιες «έξυπνες» λειτουργίες.

Αρχικά, οι δραστηριότητες του χρήστη χωρίστηκαν σε δύο κατηγορίες, τις προγραμματισμένες και τις μη προγραμματισμένες. Προγραμματισμένες είναι οι δραστηριότητες για τις οποίες ο χρήστης γνωρίζει την ακριβή ημερομηνία και ώρα της πραγματοποίησης τους. Αντίθετα ,στις μη προγραμματισμένες ο χρήστης θα μπορεί να θέσει μία συγκεκριμένη ημερομηνία και ώρα ως διορία για να την πραγματοποιήσει. Στη συνέχεια, η εφαρμογή θα δίνει τη δυνατότητα στο χρήστη να προσθέτει την τοποθεσία στην οποία θα πραγματοποιηθεί η δραστηριότητα, ώστε ο χρήστης να μπορεί κάθε στιγμή να την ελέγξει. Ο χρήστης θα μπορεί να προσθέσει τις τοποθεσίες που επιθυμεί στη βάση δεδομένων της συσκευής του και να τις χρησιμοποιήσει όποτε αυτός θέλει για τη δημιουργία των δραστηριοτήτων του. Ο χρήστης θα μπορεί βεβαίως να βάλει έναν τίτλο στη δραστηριότητα του, που να δηλώνει το περιεχόμενο της.

Φυσικά ο χρήστης θα πρέπει να ειδοποιείται για τις δραστηριότητες τις οποίες έχει προγραμματίσει. Οι ειδοποιήσεις χωρίζονται κι αυτές σε δύο κατηγορίες τις ειδοποιήσεις για τις προγραμματισμένες δραστηριότητες και τις ειδοποιήσεις για τις μη προγραμματισμένες. Στην πρώτη κατηγορία ο χρήστης θα ειδοποιείται σε δύο περιπτώσεις. Η πρώτη περίπτωση είναι η ειδοποίηση του χρήστη κάποια ώρα πριν από την έναρξη της δραστηριότητας και η δεύτερη περίπτωση αφορά την ειδοποίηση του χρήστη για την έναρξη της δραστηριότητας.

Στην κατηγορία των μη προγραμματισμένων δραστηριοτήτων οι ειδοποιήσεις είναι πιο πολύπλοκες. Στην αρχή ορίσαμε τις μη προγραμματισμένες δραστηριότητες ως τις δραστηριότητες για τις οποίες ο χρήστης γνωρίζει τη διορία και την τοποθεσία της. Ο χρήστης θα πρέπει να ειδοποιείται όταν βρίσκεται σε κοντινή απόσταση από την επιλεγμένη τοποθεσία και να του δίνεται η δυνατότητα να επιλέξει αν επιθυμεί να την πραγματοποιήσει

εκείνη τη στιγμή ή αργότερα. Οι ειδοποιήσεις αυτής της κατηγορίας δημιουργούνται με τη χρήση των υπηρεσιών τοποθεσίας. Η εφαρμογή θα πρέπει να παρακολουθεί την τρέχουσα τοποθεσία του χρήστη και να ελέγχει αν υπάρχει κάποια τοποθεσία στην κοντινή περιοχή του χρήστη που να ικανοποιεί κάποια από τις δραστηριότητες του. Αν βρεθεί κάποια τέτοια τοποθεσία, τότε η εφαρμογή θα πρέπει να ειδοποιεί το χρήστη για την ύπαρξη της και να τη σημειώνει σ' ένα χάρτη. Τέλος, αν ο χρήστης δεν έχει καταφέρει να πραγματοποιήσει κάποια μη προγραμματισμένη δραστηριότητα, θα πρέπει να ειδοποιείται με τη λήξη της διορίας που έχει τεθεί.

Σε αυτό το σημείο καταφέραμε να προσθέσουμε μία έξυπνη λειτουργία στην εφαρμογή μας. Στις δραστηριότητες ο χρήστης θα μπορεί να προσθέτει έναν τύπο δραστηριότητας ο οποίος θα την προσδιορίζει. Για παράδειγμα, αν η δραστηριότητα αφορά ένα γεύμα ο αντίστοιχος τύπος θα είναι «Εστιατόριο». Άλλοι τύποι δραστηριότητας που μπορεί να χρησιμοποιηθούν είναι «Καφετέρια», «Κινηματογράφος», «Θέατρο», «Φαρμακείο», «Ιατρείο», «Βενζινάδικο», «Καταστήματα Ηλεκτρονικών ειδών» και άλλα. Στη συνέχεια, ο χρήστης θα έχει τη δυνατότητα να θέτει στις τοποθεσίες του ένα συγκεκριμένο τύπο, ο οποίος και θα τις χαρακτηρίζει. Για παράδειγμα, με αυτόν τον τρόπο θα είναι δυνατό να θέσει τα εστιατόρια στα οποία πηγαίνει τον αντίστοιχο τύπο, «Εστιατόριο». Αυτό έχει σαν αποτέλεσμα, η συσκευή να «καταλαβαίνει» τη σύνδεση μεταξύ της τοποθεσίας και του τύπου της δραστηριότητας που αυτή ικανοποιεί.

Ο χρήστης λοιπόν όταν δημιουργεί μία δραστηριότητα θα έχει την επιλογή αρχικά να επιλέξει τον τύπο της δραστηριότητας που επιθυμεί και στη συνέχεια θα μπορεί να περιηγηθεί στις τοποθεσίες που ικανοποιούν μόνο τον συγκεκριμένο τύπο δραστηριότητας. Αυτό θα τον βοηθήσει να δημιουργεί τις δραστηριότητές του με μεγαλύτερη ευκολία και ταχύτητα.

Η κατηγοριοποίηση των τοποθεσιών βάσει του τύπου δραστηριότητας που ικανοποιούν, χρησιμεύει σε ακόμα μία λειτουργία της εφαρμογής. Όπως προαναφέραμε, ο χρήστης μπορεί να δημιουργήσει μία μη προγραμματισμένη δραστηριότητα θέτοντας τη διορία και την τοποθεσία για την πραγματοποίηση της. Θέτοντας όμως έναν τύπο για τη συγκεκριμένη δραστηριότητα έχει τη δυνατότητα να επιλέξει μία ή και περισσότερες τοποθεσίες από αυτές που ικανοποιούν τον τύπο που επιλέχθηκε. Αν ο χρήστης επιλέξει έναν τύπο και δεν επιλέξει κάποια ή κάποιες τοποθεσίες, τότε αυτόματα επιλέγονται όλες οι διαθέσιμές τοποθεσίες του συγκεκριμένου τύπου. Με αυτόν τον τρόπο, όποτε ο χρήστης βρεθεί σε κοντινή απόσταση από μία τοποθεσία που ικανοποιεί κάποια από τις μη προγραμματισμένες δραστηριότητες του θα ειδοποιείται αυτόματα για την ύπαρξη της. Ένα απλό παράδειγμα είναι ότι ο χρήστης πρέπει να προμηθευτεί κάποια φάρμακα από ένα φαρμακείο, χωρίς να τον ενδιαφέρει ποιο θα είναι αυτό. Δημιουργεί λοιπόν μία μη προγραμματισμένη δραστηριότητα, την ονομάζει

«Αντιβιοτικά», θέτει τη διορία που επιθυμεί, έστω μία βδομάδα, και επιλέγει τον τύπο «Φαρμακείο». Όταν λοιπόν ο χρήστης βρεθεί σε μία περιοχή που είναι κοντά σε κάποιο από τα φαρμακεία της βάσης δεδομένων της συσκευής ειδοποιείται για την ύπαρξη του και του δίνεται η δυνατότητα να πραγματοποιήσει τη δραστηριότητα του.

3.2.1. Αλληλεπίδραση με το server και με άλλους χρήστες

Για να ταιριάζει η εφαρμογή μας με τη σύγχρονη εποχή δίνουμε τη δυνατότητα στους χρήστες της να αλληλεπιδρούν με άλλους χρήστες. Η επικοινωνία αυτή μεταξύ των χρηστών δε γίνεται άμεσα, αλλά έμμεσα, χρησιμοποιώντας ένα server. Ο χρήστης για να χρησιμοποιήσει την εφαρμογή και τις δυνατότητες που θα προσφέρει ο server θα πρέπει αρχικά να κάνει μία τυπική εγγραφή, συμπληρώνοντας κάποια στοιχεία, όπως όνομα, επώνυμο και διεύθυνση ηλεκτρονικού ταχυδρομείου.

Αρχικά, με την ενεργοποίηση της εφαρμογής, στη βάση δεδομένων της συσκευής θα εγγράφονται κάποιες βασικές τοποθεσίες και οι αντίστοιχοι τύποι δραστηριότητας που ικανοποιούν. Με αυτόν τον τρόπο ο χρήστης θα έχει τη δυνατότητα να δημιουργήσει κάποια δραστηριότητα, χωρίς να χρειάζεται να προσθέσει κάποια τοποθεσία, ακόμα και όταν δεν έχει πρόσβαση στο διαδίκτυο. Βεβαίως, θα έχει τη δυνατότητα να δημιουργήσει μία νέα τοποθεσία και να θέσει έναν τύπο από τους υπάρχοντες για αυτήν ή και να δημιουργήσει έναν νέο τύπο που να καλύπτει τις δικές του ανάγκες.

Η αλληλεπίδραση με τους άλλους χρήστες αφορά κατά κύριο λόγο τις τοποθεσίες. Κατά τη δημιουργία μίας νέας τοποθεσίας, ο χρήστης θα έχει την επιλογή να τη διαθέτει στο server. Η τοποθεσία αυτή θα εγγράφεται τόσο στη βάση δεδομένων της συσκευής, όσο και σε αυτήν του server. Στην πλευρά του server, οι τοποθεσίες θα εγγράφονται με τέτοιον τρόπο ώστε να γνωρίζουμε ότι πρόκειται για τοποθεσίες που έχουν δημιουργηθεί από το χρήστη και να ξεχωρίζουν από τις αρχικές τοποθεσίες του server. Αν ο τύπος της δραστηριότητας που ικανοποιεί η τοποθεσία που προστέθηκε υπάρχει στο server τότε γίνεται απλά η σύνδεση τους, ενώ αν δεν υπάρχει δημιουργείται και ύστερα γίνεται η σύνδεση. Οι τοποθεσίες που θα προστίθενται από τους χρήστες δε θα περνάνε αυτόματα στις βάσεις δεδομένων των συσκευών των άλλων χρηστών. Κατά τη δημιουργία όμως μίας δραστηριότητας, ο χρήστης θα έχει τη δυνατότητα να επιλέξει μία τοποθεσία που έχει προστεθεί στο server από έναν άλλο χρήστη. Με αυτόν τον τρόπο ο χρήστης θα μπορεί να αποθηκεύσει στη βάση δεδομένων της συσκευής του μία τοποθεσία που έχει προστεθεί από έναν άλλον χρήστη.

Οι χρήστες λοιπόν, δημιουργώντας συνεχώς νέες τοποθεσίες και επιλέγοντας να τις διαθέσουν στους άλλους χρήστες μέσω του server δημιουργούν μία πολύ μεγάλη βάση

δεδομένων τοποθεσιών. Αυτό αποτελεί ένα παράδειγμα της τεχνικής του crowdsourcing. Με αυτόν τον τρόπο όσο περισσότεροι χρήστες χρησιμοποιούν την εφαρμογή και δημιουργούν νέες τοποθεσίες, τόσο αυξάνεται το πλήθος των διαθέσιμων τοποθεσιών. Αυτό φυσικά είναι θετικό για τους χρήστες αφού μπορούν να επιλέξουν ανάμεσα σε έναν τεράστιο αριθμό τοποθεσιών για τον τύπο δραστηριότητας που επιθυμούν.

3.2.2. Αλληλεπίδραση με το Facebook.

Για να εκσυγχρονίσουμε ακόμη περισσότερο την εφαρμογή μας, προσθέσαμε τη δυνατότητα αλληλεπίδρασης με το Facebook. Το Facebook είναι ένας κολοσσός του διαδικτύου που έχει πάνω από ένα δισεκατομμύριο ενεργούς χρήστες. Επίσης, έχει μία τεράστια βάση δεδομένων τοποθεσιών (Facebook Places). Ο χρήστης αρχικά θα μπορεί να κάνει χρήση του λογαριασμού του στο κοινωνικό αυτό δίκτυο για την είσοδο του στην εφαρμογή. Ένας χρήστης που θα έχει συνδεθεί στην εφαρμογή χρησιμοποιώντας το λογαριασμό του στο Facebook, θα έχει τη δυνατότητα να προσθέσει στις τοποθεσίες του μία από τις τοποθεσίες του Facebook. Αυτή η δυνατότητα θα του δίνεται κατά τη δημιουργία μίας νέας τοποθεσίας. Πατώντας το σχετικό κουμπί θα μπορεί να δει μία λίστα με τοποθεσίες του Facebook που βρίσκονται στην κοντινή του περιοχή και επιλέγοντας μία θα μπορεί να την αποθηκεύει στη συσκευή του, χαρακτηρίζοντας την και με έναν τύπο δραστηριότητας. Φυσικά, μπορεί να επιλέξει να κάνει αυτήν την τοποθεσία διαθέσιμη στο server και με αυτόν τον τρόπο θα μπορεί ένας χρήστης ο οποίος δεν έχει συνδεθεί στην εφαρμογή μέσω του λογαριασμού του στο Facebook να χρησιμοποιήσει μία τοποθεσία που προέρχεται από αυτό.

3.2.3. Συνοπτικά

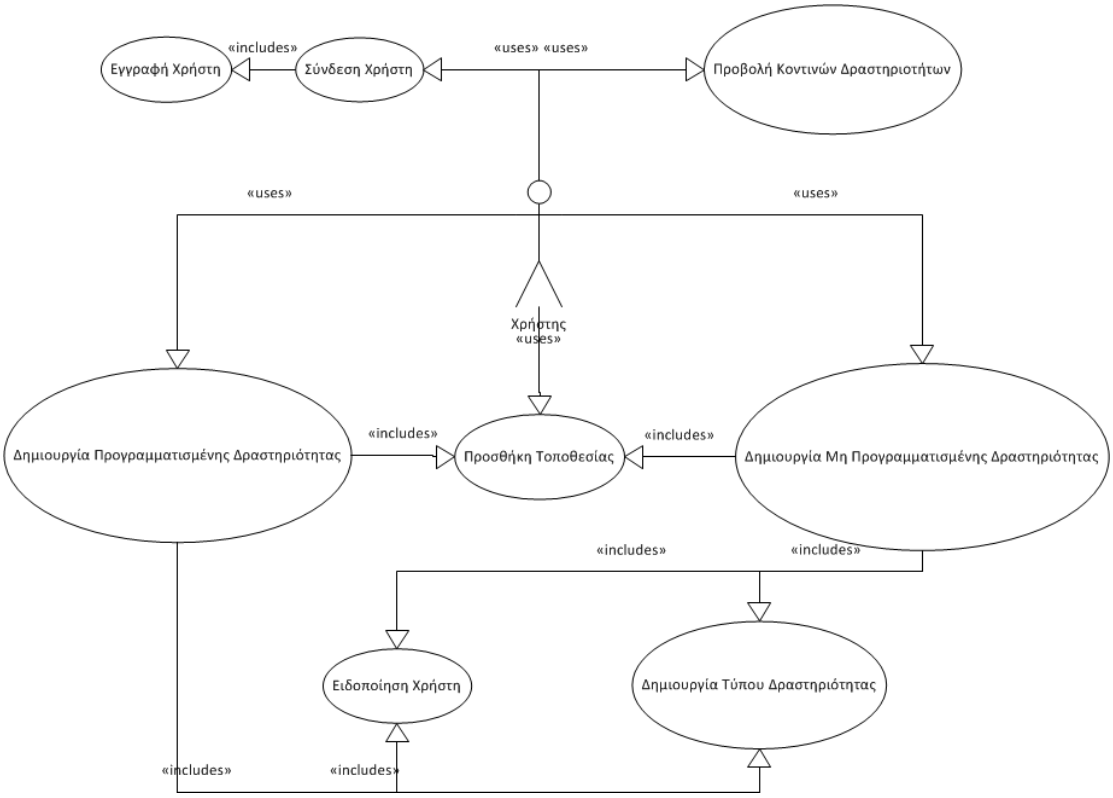
Οι λειτουργίες λοιπόν που θα πρέπει να ενσωματώνει η εφαρμογή είναι:

- Δημιουργία προγραμματισμένης δραστηριότητας.
- Δημιουργία μη προγραμματισμένης δραστηριότητας.
- Δημιουργία νέας τοποθεσίας.
- Δημιουργία νέου τύπου δραστηριότητας.
- Ειδοποίηση για όλες τις περιπτώσεις δραστηριοτήτων.
- Προβολή της λίστας κάθε κατηγορίας δραστηριοτήτων.
- Προβολή των τοποθεσιών του χρήστη.
- Εγγραφή και σύνδεση στο server της εφαρμογής ή σύνδεση μέσω Facebook.
- Λήψη των τοποθεσιών από το server.
- Αναζήτηση τοποθεσιών μέσω Facebook.

- Αποστολή τοποθεσίας στο server.
- Χρήση τοποθεσίας που έχει προστεθεί από κάποιον άλλο χρήστη.

3.3. Σενάρια χρήσης

Για την εξαγωγή των απαιτήσεων της εφαρμογής αναλύθηκαν τα κυριότερα σενάρια χρήσης και στο παρακάτω σχήμα παρουσιάζεται το διάγραμμα σεναρίων χρήσης.



Εικόνα 11 Διάγραμμα σεναρίων χρήσης.

3.3.1. Εγγραφή χρήστη

Το σενάριο αυτό προδιαγράφει τη διαδικασία που ακολουθείται για να εγγραφεί στην εφαρμογή ένας νέος χρήστης.

3.3.1.1. Βασική ροή γεγονότων

Χρήστης	Εφαρμογή	Server
1. Εκκίνηση εφαρμογής. 2. Επιλογή συνδέσμου ‘Εγγραφή’.		
	3. Μετάβαση στην φόρμα εγγραφής.	
4. Συμπλήρωση πεδίων		

φόρμας : i. Όνομα χρήστη ii. Κωδικός εισόδου iii. Διεύθυνση ηλεκτρονικού ταχυδρομείου iv. Όνομα v. Επώνυμο vi. Χώρα διαμονής vii. Πόλη		
	5. Έλεγχος συμπληρωθέντων στοιχείων. 6. Αποστολή στοιχείων στο Server.	
		7. Έλεγχος μοναδικότητας στοιχείων: i. Όνομα χρήστη ii. Διεύθυνση ηλεκτρονικού ταχυδρομείου 8. Δημιουργία εγγραφής χρήστη στη βάση δεδομένων. 9. Επιστροφή μηνύματος επιτυχούς εγγραφής χρήστη.
	10. Εμφάνιση κεντρικής οθόνης.	

3.3.1.2. Ροή διαχείρισης σφάλματος – Ελλιπής συμπλήρωση φόρμας

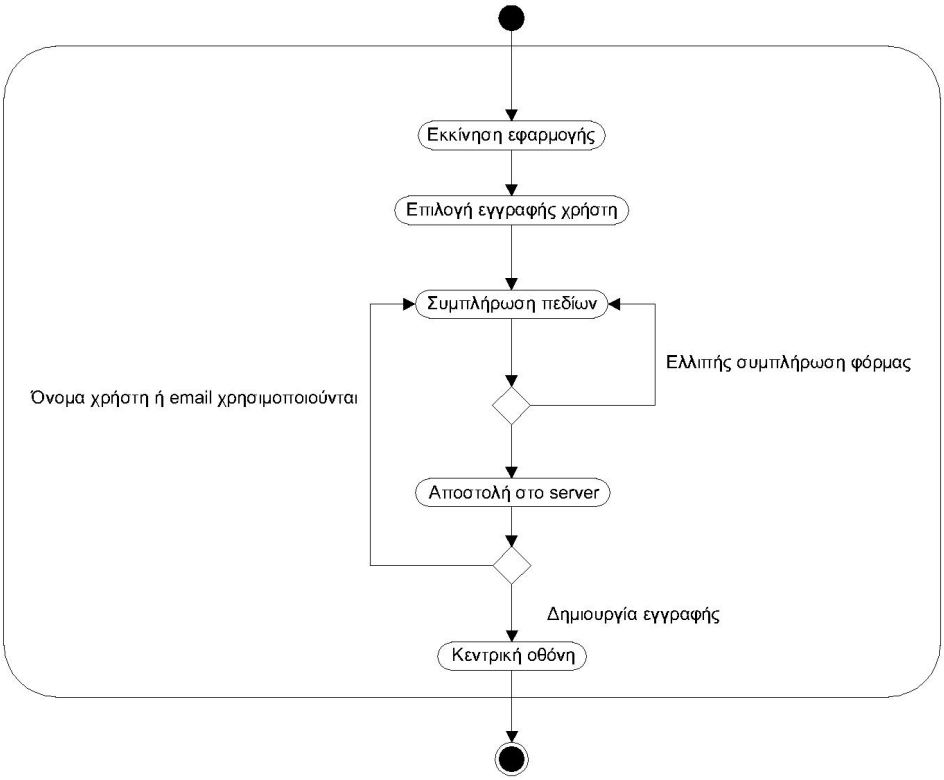
Χρήστης	Εφαρμογή
	5. Ελλιπής συμπλήρωση φόρμας. 6. Εμφάνιση μηνύματος σφάλματος. 7. Επιστροφή στη φόρμα εγγραφής χρήστη.

3.3.1.3. Ροή διαχείρισης σφάλματος – Μη μοναδικό όνομα χρήστη/διεύθυνση ηλεκρτονικού ταχυδρομείου

Χρήστης	Εφαρμογή	Server
		7. Όνομα χρήστη ή διεύθυνση ηλεκτρονικού ταχυδρομείου μη μοναδικά. 8. Επιστροφή μη επιτυχούς

		εγγραφής χρήστη.
	9. Εμφάνιση μηνύματος σφάλματος. 10. Επιστροφή στη φόρμα εγγραφής χρήστη.	

3.3.1.4. Διάγραμμα δραστηριοτήτων



Εικόνα 12 Εγγραφή χρήστη στο server.

3.3.2. Σύνδεση χρήστη

Το σενάριο αυτό προδιαγράφει τη διαδικασία που ακολουθείται για να εισέλθει στην εφαρμογή ένας χρήστης.

3.3.2.1. Βασική ροή γεγονότων

Χρήστης	Εφαρμογή	Server
1. Εκκίνηση εφαρμογής. 2. Επιλογή συνδέσμου 'Εγγραφή'.		
	3. Μετάβαση στη φόρμα σύνδεσης.	

4. Συμπλήρωση των πεδίων. i. Όνομα χρήστη ii. Κωδικός πρόσβασης		
	5. Έλεγχος συμπληρωθέντων στοιχείων. 6. Αποστολή στοιχείων στο server.	
		7. Έλεγχος εγκυρότητας στοιχείων. 8. Επιστροφή μηνύματος επιτυχούς σύνδεσης.
	9. Εμφάνιση κεντρικής οθόνης.	

3.3.2.2. Ροή διαχείρισης σφάλματος – Ελλιπής συμπλήρωση φόρμας.

Χρήστης	Εφαρμογή
	5. Ελλιπής συμπλήρωση φόρμας. 6. Εμφάνιση μηνύματος σφάλματος. 7. Επιστροφή στη φόρμα εγγραφής χρήστη.

3.3.2.3. Ροή διαχείρισης σφάλματος – Μη έγκυρα στοιχεία χρήστη.

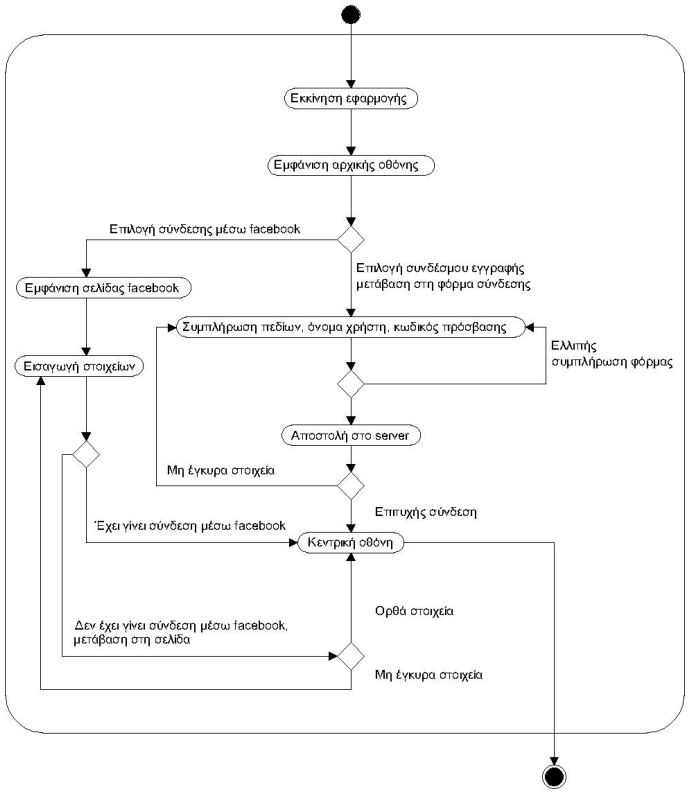
Χρήστης	Εφαρμογή	Server
		7. Μη έγκυρα στοιχεία χρήστη. 8. Επιστροφή μη επιτυχούς σύνδεσης.
	9. Εμφάνιση μηνύματος σφάλματος. 10. Επιστροφή στη φόρμα σύνδεσης χρήστη.	

3.3.2.4. Εναλλακτική ροή – Σύνδεση μέσω Facebook.

Χρήστης	Εφαρμογή	Facebook Server
1. Εκκίνηση εφαρμογής. 2. Επιλογή συνδέσμου 'Σύνδεση μέσω Facebook'.		
	3. Προώθηση στη σελίδα σύνδεσης με το Facebook.	

4. Συμπλήρωση των πεδίων: iii. Όνομα χρήστη iv. Κωδικός πρόσβασης		
		5. Έλεγχος συμπληρωθέντων στοιχείων. 6. Επιστροφή μηνύματος επιτυχούς σύνδεσης και στοιχείων χρήστη.
	7. Εμφάνιση κεντρικής οθόνης.	

3.3.2.5. Διάγραμμα δραστηριοτήτων



Εικόνα 13 Σύνδεση χρήστη.

3.3.3. Προσθήκη τοποθεσίας

Το σενάριο αυτό προδιαγράφει τη διαδικασία που ακολουθεί ένας χρήστης ώστε να δημιουργήσει μια καινούρια τοποθεσία και να την προσθέσει στις τοποθεσίες του ώστε να μπορεί να τη χρησιμοποιήσει στο μέλλον. Ο χρήστης θα έχει τη δυνατότητα αυτή από τις

οθόνες που περιέχουν τις λίστες με τις προγραμματισμένες και μη προγραμματισμένες δραστηριότητες και από την οθόνη με τη λίστα με τις τοποθεσίες.

3.3.3.1. Βασική ροή γεγονότων

Χρήστης	Εφαρμογή
1. Επιλογή προσθήκης νέας τοποθεσίας.	
	2. Μετάβαση στην οθόνη προβολής του χάρτη όπου φαίνεται η τρέχουσα τοποθεσία του χρήστη.
3. Ο χρήστης πατώντας παρατεταμένα με το δάχτυλο του σε ένα σημείο του χάρτη, διαλέγει το σημείο που επιθυμεί να προσθέσει.	
	4. Στο σημείο που επιλέχθηκε τοποθετείται μία «πινέζα» και μία ετικέτα με τις πληροφορίες τις τοποθεσίας.
5. Αν η τοποθεσία είναι αυτή που επιθυμεί, ο χρήστης πατάει το κουμπί που βρίσκεται στην ετικέτα για να την επιλέξει.	
	6. Εμφάνιση φόρμας συμπλήρωσης στοιχείων τοποθεσίας.
7. Ο χρήστης συμπληρώνει τον τίτλο και επιλέγει τον τύπο δραστηριότητας.	
	8. Εμφάνιση λίστας με τους διαθέσιμους τύπους.
9. Επιλογή από το χρήστη του επιθυμητού τύπου ή δημιουργία νέου τύπου.	
	10. Επιστροφή στη φόρμα συμπλήρωσης στοιχείων τοποθεσίας.
11. Επιβεβαίωση στοιχείων τοποθεσίας.	
	12. Δημιουργία εγγραφής τοποθεσίας στη βάση δεδομένων της εφαρμογής. 13. Επιστροφή στην οθόνη από την οποία επιλέχθηκε η δημιουργία νέας τοποθεσίας.

3.3.3.2. Εναλλακτική ροή – Προσθήκη τοποθεσίας μέσω αναζήτησης

Χρήστης	Εφαρμογή
3. Ο χρήστης επιλέγει την αναζήτηση τοποθεσίας. 4. Ο χρήστης πληκτρολογεί τη διεύθυνση της επιθυμητής τοποθεσίας ή την περιοχή.	
	5. Εμφάνιση λίστας με τοποθεσίες που ταιριάζουν στην αναζήτηση που

	πραγματοποιήθηκε.
6. Επιλογή από το χρήστη επιθυμητής τοποθεσίας.	
	7. Εμφάνιση του χάρτη και τοποθέτηση «πινέζας» στο σημείο που αντιστοιχεί στην τοποθεσία που επιλέχθηκε.

3.3.3.3. Εναλλακτική ροή – Επιλογή τοποθεσίας μέσω του Facebook.

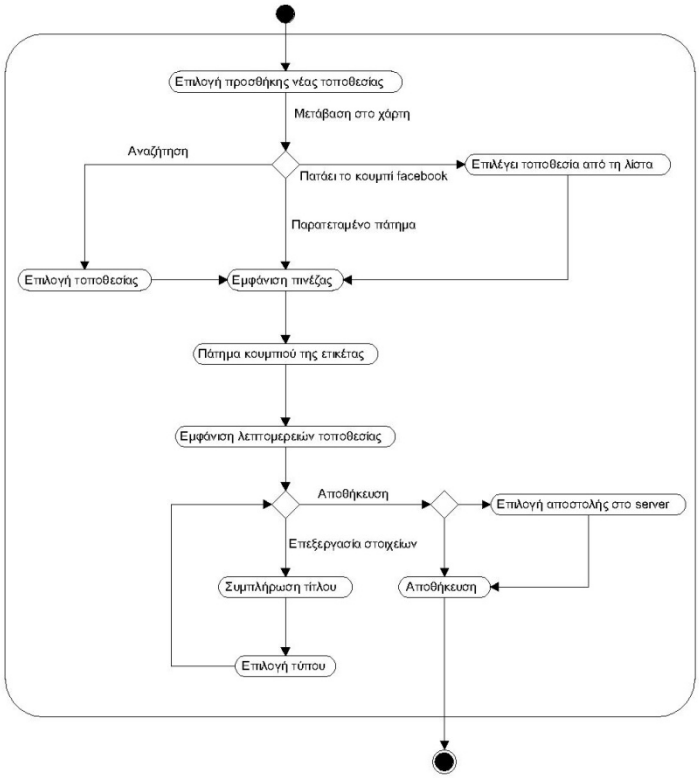
Χρήστης	Εφαρμογή
3. Ο χρήστης επιλέγει να προσθέσει τοποθεσία μέσω του Facebook, πατώντας το σχετικό κουμπί.	
	4. Εμφάνιση μιας λίστας με τοποθεσίες του Facebook, σε κοντινή απόσταση από την τρέχουσα τοποθεσία του χρήστη.
5. Επιλογή από το χρήστη της επιθυμητής τοποθεσίας.	
	6. Εμφάνιση του χάρτη και τοποθέτηση «πινέζας» στο σημείο που αντιστοιχεί στην τοποθεσία που επιλέχθηκε.

3.3.3.4. Εναλλακτική ροή – Επιλογή αποστολής της τοποθεσίας στο sever.

Χρήστης	Εφαρμογή	Server
11. Ο χρήστης επιλέγει να αποθηκεύσει στο server την τοποθεσία που δημιουργεί μαζί με τον τύπο δραστηριότητας που ικανοποιεί.		
	12. Η εφαρμογή ελέγχει την εγκυρότητα των στοιχείων.	
		13. Ο sever ελέγχει αν ο τύπος δραστηριότητας υπάρχει στη βάση δεδομένων του. Αν δεν υπάρχει το δημιουργεί. 14. Δημιουργία εγγραφής τοποθεσίας στη βάση δεδομένων του server και σύνδεση με τον αντίστοιχο τύπο

		δραστηριότητας. 15. Αποστολή μηνύματος επιτυχούς εγγραφής τοποθεσίας
	11. Δημιουργία εγγραφής τοποθεσίας στη βάση δεδομένων του κινητού. 12. Επιστροφή στην κεντρική οθόνη.	

3.3.3.5. Διάγραμμα δραστηριοτήτων



Εικόνα 14 Προσθήκη τοποθεσίας.

3.3.4. Δημιουργία τύπου δραστηριότητας

Ο χρήστης επιθυμεί να προσθέσει ένα νέο τύπο δραστηριότητας στην εφαρμογή του. Η λειτουργία αυτή είναι δυνατή από την οθόνη προβολής της λίστας με τους τύπους δραστηριότητας. Η λίστα αυτή προβάλλεται σε δύο περιπτώσεις:

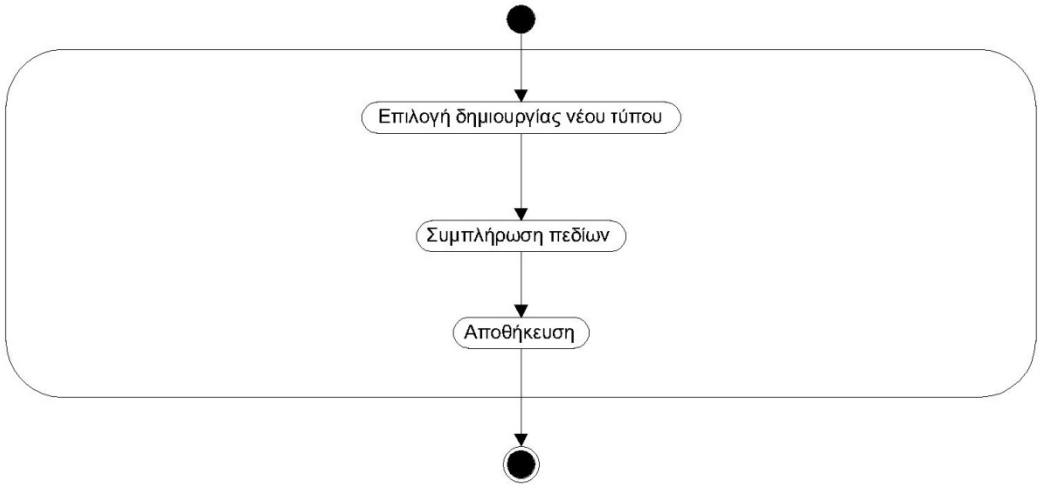
i. Όταν δημιουργούμε μία νέα τοποθεσία και επιλέγουμε τον τύπο.

ii. Όταν δημιουργούμε μια νέα δραστηριότητα και επιλέγουμε τον τύπο.

3.3.4.1. Βασική ροή γεγονότων

Χρήστης	Εφαρμογή
1. Ο χρήστης επιλέγει τη δημιουργία νέου τύπου δραστηριότητας.	
	2. Προβολή της φόρμας συμπλήρωσης στοιχείων του τύπου
3. Συμπλήρωση πεδίων φόρμας : i. Όνομα τύπου δραστηριότητας. ii. Χρόνος προειδοποίησης δραστηριότητας σε λεπτά.	
	4. Δημιουργία εγγραφής τύπου στη βάση δεδομένων της συσκευής.

3.3.4.2. Διάγραμμα δραστηριοτήτων



Εικόνα 15 Δημιουργία νέου τύπου δραστηριότητας.

3.3.5. Δημιουργία προγραμματισμένης δραστηριότητας

Ο χρήστης επιθυμεί να προσθέσει μία προγραμματισμένη δραστηριότητα. Η δυνατότητα αυτή θα δίνεται στο χρήστη από την οθόνη προβολής της λίστας των προγραμματισμένων και των μη προγραμματισμένων δραστηριοτήτων και από την οθόνη με τη λίστα των τοποθεσιών του χρήστη.

3.3.5.1. Βασική ροή γεγονότων

Χρήστης	Εφαρμογή	Server
1. Ο χρήστης επιλέγει να δημιουργήσει μια νέα προγραμματισμένη δραστηριότητα.		
	2. Προβολή φόρμας συμπλήρωσης λεπτομερειών της δραστηριότητας.	
3. Συμπλήρωση τίτλου δραστηριότητας. 4. Επιλογή του πεδίου της ημερομηνίας.		
	5. Εμφάνιση οθόνης επιλογής της ημερομηνίας και της ώρας.	
6. Επιλογή από το χρήστη της ημερομηνίας και της ώρας που θα πραγματοποιηθεί η δραστηριότητα.		
	7. Επιστροφή στη φόρμα συμπλήρωσης λεπτομερειών της δραστηριότητας.	
8. (Προαιρετικά) Επιλογή του πεδίου του τύπου.		
	9. Εμφάνιση στην οθόνη της λίστας με τους διαθέσιμους τύπους.	
10. (Προαιρετικά) Επιλογή του επιθυμητού τύπου ή δημιουργία νέου τύπου.		
	11. Πραγματοποίηση αίτησης στο server για	

	να φέρει τις τοποθεσίες που έχουν προστεθεί σε αυτόν από τους χρήστες.	
		12. Αναζήτηση των τοποθεσιών των χρηστών που ικανοποιούν το συγκεκριμένο τύπο δραστηριότητας. 13. Αποστολή των τοποθεσιών αυτών στην εφαρμογή
	14. Λήψη των τοποθεσιών που ζητήθηκαν από το server. 15. Επιστροφή στη φόρμα συμπλήρωσης λεπτομερειών της δραστηριότητας.	
16. (Προαιρετικά) Επιλογή του πεδίου της τοποθεσίας.		
	17. Εμφάνιση του χάρτη, στον οποίο είναι σημειωμένες οι τοποθεσίες που ικανοποιούν τον τύπο δραστηριότητας που έχει επιλεγεί καθώς και η τρέχουσα τοποθεσία του.	
18. (Προαιρετικά) Ο χρήστης επιλέγει την επιθυμητή τοποθεσία και πατάει το κουμπί της αντίστοιχης ετικέτας.		
	19. Επιστροφή στη φόρμα συμπλήρωσης λεπτομερειών της δραστηριότητας.	
20. Επιλογή αποθήκευσης της δραστηριότητας.		
	21. Δημιουργία εγγραφής δραστηριότητας στη βάση δεδομένων της συσκευής. 22. Δημιουργία των	

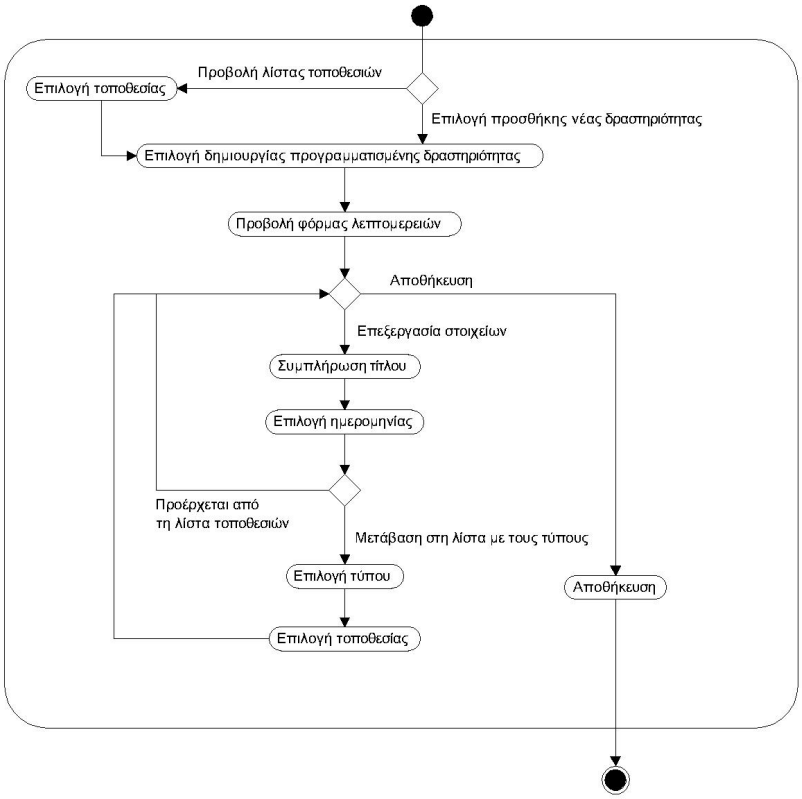
	κατάλληλων ειδοποιήσεων για τη συγκεκριμένη δραστηριότητα, ανάλογα με το εάν έχει επιλέξει τοποθεσία και τύπο. 23. Επιστροφή στην οθόνη από την οποία επιλέχθηκε η δημιουργία της νέας δραστηριότητας.	

3.3.5.2. Εναλλακτική ροή – Δημιουργία της νέας δραστηριότητας από την οθόνη προβολής της λίστας με τις τοποθεσίες του χρήστη.

Χρήστης	**Εφαρμογή**
1. Ο χρήστης επιλέγει να δει τις αποθηκευμένες τοποθεσίες του.	
	2. Εμφάνιση της λίστας με τις τοποθεσίες του χρήστη.
3. Ο χρήστης επιλέγει την τοποθεσία που θέλει να πραγματοποιήσει τη δραστηριότητα του.	
	4. Προβολή επιλογής είδους δραστηριότητας, προγραμματισμένης ή μη προγραμματισμένης.
5. Επιλογή της προγραμματισμένης από το χρήστη.	
	6. Προβολή φόρμας συμπλήρωσης λεπτομερειών της δραστηριότητας στην οποία είναι συμπληρωμένα τα πεδία του τύπου και της τοποθεσίας με τιμές αυτές που αντιστοιχούν στην τοποθεσία που επιλέχθηκε.
7. Συμπλήρωση τίτλου δραστηριότητας. 8. Επιλογή του πεδίου της ημερομηνίας.	
	9. Εμφάνιση οθόνης επιλογής της ημερομηνίας και της ώρας.
10. Επιλογή από το χρήστη της ημερομηνίας και της ώρας που θα πραγματοποιηθεί η δραστηριότητα.	
	11. Επιστροφή στη φόρμα συμπλήρωσης λεπτομερειών της δραστηριότητας.
12. Επιλογή αποθήκευσης της	

δραστηριότητας.	
	13. Δημιουργία εγγραφής δραστηριότητας στη βάση δεδομένων της συσκευής. 14. Δημιουργία των κατάλληλων ειδοποιήσεων για τη συγκεκριμένη δραστηριότητα.

3.3.5.3. Διάγραμμα δραστηριοτήτων



Εικόνα 16 Δημιουργία προγραμματισμένης δραστηριότητας.

3.3.6. Δημιουργία μη προγραμματισμένης δραστηριότητας

Ο χρήστης επιθυμεί να προσθέσει μία μη προγραμματισμένη δραστηριότητα. Η δυνατότητα αυτή θα δίνεται στο χρήστη από την οθόνη προβολής της λίστας των προγραμματισμένων και των μη προγραμματισμένων δραστηριοτήτων και από την οθόνη με τη λίστα των τοποθεσιών του χρήστη.

3.3.6.1. Βασική ροή γεγονότων

Χρήστης	Εφαρμογή	Server
1. Ο χρήστης επιλέγει να δημιουργήσει μια νέα μη προγραμματισμένη δραστηριότητα.		
	2. Προβολή φόρμας συμπλήρωσης λεπτομερειών της δραστηριότητας.	
3. Συμπλήρωση τίτλου δραστηριότητας. 4. Επιλογή του πεδίου της διορίας.		
	5. Εμφάνιση οθόνης επιλογής της ημερομηνίας και της ώρας της διορίας.	
6. Επιλογή από το χρήστη της ημερομηνίας και της ώρας μέχρι την οποία επιθυμεί να πραγματοποιήσει τη δραστηριότητα.		
	7. Επιστροφή στη φόρμα συμπλήρωσης λεπτομερειών της δραστηριότητας.	
8. (Προαιρετικά) Επιλογή του πεδίου του τύπου.		
	9. Εμφάνιση στην οθόνη της λίστας με τους διαθέσιμους τύπους.	
10. (Προαιρετικά) Επιλογή του επιθυμητού τύπου ή δημιουργία νέου τύπου.		
	11. Πραγματοποίηση αίτησης στο server για να φέρει τις τοποθεσίες που έχουν προστεθεί στο server από τους χρήστες.	
		12. Αναζήτηση των τοποθεσιών των χρηστών που ικανοποιούν το συγκεκριμένο τύπο

		δραστηριότητας. 13. Αποστολή των τοποθεσιών αυτών στην εφαρμογή
	14. Λήψη των τοποθεσιών που ζητήθηκαν από το server. 15. Επιστροφή στη φόρμα συμπλήρωσης λεπτομερειών της δραστηριότητας.	
16. (Προαιρετικά) Επιλογή του πεδίου της τοποθεσίας.		
	17. Εμφάνιση του χάρτη, στον οποίο είναι σημειωμένες οι τοποθεσίες που ικανοποιούν τον τύπο δραστηριότητας (εάν έχει επιλεγεί) καθώς και η τρέχουσα τοποθεσία του χρήστη. Εάν ο χρήστης έχει επιλέξει περισσότερες τοποθεσίες εμφανίζεται το πλήθος των τοποθεσιών που έχουν επιλεγεί.	
18. (Προαιρετικά) Επιλογή στον χάρτη προβολής των τοποθεσιών ως λίστα.		
	19. Εμφάνιση της λίστας όλων των τοποθεσιών	
20. (Προαιρετικά) Ο χρήστης επιλέγει μια ή περισσότερες τοποθεσίες και πατάει το κουμπί της αντίστοιχης ετικέτας.		
	21. Επιστροφή στη φόρμα συμπλήρωσης λεπτομερειών της δραστηριότητας.	
22. Επιλογή αποθήκευσης της δραστηριότητας.		

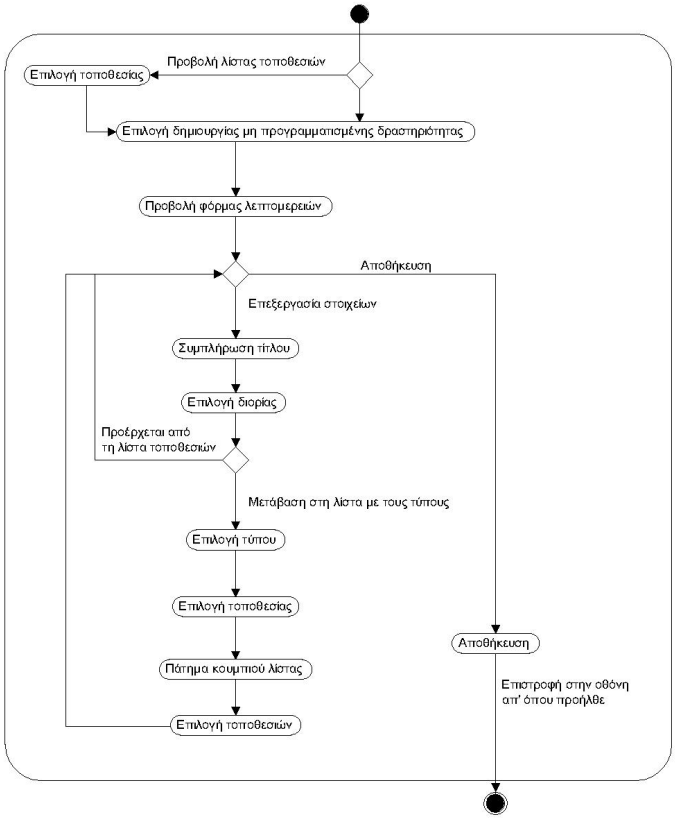
	23. Δημιουργία εγγραφής δραστηριότητας στη βάση δεδομένων της συσκευής. 24. Δημιουργία των κατάλληλων ειδοποιήσεων για τη συγκεκριμένη δραστηριότητα, ανάλογα με το εάν έχει επιλέξει τοποθεσία και τύπο. 25. Επιστροφή στην οθόνη από την οποία επιλέχθηκε η δημιουργία της νέας δραστηριότητας.	

3.3.6.2. Εναλλακτική ροή – Δημιουργία της νέας δραστηριότητας από την οθόνη προβολής της λίστας με τις τοποθεσίες του χρήστη.

Χρήστης	**Εφαρμογή**
1. Ο χρήστης επιλέγει να δει τις αποθηκευμένες τοποθεσίες του.	
	2. Εμφάνιση της λίστας με τις τοποθεσίες του χρήστη.
3. Ο χρήστης επιλέγει την τοποθεσία που θέλει να πραγματοποιήσει τη δραστηριότητα του.	
	4. Προβολή επιλογής είδους δραστηριότητας, προγραμματισμένης ή μη προγραμματισμένης.
5. Επιλογή της μη προγραμματισμένης από το χρήστη.	
	6. Προβολή φόρμας συμπλήρωσης λεπτομερειών της δραστηριότητας, στην οποία είναι συμπληρωμένα τα πεδία του τύπου και της τοποθεσίας με τιμές αυτές που αντιστοιχούν στην τοποθεσία που επιλέχθηκε.
7. Συμπλήρωση τίτλου δραστηριότητας. 8. Επιλογή του πεδίου της διορίας.	
	9. Εμφάνιση οθόνης επιλογής της ημερομηνίας και της ώρας της διορίας.

10. Επιλογή από το χρήστη της ημερομηνίας και της ώρας μέχρι την οποία επιθυμεί να πραγματοποιήσει τη δραστηριότητα.	
	11. Επιστροφή στη φόρμα συμπλήρωσης λεπτομερειών της δραστηριότητας.
12. Επιλογή αποθήκευσης της δραστηριότητας.	
	13. Δημιουργία εγγραφής δραστηριότητας στη βάση δεδομένων της συσκευής.

3.3.6.3. Διάγραμμα δραστηριοτήτων



Εικόνα 17 Δημιουργία μη προγραμματισμένης δραστηριότητας.

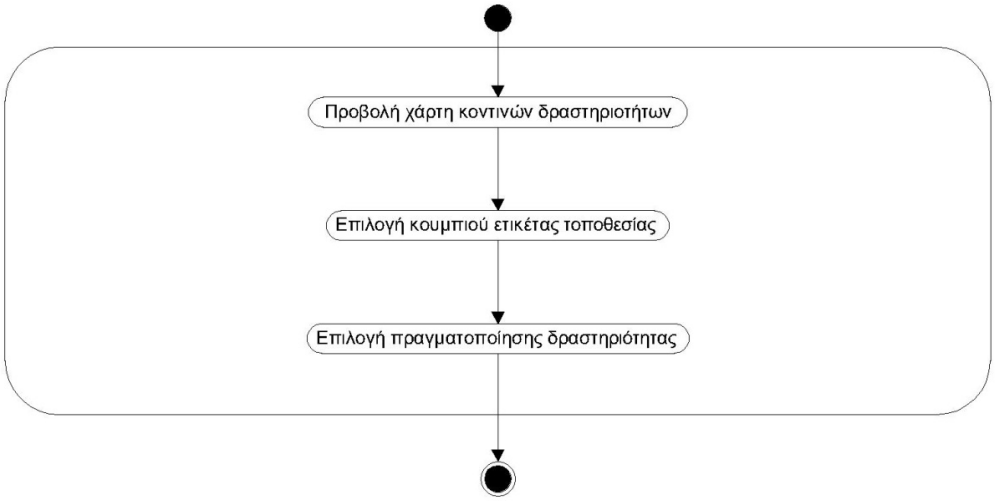
3.3.7. Προβολή κοντινών δραστηριοτήτων

Ο χρήστης επιθυμεί να δει τις δραστηριότητες και των δύο τύπων, που έχει προγραμματίσει σε τοποθεσίες που είναι κοντά στη τρέχουσα τοποθεσία του.

3.3.7.1. Βασική ροή γεγονότων

Χρήστης	Εφαρμογή
1. Ο χρήστης επιλέγει προβολή του χάρτη των κοντινών δραστηριοτήτων.	
	2. Εμφάνιση του χάρτη, με σημειωμένες τις κοντινές τοποθεσίες, στις οποίες ο χρήστης έχει προγραμματίσει κάποιες δραστηριότητες.
3. Επιλογή του κουμπιού της ετικέτας της τοποθεσίας που ενδιαφέρει το χρήστη.	
	4. Εμφάνιση μηνύματος στο χρήστη για το αν επιθυμεί να πραγματοποιήσει τη δραστηριότητα που έχει προγραμματίσει στην τοποθεσία που επέλεξε.
5. Ο χρήστης επιλέγει να πραγματοποιήσει τη δραστηριότητα.	
	6. Επιστροφή στην οθόνη με το χάρτη, στον οποίον δε σημειώνεται πλέον η τοποθεσία της δραστηριότητας που μόλις πραγματοποιήθηκε

3.3.7.2. Διάγραμμα δραστηριοτήτων



Εικόνα 18 Προβολή κοντινών δραστηριοτήτων.

3.3.8. Ειδοποίηση χρήστη

Ο χρήστης ειδοποιείται για κάποια δραστηριότητα που έχει προγραμματίσει, είτε πρόκειται για προγραμματισμένη, είτε για μη προγραμματισμένη.

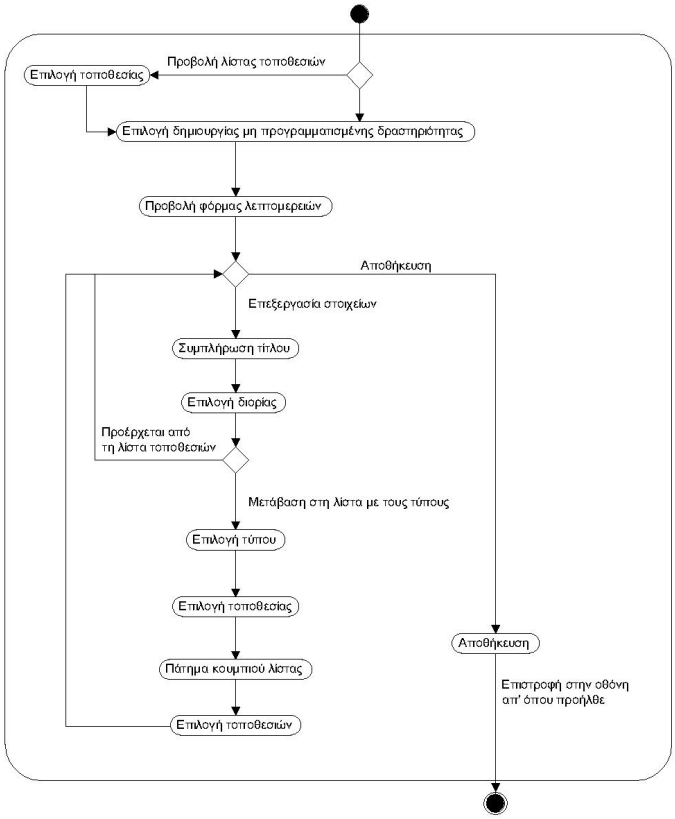
3.3.8.1. Βασική ροή γεγονότων

Χρήστης	Εφαρμογή
	1. Στην οθόνη της συσκευής εμφανίζεται μία ειδοποίηση. 2. Η ειδοποίηση ενδέχεται να αφορά στην : a. προειδοποίηση για επικείμενη έναρξη δραστηριότητας b. έναρξη μιας προγραμματισμένης δραστηριότητας c. λήξη διορίας μη προγραμματισμένης δραστηριότητας
3. Ο χρήστης επιλέγει να δει τις λεπτομέρειες της δραστηριότητας.	
	4. Προβολή της φόρμας με τις λεπτομέρειες της δραστηριότητας. 5. Εμφάνιση μηνύματος στο χρήστη αν επιθυμεί να αλλάξει την ημερομηνία και την ώρα της δραστηριότητας ή αν επιθυμεί να θέσει τη δραστηριότητα ως εκτελεσμένη.
6. (Προαιρετικά) Ο χρήστης τροποποιεί την ημερομηνία και την ώρα της δραστηριότητας.	
	7. Αποθήκευση της δραστηριότητας και επαναπρογραμματισμός των ειδοποιήσεων 8. Επιστροφή στη λίστα με τις προγραμματισμένες δραστηριότητες.
9. (Προαιρετικά) Επιλογή από το χρήστη να θέσει τη δραστηριότητα ως εκτελεσμένη.	
	10. Η εφαρμογή επισημαίνει τη δραστηριότητα ως εκτελεσμένη. 11. Επιστροφή στη λίστα με τις προγραμματισμένες δραστηριότητες.

3.3.8.2. Εναλλακτική ροή – Ειδοποίηση μη προγραμματισμένης δραστηριότητας λόγω εγγύτητας σε τοποθεσία ενδιαφέροντος

Χρήστης	Εφαρμογή
	1. Στην οθόνη της συσκευής εμφανίζεται μία ειδοποίηση. 2. Η ειδοποίηση αφορά μία μη προγραμματισμένη δραστηριότητα.
3. Ο χρήστης επιλέγει να δει τις λεπτομέρειες της δραστηριότητας.	
	4. Προβολή του χάρτη στον οποίον σημειώνονται όλες οι τοποθεσίες που βρίσκονται σε κοντινή απόσταση, και ικανοποιούν κάποια ή κάποιες από τις δραστηριότητες του.
5. Ο χρήστης επιλέγει την τοποθεσία που τον ενδιαφέρει και πατάει το κουμπί της ετικέτας.	
	6. Εμφάνιση μηνύματος στο χρήστη αν επιθυμεί να πραγματοποιήσει τη δραστηριότητα.
7. Επιλογή από το χρήστη να πραγματοποιήσει τη δραστηριότητα	
	8. Η εφαρμογή επισημαίνει τη δραστηριότητα ως εκτελεσμένη. 9. Επιστροφή στη λίστα με τις μη προγραμματισμένες δραστηριότητες.

3.3.8.3. Διάγραμμα δραστηριοτήτων



Εικόνα 19 Προβολή ειδοποιήσεων.

4

Σχεδίαση

4.1. Η εφαρμογή MyActivities στο iOS

4.1.1. Οι οθόνες της εφαρμογής

Για την εφαρμογή που σχεδιάζουμε προκειμένου να καλυφθούν οι διάφορες λειτουργίες που έχουν περιγραφεί στην ανάλυση θα πρέπει να υπάρξουν κάποιες οθόνες. Αυτές είναι οι εξής:

- **Εναρκτήρια οθόνη**, στην οποία θα δίνεται στο χρήστη η επιλογή για σύνδεση και εγγραφή μέσω του server ή για σύνδεση μέσω του Facebook.
- **Οθόνη εγγραφής και σύνδεσης**, όπου ο χρήστης θα μπορεί είτε να δημιουργήσει νέο λογαριασμό στην εφαρμογή, είτε να συνδεθεί στον υπάρχοντα λογαριασμό του μέσω του server.
- **Οθόνη με το μενού της εφαρμογής**, στην οποία θα δίνεται στο χρήστη η επιλογή να μεταβεί στη λίστα με τις προγραμματισμένες ή μη προγραμματισμένες δραστηριότητες, στις αποθηκευμένες στη βάση δεδομένων της εφαρμογής τοποθεσίες, στο χάρτη με τις κοντινές τοποθεσίες που ενδιαφέρουν το χρήστη και στα στοιχεία των λογαριασμών του χρήστη στο server και στο Facebook.
- **Οθόνη με τις τοποθεσίες**, όπου υπάρχει η λίστα με τοποθεσίες που έχουν αποθηκευθεί στη βάση δεδομένων της εφαρμογής, και δίνεται η δυνατότητα προσθήκης νέας τοποθεσίας και νέας δραστηριότητας για επιλεγμένη τοποθεσία.
- **Οθόνη με τον κοντινό χάρτη**, όπου θα εμφανίζονται μαρκαρισμένες οι τοποθεσίες στις οποίες μπορεί ο χρήστης να πραγματοποιήσει κάποια προγραμματισμένη ή μη προγραμματισμένη δραστηριότητα και θα μπορεί να δηλώσει ότι πραγματοποίησε κάποια από αυτές.
- **Οθόνη με τα στοιχεία του λογαριασμού του χρήστη στο server**, όπου μπορεί να συνδεθεί είτε να αποσυνδεθεί.
- **Οθόνη με τα στοιχεία του λογαριασμού του χρήστη στο Facebook**, όπου δίνεται η δυνατότητα στο χρήστη να συνδεθεί ή να αποσυνδεθεί.
- **Οθόνη με τις προγραμματισμένες δραστηριότητες**, όπου εμφανίζονται όλες οι προγραμματισμένες δραστηριότητες του χρήστη και του δίνεται η δυνατότητα να μεταβεί στις λεπτομέρειες κάθε μίας και να την επεξεργαστεί ή να δηλώσει ότι πραγματοποίησε κάποια από αυτές. Μπορεί επίσης να μεταβεί στην οθόνη

προσθήκης κάποιας νέας τοποθεσίας και στην οθόνη προσθήκης νέας δραστηριότητας.

- **Οθόνη με τις μη προγραμματισμένες δραστηριότητες**, όπου εμφανίζονται όλες οι μη προγραμματισμένες δραστηριότητες του χρήστη και του δίνεται η δυνατότητα να μεταβεί στις λεπτομέρειες κάθε μίας και να την επεξεργαστεί ή να δηλώσει ότι πραγματοποίησε κάποια από αυτές. Μπορεί επίσης να μεταβεί στην οθόνη προσθήκης κάποιας νέας τοποθεσίας και στην οθόνη προσθήκης νέας δραστηριότητας.
- **Οθόνη προσθήκης νέας δραστηριότητας**, όπου ο χρήστης μπορεί να επιλέξει το είδος της δραστηριότητας που θέλει να προσθέσει και ανάλογα να μεταβεί στην οθόνη με τις λεπτομέρειες προγραμματισμένης ή μη προγραμματισμένης δραστηριότητας.
- **Οθόνη με τις λεπτομέρειες προγραμματισμένης δραστηριότητας**, όπου ο χρήστης μπορεί να επεξεργαστεί τον τίτλο της δραστηριότητας, να μεταβεί στην οθόνη για τη ρύθμιση της ημερομηνίας και ώρας έναρξής της, στην οθόνη με τη λίστα των τύπων δραστηριότητας και στην οθόνη επιλογής τοποθεσίας. Του δίνεται επίσης η δυνατότητα να αποθηκεύσει τη δραστηριότητα ή να τη διαγράψει.
- **Οθόνη με τις λεπτομέρειες μη προγραμματισμένης δραστηριότητας**, όπου ο χρήστης μπορεί να επεξεργαστεί τον τίτλο της δραστηριότητας, να μεταβεί στην οθόνη για τη ρύθμιση της διορίας της, στην οθόνη με τη λίστα των τύπων δραστηριότητας και στην οθόνη επιλογής τοποθεσίας. Του δίνεται επίσης η δυνατότητα να αποθηκεύσει τη δραστηριότητα ή να τη διαγράψει.
- **Οθόνη ρύθμισης ημερομηνίας και ώρας**, όπου ο χρήστης μπορεί να θέσει μία χρονική στιγμή σαν ώρα έναρξης μίας προγραμματισμένης δραστηριότητας, ή σαν διορία μέχρι την οποία θα πρέπει να έχει πραγματοποιηθεί μία μη προγραμματισμένη δραστηριότητα.
- **Οθόνη με τους τύπους δραστηριότητας**, όπου εμφανίζονται όλοι οι τύποι δραστηριοτήτων που έχουν αποθηκευθεί στη βάση δεδομένων της εφαρμογής και δίνεται στο χρήστη η δυνατότητα να επιλέξει τύπο για μία δραστηριότητα και να επιστρέψει στην οθόνη επεξεργασίας των λεπτομερειών της. Μπορεί επίσης να μεταβεί στην οθόνη προσθήκης νέου τύπου δραστηριότητας.
- **Οθόνη προσθήκης νέου τύπου δραστηριότητας**, όπου ο χρήστης μπορεί να επεξεργαστεί τον τίτλο και το χρόνο πριν από τον οποία θα ειδοποιείται για τέτοιου τύπου δραστηριότητες και να αποθηκεύσει το συγκεκριμένο τύπο δραστηριότητας.
- **Οθόνη προσθήκης νέας τοποθεσίας**, όπου παρουσιάζεται ο χάρτης και ανάλογα με την περίπτωση κάποιες τοποθεσίες που μπορεί να ενδιαφέρουν το χρήστη είναι μαρκαρισμένες. Φαίνεται η τωρινή θέση του χρήστη και μπορεί να σημειώσει ή να αναζητήσει όποια τοποθεσία επιθυμεί και να λάβει τα λεπτομερή στοιχεία της. Με το πάτημα της ετικέτας μίας τοποθεσίας μπορεί να την επιλέξει για κάποια δραστηριότητα ή να μεταβεί στην οθόνη με τις λεπτομέρειες αυτής της τοποθεσίας αν δεν έχει ήδη καταχωρηθεί. Υπάρχει επίσης η δυνατότητα μετάβασης στη λίστα με τις τοποθεσίες της εφαρμογής ή στη λίστα με τις κοντινές τοποθεσίες του Facebook.
- **Οθόνη με τις λεπτομέρειες τοποθεσίας**, όπου ο χρήστης για μία δεδομένη τοποθεσία μπορεί να επεξεργαστεί τον τίτλο της και να μεταβεί στη οθόνη με τους τύπους δραστηριότητας για να επιλέξει ποιόν από αυτούς μπορεί να ικανοποιήσει. Δίνεται η δυνατότητα επίσης όταν θα αποθηκεύσει την τοποθεσία να επιλέξει αν θα

αποθηκευθεί μόνο στη βάση δεδομένων της εφαρμογής ή θα αποσταλεί και προς το server.

- **Οθόνη με τη λίστα τοποθεσιών** που μπορεί να ενδιαφέρουν το χρήστη, όπου ανάλογα με την περίπτωση εμφανίζονται όλες η κάποιες από τις αποθηκευμένες τοποθεσίες του και ο χρήστης μπορεί να επιλέξει μία ή περισσότερες ανάλογα με το είδος δραστηριότητας.

4.1.2. Περιγραφή κλάσεων και αλληλεπιδράσεων

4.1.2.1. Κύρια κλάση

Όνομα κλάσης	**AppDelegate**	UIResponder
Μεταβλητές	*managedObjectContext*	NSManagedObjectContext
	managedObjectModel	NSManagedObjectModel
	persistentStoreCoordinator	NSPersistentStoreCoordinator
	backgroundTask	UIBackgroundTaskIdentifier
Μέθοδοι	• ***handleActiveStateNotifications*** Διαχείριση των ειδοποιήσεων όταν η εφαρμογή είναι στο προσκήνιο. • ***handleInactiveStateNotifications*** Διαχείριση των ειδοποιήσεων όταν η εφαρμογή είναι στο παρασκήνιο. • ***persistentStoreCoordinator*** Δημιουργία *persistentStoreCoordinator.* • ***managedObjectContex*** Δημιουργία *managedObjectContext.* • ***managedObjectModel*** Δημιουργία *managedObjectModel.* • ***sessionStateChanged*** Διαχείριση της κατάστασης του *session* του *Facebook*. • ***openSession*** Έναρξη του *session* του *Facebook*. • ***scheduleLocationBasedLocalNotification*** Δημιουργία ειδοποίησης για τις κοντινές τοποθεσίες που ενδιαφέρουν το χρήστη.	

4.1.2.2. Οθόνες

Όνομα κλάσης	**LoginViewController**	UIViewController
Μεταβλητές	*loginWithFacebookButton*	UIButton
	signUpButton	UIButton
Μέθοδοι	• ***loginWithFacebook*** Γίνεται σύνδεση στην εφαρμογή μέσω του *Facebook* και μετάβαση στην κεντρική οθόνη της εφαρμογής. • ***signUp*** Μετάβαση στην οθόνη εγγραφής και σύνδεσης μέσω του *server*. • ***loginFailed*** Χειρίζεται την αποτυχία σύνδεσης στο *Facebook*.	

Όνομα κλάσης	**SignUpViewController**	UITableViewController

Μεταβλητές	*goButton*	UIBarButtonItem
Μέθοδοι	• ***performLogin*** Γίνεται η σύνδεση μέσω του *server* ενός εγγεγραμμένου χρήστη. • ***performSignUp*** Γίνεται η εγγραφή ενός νέου χρήστη μέσω του *server*. • ***goButtonPressed*** Επιλέγεται η εγγραφή ή η σύνδεση του χρήστη.	

Όνομα κλάσης	**MenuViewController**	UITableViewController
Μεταβλητές	-	
Μέθοδοι	• ***presentScheduledActivityList*** Προβολή της λίστας με τις προγραμματισμένες δραστηριότητες. • ***presentNotScheduledActivityList*** Προβολή της λίστας με τις μη προγραμματισμένες δραστηριότητες. • ***presentLocations*** Προβολή της λίστας με τις τοποθεσίες. • ***presentMyActivitiesProfile*** Προβολή των στοιχείων του χρήστη. • ***presentCurrentMap*** Μετάβαση στο χάρτη και προβολή των κοντινών τοποθεσιών στις οποίες ο χρήστης έχει προγραμματίσει κάποια δραστηριότητα. • ***presentFacebookProfile*** Προβολή των στοιχείων του λογαριασμού του χρήστη στο *Facebook*.	

Όνομα κλάσης	**MyActivitiesViewController**	UITableViewController
Μεταβλητές	*stateButton*	UIButton
Μέθοδοι	• ***login*** Μετάβαση στη φόρμα εγγραφής και σύνδεσης. • ***logout*** Αποσύνδεση από το λογαριασμό του χρήστη. • ***stateButtonPressed*** Πραγματοποίηση της σύνδεσης ή αποσύνδεσης του χρήστη.	

Όνομα κλάσης	**FacebookViewController**	UITableViewController
Μεταβλητές	*stateButton*	UIButton
Μέθοδοι	• ***login*** Σύνδεση μέσω *Facebook* • ***logout*** Αποσύνδεση από το *Facebook*. • ***stateButtonPressed*** Πραγματοποίηση της σύνδεσης ή αποσύνδεσης του χρήστη από το *Facebook*.	

Όνομα κλάσης	**LocationsViewController**	UITableViewController

Μεταβλητές	*menuButton*	UIBarButtonItem
	addNewLocationButton	UIBarButtonItem
Μέθοδοι	• ***addNewLocation*** Μετάβαση στο χάρτη για προσθήκη νέας τοποθεσίας. • ***presentMenu*** Μετάβαση στο μενού της εφαρμογής. • ***fetchedResultsController*** Λήψη όλων των τοποθεσιών από τη βάση δεδομένων της εφαρμογής. • ***addActivityForLocation*** Μετάβαση στην οθόνη προσθήκης της δραστηριότητας για συγκεκριμένη τοποθεσία.	

Όνομα κλάσης	**CurrentMapViewController**	UIViewController
Μεταβλητές	*mapView*	MKMapView
	menuButton	UIBarButtonItem
Μέθοδοι	• ***getCloseActivities*** Λήψη των κοντινών τοποθεσιών που μπορεί ο χρήστης να πραγματοποιήσει κάποια δραστηριότητα. • ***addAnnotationsToMap*** Προσθήκη σημαδιών στο χάρτη στα σημεία που αντιστοιχούν οι τοποθεσίες που ελήφθησαν. • ***zoomToFitAnnotations*** Προσαρμογή της περιοχής προβολής του χάρτη στην οθόνη, ώστε να διακρίνονται όλες οι τοποθεσίες που ελήφθησαν. • ***checkActivityAsDone*** Μαρκάρισμα μίας δραστηριότητας ως εκτελεσμένη.	

Όνομα κλάσης	**ScheduledActivityListViewController**	UITableViewController
Μεταβλητές	*addNewActivityButton*	UIBarButtonItem
	menuButton	UIBarButtonItem
	addLocationButton	UIBarButtonItem
Μέθοδοι	• ***fetchedResultsController*** Λήψη όλων των προγραμματισμένων δραστηριοτήτων από τη βάση δεδομένων της συσκευής. • ***addNewLocation*** Μετάβαση στο χάρτη για προσθήκη νέας τοποθεσίας. • ***addNewActivity*** Μετάβαση στην οθόνη προσθήκης νέας δραστηριότητας. • ***presentMenu*** Μετάβαση στο μενού της εφαρμογής. • ***getNewLocationsFromServer*** Αποστολή αίτησης στο *server* και λήψη των τοποθεσιών που προστέθηκαν πρόσφατα σε αυτόν. • ***toggleDone*** Εναλλαγή της κατάστασης μίας δραστηριότητας, ως προς το αν έχει πραγματοποιηθεί ή όχι. • ***scheduleNotificationWithActivity*** Δημιουργία των ειδοποιήσεων για μία δραστηριότητα που είχε δηλωθεί ότι πραγματοποιήθηκε.	

Όνομα κλάσης	**NotScheduledActivityListViewController**	UIViewTableController
Μεταβλητές	*addNewActivityButton*	UIBarButtonItem
	menuButton	UIBarButtonItem
	addLocationButton	UIBarButtonItem
Μέθοδοι	• ***fetchedResultsController*** Λήψη όλων των μη προγραμματισμένων δραστηριοτήτων από τη βάση δεδομένων της συσκευής. • ***addNewLocation*** Μετάβαση στο χάρτη για προσθήκη νέας τοποθεσίας. • ***addNewActivity*** Μετάβαση στην οθόνη προσθήκης νέας δραστηριότητας. • ***presentMenu*** Μετάβαση στο μενού της εφαρμογής. • ***toggleDone*** Εναλλαγή της κατάστασης μίας δραστηριότητας, ως προς το αν έχει πραγματοποιηθεί ή όχι. • ***scheduleNotificationWithActivity*** Δημιουργία της ειδοποίησης για μία δραστηριότητα που είχε δηλωθεί ότι πραγματοποιήθηκε.	

Όνομα κλάσης	**ChooseKindOfActivityViewController**	UITableViewController
Μεταβλητές	*location*	Location
Μέθοδοι	• ***addActivity*** Δημιουργία νέας δραστηριότητας, προγραμματισμένης ή μη, ανάλογα την επιλογή του χρήστη.	

Όνομα κλάσης	**AddScheduledActivityViewController**	UITableViewController
Μεταβλητές	*aScheduledActivity*	NSManagedObject
	saveButton	UIBarButtonItem
	cancelButton	UIBarButtonItem
Μέθοδοι	• ***getUsersLocations*** Αποστολή αίτησης στο *server* και λήψη των τοποθεσιών που ικανοποιούν τον επιλεγμένο τύπο δραστηριότητας και έχουν προστεθεί από τους χρήστες. • ***createLocationAnnotation*** Δημιουργία ενός αντικειμένου *MapLocation* για την προβολή της επιλεγμένης τοποθεσίας. • ***createAnnotationArray*** Δημιουργία ενός πίνακα με αντικείμενα *MapLocation* για την προβολή των τοποθεσιών που ικανοποιούν αυτόν τον τύπο δραστηριότητας. • ***saveScheduledActivity*** Αποθήκευση της προγραμματισμένης δραστηριότητας και μετάβαση στην κεντρική οθόνη. • ***scheduleNotificationForActivity*** Δημιουργία των ειδοποιήσεων για τη δραστηριότητα • ***cancelActivity*** Διαγραφή της δραστηριότητας και μετάβαση στην κεντρική οθόνη. • ***cancelNotificationForActivity*** Διαγραφή των ειδοποιήσεων της δραστηριότητας.	

Όνομα κλάσης	**AddNotScheduledActivityViewController**	UITableViewController
Μεταβλητές	*aNotScheduledActivity*	NSManagedObject
	saveButton	UIBarButtonItem
	cancelButton	UIBarButtonItem
Μέθοδοι	• ***getUsersLocations*** Αποστολή αίτησης στο *server* και λήψη των τοποθεσιών που ικανοποιούν τον επιλεγμένο τύπο δραστηριότητας και έχουν προστεθεί από τους χρήστες. • ***createNotScheduledLocationArray*** Δημιουργία ενός πίνακα με αντικείμενα *MapLocation* για την προβολή των τοποθεσιών που έχουν επιλεγεί για τη συγκεκριμένη δραστηριότητα. • ***createAnnotationArray*** Δημιουργία ενός πίνακα με αντικείμενα *MapLocation* για την προβολή των τοποθεσιών που ικανοποιούν αυτόν τον τύπο δραστηριότητας. • ***saveNotScheduledActivity*** Αποθήκευση της μη προγραμματισμένης δραστηριότητας και μετάβαση στην κεντρική οθόνη. • ***scheduleExpiredNotification*** Δημιουργία των ειδοποιήσεων για τη δραστηριότητα • ***cancelActivity*** Διαγραφή της δραστηριότητας και μετάβαση στην κεντρική οθόνη. • ***deleteExpiredNotification*** Διαγραφή των ειδοποιήσεων της δραστηριότητας.	

Όνομα κλάσης	**EditDateViewController**	UITableViewController
Μεταβλητές	*datePicker*	IBOutletDatePicker
	anActivity	NSManagedObject
	saveButton	UIBarButtonItem
Μέθοδοι	• ***saveDate*** Αποθήκευση της ημερομηνίας και ώρας έναρξης ή διορίας της δραστηριότητας ανάλογα με το είδος της και μετάβαση στην οθόνη με τις λεπτομέρειες της.	

Όνομα κλάσης	**ActivityTypeListViewController**	UITableViewController
Μεταβλητές	*anActivity*	NSManagedObject
	addNewActivityTypeButton	UIBarButtonItem
Μέθοδοι	• ***fetchedResultsController*** Λήψη όλων των τύπων των δραστηριοτήτων από τη βάση δεδομένων της συσκευής. • ***didSelectAnActivityType*** Αποθήκευση του επιλεγμένου τύπου για τη δραστηριότητα και μετάβαση στην οθόνη με τις λεπτομέρειες της. • ***addNewActivityType*** Μετάβαση στη οθόνη δημιουργίας νέου τύπου δραστηριότητας.	

Όνομα κλάσης	**AddNewActivityTypeViewController**	UITableViewController
Μεταβλητές	*anActivity*	NSManagedObject
	saveButton	UIBarButtonItem
Μέθοδοι	• ***saveNewActivityType*** Δημιουργία εγγραφής τύπου δραστηριότητας στη βάση δεδομένων της συσκευής και αποθήκευση της.	

Όνομα κλάσης	**AddLocationViewController**	UIViewController
Μεταβλητές	*trackingButton*	MKUserTrackingBarButtonItem
	mapView	MKMapView
	searchBar	UISearchBar
	facebookButton	UIButton
	listButton	UIButton
	manager	CLLocationManager
	geocoder	ClGeocoder
	placemark	CLPlacemark
	location	NSManagedObject
	anActivity	NSManagedObject
	locationArray	NSMUtableArray
	notScheduledLocationArray	NSMutableArray
	locationAnnotation	MapLocation
	placePickerController	FBPlacePickerViewController
	selectedPlace	NSObject<FBGraphPlace>
Μέθοδοι	• ***facebookButtonVisible*** Εμφάνιση ή μη του κουμπιού μετάβασης στις τοποθεσίες του *Facebook*. • ***listButtonVisible*** Εμφάνιση ή μη του κουμπιού μετάβασης στη λίστα με τις τοποθεσίες. • ***handleAnnotations*** Επιλογή των σημείων που θα εμφανιστούν στο χάρτη. • ***addAnnotationsToMap*** Προσθήκη σημαδιών στο χάρτη στα σημεία που αντιστοιχούν οι τοποθεσίες που έχουν επιλεγεί. • ***zoomToFitMapAnnotations*** Προσαρμογή της περιοχής προβολής του χάρτη στην οθόνη, ώστε να διακρίνονται όλες οι τοποθεσίες που επελέγησαν. • ***reverseGeocode*** Λήψη λεπτομερειών για συγκεκριμένη τοποθεσία στο χάρτη. • *openCallout:* Άνοιγμα της ετικέτας για μαρκαρισμένο σημείο • ***handleCalloutAccesoryControlTapped*** Διαχείριση του κουμπιού της ετικέτας. • ***handleLongPress*** Διαχείριση παρατεταμένου πατήματος της οθόνης. • ***presentList*** Προβολή της λίστας με τις τοποθεσίες που ενδιαφέρουν το χρήστη. • ***presentFacebookPlaces*** Προβολή των τοποθεσιών του *Facebook* που βρίσκοναται κοντά στο χρήστη.	

Όνομα κλάσης	**AddLocationDetailsViewController**	UITableViewController
Μεταβλητές	*aLocation*	NSManagedObject
	anActivity	NSManagedObject
	saveButton	UIBarButtonItem
	cancelButton	UIBarButtonItem
	publicButton	UISwitch
Μέθοδοι	• ***saveLocation*** Δημιουργία εγγραφής τοποθεσίας στη βάση δεδομένων της συσκευής και αποθήκευση της. • ***sendLocationToServer*** Αποστολή στο *server* της τοποθεσίας. • ***sendActivityTypeToServer*** Αποστολή στο *server* του τύπου που ικανοποιεί η τοποθεσία. • ***cancelLocation*** Διαγραφή της τοποθεσίας.	

Όνομα κλάσης	**LocationListViewController**	UIViewController
Μεταβλητές	*anActivity*	NSMangedObject
	okButton	UIBarButtonItem
Μέθοδοι	• ***fetchedResultsController*** Λήψη όλων των τοποθεσιών για τις οποίες ενδιαφέρεται ο χρήστης. • ***handleNotScheduledActivityLocations*** Δυνατότητα πολλαπλής επιλογής τοποθεσιών για τη μη προγραμματισμένη δραστηριότητα. • ***handleScheduledActivityLocation*** Αποθήκευση της επιλεγμένης τοποθεσίας για τη προγραμματισμένη δραστηριότητα και μετάβαση στην οθόνη με τις λεπτομέρειες της. • ***saveLocations*** Αποθήκευση τοποθεσιών για τη μη προγραμματισμένη δραστηριότητα και μετάβαση στην οθόνη με τις λεπτομέρειές της.	

4.1.2.3. Βοηθητικές κλάσεις

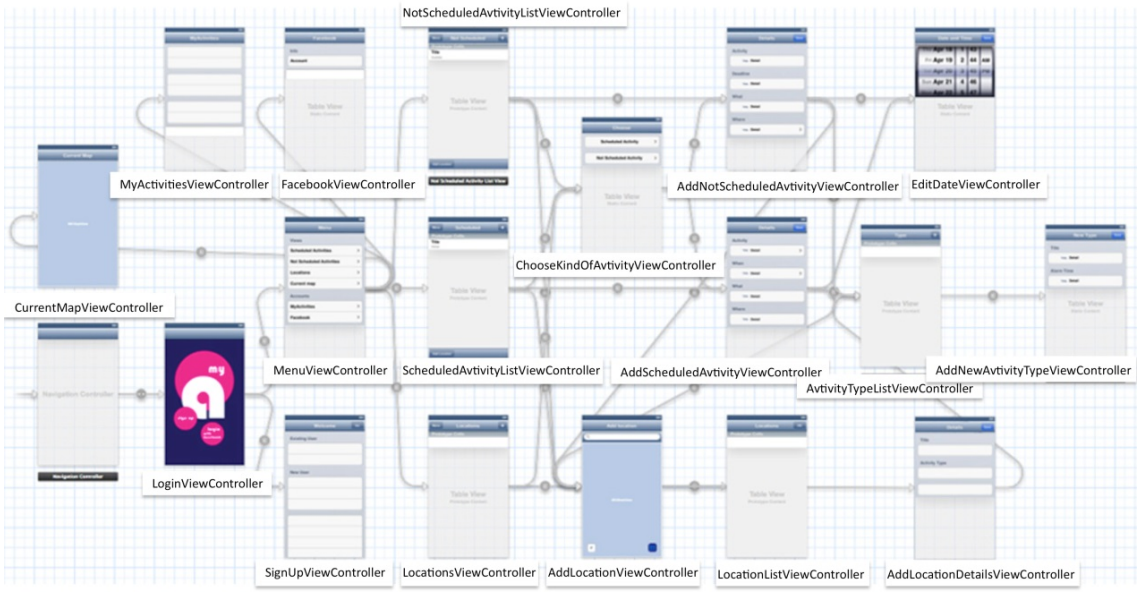
Όνομα κλάσης	**MapLocation**	NSObject
Μεταβλητές	*theTitle*	NSString
	theSubtitle	NSString
	state	NSString
	zip	NSString
	street	NSString
	city	NSString
	coordinate	CLLocationCoordinate2D
Μέθοδοι	-	

Όνομα κλάσης	**EditCell**	UITableViewCell
Μεταβλητές	*label*	UILabel
	textField	UITextField

Μέθοδοι	• ***editLabel*** Διαμόρφωση του *label*. • ***editTextField*** Διαμόρφωση του *textField*.

Όνομα κλάσης	**SignUpCell**	UITableViewCell
Μεταβλητές	*textField*	UITextField
Μέθοδοι	• ***editTextField*** Διαμόρφωση του *textField*.	

4.1.2.4. Αλληλεπιδράσεις μεταξύ των κλάσεων



Εικόνα 20 Διάγραμμα αλληλεπιδράσεων.

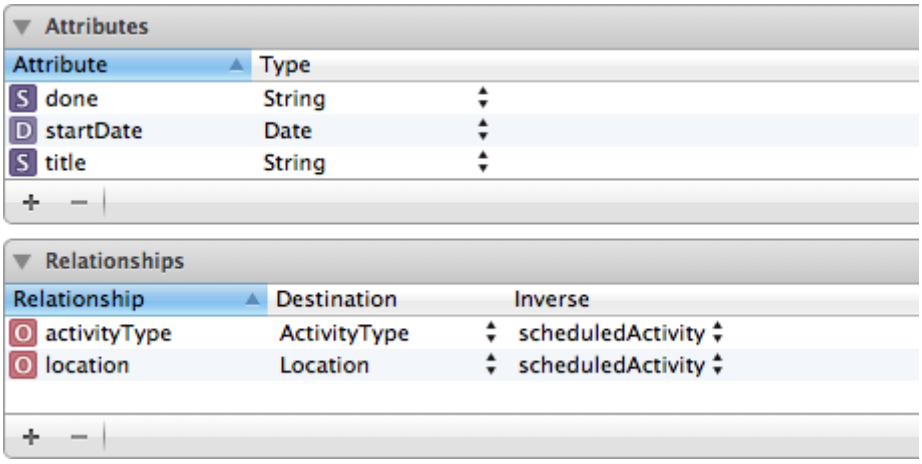
4.1.3. Σχήμα βάσης δεδομένων

Όσον αφορά τη βάση δεδομένων της εφαρμογής οι οντότητες που περιγράφονται είναι οι εξής:

i. Προγραμματισμένη δραστηριότητα (Scheduled Activity)
ii. Μη προγραμματισμένη δραστηριότητα (Not Scheduled Activity)
iii. Τοποθεσία (Location)
iv. Τύπος δραστηριότητας (Activity Type)

Κάθε μία από αυτές τις οντότητες έχει τα δικά της χαρακτηριστικά (attributes) και συνδέονται μεταξύ τους με σχέσεις που είναι πολύ σημαντικές για τη λειτουργία της εφαρμογής. Στη συνέχεια θα ασχοληθούμε με κάθε μία ξεχωριστά.

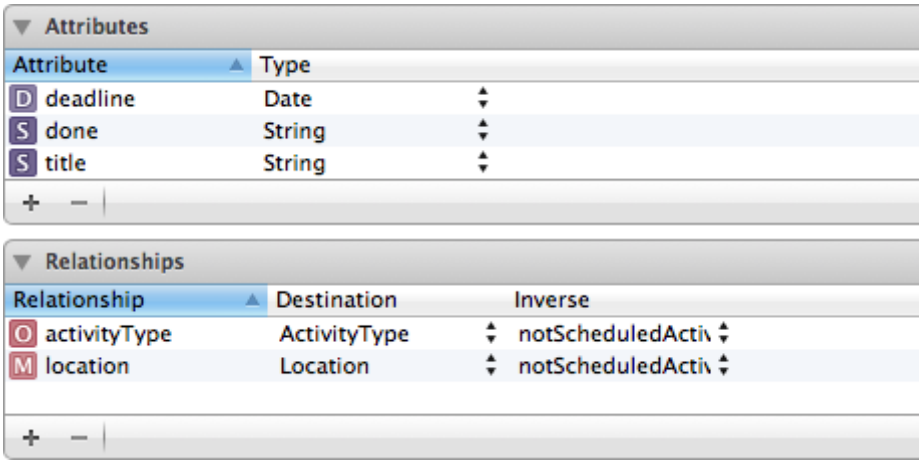
i. Προγραμματισμένη δραστηριότητα (Scheduled Activity)



Εικόνα 21 Ο πίνακας προγραμματισμένων δραστηριοτήτων και οι σχέσεις του με άλλους πίνακες.

Τα χαρακτηριστικά μιας προγραμματισμένης δραστηριότητας είναι ο τίτλος που θα της δώσει ο χρήστης, η ημερομηνία και ώρα έναξης που θα θέσει για την πραγματοποίησή της και το αν έχει πραγματοποιηθεί ή όχι η δραστηριότητα. Ο τίτλος είναι τύπου String, η ημερομηνία έναρξης είναι τύπου Date και το αν έχει πραγματοποιηθεί είναι τύπου String. Ο χρήστης μπορεί επίσης να καθορίσει και το περιεχόμενο των οντοτήτων που έχουν κάποια σχέση με τη δραστηριότητα, δηλαδή την τοποθεσία και τον τύπο της δραστηριότητας. Κάθε προγραμματισμένη δραστηριότητα μπορεί να αντιστοιχεί σε ένα τύπο και μία τοποθεσία. Ο τίτλος είναι το μόνο πεδίο που πρέπει να δωθεί κάποια τιμή υποχρεωτικά.

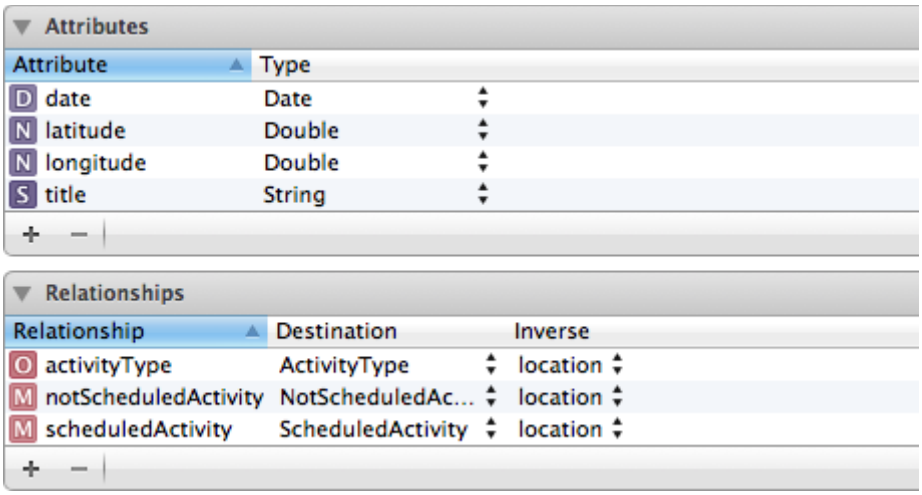
ii. Μη προγραμματισμένη δραστηριότητα (Not Scheduled Activity)



Εικόνα 22 Ο πίνακας μη προγραμματισμένων δραστηριοτήτων και οι σχέσεις του με άλλους πίνακες.

Τα χαρακτηριστικά μίας μη προγραμματισμένης δραστηριότητας είναι ο τίτλος και η διορία που θα της δώσει ο χρήστης και το αν έχει πραγματοποιηθεί ή όχι. Ο τίτλος είναι τύπου String, η διορία τύπου Date και το αν έχει πραγματοποιηθεί τύπου String. Ο χρήστης μπορεί επίσης να καθορίσει και το περιεχόμενο των οντοτήτων που έχουν κάποια σχέση με τη δραστηριότητα, δηλαδή την τοποθεσία και τον τύπο της δραστηριότητας. Κάθε μη προγραμματισμένη δραστηριότητα μπορεί να αντιστοιχεί σε ένα τύπο και μία ή περισσότερες τοποθεσίες. Ο τίτλος είναι το μόνο πεδίο που πρέπει να δωθεί κάποια τιμή υποχρεωτικά.

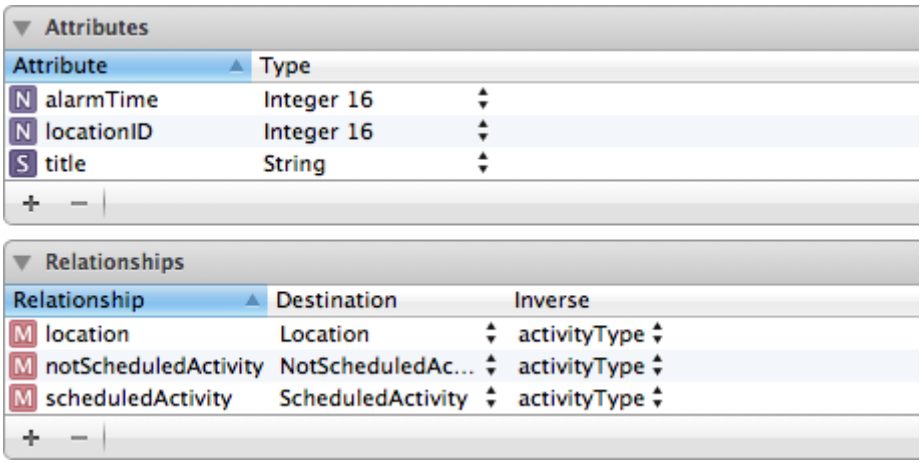
iii. Τοποθεσία (Location)



Εικόνα 23 Ο πίνακας τοποθεσιών και οι σχέσεις του με άλλους πίνακες.

Τα χαρακτηριστικά μίας τοποθεσίας είναι το γεωγραφικό της μήκος, το γεωγραφικό της πλάτος, ο τίτλος της και η τελευταία ημερομηνία που έγινε έλεγχος για αυτή την τοποθεσία. Το γεωγραφικό μήκος και πλάτος είναι τύπου Double, ο τίτλος είναι τύπου String και η ημερομηνία ελέγχου είναι τύπου Date. Οι τοποθεσίες μπορούν να συνδέονται με τις οντότητες με τις οποίες έχουν "relationship", δηλαδή τον τύπο δραστηριότητας, την προγραμματισμένη και τη μη προγραμματισμένη δραστηριότητα. Κάθε τοποθεσία μπορεί να συνδέεται με ένα μόνο τύπο δραστηριοτήτων και με μία ή περισσότερες προγραμματισμένες ή μη προγραμματισμένες δραστηριότητες. Ο τίτλος είναι το μόνο πεδίο που πρέπει να δωθεί κάποια τιμή υποχρεωτικά.

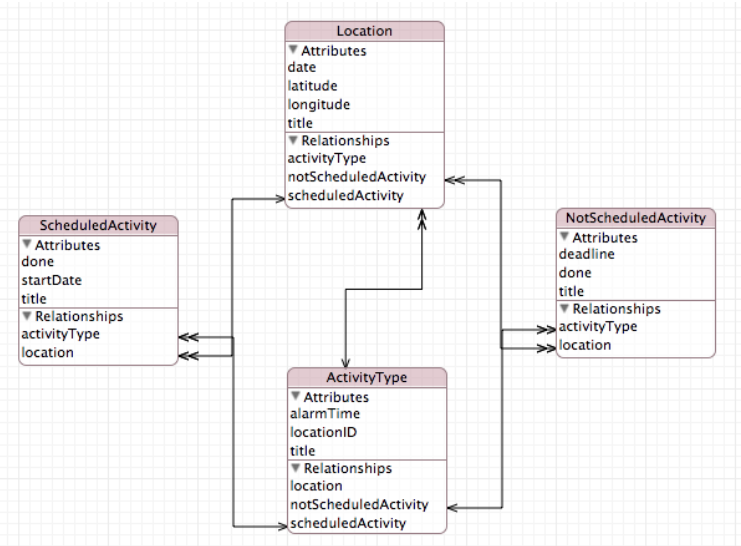
iv. Τύπος δραστηριότητας (Activity Type)



Εικόνα 24 Ο πίνακας τύπων δραστηριοτήτων και οι σχέσεις του με άλλους πίνακες.

Τα χαρακτηριστικά του τύπου δραστηριότητας είναι ο τίτλος του και τα λεπτά πριν από τα οποία θα ειδοποιείται ο χρήστης για αυτού του τύπου τις προγραμματισμένες δραστηριότητες. Ο τίτλος είναι τύπου String και τα λεπτά ειδοποίησης είναι τύπου Integer. Οι τύποι δραστηριοτήτων μπορούν να συνδέονται με τις οντότητες με τις οποίες έχουν "relationship", δηλαδή με τοποθεσίες, προγραμματισμένες και μη προγραμματισμένες δραστηριότητες. Κάθε τύπος μπορεί να αντιστοιχεί σε μία ή περισσότερες τοποθεσίες και μία ή περισσότερες προγραμματισμένες ή μη προγραμματισμένες δραστηριότητες. Και εδώ το μοναδικό πεδίο που πρέπει να συμπληρωθεί υποχρεωτικά είναι ο τίτλος.

Συνδιάζοντας όλα τα παραπάνω, το σχήμα της βάσης δεδομένων της εφαρμογής είναι όπως στην παρακάτω εικόνα**.**



Εικόνα 25 Το σχήμα της βάσης δεδομένων της εφαρμογής.

4.2. Ο server της εφαρμογής

4.2.1. Λειτουργίες του server

Σε πολλές από τις λειτουργίες της εφαρμογής είναι πολύ σημαντικός ο ρόλος του server, τόσο για τις περισσότερες δυνατότητες που προσφέρει όσο και για την αλληλεπίδραση που δημιουργείται μεταξύ των χρηστών. Συγκεκριμένα ο ρόλος του server στην εφαρμογή φαίνεται στις παρακάτω λειτουργίες:

- Ο χρήστης χρησιμοποιεί την εφαρμογή για πρώτη φορά και κάνει **εγγραφή** συμπληρώνοντας τα στοιχεία του λογαριασμού του και αποστέλλοντας αίτηση αποδοχής τους στο server. Ο server στη συνέχεια απαντά στο χρήστη αν η αίτησή του έγινε δεκτή ή όχι και το λόγο που συνέβη αυτό.
- Ο χρήστης επιλέγει να **συνδεθεί** μέσω του server στην εφαρμογή. Στέλνει το όνομα χρήστη και τον κωδικό του και ζητά από το server να συνδεθεί. Αν τα στοιχεία είναι έγκυρα ο server επιτρέπει τη σύνδεση, αλλιώς ενημερώνει το χρήστη γιατί συνέβη αυτό.
- Ο χρήστης όταν βρεθεί στην κεντρική οθόνη της εφαρμογής **ζητά** από το server να τον ενημερώσει για τις νεότερες **τοποθεσίες που έχουν προστεθεί** στη βάση δεδομένων του και δεν έχουν καταχωρηθεί από τους χρήστες. Ζητούνται μόνο εκείνες που δεν έχουν ήδη αποθηκευθεί στη βάση δεδομένων της εφαρμογής. Φυσικά αποστέλλονται και οι τύποι δραστηριοτήτων που μπορεί να ικανοποιήσει η κάθε τοποθεσία.
- Ο χρήστης όταν βρίσκεται στο στάδιο επεξεργασίας μίας δραστηριότητας και επιλέξει τύπο δραστηριότητας **ζητά** από το server τις **τοποθεσίες** που έχουν προστεθεί από τους χρήστες και έχει δηλωθεί ότι ικανοποιούν το συγκεκριμένο **τύπο δραστηριότητας**.
- Ο χρήστης μπορεί να επιλέξει να **αποθηκευθεί** μια **τοποθεσία** στη βάση δεδομένων του server εκτός από τη βάση δεδομένων τις εφαρμογής. Τότε, αν ο χρήστης δεν έχει επιλέξει τύπο δραστηριότητας που ικανοποιεί η τοποθεσία που προσθέτει, αποστέλλεται και αποθηκεύεται μόνο η τοποθεσία στο server. Αν ο χρήστης έχει επιλέξει τύπο δραστηριότητας τότε στη βάση δεδομένων του server συνδέονται η τοποθεσία που προστέθηκε και ο συγκεκριμένος τύπος. Αν ο τύπος δραστηριότητας που ικανοποιεί η τοποθεσία δεν υπάρχει στο server δημιουργείται νέα εγγραφή γι' αυτόν τον τύπο δραστηριότητας.

4.2.2. Περιγραφή κλάσεων

Όνομα κλάσης	**RegisterServlet**	HttpServlet
POST	• ***/register?data=<JSONdata>*** Λαμβάνονται τα στοιχεία με τα οποία ζητείται να δημιουργηθεί νέα εγγραφή χρήστη και ελέγχεται αν το όνομα χρήστη και η διεύθυνση ηλεκτρονικού ταχυδρομείου χρησιμοποιούνται ήδη από κάποιο άλλο εγγεγραμμένο χρήστη. Αν δε χρησιμοποιούνται, δημιουργείται νέα εγγραφή χρήστη στη βάση δεδομένων του server, αλλιώς ενημερώνεται ο χρήστης για την τροποποίηση των πεδίων της εγγραφής.	

Όνομα κλάσης	**LoginServlet**	HttpServlet
POST	• ***/login?data=<JSONdata>*** Λαμβάνονται το όνομα χρήστη και ο κωδικός με τον οποίο προσπαθεί να συνδεθεί ο χρήστης στην εφαρμογή. Αν τα στοιχεία αυτά επαληθεύονται για κάποια εγγραφή χρήστη στη βάση δεδομένων του server, τότε δίνεται η άδεια σύνδεσης και αποστέλλονται τα στοιχεία της συγκεκριμένης εγγραφής στο χρήστη. Αλλιώς αποστέλλεται μήνυμα για τη σωστή συμπλήρωση των στοιχείων.	

Όνομα κλάσης	**LocationActivityTypeServlet**	HttpServlet
GET	• ***/locationActivutyType?locationId=<id>&users=<boolean>*** Αποστέλλεται JSONArray με τις τοποθεσίες και τον τύπο δραστηριότητας που ικανοποιούν. Οι τοποθεσίες αυτές πρέπει να έχουν αύξοντα αριθμό μεγαλύτερο από id. Ενώ εάν η παράμετρος users είναι false, τότε αποστέλλονται μόνο οι τοποθεσίες που δεν έχουν προέλθει από τους χρήστες. • ***/locationActivityType?activityType=<activityTypeTitle>*** Αποστέλλεται JSONArray με τις τοποθεσίες που έχουν προστεθεί από τους χρήστες και ικανοποιούν τον τύπο δραστηριότητας που ζητείται.	
POST	• ***/ locationActivityType?data=<JSONdata>*** Ανάλογα με τη δομή JSON που αποστέλλει ο χρήστης υπάρχουν τρεις περιπτώσεις: o Εάν περιέχει μια τοποθεσία, αυτή αποθηκεύεται στη βάση δεδομένων του server και δε συνδέεται με κάποιο τύπο δραστηριότητας. o Εάν περιέχει μια τοποθεσία και τον τύπο δραστηριότητας που ικανοποιεί, τότε αυτή αποθηκεύεται και συνδέεται με τον συγκεκριμένο τύπο δραστηριότητας. Σε περίπτωση που πρόκειται για νέο τύπο δραστηριότητας, τότε δημιουργείται κι μια νέα εγγραφή στη βάση και για αυτόν.	

4.2.3. Σχήμα βάσης δεδομένων

Για τη βάση δεδομένων του server οι οντότητες που περιγράφονται είναι οι εξής:

i. Τοποθεσία (Location)

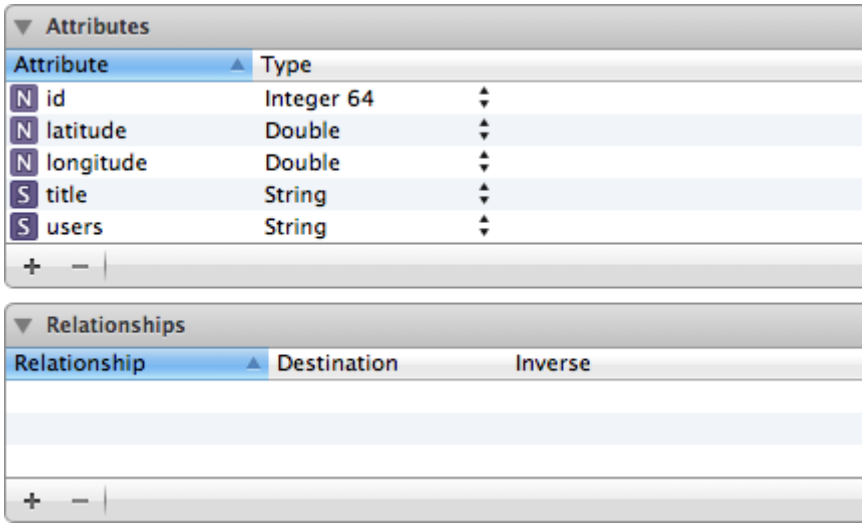
ii. Τύπος δραστηριότητας (ActivityType)

iii. Σύνδεση τοποθεσίας και τύπου δραστηριότητας (LocationActivityType)

iv. Χρήστης (User)

Όπως και για τη βάση δεδομένων της εφαρμογής έτσι και για τη βάση δεδομένων του server θα εξετάσουμε κάθε μία από τις οντότητες ξεχωριστά.

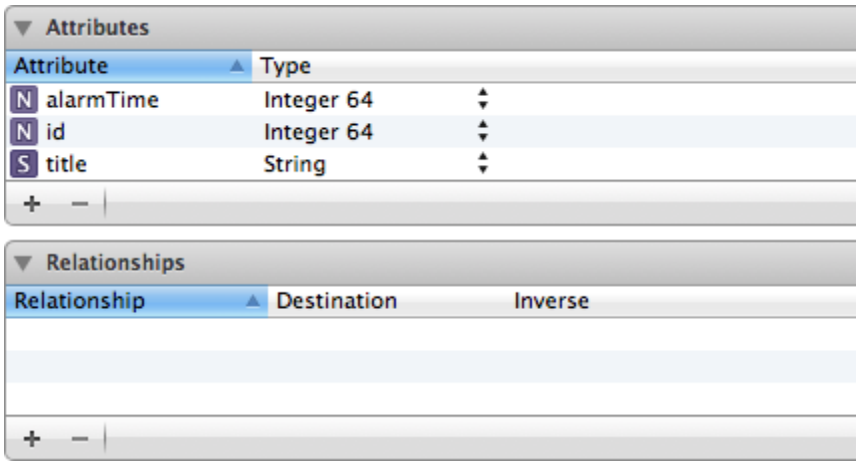
i. Τοποθεσία (Location)



Εικόνα 26 Ο πίνακας τοποθεσιών.

Τα χαρακτηριστικά μίας τοποθεσίας είναι το γεωγραφικό μήκος, το γεωγραφικό πλάτος, ο τίτλος, ο αύξων αριθμός και το αν προέρχεται από τους χρήστες ή όχι. Ο τίτλος είναι τύπου String, το γεωγραφικό μήκος και πλάτος τύπου Double, ο αύξων αριθμός τύπου Integer και το αν προέρχεται από χρήστες τύπου String.

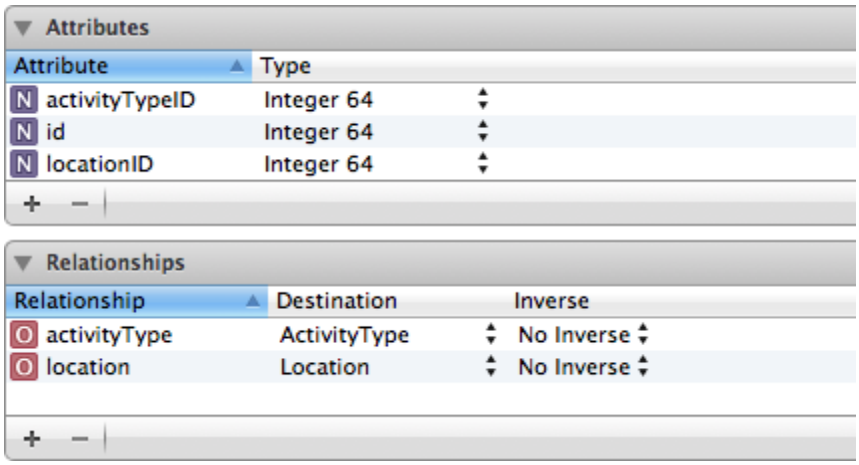
ii. Τύπος δραστηριότητας (ActivityType)



Εικόνα 27 Ο πίνακας τύπων δραστηριοτήτων

Τα χαρακτηριστικά ενός τύπου δραστηριότητας είναι ο τίτλος, τα λεπτά πριν από τα οποία θα ειδοποιείται ο χρήστης για τέτοιου τύπου δραστηριότητες και ο αύξων αριθμός. Ο τίτλος είναι τύπου String, ο αύξων αριθμός τύπου Integer και τα λεπτά ειδοποίησης τύπου Integer.

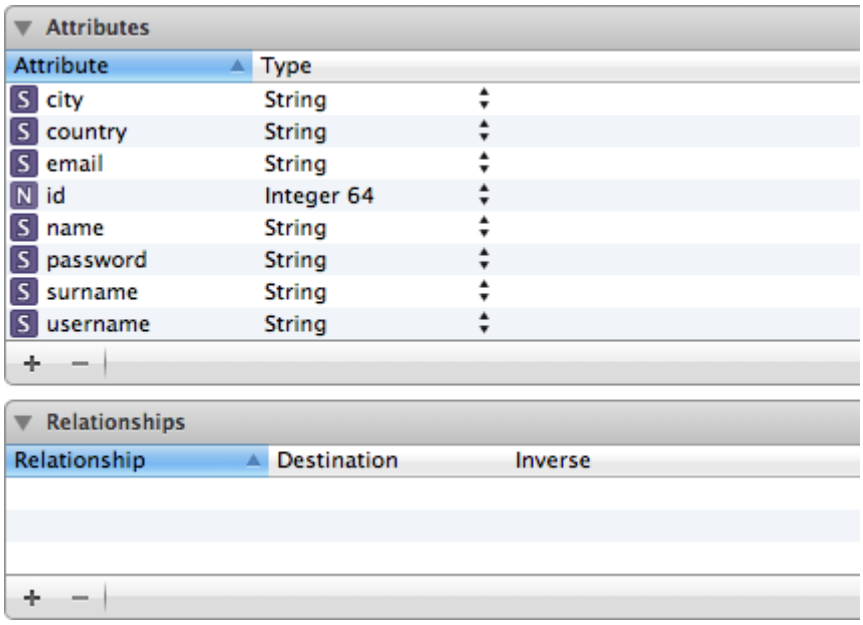
iii. Σύνδεση τοποθεσίας και τύπου δραστηριότητας (LocationActivityType)



Εικόνα 28 Ο βοηθητικός πίνακας σύνδεσης Τοποθεσιών με Τύπους Δραστηριοτήτων

Τα χαρακτηριστικά αυτής της οντότητας είναι ο αύξων αριθμός της, ο αύξων αριθμός της τοποθεσίας και του τύπου στον οποίο αντιστοιχεί. Φυσικά και οι τρεις αύξοντες αριθμοί είναι τύπου Integer. Έχει "relationship" με τον τύπο δραστηριότητας και την τοποθεσία και μπορεί να αντιστοιχεί μόνο σε ένα από αυτά.

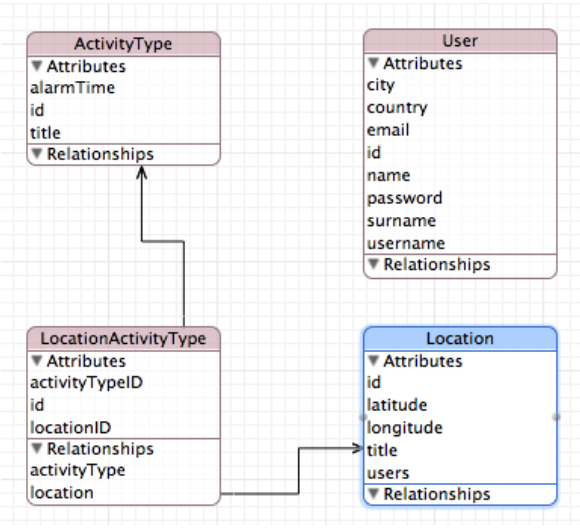
iv. Χρήστης (User)



Εικόνα 29 Ο πίνακας χρηστών

Για κάθε χρήστη που έχει εγγραφεί στο server έχει αποθηκευθεί το όνομα και το επώνυμό του, η χώρα και πόλη που μένει, η διεύθυνση ηλεκτρονικού ταχυδρομείου, το όνομα χρήστη που θα χρησιμοποιεί, ο κωδικός πρόσβασής του και ο αύξων αριθμός. Όλα τα χαρακτηριστικά είναι τύπου String εκτός από τον αύξοντα αριθμό που είναι τύπου Integer.

Συνδιάζοντας όλα τα παραπάνω, το σχήμα της βάσης δεδομένων της εφαρμογής είναι όπως στην παρακάτω εικόνα**.**



Εικόνα 30 Το σχήμα της βάσης δεδομένων του server

5

Υλοποίηση

5.1. Υλοποίηση της εφαρμογής

Υλοποιώντας την εφαρμογή για την οποία έγινε ανάλυση και σχεδίαση στα προηγούμενα κεφάλαια χρειάστηκε να έρθουμε σε επαφή με διάφορα Frameworks που υπάρχουν στο XCode. Ιδιαίτερα ενδιαφέρουσα ήταν η περιήγηση στις διάφορες λειτουργίες για τις οποίες χρειάστηκαν τα Frameworks MapKit και CoreLocation, δηλαδή για τις location-based λειτουργίες της εφαρμογής. Για τη διαχείριση της βάσης δεδομένων της εφαρμογής χρησιμοποιήθηκε το Framework CoreData και για τις λειτουργίες που αφορούν στο Facebook το Facebook SDK. Στη συνέχεια θα γίνει μία προσπάθεια μέσα από την ανάλυση κάποιων μεθόδων που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής να φανεί ο τρόπος που αυτά τα Frameworks μπορούν να εξυπηρετήσουν τις ανάγκες του προγραμματιστή και πώς εμείς επιλέξαμε να τα χρησιμοποιήσουμε.

Προσαρμογή χάρτη ώστε να φαίνονται επιλεγμένα σημεία

Μέθοδοι	handleAnnotations, addAnnotationsToMap, zoomToFitMapAnnotations
Κλάση	addLocationViewController

```
- (void)handleAnnotations:(NSArray *)array
{
    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    NSInvocationOperation *addAnnotationsToMap = [[NSInvocationOperation alloc]
initWithTarget:self selector:@selector(addAnnotationsToMap:) object:array];
    [queue addOperation:addAnnotationsToMap];
    NSInvocationOperation *zoomOperation = [[NSInvocationOperation alloc]
initWithTarget:self selector:@selector(zoomToFitMapAnnotations) object:nil];
    [zoomOperation addDependency:addAnnotationsToMap];
    [queue addOperation:zoomOperation];
}
```

```
- (void)addAnnotationsToMap:(NSArray *)array
{
    [self.mapView addAnnotations:array];
}
```

```
- (void)zoomToFitMapAnnotations
{
    MKMapRect zoomRect = MKMapRectNull;
    for (id <MKAnnotation> annotation in self.mapView.annotations)
    {
        MKMapPoint annotationPoint = MKMapPointForCoordinate(annotation.coordinate);
        MKMapRect pointRect = MKMapRectMake(annotationPoint.x, annotationPoint.y, 0.1,
0.1);
        zoomRect = MKMapRectUnion(zoomRect, pointRect);
    }
    MKCoordinateRegion region = MKCoordinateRegionForMapRect(zoomRect);
    CLLocationCoordinate2D topLeftCoord;
    topLeftCoord.latitude = -90;
    topLeftCoord.longitude = 180;
    CLLocationCoordinate2D bottomRightCoord;
    bottomRightCoord.latitude = 90;
    bottomRightCoord.longitude = -180;
    for(id<MKAnnotation> annotation in self.mapView.annotations) {
        topLeftCoord.longitude = fmin(topLeftCoord.longitude,
annotation.coordinate.longitude);
        topLeftCoord.latitude = fmax(topLeftCoord.latitude,
annotation.coordinate.latitude);
        bottomRightCoord.longitude = fmax(bottomRightCoord.longitude,
annotation.coordinate.longitude);
        bottomRightCoord.latitude = fmin(bottomRightCoord.latitude,
annotation.coordinate.latitude);
    }
    region.center.latitude = topLeftCoord.latitude - (topLeftCoord.latitude -
bottomRightCoord.latitude) * 0.5;
    region.center.longitude = topLeftCoord.longitude + (bottomRightCoord.longitude -
topLeftCoord.longitude) * 0.5;
    region.span.latitudeDelta = fabs(topLeftCoord.latitude -
bottomRightCoord.latitude) * 1.5;
    region.span.longitudeDelta = fabs(bottomRightCoord.longitude -
topLeftCoord.longitude) * 1.5;
    [self.mapView setRegion:region animated:YES];
}
```

Η μέθοδος *executeOperationQueue* δέχεται σαν όρισμα τον πίνακα με τα σημεία τα οποία θέλουμε να φανούν σαν «μαρκαρισμένα» στο χάρτη. Αφού κληθεί δημιουργείται μία *NSOperationQueue* (σειρά από επεξεργασίες) μέσω της οποίας θέλουμε να προστεθούν ως μαρκαρισμένα τα σημεία του χάρτη που έχουν επιλεγεί (*addAnnotationsToMap*) και να προσαρμοσθεί το μέγεθος της περιοχής την οποία απεικονίζει ο χάρτης ώστε να φαίνονται τα συγκεκριμένα σημεία (*zoomToFitMapAnnotations*). Στόχος όμως είναι η zoomToFitMapAnnotations να ξεκινήσει αφού έχει ολοκληρωθεί η *addAnnotationsToMap*, και αυτό επιτυγχάνεται μέσω μίας εξάρτησης της μίας μεθόδου από την άλλη *([zoomOperatio addDependency:addAnnotationsToMap]*). Στη *zoomToFitMapAnnotations* λαμβάνοντας υπ' όψιν τα σημεία που θέλουμε να φαίνονται στο χάρτη, βρίσκοντας αυτό που είναι πιο «πάνω και αριστερά» και εκείνο που είναι πιο «κάτω και δεξιά», φροντίζοντας ταυτόχρονα να υπάρχει και κάποιο περιθώριο, ορίζουμε την περιοχή που θα απεικονίζει ο χάρτης.

Δημιουργία location-based ειδοποίησης

Μέθοδοι	applicationDidEnterBackground, locationManager didUpdateLocations
Κλάση	appDelegate

```
- (void)applicationDidEnterBackground:(UIApplication *)application
{
    dispatch_block_t expirationHandler;
    expirationHandler = ^{
        [application endBackgroundTask:bgTask];
        bgTask = UIBackgroundTaskInvalid;
        bgTask = [application
beginBackgroundTaskWithExpirationHandler:expirationHandler];
    };
    bgTask = [application beginBackgroundTaskWithExpirationHandler:expirationHandler];
    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
        [[NSNotificationCenter defaultCenter]
postNotificationName:@"MyApplicationEntersBackground" object:self];
        while ([[UIApplication sharedApplication] applicationState] ==
UIApplicationStateInactive) {
            [locationManager startMonitoringSignificantLocationChanges];
            sleep(60);
        }
    });
}
```

```
- (void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray
*)locations
{
```

```
    NSError *error = nil;
    CLLocation *currentLocation = [locations lastObject];
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
    NSEntityDescription *entity = [NSEntityDescription
entityForName:@"NotScheduledActivity"

inManagedObjectContext:_managedObjectContext];
    [fetchRequest setEntity:entity];
    NSString *noString = @"NO";
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"done == %@",
noString];
    [fetchRequest setPredicate:predicate];
    NSArray *fetchedObjects = [_managedObjectContext executeFetchRequest:fetchRequest
                                                                    error:&error];

    NSMutableDictionary *infoDict = [[NSMutableDictionary alloc] init];
    if (fetchedObjects == nil) {
    } else {
        for (NotScheduledActivity *activity in fetchedObjects) {
            NSMutableArray *notificationArray = [[NSMutableArray alloc] init];
            NSMutableArray *array;
            if ([[activity valueForKey:@"Location"] count] != 0) {
                array = [NSMutableArray arrayWithArray:[[activity
valueForKey:@"Location"] allObjects]];
            } else {
                if ([activity valueForKey:@"ActivityType"] != nil) {
                    NSFetchRequest *fetchRequest1 = [[NSFetchRequest alloc] init];
                    NSEntityDescription *entity1 = [NSEntityDescription
entityForName:@"Location"

inManagedObjectContext:_managedObjectContext];
                    [fetchRequest1 setEntity:entity1];
                    NSString *activityType = [[activity valueForKey:@"ActivityType"]
valueForKey:@"title"];
                    NSPredicate *predicate1 = [NSPredicate
predicateWithFormat:@"activityType.title == %@", activityType];
                    [fetchRequest1 setPredicate:predicate1];
                    array = [NSMutableArray arrayWithArray:[_managedObjectContext
executeFetchRequest:fetchRequest1 error:&error]];
                }
            }
            for (Location *location in array) {
                if ((([[location valueForKey:@"date"] timeIntervalSince1970] + 1800) <
```

```
[[NSDate date] timeIntervalSince1970]) {
                    [location setValue:[NSDate date] forKey:@"date"];
                    CLLocation *locat = [[CLLocation alloc] initWithLatitude:[location
latitude].doubleValue longitude:[location longitude].doubleValue];
                    if ([currentLocation distanceFromLocation:locat] < 500){
                        NSDictionary *notificationDictionary = [NSDictionary
dictionaryWithObjectsAndKeys:location.latitude, @"latitude", location.longitude,
@"longitude", location.title, @"title", nil];
                        [notificationArray addObject:notificationDictionary];
                    }
                } else {
                    [location setValue:[NSDate date] forKey:@"date"];
                }
            }
            if (notificationArray.count != 0) {
                [infoDict setObject:notificationArray forKey:[activity
valueForKey:@"title"]];
            }
        }
        if (infoDict.count > 0){
            UILocalNotification *localNotif = [[UILocalNotification alloc] init];
            if (localNotif == nil)
                return;
            localNotif.fireDate = [NSDate dateWithTimeIntervalSinceNow:10];
            localNotif.timeZone = [NSTimeZone defaultTimeZone];
            localNotif.alertBody = [NSString stringWithFormat:NSLocalizedString(@"You
are near some places you may be interested in", nil)];
            localNotif.alertAction = NSLocalizedString(@"View Details", nil);
            localNotif.soundName = UILocalNotificationDefaultSoundName;
            NSString *typeOfActivity = @"notScheduled";
            [infoDict setObject:typeOfActivity forKey:@"typeOfActivity"];
            localNotif.userInfo = infoDict;
            [[UIApplication sharedApplication] scheduleLocalNotification:localNotif];
        }
    }
    [locationManager stopUpdatingLocation];
}
```

Η μέθοδος *didEnterBackground* καλείται όταν ο χρήστης θέσει την εφαρμογή στο παρασκήνιο. Τότε καλείται η μέθοδος που ανιχνεύει τις αλλαγές στην τοποθεσία του χρήστη, η *startMonitoringForSignificantChanges*. Αυτή η μέθοδος ενημερώνει την εφαρμογή όποτε συμβεί κάποια σημαντική αλλαγή στην προσωρινή θέση του χρήστη. Τις αλλαγές αυτές

μάλιστα αντιλαμβάνεται από την είσοδο στο πεδίο κάποιας κεραίας κινητής τηλεφωνίας αλλά και από τα διάφορα WiFi δίκτυα τα οποία ανιχνεύει η συσκευή και έχουν δεδομένη θέση. Όταν λοιπόν «πάρει» καινούρια θέση καλείται η μέθοδος *locationManager didUpdateLocations*. Εκεί, όπως φαίνεται στον κώδικά μας χρησιμοποιούμε το Framework CoreData, για να επιλέξουμε ποιες τοποθεσίες θα ελέγξουμε αν βρίσκονται κοντά στο χρήστη. Ζητάμε λοιπόν όλες τις μη προγραμματισμένες δραστηριότητες που δεν έχουν πραγματοποιηθεί, και τις τοποθεσίες που μπορούν να τις ικανοποιήσουν. Από αυτές τις τοποθεσίες, για εκείνες που δεν έχουν ελεγχθεί την τελευταία μισή ώρα και βρίσκονται σε μικρότερη απόσταση από 500 μέτρα από την τωρινή θέση του χρήστη δημιουργείται ειδοποίηση.

Λήψη Facebook τοποθεσιών και προσαρμογή στο χάρτη

Μέθοδοι	presentFacebookPlaces, placePickerViewControllerSelectionDidChange
Κλάση	addLocation

```
- (IBAction)presentFacebookPlaces:(id)sender
{
    if (!self.placePickerController) {
        self.placePickerController = [[FBPlacePickerViewController alloc]
initWithNibName:nil bundle:nil];
        self.placePickerController.delegate = self;
        self.placePickerController.title = @"Select a place";
    }
    self.placePickerController.locationCoordinate = self.manager.location.coordinate;
    self.placePickerController.radiusInMeters = 1000;
    self.placePickerController.resultsLimit = 50;
    if (![[[self.anActivity valueForKey:@"ActivityType"] valueForKey:@"title"]
isEqual:nil]) {
        self.placePickerController.searchText = [[self.anActivity
valueForKey:@"ActivityType"] valueForKey:@"title"];
    }
    [self.placePickerController loadData];
    [self.navigationController pushViewController:self.placePickerController
animated:true];
}
```

```
- (void)placePickerViewControllerSelectionDidChange:(FBPlacePickerViewController
*)placePicker
{
    self.selectedPlace = placePicker.selection;
    if (self.selectedPlace.count > 0) {
        CLLocation *facebookLocation = [[CLLocation alloc]
```

```
initWithLatitude:[[selectedPlace.location latitude] doubleValue]
longitude:[[selectedPlace.location longitude] doubleValue]];
        MapLocation *facebookAnnotation = [[MapLocation alloc] init];
        facebookAnnotation.coordinate = facebookLocation.coordinate;
        facebookAnnotation.theTitle = selectedPlace.name;
        facebookAnnotation.theSubtitle = selectedPlace.category;
        [self.mapView addAnnotation:facebookAnnotation];
        [self.navigationController popViewControllerAnimated:true];
    }
}
```

Όταν ο χρήστης πατήσει το κουμπί του Facebook στο χάρτη καλείται η μέθοδος *presentFacebookPlaces* η οποία με δεδομένη την προσωρινή θέση του χρήστη και για ακτίνα 1000 μέτρα γύρω από αυτή ζητά τις τοποθεσίες του Facebook που σχετίζονται με κάποιο τρόπο με τον τύπο της δραστηριότητας για την οποία ψάχνουμε τοποθεσία. Αφού επιλεγεί κάποια τοποθεσία καλείται η *placePickerViewControllerSelectionDidChange* η οποία προσθέτει μία πινέζα για αυτή την τοποθεσία.

Εμφάνιση μαρκαρισμένων σημείων στο χάρτη με ένα απλό κούνημα της συσκευής

Μέθοδοι	motionEnded withEvent
Κλάση	addLocation

```
- (void)motionEnded:(UIEventSubtype)motion withEvent:(UIEvent *)event
{
    if (motion == UIEventSubtypeMotionShake) {
        if ([self.locationArray count] != 0) {
            if (locationAnnotation != nil) {
                [self.mapView removeAnnotation:self.locationAnnotation];
            } else if ([self.notScheduledLocationArray count] != 0) {
                [self.mapView removeAnnotations:self.notScheduledLocationArray];
            }
            [self handleAnnotations:locationArray];
        } else {
            [self.mapView setUserTrackingMode:MKUserTrackingModeFollow];
        }
    }
}
```

Η μέθοδος *motionEnded:withEvent:* καλείται με ένα απλό κούνημα της συσκευής και όπως φαίνεται και παραπάνω αφαιρεί τις πινέζες από το χάρτη (*removeAnnotations*). Στη συνέχεια για ένα επιλεγμένο σύνολο τοποθεσιών προσαρμόζει το χάρτη ώστε να φαίνονται μαρκαρισμένες αυτές οι τοποθεσίες και η περιοχή που απεικονίζεται στο χάρτη να τις περιλαμβάνει *([self handleAnnotations:locationArray];)*.

Λήψη νέων τοποθεσιών από το server

Μέθοδος	getNewLocationsFromServer
Κλάση	ScheduledActivityListController

```
- (void)getNewLocationsFromServer
{
    if ([[NSUserDefaults standardUserDefaults] valueForKey:@"locationID"] == nil) {
        i = 0;
        [[NSUserDefaults standardUserDefaults] setObject:[NSNumber numberWithInt:i]
forKey:@"locationID"];
    } else {
        i = [[[NSUserDefaults standardUserDefaults] valueForKey:@"locationID"]
intValue];
    }
    NSString *str1 =
@"http://147.102.9.44:8080/dipl/locationActivityType?locationID=";
    NSString *str2 = [[NSNumber numberWithInt:i] stringValue];
    str1 = [str1 stringByAppendingString:str2];
    str1 =[str1 stringByAppendingString:@"&&users=false"];
    NSURL *url = [NSURL URLWithString:str1];
    NSURLRequest *urlRequest = [NSURLRequest requestWithURL:url];
    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    [NSURLConnection sendAsynchronousRequest:urlRequest queue:queue
completionHandler:^(NSURLResponse *response, NSData *data, NSError *error) {
        if ((data.length > 0) && (error == nil)) {
            NSArray *locationArray = [NSJSONSerialization JSONObjectWithData:data
options:0 error:nil];
            NSNumber *b = [NSNumber numberWithInt:i];
            AppDelegate *appDelegate = (AppDelegate *)[[UIApplication
sharedApplication] delegate];
            NSManagedObjectContext *managedObjectContext = [appDelegate
managedObjectContext];
            for (NSDictionary *diction in locationArray) {
                NSFetchRequest *fetchRequest1 = [[NSFetchRequest alloc] init];
                NSEntityDescription *entity1 = [NSEntityDescription
entityForName:@"Location"
```

```
inManagedObjectContext:managedObjectContext];
                [fetchRequest1 setEntity:entity1];
                NSString *title1 = [diction valueForKey:@"title"];
                NSPredicate *predicate1 = [NSPredicate predicateWithFormat:@"title ==
%@", title1];
                [fetchRequest1 setPredicate:predicate1];
                NSError *error;
                NSArray *fetchedObjects1 = [managedObjectContext
executeFetchRequest:fetchRequest1 error:&error];
                if (fetchedObjects1.count == 0) {
                    Location *loc = [NSEntityDescription
insertNewObjectForEntityForName:@"Location"
inManagedObjectContext:managedObjectContext];
                    [loc setTitle:[diction valueForKey:@"title"]];
                    [loc setLatitude:[diction valueForKey:@"latitude"]];
                    [loc setLongitude:[diction valueForKey:@"longitude"]];
                    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
                    NSEntityDescription *entity = [NSEntityDescription
entityForName:@"ActivityType"

inManagedObjectContext:managedObjectContext];
                    [fetchRequest setEntity:entity];


                    NSString *title = [diction valueForKey:@"activityTypeTitle"];
                    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"title
== %@", title];
                    [fetchRequest setPredicate:predicate];
                    NSArray *fetchedObjects = [managedObjectContext
executeFetchRequest:fetchRequest error:&error];
                    if (fetchedObjects.count == 0) {
                        ActivityType *activityType = [NSEntityDescription
insertNewObjectForEntityForName:@"ActivityType"
inManagedObjectContext:managedObjectContext];
                        [activityType setTitle:[diction
valueForKey:@"activityTypeTitle"]];
                        [activityType setAlarmTime:[diction
valueForKey:@"alarmTime"]];
                        [loc setValue:activityType forKey:@"ActivityType"];
                    } else {
                        [loc setValue:fetchedObjects.lastObject
forKey:@"ActivityType"];
                    }
                }
```

```
                if ([[diction valueForKey:@"id"] intValue] > b.intValue){
                    b = [diction valueForKey:@"id"];
                }
            }
            if (![managedObjectContext save:&error]){
            }
            [[NSUserDefaults standardUserDefaults] setInteger:b.intValue
forKey:@"locationID"];
            [[NSUserDefaults standardUserDefaults] synchronize];
        }
    }];

}
```

Μέθοδος	doGet
Servlet	LocationActivityTypeServlet

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException
        {
          try
          {
            InitialContext ctx2 = new InitialContext();
            UserTransaction utx = (UserTransaction) ctx2.lookup("java:module/UserTransaction");
            utx.begin();
            String title = null;
            title = request.getParameter("activityType");
            Long locationID = null;
            String id = request.getParameter("locationID");
            if (id != null)
            {
                locationID = Long.parseLong(id);
            }
            String users = null;
            users = request.getParameter("users");
            JSONArray j = new JSONArray();
            if (title != null)
            {
                ActivityTypeDAO activityTypeDAO = new ActivityTypeDAO();
```

```
                List<ActivityTypeDTO> activityTypes = activityTypeDAO.findByTitle(title);
                for (ActivityTypeDTO activityType : activityTypes)
                {
                LocationActivityTypeDAO locatioActivityTypeDAO = new LocationActivityTypeDAO();
                List<LocationActivityTypeDTO> locations = locatioActivityTypeDAO.findByActivityType(activityType.getId(), null);
                for (LocationActivityTypeDTO locationActivityType : locations)
                {
                    if (locationActivityType.getLocation().getUsers().equals("true"))
                    {
                        JSONObject jo = new JSONObject(locationActivityType.getLocation());
                        jo.put("activityTypeTitle", locationActivityType.getActivityType().getTitle());
                        jo.put("alarmTime", locationActivityType.getActivityType().getAlarmTime());
                        jo.put("locationID", locationActivityType.getLocation().getId());
                        j.put(jo);
                    }
                }
            }
        }else
        {
                LocationDAO locationDAO = new LocationDAO();
                List<LocationDTO> locations = locationDAO.findByParameters(locationID, users);
                for (LocationDTO location : locations)
                {
                    LocationActivityTypeDAO locatioActivityTypeDAO = new LocationActivityTypeDAO();
                    List<LocationActivityTypeDTO> locationActivityTypes = locatioActivityTypeDAO.findByLocation(location.getId());
                    for (LocationActivityTypeDTO locationActivityType : locationActivityTypes)
                    {
                        JSONObject jo = new JSONObject(locationActivityType.getLocation());
                        jo.put("activityTypeTitle", locationActivityType.getActivityType().getTitle());
                        jo.put("alarmTime", locationActivityType.getActivityType().getAlarmTime());
                        jo.put("locationID", locationActivityType.getLocation().getId());
                        jo.put("activityTypeTitle", locationActivityType.getActivityType().getTitle());
                        j.put(jo);
                     }
                }
```

```
        }
    utx.commit();
        response.getOutputStream().write(j.toString().getBytes());
                }
                catch (Exception ex)
                {
                        ex.printStackTrace();
                }
        }
```

Με την παραπάνω διαδικασία, στόχος είναι να ληφθούν οι τοποθεσίες που είναι αποθηκευμένες στο server και δεν έχουν προστεθεί από τους χρήστες. Φυσικά αυτές που ήδη έχει δε θέλει να τις ξαναπάρει. Με τη μέθοδο *getNewLocationsFromServer* ο χρήστης στέλνει στο server ένα httpRequest με το οποίο ζητάει τις τοποθεσίες που έχουν προστεθεί μετά από την τελευταία που έχει λάβει, την οποία χαρακτηρίζεται από τη locationID που είναι αποθηκευμένη στην εφαρμογή. Ζητείται επίσης μέσω του url οι τοποθεσίες που θα ληφθούν να μην έχουν προστεθεί από τους χρήστες. Αφού αποσταλεί το request το οποίο είναι ασύγχρονο, στο server καλείται η μέθοδος doGet που φαίνεται αναλυτικά παραπάνω. Εκεί ο server από τις παραμέτρους του request και συγκεκριμένα επειδή η παράμετρος activityType δεν έχει κάποια τιμή εκτελεί το αντίστοιχο μέρος της μεθόδου και μέσω των LocationDAO και LocationActivityTypeDAO που δημιουργούνται λαμβάνονται τα ζητούμενα LocationDTO. Μέσω αυτών των LocationDTO με τον ίδιο τρόπο παίρνουμε τα LocationActivityTypeDTO για τις συγκεκριμένες τοποθεσίες. Τέλος, αποστέλλονται σε μορφή JSON οι τοποθεσίες με τους τύπους δραστηριότητας που ικανοποιούν. Αυτό το αντικείμενο JSON περιέχεται στην απάντηση για το httpRequest που πραγματοποίησε ο χρήστης. Επιστρέφοντας στη μέθοδο *getNewLocationsFromServer*, η λαμβανόμενη απάντηση (JSON μορφή) μετατρέπεται σε ένα πίνακα από NSDictionaries, μέσω του οποίου αποθηκεύονται οι νέες τοποθεσίες και συνδέονται με τους αντίστοιχους τύπους δραστηριοτήτων που μπορούν να ικανοποιήσουν.

5.2. Παράδειγμα χρήσης

Στο κεφάλαιο αυτό θα περιγραφεί ένα σενάριο χρήσης της εφαρμογής, μέσω του οποίου θα γίνει πιο εύκολη η κατανόηση των λειτουργιών και της χρησιμότητας της. Θα αναλυθεί ο τρόπος με τον οποίο μπορούν να αλληλεπιδράσουν οι διάφοροι χρήστες της εφαρμογής αλλά και πώς ο καθένας μπορεί να τη χρησιμοποιήσει στα πλαίσια των δικών του αναγκών. Σε

αυτό το σενάριο λοιπόν θα εμπλακούν τρεις χρήστες, από τους οποίους οι δύο θα συνδεθούν μέσω του server της εφαρμογής και ο τρίτος μέσω του κοινωνικού δικτύου Facebook. Ο ένας από τους δύο που θα συνδεθούν μέσω του server θα είναι ο χρήστης με τον οποίο θα ασχοληθούμε περισσότερο και θα καλύψει τις περισσότερες λειτουργίες της εφαρμογής. Ο ρόλος των άλλων δύο είναι να φανεί ο τρόπος που μπορούν να επηρεάσουν τον κύριο χρήστη στο συγκεκριμένο σενάριο.

Αρχικά, θα πρέπει όλοι χρήστες να συνδεθούν στην εφαρμογή, είτε μέσω του server, είτε μέσω του Facebook. Αφού γίνει αυτό ο κύριος χρήστης θα προσθέσει τέσσερις δραστηριότητες, δύο προγραμματισμένες και δύο μη προγραμματισμένες. Στην πρώτη προγραμματισμένη δραστηριότητα που θα προσθέσει θα επιλέξει τον τύπο της δραστηριότητας και όταν κληθεί να επιλέξει τοποθεσία στο χάρτη θα είναι «μαρκαρισμένες» οι τοποθεσίες στις οποίες μπορεί να την ικανοποιήσει. Αφού επιλέξει κάποια από τις προτεινόμενες τοποθεσίες θα αποθηκεύσει αυτή την προγραμματισμένη δραστηριότητα και θα δημιουργηθούν δύο ειδοποιήσεις. Οι ειδοποιήσεις αυτές θα εμφανιστούν τις κατάλληλες χρονικές στιγμές ανάλογα με την ημερομηνία και ώρα έναρξης της δραστηριότητας, αλλά και τον τύπο της.

Στη δεύτερη προγραμματισμένη δραστηριότητα όμως που θα προσθέσει ο κύριος χρήστης θα υπάρξουν κάποιες διαφορές. Συγκεκριμένα, ο χρήστης που έχει συνδεθεί μέσω Facebook προσθέτει δημόσια μία νέα τοποθεσία για ένα συγκεκριμένο τύπο δραστηριότητας. Την τοποθεσία όμως αυτή, την έχει επιλέξει από τις τοποθεσίες του Facebook, επιλογή που ο κύριος χρήστης δεν έχει, εφόσον δεν έχει συνδεθεί μέσω Facebook. Στη συνέχεια, ο κύριος χρήστης επιλέγει σαν τύπο της δεύτερης προγραμματισμένης δραστηριότητας τον τύπο της τοποθεσίας που πρόσθεσε ο Facebook χρήστης. Έτσι, στις «μαρκαρισμένες» τοποθεσίες-προτάσεις θα υπάρχει και η τοποθεσία που πρόσθεσε πριν από λίγο ο χρήστης που έχει συνδεθεί μέσω Facebook. Ο κύριος χρήστης θα επιλέξει τη συγκεκριμένη τοποθεσία και θα αποθηκεύσει την προγραμματισμένη δραστηριότητα. Λόγω του συνδυασμού όμως του τύπου δραστηριότητας και της ημερομηνίας και ώρας έναρξής της θα δημιουργηθεί μία και όχι δύο ειδοποιήσεις όπως πριν.

Μετά από την προσθήκη των παραπάνω προγραμματισμένων δραστηριοτήτων ο κύριος χρήστης θα προσθέσει τις δύο μη προγραμματισμένες δραστηριότητες. Για την πρώτη, αφού ορίσει τον τίτλο και το χρονικό όριο μέχρι το οποίο θα ήθελε να πραγματοποιηθεί αυτή η δραστηριότητα επιλέγει τον τύπο της από τον αντίστοιχο κατάλογο. Εν τω μεταξύ, ο άλλος χρήστης που έχει συνδεθεί μέσω του server έχει προσθέσει δύο τοποθεσίες και τον τύπο δραστηριότητας που μπορούν να εξυπηρετήσουν. Για τη μία έχει επιλέξει να τη δημοσιοποιήσει στο server και τους άλλους χρήστες ενώ για την άλλη, η οποία αφορά μόνο εκείνον έχει επιλέξει να αποθηκευθεί μόνο στη βάση δεδομένων του κινητού του. Ο τύπος

δραστηριότητας που έχει επιλέξει ο κύριος χρήστης για τη μη προγραμματισμένη δραστηριότητά του και ο τύπος της δημόσιας τοποθεσίας που πρόσθεσε πριν από λίγο ο άλλος χρήστης που έχει συνδεθεί από το server είναι ίδιοι. Έτσι όταν ο κύριος χρήστης κληθεί να επιλέξει τις τοποθεσίες στις οποίες θα μπορούσε να πραγματοποιήσει τη δραστηριότατα στις τοποθεσίες-προτάσεις θα υπάρχει και η τοποθεσία του άλλου χρήστη και ανάμεσα στις άλλες που θα επιλέξει θα είναι και αυτή. Αφού ο κύριος χρήστης αποθηκεύσει μία δραστηριότητα, για το χρονικό διάστημα μέχρι την ημερομηνία που έχει τεθεί σαν όριο, θα ειδοποιείται όποτε βρίσκεται σε κοντινή απόσταση από κάποια τοποθεσία που ικανοποιεί τη δραστηριότητα αυτή. Θα ειδοποιηθεί επίσης όταν περάσει το χρονικό όριο που ο ίδιος έχει θέσει.

Όσον αφορά τη δεύτερη μη προγραμματισμένη δραστηριότητα ο κύριος χρήστης συμπληρώνει τα πεδία του τίτλου, του χρονικού ορίου και του τύπου δραστηριότητας όπως πριν, όμως σαν τοποθεσία ικανοποίησης της δραστηριότητας αφήνει την προεπιλογή “All Places”. Αυτό σημαίνει ότι όποτε βρίσκεται σε κοντινή απόσταση από κάποια τοποθεσία που ικανοποιεί δραστηριότητες αυτού του τύπου που έχει επιλέξει θα ειδοποιείται. Θα ειδοποιηθεί επίσης όταν περάσει το χρονικό όριο που ο ίδιος έχει θέσει.

Σημαντικό κομμάτι του σεναρίου αλλά και της λειτουργικότητας της εφαρμογής είναι οι ειδοποιήσεις, ο τρόπος που εμφανίζονται σε κάθε περίπτωση και οι επιλογές που δίνουν στο χρήστη για να κάνει αλλαγές στις δραστηριότητες που αφορά η κάθε ειδοποίηση. Διαφορές υπάρχουν ανάλογα με την κατάσταση στην οποία βρίσκεται η εφαρμογή, αν είναι στο προσκήνιο ή στο παρασκήνιο, και ανάλογα με το είδος της δραστηριότητας την οποία αφορά, αν είναι προγραμματισμένη ή μη προγραμματισμένη.

5.2.1. Αρχικά δεδομένα

Η εφαρμογή όταν τίθεται σε λειτουργία για πρώτη φορά θα πρέπει να έχει κάποια δεδομένα. Δε θα ήταν πρακτικό για παράδειγμα όταν κληθεί ο χρήστης να επιλέξει τον τύπο μίας δραστηριότητας ή τοποθεσίας να μην υπάρχουν κάποιες βασικές, συνήθεις κατηγορίες σαν αρχικά δεδομένα.

Συγκεκριμένα, την πρώτη φορά που ο χρήστης θα βρεθεί στην κεντρική οθόνη της εφαρμογής, θα λάβει όλα τα δεδομένα του server τα οποία έχουν καταχωρηθεί σαν δημόσια. Αυτά τα δεδομένα είναι τοποθεσίες οι οποίες συνοδεύονται από τον τύπο δραστηριοτήτων που μπορεί να πραγματοποιηθούν εκεί. Φυσικά αποθηκεύονται και αυτοί οι τύποι δραστηριοτήτων, από τους οποίους θα μπορεί να επιλέξει αργότερα ο χρήστης αν χρειαστεί. Έτσι, ακόμα και την πρώτη φορά που χρησιμοποιεί κάποιος την εφαρμογή και έχει επιλέξει τον τύπο δραστηριότητας μπορεί να έχει προτάσεις από τα αρχικά δεδομένα.

Οι τοποθεσίες που αποθηκεύει κάποιος χρήστης και έχει επιλέξει να σταλούν στο server, δεν είναι μέσα στα αρχικά δεδομένα που θα έχει κάποιος άλλος χρήστης. Είναι ο διαχειριστής του server αυτός που θα αποφασίσει ποια δεδομένα από αυτά που οι χρήστες δημιουργούν θα δημοσιοποιούνται προς όλους τους χρήστες.

Τα αρχικά δεδομένα που θα έχει ο κάθε χρήστης στο συγκεκριμένο σενάριο φαίνονται στον παρακάτω πίνακα, όπου κάθε τοποθεσία αντιστοιχίζεται με τον τύπο δραστηριότητας που μπορεί να ικανοποιήσει.

Αρχικά δεδομένα

Τοποθεσία	**Τύπος τοποθεσίας**
Plaisio Voulis	Electronics
Public Syntagma	Electronics
Kotsovolos	Electronics
Ampariza	Bar
Bartesera	Bar
Booze Cooperativa	Bar
Cine Paris	Cinema
Attikon	Cinema
Dosirak	Restaurant
Shakespeare	Restaurant
Mac Syntagma	Junk Food
Everest Syntagma	Junk Food
Souvlaki Petros	Junk Food
Thanasis	Junk Food
Oionos	Coffee
Public Café	Coffee
Da Capo	Coffee
Gas Station Syggrou	Gas Station
Aegean Syntagma	Gas Station
Pharmacy Voulis	Pharmacy
Pharmacy Plaka	Pharmacy

Zara Syntagma	Shopping
Fokas Syntagma	Shopping
Acropolis Museum	Museum
Museum of Greek Folk Art	Museum

5.2.2. Εκτέλεση σεναρίου

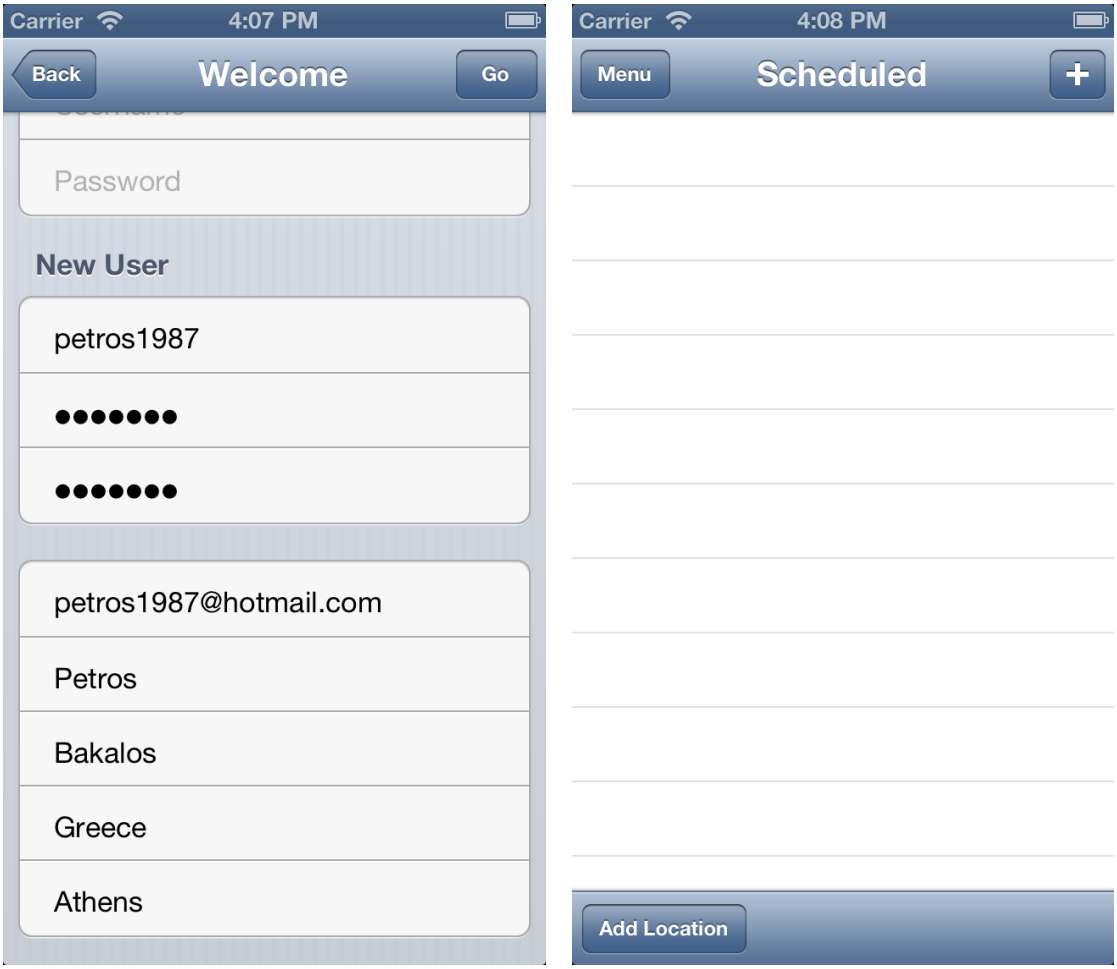
Το σενάριο του οποίου έγινε περιγραφή πιο πάνω αφορά τρεις χρήστες της εφαρμογής. Υποθέτουμε ότι τα ονόματά τους είναι Πέτρος, Χριστίνα και Σοφία. Εκτελώντας το σενάριο για τους τρεις αυτούς χρήστες θα γίνει μία προσπάθεια καλύτερης κατανόησης και περιγραφής των λειτουργιών και της χρησιμότητας της εφαρμογής.

Και οι τρεις χρήστες, αφού εγκαταστήσουν και θέσουν σε λειτουργία την εφαρμογή στο κινητό τους τηλέφωνο, οδηγούνται στην εναρκτήρια οθόνη της εφαρμογής. Εκεί ο κάθε χρήστης μπορεί να επιλέξει να συνδεθεί στην εφαρμογή είτε μέσω του λογαριασμού του στο κοινωνικό δίκτυο Facebook, είτε μέσω του server. Σε αυτό το σενάριο υποθέτουμε ότι ο Πέτρος και η Χριστίνα συνδέονται μέσω του server και η Σοφία μέσω του Facebook πατώντας τα κουμπιά “sign up” και “login with facebook” αντίστοιχα, στην εναρκτήρια οθόνη (**Εικόνα 31.α**). Ο Πέτρος και η Χριστίνα οδηγούνται στην οθόνη εγγραφής και σύνδεσης της εφαρμογής (**Εικόνα 31.β**), όπου μπορούν είτε να εγγραφούν σαν νέοι χρήστες συμπληρώνοντας τα πεδία του “section” New User, είτε να συνδεθούν συμπληρώνοντας τα πεδία του “section” Existing User.

α β

Εικόνα 31 Η εναρκτήρια οθόνη και η οθόνη εγγραφής και σύνδεσης στην εφαρμογή.

Ο Πέτρος, που δεν έχει χρησιμοποιήσει ξανά την εφαρμογή, συμπληρώνει τη φόρμα εγγραφής (**Εικόνα 32.α**) και πατάει το κουμπί "Go". Η Χριστίνα, που είχε ήδη εγγραφεί, συμπληρώνει τα πεδία username και password και πατάει το κουμπί "Go". Ο Πέτρος εφόσον τα στοιχεία του έχουν γίνει δεκτά από το server και η Χριστίνα εφόσον τα στοιχεία της είναι έγκυρα, οδηγούνται στην κύρια οθόνη της εφαρμογής, τη λίστα με τις προγραμματισμένες δραστηριότητες (**Εικόνα 32.β**).



α β

Εικόνα 32 Η φόρμα εγγραφής συμπληρωμένη και η κεντρική οθόνη της εφαρμογής.

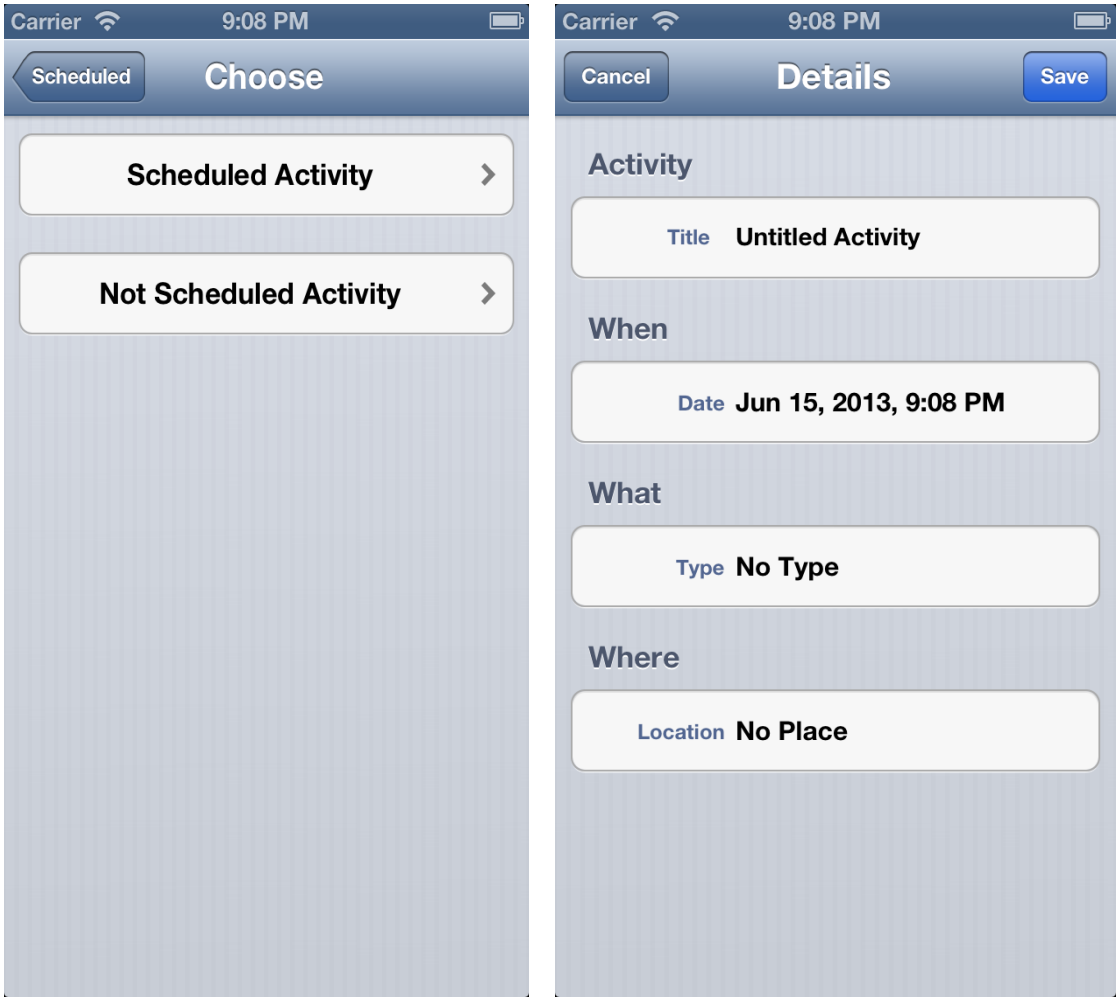
Η Σοφία, στην εναρκτήρια οθόνη είχε επιλέξει να συνδεθεί μέσω Facebook. Αν ήταν ήδη συνδεδεμένη στο Facebook, τότε από την αρχική οθόνη οδηγείται στην κεντρική, δηλαδή στη λίστα με τις προγραμματισμένες δραστηριότητες. Αν δεν ήταν ήδη συνδεδεμένη μέσω του browser της ζητήθηκε να κάνει login στη σελίδα του Facebook και στη συνέχεια οδηγήθηκε στην κεντρική οθόνη.

Όλοι οι χρήστες, όταν βρίσκονται στην κεντρική οθόνη ενημερώνονται από το server για τις τελευταίες τοποθεσίες κοινής χρήσης που προστέθηκαν στη βάση δεδομένων του server και δεν έχουν αποθηκευτεί στο κινητό. Μπορούν μάλιστα πατώντας το κουμπί Menu να οδηγηθούν στο μενού της εφαρμογής (**Εικόνα 33.α**) και από την επιλογή Locations να δουν τον κατάλογο με τις τοποθεσίες που είναι αποθηκευμένες στο κινητό, συνοδευόμενες από τον τύπο δραστηριοτήτων που μπορούν να ικανοποιήσουν (**Εικόνα 33.β**). Για τους νέους χρήστες αυτές οι τοποθεσίες αποθηκεύτηκαν όταν έφτασαν στην κεντρική οθόνη.

α β

Εικόνα 33 Το μενού της εφαρμογής και η λίστα των αποθηκευμένων τοποθεσιών.

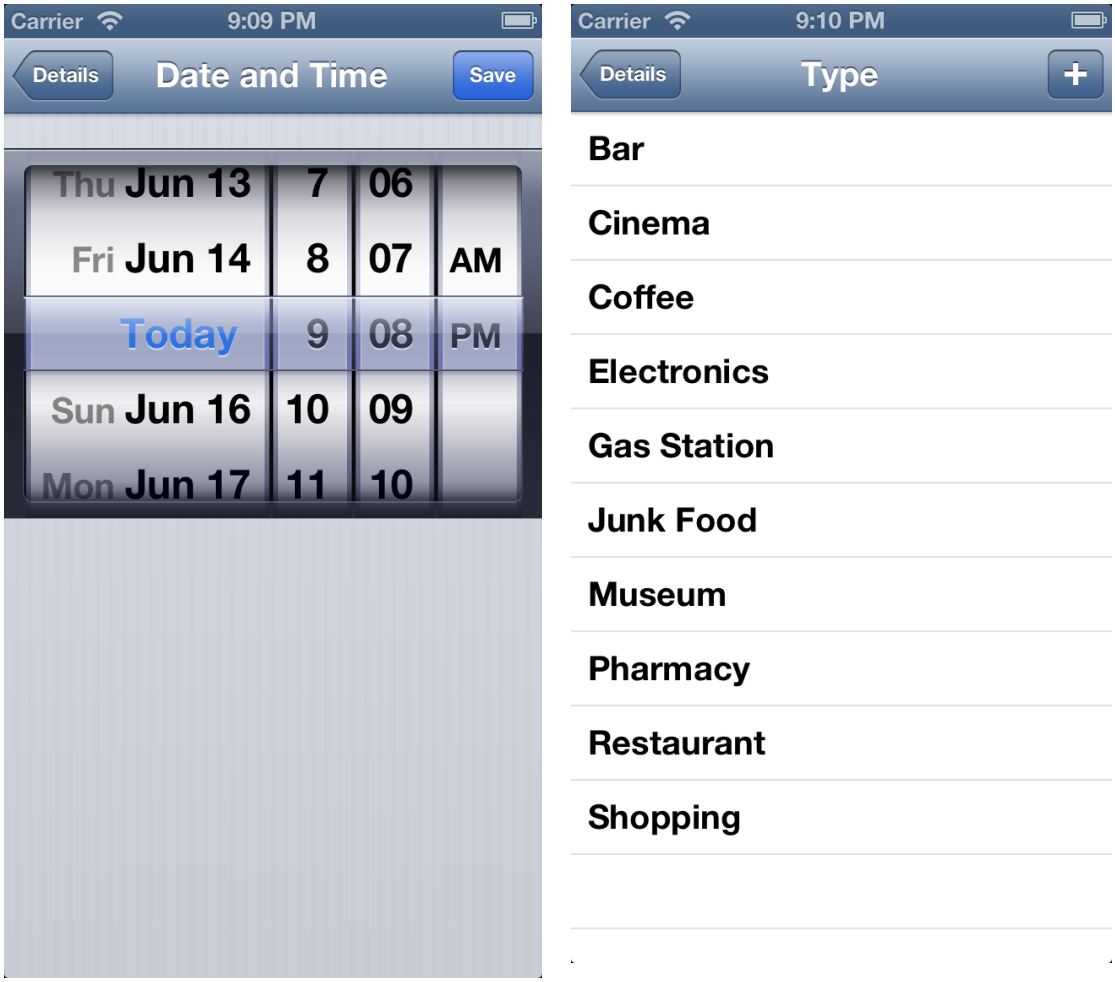
Στο σενάριό μας λοιπόν, ο κύριος χρήστης που είναι ο Πέτρος επιστρέφει στην κεντρική οθόνη και θέλει να αποθηκεύσει μία νέα προγραμματισμένη δραστηριότητα. Πατάει το κουμπί + και από την επόμενη οθόνη (**Εικόνα 34.α**) επιλέγει Scheduled Activity. Στη συνέχεια καλείται να συμπληρώσει κάποια στοιχεία για την προγραμματισμένη αυτή δραστηριότητα (**Εικόνα 34.β**).



α β

Εικόνα 34 Οθόνη επιλογής είδους δραστηριότητας και οθόνη λεπτομερειών προγραμματισμένης δραστηριότητας.

Σαν τίτλο ο Πέτρος θέτει “Coffee with Manolis”. Για την ημερομηνία έναρξης της δραστηριότητας οδηγείται στην κατάλληλη οθόνη (**Εικόνα 35.α**). Στη συνέχεια επιλέγει τον τύπο της δραστηριότητας “Coffee”. Από την οθόνη με τον κατάλογο των τύπων δραστηριοτήτων (**Εικόνα 35.β**) μπορεί να επιλέξει όποια πιστεύει ότι ταιριάζει καλύτερα. Βέβαια έχει και τη δυνατότητα να προσθέσει καινούριο δικό του τύπο δραστηριοτήτων πατώντας το κουμπί +.



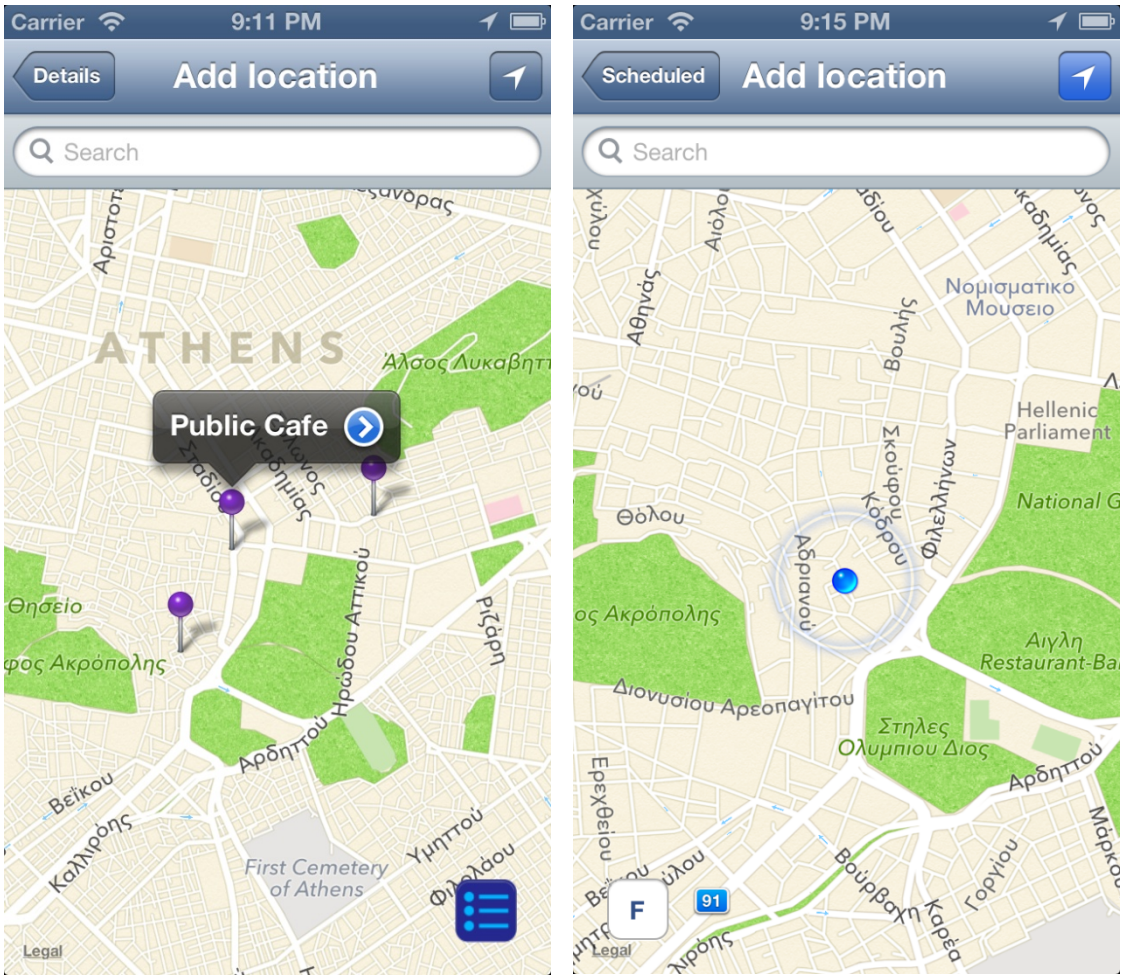
α β

Εικόνα 35 Ημερολόγιο και κατάλογος τύπων δραστηριοτήτων.

Μόνο το πεδίο της τοποθεσίας δεν έχει συμπληρώσει ο Πέτρος. Πατώντας και το τελευταίο πεδίο οδηγείται στο χάρτη (**Εικόνα 36.α**) όπου υπάρχουν μαρκαρισμένα τρία σημεία. Σε αυτές τις τρεις τοποθεσίες έχει δηλωθεί με κάποιο τρόπο ότι μπορούν να εξυπηρετήσουν δραστηριότητες τύπου "Coffee". Ο τρόπος αυτός μπορεί να είναι είτε να ανήκει στις τοποθεσίες κοινής χρήσης του server, που αυτό συμβαίνει στην περίπτωσή μας, είτε να τις έχει αποθηκεύσει κάποιος χρήστης στο server. Από τις τρεις τοποθεσίες ο Πέτρος επιλέγει το "Public Cafe".

Επιστρέφοντας στην οθόνη με τις λεπτομέρειες της προγραμματισμένης δραστηριότητας ο χρήστης πατάει το κουμπί "Save". Η δραστηριότητα αποθηκεύεται και δημιουργούνται δύο ειδοποιήσεις. Η μία θα ενημερώσει το χρήστη 45 λεπτά πριν τη στιγμή που έχει προγραμματίσει να ξεκινήσει τη δραστηριότητα. Αυτά τα 45 λεπτά είναι ο χρόνος που έχει ορισθεί για να ειδοποιείται ο χρήστης πριν από προγραμματισμένες δραστηριότητες τύπου "Coffee". Για κάθε τύπο δραστηριότητας υπάρχει ο αντίστοιχος χρόνος που θα ειδοποιείται ο

χρήστης. Η δεύτερη ειδοποίηση θα εμφανιστεί τη στιγμή που έχει προγραμματισθεί να ξεκινήσει η δραστηριότητα.

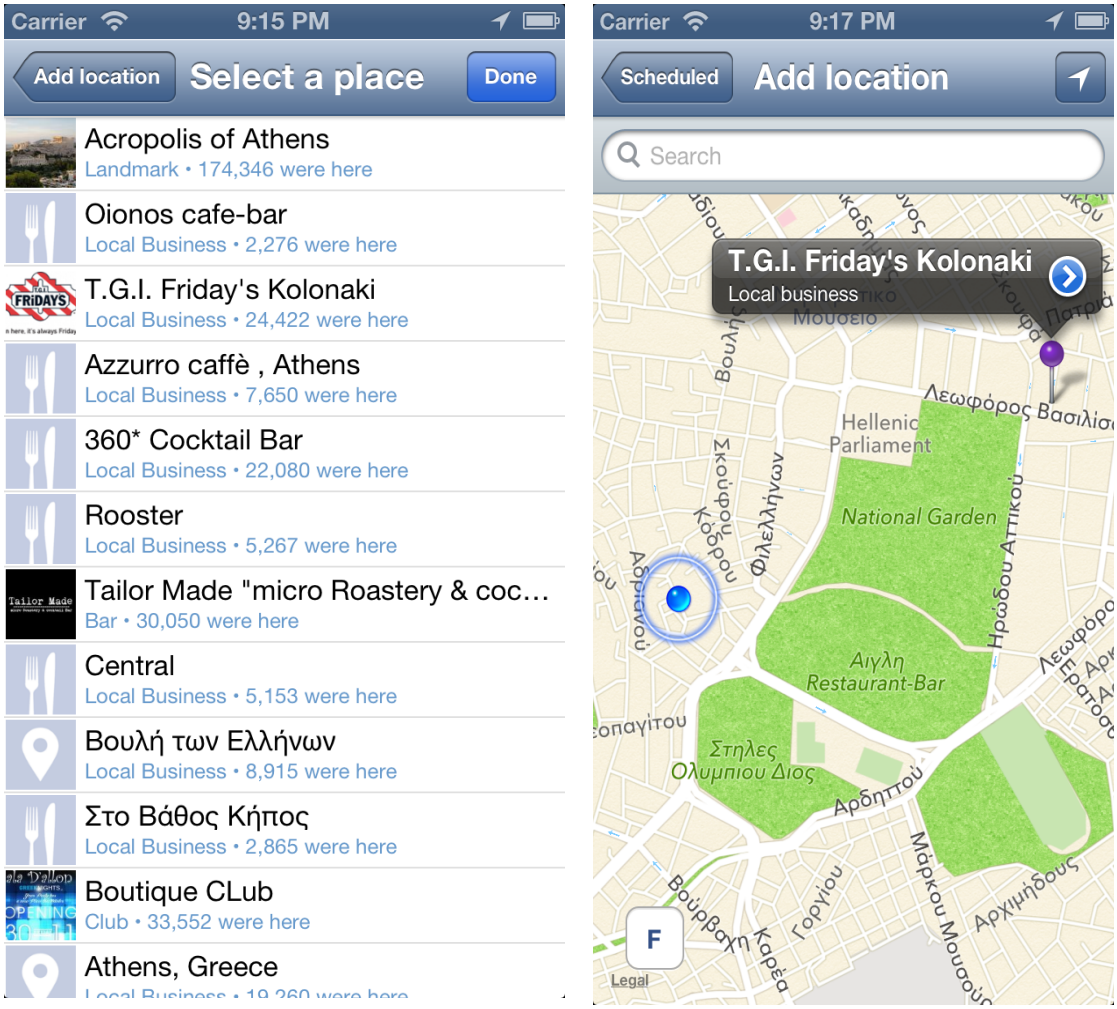


α β

Εικόνα 36 Προσθήκη τοποθεσίας για προγραμματισμένη δραστηριότητα και χάρτης με την τωρινή θέση του χρήστη.

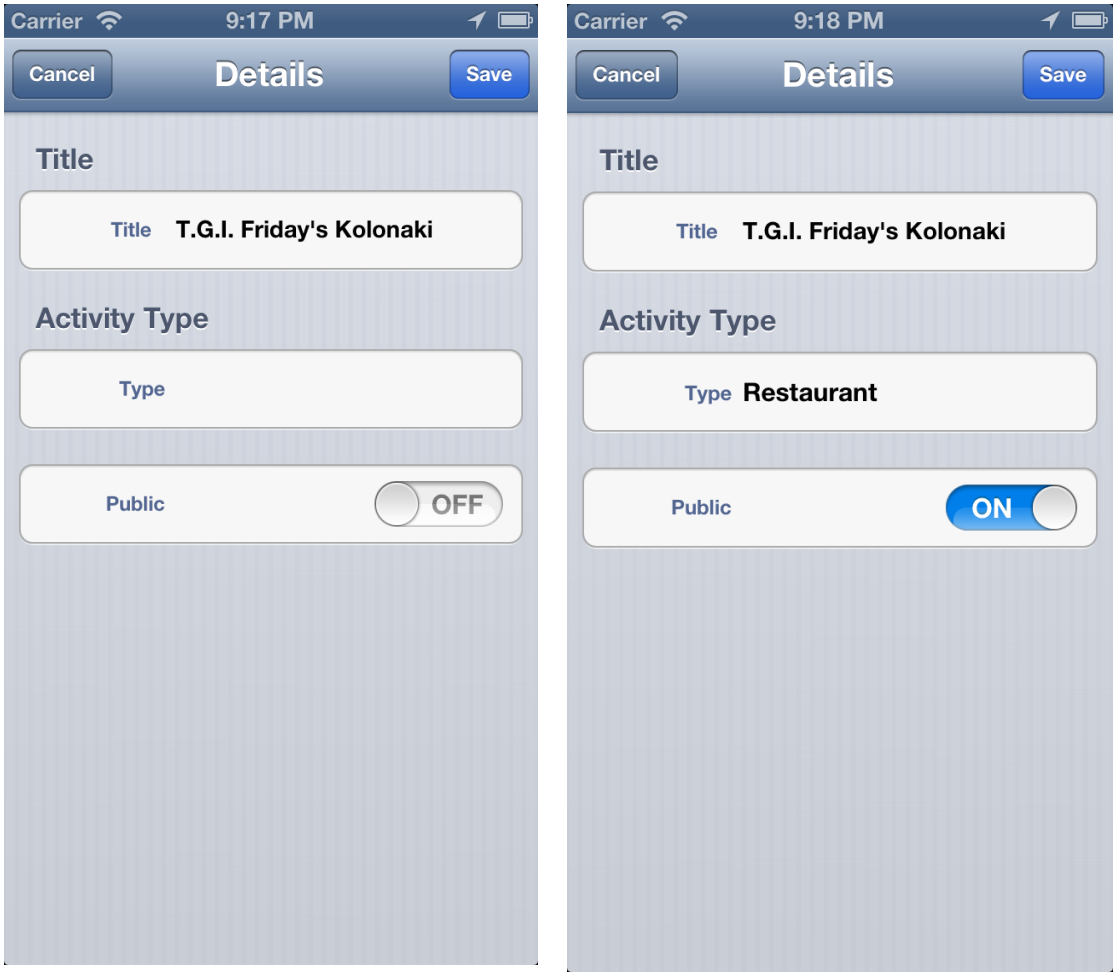
Στη συνέχεια του σεναρίου ο χρήστης που έχει συνδεθεί μέσω Facebook, δηλαδή η Σοφία θα προσθέσει μία νέα τοποθεσία. Από την κεντρική οθόνη θα πατήσει το κουμπί "Add Location" και θα οδηγηθεί στο χάρτη όπου φαίνεται η θέση της εκείνη τη στιγμή (**Εικόνα 36.β**). Πατώντας όμως το κουμπί "F" κάτω αριστερά εμφανίζεται ένας κατάλογος με τις τοποθεσίες του Facebook που βρίσκονται κοντά στην τωρινή θέση του χρήστη (**Εικόνα 37.α**). Η Σοφία επιλέγει κάποια από αυτές, στην περίπτωσή μας το εστιατόριο "Friday's" και αυτό εμφανίζεται μαρκαρισμένο στο χάρτη (**Εικόνα 37.β**). Η Σοφία όμως θέλει να αποθηκεύσει αυτή την τοποθεσία για μελλοντική χρήση και οδηγείται στην οθόνη αποθήκευσης νέας τοποθεσίας (**Εικόνα 38.α**). Εκεί σαν τύπο, από τον αντίστοιχο κατάλογο, επιλέγει "Restaurant".

Αυτή τη στιγμή αν η Σοφία πατήσει το κουμπί “Save”, θα μπορεί η ίδια να χρησιμοποιήσει αυτή την τοποθεσία για τις δραστηριότητές της ανά πάσα στιγμή. Δεν θα την έχει στείλει όμως στο server. Η Σοφία θεωρεί ότι αυτή η τοποθεσία θα μπορούσε να φανεί χρήσιμη και σε άλλους χρήστες της εφαρμογής, οπότε έχοντας το κουμπί “Public” ενεργοποιημένο, πατάει “Save” (**Εικόνα 38.β**). Η τοποθεσία και ο τύπος δραστηριοτήτων που μπορεί να ικανοποιήσει το εστιατόριο Friday’s έχουν αποθηκευθεί στο server στην κατηγορία των δεδομένων που προέρχονται από χρήστες.



α β

Εικόνα 37 Τοποθεσίες Facebook και τοποθεσία Facebook μαρκαρισμένη.



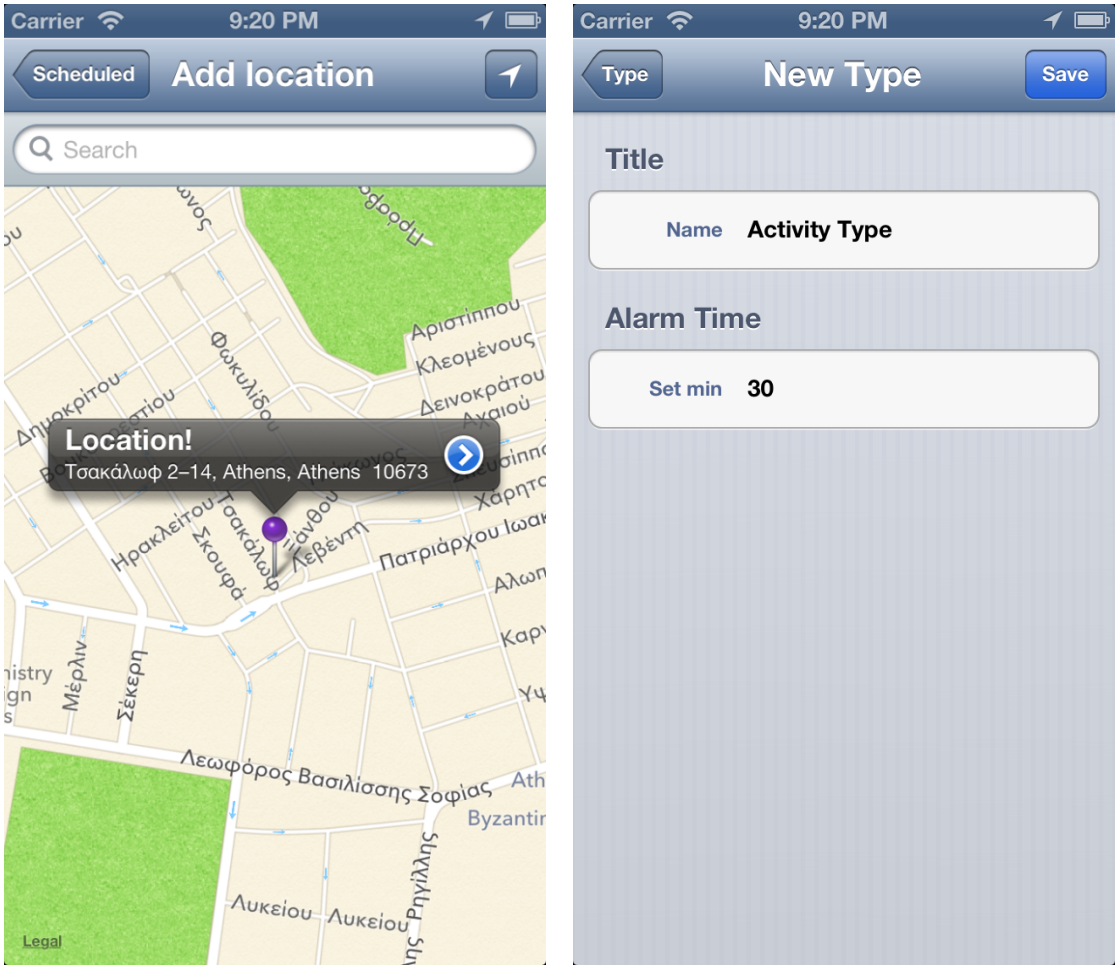
α β

Εικόνα 38 Λεπτομέρειες τοποθεσίας.

Ο κύριος χρήστης μας τώρα, ο Πέτρος, θα αποθηκεύσει τη δεύτερη προγραμματισμένη δραστηριότητά του. Αφού φτάσει στην οθόνη επεξεργασίας των λεπτομερειών μιας τέτοιας δραστηριότητας συμπληρώνει τα πεδία που αφορούν τον τίτλο, την ημερομηνία και ώρα έναρξης της δραστηριότητας και τον τύπο της. Σαν τίτλο θέτει "Dinner with Coworkers", σαν χρονική στιγμή που θα ξεκινήσει η δραστηριότητα 45 λεπτά μετά από την τωρινή ώρα και σαν τύπο επιλέγει "Restaurant" από τον αντίστοιχο κατάλογο. Αυτή τη στιγμή στη βάση δεδομένων του κινητού δεν υπάρχει αποθηκευμένη η τοποθεσία του εστιατορίου Friday's που πρόσθεσε πριν από λίγο η Σοφία. Όταν όμως ο Πέτρος πατήσει να επιλέξει την τοποθεσία που θα πραγματοποιηθεί η δραστηριότητα στις μαρκαρισμένες περιοχές του χάρτη θα υπάρχει και το εστιατόριο Friday's το οποίο και θα επιλέξει ο Πέτρος για το δείπνο με τους συναδέλφους. Εδώ, σημαντικό είναι να παρατηρήσουμε ότι ενώ η θέση του συγκεκριμένου εστιατορίου δεν ανήκει στις τοποθεσίες κοινής χρήσης του server και ενώ ο Πέτρος δεν έχει συνδεθεί μέσω του κοινωνικού δικτύου Facebook αυτή τη στιγμή έχει αποθηκεύσει μία δραστηριότητα σε αυτό το εστιατόριο. Αυτός είναι ο βασικός τρόπος που μπορούν να

αλληλεπιδρούν μεταξύ τους οι διάφοροι χρήστες. Τέλος, ο κύριος χρήστης αποθηκεύει την προγραμματισμένη δραστηριότητα και δημιουργείται μία ειδοποίηση η οποία θα εμφανιστεί τη στιγμή που θα ξεκινήσει η δραστηριότητα. Επειδή ο χρόνος που ειδοποιούνται οι χρήστες πριν από δραστηριότητες τύπου "Restaurant" είναι μία ώρα, η δεύτερη ειδοποίηση δε θα δημιουργηθεί, αφού αυτός ο χρόνος έχει περάσει.

Το επόμενο που συμβαίνει σε αυτό το σενάριο χρήσης είναι η αποθήκευση μίας νέας τοποθεσίας από τη Χριστίνα. Ακολουθώντας τα ίδια βήματα με τη Σοφία προηγουμένως η Χριστίνα φτάνει στο χάρτη όπου φαίνεται η τωρινή της θέση. Πατώντας παρατεταμένα πάνω στην οθόνη «μαρκάρεται» αυτή η τοποθεσία (**Εικόνα 39.α**) και προχωρώντας στην αποθήκευση των λεπτομερειών της οδηγείται στην αντίστοιχη οθόνη. Εκεί, σαν τίτλο θέτει "George's House". Για τον τύπο όμως της τοποθεσίας δεν υπάρχει κάποια επιλογή στον κατάλογο που να ικανοποιεί τη Χριστίνα. Έτσι, πατάει το κουμπί προσθήκης νέου τύπου και οδηγείται στην οθόνη λεπτομερειών νέου τύπου τοποθεσίας (**Εικόνα 39.β**). Σαν τίτλο θέτει "Friends" και σαν χρόνο ειδοποίησης πριν από προγραμματισμένη δραστηριότητα 30 λεπτά. Αφού αποθηκεύσει τον τύπο αυτό αποθηκεύει και την τοποθεσία χωρίς όμως να έχει επιλέξει την αποστολή αυτών των δεδομένων στο server, γιατί η τοποθεσία αυτή αφορά μόνο την ίδια τη Χριστίνα. Δε θα εμφανιστεί δηλαδή ποτέ στους άλλους χρήστες η τοποθεσία "George's House".

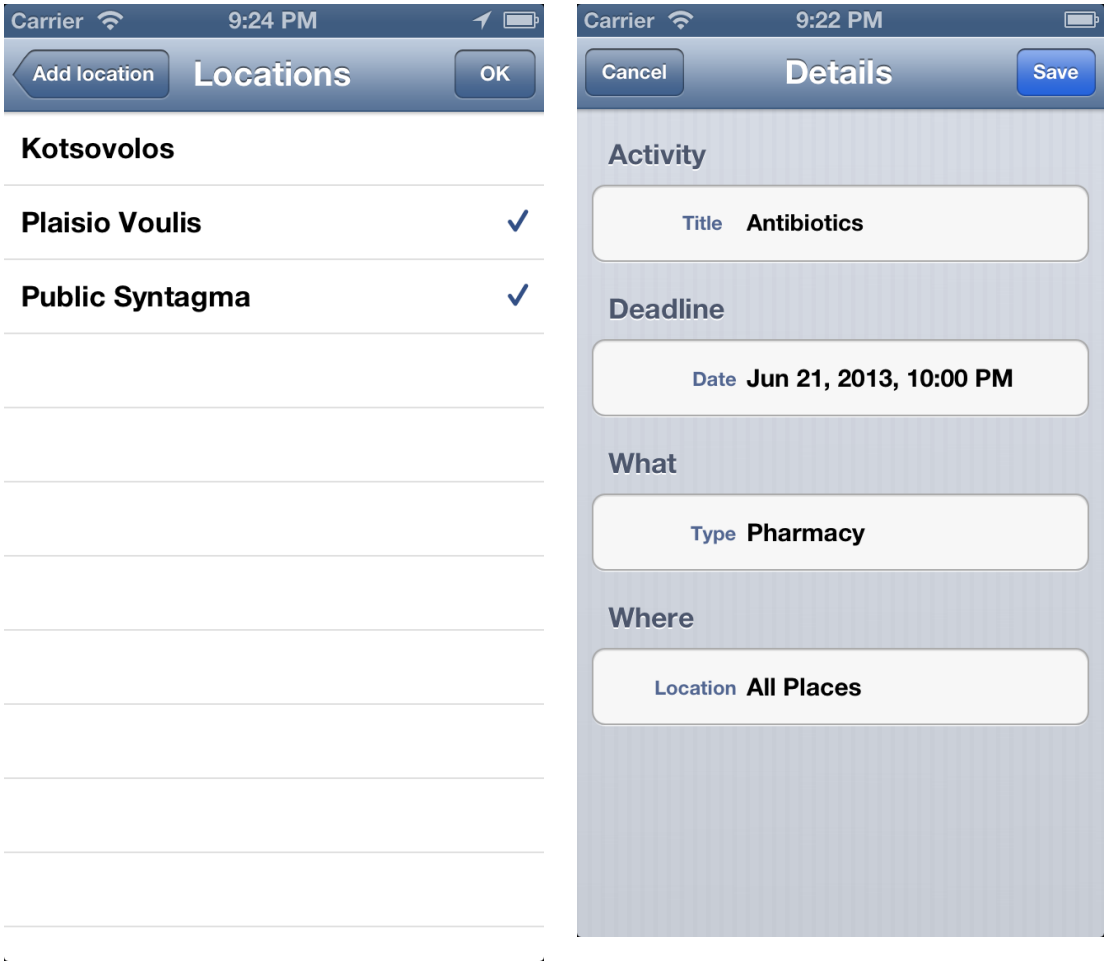


α β

Εικόνα 39 Χάρτης με μαρκαρισμένη τοποθεσία και προσθήκη νέου τύπου δραστηριοτήτων.

Επιστρέφοντας στον κύριο χρήστη, τον Πέτρο, έχουμε την προσθήκη δύο μη προγραμματισμένων δραστηριοτήτων. Για την πρώτη, ο Πέτρος αφού πατήσει το κουμπί προσθήκης νέας δραστηριότητας στην κεντρική οθόνη, επιλέγει "Not Scheduled Activity" από την οθόνη επιλογής είδους δραστηριότητας και οδηγείται στις λεπτομέρειες μη προγραμματισμένων δραστηριοτήτων. Ο Πέτρος θέλει να αγοράσει μέσα στην επόμενη εβδομάδα ένα πληκτρολόγιο. Έτσι σαν τίτλο θέτει "Keyboard" και σαν χρονικό όριο μία εβδομάδα από τώρα. Ο τύπος δραστηριότητας που επιλέγει, με τον ίδιο τρόπο όπως και στις προηγούμενες δραστηριότητες, είναι "Elerctronics". Πατώντας να επιλέξει το κατάστημα από το οποίο θα το αγοράσει, εμφανίζονται στο χάρτη μαρκαρισμένα διάφορα καταστήματα ηλεκτρονικών ειδών. Ο Πέτρος πατάει το κουμπί που τον οδηγεί στη λίστα με αυτά τα καταστήματα και επιλέγει δύο από αυτά (**Εικόνα 40.α**), το "Public Syntagma" και το "Plaisio Syntagma". Πατώντας το κουμπί "Save" από την οθόνη των λεπτομερειών μη προγραμματισμένων δραστηριοτήτων αποθηκεύεται αυτή η δραστηριότητα και δημιουργείται μία ειδοποίηση η οποία θα ενημερώσει το χρήστη όταν παρέλθει η χρονική στιγμή που έχει

θέσει σαν όριο. Το σημαντικό είναι όμως ότι όποτε ο Πέτρος βρεθεί σε κοντινή απόσταση από κάποιο από τα δύο καταστήματα που επέλεξε, και εντός του χρονικού ορίου που έχει θέσει θα ενημερωθεί με μία ειδοποίηση.



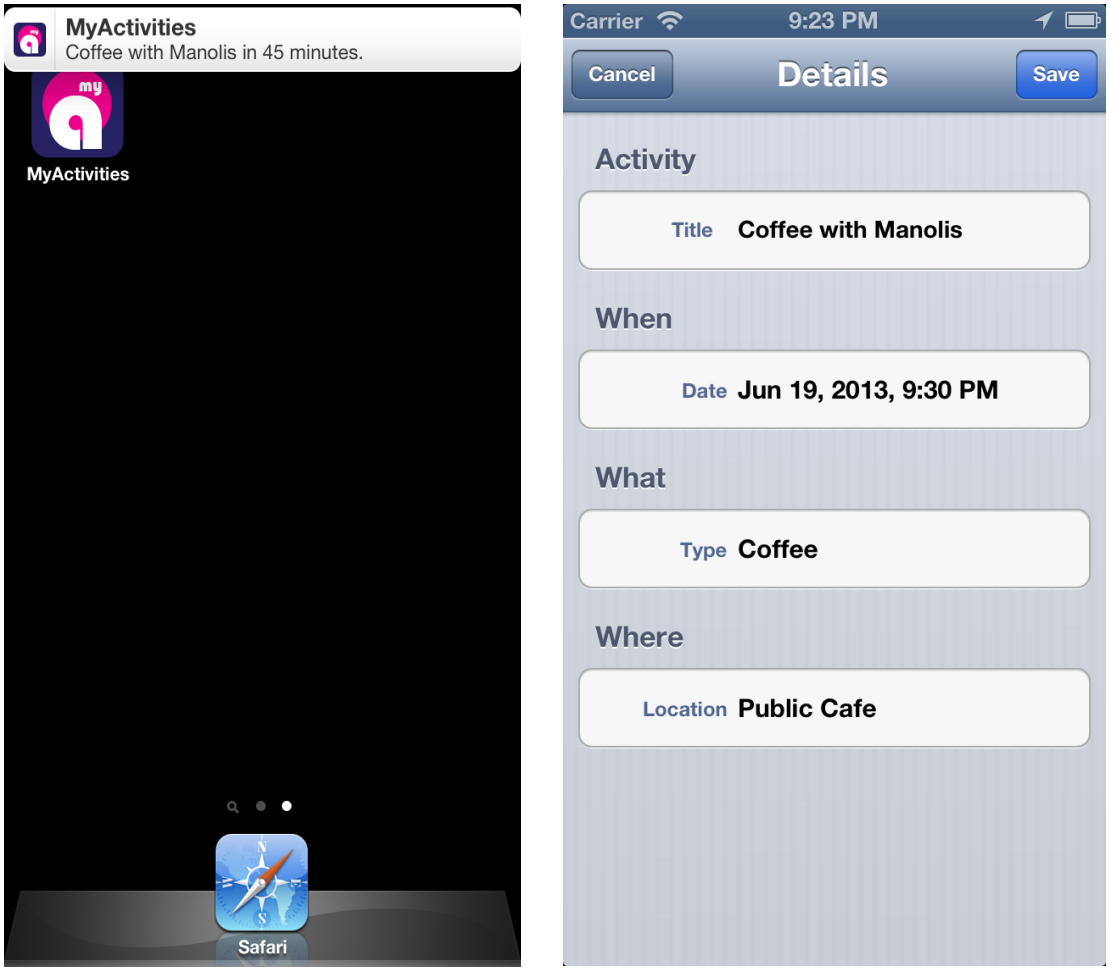
α β

Εικόνα 40 Λίστα με τοποθεσίες και λεπτομέρειες μη προγραμματισμένης δραστηριότητας.

Ο Πέτρος θα προσθέσει άλλη μία μη προγραμματισμένη δραστηριότητα, ακολουθώντας τα ίδια βήματα με πριν μέχρι το σημείο που επιλέγει την τοποθεσία που θα πραγματοποιηθεί η δραστηριότητα. Συγκεκριμένα, θέτει σαν τίτλο "Antibiotics", σαν χρονικό όριο θέτει την ημερομηνία της επόμενης ημέρας και σαν τύπο δραστηριότητας "Pharmacy". Ο Πέτρος όμως θέλει μέχρι την επόμενη μέρα να ειδοποιείται κάθε φορά που θα βρεθεί κοντά σε κάποιο φαρμακείο. Μην επιλέγοντας λοιπόν καμία συγκεκριμένη τοποθεσία, η δραστηριότητα αυτή αποθηκεύεται με την προεπιλογή "All Places" (**Εικόνα 40.β**). Έτσι, ο χρήστης θα ειδοποιείται για όλα τα φαρμακεία που βρίσκονται κοντά του σε όλη τη διάρκεια της μέρας.

Δημιουργείται επίσης μία ειδοποίηση η οποία θα ενημερώσει το χρήστη όταν παρέλθει η χρονική στιγμή που έχει θέσει σαν όριο και θα σταματήσει ο εντοπισμός για αυτή τη δραστηριότητα.

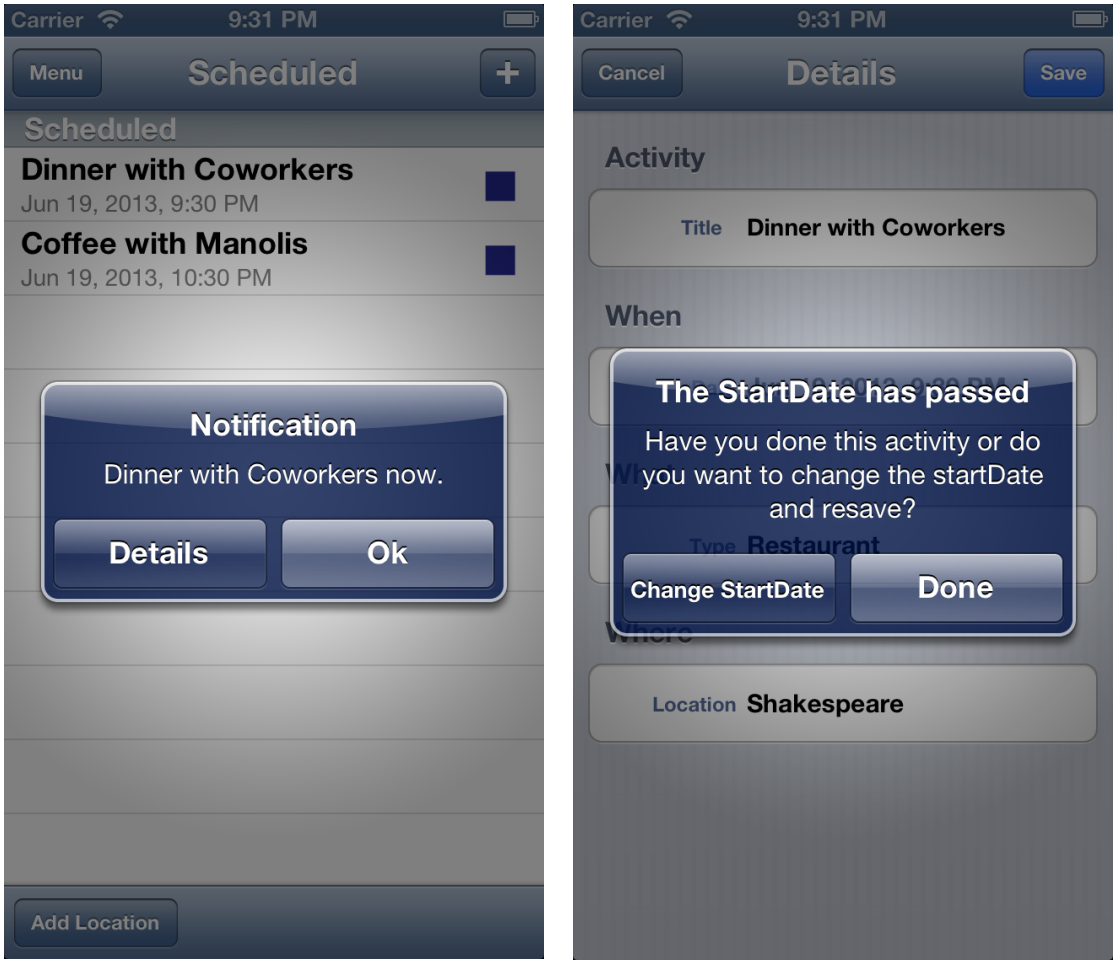
Επόμενο βήμα στο σενάριο είναι ο Πέτρος να βάλει την εφαρμογή στο παρασκήνιο. Μετά από λίγο θα εμφανιστεί μία ειδοποίηση, που θα αφορά την προγραμματισμένη δραστηριότητα “Coffee with Manolis” (**Εικόνα 41.α**). Πατώντας στην περιοχή της ειδοποίησης η εφαρμογή έρχεται στο προσκήνιο και εμφανίζεται η οθόνη με τις λεπτομέρειες της συγκεκριμένης προγραμματισμένης δραστηριότητας (**Εικόνα 41.β**). Ο Πέτρος μπορεί να διαγράψει τη δραστηριότητα, να αλλάξει όποιο από τα στοιχεία επιθυμεί, ή να μην αλλάξει τίποτα. Επιλέγει πατώντας το κουμπί “Save” να επιστρέψει στην κεντρική οθόνη χωρίς να αλλάξει κάτι.



α β

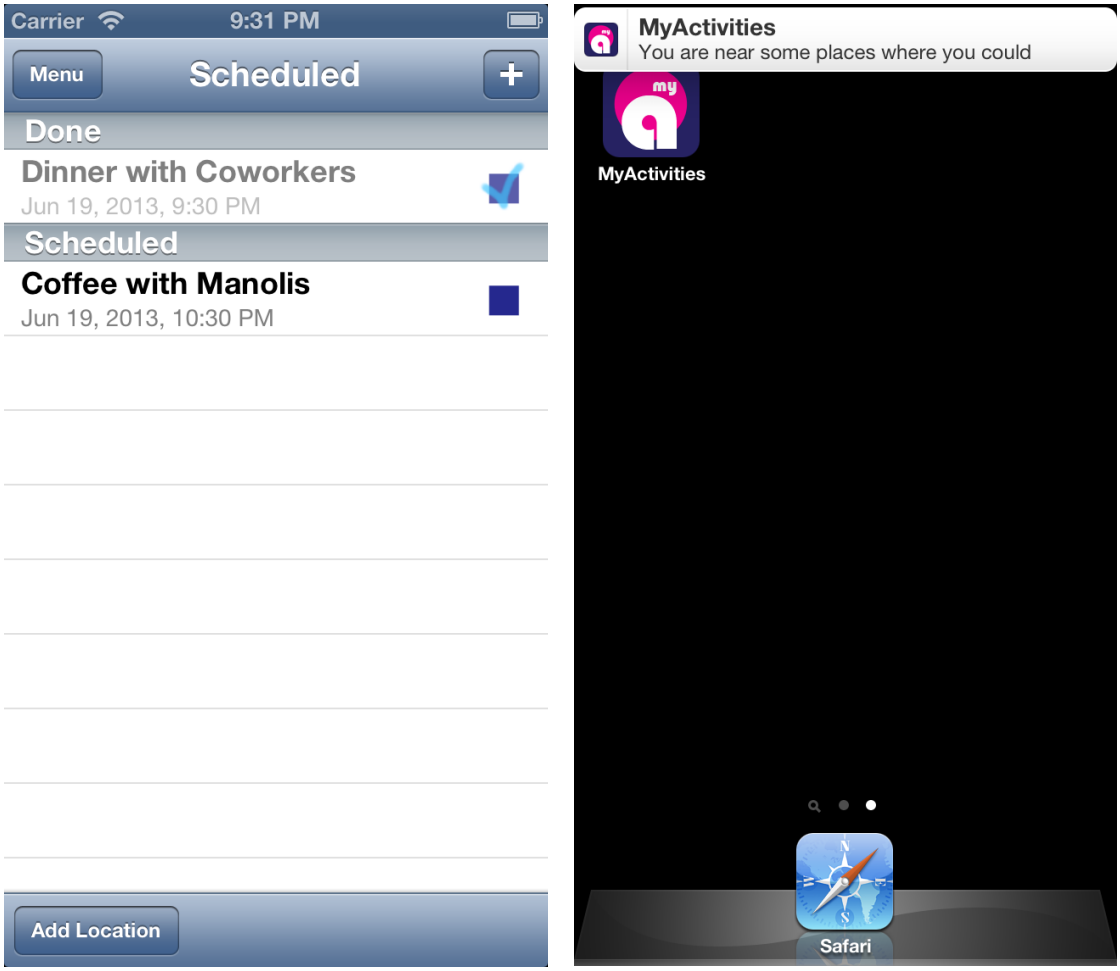
Εικόνα 41 Ειδοποίηση από το παρασκήνιο και λεπτομέρειες προγραμματισμένης δραστηριότητας.

Στη συνέχεια και ενώ βρίσκεται στο προσκήνιο, ο κύριος χρήστης μας δέχεται μία νέα ειδοποίηση, αυτή τη φορά σε μορφή Alert (**Εικόνα 42.α**). Η ειδοποίηση αυτή αφορά την προγραμματισμένη δραστηριότητα "Dinner with Coworkers" και ενημερώνει το χρήστη ότι η στιγμή που είχε αποθηκεύσει σαν ώρα έναρξης της δραστηριότητας έχει περάσει. Τώρα, ο Πέτρος βλέποντας τις λεπτομέρειες της συγκεκριμένης δραστηριότητας πατώντας το κουμπί "Save" χωρίς να έχει αλλάξει κάτι του δίνονται δύο επιλογές, να αλλάξει την ημερομηνία ή να δηλώσει ότι έχει πραγματοποιήσει τη δραστηριότητα (**Εικόνα 42.β**). Επιλέγοντας το δεύτερο οδηγείται στην οθόνη με τη λίστα των μη προγραμματισμένων δραστηριοτήτων (**Εικόνα 43.α**), όπου έχουν διαχωριστεί οι δραστηριότητες που έχουν πραγματοποιηθεί από αυτές που δεν έχουν.



α β

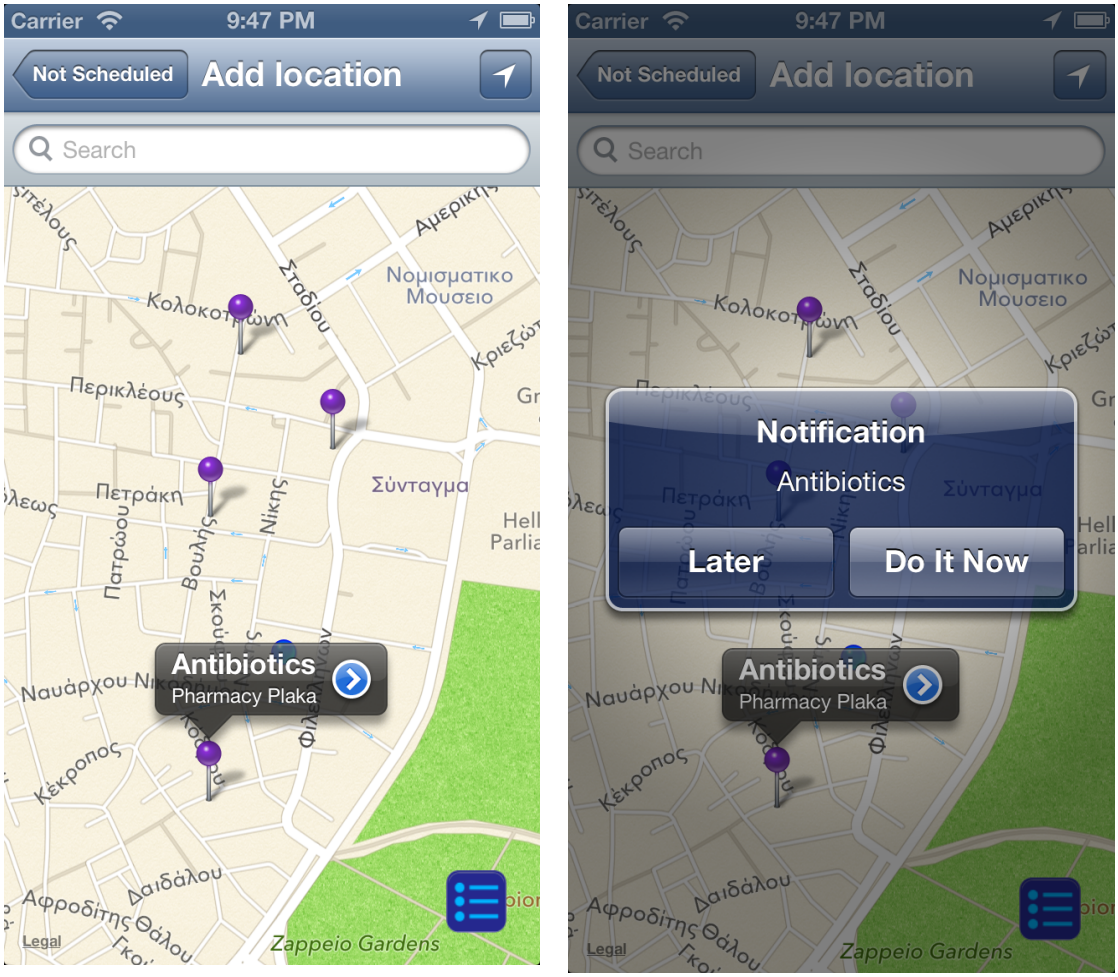
Εικόνα 42 Ειδοποίηση στο προσκήνιο και επιλογή πραγματοποίησης δραστηριότητας.



α β

Εικόνα 43 Λίστα με προγραμματισμένες δραστηριότητες και ειδοποίηση στο παρασκήνιο.

Μετά από λίγο και αφού η εφαρμογή τεθεί στο παρασκήνιο, ο κύριος χρήστης μετακινείται και στην καινούργια προσωρινή τοποθεσία του τοποθεσία και εμφανίζεται μία ειδοποίηση (**Εικόνα 43.β**). Η ειδοποίηση αυτή αφορά περισσότερες από μία δραστηριότητες. Συγκεκριμένα αφορά τις μη προγραμματισμένες δραστηριότητες "Antibiotics" και "Keyboard". Πατώντας στην περιοχή της ειδοποίησης ο Πέτρος οδηγείται στο χάρτη όπου είναι μαρκαρισμένες οι τοποθεσίες που βρίσκονται κοντά στην τωρινή θέση του και αφορούν μη προγραμματισμένες δραστηριότητες, που δεν έχουν πραγματοποιηθεί. Στην **Εικόνα 44.α** φαίνονται δύο φαρμακεία και ένα κατάστημα ηλεκτρονικών ειδών. Αν ο χρήστης επιθυμεί να πραγματοποιήσει κάποια από τις δραστηριότητες του σε κάποια από αυτές τις τοποθεσίες την επιλέγει και εμφανίζεται μία ειδοποίηση σε μορφή Alert (**Εικόνα 44.β**). Από την ειδοποίηση αυτή επιλέγει να πραγματοποιήσει τώρα τη δραστηριότητα "Antibiotics" στην τοποθεσία "Pharmacy Plaka" και επιστρέφει στη λίστα με τις μη προγραμματισμένες δραστηριότητες όπου η δραστηριότητα "Antibiotics" ανήκει τώρα στην κατηγορία αυτών που έχουν πραγματοποιηθεί (**Εικόνα 45.α**).

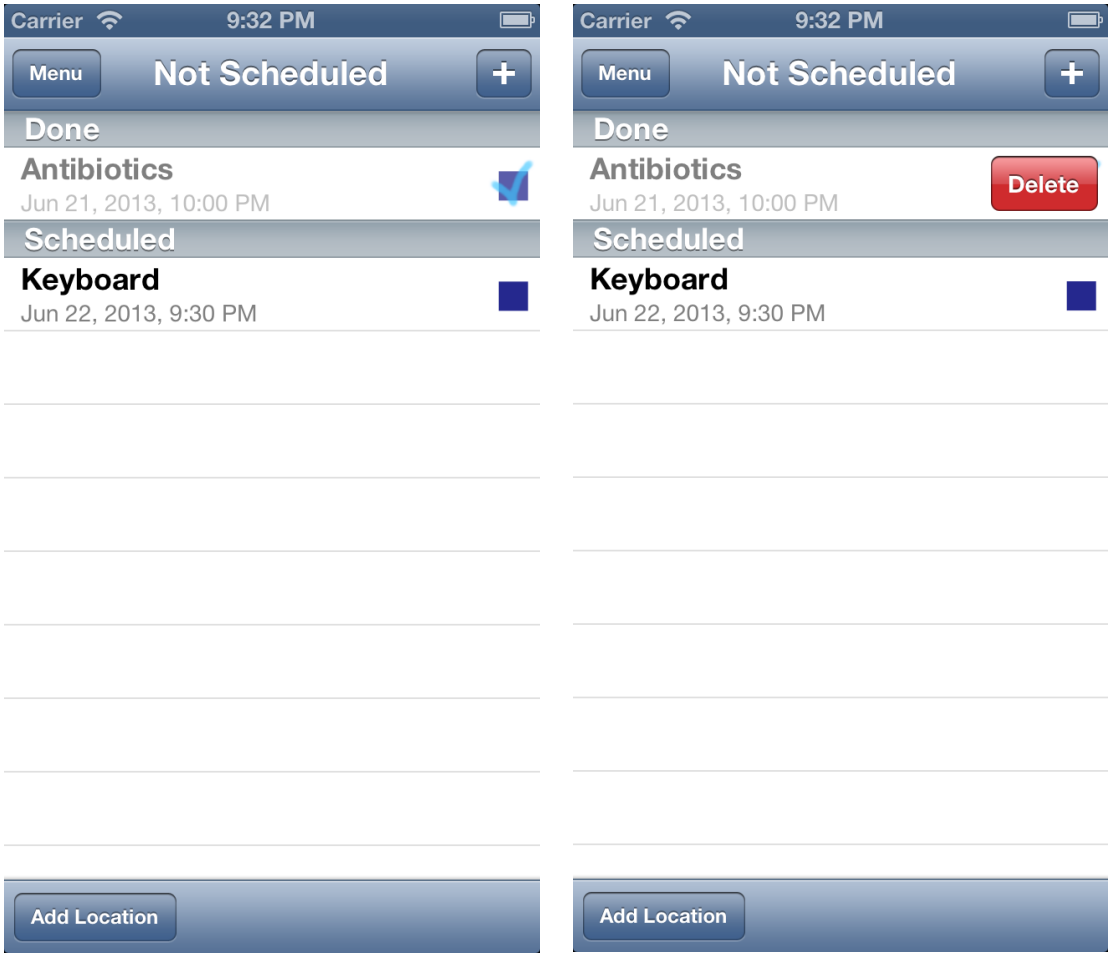


α β

Εικόνα 44 Χάρτης με μαρκαρισμένες τοποθεσίες και ειδοποίηση για πραγματοποίηση δραστηριότητας.

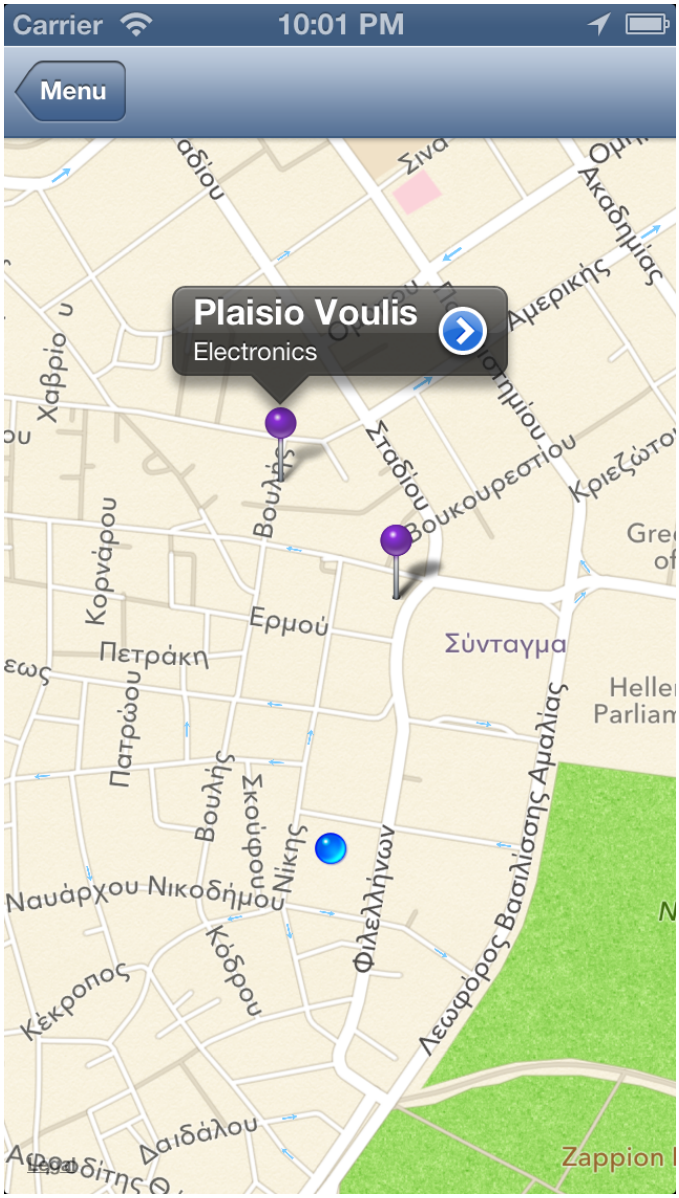
Φυσικά, ανά πάσα στιγμή ο κάθε χρήστης μπορεί να δηλώσει ότι έχει πραγματοποιήσει κάποια δραστηριότητα στις λίστες των προγραμματισμένων και μη προγραμματισμένων δραστηριοτήτων. Μπορεί ακόμα και να διαγράψει κάποια δραστηριότητα είτε έχει πραγματοποιηθεί είτε όχι (**Εικόνα 45.β**). Αυτό κάνει ο Πέτρος για τη δραστηριότητα "Antibiotics" που μόλις πραγματοποίησε.

Ο κάθε χρήστης επίσης μπορεί σε κάθε τοποθεσία στην οποία βρίσκεται μέσω του μενού να δει ένα χάρτη που εκτείνεται σε ακτίνα δύο χιλιομέτρων από την προσωρινή θέση του (**Εικόνα 46**). Σε αυτό το χάρτη θα είναι μαρκαρισμένες οι τοποθεσίες αυτής της περιοχής που μπορούν να ικανοποιήσουν κάποια από τις μη προγραμματισμένες δραστηριότητες που δεν έχουν πραγματοποιηθεί, αλλά και εκείνες στις οποίες έχει κάποια αποθηκεύσει κάποια προγραμματισμένη δραστηριότητα.



α β

Εικόνα 45 Λίστα μη προγραμματισμένων δραστηριοτήτων και διαγραφή μη προγραμματισμένης δραστηριότητας.



Εικόνα 46 Χάρτης με τις ενδιαφέρουσες κοντινές τοποθεσίες.

6

Επίλογος

6.1. Συμπεράσματα

Κατά τη συγγραφή της διπλωματικής εργασίας και την υλοποίησης της εφαρμογής ασχοληθήκαμε με πολλές τεχνολογίες και με την ενασχόληση μας με αυτές παρήχθησαν κάποια χρήσιμα συμπεράσματα.

Η διαδικασία ανάπτυξης μίας εφαρμογής απαιτεί την ανάλυση πολλών διαφορετικών διαδικασιών. Εκτός από τις λειτουργίες και τις υπηρεσίες που θα προσφέρει στο χρήστη, θα πρέπει να ικανοποιεί και πολλά ακόμα κριτήρια για να θεωρηθεί πετυχημένη. Ένα σημαντικό από αυτά είναι η λειτουργικότητα της. Ο χρήστης έχει την απαίτηση, να μπορεί να εκμεταλλευτεί τις λειτουργίες που του προσφέρονται με εύκολο τρόπο. Επίσης, το περιεχόμενο της εφαρμογής θα πρέπει να παρουσιάζεται με όμορφο τρόπο και να ταιριάζουν στη φιλοσοφία της συσκευής του. Ακόμα και η ονομασία της εφαρμογής μπορεί να επηρεάσει την επιτυχία της. Όλα αυτά απαιτούν ξεχωριστή ανάλυση και αποτελούν σημαντικό κομμάτι της ανάπτυξης μίας εφαρμογής.

Οι λειτουργίες που προσφέρουν οι υπηρεσίες τοποθεσίας, είναι ιδιαιτέρως χρήσιμες. Υπάρχουν πάρα πολλές εφαρμογές, που χρησιμοποιούν την τοποθεσία του χρήστη και ανάλογα αυτής προσφέρουν κατάλληλο περιεχόμενο με αποτέλεσμα οι χρήστες να απαιτούν από τις εφαρμογές που χρησιμοποιούν να τους προσφέρουν τέτοιου είδους υπηρεσίες. Στην Ελλάδα οι υπηρεσίες αυτές δεν είναι τόσο διαδεδομένες όσο σε άλλες χώρες και υπάρχουν βήματα που πρέπει να γίνουν ώστε να προσφέρονται οι απαραίτητες πλέον υπηρεσίες.

Για την περάτωση της εφαρμογής χρησιμοποιήθηκε και η τεχνική του crowdsourcing. Στους χρήστες παρέχεται η δυνατότητα να δημιουργούν τις δικές του τοποθεσίες και να τις διαθέτουν στο *server*. Με αυτόν τον τρόπο επιτυγχάνουμε τη δημιουργία μίας τεράστιας βάσης δεδομένων με τοποθεσίες και το τύπο της δραστηριότητας που ικανοποιούν, γεγονός που έχει κυρίως δύο οφέλη. Αρχικά, οι χρήστες έχουν τη δυνατότητα να επιλέξουν ανάμεσα σε έναν τεράστιο αριθμό από τοποθεσίες για να δημιουργήσουν τις δραστηριότητες τους. Το δεύτερο είναι ότι για τη δημιουργία της βάσης δεν απαιτείται η καταγραφή όλων των

τοποθεσιών χειροκίνητα από τους δημιουργούς της εφαρμογής, αλλά γίνεται αυτόματα, απλά με τη χρήση της εφαρμογής από τους χρήστες.

Τέλος, ασχοληθήκαμε με πολλές τεχνολογίες στα πλαίσια αυτής της διπλωματικής, και ειδικά με μερικές από αυτές για πρώτη φορά. Καταλήξαμε λοιπόν σε κάποια συμπεράσματα για την εξέλιξη και το μέλλον των τεχνολογιών αυτών. Παλιότερα, η καινοτομία στην τεχνολογία των υπολογιστών αφορούσε την παραγωγή όλο και πιο δυνατών σταθερών υπολογιστών, οι οποίοι θα μπορούσαν να φέρουν σε πέρας, όλες τις ανάγκες των χρηστών, όσο πολύπλοκες κι αν ήταν αυτές. Στη σημερινή εποχή όμως αυτό έχει διαφοροποιηθεί. Φυσικά και πολλοί χρήστες, κυρίως επαγγελματίες, έχουν ανάγκη τέτοια συστήματα, αλλά οι προτεραιότητες έχουν αλλάξει. Στο κέντρο των αναγκών του χρήστη είναι η φορητότητα. Πλέον, υπάρχει η ανάγκη για εργαλεία που ο χρήστης θα μπορεί να τα μεταφέρει μαζί του, όπως είναι τα «έξυπνα» τηλέφωνα και οι ταμπλέτες (*tablets*). Βεβαίως, είναι απαραίτητο αυτά τα εργαλεία να έχουν μεγάλη υπολογιστική ισχύ, αλλά όση χρειάζεται για τη χρήση που προορίζονται. Ίσως στο μέλλον, η ανάγκη για οικιακούς ηλεκτρονικούς υπολογιστές να μειωθεί περισσότερο και αυτοί να αντικατασταθούν από τις «έξυπνες» φορητές συσκευές.

6.2. Μελλοντικές επεκτάσεις

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας είναι χρήσιμη και εύχρηστη και μπορεί να διατεθεί στο ηλεκτρονικό κατάστημα εφαρμογών της *Apple* και να χρησιμοποιηθεί από τους χρήστες. Αυτό βεβαίως δε σημαίνει και τη διακοπή της ανάπτυξης της. Συνεχώς θα υπάρχουν ιδέες για νέα χαρακτηριστικά και δυνατότητες που θα μπορούσαν να προστεθούν στις ήδη υπάρχουσες με σκοπό φυσικά να ικανοποιούνται όσο καλύτερα γίνεται οι ανάγκες του χρήστη. Επίσης, ο προγραμματιστής έχει το χρέος να διορθώνει τα σφάλματα που προκύπτουν κατά τη χρήση της εφαρμογής και να παρέχει τις απαραίτητες αναβαθμίσεις όποτε αυτό είναι απαραίτητο. Θα αναφερθούμε πιο συγκεκριμένα σε κάποιες πιθανές μελλοντικές επεκτάσεις και διορθώσεις που μπορούν να γίνουν.

Οι εφαρμογές που διαθέτουν εγγεγραμμένους χρήστες θα πρέπει να παρέχουν ασφάλεια όσον αφορά τα προσωπικά δεδομένα των χρηστών. Ο χρήστης κατά την εγγραφή του έπρεπε να συμπληρώσει τα προσωπικά του στοιχεία και τη διεύθυνση ηλεκτρονικού ταχυδρομείου του. Ο *server* και η βάση δεδομένων που χειρίζονται αυτά τα δεδομένα του χρήστη πρέπει να τα προστατεύουν από κακόβουλο λογισμικό και από οποιαδήποτε απόπειρα υποκλοπής αυτών. Επεκτάσεις προς αυτήν την κατεύθυνση χρειάζεται να γίνουν στην περίπτωση της παρούσας εφαρμογής, αφού για να τη χρησιμοποιούν πολλοί χρήστες πρέπει να κερδίσεις την εμπιστοσύνη τους.

Μία ακόμα προέκταση που θα μπορούσε να γίνει είναι προς την κατεύθυνση των κοινωνικών δικτύων. Ο χρήστης ήδη έχει τη δυνατότητα εύρεσης μίας τοποθεσίας από το *Facebook*, αλλά υπάρχουν ακόμα πολλές λειτουργίες που θα μπορούσαν να προστεθούν. Μία από αυτές θα ήταν να μπορείς να κοινοποιήσεις στους φίλους κάποια από τις δραστηριότητες σου, ή ακόμα και να τους προσκαλέσεις. Μία άλλη προσθήκη θα μπορούσε να είναι η ενσωμάτωση άλλων κοινωνικών δικτύων, όπως το *Twitter,* το *Google+* και το *LinkedIn*. Με αυτόν το τρόπο προσελκύεις ακόμα περισσότερους χρήστες και κάνεις πιο ευέλικτη την εφαρμογή.

Η επικοινωνία με τους άλλους χρήστες γίνεται έμμεσα, μέσω του *server* ή μέσω των κοινωνικών δικτύων. Μία προσθήκη της εφαρμογής θα μπορούσε να αφορά την άμεση επικοινωνία μεταξύ των χρηστών. Χρησιμοποιώντας τεχνολογίες επικοινωνίας μεταξύ κοντινών συσκευών (*AirDrop*) ένας χρήστης θα μπορούσε να προτείνει κάποια τοποθεσία σε κάποιον άλλο, ή ακόμα και να προσθέσει μία δραστηριότητα και αυτή να προστεθεί και στο πρόγραμμα του άλλου χρήστη.

Πρόσφατα, παρουσιάστηκε η έβδομη έκδοση του λειτουργικού συστήματος, το *iOS 7*. Ο γραφικός σχεδιασμός του διαφέρει από τις προηγούμενες εκδόσεις. Είναι «μίνιμαλ» και κυριαρχεί το λευκό χρώμα. Κρίνεται λοιπόν απαραίτητο να επανασχεδιασθεί η διεπαφή χρήστη ώστε να επικοινωνεί σωστότερα με την καινούρια φιλοσοφία του *iOS 7*. Οι εφαρμογές, όπως έχουμε αναφέρει, θα πρέπει να συμβαδίζουν με τον τρόπο λειτουργίας του λειτουργικού συστήματος και με το design του, παρέχοντας στο χρήστη μία ολοκληρωμένη εμπειρία.

7

Βιβλιογραφία

[1] Apple, *The Objective-C Programming Language*, 2003

[2] Mark, David, Nutting, Jack, LaMarche, Jeff, Olsson, Fredrik, *Beginning iOS 6 Development: Exploring the iOS SDK*, 2013

[3] Mark, David, Horovitz, Alex, Kim, Kevin, LaMarche, Jeff, *More iOS 6 Development: Further Explorations of the iOS SDK*, 2013

[4] Sadun, Erica, *The Core iOS 6 Developer's Cookbook*, 2013

[5] Fritz F. Anderson, *Xcode 4 Unleashed*, 2012

[6] Joe Conway, Aaron Hillegass, *iOS Programming: The Big Nerd Ranch Guide*, Big Nerd Ranch Guides, 2012

[7] Jack Nutting , Peter Clark, *Learn Cocoa on the Mac*, 2013

[8] Giacomo Andreucci, *Pro iOS Geo Building Apps with Location Based Services*, 2013

[9] Michael Privat , Robert Warner, *Pro Core Data for iOS*, 2011

[10] Bryan Basham, Kathy Sierra and Bert Bates, *Head First Servlets and JSP*, 2012

[11] Kathy Sierra and Bert Bates, *Head First Java*, 2012

[12] Apple developer portal , http://developer.apple.com

[13] Facebook developers portal, http://developers.facebook.com

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

[15] Java, http://www.oracle.com/technetwork/java

[16] Hibernate framework, http://www.hibernate.org