ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΣΕ ΚΙΝΗΤΟ ΤΕΡΜΑΤΙΚΟ
ΜΕ ΧΡΗΣΗ *WEB SERVICES*

ΕΥΑΓΓΕΛΟΥ ΔΗΜΗΤΡΗΣ
(ΜΕ 10081)

ΠΕΙΡΑΙΑΣ 2013

ΠΕΡΙΕΧΌΜΕΝΑ

1 ΕΙΣΑΓΩΓΗ

Η παρούσα έκθεση περιγράφει τη διαδικασία ανάλυσης, σχεδιασμού και υλοποίησης εφαρμογής κινητού τερματικού στα πλαίσια της εκπόνησης πτυχιακής εργασίας του μεταπτυχιακού των *Δικτυοκεντρικών Συστημάτων* του Πανεπιστημίου Πειραιώς. Ο κύκλος ανάπτυξης λογισμικού αποτελεί ιδιαίτερα απαιτητική διαδικασία στην οποία οι εμπλεκόμενοι θα πρέπει να κατανοούν σε βάθος το πρόβλημα που καλούνται να επιλύσουν, να γνωρίζουν τις διαθέσιμες τεχνολογικές επιλογές, τη μεθοδολογία και αρχιτεκτονική του συστήματός τους καθώς επίσης και να μεριμνούν για την απόκτηση της απαιτούμενης τεχνογνωσίας για την υλοποίησή του.

Ομοίως, η ανάπτυξη εφαρμογής σε κινητό τερματικό εμπεριέχει τις δικές της προκλήσεις προκειμένου το τελικό προϊόν να είναι επιτυχημένο σε εμπορικό επίπεδο και παράλληλα να πληροί τις απαιτούμενες προδιαγραφές ποιότητας σε τεχνικό επίπεδο. Στο πρώτο σκέλος συμβάλλει σημαντικά η σύλληψη της αρχικής ιδέας, που απορρέει από την ανάλυση των αναγκών της αγοράς καθώς και τη δημιουργικότητα των δημιουργών του, που πολλές φορές αποδεικνύεται ικανή να διαμορφώνει εκ νέου το χάρτη των αναγκών των καταναλωτών, αποδίδοντας την αξία που επιζητούν και αποφέροντας παράλληλα κερδοφορία στους παρόχους του προϊόντος. Ωστόσο η πρωτοτυπία της ιδέας δε μπορεί από μόνη της να εγγυηθεί όλα τα παραπάνω καθώς η σταθερότητα, η αξιοπιστία του συστήματος και κατ' επέκταση η ικανοποίηση του πελάτη, εξαρτώνται άμεσα από τον τρόπο σχεδιασμού και υλοποίησης της. Τα εργαλεία και τα *API*'s των φορητών συσκευών δεν έχουν (προς το παρόν) το ίδιο εύρος και δυνατότητες με τα αντίστοιχα των Desktops, ενώ οι συσκευές καθ' εαυτές έχουν περιορισμένους πόρους επεξεργασίας και αποθήκευσης. Ως εκ τούτου ο μηχανικός της εφαρμογής θα πρέπει να διαμορφώνει το σύστημα του βάσει των παραπάνω περιορισμών και να βρίσκει έξυπνες λύσεις ώστε η εφαρμογή να λειτουργεί ομαλά, να καταλαμβάνει λίγους πόρους, ώστε ο τελικός χρήστης να μένει ικανοποιημένος από τη χρήση της.

Στην έκθεση θα αναφερθούν οι παράγοντες εκείνοι που συνετέλεσαν στη σύλληψη της αρχικής ιδέας, οι τεχνολογίες που χρησιμοποιήθηκαν, ορισμένα στοιχεία αρχιτεκτονικής, οι απαιτήσεις της *Microsoft* και οι έλεγχοι που διενήργησε προκειμένου να την πιστοποιήσει και να την ανεβάσει στο *marketplace*, ενώ θα παρουσιασθεί και η ίδια η εφαρμογή στα πλαίσια εκτέλεσης ενός σεναρίου χρήσης.

1.1 ΣΚΟΠΟΣ

Οι "έξυπνες συσκευές" (Smartphones) αδιαμφισβήτητα έχουν διεισδύσει στη σύγχρονη καθημερινότητα αποτελώντας αναγκαία εργαλεία υποστήριξης του χρήστη για τη διεκπεραίωση ποικίλων εργασιών. Αυτό οφείλεται στο γεγονός ότι συγκεντρώνουν ένα σημαντικό μέρος της λειτουργικότητας των ηλεκτρονικών υπολογιστών, σε ορισμένους τομείς την επεκτείνουν χάρη στους ενσωματωμένους αισθητήρες τους (γυροσκόπιο, *GPS*, ταχύμετρο κ.α.) ενώ παράλληλα διατηρούν το μικρό τους μέγεθος που επιτρέπει την εύκολη μεταφορά τους σε περιβάλλοντα που θα ήταν ανέφικτη η χρήση ενός Laptop. Μπορούν έτσι να εκτελέσουν λειτουργίες που θα αδυνατούσε ένας ηλεκτρονικός υπολογιστής ανοίγοντας έτσι ένα νέο πεδίο διερεύνησης αναγκών των καταναλωτών και ανάλογων λύσεων βασισμένων στην τεχνολογία.

Βάσει του σκεπτικού αυτού, η εν λόγω εργασία προτείνει την αξιοποίηση των δυνατοτήτων του smartphone για τη δημιουργία εφαρμογής για τη μέτρηση βασικών γεωμετρικών μεγεθών ενός κομματιού γης αυθαίρετου σχήματος και μεγέθους, με τρόπο απλό και ευκολονόητο, ώστε ο χρήστης να μη χρειάζεται να καταβάλλει ιδιαίτερη προσπάθεια για να εμπεδώσει τις λειτουργίες της και να τη χρησιμοποιήσει.

Η λειτουργικότητα αυτή, τον καιρό που ξεκίνησε η υλοποίηση (αρχές 2012) δεν υπήρχε στα Laptops διότι δε διέθεταν αισθητήρα *GPS*, αλλά, έστω ότι υπήρχε η ανάλογη δυνατότητα, θα παρέμεναν μη χρηστικά καθώς η μεταφορά τους σε υπαίθρια δύσβατα σημεία για τη λήψη μέτρησης θα δυσκόλευε σημαντικά το χρήστη. Αντίθετα, τα smartphones μπορούν να μεταφέρονται και τοποθετούνται ευκολότερα στα κατάλληλα σημεία για τη λήψη μέτρησης *GPS*.

Πέρα της βασικής τους λειτουργικότητας, τα smartphones παρέχουν τη δυνατότητα (μέσω των *API*'s) διαμόρφωσης ιδιαίτερα φιλικών προς το χρήστη *interface*s. Το τελευταίο αποτέλεσε βασική προτεραιότητα κατά την ανάπτυξη της εφαρμογής, καθώς καταβλήθηκε έντονη προσπάθεια για τη διαμόρφωση των οθονών που δε θα προβάλλουν μονάχα στατική πληροφορία, αλλά θα καθιστούν την εφαρμογή διαδραστική μέσω εναλλακτικών τρόπων αλληλεπίδρασης με το χρήστη.

Οι ανάγκες που καλύπτει η εφαρμογή και οι δυνατότητες αξιοποίησής της, περιγράφονται στην επόμενη παράγραφο.

1.2 ΠΡΟΒΛΗΜΑ ΠΡΟΣ ΕΠΙΛΥΣΗ - ΔΥΝΑΤΕΣ ΧΡΗΣΕΙΣ

Η μέτρηση της έκτασης περιοχών γης δεδομένης τοποθεσίας και με γνωστά όρια, αποτελεί αναπόσπαστο βήμα του κύκλου εργασιών που εκτελούν πολλοί διαφορετικοί επαγγελματικοί κλάδοι στα πλαίσια μεγάλου εύρους επιχειρησιακών δραστηριοτήτων. Ωστόσο τα διαθέσιμα εργαλεία και τεχνικές δεν είναι πάντοτε διαθέσιμα, ή/και προϋποθέτουν ο χρήστης να κατέχει υψηλή κατάρτιση στο χειρισμό τους και την επεξεργασία των δεδομένων από τις μετρήσεις. Ορισμένες τεχνικές με τα απαιτούμενα εργαλεία τους συνοψίζονται στον παρακάτω πίνακα:

ΜΕΘΟΔΟΣ	**ΑΠΑΙΤΟΥΜΕΝΑ ΕΡΓΑΛΕΙΑ/ΔΕΔΟΜΕΝΑ**	**ΤΕΧΝΙΚΕΣ ΓΝΩΣΕΙΣ**
Μηχανική Μέθοδος	1. Τοπογραφικό Σχέδιο 2. Εμβαδόμετρο	1. Γνώση χρήσης του εμβαδόμετρου
Γραφική Μέθοδος	1. Τοπογραφικό Σχέδιο	1. Στοιχειώδεις γνώσεις γεωμετρίας, χρονοβόρα σε πολύπλοκα σχήματα
Αναλυτική Μέθοδος	1. Τοπογραφικό Σχέδιο	1. Προχωρημένες γνώσεις γεωμετρίας
Μέτρηση με χρήση *Google Maps*	-	I. Γνώσεις χειρισμού Google Maps

Πίνακας 1: Μέθοδοι υπολογισμού εμβαδού, και άλλων γεωμετρικών μεγεθών τεμαχίων γής

Οι τρείς πρώτες τεχνικές προϋποθέτουν την ύπαρξη τοπογραφικού Σχεδίου, τη οποία διενεργούν εξειδικευμένοι επαγγελματίες, και η οποία έχει υψηλό κόστος. Η μέτρηση μέσω των εργαλείων της Google, παρ' ότι είναι εύκολη για ένα μέτριο χειριστή Η/Υ, απαιτεί τον ορισμό των σημείων με υψηλή ακρίβεια το οποίο δεν είναι ιδιαίτερα εύκολο[1].

Αντιθέτως, ο προσδιορισμός των σημείων μέσω μετρήσεων πεδίου που προτείνει η εφαρμογή, αυξάνει την ακρίβεια του προσδιορισμού των ορίων της περιοχής, ενώ βοηθά το χρήστη να λάβει υπ' όψιν τις ιδιομορφίες της που ενδεχομένως να αγνοούσε με τις μεθόδους εκτός πεδίου.

[1] Απαιτεί τη τοποθέτηση του δείκτη στο ακριβές σημείο του χάρτη ο οποίος εμπεριέχει συστημικό σφάλμα ως προς την αντιστοίχηση με τις πραγματικές γεωγραφικές συντεταγμένες.

ΣΕΝΑΡΙΑ ΑΞΙΟΠΟΙΗΣΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Υπάρχουν ποικίλα σενάρια στα οποία θα μπορούσε να χρησιμοποιηθεί η εφαρμογή που αναπτύξαμε, καθώς και χρήστες από διαφορετικούς επαγγελματικούς κλάδους:

- **Αγροτικές καλλιέργειες**: Σε ένα αγροτικό τεμάχιο ο γεωργός επιθυμεί να γνωρίζει τη συνολική ποσότητα σπόρων σταριού θα πρέπει να προμηθευτεί προκειμένου να σπείρει ένα συγκεκριμένο κομμάτι γης . Με την εφαρμογή, διενεργεί μετρήσεις και υπολογίζει το συνολικό εμβαδό, ενώ γνωρίζοντας την απαιτούμενη ποσότητα ανά τετραγωνικό μέτρο, μπορεί να υπολογίσει και τη συνολική ποσότητα προμήθειας σταριού.
- **Εκπόνηση μελέτης για φωτοβολταϊκά**: Η μελέτη για τη συνολική παραγωγή ηλεκτρικής ενέργειας από φωτοβολταϊκές εγκαταστάσεις, προαπαιτεί τη γνώση του εμβαδού της περιοχής εγκατάστασης των ηλιακών panels. Ο μηχανικός που διενεργεί τη μελέτη μπορεί με χρήση της εφαρμογής να υπολογίσει το συνολικό εμβαδό της περιοχής είτε υποτμημάτων αυτής (δεδομένου ότι τέτοιες εγκαταστάσεις δεν καλύπτουν πάντοτε το σύνολο της διαθέσιμης περιοχής). Βάσει αυτού, μπορεί να προβεί στον υπολογισμό των κιλοβατωρών.
- Άλλες εφαρμογές: Μετρήσεις για την οριοθέτηση ενός campus, αθλητικών εγκαταστάσεων, μέτρηση μεγαλύτερων εκτάσεων (στο μέγεθος πόλης), αρχιτεκτονικές μετρήσεις, άλλες μετρήσεις τοπογραφικού ενδιαφέροντος.

1.3 ΔΟΜΗ ΕΚΘΕΣΗΣ

Θα αναφερθούν όλες οι θεματικές ενότητες στις οποίες εμβαθύναμε στα διάφορα στάδια υλοποίησης της εφαρμογής. Αυτές ποικίλουν σε είδος καθώς ερευνήθηκαν και αντιμετωπίσθηκαν ποικίλων τύπων θέματα , από επιστημονικά, τεχνικά έως και επιχειρηματικά.

Στο *1ο Κεφάλαιο* διατυπώνεται το πρόβλημα προς επίλυση, πως η εφαρμογή το αντιμετωπίζει και σε ποιές περιπτώσεις θα μπορούσε η τελευταία να αξιοποιηθεί. Στο *2ο Κεφάλαιο* αναφέρονται οι αρχές λειτουργίας της εφαρμογής, που απορρέουν άμεσα από το μαθηματικό φορμαλισμό που υλοποιεί, αλλά και τα χαρακτηριστικά και περιορισμούς του συστήματος *GPS* για τον προσδιορισμό θέσης. Στο *3ο Κεφάλαιο* αναφέρονται οι τεχνολογίες που αξιοποιήθηκαν, με αναφορές στα *Web services*, το περιβάλλον ανάπτυξης, και το *Silverlight application framework* για το χτίσιμό της. Στο *4ο Κεφάλαιο* παρουσιάζεται η γενική αρχιτεκτονική της εφαρμογής, η απεικόνιση της διαδικασίας χρήσης της, το μοντέλο οντοτήτων και ο τρόπος αλληλεπίδρασής τους σε ορισμένες αντιπροσωπευτικές σελίδες. Στο *5ο Κεφάλαιο* παρουσιάζονται οι τεχνικές προδιαγραφές αλλά και οι προδιαγραφές εταιρικής πολιτικής, με τις οποίες η εφαρμογή πιστοποιήθηκε από τη Microsoft για διάθεση μέσω του marketplace. Στο *6ο Κεφάλαιο* παρουσιάζεται η εφαρμογή, όπως τη βλέπει ο χρήστης, κατά την εκτέλεση ενός σεναρίου χρήσης. Τέλος, το *7ο Κεφάλαιο* αποτελεί Παράρτημα το οποίο περιέχει κώδικα από κλάσεις και σελίδες της εφαρμογής (*XAML*, *code-behind*), προκειμένου να επιδειχθούν οι χρησιμοποιούμενες προγραμματιστικές τεχνικές και χαρακτηριστικά-ευκολίες που παρέχει το *Silverlight framework*.

2 ΑΡΧΕΣ ΛΕΙΤΟΥΡΓΙΑΣ

Η εργασία αυτή έχει ως αντικείμενο τη δημιουργία εφαρμογής για smartphone, η οποία υπολογίζει βασικά γεωμετρικά μεγέθη τεμαχίων γης, ανεξαρτήτου σχήματος και μεγέθους, τα οποία οριοθετεί ο χρήστης λαμβάνοντας στίγμα *GPS* από συνοριακά τους σημεία. Με αυτό τον τρόπο μπορεί ο χρήστης να μετρήσει εκτάσεις γης με απλό τρόπο, παρακάμπτοντας τη χρήση εξειδικευμένου τοπογραφικού εξοπλισμού που θα απαιτούσε εμπειρία χρήσης, καθώς και εκτέλεση υπολογιστικών μεθόδων με τις οποίες το πιθανότερο είναι να μην είναι εξοικειωμένος.

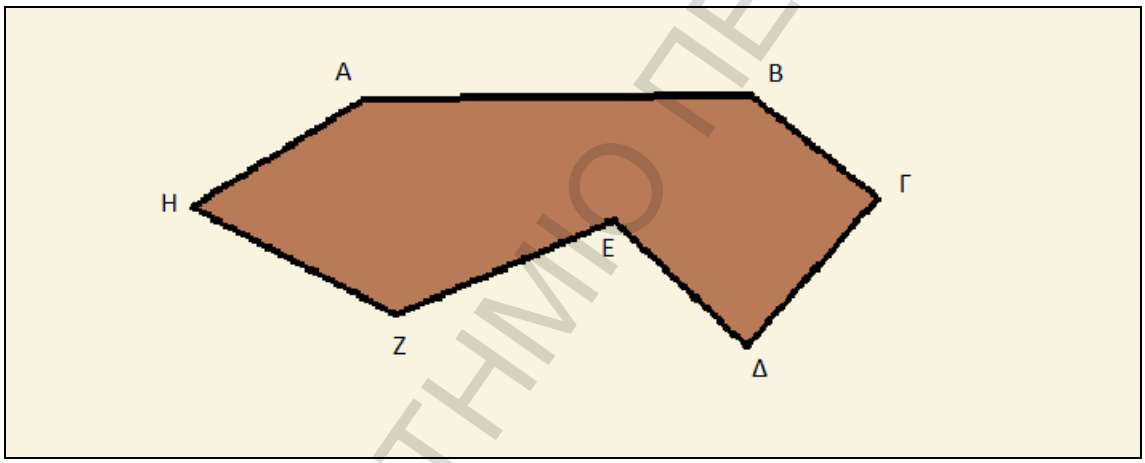
Η εφαρμογή συνδυάζει γνώσεις και μεθόδους που άπτονται διαφορετικών πεδίων επιστήμης και τεχνολογίας. Τόσο η βασική λειτουργικότητα (που περιγράφηκε στο προηγούμενο Κεφάλαιο), όσο και η πρόσθετες λειτουργίες (που θα αναφερθούν παρακάτω), αξιοποιούν διαφορετικά μαθηματικά εργαλεία και τεχνολογίες, οι οποίες από μόνες τους αποτελούν αντικείμενο ανάλυσης. Στην έκθεση αυτή θα επικεντρωθούμε στα πεδία αυτά που ως διαθέσιμα "εργαλεία" κατέστησαν δυνατή την υλοποίηση της συγκεκριμένης εφαρμογής.

Φυσικά, κάθε μέθοδος ή και εργαλείο συνοδεύονται από περιορισμούς όσον αφορά το εύρος και την ακρίβειά τους. Για παράδειγμα η ποιότητα των μετρήσεων του στίγματος του *GPS* εξαρτάται από τις συνθήκες που επικρατούν κατά τη διάρκεια της διενέργειας της μέτρησης . Ενδεικτικά, η ακρίβεια της συσκευής, είτε η σχετική θέση του χρήστη ως προς άλλα σώματα, η στιγμιαία θέση των δορυφόρων του παρόχου της υπηρεσίας, είναι ορισμένοι παράγοντες που επηρεάζουν την ακρίβεια των μετρήσεων. Δεδομένου ότι δεν μπορούν να εξαλειφθούν οι παράγοντες αυτοί, υιοθετούνται αντίστοιχες μαθηματικές μέθοδοι και πρακτικές που αντισταθμίζουν την απόκλιση από τις πραγματικές τιμές. Στις επόμενες παραγράφους θα αναφερθούν αυτά τα στοιχεία.

Στην αμέσως επόμενη παράγραφο, περιγράφονται τα βήματα της διαδικασίας υπολογισμού των γεωμετρικών μεγεθών μιας έκτασης γης μέσω της εφαρμογής καθώς και οι συμπληρωματικές λειτουργίες που αυτή επιτελεί προκειμένου να μπορεί ο χρήστης να διακινεί και αποθηκεύει εύκολα τις εξαγόμενες πληροφορίες. Οι λεπτομέρειες της υλοποίησης (αρχιτεκτονική, χρησιμοποιούμενες τεχνολογίες) θα αναφερθούν σε επόμενα κεφάλαια.

2.1 ΔΙΑΔΙΚΑΣΙΑ ΜΕΤΡΗΣΗΣ ΜΕΣΩ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Στο σχήμα απεικονίζεται μια τυχαία περιοχή γης, της οποίας το εμβαδό και άλλα μεγέθη επιθυμούμε να μετρήσουμε. Στο παράδειγμά μας έχουμε ένα 7-γωνο, απλό, μη κανονικό πολύγωνο (με άνισες πλευρές και γωνίες). Εξ' αιτίας της ιδιομορφίας του, η χαρτογράφηση αυτού του τεμαχίου είναι ιδιαίτερα δύσκολη, όπως και η μετέπειτα μαθηματική επεξεργασία για την εξαγωγή των επιθυμητών μεγεθών. Χρησιμοποιώντας την εφαρμογή, ο χρήστης αρκεί να λάβει το στίγμα *GPS* από τις 7 κορυφές (Α έως Η). Συγκεκριμένα, πρέπει να διατρέξει μια-μια τις κορυφές περιμετρικά, και τηρώντας τη διαδοχή τους, να διενεργήσει από μια μέτρηση στην κάθε μια επαναλαμβάνοντας το μέχρις ότου να φτάσει στην αρχική (δίχως να τη μετρήσει ξανά). Για παράδειγμα αν ξεκίνησε από την κορυφή Η, και κινείται ωρολογιακά, θα πρέπει να διαγράψει την πορεία Η→Α→Β→Γ→Δ→Ε→Ζ.



Εικόνα 1: Αυθαίρετο σχήμα τεμαχίου γης που μπορεί να μετρήσει η εφαρμογή. Απαιτείται μονάχα να καταγράψει το στίγμα *GPS* στα σημεία Α έως Η, τηρώντας την περιμετρική διάταξη των σημείων.

Στη συνέχεια, η εφαρμογή με χρήση ειδικών υπολογιστικών μεθόδων, υπολογίζει το συνολικό εμβαδό του πολυγώνου, των μήκη των ακμών του (ευθύγραμμα τμήματα ΑΒ, ΒΓ, ΓΔ, ΔΕ, ΕΖ, ΖΗ, ΗΑ) και την περίμετρο του. Οι εξαγόμενες πληροφορίες προβάλλονται στην οθόνη του, και απεικονίζεται το καταγεγραμμένο πολύγωνο πάνω σε ψηφιακό χάρτη.

Κατ' αυτό τον τρόπο ο χρήστης επιτυγχάνει τον υπολογισμό των βασικών γεωμετρικών μεγεθών της περιοχής του, και παράλληλα την χαρτογράφησή της, δίχως να καταβάλλει μεγάλη προσπάθεια.

ΠΡΟΣΘΕΤΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΕΦΑΡΜΟΓΗΣ

Ο υπολογισμός του εμβαδού και των μηκών μιας περιοχής αποτελεί τη βασική λειτουργικότητα της εφαρμογής. Ωστόσο οι δυνατότητες που παρέχονται από την πλατφόρμα ανάπτυξης, επιτρέπουν τη βελτίωση του τρόπου απεικόνισης και διαχείρισης των εξαγόμενων-υπολογιζόμενων πληροφοριών προς διευκόλυνση του χρήστη.

Οι πρόσθετες λειτουργίες που ενσωματώνει η εφαρμογή είναι:

- Επεξεργασία μετρήσεων κατά το στάδιο λήψης
- Προεπισκόπηση σε χάρτη της έως τώρα καταγεγραμμένης περιοχής με παράλληλη προβολή των αστικών συντεταγμένων του τρέχοντος σημείου του χρήστη
- Μετατροπή των υπολογισμένων δεδομένων σε εναλλακτικές μονάδες εμβαδού και μήκους
- Σώσιμο των πληροφοριών που εξήχθησαν κατά την ολοκλήρωση της διαδικασίας στη μνήμη της συσκευής, συνοδευόμενες από σημειώσεις υπό μορφή κειμένου που εισάγει ο χρήστης
- Αποστολή των παραπάνω πληροφοριών και μετα-πληροφοριών, σε λογαριασμό email της επιλογής του χρήστη

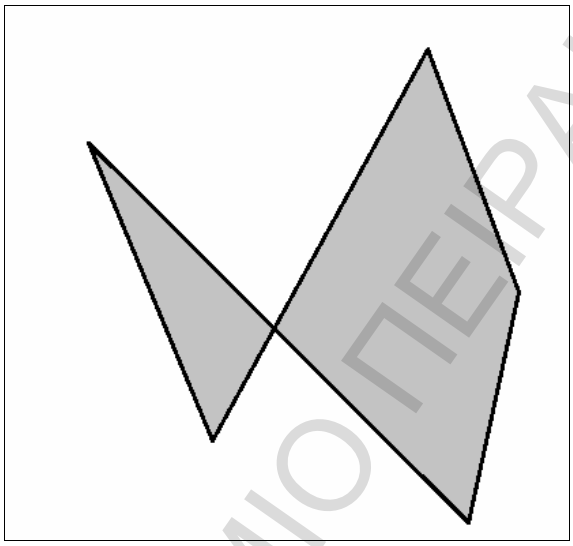
Οι παραπάνω λειτουργιές αξιοποιούν πρόσθετες δυνατότητες των smartphones μέσω του *API*, που επιτρέπουν τη δημιουργία φορμών εισαγωγής, τη μόνιμη αποθήκευση δεδομένων, την αποστολή μηνυμάτων ηλεκτρονικού ταχυδρομείου κ.α.

Στις επόμενες παραγράφους περιγράφονται και αναλύονται τα μαθηματικά και τεχνολογικά εργαλεία στα οποία η εφαρμογή οφείλει την παραπάνω λειτουργικότητα.

2.2 ΥΠΟΛΟΓΙΣΤΙΚΕΣ ΜΕΘΟΔΟΙ

ΥΠΟΛΟΓΙΣΜΟΣ ΕΜΒΑΔΟΥ ΑΠΛΟΥ ΠΟΛΥΓΩΝΟΥ

Με τον όρο *απλό πολύγωνο* εννοούμε ένα πολύγωνο του οποίου οι πλευρές δεν τέμνονται μεταξύ τους. Παράδειγμα μη-απλού πολυγώνου αποτελεί αυτό της επόμενης εικόνας.



Εικόνα 2: Παράδειγμα μη απλό πολύγωνου. Ονομάζεται έτσι διότι υπάρχει τουλάχιστο ένα ζεύγος πλευρών οι οποίες αλληλοτέμνονται.

Στα απλά πολύγωνα είναι δυνατός ο υπολογισμός του εμβαδού με χρήση μιας ιδιαίτερα απλής υπολογιστικής μεθόδου που έχει την ονομασία *Gauss Area Theorem*.

Δοθέντων των συντεταγμένων των κορυφών $(x_1, y_1), (x, y_2), \dots (x_n, y_n)$ πολυγώνου (απαιτείται να είναι διατεταγμένες κυκλικά, ωρολογιακά η μη), το συνολικό εμβαδό του πολυγώνου P υπολογίζεται από τον τύπο:

$$P = \frac{1}{2}\sum_{i-1}^{n}(x_i y_{i+1} - x_{i+1} y_i)$$

Παρατηρούμε ότι αθροίζει τη διαφορά του χιαστί γινομένου των συντεταγμένων των διαδοχικών κορυφών και γι' αυτό συναντάται στη βιβλιογραφία και ως *Shoelace Formula*.

Η εφαρμογή κάνει χρήση την συγκεκριμένης μεθόδου, καθώς χαρακτηρίζεται από απλότητα στην υλοποίησή και ταχύτητα στην εκτέλεση της. Ωστόσο εισάγει μικρό σφάλμα καθώς δε λαμβάνει υπ' όψιν την καμπυλότητα της Γης, γι αυτό και είναι κατάλληλη για περιοχές γης μεσαίου-μικρού μεγέθους (της τάξεως μερικών km^2).

ΥΠΟΛΟΓΙΣΜΟΣ ΜΗΚΟΥΣ ΜΕΤΑΞΥ ΣΗΜΕΙΩΝ ΣΤΗΝ ΕΠΙΦΑΝΕΙΑ ΤΗΣ ΓΗΣ

Ο υπολογισμός του μήκους του τόξου που οριοθετείται από δύο σημεία στη γήινη επιφάνεια γίνεται με τον *τύπο haversine*.

Έστω δύο τυχαία σημεία (φ_1, λ_1), (φ_2, λ_2) στην επιφάνεια της Γης (φ: γεωγραφικό πλάτος, λ: γεωγραφικό μήκος), και R η ακτίνα της Γής. Η ελάχιστη απόσταση d μεταξύ των σημείων που θα διάνυε κάποιος πορευμένος στην επιφάνεια δίνεται από τον τύπο:

$$d = R \times c$$

όπου

$$c = atan2(\sqrt{a}, \sqrt{1-a})$$

ενώ

$$a = sin^2\left(\frac{\Delta\varphi}{2}\right) + cos(\varphi_1) \times cos(\varphi_2) \times sin^2\left(\frac{\Delta\lambda}{2}\right)$$

Ο τύπος αυτός χαρακτηρίζεται από μεγάλη ακρίβεια διότι λαμβάνει υπ' όψη την καμπυλότητα της Γης, ωστόσο εισάγει σφάλμα που προέρχεται από την παραδοχή ότι η Γη είναι σφαιρική με σταθερή ακτίνα[2].

Η εφαρμογή υλοποιεί τη συγκεκριμένη μέθοδο, προκειμένου να υπολογίσει το μήκος των ακμών του πολυγώνου της μετρούμενης περιοχής, και κατόπιν το παράγωγο τους μέγεθος της περιμέτρου.

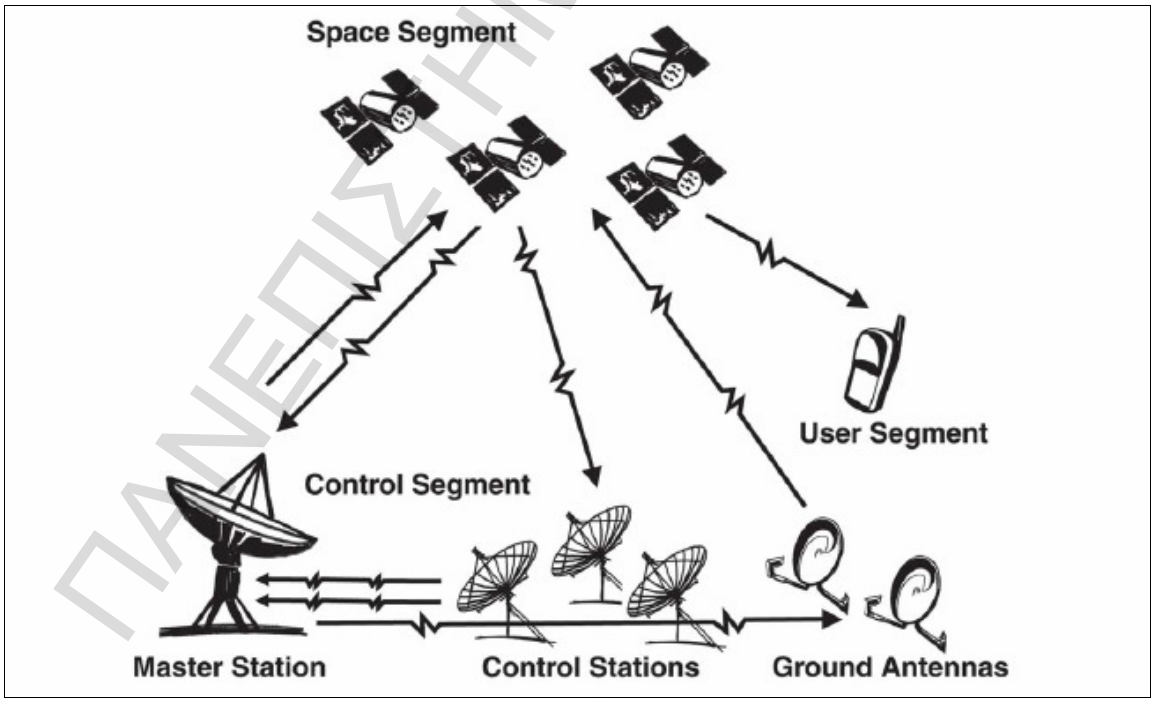
[2] Η Γη έχει ελλειψοειδές σχήμα, και κατ' επέκταση μη σταθερή ακτίνα

2.3 Η ΤΕΧΝΟΛΟΓΙΑ *GPS* - ΑΚΡΙΒΕΙΑ

Η δυνατότητα των σύγχρονων συσκευών, για την καταγραφή και τον υπολογισμό της τρέχουσας θέσης με χρήση της τεχνολογίας *GPS*, έχει ιδρύσει μια νέα κατηγορία εφαρμογών με αντικείμενο την πλοήγηση, την εύρεση κοντινών σημείων ενδιαφέροντος, την αναφορά θέσης κ.α. Η εφαρμογή που αναπτύξαμε αξιοποιεί αυτή τη δυνατότητα και γι' αυτό θα αναφέρουμε ορισμένα στοιχεία της τεχνολογίας αυτής, καθώς επηρεάζουν άμεσα τη λειτουργία της και διαμορφώνουν τις πρακτικές ορθής χρήσης της.

ΠΕΡΙΓΡΑΦΗ-ΑΡΧΕΣ ΛΕΙΤΟΥΡΓΙΑΣ

Το *GPS* (*Global Positioning System*) είναι ένα σύστημα αποτελούμενο από 29 δορυφόρους και 6 επίγειους σταθμούς. Οι δορυφόροι εκπέμπουν ραδιοκύματα που μεταφέρουν πληροφορίες θέσης, χρόνου εκπομπής του σήματος κ.α., τις οποίες επεξεργάζονται οι συσκευές-δέκτες τον χρηστών για τον υπολογισμό της τρέχουσας θέσης. Ο αριθμός χρηστών που μπορούν να εξυπηρετούνται ταυτόχρονα είναι απεριόριστος, ενώ η κάλυψη του σήματος είναι παγκόσμια. Οι επίγειοι σταθμοί είναι υπεύθυνοι για την επίβλεψη των δορυφόρων και για τυχόν διορθώσεις στην τροχιά τους.

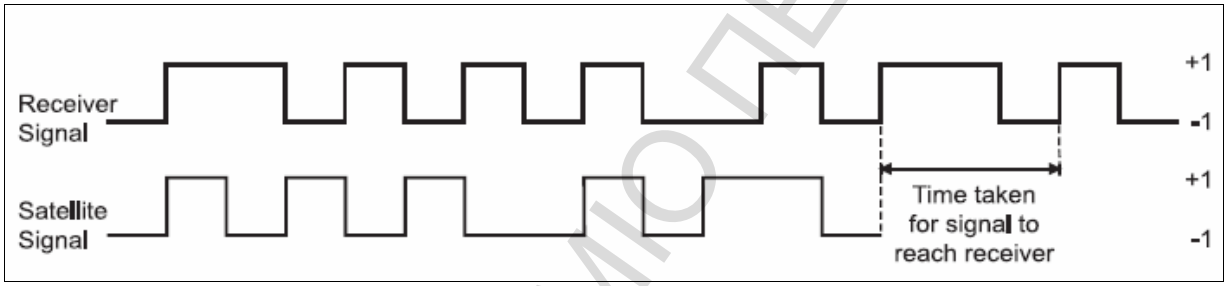


Εικόνα 3: Απεικόνιση των μερών που συναποτελούν το σύστημα *GPS*

Αναφέρουμε παρακάτω πιο αναλυτικά τον τρόπο λειτουργίας και το ρόλο των εμπλεκομένων μερών του *GPS*.

2.3.1.1 ΤΡΟΠΟΣ ΛΕΙΤΟΥΡΓΙΑΣ

Οι δορυφόροι του *GPS* περιφέρονται γύρω από τη Γη ανά 12 ώρες σε μεγάλο ύψος (12000 μιλίων) για την ευρύτερη κάλυψη του σήματος. Είναι διατεταγμένοι μεταξύ τους κατά τρόπο ώστε ανά δεδομένη στιγμή ένας δέκτης να καλύπτεται από τουλάχιστο τέσσερις δορυφόρους. Οι δορυφόροι εκπέμπουν βραχέα ραδιοκύματα σε διαφορετικές συχνότητες τα οποία μεταφέρουν μια ψευδο-τυχαία ακολουθία από bits (pseudorandom code). Η ακολουθία αυτή ταυτοποιεί μοναδικά τον κάθε δορυφόρο, και χρησιμεύει στον υπολογισμό από μεριά του δέκτη, του χρόνου μετάδοσης του σήματος σε αυτόν. Για να το επιτύχει αυτό ο δέκτης, αναπαράγει ταυτόχρονα με το δορυφόρο την ίδια ακολουθία, και κατόπιν τη συγκρίνει με την ληφθείσα προκειμένου να υπολογίσει τη χρονική υστέρηση της τελευταίας.



Εικόνα 4: Υπολογισμός της χρονικής υστέρησης της ακολουθίας του δορυφόρου ως προς την αντίστοιχη του δέκτη *GPS*

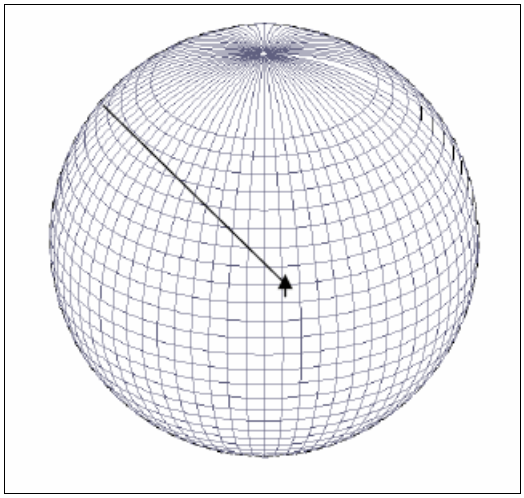
Η χρονική αυτή υστέρηση ισούται με το χρόνο μετάδοσης του σήματος από το δορυφόρο και αποτελεί τον όρο Δt στην εξίσωση για τον υπολογισμό της τρέχουσας απόστασης δέκτη-δορυφόρου:

$$d = c \times \Delta t$$

όπου c η ταχύτητα των ραδιοκυμάτων στο κενό (186000 miles/s).

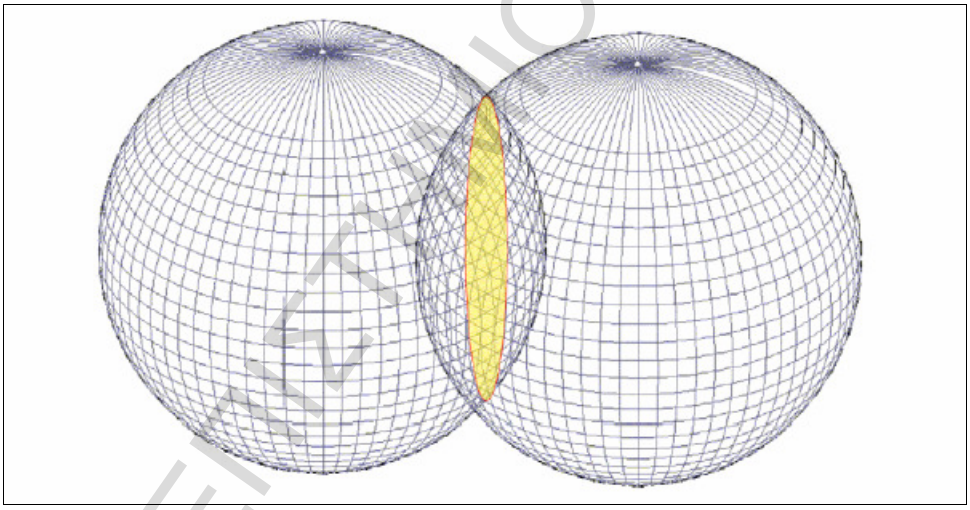
Ο δέκτης επίσης γνωρίζει την τρέχουσα θέση του δορυφόρου, καθώς με την ενεργοποίησή του, κατεβάζει από το δίκτυο *GPS* πληροφορίες των τροχιών του δικτύου των δορυφόρων, τις αποθηκεύει στη μνήμη και τις αξιοποιεί καθ' όλη τη διάρκεια της λειτουργίας του.

Η γνώση της θέσης του δορυφόρου και της απόστασης από το δέκτη επιτρέπει την εξαγωγή ενός πρώτου συμπεράσματος για τη θέση του δέκτη ως προς αυτή τουδορυφόρου. Στην επόμενη εικόνα απεικονίζεται ο γεωμετρικός τόπος των δυνατών θέσεων, που είναι μια σφαιρική επιφάνεια με το δορυφόρο στο κέντρο της περιεχόμενης σφαίρας.



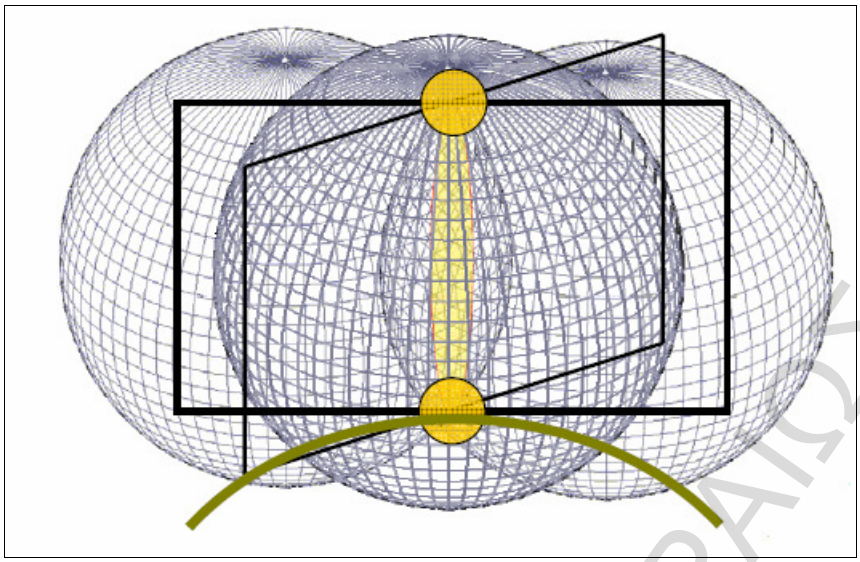
Εικόνα 5: Ο Γεωμετρικός τόπος σημείων δυνατών θέσεων του δέκτη *GPS* όταν γνωρίζουμε την απόσταση του από έναν μόνο δορυφόρο, είναι μια σφαιρική επιφάνεια.

Η εξαγόμενη αυτή πληροφορία φυσικά δεν μπορεί να αξιοποιηθεί για τη επίλυση κάποιου από τα γνωστά προβλήματα που συναντούμε, καθώς είναι πολύ αόριστη. Γι αυτό ο δέκτης αναπαράγει την παραπάνω διεργασία και για ένα δεύτερο δορυφόρο *GPS* που βρίσκεται στο οπτικό του πεδίο. Οι δυνατές θέσεις θα έχουν περιορισθεί σύμφωνα με το παρακάτω σχήμα:



Εικόνα 6: Ο Γεωμετρικός τόπος σημείων δυνατών θέσεων του δέκτη *GPS* όταν γνωρίζουμε την απόσταση του από δύο δορυφόρους, είναι ένας κύκλος.

Ο δέκτης θα πρέπει να κείτεται σε κάποιο από τα σημεία του κύκλου που σχηματίζεται από την τομή των σφαιρών προσδιορισμού θέσης. Ο συνυπολογισμός ενός ακόμη δορυφόρου δίνει ακόμη περισσότερο συγκεκριμένη εικόνα για το που βρίσκεται ο δέκτης όπως φαίνεται παρακάτω:



Εικόνα 7: Ο Γεωμετρικός τόπος σημείων δυνατών θέσεων του δέκτη GPS όταν γνωρίζουμε την απόσταση του από τρείς δορυφόρους, είναι σύνολο δύο σημείων.

Ο γεωμετρικός τόπος των δυνατών σημείων έχει περιορισθεί πλέον σε δύο μεμονωμένα σημεία, εκ των οποίων μονάχα το ένα μπορεί να βρίσκεται στη γήινη επιφάνεια και να αποτελεί αποδεκτή λύση.

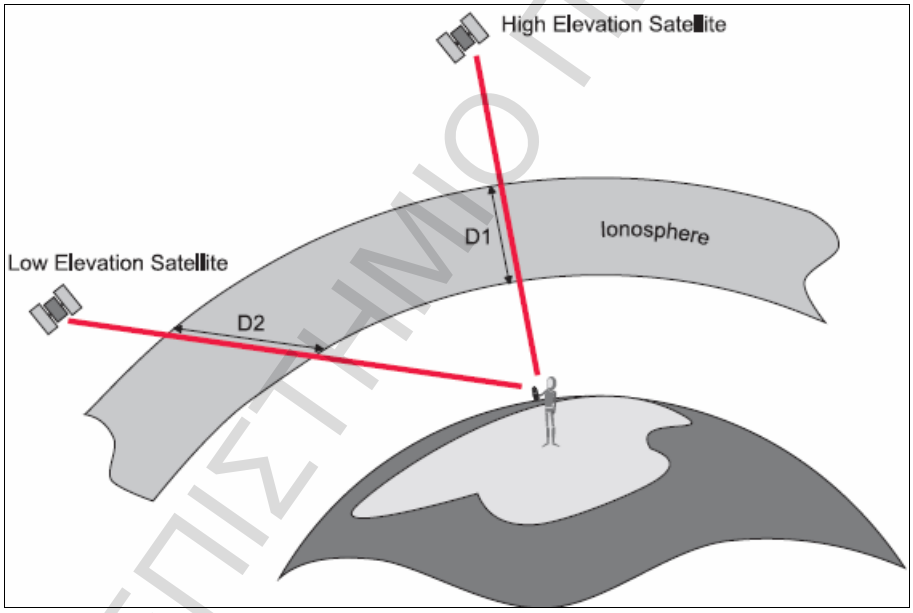
Το παραπάνω μοντέλο, περιγράφει ένα εξιδανικευμένο σύστημα στο οποίο ο δέκτης διαθέτει ένα απόλυτα συγχρονισμένο ρολόι ως προς το παγκόσμιο σύστημα ώρας (*UT*). Αυτό είναι αληθές μονάχα για τα ατομικά ρολόγια που βρίσκονται πάνω στους δορυφόρους, κι όχι για τα αντίστοιχα των δεκτών που είναι κατασκευασμένα με λιγότερο ακριβές τεχνολογίες. Κατ' επέκταση, τυχόν εσφαλμένη εκτίμηση του Δt στον τύπο, εισάγει σφάλμα στην απόσταση d και ακόλουθα στη θέση του δέκτη.

Γι' αυτό απαιτείται η ύπαρξη ενός τέταρτου δορυφόρου, ώστε να ελέγχεται εάν η σφαίρα δυνατών θέσεων που δημιουργεί, περιέχει το υπολογιζόμενο σημείο θέσης του δέκτη, και αν όχι, να υπολογίζεται ο διορθωτικός παράγοντας χρόνου ο οποίος θα εξαλείψει το σφάλμα των μετρήσεων. Συγκεκριμένα θα αφαιρεθεί από τις καταγεγραμμένες τιμές Δt, και θα συγχρονίσει ξανά το ρολόι του με την παγκόσμια ώρα. Κατ' αυτό τον τρόπο επιτυγχάνουμε τον ακριβή προσδιορισμό της θέσης του δέκτη δίχως να απαιτείται ο δέκτης να είναι απόλυτα συγχρονισμένος.

2.4 ΠΗΓΕΣ ΣΦΑΛΜΑΤΟΣ

Από την περιγραφή της λειτουργίας του *GPS*, γίνεται σαφές ότι αποτελεί ένα ιδιαίτερα πολύπλοκο σύστημα με πολλά εμπλεκόμενα μέρη εκ' των οποίων ορισμένα δημιουργούν θόρυβο, που έχει ως συνέπεια την εισαγωγή σφάλματος στις τελικές μετρήσεις. Οι πηγές αυτές του σφάλματος απαριθμούνται στην επόμενη λίστα.

- ***Παρουσία αερίων στην Ιονόσφαιρα***: Στον τύπο της απόστασης ($d = c \times \Delta t$), η ταχύτητα του φωτός είναι σταθερή και ίση με 186000 miles/s μονάχα στο απόλυτο κενό, ενώ παρουσία αερίων όπως συμβαίνει στην Ιονόσφαιρα η ταχύτητα ελαττώνεται. Η ελάττωση αυτή δεν είναι σταθερή καθώς εξαρτάται από ένα σύνολο παραγόντων. Η σχετική θέση του δορυφόρου ως προς το δέκτη καθορίζει το πάχος της Ιονόσφαιρας που διατρέχει το ηλεκτρομαγνητικό σήμα όπως φαίνεται στην επόμενη εικόνα:

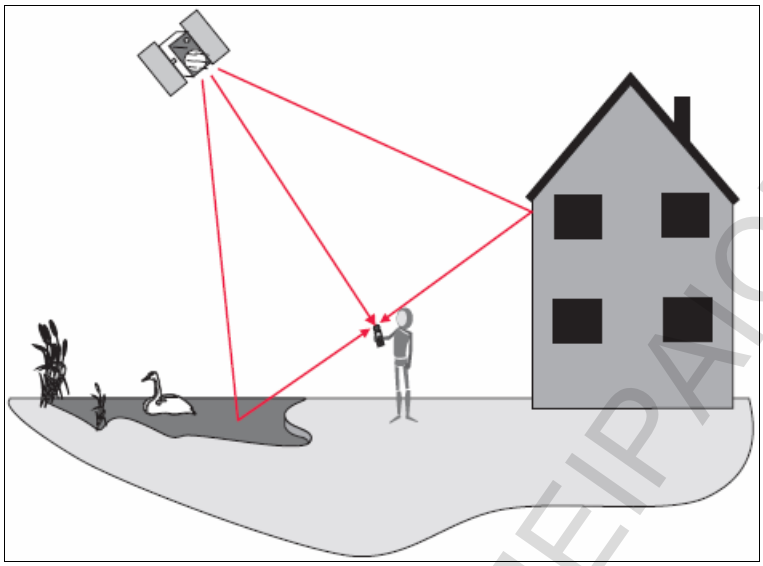


Εικόνα 8: Επίδραση του πάχους της Ιονόσφαιρας στην ακρίβεια του *GPS*. Ο δορυφόρος στα αριστερά θα εισάγει μεγαλύτερο σφάλμα απ' ότι ο δορυφόρος στα δεξιά, λόγω του ότι το σήμα του διατρέχει μεγαλύτερη απόσταση μέσα στο στρώμα της Ιονόσφαιρας.

Όσο μεγαλύτερο το πάχος, τόσο εντονότερη είναι και η καθυστέρηση του σήματος. Ωστόσο, ο συντελεστής αλληλεπίδραση του σήματος με την ιονόσφαιρα μειώνεται με την απουσία της ηλιακής ακτινοβολίας και ως εκ τούτου, οι μετρήσεις τη νύχτα είναι πιο ακριβείς.

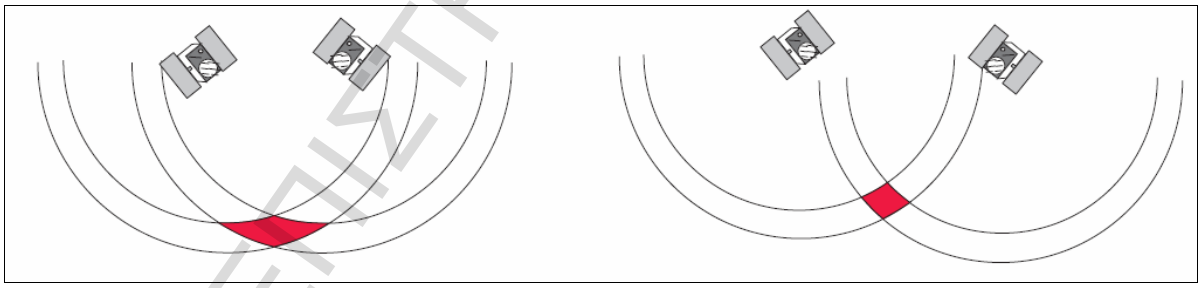
- ***Ανακλάσεις του σήματος***: Το απερχόμενο σήμα από το δορυφόρο, πρώτου φθάσει στο δέκτη, ενδέχεται να έχει υποστεί μια η και περισσότερες ανακλάσεις σε επιφάνειες σωμάτων που βρίσκονται κοντά, είτε παρεμβάλλονται στην οπτική ακτίνα μεταξύ δορυφόρου-δέκτη. Τέτοια σώματα μπορεί να είναι κτήρια, λίμνες, υπέργειες γραμμές ηλεκτρικού ρεύματος. Αντιμετωπίζεται με τη χρήση

ειδικών κεραιών που εξαλείφουν την επίδραση των σημάτων που προέρχονται από ανάκλαση στο έδαφος. Ακόμη, στα smartphones δεν υπάρχει η δυνατότητα διασύνδεσης τέτοιων κεραιών.



Εικόνα 9: Σφάλμα προερχόμενο από ανακλάσεις του σήματος σε κοντινά στο δέκτη σώματα.

- ***Γεωμετρία δορυφόρων***: Καλύτερος προσδιορισμός της θέσης επιτυγχάνεται όταν οι δορυφόροι είναι απομακρυσμένοι μεταξύ τους κι όχι ομαδοποιημένοι σε συγκεκριμένο χώρο του ουρανού, όπως είναι ορατοί από το δέκτη. Στην επόμενη εικόνα απεικονίζονται σε δύο διαστάσεις οι δύο αναφερόμενες περιπτώσεις καθώς και το πως επηρεάζουν την ακρίβεια της μέτρησης.



Εικόνα 10: Το ζεύγος δορυφόρων στα δεξιά, δίνει πιο ακριβείς μετρήσεις απ' ότι το ζεύγος στα αριστερά, καθώς απέχουν περισσότερο μεταξύ τους αφήνοντας έτσι ένα μικρότερο παράθυρο αβεβαιότητας όπως απεικονίζεται από την κοκκινισμένη περιοχή στις περιπτώσεις του σχήματος.

- ***Απόκλιση τροχιάς δορυφόρων***: Εξ' αιτίας της επίδρασης της βαρύτητας της σελήνης και τους ηλιακούς ανέμους, συχνά επηρεάζεται η τροχιά των δορυφόρων με αποτέλεσμα να αποκλίνουν από την προβλεπόμενη από το δέκτη τρέχουσα θέση. Το επίγειο σύστημα των σταθμών είναι υπεύθυνο για τον έλεγχο και διόρθωση των τροχιών μέσω σημάτων που εκπέμπουν με επίγειες κεραίες.

Οι παραπάνω πηγές σφάλματος, δεν αφήνουν ανεπηρέαστους και τους δέκτες που είναι ενσωματωμένοι στα smartphones, επηρεάζοντας έτσι τις εφαρμογές που

στηρίζουν τη λειτουργικότητά τους σε αυτούς. Στην επόμενη παράγραφο αναφέρονται οι πρακτικές για την ορθή χρήση της εφαρμογής ώστε να μειώνονται οι επιδράσεις των σφαλμάτων αυτών, καθώς και τη μέθοδο επεξεργασίας των μετρήσεων από την εφαρμογή την ίδια για την επίτευξη του σκοπού αυτού.

ΕΛΛΑΤΩΣΗ ΤΟΥ ΣΦΑΛΜΑΤΟΣ ΚΑΤΑ ΤΗ ΧΡΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Ο βασικός τρόπος εξάλειψης του σφάλματος από την εφαρμογή, είναι αθροίζοντας σε ένα πλήθος μετρήσεων αντί μιας μεμονωμένης μέτρησης για κάθε υπολογιζόμενη θέση. Συγκεκριμένα υπολογίζει το μέσο όρο 25 μετρήσεων γεωγραφικού πλάτους και μήκους ανά θέση, επιτυγχάνοντας έτσι καλύτερη σύγκλιση στην αναμενόμενη τιμή.

Παρ' ότι η άθροιση πολλών μετρήσεων αυξάνει την ακρίβεια της υπολογιζόμενης θέσης, ενδεχομένως κάποιο αποτέλεσμα να εξακολουθεί να αποκλίνει από την αναμενόμενη θέση. Τότε ο χρήστης έχει τη δυνατότητα να κάνει προεπισκόπηση των σημείων σε χάρτη ώστε να διαπιστώσει αν η υπολογιζόμενη τοποθεσία ταυτίζεται πραγματικά με την πραγματική θέση. Αν αυτό δεν ισχύει μπορεί να διαγράψει την εσφαλμένη μέτρηση και να την επαναλάβει.

Επίσης θα πρέπει να φροντίζει να μην περιορίζει το οπτικό πεδίο του δέκτη του με τον ουρανό, παρακάμπτοντας εάν είναι εφικτό ανάλογα εμπόδια, ενώ ενδείκνυται να τοποθετεί το σώμα του βόρεια ως προς το δέκτη μια και οι περισσότεροι δορυφόροι της υπηρεσίας *GPS* εντοπίζονται σ το νότο. Να διενεργεί τις μετρήσεις τις βράδυνες ώρες που δεν υπάρχει ήλιος και να επιλέγει ημέρες με χαμηλή υγρασία στην ατμόσφαιρα.

3 ΤΕΧΝΟΛΟΓΙΕΣ ΕΦΑΡΜΟΓΗΣ

Η εφαρμογή αναπτύχθηκε σε περιβάλλον *.NET Version 4.5*, με χρήση του *Visual Studio 2010 Integrated Development Environment*, που ενσωματώνει όλα τα απαραίτητα modules (*Editor, Compiler, Linker, Debugger*) για την ανάπτυξη εφαρμογών Desktop και Web τύπου, βασισμένες στις τεχνολογίες της *Microsoft* (*Visual Basic , WPF, asp κ.α.*). Επιπλέον, απαιτήθηκε η εγκατάσταση του *Windows Phone SDK Version 7.1*, το οποίο περιέχει τις απαιτούμενες βιβλιοθήκες καθώς και πρόσθετα εργαλεία ανάπτυξης για mobile εφαρμογές.

Οι βιβλιοθήκες που επιτρέπουν την ανάπτυξη mobile εφαρμογών, συμπεριλαμβάνονται στο πακέτο *Silverlight SDK Version 4*. Η τεχνολογία *Silverlight* είναι η εναλλακτική λύση της *asp.NET* για την ανάπτυξη web εφαρμογών, με χρήση γλωσσών που παραδοσιακά προορίζονται για εφαρμογές Desktop, όπως ή *C#* και η *Visual Basic*. Ειδική έκδοσή του *Silverlight* για mobile εφαρμογές, χρησιμοποιήθηκε εδώ.

Οι γλώσσες ανάπτυξης είναι η *C#* και η *XAML*. Σε *C#* γράφτηκαν όλες οι κλάσεις του συστήματος, οι οποίες επιμερίσθηκαν σε δύο *assemblies* που αντιστοιχούν στα δύο διακριτά *layers* της εφαρμογής (*Presentation Layer* και *Model Payer*).

Με την *XAML* υλοποιήθηκε το *UI*, αλλά και ένα μέρος της λειτουργικότητας της εφαρμογής παρακάμπτοντας τη συγγραφή ανάλογου κώδικα σε *C#*. Η *XAML* ακολουθεί τους κανόνες σύνταξης της *XML*, και επιτρέπει το διαχωρισμό του *UI* από τον runtime κώδικα διευκολύνοντας τη δομημένη συγγραφή του κώδικα. Επιπλέον περιέχει δηλώσεις με τις οποίες αποδίδεται και λειτουργικότητα στα στοιχεία ελέγχου του *UI* (*Buttons*, *Textboxes*) μέσω ενός εσωτερικού συστήματος διασύνδεσης των οντοτήτων, μειώνοντας την έκταση του κώδικα σε *C#* καθώς και την πιθανότητα εμφάνισης σφαλμάτων. Το μόνο αρνητικό στοιχείο της *XAML* είναι ο μεγάλος χρόνο εκμάθησης λόγω του μεγάλου της εύρους.

Αναφορικά με τα εργαλεία του *Windows Phone SDK*, χρησιμοποιήθηκε το *Microsoft Expression Blend* για τη διαμόρφωση ορισμένων από τις οθόνες της εφαρμογής. Παρέχει ευκολίες πάνω στην ανάπτυξη του *UI* επιταχύνοντας την όλη διαδικασία, ενώ χάρη στο παραθυρικό του περιβάλλον, επιτρέπει το σχεδιασμό πολύπλοκων οθονών παρακάμπτοντας τη χρήση της *XAML*.

Θα αναφέρουμε στις επόμενες παραγράφους λεπτομέρειες της λειτουργίας των ανωτέρω τεχνολογιών, που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής.

3.1 *WEB SERVICES*

Η εφαρμογή χρησιμοποιεί τα *Bing Maps Web services* προκειμένου να ενημερώνει σε πραγματικό χρόνο το χρήστη για την τρέχουσα θέση του, εκφρασμένη σε αστικές συντεταγμένες (Περιοχή, Οδός, Αριθμός κλπ.). Για τη λειτουργία αυτή απαιτείται πρόσβαση στο internet, για την αποστολή των συντεταγμένων θέσης όπως αυτές καταγράφονται από το *GPS* προς το web server, ο οποίος τις μετατρέπει σε αστικές με τη βοήθεια των αποθηκευμένων πληροφοριών στη βάση δεδομένων που διατηρεί. Δε θα ήταν εφικτή τέτοια λειτουργικότητα αν δεν είχε αναπτυχθεί η τεχνολογία των *Web services*, και γι' αυτό θα περιγράψουμε τα βασικά της χαρακτηριστικά, δίνοντας έμφαση στα *RESTful Web services* στα οποία ανήκουν τα services της *Bing Maps*.

XML-BASED *WEB SERVICES*

Τα *Web services* επεκτείνουν την υποδομή του διαδικτύου παρέχοντας τη δυναμική της διασύνδεσης εφαρμογών. Οι εφαρμογές προσπελάζουν τα services μέσω καθιερωμένων Web πρωτοκόλλων και τεχνολογιών όπως *HTTP*, *XML*, *SOAP*, *REST*, δίχως να γνωρίζουν τις λεπτομέρειες υλοποίησης τους.

Ένα *XML* Web service αποτελεί μια προγραμματιστική οντότητα η οποία παρέχει συγκεκριμένη λειτουργικότητα σε επίπεδο απλών διεργασιών, ή και σε επίπεδο επιχειρησιακής λογικής ανάλογα το σκοπό για τον οποίο αναπτύχθηκε, και μπορεί να προσπελάζεται από απεριόριστο αριθμό ετερογενών συστημάτων, που υποστηρίζουν καθιερωμένες internet τεχνολογίες και πρωτόκολλα όπως *XML*, και *HTTP*. Μπορούν να λειτουργούν τοπικά σε μια εφαρμογή, είτε να διατίθενται μέσω internet προς άλλες εφαρμογές.

Τα *XML Web services* χρησιμοποιούν μηνύματα *XML* για την επικοινωνία των συστημάτων μιας συνεδρίας, γεφυρώνοντας έτσι το χάσμα που υπάρχει μεταξύ τους λόγω των διαφορετικών μοντέλων οντοτήτων, λειτουργικών συστημάτων και γλωσσών προγραμματισμού. Με αυτό τον τρόπο είναι εφικτή η ανάπτυξη εφαρμογών, οι οποίες οικοδομούνται πάνω στα *Web services* διαφορετικών πηγών, κατά τον τρόπο που παραδοσιακά οι κατανεμημένες εφαρμογές οικοδομούνται πάνω σε ομογενείς μονάδες λειτουργικότητας (*components*).

Τα *Web services* εισάγουν μια νέα προοπτική στην ανάπτυξη κατανεμημένων συστημάτων, καθώς διασύνδεουν με γενικευμένα (αντί ειδικά) "*συμβόλαια*" και με ευρέως διαδεδομένες τεχνολογίες τις διάφορες εφαρμογές, προς χάριν της διαλειτουργικότητας. Αντιθέτως, στα παραδοσιακά grid συστήματα, όλα τα μέρη είναι διασυνδεδεμένα με πρότυπα που έχουν δημιουργηθεί για το συγκεκριμένο σύστημα, καθιστώντας τα λιγότερο ευέλικτα ως προς τη διασύνδεση τους με άλλα συστήματα.

ΠΕΔΙΑ ΑΞΙΟΠΟΙΗΣΗΣ ΤΩΝ *WEB SERVICES*

Η πιο απλή χρήση των *Web services* είναι για την εκτέλεση θεμελιωδών διεργασιών σε εφαρμογές. Για παράδειγμα, στα ηλεκτρονικά καταστήματα, μια συνηθισμένη λειτουργία είναι ο υπολογισμός των εξόδων αποστολής των προϊόντων, δεδομένων των στοιχείων επικοινωνίας. Αυτό θα απαιτούσε, τα τιμολόγια των μεταφορικών εταιριών που συνεργάζεται η επιχείρηση, να είναι διαθέσιμα ανά πάσα στιγμή στην εφαρμογή που είναι ιδιαίτερα δύσκολο, έως ανέφικτο. Με τη χρήση *Web services*, η εφαρμογή στέλνει ένα *XML* μήνυμα, μέσω του *HTTP* πρωτοκόλλου με αποδέκτη το *Web service* της μεταφορικής εταιρείας, δίνοντας ως ορίσματα τις απαιτούμενες παραμέτρους (βάρος δέματος, διαστάσεις, τοποθεσία κέντρου διανομής, προορισμός). Κατόπιν, το service θα επεξεργαστεί τα δεδομένα αφ' ότου συμβουλευτεί τον πίνακα τιμολογίων και υπολογίσει το συνολικό κόστος, και θα το επιστρέψει στην εφαρμογή μέσω μηνύματος *XML*, ώστε η τελευταία να το συμπεριλάβει στη συνολική χρέωση του πελάτη.

Επίσης, τα *Web services* μπορούν να αξιοποιηθούν για τη ολοκλήρωση ανεξάρτητων εφαρμογών μεταξύ τους. Η ευρεία ενσωμάτωση εξειδικευμένων εφαρμογών στα περισσότερα τμήματα των οργανισμών, έχει ως αποτέλεσμα τη δημιουργία πολλών χρήσιμων αλλά απομονωμένων νησίδων πληροφορίας και λειτουργικότητας. Εξ' αιτίας των διαφορετικών συνθηκών υπό τις οποίες αναπτύσσονται οι εφαρμογές αυτές, είναι ιδιαίτερα δύσκολο να ενοποιηθεί η λειτουργικότητά τους. Η τεχνολογία των *Web services* επιτρέπει την πρόσβαση στη λειτουργικότητα και τα δεδομένα των εφαρμογών αυτών μέσω κοινού *API*, με στόχο τη δημιουργία εφαρμογής που θα ολοκληρώνει αυτές τις εφαρμογές, αποκτώντας πρόσβαση σε αυτές μέσω των ιδιόκτητων *Web services* τους.

Άλλες εφαρμογές των *Web services* αποτελούν τα συστήματα που υλοποιούν πλήρη διαγράμματα ροής εργασίας, και που συναντώνται συνήθως σε οργανισμούς με πολύπλοκες επιχειρησιακές διαδικασίες. Για την ανάπτυξη τέτοιων λύσεων προτείνονται έτοιμα frameworks, τα οποία παρέχουν το ανάλογο υπόβαθρο λειτουργικότητας, ώστε το τελικό σύστημα να ανταποκρίνεται στις ανάγκες του οργανισμού και παράλληλα να είναι σταθερό. Η *Microsoft* έχει αναπτύξει το *BizTalk*

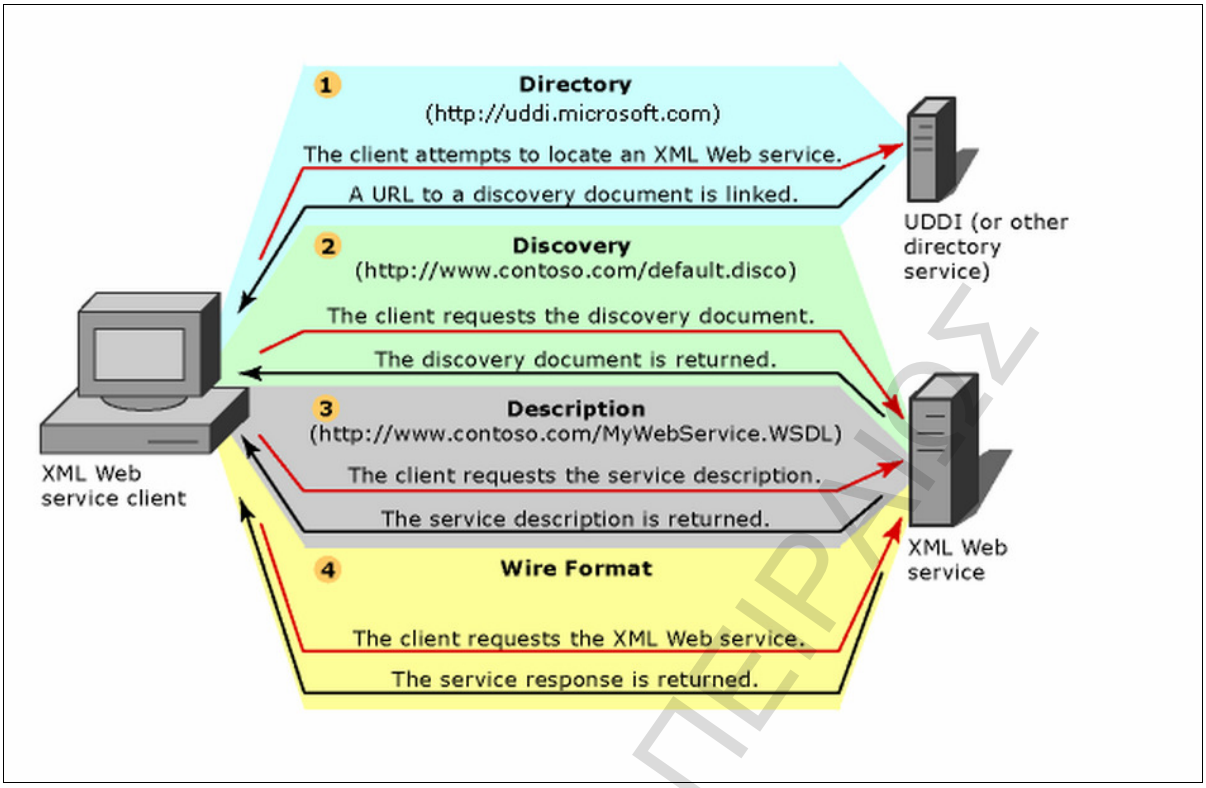
Orchestration που παρέχει έτοιμους μηχανισμούς για την ολοκλήρωση, διαχείριση και αυτοματοποίηση των επιχειρησιακών διαδικασιών, μέσω της ανταλλαγής εγγράφων και μηνυμάτων, ανάμεσα στις εφαρμογές ενός οργανισμού.

ΥΠΟΔΟΜΗ ΤΩΝ *WEB SERVICES*

Τα *Web services* αναπτύσσονται και αξιοποιούνται ανεξάρτητα από τα λειτουργικά συστήματα, τα μοντέλα οντοτήτων ή και τις γλώσσες προγραμματισμού των συστημάτων-χρηστών τους, γεγονός που συντελεί στην ευρεία τους διάδοση στο διαδίκτυο. Επίσης, για την περαιτέρω διάδοσή τους, οι developers θα πρέπει να λαμβάνουν υπ' όψιν και τις παρακάτω προδιαγραφές:

- ***Χαλαρή διασύνδεση***: Δύο συστήματα θεωρούνται χαλαρά διασυνδεδεμένα όταν η μοναδική συνθήκη που πρέπει να ικανοποιούν για την μεταξύ τους επικοινωνία, είναι να ερμηνεύουν ορθά τα μηνύματα *XML* που ανταλλάσσουν, δίχως τα τελευταία να απαιτείται να έχουν κάποια αυστηρά προτυποποιημένη μορφή, που θα απαιτούσε και τη βαθύτερη κατανόηση της λειτουργίας του έτερου συστήματος.
- ***Διαρκής προσβασιμότητα ανεξαρτήτου τοποθεσίας***: Οι servers που φιλοξενούν τα *Web services* θα πρέπει να είναι διασυνδεδεμένοι με τον παγκόσμιο ιστό αδιάκοπα, και να κατανέμονται σε διαφορετικές γεωγραφικές τοποθεσίες προκειμένου να παρέχουν την υπηρεσία με όσο το δυνατό μικρή καθυστέρηση, παρακάμπτοντας πιθανές βλάβες του δικτύου η και συσσωρευμένη κίνηση.
- ***Μηνύματα με προτυποποιημένους κανόνες σύνταξης***: Τα *XML Web services* χρησιμοποιούν μηνύματα *XML*, ώστε να εκμεταλλευτούν την ευρεία διάδοσή τους που εγγυάται και την ορθή τους ερμηνεία τους από τα αλληλεπιδρώντα συστήματα. Η ερμηνεία των μηνυμάτων αυτών είναι αρκετή, και δεν απαιτείται η γνώση των τεχνικών προδιαγραφών υλοποίησης των έτερων συστημάτων της επικοινωνίας με *Web services*.

Η υποδομή των *Web services* απεικονίζεται στην επόμενη εικόνα:



Εικόνα 11: Τα στάδια αναζήτησης, εύρεσης και χρήσης των *SOAP Web services.*

Περιγράφουμε στις επόμενες παραγράφους τα βήματα που αποτελούν τη διαδικασία αναζήτησης και κατανάλωσης των *Web services*.

3.1.3.1 DIRECTORY

Τα *XML Web service* Directories παρέχουν κεντρικές τοποθεσίες όπου οι παροχείς των *Web services* μπορούν να δημοσιεύουν πληροφορίες σχετικά με τα services που παράγουν. Τα directories αυτά μπορεί να αποτελούν τα ίδια *Web services*, προσβάσιμα προγραμματιστικά, με σκοπό να παρέχουν αποτελέσματα αναζήτησης σε εισερχόμενα αιτήματα των *client*s τους. Με τα αιτήματα αυτά μπορεί να γίνεται αναζήτηση με βάση τη ζητούμενη λειτουργικότητα, είτε με βάση τον παροχέα προκειμένου να επιστρέφεται η λίστα με τα *Web services* που ο τελευταίος έχει εκδώσει. Το πρότυπο *UDDI* ορίζει τον τρόπο δημοσίευσης και ανάκτησης πληροφοριών που αφορούν τα *XML Web services*. Συγκεκριμένα ένα *UDDI* αρχείο περιέχει τις εξής πληροφορίες:

- ***Στοιχεία παρόχου (<businessEntity>)***: Συμπεριλαμβάνονται στοιχεία επικοινωνίας, τομέας επιχειρηματικής δραστηριότητας, αναγνωριστικά της επιχείρησης και παρεχόμενα *Web services*. Παράδειγμα αποτελεί η παρακάτω εγγραφή:

```
<businessEntity businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40"
        operator="http://www.ibm.com"
        authorizedName="John Doe">
    <name>Acme Company</name>
    <description>
        We create cool Web services
    </description>
    <contacts>
        <contact useType="general info">
            <description>General Information</description>
            <personName>John Doe</personName>
            <phone>(123) 123-1234</phone>
            <email>jdoe@acme.com</email>
        </contact>
    </contacts>
  <businessServices>
  ...
  </businessServices>
  <identifierBag>
     <keyedReference
        tModelKey="UUID:8609C81E-EE1F-4D5A-B202-3EB13AD01823"
        name="D-U-N-S"
        value="123456789" />
  </identifierBag>
  <categoryBag>
     <keyedReference
        tModelKey="UUID:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2"
        name="NAICS"
        value="111336" />
   </categoryBag>
</businessEntity>
```

Εικόνα 12: Παράδειγμα Εγγραφής *businessEntity* αρχείου *UDDI*.

- ***Στοιχεία του Web service (<businessService>)*:** Περιγράφει ένα μεμονωμένο *Web service* που παρέχει ο συγκεκριμένος οργανισμός. Περιέχει όνομα και περιγραφή καθώς και το μοναδικό κλειδί που έχει αποδοθεί σε αυτό από το *UUID* πρωτόκολλο, μέσω των *attributes serviceKey* και *businessKey*.

```
<businessService serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
        businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">
    <name>Hello World Web Service</name>
    <description>A friendly Web service</description>
    <bindingTemplates>
    ...
    </bindingTemplates>
    <categoryBag />
</businessService
```

Εικόνα 13: Παράδειγμα εγγραφής *businessService* με τα στοιχεία μητρώου του Web service ως προς το πρωτόκολλο *UUID*.

- *Binding* ***information (<bindingTemplate>)***: Αποτελεί την τεχνική περιγραφή του *Web service*, καθώς αναγράφει τα πρωτόκολλα και τις διαδικτυακές διευθύνσεις στις οποίες είναι διαθέσιμο.

```
<bindingTemplate serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
        bindingKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">
     <description>Hello World SOAP Binding</description>
     <accessPoint URLType="http">
         http://localhost:8080
     </accessPoint>
     <tModelInstanceDetails>
         <tModelInstanceInfo
               tModelKey="uuid:EB1B645F-CF2F-491f-811A-4868705F5904">
         <instanceDetails>
              <overviewDoc>
                 <description>
                     references the description of the
                     WSDL service definition
                 </description>
                 <overviewURL>
                      http://localhost/helloworld.wsdl
                 </overviewURL>
              </overviewDoc>
         </instanceDetails>
         </tModelInstanceInfo>
      </tModelInstanceDetails>
</bindingTemplate>
```

Εικόνα 14: Στοιχείο *bindingTemplate* που περιγράφει τις τεχνικές προδιαγραφές του *Web service.*

- ***Service specifications (<tModel>)***: Αποτελεί μια γενικευμένη δομή που προτυποποιεί τις ως άνω δομές που συνθέτουν τις εγγραφές *UDDI*. Κατ' αυτό τον τρόπο εξυπηρετεί ως re*Source* και μπορεί να αναφέρεται σε διαφορετικά *Web service* instances, εφ' όσον είναι συμβατή με τον τύπο του *element* που την περιέχει.

```
<tModel tModelKey="uuid:xyz987..."
           operator="http://www.ibm.com"
           authorizedName="John Doe">
    <name>HelloWorldInterface Port Type</name>
    <description>
           An interface for a friendly Web service
    </description>
    <overviewDoc>
           <overviewURL>
                http://localhost/helloworld.wsdl
           </overviewURL>
     </overviewDoc>
</tModel>
```

Εικόνα 15: Παράδειγμα στοιχείου *tModel* με την τεχνική περιγραφή ενός *Web service*.

3.1.3.2 DISCOVERY

Στο στάδιο αυτό γίνεται εντοπισμός ενός ή περισσοτέρων εγγράφων μορφής *WSDL* (*Web Services Description Language*) που σχετίζονται με το *Web service* υπό αναζήτηση. Συγκεκριμένα, ενημερώνονται οι *client*s ότι υπάρχει *Web service* σχετικό με το query, και τους επιστρέφονται οι διευθύνσεις όπου υπάρχουν τα αρχεία

περιγραφής. Το αρχείο που περιέχει τα links των *WSDL* εγγράφων είναι τύπου *.disco*, που έχει σύνταξη *XML*.

```
<?xml version="1.0" encoding="utf-8" ?>
<discovery xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns="http://schemas.xmlsoap.org/disco/">
  <contractRef ref="http://www.contoso.com/Counter.asmx?wsdl"
               docRef="http://www.contoso.com/Counter.asmx"
               xmlns="http://schemas.xmlsoap.org/disco/scl/" />
    <soap address="http://www.contoso.com/Counter.asmx"
        xmlns:q1="http://tempuri.org/"
        binding="q1:CounterSoap"
        xmlns="http://schemas.xmlsoap.org/disco/soap/" />
</discovery>
```

Εικόνα 16: Παράδειγμα αρχείου *.disco*, με τα links των σχετικών εγγράφων *WSDL*.

3.1.3.3 DESCRIPTION

Τα έγγραφα *WSDL* αποτελούν αρχεία με σύνταξη *XML*, που περιέχουν την περιγραφή της σύνταξης των μηνυμάτων που καταλαβαίνουν τα *Web services* (*XML schema* ή *xsd*), ενώ αναγράφουν και το επίπεδο ποιότητας της υπηρεσίας που εξυπηρετεί ως συμβόλαιο μεταξύ παρόχου και πελάτη. Επίσης παρέχουν οδηγίες για τον τρόπο χρήσης των services από τους *clients*. Ένα σκέλος της λειτουργίας των *Web services*, έχει να κάνει με το πρωτόκολλο ανταλλαγής μηνυμάτων μεταξύ server-*client* όπως ορίζεται από την υπηρεσία. Για παράδειγμα, κλήση συγκεκριμένης μεθόδου ενός RPC service, μπορεί να απαιτεί αποστολή μηνύματος από τον *client*, τύπου *SOAP* συγκεκριμένης σύνταξης, με επακόλουθη απάντηση από το server με μήνυμα *SOAP* επίσης ειδικής σύνταξης. Ένα άλλο σενάριο είναι οι μονόδρομες αποστολές μηνυμάτων, όπου ο *client* δεν αναμένεται να λάβει απάντηση από το *Web service*, ούτε κάποιο μήνυμα σφάλματος.

Τα σχήματα των μηνυμάτων *SOAP* μπορούν να περιέχονται στα μηνύματα καθ' αυτά, είτε να βρίσκονται σε εξωτερική τοποθεσία οπότε εισάγονται δυναμικά στο αρχείο *WSDL*. Ακόμη στα τελευταία, προαιρετικά περιέχονται οι τα σημεία προσπέλασης των *Web services*, τα οποία έχουν μορφή που εξαρτάται από το πρωτόκολλο στο οποίο έχει υλοποιηθεί το εκάστοτε service.

3.1.3.4 WIRE FORMAT

Αποτελεί το πρωτόκολλο επικοινωνίας στο οποίο υλοποιείται το *Web service*. Προκειμένου να είναι προσπελάσιμο από όσο το δυνατόν περισσότερα συστήματα, επιλέγονται πρωτόκολλα ευρείας αποδοχής όπως το *http*. Συγκεκριμένα χρησιμοποιούνται οι μέθοδοι *HTTP-GET* και *HTTP-POST*, για την κωδικοποίηση και την μετάδοση παραμέτρων υπό τη μορφή ζευγών key/value. Κάθε τέτοιο μήνυμα

περιέχει επικεφαλίδες (*headers*) με τα ζητούμενα/επιστρεφόμενα από τον *client* (ή server) δεδομένα.

Τα αιτήματα *HTTP-GET* περνούν τις παραμέτρους κωδικοποιημένα σύμφωνα με τη κωδικοποίηση MIME type *application/x-www-form-urlencoded*, οι οποίες επικολλώνται στο τέλος του URL του server που δέχεται το αίτημα. Με τον τρόπο αυτό, οι επιμέρους χαρακτήρες τους δεν αντιμετωπίζονται ως *ειδικοί χαρακτήρες* και περνούν ως απλό κείμενο στο server. Ομοίως και τα αιτήματα *HTTP-POST* κωδικοποιούν τις παραμέτρους με τη διαφορά ότι τα ζεύγη key/value περιέχονται στο σώμα του μηνύματος *http*, κι όχι στο URL.

*REST*FUL *WEB SERVICES*

Η τεχνολογία *REST* αποτελεί ένα αρχιτεκτονικό μοντέλο δημιουργίας *Web services*, που περιστρέφεται γύρω από τους πόρους ενός συστήματος, και ορίζει το πως οι πόροι αυτοί θα προσπελάζονται και μεταφέρονται μέσω του πρωτοκόλλου *HTTP* από ένα σύνολο *client*s με διαφορετική υλοποίηση. Τα τελευταία χρόνια έχει κερδίσει έδαφος έναντι της τεχνολογίας *SOAP*, λαμβάνοντας υπ' όψιν τον αριθμό των *Web services* που τη χρησιμοποιούν, και αυτό οφείλεται κυρίως στην απλότητα της υλοποίησής της. Ένα service που υλοποιεί το πρότυπο *REST*, στην πιο καθαρόαιμη μορφή του, διέπεται από τις εξής τέσσερις αρχές σχεδιασμού:

- Χρησιμοποιεί αποκλειστικά και μονοσήμαντα τις μεθόδους *HTTP*, όπως προβλέπει το εν λόγω πρωτοκόλλου
- Δε διατηρεί πληροφορίες συνεδρίας με τους *client*s (*stateless*)
- Οι πόροι είναι ιεραρχικά δομημένοι
- Υποστηρίζει μηνύματα *XML* ή JSON η και τα δύο

Στις επόμενες παραγράφους περιγράφουμε αναλυτικά τις παραπάνω αρχές σχεδιασμού.

3.1.4.1 ΑΠΟΚΛΕΙΣΤΙΚΗ ΧΡΗΣΗ ΤΩΝ *HTTP* METHODS

Τα *REST*ful *Web services* πρέπει να χρησιμοποιούν τις μεθόδους *HTTP* σύμφωνα με τα *specs* του συγκεκριμένου πρωτοκόλλου (*RFC 2616*). Συγκεκριμένα, υπάρχει μονοσήμαντη αντιστοίχιση μεταξύ των λειτουργιών CRUD (create, read, update, delete) με τις μεθόδους *HTTP*. Ο παρακάτω πίνακας περιέχει τις συσχετίσεις αυτές:

OPERATION	***HTTP* METHOD**
Create a re*Source* on the server	*POST*
Retrieve a re*Source*	*GET*
Change the state of a re*Source*	PUT

Remove a re*Source*	DELETE

Πίνακας 2: Προβλεπόμενες σχέσεις μεταξύ μεθόδων http και λειτουργιών του service, όπως ορίζει το πρωτόκολλο REST.

Έχει ιδιαίτερη σημασία η τήρηση αυτού του κανόνα, καθώς μπορεί να επιφέρει μη επιθυμητές αλλαγές στην κατάσταση των πόρων του web server. Για παράδειγμα, σύμφωνα με το πρότυπο *HTTP*/1.1, προβλέπεται ότι σε ένα αίτημα *GET* το *URI* αναζητά ένα συγκεκριμένο πόρο, ενώ οι παράμετροι στο query string μπορούν να ορίσουν κριτήρια αναζήτησης περισσότερους πόρους. Ωστόσο υπάρχουν περιπτώσεις όπου η μέθοδος αυτή έχει χρησιμοποιηθεί για transactional operations στο server, όπως την εισαγωγή εγγραφών. Παράδειγμα αυτού αποτελεί το επόμενο URI:

```
GET /adduser?name=Robert HTTP/1.1
```

Εικόνα 17: Παράδειγμα μη ορθής χρήσης της μεθόδου GET, η οποία χρησιμοποιείται για την εκτέλεση transactional λειτουργίας στο server.

Εδώ, καλείται μια απομακρυσμένη διαδικασία μέσω της *GET*, η οποία προκαλεί μεταβολές στο server, προσθέτοντας και αποθηκεύοντας ένα νέο χρήστη. Πέρα από την παραβίαση των κανόνων *REST* σε αυτή τη δήλωση, εισάγεται και ένα σημαντικός κίνδυνος, καθώς αυτή η κλήση θα μπορούσε να διενεργηθεί από μια μηχανή αναζήτησης στα πλαίσια του *web caching*, όπου ενημερώνεται και καταχωρεί σε καταλόγους τις υπάρχουσες ιστοσελίδες και εξυπηρετεί στο να επιστρέφει σε σύντομο χρόνο τα αποτελέσματα αναζήτησης.

Η σωστή δήλωση για την τέλεση της συγκεκριμένης λειτουργίας στο server, θα ήταν στο αίτημα να χρησιμοποιηθεί η μέθοδος *POST*, τα ζεύγη name/value των παραμέτρων να αναπαρασταθούν ως στοιχεία *XML*, και το *URI* του αιτήματος να δείχνει στο στοιχείο όπου θα προστεθεί η εγγραφή:

```
POST /users HTTP/1.1
Host: myserver
Content-Type: application/xml
<?xml version="1.0"?>
<user>
  <name>Robert</name>
</user>
```

Εικόνα 18: Εισαγωγή εγγραφής στο server, με χρήση της μεθόδου *POST* με τις παραμέτρους εισαγωγής να περιέχονται στο σώμα του μηνύματος.

Αφ' ότου ολοκληρωθεί αυτή η κλήση, κάποια άλλη εφαρμογή μπορεί να προσπελάσει τον πόρο που καταχωρήθηκε, χρησιμοποιώντας ένα νέο *URI* στο οποίο διαφαίνεται ότι ο πόρος αυτός περιέχεται μέσα στο στοιχείο */users* .

```
GET /users/Robert HTTP/1.1
Host: myserver
Accept: application/xml
```

Εικόνα 19: Το URI αναζήτησης του πόρου */Robert* που βρίσκεται μέσα στον πόρο */users*.

Αυτή είναι η προβλεπομένη χρήση της *GET*, στην οποία πρέπει να διενεργούνται μονάχα λειτουργίες ανάκτησης πληροφοριών (*read operations*) και όχι εισαγωγής ή

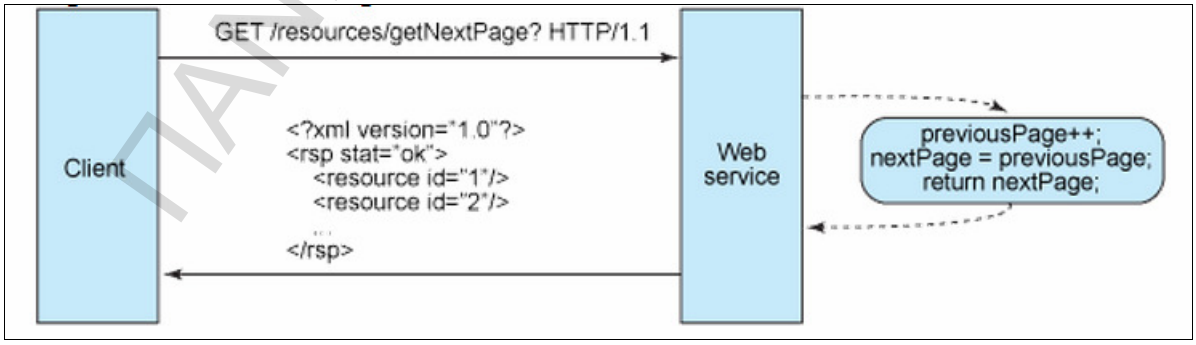
επεξεργασίας. Αυτό εξασφαλίζει ότι δε θα προκύψουν απρόβλεπτες μεταβολές στην κατάσταση των πόρων με ανεπιθύμητες συνέπειες.

3.1.4.2 ΜΗ ΔΙΑΤΗΡΗΣΗ ΠΛΗΡΟΦΟΡΙΩΝ ΣΥΝΕΔΡΙΑΣ ΜΕ ΤΟΥΣ *CLIENTS*

Τα *Web services* πρέπει να ανταποκρίνονται σε περιπτώσεις ιδιαίτερα υψηλής ζήτησης, και ως εκ τούτου φιλοξενούνται σε υποδομές από servers με *load balancing*, εφεδρείες και *proxies*, σχηματίζοντας ειδικές τοπολογίες ώστε να εξυπηρετούν με μικρό χρόνο απόκρισης τα *requests* των εφαρμογών. Η χρήση βοηθητικών servers θέτει περιορισμό όσον αφορά το εάν μπορούν να αποθηκεύουν πληροφορίες συνεδρίας μεταξύ service και εφαρμογής. Αυτό συμβαίνει διότι ανά πάσα στιγμή, διαφορετικός server ενδέχεται να εξυπηρετεί μια δεδομένη συνεδρία, ενώ ο συγχρονισμός του με τους άλλους servers της συνεδρίας δεν είναι είναι πάντοτε εφικτός. Όταν λοιπόν απαιτείται κάποιο κομμάτι πληροφορίας το οποίο δεν έχει επικοινωνηθεί ακόμη στον ενεργό server της συνεδρίας λόγω καθυστέρησης στο συγχρονισμό, αναμένεται να παρουσιασθεί καθυστέρηση είτε διακοπή της παρεχόμενης υπηρεσίας.

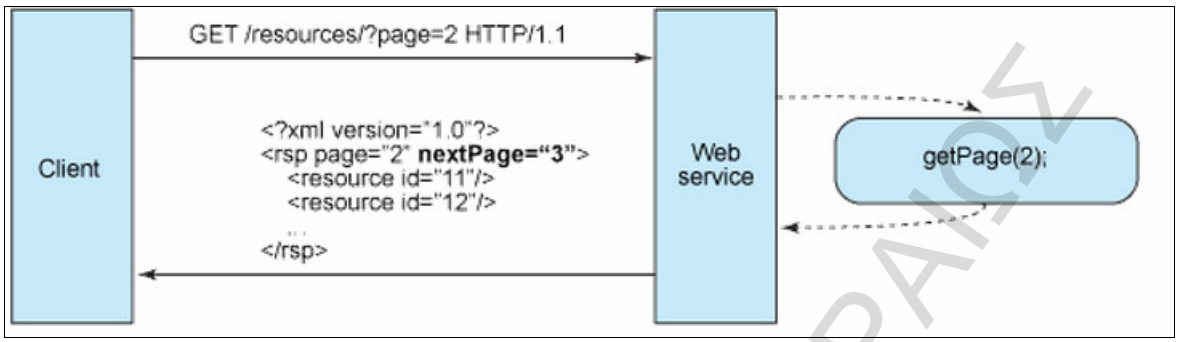
Ως εκ τούτου, τα *requests* θα πρέπει να είναι πλήρη ως προς την πληροφορία που μεταφέρουν υπό την έννοια ότι ο server θα τα διεκπεραιώσει δίχως να χρειαστεί να ανατρέξει σε άλλες πληροφορίες που θα αφορούν την κατάσταση της συγκεκριμένης συνεδρίας με τον *client*. Η πληροφορία των *requests* θα είναι εκφρασμένη ως παράμετροι, δεδομένα και context, και θα περιέχεται στις επικεφαλίδες και το σώμα του αιτήματος. Το παραπάνω μοντέλο βελτιώνει την απόδοση του συστήματος των servers και απλοποιεί την αρχιτεκτονική και υλοποίησή τους καθώς παρακάμπτεται η ανάγκη του συγχρονισμού τους.

Στην επόμενη εικόνα απεικονίζεται ένα *stateful Web service*, από το οποίο η εφαρμογή ζητά την επόμενη σελίδα από ένα σελιδοποιημένο σύνολο αποτελεσμάτων, θεωρώντας ότι το service κρατά την τρέχουσα σελίδα της συνεδρίας σε μια μεταβλητή με το όνομα *previousPage*.



Εικόνα 20: Παράδειγμα *stateful Web service*.

Αντιθέτως, σε ένα *stateless Web service,* η ευθύνη της διατήρησης του state άπτεται στον *client*. Σε ένα *REST*ful service, παρέχεται *interface* που εξωθεί τον *client* να διατηρεί το state, ενώ στα αιτήματα του να συμπεριλαμβάνει ως παραμέτρους τις πληροφορίες κατάστασης της συνεδρίας. Στο παράδειγμά μας, η *stateless* εκδοχή θα ήταν ο *client* να μεταφέρει μέσω του αιτήματος τη ζητούμενη σελίδα αντί την επόμενη, που είναι πιο αόριστη πληροφορία.



Εικόνα 21: Παράδειγμα *stateless Web service*.

Ο server θα αποκρινόταν επιστρέφοντας το link της επόμενης σελίδας, αφήνοντας στο χρήστη την ευθύνη διατήρησης της πληροφορίας αυτής.

3.1.4.3 ΙΕΡΑΡΧΙΚΑ ΔΟΜΗΜΕΝΟΙ ΠΟΡΟΙ

Τα *URI*s που υλοποιεί ένα *Web service* καθορίζουν το βαθμό στον οποίο μπορεί κάποιος με απλή διαίσθηση να κατανοήσει τη λειτουργικότητά του. Τα *URI*s πρέπει να ικανοποιούν τη συνθήκη της διαισθητικότητας προκειμένου να αφομοιώνεται εύκολα η χρήση τους, και επιτυγχάνεται όταν σχεδιάζονται ιεραρχικά, σαν ένας λογικός γράφος, ο οποίος να τηρεί τη λογική σχέση κτήσης μεταξύ προϊστάμενων και υφιστάμενων οντοτήτων. Για παράδειγμα, ένα service με θέματα συζήτησης για *thread*ing θα μπορούσε να έχει την παρακάτω γενική μορφή:

```
http://www.myservice.org/discussion/topics/{topic}
```

Εικόνα 22: Παράδειγμα *URI* το οποίο χάρη στην ιεραρχική του δομή, μπορεί να ερμηνευτεί διαισθητικά.

Ο κεντρικός κόμβος ονομάζεται */dicussion* και περιέχει ένα κόμβο */topics*, που περιέχει άλλες ποικίλες κατηγορίες όπως */technology*, */sports*, */politics* κλπ. Επομένως είναι εύκολο να αιτηθεί κάποιος μια κατηγορία συζήτησης, επικολλώντας απλώς το όνομα της κατηγορίας στη θέση του */{topic}*.

3.1.4.4 ΥΠΟΣΤΗΡΙΞΗ *XML*, JSON

Η αναπαράσταση ενός πόρου ενός *Web service* αντανακλά την κατάσταση του πόρου και τις ιδιότητές του τη στιγμή που ζητήθηκε από τον *client*. Παραδείγματα

αποτελούν η αναπαράσταση μιας εγγραφής βάσης δεδομένων με στοιχείο *XML*, όπου τα tags αντιστοιχούν στα columns και οι τιμές των rows στις τιμές των *elements*. Στην περίπτωση που στον server υπάρχει μοντέλο οντοτήτων, πρέπει να διατηρούνται οι σχέσεις μεταξύ των οντοτήτων κατά την αναπαράστασή και μεταφορά τους στην εφαρμογή του *client*. Στο παράδειγμα με τα θέματα συζήτησης για *thread*ing, μια αποδεκτή αναπαράσταση θα ήταν η ακόλουθη:

```
<?xml version="1.0"?>
<discussion date="{date}" topic="{topic}">
  <comment>{comment}</comment>
  <replies>
    <reply from="joe@mail.com" href="/discussion/topics/{topic}/joe"/>
    <reply from="bob@mail.com" href="/discussion/topics/{topic}/bob"/>
  </replies>
</discussion>
```

Εικόνα 23: Παράδειγμα αναπαράστασης του μοντέλου οντοτήτων ενός service για discussion forum.

Το κεντρικό *element /discussion* περιέχει *attributes* που περιγράφουν το θέμα και την ημερομηνία δημιουργίας του, ενώ περιέχει και *elements* τύπου */reply* που ενσωματώνουν links με τις απαντήσεις που δόθηκαν σε αυτό το θέμα. Σε αυτή τη μορφή είναι εφικτή η διαισθητική ερμηνεία του μηνύματος, και ως εκ τούτου γίνεται πιο ελκυστική η χρήση του service για τους developers των εφαρμογών.

Επίσης, θα πρέπει τα*Web services* να υλοποιούνται σε διαφορετικές εκδόσεις όσον αφορά τον υποστηριζόμενο τύπο μηνυμάτων, ώστε να είναι συμβατά με μεγαλύτερο εύρος εφαρμογών. Γι' αυτό πρέπει να υλοποιούνται σε διαφορετικά formats, όπου στο καθένα η επικεφαλίδα *HTTP Access header* να λαμβάνει την ανάλογη τιμή MIME type. Τα MIME types που μπορούν να χρησιμοποιούν τα *REST*ful *Web services* είναι τα παρακάτω:

MIME-Type	Content-Type
JSON	application/json
XML	application/xml
XHTML	application/xhtml+xml

Πίνακας 3: Αποδεκτοί τύποι μηνυμάτων με βάση το MIME Type των *RESTful* Web services.

Κατ' αυτό τον τρόπο, ένα *Web service* μπορεί να χρησιμοποιηθεί από διαφορετικούς *client*s σε διαφορετικές πλατφόρμες και συσκευές.

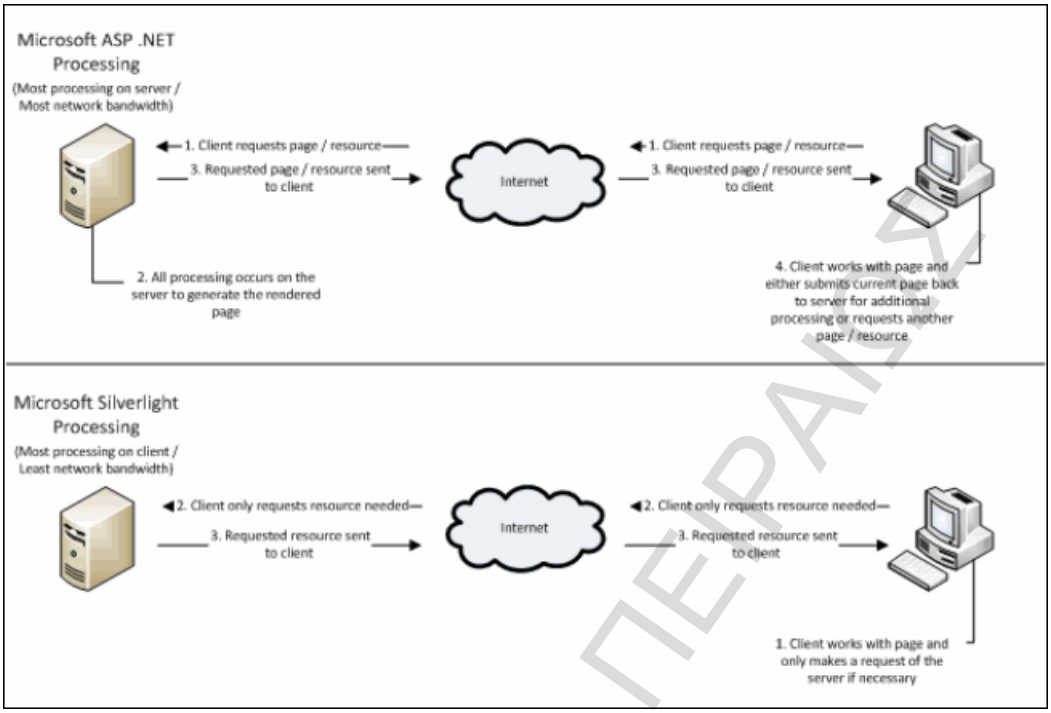
3.2 *SILVERLIGHT*

Η τεχνολογία *Silverlight*, επιτρέπει τη δημιουργία εφαρμογών web με πληθωρικό *interface* και λειτουργικότητα που μπορεί να ανταγωνιστεί επάξια την αντίστοιχη των desktop εφαρμογών, καθιστώντας την κατάλληλη για την ανάπτυξη επιχειρησιακών εφαρμογών, εφαρμογών *multimedia* αλλά και *mobile*. Αποτελεί ένα plug-in το οποίο μπορεί να εγκατασταθεί σε όλους τους δημοφιλείς *browser*s (*Google Chrome*, *Mozilla Firefox* κ.α.), ανεξαρτήτως λειτουργικού συστήματος, αρκεί ο χρήστης να επικυρώσει την εγκατάστασή του η οποία λαμβάνει χώρα μια φορά. Έκτοτε το plug-in αυτό, δεν καταναλώνει πόρους συστήματος εφ' όσον ο χρήστης πλοηγείται σε σελίδες οι οποίες δεν έχουν αναπτυχθεί σε *Silverlight*. Το plug-in είναι αυτόνομο, δεν απαιτεί προεγκατάσταση άλλου λογισμικού καθώς περιέχει μια περιορισμένη πλην όμως, πλήρη έκδοση του *.NET framework*.

ΛΕΙΤΟΥΡΓΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Μια εφαρμογή *Silverlight* εκτελείται στον *browser* του χρήστη ενώ φιλοξενείται σε κανονικούς web servers, όχι απαραίτητα σε *IIS* (όπως οι εφαρμογές *asp.NET*), ούτε απαιτεί την εγκατάσταση της *asp.NET* στο server. Η πρόσβαση σε βάσεις δεδομένων είτε σε κλάσεις και μεταβλητές του server, απαιτεί τη διαμεσολάβηση λογισμικού, πιθανότατα τύπου *Web service*.

Υπάρχουν διαφορές ως προς τον τρόπο εκτέλεσης μιας εφαρμογής *Silverlight* σε σύγκριση με τις web εφαρμογές άλλων τεχνολογιών (όπως *asp* είτε *PHP*). Στις δεύτερες, ο web server εκτελεί γηγενώς τον *asp* (ή *PHP*) κώδικα παράγοντας κώδικα *HTML*, τον οποίο αποστέλλει στον *client* που τον εκτελεί μέσω του *browser*. Εν, αντιθέσει, σε μια *Silverlight* εφαρμογή, κατόπιν αιτήματος του *client*, ο server θα αποστείλει την εφαρμογή στον *client*, η οποία θα αρχικοποιηθεί μέσω του προεγκατεστημένου *Silverlight* plug-in στον *browser*, και έκτοτε η επικοινωνία με το server θα διενεργείται μονάχα όταν απαιτούνται περαιτέρω δεδομένα από τον τελευταίο. Στο παρακάτω διάγραμμα αναπαριστώνται οι διαφορετικές προσεγγίσεις των αναφερόμενων τεχνολογιών ανάπτυξης web applications.



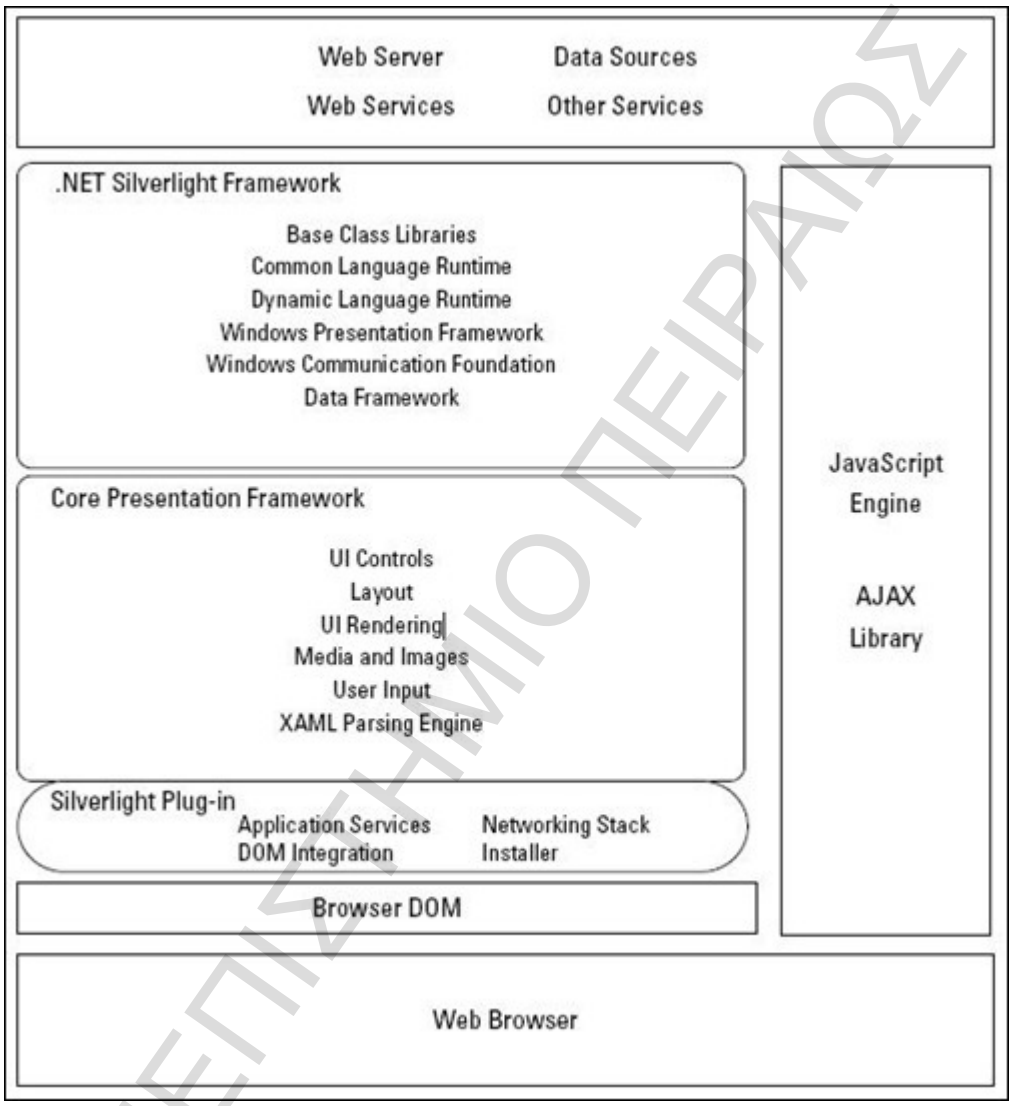
Εικόνα 24: Η αρχιτεκτονική client-server που υλοποιείται με χρήση της *asp.NET* (επάνω) του *Silverlight* (κάτω).

Πιο αναλυτικά, το αρχείο που αποστέλλεται από το server στον *client* είναι συμπιεσμένο τύπου *.xap*. Αυτό περιέχει συμπιεσμένο τον κώδικα της εφαρμογής μαζί με ένα *XML* manifest αρχείο που περιγράφει τις λειτουργικές προδιαγραφές της εφαρμογής για λογαριασμό του *browser*.

Το *Silverlight* plug-in περιέχει μια περιορισμένη-ελαφριά έκδοση του *.NET Framework* δίνοντας τη δυνατότητα εκτέλεσης *managed code* (C#, *Visual Basic*) είτε *compliled code* (*C/C++*). Ακόμη διαθέτει την τεχνολογία *Dynamic Language Runtime*, που παρέχει τη δυνατότητα συγγραφής κώδικα σε μη παραδοσιακές γλώσσες όπως *IronRuby* ή *IronPython*. Για κάθε σελίδα υπάρχει ένα αρχείο *code-behind* που μπορεί να αξιοποιηθεί για την υλοποίηση των *handlers* που είναι συσχετισμένοι με τα *events* που μεταδίδονται από τη σελίδα (αντίστοιχα όπως με τις εφαρμογές *asp.NET*).

ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Το *Silverlight* αποτελεί συνδυασμό των τεχνολογιών *.NET framework*, *vector animations*, πολυμέσων, *Javascirpt* και *Ajax*. Στο παρακάτω σχήμα αναπαρίσταται η αρχιτεκτονική του *Silverlight*.



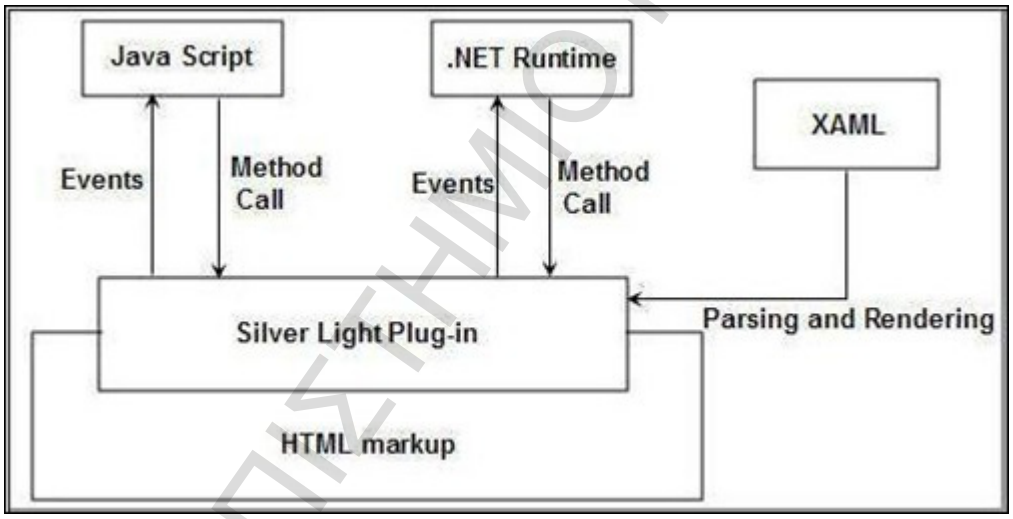
Εικόνα 25: Αρχιτεκτονική του Silverlight plugin.

Περιγράφουμε τα κυριότερα μέρη που συνθέτουν το *Silverlight*.

- ***.NET framework* Components**: το *Silverlight* ενσωματώνει λειτουργίες του *.NET*, με βασικότερη την *Windows Presentation Foundation* (*WPF*). Το *WPF* αποτελεί την τελευταία γενιά εργαλείων δημιουργίας πληθωρικών *UI*'s. Για Όλα τα *user controls* (*Checkbox*, *Button*, *Textbox*) όπως και η *XAML* προέρχονται από τη *WPF*. Δανείζεται επίσης την *WCF* για πρόσβαση σε δεδομένα, το *Common Language Runtime* (*CLR*) για *memory management* και *garbage collection*, ενώ οι *Base Class Libraries* περιέχουν

τις στοιχειώδεις συναρτήσεις για επεξεργασία των *strings*, γενικούς αλγόριθμους, *containers* κ.α.

- ***Presentation Core***: Αναπαράγει τα *2d animations*, εικόνες, πολυμέσα και διαχειρίζεται το *input* που προέρχεται από τα *user controls* (πληκτρολόγιο κλπ). Δεν περιέχει τις κλάσεις των *user controls*, αλλά τις κλάσεις που πάνω στις οποίες οικοδομούνται τα τελευταία.
- ***Javascript & Ajax***: Αλληλεπιδρά με τις τεχνολογίες αυτές ενώ ταυτόχρονα δανείζεται λειτουργικές τους δυνατότητες.
- ***Hosting***: Εμπεριέχει τη λειτουργικότητα φιλοξενίας *Silverlight* εφαρμογών, επεκτείνοντας προς αυτή την κατεύθυνση τη συνολική λειτουργικότητα του *browser*. Μεριμνεί για τη δημιουργία του *UI* έχοντας ως *input* τον κώδικα *XAML* και *output* τον κώδικα *HTML*. Όταν ο χρήστης αλληλεπιδρά με την εφαρμογή, μεταδίδει *events* στο *.NET framework* (ή και τη *Javascript*), το οποίο κάνει τις ανάλογες κλήσεις στο runtime, που εκτελεί τις επιθυμητές διεργασίες ενημερώνοντας την οθόνη μέσω του κώδικα *HTML*. Το σχήμα αναπαριστά την αναφερόμενη διαδικασία.



Εικόνα 26: Λειτουργία Hosting του Silverlight plug-in.

3.3 *XAML* (EXTENSIBLE APPLICATION MARKUP LANGUAGE)

Η *XAML* αποτελεί υλοποίηση της *XML* για τον σχεδιασμό του *UI* των εφαρμογών. Ενώ στα κλασσικά προγραμματιστικά περιβάλλοντα με αντικειμένοστρεφείς γλώσσες (Java, *C#*), κοινός κώδικας δημιουργεί το *UI* και του προσδίδει λειτουργικότητα, στο περιβάλλον του *Silverlight* και χάρη στην *XAML*, υπάρχει διαχωρισμός μεταξύ του *UI* και της λειτουργικότητας κατά τρόπο που μπορούν να αναπτυχθούν ανεξάρτητα. Η *XAML* προσφέρει ένα *δηλωτικό* σύστημα των *UI* components αντί να χρησιμοποιεί *runtime object* s, που την καθιστά εύχρηστη και κατάλληλη για την ταχεία δημιουργία των *UI*'s.

Κάθε *element* της *XAML* δημιουργεί από ένα *object* με κλάση η οποία ορίζεται στο *PresentationFramework assembly*. Το αρχείο *XAML* στο οποίο ορίζεται το *element* αυτό, συμπληρώνεται από ένα αρχείο *code-behind*, το οποίο διαθέτει ορισμούς κλάσεως τύπου *partial* και στις οποίες υλοποιείται η λειτουργικότητα του *UI control*.

ΣΥΝΤΑΞΗ-ΛΕΙΤΟΥΡΓΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Όπως οι κλάσεις έχουν *properties* που καθορίζουν τη λειτουργικότητα τους, έτσι και τα *objects* που παράγει η *XAML* έχουν *properties*, που αντιστοιχούν στα *properties* και *attributes* των αντίστοιχων *elements*, και αναπαριστούν τετριμμένους τύπους (αποθηκεύονται σε *string*), είτε σύνθετους τύπους αντίστοιχα. Ένα τυπικό παράδειγμα *XAML element* που περιέχει την αναφερόμενη σύνταξη είναι το παρακάτω:

```
<Button Name="myButton" Content="Click Me!">
    <Button.Foreground>
        <SolidColorBrush Color="Blue" />
    </Button.Foreground>
</Button>
```

Εικόνα 27: Παράδειγμα στοιχείου XAML, τύπου *Button*.

Το στοιχείο *Button*, έχει όνομα "*myButton*" που εξυπηρετεί ως *reference* του συγκεκριμένου στοιχείου στον *runtime* κώδικα για την περαιτέρω διαχείρισή του. Ακόμη έχει ένα *property Foreground* τύπου *Brush* (αντί *String*), και ως εκ τούτου μπορεί να οριστθεί μονάχα ως *property element*.

Επίσης η *XAML* υποστηρίζει συλλογές αντικειμένων, οι οποίες δηλώνονται ως αντικείμενο *Children* με τύπο *UIElementCollection* όπως στο παρακάτω παράδειγμα.

```
<StackPanel>
    <StackPanel.Children>
        <Button Content="Button 1" />
        <Button Content="Button 2" />
    </StackPanel.Children>
</StackPanel>
```

Εικόνα 28: Παράδειγμα συλλογής αντικειμένων *XAML*.

Πολύ συχνά, οι δηλώσεις των *XAML elements* είναι μακροτενείς και πολύπλοκες, σε σημείο που αν δηλώνονται *inline* εντός άλλων *elements*, να χαλάνε τη συνοχή του κώδικα και να τον κάνουν δυσανάγνωστο. Γι αυτό το λόγο υπάρχουν τα *markup extensions* τα οποία είναι *strings* με ειδικό format που μεταχειρίζονται με ανάλογο τρόπο από τον *XAML parser*, και παραπέμπουν σε άλλες δομές *XAML*. Ένα τυπικό παράδειγμα είναι το παρακάτω:

```
<Button Style="{StaticResource myButtonStyle}"/>
```

Εικόνα 29: Παράδειγμα ενός *markup extension* για την απόδοση στυλ σε ένα *Button*.

Εδώ δηλώνεται ένα *Button* το οποίο έχει Style που ορίζεται από το *StaticReSource myButtonStyle*. Τα *staticReSources* είναι πόροι που καλούνται κατά το φόρτωμα της εφαρμογής και δηλώνονται σε άλλο σημείο του κώδικα *XAML* (ενδεχομένως και άλλο αρχείο).

Σε ορισμένες περιπτώσεις τα *UI control*s υπολείπονται ορισμένων *properties* που είναι αναγκαία για την αξιοποίησή τους στο *UI* κατά τον επιθυμητό τρόπο. Γι αυτό υπάρχουν τα *attached properties* τα οποία επεκτείνουν τις δυνατότητες επεξεργασίας των *UI control*s με βάση το πλαίσιο στο οποίο χρησιμοποιούνται. Παραθέτουμε ένα παράδειγμα:

```
<DockPanel>
    <Button DockPanel.Dock="Left"/>
    <Button DockPanel.Dock="Right" />
</DockPanel>
```

Εικόνα 30: Το στοιχείο DockPanel.*Dock* αποτελεί *attached property* του στοιχείου *DockPanel*.

Ενώ τα *Button*s δεν έχουν *properties* για την τοποθέτησή τους στο *UI*, τους αποδίδεται το *DockPanel.Dock* attached *property*, ώστε να ενημερώσουν το πατρικό *element* (*DockPanel*) για τη θέση που θα έχουν σε αυτό.

Όπως συμβαίνει με την *XML*, οι δηλώσεις των *namespaces* δηλώνονται ως *attributes* στο *root element* του δέντρου. Χρησιμεύουν στο να κατηγοριοποιούν τις εισαγόμενες κλάσεις με βάση τα *assemblies* στα οποία αυτές ανήκουν, με τα οποία έχει διασυνδεθεί η εφαρμογή. Παραθέτουμε ένα παράδειγμα:

```
<Window x:Class="XAMLTest.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Main Window">

</Window>
```

Εικόνα 31: Ορισμός των namespaces αρχείου XAML.

Στο παράδειγμα, δηλώνονται δύο *namespaces*. Το πρώτο δεν έχει *prefix* και αποτελεί το *default namespace*. Σε αυτό ανήκει το *Window element*. Μπορούν να δηλωθούν και *namespaces* από το χρήστη, με τη γενική μορφή:

```
xmlns:myCode="clr-namespace:MyNamespace;assembly=MyAssembly"
```

Εικόνα 32: Custom namespace ορισμένο από το χρήστη.

Στο παράδειγμά αυτό, *myCode* είναι το *prefix* των εισαγόμενων κλάσεων, το *CLR-namespace* δηλώνει στον *XAML parser* ότι θα εισαχθεί ένα *.NET namespace*, ενώ το *MyAssembly* είναι το όνομα του *assembly* προς εισαγωγή.

DATA-BINDING

Η τεχνική αυτή επιτρέπει την αυτόματη ενημέρωση των *control*s του *interface*, με απλές δηλώσεις αντικειμένων σε *XAML*, αντί του κλασσικού προγραμματισμού της λειτουργικότητας των *control*s του *UI*. Για να το επιτύχει αυτό, το *Silverlight* εισάγει δύο νέες κλάσεις, τη *System.Windows.DependencyObject* και τη *System.Windows.DependencyProperties* οι οποίες παρέχουν την επιπλέον λειτουργικότητα που απαιτείται από τα *objects* που συμμετέχουν σε αυτό το νέο σύστημα.

3.3.2.1 DEPENDENCYOBJECT , DEPENDENCYPROPERTY

Τα τετριμμένα *object* s (ή *Plain Old CLR Object* s), αποτελούν υποκλάση της *System.Object* η οποία δεν υποστηρίζει *data-binding*. Γι' αυτό εισήχθη η κλάση *DependencyObject* η οποία ενσωματώνει τις απαιτούμενες λειτουργίες. Ένα *DependencyObject* είναι αλληλένδετο με ένα η περισσότερα *DependencyProperty* που αποτελούν *properties* που δηλώνονται με συγκεκριμένο τρόπο και έχουν επιπλέον ιδιότητες. Αυτές οι ιδιότητες είναι που κάνουν εφικτή την υλοποίηση των *attached properties* της Παρ. 3.3.1.

Τα *DependencyObject* αποτελούν υποκλάση της *System.Threading.DispatcherObject* η οποία τα συσχετίζει με ένα *Dispatcher object* το οποίο διαχειρίζεται τη στοίβα εργασιών ενός *thread*. Ένα *DispatcherObject* δε μπορεί να προσπελασθεί από άλλο *thread* πέρα από αυτό του *Dispatcher* του, διασφαλίζοντας έτσι την αποκλειστική πρόσβαση από ένα μονάχα *thread*, αποφεύγοντας έτσι ενδεχόμενα *race conditions*. Στο *Silverlight*, την αλληλεπίδραση με το *UI* την αναλαμβάνει ένα *thread*, ενώ το *rendering* του *interface* την αναλαμβάνει άλλο *thread*. Δεδομένου ότι τα *UI control*s υλοποιούν την *DependencyObject* (και άρα την *DispatcherObject*) , δε μπορούν να προσπελασθούν από άλλο *thread* πέραν του *UI*.

3.3.2.2 ΔΗΛΩΣΕΙΣ DATA-BINDING

Ο συνήθης τρόπος δήλωσης *data-binding* είναι με *markup exntensions*, όπως φαίνεται στην επόμενη εικόνα:

```
<Button Content="{Binding Source=myBindingSource, Path=myContent}" />
```

Εικόνα 33: Ορισμός *data-binding* μεταξύ της ιδιότητας *Content* ενός *Button* και ενός custom *CLR Object*.

Στο παράδειγμα αυτό, με απλή δήλωση *XAML* διαβάζεται η τιμή *myContent* από την πηγή *myBindingSource*, και αποδίδεται στο *DependencyProperty Content* του *DependencyObject Button*. Με τη δήλωση αυτή το framework διενεργεί παράλληλα και άλλες παραμετροποιήσεις σε αυτή τη συσχέτιση, που αν διενεργούνταν με κλασσικό προγραμματισμό, θα απαιτείτο σημαντικά περισσότερος κόπος από τον developer ελαττώνοντας έτσι την παραγωγικότητά του. Ακόμη, χάρη στο σύστημα αυτό, αποφεύγεται η απόδοση ονομάτων στα *elements* για τη μετέπειτα προσπέλαση σε αυτά από τον κώδικα παραμετροποίησης τους.

Μια ιδιαίτερα σημαντική παράμετρος μιας συσχέτισης *data-binding*, είναι και η κληρονομικότητα των *DependencyProperties.* Δεδομένου ότι τα *DependencyObject s* δομούνται ιεραρχικά και εφ' όσον ένα *DependencyProperty* παραμετροποιηθεί ως "κληρονομήσιμο", αυτομάτως θα είναι ορισμένο και στα περιεχόμενα *elements* του πατρικού *DependencyObject* .

Ο μηχανισμός *data-binding* επιτρέπει την απόδοση τιμών στα οπτικά χαρακτηριστικά των *UI elements*. Παραθέτουμε ένα παράδειγμα:

```
<Style x:Key="myButtonStyle">
    <Setter Property="Control.Background" Value="Blue" />
</Style>
...
<Button Style="{StaticResource myButtonStyle}" />
```

Εικόνα 34: Ορισμός στυλ, και απόδοσή του σε στοιχείο *Button* μέσω *data-binding*.

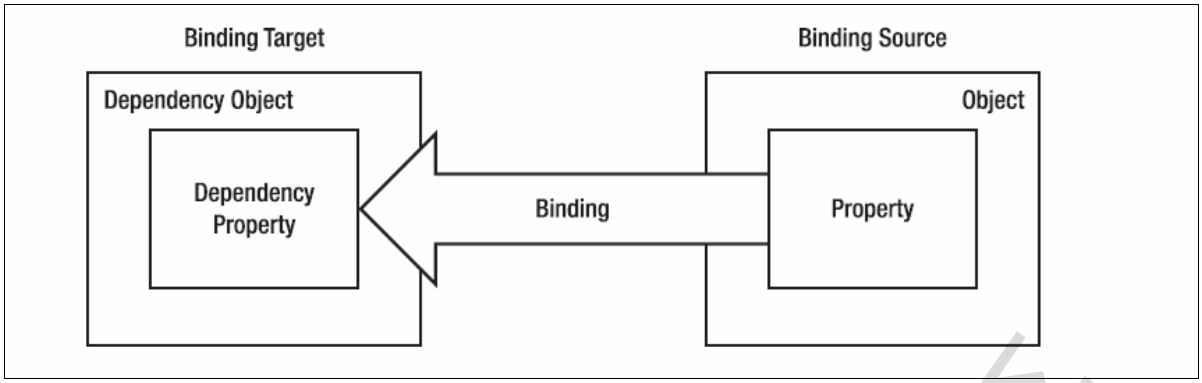
Πρώτα δηλώνεται ένα στυλ *myButtonStyle* ως *StaticReSource*. Ένα στοιχείο *<Style>* αποτελεί μια συλλογή από *<Setter>* (ζεύγος *property/value* που αποδίδονται σε ένα Target *control*).Κατόπιν, το συγκεκριμένο *Style* αποδίδεται σε ένα *Button* με *extension markup*.

3.3.2.3 *DATA-BINDING* MODES

Υπάρχουν τέσσερα διαφορετικά modes του *data-binding* που καθορίζουν την κατεύθυνση μετάδοσης της πληροφορίας μεταξύ *Target-Source*.

3.3.2.3.1 ONE-WAY

Χρησιμοποιείται όταν μεταδίδονται δεδομένα προς το χρήστη κι όχι αντίστροφα. Αυτό είναι το *default mode* όταν το *Source* δεν έχει *setter function* (κι επομένως ορίζεται προγραμματιστικά ως *read-only*). Το παρακάτω σχήμα απεικονίζει αυτή την λειτουργία.



Εικόνα 35: *OneWay mode* σε συσχέτιση *data-binding*.

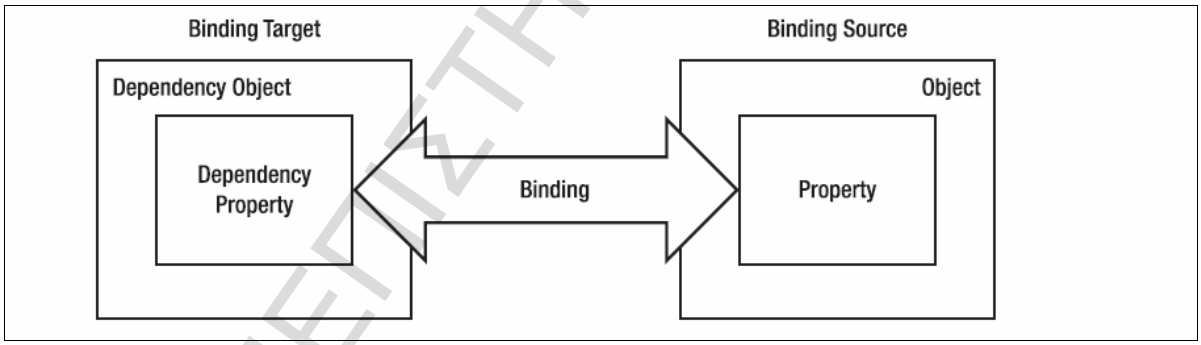
Ενώ ένα παράδειγμα ανάλογης δήλωσης είναι η επόμενη:

```
<Button Content="{Binding Path=myProperty, BindingMode=OneWay}" />
```

Εικόνα 36: Παραμετροποίηση της σχέσης *data-binding* ως *OneWay*.

3.3.2.3.2 TWO-WAY

Σε αυτό τον τύπο έχουμε πλήρες συγχρονισμό μεταξύ *Source-Target*. Αλλαγές που λαμβάνουν χώρα στη μια μεριά, λαμβάνουν αντίστοιχα χώρα και στην άλλη. Είναι ο *default* τύπος *data-binding* όταν χρησιμοποιούνται *editable UI control*s όπως *Textbox* και *Checkbox*, εφ' όσον η *property* του *Source* που συμμετέχει στη συσχέτιση διαθέτει *setter function*.



Εικόνα 37: TwoWay mode σε συσχέτιση *data-binding*.

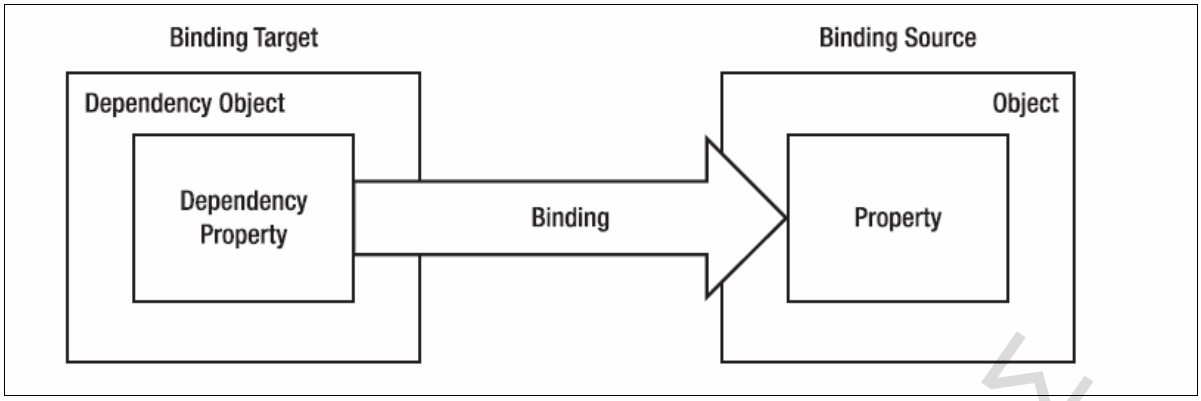
Ένα παράδειγμα δήλωσης *TwoWay* είναι η παρακάτω:

```
<Button Content="{Binding Path=myProperty, BindingMode=TwoWay}" />
```

Εικόνα 38: Παραμετροποίηση της σχέσης *data-binding* ως *TwoWay*.

3.3.2.3.3 ONE-WAY-TO-*SOURCE*

Αποτελεί την ανεστραμμένη εκδοχή της περίπτωσης *OneWay*. Το *UI control* (*Target*) ενημερώνει το *Source object* , παρ' ότι το τελευταίο δεν αποτελεί *DependencyProperty*. Το σχήμα απεικονίζει αυτή τη σχέση:



Εικόνα 39: OneWayTo*Source* mode σε συσχέτιση *data-binding*.

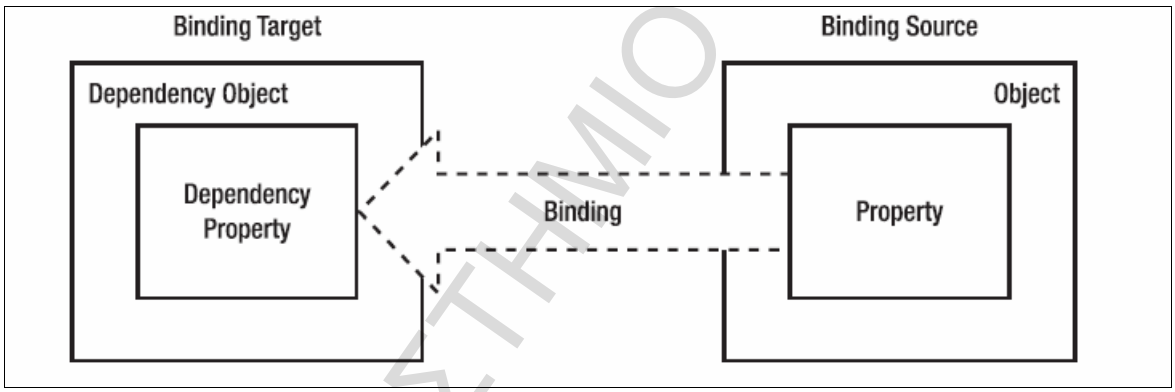
Η επόμενη δήλωση υλοποιεί ένα *OneWayToSource data-binding*:

```
<Button Content="{Binding Path=myProperty, BindingMode=OneWayToSource}" />
```

Εικόνα 40: Παραμετροποίηση της σχέσης *data-binding* ως *OneWaytoSource*.

3.3.2.3.4 ONE-TIME

Έχει τα χαρακτηριστικά του *OneWay data-binding* με τη διαφορά ότι το *UI control* ενημερώνεται μονάχα μια φορά όταν φορτώνει το συγκεκριμένο *View*, είτε όποτε το *DataContext* του αλλάζει. Απεικονίζουμε το συγκεκριμένο mode σε σχήμα:



Εικόνα 41: OneTime mode σε συσχέτιση *data-binding*.

ενώ παράδειγμα αντίστοιχης δήλωσης είναι η επόμενη:

```
<Button Content="{Binding Path=myProperty, BindingMode=OneTime}" />
```

Εικόνα 42: Παραμετροποίηση της σχέσης *data-binding* ως *OneTime*.

3.3.2.4 DATA CONTEXT

Τα *UI control*s που παρέχονται από το *namespace System.Windows.Controls*, κληρονομούν την κλάση *FrameworkElement*, η οποία έχει την *property DataContext* τύπου *System.Object*, και η οποία αποτελεί τη *default* πηγή της συσχέτισης *data-binding* για τα *UI control*s. Παραθέτουμε ένα

παράδειγμα:

```
<Window x:Class="MvvmWpfApp.Window1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:viewModel="clr-namespace:MvvmWpfApp.ViewModel;assembly=MvvmWpfApp.ViewModel"
        Title="Window1" Height="300" Width="300">
    <Window.Resources>
        <viewModel:SampleViewModel x:Key="sampleViewModel1" />
        <viewModel:SampleViewModel x:Key="sampleViewModel2" />
    </Window.Resources>

    <Grid DataContext="{StaticResource sampleViewModel1}">

        <StackPanel Orientation="Vertical">

            <StackPanel Orientation="Horizontal">
                <Button Name="button1" />
                <Button Name="button2" />
            </StackPanel>

            <StackPanel Orientation="Horizontal" DataContext="{StaticResource
 sampleViewModel2}">
                <Button Name="button3" />
                <Button Name="button4" />
            </StackPanel>

        </StackPanel>
    </Grid>
</Window>
```

Εικόνα 43: Η ιδιότητα *DataContext* κληρονομείται από τα περιεχόμενα element ενός *UI control*.

Στο παράδειγμα, τα *Button1* και *Button2* κληρονομούν το *DataContext* από το πατρικό *StackPanel* που με τη σειρά του το κληρονομεί από το πατρικό *element Grid*, ενώ τα *Button3* και *Button4* το επαναθέτουν δίνοντάς του τιμή *sampleViewModel2*.

3.3.2.5 *DATA-BINDING* PARAMETERS

Μια συσχέτιση *data-binding* μπορεί να παραμετροποιηθεί κατά πολλούς τρόπους, καθιστώντας έτσι ένα χρήσιμο εργαλείο για την ενημέρωση του *UI* σε ποικίλες περιπτώσεις και υπό ιδιαίτερες συνθήκες. Αναφέρουμε τις πιο συνήθεις παραμετροποιήσεις που έλαβαν χώρα στην εφαρμογή μας.

ElementName: Επιτρέπει τη δημιουργία συσχέτισης *data-binding* μεταξύ δύο *UI control*s (όπου ως πηγή ορίζεται το *DependencyProperty* άλλου *UI control*, και ως στόχος το *UI control* που δηλώνει τη συσχέτιση). Παράδειγμα:

```
<TextBox Text="Hello!" Name="textBox1" />
<TextBox Text="{Binding ElementName=textBox1, Path=Text}" />
```

Εικόνα 44: Δήλωση συσχέτισης *data-binding* με παραμετροποίηση του *ElementName*.

Το δεύτερο *Textbox* έχει για τιμή του property *Text*, την τιμή του αντίστοιχου *property Text* του πρώτου, καθώς αποτελεί την πηγή της μεταξύ των συσχέτισης.

FallbackValue: Υπάρχουν περιπτώσεις όπου το *data-binding* μπορεί να αποτύχει (ενδεχόμενη διακοπή στην υπηρεσία που παρέχει τα δεδομένα, *sensor failure*, *Web service* κλπ). Σε αυτή την περίπτωση το *UI control* θα λάβει την *default* τιμή του τύπου του (π.χ. για *string* είναι το κενό ''). Η επιλογή αυτή επιτρέπει να λάβει κάποια άλλη τιμή, η οποία θα ενημερώνει το χρήστη σχετικά, κάνοντας έτσι το *interface* πιο φιλικό. Παράδειγμα:

```
<TextBox Text="{Binding Path=myTextProperty, FallbackValue='No value found'}" />
```

Εικόνα 45: Ορισμός του *FallbackValue* που ενημερώνει το χρήστη για διακοπές στην υπηρεσία.

Στο *Textbox* του παραδείγματος, θα εμφανισθεί η τιμή "*No value found*", σε περίπτωση αποτυχίας του *data-binding*.

IsAsync: Αποδίδοντας την τιμή *true*, ανακτούμε την τιμή του *Source* ασύγχρονα δίχως να μπλοκάρει το *thread* που αναλαμβάνει εκτέλεση των λειτουργιών του *UI*. Στο μεταξύ εμφανίζεται η τιμή *FallbackValue*, γι' αυτό και είναι αναγκαία η αξιοποίηση της συγκεκριμένης λειτουργίας.

```
<TextBox Text="{Binding Path=myTextProperty, IsAsync=true, FallbackValue='Please wait…'}" />
```

Εικόνα 46: Ορισμός της συσχέτισης *data-binding* ως ασύγχρονη, ώστε να μην παγώνει το *UI* σε χρονοβόρες διεργασίες του.

Path: Καθορίζει το μονοπάτι του αντικειμένου, με αφετηρία το object *DataContext* (είτε το *Source* εφ' όσον έχει ορισθεί), με το οποίο θα συγχρονιστεί το *UI control* της συσχέτισης. Για παράδειγμα:

```
<TextBox Text="{Binding Path=myObject.MySubObject.MyBoundProperty}" />
```

Εικόνα 47: Με το *property Path* συγκεκριμενοποιείται το αντικείμενο με το οποίο θα συγχρονισθεί το *UI control* της συσχέτισης.

Το *element Textbox*, θα έχει την τιμή *myObject .MySumObject .MyBoundProperty*, όπου το *myObject* θα περιέχεται στο *DataContext* που κληρονομεί το συγκεκριμένο *Textbox*. Μπορεί επίσης να δείχνει και στοιχείο κάποιας συλλογής όπως φαίνεται στο επόμενο παράδειγμα:

```
<TextBox Text="{Binding Path=myIndexedProperty[2]}" />
```

Εικόνα 48: Προσδιορισμός στοιχείου συλλογής μέσω της παραμέτρου *Path*.

Το τρίτο στοιχείο της συλλογής ενημερώνει την τιμή του Text *property* του *Textbox*.

Source: Η παράμετρος *DataContext* δίνει το *default Source* της σχέσης *data-binding*, ωστόσο κάποιες φορές χρειάζεται να επαναορίζεται το *Source*, ανάλογα τις εκάστοτε ανάγκες των *UI controls*. Αυτό γίνεται αποδίδοντας τιμή στην παράμετρο *Source*.

```
<Window xmlns:viewModel="clr-namespace:BusinessModel;assembly=MyBusinessModel"
…>
    <Window.Resources>
       <viewModel:BusinessObject x:Key="myObject" />
    </Window.Resources>
    <TextBox Text="{Binding Path=MyTextProperty, Source={StaticResource myObject}}" />
</Window>
```

Εικόνα 49: Ορισμός της πηγής της συσχέτισης *data-binding*.

RelativeSource: Είναι εφικτός ο σχετικός προσδιορισμός ενός *Source object* ως προς το Target *object* στο γράφο των *UI elements*, και όχι με απόλυτα μονοπάτια όπως στα έως τώρα παραδείγματα. Η πιο κοινή χρήση της παραμέτρου αυτής είναι κατά τη διασύνδεση δύο διαφορετικών παραμέτρων του ιδίου *UI control*.

```
<TextBox Text="{Binding Path=Foreground, RelativeSource={x:Static RelativeSource.Self}}" />
```

Εικόνα 50: Ορισμός πηγής συσχέτισης ως προς το στοιχείο *TextBox* που δηλώνει τη συσχέτιση.

Η *property Text* συνδέεται με την *property RelativeSource* στο ίδιο *Textbox element*.

TargetNullValue: Όταν η πηγή της συσχέτισης δίνει *NULL*, η παράμετρος *TargetNullValue* παρέχει την τιμή που θα αναγράφεται στο *UI control*, ώστε να ενημερώνεται σχετικά ο χρήστης, αντί να βλέπει τιμές που δεν τον ενημερώνουν πραγματικά για την κατάσταση τηςυπηρεσίας.

```
<TextBox Text="{Binding Path=FirstName, TargetNullValue='Please enter your first name…'}" />
```

Εικόνα 51: Ορισμός του *FallbackValue* που ενημερώνει το χρήστη για την κατάσταση της υπηρεσίας.

Στο *Textbox* του παραδείγματος, εμφανίζεται η επιγραφή "*Please enter your first name*", σε περίπτωση που ο χρήστης δεν έχει καταχωρήσει κάποιο όνομα (το οποίο αυτόματα θα ενημέρωνε την πηγή της συσχέτισης *data-binding*).

UpdateSourceTrigger: Προσδιορίζει το *event* το οποίο θα προκαλέσει την ενημέρωση του *Source* του *data-binding*. Υπάρχουν τρείς δυνατές τιμές:

i. ***PropertyChanged***: Ενημερώνει το *Source* εφ' όσον αλλάξει η τιμή του *binding Target.*
ii. ***LostFocus:*** Ενημερώνει το *Source* εφ' όσον το φύγει ο έλεγχος από το *binding* Target.
iii. ***Explicit:*** Καλείται η μέθοδος *UpdateSource* χειροκίνητα από τον *code-behind*.

Converter: Επιτρέπει την μετατροπή των μεταδιδόμενων δεδομένων, από τη μορφή στην οποία βρίσκονται στην πηγή, σε εκείνην που πρέπει να έχουν όταν ενημερώνουν το *UI control*. Παραθέτουμε ένα παράδειγμα:

```
<TextBox Text="{Binding myTextProperty}" Visibility="{Binding Path=isTextVisible,↵
 Converter={StaticResource myBooleanToVisibilityConverter}}" />
```

Εικόνα 52: Παράδειγμα μετατροπής των μεταδιδόμενων δεδομένων, από την πηγή στο *UI element*, μέσω της παραμέτρου *Converter*.

Εδώ, μετατρέπεται η *boolean* μεταβλητή σε *enumeration Visibility*, προκειμένου να είναι ορατό (ή όχι) το *Textbox element*.

3.3.2.6

4 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΕΦΑΡΜΟΓΗΣ

Η αρχιτεκτονική λογισμικού είναι από τους βασικότερους παράγοντες για την σταθερότητα του και την αποδοχή του από τους χρήστες. Μια ορθολογική αρχιτεκτονική πέρα από την ομαλή λειτουργία του λογισμικού, μπορεί να εγγυηθεί ότι θα λειτουργεί γρήγορα, θα καταναλώνει λίγους πόρους δίχως να εμποδίζει άλλες λειτουργίες, ενώ θα διαμορφώνει το έδαφος για μελλοντικές αλλαγές ή προσθήκες με μικρό κόπο, δίχως να χρειάζονται αλλαγές στην υπάρχουσα δομή του.

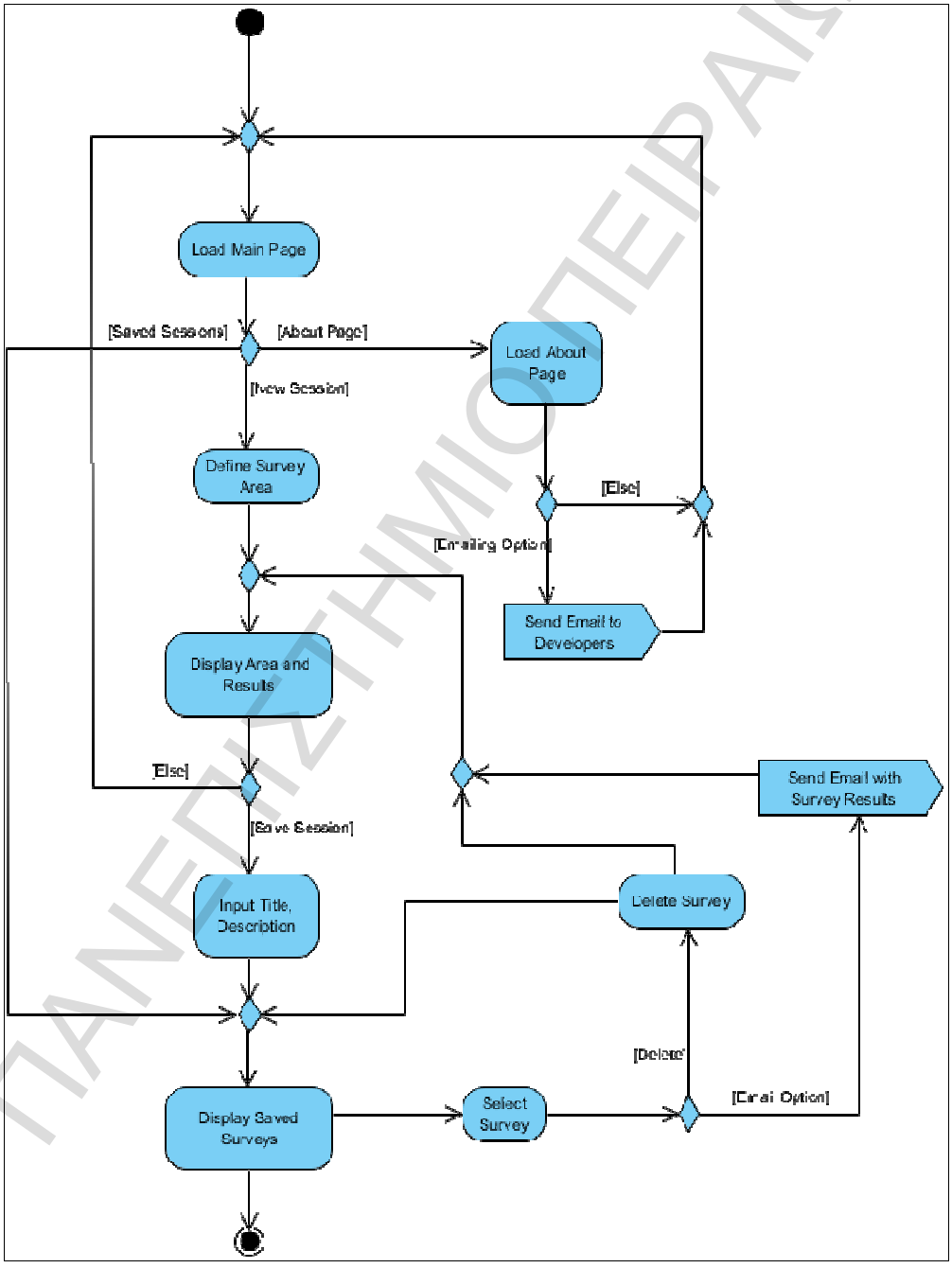
Σε παλαιότερες εποχές, όταν οι πλατφόρμες ανάπτυξης και οι γλώσσες προγραμματισμού είχαν πιο περιορισμένες δυνατότητες, τα λογισμικά δομούνταν κατά τρόπο που ο ίδιος κώδικας που ήταν υπεύθυνος για το *interface*, αναλάμβανε τη διαχείριση λειτουργιών ανάκτησης-αποθήκευσης των δεδομένων στη βάση καθώς και την επεξεργασία του δεδομένων που προέρχονται από το χρήστη και προορίζονταν για το δίσκο και αντίστροφα. Αυτός ο "μονολιθικός" τρόπος σχεδιασμού δε συνίσταται, παρά μόνο σ τις περιπτώσεις όπου το μέγεθος της εφαρμογής είναι μικρό και ο διαθέσιμος χρόνος υλοποίησης είναι περιορισμένος.

Η εφαρμογή μας εκμεταλλευόμενη τις δυνατότητες που παρέχει η πλατφόρμα ανάπτυξης (*VS 2010*), δομείται σε 3 στρώματα λογισμικού (*layers*). Το χαμηλότερο είναι το ιθαγενές *API* της πλατφόρμας (D*ata Access*) ,και είναι υπεύθυνο για τις read/write λειτουργίες στο δίσκο. Το επόμενο στρώμα είναι το *API* που δημιουργήσαμε εμείς για τις read/write λειτουργίες των δεδομένων στη μνήμη (*Model*) και αναπαριστά τις οντότητες όπως συναντώνται στο πραγματικό σύστημα, ενώ το τρίτο στρώμα είναι υπεύθυνο για τη συλλογή/εκτύπωση των δεδομένων από/προς το χρήστη και την ενδιάμεση επεξεργασία τους καθώς ανακτώνται/στέλνονται στο μοντέλο (*Presentation Layer*).

Ενώ το *Presentation Layer* δύναται να διαχωριστεί περαιτέρω σε δύο επιμέρους στρώματα (*View* και *Presenter layers* αντίστοιχα), χρησιμοποιήθηκαν οι ευκολίες που διαθέτει η πλατφόρμα για γρήγορη ανάπτυξη (παραγωγή αυτόματου κώδικα, *code-behind* files κ.α.), και οι οποίες ενοποιούν τα δύο αναφερόμενα στρώματα, καθώς υπήρχε προθεσμία στην υλοποίηση του έργου (διαγωνισμός ανάπτυξης mobile εφαρμογής της *Microsoft* 2012, στο *Τμήμα Ψηφιακών Συστημάτων* του *ΠΑ.ΠΕΙ*.).

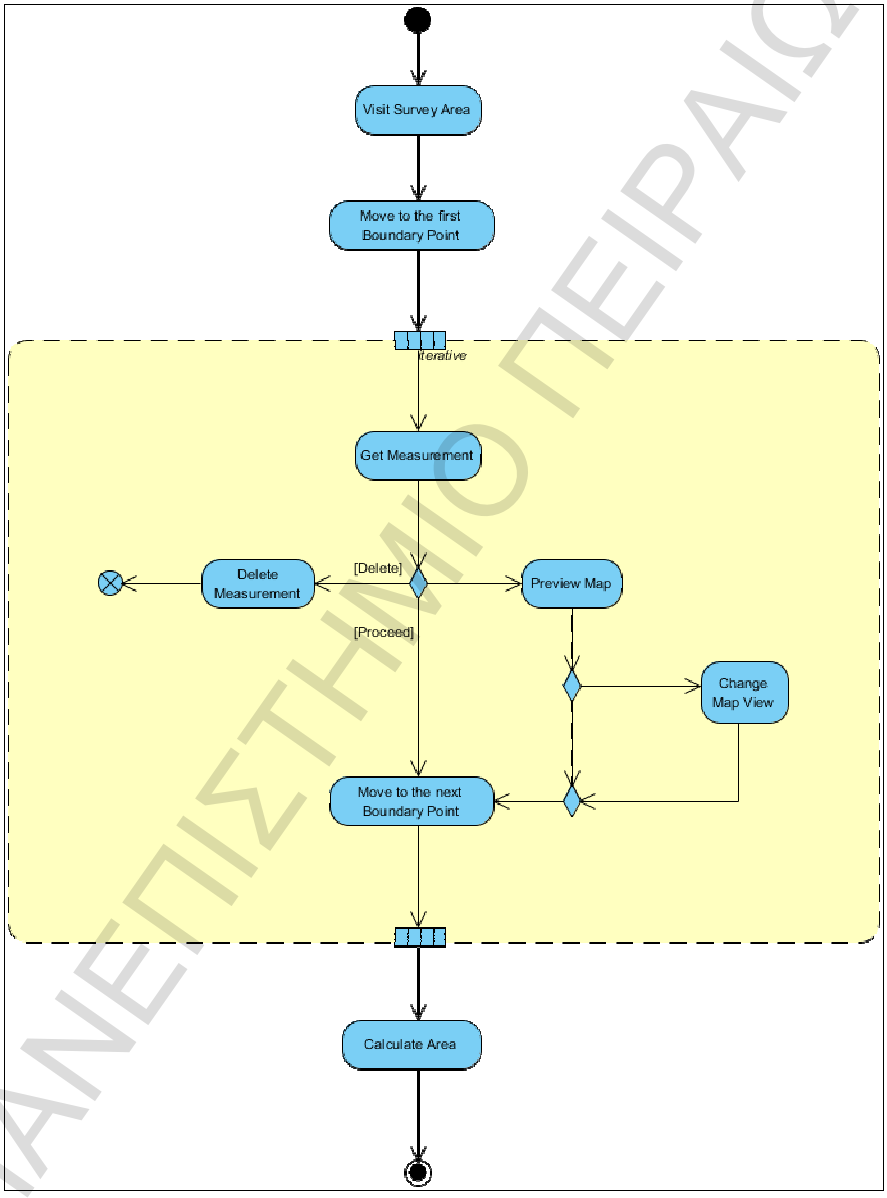
4.1 ΔΙΑΔΙΚΑΣΙΑ

Τα δυνατά σενάρια χρήσης της εφαρμογής μπορούν να αναπαρασταθούν από *Διαγράμματα Δράσεων* της γλώσσας μοντελοποίησης *UML*. Η μοντελοποίηση είναι σημαντική καθώς διαμορφώνει τον τρόπο υλοποίησης του συστήματος και κατ' επέκταση το παραδοτέο στο χρήστη προϊόν. Παραθέτουμε εδώ το ανάλογο διάγραμμα, ενώ η απεικόνιση της διαδικασίας με τις οθόνες του χρήστη, παρατίθεται στο κεφάλαιο 6:



Εικόνα 53: Δράσεις που εκτελούνται κατά τη χρήση της εφαρμογής.

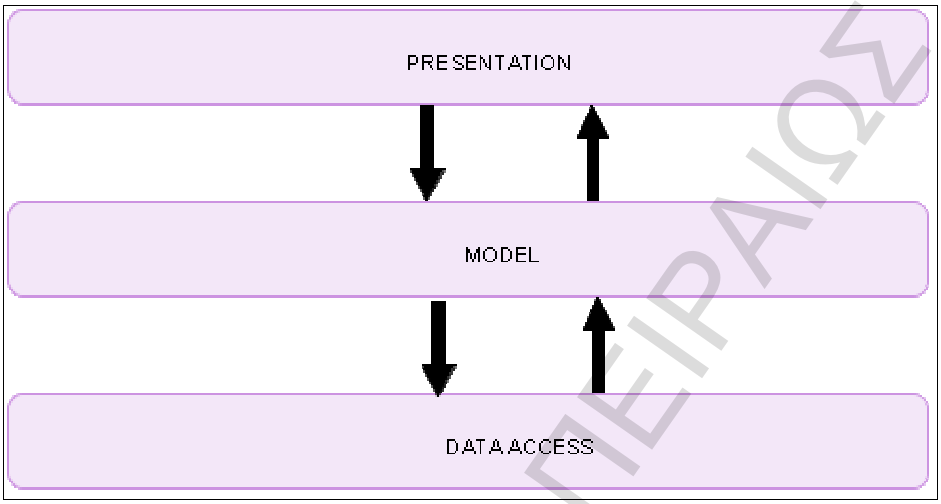
Η δραστηριότητα *Define Survey Area* μπορεί να αναλυθεί κι άλλο, καθώς περιέχει περαιτέρω δράσεις που εκτελεί η εφαρμογή αλλά και ο χρήστης. Αντιστοιχεί στην οθόνη που γίνεται η συλλογή των σημείων συνόρου της περιοχής, και η οποία πρέπει να είναι ιδιαίτερα ευέλικτη ώστε να επιτρέπονται διαγραφές σημείων, προεπισκόπηση περιοχής και να παρέχονται κι άλλες ευκολίες. Αυτό προσδίδει πολυπλοκότητα στο σύστημα, που αποτυπώνεται στο επόμενο διάγραμμα.



Εικόνα 54: Διάγραμμα δράσεων της δραστηριότητας *Define Survey Area*.

4.2 ΔΟΜΗ

Η εφαρμογή δομήθηκε σε 3 *layers* κώδικα που το καθένα επιτελεί συγκεκριμένες λειτουργίες, και συσχετίζεται με τα υπόλοιπα κατά τον τρόπο που απεικονίζει η εικόνα:



Εικόνα 55: Γενική αρχιτεκτονική εφαρμογής σε *layers*.

Τα βέλη δείχνουν τους συσχετισμούς που υπάρχουν μεταξύ των *layers* και τη φορά διακίνησης των δεδομένων. Αυτό το σχήμα είναι γενικό, ανταποκρίνεται συνολικά στην εφαρμογή κι όχι στις επιμέρους σελίδες (καθώς δεν είναι όλοι οι συσχετισμοί ενεργοί), και δεν αναλύει τις κλάσεις από τις οποίες αποτελείται το κάθε ένα *layer*. Αυτό θα γίνει στις επόμενες παραγράφους.

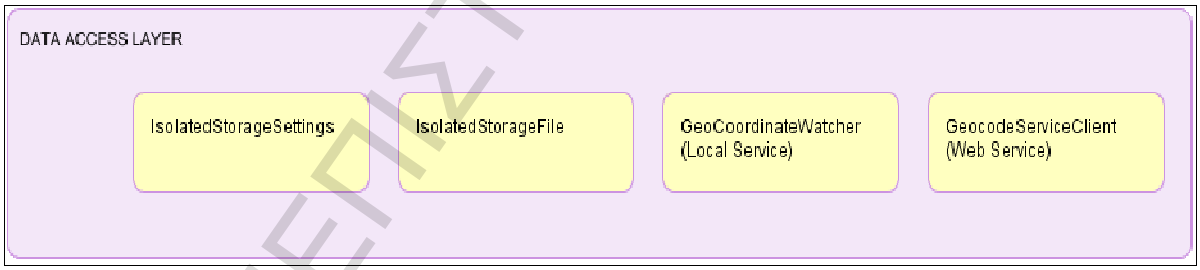
Περιγράφουμε τώρα το κάθε ένα *layer* ως προς τη λειτουργικότητα του ξεκινώντας από κάτω προς τα πάνω.

DATA ACCESS LAYER

Το στρώμα αυτό είναι υπεύθυνο για την αποθήκευση/ανάκτηση δεδομένων από/προς τις μονάδες αποθήκευσης (*persistent storage*), δίνοντας ένα πλήρες *interface* από μεθόδους για τη διαχείρισή τους. Με αυτό τον τρόπο παρακάμπτεται η ανάγκη να γνωρίζει ο προγραμματιστής λεπτομέρειες σχετικά με τον τρόπο που υλοποιείται η αποθήκευση, τις μεθόδους που χρησιμοποιεί εσωτερικά κλπ. Πέρα από αποθηκευτικά μέσα, στο στρώμα αυτό περιλαμβάνονται τα *Web services* καθώς και άλλες πηγές δεδομένων. Στην εφαρμογή μας το στρώμα αυτό αποτελείται από τα εξής στοιχεία:

- ***IsolatedStorageSettings***: Παρέχει ένα *container* τύπου *Dictionary<TKey, TValue>* στο δεσμευμένο χώρο της εφαρμογής, για την αποθήκευση μεταβλητών ως ζεύγη key/value. Συνήθως χρησιμοποιείται για την αποθήκευση μεμονωμένων δεδομένων που δεν ανήκουν σε συλλογή. Στην εφαρμογή χρησιμοποιείται για την αποθήκευση απλών τύπων όπως το μετρητή των αρχείων *.png*, τη ρύθμιση ενεργοποίησης η όχι του *GPS* κ.α.
- ***IsolatedStorageFile***: Παρέχει ένα *virtual directory* για την αποθήκευση αρχείων δομημένων σε *directories*. Σε αυτό αποθηκεύονται οι παρελθούσες μετρήσεις που επέλεξε να σώσει ο χρήστης (εμβαδό, συντεταγμένες, περιγραφή). Επίσης αποθηκεύονται και οι φωτογραφίες *.png*.
- ***GeocoordinateWatcher*** (*local service*): Επικοινωνεί με τον *sensor* του *GPS* της συσκευής, και επιστρέφει τις γεωγραφικές συντεταγμένες της τρέχουσας θέσης. Δε δέχεται δεδομένα, μονάχα στέλνει στο *Model Layer*.
- ***GeocodeService*** (*Web service*): Βρίσκει την αστική διεύθυνση (Διεύθυνση, Περιοχή, Πόλη, Χώρα) που αντιστοιχεί στο ζεύγος γεωγραφικών συντεταγμένων που του δίνονται (*latitude*, *longitude*). Στην εφαρμογή χρησιμοποιείται για την προβολή των αστικών συντεταγμένων του τρέχοντος σημείου του δέκτη. Μονάχα επιστρέφει δεδομένα.

Στην επόμενη εικόνα αναπαριστώνται οι αναφερόμενες οντότητες.

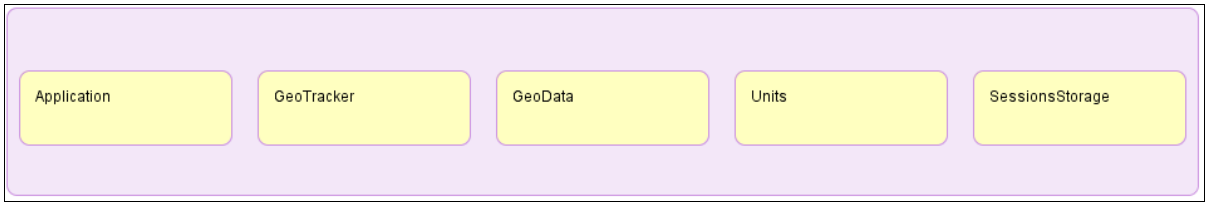


Εικόνα 56: Τα μέρη που συναποτελούν το *Data Access Layer*.

MODEL *LAYER*

Το στρώμα αυτό πραγματοποιεί την ανάκτηση/ αποθήκευση δεδομένων από/προς τη μνήμη. Σε αυτό υλοποιούνται οι οντότητες του συστήματος, όπως ορίσθηκαν κατά το αρχικό στάδιο της ανάλυσης. Μέσω του μηχανισμού *Data-binding* που εισήχθη πρώτη φορά στο περιβάλλον *VS2010*, ορισμένες από τις οντότητες είναι συνδεδεμένες με άλλα *object* s του *UI*, κατά τρόπο που μεταβολές λαμβάνουν χώρα στο *Model Layer* ενημερώνουν αυτόματα το *UI*. Αναφέρουμε τις οντότητες που συνθέτουν το *Model Layer*:

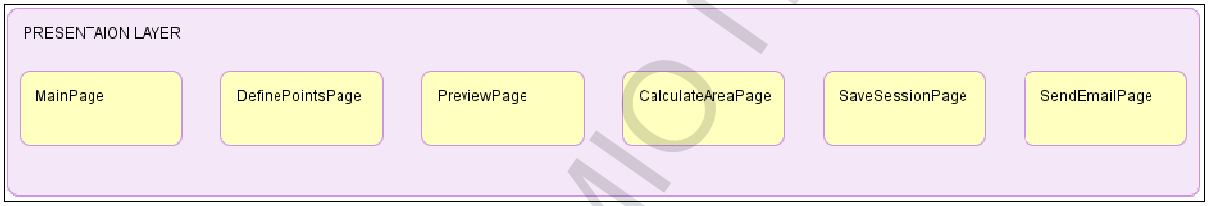
- ***Application***: Περιέχει δεδομένα και μεθόδους που είναι διαθέσιμα πάντοτε, ανεξάρτητα από τη σελίδα στην οποία η οποία βρίσκεται ο έλεγχος. Περιέχει τις τρέχουσες ρυθμίσεις του χρήστη (εμφάνιση χάρτη, μονάδες μέτρησης, μονοπάτια), καθώς και κώδικα που εκτελείται κατά την εκκίνηση (ή το κλείσιμο), την ενεργοποίηση (ή την απενεργοποίηση της εφαρμογής) όπου λαμβάνει χώρα ή ανάκτηση (ήαποθήκευση) δεδομένων στο *Data Access Layer*.
- ***GeoTracker***: Περιέχει *reference* στο *GeocoordinateWatcher*, από το οποίο αντλεί την τρέχουσα θέση του δέκτη καθώς και την κατάσταση του (ενεργό/ανενεργό, δίχως σήμα κλπ.). Είναι συνδεδεμένος με τα αντίστοιχα *Textboxes* του *UI* μέσω του μηχανισμού *Data-binding*, που επιτρέπει την άμεση, με μικρό επεξεργαστικό κόστος ενημέρωση του *UI* των συνεχόμενων μεταβολών θέσης.
- ***GeoData***: Διατηρεί τα σημεία που συλλέγει ο χρήστης όταν καταγράφει μια περιοχή, και υπολογίζει τα γεωμετρικά μεγέθη της εν λόγω περιοχής τα οποία αποθηκεύει στην μνήμη. Μέσω του μηχανισμού *Data-binding* οι μεταβολές αυτές ενημερώνουν τα αντίστοιχα *object* s στο *UI*, που είναι υπεύθυνα για την προβολή των μετρήσεων στο χρήστη.
- ***Units***: Περιέχει τις ρυθμίσεις προβολής που έχει ορίσει ο χρήστης. Συγκεκριμένα, αποθηκεύει τις επιλεγόμενες μονάδες μήκους και εμβαδού, και περιέχει τις συναρτήσεις μετατροπής που χρησιμοποιεί ο μηχανισμός *data-binding*, ώστε τα τελικά αποτελέσματα να προβάλλονται σε αυτές τις μονάδες. Ακόμη, περιέχει τον τύπο του χάρτη προβολής.
- ***SessionsStorage***: Σε αυτό το *object* διαβάζονται τα αποθηκευμένα στο δίσκο δεδομένα. Συγκεκριμένα, περιέχει συλλογή από τις μετρήσεις που έχει πραγματοποιήσει ο χρήστης. Κάθε μια μέτρηση αναπαρίσταται με ένα αντικείμενο *Session*, το οποίο περιέχει το μοναδικό αναγνωριστικό αριθμό της μέτρησης, τον τίτλο και περιγραφή που έδωσε ο χρήστης, τα σημεία που οριοθετούν την περιοχή καθώς και τις επιλεγμένες μονάδες μέτρησης. Περιέχει μια αναφορά στο *object IsolatedStorageFile* του *Data Access Layer* προκειμένου να καλεί τις ανάλογες μεθόδους ανάκτησης/αποθήκευσης των δεδομένων στο δίσκο.



Εικόνα 57: Τα μέρη που συναποτελούν το *Model Layer*.

PRESENTATION LAYER

Σε αυτό το στρώμα υπάρχουν οι κλάσεις που υλοποιούν το *UI* (*pages*) καθώς και οι μέθοδοι που καλούνται (*event handlers*) κατά τον σκανδαλισμό *events* που λαμβάνουν χώρα όταν ο χρήστης αλληλεπιδρά με τα στοιχεία ελέγχου της οθόνης (*Buttons*, *Checkboxes* κλπ.). Το περιβάλλον ανάπτυξης επιτρέπει την "εσωτερική διασύνδεση" ορισμένων στοιχείων ελέγχου του *Presentation Layer* με κάποιες οντότητες του *Model Layer*, μέσω της γλώσσας *XAML*, που εκτός από το χτίσιμο του περιγράμματος των σελίδων, χρησιμοποιείται και για την απόδοση λειτουργικότητας στο *UI*. Ως εκ τούτου παρακάμπτεται η ανάγκη για τον ορισμό *event handlers* για κάθε ένα στοιχείο ελέγχου. Όλες οι σελίδες της εφαρμογής κληρονομούν την υπερκλάση *PhoneApplicationPage* που ορίζει τα βασικά στοιχεία προβολής της σελίδας (διαστάσεις, προσανατολισμός, ονομασία σελίδας κ.α.). Οι κλάσεις του *Presentation Layer* παρατίθενται στο παρακάτω διάγραμμα:



Εικόνα 58: Τα μέρη που συναποτελούν το *Presentation Layer*.

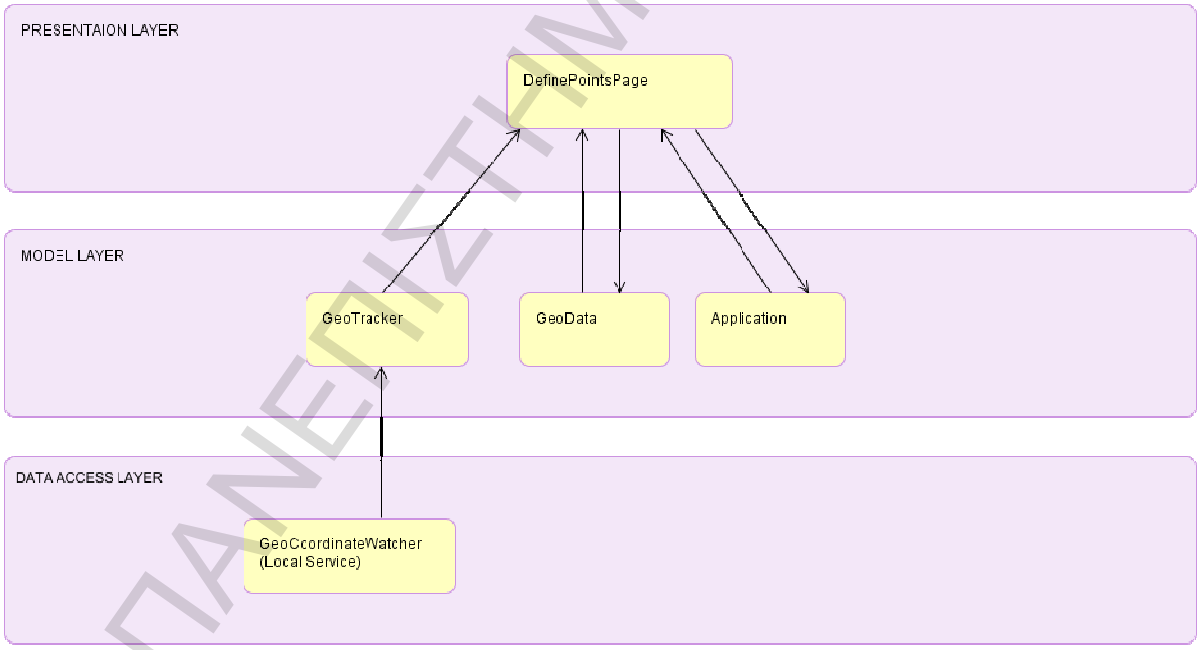
Οι οθόνες που παρουσιάζονται στο κεφάλαιο 6, περιγράφουν αναλυτικά τη λειτουργικότητα των *Views* του *Presentation Layer*.

4.3 ΑΛΛΗΛΕΠΙΔΡΑΣΗ ΜΕΡΩΝ ΣΥΣΤΗΜΑΤΟΣ

Όπως αναφέρθηκε παραπάνω, τα *layers* δημιουργήθηκαν για την ορθολογικότερη αξιοποίηση των πόρων της συσκευής, και για την ευκολότερη ανάπτυξη και περαιτέρω συντήρηση της ίδιας της εφαρμογής. Η γενική αρχιτεκτονική απεικονίζεται στο σχήμα Εικόνα 55 στην Παρ. 4.1, ωστόσο ανά σελίδα, διαφέρουν οι κλάσεις που συσχετίζονται, η φορά διακίνησης των δεδομένων μεταξύ τους, και κατ' επέκταση ο τρόπος που αλληλεπιδρούν τα διάφορα *layers*. Ενδεικτικά, θα αναφέρουμε δύο από τις σελίδες της εφαρμογής, και συγκεκριμένα αυτές που υλοποιούν την κύρια λειτουργικότητά της, αναλύοντας τις αλληλεπιδράσεις που λαμβάνουν χώρα μεταξύ των διαφόρων κλάσεων, προκειμένου να επιτύχουν την επιθυμητή λειτουργικότητα.

ΟΡΙΣΜΟΣ ΠΕΡΙΟΧΗΣ ΜΕΤΡΗΣΗΣ (ΣΕΛΙΔΑ ΕΦΑΡΜΟΓΗΣ)

Στη σελίδα αυτή ο χρήστης καταγράφει το στίγμα *GPS* από τα συνοριακά σημεία της μετρούμενης περιοχής. Για κάθε μια καταγραφή λαμβάνουν χώρα πολλαπλές αλληλεπιδράσεις μεταξύ των *layers* της εφαρμογής. Στο επόμενο σχήμα απεικονίζονται οι εμπλεκόμενες κλάσεις και οι αλληλεπιδράσεις τους:



Εικόνα 59: Διάγραμμα με τις κλάσεις της σελίδας *DefinePointsPage*.

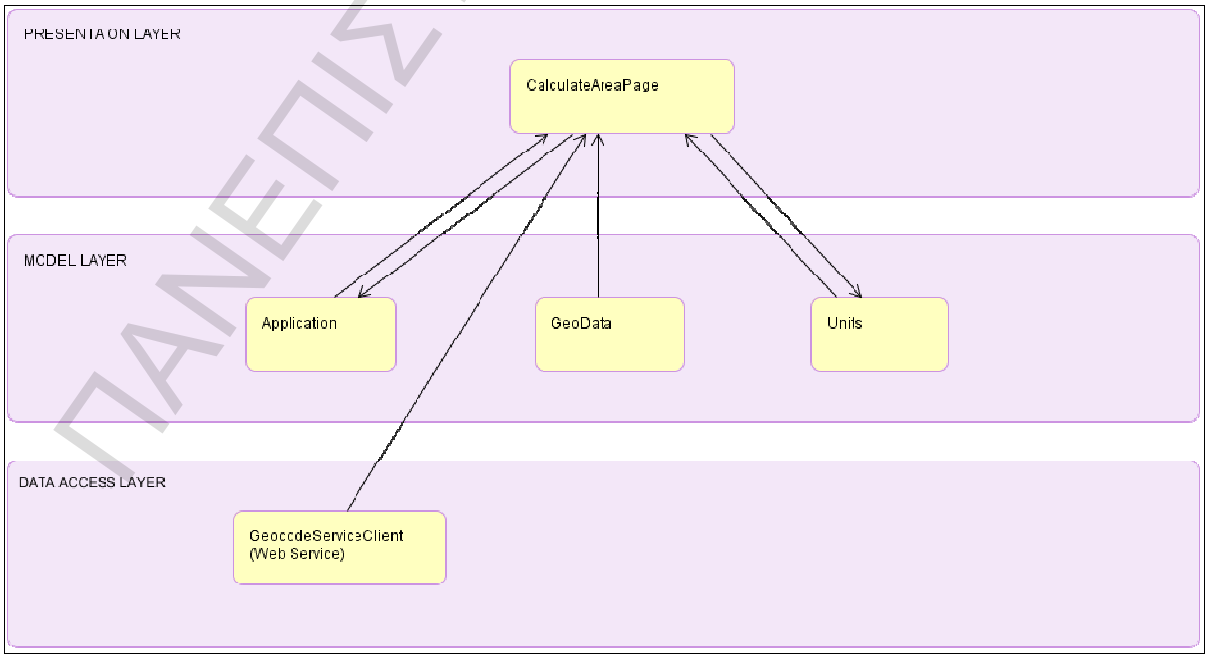
Η διαδικασία ορισμού μιας νέας περιοχής μέτρησης είναι η εξής:

1. Ο χρήστης πατάει το *Button* για την καταγραφή του τρέχοντος στίγματος *GPS*, το οποίο ορίζεται στην κλάση *DefinePointsPage* στο *Presentation Layer* (*PL*). Η τελευταία καλεί την μέθοδο *UpdateCurrentPosition()* της *GeoTracker*.

2. Η *GeoTracker* καλέι διαδοχικά τις μεθόδους *Stop()* και *Start()* της *GeoCoordinateWatcher* προκειμένου να ανανεωθεί η μέτρηση της τρέχουσας θέσης.
3. Η *GeoCoordinateWatcher* ενημερώνει την *property Position* με την τρέχουσα θέση, την οποία διαβάζει η *GeoTracker* αποθηκεύοντάς τη σε ένα *array*.
4. Τα βήματα 2 και 3 επαναλαμβάνονται 24 φορές ώστε να έχουν συλλεχθεί 24 μετρήσεις της τρέχουσας θέσης.
5. Η *GeoTracker* υπολογίζει το μ.ο. των μετρήσεων που έχουν αποθηκευθεί προσωρινά στην *array* του βήματος *3*, και ενημερώνει την *property* της με όνομα *currentPosition* με την τιμή αυτή.
6. Η *DefinePointsPage* καλεί τη μέθοδο *Add()* της *GeoData* δίνοντας σαν όρισμα την τιμή της νέας μέτρησης. Η *GeoData* εισάγει την τιμή αυτή στην *property SelectedPointsSeq*. Μέσω του *Data-binding* ενημερώνεται αυτόματα το αντίστοιχο *Listbox* του *UI* γνωστοποιώντας στο χρήστη ότι η καταχώρηση της νέας τιμής πραγματοποιήθηκε.
7. Τα βήματα 1 έως 6 επαναλαμβάνονται για όλα τα σημαία στα οποία ο χρήστης καταγράφει το σήμα *GPS*.

ΥΠΟΛΟΓΙΣΜΟΣ ΕΜΒΑΔΟΥ ΠΕΡΙΟΧΗΣ (ΣΕΛΙΔΑ ΕΦΑΡΜΟΓΗΣ)

Στη σελίδα αυτή υπολογίζονται τα γεωμετρικά μεγέθη της μετρούμενης περιοχής, ενώ αναπαρίσταται η τελευταία στο χάρτη υπό μορφή πολυγώνου, με κορυφές τα επιλεγμένα από το χρήστη σημεία. Η αλληλουχία των βημάτων της διεργασίας, είναι η εξής:



Εικόνα 60: Διάγραμμα με τις κλάσεις της σελίδας *CalculateAreaPage*.

1. Κατά το φόρτωμα της σελίδας, η *CalculateAreaPage* "ρωτά" την *Application* για τα συλλεγόμενα από το χρήστη σημεία της υπό μέτρηση περιοχής.
2. Η Application επιστρέφει μια *reference* στη *GeoData* η οποία μέσω του μηχανισμού *Data-binding* ενημερώνει το χάρτη αναπαριστώντας την υπό μέτρηση περιοχή
3. Η *CalculateAreaPage* καλεί τις μεθόδους *CalculateArea*() και *CalculatePerimeter*() της *GeoData*, η οποία ενημερώνει τις αντίστοιχες *properties*.
4. Μέσω *Data-binding*, καλούνται οι μέθοδοι *FormatAreaResult*() και *FormatDistanceResult*() δεχόμενες ως όρισμα τις υπολογιζόμενες τιμές του εμβαδού και της περιμέτρου αντίστοιχα, που είναι αποθηκευμένες στην *GeoData*. Κατόπιν, οι τιμές αυτές καταχωρούνται στα αντίστοιχα *Textboxes* του *UI*, ενημερώνοντας τον χρήστη.
5. Η *CalculateAreaPage* καλεί τη μέθοδο *ReverseGeocodeAsync* της *GeocodeService* με όρισμα τη θέση του τελευταίου σημείου που ελήφθη από το χρήστη κατά την καταγραφή της μετρούμενης περιοχής. Ο *Bing Maps Server* απαντά στο ερώτημα επιστρέφοντας την αστική διεύθυνση του σημείου (Οδός, περιοχή κλπ.), η οποία καταχωρείται στο ανάλογο *Textbox* του *UI*.

5 TESTING & ΠΙΣΤΟΠΟΙΗΣΗ ΑΠΟ *MICROSOFT*

Η εφαρμογή που αναπτύξαμε, ανέβηκε στο Windows *marketplace* αφού πρώτα πιστοποιήθηκε από τους τεχνικούς της *Microsoft* στην Αμερική, σε μια τυποποιημένη και μακροσκελή διαδικασία. Η τελευταία αποσκοπεί στο να εξασφαλίσει ότι οι εφαρμογές που θα διατίθενται από το επίσημο *marketplace*, θα είναι σταθερές με μικρές απαιτήσεις σε πόρους της συσκευής του χρήστη, δε θα είναι κακόβουλες, θα τηρούν την υφιστάμενη νομοθεσία αναφορικά με το περιεχόμενό τους και την επεξεργασία των προσωπικών δεδομένων του χρήστη, ενώ παράλληλα θα ακολουθούν και συγκεκριμένα λειτουργικά χαρακτηριστικά.

Οι προδιαγραφές που αναφέρθηκαν εφαρμόζουν διαφορετικά ανά εφαρμογή, καθώς το είδος των πληροφοριών που αυτή επεξεργάζεται, τα *sensors* που ενεργοποιεί, η τεχνολογία που χρησιμοποιεί (*XNA* για games ή *Silverlight* για εφαρμογές), διαμορφώνουν και τους ανάλογους περιορισμούς. Στις επόμενες παραγράφους θα αναφέρουμε τις προδιαγραφές εκείνες που ελήφθησαν υπ' όψιν κατά την υλοποίηση της παρούσας εφαρμογής.

5.1 ΠΡΟΔΙΑΓΡΑΦΕΣ ΕΤΑΙΡΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

Θεσπίστηκαν για να προστατεύουν το *repository* διανομής των εφαρμογών (*windows marketplace*) και των χρηστών του, και για την καθιέρωση κοινού συστήματος λειτουργικών χαρακτηριστικών των εφαρμογών, που θα καλύπτουν κατά το μέγιστο δυνατό τις καταγεγραμμένες ανάγκες των χρηστών.

Οι προδιαγραφές που επηρέασαν τη εφαρμογή μας είναι:

1. Η εφαρμογή θα πρέπει να μη υπονομεύει την ασφάλεια των συσκευών, του *marketplace*, είτε να δημιουργεί την πιθανότητα βλαψίματος του χρήστη της
2. Η καταγραφή της τρέχουσα τοποθεσίας θα πρέπει να υλοποιείται αποκλειστικά μέσω του windows phone *API*.
3. Η τρέχουσα τοποθεσία αποτελεί προσωπικό δεδομένο, και ως εκ τούτου οι χρήστες θα πρέπει να ενημερώνονται από την εφαρμογή για τον τρόπο αξιοποίησης των δεδομένων τους, ενώ θα πρέπει να υπάρχει ενσωματωμένη η δυνατότητα απενεργοποίησης του *sensor* εντοπισμού (*GPS*), λειτουργία που θα πρέπει να γνωστοποιείται στο χρήστη μέσω του *Privacy Policy* της εφαρμογής.

4. Τα συλλεγόμενα δεδομένα τοποθεσίας θα πρέπει να χρησιμοποιούνται αποκλειστικά και μόνο για την επικαλούμενη λειτουργικότητα κι όχι για άλλους σκοπούς.
5. Να είναι πλήρως λειτουργική όταν ανακτάται από το *marketplace*
6. Η κοστολόγησή της να μην είναι παράλογη, αντιθέτως να ρυθμίζεται από τη φύση και το εύρος των λειτουργιών της.

5.2 ΤΕΧΝΙΚΕΣ ΠΡΟΔΙΑΓΡΑΦΕΣ

Αναφέρονται στον επόμενο πίνακα οι αντίστοιχες προδιαγραφές, και οι έλεγχοι που πραγματοποιήθηκαν από τους testers της *Microsoft*, για την πιστοποίηση της εφαρμογής. Αναφέρουμε εδώ πως αντίστοιχοι έλεγχοι διενεργήθηκαν και από μεριά μας πρώτου υποβληθεί η εφαρμογή, προκειμένου να εξασφαλισθεί η ολοκλήρωση της πιστοποίησης από τους *testers*. Οι έλεγχοι διενεργήθηκαν τόσο σε πραγματική συσκευή, όσο και στους διαθέσιμους *emulators* της πλατφόρμας ανάπτυξης.

Απαίτηση	**Διενεργούμενος Έλεγχος**
Να είναι λειτουργική σε διαφορετικές συσκευές που πληρούν τις απαιτήσεις υλικού (μέγεθος μνήμης) και συμβατή σε εκείνες που έχουν κατώτερες προδιαγραφές.	1. Εγκατάσταση σε δύο και παραπάνω συσκευές που πληρούν τις αναφερόμενες απαιτήσεις υλικού. 2. Διαδοχική εγκατάσταση και απεγκατάσταση δίχως να παρουσιασθούν σφάλματα. 3. Φόρτωμα της εφαρμογής και χρήση της δίχως να παρατηρείται κάποια ασυμβατότητα με τη συσκευή που θα οδηγούσε σε "πάγωμα" της τελευταίας.
Να χειρίζεται κάθε *exception* που μπορεί να "σκανδαλισθεί" (ανεξάρτητα αν η πηγή είναι το υπάρχον *API* είτε αυτό που δημιούργησε ο χρήστης), αντί να κλείσει απρόσμενα η εφαρμογή.	1. Φόρτωμα της εφαρμογής, διαρκής πλοήγηση στις σελίδες, κλείσιμο. 2. Έλεγχος δυνατότητας απόκρισης της εφαρμογής ως προς την εισαγωγή δεδομένων και την πλοήγηση σε περίπτωση που προκύψει *exception*. Σε περίπτωση που "παγώσει", η πιστοποίηση αποτυγχάνει.

Θα πρέπει να εμφανίσει την πρώτη οθόνη εντός 3s από τη στιγμή έναρξης της φόρτωσης και να μπορεί να ανταποκρίνεται σε ενέργειες του χρήστη εντός 20s από τη στιγμή εκείνη.	1. Χρονομέτρηση του διαστήματος μέχρι την εμφάνισης της αρχικής οθόνης από την έναρξη της φόρτωσης. 2. Χρονομέτρηση του διαστήματος στο οποίο μπορεί να ανταποκρίνεται σε ενέργειες του χρήστη.
Το φόρτωμα της εφαρμογής να ικανοποιεί την αμέσως προηγούμενη απαίτηση ενώ έχει προηγηθεί πρόσφατο κλείσιμο από το χρήστη.	1. Φόρτωμα εφαρμογής, πλοήγηση και κλείσιμο με χρήστη του *Back Button*. 2. Άμεσο φόρτωμα της εφαρμογής και έλεγχος ότι υπό αυτές τις συνθήκες ικανοποιεί την αμέσως προηγούμενη προδιαγραφή.
Το φόρτωμα της εφαρμογής να ικανοποιεί την αμέσως προηγούμενη απαίτηση ενώ έχει προηγηθεί απενεργοποίηση της (η απενεργοποίηση μπορεί να προκληθεί με το πάτημα του *Start Button* από το χρήστη, είτε από την κλήση κάποιου *Launcher* η *Chooser* από τη συσκευή).	1. Φόρτωμα της συσκευής. 2. Απενεργοποίηση πατώντας το *Start Button*, φόρτωμα της από το menu των διαθέσιμων εφαρμογών. 3. Έλεγχος ότι πληροί την απαίτηση εμφάνισης αρχικής οθόνης και αποκρισιμότητας εντός των προβλεπόμενων χρόνων.
Το *Back Button* πρέπει να πλοηγεί το χρήστη ως ακολούθως: 1. Στην περίπτωση που είναι στην αρχική σελίδα, να κλείνει έξοδο από την εφαρμογή. 2. Αν είναι σε ενδιάμεση σελίδα να τον πλοηγεί σε κάποια προηγούμενη στη στοίβα των σελίδων (κι όχι αναγκαστικά σε αυτή που βρισκόταν ακριβώς προηγουμένως). 3. Αν έχει κάποιο *dialog*/*context menu* ανοικτό, να το κλείνει παραμένοντας στην τρέχουσα σελίδα.	1. Φόρτωμα εφαρμογής, πάτημα του *Back Button* μετά το πλήρες φόρτωμα, επιβεβαίωση ότι θα κλείσει κανονικά. 2. Φόρτωμα εφαρμογής, πλοήγηση σε ενδιάμεση σελίδα, πάτημα *Back Button*, έλεγχος ότι επανέρχεται σε προηγούμενη σελίδα ως προς τη στοίβα των σελίδων. 3. Φόρτωμα εφαρμογής άνοιγμα κάποιου *dialog*/*context menu*, πάτημα του *Back Button*, κι έλεγχος ότι θα κλείσει το *dialog*/*context menu* που είναι ανοικτό.

Θα πρέπει να μη ξεπεράσει το όριο των 90 Mb χρησιμοποιούμενης μνήμης σε μια δεδομένη στιγμή.	1. Χρήση ειδικού εργαλείου για monitoring (διαθέσιμο στο *Visual Studio 2010*).
Θα πρέπει να μη καθυστερεί η εμποδίζει το χρήστη να πραγματοποιεί είτε να δέχεται είτε να τερματίζει τηλεφωνικές κλήσεις.	1. Επιβεβαίωση ύπαρξης σήματος τηλεφωνίας. 2. Φόρτωμα εφαρμογής. 3. Λήψη τηλεφωνικής κλήσης. 4. Έλεγχος ποιότητας κλήσης ως προς τυχόν παρεμβολές είτε δονήσεις της συσκευής. 5. Τερματισμός κλήσης. 6. Επιβεβαίωση ομαλής επαναφοράς της εφαρμογής στο προσκήνιο. 7. Απενεργοποίηση εφαρμογής πατώντας το *Start Button*. 8. Πραγματοποίηση κλήσης.
Θα πρέπει να μη καθυστερεί η εμποδίζει το χρήστη να στέλνει *SMS* η *MMS*.	1. Επιβεβαίωση ύπαρξης σήματος τηλεφωνίας. 2. Επιβεβαίωση ότι δεν είναι σε λειτουργία πτήσης. 3. φόρτωμα εφαρμογής. 4. Απενεργοποίηση πατώντας το *Start Button*. 5. Έλεγχος δυνατότητας αποστολής *SMS*/*MMS* προς άλλο διαθέσιμο τηλέφωνο.
Θα πρέπει να μη καθυστερεί η εμποδίζει το χρήστη να δέχεται *SMS* η *MMS*.	1. Επιβεβαίωση ύπαρξης σήματος τηλεφωνίας. 2. Επιβεβαίωση ότι δεν είναι σε λειτουργία πτήσης. 3. Φόρτωμα εφαρμογής. 4. Αποστολή *SMS*/*MMS* στη συσκευή και αναμονή διάρκειας 10 min. 5. Απενεργοποίηση εφαρμογής πατώντας το *Start Button*. 6. Επιβεβαίωση ότι η ειδοποίηση για το εισερχόμενο *SMS*/*MMS*, εμφανίζεται στην οθόνη εντός 5s αφ' ότου κλείσει η εφαρμογή.

Η εφαρμογή θα πρέπει να μην κλείνει απρόσμενα όταν δέχεται τηλεφωνική κλήση, *SMS* ή *MMS*.	1. Επιβεβαίωση ύπαρξης σήματος τηλεφωνίας. 2. Επιβεβαίωση ότι δεν είναι σε λειτουργία πτήσης. 3. Λήψη τηλεφωνικής κλήσης/*SMS*/*MMS*. 4. Επιβεβαίωση ότι η εφαρμογή δε θα "παγώσει" είτε κλείσει απρόσμενα. 5. Σε περίπτωση εισερχόμενης κλήσης, να απαντηθεί η κλήση επιβεβαιώνοντας παράλληλα ότι η εφαρμογή είναι στο προσκήνιο, ενώ σε περίπτωση λήψης *SMS*/*MMS*, άνοιγμα του εισερχόμενου μηνύματος και επιβεβαίωση ομαλής επαναφοράς στην εφαρμογή πατώντας το *Back Button*.
Το περιεχόμενο της εφαρμογής (κείμενο και *controls*) θα πρέπει να είναι πάντοτε ορατά, ανεξάρτητα του *theme* που έχει επιλέξει ο χρήστης (υπάρχουν 2 είδη *themes Dark/Whtie background*).	1. Μετάβαση στις ρυθμίσεις της συσκευής και ενεργοποίηση του *Dark/White background theme*. 2. Φόρτωμα της εφαρμογής 3. Επιβεβαίωση ότι το κείμενο και τα *controls* της εφαρμογής είναι ευδιάκριτα.
Θα πρέπει να αναγράφονται το όνομα της εφαρμογής, η έκδοση, και στοιχεία επικοινωνίας σε περίοπτη θέση εντός της εφαρμογής.	1. Φόρτωμα εφαρμογής 2. Επιβεβαίωση ότι η εφαρμογή αναγράφει τις απαιτούμενες πληροφορίες σε περίοπτη θέση.

Εικόνα 61: Απαιτούμενες τεχνικές προδιαγραφές των εφαρμογών του *Windows marketplace.*

5.3 ΕΠΙΠΡΟΣΘΕΤΕΣ ΠΡΟΔΙΑΓΡΑΦΕΣ

Οι πρόσθετες προδιαγραφές περιλαμβάνουν:

Απαίτηση	**Διενεργούμενος Έλεγχος**
Σε περίπτωση που ο *GPS* sensor απενεργοποιηθεί από τις ρυθμίσεις της συσκευής, η εφαρμογή να παραμένει λειτουργική (προτείνεται να ενημερώνει το χρήστη για τη μη-διαθεσιμότητα του *GPS*)	1. Μετάβαση στις ρυθμίσεις και ενεργοποίηση του *GPS* 2. Φόρτωμα εφαρμογής 3. Μετάβαση στις ρυθμίσεις και απενεργοποίηση του *GPS* 4. Επαναφορά στην εφαρμογή και έλεγχος της αποκρισιμότητάς της
Ελαχιστοποίηση χρήσης μπαταρίας όταν η οθόνη της συσκευής είναι κλειδωμένη	1. Φόρτωμα εφαρμογής 2. Κλείδωμα οθόνης συσκευής 3. Έλεγχος ότι δεν τρέχουν μη-χρειαζούμενες διεργασίες (στην εφαρμογή αυτή απενεργοποιήσαμε τη διεργασία για την τρέχουσα θέση, που ενεργοποιεί τον αισθητήρα *GPS*)
Ο ελάχιστος χρόνος αποφόρτισης της μπαταρίας θα πρέπει να είναι 120 min με την εφαρμογή να είναι ενεργή και με την οθόνη της συσκευής κλειδωμένη	1. Πλήρης φόρτιση μπαταρίας 2. Ενεργοποίηση της λειτουργίας πτήσης στη συσκευή 3. Φόρτωμα εφαρμογής 4. Κλείδωμα οθόνης 5. Επιβεβαίωση ότι η μπαταρία θα αποφορτιστεί σε τουλάχιστο 120 min

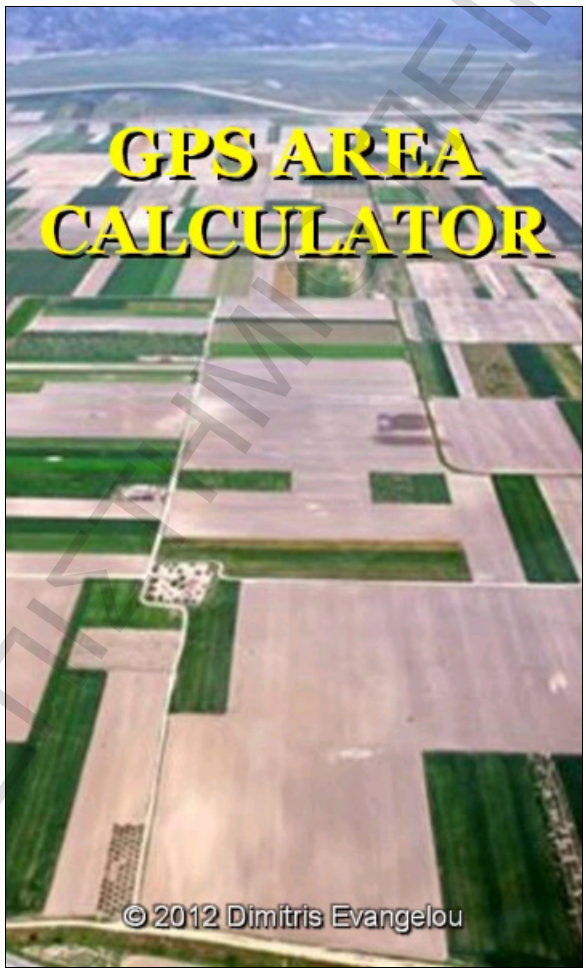
Εικόνα 62: Λοιπές απαιτούμενες προδιαγραφές των εφαρμογών του Windows marketplace.

6 ΠΑΡΟΥΣΙΑΣΗ *GPS* AREA CALCULATOR

Στο κεφάλαιο αυτό θα παρουσιάσουμε τις σελίδες της εφαρμογής ενώ παράλληλα θα εκτελούμε μια περίπτωση χρήσης για την πληρέστερη επεξήγηση των λειτουργικών λεπτομερειών. Η περίπτωση χρήσης περιλαμβάνει τον υπολογισμό του εμβαδού και της περιμέτρου που καταλαμβάνει κάποια παραλία που επιλέξαμε τυχαία. Ακολουθούν τα βήματα:

ΟΘΟΝΗ ΜΕΤΑΒΑΣΗΣ

Επιλέγοντας την εφαρμογή *GPS*AreaCalculator για ενεργοποίηση, εμφανίζεται η παρακάτω οθόνη (η οποία παραμένει επί 1.2 sec στην οθόνη):

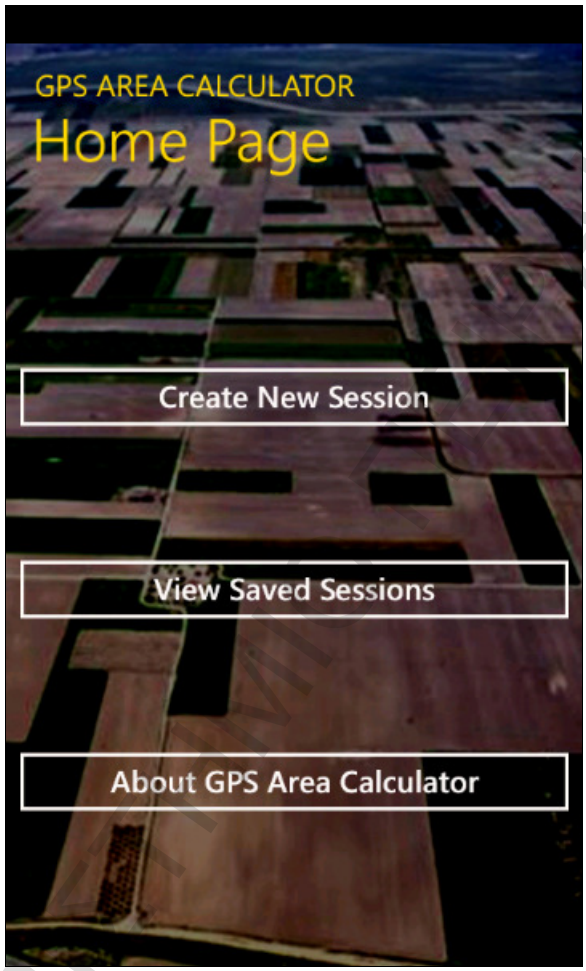


Εικόνα 63: Οθόνη μετάβασης.

Η οθόνη αυτή (*splashscreen*) κάνει πιο ομαλή τη μετάβαση στις κύριες σελίδες της εφαρμογής όσο η τελευταία φορτώνει. Είναι προαιρετική όσον αφορά την πιστοποίηση της εφαρμογής και δεν επιτελεί κάποια άλλη λειτουργία.

ΑΡΧΙΚΗ ΣΕΛΙΔΑ

Μόλις φορτώσει η εφαρμογή, εμφανίζεται η αρχική οθόνη που περιέχει το κύριο menu:



Εικόνα 64: *MainPage*

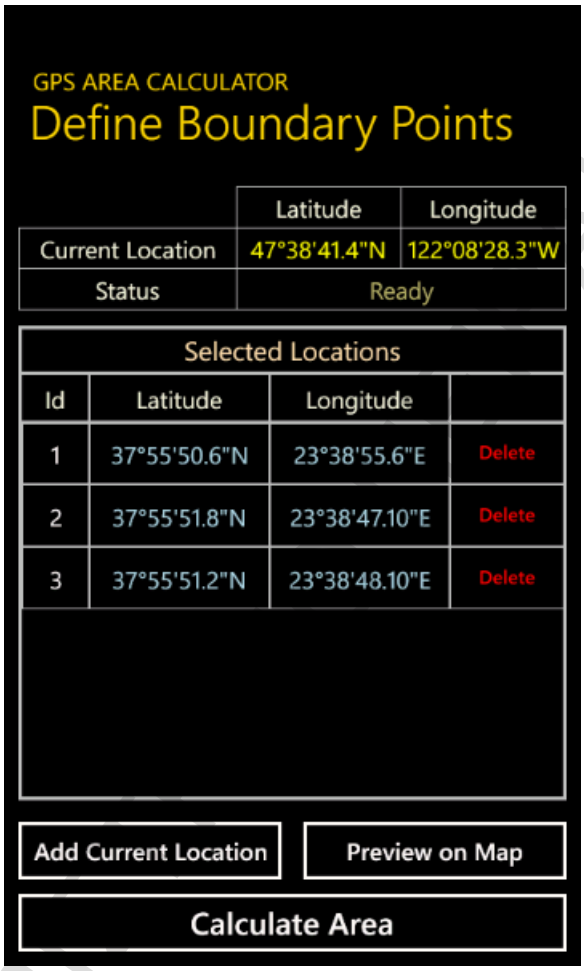
Από εδώ ο χρήστης μπορεί να:

- αρχικοποιήσει μια νέα διαδικασία υπολογισμού μιας περιοχής,
- προβάλλει τις παρελθούσες μετρήσεις και τις οποίες έχει αποθηκεύσει στη μνήμη
- να διαβάσει το "*About*" της εφαρμογής, στο οποίο μπορεί να ενημερωθεί για την τρέχουσα έκδοση και να απενεργοποιήσει τη δυνατότητα καταγραφής του σήματος *GPS*

Για μια νέα μέτρηση επιλέγει "*Create New Session*".

ΚΑΤΑΓΡΑΦΗ ΣΗΜΕΙΩΝ

Επιλέγοντας το "*Create New Session*" της αρχικής σελίδας μεταβαίνει στη σελίδα της καταγραφής των συνοριακών σημείων της περιοχής υπό μέτρηση:



Εικόνα 65: *DefineSurveyArea*

Στη σελίδα εμφανίζονται δύο πίνακες. Στον πρώτο πίνακα ενημερώνεται για την τρέχουσα θέση του (γραμμή *Current Location*) καθώς η συσκευή διενεργεί συνεχόμενες λήψεις στίγματος γι αυτό το σκοπό, ενώ στη γραμμή *Status* ενημερώνεται για την τρέχουσα κατάσταση του αισθητήρα *GPS*. Οι δυνατές καταστάσεις είναι:

1. ***Ready***: Ο χρήστης μπορεί να διενεργήσει μια νέα καταγραφή της τρέχουσας θέσης υψηλής ακρίβειας.
2. ***Tracking (hold your position)***: Ο αισθητήρας *GPS* κάνει καταγραφή, διάστημα στο οποίο δε μπορεί να γίνει νέα λήψη. Δεδομένου ότι κάνει λήψη 25 στιγμάτων προκειμένου να υπολογίσει το μέσο όρο τους, διεργασία αυτή

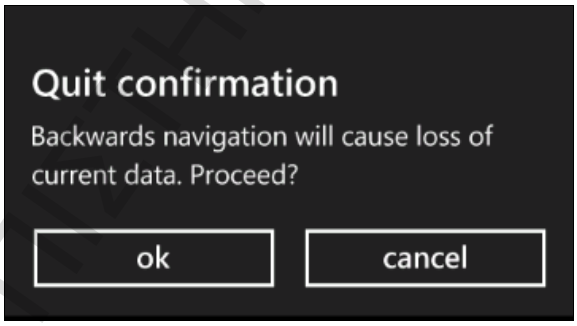
διαρκεί από 2s έως 3s. Ο χρήστης πρέπει να παραμένει ακίνητος για να μην αλλοιώνονται οι μετρήσεις.

3. ***Disabled***: Ενημερώνει το χρήστη πως δε μπορεί να διενεργήσει νέα καταγραφή της θέσης του καθώς έχει απενεργοποιήσει τον αισθητήρα *GPS* από τις ρυθμίσεις της συσκευής. Επίσης παύει και η ένδειξη της τρέχουσας θέσης (πεδίο *Current Location*).

Στο δεύτερο πίνακα βλέπει τις έως τώρα καταγεγραμμένες μετρήσεις του που αντιστοιχούν στα σημεία οριοθέτησης της περιοχής. Οι συντεταγμένες (*latitude*, *longitude*) προβάλλονται με το *format* των μοιρών, και είναι το αποτέλεσμα της άθροισης 25 μετρήσεων, πρακτική που μειώνει το σφάλμα των επί μέρους μετρήσεων. Επίσης παρέχεται η δυνατότητα, να διαγράψει μια μέτρηση εφ' όσον παρατηρήσει ότι δεν ανταποκρίνεται στην πραγματική του θέση. Σε αυτό εξυπηρετεί η επιλογή *Preview On Map* στην οποία προβάλλει τις έως τώρα καταγεγραμμένες μετρήσεις σε χάρτη.

Μια νέα καταγραφή θέσης πραγματοποιείται πατώντας το κουμπί "*Add Current Location*" , ενώ η διενέργεια των γεωμετρικών μεγεθών από τη συσκευή διενεργείται πατώντας το κουμπί "*Calculate Area*".

Αν εφ' όσον έχει διενεργηθεί μια τουλάχιστο μέτρηση, ο χρήστης πατήσει το *Back Button* της συσκευής, θα εμφανισθεί μια οθόνη επιβεβαίωσης που προειδοποιεί το χρήστη για την επερχόμενη διαγραφή των δεδομένων που συνέλλεξε έως εκείνη τη στιγμή:



Εικόνα 66: Επιβεβαίωση για την έξοδο από τη συγκεκριμένη συνεδρία μετρήσεων.

Προστατεύεται έτσι από τυχόν μη ηθελημένη ενέργεια εξόδου από την εφαρμογή.

Για να επαληθεύει την εγκυρότητα των μετρήσεων του μακροσκοπικά, μεταβαίνει στην προεπισκόπηση των σημείων (παράγραφος 6.1.4).

ΠΡΟΕΠΙΣΚΟΠΗΣΗ ΣΕ ΧΑΡΤΗ

Κατά τη διάρκεια καταγραφής των συνοριακών σημείων, ο χρήστης ενδεχομένως να μη θυμάται αν πράγματι κατέγραψε ένα επιθυμητό σημείο, είτε να θέλει να επιβεβαιώσει ότι οι καταγραφές που έκανε έως εκείνη τη στιγμή είναι ακριβείς. Σε αυτό εξυπηρετεί η σελίδα της προεπισκόπησης στην οποία οι τελευταίες προβάλλονται σε χάρτη:

Εικόνα 67: *PreviewPage*

Στο παράδειγμά μας, ο χρήστης έχει έως τώρα καταγράψει τρία συνοριακά σημεία, και το σχηματιζόμενο πολύγωνο καλύπτει ένα μέρος της έκτασης που τον ενδιαφέρει.

Στη σελίδα δίνεται η δυνατότητα για μεγέθυνση του χάρτη ώστε να επιβεβαιώσει ότι το σημάδι ανταποκρίνεται στην πραγματική θέση στην οποία έλαβε τη μέτρηση. Αν αυτό δε συμβαίνει για κάποια μέτρηση, μπορεί να επανέλθει στη σελίδα καταγραφής (*πατώντας Back to Points Selection*) στην οποία μπορεί να τη διαγράψει και την επαναλάβει.

ΥΠΟΛΟΓΙΣΜΟΣ ΓΕΩΜΕΤΡΙΚΩΝ ΜΕΓΕΘΩΝ

Από τη σελίδα καταγραφής των σημείων, μπορεί ο χρήστης να επιλέξει τον τελικό υπολογισμό των μεγεθών της περιοχής γης που μελετά. Στο παράδειγμά μας η παρακάτω οθόνη εμφανίζεται:

Εικόνα 68: *CalculateAreaPage.*

Ο χρήστης έχει εικόνα του πολύγονου της υπό διερεύνηση περιοχής, ενώ στον πίνακα κάτω από το χάρτη βλέπει τις υπολογισμένες από την εφαρμογή τιμές του εμβαδού και της περιμέτρου (*Εμβαδό*: 13216 m^2, *Περίμετρος*: 675 m).

Οι τιμές αναγράφονται αυτομάτως σε μέτρα και τετραγωνικά μέτρα, αλλά είναι εφικτή η αναγραφή τους και σε άλλες μονάδες μήκους και εμβαδού. Αυτό το ρυθμίζει ο χρήστης με την επιλογή *Change Prefs*:

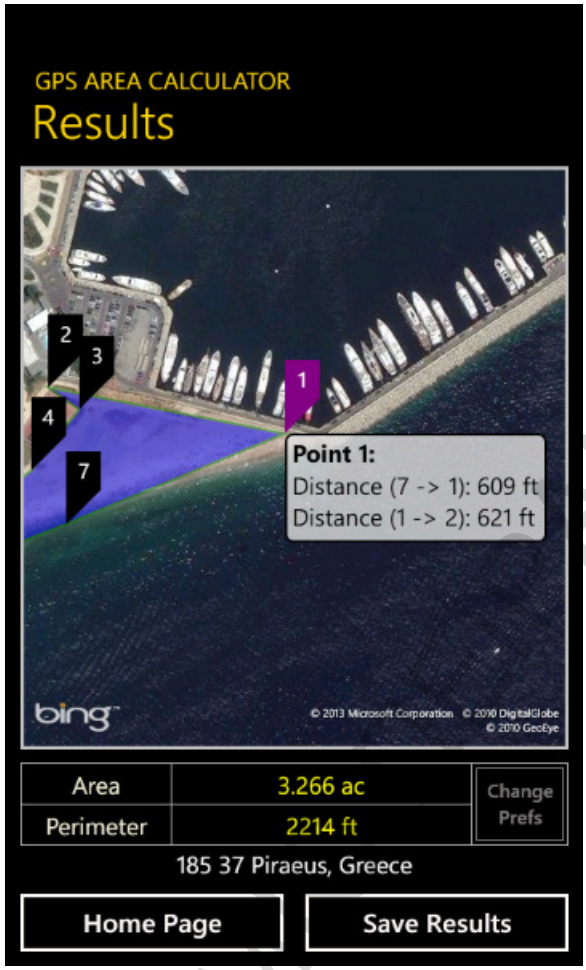
Εικόνα 69: *ChangePrefs.*Ο χρήστης μπορεί να επιλέξει άλλες μονάδες μέτρησης όπως και διαφορετικό τύπο χάρτη.

Με αυτό τον τρόπο η εφαρμογή υπολογίζει τα αποτελέσματα και σε άλλες βασικές μονάδες μέτρησης (όπως πόδια ή χιλιόμετρα για το μήκος και στρέμματα και εκτάρια για το εμβαδό).

Ο χρήστης έχει επίσης τη δυνατότητα να αλλάξει τον τύπο του χάρτη, ώστε να μη φαίνονται τα γεωμορφικά χαρακτηριστικά (που είναι και η *default* επιλογή). Έστω ότι επιλέγει *Aerial mode* για το χάρτη, πόδια για το μήκος και στρέμματα για το εμβαδό. Πατώντας *OK* κατοχυρώνονται οι αλλαγές και προβάλλεται η παρακάτω σελίδα:

Εικόνα 70: *CalculateAreaPage* με εναλλακτικές μονάδες μέτρησης και τύπο χάρτη.

Οι κορυφές του πολυγώνου έχουν αριθμηθεί με τη σειρά με την οποία ο χρήστης τις κατέγραψε. Η εφαρμογή υπολογίζει και τα μήκη των ακμών που ενώνουν τις κορυφές, και ο χρήστης μπορεί να τα προβάλλει πατώντας επάνω σε μια από τις δύο κορυφές για την ακμή που τον ενδιαφέρει. Η επόμενη οθόνη απεικονίζει τη λειτουργία αυτή:



Εικόνα 71: *CalculateAreaPage* μεένδειξη της απόστασης ανάμεσα σε δύο διαδοχικά συνοριακά σημεία.

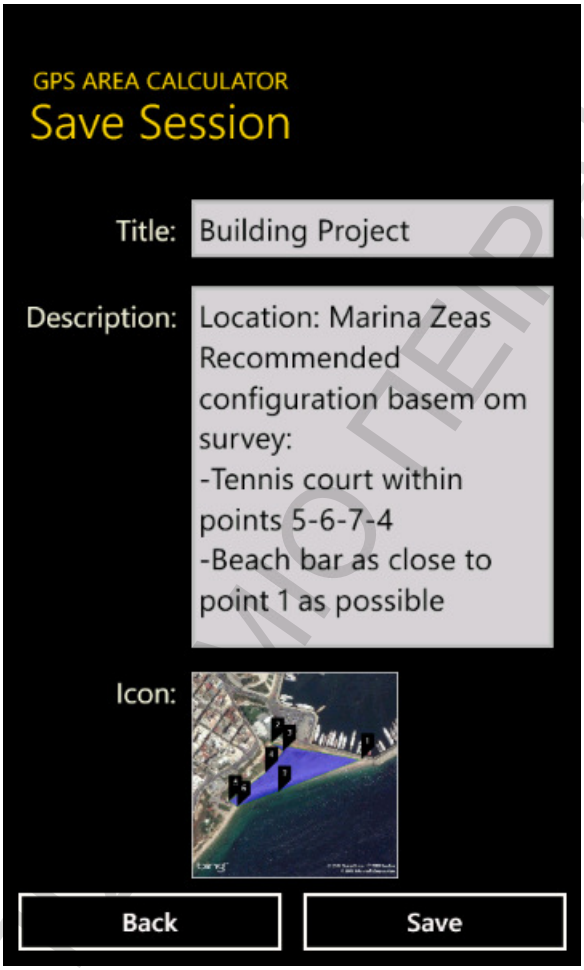
Στο παράδειγμά μας, πατώντας επάνω στην κορυφή #1, εμφανίζεται *bubble* που προβάλλει το μήκος των ακμών *7→1* και *1→2* στη μονάδα μήκους της επιλογής του χρήστη.

Να σημειωθεί ότι το στοιχείο χάρτη που υπάρχει στην οθόνη είναι διαδραστικό, και μπορεί ο χρήστης να περιηγηθεί σε αυτόν και να τον μεγεθύνει, με εφαρμογή βασικών *gestures* που υποστηρίζονται από τα *smartphones*.

Ακόμη, κάτω από τον πίνακα των αποτελεσμάτων, αναγράφεται η τρέχουσα θέση του χρήστη. Η πληροφορία αυτή αντλείται από αντίστοιχο *Web service*, και περιλαμβάνει κατά σειρά (εφ' όσον είναι διαθέσιμα) τα παρακάτω στοιχεία: Αρ. Οδού, Τ.Κ., Τόπο, Χώρα.. (σε διαφορετικές χώρες, διαφοροποιούνται και οι ανάλογες πληροφορίες, π.χ. στην Αμερική αναγράφεται και η Πολιτεία).

ΑΠΟΘΗΚΕΥΣΗ ΣΤΗ ΜΝΗΜΗ

Πατώντας την επιλογή για το σώσιμο της μέτρησής του, ο χρήστης μεταβαίνει στην επόμενη σελίδα στην οποία καλείται να επιβεβαιώσει την ενέργειά του, και προαιρετικά να εισάγει μια σύντομη περιγραφή για τη μέτρησή του.

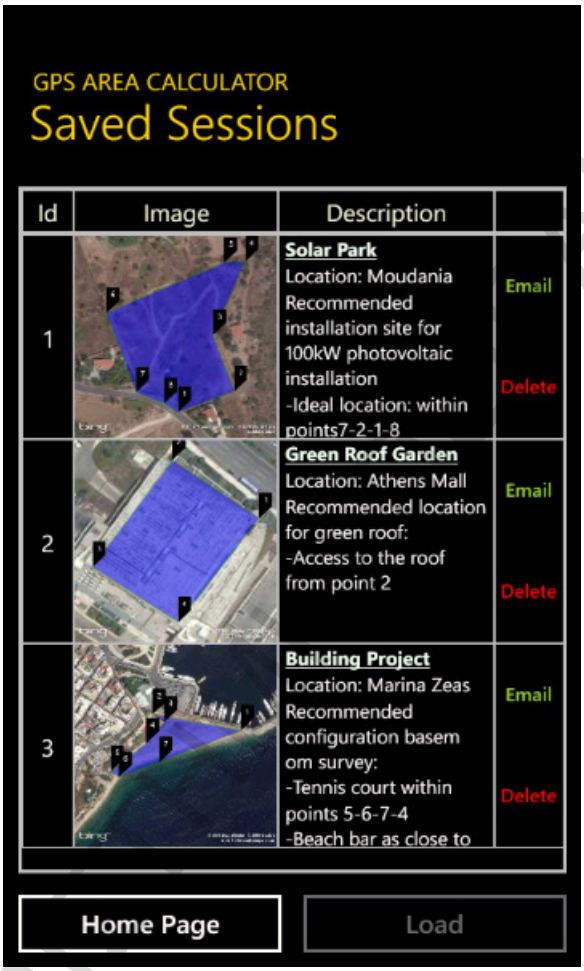


Εικόνα 72: SaveSessionPage.

Με αυτόν το τρόπο, μπορεί να διατηρεί σημειώσεις για τις μετρήσεις του, και να τις στοιχειοθετεί με ανάλογη περιγραφή, υποκαθιστώντας τη χρήση χαρτιού. Ακόμη, η εφαρμογή αποθηκεύσει αυτόματα εικονίδιο που απεικονίζει την καταγεγραμμένη περιοχή, ώστε να παρέχει πληρέστερη εικόνα στο χρήστη.

ΑΠΟΘΗΚΕΥΜΕΝΕΣ ΜΕΤΡΗΣΕΙΣ

Οι παρελθούσες μετρήσεις που έχουν αποθηκευτεί από το χρήστη της εφαρμογής, συνοψίζονται στη σελίδα με τίτλο *Saved Sessions*:



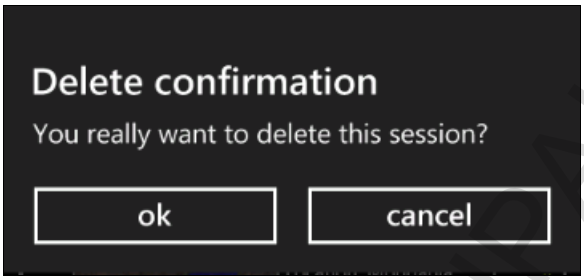
Εικόνα 73: *SavedSessionsPage*.

Οι επιμέρους περιοχές ταξινομούνται σε λίστα με τη σειρά με την οποία καταγράφηκαν. Οι περιγραφές του χρήστη και το εικονίδιο που αποθήκευσε για κάθε περιοχή, βοηθούν το χρήστη να απομνημονεύσει τις συνθήκες υπό τις οποίες διενεργήθηκε η κάθε καταγραφή. Από αυτό το σημείο μπορεί να ενεργήσει ως ακολούθως:

- Να διαγράψει μια μέτρηση
- Να ανοίξει σε πλήρη προβολή μια μέτρηση
- Να αποστείλει με email τα αποτελέσματα της μέτρησης

6.1.7.1ΔΙΑΓΡΑΦΗ ΜΕΤΡΗΣΗΣ

Για κάθε μια από τις αποθηκευμένες καταγραφές περιοχής, δίνεται η δυνατότητα διαγραφής του από τη μνήμη της συσκευής. Πατώντας *Delete* στην επιλεγμένη καταγραφή, εμφανίζεται ο παρακάτω διάλογος επιβεβαίωσης:



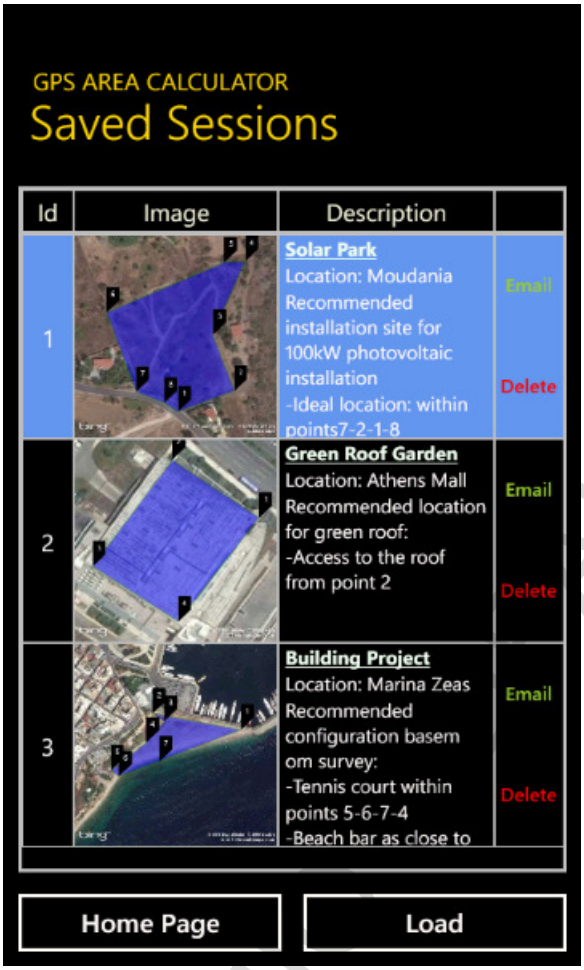
Εικόνα 74: Επιβεβαίωση διαγραφής καταγεγραμένης περιοχής.

Πατώντας *ok* εκτελείται το αίτημά του.

6.1.7.2 ΠΛΗΡΗΣ ΠΡΟΒΟΛΗ ΜΕΤΡΗΣΗΣ

Από τη σελίδα των αποθηκευμένων μετρήσεων, ο χρήστης μπορεί να επιλέξει-ανοίξει σε πλήρη προβολή την επιθυμητή γι' αυτόν καταγραφή. Να σημειωθεί πως τα αποτελέσματα θα προβληθούν με τις αντίστοιχες μετρικές μονάδες και τύπο χάρτη που ήταν επιλεγμένα κατά την αποθήκευση.

Έστω ότι ο χρήστης επιθυμεί να προβάλλει την καταγραφή *#1*. Θα πρέπει να πατήσει επάνω στην αντίστοιχη εγγραφή της λίστας, οπότε η τελευταία θα αποκτήσει μπλε *background* (ως ένδειξη ότι είναι επιλεγμένη) και θα εμφανισθεί η ακόλουθη οθόνη:



Εικόνα 75: *SavedSessionsPage*. Έχει επιλεγεί η καταγραφή *#1*για επεξεργασία.

Στη συνέχεια πατώντας, το κουμπί *Load* γίνεται η πλήρης προβολή της μέτρησης (να σημειωθεί πως το άνοιγμα μέτρησης επιτυγχάνεται εναλλακτικά κάνοντας *double-tap* στην αντίστοιχη εγγραφή που την περιγράφει, παρακάμπτοντας τη χρήση του *Load Button*).

Εικόνα 76: Εμφάνιση επιλεγμένης καταγραφής, με επιλογή από τη λίστα της σελίδας *SavedSessionsPage*.

Στη σελίδα αυτή πραγματοποιείται η ανάκτηση των δεδομένων από τη μνήμη της συσκευής, τα οποία προβάλλονται στο χρήστη με τις ρυθμίσεις προβολής που είχε επιλέξει κατά την αποθήκευσή της καταγραφής.

Η σελίδα αυτή έχει αρκετές ομοιότητες με την *CalculateAreaPage* (Παρ. 6.1.5) με τη διαφορά ότι σαν επικεφαλίδα φέρει τον τίτλο που απέδωσε ο χρήστης στην καταγραφή.

6.1.7.3 ΑΠΟΣΤΟΛΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΜΕ EMAIL

Η τρίτη επιλογή στη σελίδα των αποθηκευμένων μετρήσεων (παρ. 6.1.7), είναι η αποστολή μηνύματος ηλεκτρονικού ταχυδρομείου με τα δεδομένα και αποτελέσματα της μέτρησης. Αυτό επιτυγχάνεται πατώντας *Email* στη γραμμή που περιγράφει την επιθυμητή μέτρηση. Έστω, στο παράδειγμά μας επιλέγουμε τη μέτρηση που μόλις διενεργήσαμε (*#3*). Θα εμφανισθεί η ακόλουθη οθόνη:

Εικόνα 77: *SendEmailPage*.

Ο χρήστης μπορεί να επιλέξει παραλήπτη εναλλακτικά με δύο τρόπους:

- από την αντίστοιχη υπηρεσία που είναι ενσωματωμένη στο λειτουργικό σύστημα, πατώντας *View Contacts*
- Πληκτρολογώντας την επιθυμητή διεύθυνση στο *Textbox* της οθόνης

Έχοντας εκτελέσει μια από τις προηγούμενες μεθόδους, και πατώντας *Send*, αποστέλλεται το παρακάτω ηλεκτρονικό μήνυμα:

TITLE:
Building Project

DESCRIPTION:
Location: Marina Zeas
Recommended configuration basen on survey:
-Tennis court within points 5-6-7-4
-Beach bar as close to point 1 as possible

AREA:
13216 m²

PERIMETER:
675 m

EDGES' LENGTHS:
(1 -> 2): 189 m
(2 -> 3): 30 m
(3 -> 4): 60 m
(4 -> 5): 97 m
(5 -> 6): 21 m
(6 -> 7): 91 m
(7 -> 1): 186 m

GPS POINTS (Latitude, Longitude):
1: (37.930722, 23.648769)
2: (37.931048, 23.646655)
3: (37.930892, 23.646934)
4: (37.930464, 23.64651)
5: (37.929906, 23.645663)
6: (37.929783, 23.64585)
7: (37.930083, 23.646816)

Map display of the above points can be achieved by inserting each one of them (without the parentheses) into the "Search" field of a Bing Maps window and then pressing the Search button.

Εικόνα 78: Παράδειγμα ηλεκτρονικού μηνύματος που αποστέλλει ο χρήστης σε λογαριασμό της επιλογής του, με τα αποτελέσματα μιας καταγραφής.

Η μορφή του απεσταλμένου μηνύματος, αποσκοπεί στο να μεταφέρει στο χρήστη όλα τα διαθέσιμα δεδομένα που σχετίζονται με τη μέτρηση, διατηρώντας έτσι την πληρέστερη δυνατή εικόνα της καταγραφής. Αποτελείται από τις εξής ενότητες:

1. Τον τίτλο της καταγραφής, όπως τον απέδωσε ο χρήστης κατά το σώσιμο.
2. Την περιγραφή της καταγραφής.
3. Το αποτέλεσμα του εμβαδού της υπό διερεύνησης περιοχής, αναγραφόμενη με τις επιλεγμένες (κατά το στάδιο του σωσίματος) μονάδες μέτρησης του εμβαδού.
4. Τη συνολική περίμετρο της περιοχής, με τη μονάδα μήκους της επιλογής του χρήστη.
5. Τα μήκη των επιμέρους ακμών που αναπαριστώνται με τα ζεύγη των κορυφών που τις συνθέτουν. Οι κορυφές αναπαριστώνται με τους αριθμούς που τους αποδόθηκαν κατά την καταγραφή.
6. Οι κορυφές του πολυγώνου της περιοχής με τις συντεταγμένες *GPS*[3].

Γενικότερα, η χρησιμότητα της αποστολής των αποτελεσμάτων μέσω email, άπτεται στο γεγονός ότι οι πληροφορίες μπορούν πλέον να διακινούνται ευρύτερα και να αποθηκεύονται σε εναλλακτικούς προορισμούς, αντί να παραμένουν μόνιμα στη συσκευή που τις κατέγραψε, επιτρέποντας στο χρήστη να τις μοιραστεί και με άλλους χρήστες.

[3] Το API του Windows Phone δεν υποστηρίζει attachments στα απεσταλμένα emails και ως εκ τούτου δεν συμπεριλήφθηκε η εικόνα (αρχείο .png) που αναπαριστά την καταγεγραμμένη περιοχή στο χάρτη.

7 ΠΑΡΑΡΤΗΜΑ: ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

Παραθέτουμε μέρος από τον κώδικα που συγγράφηκε στα πλαίσια της διπλωματικής εργασίας. Ξεκινούμε ορισμένες από τις οντότητες του *Model Layer*, και συνεχίζουμε με το *Presentation Layer*. Για κάθε μία από τις οθόνες που θα παρουσιάσουμε, θα παραθέσουμε τόσο το αρχείο *XAML* όσο και το *code-behind file*.

7.1 MODEL LAYER

APPLICATION CLASS

```
using System.Windows;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using Classes;
using System.Windows.Media.Imaging;
using System.IO.IsolatedStorage;
using System;
using System.Threading;

namespace FrontEnd2
{
    public partial class App : Application
    {
        const string imagesDirectoryName = "sessions_images";
        const string imageCounterKey = "sessionImageCounter";
        public WriteableBitmap tempImageBitmap;
        public GeoData geoData;
        public Units metricUnits;
        public SessionsStorage sessionsStorage;
        public bool LocationServicesEnabled;

        public string generateImageUri()
        {
            int imageRegistry;

Int32.TryParse(IsolatedStorageSettings.ApplicationSettings[imageCounterKey].ToString()
, out imageRegistry);
            imageRegistry++;
            string imageUri = string.Format("{0}/img{1}.jpg", imagesDirectoryName,
imageRegistry);
            IsolatedStorageSettings.ApplicationSettings[imageCounterKey] =
imageRegistry.ToString();
            return imageUri;
        }

        public PhoneApplicationFrame RootFrame { get; private set; }

        public App()
        {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard Silverlight initialization
            InitializeComponent();
```

```
            // Phone-specific initialization
            InitializePhoneApplication();

            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {

                PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
            }

        }

        private void Application_Launching(object sender, LaunchingEventArgs e)
        {
            //Check if the user wishes to have location tracking
            if
(IsolatedStorageSettings.ApplicationSettings.Contains("LocationServicesEnabled"))

Boolean.TryParse(IsolatedStorageSettings.ApplicationSettings["LocationServicesEnabled"
].ToString(), out LocationServicesEnabled);
            else
            {
                IsolatedStorageSettings.ApplicationSettings["LocationServicesEnabled"]
= "true";
                LocationServicesEnabled = true;
            }
            //~Check if the user wishes to have location tracking

            sessionsStorage = new SessionsStorage();

            //The directory that holds the preview images of the stored sessions
            if (!sessionsStorage.iso.DirectoryExists(imagesDirectoryName))
            {
                sessionsStorage.iso.CreateDirectory(imagesDirectoryName);
                IsolatedStorageSettings.ApplicationSettings[imageCounterKey] = "0";
            }

            Thread.Sleep(1200);
        }

        private void Application_Activated(object sender, ActivatedEventArgs e)
        {

        }

        // Code to execute when the application is deactivated (sent to background)
        // This code will not execute when the application is closing
        private void Application_Deactivated(object sender, DeactivatedEventArgs e)
        {
            sessionsStorage.Save();
        }

        // Code to execute when the application is closing (eg, user hit Back)
        // This code will not execute when the application is deactivated
        private void Application_Closing(object sender, ClosingEventArgs e)
        {
            sessionsStorage.Save();
        }

        // Code to execute if a navigation fails
        private void RootFrame_NavigationFailed(object sender,
NavigationFailedEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
```

```
            {
                // A navigation has failed; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        // Code to execute on Unhandled Exceptions
        private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the
splash
            // screen to remain active until the application is ready to render.
            RootFrame = new PhoneApplicationFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender,
NavigationEventArgs e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        #endregion
    }
}
```

Εικόνα 79: *Model Layer: Application class*

GEOTRACKER CLASS

```
using System;
using System.ComponentModel;
using System.Device.Location;
using System.Windows.Data;
using System.Globalization;

namespace Classes
{
    public class GeoTracker : INotifyPropertyChanged
    {
        GeoCoordinateWatcher watcher;

        public void UpdateCurrentStatus()
        {
            CurrentStatus = watcher.Status;
        }

        public GeoPosition<GeoCoordinate> currentPosition;
        public GeoPosition<GeoCoordinate> CurrentPosition
        {
            get
            {
                return currentPosition;
            }
            set
            {
                if (currentPosition != value)
                {
                    currentPosition = value;
                    OnPropertyChanged("CurrentPosition");
                }
            }
        }

        private GeoPositionStatus currentStatus;
        public GeoPositionStatus CurrentStatus
        {
            get
            {
                return currentStatus;
            }
            set
            {
                if (currentStatus != value)
                {
                    currentStatus = value;
                    OnPropertyChanged("CurrentStatus");
                }
            }
        }

        private bool IsActive_PositionChangedEvt;
        private bool IsActive_StatusChangedEvt;

        #region Constructors
        public GeoTracker() : this(true, true) { }

        public GeoTracker(bool isActive_PositionChangedEvt, bool
isActive_StatusChangedEvt)
        {
```

```
            IsActive_PositionChangedEvt = isActive_PositionChangedEvt;
            IsActive_StatusChangedEvt = isActive_StatusChangedEvt;
            Initialize();
        }
        #endregion ~Constructors

        #region Basic Functions
        public void Initialize()
        {
            if (watcher == null)
                watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.High) {
MovementThreshold = 40 };
            if (IsActive_PositionChangedEvt)
                watcher.PositionChanged += new
EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);
            if (IsActive_StatusChangedEvt)
                watcher.StatusChanged += new
EventHandler<GeoPositionStatusChangedEventArgs>(watcher_StatusChanged);
        }

        public void finalize()
        {
            watcher.StatusChanged -= watcher_StatusChanged;
            watcher.PositionChanged -= watcher_PositionChanged;
            if (watcher != null)
            {
                watcher.Dispose();
                watcher = null;
            }
        }

        public void Start()
        {
            watcher.Start();
        }

        public void Stop()
        {
            if (currentStatus != GeoPositionStatus.Disabled)
            {
                currentStatus = GeoPositionStatus.Disabled;
                OnPropertyChanged("CurrentStatus");
                watcher.Stop();
            }
        }

        public event PropertyChangedEventHandler PropertyChanged;
        private void OnPropertyChanged(string propertyName)
        {
            if (this.PropertyChanged != null)
            {
                this.PropertyChanged(this, new
PropertyChangedEventArgs(propertyName));
            }
        }

        #endregion ~Basic Functions

        #region Live Position Tracking
        void watcher_PositionChanged(object sender,
GeoPositionChangedEventArgs<GeoCoordinate> e)
        {
            currentPosition = e.Position;
            OnPropertyChanged("CurrentPosition");
        }
```

```
        void watcher_StatusChanged(object sender, GeoPositionStatusChangedEventArgs e)
        {
            currentStatus = e.Status;
            if ((currentStatus == GeoPositionStatus.Disabled) || (currentStatus ==
GeoPositionStatus.NoData)
                || (currentStatus == GeoPositionStatus.Initializing))
            {
                CurrentPosition = null;
                OnPropertyChanged("CurrentPosition");
            }
            OnPropertyChanged("CurrentStatus");
        }
        #endregion ~Live Position Tracking

        #region Instant Position Tracking
        const int noTracks = 24;
        public void UpdateCurrentPosition()
        {
            double[] tempArray = new double[2];
            for (int i = 0; i < noTracks; i++)
            {
                Stop();
                finalize();
                Initialize();
                Start();
                tempArray[0] += watcher.Position.Location.Latitude;
                tempArray[1] += watcher.Position.Location.Longitude;
            }
            currentPosition.Location = new GeoCoordinate(tempArray[0] / noTracks,
tempArray[1] / noTracks);
            OnPropertyChanged("CurrentPosition");

        }
        #endregion ~Instant Position Tracking
    }

    #region Rendering Classes
    public class StatusConverter : IValueConverter
    {
        public object Convert(object o, Type type, object parameter, CultureInfo
culture)
        {
            GeoPositionStatus status = (GeoPositionStatus)o;
            switch (status)
            {
                case GeoPositionStatus.Initializing:
                    return "Ready";
                case GeoPositionStatus.Ready:
                    return "Ready";
                case GeoPositionStatus.Disabled:
                    return "Disabled";
                case GeoPositionStatus.NoData:
                    return "Server Not Reachable";
            }
            return type.Name;
        }

        public object ConvertBack(object o, Type type, object parameter, CultureInfo
culture)
        {
            return null;
        }
    }
```

```
    public class CoordinateConverter : IValueConverter
    {
        public object Convert(object o, Type type, object coordType, CultureInfo
culture)
        {
            double coordinate = (double)o;
            bool neg = coordinate < 0d;

            // Work with a positive number
            coordinate = Math.Abs(coordinate);

            // d/m/s/t Format
            double d = Math.Floor(coordinate);
            coordinate -= d;
            coordinate *= 60;
            double m = Math.Floor(coordinate);
            coordinate -= m;
            coordinate *= 60;
            double s = Math.Floor(coordinate);
            coordinate -= s;
            coordinate *= 10;
            double t = Math.Round(coordinate);

            // Create padding character
            char pad;
            char.TryParse("0", out pad);
            // Create d/m/s strings
            string dd = d.ToString();
            string mm = m.ToString().PadLeft(2, pad);
            string ss = s.ToString().PadLeft(2, pad);
            string tt = t.ToString();

            // Append d/m/s
            string dms = string.Format("{0}°{1}'{2}.{3}\"", dd, mm, ss, tt);

            // Append compass heading
            switch ((string)coordType)
            {
                case "Latitude":
                    dms += neg ? "S" : "N";
                    break;
                case "Longitude":
                    dms += neg ? "W" : "E";
                    break;
            }
            // Return formated string
            return dms;
        }

        public object ConvertBack(object o, Type type, object parameter, CultureInfo
culture)
        {
            return null;
        }
    }

    public class TimestampConverter : IValueConverter
    {
        public object Convert(object o, Type type, object parameter, CultureInfo
culture)
        {
            return String.Format("{0}", o);

        }
```

```
        public object ConvertBack(object o, Type type, object parameter, CultureInfo
culture)
        {
            return null;
        }
    }
    #endregion
}
```

Εικόνα 80: *Model Layer: GeoTracker class*

GEODATA CLASS

```
using System;
using System.Collections.ObjectModel;
using System.Device.Location;
using System.ComponentModel;
using System.Linq;

namespace Classes
{

    public class GeoData : INotifyPropertyChanged
    {
        private const int METHODS_SWITCHING_POINT = 839935;

        public GeoData()
        {
            GeoPoint.ResetI();
        }

        private double area;
        public double Area
        {
            get
            {
                return area;
            }
            set
            {
                area = value;
            }
        }

        private double perimeter;
        public double Perimeter
        {
            get
            {
                return perimeter;
            }
            set
            {
                perimeter = value;
            }
        }

        public void CalculateArea()
        {
            //At least 3 selected points must exist
            if (SelectedPointsSeq.Count >= 3)
            {
                double calculatedArea =
```

```
EarthGeometer.SphericalPolygonArea2(SelectedPointsSeq.Select(
                    x => EarthGeometer.ToRad(x.Location.Latitude)).ToArray(),
                    SelectedPointsSeq.Select(x =>
EarthGeometer.ToRad(x.Location.Longitude)).ToArray());

                if (calculatedArea < METHODS_SWITCHING_POINT)
                {
                    Area = calculatedArea;
                }
                else
                    Area =
EarthGeometer.SphericalPolygonArea(SelectedPointsSeq.Select(
                    x => EarthGeometer.ToRad(x.Location.Latitude)).ToArray(),
                    SelectedPointsSeq.Select(x =>
EarthGeometer.ToRad(x.Location.Longitude)).ToArray());
                OnPropertyChanged("Area");
            }
        }

        public void CalculatePerimeter()
        {
            if (SelectedPointsSeq.Count >= 3)
            {
                Perimeter = 0;
                for (int i = 0; i < SelectedPointsSeq.Count; i++)
                {
                    GeoPoint geoPoint1 = SelectedPointsSeq.ElementAt(i);
                    GeoPoint geoPoint2 = SelectedPointsSeq.Skip(i +
1).FirstOrDefault() ?? SelectedPointsSeq.First();
                    Perimeter += EarthGeometer.GetDistanceM(
                      geoPoint1.Location.Latitude, geoPoint1.Location.Longitude,
                        geoPoint2.Location.Latitude, geoPoint2.Location.Longitude);
                }
                OnPropertyChanged("Perimeter");
            }
        }

        private ObservableCollection<GeoPoint> selectedPointsSeq = new
ObservableCollection<GeoPoint>();
        public ObservableCollection<GeoPoint> SelectedPointsSeq
        {
            get
            {
                return selectedPointsSeq;
            }
            set
            {
                selectedPointsSeq = value;
            }
        }

        public void Add(GeoCoordinate location)
        {
            SelectedPointsSeq.Add(new GeoPoint(location));
        }

        public void RemoveItem(int objectId)
        {
            if (objectId - 1 < SelectedPointsSeq.Count)
            {
                GeoPoint.DecrementI();
                for (int i = objectId; i < SelectedPointsSeq.Count; i++)
                    SelectedPointsSeq[i].Id--;
                SelectedPointsSeq.RemoveAt(objectId - 1);
            }
```

```
        }

        public class GeoPoint : INotifyPropertyChanged
        {
            private static int i = 0;
            private int id;
            private GeoCoordinate location { get; set; }

            public GeoPoint(GeoCoordinate geoCoordinate)
            {
                i++;
                Id = i;
                location = geoCoordinate;
            }

            public int Id
            {
                get
                {
                    return id;
                }
                set
                {
                    id = value;
                    OnPropertyChanged("Id");
                }
            }

            public GeoCoordinate Location
            {
                get
                {
                    return location;
                }
                set
                {
                    location = value;
                    OnPropertyChanged("Location");

                }
            }

            public static void DecrementI()
            {
                i--;
            }

            public static void ResetI()
            {
                i = 0;
            }

            public event PropertyChangedEventHandler PropertyChanged;
            private void OnPropertyChanged(string propertyName)
            {
                if (this.PropertyChanged != null)
                {
                    this.PropertyChanged(this, new
PropertyChangedEventArgs(propertyName));
                }
            }
        }

        public static GeoCoordinate GenerateRandomLocation()
        {
```

```
            return new GeoCoordinate(new Random().Next(0, 90), new Random().Next(0,
180));
        }

        public event PropertyChangedEventHandler PropertyChanged;
        private void OnPropertyChanged(string propertyName)
        {
            if (this.PropertyChanged != null)
            {
                this.PropertyChanged(this, new
PropertyChangedEventArgs(propertyName));
            }
        }

        public void Reset()
        {
            Area = 0;
            Perimeter = 0;
            GeoPoint.ResetI();
            selectedPointsSeq.Clear();
        }
    }

}
```

Εικόνα 81: *Model Layer: GeoData class*

7.2 PRESENTATION LAYER

MAINPAGE

```
<phone:PhoneApplicationPage
    x:Class="FrontEnd2.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True" Loaded="PhoneApplicationPage_Loaded">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">

        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Image Margin="0,-30,0,0" Grid.RowSpan="2" Source="Landfield.png"
Stretch="UniformToFill" />

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="GPS AREA CALCULATOR"
Style="{StaticResource PhoneTextNormalStyle}" FontSize="25" Foreground="Gold" />
            <TextBlock x:Name="PageTitle" Text="Home Page" Margin="9,-7,0,0"
```

```
Style="{StaticResource PhoneTextTitle1Style}" FontSize="50" Foreground="Gold"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" >
            <Grid.RowDefinitions>
                <RowDefinition Height="*"></RowDefinition>
                <RowDefinition Height="*"></RowDefinition>
                <RowDefinition Height="*"></RowDefinition>
            </Grid.RowDefinitions>
            <Button Style="{StaticResource ButtonStyleThemeUnaware}" Foreground="Snow"
BorderBrush="Snow" Content="Create New Session" Height="72"  Name="button1"
Click="button1_Click" Grid.Row="0" Margin="0,100,0,0"/>
            <Button Style="{StaticResource ButtonStyleThemeUnaware}" Foreground="Snow"
BorderBrush="Snow" Content="View Saved Sessions" Height="72" Name="button2"
Click="button2_Click"  Grid.Row="1"/>
            <Button Style="{StaticResource ButtonStyleThemeUnaware}" Foreground="Snow"
BorderBrush="Snow" Content="About GPS Area Calculator" Height="72" Name="button3"
Click="button3_Click"  Grid.Row="2" Margin="0,-100,0,0"  />

            <Popup Grid.RowSpan="3" Name="popupAbout" IsOpen="False">
                <Border  Padding="5,0,5,0" BorderBrush="White" BorderThickness="2"
Background="Black" Height="629" Width="456" Margin="12,0">
                    <Grid>
                        <Grid.RowDefinitions>
                            <RowDefinition Height="*"/>
                            <RowDefinition Height="Auto"/>
                        </Grid.RowDefinitions>

                        <ScrollViewer VerticalScrollBarVisibility="Visible"  >
                            <StackPanel Grid.Row="0" >
                                <TextBlock FontSize="24" Foreground="Honeydew"
TextDecorations="Underline">Overview</TextBlock>
                                <TextBlock Foreground="Snow" TextWrapping="Wrap">
GPS Area Calculator enables the easy calculation of the critical attributes that
describe land plots, along with the display, storage and delivery of the related
information. Assuming that the plot of study forms a polygon, the user has to record
the GPS signal at the vertices of this polygon, and the app evaluates its area,
perimeter and length of every edge. The land measured, is displayed on the map, and
there is flexibility regarding the map view mode (aerial or road) and the form of the
numerical results (area and distance metric units). The user is given the option to
store data in the device along with personal notes, and share it via email.
                                </TextBlock>
                                <TextBlock FontSize="24" Foreground="Honeydew"
TextDecorations="Underline" Margin="0,15,0,0">Accuracy</TextBlock>
                                <TextBlock Foreground="Snow" TextWrapping="Wrap">
Like any scientific instrument, GPS Area Calculator derives results that ...
                                </TextBlock>
                                <TextBlock FontSize="24" Foreground="Honeydew"
TextDecorations="Underline" Margin="0,15,0,0">Version</TextBlock>
                                <TextBlock Foreground="Snow" TextWrapping="Wrap">1.1
                                </TextBlock>

                                <TextBlock FontSize="24" Foreground="Honeydew"
TextDecorations="Underline" Margin="0,15,0,0">Privacy Policy</TextBlock>
                                <TextBlock Foreground="Snow" TextWrapping="Wrap">
                                    Location information is not at any time disclosed
to any "Third Party". Sharing is only possible through the emailing option which is
available at the user’s discretion.
                                </TextBlock>

                                <CheckBox Checked="CheckBox_Checked"
Unchecked="CheckBox_Unchecked" Name="theCheckbox" IsChecked="{Binding }">
                                    <TextBlock Foreground="Beige" TextWrapping="Wrap">
                                    Location Tracking Enabled
```

```
                                    </TextBlock>
                                </CheckBox>

                                <TextBlock FontSize="24" Foreground="Honeydew"
TextDecorations="Underline" Margin="0,15,0,0">Contact</TextBlock>
                                <TextBlock Foreground="Snow" TextWrapping="Wrap">
                                We would appreciate your feedback regarding the
application, and we are always open to new ideas, so please don't hesitate to contact
us:
                                </TextBlock>
                                <HyperlinkButton Foreground="LightBlue"
Content="dimevag@hotmail.com"  Click="HyperlinkButton_Click"></HyperlinkButton>

                                <TextBlock FontSize="24" Foreground="Honeydew"
TextDecorations="Underline" Margin="0,15,0,0">Acknowledgments</TextBlock>
                                <TextBlock Foreground="Snow" TextWrapping="Wrap"
Margin="0,0,0,50">
                                Special thanks to Andreas Kontarinis and Ioannis
Kapranos for their invaluable support.
                                </TextBlock>

                            </StackPanel>
                        </ScrollViewer>
                        <Button  Grid.Row="1" Style="{StaticResource
ButtonStyleThemeUnaware}" Foreground="Snow" BorderBrush="Snow" Content="Back"
Height="72" Width="173"  Name="btnOK" Click="btnOK_Click" />
                    </Grid>
                </Border>
            </Popup>
        </Grid>
    </Grid>
</phone:PhoneApplicationPage>
```

Εικόνα 82: *Presentation Layer: MainPage (XAML)*

```
using System;
using System.Linq;
using System.Windows;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Tasks;
using System.IO.IsolatedStorage;

namespace FrontEnd2
{
    public partial class MainPage : PhoneApplicationPage
    {
        public bool LocationServicesEnabled;

        public MainPage()
        {
            InitializeComponent();

        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new Uri("/Page2.xaml", UriKind.Relative));
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            while (this.NavigationService.BackStack.Any())
```

```
            {
                this.NavigationService.RemoveBackEntry();
            }

            LocationServicesEnabled = (Application.Current as
App).LocationServicesEnabled;
            theCheckbox.DataContext = LocationServicesEnabled;
        }

        private void button2_Click(object sender, RoutedEventArgs e)
        {
            string url = string.Format("/Page4.xaml?SessionEntry={0}", "false");
            NavigationService.Navigate(new Uri(url, UriKind.Relative));
        }

        private void button3_Click(object sender, RoutedEventArgs e)
        {
            popupAbout.IsOpen = true;
        }

        private void btnOK_Click(object sender, RoutedEventArgs e)
        {
            popupAbout.IsOpen = false;
        }

        private void HyperlinkButton_Click(object sender, RoutedEventArgs e)
        {

            var task = new EmailComposeTask
            {
                Subject = "User Feedback (Source app: GPS Area Calculator v1.1)",
                To = "dimevag@hotmail.com"
            };
            task.Show();
        }


        protected override void OnBackKeyPress(System.ComponentModel.CancelEventArgs
e)
        {
            if (popupAbout.IsOpen)
            {
                popupAbout.IsOpen = false;
                e.Cancel = true;
            }
            base.OnBackKeyPress(e);
        }

        private void CheckBox_Checked(object sender, RoutedEventArgs e)
        {
            //            MessageBox.Show((theCheckbox.IsChecked).ToString());
            MessageBox.Show((LocationServicesEnabled).ToString());
            (Application.Current as App).LocationServicesEnabled = true;
            IsolatedStorageSettings.ApplicationSettings["LocationServicesEnabled"] =
"true";
        }

        private void CheckBox_Unchecked(object sender, RoutedEventArgs e)
        {
            //          MessageBox.Show((theCheckbox.IsChecked).ToString());
            MessageBox.Show((LocationServicesEnabled).ToString());
            (Application.Current as App).LocationServicesEnabled = false;
            IsolatedStorageSettings.ApplicationSettings["LocationServicesEnabled"] =
"false";
        }
```

```
    }
}
```

Εικόνα 83: *Presentation Layer: MainPage (code-behind)*

DEFINEPOINTSPAGE

```
<phone:PhoneApplicationPage
    x:Class="FrontEnd2.Page2"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True"
    xmlns:Classes="clr-namespace:Classes;assembly=Classes"
Loaded="PhoneApplicationPage_Loaded" Unloaded="PhoneApplicationPage_Unloaded"
    xmlns:local="clr-namespace:FrontEnd2"
    >

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Grid.Resources>
            <local:ButtonStatusConverter x:Key="ButtonStatusConverter"/>
            <Classes:CoordinateConverter x:Key="coordinateConverter" />
            <Classes:StatusConverter x:Key="statusConverter" />

            <DataTemplate x:Key="theTemplate">
                <ListBoxItem Margin="0,0,0,0" >
                    <StackPanel Orientation="Horizontal" >

                        <Border BorderBrush="Silver" BorderThickness="1"
Padding="5,0,5,0" Width="56">
                            <TextBlock Text="{Binding Id}"
HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="Snow"/>
                        </Border>
                        <Border BorderBrush="Silver" BorderThickness="1"
Padding="5,0,5,0" Width="150">
                            <TextBlock  Text="{Binding Location.Latitude,
Converter={StaticResource coordinateConverter}, ConverterParameter=Latitude}"
HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="LightBlue"/>
                        </Border>
                        <Border BorderBrush="Silver" BorderThickness="1"
Padding="5,0,5,0" Width="150">
                            <TextBlock Text="{Binding Location.Longitude,
Converter={StaticResource coordinateConverter}, ConverterParameter=Longitude}"
HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="LightBlue"/>
                        </Border>
                        <Border BorderBrush="Silver" BorderThickness="1" Padding="0"
```

```
Width="100">
                            <!--<Button BorderThickness="0" Content="X"
HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="Red"
Click="ButtonIntoList_Click"/>-->
                            <Button  Style="{StaticResource ButtonStyleThemeUnaware}"
Margin="-18,-2,-12,-2" HorizontalAlignment="Center" VerticalAlignment="Center"
Name="btnDetete" BorderThickness="0" FontSize="16"  Click="ButtonIntoList_Click">
                                <TextBlock Margin="-8,0,-8,0" Text="Delete"
Foreground="Red" />
                            </Button>
                        </Border>

                    </StackPanel>
                </ListBoxItem>
            </DataTemplate>

        </Grid.Resources>
        <Image Margin="0,-30,0,0" Grid.RowSpan="5" Source="BlackBackground.png"
Stretch="UniformToFill" Visibility="{StaticResource PhoneLightThemeVisibility}" />
        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="GPS AREA CALCULATOR"
Style="{StaticResource PhoneTextNormalStyle}" Foreground="Gold"/>
            <TextBlock x:Name="PageTitle" Text="Define Boundary Points" Margin="9,-
7,0,0" Style="{StaticResource PhoneTextTitle1Style}" FontSize="40" Foreground="Gold"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Grid Name="tableCurrentPosition" >
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="137"/>
                    <ColumnDefinition Width="137"/>
                </Grid.ColumnDefinitions>
                <Grid.RowDefinitions>
                    <RowDefinition />
                    <RowDefinition />
                    <RowDefinition />
                </Grid.RowDefinitions>

                <Border BorderBrush="Silver" BorderThickness="0,0,1,1"
Grid.Column="0" Grid.Row="0">
                </Border>
                <Border BorderBrush="Silver" BorderThickness="0,1,1,1"
Grid.Column="1" Grid.Row="0">
                    <TextBlock Margin="3" HorizontalAlignment="Center"
Name="txtBlockColumnLatitude" Text="Latitude" VerticalAlignment="Top"
Foreground="Beige" />
                </Border>
                <Border BorderBrush="Silver" BorderThickness="0,1,1,1"
Grid.Column="2" Grid.Row="0">
                    <TextBlock Margin="3"  HorizontalAlignment="Center"
Name="txtBlockColumnLongitude" Text="Longitude" VerticalAlignment="Top"
Foreground="Beige"/>
                </Border>
                <Border BorderBrush="Silver" BorderThickness="1,0,1,1"  Grid.Row="1"
Grid.Column="0">
                    <TextBlock Margin="3"  HorizontalAlignment="Center"
Name="textBlock1" Text="Current Location" VerticalAlignment="Top" Foreground="Beige"
/>
                </Border>
                <Border BorderBrush="Silver" BorderThickness="0,0,1,1"  Grid.Row="1"
Grid.Column="1">
                    <TextBlock Margin="3"  HorizontalAlignment="Center"
```

```
Name="txtBlockCurrentLocationLatitude" Text="{Binding
Path=CurrentPosition.Location.Latitude, FallbackValue=-, Converter={StaticResource
coordinateConverter}, ConverterParameter=Latitude}" VerticalAlignment="Top"
Foreground="Yellow"/>
                </Border>
                <Border BorderBrush="Silver" BorderThickness="0,0,1,1"  Grid.Row="1"
Grid.Column="2">
                    <TextBlock Margin="3"  HorizontalAlignment="Center"
Name="txtBlockCurrentLocationLongitude" Text="{Binding
Path=CurrentPosition.Location.Longitude, FallbackValue=-, Converter={StaticResource
coordinateConverter}, ConverterParameter=Longitude}" VerticalAlignment="Top"
Foreground="Yellow" />
                </Border>
                <Border BorderBrush="Silver" BorderThickness="1,0,1,1"  Grid.Row="2"
Grid.Column="0" >
                    <TextBlock Margin="3"  HorizontalAlignment="Center"
Name="txtBlockGPSStatusLabel" Text="Status" Foreground="Beige" />
                </Border>
                <Border BorderBrush="Silver" BorderThickness="0,0,1,1"  Grid.Row="2"
Grid.Column="1" Grid.ColumnSpan="2" >
                    <Grid>
                        <TextBlock Margin="3"  HorizontalAlignment="Center"
Name="txtBlockGPSStatus" Text="{Binding Path=CurrentStatus, Converter={StaticResource
statusConverter}}" Foreground="DarkKhaki" />
                        <TextBlock Margin="3"  HorizontalAlignment="Center"
Name="txtBlockGPSStatusTracking" Text="Tracking (hold your position)" Foreground="Red"
Visibility="Collapsed"/>
                    </Grid>
                </Border>
            </Grid>
        </Grid>

        <Border Margin="12,92,12,6" Grid.Row="2" BorderBrush="Silver"
BorderThickness="3" >
            <ListBox  Name="SelectedGPSLocationsView" ItemTemplate="{StaticResource
theTemplate}" Margin="0,-2,0,8">
            </ListBox>
        </Border>

        <Button Style="{StaticResource ButtonStyleThemeUnaware}" Foreground="Snow"
BorderBrush="Snow" Grid.Row="3" Content="Add Current Location" Height="72"
HorizontalAlignment="Left"  Name="button1" VerticalAlignment="Top" Width="243"
FontSize="20" Click="button1_Click" IsEnabled="{Binding Text,
ElementName=txtBlockGPSStatus, Converter={StaticResource ButtonStatusConverter},
FallbackValue=False}" />
        <Button Style="{StaticResource ButtonStyleThemeUnaware}" Foreground="Snow"
BorderBrush="Snow" Grid.Row="3" Content="Preview on Map" Height="72" Name="button2"
VerticalAlignment="Top" FontSize="20" Click="button2_Click" Margin="237,0,0,0" />

        <Button Style="{StaticResource ButtonStyleThemeUnaware}" Foreground="Snow"
BorderBrush="Snow" Content="Calculate Area" Grid.Row="3" Height="72"
HorizontalAlignment="Left"  Name="calculateAreaButton" VerticalAlignment="Top"
Width="480" FontSize="26" Click="button3_Click" Margin="0,60,0,0" Grid.RowSpan="2" />
        <Border BorderBrush="Silver" BorderThickness="3,3,3,0" Height="40"
Margin="12,0,12,361" Grid.Row="2" VerticalAlignment="Bottom">
            <TextBlock Margin="-3,-3,-3,0" TextWrapping="Wrap" Text="Selected
Locations" HorizontalAlignment="Center" VerticalAlignment="Center"
Foreground="NavajoWhite" FontSize="22" />
        </Border>
        <Border BorderBrush="Silver" BorderThickness="3,3,3,0" Height="48"
Margin="12,51,12,0" Grid.Row="2" VerticalAlignment="Top"></Border>
        <StackPanel Orientation="Horizontal" Margin="14,121,16,249"
Grid.Row="2"></StackPanel>
        <Border BorderBrush="Silver" BorderThickness="1" Padding="5,3,5,3"
Margin="14,53,409,320" Grid.Row="2">
```

```
            <TextBlock Text="Id" HorizontalAlignment="Center"
VerticalAlignment="Center" Foreground="Beige" />
        </Border>
        <Border BorderBrush="Silver" BorderThickness="1" Padding="5,3,5,3" Width="150"
Margin="71,53,259,320" Grid.Row="2">
            <TextBlock Text="Latitude" HorizontalAlignment="Center"
VerticalAlignment="Center" Foreground="Beige" />
        </Border>
        <Border BorderBrush="Silver" BorderThickness="1" Padding="5,3,5,3" Width="150"
Margin="221,53,109,320" Grid.Row="2">
            <TextBlock Text="Longitude" HorizontalAlignment="Center"
VerticalAlignment="Center" Foreground="Beige" />
        </Border>
        <Border BorderBrush="Silver" BorderThickness="1" Padding="0"
Margin="371,53,12,320" Grid.Row="2" />
    </Grid>
</phone:PhoneApplicationPage>
```

Εικόνα 84: *Presentation Layer: DefinePointsPage (XAML)*

```
using System;
using System.Windows;
using System.Windows.Controls;
using Microsoft.Phone.Controls;
using Classes;
using System.Device.Location;
using System.Windows.Navigation;
using System.Windows.Data;
using System.Globalization;

namespace FrontEnd2
{
    public partial class DefinePointsPage  : PhoneApplicationPage
    {
        App a = Application.Current as App;

        public GeoTracker geoTracker = new GeoTracker();

        GeoData geoData;
        Units metricUnits;
        bool LocationServicesEnabled;


        public Page2()
        {
            InitializeComponent();

            txtBlockCurrentLocationLatitude.DataContext = geoTracker;
            txtBlockCurrentLocationLongitude.DataContext = geoTracker;
            txtBlockGPSStatus.DataContext = geoTracker;

            a.geoData = new GeoData();
            geoData = a.geoData;
            geoData.Reset();

            a.metricUnits = new Units();
            metricUnits = a.metricUnits;
            metricUnits.Reset();

            SelectedGPSLocationsView.ItemsSource = geoData.SelectedPointsSeq;
        }
```

```
        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            //Current Location display
            geoTracker.Initialize();
            geoTracker.Start();

            LocationServicesEnabled = a.LocationServicesEnabled;
        }

        protected override void
OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            if (!e.IsNavigationInitiator)
            {
                geoTracker.Initialize();
                geoTracker.Start();
            }
        }


        protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
        {
            if (!e.IsNavigationInitiator)
            {
                geoTracker.Stop();
                geoTracker.finalize();
            }
            base.OnNavigatingFrom(e);

        }

        protected override void OnBackKeyPress(System.ComponentModel.CancelEventArgs
e)
        {
            try
            {
                if (geoData.SelectedPointsSeq.Count > 0)
                {
                    if (MessageBox.Show("Backwards navigation will cause loss of
current data. Proceed?", "Quit confirmation", MessageBoxButton.OKCancel) ==
MessageBoxResult.Cancel)
                    {
                        e.Cancel = true;
                    }
                }
                base.OnBackKeyPress(e);

            }
            catch (Exception)
            {
                e.Cancel = true;
            }

        }

        private void PhoneApplicationPage_Unloaded(object sender, RoutedEventArgs e)
        {
            geoTracker.Stop();
            geoTracker.finalize();
        }

        private void ButtonIntoList_Click(object sender, RoutedEventArgs e)
        {
            Button button = (Button)sender;
            Border border = (Border)button.Parent;
```

```
            StackPanel stackPanel = (StackPanel)border.Parent;
            ListBoxItem listboxItem = (ListBoxItem)stackPanel.Parent;

            if (MessageBox.Show("You really want to delete this location?", "Delete
confirmation", MessageBoxButton.OKCancel) == MessageBoxResult.OK)
            {
                int index = ((Classes.GeoData.GeoPoint)listboxItem.DataContext).Id;
                geoData.RemoveItem(index);
            }
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            if (LocationServicesEnabled)
            {
                if ((geoTracker.CurrentStatus == GeoPositionStatus.Ready) ||
(geoTracker.CurrentStatus == GeoPositionStatus.Initializing))
                {
                    button1.IsEnabled = false;
                    txtBlockGPSStatus.Visibility = Visibility.Collapsed;
                    txtBlockGPSStatusTracking.Visibility = Visibility.Visible;
                    Dispatcher.BeginInvoke(() =>
                   {
                        geoTracker.UpdateCurrentPosition();
                        geoData.Add(geoTracker.currentPosition.Location);

SelectedGPSLocationsView.ScrollIntoView(SelectedGPSLocationsView.Items[SelectedGPSLoca
tionsView.Items.Count - 1]);
                        txtBlockGPSStatusTracking.Visibility = Visibility.Collapsed;
                        txtBlockGPSStatus.Visibility = Visibility.Visible;
                        button1.IsEnabled = true;
                   });
                }
                else if (geoTracker.CurrentStatus == GeoPositionStatus.NoData)
                    MessageBox.Show("GPS signal is low. Try recording from an
alternative location.", "System notification", MessageBoxButton.OK);
                else
                    MessageBox.Show("GPS receiver is inactive. Check your device
settings.", "System notification", MessageBoxButton.OK);
            }
            else
                MessageBox.Show("Location tracking is disabled. To enable it go to
\"About GPS Area Calculator\" under the \"Privacy Policy\" section.", "System
notification", MessageBoxButton.OK);
        }

        private void button2_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new Uri("/Page2a.xaml", UriKind.Relative));
        }

        private void button3_Click(object sender, RoutedEventArgs e)
        {
            if (geoData.SelectedPointsSeq.Count < 3)
                MessageBox.Show("You must record at least three GPS points to
calculate the area.", "System notification", MessageBoxButton.OK);
            else
                NavigationService.Navigate(new Uri("/Page3.xaml", UriKind.Relative));
        }
    }

    public class ButtonStatusConverter : IValueConverter
    {
        public object Convert(object o, Type type, object parameter, CultureInfo
culture)
```

```
        {
            string status = (string)o;
            if (status == "Ready")
                return true;
            else if ((status == "Disabled") || (status == "Server Not Reachable"))
                return false;
            return false;

        }

        public object ConvertBack(object o, Type type, object parameter, CultureInfo
culture)
        {
            return null;
        }
    }
}
```

Εικόνα 85: *Presentation Layer: DefinePointsPage (code-behind)*

CALCULATEAREAPAGE

```
<phone:PhoneApplicationPage
    x:Class="FrontEnd2.Page3"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True"
    xmlns:my="clr-
namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps"
Loaded="PhoneApplicationPage_Loaded"
    xmlns:Classes="clr-namespace:Classes;assembly=Classes"
    x:Name="Root">

    <Grid x:Name="LayoutRoot" Background="Transparent" >
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>

        </Grid.RowDefinitions>

        <Grid.Resources>
            <my:ApplicationIdCredentialsProvider ApplicationId="AhwhcRkC-
SOyz8i_fSU3pXZxer9sIBQiS_qfefmpQsVfPnraVZuwMv_Hb-MGBBtC" x:Key="MapCredentials" />
            <!-- PUSHPIN TEMPLATE -->
            <DataTemplate x:Key="PinTemplate">
                <my:Pushpin Content="{Binding Id}" Location="{Binding Location}"
MouseLeftButtonDown="Pushpin_MouseLeftButtonDown"/>
            </DataTemplate>
            <!-- ~PUSHPIN TEMPLATE -->

            <Classes:EnumBooleanConverter x:Key="enumBooleanConverter" />
```

```
        </Grid.Resources>
        <Image Margin="0,-30,0,0" Grid.RowSpan="4" Source="BlackBackground.png"
Stretch="UniformToFill" Visibility="{StaticResource PhoneLightThemeVisibility}" />
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,12">
            <TextBlock x:Name="ApplicationTitle" Text="GPS AREA CALCULATOR"
Style="{StaticResource PhoneTextNormalStyle}" Foreground="Gold"/>
            <TextBlock x:Name="PageTitle" Text="Results" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextTitle1Style}" FontSize="40" Foreground="Gold"  />
        </StackPanel>

        <!-- MAP WITH CONTENTS -->
        <Border Grid.Row="1" BorderThickness="3" BorderBrush="Silver" Margin="12,0">
            <my:Map Name="SubjectMap" CredentialsProvider="{StaticResource
MapCredentials}" >
                <my:MapLayer Name="mapLayer" >
                    <my:MapPolygon Name="AreaMapPolygon" Fill="Blue" Stroke="Green"
StrokeThickness="2" Opacity="0.5" Visibility="Collapsed" />
                    <my:MapItemsControl Name="mapPushpins" ItemsSource="{Binding
SelectedPointsSeq}" ItemTemplate="{StaticResource PinTemplate}" />
                    <Popup Name="popupSelectedPushpinInfo" Grid.Row="1" >
                        <Border CornerRadius="6" BorderBrush="Black"
BorderThickness="2" Background="#CAE5E5E5" Tap="Border_Tap" >
                            <StackPanel >
                                <TextBlock Foreground="Black" FontWeight="Bold"
Name="txtBlockPushpinName" Margin="4,0,4,0" />
                                <TextBlock Foreground="Black"
Name="txtBlockPushpinDetails" Margin="4,0,4,-20" />
                            </StackPanel>
                        </Border>
                    </Popup>
                </my:MapLayer>

            </my:Map>
        </Border>

        <!-- ~MAP WITH CONTENTS -->

        <Popup Name="popupNumericalResultsFormat" Grid.Row="1" Grid.RowSpan="2"
IsOpen="False" Margin="12,0,12,0">
            <Border BorderBrush="White" BorderThickness="2" Background="Black"
Height="594" Width="456" >
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition  Height="Auto"/>
                        <RowDefinition Height="*"/>
                        <RowDefinition Height="Auto"/>
                    </Grid.RowDefinitions>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition />
                    </Grid.ColumnDefinitions>

                    <TextBlock Grid.Row="0" Grid.ColumnSpan="2"
HorizontalAlignment="Center" FontSize="28" Margin="0,0,0,24" Foreground="NavajoWhite"
Text="View Preferences" />

                    <Grid Grid.Row="1">
                        <Grid.RowDefinitions>
                            <RowDefinition Height="*"/>
                            <RowDefinition Height="*"/>
                            <RowDefinition Height="Auto"/>
                        </Grid.RowDefinitions>

                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="244"/>
```

```
                        <ColumnDefinition Width="*"/>
                    </Grid.ColumnDefinitions>

                    <StackPanel x:Name="stackPanelRadioButtons" Grid.Row="2"
Grid.ColumnSpan="2" Orientation="Horizontal" Height="90" >
                        <TextBlock FontSize="24" Margin="6,0,0,0"
Foreground="Beige" VerticalAlignment="Center" Width="180">Map Mode:</TextBlock>
                        <RadioButton Style="{StaticResource
RadioButtonStyleThemeUnaware}" Name="radioButton2" GroupName="radioButtonGroup"
Foreground="Snow" Width="130" IsChecked="{Binding Path=SelectedMapMode,
Converter={StaticResource enumBooleanConverter}, ConverterParameter=Aerial,
Mode=OneWay}" >Aerial</RadioButton>
                        <RadioButton Style="{StaticResource
RadioButtonStyleThemeUnaware}" Name="radioButton1" GroupName="radioButtonGroup"
Foreground="Snow" Width="130" IsChecked="{Binding Path=SelectedMapMode,
Converter={StaticResource enumBooleanConverter}, ConverterParameter=Road,
Mode=OneWay}" >Road</RadioButton>
                    </StackPanel>

                    <TextBlock Grid.Row="0" Grid.Column="0" FontSize="24"
Margin="6,0,0,0" Foreground="Beige">Area expressed in:</TextBlock>
                    <ScrollViewer Grid.Row="0" Grid.Column="1" >
                        <ListBox FontSize="24" Foreground="Snow"
SelectedIndex="{Binding SelectedAreaUnit, Mode=TwoWay}"
x:Name="listBoxSelectedAreaUnit" >
                            <ListBoxItem Margin="0,0,0,10" >Square
Meters</ListBoxItem>
                            <ListBoxItem Margin="0,0,0,10" >Acres</ListBoxItem>
                            <ListBoxItem Margin="0,0,0,10" >Hectares</ListBoxItem>
                            <ListBoxItem Margin="0,0,0,10" >Square
Kilometers</ListBoxItem>
                        </ListBox>
                    </ScrollViewer>

                    <TextBlock Grid.Row="1" Grid.Column="0" FontSize="24"
Margin="6,0,0,0" Foreground="Beige">Distance expressed in:</TextBlock>
                    <ScrollViewer Grid.Row="1" Grid.Column="1">
                        <ListBox FontSize="24" Foreground="Snow"
SelectedIndex="{Binding SelectedDistanceUnit, Mode=TwoWay}"
x:Name="listBoxSelectedDistanceUnit" >
                            <ListBoxItem Margin="0,0,0,10" >Feet</ListBoxItem>
                            <ListBoxItem Margin="0,0,0,10" >Meters</ListBoxItem>
                            <ListBoxItem Margin="0,0,0,10"
>Kilometers</ListBoxItem>
                            <ListBoxItem Margin="0,0,0,10" >Miles</ListBoxItem>
                        </ListBox>
                    </ScrollViewer>
                </Grid>
                <Button Style="{StaticResource ButtonStyleThemeUnaware}"
Foreground="Snow" BorderBrush="Snow" Content="OK" Grid.Row="2" Height="72"
HorizontalAlignment="Left"  Name="btnSavePreferences" Width="220" FontSize="22"
Click="btnSavePreferences_Click" />
                <Button Style="{StaticResource ButtonStyleThemeUnaware}"
Foreground="Snow" BorderBrush="Snow" Content="Cancel" Grid.Row="2" Height="72"
HorizontalAlignment="Right"  Name="btnCancel" Width="220" FontSize="22"
Click="btnCancel_Click" />
            </Grid>
        </Border>
    </Popup>

    <Grid x:Name="NumericalResultsPanel" Grid.Row="2" Margin="12,12,12,0">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="126"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="80"/>
```

```
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition />
                <RowDefinition Height="34" />
                <RowDefinition />
            </Grid.RowDefinitions>

            <Border BorderBrush="Silver" BorderThickness="1,1,1,1" Margin="0,-1,0,1">
                <TextBlock Margin="3"  HorizontalAlignment="Center"
Name="txtBlockAreaLabel" Text="Area" VerticalAlignment="Top" Foreground="Beige" />
            </Border>
            <Border BorderBrush="Silver" BorderThickness="0,1,1,1"  Grid.Column="1"
Margin="0,-1,0,1">
                <TextBlock Margin="3" HorizontalAlignment="Center"
Name="txtBlockAreaValue" VerticalAlignment="Top" Foreground="Yellow" />
            </Border>
            <Border BorderBrush="Silver" BorderThickness="1,0,1,1" Margin="0,34,0,1"
Grid.RowSpan="2">
                <TextBlock Margin="3"  HorizontalAlignment="Center"
Name="txtBlockPerimeterLabel" Text="Perimeter" VerticalAlignment="Top"
Foreground="Beige" />
            </Border>
            <Border BorderBrush="Silver" BorderThickness="0,0,1,1" Grid.Column="1"
Margin="0,34,0,1" Grid.RowSpan="2">
                <TextBlock Margin="3"  HorizontalAlignment="Center"
Name="txtBlockPerimeterValue"  VerticalAlignment="Top" Foreground="Yellow"/>
            </Border>
            <Border BorderBrush="Silver" BorderThickness="0,1,1,1" Grid.Column="2"
Grid.RowSpan="2" Margin="0,-1,0,1">
                <Button BorderBrush="Snow" Margin="-9" BorderThickness="1"
Click="Button_Click" Opacity="0.5" Style="{StaticResource ButtonStyleThemeUnaware}">
                    <TextBlock FontSize="16" Foreground="Snow" TextAlignment="Center"
VerticalAlignment="Center" HorizontalAlignment="Center" Margin="-3" Text="Change
Prefs"  Name="txtBlockChangeUnits"  TextWrapping="Wrap"  />
                </Button>

            </Border>
        </Grid>

        <Grid Grid.Row="3">
            <TextBlock HorizontalAlignment="Center"
x:Name="txtblck_regIndic">HAHA</TextBlock>
        </Grid>
        <Button Style="{StaticResource ButtonStyleThemeUnaware}" Foreground="Snow"
BorderBrush="Snow" Content="Home Page" Grid.Row="4" Height="72"
HorizontalAlignment="Left"  Name="btnBackToMainPage" VerticalAlignment="Top"
Width="243" FontSize="22" Click="btnBackToMainPage_Click_1" />
        <Button Style="{StaticResource ButtonStyleThemeUnaware}" Foreground="Snow"
BorderBrush="Snow" Content="Save Results" Height="72" HorizontalAlignment="Left"
Margin="237,0,0,0" Name="btnStoreResults" VerticalAlignment="Top" Width="243"
Grid.Row="4" FontSize="22" Click="btnStoreResults_Click" />
    </Grid>

</phone:PhoneApplicationPage>
```

Εικόνα 86: *Presentation Layer: CalculateAreaPage (XAML)*

```
using System;
using System.Linq;
using System.Windows;
using System.Windows.Input;
using Microsoft.Phone.Controls;
```

```
using Classes;
using Microsoft.Phone.Controls.Maps;
using System.Windows.Media.Imaging;
using System.Windows.Media;
using FrontEnd2.ServiceReference1;
using System.Device.Location;
namespace FrontEnd2
{
    public partial class CalculateAreaPage : PhoneApplicationPage
    {
        App a = Application.Current as App;

        GeoData geoData;
        Units metricUnits;
        SessionsStorage sessionsStorage;

        GeocodeServiceClient geocodeService = null;
        String ApplicationId = "AhwhcRkC-
SOyz8i_fSU3pXZxer9sIBQiS_qfefmpQsVfPnraVZuwMv_Hb-MGBBtC";

        public Page3()
        {
            InitializeComponent();

            geocodeService = new
GeocodeServiceClient("BasicHttpBinding_IGeocodeService");
            geocodeService.ReverseGeocodeCompleted += new
EventHandler<ReverseGeocodeCompletedEventArgs>(geocodeService_ReverseGeocodeCompleted)
;
        }

        private void geocodeService_ReverseGeocodeCompleted(object sender,
ReverseGeocodeCompletedEventArgs e)
        {
            GeocodeResponse geocodeResponse = e.Result;
            if (geocodeResponse.Results.Count > 0)
            {
                txtblck_regIndic.Text = geocodeResponse.Results[0].DisplayName;
            }
            else
            {
                txtblck_regIndic.Text = "N/A";
            }
        }

        private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
        {
            geoData = a.geoData;
            metricUnits = a.metricUnits;
            sessionsStorage = a.sessionsStorage;

            //Intialize map and Preferences popup
            SubjectMap.DataContext = geoData;

            if (metricUnits.SelectedMapMode == MapModes.Road)
                SubjectMap.Mode = new RoadMode();
            else
                SubjectMap.Mode = new AerialMode();

            LocationCollection mapPolygonLocations =
formatGeoDataToLocationCollection();
            AreaMapPolygon.Locations = mapPolygonLocations;
            AreaMapPolygon.Visibility = Visibility.Visible;

            stackPanelRadioButtons.DataContext = metricUnits;
```

```
            //~Intialize map

            //Calculate Geo Results
            geoData.CalculateArea();
            geoData.CalculatePerimeter();
            //~Calculate Geo Results

            //Display Numerical results
            listBoxSelectedAreaUnit.DataContext = metricUnits;
            listBoxSelectedDistanceUnit.DataContext = metricUnits;

            txtBlockAreaValue.Text = metricUnits.FormatAreaResult(geoData.Area);
            txtBlockPerimeterValue.Text =
metricUnits.FormatDistanceResult(geoData.Perimeter);
            //~Display Numerical results


SubjectMap.SetView(LocationRect.CreateLocationRect(mapPolygonLocations.ToArray()));
            ReverseGeocodeRequest reverseGeocodeRequest = new ReverseGeocodeRequest();
            reverseGeocodeRequest.Credentials = new Credentials();
            reverseGeocodeRequest.Credentials.ApplicationId = ApplicationId;
            reverseGeocodeRequest.Location =
geoData.SelectedPointsSeq.Last().Location;
            geocodeService.ReverseGeocodeAsync(reverseGeocodeRequest);
        }

        private LocationCollection formatGeoDataToLocationCollection()
        {
            LocationCollection locationCollection = new LocationCollection();
            foreach (Classes.GeoData.GeoPoint geoPoint in a.geoData.SelectedPointsSeq)
                locationCollection.Add(geoPoint.Location);
            return locationCollection;
        }

        int prevAreaUnit;
        int prevDistanceUnit;
        Pushpin selectedPushpin;
        MapModes prevMapMode;

        private void btnCancel_Click(object sender, RoutedEventArgs e)
        {
            if (listBoxSelectedAreaUnit.SelectedIndex != prevAreaUnit)
                listBoxSelectedAreaUnit.SelectedIndex = prevAreaUnit;
            if (listBoxSelectedDistanceUnit.SelectedIndex != prevDistanceUnit)
                listBoxSelectedDistanceUnit.SelectedIndex = prevDistanceUnit;

            if ((radioButton1.IsChecked == true) && (prevMapMode != MapModes.Road))
                radioButton2.IsChecked = true;
            else if ((radioButton2.IsChecked == true) && (prevMapMode !=
MapModes.Aerial))
                radioButton1.IsChecked = true;

            popupNumericalResultsFormat.IsOpen = false;
            btnBackToMainPage.IsEnabled = true;
            btnStoreResults.IsEnabled = true;
        }

        private void btnSavePreferences_Click(object sender, RoutedEventArgs e)
        {
            if (listBoxSelectedAreaUnit.SelectedIndex != prevAreaUnit)
                txtBlockAreaValue.Text = metricUnits.FormatAreaResult(geoData.Area);
            if (listBoxSelectedDistanceUnit.SelectedIndex != prevDistanceUnit)
            {
                txtBlockPerimeterValue.Text =
metricUnits.FormatDistanceResult(geoData.Perimeter);
```

```
                if (popupSelectedPushpinInfo.IsOpen == true)
                {
                    UpdatePopupSelectedPushpinInfo(selectedPushpin);
                }
            }

            if ((radioButton1.IsChecked == true) && (prevMapMode == MapModes.Aerial))
            {
                metricUnits.SelectedMapMode = MapModes.Road;
                SubjectMap.Mode = new RoadMode();
            }
            else if ((radioButton2.IsChecked == true) && (prevMapMode ==
MapModes.Road))
            {
                metricUnits.SelectedMapMode = MapModes.Aerial;
                SubjectMap.Mode = new AerialMode();
            }

            popupNumericalResultsFormat.IsOpen = false;
            btnBackToMainPage.IsEnabled = true;
            btnStoreResults.IsEnabled = true;
        }

        private void Pushpin_MouseLeftButtonDown(object sender, MouseButtonEventArgs
e)
        {
            Pushpin pushpin = (Pushpin)sender;
            if ((pushpin == selectedPushpin) && (popupSelectedPushpinInfo.IsOpen))
            {
                popupSelectedPushpinInfo.IsOpen = false;
                pushpin.Background = new SolidColorBrush(Colors.Black);
            }
            else
            {
                UpdatePopupSelectedPushpinInfo(pushpin);
                if (selectedPushpin != null)
                    selectedPushpin.Background = new SolidColorBrush(Colors.Black);
                pushpin.Background = new SolidColorBrush(Colors.Purple);
                MapLayer.SetPosition(popupSelectedPushpinInfo, pushpin.Location);
                selectedPushpin = pushpin;
                popupSelectedPushpinInfo.IsOpen = true;
            }
        }

        private void UpdatePopupSelectedPushpinInfo(Pushpin pushpin)
        {
            int selectedArrIndex = (int)(pushpin).Content - 1;
            int prevArrIndex = (selectedArrIndex == 0) ?
geoData.SelectedPointsSeq.Count - 1 : selectedArrIndex - 1;
            int nextArrIndex = (selectedArrIndex == (geoData.SelectedPointsSeq.Count -
1)) ? 0 : selectedArrIndex + 1;

            double distance_prev = EarthGeometer.GetDistanceM(

geoData.SelectedPointsSeq.ElementAt(selectedArrIndex).Location.Latitude,

geoData.SelectedPointsSeq.ElementAt(selectedArrIndex).Location.Longitude,
                geoData.SelectedPointsSeq.ElementAt(prevArrIndex).Location.Latitude,
                geoData.SelectedPointsSeq.ElementAt(prevArrIndex).Location.Longitude
                );

            double distance_next = EarthGeometer.GetDistanceM(

geoData.SelectedPointsSeq.ElementAt(selectedArrIndex).Location.Latitude,
```

```
geoData.SelectedPointsSeq.ElementAt(selectedArrIndex).Location.Longitude,
                geoData.SelectedPointsSeq.ElementAt(nextArrIndex).Location.Latitude,
                geoData.SelectedPointsSeq.ElementAt(nextArrIndex).Location.Longitude
                );

            txtBlockPushpinName.Text = string.Format("Point {0}:", selectedArrIndex +
1);
            txtBlockPushpinDetails.Text = string.Format("Distance ({0} -> {1}):
{2}\n", prevArrIndex + 1, selectedArrIndex + 1,
metricUnits.FormatDistanceResult(distance_prev));
            txtBlockPushpinDetails.Text += string.Format("Distance ({0} -> {1}):
{2}\n", selectedArrIndex + 1, nextArrIndex + 1,
metricUnits.FormatDistanceResult(distance_next));
        }

        private void Border_Tap(object sender, GestureEventArgs e)
        {
            selectedPushpin.Background = new SolidColorBrush(Colors.Black);
            popupSelectedPushpinInfo.IsOpen = false;
        }

        protected override void OnBackKeyPress(System.ComponentModel.CancelEventArgs
e)
        {
            if (popupNumericalResultsFormat.IsOpen)
            {
                btnCancel_Click(new object(), new RoutedEventArgs());
                e.Cancel = true;
            }
            else
                if (popupSelectedPushpinInfo.IsOpen)
                {
                    popupSelectedPushpinInfo.IsOpen = false;
                    e.Cancel = true;
                }
            base.OnBackKeyPress(e);
        }

        private void btnStoreResults_Click(object sender, RoutedEventArgs e)
        {

            if (popupSelectedPushpinInfo.IsOpen)
            {
                Pushpin_MouseLeftButtonDown(selectedPushpin, null);

            }
            a.tempImageBitmap = new WriteableBitmap(SubjectMap, null);

            NavigationService.Navigate(new Uri("/Page5.xaml", UriKind.Relative));
        }

        private void btnBackToMainPage_Click_1(object sender, RoutedEventArgs e)
        {

            NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            prevAreaUnit = metricUnits.SelectedAreaUnit;
            prevDistanceUnit = metricUnits.SelectedDistanceUnit;
            prevMapMode = metricUnits.SelectedMapMode;

            btnBackToMainPage.IsEnabled = false;
            btnStoreResults.IsEnabled = false;
```

```
            popupNumericalResultsFormat.IsOpen = true;
        }

    }
}
```

Εικόνα 87: *Presentation Layer: CalculateAreaPage (code-behind)*

8 ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

9 ΒΙΒΛΙΟΓΡΑΦΙΑ

1.Nick Randlolf, Christopher Fairbairn (2011). ***Professional Windows Phone 7 Application Development***. Indiana: Wiley.

BIBLIOGRAPHY \l 1033 2. Gary McLean Hall (2010)**.** ***Pro WPF and Silverlight MVVM - Effective Application Development with Model-View-ViewModel****.* New York: Apress.

BIBLIOGRAPHY \l 1033 3. Fablo Claudio Feracchiati, Emmanuele Garofalo (2010)**.** ***Windows Phone 7 Recipes****.* New York: Apress.

BIBLIOGRAPHY \l 1033 4. Mamta Dalal, Ashish Ghoda (2011)**.** ***XAML Developer*** *Reference.* California: O' Reily.

BIBLIOGRAPHY \l 1033 5. Sam Noble, Sam Burton, Allen Jones (2008)**.** ***WPF Recipes in C#****.* New York: Apress.

BIBLIOGRAPHY \l 1033 6. Leonard Richardson, Sam Ruby (2007)**.** ***RESTful Web Services****.* California: O' Reily.

BIBLIOGRAPHY \l 1033 7. Martin Fowler, Kendal Scott (1999)**.** *UML* ***Distilled: A Brief GUIde to the Standar*** *Object .* Boston: Addison Wesley.

BIBLIOGRAPHY \l 1033 8. Faraz Rasheed (2006)**.** ***C# School****.* Fueng*UI*rola, Spain: Synchron Data.

BIBLIOGRAPHY \l 1033 9. *HTTP*://www.ibm.com/developerworks/webservices/library/ws-*REST*ful. ***RESTful Web Services***. Developer Works. 06 November 2008.

BIBLIOGRAPHY \l 1033 10. *HTTP*://msdn.*Microsoft*.com/en-us/library/w9fdtx28(v=*asp*net.11).*asp*x. ***XML Web Services OverView***. *Microsoft*-Developer Network.

BIBLIOGRAPHY \l 1033 11. *HTTP*://www.trimble.com/*GPS*_tutorial/why*GPS*.*asp*x. ***Trimble GPS Tutorial***. Trimble.

BIBLIOGRAPHY \l 1033 12. *HTTP*://msdn.*Microsoft*.com/en-US/library/windowsphone/develop/hh184843(v=vs.105).*asp*x. ***App Certification Req****UI****rements for Windows Phone****. Microsoft*-Dev Center.