Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

Σύστημα Οργάνωσης και Δρομολόγησης Δραστηριοτήτων με χρήση GPS σε πλατφόρμα Android.

Μώρος-Σμυρνάκης Αθανάσιος ΑΜ2514
Βασιλακής Ιωάννης ΑΜ2624

Επιβλέπον Καθηγητής: Νικόλαος Παπαδάκης
Επιτροπή Αξιολόγησης:
Ημερομηνία Παρουσίασης:

Ευχαριστίες

Στον καθηγητή μας κ. Νίκο Παπαδάκη για την καθοδήγησή του, στους Κώστα Καλαφάτη, Πέτρο Βαρχαλαμά, Οδυσσέα Λάκκα, Κωνσταντίνο Μήτση, Κέλλυ Καλουτά, Θανάση Τριανταφυλλάκο, Τζούλιο Ίμπρο για την ανεκτίμητη βοήθεια τους, και πάνω από όλα στην οικογένειά μας που πιστεύοντας σε εμάς παρείχαν αμέριστη συμπαράσταση και υποστήριξη όλα αυτά τα χρόνια, δίχως των οποίον τίποτε από όλα όσα επιτεύχθηκαν δεν θα ήταν δυνατά˙ σε όλους αυτούς λοιπόν θα θέλαμε να αφιερώσουμε αυτήν την εργασία λέγοντας ένα μεγάλο, ένα γιγάντιο ευχαριστώ και ότι ελπίζουμε κάποια μέρα να μπορέσουμε να τους το ξεπληρώσουμε.

Abstract

This thesis focuses on the development of an Android based application that can schedule everyday appointments while being able to navigate to the nearest point of interest in the city of Heraklion always taking into account the future appointments of the user and the time it takes to drive there depending on the time of day. Furthermore the various technologies that strive to achieve similar goals in the aspect of navigation and appointment management on handheld devices are detailed here.

The purpose of this thesis is to describe the development of a location aware android application, which can be expressed as latitude and longitude, as well as comparing its functionality and usability to the already established applications of similar purpose, while detailing the evolution of such applications on handheld devices that have been developed over the years.

Keywords: **Navigation**, **Android**, **Google Maps**, **Electronic Organizer**, **Smartphone Application**, **Appointment Scheduling System**

Σύνοψη

Η πτυχιακή αυτή εργασία εστιάζει στην ανάπτυξη μίας εφαρμογής βασισμένης στην πλατφόρμα Android, η οποία θα έχει την δυνατότητα προγραμματισμού διαφόρων καθημερινών δραστηριοτήτων ενώ παράλληλα θα μπορεί να δρομολογήσει τον χρήστη στα συγκεκριμένα σημεία αυτών των δραστηριοτήτων έχοντας υπόψη τις μελλοντικές προγραμματισμένες δραστηριότητες του χρήστη καθώς και τον χρόνο που χρειάζεται για να μεταβεί σε αυτές ανάλογα με την τρέχουσα ώρα. Επιπλέον περιγράφει τις διάφορες τεχνολογίες που έχουν αναπτυχθεί για την επίτευξη παρόμοιας λειτουργικότητας σε κινητές συσκευές όπως ο χρονοπρογραμματισμός δραστηριοτήτων και η δρομολόγηση του χρήστη σε αυτές.

Ο σκοπός της πτυχιακής αυτής εργασίας είναι να περιγράψει σαφώς την ανάπτυξη μιας android εφαρμογής η οποία θα έχει τη δυνατότητα χωρικής αντίληψης σε έκφραση γεωγραφικού μήκους και πλάτους, καθώς και την σύγκριση της λειτουργικότητας και της ευχρηστίας μιας τέτοιας εφαρμογής σε σχέση με τις τρέχουσες καθιερωμένες εφαρμογές παρόμοιας χρήσης, ενώ γίνεται αναφορά στην εξέλιξη τέτοιων εφαρμογών για κινητές συσκευές που έχουν αναπτυχθεί μέσα στα χρόνια.

Λέξεις Κλειδιά: **Πλοήγηση**, **Android**, **Χάρτες Google**, **Ηλεκτρονική Ατζέντα**, **Εφαρμογή Smartphone**, **Προγραμματισμός Συναντήσεων**

Περιεχόμενα

Περιεχόμενα **5**

1. Εισαγωγή **7**

1.1 Περίληψη 7
1.2 Κίνητρο Διεξαγωγής της Εργασίας 8
1.3 Σκοπός και Στόχοι της Εργασίας 8

2. Εξέλιξη της Πλοήγησης **9**

2.1 Η έννοια της Πλοήγησης και της Δρομολόγησης 9
2.2 Η ανάγκη για πλοήγηση 9
2.3 Η Ιστορία της Πλοήγησης 10
2.4 Όργανα Πλοήγησης 11
2.5 Μέσα πλοήγησης στον 20^{ο} αιώνα 13
2.5.1 To Sonar 13
2.5.2 To Radar 14
2.6 G.P.S. Global Positioning System 15
2.6.1 Βασική Δομή και Λειτουργία του G.P.S. 15
2.6.2 Η Ιστορία και η εξέλιξη του G.P.S. 15
2.6.3 Το Χρονικό του G.P.S. 17
2.6.4 Η Λειτουργία του NAVSTAR 19

3. Τεχνολογική Στάθμιση **21**

3.1 Κινητές Συσκευές 21
3.1.1 Η εξέλιξη των κινητών συσκευών 23
3.2. Η εξέλιξη των λειτουργικών συστημάτων σε κινητές συσκευές 28
3.3 Η πλατφόρμα Android 30
3.3.1 Η γέννηση του Android 30
3.3.2 Τι είναι το Android 30
3.3.3 Χαρακτηριστικά του Android 31
3.3.4 Αρχιτεκτονική του Android 32
3.3.5 Activities and their Lifecycle 34
3.4 Σύγκριση Σύγχρονων mobile OS με την πλατφόρμα Android 36
3.4.1 Symbian 36
3.4.2 RIM BlackBerry 37
3.4.3 Palm OS - WebOS 37
3.4.4 Windows Mobile 38
3.4.5 iOS 38

4. Σχεδιασμός Εφαρμογής **41**

4.1 Θεωρίες 42
4.2 Αλγόριθμοι 42
4.3 Ανάλυση Προβλήματος 43

4.4 Βασικές Λειτουργίες 44
4.5 Βασική Δομή Εφαρμογής: Οι Ρόλοι των Activities 45
4.6 Προσδιορισμός Απαιτήσεων 47
4.6.1 LoginActivity *47*
4.6.2 SignUpActivity *47*
4.6.3 Profile *47*
4.6.4 Organizer *48*
4.6.5 NewAppointment *48*
4.6.6 DateTimePick *48*
4.6.7 MoviePicker *48*
4.6.8 PlanMap *49*
4.6.9 InstantMap *49*
5.2 Έλεγχος Εγκυρότητας Ραντεβού 50
5.2.1 Τι είναι ο Έλεγχος Εγκυρότητας Ραντεβού; *50*
5.2.2 Σκοπός του Ελέγχου Εγκυρότητας Ραντεβού *50*
5.2.3 Έλεγχος Χρόνου *50*
5.2.4 Έλεγχος Κατεύθυνσης *60*
5.2.5 Έλεγχος Ωραρίου *65*
4.7 Ανάλυση Σεναρίων Χρήσης 67
4.8 Απαιτήσεις Συστήματος 76
4.9 Εργαλεία και Τεχνολογίες που χρησιμοποιήθηκαν 77

5. Υλοποίηση Εφαρμογής 78

5.1 Supporting Multiple Sizes of Screens 80
5.3 Διαχείριση Λογαριασμών 86
5.3.1 LoginActivity *86*
5.2.2 SignUpActivity *90*
5.2.3 Profile *93*
5.3 Διαχείριση Ραντεβού 95
5.3.1 Organizer *95*
5.3.2 NewAppointment *100*
5.3.3 DateTimePick *104*
5.3.4 MoviePicker *106*
5.5 Διαχείριση Δρομολόγησης/Χαρτών 110
5.4.2 PlanMap *110*
5.4.2 InstantMap *116*

6. Αποτελέσματα 118

8.1 Συμπεράσματα 118
8.2 Προτάσεις για Μελλοντικές Επεκτάσεις 119

7. Βιβλιογραφία 120

1. Εισαγωγή

1.1 Περίληψη

Η εργασία αυτή αποτελεί μια προσπάθεια ανάπτυξης μιας εφαρμογής με δυνατότητα χωρικής αντίληψης εκφρασμένη σε γεωγραφικό ύψος και πλάτος αλλά και συγκεκριμένων σημείων ενδιαφέροντος στα οποία ο χρήστης θα μπορεί να ζητήσει δρομολόγηση από οπουδήποτε. Τέτοια σημεία είναι τα supermarket, τα πρατήρια βενζίνης καθώς και οι κινηματογράφοι που βρίσκονται μέσα στα όρια της πόλης του Ηρακλείου. Η εφαρμογή θα έχει επίσης την δυνατότητα χρονοπρογραμματισμού των υποχρεώσεων του χρήστη με τέτοιο τρόπο ώστε να μπορεί να βρίσκεται την επιλεγμένη ώρα σε όλα τα καταχωρημένα ραντεβού του όσον αφορά τα συγκεκριμένα σημεία ενδιαφέροντος παίρνοντας πάντα υπόψη το ωράριο λειτουργίας του επιλεγμένου σημείου, την κατεύθυνση σε σχέση με την τρέχουσα θέση του χρήστη, το χρόνο μετακίνησης, και το χρόνο στάθμευσης ανάλογα με την επιλεγμένη ώρα.

Σαν δεύτερο σκέλος γίνεται αναφορά στα βασικά μέρη που αποτελούν τον πυρήνα πίσω από τη λογική υλοποίησης της εφαρμογής. Το πρώτο βασικό μέρος είναι η πλοήγηση ως όλον, η οποία περιλαμβάνει ποικίλες πτυχές όπως σημειολογία, αλγόριθμοι συντομότερης διαδρομής και άλλα τα οποία θα αναλυθούν εκτενέστερα σε παρακάτω κεφάλαιο. Το δεύτερο μέρος έχει να κάνει με τον χρονοπρογραμματισμό το οποίο περιλαμβάνει πολλαπλές συνθήκες για να επιτεύξουμε το επιθυμητό αποτέλεσμα, το οποίο στην περίπτωσή μας είναι η οργάνωση των ραντεβού του χρήστη με τέτοιο τρόπο ώστε να μπορεί να τα προλάβει όλα. Το τρίτο μέρος είναι η τεχνολογία στης οποία θα εργαστούμε, η πλατφόρμα Android, την οποία θα αναλύσουμε παρακάτω. Δεν θα μείνουμε όμως μόνο στην συγκεκριμένη τεχνολογία, αλλά θα επιχειρήσουμε να διερευνήσουμε και τις υπόλοιπες εναλλακτικές τεχνολογίες παρόμοιας λειτουργικότητας συγκρίνοντας την παρούσα αλλά και την αρχική εξέλιξη που έχουν διατρέξει ανά τα χρόνια.

1.2 Κίνητρο Διεξαγωγής της Εργασίας

Το κίνητρο διεξαγωγής της εργασίας είχε αρχικά εκπαιδευτικό χαρακτήρα. Η τεχνολογία της πλατφόρμας Android είναι πλέον μια από τις πιο διαδεδομένες με ραγδαία εξέλιξη καθώς η χρήση των smartphones αυξάνεται με ταχύτατους ρυθμούς. Είναι πασιφανές λοιπόν ότι η εξοικείωση με μια τόσο διαδεδομένη τεχνολογία θα είχε μεγάλο όφελος από επαγγελματική άποψη, δεν ήταν όμως ο μόνος λόγος που επιλέχθηκε. Η ανάπτυξη μιας τέτοιας εφαρμογής αποτελεί μια ολοκληρωμένη προσπάθεια δημιουργίας κάτι εντελώς καινούριου σχεδόν ‘από το μηδέν’, αυτό λοιπόν σχηματίζει ένα σπάνιο είδος δημιουργικότητας που προσωπικά σαν προγραμματιστές επιζητάμε, ιδιαίτερα σε έναν κλάδο που η κατάτμηση και εξειδίκευση στον τομέα της ανάπτυξης λογισμικού έχει ‘τμηματικό χαρακτήρα’, κάτι που υποσκιάζει σε κάποιον βαθμό την αίσθηση δημιουργικότητας όταν συντίθεται κάτι από την αρχή ως το τέλος εξετάζοντας προσωπικά όλες τις πτυχές και τα στάδια της ανάπτυξης του λογισμικού.

1.3 Σκοπός και Στόχοι της Εργασίας

Ο σκοπός της πτυχιακής εργασίας είχε και αυτός στην αρχή χαρακτήρα εκπαιδευτικό, για τους λόγους που αναφέρθηκαν παραπάνω. Στην πορεία όμως ο σκοπός αυτός εξελίχθηκε και πήρε ως ένα βαθμό μια πιο διερευνητική μορφή. Το αντικείμενο το οποίο μελετάται βασίζεται σε υπάρχουσες ευρύτατα διαδεδομένες τεχνολογίες που έχουν καθιερωθεί στον χώρο ανάπτυξης λογισμικού όσον αφορά κινητές συσκευές αλλά και εργαλεία πλοήγησης γενικότερα. Η έννοια της διερεύνησης λοιπόν είχε τη μορφή σύγκρισης όλων των καθιερωμένων τεχνολογιών αρχικά μεταξύ τους και τέλος με την παρούσα εφαρμογή, βλέποντας έτσι τις ελλείψεις και τα περιθώρια βελτίωσης όχι μόνο στην εφαρμογή αλλά και ευρύτερα στις πλέον καθιερωμένες τεχνολογίες που δεσπόζουν στον τομέα ανάπτυξης λογισμικού για κινητές συσκευές.

2. Εξέλιξη της Πλοήγησης

2.1 Η έννοια της Πλοήγησης και της Δρομολόγησης

Η πλοήγηση αρχικά είχε συσχετιστεί αποκλειστικά με την οδήγηση πλοίων σε γνωστά ή και άγνωστα νερά μέχρι τον τελικό προορισμό τους με χρήση διαφόρων μεθόδων στηριζόμενες κυρίως στην Αστρονομία και την Φυσική. Με την πάροδο όμως των ετών και τις δραστικές αλλαγές που έφερε η τεχνολογική άνοδος στις μετακινήσεις η πλοήγηση πήρε μια πιο ευρύτερη έννοια η οποία σχετίζεται πλέον με την οδήγηση στο σωστό προορισμό για κάθε είδους μετακίνηση.

Η δρομολόγηση αναφέρεται κυρίως στην εύρεση δρόμου με την ευρύτερη έννοια, είτε αυτός είναι φυσικός δρόμος διέλευσης αυτοκινήτων, είτε εύρεση μονοπατιού για πεζούς σε μια πόλη, είτε την πιο αφηρημένη έννοια της εύρεσης συντομότερου μονοπατιού στα δίκτυα ηλεκτρονικών υπολογιστών.

Οι ομοιότητες μεταξύ των δύο εννοιών είναι πιο πολλές από τις διαφορές. Η δρομολόγηση αφορά αποκλειστικά καθοδήγηση πάνω σε κάποιο δρόμο ή μονοπάτι, το οποίο μπορεί να είναι ή φυσικό ή ιδεατό στην περίπτωση των δικτύων. Η πλοήγηση από την άλλη πλευρά αναφέρεται πιο αφηρημένα στην εύρεση προορισμού και την οδήγηση προς αυτόν ανεξαρτήτως ύπαρξης δρόμου ή μονοπατιού. Είναι σαφώς πιο συνηθισμένο να χρησιμοποιούμε την έννοια της δρομολόγησης όταν αναφερόμαστε σε χερσαία μέσα, καθώς οι περισσότερες χερσαίες μετακινήσεις γίνονται πάνω σε κάποιου είδους δρόμου, αυτό όμως δεν σημαίνει πως η χρήση της έννοιας της πλοήγησης για χερσαία καθοδήγηση είναι λανθασμένη. Παρακάτω θα αναφερθούμε στην πλοήγηση ως όλον συμπεριλαμβανομένου σαφώς και της δρομολόγησης.

2.2 Η ανάγκη για πλοήγηση

Η ανάγκη για πλοήγηση προέρχεται προφανώς από την ανάγκη για μετακίνηση. Η ραγδαία εξέλιξη της τεχνολογίας στον τομέα των μέσων μεταφοράς έκανε σαφώς την ανάγκη για ακριβής πλοήγηση ολοένα και πιο σημαντική, δεν ξεκίνησε όμως από εκεί. Η ανάγκη για πλοήγηση είναι τόσο παλιά όσο και η ανάγκη για μετακίνηση, έτσι παρατηρούμε ότι από τα αρχαία ακόμη χρόνια ξεκίνησαν να αναπτύσσονται μέθοδοι, στηριζόμενοι στις αναπτυσσόμενες επιστήμες της Αστρονομίας και της Φυσικής, που θα δίνανε τη βάση για την εξέλιξη της πλοήγησης σε τέτοιο βαθμό που να την καθιστά απαραίτητο εργαλείο στις μετακινήσεις παντός είδους τόσο στα αρχαία χρόνια όσο και στις νεότερες εποχές.

2.3 Η Ιστορία της Πλοήγησης

Οι πρώτοι πολιτισμοί που ασχολήθηκαν με την τέχνη της πλοήγησης ήταν ναυτικοί λαοί. Υπάρχουν ιστορικές αναφορές για αρχαίους Έλληνες, Φοίνικες και Αιγύπτιους που ταξίδευαν στα νερά της Μεσογείου με τίποτα παραπάνω από τη θέση των αστεριών. Ο προσανατολισμός των πλοίων με βάση την θέση των αστεριών αναφέρεται στην Ομήρου Οδύσσεια όταν η Καλυψώ λέει στον Οδυσσέα να κρατήσει την Άρκτο στα αριστερά του ενώ παρατηρεί την θέση των Πλειάδων. Οι αρχαίοι Μινωίτες ήταν ακόμα ένα παράδειγμα ναυτικού λαού που κατεύθυναν τα πλοία τους με βάση την θέση συγκεκριμένων άστρων, κάτι που μπορεί να παρατηρηθεί και στην αρχιτεκτονική τους, πολλά από τα κτήρια τους ήταν χτισμένα με κατεύθυνση του σημείου ανατολής κάποιου συγκεκριμένου αστερισμού. Οι αρχαίοι ναυτικοί λοιπόν είχαν αναπτύξει μεθόδους προσέγγισης της τοποθεσίας τους που υπολογιζόταν είτε παρατηρώντας τις θέσεις των αστεριών είτε την μορφή των ακτών ή με την προσεγγιστική μέθοδο του ‘στίγμα εξ αναμετρήσεως’ ή dead reckoning όπως έγινε γνωστό μεταγενέστερα. Η τεχνική του dead reckoning απαιτούσε γνώση μόνο της ταχύτητας του πλοίου και του προσανατολισμού του, καθιστώντας την έτσι ιδιαίτερα εύχρηστη, κάνοντας λοιπόν τους κατάλληλους υπολογισμούς μετρώντας τον χρόνο σε ένα σταθερό χρονικό διάστημα και υπολογίζοντας την απόσταση του πλοίου από μια σταθερή θέση μπορούσαν να εντοπίσουν προσεγγιστικά την θέση του πλοίου. Η μέθοδος όμως αυτή έδινε μεγάλο περιθώριο σφάλματος καθώς στην θάλασσα αυτού του τύπου οι υπολογισμοί δεν ήταν δυνατό να είναι πάντοτε ακριβείς λόγω απρόβλεπτων ανέμων θαλάσσιων ρευμάτων και άλλων παραγόντων οι οποίοι είχαν ως αποτέλεσμα το πλοίο να αποκλίνει από την πορεία του παράγοντας έτσι σημαντικά λάθη στον υπολογισμό της θέσης του.

Μεταγενέστερα στα χρόνια των εξερευνητών η τέχνη της πλοήγησης πήρε ακόμη μεγαλύτερη σημασία, με χαρακτηριστικά εργαλεία πλοήγησης της εποχής την πυξίδα τον εξάντα και βεβαίως την εξέλιξη της χαρτογραφίας. Αξιοσημείωτο είναι το μεγάλο εγχείρημα της εποχής για την εύρεση ακριβούς γεωγραφικού μήκους που καθιστούσε το κύριο πρόβλημα για την πλοήγηση κάθε είδους πλοίου της εποχής. Μετά από χρόνια θυελλωδών ερευνητικών εγχειρημάτων δύο ήταν οι κύριες μέθοδοι υπολογισμού που αναπτύχθηκαν. Η πρώτη μέθοδος χρησιμοποιήθηκε για αρκετά χρόνια πριν την ανακάλυψη της δεύτερης μεθόδου και περιελάμβανε τον υπολογισμό του γεωγραφικού μήκους με βάση την σελήνη. Η μέθοδος προϋπόθετε πληθώρα από μαθηματικούς υπολογισμούς γωνίας θέσης και απόστασης της σελήνης ώστε να βρεθεί προσεγγιστικά η θέση του πλοίου ως προς το γεωγραφικό μήκος, έδινε όμως μεγάλο περιθώριο σφάλματος γι αυτό και εγκαταλείφτηκε μετά την ανακάλυψη της δεύτερης μεθόδου. Η δεύτερη μέθοδος ήταν γνωστή για αρκετά χρόνια, προϋπέθετε την μέτρηση του χρόνου σε ένα σταθερό σημείο, για παράδειγμα το λιμάνι απόπλευσης, και έπειτα την μέτρηση του χρόνου κατά τη διάρκεια του ταξιδιού. Μέσα λοιπόν από μια σειρά υπολογισμών μπορούσαν να έχουν το γεωγραφικό μήκος με ικανοποιητική ακρίβεια, το πρόβλημα όμως ήταν πρακτικό. Στην ξηρά η μέθοδος λειτουργούσε θαυμάσια, στην θάλασσα όμως με τις αλλαγές της θερμοκρασίας και την έντονη υγρασία τα ρολόγια της εποχής είχαν μεγάλες απώλειες με αποτέλεσμα να εισάγεται μεγάλος συντελεστής σφάλματος στον υπολογισμό του γεωγραφικού μήκους.

Την λύση ήρθε να δώσει ένας αυτοδίδακτος Άγγλος ωρολογοποιός με το όνομα John Harrison. Ο Harrison εφεύρε έναν τύπο χρονομέτρου φτιαγμένο από ειδικά κράματα μετάλλων τα οποία είχαν τέτοιες ιδιότητες ώστε όταν το ένα διαστελλόταν λόγω θερμοκρασίας το άλλο συστελλόταν με αποτέλεσμα το χρονόμετρο να μην έχει σημαντικές απώλειες ακόμα και στις αντίξοες συνθήκες που συναντούσε κανείς σε μακρινά ταξίδια στην θάλασσα, με αποτέλεσμα να υπάρξει επιτέλους ο πρώτος αξιόπιστος τρόπος υπολογισμού γεωγραφικού μήκους στη θάλασσα.

Sobel, D. (1998), Brunner, W. (2005)

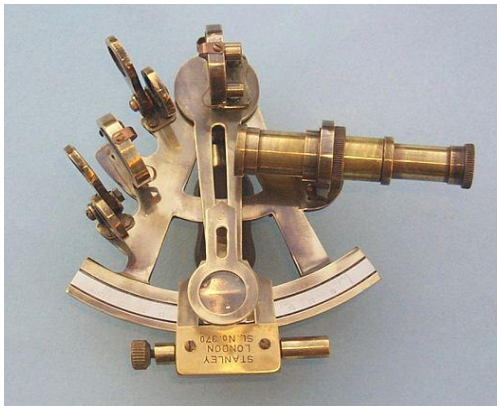
2.4 Όργανα Πλοήγησης

Εδώ θα αναφέρουμε επιγραμματικά κάποια από τα πιο χαρακτηριστικά όργανα πλοήγησης που χρησιμοποιήθηκαν από αρχαίους αλλά και από νεότερους πλοηγούς που υπό μια ευρύτερη έννοια κάποιος θα μπορούσε να θεωρήσει προκάτοχους του σύγχρονου G.P.S.

Αστρολάβος
225π.Χ.
Ἀπολλώνιος ὁ Περγαῖος

Μαγνητική Πυξίδα
206π.Χ.
Αρχαία Κίνα, Δυναστεία Χαν

Ηλιακή Πυξίδα
1836μ.Χ.
William Austin Burt



Εξάντας
1757μ.Χ.
John Bird

Οκτάντας
1731μ.Χ.
John Hadley και Thomas Godfrey

Γυροσκόπιο
1743μ.Χ.
John Serson

Morrison, J. E. (2007), Vardalas, J. (2013)

2.5 Μέσα πλοήγησης στον 20ο αιώνα

Η πληθώρα των σημερινών θαλάσσιων μέσων πλοηγείται με τη χρήση οργάνων των οποίον η εφεύρεση είδε φώς κατά μέσο όρο τον 20ο αιώνα. Τέτοιες τεχνολογίες όπως το sonar το radar και το G.P.S. άλλαξαν ριζικά τον τρόπο πλοήγησης και κατά συνέπεια των μεταφορών γενικότερα. Οι περισσότερες από αυτές τις τεχνολογίες αναπτύχθηκαν αρχικά για στρατιωτικούς λόγους, ανίχνευση εχθρικών σκαφών κλπ, με το πέρασμα τον ετών όμως η χρήση κάποιων από αυτών των τεχνολογιών έγινε διαθέσιμη στο ευρύτερο κοινό, είτε για εμπορική είτε για προσωπική χρήση.

2.5.1 Το Sonar

Η πρώτη μελέτη που έβαλε τα θεμέλια για την γέννηση της τεχνολογίας του Sonar έγινε από τον Daniel Colloden το 1822 ο οποίος μέτρησε την ταχύτητα με την οποία διασχίζει ο ήχος μια συγκεκριμένη απόσταση υποβρυχίως στην λίμνη της Γενεύης στην Ελβετία.

Το πρώτο Sonar κατασκευάστηκε το 1906 από τον Lewis Nixon σαν ένα εργαλείο ανίχνευσης παγόβουνων, το πραγματικό ενδιαφέρον όμως για την συγκεκριμένη τεχνολογία αναδύθηκε στον πρώτο παγκόσμιο πόλεμο καθώς η ανάγκη ανίχνευσης υποβρύχιων εχθρικών σκαφών ήταν απαράμιλλης σημασίας. Το 1915 ο Paul Langévin κατασκεύασε το πρώτο sonar ικανό να ανιχνεύσει υποβρύχια, το οποίο ονομάστηκε «ηχώ τοποθεσίας για ανίχνευση υποβρυχίων» (echo location to detect submarines).

Τα πρώτα sonar ήταν παθητικές ακουστικές συσκευές που σημαίνει κανένα σήμα δεν αποστελλόταν από την ίδια την συσκευή, απλώς ανίχνευε παθητικά υπάρχοντα ηχητικά κύματα. Το 1918 και η Βρετανία και η Αμερική είχαν καταφέρει να κατασκευάσουν ενεργητικές συσκευές sonar οι οποίες απόστελλαν ανεξάρτητα ηχητικά σήματα τα οποία μετά ανίχνευαν καθώς επέστρεφαν στον ανιχνευτή. Ο όρος «Sonar» χρησιμοποιήθηκε για πρώτη φορά από τους Αμερικανούς στον δεύτερο παγκόσμιο πόλεμο και ήταν τα αρχικά για το «Sound Navigation and Ranging» που σημαίνει Ηχητική Πλοήγηση και Μέτρηση Απόστασης.

Bellis, M. (2017)

2.5.2 To Radar

Η ανάπτυξη της τεχνολογίας του radar άρχισε την δεκαετία του 1930, η βασική ιδέα όμως που έβαλε τα θεμέλια για τη δημιουργία του είχε συλληφθεί αρκετά χρόνια νωρίτερα από την εποχή του Heinrich Hertz ο οποίος στα τέλη της δεκαετίας του 1880 επιχείρησε να επαληθεύσει πειραματικά τις θεωρίες του Σκοτσέζου φυσικού James Clerk Maxwell. Οι εξισώσεις του Maxwell για το ηλεκτρομαγνητικό πεδίο καθόριζαν ότι και το φως και τα ραδιοκύματα υπακούουν στους ίδιους θεμελιώδεις νόμους αλλά οι συχνότητές τους διαφέρουν κατά μεγάλο βαθμό. Η μελέτη του Maxwell έφτανε στο συμπέρασμα ότι τα ραδιοκύματα μπορούν να ανακλαστούν από μεταλλικές επιφάνειες και να διαθλαστούν από διηλεκτρικά μέσα ακριβώς όπως γίνεται και με τις ακτίνες του φωτός. Ο Hertz λοιπόν έδειξε πειραματικά αυτές τις ιδιότητες το 1888, χρησιμοποιώντας ραδιοκύματα με μήκος κύματος 66cm.

Η πρακτική εφαρμογή των πειραμάτων του Hertz προσέλκυσε το ενδιαφέρον πολλών. Το 1904 η πατέντα για έναν ανιχνευτή εμποδίων ως βοηθητική συσκευή πλοήγησης πλοίων δόθηκε στον Γερμανό μηχανικό Christian Hülsmeyer ο οποίος παρουσίασε την εφεύρεση του στο ναυτικό της τότε Γερμανίας, δεν υπήρχε όμως ιδιαίτερο ενδιαφέρον ούτε στρατιωτικής ούτε εμπορικής φύσης για την εφεύρεση μέχρι τις αρχές του 1930 όταν αναπτύχθηκαν βομβαρδιστικά μεγάλης εμβέλειας. Αυτή ήταν η αφορμή για την έναρξη της έρευνας στην τεχνολογία του radar.

Η πρώτη παρατήρηση της δράσης του radar έγινε στο Ναυτικό Ερευνητικό Εργαστήριο των Ηνωμένων Πολιτειών της Αμερικής (U.S. Naval Research Laboratory) στην Washington το 1922. Οι ερευνητές τοποθέτησαν έναν πομπό ραδιοκυμάτων στην μια όχθη του ποταμού Potomac ενώ στην απέναντι όχθη ήταν τοποθετημένος ένας δέκτης. Το πείραμα στέφθηκε με επιτυχία αλλά το αμερικάνικο ναυτικό δεν προτάθηκε να χρηματοδοτήσει το πρόγραμμα. Αργότερα το 1930 στο ίδιο εργαστήριο ο L.A. Hyland παρατήρησε πως όταν ένα αεροσκάφος περνάει μέσα από ακτίνα μεταδιδόμενης κεραίας υπήρχαν διακυμάνσεις στην πλευρά του δέκτη, και πάλι όμως το ενδιαφέρον από τις αμερικάνικες αρχές για χρήση μιας τέτοιας τεχνολογίας ήταν περιορισμένο. Μόνο όταν ανακαλύφθηκε η ο τρόπος χρήσης μιας μοναδικής κεραίας για εκπομπή και λήψη ταυτόχρονα φάνηκε η αξία μιας τέτοιας τεχνολογίας, ένα σύστημα που θα μπορούσε να ανιχνεύσει κάθε είδους όχημα στον αέρα, στην ξηρά και στη θάλασσα. Παρόμοιου τύπου έρευνες διεξάγονταν στην Βρετανία, την Γερμανία και την Σοβιετική Ένωση. Μέχρι τα τέλη της δεκαετίας του 1930 είχαν όλες αναπτύξει κάποιου είδους radar.

Η εξέλιξη του radar συνεχίστηκε και μετά τον πόλεμο με αξιοσημείωτα χρονικά σημεία το 1970 που βελτιώθηκε σε μεγάλο βαθμό η ακρίβειά του κάνοντας χρήση του φαινομένου Doppler, έτσι οι εφαρμογές του radar διευρύνθηκαν και σε άλλους τομείς πλοήγησης στην αεροπορία, την ναυσιπλοΐα και ειδικά στο καινούριο για τότε εγχείρημα, την εξερεύνηση του διαστήματος.

Skolnik, M.I. (2016)

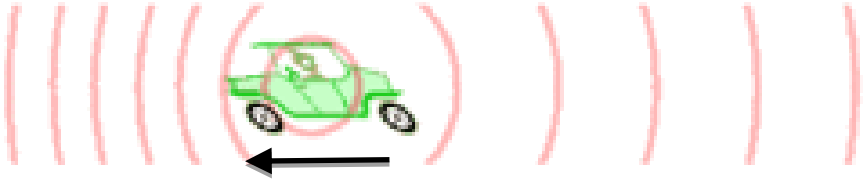
2.6 G.P.S. Global Positioning System

2.6.1 Βασική Δομή και Λειτουργία του G.P.S.

Το G.P.S.απαρτίζεται από 24 δορυφόρους διατεταγμένους σε σταθερές τροχιές γύρω από τη Γη οι οποίοι μπορούν να υπολογίσουν την θέση του χρήστη σε ένα πεδίο τριών διαστάσεων. Η πλοήγηση και εύρεση θέσης του χρήστη γίνεται δυνατή από τον υπολογισμό της απόστασης από τη θέση του χρήστη μέχρι τους ορατούς δορυφόρους που βρίσκονται σε τροχιά. Υπολογίζοντας την απόσταση του χρήστη από τους δορυφόρους είναι δυνατόν να προσδιοριστούν τριών ειδών συντεταγμένες της θέσης του γεωγραφικό μήκος, γεωγραφικό πλάτος και υψόμετρο, όπως και χρόνος G.P.S. σε σχέση με το ρολόι του δορυφόρου.

2.6.2 Η Ιστορία και η εξέλιξη του G.P.S.

Το Global Positioning System αναπτύχθηκε αρχικά από το υπουργείο άμυνας της Αμερικής με στόχο την ενίσχυση της ακρίβειας οπλικών συστημάτων και την μετρίαση της ραγδαίας αύξησης πλοηγικών συστημάτων στον στρατό, η αφορμή όμως με την οποία ξεκίνησε η έρευνα στο G.P.S. είναι χαρακτηριστικά αξιοσημείωτη. Όλα ξεκίνησαν με την εκτόξευση του σοβιετικού δορυφόρου Sputnik το 1957. Ότι στην αρχή έμοιαζε με σημαντική ήττα των Αμερικανών στον Ψυχρό Πόλεμο αποδείχτηκε τελικά καταλυτικό για μια από τις σημαντικότερες τεχνολογίες του 20^{ου} αιώνα που θα άλλαζε για πάντα την έννοια της πλοήγησης. Τον Οκτώβρη του 1957 λοιπόν, μια ομάδα επιστημόνων από το MIT παρατήρησε πως η συχνότητα των ραδιοκυμάτων που εκπεμπόταν από το σοβιετικό δορυφόρο αυξάνονταν όσο πλησίαζε και μειώνονταν όταν απομακρυνόταν. Αυτό ήταν το αποτέλεσμα του φαινομένου Doppler το οποίο ορίζεται ως η αλλαγή στην συχνότητα ή το μήκος ενός κύματος που επισημαίνεται από έναν παρατηρητή που κινείται σχετικά με την πηγή του κύματος, ένα χαρακτηριστικό παράδειγμα του φαινομένου Doppler είναι η αλλαγή στον τόνο μιας σειρήνας ή κόρνας ενός οχήματος όταν πλησιάζει, όταν περνά και τέλος όταν απομακρύνεται από έναν σταθερό παρατηρητή.



Εκείνη λοιπόν η παρατήρηση έδωσε την ιδέα στους επιστήμονες ότι οι δορυφόροι μπορούν να ανιχνεύονται από το έδαφος μετρώντας την συχνότητα των ραδιοκυμάτων που εξέπεμπαν και αντίστροφα, οι θέση των δεκτών στο έδαφος μπορεί να ανιχνεύεται από τους δορυφόρους. Έτσι λοιπόν στις αρχές του 1960 το υπουργείο άμυνας της Αμερικής άρχισε την ανάπτυξη του προκατόχου του G.P.S. με σκοπό δημιουργίας ενός παγκοσμίου, συνεχώς διαθέσιμου, υψηλής ακρίβειας συστήματος πλοήγησης και εύρεσης θέσης. Το αμερικάνικο ναυτικό χρηματοδότησε δύο προγράμματα. Το TRANSIT και το TIMATION.

Το TRANSIT έγινε το πρώτο λειτουργικό σύστημα πλοήγησης και εύρεσης θέσεως βασισμένο σε δίκτυο δορυφόρων, αναπτύχθηκε από τον John Hopkins στο Εργαστήριο Εφαρμοσμένης Φυσικής (Applied Physics Laboratory) υπό την αιγίδα του Richard Kirschner. Το σύστημα αποτελούταν από εφτά χαμηλού υψόμετρου πολικής τροχιάς δορυφόρους που εξέπεμπαν σταθερά ραδιοκύματα. Συγκεκριμένοι σταθμοί ελέγχου ανίχνευαν τους δορυφόρους από το έδαφος και ενημέρωναν συνεχώς τις παραμέτρους τροχιάς των δορυφόρων. Οι χρήστες του TRANSIT έβρισκαν την θέση τους μετρώντας τη μεταβολή Doppler (Doppler Shift) των σημάτων που εξέπεμπαν οι δορυφόροι. Το TRANSIT ήταν αρχικά σχεδιασμένο για την ανίχνευση βαλλιστικών πυραύλων σε υποβρύχια και άλλα πλοία στη θάλασσα, έγινε όμως διαθέσιμο και στο κοινό για προσωπική χρήση το 1967. Το σύστημα όμως αυτό είχε σοβαρές ελλείψεις, ήταν αργό με αποτέλεσμα να απαιτεί μεγάλο χρόνο ανίχνευσης, διέθετε δυνατότητα εύρεσης θέσης μόνο δύο διαστάσεων, που σημαίνει μόνο γεωγραφικό πλάτος και μήκος, είχε περιορισμένη κάλυψη λόγω της διακεκομμένης διαθεσιμότητας των σημάτων, και οι χρήστες έπρεπε να διορθώσουν το στίγμα τους ανάλογα με την ταχύτητα τους κάτι που δεν ήταν καθόλου πρακτικό για την ανίχνευση αεροσκαφών και άλλων γρήγορα κινούμενων χρηστών.

Το TIMATION ήταν ο δεύτερος προκάτοχος του G.P.S.. Ήταν ένα σύστημα πλοήγησης βασισμένο στο διάστημα το οποίο αναπτυσσόταν από το αμερικάνικο ναυτικό από το 1964.Το πρόγραμμα ενσωμάτωσε δύο πειραματικούς δορυφόρους που χρησιμοποιήθηκαν την ανάπτυξη ρολογιών μεγάλης σταθερότητας, μεταφοράς χρόνου, πλοήγηση σε πεδίο δύο διαστάσεων και επίδειξη της τεχνολογίας για πλοήγηση σε πεδίο τριών διαστάσεων. Ο πρώτος TIMATION δορυφόρος εκτοξεύθηκε το 1967 μεταφέροντας ταλαντωτές υψηλής σταθερότητας από κρυστάλλους quartz. Τα ατομικά ρολόγια είχαν καλύτερη σταθερότητα συχνότητας από προηγούμενα ρολόγια κάτι που βελτίωσε σε μεγάλο βαθμό την πρόβλεψη τροχιάς των δορυφόρων που ονομάστηκε ephemerides και αργότερα θα επέκτεινε το περιθώριο χρόνου που απαιτούταν ανάμεσα στις ενημερώσεις από σταθμούς ελέγχου στο έδαφος και τους δορυφόρους.

Την ίδια περίοδο η αμερικάνικη αεροπορία δούλευε σε δικό της πρόγραμμα μιας παρόμοιας τεχνολογίας που ονομάστηκε System 621B και παρείχε για πρώτη φορά πλοήγηση σε πεδίο τριών διαστάσεων, (γεωγραφικό πλάτος, γεωγραφικό μήκος και υψόμετρο) με υπηρεσία συνεχούς κάλυψης.

Υπήρχε αρκετή σύγχυση για το ποιος θα ηγηθεί του συστήματος καθώς και οι τρεις δυνάμεις του αμερικάνικου στρατού αεροπορία ναυτικό και χερσαίος στρατός ανέπτυσσαν ανεξάρτητα το δικό τους πρόγραμμα. Για να συντονιστεί η προσπάθεια το υπουργείο αμύνης καθιέρωσε μια επιτροπή και από τις τρείς μεριές αεροπορίας ναυτικού και χερσαίου στρατού το 1968 η οποία ονομάστηκε NAVSEG (Navigation Satellite Executive Group) και πέρασε τα επόμενα χρόνια προσπαθώντας να αποφασίσει σχετικά με τις ειδικές προδιαγραφές του προγράμματος πλοήγησης. Η λύση δόθηκε τον Απρίλιο του 1973 από τον αναπληρωτή γραμματέα αμύνης ο οποίος όρισε την αεροπορία ως κύριο ηγετικό μέλος του προγράμματος. Έτσι ήρθε στο φώς το πρώτο ενιαίο σύστημα παίρνοντας στοιχεία και από τα τρία προγράμματα, του ναυτικού (TIMATION), της αεροπορίας (System 621B), και του

χερσαίου στρατού (SECOR, Sequential Correlation of Range) το ενιαίο αυτό σύστημα ονομάστηκε DNSS (Defense Navigation Satellite System). Το σύστημα σχεδιάστηκε έτσι ώστε να πάρει τα καλύτερα στοιχεία και από τα τρία προγράμματα, έτσι ενσωμάτωσε τη δομή των σημάτων και συχνοτήτων από το System 621B της αεροπορίας, της προτεινόμενες τροχιές βασισμένες στο TIMATION του ναυτικού αλλά σε μεγαλύτερο υψόμετρο δίνοντας περίοδο 12 ωρών αντί για 8 που ήταν αρχικά σχεδιασμένο το TIMATION. Ενώ και τα δύο συστήματα είχαν προτείνει την χρήση ατομικών ρολογιών στους δορυφόρους μόνο το ναυτικό είχε δοκιμάσει πρακτικά την ιδέα.

Scott, P., Frost, G.P., Lachow, I., Frelinger, D.F., Fossum, D., Pnto, D.W., Pinto, M.M. (1995)

2.6.3 Το Χρονικό του G.P.S.

- Τον Δεκέμβρη του 1973 το υπουργείο αμύνης της Αμερικής έδωσε την επίσημη άδεια για να προχωρήσει η πρώτη από τις τρείς φάσεις της ανάπτυξης του προγράμματος NAVSTAR G.P.S.

- Την περίοδο 1978-1985 εκτοξεύθηκαν 11 ακόμη δοκιμαστικοί δορυφόροι με σκοπό τη δοκιμή του συστήματος NAVSTAR που αποκαλούταν πλέον απλά “G.P.S.”. Οι δορυφόροι έφεραν ατομικά ρολόγια για ακριβής μέτρηση των χρόνων μετάδοσης, επίσης κάποιοι από αυτούς τους δορυφόρους έφεραν αισθητήρες για την ανίχνευση εκτόξευσης και εκπυρσοκρότησης πυρηνικών όπλων.

- Το 1983 λίγο μετά την κατάρριψη Κορεάτικου αεροσκάφους που είχε ξεφύγει εκτός πορείας πάνω από τον εναέριο χώρο της Ρωσίας από την τότε Σοβιετική Ένωση, ο πρόεδρος Reagan της Αμερικής καθιέρωσε την ελεύθερη χρήση του G.P.S. για όλα τα επιβατικά και εμπορικά αεροσκάφη.

- Το 1985 η αμερικάνικη κυβέρνηση κάνει σύμβαση με ιδιωτικές εταιρείες για την ανάπτυξη φορητών δεκτών G.P.S. Μετά από χρόνια δοκιμών το 1989 η αμερικάνικη αεροπορία εκτόξευσε τον πρώτο πλήρως λειτουργικό δορυφόρο στο διάστημα. Την ίδια χρονιά η Magellan Corporation διεκδικεί τον τίτλο της πρώτης εταιρείας που προσφέρει στο καταναλωτικό κοινό φορητή συσκευή πλοήγησης στηριζόμενη στην τεχνολογία G.P.S.

- Το 1990 το υπουργείο άμυνας της Αμερικής φοβούμενο για το ενδεχόμενο στρατιωτικής χρήσης του G.P.S. από εχθρικές δυνάμεις μειώνει την ακρίβεια του συστήματος.

- Το 1995 το πρώτο δίκτυο 27 πλήρως λειτουργικών δορυφόρων G.P.S. ολοκληρώνεται. Από τους 27 δορυφόρους 3 θα λειτουργήσουν σαν εφεδρικοί για την γρήγορη αντικατάσταση κάποιου ενεργού δορυφόρου του δικτύου των 24 σε περίπτωση βλάβης.

- Το 1998 ο αντιπρόεδρος της Αμερικής Al Gore ανακοίνωσε το σχέδιο να κάνει τους δορυφόρους G.P.S. να εκπέμπουν δύο επιπρόσθετα σήματα για ελεύθερη μη στρατιωτική χρήση, συγκεκριμένα για την βελτίωση της ασφάλειας εμπορικών αεροσκαφών.

- Το 1999 η κατασκευαστική εταιρεία κινητών τηλεφώνων Benefon λάνσαρε για πρώτη φορά τηλέφωνο με ενσωματωμένο δέκτη G.P.S. το "Benefon Esc!"

- Το 2000 το υπουργείο άμυνας της Αμερικής καταργεί την επιτηδευμένη μείωση της ακρίβειας του συστήματος. Το G.P.S. έγινε 10 φορές πιο ακριβές και σύντομα όλο και περισσότερες βιομηχανίες άρχισαν να το χρησιμοποιούν.

- Το 2001 καθώς οι G.P.S. δέκτες γίνονταν ολοένα φθηνότεροι και μικρότεροι, ιδιωτικές εταιρίες άρχισαν να λανσάρουν πληθώρα προϊόντων G.P.S. όπως φορητούς πλοηγούς G.P.S. για το αυτοκίνητο και άλλα.

- Το 2004 η εταιρεία Qualcomm ανακοίνωσε ότι είχε αναπτύξει και δοκιμάσει τεχνολογία «υποβοηθούμενου G.P.S.» δίνοντας την δυνατότητα στους χρήστες να χρησιμοποιήσουν το σήμα του κινητού τηλεφώνου μαζί με το σήμα G.P.S. για να προσδιορίσουν την θέση τους με ακρίβεια μόλις μερικών μέτρων.

- Το 2005 ο πρώτος από τη νέα γενιά G.P.S. δορυφόρων εκτοξεύτηκε από το ακρωτήριο Canaveral. Η νέα γενιά δορυφόρων θα εξέπεμπε ένα δεύτερο σήμα αφιερωμένο σε ελεύθερη χρήση για το κοινό.

- Το 2009 το Government Accountability Office έβγαλε αναφορά προειδοποιώντας πως η νέα αναβάθμιση του συστήματος με κόστος 5,8 δισεκατομμυρίων δολαρίων είχε τόσα τεχνικά προβλήματα που υπήρχε κίνδυνος το 2010 κάποιοι από τους δορυφόρους να παρουσιάσουν ανεπανόρθωτες βλάβες με αποτέλεσμα να βγουν εκτός λειτουργίας.

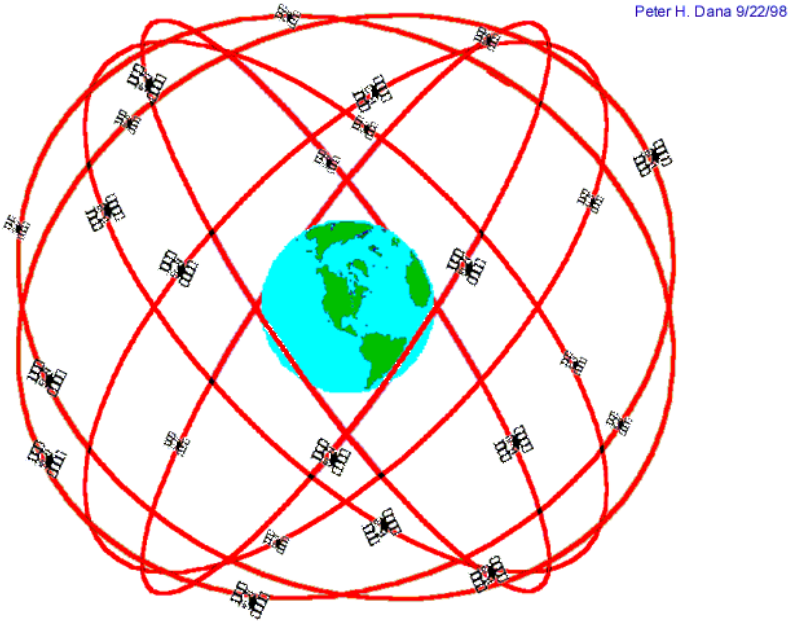
- Το 2010-2011 η αμερικάνικη αεροπορία εκτόξευσε δύο G.P.S. δορυφόρους με κύριο ρόλο να κρατήσουν το σύστημα λειτουργικό μέχρι την επόμενη γενιά των Block III δορυφόρων η εκτόξευση των οποίον θα άρχιζε το 2014. Η νέα γενιά θα ενίσχυε το σήμα για την ελεύθερη χρήση στο κοινό ενώ παράλληλα θα βελτίωνε την επίδοση του υπάρχοντος συστήματος.

- Το 2012 η αμερικάνικη αεροπορία διαθέτει δίκτυο από 31 λειτουργικούς G.P.S. δορυφόρων συν 3 επιπλέον αποσυρμένους που υπάρχει δυνατότητα να ενεργοποιηθούν αν παραστεί ανάγκη.

Sullivan, M. (2012)

2.6.4 Η Λειτουργία του NAVSTAR

Η τεχνολογία G.P.S. Global Positioning System αποτελεί ένα από τα σημαντικότερα εργαλεία πλοήγησης του σύγχρονου κόσμου, αποτελείται από τρία κεντρικά κομμάτια, τον έλεγχο, το διάστημα, και τους χρήστες. Το κομμάτι του διαστήματος αποτελείται από 24 δορυφόρους NAVSTAR σε 6 πεδία τροχιάς. Οι δορυφόροι περιφέρονται γύρω από τη Γη με περίοδο 12 ωρών σε ύψος 20.200 χιλιομέτρων από την επιφάνεια της Γης με κλίση 55 μοιρών σε σχέση με τον ισημερινό. Κάθε δορυφόρος περνάει από το ίδιο σημείο της Γης περίπου μια φορά την ημέρα. Οι δορυφόροι είναι έτσι διατεταγμένοι ώστε η απόσταση μεταξύ τους να είναι τέτοια που να αφήνει ένα ελάχιστο 5 δορυφόρων σε εμβέλεια οποιουδήποτε μέρους της γης.



Το σύστημα G.P.S. λειτουργεί μετρώντας το χρόνο που χρειάζεται ένα κωδικοποιημένο σήμα να φτάσει στην Γη από τους δορυφόρους. Ο δέκτης το επιτυγχάνει αυτό παράγοντας μια σειρά από κωδικοποιημένα σήματα ίδια με αυτά που εξέπεμψε ο δορυφόρος, έπειτα υπολογίζει την χρονοκαθυστέρηση ανάμεσα στα δικά του σήματα και τα σήματα που έλαβε από τον δορυφόρο, υπολογίζοντας την απόκλιση που πρέπει να εισάγει στα δικά του σήματα για να ταιριάξουν με αυτά του δορυφόρου. Ένας δέκτης G.P.S. μπορεί θεωρητικά να υπολογίσει την θέση του σε χώρο τριών διαστάσεων μετρώντας την απόστασή του από τρεις διαφορετικούς δορυφόρους, στην πράξη όμως χρειάζεται ένας επιπλέον δορυφόρος για να καλύψει την χρονική απόκλιση ανάμεσα στα ρολόγια του δέκτη και αυτά των δορυφόρων. Η τέταρτη μέτρηση δίνει την δυνατότητα στον υπολογιστή του δέκτη να

επιλύσει προς την χρονική απόκλιση και να την αφαιρέσει από την τελική λύση. Ο υπολογισμός της ταχύτητας του χρήστη γίνεται παίρνοντας τον ρυθμό μεταβολής μετρήσεων ψευδό-εμβέλειας (pseudo-range measurements) ανά το χρόνο. Αυτές οι μετρήσεις ψευδό-εμβέλειας εκτελούνται σημειώνοντας την διαφορά φάσης στις μετρήσεις μέσα σε ένα συγκεκριμένο χρονικό διάστημα (η οποία είναι η μέση συχνότητα Doppler).

Οι δορυφόροι G.P.S. εκπέμπουν δύο τύπους σημάτων: Το χαμηλής ακρίβειας C/A-Code (Coarse Acquisition Code) που αποτελεί την συμβατική υπηρεσία εύρεσης θέσης που ονομάζεται «Standard Positioning Service (SPS)» το οποίο διατίθεται παγκοσμίως ελεύθερα σε όλους τους χρήστες δωρεάν, και το υψηλής ακρίβειας P-Code (Precision Code) το οποίο αποτελεί την υπηρεσία εύρεσης ακριβούς θέσεως που ονομάζεται Precise Positioning Service (PPS) και είναι διαθέσιμη αποκλειστικά σε στρατιωτικές δυνάμεις ομοσπονδιακές υπηρεσίες και άλλες κρατικές υπηρεσίες που ελέγχονται άμεσα από την Αμερικάνικη κυβέρνηση. Τα χαμηλής ακρίβειας σήματα χρησιμοποιούν την συχνότητα L1 στα 1575,42MHz ενώ τα υψηλής ακρίβειας χρησιμοποιούν μαζί με την συχνότητα L1 και την συχνότητα L2 στα 1227,60MHz.

Δύο παράγοντες επηρεάζουν την ακρίβεια του συστήματος λάθη μέσα στα ίδια τα G.P.S. σήματα, και η γεωμετρία των τεσσάρων NAVSTAR δορυφόρων που χρησιμοποιούνται για τον προσδιορισμό θέσης. Ο πρώτος παράγοντας περιλαμβάνει λάθη στο ίδιο το G.P.S. σήμα τα οποία μπορούν να προκληθούν από σφάλματα στον υπολογισμό χρόνου στο ρολόι του δορυφόρου, διαφορά στην αναμενόμενη και την πραγματική θέση του δορυφόρου, ατμοσφαιρικές καθυστερήσεις, φαινόμενο multipath fading και θόρυβο στον δέκτη. Ο δεύτερος παράγοντας, η γεωμετρία των δορυφόρων είναι σημαντική επειδή ο G.P.S. δέκτης καθορίζει την θέση του χρήστη βάσει τριγωνισμού (triangulation), οπότε όσο πιο μακριά είναι οι 4 δορυφόροι μεταξύ τους τόσο καλύτερη ακρίβεια θα έχει ο δέκτης.

Το κομμάτι του ελέγχου ανιχνεύει τους δορυφόρους και τους παρέχει περιοδικές ενημερώσεις διορθώνοντας τις αποκλίσεις στα ρολόγια τους και τις σταθερές ephemeris που ελέγχουν την τροχιά του δορυφόρου. Οι τοποθεσίες των σταθμών ελέγχου είναι γνωστές με υψηλό βαθμό ακρίβειας από τους δορυφόρους και ο κάθε σταθμός είναι εξοπλισμένος με το δικό του ατομικό ρολόι. Κάθε σήμα δορυφόρου G.P.S. διαβάζεται από 4 ή 5 σταθμούς ελέγχου, Επειδή οι συντεταγμένες θέσης και χρόνου των σταθμών είναι γνωστές από τους δορυφόρους οι μετρήσεις ψευδό-εμβέλειας που γίνονται από τον κάθε σταθμό ελέγχου μπορούν να συνδυαστούν για να δημιουργήσουν ένα αντίστροφο αποτέλεσμα θέσης το οποίο μπορούν να χρησιμοποιήσουν για να διορθώσουν τα σφάλματα χρόνου και θέσης των δορυφόρων.

Το κομμάτι του χρήστη αποτελείται από τον G.P.S. δέκτη και τον βοηθητικό εξοπλισμό όπως κεραίες. Εδώ θα περιγραφεί η λειτουργία του δέκτη και θα εξεταστεί ο βρόγχοι ανίχνευσης σήματος. Ο βρόγχος ανίχνευσης αποτελεί έναν μηχανισμό που δίνει τη δυνατότητα στον δέκτη να ανιχνεύει ένα σήμα που αλλάζει συχνότητα ή χρόνο. Είναι μία συσκευή ανατροφοδότησης που συγκρίνει ένα εισερχόμενο σήμα με ένα τοπικό (ένα εξωτερικό σήμα με ένα που έχει φτιαχτεί εσωτερικά), έπειτα παράγει ένα σήμα σφάλματος το οποίο είναι η διαφορά μεταξύ των δύο σημάτων, και χρησιμοποιεί αυτό το σήμα για να προσαρμόσει το εσωτερικό σήμα ώστε να ταιριάξει με το εξωτερικό έτσι ώστε το σφάλμα να μηδενιστεί ή ελαχιστοποιηθεί σε ικανοποιητικό βαθμό.

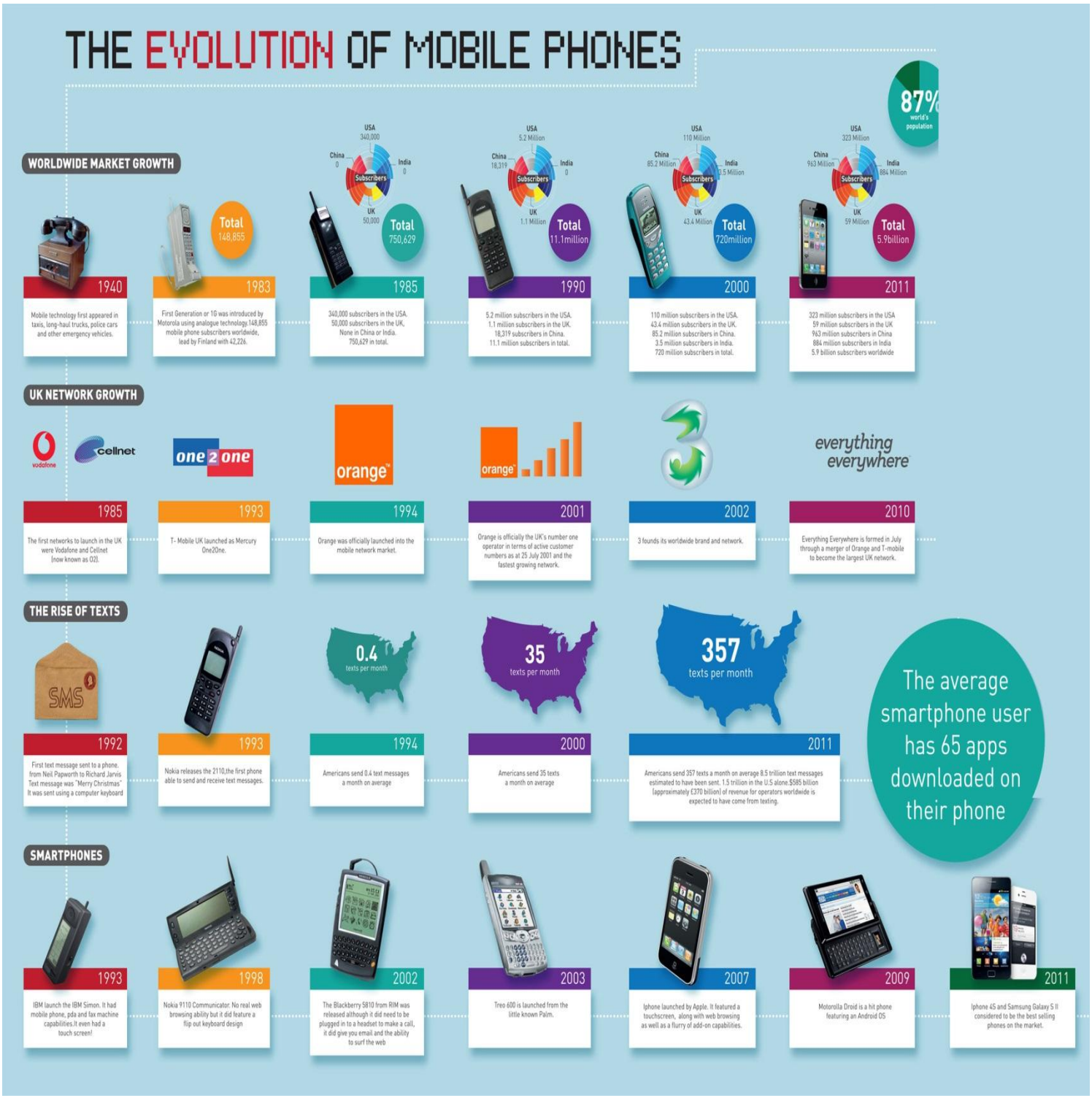
Mai, T. (2012), Scott, P., Frost, G.P., Lachow, I., Frelinger, D.F., Fossum, D., Pnto, D.W., Pinto, M.M. (1995)

3. Τεχνολογική Στάθμιση

3.1 Κινητές Συσκευές

Οι κινητές συσκευές αποτελούν μια τεχνολογία που έχει ενσωματωθεί εδώ και αρκετά χρόνια στον πυρήνα την καθημερινής μας ζωής. Είναι προφανές ότι η ραγδαία εξέλιξη της συγκεκριμένης τεχνολογίας έχει επιφέρει σημαντικές αλλαγές στην προσωπική, επαγγελματική και κοινωνική ζωή του καθενός. Ποιο θα είναι το αποτέλεσμα μιας τέτοιας τάσης στην κοινωνική αλλά και βιολογική ανάπτυξη του κόσμου, κανείς δεν μπορεί να ξέρει. Στο παρακάτω κεφάλαιο θα αναφερθεί η εξέλιξη της τεχνολογίας των κινητών συσκευών και οι σημαντικές καινοτομίες που την έφεραν στο επίπεδο που βρίσκεται σήμερα εξετάζοντας την ανάπτυξη και του hardware αλλά και του software ανά τα χρόνια.

Στην παρακάτω εικόνα φαίνεται ένα χρονικό από τα πιο αντιπροσωπευτικά μοντέλα, τεχνολογίες, και καινοτομίες της κάθε εποχής.



3.1.1 Η εξέλιξη των κινητών συσκευών

Η τεχνολογία των κινητών τηλεφώνων έχει περάσει πολλά στάδια εξέλιξης για να φτάσει στο στάδιο το οποίο βρίσκεται σήμερα. Τα στάδια αυτά στην βιομηχανία των κινητών τηλεφώνων ονομάζονται «γενιές» (Generations) ή απλά "G". Οι γενιές αναφέρονται στην ανάπτυξη της τεχνολογίας του υλικού και των δυνατοτήτων των δικτύων κυψέλης. Στο παρών κεφάλαιο θα αφοσιωθούμε κατά κύριο λόγο στην τεχνολογία υλικού που αναφέρεται κυρίως στην τεχνολογία υλικού των ίδιων των συσκευών ενώ οι γενιές περιλαμβάνουν την τεχνολογία της κινητής τηλεφωνίας σε ένα ευρύτερο πλαίσιο που συμπεριλαμβάνει τεχνολογία υλικού, τεχνολογία δικτύου και άλλους τομείς οι οποίοι δεν έχουν άμεση σχέση με το αντικείμενο της παρούσας εργασίας γι αυτό και θα παραληφθούν. Η τεχνολογία υλικού λοιπόν θα χωριστεί σε κατηγορίες ανάλογα με το χρονικό διάστημα στο οποίο παρουσιάστηκε.

Η πρώτη εποχή ήταν η εποχή του κινητού «τούβλου» 1973-1988 χαρακτηριζόμενου έτσι λόγω του σχήματός του. Τέτοιου τύπου συσκευές ήταν ογκώδης και άκομψες και καθόλου πρακτικές σε σχέση με τις σημερινές συσκευές που διατίθεται σε τελείως διαφορετικού είδους σχεδιασμό, για εκείνη την εποχή όμως αποτελούσε κάτι το καινοτόμο και σαφώς έγινε η βάση για την εξέλιξη του κινητού όπως το ξέρουμε σήμερα. Οι συσκευές αυτές απαιτούσαν ογκώδεις μπαταρίες γεγονός που αύξανε σημαντικά το βάρος της ίδιας της συσκευής. Ο όγκος της μπαταρίας ήταν απαραίτητα τόσο μεγάλος επειδή η συσκευή απαιτούσε αρκετή ενέργεια για να εκπέμψει σήμα αρκετά ισχυρό ώστε να φτάσει μέχρι τον κοντινότερο αναμεταδότη ο οποίος θα μπορούσε να βρίσκεται σε αρκετά μεγάλη απόσταση αφού το πλήθος τους εκείνα τα χρόνια ήταν ιδιαίτερα περιορισμένο. Στην παρακάτω εικόνα βλέπουμε ένα από τα πιο χαρακτηριστικά μοντέλα της εποχής το Motorola DynaTAC 8000X

Motorola DynaTAC 8000X
1983

Η επόμενη εποχή ανήκε στα κινητά «πλάκα» και αναφέρεται στο χρονικό διάστημα 1988-1998. Η συγκεκριμένη εποχή αποτέλεσε ένα από τα πιο καθοριστικά άλματα τεχνολογίας στις κινητές συσκευές. Τα κινητά πλέον είχαν μακρύ λεπτό ορθογώνιο σχήμα, ένα μοντέλο σχεδιασμού που λίγο ως πολύ έχει επικρατήσει μέχρι και σήμερα. Σε αυτό το σημείο οι πάροχοι δικτύου ξεκίνησαν να βλέπουν την καθαρή αξία και το κέρδος της τεχνολογίας κεφαλωτού δικτύου έτσι ξεκίνησε μια παγκόσμια στροφή προς την αυξανόμενη χρήση κινητών συσκευών. Το δίκτυο εξελίχθηκε στην δεύτερη γενιά (2G) τεχνολογίας και η ζήτηση για κινητές συσκευές οι οποίες πλέον μπορούσαν να χωρέσουν στην τσέπη του καθενός αυξήθηκε δραματικά. Η εποχή όμως ήταν καινοτόμα και σε άλλες καθοριστικές λειτουργίες του τηλεφώνου. Για πρώτη φορά εισάχθηκε η ιδέα του κινητού όχι μόνο σαν τηλέφωνο αλλά σαν συσκευή ικανή για πρόσθετες λειτουργίες. Αρχικά η ιδέα πίσω από το SMS ήταν για μια υπηρεσία που να μπορεί να στέλνει γραπτή ειδοποίηση για την λήψη νέου μηνύματος τηλεφωνητή ή και άλλες μικρού μεγέθους ειδοποιήσεις από τον πάροχο στον πελάτη. Στις αρχές του 1990 όμως, λόγω ορισμένων παραβλέψεων η υπηρεσία αυτή δεν χρεωνόταν στους πελάτες. Εξοικειωμένοι χρήστες λοιπόν στην Ευρώπη ανακάλυψαν ότι μπορούσαν να στείλουν μηνύματα χωρίς καμία χρέωση αντί να κάνουν κανονική κλήση το κόστος της οποίας για εκείνα τα χρόνια ήταν αρκετά υψηλό, έτσι λοιπόν το μήνυμα 140 χαρακτήρων που χρησιμοποιούμε ακόμη και σήμερα γεννήθηκε. Η δυνατότητα αποστολής αλλά και λήψης γραπτών μηνυμάτων με την υπηρεσία SMS (Short Message Service) άνοιξε νέους ορίζοντες για την εξέλιξη των κινητών τηλεφώνων. Στην παρακάτω εικόνα φαίνεται ένα χαρακτηριστικό μοντέλο κινητού τηλεφώνου της συγκεκριμένης εποχής το Nokia 1011.

Nokia 1011
1993

Η τρίτη εποχή αποτελούταν από κινητά τα οποία είχαν πλέον πολλαπλές λειτουργίες. Μέχρι στιγμής όλα τα κινητά περιλάμβαναν κλήσεις, SMS, και το χαρακτηριστικό παιχνίδι «φιδάκι» που σημάδεψε τα κινητά της εποχής. Η εποχή του κινητού πολλαπλών χρήσεων όμως διεύρυνε τα όρια και τις χρήσεις του κινητού με πληθώρα νέων εφαρμογών όπως προγράμματα αναπαραγωγής μουσικής, λήψης και επεξεργασίας φωτογραφιών από κάμερα η οποία ήταν πλέον ενσωματωμένη στην ίδια τη συσκευή, πρόσβαση στο διαδίκτυο και τόσες άλλες υπηρεσίες που σήμαναν την αρχή μιας νέας εποχής που η χρήση του κινητού δεν περιοριζόταν μόνο στην επικοινωνία. Υπήρχαν όμως και τα αρνητικά στοιχεία, λόγω περιορισμένης υπολογιστικής ισχύς από πλευρά υλικού, κακού σχεδιασμού ή μίξη των δύο. Πολλές από τις προσφερόμενες εφαρμογές ήταν αργές και δύσχρηστες, πράγμα που περιόριζε σε μεγάλο βαθμό τη χρήση τους. Η πρόσβαση στο διαδίκτυο για παράδειγμα, θεωρητικά ήταν πλέον δυνατή, αλλά λόγω των περιορισμών του υλικού της συσκευής η αναζήτηση στο διαδίκτυο γινόταν με πολύ αργούς ρυθμούς πράγμα που καθιστούσε μια τέτοια λειτουργία ιδιαίτερα δύσχρηστη με αποτέλεσμα πολύ λίγοι χρήστες να το χρησιμοποιούν κατά αυτόν το τρόπο. Η εποχή όμως έδωσε τις βάσεις για την ανάπτυξη εφαρμογών σε κινητές συσκευές που αργότερα θα εξελίσσονταν με ραγδαίο ρυθμό μέχρι το επίπεδο που βρίσκονται σήμερα. Παρακάτω απεικονίζεται ένα από τα πιο χαρακτηριστικά τηλέφωνα της εποχής το Motorola RAZR.

Motorola RAZR
2003

Η τέταρτη εποχή περιλαμβάνει τα smartphone υπό την ευρύτερη έννοια και συμπίπτει χρονολογικά για κάποιο διάστημα με την τρίτη εποχή, ξεκινάει από το 2002 μέχρι και το σήμερα. Δεν ήταν πότε απολύτως καθορισμένο ποια συσκευή μπορεί να χαρακτηριστεί ως smartphone και ποια δεν μπορεί, κάτι που εξηγεί το γεγονός ότι συμπίπτει με την τρίτη εποχή, πολλές συσκευές συνδύαζαν τα χαρακτηριστικά και των δύο εποχών. Ενώ τα smartphones έχουν τα ίδια χαρακτηριστικά με τα κινητά πολλαπλών λειτουργιών της τρίτης γενιάς (κλήσεις, SMS, web browsing, camera, applications κλπ) η διαφορές τους συνήθως παρατηρείται σε χαρακτηριστικά όπως συγκεκριμένο λειτουργικό σύστημα, μεγαλύτερη οθόνη, χρήση πληκτρολογίου QWERTY ή γραφίδας, πρόσβαση σε WiFi και άλλα χαρακτηριστικά με σκοπό να βελτιώσουν την ευχρηστία και την λειτουργικότητα της συσκευής κάτι που δεν κατάφερναν πάντα. Το smartphone αρχικά είχε περιορισμένη απήχηση στο καταναλωτικό κοινό λόγω πολλών παραγόντων μεταξύ των οποίων το αυξημένο κόστος, ο ανορθόδοξος σχεδιασμός κάποιων από αυτά τα μοντέλα όπως και ο διαφορετικός τρόπος χρήσης και πρόσβασης που ξέφευγε από τις συνηθισμένες γνώριμες μεθόδους προσβασιμότητας του κινητού. Στην παρακάτω εικόνα απεικονίζεται ένα από τα πιο αντιπροσωπευτικά smartphone της εποχής το Nokia e62

Nokia e62

Την αρχή της πέμπτης εποχής των τηλεφώνων αφής αποτελεί αναμφισβήτητα μια από τις σημαντικότερες αλλαγές στην ιστορία των κινητών τηλεφώνων, η αρχή της παρατηρείται στα μέσα του 2007 με την κυκλοφορία του πρώτου iPhone το οποίο εισήγαγε ριζοσπαστικές καινοτομίες στην τεχνολογία των κινητών τηλεφώνων αφής. Άλλοι υποστηρίζουν πως τα πρώτα βήματα έγιναν από την κυκλοφορία του LG PRADA το 2006, που διέθετε την πρώτη capacitive touchscreen. Η συνταρακτική επιτυχία της πέμπτης γενιάς κινητών τηλεφώνων όμως ήταν γεγονός αναμφισβήτητο από όλους. Κάπως έτσι λοιπόν σηματοδοτήθηκε η αρχή μιας γενιάς κινητών τηλεφώνων που επεκτείνεται μέχρι τις σημερινές μέρες.

LG PRADA
2006

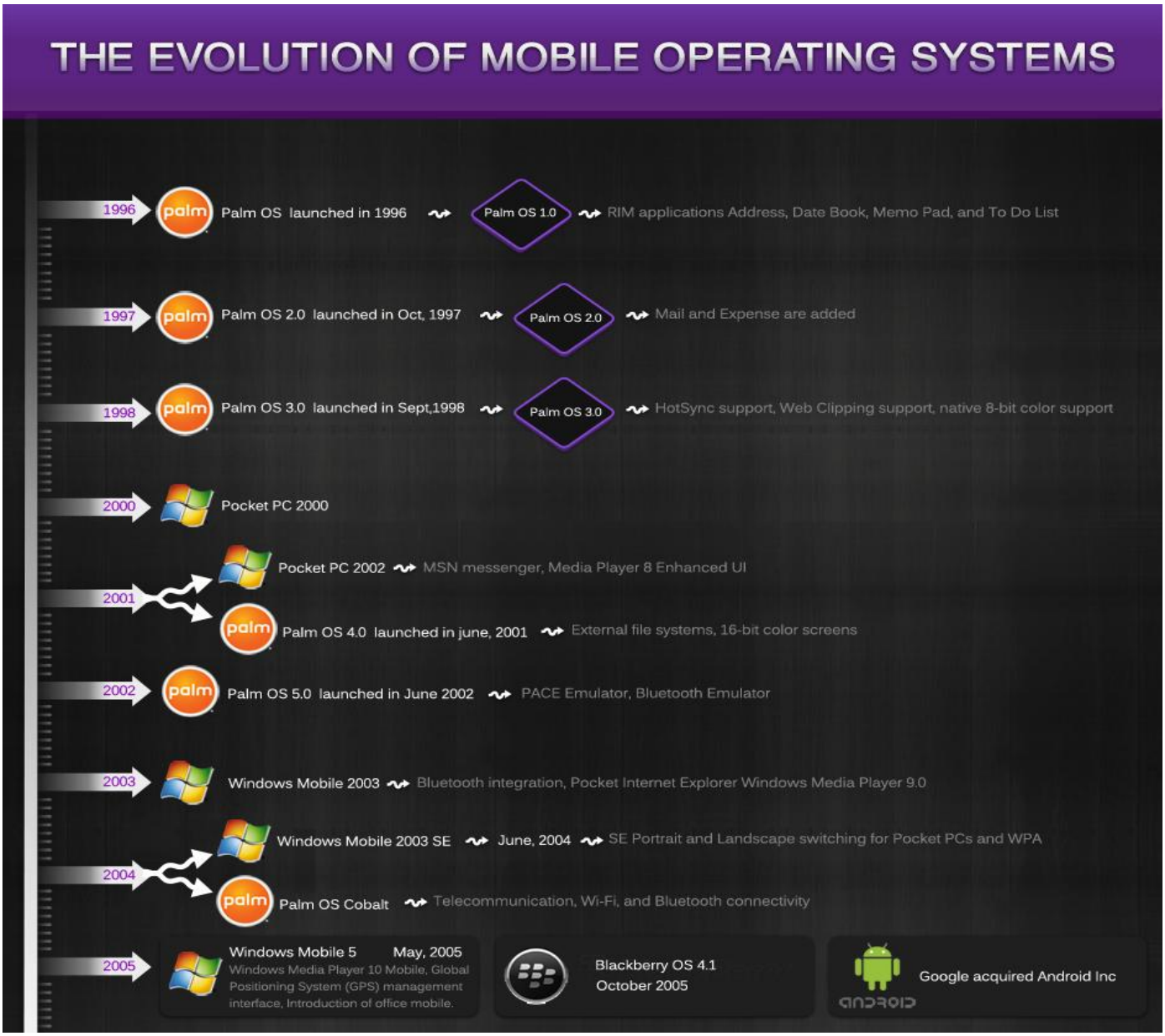
iPhone
2007

Fling, B. (2009)

3.2. Η εξέλιξη των λειτουργικών συστημάτων σε κινητές συσκευές

Η ιστορία των λειτουργικών συστημάτων κινητών συσκευών εκτείνεται από τα μέσα της δεκαετίας του 90 με την κυκλοφορία του πρώτου Palm OS μέχρι και το σήμερα με τα πολυσύνθετα Android και iOS. Στην παρακάτω εικόνα φαίνεται η εξέλιξη αυτών των λειτουργικών συστημάτων από την πρώτη κυκλοφορία του Palm OS το 1996 μέχρι και την κυκλοφορία του Android Honeycomb το 2011.

Chart by: https://www.xcubelabs.com/mobile-operating-systems/




2007
iPhone
The first version of iPhone was introduced on January 9, 2007
Windows Mobile 6
Feb, 2007
.NET Compact Framework v2 SP2 Microsoft SQL Server 2005 Compact Edition Windows Live
Blackberry OS 4.2
April 2007
Voice Notes option, Email and SMS in separate folders
2007
2008
iPhone
iPhone OS 2.0
June, 2008
Third-party application support, Sync Google contacts
Blackberry OS 4.5
Sep 2008
HTML e-mails, faster performance and improved multi-tasking, Microsoft Office Documents ToGo
Windows Mobile 6.1
April, 2008
Threaded SMS, full page zooming in Internet Explorer and 'Domain Enroll'
Android 1.0
symbian OS
Symbian^1
Desktop interactive widgets, FaceBook IM chat
2008
2009
Android 1.1
Feb, 2009
Support for saving attachments from MMS, marquee in layouts, API changes
Blackberry OS 5
Oct 2009
Wireless sync, BlackBerry Enterprise Server 5, almost revamped Web browser
Windows Mobile 6.5
May, 2009
Internet Explorer Mobile 6 and Multi touch support
Samsung Bada 1.0
Nov, 2009
Samsung announced the Bada platform on 10 November 2009
iPhone OS 3.0
June, 2009
Push notifications, cut, copy and paste, Turn-by-Turn navigation, Voice memos
HP Web OS
Synergy app, multi-touch gestures and multi tasking
Android 1.5 (Cupcake)
April, 2009
Bluetooth A2DP and AVRCP support, Uploading videos to YouTube and pictures to Picasa
Android 1.6 (Donut)
Sept, 2009
WVGA screen resolution support, Google free turn-by-turn navigation
Android 2.0 / 2.1 (Eclair)
Oct, 2009
HTML5 support, Microsoft Exchange Server, Bluetooth 2.1
2010
iPhone
iOS 4.0 June, 2010
Multi-Tasking, folders
Blackberry OS 6
August 2010
New media interface, stronger social media integration, Multiple contact lists, trackpad support for swipe gestures
Windows Phone 7
Oct, 2010
Tiled UI, loud-based services support, Multitasking
Android 2.2 (Froyo)
May, 2010
USB tethering and Wi-Fi hotspot functionality, Adobe Flash 10.1 support
Android 2.3 (GingerBread)
Dec, 2010
Multi-touch software keyboard, Support for extra-large screen sizes and resolutions
symbian OS
Symbian^2
Royalty-free version
Symbian^3 – Feb, 2010
Native WebKit based browser, 2D and 3D graphics architecture, UI improvements, and support for external displays through HDMI
2011
Android 3.0 (HoneyComb) Expected mid 2011
Optimized tablet support with a new user interface, Three dimensional desktop, video chat with Gtalk support
[x]cube LABS

3.3 Η πλατφόρμα Android

3.3.1 Η γέννηση του Android

Το 1998 στο Palo Alto της Καλιφόρνιας ιδρύθηκε μια μικρή εταιρεία, η Android Inc. από τους Andy Rubin, Rich Miner, Nick Sears, και Chris White. Ο Andy Rubin είχε πει χαρακτηριστικά σε μια συνέντευξή του «Υπάρχει μεγάλη προοπτική στην ανάπτυξη εξυπνότερων κινητών συσκευών που θα έχουν περισσότερη αντίληψη της θέσης και των προτιμήσεων του χρήστη. Όταν η άνθρωποι γίνονται έξυπνοι αυτή η πληροφορία αρχίζει να ενσωματώνεται μέσα στα ίδια τα καταναλωτικά προϊόντα». Κάποια χρόνια αργότερα τον Ιούλιο του 2005 η Android Inc. αγοράστηκε από την Google, δύο χρόνια μετά, το Νοέμβριο του 2007 η Open Handset Alliance σχηματίστηκε για να προωθήσει ένα ελεύθερο λειτουργικό σύστημα ανοιχτού κώδικα βασισμένο στο Linux, το οποίο θα ήταν προορισμένο για κινητές συσκευές, το Android. Η Open Handset Alliance αποτελεί μια συμμαχία από δεκάδες εταιρείες μερικές από τις οποίες είναι οι Intel, Motorola, NVIDIA, Texas Instruments, LG, Samsung Sprint Nextel και T-Mobile. Το πρώτο τηλέφωνο που κυκλοφόρησε με το νέο αυτό λειτουργικό σύστημα ήταν το T-Mobile G1 τον Οκτώβριο του 2008.

3.3.2 Τι είναι το Android

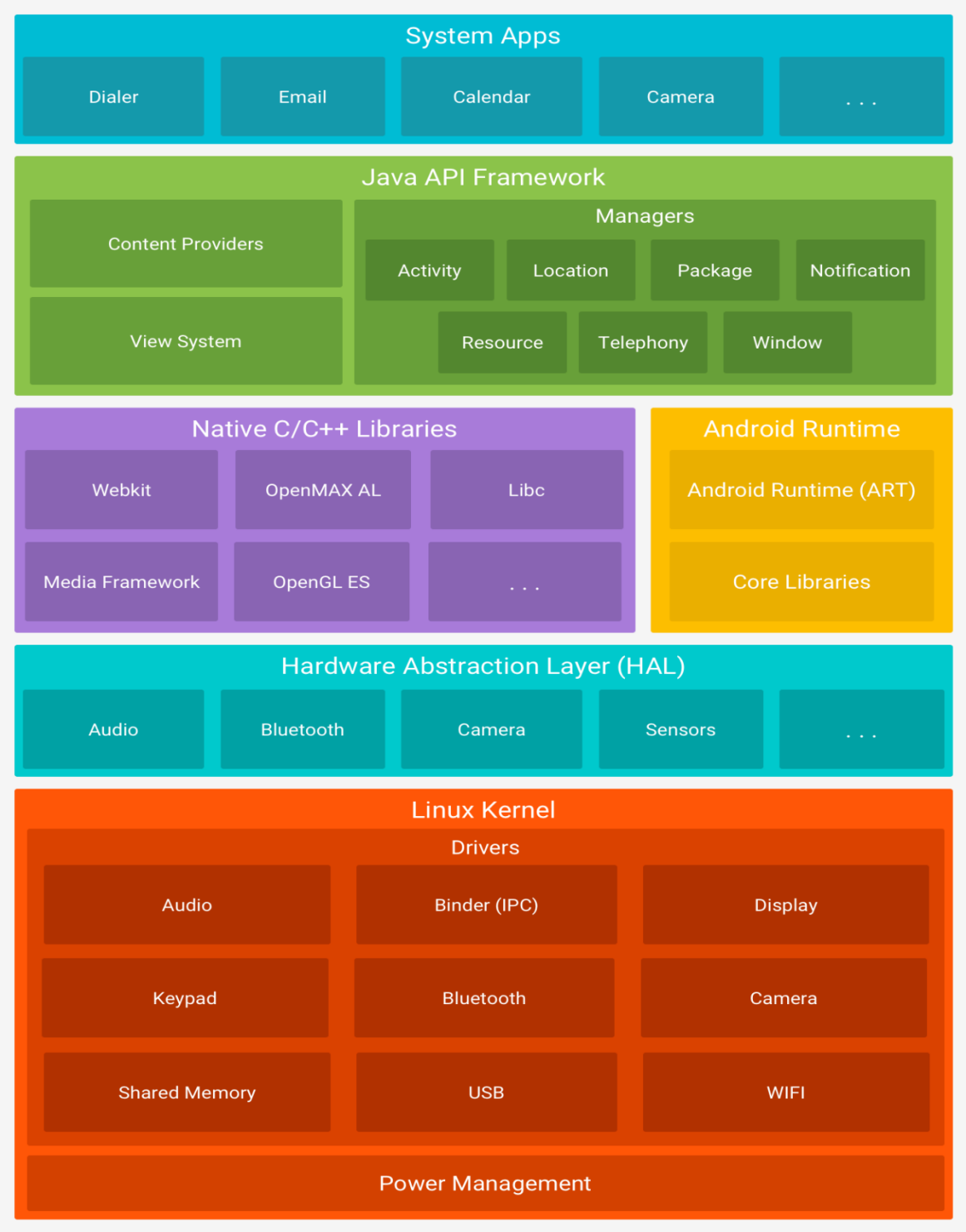
Το Android αποτελεί μια στοίβα λογισμικού για κινητές συσκευές που περιλαμβάνει λειτουργικό σύστημα, middleware και key applications. Είναι μια πλήρης πλατφόρμα ανοιχτού κώδικα σχεδιασμένη για κινητές συσκευές η οποία είναι δωρεάν. Κινητά τηλέφωνα βασισμένα στο Android απαιτούν ασύρματα δίκτυα τρίτης γενιάς (3G) για να μπορούν να χρησιμοποιήσουν όλες τις “smartphone” δυνατότητές τους που περιλαμβάνουν όπως one-touch Google searches, Google Docs Google Earth και Google Street View. Υπάρχει επίσης η δυνατότητα ανάπτυξης εφαρμογών από τρίτους, γραμμένων σε Java βασισμένων σε Linux Kernel χρησιμοποιώντας Google Android API.

3.3.3 Χαρακτηριστικά του Android

- Η πλατφόρμα είναι συμβατή με μεγαλύτερες VGA, 2D και 3D βιβλιοθήκες γραφικών σε OpenGL ES 1.0 specification και smartphone layouts
- Το λογισμικό βάσης δεδομένων SQLite χρησιμοποιείται για αποθήκευση δεδομένων.
- Η πλατφόρμα υποστηρίζει τεχνολογίες συνδεσιμότητας συμπεριλαμβανομένου GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth και WiFi
- SMS και MMS είναι οι διαθέσιμες μορφές μηνυμάτων
- Λογισμικό γραμμένο σε Java μπορεί να γίνει compile στο Dalvik virtual machine το οποίο αποτελεί μια εξειδικευμένη υλοποίηση σχεδιασμένη για χρήση σε κινητές συσκευές.
- Η πλατφόρμα υποστηρίζει τις εξής μορφές πολυμέσων: MPEG-4, H.264, MP3, AAC, MIDI, OGG, AMR, JPEG, PNG, GIF.
- Η πλατφόρμα διαθέτει το Android Market το οποίο αποτελεί ένα σύστημα κατανομής ανοιχτού περιεχομένου που δίνει τη δυνατότητα στους καταναλωτές να αναζητήσουν να αγοράσουν να κατεβάσουν και να εγκαταστήσουν περιεχόμενο διαφόρων ειδών σε μορφή εφαρμογών.

3.3.4 Αρχιτεκτονική του Android

Το παρακάτω σχήμα απεικονίζει την αρχιτεκτονική του Android στην οποία περιλαμβάνονται τα κύρια μέρη του λειτουργικού συστήματος. Υπάρχουν τέσσερα επίπεδα σε αυτήν την αρχιτεκτονική με το Linux Kernel στη βάση και την εφαρμογή στην κορυφή.



Το επίπεδο της εφαρμογής (Application) περιέχει ένα σύνολο από εφαρμογές πυρήνα (core applications) όπως ο email client, SMS manager, calendar, maps, browser, contacts και άλλα. Όλες οι εφαρμογές είναι γραμμένες σε γλώσσα Java.

Στο επίπεδο του Java API Framework, οι developers έχουν πλήρη δικαιώματα πρόσβασης στον πυρήνα του application framework. Αυτό το application framework απλοποιεί την επαναχρησιμοποίηση των συνθετικών μερών. Κάθε developer μπορεί να δημοσιεύσει τις δυνατότητες της εφαρμογής τους και έτσι άλλοι developer μπορεί να τις χρησιμοποιήσουν. Το framework layer αποτελείται από υπηρεσίες που περιλαμβάνουν views, content provider, resource provider, notification manager και activity manager.

Οι βιβλιοθήκες (libraries) και το Runtime layer περιλαμβάνουν ένα σύνολο από γλώσσες C και C++ και μερικές από τις βιβλιοθήκες πυρήνα (core libraries), βιβλιοθήκες 3D, SQL, Surface manager κλπ. Το Dalvik virtual machine χρησιμοποιείται για να κάνει compile προγράμματα γραμμένα σε γλώσσα Java.

Το hardware abstraction layer (HAL) παρέχει βασικά interfaces που δίνουν πρόσβαση σε δυνατότητες υλικού (hardware capabilities) στο υψηλότερου επιπέδου Java API framework. Το HAL αποτελείται από πολλαπλές βιβλιοθήκες κάθε μια από τις οποίες υλοποιεί ένα interface για ένα συγκεκριμένου τύπου hardware component όπως η camera ή το Bluetooth.

Και τέλος το Linux Kernel είναι το λειτουργικό σύστημα που διαχειρίζεται το φυσικό κομμάτι του υλικού (physical hardware) μαζί με μια πληθώρα υπηρεσιών όπως security, networking memory management, drivers για διάφορες συσκευές και Power management. Το kernel λειτουργεί επίσης σαν abstraction layer μεταξύ του υλικού (hardware) και της υπόλοιπης στοίβας λογισμικού (software stack).

3.3.5 Activities and their Lifecycle

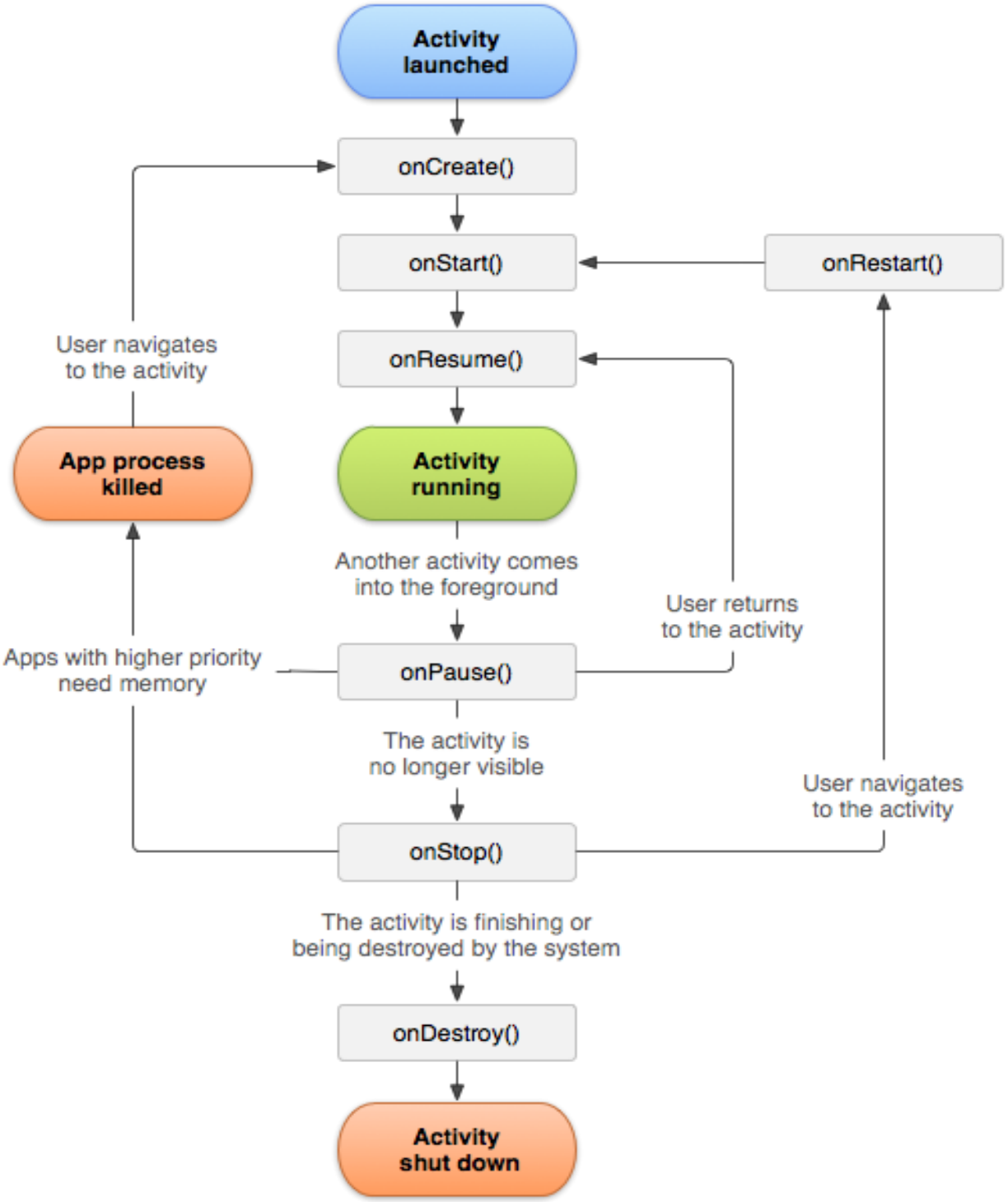
Το Activity είναι μια κλάση που αντιπροσωπεύει μια συγκεκριμένη λειτουργία που μπορεί να κάνει ο χρήστης. Σχεδόν όλα τα activities είναι διαδραστικά, έτσι η κλάση του activity φροντίζει και για την δημιουργία του UI (User Interface). Τα activities παρουσιάζονται συχνά στον χρήστη ως παράθυρα πλήρους οθόνης μπορούν όμως να χρησιμοποιηθούν και με άλλους τρόπους όπως αναδυόμενα ανεξάρτητα παράθυρα ανεξάρτητα από το κεντρικό.

Το σύστημα διαχειρίζεται τα activities ως στοίβα, το λεγόμενο activity stack. Όταν αρχίζει ένα νέο activity τοποθετείται στην κορυφή της στοίβας και γίνεται το τρέχων activity (running activity), το προηγούμενο activity μένει από κάτω στη στοίβα και δεν εμφανίζεται μέχρι να κλείσει το νέο activity.

Ένα activity έχει τέσσερεις βασικές καταστάσεις:

- Αν το activity είναι σε πρώτο πλάνο στην οθόνη (στην κορυφή της στοίβας) θεωρείται ως ενεργή (active or running).
- Αν το activity δεν είναι εστιασμένο αλλά εμφανίζεται ακόμα στην οθόνη (σε περίπτωση που ένα διαφανές ή μικρότερου μεγέθους activity έχει εστιαστεί πάνω από το αναφερόμενο activity), τότε θεωρείται σε παύση (paused). Ένα activity σε παύση είναι «απόλυτα ζωντανό» που σημαίνει πως διατηρεί όλες τις καταστάσεις και τις πληροφορίες των μελών τις, επίσης παραμένει συνδεδεμένη στο window manager, αλλά το σύστημα μπορεί να την κλείσει ή να την «σκοτώσει» όπως λέμε σε περιπτώσεις υπερβολικά χαμηλής διαθέσιμης μνήμης.
- Αν το activity είναι πλήρως καλυμμένο από κάποιο άλλο activity τότε σταματάει, βρίσκεται δηλαδή σε κατάσταση “stopped”, που σημαίνει πως δεν είναι πλέον ορατό στον χρήστη άρα το παράθυρό του είναι αόρατο στο χρήστη και μπορεί να σκοτωθεί από το σύστημα αν χρειαστεί μνήμη κάπου αλλού.
- Αν το activity είναι σε παύση ή σταματημένο (paused or stopped), το σύστημα μπορεί να αφαιρέσει το activity από τη μνήμη είτε ζητώντας από το χρήστη να κλείσει το activity (καλώντας την μέθοδο finish) είτε σκοτώνοντας απλά την διεργασία του. Όταν εμφανιστεί πάλι στον χρήστη πρέπει να έχει γίνει πλήρης επανεκκίνηση και επαναφορά στην προηγούμενή του κατάσταση.

Παρακάτω φαίνεται το διάγραμμα που περιγράφει τις κύριες καταστάσεις ενός activity, τα τετράγωνα αντιπροσωπεύουν τις callback methods που μπορούν να υλοποιηθούν για να εκτελεστούν διάφορες λειτουργίες όταν το activity αλλάζει κατάσταση.



3.4 Σύγκριση Σύγχρονων mobile OS με την πλατφόρμα Android

3.4.1 Symbian

Το Symbian υπήρξε μια από τις παλαιότερες πλατφόρμες smartphone. Η ιστορία του άρχισε σαν επιπρόσθετη ανάπτυξη του λειτουργικού συστήματος EPOC της PSION η οποία το 1997-1998 σε συνεργασία με τις κατασκευαστικές εταιρίες κινητών τηλεφώνων Ericsson, Motorola και Nokia μετασχηματίσανε το EPOC σε λειτουργικό σύστημα κατάλληλο να τρέχει σε κινητές συσκευές, το Symbian. Από την αρχή το Symbian είχε τρία κύρια γραφικά περιβάλλοντα: Nokia S60, Ericsson UIQ και MOAP(S) από τον γιαπωνέζικο πάροχο κινητών NTT. Με τον καιρό οι υπόλοιπες εταιρείες αποχώρησαν, έτσι η Nokia αγόρασε τα πλήρη δικαιώματα παίρνοντας το όνομα Symbian Foundation.

Τον Ιούλιο του 2008 η Symbian Foundation ανακοίνωσε το μεγαλύτερο βήμα στην εξέλιξη του Symbian OS από την μέρα της δημιουργίας του, μετατρέποντάς το σε πλατφόρμα ανοιχτού κώδικα ενώ σχεδίαζε την ολοκληρωτική παράδοση του ανοιχτού κώδικα σε δύο χρόνια.

Αντιθέτως όμως με το Android το οποίο είναι πραγματικά ελεύθερο λειτουργικό σύστημα ανοιχτού κώδικα, το Symbian δεν έχει φτάσει ακόμα στο στάδιο του ελεύθερου λειτουργικού συστήματος πλήρης στοίβας ανοιχτού κώδικα. Η Symbian Foundation ανακοίνωσε την κυκλοφορία του beta version security package τον Ιούλιο του 2009 υπό την Eclipse Public License (EPL). Η EPL δίνει τη δυνατότητα στο package να προσπεράσει κανονισμούς εξαγωγής σε κρυπτογραφικά προϊόντα από το Ηνωμένο Βασίλειο, την βάση του Symbian, όπως αναφέρεται σε δημόσιους κανονισμούς αδειοδότησης. Το σημερινό Symbian OS όμως περιλαμβάνει πολλούς ιδιοκτήτους κώδικες που θα χρειαστούν να αδειοδοτηθούν υπό την EPL για να γίνει το Symbian OS πραγματικά λειτουργικό σύστημα ανοιχτού κώδικα.

Σε σύγκριση με το Android το Symbian είναι ένα multitasking OS με δυνατότητα να εκτελεί πολλαπλές εφαρμογές ταυτόχρονα. Η πλατφόρμα υποστηρίζει ποικίλες προγραμματιστικές γλώσσες όπως C/C++ για μεταφορά ήδη υπαρχόντων εφαρμογών UNIX, και Java για μεταφορά εφαρμογών Java ME. Η κύρια γλώσσα προγραμματισμού για την πλατφόρμα όμως είναι η Symbian C++, μια γλώσσα της οποίας η καμπύλη εκμάθησης είναι ιδιαίτερα κρημνώδης, κάτι που κάνει την πλατφόρμα Android να προτιμάται περισσότερο σε γενικό βαθμό από τους developers επειδή εγγυάται σχεδόν πάντοτε ένα σταθερό περιβάλλον εφαρμογής σε όλες τις συσκευές Android. Το virtual machine του Android παρέχει ένα επίπεδο για προγραμματιστές ώστε οι developers να μην χρειάζεται να ανησυχούν για το υλικό (hardware) που βρίσκεται από κάτω πάνω στο οποίο δουλεύει το Android. Έτσι δεν χρειάζεται ανάπτυξη της εφαρμογής ξανά, για την μεταφορά μιας εφαρμογής ανάμεσα σε συσκευές Android. Οι developers στην πλατφόρμα Android έχουν δυνατότητα πρόσβασης όλων των κατωτέρων επιπέδων του Android Framework μέχρι και το Linux Kernel σε αντίθεση με το Symbian που δίνει πρόσβαση μόνο μέχρι το επίπεδο του middleware. Οι developers λοιπόν μπορούν να δημιουργήσουν περισσότερες λειτουργίες έχοντας πρόσβαση σε περισσότερα επίπεδα του Android OS.

3.4.2 RIM BlackBerry

Άλλο ένα ανταγωνιστικό OS στην αγορά των smartphones ήταν το BlackBerry από την RIM (Research in Motion). Το BlackBerry κυκλοφόρησε το 2008 με την επαναστατική για τότε πλατφόρμα του που εστίαζε στην διαχείριση των email ενώ διατηρούσε αρκετά καλές επιδόσεις από θέμα ταχύτητας και αξιοπιστίας. Το BlackBerry ήταν ένα σύστημα σχεδιασμένο κυρίως για επαγγελματική χρήση, που εστιάζει στην ταχύτητα και την αξιοπιστία του συστήματος αλλά για περιορισμένου τύπου χρήσεις.

Όσον αφορά το λειτουργικό σύστημα ενώ το RIM είναι ιδιόκτητο σύστημα οι developers μπορούν να δημιουργήσουν εφαρμογές χρησιμοποιώντας APIs όπως τα Novell Group Wise, Lotus Notes και τα ιδιόκτητα BlackBerry APIs. Η αναπτυσσόμενη εφαρμογή όμως που δημιουργείται με τη χρήση κάποιου είδους περιορισμένης λειτουργικότητας πρέπει να φέρει ψηφιακή υπογραφή, ώστε να μπορεί συνδέεται με το ανάλογο developer account στην RIM. Η υπογραφή εγγυάται την αυθεντικότητα της εφαρμογής, όχι όμως και την ποιότητα ή την ασφάλεια του κώδικα.

Ενώ για το Android, υπάρχει η δυνατότητα στους developers να αναπτύξουν διαχειριζόμενο κώδικα σε γλώσσα Java ελέγχοντας την συσκευή από βιβλιοθήκες Java ανεπτυγμένες από την Google. Το Android προσφέρει λειτουργικό σύστημα πλήρους στοίβας που σημαίνει ότι το Android προσφέρει περισσότερο API σε σχέση με το RIM. Και εδώ developers στην πλατφόρμα Android έχουν δυνατότητα πρόσβασης όλων των κατωτέρων επιπέδων του Android Framework μέχρι και το Linux Kernel σε αντίθεση με το RIM που δίνει πρόσβαση μόνο μέχρι το επίπεδο του middleware. Οι developers λοιπόν μπορούν να δημιουργήσουν περισσότερες λειτουργίες έχοντας πρόσβαση σε περισσότερα επίπεδα του Android OS.

3.4.3 Palm OS - WebOS

Το Palm OS ήταν ο βασιλιάς της χαμένης εποχής των PDA. Με την πρόοδο συσκευών με περισσότερες δυνατότητες η ανάγκη ύπαρξης μιας δεύτερης συσκευής για την οργάνωση προσωπικών πληροφοριών (κατάλογος τηλεφώνων, ημερολόγιο συναντήσεων κλπ) μειώθηκε σημαντικά. Η Palm προσπάθησε να κάνει την μετάβαση στην αγορά των κινητών τηλεφώνων με την σειρά Palm Treo αλλά απέτυχε.

WebOS:

Σαν ομοιότητα μπορεί να παρατηρηθεί ότι και οι δύο πλατφόρμες είναι ανοιχτού κώδικα βασισμένες σε Linux χρησιμοποιώντας γλώσσα Java. Το WebOS όμως δεν αποτελεί λειτουργικό σύστημα πλήρους στοίβας όπως το Android.

3.4.4 Windows Mobile

Το Windows Mobile αποτελεί μια ιδιόκτητη ανοιχτή τεχνολογία αρχιτεκτονικής ευρέως χρησιμοποιημένη σε Nigh Level Operating Systems (HLOS) για smartphones. Το Windows Mobile θεωρείται ιδιόκτητο αφού η αρχιτεκτονική και η ανάπτυξή του ελέγχονται από τη Microsoft. Το λειτουργικό σύστημα είναι ανοικτά και ελεύθερα αδειοδοτημένο σε πάνω από 20 OEMs που κατασκευάζουν κινητές συσκευές βασισμένες στο Windows Mobile.

Από την πλευρά της ασφάλειας η Microsoft και η Windows Mobile συνεργάστηκαν με σκοπό την διαχείριση της ασφάλειας μέσω του Exchange Activesync. Αυτό δίνει τη δυνατότητα στους χρήστες να ενεργοποιήσουν client-based authentication και να το χρησιμοποιήσουν μαζί με άλλες λειτουργίες ασφάλειας όπως κωδικός συσκευής απομακρυσμένη εκκαθάριση (remote wipe) για τη διαγραφή όλων των προσωπικών δεδομένων του χρήστη σε περίπτωσης απώλειας ή κλοπής της συσκευής. Το Android από την άλλη πλευρά εκτελεί διάφορες λειτουργίες βασισμένες στο μοντέλο αίτησης άδειας από τον χρήστη (permission-based). Για την εκτέλεση δηλαδή συγκεκριμένων λειτουργιών που είναι επίφοβες από θέμα ασφάλειας γίνεται αίτηση άδειας από τον χρήστη.

3.4.5 iOS

Όσον αφορά την αδειοδότηση του λειτουργικού συστήματος η σύγκριση γίνεται αρχικά ανάμεσα σε ένα ιδιόκτητο λειτουργικό σύστημα και ένα σύστημα ανοιχτού κώδικα. Το Android ως ελεύθερο λειτουργικό σύστημα ανοιχτού κώδικα δίνει την ελευθερία στους developers να προσαρμόσουν το λειτουργικό σύστημα πάνω στο υλικό (hardware) στο οποίο θέλουν να δουλέψουν. Κάτι που μπορεί να οδηγήσει σε περισσότερες καινοτομίες αλλά διατρέχει παράλληλα τον κίνδυνο να οδηγήσει σε κατακερματισμό κάτι που πρέπει να ελέγχεται από την OHA. Ως ιδιόκτητο λειτουργικό σύστημα το iOS προορίζεται μόνο για συγκεκριμένο hardware (iPhone iPad κλπ). Σε σύγκριση με το Android υπάρχει περισσότερη ευκολία για την Apple να ενσωματώσει ένα μοντέλο αδιάλειπτης συνέργιας και εμπειρίας χρήσης ανάμεσα στις υπηρεσίες της αφού το σύστημα είναι στοχευόμενο σε συγκεκριμένο υλικό. Από την άλλη πλευρά η ελευθερία πρόσβασης και παραμετροποίησης σε όλα τα επίπεδα του framework που παρέχει η πλατφόρμα Android της δίνει μεγάλο πλεονέκτημα σε σχέση με το iOS όσον αφορά την εξέλιξη της ανάπτυξης εφαρμογών, της μεταφερσιμότητας και της παραμετροποίησης του λειτουργικού συστήματος. Το κόστος ανάπτυξης λογισμικού είναι ακόμη ένα σημαντικό πλεονέκτημα του Android αφού βασίζεται σε ελεύθερο λογισμικό ανοιχτού κώδικα σε αντίθεση με το ιδιόκτητο μοντέλο του iOS. Στον παρακάτω πίνακα παραβάλλονται οι διαφορές των δύο λειτουργικών.

Comparison Chart by http://www.diffen.com/difference/Android_vs_iOS

	Android	**iOS**
Customizability	A lot. Can change almost anything.	Limited unless jailbroken
Developer	Google	Apple Inc.
Initial release	September 23, 2008	July 29, 2007
Source model	Open source	Closed, with open source components.

	Android	iOS
OS family	Linux	OS X, UNIX
Widgets	Yes	No, except in NotificationCenter
File transfer	Easier than iOS. Using USB port and Android File Transfer desktop app. Photos can be transferred via USB without apps.	More difficult. Media files can be transferred using iTunes desktop app. Photos can be transferred out via USB without apps.
Available on	Many phones and tablets. Major manufacturers are Samsung, Motorola, LG, HTC and Sony.. Nexus and Pixel line of devices is pure Android, others bundle manufacturer software.	iPod Touch, iPhone, iPad, Apple TV (2nd and 3rd generation)
Calls and messaging	Google Hangouts. 3rd party apps like Facebook Messenger, WhatsApp, Google Duo and Skype all work on Android and iOS both.	iMessage, FaceTime (with other Apple devices only). 3rd party apps like Google Hangouts, Facebook Messenger, WhatsApp, Google Duo and Skype all work on Android and iOS both.
Internet browsing	Google Chrome (or Android Browser on older versions; other browsers are available)	Mobile Safari (Other browsers are available)
App store , Affordability and interface	Google Play – 1,000,000+ apps. Other app stores like Amazon and Getjar also distribute Android apps. (unconfirmed ".APKs")	Apple app store – 1,000,000+ apps
Available language(s)	100+ Languages	34 Languages
Video chat	Google Duo and other 3rd party apps	FaceTime (Apple devices only) and other 3rd party apps
Voice commands	Google Now, Google Assistant	Siri
Maps	Google Maps	Apple Maps (Google Maps also available via a separate app download)
Latest stable release and Updates	Android 8.0.0, Oreo (Aug 21, 2017)	11 (Sep 19, 2017)
Alternative app stores and side loading	Several alternative app stores other than the official Google Play Store. (e.g. Aptoide, Galaxy Apps)	Apple blocks 3rd party app stores. The phone needs to be jailbroken if you want to download apps from other stores.

	Android	iOS
Battery life and management	Many Android phone manufacturers equip their devices with large batteries with a longer life.	Apple batteries are generally not as big as the largest Android batteries. However, Apple is able to squeeze decent battery life via hardware/software optimizations.
Open source	Kernel, UI, and some standard apps	The iOS kernel is not open source but is based on the open-source Darwin OS.
File manager	Yes. (Stock Android File Manager included on devices running Android 7.1.1)	Not available
Photos & Videos backup	Apps available for automatic backup of photos and videos. Google Photos allows unlimited backup of photos. OneDrive, Amazon Photos and Dropbox are other alternatives.	Up to 5 GB of photos and videos can be automatically backed up with iCloud. All other vendors like Google, Amazon, Dropbox, Flickr and Microsoft have auto-backup apps for both iOS and Android.
Security	Android software patches are available soonest to Nexus device users. Manufacturers tend to lag behind in pushing out these updates. So at any given time a vast majority of Android devices are not running updated fully patched software.	Most people will never encounter a problem with malware because they don’t go outside the Play Store for apps. Apple's software updates support older iOS devices also.
Rooting, bootloaders, and jailbreaking	Access and complete control over your device is available and you can unlock the bootloader.	Complete control over your device is not available.
Cloud services	Native integration with Google cloud storage. 15GB free, $2/mo for 100GB, 1TB for $10. Apps available for Amazon Photos, OneDrive and Dropbox.	Native integration with iCloud. 5GB free, 50GB for $1/mo, 200GB for $3/mo, 1TB for $10/mo. Apps available for Google Drive and Google Photos, Amazon Photos, OneDrive and Dropbox.
Working state	Current	Current
Interface	Touch Screen	Touch Screen
Supported versions	Android 5.0 & later (Android 4.4 is also supported but with patches)	iOS 8 & later
First version	Android 1.0, Alpha	iOS 1.0

Abdullah Humayun, M.Y, Dang, T.T.P., Himawan, A.G., Koirala, Y., Ridwan, R.M. & Wibiyanto, D. (2009), Android Documentation: https://developer.android.com/guide/platform/index.html

4. Σχεδιασμός Εφαρμογής

Σε αυτό το κεφάλαιο θα περιγραφεί η αρχική ανάλυση του προβλήματος που οδήγησε στο σχεδιασμό του πυρήνα βασικών λειτουργιών γύρω από τις οποίες αναπτύχθηκε η εφαρμογή. Θα γίνει περιγραφή των βασικών δομικών στοιχείων της εφαρμογής, των τρόπων υλοποίησης, της μεθοδολογίας που ακολουθήθηκε καθώς και των τρόπων αντιμετώπισης προβλημάτων είτε σχεδιαστικών είτε υλοποίησης. Επιπλέον θα γίνει αναφορά στη βασική δομή των κλάσεων όπως και στα βασικά σενάρια χρήσης που χρησιμοποιήθηκαν στον σχεδιασμό και την αντιμετώπιση προβλημάτων μέσω μοντέλων UML.

Η Μεθοδολογία που ακολουθήθηκε μπορεί να περιγραφεί από τα βασικά στάδια ανάπτυξης λογισμικού τα οποία είναι:

- Προσδιορισμός και Ανάλυση Απαιτήσεων (Requirement gathering and analysis)
- Σχεδιασμός (Design)
- Υλοποίηση (Implementation/Coding)
- Δοκιμή Λειτουργίας (Testing)
- Παράδοση της τελικής εφαρμογής (Deployment)
- Συντήρηση (Maintenance)

Στο συγκεκριμένο κεφάλαιο θα γίνει κυρίως ανάλυση του Σχεδιασμού και προσδιορισμός απαιτήσεων, ενώ σε επόμενο κεφάλαιο θα αναλυθεί εις βάθος το στάδιο της Υλοποίησης.

4.1 Θεωρίες

Σε αυτό το σημείο θα αναλυθούν οι θεωρίες που μελετήθηκαν καθώς και το μέρος της εφαρμογής στο οποίο εφαρμόστηκαν.

Σαν γενικότερες γνώσεις μελετήθηκε η θεωρία της αντικειμενοστρέφειας η οποία εφαρμόστηκε σε όλη την έκταση της εφαρμογής καθώς αποτελεί ένα απαραίτητο δομικό στοιχείο στην ανάπτυξη λογισμικού σε μια αντικειμενοστραφής γλώσσα προγραμματισμού όπως η Java η οποία χρησιμοποιήθηκε στην παρούσα περίπτωση. Συνοπτικά αναφέρονται οι έννοιες της κληρονομικότητας (inheritance), του πολυμορφισμού (polymorphism), της αφαιρετικότητας (abstraction), και της ενθυλάκωσης (encapsulation) που περιλαμβάνονται στην θεωρία της αντικειμενοστρέφειας.

Σαν πιο ειδικές γνώσεις εξίσου απαραίτητες για την ανάπτυξη μιας εφαρμογής σε πλατφόρμα Android ήταν επίγνωση της αρχιτεκτονικής του Android όπως του ιδιόμορφου lifecycle των activities που αναλύεται στο κεφάλαιο 3.3.

Απαραίτητες ήταν επίσης οι μαθηματικές θεωρίες συνόλων όπως και συγκεκριμένοι κλάδοι της Γεωμετρίας όπως η Επιπεδομετρία που χρησιμοποιήθηκαν αρκετά στον έλεγχο εγκυρότητας των ραντεβού για τον υπολογισμό της απόστασης, της κατεύθυνσης και του χρόνου μετάβασης από ένα σημείο σε ένα άλλο.

Η κανονικοποίηση βάσης δεδομένων υπήρξε επίσης μια σημαντική θεωρία που μελετήθηκε και εφαρμόστηκε στην τοπική βάση της εφαρμογής για την ομαλή λειτουργία της αποθήκευσης και της ανάκτησης δεδομένων.

4.2 Αλγόριθμοι

Στην εφαρμογή χρησιμοποιήθηκαν αλγόριθμοι ταξινόμησης για τις λίστες των σημείων με χρήση της κλάσης Comparator από τη βιβλιοθήκη java.util για την ταξινόμηση των σημείων στις λίστες. Επιπλέον χρησιμοποιήθηκαν αλγόριθμοι εύρεσης συντομότερης διαδρομής όπως Djikstra και αλγόριθμοι αναζήτησης για την επιλογή του κοντινότερου σημείου στο στάδιο της δρομολόγησης.

4.3 Ανάλυση Προβλήματος

Το πρόβλημα χωρίζεται σε τρεις βασικές κατηγορίες ανάλογα με την φύση των λειτουργιών που απαιτούνταν:

- Διαχείριση Δραστηριοτήτων
- Δρομολόγηση
- Αποθήκευση
- Εμφάνιση

Η διαχείριση δραστηριοτήτων ήταν ένα μεγάλο κομμάτι μερικά από τα βασικότερα προβλήματα που το αφορούσαν ήταν τα εξής:

- Πως θα λειτουργεί η διαχείριση των ραντεβού όσον αφορά τη δημιουργία και τη διαγραφή τους;
- Πως θα ικανοποιούνται όλες οι συνθήκες ώστε το ραντεβού να είναι έγκυρο;
- Πως θα διαφοροποιούνται τα ραντεβού;
- Πως θα υπάρχει πρόσβαση στα ραντεβού μετά τη δημιουργία τους;

Η δρομολόγηση βρίσκεται στον πυρήνα του σκοπού της εφαρμογής πληθώρα προβλημάτων μπορούν να αναλυθούν, τα βασικότερα των οποίων θα αναφερθούν επιγραμματικά:

- Με ποια μορφή θα γίνεται η επικοινωνία με τις υπηρεσίες δρομολόγησης;
- Πως θα ανιχνεύεται η τρέχουσα θέση του χρήστη;
- Με ποιο τρόπο θα γίνεται ο υπολογισμός της απόστασης;
- Πως θα γίνεται ο υπολογισμός του κοντινότερου σημείου;
- Πώς θα επιστρέφεται η ακριβής διαδρομή στον χρήστη;
- Πως θα αποφευχθεί η υπέρβαση του ορίου όσον αφορά τον αριθμό των αιτήσεων θέσης στην υπηρεσία;

Η αποθήκευση αναφέρεται στην διατήρηση κάθε τύπου δεδομένου που χρειάζεται για την λειτουργία της εφαρμογής. Διακρίνονται τα εξής προβλήματα:

- Ποια δεδομένα είναι απαραίτητο να διατηρηθούν;
- Σε ποιο στάδιο της εφαρμογής μέσα στο activity lifecycle απαιτείται να είναι διαθέσιμα τα δεδομένα;
- Ποια δεδομένα χρειάζεται να διατηρηθούν ασύγχρονα;
- Με ποιο τρόπο θα διατηρούνται τα δεδομένα κατά την περιστροφή της συσκευής;
- Με ποιο τρόπο θα γίνεται η διατήρηση και μεταφορά δεδομένων ανάμεσα στα activities;
- Ποιο μοντέλο ασύγχρονης αποθήκευσης είναι καταλληλότερο για την εφαρμογή;

Η Εμφάνιση αναφέρεται στο κομμάτι της διεπαφής με την οποία αλληλεπιδρά ο χρήστης, αποτελεί ένα βασικό μέρος καθώς καθορίζει σε μεγάλο βαθμό την ευχρηστία της εφαρμογής, διακρίνονται λοιπόν τα εξής προβλήματα:

- Ποια είναι η ιδανική διάταξη των στοιχείων διεπαφής που να αποδίδει την αναμενόμενη λειτουργικότητα ενώ παράλληλα υπακούει στους βασικούς κανόνες των θεωριών διεπαφής ανθρώπου-μηχανής;
- Πως θα επιτευχθεί συμβατότητα για πολλαπλά μεγέθη οθονών ενώ παράλληλα διατηρείται ομοιομορφία στο σχεδιασμό της διεπαφής;
- Με ποιο τρόπο θα σχεδιαστούν εξειδικευμένες διεπαφές για κατακόρυφο και οριζόντιο προσανατολισμό σε πολλαπλά μεγέθη οθόνης;

4.4 Βασικές Λειτουργίες

Οι βασικές λειτουργίες της εφαρμογής πηγάζουν από τα κύρια προβλήματα που αναφέρονται στην υποκεφάλαιο 4.1 και αποτελούν τον πυρήνα γύρω από τον οποίο αναπτύχθηκε η εφαρμογή. Η εφαρμογή ουσιαστικά πρέπει να είναι ικανή να εκτελέσει τέσσερεις βασικές λειτουργίες:

1) Διαχείριση Λογαριασμών: Αποτελείται από τη δημιουργία, την προβολή και την επεξεργασία μοναδικού λογαριασμού στον οποίο θα αποθηκεύονται όλες οι προσωπικές πληροφορίες του χρήστη καθώς και η λίστα των ραντεβού του.

2) Διαχείριση Ραντεβού: Αποτελείται από τη δημιουργία, τη διαγραφή και τη προβολή των ραντεβού που αντιστοιχούν στον λογαριασμό του συγκεκριμένου χρήστη.

3) Έλεγχος εγκυρότητας ραντεβού: Αποτελείται από πλήθος ελέγχων που εκτελούνται κατά την δημιουργία και την δρομολόγηση των ραντεβού. Οι έλεγχοι λειτουργούν με βάση συγκεκριμένες συνθήκες αν όλες οι συνθήκες ικανοποιούνται τότε το ραντεβού είναι έγκυρο, που σημαίνει πως ο χρήστης προλαβαίνει να πάει στο ραντεβού τη συγκεκριμένη ώρα και δεν υπάρχει άλλο ραντεβού την ίδια ώρα. Ο έλεγχος εγκυρότητας ραντεβού είναι ένα μεγάλο μέρος της εφαρμογής και θα αναλυθεί εκτενέστερα σε επόμενο κεφάλαιο.

4) Δρομολόγηση: Αποτελείται από την καθοδήγηση του χρήστη στο κοντινότερο σημείο του επιλεγμένου ραντεβού από τη τρέχουσα θέση αφού έχει καθοριστεί πως το ραντεβού είναι έγκυρο.

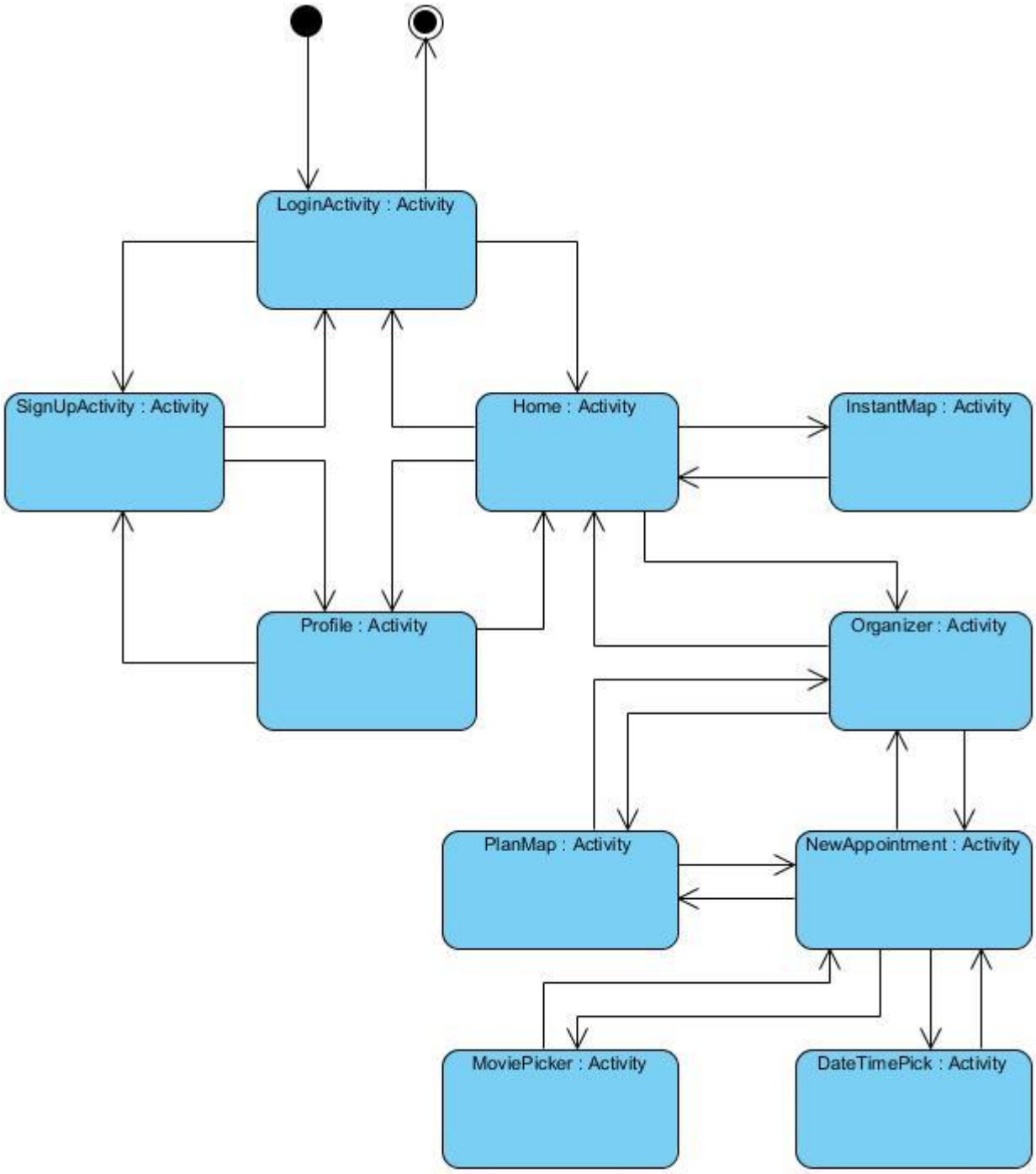
4.5 Βασική Δομή Εφαρμογής: Οι Ρόλοι των Activities

Η εφαρμογή απαρτίζεται από 10 βασικά activities. Τα activities αυτά αποτελούν τις 10 βασικές οθόνες αλληλεπίδρασης που μπορεί να συναντήσει ο χρήστης. Το κάθε activity αντιπροσωπεύει συγκεκριμένους ρόλους σε σχέση με τη λειτουργία της εφαρμογής, παρακάτω γίνεται μια σύντομη αναφορά στα 10 βασικά activities της εφαρμογής:

- **LoginActivity**: Ρόλος του συγκεκριμένου activity είναι να παρέχει στον χρήστη τη δυνατότητα να εισάγει τα credentials του για να συνδεθεί στο λογαριασμό του ή αν δεν έχει λογαριασμό να πατήσει το κουμπί που τον μεταφέρει στο activity υπεύθυνο για sign up.
- **SignUpActivity**: Ο κύριος ρόλος του συγκεκριμένου activity είναι να παρέχει στο χρήστη τη δυνατότητα δημιουργίας νέου λογαριασμού, χρησιμοποιείται επίσης και για την επεξεργασία υπάρχοντος λογαριασμού.
- **Home**: Το Home Activity είναι στην ουσία ένας κόμβος από τον οποίο υπάρχει πρόσβαση σε όλες τις βασικές λειτουργίες της εφαρμογής, χρησιμοποιείται δηλαδή σαν αρχική σελίδα.
- **Profile**: Ο ρόλος του Profile είναι να δώσει στον χρήστη τη δυνατότητα προβολής των στοιχείων του λογαριασμού, εδώ υπάρχει επίσης και το κουμπί που τον μεταφέρει στην κατάλληλη σελίδα για επεξεργασία των στοιχείων αυτών.
- **InstantMap**: Στο activity InstantMap ο χρήστης μπορεί να δρομολογηθεί γρήγορα στο κοντινότερο σημείο που θα επιλέξει από την τρέχουσα θέση του χωρίς να λαμβάνονται υπόψη οι υπόλοιπες υποχρεώσεις του (ώρες ραντεβού, χρόνος μετάβασης κλπ).
- **Organizer**: Ο βασικός ρόλος του Organizer είναι να δίνει στον χρήστη τη δυνατότητα πλήρους διαχείρισης των ραντεβού του. Εδώ ο χρήστης μπορεί να προβάλει τα υπάρχοντα ραντεβού του, να δρομολογηθεί σε αυτά, να διαγράψει επιλεγμένα ραντεβού και να δημιουργήσει καινούρια.
- **NewAppointment**: Το NewAppointment αποτελεί μια φόρμα δημιουργίας νέου ραντεβού, είναι επίσης ο κόμβος που δίνει πρόσβαση στα activities υπεύθυνα για την επιλογή ώρας και μέρας ραντεβού, την επιλογή ταινίας και τέλος το activity υπεύθυνο για τον έλεγχο εγκυρότητας και την δρομολόγηση του ραντεβού.
- **DateTimePick**: Αποτελεί το activity υπεύθυνο για την επιλογή μέρας και ώρας του ραντεβού όταν αυτό περιλαμβάνει σημείο τύπου supermarket ή βενζινάδικου, καθώς η επιλογή μέρας για σημείο cinema γίνεται σε ξεχωριστό activity.
- **MoviePicker**: Το MoviePicker είναι το activity που διαχειρίζεται την επιλογή ταινίας, κινηματογράφου, μέρας και ώρας για ραντεβού τύπου cinema.
- **PlanMap**: Το PlanMap διαχειρίζεται τον έλεγχο εγκυρότητας του ραντεβού, καθώς και τη δρομολόγηση του χρήστη στα ραντεβού. Το activity αυτό χρησιμοποιείται σε δύο περιπτώσεις: στην δημιουργία νέου ραντεβού για τον έλεγχο εγκυρότητας, και στην δρομολόγηση υπάρχοντος ραντεβού από το Organizer όπου γίνεται και πάλι έλεγχος εγκυρότητας πριν τη δρομολόγηση.

Παρακάτω παραβάλλεται ένα Abstract State Machine Diagram που απεικονίζει το Activity Lifecycle της εφαρμογής.

NavIraklio Activity Lifecycle - Figure 1



4.6 Προσδιορισμός Απαιτήσεων

Το κάθε activity επιτελεί όπως αναφέρθηκε κάποιο συγκεκριμένο ρόλο, οπότε είναι περισσότερο χρήσιμο να γίνει προσδιορισμός των απαιτήσεων ως προς το κάθε activity ξεχωριστά από το να οριστούν γενικότερες απαιτήσεις τις εφαρμογής χωρίς καμία κατηγοριοποίηση. Οπότε στο κεφάλαιο αυτό θα γίνει ο απαραίτητος προσδιορισμός απαιτήσεων ως προς τα activities και τους ρόλους που επιτελούν.

4.6.1 LoginActivity

Ο χρήστης πρέπει να έχει τη δυνατότητα εισόδου στον λογαριασμό του ενώ παράλληλα να υπάρχει εμφανής τρόπος για να γίνεται δημιουργία νέου λογαριασμού. Επιπλέον πρέπει να γίνεται έλεγχος εγκυρότητας των πεδίων (κενά πεδία, μικρό username κλπ) και τέλος να γίνεται ο έλεγχος ταυτότητας του λογαριασμού (authentication).

4.6.2 SignUpActivity

Ως απαιτήσεις ορίζονται ο έλεγχος εγκυρότητας των πεδίων, η ομοιομορφία στον σχεδιασμό του layout όσον αφορά τη φόρμα, ο διαχωρισμός ρόλου του activity ανάλογα με τη λειτουργία που ζητείται να εκτελέσει (δημιουργία ή επεξεργασία λογαριασμού), ο έλεγχος για υπάρχουσα εγγραφή με το ίδιο primary key όπως το username που έχει οριστεί στη περίπτωση δημιουργίας νέου λογαριασμού και η αποτροπή αλλαγής του username στη περίπτωση επεξεργασίας λογαριασμού, και τέλος η εισαγωγή ή ενημέρωση της εγγραφής στη βάση δεδομένων.

4.6.3 Profile

Το activity πρέπει να παρουσιάζει όλα τα στοιχεία του ραντεβού με εμφανή και ξεκάθαρο τρόπο ταξινομημένα ανάλογα με την ιδιότητά τους, επίσης πρέπει να δίνει τη δυνατότητα εισαγωγής και τροποποίησης των στοιχείων αυτών μέχρι το σημείο επικύρωσης του ραντεβού.

4.6.4 Organizer

Σαν απαιτήσεις ορίζονται πρώτα η δομή του ραντεβού που πρέπει να περιλαμβάνει: συντεταγμένες μέρους, τύπος μέρους (supermarket, cinema κλπ), διεύθυνση ,ωράριο λειτουργίας μέρους, , μέρα και ώρα έναρξης, διάρκεια, και μόνο σε ραντεβού με τύπο μέρους cinema, τίτλος ταινίας και όνομα κινηματογράφου. Έπειτα υπάρχουν οι τρεις βασικές λειτουργίες που αναφέρθηκαν και στο ρόλο του activity. Το Organizer πρέπει να δίνει τη δυνατότητα προβολής όλων των καταχωρημένων ραντεβού, διαγραφής επιλεγμένων από το χρήστη ραντεβού, και δρομολόγησης ραντεβού επίσης επιλεγμένων από το χρήστη. Επιπλέον πρέπει να υπάρχει κάποιος τρόπος αποθήκευσης και ανάκτησης της λίστας των ραντεβού ώστε να ανταποκρίνεται στα καταχωρημένα ραντεβού του συνδεδεμένου χρήστη. Ισχύουν και πάλι οι απαιτήσεις για εύχρηστο και καθαρό layout.

4.6.5 NewAppointment

Το activity πρέπει να παρουσιάζει όλα τα στοιχεία του ραντεβού με εμφανή και ξεκάθαρο τρόπο ταξινομημένα ανάλογα με την ιδιότητά τους, επίσης πρέπει να δίνει τη δυνατότητα εισαγωγής και τροποποίησης των στοιχείων αυτών μέχρι το σημείο επικύρωσης του ραντεβού.

4.6.6 DateTimePick

Οι απαιτήσεις που ορίζονται για το DateTimePick είναι να δίνει την δυνατότητα επιλογής μέρας και ώρας ραντεβού ενώ παράλληλα ελέγχει την εγκυρότητα της επιλεγμένης μέρας και ώρας η οποία πρέπει να αποστέλλεται με κάποιο τρόπο πίσω στο activity NewAppointment.

4.6.7 MoviePicker

Το activity πρέπει να διαθέτει το κατάλληλο διαδραστικό περιβάλλον ώστε ο χρήστης να μπορέσει να επιλέξει ταινία η οποία να ανταποκρίνεται στους κινηματογράφους στους οποίους προβάλλεται όπως και στις ώρες και μέρες προβολής. Το περιβάλλον αυτό πρέπει να είναι έτσι σχεδιασμένο ώστε να δίνει τις απαραίτητες πληροφορίες που χρειάζεται ο μέσος θεατής για να επιλέξει μια ταινία. Ο χρήστης πρέπει να έχει τη δυνατότητα προγραμματισμού ακριβούς ημερομηνίας ραντεβού cinema μέχρι και δύο μήνες μετά. Επιπλέον πρέπει να γίνονται οι κατάλληλοι έλεγχοι ώστε να αποτραπεί επιλογή μη έγκυρης ώρας ή μέρας προβολής (για παράδειγμα παρελθοντική ημερομηνία).

4.6.8 PlanMap

Το PlanMap απαιτείται να διεξάγει αυστηρό έλεγχο εγκυρότητας οι συνθήκες του οποίου πρέπει να είναι τέτοιες ώστε να μην υπάρχει απολύτως καμία πιθανότητα να καταχωρηθεί ραντεβού που να επικαλύπτεται από άλλο είτε λόγω ώρας έναρξης είτε λόγω χρόνου μετάβασης, επίσης δεν πρέπει να καταχωρηθεί ραντεβού το οποίο αναφέρεται σε παρελθοντική ημερομηνία όπως και ραντεβού για το οποίο δεν προλαβαίνει να φτάσει εγκαίρως ο χρήστης λόγω χρόνου μετάβασης, επίσης πρέπει να λαμβάνονται υπόψη τα ωράρια καταστημάτων και ώρες αιχμής όσον αφορά την κίνηση στους δρόμους. Το PlanMap απαιτείται επίσης εφόσον καθορίσει την εγκυρότητα του ραντεβού να κάνει εύρεση του κοντινότερου ζητούμενου σημείου και να εμφανίσει στον χάρτη τη διαδρομή που πρέπει να ακολουθήσει ο χρήστης οδικώς για να φτάσει ως εκεί. Επιπλέον πρέπει να εγγυάται η διαθεσιμότητα των components του δέκτη GPS καθώς και της σύνδεσης internet προτού γίνει επικοινωνία με την υπηρεσία.

4.6.9 InstantMap

Το InstantMap απαιτείται να δίνει στον χρήστη κάποιο τρόπο εισαγωγής του επιθυμητού τύπου μέρους στο οποίο θέλει να δρομολογηθεί, σε αντίθεση με το PlanMap δεν πρέπει να γίνεται έλεγχος εγκυρότητας ραντεβού. Πρέπει να βρίσκει και να επιστρέφει την διαδρομή που πρέπει να ακολουθήσει ο χρήστης οδικώς για να μεταβεί στο κοντινότερο σημείο του οποίου η εύρεση πρέπει να έχει γίνει μετά από αίτημα του χρήστη για δρομολόγηση στον επιλεγμένο τύπο μέρους. Πρέπει και εδώ να εγγυάται η διαθεσιμότητα των components του δέκτη GPS καθώς και της σύνδεσης internet προτού γίνει επικοινωνία με την υπηρεσία.

5.2 Έλεγχος Εγκυρότητας Ραντεβού

5.2.1 Τι είναι ο Έλεγχος Εγκυρότητας Ραντεβού;

Ο έλεγχος εγκυρότητας ραντεβού πραγματοποιείται στο activity PlanMap και είναι ένα σύνολο από αρνητικές συνθήκες που δεν πρέπει να ικανοποιούνται έτσι ώστε το ραντεβού να είναι έγκυρο που σημαίνει ότι ο χρήστης είναι εφικτό να το διεκπεραιώσει στο διαθέσιμο χρονικό διάστημα από την ώρα καταχώρησης μέχρι την ώρα έναρξης του ραντεβού συνυπολογίζοντας παράγοντες όπως χρόνο μετάβασης, ωράριο καταστημάτων, ώρα αυξημένης κίνησης, κατεύθυνση κ.α. Επίσης η εκπλήρωση του εξεταζόμενου ραντεβού δεν θα πρέπει να εμποδίζει την διεκπεραίωση των υπολοίπων ήδη καταχωρημένων ραντεβού.

5.2.2 Σκοπός του Ελέγχου Εγκυρότητας Ραντεβού

Ο σκοπός του ελέγχου εγκυρότητας είναι να αποτρέψει την καταχώρηση ενός ραντεβού που για οποιοδήποτε λόγο δεν μπορεί να διεκπεραιωθεί από το χρήστη και να αποφευχθεί η παρεμβολή αυτού του ραντεβού στις υπόλοιπες υποχρεώσεις του χρήστη όσον αφορά τα ήδη καταχωρημένα ραντεβού.

5.2.3 Έλεγχος Χρόνου

Ο έλεγχος εγκυρότητας ραντεβού όπως προαναφέρθηκε εξετάζει ένα σύνολο αρνητικών συνθηκών. Αυτές οι συνθήκες θα αναλυθούν εδώ και θα εξεταστεί πως ο έλεγχος όλων αυτών θα οδηγήσει στον καθορισμό της εγκυρότητας του ραντεβού. Σε πρώτο σκέλος θα γίνει καθορισμός των κύριων σεναρίων στα οποία το ραντεβού είναι άκυρο, τέτοια σενάρια είναι:

1) **Ο χρήστης καταχώρησε παρελθοντικό χρονικό σημείο.**
2) **Ο χρήστης δεν προλαβαίνει να μεταβεί στο ραντεβού**
3) **Υπάρχει άλλο ραντεβού την ίδια ή κοντινή ώρα με το επιλεγμένο.**
4) **Το κατάστημα είναι κλειστό για την επιλεγμένη ώρα**

Τίθενται τα στοιχεία:

- **appointTimeStart:** Το χρονικό σημείο έναρξης του τρέχοντος ραντεβού
- **appointTimeEnd:** Το χρονικό σημείο λήξης του τρέχοντος ραντεβού
- **nowTime:** Το τρέχον χρονικό σημείο στο οποίο βρίσκεται ο χρήστης τώρα, το χρονικό σημείο στο οποίο γίνεται ο έλεγχος εγκυρότητας.

1) **Ο χρήστης καταχώρησε παρελθοντικό χρονικό σημείο:** Εδώ υπάρχει το χρονικό σημείο έναρξης του ραντεβού που ονομάζεται appointTimeStart και το χρονικό σημείο το οποίο γίνεται ο έλεγχος εγκυρότητας και ονομάζεται nowTime και αποτελεί την τρέχουσα ώρα του χρήστη.

Από αυτό το σενάριο βγαίνει η πρώτη συνθήκη που αν ισχύει το ραντεβού βγαίνει άκυρο:

$$I)\ appointTimeStart < nowTime$$

2) **Ο χρήστης δεν προλαβαίνει να μεταβεί στο ραντεβού:** Εδώ υπάρχουν και πάλι τα χρονικά σημεία nowTime και appointTime όπως και πριν αλλά εισάγεται ένας νέος παράγοντας, ο χρόνος μετάβασης από τη θέση του χρήστη στο σημείο του τρέχοντος ραντεβού. Από τι εξαρτάται όμως ο χρόνος μετάβασης; Οι παράγοντες που επηρεάζουν τον χρόνο μετάβασης είναι η απόσταση και η μέση ταχύτητα του χρήστη.

Τίθενται τα εξής στοιχεία:

- **appointTimeStart:** Το χρονικό σημείο έναρξης του τρέχοντος ραντεβού
- **appointTimeEnd:** Το χρονικό σημείο λήξης του τρέχοντος ραντεβού
- **nowTime:** Το τρέχον χρονικό σημείο στο οποίο βρίσκεται ο χρήστης τώρα, το χρονικό σημείο στο οποίο γίνεται ο έλεγχος εγκυρότητας.
- **A:** Η τωρινή θέση του χρήστη στο χώρο
- **B:** Η θέση του μέρους του τρέχοντος ραντεβού στο χώρο
- **d(AB):** Η απόσταση του χρήστη από το σημείο του τρέχοντος ραντεβού
- **t(AB):** Ο χρόνος μετάβασης του χρήστη από το σημείο Α στο σημείο Β
- **v:** Η μέση ταχύτητα του χρήστη.

Η μέση ταχύτητα του χρήστη εξαρτάται από την τρέχουσα ώρα, αν η τρέχουσα ώρα είναι μεταξύ 8:00 και 9:00 το πρωί ή μεταξύ 13:00 και 15:00 το μεσημέρι που παρατηρείται αυξημένη κίνηση, τότε η μέση ταχύτητα όπως έχει προσεγγιστεί με βάση το άρθρο Scientific American: Cities where it's faster to walk than drive. by Tali Trigg είναι 14χλμ/ώρα διαφορετικά όλες τις άλλες ώρες η μέση ταχύτητα υπολογίζεται προσεγγιστικά στα 25χλμ/ώρα

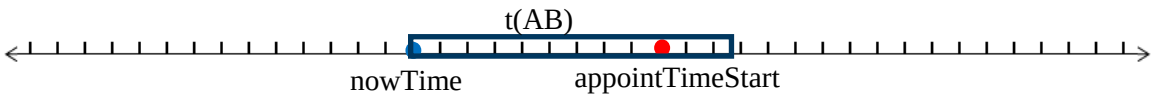
$$\text{Αν } 8{:}00 \le \text{nowTime} \le 9{:}00 \text{ ή } 13{:}00 \le \text{nowTime} \le 15{:}00 \text{ τότε}$$
$$\text{v} = 14\text{χλμ/ώρα}$$
$$\text{Αλλιώς v} = 25\text{χλμ/ώρα}$$

Άρα ο χρόνος μετάβασης εξαρτάται από την απόσταση d(AB) και την τρέχουσα ώρα.
Ο τύπος υπολογισμού του χρόνου μετάβασης είναι

$$t(AB) = d(AB)/v$$

Με γνωστό τον τρόπο υπολογισμού του χρόνου μετάβασης από τη θέση του χρήστη στο τρέχον ραντεβού t(AB) μπορεί να βγει η συνθήκη κατά την οποία ο χρήστης δεν προλαβαίνει να φτάσει εγκαίρως στο ραντεβού το οποίο θεωρείται μη έγκυρο:

$$II)\ \big(nowTime + t(AB)\big) > appointTime$$



3) **Υπάρχει άλλο ραντεβού την ίδια ή κοντινή ώρα με το επιλεγμένο:** Εδώ διακρίνονται οι περισσότερες συνθήκες, η ικανοποίηση των οποίων οδηγεί σε ακύρωση του ραντεβού. Αρχικά καθορίζεται ότι υπάρχει ένα δεύτερο ραντεβού που με κάποιο τρόπο εμποδίζει τη διεκπεραίωση του τρέχοντος ραντεβού και είναι ήδη καταχωρημένο.

Τίθενται τα εξής στοιχεία:

- **appointTimeStart:** Το χρονικό σημείο έναρξης του τρέχοντος ραντεβού.
- **appointTimeEnd:** Το **χρονικό** σημείο λήξης του τρέχοντος ραντεβού.
- **appointDuration:** Η **διάρκεια** του τρέχοντος ραντεβού
- **appointTime2Start:** Το **χρονικό** σημείο έναρξης του δεύτερου ραντεβού.
- **appointTime2End:** Το **χρονικό** σημείο λήξης του δεύτερου ραντεβού.
- **appoint2Duration:** Η **διάρκεια** του τρέχοντος ραντεβού
- **Α:** Η τωρινή θέση του **χρήστη** στο χώρο
- **Β:** Η θέση του μέρους του **τρέχοντος** ραντεβού στο χώρο
- **Γ:** Η θέση του μέρους του δεύτερου ραντεβού στο χώρο
- **d(ΑΒ):** Η απόσταση του **χρήστη** από το σημείο του τρέχοντος ραντεβού
- **d(ΑΓ):** Η απόσταση του **χρήστη** από το σημείο του δεύτερου ραντεβού
- **d(ΒΓ):** Η απόσταση από τη θέση του τρέχοντος ραντεβού μέχρι τη θέση του δεύτερου ραντεβού
- **t(ΑΒ):** Ο χρόνος **μετάβασης** του χρήστη από την αρχική του θέση μέχρι το σημείο του τρέχοντος ραντεβού
- **t(ΑΓ):** Ο χρόνος **μετάβασης** του χρήστη από την αρχική του θέση μέχρι το σημείο του δεύτερου ραντεβού
- **t(ΒΓ):** Ο χρόνος **μετάβασης** από τη θέση του τρέχοντος ραντεβού μέχρι τη θέση του δεύτερου ραντεβού
- **v:** Η μέση ταχύτητα του **χρήστη** ανάλογα με την τρέχουσα ώρα.

Κώδικας υπολογισμού χρόνου μετάβασης

```
public int calculateTravelTime(LatLng origin, LatLng destination,
boolean rushHour) {
    int travelTime;
    double avgSpeed, distance;
    if (rushHour) {
        avgSpeed = 3.88888;
    }         // rush hour avg speed 14km/h == 233.3333m/min  ==
3.8888m/sec
    else {
        avgSpeed = 6.94444;          //normal avg speed 25km/h ==
416.6666m/min ==6.94444m/sec
    }
    distance = distFrom(origin, destination);
    travelTime = (int) Math.ceil(distance / avgSpeed);

    return travelTime;
}

mainToSecondTravelTime = calculateTravelTime(destinationLatLng,
receivedAppointListLatLng, isRushHour);
originToSecondTravelTime = calculateTravelTime(origin,
receivedAppointListLatLng, isRushHour);
```

Ορισμός βασικών μεταβλητών κώδικα:

Επειδή η δημιουργία των μεταβλητών είναι σχετικά πολύπλοκη και χρειάζεται παράθεση ποικίλων κομματιών κώδικα σε διαφορετικά σημεία, πράγμα που ενδέχεται να εισάγει αχρείαστη πολυπλοκότητα όσον αφορά την κατανόηση της μεθόδου, για το λόγο αυτόν θα γίνει μια σύντομη περιγραφή των ρόλων των μεταβλητών που χρησιμοποιούνται.

listDateTimeFrame[0]: Το χρονικό σημείο έναρξης του δεύτερου ραντεβού επεκταμένο κατά 5 λεπτά προβλεπόμενα για χρόνο στάθμευσης

listDateTimeFrame[1]: Το χρονικό σημείο λήξης του δεύτερου ραντεβού επεκταμένο κατά 5 λεπτά προβλεπόμενα για χρόνο ξεπαρκαρίσματος.

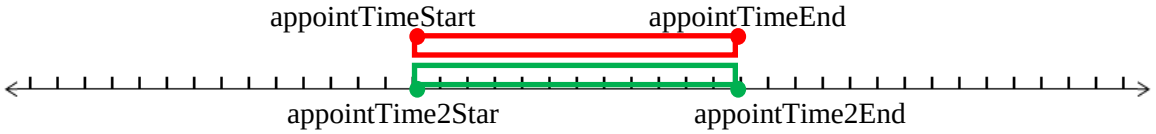
appointTimeFrame[0]: Το χρονικό σημείο έναρξης του τρέχοντος ραντεβού επεκταμένο κατά 5 λεπτά προβλεπόμενα για χρόνο στάθμευσης.

appointTimeFrame[1]: Το χρονικό σημείο λήξης του τρέχοντος ραντεβού επεκταμένο κατά 5 λεπτά προβλεπόμενα για χρόνο ξεπαρκαρίσματος.

Έχοντας αυτά τα στοιχεία αναλύονται οι υπόλοιπες συνθήκες όπου:

4) Το τρέχον ραντεβού να έχει ακριβώς τα ίδια χρονικά σημεία έναρξης και λήξης

$$III)\ appointTimeStart = appointTime2Start \ \ \boldsymbol{και}\ \ appointTimeEnd = appointTime2End$$

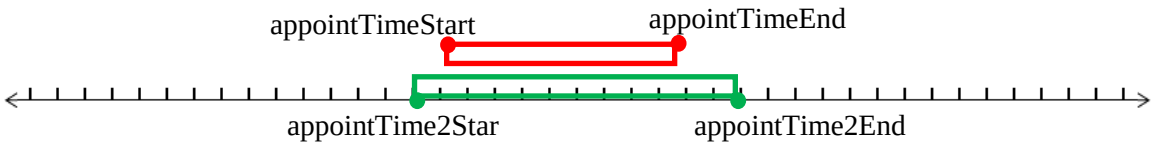


Κώδικας συνθήκης

```
appSame = appointTimeFrame[0] == listDateTimeFrame[0] &&
appointTimeFrame[1] == listDateTimeFrame[1];
```

5) Το τρέχον ραντεβού να έχει χρονικό σημείο έναρξης μετά την έναρξη του δεύτερου ραντεβού και χρονικό σημείο λήξης πριν την λήξη του δεύτερου ραντεβού

$$IV)\ appointTimeStart > appointTime2Start \ \ \boldsymbol{και}\ \ appointTimeEnd < appointTime2End$$

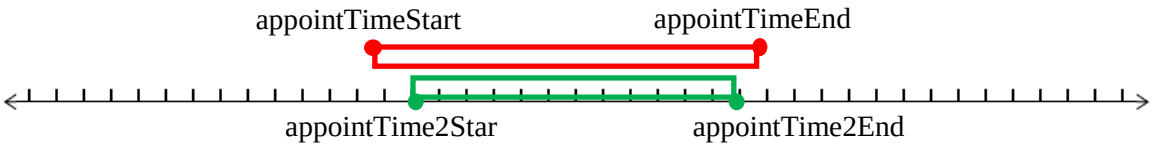


Κώδικας συνθήκης

```
appInside = listDateTimeFrame[0].before(appointTimeFrame[0]) && ap-
pointTimeFrame[0].before(listDateTimeFrame[1]) && ap-
pointTimeFrame[1].after(listDateTimeFrame[0]) && ap-
pointTimeFrame[1].before(listDateTimeFrame[1]);
```

6) **Το τρέχον ραντεβού να έχει χρονικό σημείο έναρξης πριν την έναρξη του δεύτερου ραντεβού και χρονικό σημείο λήξης μετά την λήξη του δεύτερου ραντεβού**

$$V)\ appointTimeStart < appointTime2Start \ \ \textbf{και} \ \ appointTimeEnd > appointTime2End$$

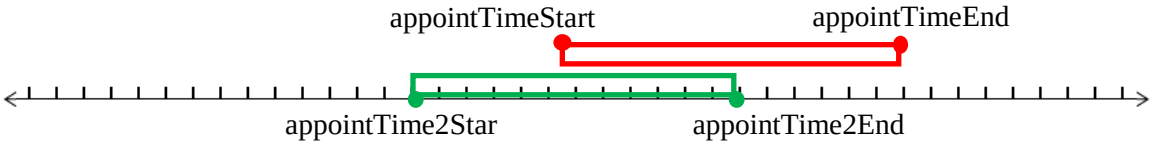


Κώδικας συνθήκης

```
appInclusive = listDateTimeFrame[0].after(appointTimeFrame[0]) && list-
DateTimeFrame[0].before(appointTimeFrame[1]) && list-
DateTimeFrame[1].after(appointTimeFrame[0]) && list-
DateTimeFrame[1].before(appointTimeFrame[1]);
```

7) **Το δεύτερο ραντεβού να έχει χρονικό σημείο λήξης μετά την έναρξη και πριν τη λήξη του τρέχοντος ραντεβού.**

$$VI)\ appointStart < appointTime2End < appointEnd$$

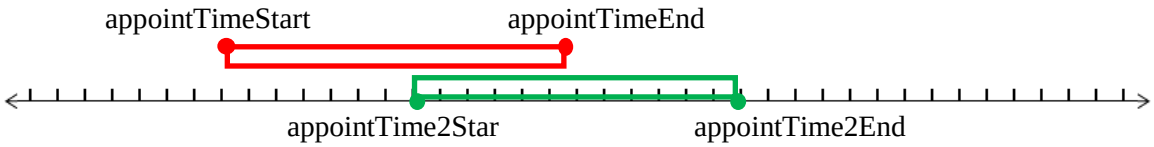


Κώδικας συνθήκης

```
appBeforeWithoutTravelTime = list-
DateTimeFrame[1].before(appointTimeFrame[1]) && ap-
pointTimeFrame[0].before(listDateTimeFrame[1]);
```

8) **Το δεύτερο ραντεβού να έχει χρονικό σημείο έναρξης μετά την έναρξη και πριν τη λήξη του τρέχοντος ραντεβού.**

$$VII)\ appointStart < appointTime2Start < appointEnd$$



Κώδικας συνθήκης

```
appAfterWithoutTravelTime = list-
DateTimeFrame[0].after(appointTimeFrame[0]) && list-
DateTimeFrame[0].before(appointTimeFrame[1]);
```

Ορίζονται επίσης οι περιπτώσεις με τις οποίες σχετίζεται ο χρόνος μετάβασης:

- Τα ραντεβού επικαλύπτονται ακόμα και αν δεν συνυπολογιστεί ο χρόνος μετάβασης:
  Εδώ υπάγονται οι περιπτώσεις 4) και 5) που αναφέρθηκαν παραπάνω. Πρόκειται για περιπτώσεις που ο υπολογισμός του χρόνου μετάβασης είναι περιττός αφού σε κάθε περίπτωση το ραντεβού κρίνεται άκυρο.
- Τα ραντεβού επικαλύπτονται μόνο όταν συνυπολογιστεί ο χρόνος μετάβασης:
  Σε αυτές τις περιπτώσεις τα ραντεβού τα ίδια δεν επικαλύπτονται αλλά ο χρόνος μετάβασης δεν επαρκεί για να μεταβεί ο χρήστης από το ένα στο άλλο.

Τίθενται τα στοιχεία:

- **appointArriveTime:** Το χρονικό σημείο άφιξης χρήστη στο τρέχον ραντεβού. Υπολογίζεται από το τρέχον χρονικό σημείο του χρήστη (nowTime) και το χρόνο μετάβασης από την τρέχουσα θέση του χρήστη στη θέση του τρέχοντος ραντεβού t(AB):

$$\text{appointArriveTime} = nowTime + t(AB)$$

```
Calendar cal = Calendar.getInstance();
cal.setTime(listDateTimeFrame[1]);
cal.add(Calendar.SECOND, mainToSecondTravelTime);
arrivalMainBefore = cal.getTime();
```

- **app2BeforeAppointArrive:** Το χρονικό σημείο άφιξης χρήστη στο δεύτερο ραντεβού όταν το δεύτερο ραντεβού είναι πριν το τρέχον ραντεβού. Το σημείο αυτό υπολογίζεται από το τρέχον χρονικό σημείο του χρήστη (nowTime) και τον χρόνο μετάβασης από τη τρέχουσα θέση του χρήστη μέχρι τη θέση του δεύτερου ραντεβού t(ΑΓ):

$$\text{app2BeforeAppointArriveTime} = nowTime + t(A\Gamma),$$
$$\text{όταν } nowTime < appoint2Start < appointTimeStart$$

```
Calendar cal = Calendar.getInstance();
cal.setTime(nowTime);
cal.add(Calendar.SECOND, originToSecondTravelTime);
arrivalSecondBefore = cal.getTime();
```

- **app2AfterAppointArrive:** Το χρονικό σημείο άφιξης χρήστη στο δεύτερο ραντεβού όταν το δεύτερο ραντεβού είναι μετά το τρέχον ραντεβού. Το σημείο αυτό υπολογίζεται από τη λήξη του τρέχοντος ραντεβού (appointEnd) και τον χρόνο μετάβασης από τη θέση του τρέχοντος ραντεβού μέχρι τη θέση του δεύτερου ραντεβού t(ΒΓ):

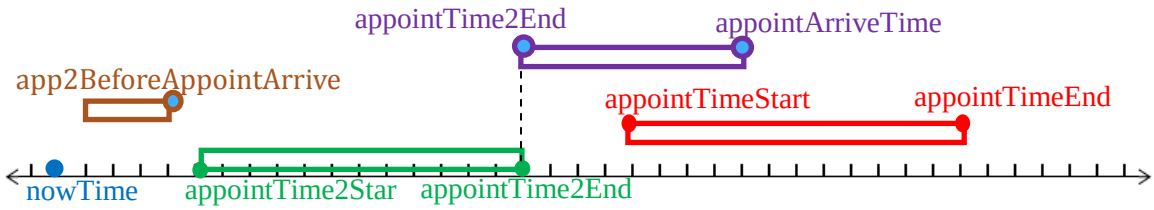
$$\text{app2BeforeAppointArriveTime} = appointEnd + t(B\Gamma),$$
$$\text{όταν appointTimeEnd} < appoint2Start$$

```
Calendar cal = Calendar.getInstance();
cal.setTime(appointTimeFrame[1]);
cal.add(Calendar.SECOND, mainToSecondTravelTime);
arrivalSecondAfter = cal.getTime();
```

Διακρίνονται οι περιπτώσεις:

9) Το χρονικό σημείο άφιξης χρήστη στο τρέχον ραντεβού να είναι μετά την έναρξη του τρέχοντος ραντεβού. Η έναρξη του δεύτερου ραντεβού να είναι πριν την έναρξη του τρέχοντος ραντεβού και μετά το τρέχον χρονικό σημείο του χρήστη. Το χρονικό σημείο άφιξης χρήστη στο δεύτερο ραντεβού να είναι πριν το χρονικό σημείο έναρξης του δεύτερου ραντεβού.

$$IX)\ appointTimeStart < \text{appointArriveTime}$$
$$\textbf{και}$$
$$nowTime < appointTime2Start < appointTimeStart$$
$$\textbf{και}$$
$$appointTime2Start < \text{app2BeforeAppointArrive}$$

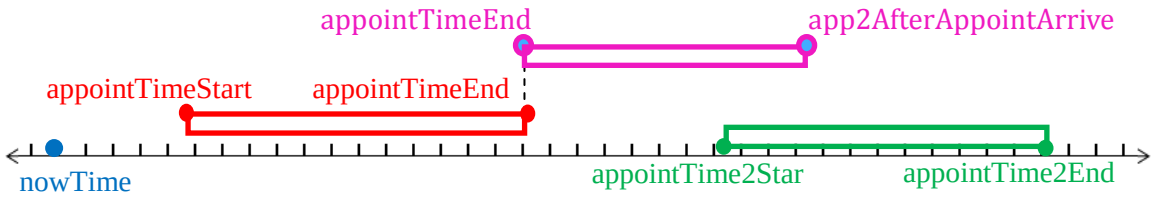


Συνθήκη μέσα στον κώδικα:

```
appBeforeWithTravelTime =
arrivalMainBefore.after(appointTimeFrame[0]) &&
listDateTimeFrame[0].before(appointTimeFrame[0]) &&
listDateTimeFrame[0].after(nowTime) &&
arrivalSecondBefore.before(listDateTimeFrame[0]);
```

10) Το χρονικό σημείο έναρξης του δεύτερου ραντεβού να είναι μετά το τρέχον χρονικό σημείο του χρήστη και μετά τη λήξη του τρέχοντος ραντεβού. Το χρονικό σημείο άφιξης χρήστη στο δεύτερο ραντεβού να είναι μετά την έναρξη του δεύτερου ραντεβού.

$$X)\ appointTimeEnd < appointTime2Start$$
$$\boldsymbol{και}$$
$$nowTime < appointTime2Start$$
$$\boldsymbol{και}$$
$$appointTime2Start < \text{app2AfterAppointArrive}$$



Συνθήκη μέσα στον κώδικα:

```
arrivalSecondAfter.after(listDateTimeFrame[0]) &&
listDateTimeFrame[0].after(appointTimeFrame[1]) &&
listDateTimeFrame[0].after(nowTime);
```

5.2.4 Έλεγχος Κατεύθυνσης

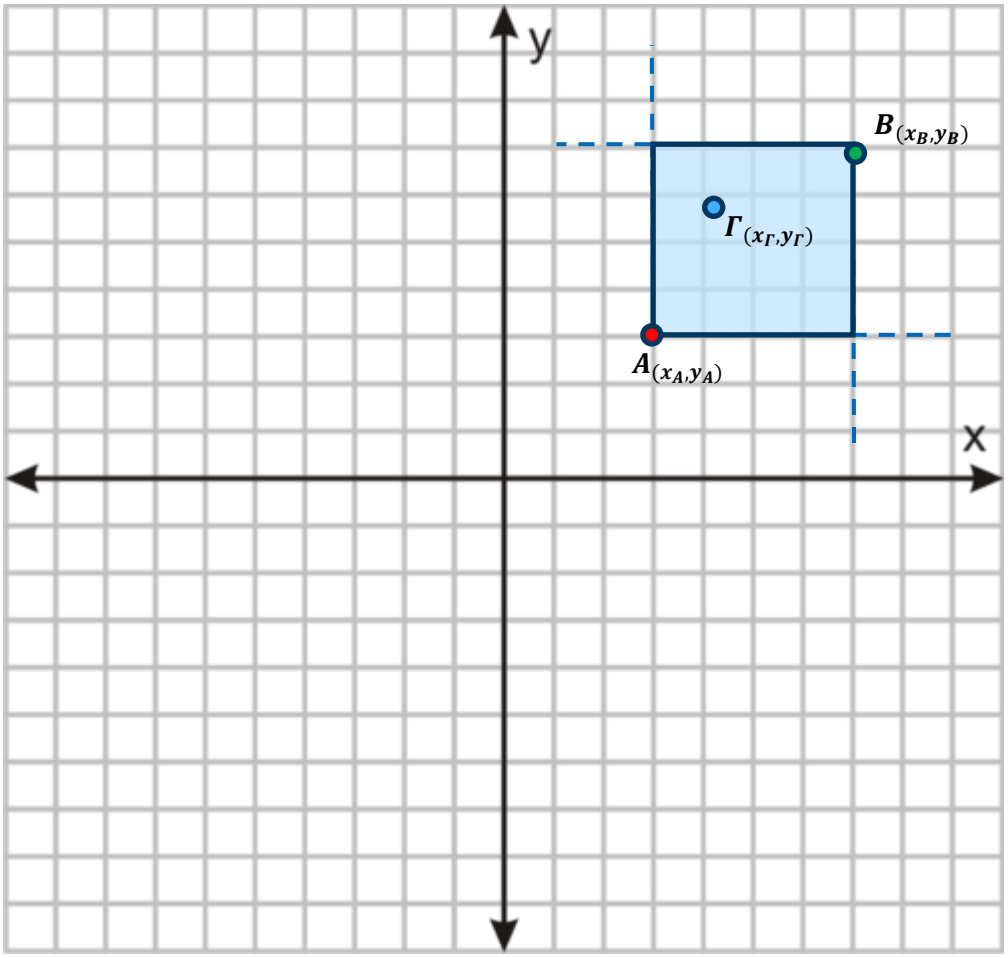
Στις περιπτώσεις 7) και 8) όταν δηλαδή τα ραντεβού δεν επικαλύπτονται κανονικά αλλά δεν επαρκεί ο χρόνος μετάβασης από το ένα στο άλλο, ζητείται η εύρεση κοντινότερης θέσης στο χάρτη ως προς τη θέση του δεύτερου ραντεβού (αντί για την τρέχουσα θέση του χρήστη που ήδη ελέγξαμε). Ο νέος προορισμός πρέπει να βρίσκεται σε τέτοια θέση που να μην ικανοποιούνται οι συνθήκες 7) και 8) . Επίσης πρέπει να βρίσκεται στην ίδια κατεύθυνση με το δεύτερο ραντεβού ώστε ο χρήστης να μη βγει από το δρόμο του. Αν δεν βρεθεί τέτοιος προορισμός τότε γίνεται ξανά αναζήτηση κοντινότερης θέσης στο χάρτη ως προς τη θέση του δεύτερου ραντεβού χωρίς τον περιορισμό για ίδια κατεύθυνση με την πορεία του χρήστη. Εάν βρεθεί τέτοιος προορισμός ο χρήστης λαμβάνει το κατάλληλο μήνυμα και επιλέγει αν θέλει να βγει από το δρόμο του προλαβαίνοντας έτσι και τα δύο ραντεβού ή αν θέλει να ακυρώσει την καταχώρηση του τρέχοντος ραντεβού.

Ο έλεγχος εγκυρότητας του νέου προορισμού ως προς την κατεύθυνση γίνεται ως εξής:

- x: Γεωγραφικό Πλάτος
- y: Γεωγραφικό Μήκος
- Έστω x_A , y_A οι συντεταγμένες του σημείου Α το οποίο είναι η τρέχουσα θέση του χρήστη στο χώρο.
- Έστω x_B , y_B οι συντεταγμένες του σημείου Β το οποίο είναι η θέση του δεύτερου ραντεβού στο χώρο.
- Έστω $x_Γ$, $y_Γ$ οι συντεταγμένες του σημείου Γ το οποίο είναι η θέση του νέου προορισμού για το τρέχον ραντεβού στο χώρο.

Για να βρίσκεται το σημείο Γ στην ίδια κατεύθυνση που θα ακολουθήσει ο χρήστης για να μεταβεί από την τρέχουσα θέση του (Α) στη θέση του δεύτερου ραντεβού (Β) πρέπει να ισχύουν οι παρακάτω συνθήκες.

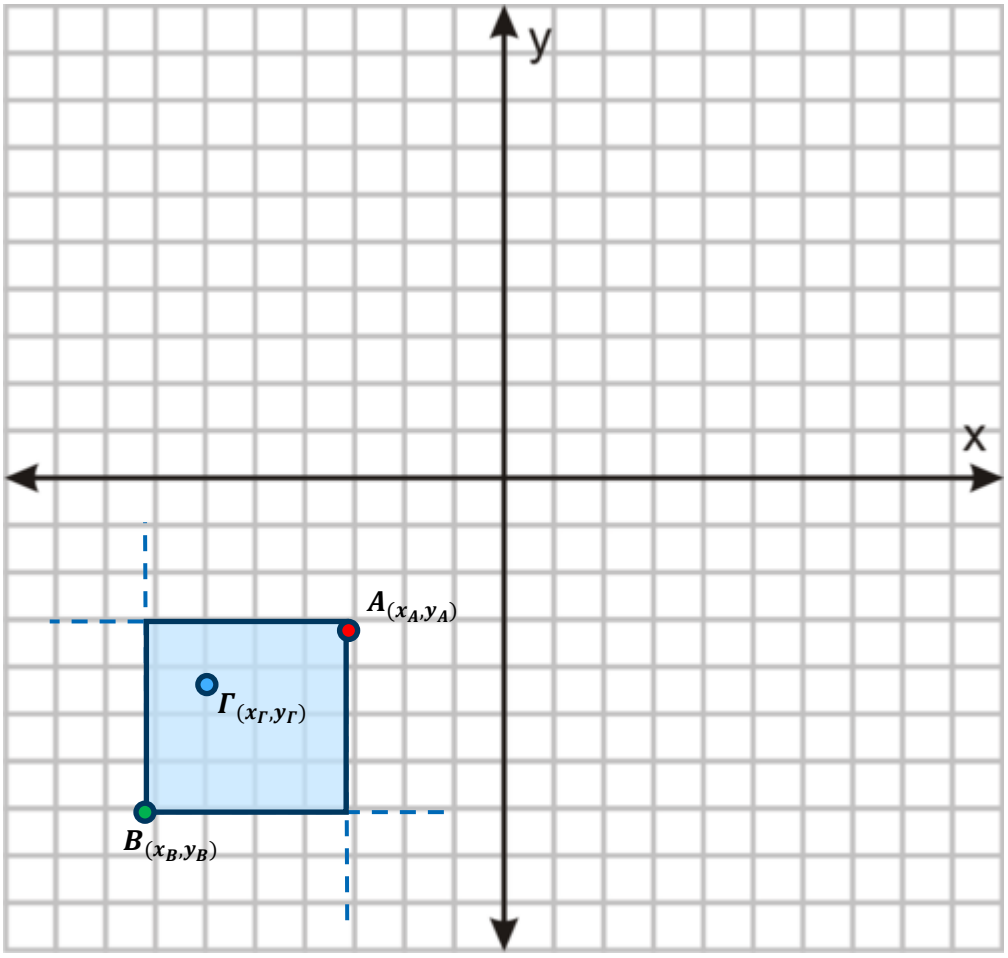
$$Αν \;\; x_A < x_B \;\; και \;\; y_A < \;\; y_B \;\; τότε \;\; πρέπει \; να \; ισχύει$$
$$x_A \leq x_Γ \leq \; x_B \;\; και \;\; y_A \leq y_Γ \leq \; y_B$$



Συνθήκη μέσα στον κώδικα:

```
boolean xyPosB = xB > xA && yB > yA && xC >= xA && xC <= xB && yC >= yA
&& yC <= yB;        // if(xA<xB) and yA<yB then for secondary to be in
the same direction: xA<=xC<=xB  and yA<=yC<=yB  must be true
```

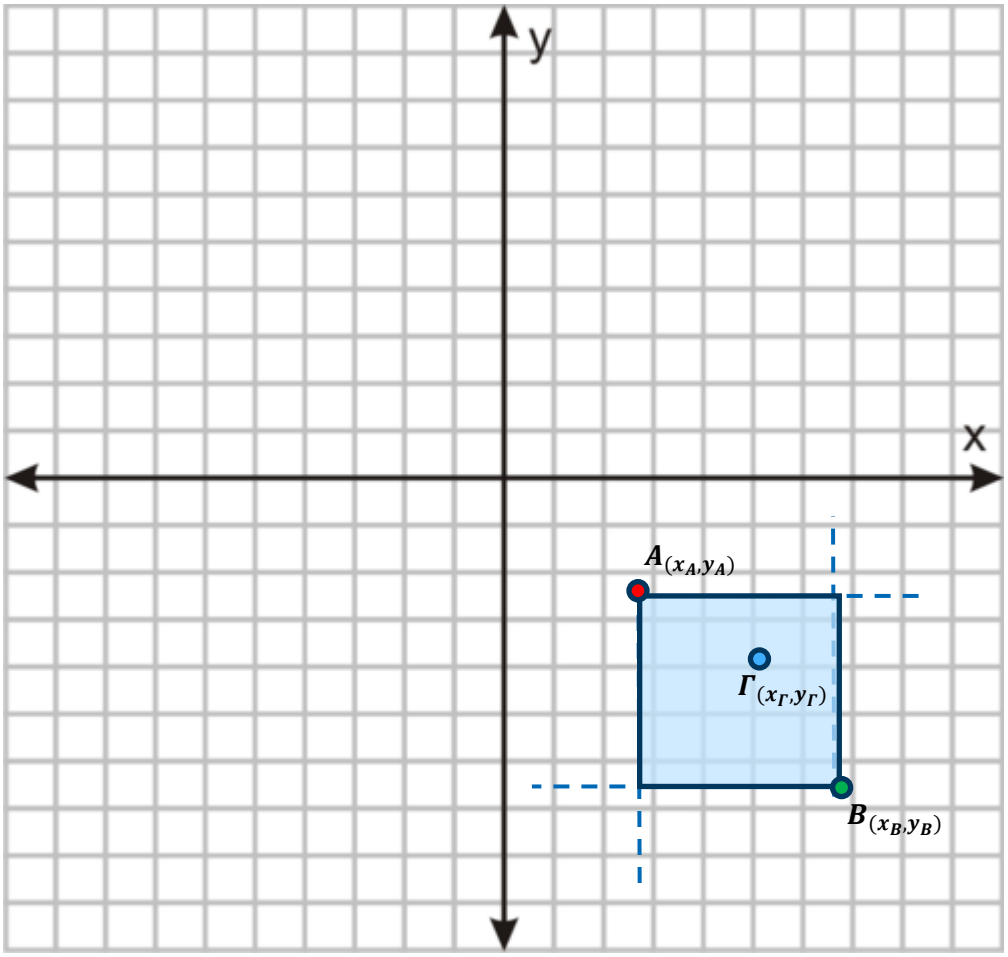
$$Αν \;\; x_A > x_B \;\; και \;\; y_A > \;\; y_B \;\; τότε \;\; πρέπει \; να \; ισχύει$$
$$x_B \leq x_Γ \leq \; x_A \;\; και \;\; y_B \leq y_Γ \leq \; y_A$$



Συνθήκη μέσα στον κώδικα:

```
boolean xyNegB = xB < xA && yB < yA && xC <= xA && xC >= xB && yC <= yA
&& yC >= yB;       // if(xA>xB) and yA>yB then for secondary to be in
the same direction: xB<=xC<=xA  and yB<=yC<=yA  must be true
```

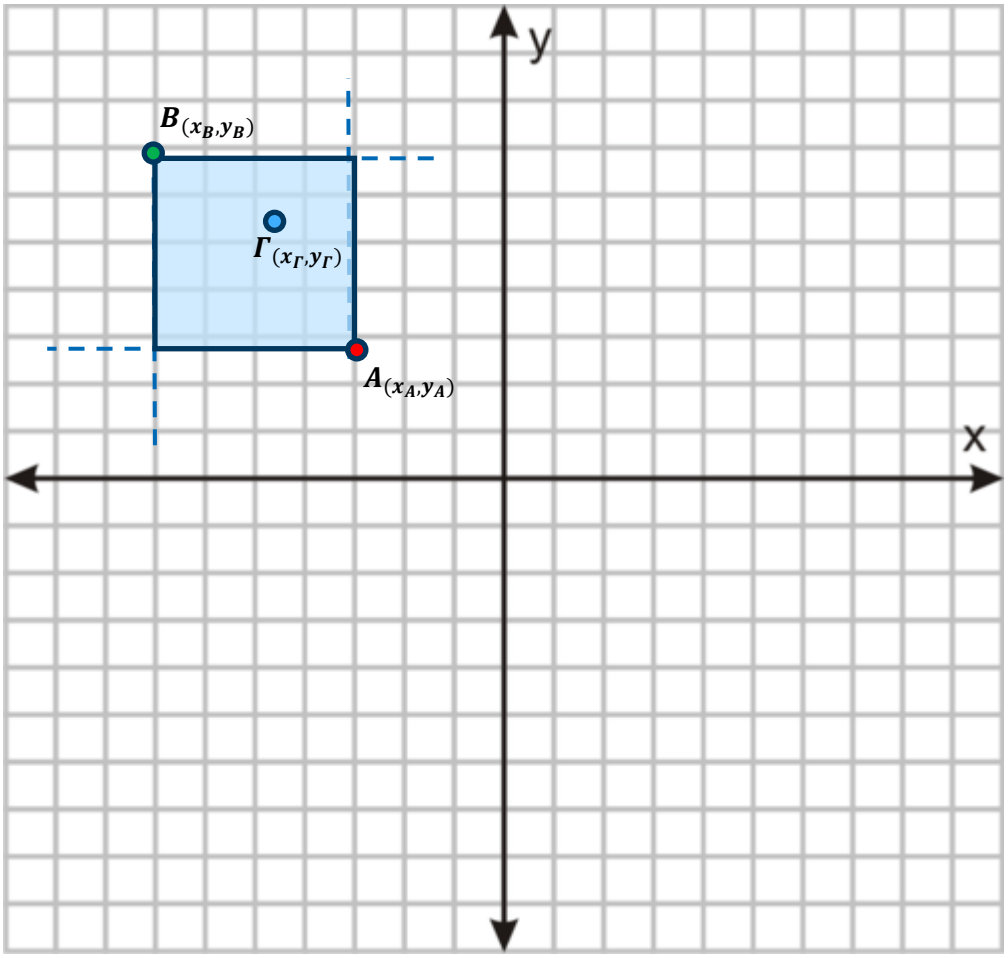
$$Αν \;\; x_A < x_B \;\; και \;\; y_A > \;\; y_B \;\; τότε \;\; πρέπει \; να \; ισχύει$$
$$x_A \leq x_Γ \leq \; x_B \;\; και \;\; y_B \leq y_Γ \leq \; y_A$$



Συνθήκη μέσα στον κώδικα:

```
boolean xPosyNegB = xB > xA && yB < yA && xC >= xA && xC <= xB && yC <=
yA && yC >= yB;     // if(xA<xB) and yA>yB then for secondary to be in
the same direction: xA<=xC<=xB  and yB<=yC<=yA  must be true
```

$$Αν \;\; x_A > x_B \;\; και \;\; y_A < \;\; y_B \;\; τότε \;\; πρέπει \; να \; ισχύει$$
$$x_B \leq x_Γ \leq \; x_A \;\; και \;\; y_A \leq y_Γ \leq \; y_B$$



Συνθήκη μέσα στον κώδικα:

```
boolean xNegyPosB = xB < xA && yB > yA && xC <= xA && xC >= xB && yC >=
yA && yC <= yB;     // if(xA>xB) and yA<yB then for secondary to be in
the same direction: xB<=xC<=xA  and yA<=yC<=yB  must be true
```

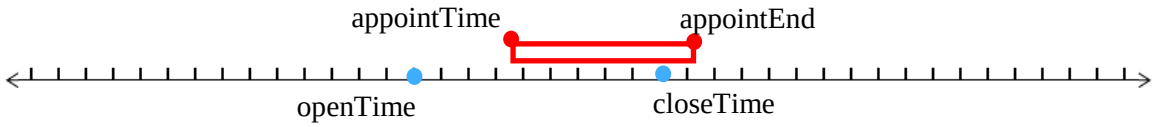
5.2.5 Έλεγχος Ωραρίου

4) Το κατάστημα είναι κλειστό για την επιλεγμένη ώρα: Η τελική συνθήκη ελέγχει αν το μέρος (supermarket/gas station/cinema) είναι ανοιχτό για την επιλεγμένη ώρα και αντιθέτως με τις προηγούμενες αποτελεί μια θετική συνθήκη, που σημαίνει πως αν ισχύει το ραντεβού είναι έγκυρο όσον αφορά το ωράριο, κάτι που δεν αποκλείει όμως το ραντεβού από το να κριθεί άκυρο από τις παραπάνω συνθήκες.
Τίθενται τα εξής στοιχεία:

- **openHour:** Ώρα ανοίγματος καταστήματος/μέρους.
- **closeHour:** Ώρα κλεισίματος καταστήματος/μέρους.
- **appointTime:** χρονικό σημείο άφιξης χρήστη για το τρέχον ραντεβού.
- **appointEnd:** χρονικό σημείο λήξης του τρέχοντος ραντεβού.

Το ραντεβού για να είναι έγκυρο πρέπει να ισχύει:

$$openTime < appointTime < closeTime$$
$$\boldsymbol{και}$$
$$openTime < appointTime < closeTime$$



Επειδή το αντικείμενο Date καταχωρεί εκτός από ώρα και ημερομηνία, αν το closeTime είναι πριν το openTime, που πρακτικά σημαίνει πως το μέρος είναι ανοιχτό κατά τα μεσάνυχτα με την αλλαγή της μέρας, τότε η συνθήκη θα είναι λανθασμένη αφού το openTime θα συγκρίνεται με το closeTime της ίδιας μέρας. Γι αυτό το λόγο έχει υλοποιηθεί ένα επιπλέον "if" που προσθέτει μια μέρα στο date της closeTime εάν ισχύει $closeTime < openTime$.

Υπάρχουν επίσης και τα μέρη που είναι ανοιχτά 24 ώρες, σε αυτά το openTime και το closeTime είναι τα ίδια, οπότε η τελική συνθήκη γίνεται ως εξής:

$$openTime < appointTime < closeTime \quad \textbf{και} \quad openTime < appointTime < closeTime \quad \text{ή} \quad openTime = closeTime$$

Κώδικας ελέγχου ωραρίου:

```
public boolean checkIfopen(Place testedPlace, Date appointTime, Date
appointEnd) {
    Calendar calendar = Calendar.getInstance();
    Date openTime, closeTime;
    calendar.setTime(appointTime);
    calendar.set(Calendar.HOUR_OF_DAY, testedPlace.getOpenHour());
    openTime = calendar.getTime();
    calendar.setTime(appointTime);
    calendar.set(Calendar.HOUR_OF_DAY, testedPlace.getCloseHour());
    closeTime = calendar.getTime();
    if (closeTime.before(openTime)) {
        calendar.add(Calendar.DATE, 1);
        closeTime = calendar.getTime();
    }
    boolean isOpen = appointTime.after(openTime) &&
appointTime.before(closeTime) && appointEnd.after(openTime) &&
appointEnd.before(closeTime) || closeTime.equals(openTime);
    return isOpen;
}
```

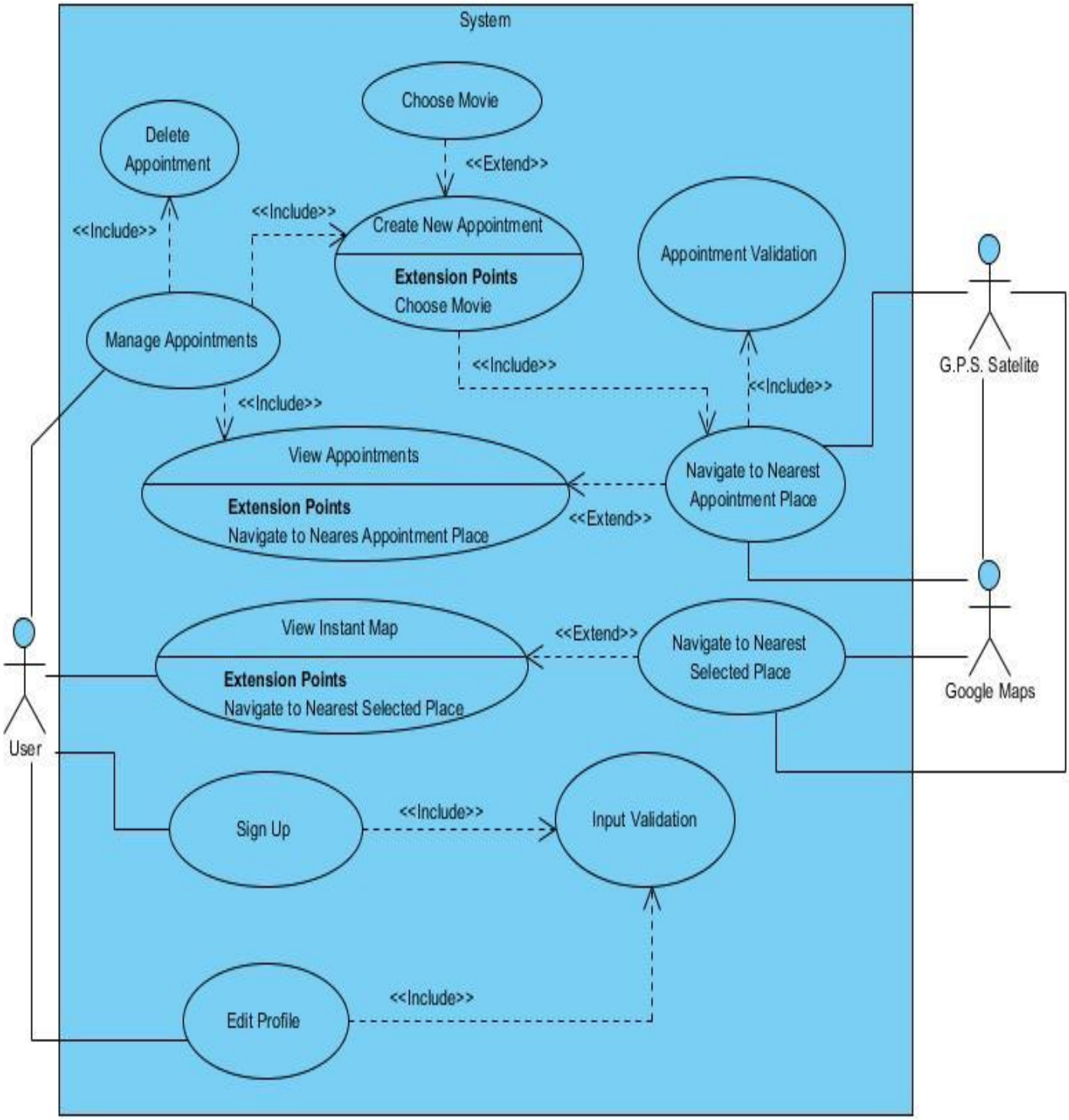
4.7 Ανάλυση Σεναρίων Χρήσης

Ένα από τα δυσκολότερα κομμάτια στον σχεδιασμό εφαρμογών είναι η ανάλυση σεναρίων χρήσης κάτι που οφείλεται σε μεγάλο μέρος στην απρόβλεπτη φύση του ανθρώπινου παράγοντα που χαρακτηρίζει τον χρήστη. Το δύσκολο κομμάτι λοιπόν κατά την σχεδίαση είναι η πρόβλεψη όλων των πιθανών συμπεριφορών του χρήστη που λόγω πολλών παραγόντων (απειρία, περιορισμένη εξοικείωση με το αντικείμενο κ.α.) μπορεί να οδηγήσει σε σενάρια χρήσης τα οποία ο προγραμματιστής σαν καταρτισμένος χρήστης δεν θα σκεφτόταν καν να εκτελέσει. Η δυσκολία λοιπόν στη σχεδίαση σεναρίων χρήσης έγκειται στην διαδικασία να μπει ο σχεδιαστής/προγραμματιστής στην νοοτροπία ενός άπειρου ή και κακόβουλου χρήστη και να προβλέψει ποιες ενέργειες μπορεί να εκτελέσει ώστε να περιορίσει και τις κατάλληλες λειτουργίες τις εφαρμογής να τον καθοδηγήσουν στην σωστή λειτουργικότητα και κυρίως να αποφευχθούν λογικά λάθη στη δομή της εφαρμογής που μπορούν ενδεχομένως να προκληθούν από κακή χρήση.

Τα βασικά σενάρια χρήσης της συγκεκριμένης εφαρμογής σχετίζονται άμεσα με τις βασικές λειτουργίες που αναφέρθηκαν στο κεφάλαιο 4.2. Οι βασικές χρήσεις διακρίνονται σε:

- Δημιουργία Λογαριασμού
- Προβολή Λογαριασμού
- Επεξεργασία Λογαριασμού
- Άμεση Δρομολόγηση
- Δημιουργία Ραντεβού
- Προβολή υπαρχόντων ραντεβού
- Δρομολόγηση σε υπάρχων ραντεβού
- Διαγραφή ραντεβού

Στο παρακάτω Use Case Diagram απεικονίζονται τα βασικότερα σενάρια χρήσης της εφαρμογής.

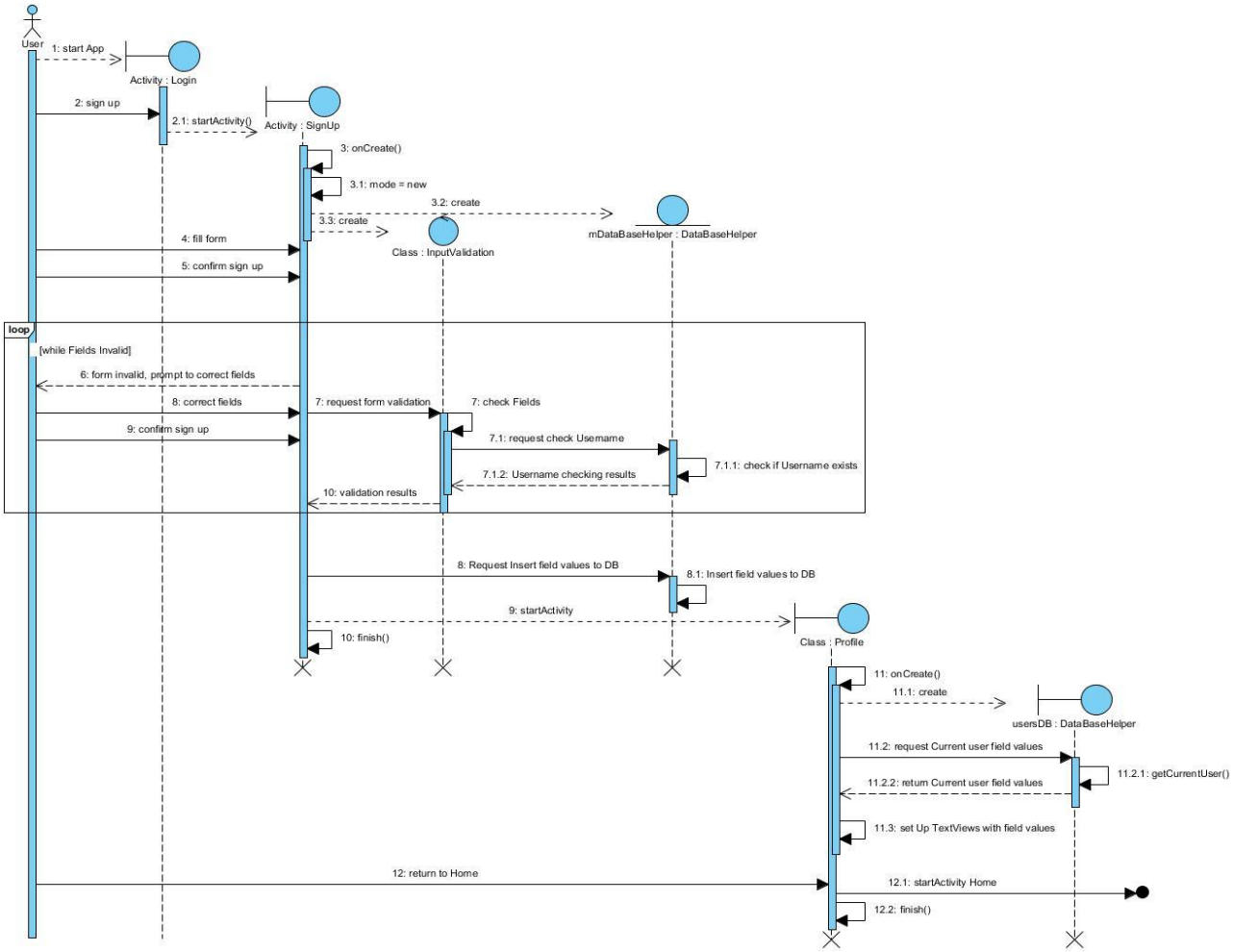


General Use Case Diagram - Figure 1

Παρακάτω θα πραγματοποιηθεί η ανάλυση των πιο βασικών σεναρίων χρήσης με τη βοήθεια των Sequence Diagrams.

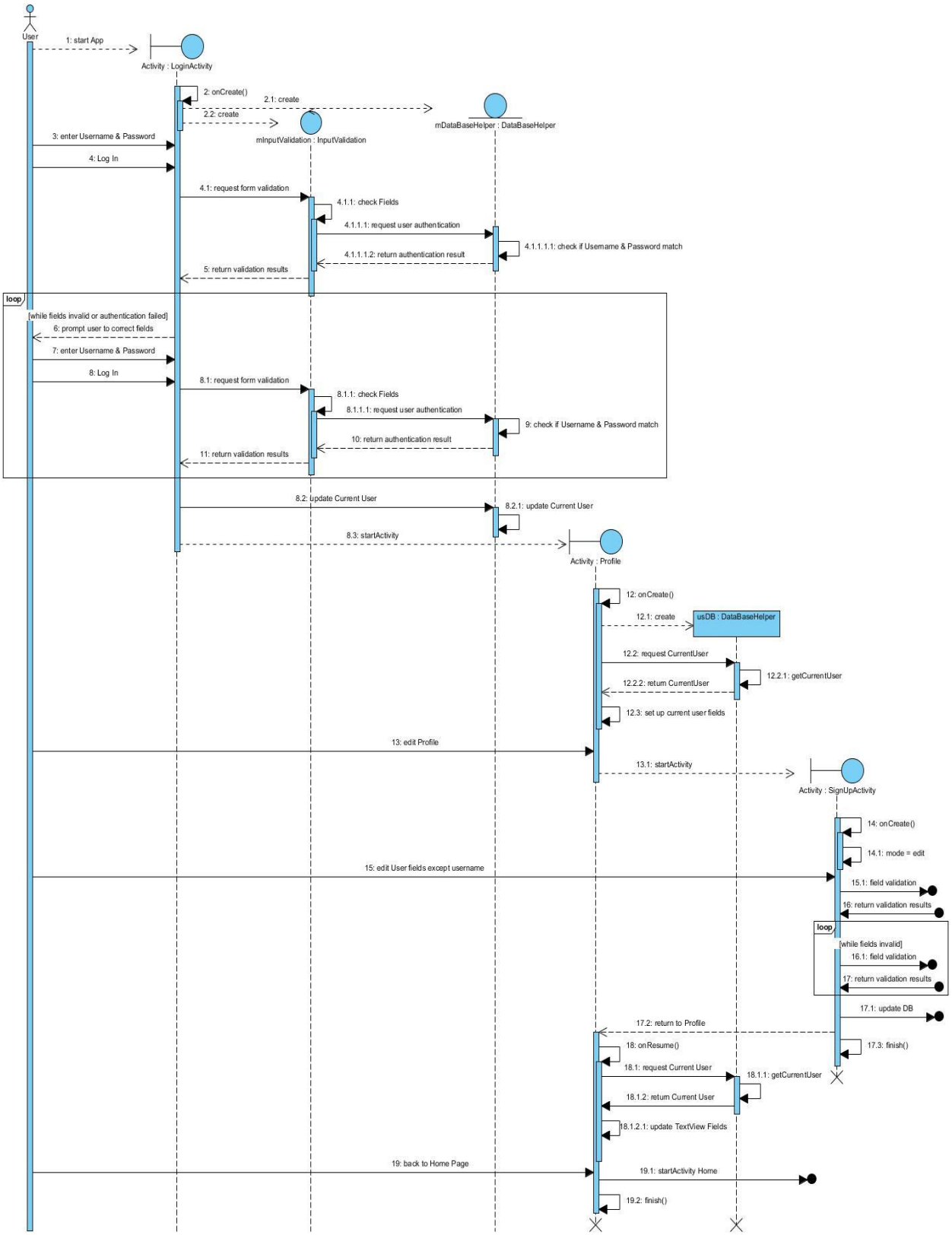
➢ **Δημιουργία και Προβολή Λογαριασμού**: Εδώ φαίνονται διαδοχικά τα βήματα του χρήστη. Εκτελώντας την εφαρμογή πηγαίνει στο LoginActivity από εκεί μεταβαίνει στην SignUpActivity για δημιουργία λογαριασμού, αφού συμπληρώσει όλα τα πεδία γίνεται επικυρωθούν από την InputValidation γίνεται η δημιουργία του λογαριασμού και αυτόματα μεταβαίνει στο activity του Profile όπου φαίνονται τα στοιχεία του λογαριασμού του και μπορεί να συνεχίσει σε περεταίρω λειτουργίες της εφαρμογής.

Sign Up Sequence Diagram - Figure_1



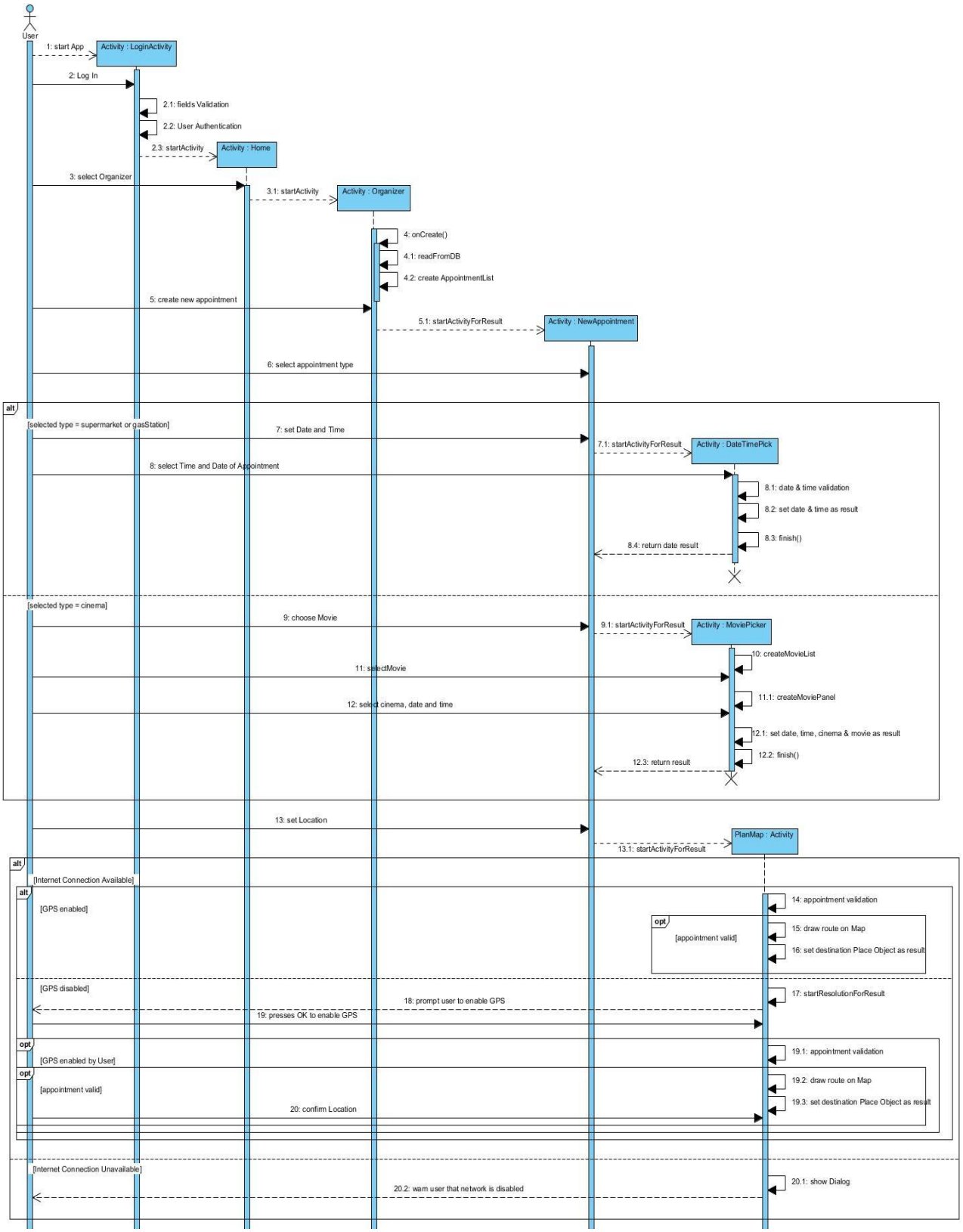
➢ **Επεξεργασία Λογαριασμού:** Παρακάτω φαίνονται τα βήματα που ακολουθεί ο χρήστης υπάρχοντος λογαριασμού που θέλει να επεξεργαστεί τα προσωπικά του στοιχεία. Αρχικά συνδέεται από το LoginActivity έπειτα από το Home μεταβαίνει στο Profile όπου επιλέγει edit profile το οποίο τον μεταφέρει σε μια παραμετροποιημένη έκδοση του SignUpActivity όπου αλλάζει όποια στοιχεία θέλει εκτός του username το οποίο πρέπει να παραμένει πάντοτε μοναδικό, όταν τελειώσει επικυρώνει τις αλλαγές και επιστρέφει στο Home αφού προβάλει πρώτα τις αλλαγές στο Profile.

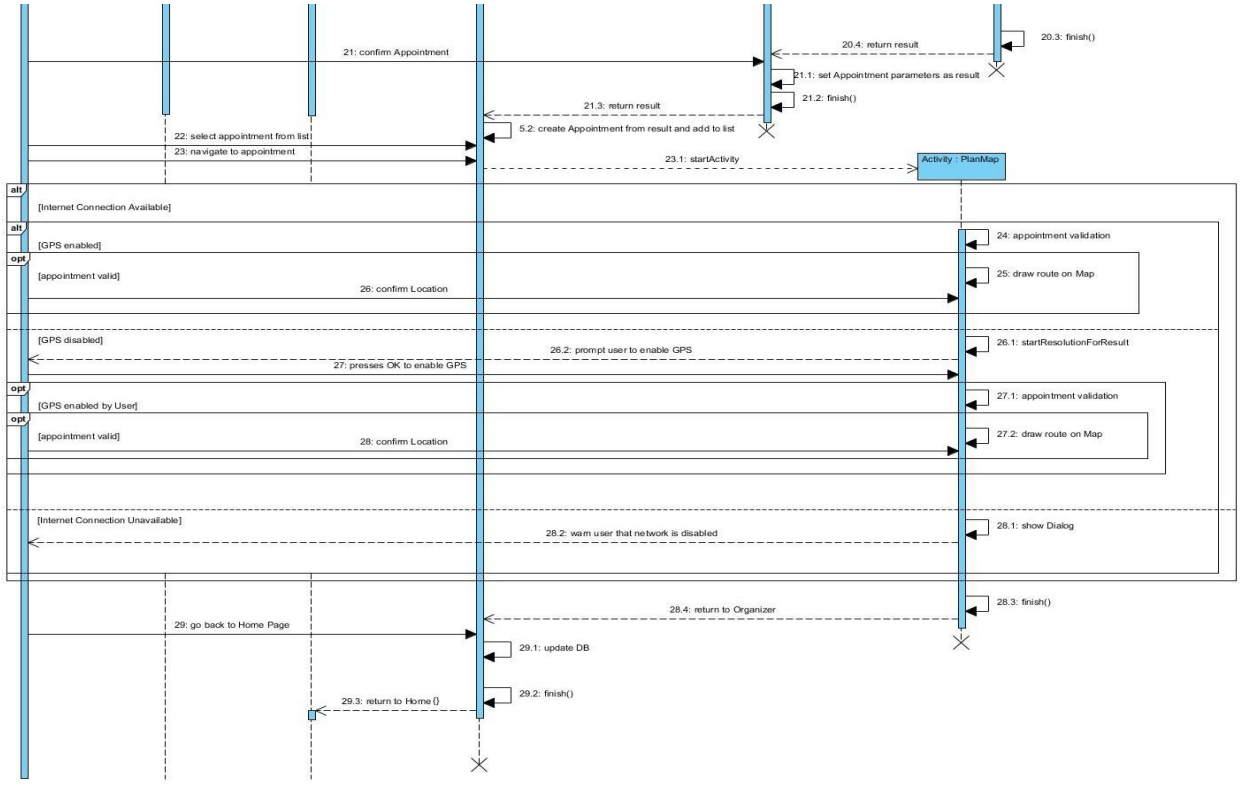
Edit Profile Sequence Diagram - Figure 1



- **Δημιουργία, Προβολή και Δρομολόγηση Νέου Ραντεβού:** Εδώ περιγράφονται τα βήματα πρώτα για τη δημιουργία και έπειτα για την προβολή και δρομολόγηση νέου ραντεβού σε χρήστη με υπάρχων λογαριασμό. Μετά από τις διαδικασίες του Login ο χρήστης μεταβαίνει στο Home, από εκεί επιλέγει το Organizer, εδώ φαίνονται τα υπάρχοντα ραντεβού του χρήστη καθώς και οι επιλογές διαχείρισής τους. Ο χρήστης επιλέγει δημιουργία νέου ραντεβού και μεταβαίνει στο NewAppointment εκεί επιλέγει τον τύπο του ραντεβού και ανάλογα με το μέρος που θα επιλέξει θα εμφανιστεί είτε η επιλογή για μετάβαση στο DateTimePick όπου επιλέγει μέρα και ώρα αν πρόκειται για ραντεβού τύπου supermarket ή gas station, είτε η επιλογή για μετάβαση στο MoviePicker όπου του δίνεται η δυνατότητα επιλογής ταινίας κινηματογράφου ώρας και μέρας. Αφού γίνει επιλογή όλων αυτών πατάει το set location για να επικυρώσει το ραντεβού και τη δρομολόγησή του, εκεί αφού ελεγχθεί αν έχει ενεργοποιήσει τα κατάλληλα components της συσκευής που απαιτούνται για την ομαλή λειτουργία του χάρτη, γίνεται ο έλεγχος εγκυρότητας του ραντεβού, αν το ραντεβού είναι έγκυρο, τότε δρομολογείται στο κοντινότερο σημείο που επέλεξε και επιστρέφει στο NewAppointment έχοντας αποθηκευτεί αυτόματα ο προορισμός του. Εκεί επιβεβαιώνει το ραντεβού και έτσι γίνεται επιστροφή στο Organizer όπου αποθηκεύεται στη λίστα από την οποία μπορεί να ζητήσει δρομολόγηση σε συγκεκριμένο ραντεβού. Στην περίπτωση αυτή ανοίγει και πάλι το PlanMap όπου ελέγχεται και πάλι η εγκυρότητα του ραντεβού και έπειτα γίνεται κανονικά η δρομολόγηση του χρήστη από την τρέχουσα θέση του.

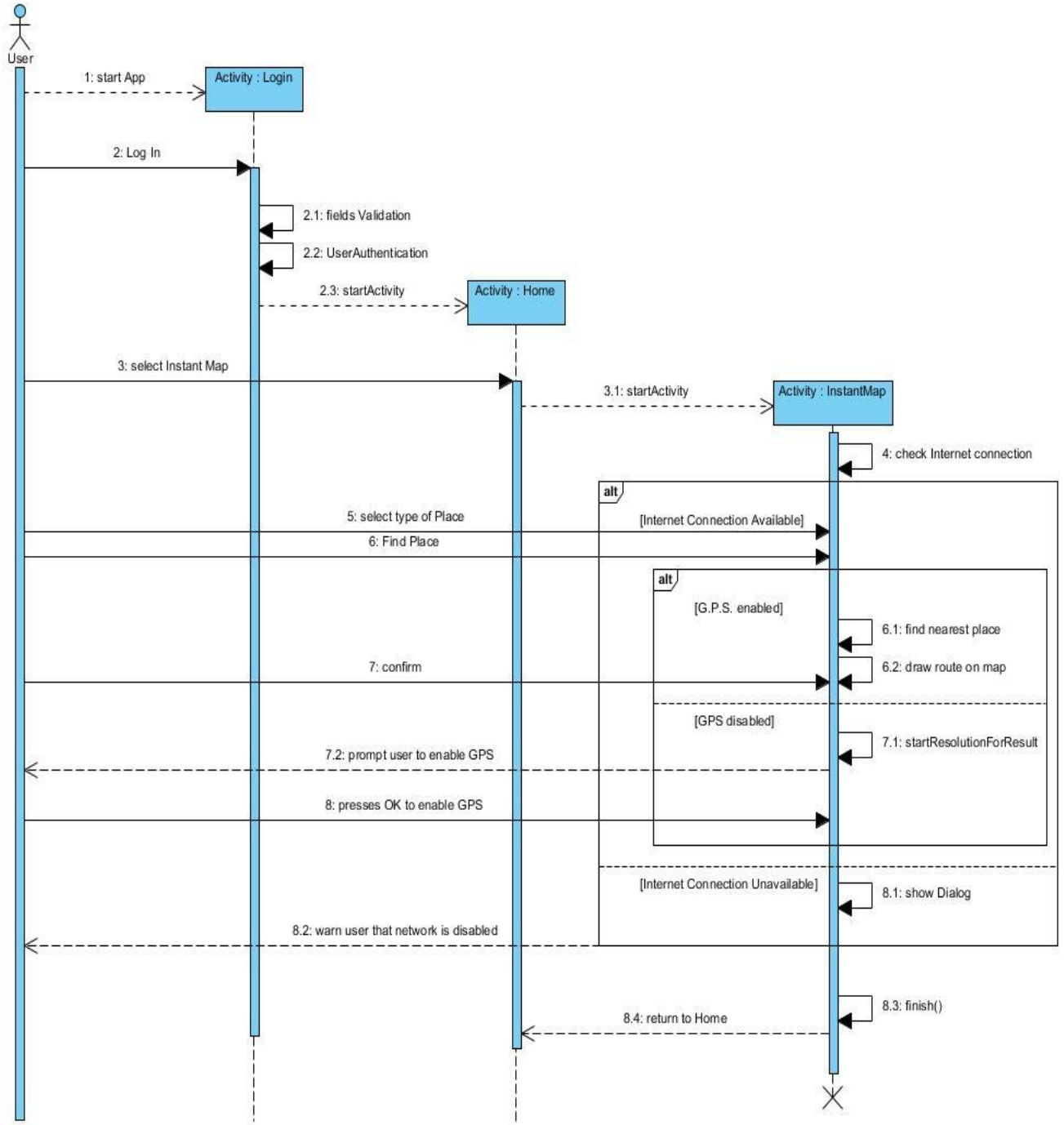
New Appointment Sequence Diagram -Figure 1




21: confirm Appointment
20.4: return result
20.3: finish()
21.1: set Appointment parameters as result
21.2: finish()
21.3: return result
5.2: create Appointment from result and add to list
22: select appointment from list
23: navigate to appointment
23.1: startActivity
Activity : PlanMap
alt
[Internet Connection Available]
alt
[GPS enabled]
opt
[appointment valid]
24: appointment validation
25: draw route on Map
26: confirm Location
[GPS disabled]
26.1: startResolutionForResult
26.2: prompt user to enable GPS
27: presses OK to enable GPS
opt
[GPS enabled by User]
opt
[appointment valid]
27.1: appointment validation
27.2: draw route on Map
28: confirm Location
[Internet Connection Unavailable]
28.1: show Dialog
28.2: warn user that network is disabled
28.3: finish()
28.4: return to Organizer
29: go back to Home Page
29.1: update DB
29.2: finish()
29.3: return to Home ()

➢ **Άμεση Δρομολόγηση:** Ο χρήστης αφού συνδεθεί με το λογαριασμό του επιλέγει το Instant Map από το Home, εμφανίζεται ο χάρτης με τις επιλογές μέρους, εκεί επιλέγει κάποιο τύπο μέρους (supermarket/gas station/cinema) και ο χάρτης τον δρομολογεί στο κοντινότερο ανάλογα με τη θέση του χρήστη ανεξαρτήτως των υπολοίπων ραντεβού.

Instant Map use Sequence Diagram -Figure 1



4.8 Απαιτήσεις Συστήματος

Οι απαιτήσεις συστήματος της εφαρμογής διαχωρίζονται σε απαιτήσεις υλικού (hardware) και απαιτήσεις λογισμικού (software).Οι απαιτήσεις υλικού μπορούν να περιγραφούν πλήρως από τις ελάχιστες προδιαγραφές που απαιτεί το Android 6.0 Marshmallow για το οποίο είναι σχεδιασμένη και η εφαρμογή. Παραθέτεται λοιπόν ο πλήρης πίνακας απαιτήσεων υλικού από το επίσημο Android Compatibility Definition Document για το Android 6.0 Marshmallow:

Android 6.0 Hardware Requirements 1

Category	Feature	Section	Handheld	Television	Watch	Automotive	Other
Input	D-pad	7.2.2. Non-touch Navigation		MUST			
	Touchscreen	7.2.4. Touchscreen input	MUST		MUST		SHOULD
	Microphone	7.8.1. Microphone	MUST	SHOULD	MUST	MUST	SHOULD
Sensors	Accelerometer	7.3.1 Accelerometer	SHOULD		SHOULD		SHOULD
	GPS	7.3.3. GPS	SHOULD			SHOULD	
Connectivity	Wi-Fi	7.4.2. IEEE 802.11	SHOULD	MUST		SHOULD	SHOULD
	Wi-Fi Direct	7.4.2.1. Wi-Fi Direct	SHOULD	SHOULD			SHOULD
	Bluetooth	7.4.3. Bluetooth	SHOULD	MUST	MUST	MUST	SHOULD
	Bluetooth Low Energy	7.4.3. Bluetooth	SHOULD	MUST	SHOULD	SHOULD	SHOULD
	USB peripheral/host mode	7.7. USB	SHOULD			SHOULD	SHOULD
Output	Speaker and/or Audio output ports	7.8.2. Audio Output	MUST	MUST		MUST	MUST

Οι μόνες διαφορές στις απαιτήσεις υλικού σε σχέση με την παρούσα εφαρμογή είναι η απαίτηση ύπαρξης GPS Sensor και κάποιου είδους internet connectivity είτε από WiFi είτε από mobile internet.

Οι απαιτήσεις λογισμικού είναι η ελάχιστη έκδοση Android, για την οποία σχεδιάστηκε και η εφαρμογή, να είναι Android 6.0 Marshmallow.

***Υποσημείωση**: Η εφαρμογή λειτουργεί και σε Android 5.0 Lollipop αλλά η ομαλή λειτουργία της δεν μπορεί να εγγυηθεί σε αυτήν την έκδοση λόγω αρκετών προβλημάτων συμβατότητας.*

4.9 Εργαλεία και Τεχνολογίες που χρησιμοποιήθηκαν

- Java 8: Η κύρια πλατφόρμα development της Java 8.
- Android Studio: Το κύριο IDE (Integrated Development Environment) στο οποίο περιλαμβάνονται και τα Android SDK Tools που μαζί χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.
- SQLite: Το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που χρησιμοποιήθηκε για τη διαχείριση και την αποθήκευση των δεδομένων τοπικά, μέσω SQL ερωτημάτων.
- Google Maps: Η υπηρεσία που χρησιμοποιήθηκε για την διαχείριση των χαρτών καθώς και της δρομολόγησης.
- Photoshop CS6: Το εργαλείο σχεδιασμού που χρησιμοποιήθηκε για την δημιουργία των background images και το icon της εφαρμογής.

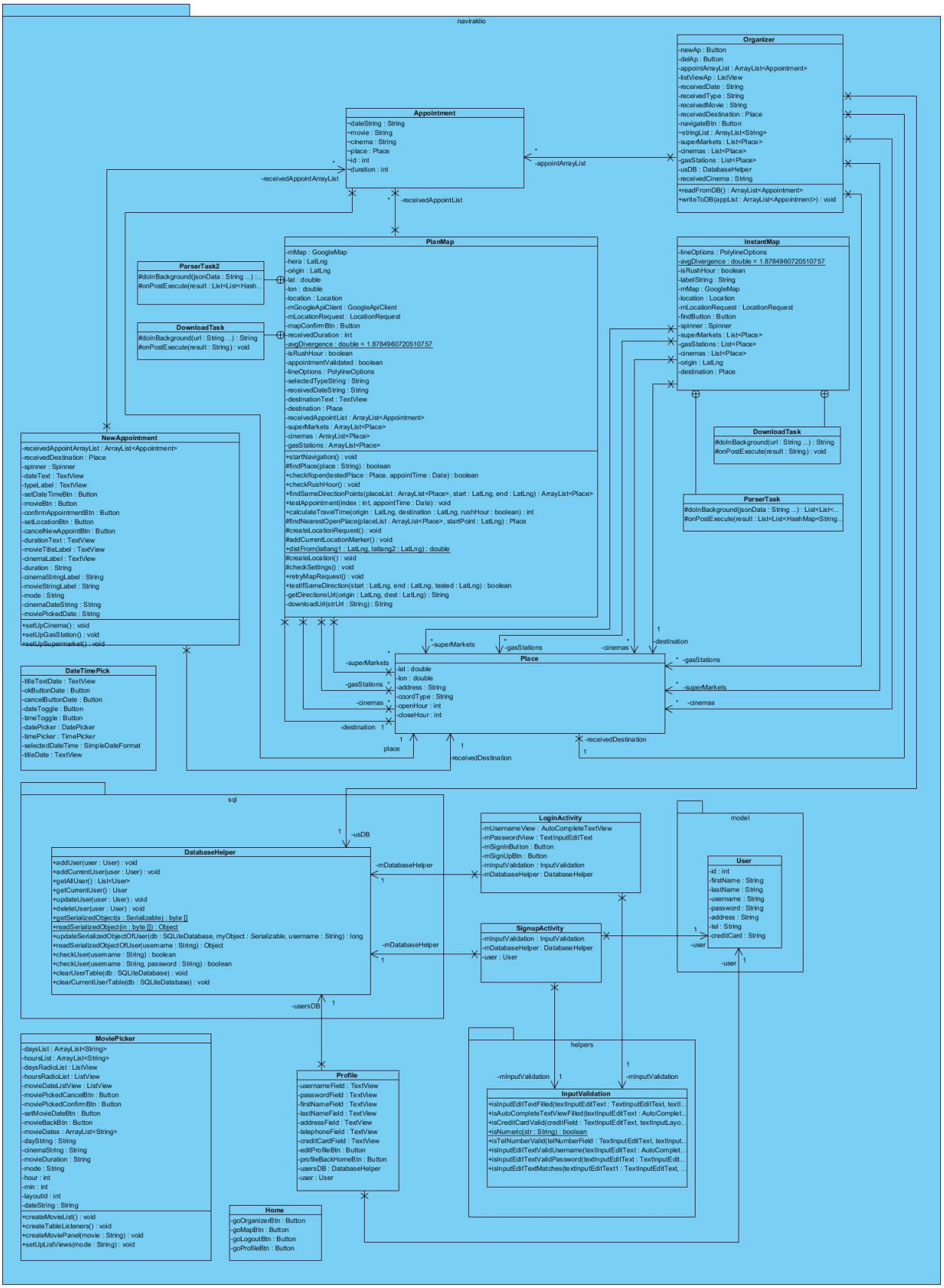
5. Υλοποίηση Εφαρμογής

Σε αυτό το κεφάλαιο θα αναλυθεί το στάδιο της υλοποίησης και ένα κομμάτι από τις δοκιμές λειτουργίας. Αρχικά θα περιγραφούν τεχνικές υλοποίησης που χρησιμοποιήθηκαν για την υποστήριξη οθονών πολλαπλών μεγεθών, έπειτα θα περιγραφεί η κύρια υλοποίηση των activities με βάση την συσχετιζόμενη λειτουργία για την οποία είναι σχεδιασμένα στην εφαρμογή μαζί με μια σύντομη αναφορά των κεντρικών τους ρόλων που αναφέρονται και στο κεφάλαιο 4.5. Τα Activities κατηγοριοποιούνται ανάλογα με τη συσχετιζόμενη λειτουργία ως εξής:

- **Διαχείριση Λογαριασμών:** LoginActivity, SignUpActivity, Profile.
- **Διαχείριση Ραντεβού:** Organizer, NewAppointment, DateTimePick, MoviePicker
- **Διαχείριση Πλοήγησης/Χαρτών:** InstantMap, PlanMap

Μετά από τη δημιουργία των κεντρικών activities έγινε η υλοποίηση των επιπλέον απαραίτητων στοιχείων όπως η βάση δεδομένων, η διόρθωση λαθών λογικής κυρίως στο κομμάτι των συνθηκών για τον έλεγχο εγκυρότητας ραντεβού, η υλοποίηση επιπλέον ελέγχων για την αποφυγή σφαλμάτων λόγω αδυναμίας σύνδεσης GPS ή Internet, και τέλος η τελειοποίηση του γραφικού περιβάλλοντος και οι δοκιμές λειτουργίας σε βασικά σενάρια χρήσης.

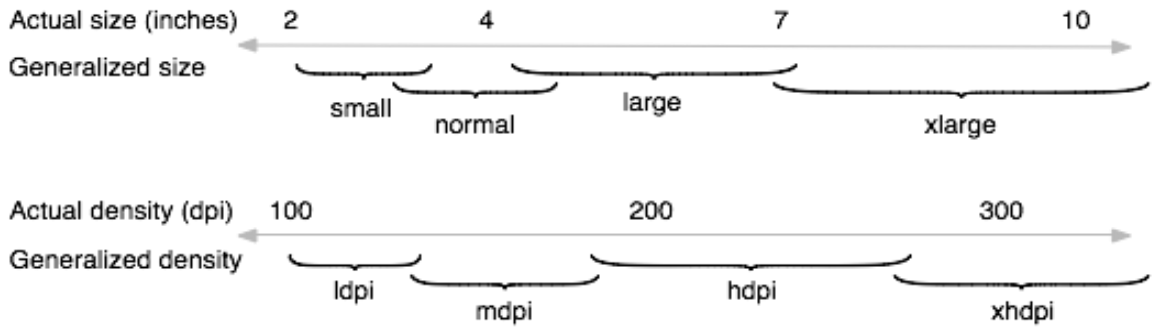
Παρακάτω παρατίθεται το γενικό class diagram που περιγράφει όλες τις κλάσεις, τις σημαντικότερες μεταβλητές και μεθόδους καθώς και τις σχέσεις που υπάρχουν μεταξύ τους.


naviraklio
Organizer
-newAp : Button
-delAp : Button
-appointArrayList : ArrayList<Appointment>
-listViewAp : ListView
-receivedDate : String
-receivedType : String
-receivedMovie : String
-recievedDestination : Place
-navigateBtn : Button
-stringList : ArrayList<String>
-superMarkets : List<Place>
-cinemas : List<Place>
-gasStations : List<Place>
-usDB : DatabaseHelper
-receivedCinema : String
+readFromDB() : ArrayList<Appointment>
+writeToDB(appList : ArrayList<Appointment>) : void
Appointment
-dateString : String
-movie : String
-cinema : String
-place : Place
-id : int
-duration : int
-receivedAppointArrayList
-appointArrayList
-receivedAppointList
PlanMap
-mMap : GoogleMap
-hera : LatLng
-origin : LatLng
-lat : double
-lon : double
-location : Location
-mGoogleApiClient : GoogleApiClient
-mLocationRequest : LocationRequest
-mapConfirmBtn : Button
-recievedDuration : int
-avgDivergence : double = 1.878496072051075 7
-isRushHour : boolean
-appointmentValidated : boolean
-lineOptions : PolylineOptions
-selectedTypeString : String
-receivedDateString : String
-destinationText : TextView
-destination : Place
-receivedAppointList : ArrayList<Appointment>
-superMarkets : ArrayList<Place>
-cinemas : ArrayList<Place>
-gasStations : ArrayList<Place>
+startNavigation() : void
#findPlace(place : String) : boolean
+checkIfopen(testedPlace : Place, appointTime : Date) : boolean
+checkRushHour() : void
+findSameDirectionPoints(placeList : ArrayList<Place>, start : LatLng, end : LatLng) : ArrayList<Place>
+testAppointment(index : int, appointTime : Date) : void
+calculateTravelTime(origin : LatLng, destination : LatLng, rushHour : boolean) : int
#findNearestOpenPlace(placeList : ArrayList<Place>, startPoint : LatLng) : Place
#createLocationRequest() : void
#addCurrentLocationMarker() : void
+distFrom(latlang1 : LatLng, latlang2 : LatLng) : double
#createLocation() : void
#checkSettings() : void
+retryMapRequest() : void
+testIfSameDirection(start : LatLng, end : LatLng, tested : LatLng) : boolean
-getDirectionsUrl(origin : LatLng, dest : LatLng) : String
-downloadUrl(strUrl : String) : String
ParserTask2
#doInBackground(jsonData : String ...) :...
#onPostExecute(result : List<List<Hash...
DownloadTask
#doInBackground(url : String ...) : String
#onPostExecute(result : String) : void
InstantMap
-lineOptions : PolylineOptions
-avgDivergence : double = 1.878496072051075 7
-isRushHour : boolean
-labelString : String
-mMap : GoogleMap
-location : Location
-mLocationRequest : LocationRequest
-findButton : Button
-spinner : Spinner
-superMarkets : List<Place>
-gasStations : List<Place>
-cinemas : List<Place>
-origin : LatLng
-destination : Place
DownloadTask
#doInBackground(url : String ...) : String
#onPostExecute(result : String) : void
ParserTask
#doInBackground(jsonData : String ...) : List<List<...
#onPostExecute(result : List<List<HashMap<String...
NewAppointment
-receivedAppointArrayList : ArrayList<Appointment>
-receivedDestination : Place
-spinner : Spinner
-dateText : TextView
-typeLabel : TextView
-setDateTimeBtn : Button
-movieBtn : Button
-confirmAppointmentBtn : Button
-setLocationBtn : Button
-cancelNewAppointBtn : Button
-durationText : TextView
-movieTitleLabel : TextView
-cinemaLabel : TextView
-duration : String
-cinemaStringLabel : String
-movieStringLabel : String
-mode : String
-cinemaDateString : String
-moviePickedDate : String
+setUpCinema() : void
+setUpGasStation() : void
+setUpSupermarket() : void
Place
-lat : double
-lon : double
-address : String
-coordType : String
-openHour : int
-closeHour : int
-superMarkets
-gasStations
-cinemas
-destination
-receivedDestination
place
DateTimePick
-titleTextDate : TextView
-okButtonDate : Button
-cancelButtonDate : Button
-dateToggle : Button
-timeToggle : Button
-datePicker : DatePicker
-timePicker : TimePicker
-selectedDateTime : SimpleDateFormat
-titleDate : TextView
sql
DatabaseHelper
+addUser(user : User) : void
+addCurrentUser(user : User) : void
+getAllUser() : List<User>
+getCurrentUser() : User
+updateUser(user : User) : void
+deleteUser(user : User) : void
+getSerializedObject(s : Serializable) : byte []
+readSerializedObject(in : byte []) : Object
+updateSerializedObjectOfUser(db : SQLiteDatabase, myObject : Serializable, username : String) : long
+readSerializedObjectOfUser(username : String) : Object
+checkUser(username : String) : boolean
+checkUser(username : String, password : String) : boolean
+clearUserTable(db : SQLiteDatabase) : void
+clearCurrentUserTable(db : SQLiteDatabase) : void
-usDB
-mDatabaseHelper
-usersDB
LoginActivity
-mUsernameView : AutoCompleteTextView
-mPasswordView : TextInputEditText
-mSignInButton : Button
-mSignUpBtn : Button
-mInputValidation : InputValidation
-mDatabaseHelper : DatabaseHelper
SignupActivity
-mInputValidation : InputValidation
-mDatabaseHelper : DatabaseHelper
-user : User
model
User
-id : int
-firstName : String
-lastName : String
-username : String
-password : String
-address : String
-tel : String
-creditCard : String
-user
MoviePicker
-daysList : ArrayList<String>
-hoursList : ArrayList<String>
-daysRadioList : ListView
-hoursRadioList : ListView
-movieDateListView : ListView
-moviePickedCancelBtn : Button
-moviePickedConfirmBtn : Button
-setMovieDateBtn : Button
-movieBackBtn : Button
-movieDates : ArrayList<String>
-dayString : String
-cinemaString : String
-movieDuration : String
-mode : String
-hour : int
-min : int
-layoutId : int
-dateString : String
+createMovieList() : void
+createTableListeners() : void
+createMoviePanel(movie : String) : void
+setUpListViews(mode : String) : void
Profile
-usernameField : TextView
-passwordField : TextView
-firstNameField : TextView
-lastNameField : TextView
-addressField : TextView
-telephoneField : TextView
-creditCardField : TextView
-editProfileBtn : Button
-profileBackHomeBtn : Button
-usersDB : DatabaseHelper
-user : User
helpers
InputValidation
+isInputEditTextFilled(textInputEditText : TextInputEditText, textI...
+isAutoCompleteTextViewFilled(textInputEditText : AutoComplet...
+isCreditCardValid(creditField : TextInputEditText, textInputLayo...
+isNumeric(str : String) : boolean
+isTelNumberValid(telNumberField : TextInputEditText, textInput...
+isInputEditTextValidUsername(textInputEditText : AutoComplet...
+isInputEditTextValidPassword(textInputEditText : TextInputEdit...
+isInputEditTextMatches(textInputEditText1 : TextInputEditText, ...
-mInputValidation
Home
-goOrganizerBtn : Button
-goMapBtn : Button
-goLogoutBtn : Button
-goProfileBtn : Button

5.1 Supporting Multiple Sizes of Screens

Η υποστήριξη οθονών διαφορετικών μεγεθών στην ανάπτυξη εφαρμογών Android γινόταν παλαιότερα με την χρήση προκαθορισμένης δομής φακέλων οι οποίοι επιλέγονταν ανάλογα με το μέγεθος της οθόνης της συσκευής στην οποία εκτελούταν η εφαρμογή. Για απλοποίηση του σχεδιασμού δημιουργήθηκαν δύο κατηγορίες: η πρώτη αναφερόταν στο μέγεθος της οθόνης ως προς το μήκος της διαγωνίου της η δεύτερη κατηγορία στην πυκνότητα των pixels σε ένα συγκεκριμένο εμβαδό επιφάνειας στην οθόνη που αποδίδεται με την μονάδα dpi (dots per inch). Δημιουργήθηκαν λοιπόν 4 γενικευμένα μεγέθη οθόνης διαγωνίου και 6 γενικευμένα μεγέθη πυκνότητας pixel τα οποία φαίνονται στο παρακάτω σχήμα.

Generalized Screen Sizes 1



Ο παρακάτω πίνακας αποδίδει συγκεκριμένα τους qualifiers που χρησιμοποιούνταν, πολλοί από αυτούς χρησιμοποιούνται και σήμερα

Screen characteristic	Qualifier	Description
Size	`small`	Resources for *small* size screens.
	`normal`	Resources for *normal* size screens. (This is the baseline size.)
	`large`	Resources for *large* size screens.
	`xlarge`	Resources for *extra-large* size screens.
Density	`ldpi`	Resources for low-density (*ldpi*) screens (~120dpi).
	`mdpi`	Resources for medium-density (*mdpi*) screens (~160dpi). (This is the baseline density.)
	`hdpi`	Resources for high-density (*hdpi*) screens (~240dpi).
	`xhdpi`	Resources for extra-high-density (*xhdpi*) screens (~320dpi).
	`xxhdpi`	Resources for extra-extra-high-density (*xxhdpi*) screens (~480dpi).
	`xxxhdpi`	Resources for extra-extra-extra-high-density (*xxxhdpi*) uses (~640dpi).

	nodpi	Resources for all densities. These are density-independent resources. The system does not scale resources tagged with this qualifier, regardless of the current screen's density.
	tvdpi	Resources for screens somewhere between mdpi and hdpi; approximately 213dpi. This is not considered a "primary" density group. It is mostly intended for televisions and most apps shouldn't need it—providing mdpi and hdpi resources is sufficient for most apps and the system will scale them as appropriate. If you find it necessary to provide tvdpi resources, you should size them at a factor of 1.33*mdpi. For example, a 100px x 100px image for mdpi screens should be 133px x 133px for tvdpi.
Orientation	land	Resources for screens in the landscape orientation (wide aspect ratio).
	port	Resources for screens in the portrait orientation (tall aspect ratio).
Aspect ratio	long	Resources for screens that have a significantly taller or wider aspect ratio (when in portrait or landscape orientation, respectively) than the baseline screen configuration.
	notlong	Resources for use screens that have an aspect ratio that is similar to the baseline screen configuration.

Με την άνοδο όμως των tablet με διαγώνιο οθόνης 7” η παραπάνω μέθοδος παρουσίασε προβλήματα καθώς το μέγεθος των 7 ιντσών βρίσκεται στο μεταίχμιο μεταξύ της γενικευμένης κατηγορίας large 4”-7” και xlarge 7”-10”. Για τον λόγο αυτόν όπως και για καλύτερη παραμετροποίηση των διεπαφών μεγάλης οθόνης όπως αυτή στα tablet η παραπάνω μέθοδος κατηγοριοποίησης έγινε deprecated. Από το Android 3.2 και μετά ενδείκνυται μια νέα μέθοδος η οποία κατηγοριοποιεί τα layouts ανάλογα με το πλάτος ή ύψος που είναι διαθέσιμα στο παράθυρο της εφαρμογής. Τα μεγέθη αυτά αποδίδονται σε dp (density independent pixels – pixels ανεξάρτητα από την πυκνότητα της οθόνης) και δεν χαρακτηρίζουν το φυσικό μέγεθος της οθόνης.

Τα πιο χαρακτηριστικά παραδείγματα των νέων αυτών qualifiers είναι:

- 320dp: μια τυπική οθόνη κινητού (240x320 ldpi, 320x480 mdpi, 480x800 hdpi, κλπ).
- 480dp: ένα tweener tablet όπως το Streak (480x800 mdpi).
- 600dp: ένα 7” tablet (600x1024 mdpi).
- 720dp: ένα 10” tablet (720x1280 mdpi, 800x1280 mdpi, κλπ).

Παρακάτω παραθέτεται ο πίνακας με τους νέους qualifiers

Screen configuration	Qualifier values	Description

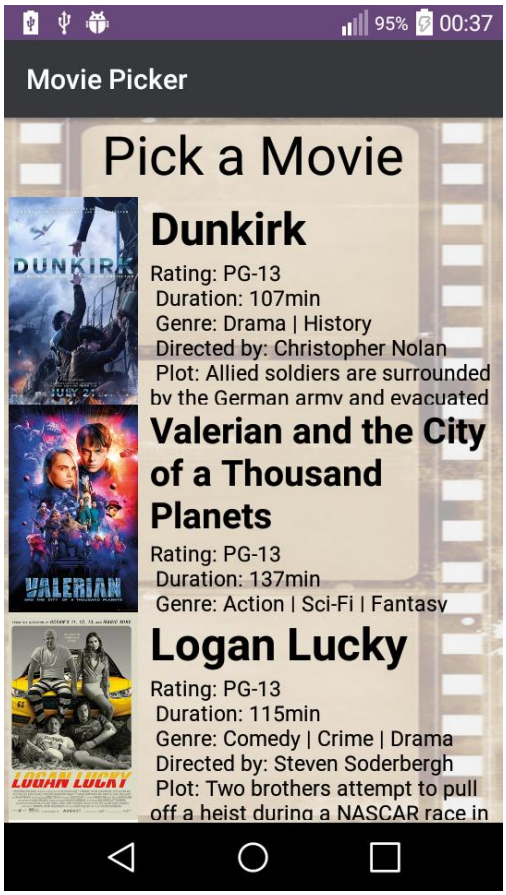
smallestWidth	`sw<N>dp`
	
	Examples:
	`sw600dp`
	`sw720dp`
		The fundamental size of a screen, as indicated by the shortest dimension of the available screen area. Specifically, the device's smallestWidth is the shortest of the screen's available height and width (you may also think of it as the "smallest possible width" for the screen). You can use this qualifier to ensure that, regardless of the screen's current orientation, your application's has at least `<N>` dps of width available for its UI.
		
		For example, if your layout requires that its smallest dimension of screen area be at least 600 dp at all times, then you can use this qualifier to create the layout resources, `res/layout-sw600dp/`. The system will use these resources only when the smallest dimension of available screen is at least 600dp, regardless of whether the 600dp side is the user-perceived height or width. The smallestWidth is a fixed screen size characteristic of the device; **the device's smallestWidth does not change when the screen's orientation changes**.
		
		The smallestWidth of a device takes into account screen decorations and system UI. For example, if the device has some persistent UI elements on the screen that account for space along the axis of the smallestWidth, the system declares the smallestWidth to be smaller than the actual screen size, because those are screen pixels not available for your UI.
		
		This is an alternative to the generalized screen size qualifiers (small, normal, large, xlarge) that allows you to define a discrete number for the effective size available for your UI. Using smallestWidth to determine the general screen size is useful because width is often the driving factor in designing a layout. A UI will often scroll vertically, but have fairly hard constraints on the minimum space it needs horizontally. The available width is also the key factor in determining whether to use a one-pane layout for handsets or multi-pane layout for tablets. Thus, you likely care most about what the smallest possible width will be on each device.

Available screen width	`w<N>dp`
	
	Examples:
	`w720dp`
	`w1024dp`
		Specifies a minimum available width in dp units at which the resources should be used—defined by the `<N>` value. The system's corresponding value for the width changes when the screen's orientation switches between landscape and portrait to reflect the current actual width that's available for your UI.
		
		This is often useful to determine whether to use a multi-pane layout, because even on a tablet device, you often won't want the same multi-pane layout for portrait orientation as you do for landscape. Thus, you can use this to specify the minimum width required for the layout, instead of using both the screen size and orientation qualifiers together.

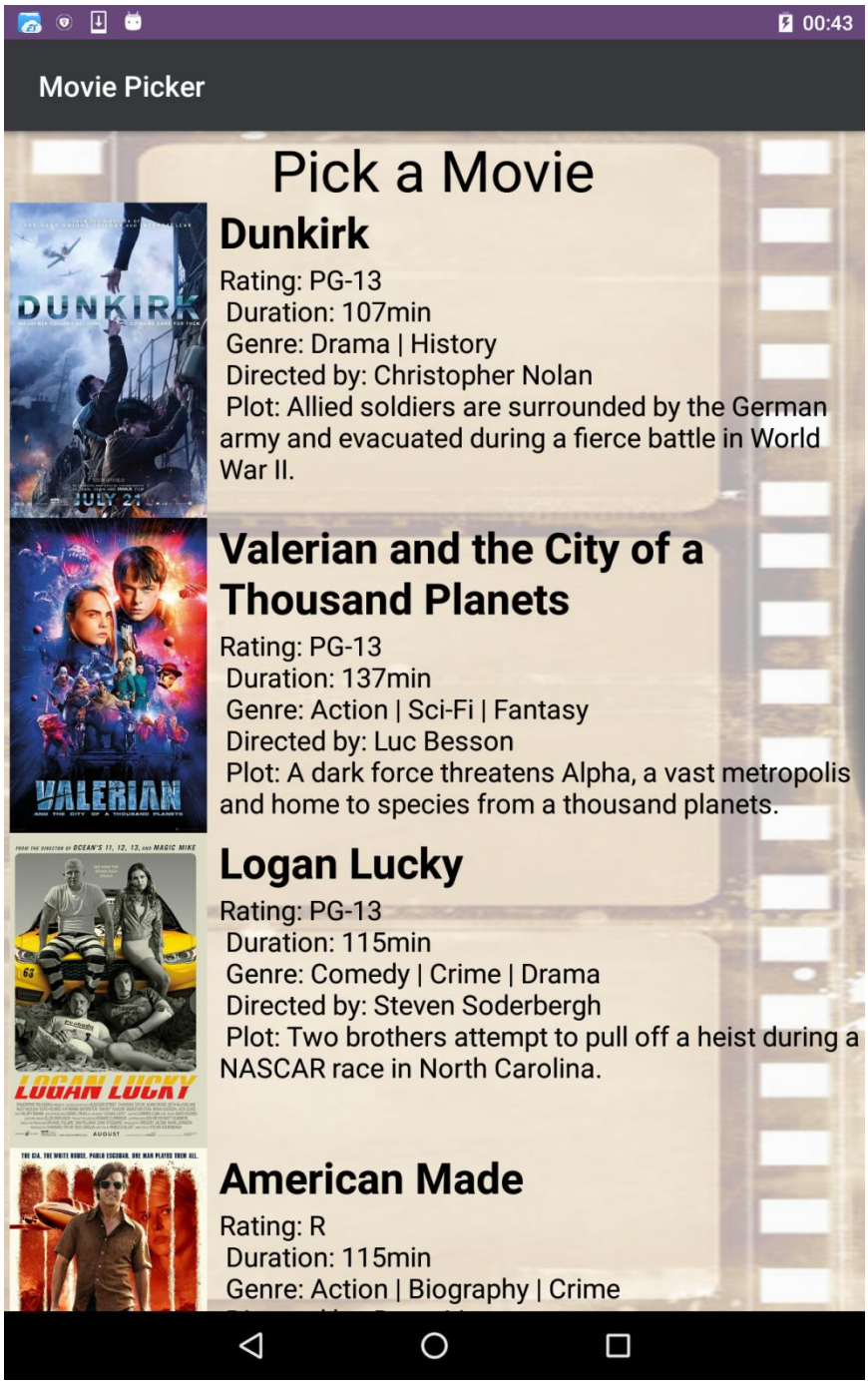
Available screen height	`h<N>dp`
	
	Examples:
	`h720dp`
	`h1024dp`
	etc.
		Specifies a minimum screen height in dp units at which the resources should be used—defined by the `<N>` value. The system's corresponding value for the height changes when the screen's orientation switches between landscape and portrait to reflect the current actual height that's available for your UI.
		
		Using this to define the height required by your layout is useful in the same way as `w<N>dp` is for defining the required width, instead of using both the screen size and orientation qualifiers. However, most apps won't need this qualifier, considering that UIs often scroll vertically and are thus more flexible with how much height is available, whereas the width is more rigid.

Στην παρούσα εφαρμογή έγινε χρήση του qualifier sw600dp για τον διαχωρισμό των layout στοχευόμενα σε tablet ενώ το default layout σχεδιάστηκε για τις ανάγκες συσκευών μικρότερης οθόνης όπως κινητά. Επίσης έγινε χρήση του qualifier land για τον ξεχωριστό σχεδιασμό των layout οριζόντιου προσανατολισμού.

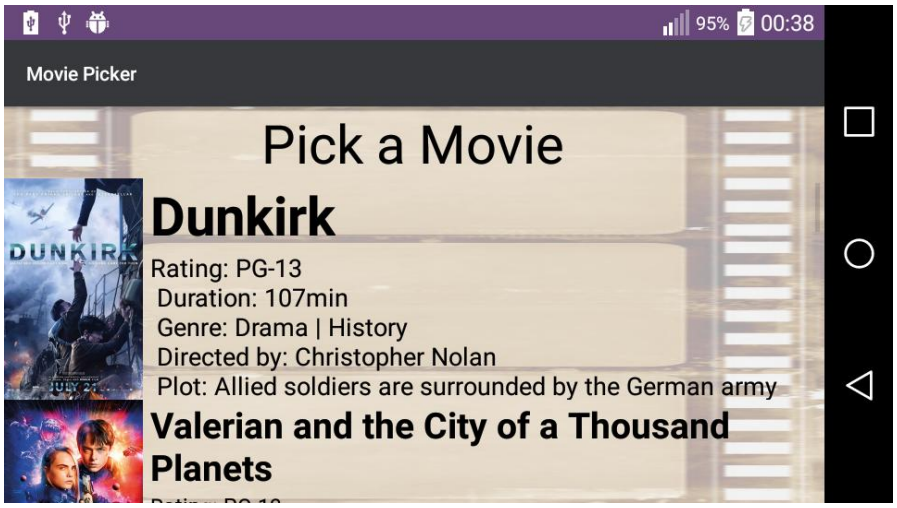
Παρακάτω παρατίθενται παραδείγματα οθονών σε πραγματικό μέγεθος σε κλίμακα 1:1 από την παρούσα εφαρμογή, στα οποία φαίνεται ο διαχωρισμός των layout ανάλογα με το μέγεθος και τον προσανατολισμό της οθόνης κάτι που δίνει μεγάλη ελευθερία και περιθώριο βελτιστοποίησης της ευχρηστίας και της προσαρμοστικότητας της εφαρμογής σε συσκευές διαφορετικού υλικού.



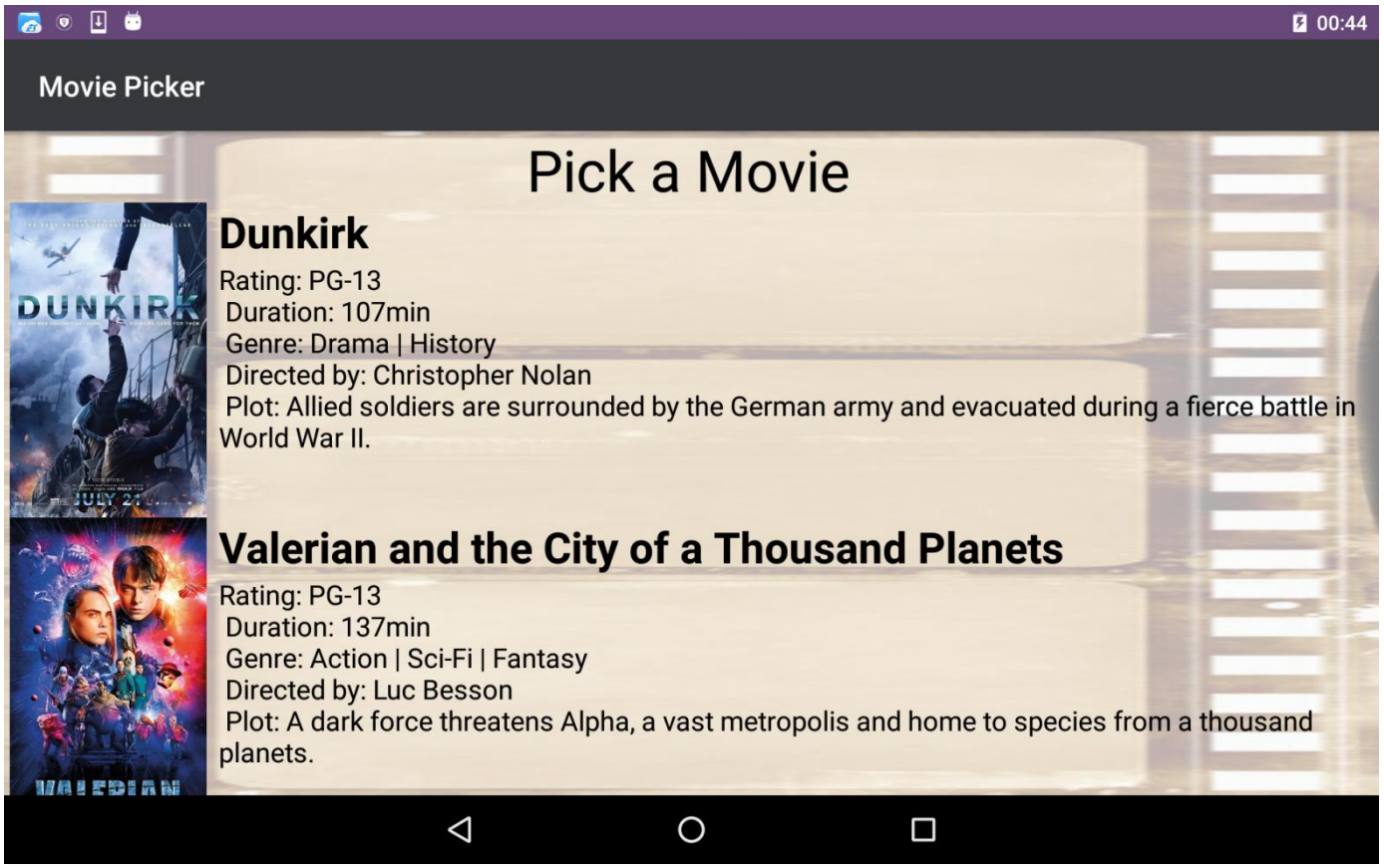
Default portrait layout
Qualifier: layout



Tablet portrait layout
Qualifier: layout-sw600dp



Default landscape layout
Qualifier: layout-land



Tablet landscape layout
Qualifier: layout-sw600dp-land

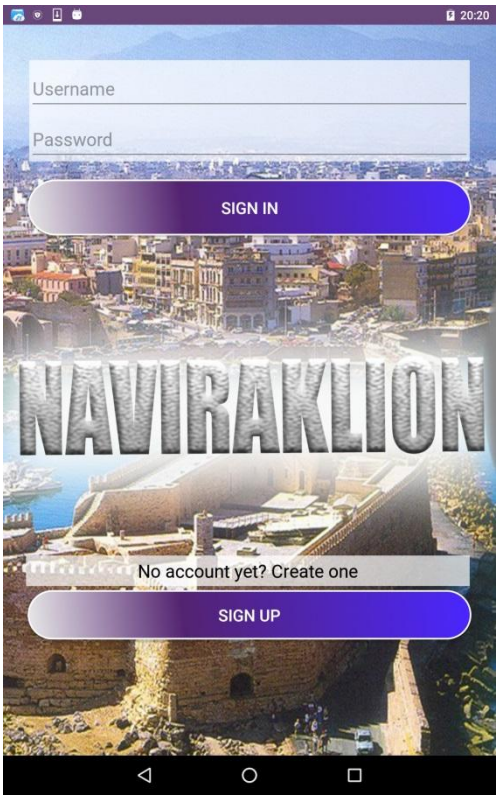
5.3 Διαχείριση Λογαριασμών

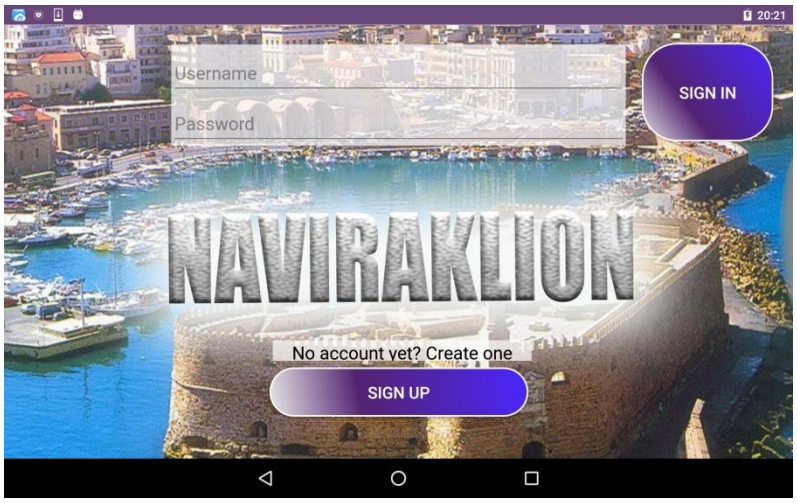
5.3.1 LoginActivity

Ρόλος: Το LoginActivity αποτελεί την πρώτη οθόνη που συναντάει ο χρήστης όταν ξεκινήσει την εφαρμογή και είναι υπεύθυνη για την σύνδεση και την πιστοποίηση ταυτότητας του χρήστη.

Εκτέλεση: Αρχικά ο σχεδιασμός βασίστηκε στο Login Activity template που προσφέρει το Android Studio, το οποίο ακολουθεί τα πρότυπα του Material Design που προτείνει η Google. Έτσι ξεκίνησε ο σχεδιασμός του layout ο οποίος έπρεπε να ικανοποιήσει τις παραπάνω απαιτήσεις. Σχεδιάστηκε το layout για όλα τα μεγέθη οθόνης το οποίο απαρτιζόταν κυρίως από δύο κουμπιά, ένα για είσοδο του χρήστη (sign in) και ένα για εγγραφή νέου χρήστη (sign up), και δύο πεδία για συμπλήρωση των username και password, αργότερα προστέθηκαν και άλλες λεπτομέρειες όπως background images, customized button styles, labels κ.α. Το τελικό αποτέλεσμα του layout φαίνεται στις παρακάτω εικόνες.

LoginActivity Layout 1





Στη συνέχεια διαμορφώθηκε ο βασικός κώδικας ορισμού των views, των απαραίτητων listeners για τα κουμπιά, και η εισαγωγή ενός progressbar και η παραμετροποίηση του ώστε να εμφανίζεται κατά τη διάρκεια επιτυχούς εισόδου και να αποκρύπτεται πάλι όταν η εφαρμογή γυρίσει πίσω στο LoginActivity κάτι που επιτεύχθηκε καλώντας setVisibility(View.GONE) στην callback μέθοδο onResume() του activity. Το δεύτερο σκέλος ήταν η δημιουργία ελέγχου εγκυρότητας πεδίων, έτσι φτιάχτηκε η κλάση InputValidation η οποία καλείται στην LoginActivity και περιέχει μεθόδους που ελέγχουν το μέγεθος της συμβολοσειράς που εισήγαγε ο χρήστης, είναι πολύ μικρό (ορισμένο ελάχιστο είναι 5 χαρακτήρες για username και 6 για password) ή και μηδενικό. Παρακάτω φαίνεται ο κώδικας με τις μεθόδους ελέγχου της κλάσης InputValidation που χρησιμοποιείται.

Field Validation 1

```
public boolean isInputEditTextFilled(TextInputEditText textInputEdit-
Text, TextInputLayout textInputLayout, String message) {
    String value = textInputEditText.getText().toString().trim();
    if (value.isEmpty()) {
        textInputLayout.setError(message);
        hideKeyboardFrom(textInputEditText);
        return false;
    } else {
        textInputLayout.setErrorEnabled(false);
    }

    return true;
}

public boolean isAutoCompleteTextViewFilled(AutoCompleteTextView tex-
tInputEditText, TextInputLayout textInputLayout, String message) {
    String value = textInputEditText.getText().toString().trim();
    if (value.isEmpty()) {
        textInputLayout.setError(message);
        hideKeyboardFrom(textInputEditText);
        return false;
    } else {
        textInputLayout.setErrorEnabled(false);
    }

    return true;
}

public boolean isInputEditTextValidUsername(AutoCompleteTextView tex-
tInputEditText, TextInputLayout textInputLayout, String message) {
    String value = textInputEditText.getText().toString().trim();
    if (value.isEmpty() || textInputEditText.length() < 5) {
        textInputLayout.setError(message);
        hideKeyboardFrom(textInputEditText);
        return false;
    } else {
        textInputLayout.setErrorEnabled(false);
    }
    return true;
}

public boolean isInputEditTextValidPassword(TextInputEditText
textInputEditText, TextInputLayout textInputLayout, String message) {
    String value = textInputEditText.getText().toString().trim();
    if (value.isEmpty() || textInputEditText.length() < 6) {
        textInputLayout.setError(message);
        hideKeyboardFrom(textInputEditText);
        return false;
    } else {
        textInputLayout.setErrorEnabled(false);
    }
    return true;
}
```

Το τρίτο σκέλος ήταν το authentication, όπου γίνεται επικοινωνία με τη βάση μέσω μεθόδων της κλάσης DatabaseHelper η οποία ουσιαστικά είναι υπεύθυνη για όλες τις ενέργειες που σχετίζονται με την βάση δεδομένων. Η λογική του authentication είχε δύο μέρη, το πρώτο είναι να κάνει αναζήτηση στον πίνακα χρηστών της βάσης ελέγχοντας αν υπάρχει username και password στην ίδια εγγραφή του πίνακα που να ταιριάζει απόλυτα με τα στοιχεία που εισήγαγε ο χρήστης στα αντίστοιχα πεδία. Η παραπάνω λογική επιτυγχάνεται ελέγχοντας την τιμή επιστροφής της παρακάτω μεθόδου checkUser η οποία επιστρέφει true σε περίπτωση που βρέθηκε ζευγάρι username/password που να ταιριάζει στο input του χρήστη και false αν δεν βρέθηκε.

Authentication Code 1

```
public boolean checkUser(String username, String password) {

    // array of columns to fetch
    String[] columns = {
            COLUMN_USER_FIRST_NAME,
            COLUMN_USER_LAST_NAME,
            COLUMN_USER_USERNAME,
            COLUMN_USER_PASSWORD,
            COLUMN_USER_ADDRESS,
            COLUMN_USER_TEL,
            COLUMN_USER_CREDIT_CARD
    };
    SQLiteDatabase db = this.getReadableDatabase();
    // selection criteria
    String selection = COLUMN_USER_USERNAME + " = ?" + " AND " +
COLUMN_USER_PASSWORD + " = ?";

    // selection arguments
    String[] selectionArgs = {username, password};

    // query user table with conditions
    /**
     * Here query function is used to fetch records from user table
this function works like we use sql query.
     * SQL query equivalent to this query function is
     * SELECT user_id FROM user WHERE user_username = 'username-
example' AND user_password = 'qwerty';
     */
    Cursor cursor = db.query(TABLE_USER, //Table to query
            columns,                    //columns to return
            selection,                  //columns for the WHERE clause
            selectionArgs,              //The values for the WHERE
clause
            null,                       //group the rows
            null,                       //filter by row groups
            null);                      //The sort order
```

```
    int cursorCount = cursor.getCount();
    if (cursorCount > 0) {
        if (cursor.moveToFirst()) {
            do {
                this.clearCurrentUserTable(this.getWritableDatabase());
                String firstNameResult =
cursor.getString(cursor.getColumnIndex(COLUMN_USER_FIRST_NAME));
                String lastNameResult =
cursor.getString(cursor.getColumnIndex(COLUMN_USER_LAST_NAME));

                String usernameResult =
cursor.getString(cursor.getColumnIndex(COLUMN_USER_USERNAME));

                String passwordResult =
cursor.getString(cursor.getColumnIndex(COLUMN_USER_PASSWORD));

                String addressResult =
cursor.getString(cursor.getColumnIndex(COLUMN_USER_ADDRESS));

                String telResult =
cursor.getString(cursor.getColumnIndex(COLUMN_USER_TEL));

                String creditCardResult =
cursor.getString(cursor.getColumnIndex(COLUMN_USER_CREDIT_CARD));

                this.addCurrentUser(new User(firstNameResult,
lastNameResult, usernameResult, passwordResult, addressResult,
telResult, creditCardResult));

            } while (cursor.moveToNext());
        }

    }


    cursor.close();
    db.close();
    return cursorCount > 0;

}
```

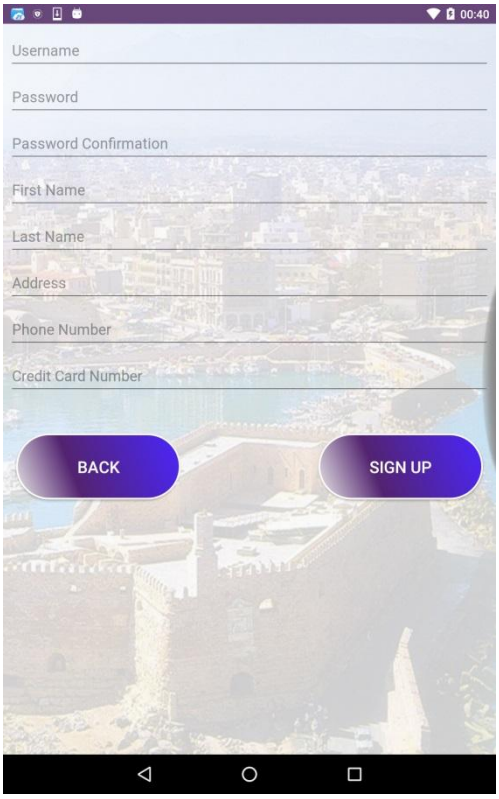
Το δεύτερο μέρος του authentication περιλαμβάνει την αναδημιουργία ενός δεύτερου πίνακα, του CurrentUser ο οποίος διαθέτει πάντοτε μια εγγραφή που περιγράφει τον τρέχον χρήστη για να μπορούν να ανακληθούν ανά πάσα στιγμή τα ανάλογα δεδομένα που σχετίζονται με τον συνδεδεμένο χρήστη. Αφού αναδημιουργηθεί ο πίνακας (Drop Table και Create Table) γίνεται αντιγραφή της εγγραφής από τον πίνακα χρηστών που αντιστοιχήθηκε επιτυχώς στα στοιχεία που έδωσε ο χρήστης, στον πίνακα CurrentUser και η εφαρμογή προχωρεί στο επόμενο activity, το Home.

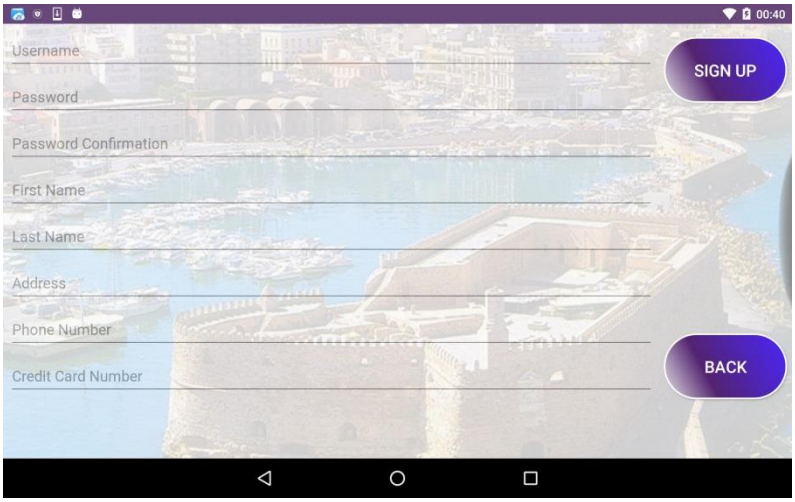
5.2.2 SignUpActivity

Ρόλος: Το SignUpActivity αποτελεί το activity ρόλος του οποίου είναι η δημιουργία ή επεξεργασία προσωπικών στοιχείων του λογαριασμού μέσω συμπλήρωσης μιας φόρμας.

Εκτέλεση: Πρώτο σκέλος ήταν ο σχεδιασμός του layout κάτι που χρειάστηκε ιδιαίτερη παραμετροποίηση ανάλογα με το μέγεθος της οθόνης καθώς έπρεπε να αποτραπεί η μετατόπιση των κουμπιών με την ανάδυση του πληκτρολογίου ενώ παράλληλα να υπάρχει ενεργό scrolling για πρόσβαση όλων των πεδίων ακόμα και όταν μέρος της οθόνης είναι καλυμμένο από το πληκτρολόγιο, εν τέλει οι στόχοι αυτοί επιτεύχθηκαν και το τελικό αποτέλεσμα των layout φαίνεται στις παρακάτω εικόνες.

SignUpActivity Layout 1





Σαν δεύτερο σκέλος ήταν η υλοποίηση ελέγχου εγκυρότητας πεδίων, εδώ υπήρχαν οι εξής περιορισμοί:

- Το username δεν πρέπει να είναι μικρότερο από 5 χαρακτήρες
- Το password δεν πρέπει να είναι μικρότερο από 6 χαρακτήρες
- Το πεδίο επιβεβαίωσης κωδικού πρέπει να ταιριάζει απόλυτα με το πεδίο κωδικού
- Το τηλέφωνο πρέπει να περιλαμβάνει μόνο νούμερα με μήκος από 10 μέχρι 15 ψηφία όπως υποδεικνύουν τα διεθνή πρότυπα για τηλεφωνικούς αριθμούς.
- Ο αριθμός πιστωτικής κάρτας πρέπει να περιλαμβάνει αποκλειστικά νούμερα μήκους 16 ψηφίων.

Οι έλεγχοι αυτοί εκτελέστηκαν με συνθήκες που χρησιμοποιούσαν για παραμέτρους τις επιστροφές ειδικών μεθόδων της κλάσης InputValidation. Παρακάτω παρατίθενται κομμάτια κώδικα που περιγράφουν μερικές από αυτές τις μεθόδους.

Field Validation 2

```
public boolean isCreditCardValid(TextInputEditText creditField,
TextInputLayout textInputLayout) {
    String creditString = creditField.getText().toString().trim();
    boolean creditValid = creditString.length() == 16 &&
isNumeric(creditString);
    if (!creditValid) {
        textInputLayout.setError("Invalid Credit Card Number");
    } else {
        textInputLayout.setErrorEnabled(false);
    }
    return creditValid;
}


public static boolean isNumeric(String str) {
    try {
        long d = Long.parseLong(str);
    } catch (NumberFormatException nfe) {
        return false;
    }
    return true;
}

public boolean isTelNumberValid(TextInputEditText telNumberField,
TextInputLayout textInputLayout) {
    String telString = telNumberField.getText().toString().trim();
    boolean telValid = telNumberField.length() <= 15 &&
telNumberField.length() >= 10 && isNumeric(telString);
    if (!telValid) {
        textInputLayout.setError("Invalid Phone Number");
    } else {
        textInputLayout.setErrorEnabled(false);
    }
    return telValid;
}
```

```
public boolean isInputEditTextMatches(TextInputEditText
textInputEditText1, TextInputEditText textInputEditText2,
TextInputLayout textInputLayout, String message) {
    String value1 = textInputEditText1.getText().toString().trim();
    String value2 = textInputEditText2.getText().toString().trim();
    if (!value1.contentEquals(value2)) {
        textInputLayout.setError(message);
        hideKeyboardFrom(textInputEditText2);
        return false;
    } else {
        textInputLayout.setErrorEnabled(false);
    }
    return true;
}
```

Το επόμενο κομμάτι ήταν ο διαχωρισμός της λειτουργίας του SignUpActivity, αυτό επιτυγχάνεται στέλνοντας ένα parcelable extra που περιέχει ένα instance του current user μαζί με το intent όταν χρειάζεται να κληθεί το activity για επεξεργασία λογαριασμού, το activity με τη σειρά του ελέγχει αν αυτό το extra είναι null, σε περίπτωση που δεν είναι εκχωρεί σε ένα string ελέγχου το mode την τιμή “edit” υποδεικνύοντας στους παρακάτω ελέγχους ότι το activity θα χρησιμοποιηθεί για επεξεργασία λογαριασμού, διαφορετικά αν το parcelable extra είναι ίσο με null το mode παίρνει την τιμή “new” και το activity χρησιμοποιείται για τη δημιουργία νέου λογαριασμού.

Σε αυτό το σημείο το activity διαχωρίζεται σε δύο λειτουργικότητες ανάλογα με την τιμή του mode:

- Αν το mode έχει τιμή new σημαίνει πως το activity θα χρησιμοποιηθεί για δημιουργία λογαριασμού, τότε γίνεται έλεγχος ύπαρξης του εισαγόμενου username σε όλο τον πίνακα users της βάσης δεδομένων (το username πρέπει να είναι μοναδικό καθώς αποτελεί το primary key στον πίνακα users). Αυτό επιτυγχάνεται με κλήση μιας overloaded μεθόδου της DatabaseHelper με παρόμοια λειτουργικότητα όπως η μέθοδος με το ίδιο όνομα που αναφέρθηκε στο 5.2.1 για το authentication, η διαφορά σε αυτή την μέθοδο είναι ότι κάνει αναζήτηση μόνο του username αντί για username και password στην ίδια εγγραφή όπως έκανε η παραπάνω μέθοδος στο LoginActivity. Αν το username προς δημιουργία δεν υπάρχει σε καμία εγγραφή του πίνακα users της βάσης και έχει ήδη ικανοποιήσει τις συνθήκες εγκυρότητας πεδίων που αναφέρθηκαν παραπάνω τότε γίνεται εισαγωγή νέας εγγραφής στο πίνακα με τα πεδία που έχει εισάγει ο χρήστης, αναδημιουργείται ο πίνακας CurrentUser και εισάγονται σαν μοναδική εγγραφή του τα δεδομένα των πεδίων της φόρμας, έπειτα το activity τερματίζει και η εφαρμογή προχωρεί στο Profile Activity όπου φαίνονται τα στοιχεία που εισάχθηκαν.

- Αν το mode έχει τιμή edit σημαίνει πως το activity θα χρησιμοποιηθεί για επεξεργασία λογαριασμού, τότε γίνεται γεμίζουν τα πεδία με τα περιεχόμενα της μοναδικής εγγραφής του πίνακα CurrentUser και η επεξεργασία του πεδίου username απενεργοποιείται ώστε να παραμείνει μοναδικό όπως απαιτείται σαν primary key στον πίνακα users της βάσης. Αφού γίνει και πάλι ο έλεγχος εγκυρότητας πεδίων χρησιμοποιείται η μέθοδος updateUser της κλάσης DatabaseHelper η οποία κάνει αναζήτηση για την εγγραφή με το username που αναγράφεται στο πεδίο και ενημερώνονται τα υπόλοιπα περιεχόμενα της εγγραφής με τα στοιχεία των πεδίων, τέλος γίνεται αναδημιουργία του πίνακα CurrentUser και εισαγωγή της νέας ενημερωμένης μοναδικής εγγραφής του και το activity τερματίζει. Παρακάτω παρατίθεται ένα κομμάτι κώδικα από την μέθοδο updateUser

updateUser method 1

```
public void updateUser(User user) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(COLUMN_USER_FIRST_NAME, user.getFirstName());
    values.put(COLUMN_USER_LAST_NAME, user.getLastName());
    values.put(COLUMN_USER_USERNAME, user.getUsername());
    values.put(COLUMN_USER_PASSWORD, user.getPassword());
    values.put(COLUMN_USER_ADDRESS, user.getAddress());
    values.put(COLUMN_USER_TEL, user.getTel());
    values.put(COLUMN_USER_CREDIT_CARD, user.getCreditCard());

    // updating row
    db.update(TABLE_USER, values, COLUMN_USER_USERNAME + " = ?",
            new String[]{String.valueOf(user.getUsername())});
    db.close();
}
```

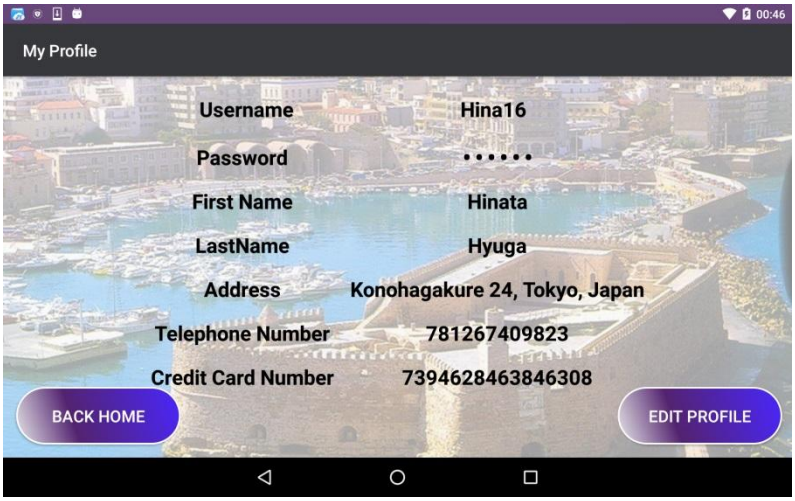
5.2.3 Profile

Ρόλος: Ο ρόλος του Profile είναι η προβολή των προσωπικών στοιχείων του λογαριασμού χρήστη.

Εκτέλεση: Η υλοποίηση του Profile ήταν σχετικά απλή, το layout περιλαμβάνεται από ένα TableLayout στο οποίο έχουν εισαχθεί TextViews που αναγράφουν τους τίτλους και τα περιεχόμενα των στοιχείων του λογαριασμού, εδώ αξίζει να σημειωθεί ότι δεν χρειάστηκε υλοποίηση ελέγχου για διατήρηση δεδομένων στα περιεχόμενα των TextViews κατά την περιστροφή της συσκευής όπως εφαρμόστηκε σε άλλα activities, για τον λόγο ότι τα περιεχόμενα αυτά ενημερώνονται μέσω της βάσης κάθε φορά που καλείται η callback onCreate κάτι που συμβαίνει κατά την αλλαγή προσανατολισμού του layout.

Παρακάτω φαίνονται τα υποδείγματα layout για το Profile Activity.

Profile Layout 1



Τα περιεχόμενα των κελιών όπως προαναφέρθηκε ενημερώνονται άμεσα από τη βάση καλώντας την μέθοδο getCurrentUser της DatabaseHelper η οποία επιστρέφει την μοναδική εγγραφή του πίνακα CurrentUser που περιέχει τα στοιχεία του χρήστη που αυτά με τη σειρά τους εμφανίζονται στην οθόνη. Η ενημέρωση των κελιών από τη βάση γίνεται σε δύο σημεία, το πρώτο είναι στην onCreate και το δεύτερο στην onRestore ώστε να ενημερωθούν και στη δημιουργία αλλά και στην επαναφορά του activity που γίνεται μετά από επεξεργασία λογαριασμού. Επιπλέον έχουν εισαχθεί δύο κουμπιά για να υπάρχει η δυνατότητα μετάβασης στην φόρμα επεξεργασίας λογαριασμού (SignUpActivity) ή στην αρχική σελίδα (Home Activity) αντίστοιχα.

5.3 Διαχείριση Ραντεβού

5.3.1 Organizer

Ρόλος: Το Organizer έχει ως ρόλο την γενική διαχείριση των ραντεβού, αποτελείται από μια λίστα των υπαρχόντων ραντεβού και μέσω κατάλληλων κουμπιών δίνει τη δυνατότητα δημιουργίας καινούριου, διαγραφής ή και δρομολόγησης σε υπάρχον ραντεβού από τη λίστα.

Εκτέλεση: Το πρώτο σκέλος της υλοποίησης του Organizer ήταν να σχεδιαστεί ο τρόπος με τον οποίο θα αντιπροσωπεύονται τα ραντεβού σαν οντότητες, εδώ χρειάστηκε να γίνει ανάλυση των στοιχείων των οποίων απαρτίζουν ένα ραντεβού. Το ραντεβού όπως ορίζεται στις απαιτήσεις πρέπει να έχει συγκεκριμένα πεδία, στην υλοποίηση αυτά τα στοιχεία χωριστήκαν σε δύο κατηγορίες: τα στοιχεία που αφορούν αποκλειστικά το μέρος τα οποία τείνουν να είναι πιο στατικά και τα στοιχεία που μεταβάλλονται ανάλογα με το ραντεβού που εξαρτώνται κυρίως από τις επιλογές του χρήστη. Έτσι δημιουργήθηκαν οι κλάσεις Place και Appointment. Η κλάση Place περιέχει τα πεδία που χαρακτηρίζουν το μέρος και παραμένουν πάντοτε στατικά, αυτά είναι: οι συντεταγμένες μέρους, η διεύθυνση, ο τύπος μέρους και το ωράριο. Η κλάση Appointment αντιπροσωπεύει ένα ολοκληρωμένο ραντεβού, γι αυτό περιέχει ένα αντικείμενο της κλάσης Place που αντιπροσωπεύει το προορισμό του ραντεβού, επίσης διαθέτει τα πεδία μέρα και ώρα έναρξης, διάρκεια, ένα id για διευκόλυνση ταξινόμησης, και σε περιπτώσεις που ο προορισμός είναι κινηματογράφος τίτλος ταινίας και όνομα κινηματογράφου. Παρακάτω παρατίθενται οι constructors των κλάσεων Place και Appointment για αναφορά των πεδίων.

```
public Place(double lat,double lon, String address, String coordType,
int openHour,  int closeHour) {
    this.lat=lat;
    this.lon=lon;
    this.address=address;
    this.coordType=coordType;
    this.openHour=openHour;
    this.closeHour=closeHour;
}
```

```
public Appointment(int id, String dateString, int duration, Place
place) {

    this.dateString = dateString;
    this.id = id;
    this.place = place;
    this.duration = duration;
}
```

Εφόσον καθορίστηκε η δομή του ραντεβού το επόμενο βήμα ήταν ο καθορισμός δομής δεδομένων που θα χρειαζόταν για την διαχείριση πολλαπλών ραντεβού. Έτσι επιλέχθηκε η χρήση ενός ArrayList αντικειμένων Appointment. Η λίστα αυτή έχει όνομα appointArrayList, η λίστα στη συνέχεια μετατρέπεται σε λίστα από string και τοποθετείται μέσα σε ένα ArrayAdapter που διαχειρίζεται τα περιεχόμενα της λίστας μέσα σε ένα ListView που υλοποιήθηκε για την προβολή των ραντεβού από το χρήστη. Στη συνέχεια προστέθηκαν τα κουμπιά με τις βασικές λειτουργίες που απαιτούνταν για τη διαχείριση των ραντεβού, διαγραφή, δρομολόγηση και δημιουργία ραντεβού από τη λίστα. Για να επιλεχθεί ένα συγκεκριμένο ραντεβού από τη λίστα έγινε χρήση της ιδιότητας SINGLE CHOICE της ListView η οποία δίνει τη δυνατότητα επιλογής ενός στοιχείου κάθε φορά από τη λίστα και αναλόγως το

τι λειτουργία θα επιλέξει ο χρήστης μετά γίνεται και η ανάλογη ενέργεια στο επιλεγμένο ραντεβού. Το επόμενο κομμάτι ήταν η υλοποίηση κάποιου τρόπου αποθήκευσης και ανάκτησης της λίστας των ραντεβού ώστε ο κάθε λογαριασμός να έχει τη δικιά του λίστα με τα καταχωρημένα ραντεβού που έχει επιλέξει. Σε αυτό το σημείο έπρεπε να χρησιμοποιηθεί η βάση δεδομένων αλλά η λίστα των ραντεβού που χρειαζόταν να αποθηκευτεί έπρεπε να καταχωρηθεί στη βάση σε μορφή αντικειμένων κλάσεων, αυτό επιτεύχθηκε κάνοντας implement το interface Serialiazable στις κλάσεις Appointment και Place αυτό έδωσε τη δυνατότητα τα αντικείμενα να μετατραπούν σε byte array και έτσι να μπορέσουν να αποθηκευτούν σε ένα ιδιαίτερο column τύπου BLOB του πίνακα users το user_appointments, και όποτε χρειαζόταν η ανάκλησή τους να μετατρέπεται το byte array σε Serialized Object και να επιστρέφεται στην ίδια μορφή την οποία ήταν πριν το αποθηκεύσουμε. Τα δύο επικοινωνίας με τη βάση γίνονται κατά την callback method onCreate (η ανάκτηση από τη βάση) και κατά την callback method onDestroy (η αποθήκευση στη βάση) δηλαδή όταν δημιουργείται το activity διαβάζει τα δεδομένα από τη βάση και όταν τερματίσει τα αποθηκεύει.

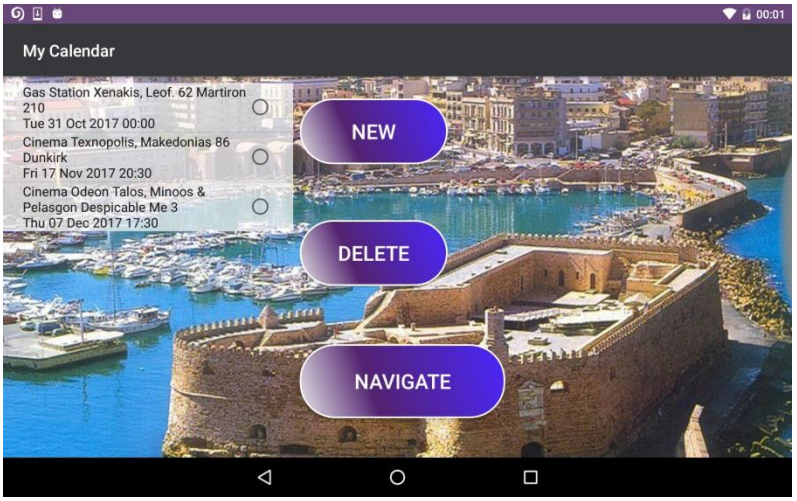
Οι μετατροπές των αντικειμένων σε Serialiazable και byte array back and forth που αναφέρθηκαν παραπάνω γίνονται με τη βοήθεια των παρακάτω μεθόδων της κλάσης DatabaseHelper.

```
public static byte[] getSerializedObject(Serializable s) {
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    ObjectOutputStream oos = null;
    try {
        oos = new ObjectOutputStream(baos);
        oos.writeObject(s);
    } catch (IOException e) {
        Log.e("IO Exception", e.getMessage(), e);
        return null;
    } finally {
        try {
            oos.close();
        } catch (IOException e) {
        }
    }
    byte[] result = baos.toByteArray();
    Log.i("getSerializedObject", "Object " +
s.getClass().getSimpleName() + "written to byte[]: " + result.length);
    return result;
}
public static Object readSerializedObject(byte[] in) {
    Object result = null;
    ByteArrayInputStream bais = new ByteArrayInputStream(in);
    ObjectInputStream ois = null;
    try {
        ois = new ObjectInputStream(bais);
        result = ois.readObject();
    } catch (Exception e) {
        result = null;
    } finally {
        try {
            ois.close();
        } catch (Throwable e) {
        }
    }
    return result;
}

public long updateSerializedObjectOfUser(SQLiteDatabase db,
Serializable myObject, String username) {
    try {
        db.beginTransaction();
        ContentValues values = new ContentValues();
        values.put(COLUMN_USER_APPOINTMENTS,
DatabaseHelper.getSerializedObject(myObject));
        long id = db.update(TABLE_USER, values, COLUMN_USER_USERNAME +
" = ?",
                new String[]{String.valueOf(username)});
        if (id >= 0) db.setTransactionSuccessful();
        return id;
    } catch (Exception e) {
        Log.e("SerializedDataInput", e.getMessage(), e);
        // ignore this and roll back the transaction
    } finally {
        try {
            db.endTransaction();
        } catch (Exception e) {
            return -1;
        }
    }
    return -1;
}
```

```
public Object readSerializedObjectOfUser(String username) {
    Object returnedObject=null;
    // array of columns to fetch
    String[] columns = {
            COLUMN_USER_USERNAME,
            COLUMN_USER_APPOINTMENTS,
    };
    SQLiteDatabase db = this.getReadableDatabase();
    // selection criteria
    String selection = COLUMN_USER_USERNAME + " = ?";

    // selection arguments
    String[] selectionArgs = {username};

    // query user table with conditions
    /**
     * Here query function is used to fetch records from user table
this function works like we use sql query.
     * SQL query equivalent to this query function is
     * SELECT user_id FROM user WHERE user_username = 'username-
example' AND user_password = 'qwerty';
     */
    Cursor cursor = db.query(TABLE_USER, //Table to query
            columns,                    //columns to return
            selection,                  //columns for the WHERE clause
            selectionArgs,              //The values for the WHERE
clause
            null,                       //group the rows
            null,                       //filter by row groups
            null);                      //The sort order

    int cursorCount = cursor.getCount();
    if (cursorCount > 0) {
        if (cursor.moveToFirst()) {
            do {
                byte[] returnedBlob =
cursor.getBlob(cursor.getColumnIndex(COLUMN_USER_APPOINTMENTS));
                if(returnedBlob!=null){
                    returnedObject =
DatabaseHelper.readSerializedObject(returnedBlob);
                }

            } while (cursor.moveToNext());
        }
    }
    cursor.close();
    db.close();
    return returnedObject;
}
```

Σαν επόμενο βήμα ήταν η υλοποίηση των βασικών λειτουργιών διαχείρισης ραντεβού, η δημιουργία η δρομολόγηση και η διαγραφή. Η διαγραφή επιτεύχθηκε με την υλοποίηση ενός κουμπιού το οποίο όταν πατιέται διαβάζει την τρέχουσα θέση του επιλεγμένου από τον χρήστη στοιχείο στο ListView, το διαγράφει από τη λίστα με τα string στην οποία είναι συνδεδεμένο το adapter, κάνει notifyDataSetChanged για να ενημερώσει το adapter και τα περιεχόμενα της ListView και τέλος διαγράφει το ίδιο στοιχείο από την appointArrayList που είναι αποθηκευμένα τα αντικείμενα των ραντεβού. Η δρομολόγηση ξεκινάει με το πάτημα του κουμπιού Navigate το οποίο διαβάζει το επιλεγμένο στοιχείο στη λίστα και στέλνει όλη τη λίστα των ραντεβού σαν Extras σε Intent στο activity PlanMap που διαχειρίζεται αποκλειστικά την δρομολόγηση με συνθήκες όλων των ραντεβού, η λειτουργία του PlanMap θα αναλυθεί παρακάτω. Όταν τελειώσει την δρομολόγηση το PlanMap η εφαρμογή επιστρέφει στο activity Organizer. Η δημιουργία νέου ραντεβού πραγματοποιείται με το πάτημα του κουμπιού new το οποίο στέλνει και αυτό όλη τη λίστα των ραντεβού σαν Parcelable Extras σε Intent στο activity NewAppointment από το οποίο αρχίζει η διαδικασία δημιουργίας νέου ραντεβού η οποία θα περιγραφεί παρακάτω. Όταν τελειώσει η διαδικασία δημιουργίας ραντεβού το activity NewAppointment επιστρέφει ένα result code, αν αυτό έχει τιμή RESULT_OK σημαίνει πως το ραντεβού δημιουργήθηκε με επιτυχία, σε αυτή τη περίπτωση εισάγεται μια νέα εγγραφή στη λίστα appointArrayList από τις παραμέτρους που επιστράφηκαν από το NewAppointment Activity και ενημερώνονται με τη σειρά τους η stringlist και το adapter για να εμφανίσουν το νέο ραντεβού στο ListView. Σε περίπτωση που δεν επιστράφηκε RESULT_OK τότε οι λίστες μένουν ως έχουν. Παρακάτω απεικονίζεται η οθόνη του activity με τρία καταχωρημένα ραντεβού.

Organizer Activity Layout 1

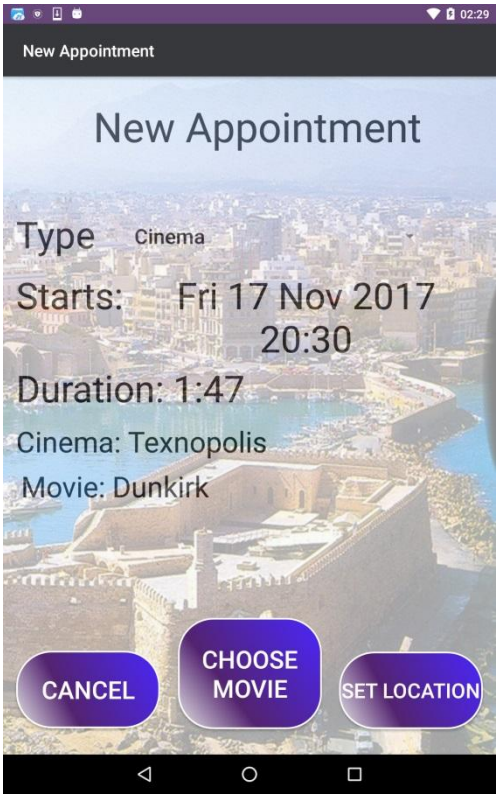


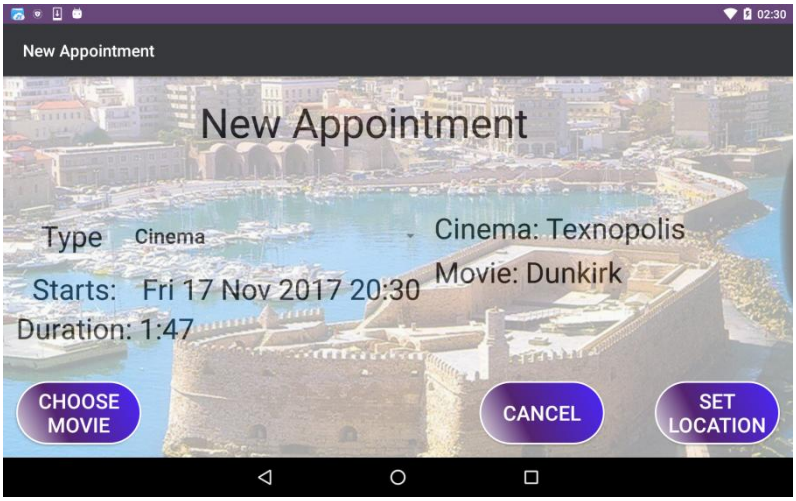
5.3.2 NewAppointment

Ρόλος: Το NewAppointment αποτελεί μια φόρμα με τα βασικά στοιχεία ενός ραντεβού, ο κύριος ρόλος του είναι να προβάλει στο χρήστη τα στοιχεία που έχει επιλέξει ενώ παράλληλα να του δώσει τη δυνατότητα να εισάγει καινούρια ή να μεταβάλει υπάρχοντα στοιχεία όσον αφορά το καινούριο ραντεβού.

Εκτέλεση: Για το NewAppointment αρχικά σχεδιάστηκε το layout το οποίο περιλαμβάνει ένα spinner για την επιλογή τύπου του ραντεβού, TextViews με τα στοιχεία ώρας, μέρας, διάρκειας, και δύο παραπάνω TextViews που αναγράφουν τον τίτλο της ταινίας και το όνομα του κινηματογράφου και εμφανίζονται μόνο όταν έχει επιλεγεί ραντεβού τύπου cinema. Το layout διαθέτει επίσης κουμπιά για επιλογή ταινίας, ημέρας, ώρας, για εύρεση κοντινότερου σημείου, έλεγχο εγκυρότητας ραντεβού και τέλος για επιβεβαίωση καταχώρησης του ραντεβού, πολλά από αυτά τα κουμπιά αποκρύπτονται και επανεμφανίζονται ανάλογα με το στάδιο στο οποίο βρίσκεται το activity σε σχέση με την επιλογή των στοιχείων του ραντεβού, τα διάφορα στάδια του activity θα αναλυθούν παρακάτω. Παρατίθενται τα υποδείγματα του layout σε ένα στάδιο επιλεγμένου ραντεβού cinema.

NewAppointment Activity Layout 1





Σαν πρώτο βήμα στο activity γίνεται η επιλογή τύπου ραντεβού από το spinner, όταν ο χρήστης επιλέξει τον τύπο του μέρους που τον ενδιαφέρει (supermarket ,gas station, cinema) , τότε το layout εμφανίζει και αποκρύπτει τα ανάλογα views που αντιστοιχούν στον τύπο ραντεβού που επιλέχθηκε. Εδώ υπάρχουν 3 κύρια modes ανάλογα με τον τύπο του μέρους και ελέγχονται με το string "mode", όταν επιλεχθεί supermarket το mode παίρνει τιμή "supermarket" όταν επιλεχθεί κινηματογράφος το mode παίρνει τιμή "cinema" κοκ.

Στο mode supermarket εμφανίζεται το spinner τα TextViews για την ημερομηνία και ώρα έναρξης και τη διάρκεια του ραντεβού καθώς και τα κουμπιά Set Date & Time και Set Location. Το κουμπί Set Date & Time μεταφέρει την εφαρμογή στο activity υπεύθυνο για επιλογή ημέρας και ώρας για τα ραντεβού τύπου supermarket και gas station που ονομάζεται DateTimePick Activity και θα περιγραφεί παρακάτω. Το κουμπί Set Location μεταφέρει την εφαρμογή στο PlanMap Activity που είναι υπεύθυνο για τον έλεγχο εγκυρότητας του ραντεβού καθώς και την εύρεση του κοντινότερου προορισμού. Το PlanMap Activity θα αναλυθεί εκτενώς παρακάτω. Στο mode gas station υπάρχουν ακριβώς τα ίδια Views με τη διαφορά πως το πεδίο duration αλλάζει από 40 λεπτά που είναι ο αναμενόμενος χρόνος ραντεβού για το supermarket σε 5 λεπτά που είναι η διάρκεια ραντεβού για gas station.

Στο mode cinema εμφανίζονται πάλι τα ίδια Views με τις διαφορές ότι πρώτον εμφανίζονται και 2 επιπλέον TextViews για την αναγραφή τίτλου ταινίας και όνομα κινηματογράφου και δεύτερον αποκρύπτεται το κουμπί Set Date & Time και στη θέση του εμφανίζεται το κουμπί Choose Movie που ξεκινάει το MoviePicker Activity στο οποίο ο χρήστης μπορεί να επιλέξει ταινία, κινηματογράφο, ώρα και μέρα προβολής.

Το activity διαθέτει και ένα τέταρτο mode με την τιμή "ragnarok" όταν καλείται το συγκεκριμένο mode σημαίνει πως επιστράφηκαν επιτυχώς οι συντεταγμένες του κοντινότερου μέρους και ότι το ραντεβού έχει περάσει τον έλεγχο εγκυρότητας από το PlanMap. Ο ρόλος του συγκεκριμένου σταδίου είναι να κλειδώνει τις επιλογές των στοιχείων του ραντεβού που έκανε ο χρήστης και να ζητά από τον ίδιο την τελευταία επιβεβαίωση καταχώρησης του ραντεβού στη λίστα. Στο συγκριμένο στάδιο το activity αποκρύπτει όλα τα διαδραστικά εργαλεία επεξεργασίας στοιχείων του ραντεβού τα οποία είναι το spinner και τα κουμπιά Set Date & Time, Set Location και Choose Movie. Στη θέση του Set Location εμφανίζεται το κουμπί Confirm το οποίο όταν πατηθεί επιβεβαιώνεται η καταχώρηση του ραντεβού το οποίο αποστέλλεται μέσω Extras σε Intent στην λίστα του Organizer Activity. Όσον αφορά τα TextViews κατά το mode ragnarok εμφανίζονται μόνο αυτά στα οποία αναγράφονταν τα στοιχεία του ραντεβού πριν καλεστεί το ragnarok, αυτό γίνεται με τη χρήση ενός δεύτερου string ελέγχου με το όνομα preMode στο οποίο αποθηκεύεται το mode στο οποίο βρισκόταν το activity πριν μεταβεί στο PlanMap επομένως πριν γίνει η επιστροφή αποτελεσμάτων από το PlanMap και καλεστεί το mode ragnarok.

Τέλος αξίζει να σημειωθεί η διαδικασία με την οποία διατηρούνταν τα περιεχόμενα των TextViews κατά την περιστροφή του layout, όπως και η κατάσταση των κουμπιών (ορατά ή αόρατα). Κάθε φορά που περιστρέφεται η συσκευή το layout αλλάζει προσανατολισμό, κατά την αλλαγή αυτή αναδημιουργείται καλώντας εκ νέου την callback method onCreate, όταν γίνεται αυτό κάποια στοιχεία τα οποία έχουν καθοριστεί ως μη στατικά διατηρούνται, τέτοια στοιχεία είναι για παράδειγμα τα περιεχόμενα ενός EditText που αναμένεται να έχουν αλλάξει από την έναρξη της εφαρμογής μέχρι να χρειαστεί περιστροφή καθώς αλλάζουν άμεσα από το χρήστη. Η κατάσταση όμως ορισμένων στοιχείων που θεωρούνται πιο στατικά δεν αποθηκεύονται, τέτοια στοιχεία είναι τα TextViews που χρησιμοποιούνται στο παρόν activity για προβολή μεταβλητών δεδομένων, άρα χρειάζεται να γίνει η διατήρηση αυτών των δεδομένων με κάποιο τρόπο.

Η διατήρηση αυτή επιτεύχθηκε με την αποθήκευση και ανάκληση αυτών των δεδομένων στην savedInstantState με τις callback methods onRestoreInstanceState και onSaveInstanceState καθώς και υλοποίηση ανάλογων εκχωρήσεων μέσα σε συνθήκες στην onCreate. Η αποθήκευση και ανάκληση αυτών των δεδομένων φαίνονται στα παρακάτω κομμάτια κώδικα.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_new_appointment);
    if (savedInstanceState != null) {
        dateString = savedInstanceState
                .getString("dateStringKey", "");
        cinemaStringLabel = savedInstanceState
                .getString("cinemaStringLabelKey", "");
        movieStringLabel = savedInstanceState
                .getString("movieStringLabelKey", "");
        duration = savedInstanceState
                .getString("durationStringKey", "");
        movieDuration = savedInstanceState
                .getString("movieDurationStringKey", "");
        mode = savedInstanceState
                .getString("modeKey");
        preMode = savedInstanceState
                .getString("preModeKey");
        moviePickedDate = savedInstanceState
                .getString("moviePickedDateKey");
        receivedDestination = savedInstanceState
                .getParcelable("destinationKey");
    } else {
        dateString = "";
        duration = "";
        cinemaStringLabel = "";
        movieStringLabel = "";
        movieDuration="";
        mode = "Supermarket";
        moviePickedDate = "";
        preMode="";
        receivedDestination = null;
    }
```

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    savedInstanceState
                .putString("dateStringKey",dateString);
    savedInstanceState
                .putString("cinemaStringLabelKey",cinemaStringLabel);
    savedInstanceState
                .putString("movieStringLabelKey",movieStringLabel);
    savedInstanceState
                .putString("durationStringKey",duration);
    savedInstanceState
                .putString("modeKey",mode);
    savedInstanceState
                .putString("moviePickedDateKey",moviePickedDate);
    savedInstanceState
                .putParcelable("destinationKey",receivedDestination);
    savedInstanceState
                .putString("movieDurationKey",movieDuration);
    savedInstanceState
                .putString("preModeKey",preMode);
}

@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    dateString = savedInstanceState
                .getString("dateStringKey");
    cinemaStringLabel = savedInstanceState
                .getString("cinemaStringLabelKey");
    movieStringLabel = savedInstanceState
                .getString("movieStringLabelKey");
    duration = savedInstanceState
                .getString("durationStringKey");
    mode = savedInstanceState
                .getString("modeKey");
    preMode = savedInstanceState
                .getString("preModeKey");
    moviePickedDate = savedInstanceState
                .getString("moviePickedDateKey");
    receivedDestination = savedInstanceState
                .getParcelable("destinationKey");
    movieDuration=savedInstanceState
                .getString("movieDurationKey");
}
```

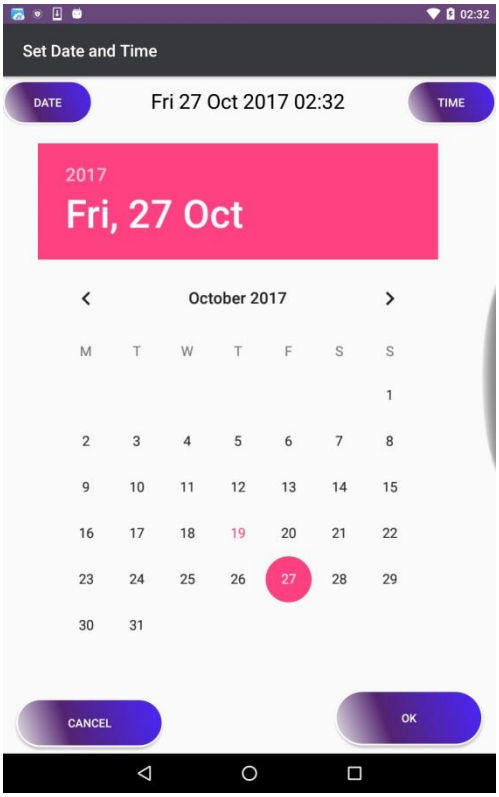
5.3.3 DateTimePick

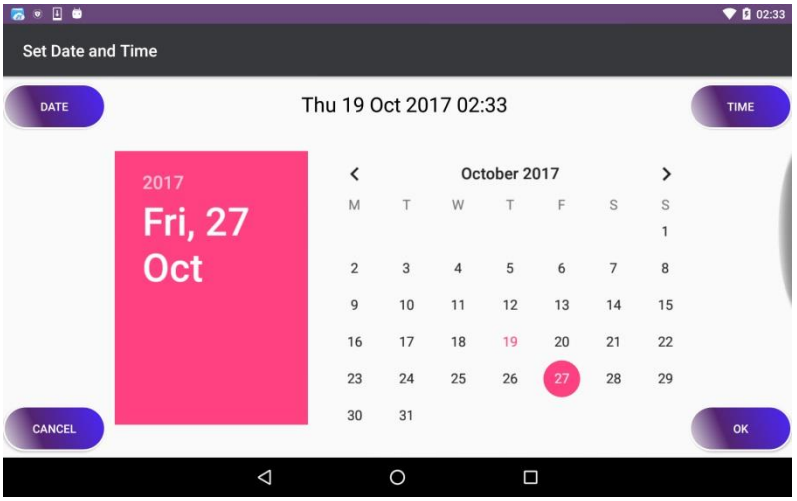
Ρόλος: Το DateTimePick είναι υπεύθυνο για την επιλογή μέρας και ώρας του επιλεγμένου ραντεβού για τύπους supermarket και gas station

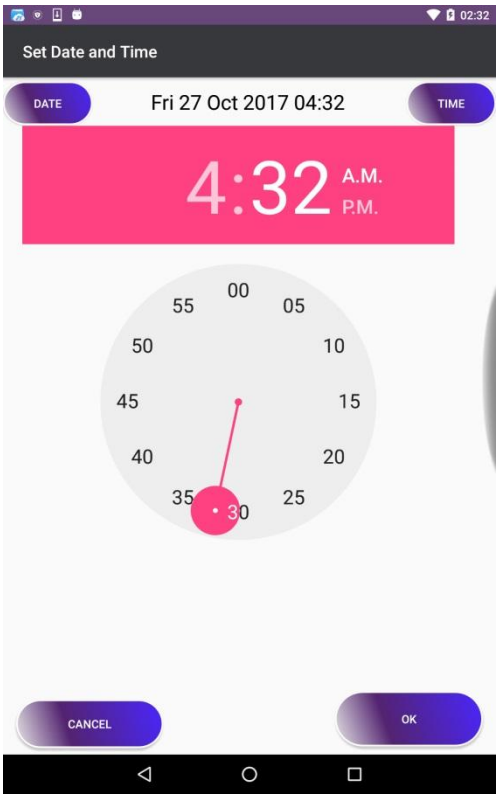
Εκτέλεση: Ο σχεδιασμός του layout για το DateTimePick Activity είναι σχετικά απλός, περιλαμβάνει ένα DatePicker και ένα TimePicker τα οποία βρίσκονται στην ίδια θέση και εναλλάσσουν ορατότητα με το πάτημα των αντίστοιχων κουμπιών Date και Time, υπάρχει επίσης ένα TextView που δείχνει την επιλεγμένη ώρα με χρήση SimpleDateFormat που μορφοποιεί το αντικείμενο Date σε string της μορφής "ημέρα εβδομάδας, ημέρα μήνα, μήνας, έτος, ώρα, λεπτά", επίσης υπάρχουν τα κουμπιά OK και Cancel για καταχώρηση της επιλεγμένης ώρας ή ακύρωση αντίστοιχα. Οι προκαθορισμένες τιμές των DatePicker και TimePicker αντιστοιχούν στην τρέχουσα ημερομηνία και ώρα. Υπάρχουν επίσης έλεγχοι που αποτρέπουν τον χρήστη από το να εισάγει παρελθοντική ημερομηνία και ώρα, ειδοποιώντας τον με κατάλληλο μήνυμα. Τέλος η λειτουργία του κουμπιού OK αποτελείται από το διάβασμα του string που βρίσκεται στο TextView το οποίο περιγράφει την επιλεγμένη ημερομηνία και ώρα και αφού γίνει ο έλεγχος εγκυρότητας αποστέλλει αυτό το string σαν Extra σε Intent πίσω στο NewAppointment Activity όπου ενημερώνονται τα κατάλληλα πεδία. Γίνεται και εδώ χρήση παρόμοιων τεχνικών στην onCreate για την διατήρηση τιμών στο TextView κατά την περιστροφή της συσκευής και την αλλαγή προσανατολισμού του layout.

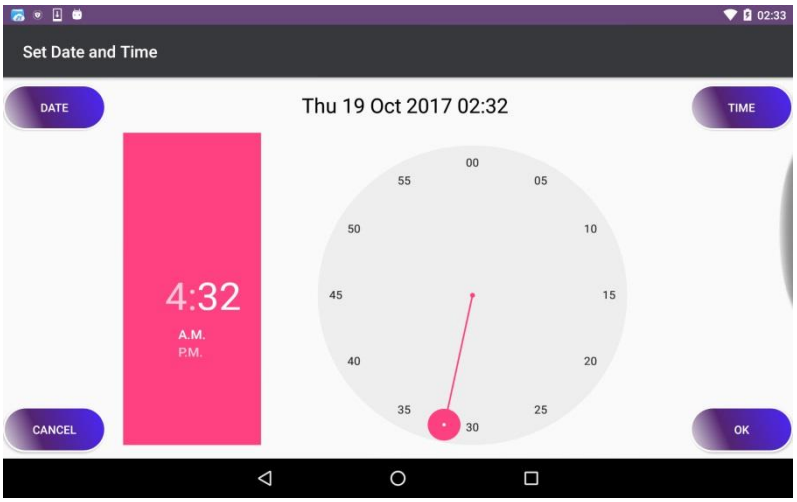
Παρακάτω παρατίθενται οι οθόνες του DateTimePick κατά την επιλογή μέρας και ώρας αντίστοιχα.

DateTimePick Activity Layouts 1









5.3.4 MoviePicker

Ρόλος: Ο ρόλος του MoviePicker είναι η διαχείριση επιλογής ταινίας, κινηματογράφου, μέρας και ώρας που επιθυμεί ο χρήστης.

Εκτέλεση: Η υλοποίηση του MoviePicker αποτελείται από δύο κύρια layout τα οποία εναλλάσσονται αναλόγως με το στάδιο επιλογής στο οποίο βρίσκεται ο χρήστης.

Στο πρώτο στάδιο ο χρήστης χρειάζεται να επιλέξει την ταινία, αυτό γίνεται με πάτημα πάνω στο στοιχείο μιας scrollable λίστας η οποία διαθέτει γραμμές με την εικόνα της ταινίας, τον τίτλο και την περιγραφή της. Η λίστα αυτή αποτελείται από ένα ScrollView μέσα στο οποίο βρίσκεται ένα TableLayout, αυτό περιλαμβάνεται από TableRows που περιέχουν τα στοιχεία της κάθε ταινίας τα οποία είναι: ένα ImageView για την εικόνα της ταινίας, ένα TextView για τον τίτλο της και ένα TextView με μια σύντομη περιγραφή και άλλες βασικές πληροφορίες για την ταινία όπως όνομα σκηνοθέτη είδος ταινίας διάρκεια κ.α. Σε κάθε ImageView και TextView που εμπεριέχονται στα TableRows υλοποιείται ένας ClickListener που κάνει το activity να αλλάξει layout με χρήση της μεθόδου createMoviePanel η οποία παίρνει σαν όρισμα ένα string με την επιλεγμένη ταινία και εκτελεί όλες τις λειτουργίες της επόμενης οθόνης από το δεύτερο layout που παραμένει όμως στο ίδιο activity.

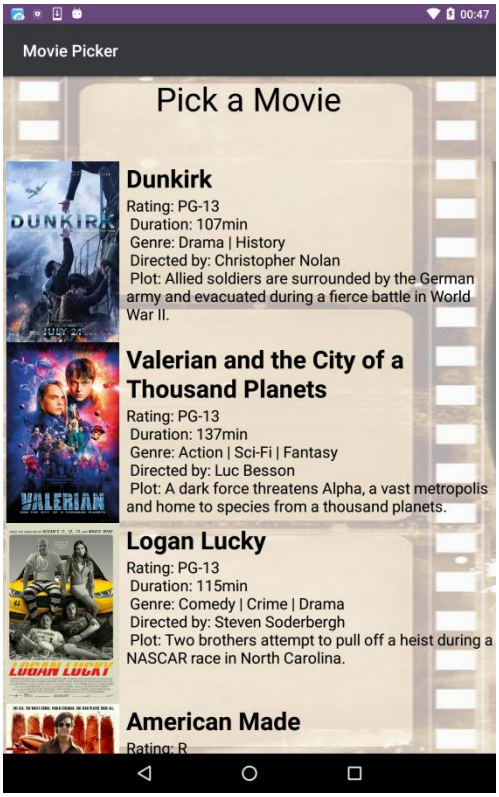
Το δεύτερο στάδιο περιλαμβάνει την επιλογή διαθέσιμου κινηματογράφου, ώρας, μέρας της εβδομάδας και τέλος ημερομηνίας για την οποία θέλει να κλείσει το ραντεβού ο χρήστης. Το layout αυτό περιλαμβάνεται από ένα ImageView με την εικόνα της ταινίας, ένα TextView με τον τίτλο της και ένα TextView με μια πιο αναλυτική περιγραφή της πλοκής όπως και άλλων πληροφοριών της ταινίας. Η επιλογή μέρας της εβδομάδας γίνεται με τη χρήση μιας προκαθορισμένης ListView τα περιεχόμενα της οποίας αλλάζουν αναλόγως με την ταινία που έχει επιλεχθεί για να ταιριάζουν στις πραγματικές μέρες εβδομάδος στις οποίες προβάλλεται η ταινία, δίπλα από την λίστα με τις διαθέσιμες μέρες της εβδομάδος υπάρχει μια λίστα με τις ώρες προβολής και το όνομα του κινηματογράφου για την επιλεγμένη μέρα. Το layout διαθέτει επίσης το κουμπί Cancel που όταν πατηθεί αλλάζει και πάλι το layout και μεταβαίνει στην προηγούμενη οθόνη επιλογής ταινίας. Στη δεξιά γωνία της οθόνης υπάρχει το κουμπί Set Date το οποίο όταν πατηθεί γίνεται έλεγχος αν έχουν επιλεχθεί στοιχεία και στη λίστα μερών της εβδομάδος και στη λίστα ωρών προβολής, αν αυτό συμβαίνει αποκρύπτονται οι δύο λίστες μαζί με τα κουμπιά και στην θέση τους εμφανίζεται μια νέα ListView με τις ημερομηνίες που αντιστοιχούν στην επιλεγμένη μέρα εβδομάδος και ώρα προβολής για τους επόμενους δύο μήνες, ο υπολογισμός των ημερομηνιών αυτών γίνεται από τον παρακάτω κώδικα.

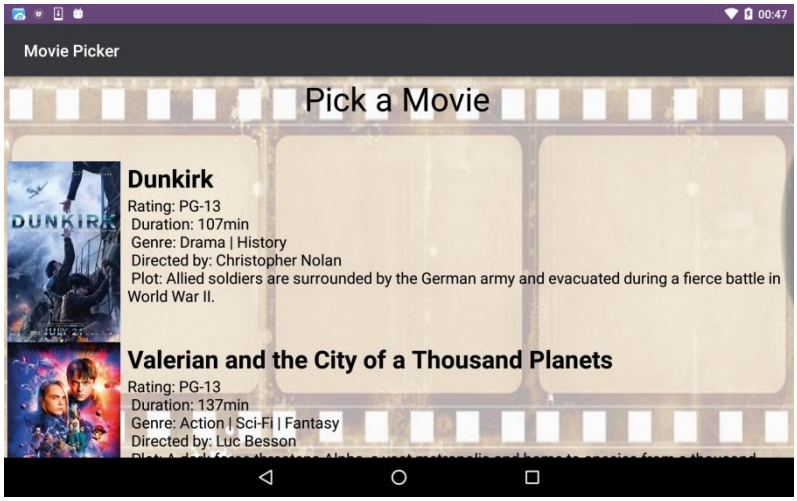
```
@Override
public void onClick(View v) {
    int mins, movieHour, movieMin;
    String tempStr;
    Date temp;
    if (daysRadioList
                .getCheckedItemPosition() != -1) {
        if (hoursRadioList
                .getCheckedItemPosition() != -1) {
            dayString = daysList.get(daysRadioList
                .getCheckedItemPosition());
            cinemaString = hoursList.get(hoursRadioList
                .getCheckedItemPosition());
            cinemaString = hoursList.get(hoursRadioList
                .getCheckedItemPosition()).substring(0
                ,cinemaString.indexOf(":") - 2);
            tempStr = hoursList.get(hoursRadioList
                .getCheckedItemPosition());
            movieHour = Integer.parseInt(hoursList
                .get(hoursRadioList.getCheckedItemPosition())
                .substring(tempStr.indexOf(":") - 2
                ,tempStr.indexOf(":")));
            movieMin = Integer.parseInt(hoursList
                .get(hoursRadioList.getCheckedItemPosition())
                .substring(tempStr.indexOf(":") + 1
                ,tempStr.indexOf(":") + 3));

            mins = Integer
                .parseInt(moviePickedInfo
                .getText().toString()
                .substring(moviePickedInfo.getText()
                .toString().indexOf("Duration: ") + 10
                ,moviePickedInfo.getText()
                .toString().indexOf("min")));
            hour = mins / 60;
            min = mins % 60;
            movieDuration = (mins / 60) + ":" + (mins % 60);
            SimpleDateFormat dayOfWeek = new SimpleDateFormat("EEEE"
                , Locale.getDefault());
            Calendar cal = Calendar.getInstance();
            Date currDate = cal.getTime();
            while (!dayOfWeek.format(currDate).equals(dayString)) {
                cal.setTime(currDate);
                cal.add(Calendar.DATE, 1);
                currDate = cal.getTime();
            }
            cal.set(Calendar.HOUR_OF_DAY, movieHour);
            cal.set(Calendar.MINUTE, movieMin);
            movieDates = new ArrayList<>();
            SimpleDateFormat fullDate = new SimpleDateFormat("EEE dd
                MMM yyyy HH:mm", Locale.getDefault());
            TimeZone timeZone;
            timeZone = TimeZone.getTimeZone("Europe/Athens");
            fullDate.setTimeZone(timeZone);
            for (int i = 0; i < 10; i++) {
                temp = cal.getTime();
                movieDates.add(fullDate.format(temp));
```

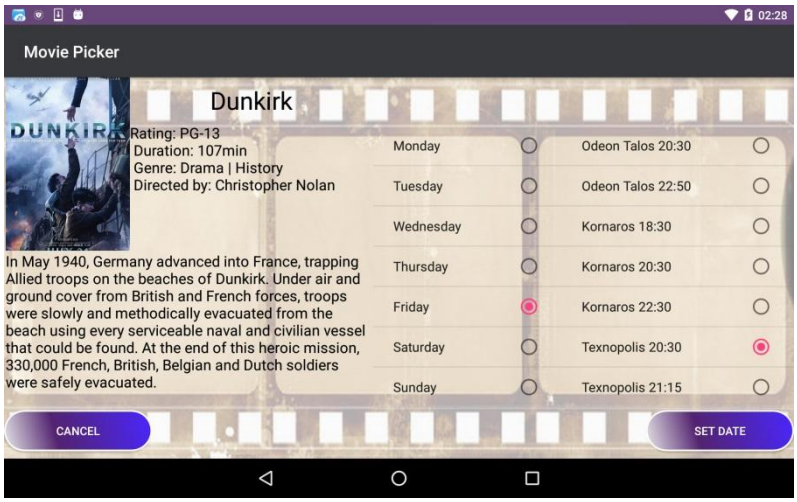
```
            cal.add(Calendar.WEEK_OF_YEAR, 1);
                }
                mode = "DatePick";
                setUpListViews(mode);

            }

        }
}
```

Στη θέση του κουμπιού Set Date εμφανίζεται το κουμπί Confirm το οποίο όταν πατηθεί αποστέλλει όλα τα επιλεγμένα στοιχεία του ραντεβού στο activity NewAppointment και τερματίζει το activity MoviePicker εφόσον υπάρχει επιλεγμένο στοιχείο στη λίστα με τις ημερομηνίες. Στη θέση του κουμπιού Cancel εμφανίζεται το κουμπί Back του οποίου η λειτουργία είναι όταν πατηθεί να επιστρέψει το layout στην προηγούμενη κατάσταση εμφανίζοντας τα προηγούμενα κουμπιά και λίστες ενώ αποκρύπτει τα νέα κουμπιά και ListView που είχαν εμφανιστεί με το πάτημα του Set Date. Οι επιλογές που κάνει ο χρήστης στις λίστες καθώς και τα περιεχόμενα των TextViews διατηρούνται με τον ίδιο τρόπο που αναφέρθηκε στην περιγραφή του NewAppointment στο κεφάλαιο 5.3.2. Παρακάτω απεικονίζονται τα δύο layout του MoviePicker κατά την επιλογή ταινίας και ώρας προβολής.

MoviePicker Layouts 1







5.5 Διαχείριση Δρομολόγησης/Χαρτών

5.4.2 PlanMap

Ρόλος: Ο ρόλος του PlanMap σε πρώτο σκέλος είναι να κάνει έλεγχο εγκυρότητας του ραντεβού και έπειτα κάνοντας δρομολόγηση να βρει το κοντινότερο σημείο για τον τύπο μέρους που έχει καθοριστεί από το ραντεβού. Σε δεύτερο σκέλος είναι να χρησιμοποιηθεί για την ίδια την δρομολόγηση υπάρχοντος ραντεβού αφού κάνει όμως πάλι τον απαραίτητο έλεγχο εγκυρότητας.

Εκτέλεση: Σε πρώτο βήμα σχεδιάστηκε το UI του PlanMap το οποίο περιελάμβανε το fragment του χάρτη, ένα TextView για την προβολή πληροφοριών προορισμού και ένα κουμπί για επιβεβαίωση της καταχώρησης τοποθεσίας από τον χρήστη. Έπειτα υλοποιήθηκαν οι απαραίτητες λειτουργίες του χάρτη. Η πρώτη ήταν η εύρεση της τρέχουσας θέσης του χρήστη, για αυτή χρησιμοποιήθηκε το fusedLocationApi από το GoogleApi, προτού γίνει η κλήση όμως για δημιουργία αντικειμένου location με τη θέση του χρήστη έπρεπε να γίνει έλεγχος διαθεσιμότητας ενεργού δικτύου με πρόσβαση στο internet και ενεργού GPS δέκτη για πρόσβαση στο δορυφόρο. Εδώ υπάρχουν δύο επιπλέον callback methods που σχετίζονται με το lifecycle του χάρτη, η onMapReady που καλείται όταν ο χάρτης είναι έτοιμος προς χρήση και παρέχει ένα αντικείμενο της GoogleMap το οποίο δεν είναι null, και η onConnected η οποία καλείται ασύγχρονα όταν το connection request στην υπηρεσία ολοκληρωθεί επιτυχώς. Πρώτα από όλα πρέπει να καθοριστεί ότι υπάρχει διαθέσιμο δίκτυο με σύνδεση στο internet ώστε η επικοινωνία με την υπηρεσία να είναι εφικτή, καλείται λοιπόν ένας έλεγχος διαθεσιμότητας δικτύου στην μέθοδο onCreate προτού δημιουργηθεί οποιοδήποτε αντικείμενο σχετικό με τον χάρτη, αν υπάρχει ήδη ενεργό δίκτυο η ροή προχωράει κανονικά, αν δεν υπάρχει καλείται ένα alertDialog το οποίο πληροφορεί τον χρήστη ότι δεν είναι ενεργοποιημένο κάποιου είδους συνδεσιμότητα στο internet και το activity τερματίζει. Αφού περάσει ο έλεγχος συνδεσιμότητας δικτύου γίνεται ο έλεγχος δέκτη GPS μέσα στην onConnected αλλά πριν δημιουργηθεί αντικείμενο location για τη θέση χρήστη. Αν το GPS είναι ήδη ενεργοποιημένο τότε η ροή προχωράει κανονικά, αν δεν είναι τότε καλείται η μέθοδος startResolutionForResult η οποία εμφανίζει ένα dialog στον χρήστη προειδοποιώντας τον ότι η εφαρμογή χρειάζεται ενεργοποίηση του δέκτη GPS, πατώντας την επιλογή OK του dialog ο χρήστης ενεργοποιεί αυτόματα το GPS της συσκευής του χωρίς να ανοίξει τις ρυθμίσεις ή κάποια άλλη εξωτερική εφαρμογή.

Αφού τελειώσει ο έλεγχος συνδεσιμότητας η ροή περνάει στον έλεγχο αδειών (permissions) σε παλαιότερες εκδόσεις τα permissions ορίζονταν κατά την εγκατάσταση της εφαρμογής από το Manifest, από την έκδοση Android 6.0 και μετά όλες οι εφαρμογές υποχρεούνται να ζητάνε permissions στο runtime (κατά την εκτέλεση της εφαρμογής) από τον χρήστη. Στην παρούσα εφαρμογή έχουν υλοποιηθεί και οι δύο τρόποι για να καλυφθούν και συσκευές με παλαιότερο λειτουργικό σύστημα. Ο έλεγχος permission γίνεται για πρώτη φορά στο Home Activity, και μετά στο InstantMap ή PlanMap αναλόγως με το ποιο θα επιλεγεί πρώτο. Παρακάτω φαίνονται οι έλεγχοι συνδεσιμότητας, αδειών και η υλοποίηση του fusedLocationApi μέσα στον κώδικα:

Έλεγχος Δικτύου και GPS 1

```
if (!isNetworkConnected()) {
    exitMap("noInternet");
}
```

```
protected void checkSettings() {
    if (!isNetworkConnected()) {
        return;
    }
    createLocationRequest();
    LocationSettingsRequest.Builder builder = new
LocationSettingsRequest.Builder()
            .addLocationRequest(mLocationRequest);
    SettingsClient client = LocationServices.getSettingsClient(this);
    Task<LocationSettingsResponse> task =
client.checkLocationSettings(builder.build());
    task.addOnSuccessListener(this, new
OnSuccessListener<LocationSettingsResponse>() {
        @Override
        public void onSuccess(LocationSettingsResponse
locationSettingsResponse) {
            createLocationRequest();
            if (location == null) {
                Toast.makeText(PlanMap.this, "Location Request Failed,
Retrying...", Toast.LENGTH_SHORT);
                for (int i = 0; i < 4; i++) {
                    if (location != null) {
                        break;
                    } else if (location == null && i == 3) {
                        exitMap("connection_failed");
                    } else {
                        try {
                            Thread.sleep(2000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                        createLocationRequest();
                    }
                }
            }
        }
    });
    task.addOnFailureListener(this, new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            int statusCode = ((ApiException) e).getStatusCode();
            switch (statusCode) {
                case CommonStatusCodes.RESOLUTION_REQUIRED:
                    try {
                        ResolvableApiException resolvable =
(ResolvableApiException) e;
                      resolvable.startResolutionForResult(PlanMap.this,
                                REQUEST_CHECK_SETTINGS);
                    } catch (IntentSender.SendIntentException sendEx) {
                    }
                    break;
                case
LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:
                    break;
            }
        }
    });
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    // Check which request we're responding to
    if (requestCode == REQUEST_CHECK_SETTINGS) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {

            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            mapInit();
            startNavigation();
        }
    } else {
        Toast toasty = Toast.makeText(PlanMap.this,
"gps remains disabled", Toast.LENGTH_SHORT);
        toasty.show();
    }
}
```

Permission Check & Create User Location 1

```
protected void createLocation() {
    if (ContextCompat.checkSelfPermission(PlanMap.this,
android.Manifest.
            permission.ACCESS_FINE_LOCATION) != PackageManager.
            PERMISSION_GRANTED) {
        if
(ActivityCompat.shouldShowRequestPermissionRationale(PlanMap.this,
                Manifest.permission.ACCESS_FINE_LOCATION)) {
            Toast toast = Toast.makeText(PlanMap.this,
"You previously revoked the Location permission", Toast.LENGTH_SHORT);
            toast.show();
            ActivityCompat.requestPermissions(PlanMap.this,
                    new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    PERMISSIONS REQUEST ACCESS FINE LOCATION);
        } else {
            ActivityCompat.requestPermissions(PlanMap.this,
                    new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    PERMISSIONS REQUEST ACCESS FINE LOCATION);
        }
    }

    location = FusedLocationApi.getLastLocation(mGoogleApiClient);
}
```

```
@Override
public void onRequestPermissionsResult(int requestCode, String
permissions[],
                                       int[] grantResults) {
    switch (requestCode) {
        case PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION: {
            if (grantResults.length > 0
                    && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                Toast toast = Toast.makeText(PlanMap.this,
"permission granted", Toast.LENGTH_SHORT);
                toast.show();
            } else {
            }
            return;
        }
    }
}
```

Αφού ολοκληρωθούν οι έλεγχοι και η εύρεση της τρέχουσας θέσης του χρήστη το activity προχωράει στον έλεγχο εγκυρότητας ραντεβού ο τρόπος λειτουργίας του οποίου αναλύθηκε στο κεφάλαιο 5.2. Εδώ το activity πληροφορεί τον χρήστη για την εγκυρότητα του ραντεβού συνήθως με χρήστη μηνυμάτων Toast, εκτός από την περίπτωση αδυναμίας εύρεσης προορισμού στην ίδια κατεύθυνση με την πορεία του χρήστη που αναφέρεται στο κεφάλαιο 5.2.4 Έλεγχος κατεύθυνσης. Στη συγκεκριμένη περίπτωση ο χρήστης ερωτάται αν θέλει να βγει από το δρόμο του για να προλάβει και τα δύο ραντεβού ή αν θέλει να ακυρώσει το τρέχον ραντεβού, το ερώτημα γίνεται με χρήση ενός Dialog στο οποίο ο χρήστης μπορεί να επιλέξει τι θέλει να κάνει. Σε περίπτωση που το ραντεβού είναι έγκυρο το activity προσπαθεί να βρει το κοντινότερο σημείο με τύπο μέρους (supermarket/cinema/gas station) που να ανταποκρίνεται στο επιλεγμένο για το τρέχον ραντεβού.

Εδώ πρέπει να σημειωθεί πως οι συντεταγμένες σημείων ενδιαφέροντος είναι αποθηκευμένες στατικά μαζί με όλα τα άλλα στοιχεία του σημείου σε τρεις λίστες με αντικείμενα της κλάσης Place, για το λόγο ότι αν γινόταν request στην υπηρεσία της Google κάθε φορά που χρειαζόταν να βρεθούν τα κοντινότερα στον χρήστη σημεία, τότε υπήρχε ο κίνδυνος να υπερβεί το όριο των ημερησίων requests που θέτει η Google για accounts ελεύθερης χρήσης. Για το λόγο αυτό καταχωρήθηκαν στατικά τα σημεία ενδιαφέροντος γύρω από την πόλη του Ηρακλείου.

Το επόμενο πρόβλημα ήταν ο τρόπος υπολογισμού απόστασης από τον χρήστη στα σημεία ενδιαφέροντος ώστε να βρεθεί το κοντινότερο. Υπήρχαν δύο πιθανοί τρόποι υλοποίησης: Ο πρώτος και πιο ακριβής ήταν να γίνεται request στο DirectionsApi της Google το οποίο θα επέστρεφε την ακριβή διαδρομή του χρήστη για κάθε πιθανό σημείο κάτι που θα ανταποκρινόταν σχεδόν απόλυτα στην πραγματική διαδρομή που έπρεπε να εκτελέσει ο χρήστης. Εδώ όμως υπήρχε πάλι το πρόβλημα περιορισμού των ημερησίων requests που μπορούν να γίνουν στο DirectionsApi της Google για free developer account key το οποίο χρησιμοποιεί η εφαρμογή. Η δεύτερη μέθοδος ήταν να υπολογιστεί η απόσταση του χρήστη από όλα τα σημεία με κλασσική Ευκλείδεια Γεωμετρία, από τον τύπο της απόστασης δύο σημείων πάνω σε μια κυρτή επιφάνεια (που είναι η επιφάνεια της Γης) με γνωστές διαστάσεις. Αυτός ο τρόπος όμως θα έχανε σημαντικά σε ακρίβεια σε σχέση με την πραγματική διαδρομή με τις στροφές των δρόμων τις κατευθύνσεις κλπ.

Γι αυτό επιχειρήθηκε μια προσεγγιστική μελέτη απόστασης. Η μελέτη αυτή έγινε υπολογίζοντας την πραγματική απόσταση χρησιμοποιώντας ένα σύνολο από δείγματα όλων των σημείων ενδιαφέροντος από ένα σταθερό σημείο του χρήστη στο κέντρο του Ηρακλείου. Ο υπολογισμός της πραγματικής απόστασης έγινε με χρήση του DirectionsApi της Google το οποίο επέστρεψε για κάθε δείγμα την ακριβή διαδρομή το μήκος της οποίας υπολογίστηκε και αποθηκεύτηκε σε ένα πίνακα. Το δεύτερο μέρος της μελέτης ήταν ο υπολογισμός απόστασης από τα ίδια σημεία με χρήση Ευκλείδειας Γεωμετρίας υπολογίζοντας την απόσταση των σημείων στην κυρτή επιφάνεια της Γης. Ο υπολογισμός του δεύτερου τρόπου έγινε με χρήση της συνάρτησης distanceBetween της κλάσης Location που παρέχεται από το API της Google. Το τελευταίο κομμάτι ήταν η διαίρεση των δύο μετρήσεων μεταξύ τους ώστε να βγει μια τιμή απόκλισης της Ευκλείδειας σε σχέση με την πραγματική. Έπειτα βρέθηκε το ο μέσος όρος αυτών των αποκλίσεων ο οποίος χρησιμοποιήθηκε για προσέγγιση ακρίβειας των αποτελεσμάτων της μεθόδου εύρεσης απόστασης με Ευκλείδεια Γεωμετρία ο οποίος υλοποιήθηκε σαν τελικός τρόπος εύρεσης κοντινότερου σημείου ενδιαφέροντος. Παρακάτω παρατίθεται ο κώδικας εύρεσης κοντινότερου σημείου και υπολογισμού απόστασης, η μεταβλητή avgDivergence είναι η μέση απόκλιση που βρέθηκε από την προσεγγιστική μελέτη που προαναφέρθηκε. Εδώ πρέπει να σημειωθεί πως η εύρεση κοντινότερου σημείου γίνεται μόνο για ραντεβού τύπου supermarket ή gas station καθώς τα ραντεβού cinema έχουν ήδη προεπιλεγμένο μέρος που έχει επιλεγεί από το χρήστη για προβολή συγκεκριμένης ταινίας οπότε δεν μπορεί να αλλάξει.

```
protected Place findNearestOpenPlace(ArrayList<Place> placeList, LatLng
startPoint) throws ParseException {
    double dist = 0, tempDist;
    Place nearestPlace = null;
    boolean nullFlag = false;
    Date appointTime,appointEnd;
    appointTime = dateFormat.parse(receivedDateString);
    Calendar cal = Calendar.getInstance();
    cal.setTime(appointTime);
    cal.add(Calendar.SECOND,receivedDurationInSec);
    appointEnd = cal.getTime();
    for (int i = 0; i < placeList.size(); i++) {
        if (checkIfopen(placeList.get(i), appointTime,appointEnd)) {
            dist = distFrom(startPoint,
            new LatLng(placeList.get(i).getLat(),
            placeList.get(i).getLon()));
            nearestPlace = placeList.get(i);
            nullFlag = false;
            break;
        } else {
            nearestPlace = null;
            nullFlag = true;
        }
    }
    if (!nullFlag) {
        for (int i = 0; i < placeList.size(); i++) {
            if (checkIfopen(placeList.get(i), appointTime,appointEnd))
{
```

```
                tempDist = distFrom(startPoint,
        new LatLng(placeList.get(i).getLat(),
        placeList.get(i).getLon()));
                if (tempDist < dist) {
                    dist = tempDist;
                    nearestPlace = placeList.get(i);
                }
            }
        }
    }
    return nearestPlace;
}

public static double distFrom(LatLng latlang1, LatLng latlang2) {
    float[] results = new float[3];
    Location.distanceBetween(latlang1.latitude, latlang1.longitude,
    latlang2.latitude, latlang2.longitude, results);
    double dist;
    dist = (double) results[0];
    dist = dist * avgDivergence;
    return dist;
}
```

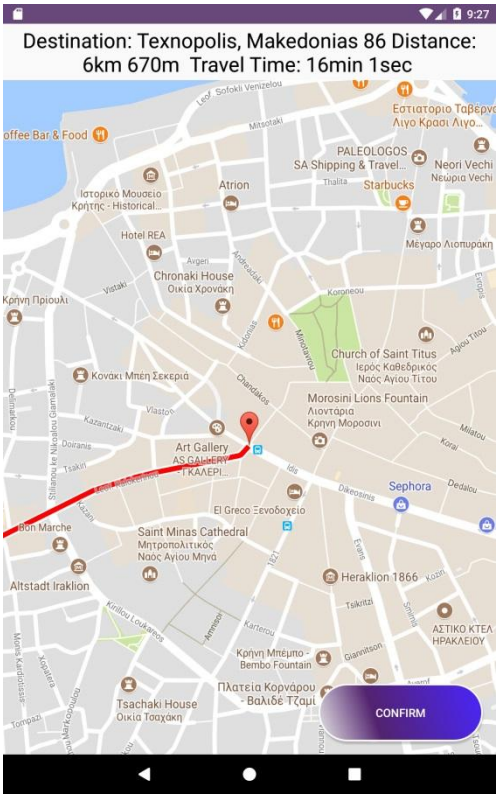
Τέλος αφού βρεθεί το κοντινότερο σημείο ενδιαφέροντος γίνεται ένα request στο DirectionsApi της Google με την αρχή και το τέλος της διαδρομής που πρέπει να σχεδιαστεί στο χάρτη, αν το request γίνει επιτυχώς η υπηρεσία επιστρέφει ένα πίνακα από σημεία τα οποία μετατρέπονται σε polylines και σχεδιάζονται στο χάρτη μαζί με δύο markers στην αρχή και στο τέλος της διαδρομής. Υπολογίζεται επίσης και η πραγματική απόσταση διαδρομής καθώς και ο χρόνος μετάβασης του χρήστη στοιχεία τα οποία εμφανίζονται στο TextView μαζί με τις πληροφορίες του προορισμού που επιλέχθηκε. Αν υπάρξει πρόβλημα επικοινωνίας με την υπηρεσία επαναλαμβάνεται το αίτημα μέχρι να γίνει επιτυχής επικοινωνία ή μέχρι ο αριθμός των αιτημάτων να ξεπεράσει τα 20 όπου εμφανίζεται μήνυμα στον χρήστη ότι η επικοινωνία με την υπηρεσία δεν είναι δυνατή αυτή τη στιγμή αλλά μπορεί να δοκιμάσει αργότερα.

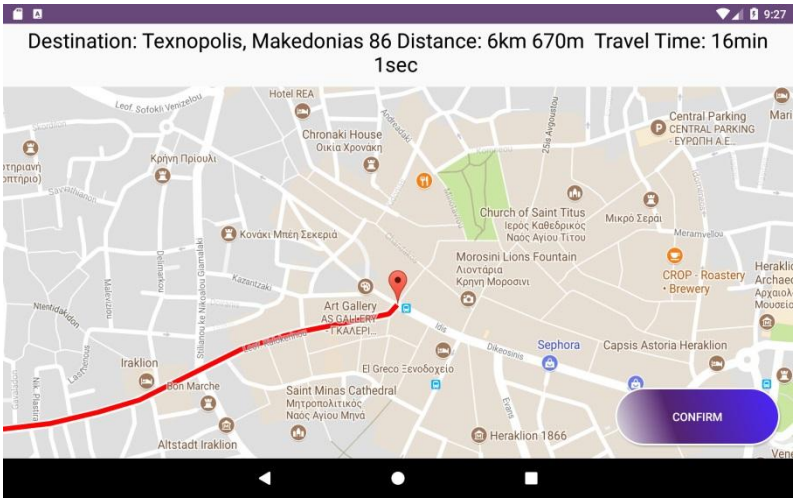
Το activity αποθηκεύει τον πίνακα με τα σημεία διαδρομής, θέσης χρήστη, προορισμού καθώς και τα περιεχόμενα του TextView με τη χρήση της savedInstanceState που περιγράφηκε στο κεφάλαιο 5.3.1 του activity Organizer. Έτσι η διαδρομή αναδημιουργείται στον χάρτη μαζί με όλα τα περιεχόμενα του TextView κάθε φορά που το layout αλλάζει προσανατολισμό κατά την περιστροφή της συσκευής χωρίς όμως να γίνονται επιπρόσθετα requests στην υπηρεσία.

Και σε τελικό στάδιο αφού έχει βρεθεί προορισμός, ο χρήστης μπορεί να πατήσει το κουμπί Confirm για επιβεβαίωση, έτσι το activity τερματίζει αποστέλλοντας τις παραμέτρους προορισμού σε μορφή ενός αντικειμένου Place σαν Extra σε Intent στο activity NewAppointment.

Παρακάτω φαίνεται ένα παράδειγμα χρήσης του PlanMap σε κάθετο και οριζόντιο προσανατολισμό.

PlanMap Layouts 1





5.4.2 InstantMap

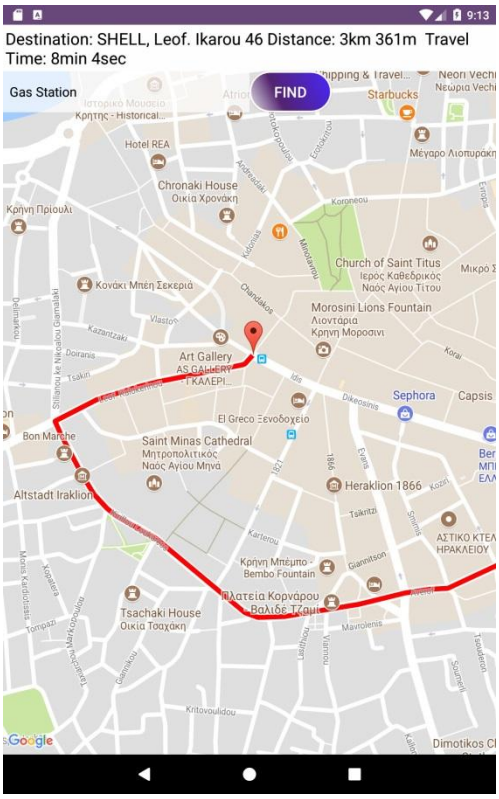
Ρόλος: Ρόλος του InstantMap είναι η εύρεση κοντινότερου σημείου ενδιαφέροντος και δρομολόγηση του χρήστη σε αυτό με τον χρήστη να επιλέγει τον τύπο σημείου ενδιαφέροντος και τη δρομολόγηση να γίνεται χωρίς να εκτελείται έλεγχος εγκυρότητας ραντεβού. Αυτό έχει ως σκοπό να δώσει στον χρήστη έναν τρόπο γρήγορης εύρεσης σημείου ενδιαφέροντος χωρίς να μπει στη διαδικασία να δημιουργήσει νέο ραντεβού.

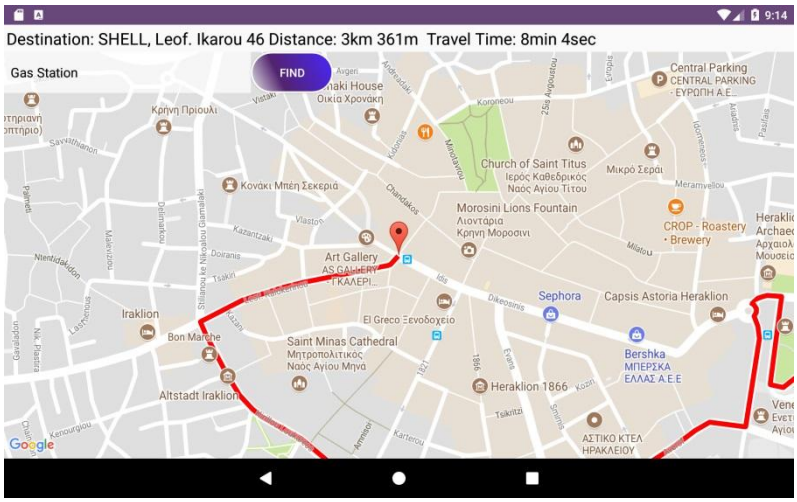
Εκτέλεση: Η υλοποίηση του InstantMap είναι ακριβώς η ίδια με αυτή του PlanMap με τη διαφορά ότι ο τύπος μέρους καθορίζεται από ένα spinner το οποίο επιλέγεται από το χρήστη. Η διαδικασία της δρομολόγησης γίνεται μετά από το πάτημα του κουμπιού Find σε αντίθεση με το PlanMap όπου γίνεται αυτόματα με την έναρξη του activity. Η βασική διαφορά στην υλοποίηση του InstantMap είναι ότι δεν υλοποιείται έλεγχος εγκυρότητας ραντεβού για τους λόγους που αναφέρθηκαν παραπάνω στο ρόλο του activity.

Οι υπόλοιποι έλεγχοι πραγματοποιούνται κανονικά και εδώ καθώς είναι απαραίτητοι για την ομαλή λειτουργία του χάρτη και της δρομολόγησης, οι έλεγχοι αυτοί όπως ο έλεγχος διαθεσιμότητας δικτύου, GPS και έλεγχος permissions αναφέρθηκαν στην υλοποίηση του PlanMap.

Υπάρχει και εδώ το TextView στο οποίο αναγράφεται η διεύθυνση, η απόσταση και ο χρόνος μετάβασης για τον προορισμό που βρέθηκε. Παρακάτω παρατίθενται παραδείγματα χρήσης του InstantMap Activity σε κάθετο και οριζόντιο προσανατολισμό.

InstantMap Layouts 1





6. Αποτελέσματα

6.1 Συμπεράσματα

Η διεξαγωγή της εργασίας ήταν μια διαδικασία κατά την οποία αποκομίστηκαν βαθύτερες γνώσεις στο αντικείμενο της ανάπτυξης λογισμικού σε περιβάλλον περιορισμένης υπολογιστικής ισχύς όπως είναι οι κινητές συσκευές στις οποίες δεσπόζει η πλατφόρμα Android. Επιπλέον η μελέτη και η σύγκριση όλων των διαθέσιμων τεχνολογιών που χρησιμοποιούνται σε κινητές συσκευές έδωσε τη δυνατότητα εξέτασης αυτών κάτω από ένα διαφορετικό πρίσμα που θα μπορούσε να θεωρηθεί πιο αντικειμενικό σε σχέση με μια μελέτη που μελετά αποκλειστικά την τεχνολογία την οποία χρησιμοποιεί.

Σαν συμπεράσματα αποκομίζεται πως όλες οι τεχνολογίες έχουν τα πλεονεκτήματα και τα μειονεκτήματά τους, οι δεσπόζουσες τεχνολογίες αυτής της εποχής όσον αφορά τα λειτουργικά συστήματα στις κινητές συσκευές φαίνεται να είναι το Android της Google και το iOS της Apple κάτι που δε σημαίνει απαραίτητα ότι είναι οι ιδανικές τεχνολογίες για όλες τις συσκευές, το Android καλύπτει μεγαλύτερη γκάμα συσκευών όσον αφορά την προσαρμοστικότητα στο υλικό και προσφέρει πολύ μεγαλύτερη ελευθερία παραμετροποίησης και στον developer αλλά και στον χρήστη, ενώ το iOS όντας κλειστό λειτουργικό σύστημα απευθύνεται σε πιο περιορισμένο εύρος όσον αφορά το υλικό και σαφώς οι ανάγκες που καλύπτει τείνουν πιο πολύ στη σταθερότητα του λειτουργικού με κόστος όμως την αφαίρεση σε μεγάλο μέρος της ελευθερίας παραμετροποίησης και προσαρμοστικότητας του λογισμικού για κάλυψη πιο εξειδικευμένων και εξατομικευμένων αναγκών, κάτι που προσφέρει σε μεγάλο βαθμό η πλατφόρμα Android. Εν τέλει όμως η προοριζόμενη χρήση του κάθε καταναλωτή είναι αυτή που θα καθορίσει τις ανάγκες του για να επιλέξει ανάμεσα σε αυτές τις δύο ή και σε διαφορετικές τεχνολογίες.

Όσον αφορά την ίδια την πλατφόρμα Android, μετά από εκτεταμένη χρήση και εξοικείωση με πολλές από τις λειτουργίες και δυνατότητες που προσφέρει, συμπεραίνεται πως το Android αποτελεί ένα ολοκληρωμένο σύστημα λογισμικού με δικά του αυτοδιαχειριζόμενα components που είναι ταυτόχρονα προσβάσιμα μέχρι και τα κατώτερα επίπεδα αρχιτεκτονικής, πράγμα που το καθιστά ιδιαιτέρως παραμετροποιήσιμο, επεκτάσιμο και ευπροσάρμοστο στις ανάγκες του developer ο οποίος έχοντας αυτού του είδους την ελευθερία στα χέρια του, δύναται να αναπτύξει λογισμικό το οποίο θα είναι πολύ πιο κοντά στους αρχικούς στόχους και θα ανταποκρίνεται πολύ καλύτερα στις απαιτήσεις που έχουν τεθεί. Η φύση της αρχιτεκτονικής του Android δίνει τη δυνατότητα αυτοδιαχείρισης σε components που δεν χρειάζεται να τροποποιηθούν κάτι που μειώνει σε σημαντικό βαθμό την πολυπλοκότητα της διαδικασίας υλοποίησης και σχεδιασμού, ενώ παράλληλα δίνεται σχεδόν απόλυτη ελευθερία στην προσαρμογή και τροποποίηση των component που χρειάζεται συγκεκριμένα ο developer, έτσι δίνεται το περιθώριο επεκτασιμότητας και βελτιστοποίησης της εφαρμογής σε πολύ μεγάλο βαθμό, κάτι που θα ήταν αρκετά πιο δύσκολο σε άλλα πιο στατικά μοντέλα αρχιτεκτονικής που προσφέρονται από διαφορετικές τεχνολογίες στον ίδιο τομέα της ανάπτυξης λογισμικού κινητών συσκευών.

Η εφαρμογή η ίδια αποτελεί ένα σύστημα δρομολόγησης και διαχείρισης ραντεβού, παραμετροποιημένη έτσι ώστε να δίνει ευκολία στη χρήση, εύκολη πρόσβαση στα πιο συχνά σενάρια χρήσης ενώ παράλληλα προσφέρει αξιοπιστία ως προς την διατήρηση και προβολή των πληροφοριών που αιτείται ο χρήστης σε κάθε περίπτωση μέσα στα πλαίσια των δυνατοτήτων που αναγράφονται για την παρούσα εφαρμογή.

6.2 Προτάσεις για Μελλοντικές Επεκτάσεις

- **Δυναμικό Μοντέλο:** Σε ένα πλαίσιο που υπάρχει κάποια μηνιαία χρηματοδότηση για την περαιτέρω ανάπτυξη της εφαρμογής θα ήταν εφικτή η μετατροπή ανάκτησης δεδομένων από το στατικό μοντέλο το οποίο έχει σε κάτι πιο δυναμικό. Αυτό σημαίνει πως είναι δυνατό οι πληροφορίες που χρειάζονται σχετικά με τα σημεία ενδιαφέροντος να ανακτούνται δυναμικά μέσω του PlacesApi της Google,κάτι ανάλογο θα μπορούσε να γίνει και για την ενημέρωση της λίστας διαθέσιμων ταινιών και ωραρίων προβολής. Για να αποτραπεί όμως το ενδεχόμενο αδυναμίας σύνδεσης στην υπηρεσία λόγω υπέρβασης του ορίου των ημερησίων requests θα πρέπει να αναβαθμιστεί το developer account εφαρμόζοντας μια μηνιαία χρηματική συνδρομή στη Google για αύξηση του ορίου ημερησίων requests. Με το συγκεκριμένο μοντέλο θα γίνει δυνατή και η υλοποίηση του πρώτου τρόπου εύρεσης κοντινότερης απόστασης μέσω requests στο DirectionsApi έτσι θα έχουμε πολύ μεγαλύτερη ακρίβεια στον υπολογισμό αποστάσεων κάτι που θα βελτιώσει σε μεγάλο βαθμό την αξιοπιστία των αποτελεσμάτων της εφαρμογής. Επιπλέον με το μοντέλο χρήσης του PlacesApi θα υπάρξει η δυνατότητα διεύρυνσης του είδους των σημείων ενδιαφέροντος σε όλους τους διαθέσιμους τύπους που παρέχει η υπηρεσία χωρίς να υπάρχει ο περιορισμός μόνο τριών στατικών λιστών όπως γίνεται στην παρούσα εφαρμογή, επίσης η χρήση αυτ'ης της υπηρεσίας θα επιτρέψει στην εφαρμογή να ανταποκρίνεται σχεδόν σε οποιοδήποτε μέρος του κόσμου (το οποίο καλύπτεται από την υπηρεσία) σε αντίθεση με την τρέχουσα εφαρμογή που καλύπτει αποκλειστικά την πόλη του Ηρακλείου.

- **Υλοποίηση των υπηρεσιών Local Guides και Google Maps Contribution:** Σαν extra feature θα μπορούσε να προστεθεί η δυνατότητα σύνδεσης με τις υπηρεσίες Local Guides και Google Maps Contribution ώστε να δοθεί η δυνατότητα στον χρήστη να παρέχει χρήσιμο feedback το οποίο μπορεί να χρησιμοποιηθεί μετά και από άλλους χρήστες αλλά και από developers σαν πρόσθετη πηγή πληροφοριών σχετικά με τις εκάστοτε τοποθεσίες.

- **Σύνδεση Λογαριασμών Χρήστη:** Τέλος θα μπορούσε να προστεθεί η δυνατότητα σύνδεσης με λογαριασμούς Google ή κοινωνικών δικτύων μέσω των οποίων θα μπορούσαν να υλοποιηθούν ποικίλες λειτουργίες κοινωνικής δικτύωσης. Επιπλέον μια τέτοια λειτουργία θα προσέφερε χρήσιμο feedback για τις προτιμήσεις και τις ανάγκες του χρήστη στοιχεία τα οποία θα μπορούσαν να χρησιμοποιηθούν για την εξατομίκευση των λειτουργιών και του σχεδιασμού της εφαρμογές στις ανάγκες του κάθε χρήστη ξεχωριστά με σκοπό τη βελτίωση της εμπειρίας χρήσης.

7. Βιβλιογραφία

[1] Abdullah Humayun, M.Y, Dang, T.T.P., Himawan, A.G., Koirala, Y., Ridwan, R.M. & Wibiyanto, D. (2009) A Future Trajectory of Google Android: A study from operating system, application stores and handset manyfacturers. International University of Japan

[2] Bellis, M. (2017) The History of Sonar. History and Culture
Διαθέσιμο στο: https://www.thoughtco.com/the-history-of-sonar-1992436

[3] Brunner, W. (2005) Longitude by the Method of Lunar Distance. Starpath School of Navigation

[4] Fling, B. (2009) Mobile Design and Development. California: O'Reilly Media Inc
Grundstrom, P. (2010) Mobile Development for iPhone and Android: A comparison of 3rd party development for the iPhone and Android platforms. Royal Institute of Technology School of Computer Science and Communication

[5] Kalafatis K., Kalouta K. (2013) Εμπλουτισμός Διεπαφών Ανεύρεσης Δεδομένων Σε Κοινωτικές Υπηρεσίες Δικτύωσης.
Σχολή Τεχνολογικών Εφαρμογών Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων
Διαθέσιμο στο: http://nefeli.lib.teicrete.gr/browse/stef/epp/2013/

[6] Mai, T. (2012) Global Positioning System History. National Aeronautics and Space Administration
Διαθέσιμο στο: https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html

[7] Morrison, J. E. (2007) The Astoralabe. Classical Science Press

[8] Scott, P., Frost, G.P., Lachow, I., Frelinger, D.F., Fossum, D., Pnto, D.W., Pinto, M.M. (1995) The Global Positioning System: Assessing National Policies. Santa Monica, CA: RAND Corporation.
Διαθέσιμο στο: https://www.rand.org/pubs/monograph_reports/MR614.html

[9] Skolnik, M.I. (2016) Radar. Encyclopædia Britannica
Διαθέσιμο στο: https://www.britannica.com/technology/radar/History-of-radar

[10] Sobel, D. (1998) A Brief History of Early Navigation. Johns Hopkins APL technical Digest, 19(1), pp. 11-13

[11] Sullivan, M. (2012) A brief history of GPS. PC World (IDG Communications)
Διαθέσιμο στο: https://www.pcworld.com/article/2000276/a-brief-history-of-gps.html

[12] Varchalamas P. (2013) Development of a Natural User Interface (NUI) and creation of a Context Information Manager, using Microsoft Kinect.
Σχολή Τεχνολογικών Εφαρμογών Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων
Διαθέσιμο στο: http://nefeli.lib.teicrete.gr/browse/stef/epp/2013/

[13] Vardalas, J. (2013) A History of the Magnetic Compass. Tech history
Διαθέσιμο στο: http://theinstitute.ieee.org/tech-history/technology-history/a-history-of-the-magnetic-compass