Πανεπιστήμιο Πειραιώς

Τμήμα Ψηφιακών Συστημάτων

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ

Κατεύθυνση: **Προηγμένα Πληροφοριακά Συστήματα**

Ανάπτυξη εφαρμογής για έξυπνα κινητά για το σχεδιασμό και την παρακολούθηση στρατιωτικών επιχειρήσεων σε πραγματικό χρόνο «PERSEUS»

.Σπυρίδων Παναγιωτόπουλος ,

ΑΜ: ΜΕ1748

Μεταπτυχιακή Διπλωματική Εργασία

Επιβλέπων: Δημοσθένης Κυριαζής, Επίκουρος Καθηγητής

Πειραιάς 2019

Περίληψη

Η εφαρμογή στην οποία αναφέρεται το παρών έγγραφο αποτελεί την διπλωματική εργασία η οποία πραγματοποιήθηκε στα πλαίσια του μεταπτυχιακού προγράμματος Πανεπιστημίου Πειραιώς, Τμήμα Ψηφιακών Συστημάτων και Κατεύθυνση «Προηγμένα Πληροφοριακά Συστήματα».

Το θέμα της εργασίας καθορίστηκε από την ΓΕΣ/ΔΕΠΛΗ στα πλαίσια της μετεκπαίδευσης μου, και αφορά την ανάπτυξη εφαρμογής για κινητά τηλέφωνα που θα παρέχει την δυνατότητα σχεδίασης και παρακολούθησης ασκήσεων και επιχειρήσεων, σε καιρό ειρήνης και καιρό πολέμου αντίστοιχα. Η εφαρμογή αυτή μπορεί να λειτουργήσει αυτόνομα ως εργαλείο σχεδίασης επί χάρτη αλλά και ως πλήρη εφαρμογή επιχειρησιακής σχεδίασης και διεξαγωγής της αποστολής.

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε περιβάλλον Android Studio 3.1.4 για την κινητή εφαρμογή, IntelliJ Idea 2017.3.3 για την ανάπτυξη του Εξυπηρετητή, MariaDB για την βάση δεδομένων, CouchBase Lite για την αποθήκευση δεδομένων εντός της συσκευής καθώς και το περιβάλλον Github για έλεγχο εκδόσεων και ενσωμάτωσης της εφαρμογής.

Τέλος, το σχεδιαστικό κομμάτι της εφαρμογής βασίστηκε στο Extensible Map Platform Android Development Kit, ώστε να παρέχεται διαλειτουργικότητα με τα νέα και προς ανάπτυξη συστήματα του Στρατού Ξηράς αλλά και του ΝΑΤΟ.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου κ. Δημοσθένη Κυριαζή για την εμπιστοσύνη που έδειξε, αναθέτοντάς μου την διπλωματική αυτή εργασία, καθώς και για τις συμβουλές του κατά την εκπόνησή της. Επίσης, θα ήθελα να ευχαριστήσω θερμά τον μηχανικό λογισμικού και ερευνητή κ. Ανδρέα Μενύχτα για την πολύτιμη βοήθεια και καθοδήγηση που μου παρείχε, αναφορικά με τα πιο τεχνικά κομμάτια της διπλωματικής. Τέλος, θα ήθελα να ευχαριστήσω τον κ. Σιγάλα Ιωάννη, για την καθοδήγηση του σε θέματα επιτελικού ενδιαφέροντος ως προς την εφαρμογή και τον κ. Γεώργιο Μέτση, για την βοήθεια στην επίτευξη διαλειτουργικότητας μεταξύ των εφαρμογών μας.

Πίνακας Περιεχομένων

Λίστα Σχημάτων

1. Εισαγωγή

1.1 Ιστορικό Υπόβαθρο

Σύμφωνα με τον Κλαούσεβιτς στο βιβλίο του «Για τον Πόλεμο» [1], ο Πόλεμος είναι μια πράξη βίας που έχει ως σκοπό την επιβολή της θέλησης μας επί του αντιπάλου μας. Και επί αυτού του σκοπού, η βία εξοπλίζεται με τις εφευρέσεις της τέχνης και της επιστήμης για να αντιμετωπίσει την βία.

Αν και έχουν περάσει πάνω από 140 χρόνια από την έκδοση του βιβλίου του, ο ορισμός αυτός είναι απόλυτα αληθής τώρα όσο ήταν και τότε. Ο σκοπός του πολέμου δεν έχει αλλάξει, αλλά αυτό που έχει αλλάξει είναι ο τρόπος διεξαγωγής των πολεμικών επιχειρήσεων.

Στην αυγή του Α' Παγκοσμίου Πολέμου [2] η τεχνολογία είχε αρχίσει να προοδεύει με γοργούς ρυθμούς, γεγονός που οδήγησε σε ραγδαίες εξελίξεις κατά την διάρκεια των επιχειρήσεων. Οι πολύχρωμες και εντυπωσιακές στολές των Ναπολεόντειων πολέμων έδωσαν θέση στις μονόχρωμες και πρακτικές φόρμες, που χρησιμοποιούνται με διάφορες παραλλαγές ακόμα και σήμερα. Τα άριστα παρατεταμένα τμήματα που έβαλαν μεταξύ τους σε κοντινή απόσταση έδωσαν τόπο στον πόλεμο των χαρακωμάτων, τα οποία με την σειρά τους κατέρρευσαν με την είσοδο των πρώιμων Βρετανικών αρμάτων στο πεδίο της μάχης. Είναι χαρακτηριστικό ότι η αδυναμία της ηγεσίας των Γαλλικών ενόπλων δυνάμεων να αντιληφθεί τις εξελίξεις οδήγησε σε δραματικές απώλειες στην αρχή του πολέμου εναντίων των Γερμανικών δυνάμεων, οι οποίες είχαν ακολουθήσει τις εξελίξεις. Αλλά και με την σειρά τους οι Γερμανοί βρέθηκαν απροετοίμαστοι να αντιμετωπίσουν τα βρετανικά άρματα, τα οποία με σχετική ευκολία διέσπασαν τα χαρακώματα.

Στον Β' Παγκόσμιο Πόλεμο[3] η ανάγκη για τεχνολογική υπεροχή ήταν φανερή στις μεγάλες δυνάμεις που έλαβαν μέρος. Πέραν από τους εξοπλισμούς και τα όπλα που εφευρέθηκαν, με αποκορύφωμα την πυρηνική βόμβα, μεγάλη βαρύτητα έπεσε στις επικοινωνίες.

Τα ηλεκτρονικά ήρθαν στο προσκήνιο κατά τον Β' Παγκόσμιο Πόλεμο. Ευρέως γνωστό παράδειγμα [4] είναι η δημιουργία από τους Βρετανούς του προγόνου του σύγχρονου ηλεκτρονικού υπολογιστή για την αποκρυπτογράφηση των Γερμανικών κωδικών "Enigma".

Όμως οι εφευρέσεις δεν σταμάτησαν εκεί. Πλήθος πρώιμων ψηφιακών υπολογιστών χρησιμοποιήθηκαν, κυρίων για τον υπολογισμό πινάκων και καθορισμό τροχιάς βλημάτων πυροβολικού. Παράλληλα, τα πρώτα Radar αναπτύχθηκαν και άλλαξαν τον τρόπο διεξαγωγής των επιχειρήσεων, ενώ η σμίκρυνση της τεχνολογίας οδήγησε στην δυνατότητα να φέρει ο απλός στρατιώτης ασύρματες συσκευές επικοινωνίας με την ηγεσία. Τέλος, η υποκλοπή και παρεμβολή των πληροφοριών αυτών απέκτησε τεράστια σημασία, οδηγώντας στην ανάπτυξη συσκευών που αναλάμβαναν αυτές τις λειτουργίες.

Στην σύγχρονη εποχή πλέον, ορισμένες χώρες έχουν την δυνατότητα να εξολοθρεύουν ολόκληρες περιοχές χωρίς να έχουν καν φυσική παρουσία, με την πηγή της ισχύος να βρίσκεται εκατοντάδες χιλιόμετρα μακριά. Αυτό καθίσταται δυνατόν χάρις στην παροχή πληροφοριών που εξασφαλίζονται από σύγχρονες συσκευές και μεθόδους. Κυριολεκτικά, με το πάτημα ενός κουμπιού.

Η πληροφορία πάντα υπήρξε ένας από τους σημαντικότερους παράγοντες νίκης στο πεδίο της μάχης. Αυτό δεν άλλαξε στην σύγχρονη εποχή, άλλα αυτό που άλλαξε είναι η ταχύτητα μετάδοσης, ο όγκος και η ποιότητα των πληροφοριών και η ταχύτητα αξιολόγησης τους. Δορυφορικές εικόνες, δημοσιεύσεις, σχόλια, αναμεταδόσεις, οποιαδήποτε πληροφορία έχει πλέον αξία.

Η σχεδόν στιγμιαία μετάδοση των πληροφοριών έχει μεταβάλλει, πιθανώς μόνιμα, το σύγχρονο πεδίο των επιχειρήσεων. Ειδήσεις πραγματικού χρόνου, μέσα κοινωνικής δικτύωσης και το Internet παρέχουν την δυνατότητα για ταχεία ανταλλαγή πληροφοριών, καθιστώντας ακόμα και απλούς πολίτες έμμεσα συμμετέχοντες στις πολεμικές συγκρούσεις. Με δυνατότητα αναζήτησης πολλαπλών πηγών, εγχώρια ή από το εξωτερικό, φιλτράροντας μη ενδιαφέροντες ειδήσεις και επικεντρώνοντας στα σημαντικά, η εύρεση πληροφοριών δεν υπήρξε ποτέ ευκολότερη.

Εκεί που παλαιότερα ομάδες ανθρώπων έψαχναν κάθε πληροφορία ξεχωριστά, υπολογιστικά δίκτυα εργάζονται ακούραστα, συλλέγοντας πληροφορίες από πηγές στο διαδίκτυο και κατηγοριοποιώντας τις με τέτοια ταχύτητα, που καθιστά ακόμα και σχετικά ασήμαντες πληροφορίες χρήσιμες, λόγω χρονικής συγκυρίας.

Καμία άλλη συσκευή δεν ταιριάζει τόσο πολύ σε αυτή την νέα πραγματικότητα από το κινητό τηλέφωνο, μια συσκευή η οποία βρίσκεται στα χέρια των περισσοτέρων ανθρώπων σήμερα. Το κινητό τηλέφωνο. Συσκευή με τρομερές δυνατότητες παραμετροποίησης, καθώς παρέχει την δυνατότητα στον χρήστη να επιλέγει τις εφαρμογές που επιθυμεί και έχει ανάγκη. Επιτρέπει την στιγμιαία επικοινωνία, τόσο ως τηλέφωνο αλλά και ως πλατφόρμα εκτέλεσης εφαρμογών κοινωνικής δικτύωσης. Επίσης, επιτρέπει την εύκολη λήψη φωτογραφιών, βίντεο και ήχου, καθώς και την διανομή τους. Εκτιμάται ότι [5] ο στρατιώτης της ψηφιακής εποχής θα εκμεταλλευτεί στρατιωτικά κινητά τηλέφωνα προσαρμοσμένα για τις ανάγκες της αποστολής, και θα βασίζεται σε ένα δίκτυο ειδικών, που θα παρέχουν υποστήριξη από τα μετόπισθεν.

Με την χρήση αυτών των συσκευών, ο στρατιώτης θα μπορεί να έχει βελτιωμένη και ενημερωμένη επίγνωση της κατάστασης, και πληροφορίες πραγματικού χρόνου, βελτιώνοντας τις πιθανότητες επιτυχούς ολοκλήρωσης της αποστολής. Παράλληλα εκτιμάται ότι θα επιτρέψει στις ένοπλες δυνάμεις να εκτελούν στοχευμένες και υψηλής ακρίβειας αποστολές, μειώνοντας τον απαιτούμενο χρόνο και το κόστος μιας αποστολής.

Το σύγχρονο περιβάλλον μάχης είναι ταχέως εξελισσόμενο, με όπλα μεγάλου βεληνεκούς και αεροκίνητες δυνάμεις να μειώνουν το εμπόδιο της απόστασης, ενώ οι ασύμμετρες απειλές απαιτούν ταχεία δυνατότητα αντίδρασης και ευελιξία. Αυτό καθιστά προφανές ότι η ταχύτατη μετάδοση της πληροφορίας είναι κρίσιμη, και οποιαδήποτε αδυναμία των Ενόπλων Δυνάμεων μιας χώρας να ανταποκριθεί σε αυτόν τον τομέα μπορεί να αποβεί καταστροφική.

1.2 Στρατιωτικό Υπόβαθρο

1.2.1 **Πληροφορίες**

Οι πληροφορίες διαδραματίζουν κρίσιμο ρόλο στις επιχειρήσεις, και η απαίτηση για πληροφορίες είναι διαρκής. Παρακάτω παρατίθενται τα βασικά χαρακτηριστικά, μέθοδοι και ορισμοί που χρησιμοποιούνται στην στρατιωτική ανάλυση και επεξεργασία των πληροφοριών.

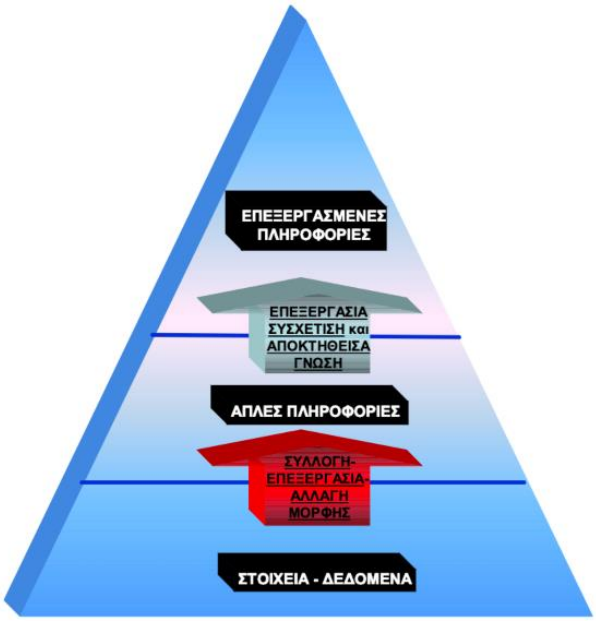
1.2.2 **Πηγές Πληροφοριών**

- **Πηγές από Ακουστικά Μέσα Συλλογής**
- **Πηγές από Ανθρώπινη Δραστηριότητα**
- **Πληροφορίες από Εικόνες,** είτε προέρχονται από φωτογραφίες, δορυφορική εικόνα, εικόνα ραντάρ, ηλεκτροπτικές και θερμικές συσκευές.
- **Πληροφορίες από Επικοινωνίες του Εχθρού,** που προέρχονται από υποκλοπές στα επικοινωνιακά συστήματα του εχθρού.
- **Πληροφορίες Ηλεκτρονικών Εκπομπών,** που προέρχονται από ηλεκτρονικές εκπομπές εκτός αυτών που προέρχονται από επικοινωνίες και εκπομπές ραντάρ.
- **Πληροφορίες από Ραντάρ**
- **Πληροφορίες από Ανοικτές Πηγές,** όπως μέσα μαζικής ενημέρωσης και το διαδίκτυο.
- **Πληροφορίες Τεχνολογίας,** που αφορούν ξένες τεχνολογικές εξελίξεις και επιχειρησιακές δυνατότητες υλικού με πρακτική στρατιωτική εφαρμογή.

1.2.3 **Κατηγοριοποίηση Πληροφοριών**

- **Απλές πληροφορίε**ς είναι τα μη επεξεργασμένα στοιχεία που προέρχονται από οποιαδήποτε πηγή. Ειδικά στο επιχειρησιακό περιβάλλον, ο μεγαλύτερος όγκος πληροφοριών είναι απλές, και έγκειται στο Διοικητή να προσδιορίσει ποιες από αυτές τις πληροφορίες ανήκουν στις Κρίσιμες Πληροφοριακές Απαιτήσεις του, με βάσει τις οποίες θα συνταχθεί το επιχειρησιακό του σχέδιο.
- **Επεξεργασμένες πληροφορίες** είναι τα προϊόντα που προέκυψαν από την επεξεργασία απλών πληροφοριών και αφορούν ξένα κράτη, εχθρικές ή ενδεχόμενα εχθρικές δυνάμεις ή περιοχές πραγματικών ή ενδεχομένων επιχειρήσεων.

Παρακάτω απεικονίζεται η σχέση μεταξύ στοιχείων, απλών πληροφοριών και επεξεργασμένων πληροφοριών.



Εικόνα 1 Πληροφορίες

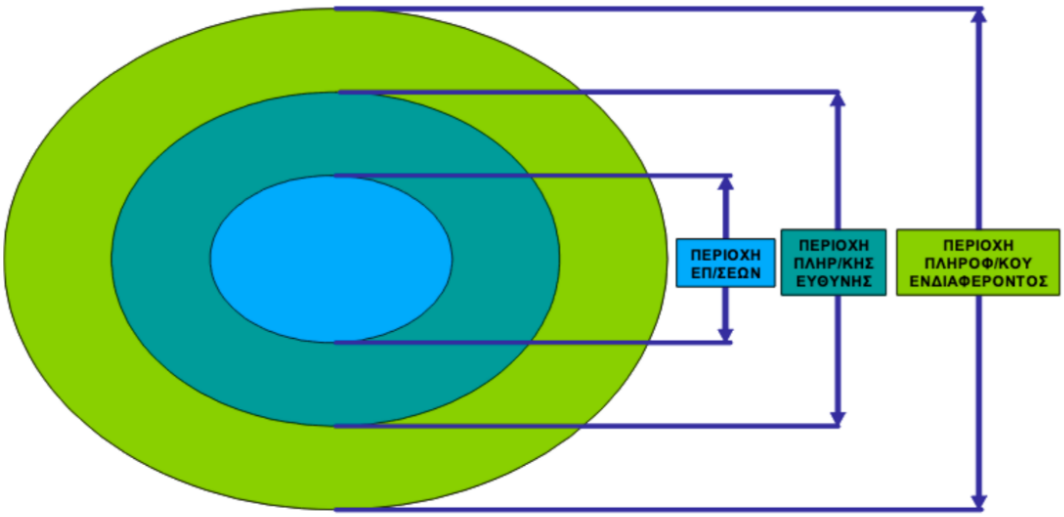
1.2.4 **Τύποι Επεξεργασμένων Πληροφοριών:**

- **Βασικές Πληροφορίες:** Διατηρούνται σε βάση δεδομένων και ανανεώνονται διαρκώς σε καιρό ειρήνης. Αφορούν κυρίων αμετάβλητα δεδομένα, όπως π.χ. το έδαφος, και χρησιμοποιούνται για την προετοιμασία μιας επιχείρησης. Αποτελούν πληροφορίες γενικού και μόνιμου ενδιαφέροντος.
- **Τρέχουσες Πληροφορίες:** Χρησιμοποιούνται για την κάλυψη πληροφοριακών αναγκών τρεχουσών επιχειρήσεων και αναφέρονται σε τρέχουσα κατάσταση, η οποία χαρακτηρίζεται από σημαντικές και ταχείες αλλαγές, που απαιτούν συνεχή ανάλυση και εκτίμηση.
- **Πληροφορίες Στόχου:** Οι πληροφορίες που εντοπίζουν βασικά στοιχεία ενός στόχου ή συμπλέγματος στόχων και δείχνουν την σχετική σπουδαιότητα και τρωτότητα του.

1.2.5 **Κατηγοριοποίηση Περιοχών**

- **Περιοχή Επιχειρήσεων:** Η περιοχή στην οποία ο Διοικητής είναι υπεύθυνος να αναπτύξει τα τμήματα του για την επιτυχή εκτέλεση της αποστολής του.
- **Περιοχή Πληροφοριακής Ευθύνης:** Ορίζεται ως η περιοχή στην οποία ο Διοικητής είναι υπεύθυνος να χρησιμοποιήσει τα μέσα του για να παράγει επεξεργασμένες πληροφορίες. Καθορίζεται με βάση τις δυνατότητες και τα διαθέσιμα μέσα του τμήματος του. Σε αυτήν την περιοχή ορίζεται η κύρια προσπάθεια του επιτελείου του.
- **Περιοχή Πληροφοριακού Ενδιαφέροντος:** Ορίζεται ως η περιοχή για την οποία ο Διοικητής χρειάζεται επεξεργασμένες πληροφορίες για τις εξελίξεις που πιθανώς επηρεάσουν το αποτέλεσμα τρεχουσών ή μελλοντικών επιχειρήσεων. Συνήθως δεν είναι δυνατόν να αποκτήσει αυτές τις πληροφορίες με τα δικά του μέσα, οπότε πρέπει το επιτελείο του να απευθυνθεί στα γειτονικά ή φίλια κλιμάκια για αυτές.

Η σχέση μεταξύ των περιοχών μπορεί να απεικονισθεί απλά με το παρακάτω σχήμα:



Εικόνα 2 Περιοχές

1.2.6 **Κύκλος των Πληροφοριών**

Ο κύκλος πληροφοριών είναι η διαδικασία μέσα από την οποία αξιοποιούνται οι απλές πληροφορίες, ώστε αυτές να καταστούν το ταχύτερο δυνατόν επεξεργασμένες. Χωρίζεται σε τέσσερα διακριτά στάδια:

- **Διεύθυνση**, η οποία περιλαμβάνει τον καθορισμό των πληροφοριακών απαιτήσεων, τον σχεδιασμό της προσπάθειας συλλογής και την έκδοση διαταγών και απαιτήσεων στα όργανα συλλογής.
- **Συλλογή**, κατά την οποία τα όργανα συλλογής εκμεταλλεύονται τις πηγές και διακινούν τις απλές πληροφορίες στον κατάλληλο φορέα επεξεργασίας.
- **Επεξεργασία,** στην οποία οι απλές πληροφορίες που έχουν συλλεχθεί από τα προηγούμενα στάδια μετατρέπονται σε επεξεργασμένες πληροφορίες.
- **Εκμετάλλευση,** στην οποία γίνεται εκτίμηση και έγκαιρη διάθεση των επεξεργασμένων πληροφοριών, σε κατάλληλη μορφή και με οποιοδήποτε κατάλληλο μέσο σε όσους τις χρειάζονται.

Η παραπάνω ακολουθία είναι κυκλική, καθώς οι πληροφορίες απαιτούν συνεχή επανεκτίμηση και ανανέωση ώστε να παραμείνουν επίκαιρες.

1.2.7 **Μεταβλητές Επιχειρήσεων:**

- **Αποστολή,** στην οποία ο Διοικητής αναλύει την προειδοποιητική διαταγή ή διαταγή επιχειρήσεων του προϊσταμένου του, για να προσδιορίσει τον τρόπο με τον οποίο το τμήμα του θα συμβάλλει στην αποστολή.
    - **Αποστολή Προϊσταμένου και Πρόθεση Προϊσταμένου Διοικητού**, αναζητώντας την πρόθεση του άμεσα και επόμενα προϊστάμενου κλιμακίου (δύο κλιμάκια άνω) ή συμπεραίνοντας τις αν αυτές οι πληροφορίες δεν είναι διαθέσιμες.

- **Ιδέα Ενεργείας Προϊσταμένου κλιμακίου**, η οποία καθορίζει τον τρόπο με τον οποίο η αποστολή του τμήματος συνεισφέρει στην επιτυχία της αποστολής του προϊσταμένου.
- **Επιβαλλόμενα, Απορρέοντα και Ουσιώδη έργα**, τα οποία ο διοικητής τα εξάγει από τις προειδοποιητικές διαταγές και διαταγές επιχειρήσεων που έχει λάβει. Από τα επιβαλλόμενα και τα απορρέοντα έργα πρέπει να ξεχωρίσει τα ουσιώδη έργα, αδυναμία ολοκλήρωσης των οποίων οδηγεί και σε αποτυχία της αποστολής.
- **Δεσμεύσεις**, που πιθανώς να έχουν τεθεί στο τμήμα του Διοικητή είτε με την μορφή απαίτησης, ή απαγόρευσης.

- **Εχθρός,** όπου οι απαιτούμενες πληροφορίες διαφέρουν ανάλογα με το κλιμάκιο το οποίο εκτελεί την σχεδίαση. Για μικρά κλιμάκια, είναι απαραίτητη η γνώση για την σύνθεση, διάταξη, ισχύ, ικανότητα ενίσχυσης, πρόσφατες δραστηριότητες, και τρόπους ενεργεία του εχθρού. Οι πληροφορίες αυτές μπορεί να προέρχονται από προϊστάμενα ή γειτονικά κλιμάκια, να αποτελούν προϊόν επεξεργασίας πληροφοριών ή να έχουν συναχθεί από την εμπειρία του Διοικητή.
- **Καιρός-Έδαφος,** όπου εξετάζονται:
  - **Παρατήρηση και Πεδία Βολής,** για τις φίλιες και τις εχθρικές δυνάμεις.
  - **Προσβάσεις,** στις οποίες περιλαμβάνονται χερσαίες, εναέριες και υπόγειες προσβάσεις.
  - **Εδάφη Τακτικής Σημασίας,** τα οποία είναι τοποθεσίες ή περιοχές η κατάληψη των οποίων προσφέρει αξιοσημείωτο τακτικό πλεονέκτημα στον εχθρό.
  - **Εδάφη Ζωτικής Σημασίας,** τα σπουδαιότερο εδάφη τακτικής σημασίας μιας αμυντικής τοποθεσίας, η διατήρηση των οποίων είναι απαραίτητη και για την αποστολή και η κατάληψη τους από τον εχθρό θέτει σε σοβαρό κίνδυνο όλη την αμυντική διάταξη.
  - **Κωλύματα,** οποιοδήποτε εμπόδιο που σχεδιάζεται ή χρησιμοποιείται για να διακόψει, καθηλώσει, στρέψει ή απαγορεύσει την κίνηση της εχθρικής δύναμης.

    - **Κάλυψη και Απόκρυψη,** εδάφη τα οποία περιορίζουν τα πεδία βολής και εξετάζονται και για τις φίλιες και για τις εχθρικές δυνάμεις.
    - **Καιρός,** Και ιδιαίτερα τα στοιχεία του καιρού που αφορούν ορατότητα, ανέμους, βροχοπτώσεις, κάλυψη από σύννεφα, θερμοκρασία και υγρασία, παράγοντες που μπορούν να επηρεάσουν την διεξαγωγή των επιχειρήσεων.
- **Διατιθέμενες Δυνάμεις και Υποστήριξη,** όπου ο Διοικητής, χρησιμοποιώντας την γνώση που έχει για το τμήμα του ως προς το ηθικό, εκπαίδευση, ισχυρά και τρωτά του σημεία, καθορίζει την μαχητική ισχύ της δύναμης του.
- **Διατιθέμενος Χρόνος,** στον οποίο ο Διοικητής εξετάζει όλες τις υποχρεώσεις του και εκτιμά πόσος χρόνος είναι διαθέσιμος για την εκπλήρωση τους.

1.2.8 **Διαδικασία Διοίκησης και Ελέγχου των Επιχειρήσεων**

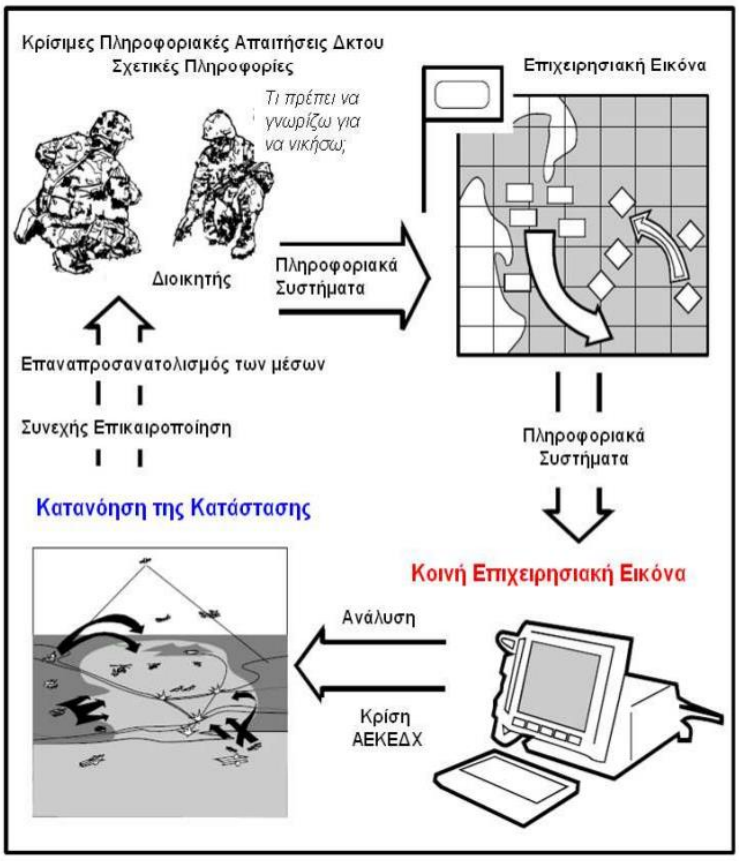
Η Διαδικασία Διοίκησης και Ελέγχου[7] των Επιχειρήσεων αποτελείται από τις εξής κύριες Δραστηριότητες:

- **Σχεδίαση**: Είναι η διαδικασία κατανόησης της κατάστασης, νοερής σύλληψης ενός μελλοντικού επιθυμητού αποτελέσματος και η εξεύρεση αποτελεσματικών τρόπων για την επίτευξη αυτού του αποτελέσματος.

- **Προπαρασκευή**: Αποτελείται από τις ενέργειες στις οποίες προβαίνουν τα επιτελεία και οι μονάδες για την βελτίωση της ικανότητάς τους να εκτελέσουν επιχείρηση, και η οποία μπορεί να περιλαμβάνει, χωρίς να περιορίζεται σε αυτά, βελτίωση και δοκιμή σχεδίων, συλλογή πληροφοριών, αναγνωρίσεις και συντήρηση υλικών.

- **Διεξαγωγή**: Η διεξαγωγή υλοποιεί το σχέδιο, εφαρμόζοντας τη μαχητική ισχύ για την επίτευξη της αποστολής. Η διεξαγωγή εστιάζεται σε συντονισμένη ενέργεια για απόκτηση και διατήρηση της πρωτοβουλίας, δημιουργία και διατήρηση της ορμητικότητας και εκμετάλλευση της επιτυχίας.

- **Αξιολόγηση**: Διαρκής παρακολούθηση και εκτίμηση της εξέλιξης της επιχείρησης, η οποία περιλαμβάνει συνεχή ανάλυση του επιχειρησιακού περιβάλλοντος.



Εικόνα 3 Διαδικασία Ελέγχου

Οι διαδικασίες αυτές κατά την έναρξη είναι διαδοχικές, αλλά με την συνέχιση των επιχειρήσεων η εκτέλεση τους από το επιτελείο γίνεται ταυτόχρονη. Η σχεδίαση είναι συνεχής, η προπαρασκευή αρχίζει με την λήψη διαταγής από ένα τμήμα, επικαλύπτει την σχεδίαση και συνεχίζει κατά την διεξαγωγή για υφιστάμενα τμήματα (τα οποία έχουν λάβει αργότερα την αντίστοιχη διαταγή τους). Η διεξαγωγή είναι η εκτέλεση του σχεδίου, ενώ η αξιολόγηση είναι και αυτή συνεχής, και επηρεάζει τις άλλες τρεις δραστηριότητες.

Η Διοίκηση και ο έλεγχος απαιτούν από τον διοικητή να έχει κατανοήσει το επιχειρησιακό περιβάλλον στο οποίο επιχειρεί. Η κατανόηση αυτή οφείλεται στην γνώση η οποία προκύπτει από την ανάλυση των πληροφοριών, στις οποίες έχει εφαρμοστεί η κρίση του Διοικητή, με σκοπό να προσδιοριστούν οι σχέσεις μεταξύ των μεταβλητών της αποστολής και να υποβοηθηθεί η λήψη απόφασης για την εκπλήρωση της αποστολής.

<table>
<tr><th>ΣΧΕΔΙΑΣΗ</th><th>ΠΡΟΠΑΡΑΣΚΕΥΗ</th><th>ΔΙΕΞΑΓΩΓΗ</th></tr>
<tr><td>• ΔΙΛΑ
• Σχέδια και Διαταγές</td><td>• Επιχειρήσεις Αναγνωρίσεως
• Επιχειρήσεις Ασφαλείας
• Προστασία Δύναμης
• Αναθεώρηση και Βελτίωση Σχεδίου
• Συντονισμός και αποκατάσταση Συνδέσμου
• Δοκιμές
• Τακτική Συγκρότηση
• Εκπαίδευση
• Κινήσεις Τμημάτων
• Έλεγχοι και Επιθεωρήσεις
• Προπαρασκευές ΔΜ
• Σύζευξη νεοεισερχόμενου προσωπικού και τμημάτων</td><td>• Απόφαση
- Εκτέλεση
- Τροποποιήσεις
• Διεύθυνση
- Εφαρμογή ΜΙ
- Συγχρονισμός επιχείρησης
- Διατήρηση της συνέχειας</td></tr>
<tr><td>ΑΞΙΟΛΟΓΗΣΗ
• Παρακολούθηση της κατάστασης
• Παρακολούθηση κριτηρίων επιτυχίας
• Αξιολόγηση ΤΕ</td><td>ΑΞΙΟΛΟΓΗΣΗ
• Παρακολούθηση προπαρασκευών
• Αξιολόγηση προπαρασκευών</td><td>ΑΞΙΟΛΟΓΗΣΗ
• Παρακολούθηση επιχείρησης
• Αξιολόγηση προόδου</td></tr>
<tr><td colspan="3">ΔΙΑΡΚΗΣ ΑΞΙΟΛΟΓΗΣΗ</td></tr>
<tr><td colspan="3">Αντίληψη της κατάστασης – πόροι, λύσεις
Παρακολούθηση της κατάστασης/επιχείρησης, κριτήρια επιτυχίας, μεταβλητές
Αξιολόγηση – πρόβλεψη, απόκτηση-διατήρηση-εκμετάλλευση πρωτοβουλίας</td></tr>
</table>

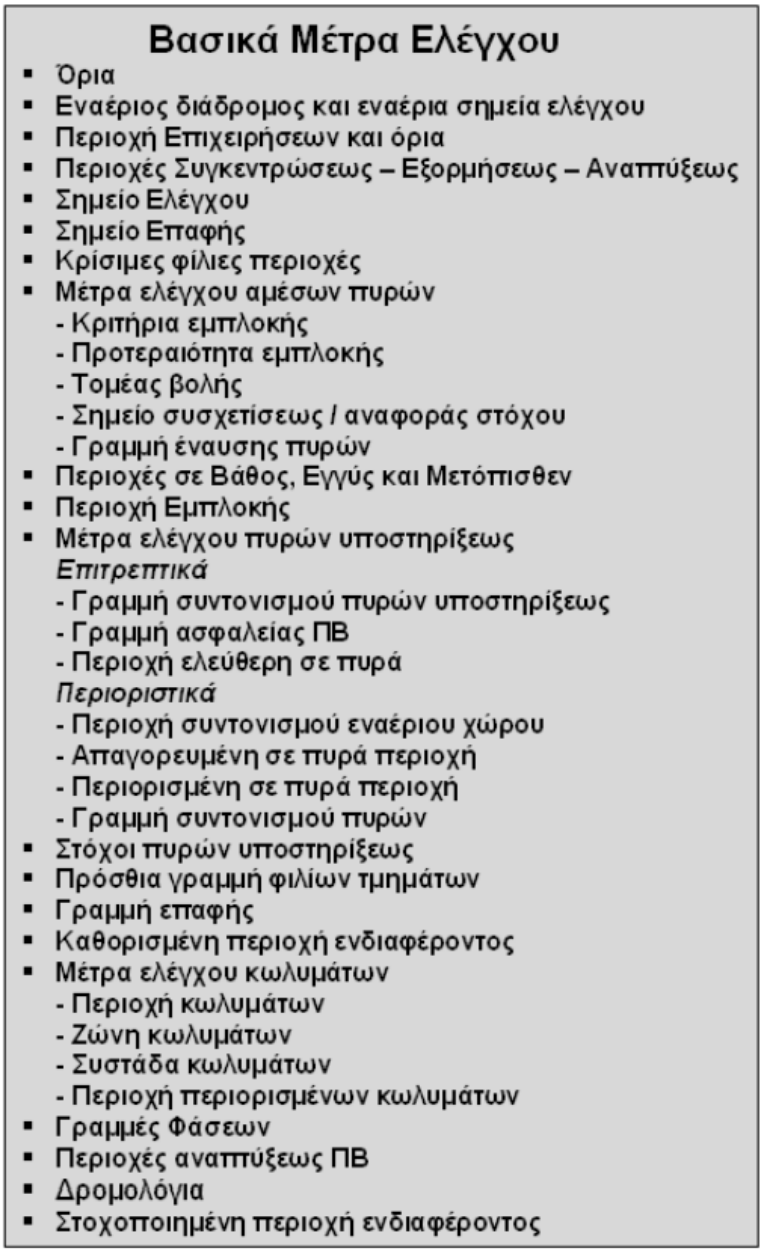
Εικόνα 4 Διαδικασίες Διόικησης και Ελέγχου

1.2.9 **Κύρια συστατικά Σχεδίου Επιχειρήσεων**

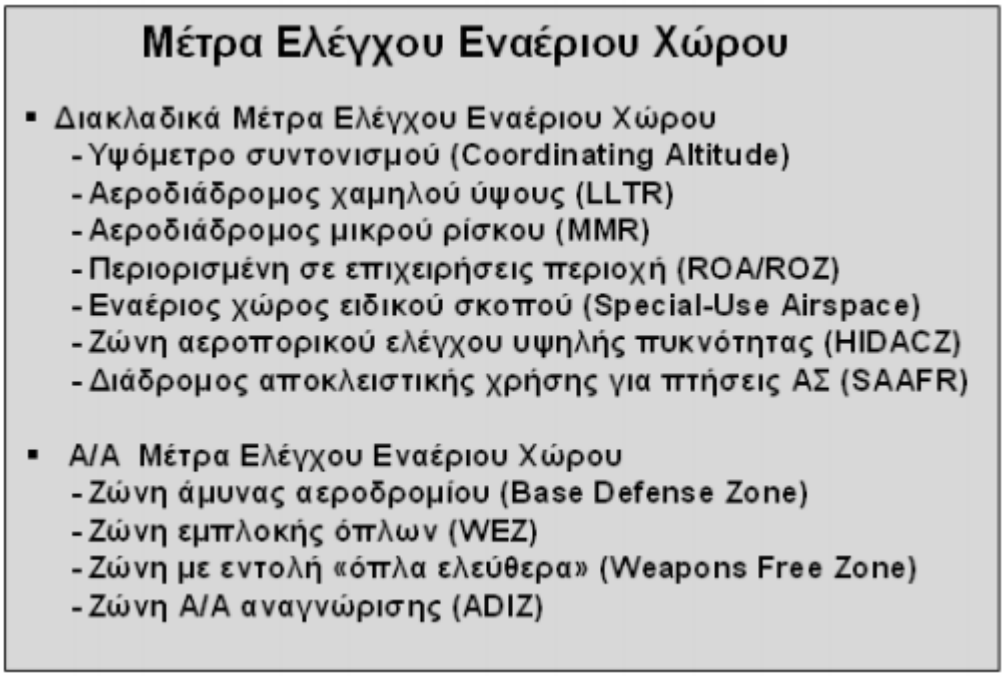
- **Διατύπωση της Αποστολής:** Δημιουργείται από την διεύρυνση του Διοικητή στα πλαίσια πρόθεσης προϊστάμενου Διοικητή δύο κλιμάκια άνω του δικού του. Αυτή οδηγεί σε σαφή διατύπωση της αποστολής, η οποία πρέπει να απαντά στα ερωτήματα Ποιος, Τι, Που, Πότε και Γιατί.
- **Πρόθεση του Διοικητή:** Σαφής και περιεκτική δήλωση του Τι πρέπει μια μονάδα να κάνει, και οι οποίες αντιπροσωπεύουν την επιθυμητή τελική κατάσταση.
- **Ιδέα Ενεργείας:** Κατευθύνει τον τρόπο με τον οποίο οι υφιστάμενοι καλούνται να συνεργαστούν για την εκπλήρωση της αποστολής και προσδιορίζει την αλληλουχία ενεργειών για την επίτευξη της επιθυμητής τελική ενέργειας.

- **Έργα Υφισταμένων:** Έργο είναι μια σαφώς προσδιορισμένη και μετρήσιμη ενέργεια, η οποία πραγματοποιείται από άτομα ή τμήματα, και κατευθύνουν τις μονάδες να εκτελέσουν συγκεκριμένες ενέργειες.
- **Συντονιστικές Οδηγίες:** Οδηγίες που έχουν εφαρμογή μεταξύ δύο ή περισσότερων μονάδων.
- **Μέτρα Ελέγχου:** Αναθέτουν ευθύνες, συντονίζουν ενέργειες, επισημαίνουν περιορισμούς ή καθορίζουν οδηγίες για την ρύθμιση ελευθερίας ενεργειών μεταξύ των τμημάτων. Δύνανται να δώσουν ελευθερία στους υφισταμένους να διεξάγουν επιχειρήσεις εντός της ανατεθείσας περιοχής επιχειρήσεων, χωρίς την ανάγκη επιπρόσθετου σχεδιασμού. Ο Διοικητής καθορίζει το ελάχιστο των μέτρων ελέγχου που απαιτούνται για τον βασικό συντονισμό των μονάδων. Τα μέτρα ελέγχου μπορεί να είναι επιτρεπτικά, να επιτρέπουν δηλαδή συγκεκριμένες ενέργειες, ή περιοριστικά, να περιορίζουν την εκτέλεση συγκεκριμένων ενεργειών.

Τα Μέτρα Ελέγχου και Συντονισμού φαίνονται στις παρακάτω εικόνες:



Εικόνα 5 Βασικά Μέτρα Ελέγχου



Εικόνα 6 Μέτρα Ελέγχου Εναερίου Χώρου

Εικόνα 7 Ειδικά Μέτρα Ελέγχου

1.3 Χρησιμοποιούμενες Τεχνολογίες

1.3.1 **Android**

Το Android[8] είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει στον πυρήνα του λειτουργικό Linux. Αναπτύχθηκε από την Google και μετέπειτα από την Open Handset Alliance, επιτρέποντας στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας έτσι την συσκευή μέσω πολλαπλών βιβλιοθηκών λογισμικού ανεπτυγμένων από την ίδια την Google. Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα smart phones και τα tablet, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις, αυτοκίνητα και ρολόγια χειρός. Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, συνηθισμένους Η/Υ και πολλές άλλες ηλεκτρονικές συσκευές.

Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο. Οι συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS X μαζί.

Το Android, έκδοση 6 [9], "Marshmallow" είναι πρακτικά η έκτη και βασική έκδοση του λειτουργικού συστήματος από την Google. Με πρώτη παρουσίαση τον Μάιο του 2015 στο συνέδριο της Google, η επίσημη κυκλοφορία του έγινε τον Οκτώβριο του 2015.

Βασικά χαρακτηριστικά της έκδοσης αυτής, βελτίωση της συνολικής εμπειρίας του χρήστη. Έγινε εισαγωγή μία εντελώς καινούργιας αρχιτεκτονικής όσων αφορά τα δικαιώματα των εφαρμογών, νέα APIs, νέο σύστημα διαχείρισης ενέργειας (κάτι που καταφέρνει δραστική μείωση της δραστηριότητας της συσκευής όταν αυτή δεν χρησιμοποιείται), υποστήριξη αναγνώρισης δακτυλικών αποτυπωμάτων και USB τύπου C αλλά και δυνατότητα μεταφοράς δεδομένων και εφαρμογών με κάρτα μνήμης microSD.

1.3.2 **Java**

Η Java[10] είναι μία γλώσσα προγραμματισμού που χαρακτηρίζεται από τα εξής: απλή, αντικειμενοστραφής, συμβατή με δικτυακά πρωτόκολλα, ουδέτερη της υποκείμενης αρχιτεκτονικής, φορητή, ασφαλής, υψηλής απόδοσης, δυναμική, μεταγλωττισμένη και

πολυνηματική. Προϊόν της Sun Microsystems, ξεκίνησε σαν μέρος ενός μεγαλύτερου σχεδίου που αφορούσε την ανάπτυξη λογισμικού για μικρές, αξιόπιστες, φορητές, πραγματικού χρόνου συσκευές που στην αρχή βασιζόντουσαν στην C++. Αρκετά προβλήματα όμως παρουσιάστηκαν και η γλώσσα C++ δεν μπόρεσε να εφαρμοστεί τελικά. Χρειάστηκε να αναπτυχθεί μία νέα γλώσσα, η Java. Η Java, στην τελική της μορφή, βρήκε περαιτέρω εφαρμογή στην επίλυση μερικών προβλημάτων του σημερινού προγραμματισμού, όπως animation, την αλληλεπίδραση πραγματικού χρόνου και την εξερεύνηση του διαδικτύου.

1.3.3 **SQL**

Η SQL (Structured Query Language)[11] είναι μία γλώσσα χειρισμού δεδομένων. Σχεδιάστηκε για χρήση σε σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων και η οποία, αρχικά, βασίστηκε στη σχεσιακή άλγεβρα. Η γλώσσα περιλαμβάνει δυνατότητες ανάκτησης και ενημέρωσης δεδομένων, δημιουργίας και τροποποίησης σχημάτων και σχεσιακών πινάκων, αλλά και ελέγχου πρόσβασης στα δεδομένα. Η SQL έγινε η πιο ευρέως χρησιμοποιούμενη γλώσσα για τις σχεσιακές βάσεις δεδομένων.

1.3.4 **MariaDB**

Η MariaDB[12] είναι ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων. Το πρόγραμμα τρέχει έναν εξυπηρετητή (server) παρέχοντας πρόσβαση πολλών χρηστών σε ένα σύνολο βάσεων δεδομένων. Ξεκίνησε σαν κλάδος της MySQL από τον ιδιοκτήτη της όταν αυτή πουλήθηκε στην Oracle. Παρείχε όλες τις δυνατότητες που έχουν εκδόσεις της MySQL.

1.3.5 **NoSQL**

Τα NoSQL[13] συστήματα αναπτύχθηκαν και εξελίχθηκαν παράλληλα από τις μεγαλύτερες εταιρίες πληροφορικής στον κόσμο (Google/Amazon) που είχαν ανάγκη να διαχειρίζονται έναν τεράστιο όγκο δεδομένων, κάτι στο οποίο δεν βόλευαν τα παραδοσιακά μέχρι τότε Συστήματα Διαχείρισης Βάσεων Δεδομένων. Τα NoSQL συστήματα έχουν φτιαχτεί έτσι ώστε να μπορούν να διαχειριστούν μεγάλες ποσότητες δεδομένων χωρίς κατ’ ανάγκη να διατηρούν μία συγκεκριμένη δομή. Επίσης, οι μέθοδοι υλοποίησης και εφαρμογής

τους αξιοποιούν μια αρχιτεκτονική που επιτρέπει, και ίσως διευκολύνει, την κατανεμημένη λειτουργία του συστήματος. Έτσι με αυτόν τον τρόπο το σύστημα μπορεί θεωρητικά να έχει καλύτερες επιδόσεις, από την στιγμή που μπορούν να προστεθούν θεωρητικά άπειροι servers όπου κατανεμημένα θα επεξεργάζονται τα δεδομένα του συστήματος.

1.3.6 **Couchbase**

Ένας Couchbase Server[14], είναι μια ανοιχτού κώδικα, κατανεμημένη multi-model NoSQL Βάση Δεδομένων βασισμένη σε έγγραφα. Αποτελείται από σύνολο προγραμμάτων που έχει γίνει βέλτιστο για διαδραστικές εφαρμογές. Αυτές οι εφαρμογές μπορούν να παρέχουν την εξυπηρέτηση πολλαπλών παράλληλων χρηστών, δημιουργώντας, αποθηκεύοντας, προσπελάζοντας, μετατρέποντας και παρουσιάζοντας δεδομένα.

Ειδικότερα για κινητές συσκευές έχει αναπτυχθεί η Couchbase Lite, χειρισμός της οποίας μπορεί να γίνει με τις γλώσσες Swift, Java, C#, Objective-c. Παρέχει δυνατότητες κρυπτογράφησης, Συχγρονισμός Peer-to-Peer και Replication Checkpoint Reset, που είναι απαιτήσεις για την μελλοντική αναβάθμιση της εφαρμογής.

1.3.7 **REST (Representational State Transfer)**

Το REST[15] είναι ένα αρχιτεκτονικό μοντέλο που παρέχει τυποποίηση μεταξύ συστημάτων στο δίκτυο, επιτρέποντας στα συστήματα να επικοινωνούν μεταξύ τους.

Η βασική ιδέα είναι ότι τα συστήματα που υποστηρίζουν το REST είναι stateless και διαχωρίζουν τις ανησυχίες πελάτη – εξυπηρετητή. Αυτό σημαίνει ότι δεν απαιτείται η γνώση της κατάστασης του καθενός από τον άλλο, όπως επίσης δεν απαιτείται και η γνώση πρότερων μηνυμάτων.

Σε αυτήν την αρχιτεκτονική η υλοποίηση του πελάτη και η υλοποίηση του εξυπηρετητή μπορούν να γίνουν ξεχωριστά, χωρίς να απαιτείται γνώση του ενός για τον άλλο, αρκεί να γνωρίζει κάθε πλευρά την μορφή μνημάτων που θα αποσταλούν. Επίσης, κάθε πελάτης που θα καλέσει το ίδιο Rest endpoint με τις ίδιες μεθόδους θα λάβει το ίδιο αποτέλεσμα.

1.3.7.1 Επικοινωνία Πελάτη – Εξυπηρετητή

Στην αρχιτεκτονική REST ο πελάτης στέλνει αιτήματα για να λάβει ή να μεταβάλλει κάποιους πόρους, και ο εξυπηρετητής απαντάει στα σχετικά αιτήματα. Μια σύνοψη της διαδικασίας εμφανίζεται παρακάτω:

- **Αποστολή Αιτήσεων (REQUEST)**

  Ο πελάτης στέλνει ένα HTTP αίτημα στον εξυπηρετητή για να εκτελέσει την απαιτούμενη ενέργεια. Το αίτημα περιλαμβάνει:

  - Μια εντολή HTTP για την δράση που θέλει να εκτελέσει. Οι ποιο συχνά χρησιμοποιούμενες εντολές είναι οι:
    - **GET**: Λήψη πόρου (με id) ή συλλογής πόρων
    - **POST**: Δημιουργία νέου πόρου
    - **PUT**: Μεταβολή κάποιου πόρου (με id)
    - **DELETE**: Διαγραφή κάποιου πόρου (με id)
  - Κεφαλίδα, που μεταφέρει επιπλέον δεδομένα για το αίτημα. Παρακάτω παρουσιάζονται ορισμένες συχνά χρησιμοποιούμενες παράμετροι κεφαλίδων, τα MIME (Multipurpose Internet Mail Extension):
    - **image** (image/png, image/jpeg)
    - **audio** (audio/wav, image/mpeg)
    - **video** (video/mp4, video/ogg)
    - **application** (application/xml, application/octet-stream, application/pdf, application/json)
  - Το μονοπάτι (path) προς τον πόρο
  - Ένα προαιρετικό σώμα κειμένου (message body) που περιέχει δεδομένα

- **Αποστολή Απάντησης (RESPONSE)**

Ο Εξυπηρετητής επεξεργάζεται το αίτημα και αναλόγως της κατάληξης απαντάει στον Πελάτη. Η απάντηση περιλαμβάνει:

- Τύπο Περιεχομένου (Content-Type): Αν ο εξυπηρετητής απαντάει με αποστολή δεδομένων, απαιτείται ο τύπος περιεχομένου. Αυτοί οι τύποι περιεχομένου είναι τύποι MIME όπως παρατέθηκαν παραπάνω.

- Κωδικός Κατάστασης (Status Code): Ένας αριθμητικός κωδικός για συνοπτική περιγραφή της τελικής κατάστασης του αιτήματος. Οι ποιο συχνοί κωδικοί παρατίθενται παρακάτω:

    - **200 (OK):** Προκαθορισμένη απόκριση επιτυχίας HTTP αίτησης. Επιστρέφεται πάντα στην περίπτωση επιτυχίας της GET ή PUT εντολής.

    - **201 (CREATER):** Επιστρέφεται όταν η αίτηση δημιουργίας του πόρου ήταν επιτυχής. Επιστρέφεται πάντα στην περίπτωση επιτυχίας της POST εντολής.

    - **204 (NO CONTENT):** Επιστρέφεται όταν η αίτηση ήταν επιτυχής, αλλά δεν αναμένεται επιστροφή πόρου. Επιστρέφεται πάντα στην περίπτωση επιτυχίας της DELETE εντολής.

    - **400 (BAD CONTENT):** Επιστρέφεται όταν δεν μπορεί να γίνει επεξεργασίας του αιτήματος λόγω κάποιου λάθους όπως κακή σύνταξη αιτήματος, υπέρβαση μεγέθους ή άλλο λάθος στον πελάτη.

    - **403 (FORBIDDEN):** Επιστρέφεται όταν ο πελάτης δεν έχει δικαίωμα να προσπελάσει τον πόρο.

    - **404 (NOT FOUND):** Επιστρέφεται όταν ο πόρος δεν μπορεί να βρεθεί. Μπορεί να έχει σβηστεί ή να είναι λάθος το μονοπάτι.

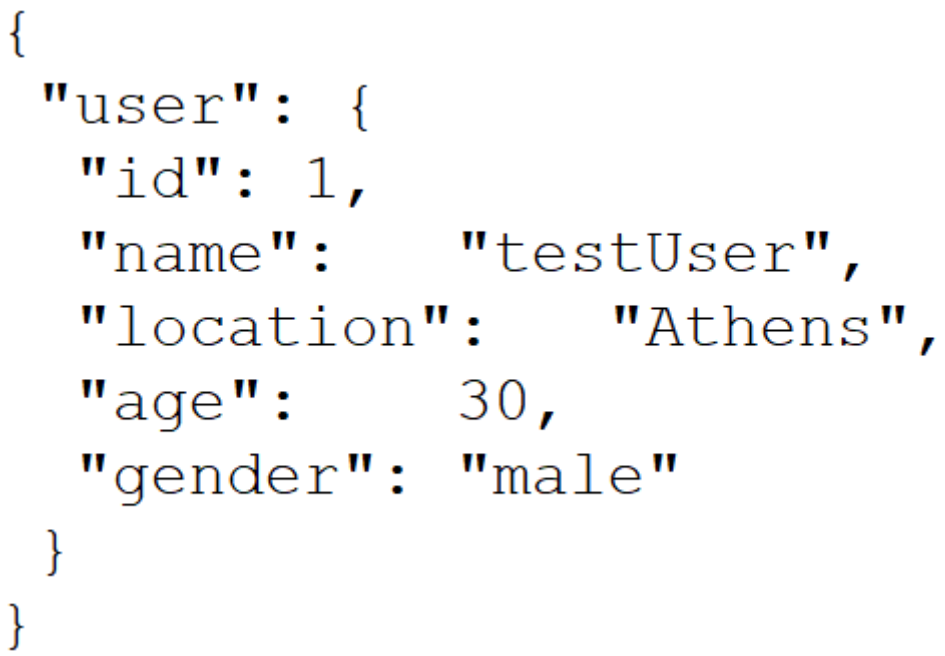
    - **500 (INTERNAL SERVER ERROR):** Γενικό μήνυμα σφάλματος που παρουσιάζεται σε περίπτωση μη αναμενόμενης αποτυχίας για την οποία δεν υπάρχουν παραπάνω πληροφορίες.

1.3.8 **JSON**

Στην πληροφορική, JavaScript Object Notation ή JSON[16] είναι ένα ανοικτό μορφότυπο το οποίο χρησιμοποιεί κείμενο, εύκολα

προσπελάσιμο και αναγνώσιμο από τον άνθρωπο, για τη μετάδοση αντικειμένων (δεδομένων) που αποτελούνται από ζεύγη χαρακτηριστικών και τύπου δεδομένων συστοιχιών (ή άλλων σειριοποιήσιμων τιμών). Πρόκειται για ένα πολύ κοινό μορφότυπο δεδομένων που χρησιμοποιείται για την ασύγχρονη επικοινωνία μεταξύ πελάτη και εξυπηρετητή και γενικότερα στο διαδίκτυο.

Ένα παράδειγμα JSON αντικειμένου φαίνεται παρακάτω:



Εικόνα 8 Παράδειγμα JSON

1.3.9 **Retrofit**

Μια type-safe[17] βιβλιοθήκη-πελάτης για την κατανάλωση REST Services σε Java και Android. Αναπτύσσεται από την Square. Μοντελοποιεί τα Rest Endpoints ως Java interfaces, καθιστώντας την χρήση τους ιδιαίτερα εύκολη και δέχεται τα μεταδεδομένα για την κάθε υπηρεσία με την μορφή @Annotations.

1.3.10 **Extensible Map Platform (EMP) Android Development Kit**

Ένα Standard Development Kit[18] που αναπτύχθηκε για την ενσωμάτωση σε εφαρμογές ικανοτήτων γεοχωρικών αναπαραστάσεων δεδομένων και γραφικών τακτικής, όπως χρησιμοποιούνται από τις ένοπλες δυνάμεις των ΗΠΑ. Από τον

Οκτώβριο του 2009 το Υπουργείο Άμυνας των ΗΠΑ έδωσε άδεια να ανέβει το λογισμικό στο GitHub ως Open Source.

1.3.11 **Dagger 2**

Ένα προγραμματιστικό πλαίσιο[19] εισαγωγής εξαρτήσεων (dependency injection), για Android και Java. Αναπτύσσεται κυρίως από την Google και προσφέρει γρήγορη compile-time εισαγωγή εξαρτήσεων με την χρήση ειδικών διεπαφών @Component.

1.3.12 **RxJava 2**

Υλοποίηση για Java της ReactiveX[20], βιβλιοθηκών που επιτρέπουν την σύνθεση ασύγχρονων και βασισμένων σε γεγονότα προγραμμάτων με την χρήση Observable Sequences. Είναι ελαφριά και υποστηρίζει πληθώρα γλωσσών που βασίζονται στο Java Virtual Machine (JVM) όπως Scala, Kotlin και φυσικά Java.

1.3.13 **Wildfly Application Server**

Application Server[21] ανοιχτού κώδικα, ειδικά σχεδιασμένος για Java based Εφαρμογές. Αναπτύχθηκε και συντηρείται έως σήμερα από την Red Hat και επιμέρους κοινότητες. Αποτελεί την community edition του Jboss AS.

1.3.14 **Android Studio**

Το Android Studio[22] είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android. Ανακοινώθηκε στις 16 Μαϊου 2013 στο συνέδριο Google I/O. Το Android Studio είναι διαθέσιμο ελεύθερα με την άδεια Apache.

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

1.3.15 **IntelliJ IDEA**

Το IntelliJ IDEA[23] είναι ένα περιβάλλον ανάπτυξης (IDE) της Java για παραγωγή λογισμικού υπολογιστή. Είναι προϊόν της JetBrains (παλιότερα γνωστή ως IntelliJ) και είναι διαθέσιμο σαν προϊόν της Apache.

Τα ενδιαφέροντα χαρακτηριστικά του Intellij IDEA, είναι η δυνατότητα του για διόρθωση του κώδικα κατά την διαδικασία γραφής του (ακόμη και με παραπομπές), πληθώρα εργαλείων που μας επιτρέπουν προσπέλαση σε πολλές, και διαφορετικών ειδών, βάσεις δεδομένων αλλά και υποστήριξη πάρα πολλών διαφορετικών γλωσσών προγραμματισμού (μέσω plugins) που καθιστούν το IntelliJ IDEA πραγματικό πολυεργαλείο για τον σύγχρονο προγραμματιστή.

1.3.16 **GitHub**

Το Git είναι ένα σύστημα ελέγχου εκδόσεων με έμφαση στην ταχύτητα, στην ακεραιότητα των δεδομένων και στην υποστήριξη για κατανεμημένες μη γραμμικές ροές εργασίας. Το Git σχεδιάστηκε και αναπτύχθηκε αρχικά από τον Linus Torvalds για τη ανάπτυξη του πυρήνα Linux το 2005 και έχει γίνει από τότε το πιο διαδεδομένο σύστημα ελέγχου εκδόσεων για ανάπτυξη λογισμικού.

Το GitHub[24] ειναι ένα hosting service, βασισμένο στο διαδίκτυο, που επιτρέπει τον έλεγχο έκδοσης χρησιμοποιώντας το Git. Βασικά χρησιμοποιείται για κώδικα υπολογιστή. Υποστηρίζει όλες τις εντολές του Git, παρέχει επιπλέον δυνατότητες και είναι εξαιρετικά δημοφιλές, επομένως συναντάται και συχνά.

2. Προδιάγραφες και Αρχιτεκτονική Συστήματος

2.1 Προδιαγραφές Συστήματος

Πριν γίνει η περιγραφή της αρχιτεκτονικής της εφαρμογής στα συστατικά μέρη που την αποτελούν και πως αυτά επικοινωνούν μεταξύ τους, θα γίνει μια σύντομη περιγραφή των προδιαγραφών λειτουργίας του συστήματος, με έμφαση στον Χρήστη της Εφαρμογής, και συγκεκριμένα στα χαρακτηριστικά του και τις απαιτήσεις του.

Τα βασικά χαρακτηριστικά του χρήστη παρουσιάζονται στην παρακάτω λίστα:

- Ο χρήστης είναι γνώστης των βασικών κανονισμών και εγχειριδίων εκστρατείας του Στρατού Ξηράς, και συγκεκριμένα αυτά που αφορούν την ανάλυση, σχεδίαση και διεξαγωγή επιχειρήσεων του Στρατού Ξηράς.
- Ο χρήστης ενδιαφέρεται για την σχεδίαση επιχείρησης σε πραγματικό χρόνο, με δυνατότητα μετάδοσης και ανταλλαγής πληροφοριών και δεδομένων.
- Ο χρήστης έχει την δυνατότητα φόρτισης της συσκευής, ώστε να είναι πάντα διαθέσιμη.
- Ο χρήστης έχει μια βασική εξοικείωση με την χρήση έξυπνων συσκευών κινητής τηλεφωνίας.
- Ο χρήστης διαθέτει έξυπνη συσκευή κινητής τηλεφωνίας Android, έκδοσης Marshmallow ή καλύτερης κατάλληλα διαμορφωμένη από την Υπηρεσία.
-

Με βάση τα παραπάνω χαρακτηριστικά, οι απαιτήσεις του χρήστη καθορίζονται παρακάτω:

- Η εφαρμογή να τηρεί τις συμβάσεις που προβλέπονται στους οικείους κανονισμούς, με έμφαση στον Στρατιωτικό Κανονισμό 43-1, «Στρατιωτικές Συνθηματικές Παραστάσεις»
- Να μπορεί να ανταλλάσσει δεδομένα με τις υπόλοιπες επιχειρησιακές εφαρμογές του Στρατού Ξηράς.
- Να παρέχει όλα τα δεδομένα και λειτουργίες που προσφέρονται από τις παλαιότερες εφαρμογές σχεδίασης επιχειρήσεων.

- Το περιβάλλον διεπαφής να μην διαφέρει σημαντικά από τις λοιπές εφαρμογές σχεδίασης, ώστε να είναι δυνατή η εύκολη μετάβαση από το ένα περιβάλλον στο άλλο.
- Το περιβάλλον διεπαφής να είναι εύχρηστο ακόμα και από άτομα που δεν έχουν σημαντική εμπειρία στην χρήση έξυπνων συσκευών κινητής τηλεφωνίας.
- Να επιτρέπει την σχεδίαση ακόμα και υπό αντίξοες συνθήκες.
- Να παρέχει τις μέγιστες δυνατές πληροφορίες που είναι απαραίτητες για την σχεδίαση, χωρίς να επιβαρύνει το δίκτυο.
- Να είναι εύκολα αναβαθμίσιμο, ώστε να ταιριάζει στο σύγχρονο πεδίο των επιχειρήσεων.
- Να προσφέρει δυνατότητα χρήσης του και εκτός σύνδεσης, ως εργαλείο σχεδίασης.
- Να είναι συμβατό με τις περισσότερες συσκευές κινητής τηλεφωνίας που χρησιμοποιούνται.
- Να προσφέρει την μέγιστη δυνατή ασφάλεια από πλευράς εφαρμογής.
- Να παραμετροποιεί τις διαθέσιμες λειτουργίες βάση των δικαιωμάτων του χρήστη.
- Να μην καταναλώνει υπερβολικά ενέργεια, ώστε να μπορεί να χρησιμοποιηθεί για αρκετές ώρες και στο περιβάλλον επιχειρήσεων.

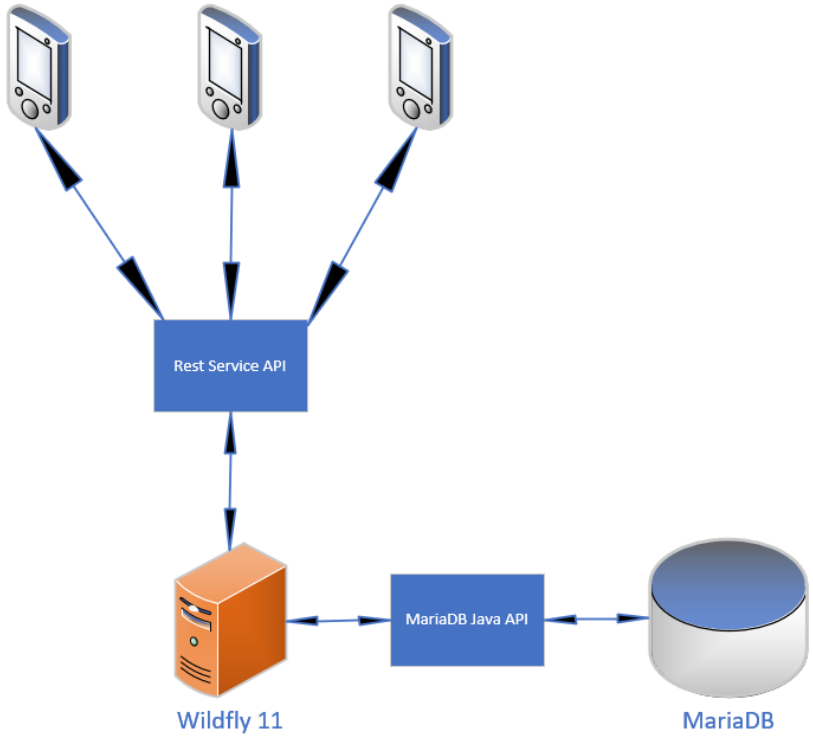
2.2 Αρχιτεκτονική του Συστήματος

Η αρχιτεκτονική τής εφαρμογής βασίζεται στο μοντέλο αρχιτεκτονικής Πελάτη - Εξυπηρετητή. Ο Εξυπηρετητής είναι υπεύθυνος για την λήψη των αιτήσεων του Πελάτη, την επεξεργασία αυτών των αιτήσεων, και την αποστολή των απαντήσεων στον Πελάτη με τα δεδομένα που αφορούν το αίτημα του. Για την υλοποίηση των αιτημάτων και απαντήσεων χρησιμοποιήθηκε διεπαφή προγραμματισμού εφαρμογών τύπου REST, λόγω πλήρης κάλυψης των αναγκών αλλά και για συμβατότητα με τα υπόλοιπα συστήματα του Στρατού Ξηράς.

Για την κάλυψη των απαιτήσεων δεδομένων του Πελάτη, ο εξυπηρετητής επικοινωνεί με την Βάση Δεδομένων, η οποία περιλαμβάνει όλα τα απαραίτητα δεδομένα που αφορούν τον χρήστη. Τα δεδομένα αυτά αφορούν στοιχεία χρηστών, δικαιωμάτων, μονάδων, εγκαταστάσεων και επιχειρήσεων.

Άμεση επικοινωνία με την βάση έχει μόνο ο Εξυπηρετητής. Ο Πελάτης μπορεί μόνο να επικοινωνεί με τον Εξυπηρετητή και να του αποστέλλει αιτήματα καταχώρησης ή ανάκτησης δεδομένων. Ο Εξυπηρετητής ελέγχει το αίτημα, και κατόπιν επιστρέφει κατάλληλα επεξεργασμένα δεδομένα στον πελάτη.

Η βασική αρχιτεκτονική παρουσιάζεται στο παρακάτω σχήμα:



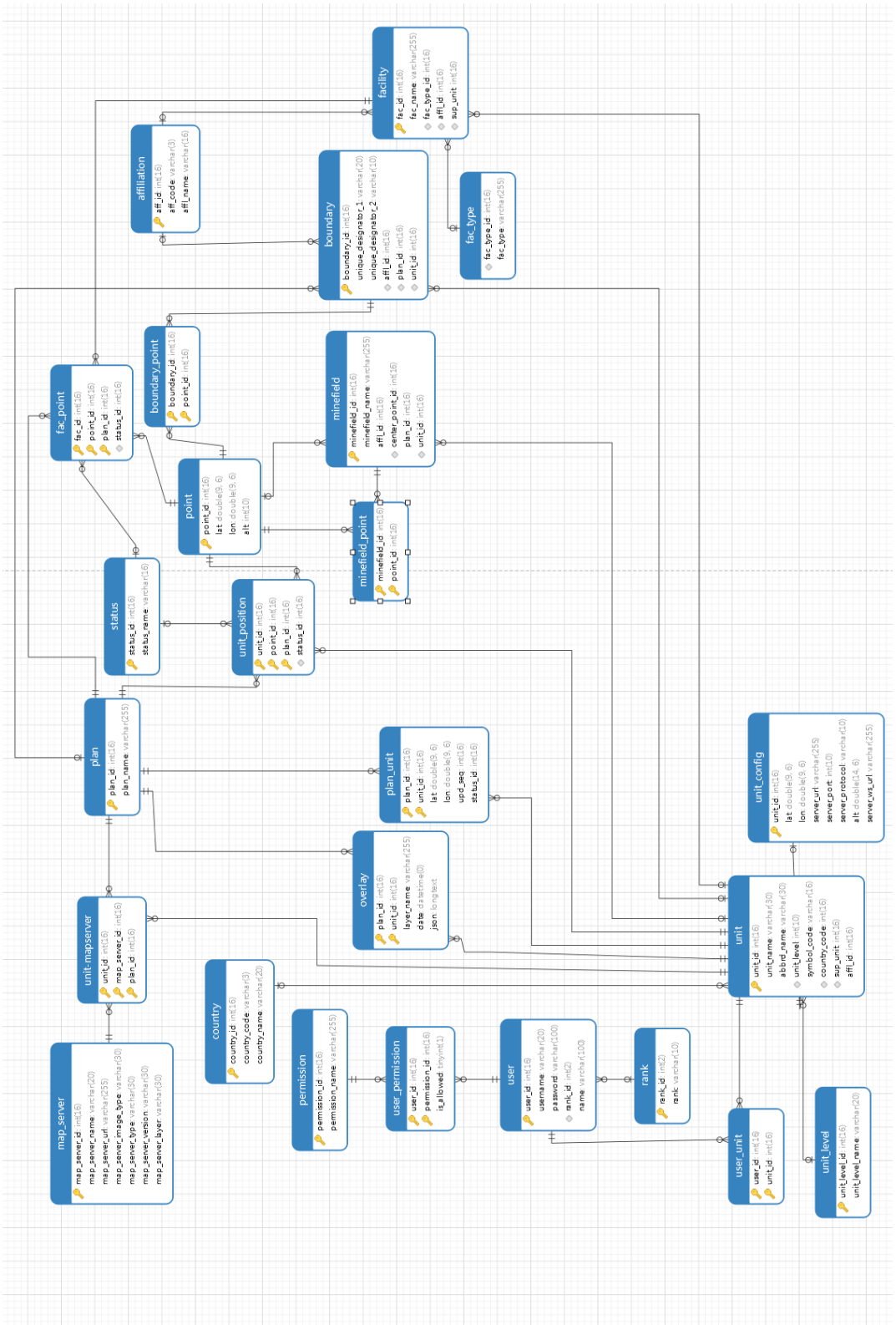
Εικόνα 9 Αρχιτεκτονική Συστήματος

Μετά την παραπάνω γενική περιγραφή, ακολουθεί ανάλυση του Εξυπηρετητή, της Βάσης Δεδομένων και της Εφαρμογής με περισσότερες λεπτομέρειες.

2.2.1 **Βάση Δεδομένων**

Η Βάση Δεδομένων περιέχει όλα τα δεδομένα επιχειρήσεων που χρειάζεται ο χρήστης για σχεδιασμό και παρακολούθηση μιας επιχείρησης, όπως προαναφέρθηκε. Για τον σκοπό αυτό σχεδιάστηκε σε εξυπηρετητή βάσης δεδομένων MariaDB,

κατάλληλη βάση για την επίτευξη αυτού του σκοπού, με το παρακάτω σχήμα βάσης:



Εικόνα 10 Σχήμα Βάσης Δεδομένων

Η παρακάτω βάση δεδομένων είναι μια εξαιρετικά απλοποιημένη έκδοση της πραγματικής και εξομοιώνει την βάση δεδομένων στην οποία βασίζεται ο στρατός ξηράς. Για λόγους ασφάλειας του συστήματος από εξωτερικές απειλές δεν δίνεται η δυνατότητα χρησιμοποίησης της υπάρχουσας βάσης, καθώς θα υπήρχε διαρροή της δομής της στο διαδίκτυο με τις αναμενόμενες συνέπειες. Επίσης, δεν ήταν δυνατή η ανάπτυξη εξαρχής μιας πραγματικής βάσης για το μοντέλο MIP 3.0, καθώς η πολυπλοκότητα της απαιτεί χρονικό διάστημα πέραν του δοθέντος για ομάδα χρηστών, η οποία θα ασχολείται εξολοκλήρου με αυτή. Παρόλα αυτά οι παραπάνω πίνακες περιλαμβάνουν όλα τα τελικά δεδομένα που απαιτούνται για την πλήρη λειτουργία της εφαρμογής.

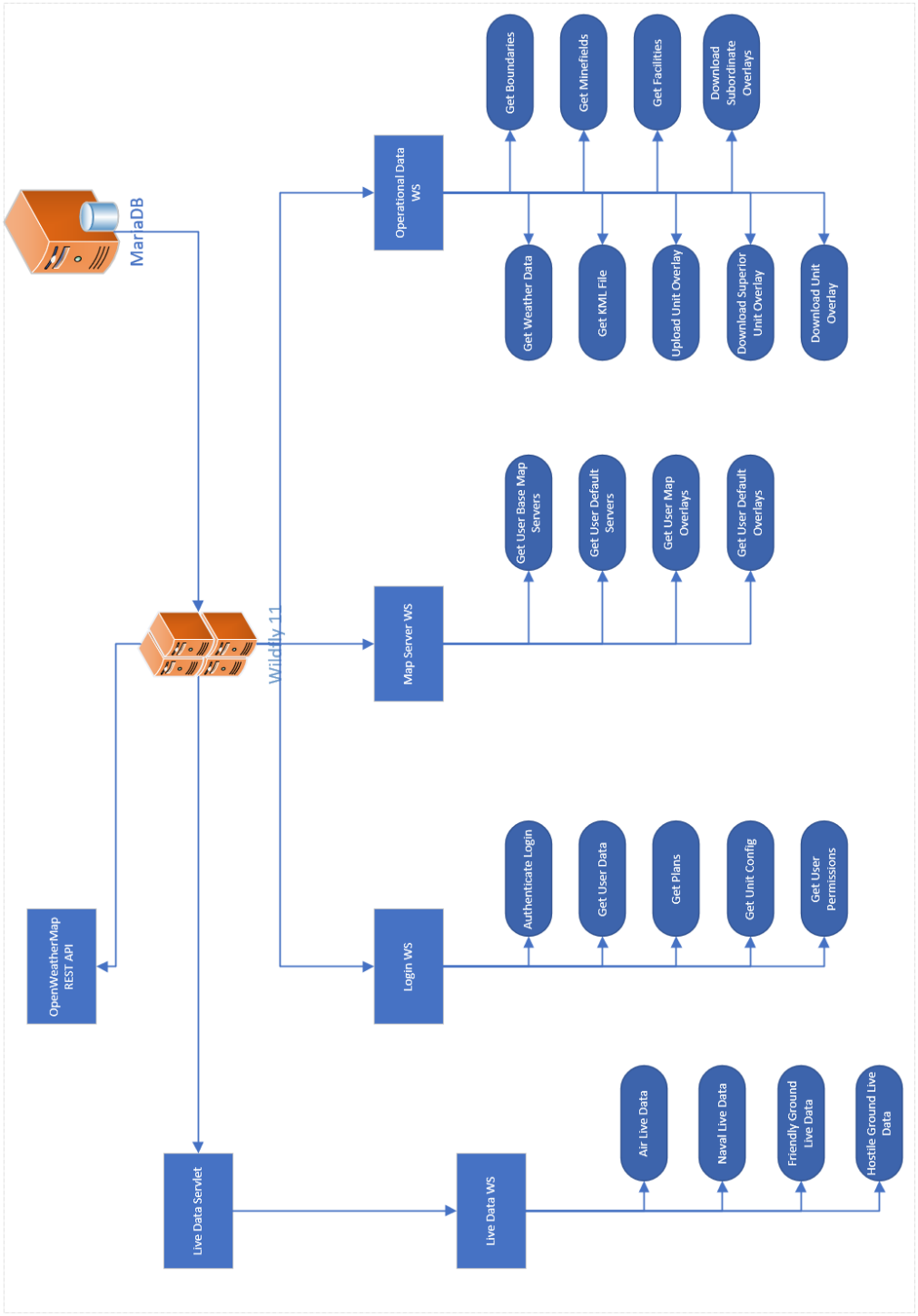
Στο παραπάνω απλοποιημένο μοντέλο εμφανίζονται οι παρακάτω πίνακες:

- **Affiliation,** lookup πίνακας για την σχέση ενός στοιχείου ως προς τον χρήστη (Φίλιος, Ουδέτερος, Εχθρικός, κ.ο.κ.).
- **Boundary,** πίνακας που περιέχει στοιχεία για τα όρια μεταξύ μονάδων.
- **Boundary_Point,** πίνακας συσχέτισης ενός ορίου με τα σημεία του.
- **Country,** lookup πίνακας για την χώρα προέλευσης ενός στοιχείου.
- **Fac_point,** πίνακας συσχέτισης μίας εγκατάστασης με τα σημεία της.
- **Fac_type,** πίνακας συσχέτισης μιας εγκατάστασης με τον τύπο της
- **Facility,** πίνακας που περιέχει τις εγκαταστάσεις.
- **Map_server,** πίνακας που περιέχει τις διευθύνσεις εξυπηρετητών χαρτών για χάρτες υποβάθρου και διαφανή χαρτών.
- **Minefield,** πίνακας που περιέχει τα ναρκοπέδια.
- **Minefield_point,** πίνακας συσχέτισης ναρκοπεδίου με τα σημεία του.
- **Overlay,** πίνακας που περιέχει τα διαφανή σχεδίων.
- **Permission,** πίνακας που περιέχει τα δικαιώματα πρόσβασης σε λειτουργίες της εφαρμογής.
- **Plan,** πίνακας που περιέχει τα φορτωμένα Σχέδια επιχειρήσεων.

- **Plan_Unit,** πίνακας συσχέτισης κάθε μονάδας με το αντίστοιχο σχέδιο επιχειρήσεων.
- **Point,** πίνακας που περιλαμβάνει σημεία στοιχείων στο χάρτη.
- **Rank,** lookup πίνακας για τον βαθμό που φέρει κάθε χρήστης.
- **Status,** lookup πίνακας για την κατάσταση(λειτουργικό, κατεστραμμένο κ.ο.κ.) που βρίσκεται ένα στοιχείο.
- **Unit,** πίνακας που περιέχει όλες τις μονάδες.
- **Unit_config,** πίνακας που περιέχει όλα τα δεδομένα ρύθμισης της εφαρμογής για χρήστες που ανήκουν στην ίδια μονάδα.
- **Unit_level,** lookup πίνακας για το επίπεδο μιας μονάδας.
- **Unit_position,** πίνακας συσχέτισης μονάδας με την θέση της για ένα συγκεκριμένο σχέδιο επιχείρησης.
- **Unit_mapserver,** πίνακας συσχέτισης μονάδας με τον εξυπηρετητή χαρτών της για ένα συγκεκριμένο σχέδιο επιχείρησης.
- **User,** πίνακας που περιέχει τους χρήστες της εφαρμογής.
- **User_permission,** πίνακας συσχέτισης χρήστη με τα δικαιώματα που έχει.
- **User_unit,** πίνακας συσχέτισης χρήστη με την μονάδα στην οποία ανήκει.

2.2.2 **Εξυπηρετητής**

Ο Εξυπηρετητής είναι ένας Application Server Wildfly 11, στον οποίο έχει γίνει εγκατάσταση το module για την εξυπηρέτηση των αιτήσεων του χρήστη. Για να επιτύχει αυτό οι λειτουργίες του χωρίζονται σε λειτουργίες παροχής στατικών δεδομένων και σε λειτουργίες παροχής πραγματικών δεδομένων, όπως παρακάτω:



Εικόνα 11 Αρχιτεκτονική Εξυπηρετητή

Ο Εξυπηρετητής επικοινωνεί με την βάση δεδομένων, καθώς και με το API του OpenWeatherMap για την ανάκτηση δεδομένων καιρού.

Κατόπιν πρώτης αίτησης οποιουδήποτε χρήστη, αρχίζει να τρέχει το Live Data Servlet, το οποίο παρέχει ένα σύνολο δεδομένων μονάδως επιφανείας, αέρος και θάλλασας. Αυτά τα δεδομένα

μεταβάλλονται σε πραγματικό χρόνο, όπως στο επιχειρησιακό περιβάλλον.

Επίσης ο εξυπηρετητής παρέχει διεπαφές τύπου REST για τέσσερις κατηγορίες υπηρεσιών:

- **Live Data WS:** Τα δεδομένα που παράγονται από το servlet παρέχονται στους χρήστες που έχουν δικαίωμα και τις αιτούνται.

- **Login WS:** Οι υπηρεσίες που έχουν σχέση με την αυθεντικοποίηση του χρήστη, καθώς και με τις ρυθμίσεις του προ της εισόδου στην κύρια εφαρμογή.

- **MapServer WS:** Οι υπηρεσίες που έχουν σχέση με την παροχή δεδομένων για τους εξυπηρετητές χαρτών του χρήστη.

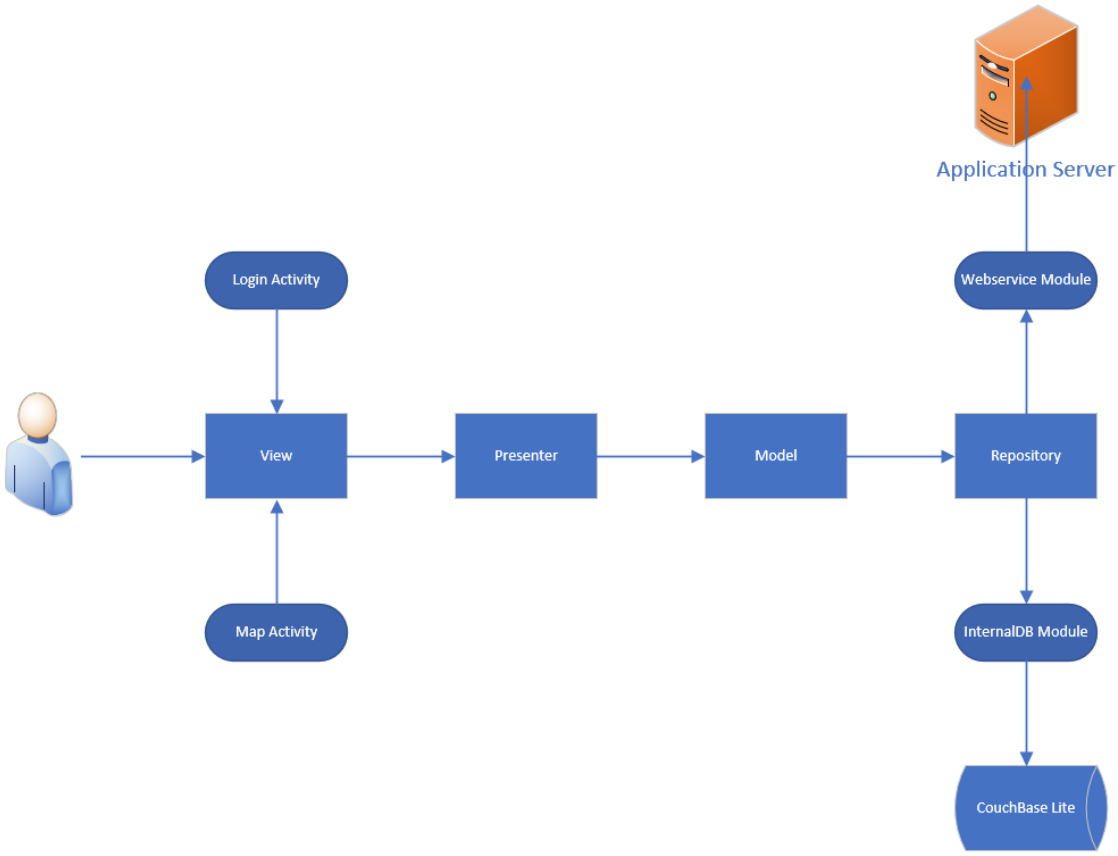
- **Operational Data WS:** Οι υπηρεσίες που παρέχουν τα επιχειρησιακά δεδομένα και λειτουργίες στον χρήστη.

Όπως ισχύει και για την βάση δεδομένων, ο εξυπηρετητής εξομοιώνει τον πραγματικό εξυπηρετητή που χρησιμοποιείται από τον Στρατό Ξηράς για λόγους ασφαλείας. Παρόλα αυτά, τα παραγόμενα αντικείμενα τύπου JSON δεν διαφέρουν από αυτά που χρησιμοποιούνται από τον πραγματικό εξυπηρετητή, οπότε και η διαφορά του με τον πραγματικό δεν είναι σημαντική και δεν θα απαιτηθεί τροποποίηση της εφαρμογής για την εκμετάλλευση των κανονικών δεδομένων.

2.2.3 **Εφαρμογή**

Η εφαρμογή βασίζεται στην αρχιτεκτονική Μοντέλο – Όψη – Παρουσιαστής (Model – View – Presenter), η οποία χρησιμοποιείται κυρίως για διεπαφές χρήστη.

Το Μοντέλο είναι διεπαφή για τον καθορισμό των δεδομένων και επικοινωνεί με την κατάλληλη αποθήκη δεδομένων (Repository). Η Όψη είναι διεπαφή υπεύθυνη για την παρουσίαση των δεδομένων και την δρομολόγηση τους στον Παρουσιαστή για επίδραση πάνω στα δεδομένα. Ο Παρουσιαστής είναι ο ενδιάμεσος μεταξύ του Μοντέλου και της Όψης και δρα και στα δύο.



Εικόνα 12 Αρχιτεκτονική Εφαρμογής

Οι δύο βασικές όψεις είναι οι:

- **Login Activity,** που εκτελεί την αυθεντικοποίηση του χρήστη και την αρχικοποίηση των δικαιωμάτων και ρυθμίσεων του.
- **Map Activity,** η οποία είναι ολόκληρη η εφαρμογή σχεδίασης και παρακολούθησης.

Η αποθήκη πληροφοριών επικοινωνεί με τις:

- **InternalDB Module** για την ανάκτηση των αποθηκευμένων διαφανών στην εσωτερική βάση της συσκευής
- **Webservice Module,** για την ανταλλαγή δεδομένων με τον Εξυπηρετητή.

3. Υλοποίηση του Συστήματος

3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναλυθεί η υλοποίηση της εφαρμογής Σχεδίασης και Παρακολούθησης Επιχειρήσεων-Ασκήσεων. Θα δειχθεί πως οι προδιαγραφές και οι απαιτήσεις που αναφέρθηκαν στο 2ο Κεφάλαιο χρησιμοποιήθηκαν για την υλοποίηση της αρχιτεκτονικής του συστήματος.

Η ανάπτυξη της εφαρμογής έγινε σε γλώσσα προγραμματισμού Java, έκδοση 1.8, τόσο για την εφαρμογή του Πελάτη όσο και για τον Εξυπηρετητή.

Η εφαρμογή τρέχει σε λειτουργικό περιβάλλον Android, έκδοση Marshmallow (API 26). Επιλέχθηκε για πολλούς λόγους, μεταξύ των οποίων είναι ότι είναι ευρέως διαδεδομένο, μπορεί να εγκατασταθεί σε πληθώρα συσκευών, που μπορούν να αγοραστούν για να καλύπτουν πολλαπλές προδιαγραφές και μπορεί να παραμετροποιηθεί και πέραν της εφαρμογής με άλλες εφαρμογές που έχουν αναπτυχθεί εσωτερικά και εξωτερικά για την κάλυψη περισσοτέρων αναγκών του επιχειρησιακού περιβάλλοντος.

Ο Εξυπηρετητής είναι ο Wildfly 11, Java Application Server, συμβατός με την προαναφερθείσα έκδοση της γλώσσας, στον οποίο έχουν γίνει οι απαραίτητες ρυθμίσεις για να επικοινωνεί με την βάση δεδομένων, και ο οποίος χρησιμοποιείται για την εγκατάσταση και εκτέλεση του module το οποίο αποτελεί τον εξυπηρετητή της εφαρμογής.

Για την επίτευξη της απαίτησης δυνατότητας εύκολης αναβάθμισης της εφαρμογής, χρησιμοποιήθηκε η αρχιτεκτονική Μοντέλο – Όψη – Παρουσιαστής, ώστε μελλοντικές αναβαθμίσεις είτε στην διεπαφή του χρήστη είτε στα δεδομένα να επηρεάσει μόνο το αναβαθμιζόμενο κομμάτι, και να μην απαιτεί δομικές αλλαγές σε ολόκληρη την εφαρμογή.

Για την επίτευξη της διαλειτουργικότητας τόσο με τα εγχώρια συστήματα όσο και με τα συστήματα του ΝΑΤΟ η εφαρμογή βασίστηκε στο Extensible Map Platform Android Development Kit, το οποίο παρέχει τις βασικές συνθηματικές παραστάσεις, όπως αυτές χρησιμοποιούνται από τις συμμαχικές δυνάμεις, καθώς και

βασικές λειτουργίες χάρτη βασιζόμενο στην μηχανή Worldwind της NASA.

Τέλος, η διεπαφή χρήστη της εφαρμογής σχεδιάστηκε να είναι όσο δυνατόν παρόμοια με τις υπάρχουσες εφαρμογές του Στρατού Ξηράς, λαμβάνοντας όμως υπόψιν την ιδιαιτερότητα της ως εφαρμογή έξυπνης κινητής υπολογιστικής.

Η εφαρμογή υλοποιεί τις παρακάτω διεπαφές:

- **Login:**
    - **Μοντέλου**

```
interface Model {
    Completable checkServerStatus(String url, int port);

    Observable<Boolean> login(String username, String password);

    Observable<UserData> getUserData(String username, String password);

    boolean initializeAppState(ImmutableAppState state);

    void unsubscribe();

    void subscribe(Disposable d);

    Observable<Config> getUnitConfig(String username, String password, long unitId);

    void updateAppConfig(Config config);

    Observable<List<Plan>> getPlans(String username, String password);

    Observable<UserPermissions> getUserPermissions(String username, String password);
}
```

Εικόνα 13 Login Μοντέλο

- **Όψη**

```
interface View {

    String getUsername();

    void setUsername(String userName);

    String getPassword();

    void setPassword(String password);

    void showInputErrorMessage();

    void showLoginFailedMessage();

    void showLoginErrorMessage(Throwable t);

    void showPlanDialog(List<Plan> planList);

    void changeToMapActivity();

    void showServerChangeWindow();

    void logErrorMessage(Throwable t);
}
```

Εικόνα 14 Login Όψη

- **Παρουσιαστής**

```
interface Presenter {

    void setView(LoginActivityMVP.View view);

    void loginButtonClicked(SharedPreferences config);

    void initializeAppState(Plan plan, Config config, UserData userData,
                            UserPermissions permissions);

    void initializeOfflineAppState(String username);

    void getUserData(Plan plan);

    void getAppConfigFromServer(Plan plan, UserData userData, UserPermissions permissions);

    void updateAppConfig(Config config);

    void unsubscribe();

    void subscribe(Disposable d);
}
```

Εικόνα 15 Login Παρουσιαστής

- **Map:**
    - **Μοντέλου**

```
interface Model {
    Observable<LiveData> getAirData(String username, String password, int planId);

    Observable<LiveData> getNavalData(String username, String password, int planId);

    Observable<LiveData> getFriendlyGroundData(String username, String password, int planId);

    Observable<LiveData> getHostileGroundData(String username, String password, int planId);

    Observable<List<MapServer>> getUserBaseMapServers(String username, String password);

    Observable<List<MapServer>> getUserDefaultMapServer(String username, String password);

    Observable<List<MapServer>> getUserMapOverlays(String username, String password);

    Observable<List<MapServer>> getDefaultUserMapOverlays(String username, String password);

    void updateState(ImmutableAppState appState);

    ImmutableAppState getCurrentAppState();

    Completable saveOverlay(Overlay o, int overlayType, String fileName, String username,
                            int planId, String planName, long unitId, DateTime date);

    Observable<OverlayPOJO> getUserOverlayList(String username, long unitId);

    Observable<ExportableOverlay> loadOverlay(String fileName, String username, int planId,
                                              long unitId);

    Observable<WeatherData> getWeatherData(double lat, double lon);

    void unsubscribe();
```

Εικόνα 16 Map Μοντέλο 1

```
    void subscribe(Disposable d);

    Observable<Boolean> checkPermission(String username, String password, String permissionKey);

    Observable<Boolean> uploadUnitOverlay(String username, String password, long planId,
                                          long unitId, String date,
                                          String layerName, String overlay);

    Observable<String> downloadUnitOverlay(String username, String password, long planId,
                                           long unitId);

    Observable<String> downloadSuperiorOverlay(String username, String password, long unitId,
                                               long planId);

    Observable<OverlayList> downloadSubordinateOverlays(String username, String password,
                                                        long unitId, long planId);

    Observable<SymbolList> getBoundaries(String username, String password, long unitId,
                                         long planId);

    Observable<SymbolList> getMinefields(String username, String password, long unitId,
                                         long planId);

    Observable<SymbolList> getFacilities(String username, String password, long unitId,
                                         long planId, String facType);
}
```

Εικόνα 17 Map Μοντέλο 2

- **Όψη**

```
interface View {

    void sendToast(String message);

    void displayLiveDataOnMap(List<IFeature> featureList, int overlayID);

    void addFeatureToOverlay(IFeature feature, int overlayType);

    void addFeaturesToOverlay(List<IFeature> featuresList, int overlayType);

    void editLayerFeature(IFeature feature, int overlayType,
                          MapActivity.FeatureEditorListener listener);

    void removeFeatureFromOverlay(IFeature feature, int overlayType);

    void addOverlayToMap(Overlay overlay, int overlayType);

    void removeOverlayFromMap(int overlayType);

    void replaceOverlay(Overlay overlay, int overlayID);

    Overlay getOverlay(int overlayType);

    IMap getMap();

    void showUserOverlayList(List<OverlayPOJO> overlayList);

    void handleDataRetrievalError(int liveDataType);

    void showBaseMapDialog(List<MapServer> mapServerList);

    void showUserOverlayMapDialog(List<MapServer> mapServerList);

    void initializeMapState(List<MapServer> baseMapServer, List<MapServer> overlayList);

    void displayWeatherData(Point p);

    void changBaseMapService(MapServer server);
}
```

Εικόνα 18 Map Όψη

- **Παρουσιαστής**

```
interface Presenter {

    void setView(MapActivityMVP.View view);

    void requestLiveData(int type);

    void updateMapState(IMap map);

    void updateMapState(ImmutableMapState mapState);

    void unsubscribe();

    void subscribe(Disposable d);

    void saveOverlay(Overlay overlay, String filename);

    void loadOverlay(OverlayPOJO overlayPOJO);

    void getUserOverlayList();

    void logErrorMessage(Throwable throwable);

    void logMessage(String message);

    void getUserBaseMapServers();

    void getUserMapOverlays();

    void getDefaultMapServices();

    void downloadUnitOverlay();

    ImmutableAppState getCurrentAppState();

    void uploadUnitOverlay();

    void downloadSuperiorOverlay();

    void downloadSubordinateOverlays();

    void getWeatherData(double lat, double lon);

    void getBoundaries();

    void getMinefields();

    void getFacilities(String facType);

}
```

Εικόνα 19 Map Presenter

Παράλληλα έγινε χρήση του σχεδιαστικού πρότυπου Dependency Injection, το οποίο αποσυνδέει τις κλάσεις μεταξύ τους, παρέχοντας την εξάρτηση στην κλάση που την χρησιμοποιεί αντί να αναμένει από την κλάση να την δημιουργήσει κατά την εκτέλεση. Έτσι αν το αντικείμενο-εξάρτηση αλλάξει, οι αλλαγές στον κώδικα θα είναι ελάχιστες.

Εν συνεχεία χρησιμοποιήθηκε η βιβλιοθήκη RxJava για την παροχή δυνατότητας διαχείρισης ασύγχρονων ροών (Asynchronous streams) με λειτουργίες συναρτησιακού προγραμματισμού. Με αυτήν την βιβλιοθήκη η διαχείριση αιτήσεων και απαντήσεων γίνεται στο ίδιο σημείο, αφαιρώντας την πολυπλοκότητα του βασικού AsyncTask του Android, μειώνει τις πιθανότητες διαρροής μνήμη και επιτρέπει τον συνδυασμό μεθόδων, οδηγώντας σε βελτιωμένη απόκριση στην εφαρμογή.

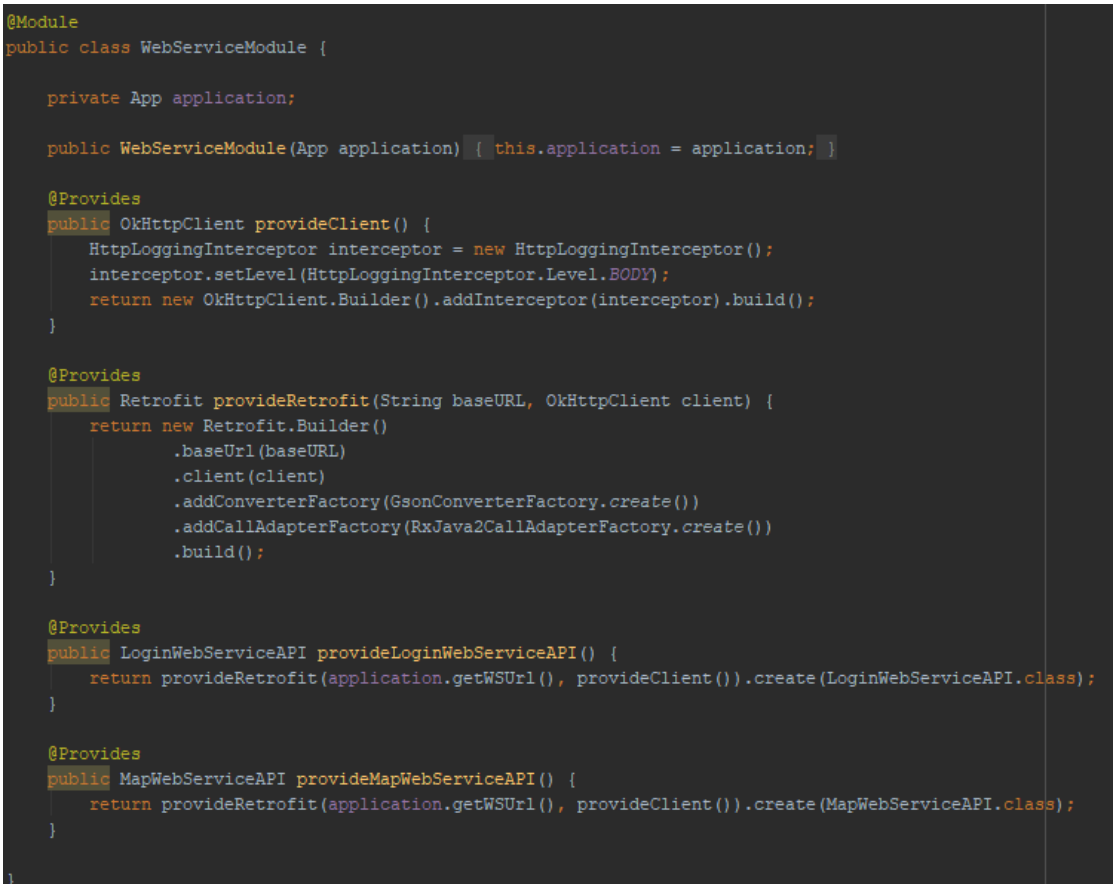
Για τον ομοιόμορφο τρόπο χειρισμού των κλήσεων Υπηρεσιών Δικτύου Rest, έχει δημιουργηθεί μια κλάση WSMessageBody, η οποία παρουσιάζεται παρακάτω:

```
private String username, password, permission, layerName, date, overlay, facType;
private long planId, userId;
private long unitId;
private double lat, lon;

//constructors, getters and setters
```

Εικόνα 20 WSMessageBody

Ανάλογα τα δεδομένα που θέλουμε να περάσουμε στο σώμα της POST Μεθόδου που θα καλέσουμε, η εφαρμογή θα γεμίσει τα κατάλληλα πεδία και θα τα αποστείλει για την εκτέλεση της μεθόδου.

Για την χρησιμοποίηση των Υπηρεσιών Δικτύου Rest από το Android χρησιμοποιείται η βιβλιοθήκη Retrofit. Ο Πελάτης της υπηρεσίας δημιουργείται κεντρικά στην εφαρμογή, και παρέχεται στα modules της εφαρμογής όπως παρακάτω:



```
@Module
public class WebServiceModule {

    private App application;

    public WebServiceModule(App application) { this.application = application; }

    @Provides
    public OkHttpClient provideClient() {
        HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();
        interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
        return new OkHttpClient.Builder().addInterceptor(interceptor).build();
    }

    @Provides
    public Retrofit provideRetrofit(String baseURL, OkHttpClient client) {
        return new Retrofit.Builder()
                .baseUrl(baseURL)
                .client(client)
                .addConverterFactory(GsonConverterFactory.create())
                .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
                .build();
    }

    @Provides
    public LoginWebServiceAPI provideLoginWebServiceAPI() {
        return provideRetrofit(application.getWSUrl(), provideClient()).create(LoginWebServiceAPI.class);
    }

    @Provides
    public MapWebServiceAPI provideMapWebServiceAPI() {
        return provideRetrofit(application.getWSUrl(), provideClient()).create(MapWebServiceAPI.class);
    }

}
```

Εικόνα 21 Παροχέας Retrofit

Η εφαρμογή έχει στηθεί σύμφωνα με το State Pattern, βάση του οποίου η εφαρμογή αλλάζει την συμπεριφορά της όταν αλλάζει η εσωτερική κατάσταση της. Όλες οι προηγούμενες καταστάσεις αποθηκεύονται σε Store, επιτρέποντας σε μελλοντική επέκταση τον έλεγχο αλλά και την μετάβαση σε προηγούμενη κατάσταση.

Τέλος, τα αντικείμενα τα οποία αποτελούν μεταβλητές του γενικότερου αντικειμένου AppState δημιουργούνται με την χρήση της βιβλιοθήκης Immutables, για να μην είναι δυνατή η κατά λάθος τροποποίηση μιας κατάστασης, με αποτέλεσμα να χαθεί μια ενδιάμεση φάση της εφαρμογής χωρίς να έχει αποθηκευτεί.

3.2 Υλοποίηση Συστήματος Αυθεντικοποίησης Χρήστη

Για την αυθεντικοποίηση του χρήστη απαιτείται η παροχή username και password. Μόλις ο χρήστης τα παράσχει, και πατήσει το κομβίο Login, εκτελούνται η παρακάτω μέθοδος:

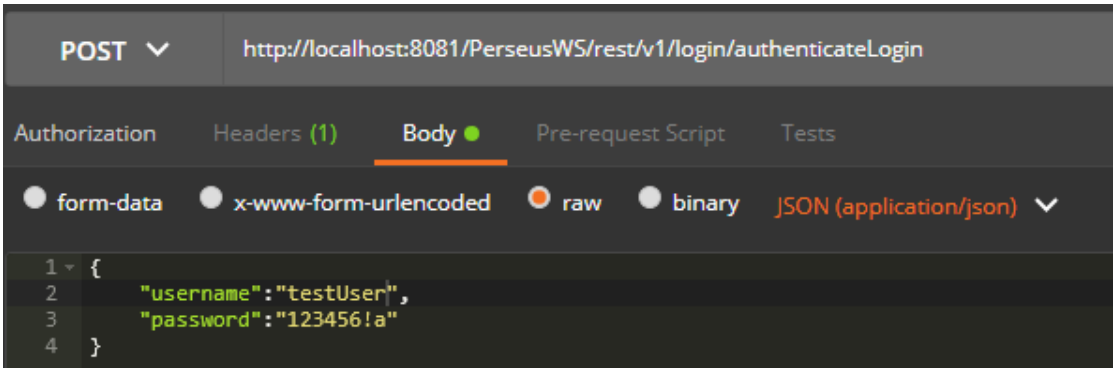
```
@Override
public void loginButtonClicked(SharedPreferences config) {
    if (view.getUsername().trim().isEmpty() || view.getPassword().trim().isEmpty()) {
        view.showInputErrorMessage();
    } else {
        subscribe(model.checkServerStatus(config.getString(Constants.SERVER_URL_KEY,  defValue: "10.0.2.2"),
                config.getInt(Constants.SEVER_PORT_KEY,  defValue: 8080))
                .subscribeOn(Schedulers.newThread())
                .doOnError(throwable -> view.showLoginErrorMessage(throwable))
                .subscribe(() ->
                        subscribe(model.login(view.getUsername(), view.getPassword())
                                .subscribeOn(Schedulers.newThread())
                                .subscribe(loginSuccess -> {
                                    if (loginSuccess) {
                                        subscribe(model.getPlans(view.getUsername(), view.getPassword())
                                                .subscribe(planLst -> view.showPlanDialog(planLst)
                                                        , throwable -> view.showLoginErrorMessage(throwable)));
                                    } else {
                                        view.showLoginFailedMessage();
                                    }
                                }, throwable -> {
                                    view.showLoginErrorMessage(throwable);
                                    view.showServerChangeWindow();
                                }))
                        , throwable -> {
                            view.logErrorMessage(throwable);
                            view.showServerChangeWindow();
                        }
                ));
    }
}
```

```
@Override
public Observable<Boolean> login(String username, String password) {
    WSMessageBody messageBody = new WSMessageBody(username, password);
    return loginWebServiceAPI.authenticateLogin(Constants.JSON_CONTENT_TYPE, messageBody);
}
```

```
@POST("v1/login/getUserData")
Observable<UserData> getUserData(@Header("Content-Type") String content_type,
                                 @Body WSMessageBody WSMessageBody);
```

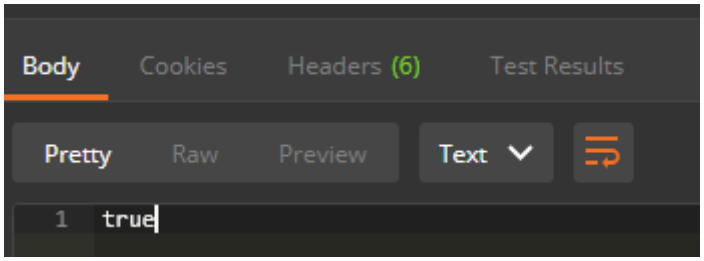
Εικόνα 22 Login Webservice

Η αποστολή του αιτήματος γίνεται μέσω Retrofit στο endpoint.



Εικόνα 23 Login Request

Και αφού ο Εξυπηρετητής επεξεργαστεί το αίτημα, αποστέλλει την απάντηση.



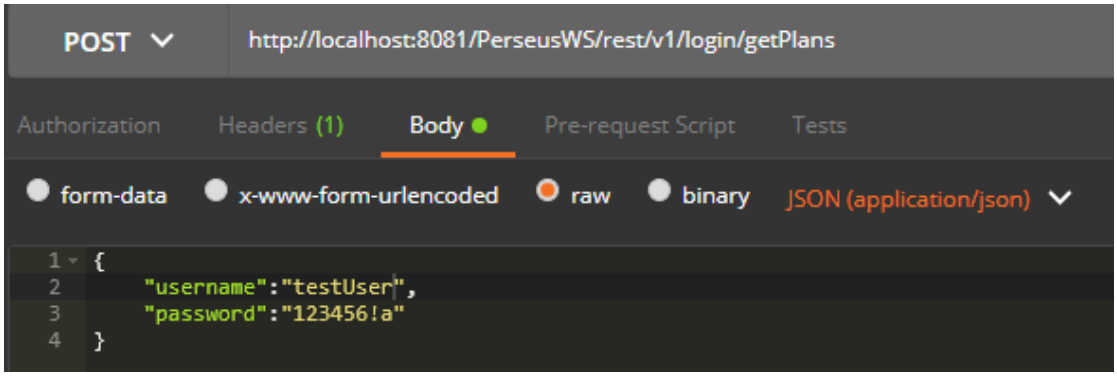
Εικόνα 24 Login Response

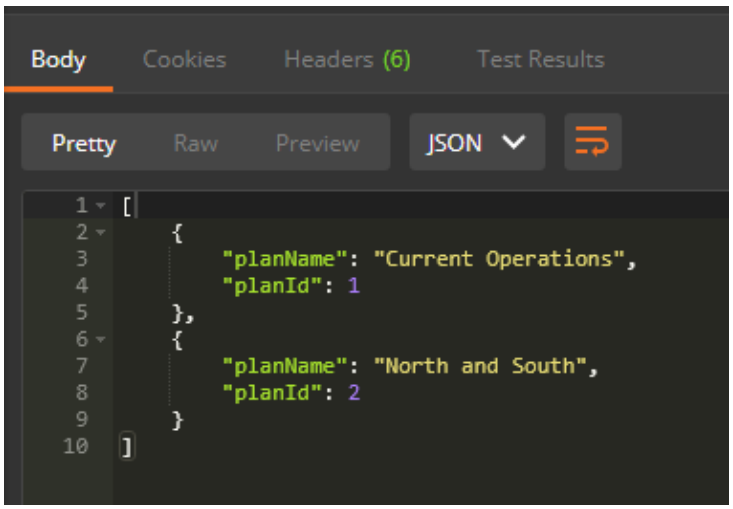
3.3 Είσοδος στην Εφαρμογή (Online)

Αφού αυθεντικοιποιηθεί ο χρήστης, η εφαρμογή ακολουθάει τα παρακάτω βήματα:

- Λαμβάνει τα διαθέσιμα σχέδια επιχειρήσεων (Plans) από την βάση.

```
@POST("v1/login/getPlans")
Observable<List<Plan>> getPlans(@Header("Content-Type") String content_type,
                                @Body WSMessageBody WSMessageBody);
```

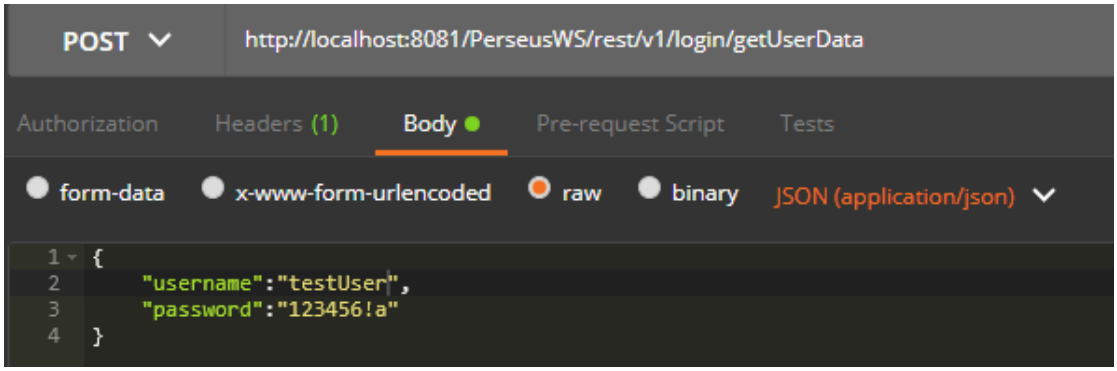


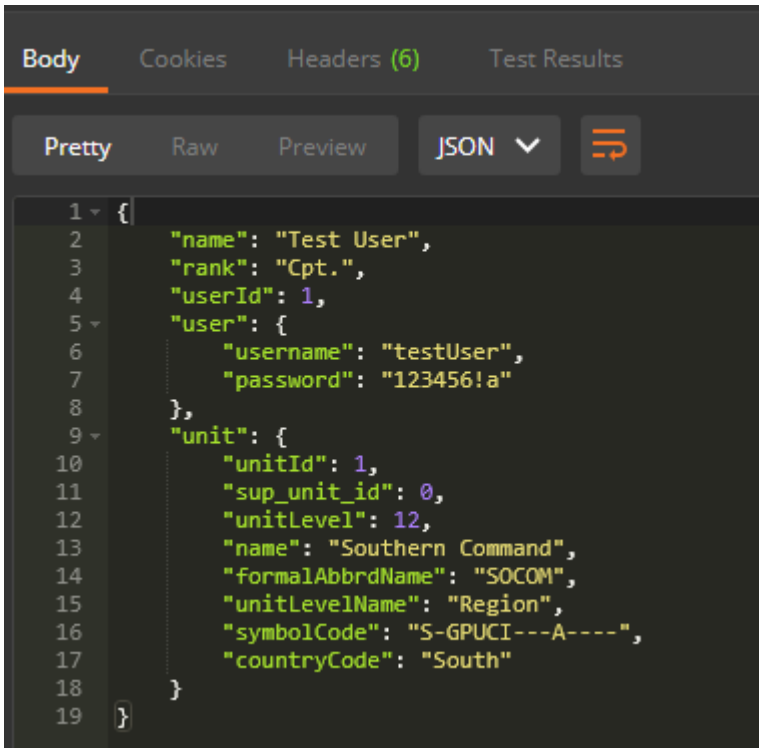


Εικόνα 25 getPlans Webservice

- Λαμβάνει τα δεδομένα του χρήστης:

```
@POST("v1/login/getUserData")
Observable<UserData> getUserData(@Header("Content-Type") String content_type,
                                 @Body WSMessageBody WSMessageBody);
```

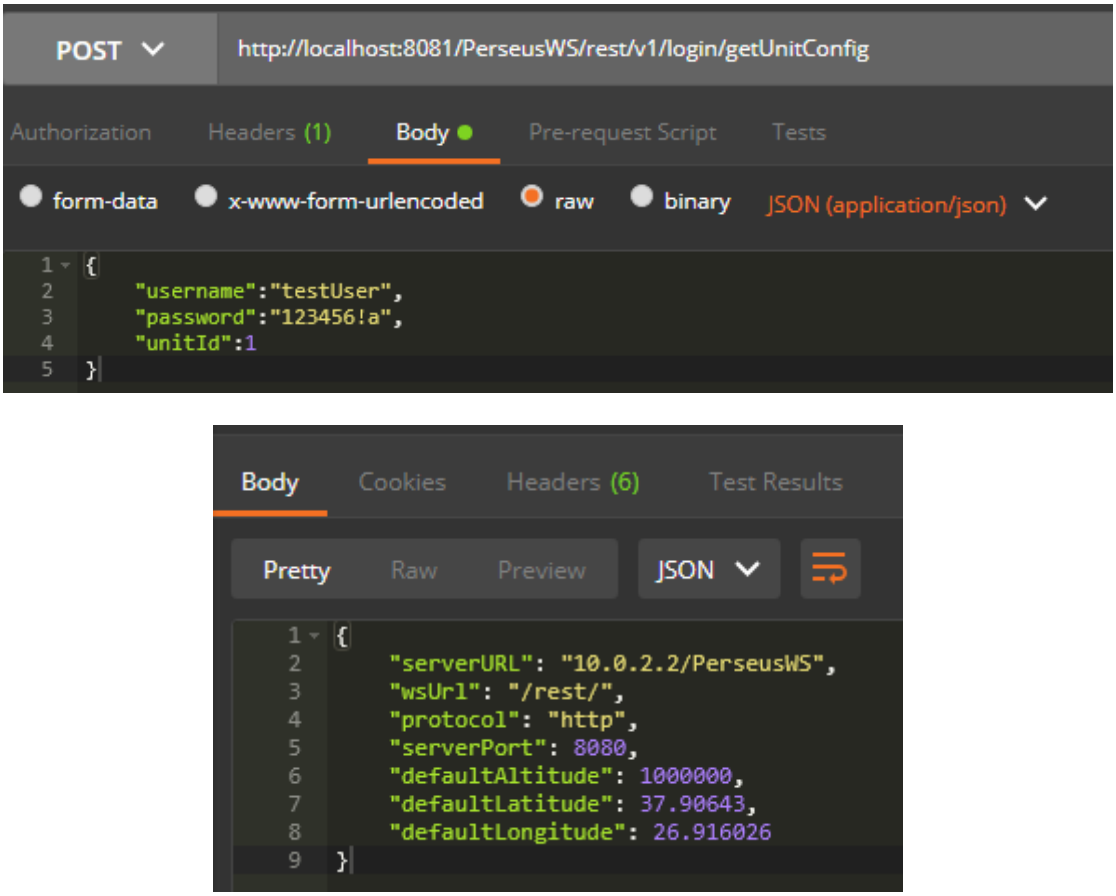




Εικόνα 26 getUserData Webservice

- Λαμβάνει τις ρυθμίσεις του Χρήστη, βάση της Μονάδας που ανήκει

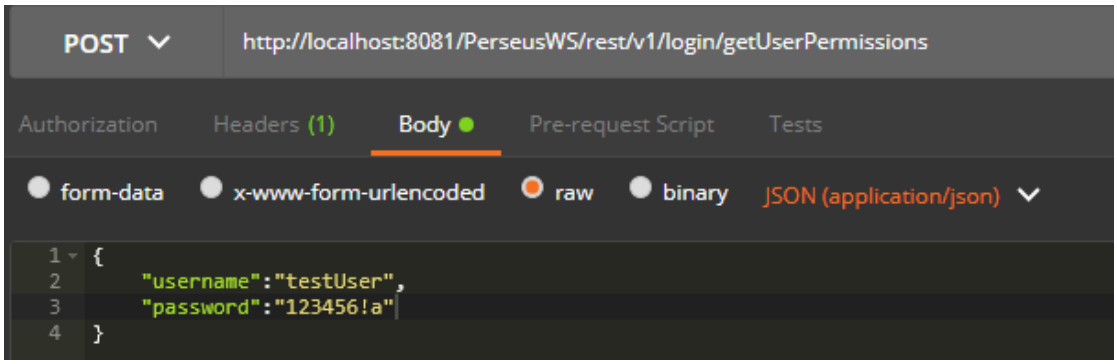
```
@POST("v1/login/getUnitConfig")
Observable<Config> getUnitConfig(@Header("Content-Type") String content_type,
                                 @Body WSMessageBody WSMessageBody);
```

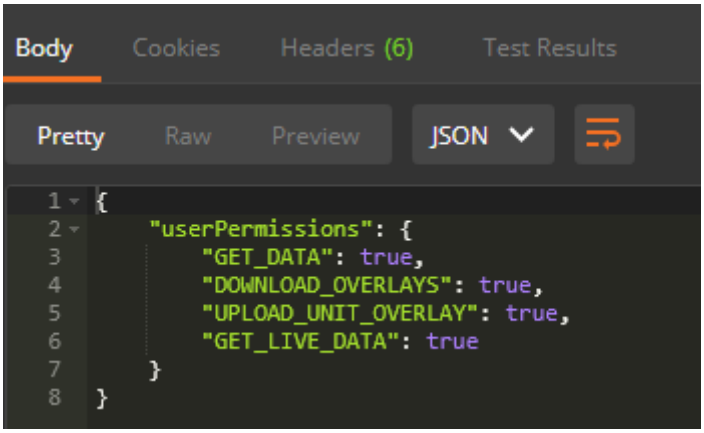


Εικόνα 27 getUserConfig Webservice

- Λαμβάνει τα δικαιώματα του Χρήστη:

```
@POST("v1/login/getUserPermissions")
Observable<UserPermissions> getUserPermissions(@Header("Content-Type") String content_type,
                                               @Body WSMessageBody WSMessageBody);
```





Εικόνα 28 getUserPermissions Webservice

- Ρυθμίζει και εκκινεί την εφαρμογή:

```
@Override
public void initializeAppState(Plan plan, Config config, UserData userData,
                               UserPermissions permissions) {
    ImmutableAppState state = ImmutableAppState.builder()
            .user(ImmutableUser.builder()
                    .userId(userData.getUserId())
                    .rank(userData.getRank())
                    .unit(ImmutableUnit.builder()
                            .unitFormalAbbrdName(userData.getUnitData().getFormalAbbrdName())
                            .unitFormalName(userData.getUnitData().getName())
                            .unitId(userData.getUnitData().getUnitId())
                            .unitLevel(userData.getUnitData().getUnitLevel())
                            .superiorUnit(userData.getUnitData().getSupUnitId())
                            .unitLevelName(userData.getUnitData().getUnitLevelName())
                            .countryCode(userData.getUnitData().getCountryCode())
                            .build())
                    .name(userData.getName())
                    .password(userData.getUser().getPassword())
                    .username(userData.getUser().getUsername())
                    .permissions(permissions)
                    .build())
            .plan(plan)
            .config(config)
            .stateId(0)
            .offlineMode(false)
            .build();

    if (model.initializeAppState(state)) {
        view.changeToMapActivity();
    }
}
```

Εικόνα 29 Κώδικας εκκίνησης Εφαρμογής

3.4 Αλλαγή εξυπηρετητή

Αν η εφαρμογή δεν μπορέσει να επικοινωνήσει με τον Εξυπηρετητή, δίνει την δυνατότητα στον χρήστη είτε να κάνει είσοδο σε offline mode ή να αλλάξει διεύθυνση εξυπηρετητή.

```
@Override
public void showServerChangeWindow() {
    runOnUiThread(() -> {
        AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
        builder.setTitle("Server not Reachable");

        final EditText input = new EditText( context: this);

        input.setInputType(InputType.TYPE_CLASS_TEXT);
        builder.setView(input);

        builder.setPositiveButton( text: "Change Server", (dialog, which) -> {
            String uri = input.getText().toString();
            if (uri.length() > 0) {
                try {
                    URL URL = new URL(uri);

                    String protocol = URL.getProtocol();
                    int port = URL.getPort();
                    String serverURL = URL.getHost();
                    String wsURL = URL.getPath();

                    Config config = new Config();
                    config.setProtocol(protocol);
                    config.setServerPort(port);
                    config.setServerURL(serverURL);
                    config.setWsUrl(wsURL);
                    presenter.updateAppConfig(config);

                    dialog.dismiss();

                    AlarmManager mgr = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
                    mgr.set(AlarmManager.RTC,  triggerAtMillis: System.currentTimeMillis() + 1000,
                            PendingIntent.getActivity(this.getBaseContext(),  requestCode: 0,
                                    new Intent(getIntent()), getIntent().getFlags()));
                    android.os.Process.killProcess(android.os.Process.myPid());
                } catch (MalformedURLException e) {
                    e.printStackTrace();
                }
            } else {
                Toast.makeText( context: this,
                        text: "Please enter a proper Server URL to continue",
                        Toast.LENGTH_SHORT).show();
            }
        });

        builder.setNegativeButton( text: "Log in Offline Mode.", (dialog, which) -> {
            dialog.dismiss();
            presenter.initializeOfflineAppState( username: "offline");
        });

        builder.show();
    });
}
```

Εικόνα 30 Κώδικας Αλλαγή Εξυπηρετητή

```java
@Override
public void initializeOfflineAppState(String username) {
    Plan offlinePlan = new Plan();
    offlinePlan.setPlanId(-1);
    offlinePlan.setPlanName("Offline Plan");

    Config offlineConfing = new Config();
    offlineConfing.setDefaultAltitude(1000000);
    offlineConfing.setDefaultLatitude(38);
    offlineConfing.setDefaultLongitude(28);

    ImmutableAppState state = ImmutableAppState.builder()
            .user(ImmutableUser.builder()
                    .userId(-11)
                    .rank(" ")
                    .unit(ImmutableUnit.builder()
                            .unitFormalAbbrdName(" ")
                            .unitFormalName(" ")
                            .unitId(-11)
                            .unitLevel(-1)
                            .superiorUnit(-1)
                            .unitLevelName(" ")
                            .countryCode(" ")
                            .build())
                    .name("Offline User")
                    .password(" ")
                    .username(username)
                    .permissions(new UserPermissions())
                    .build())
            .plan(offlinePlan)
            .config(offlineConfing)
            .stateId(0)
            .offlineMode(true)
            .build();

    if (model.initializeAppState(state)) {
        view.changeToMapActivity();
    }
}
```

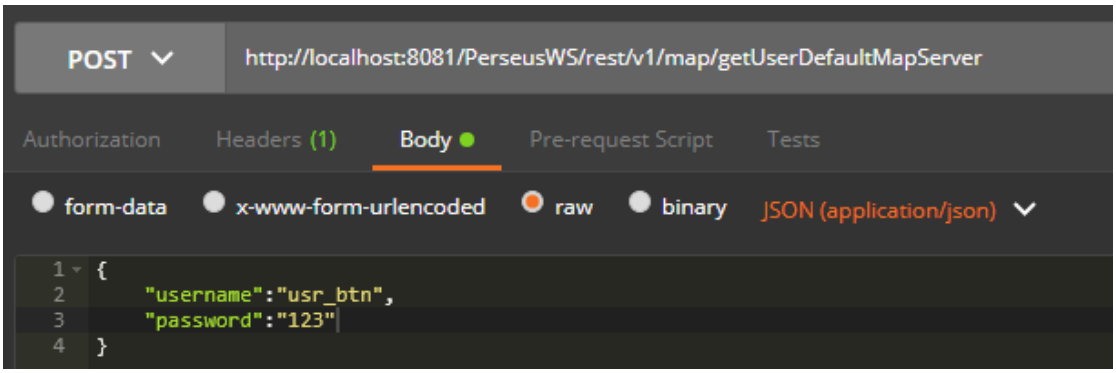
Εικόνα 31 Κώδικας Εκκίνησης σε Offline Mode

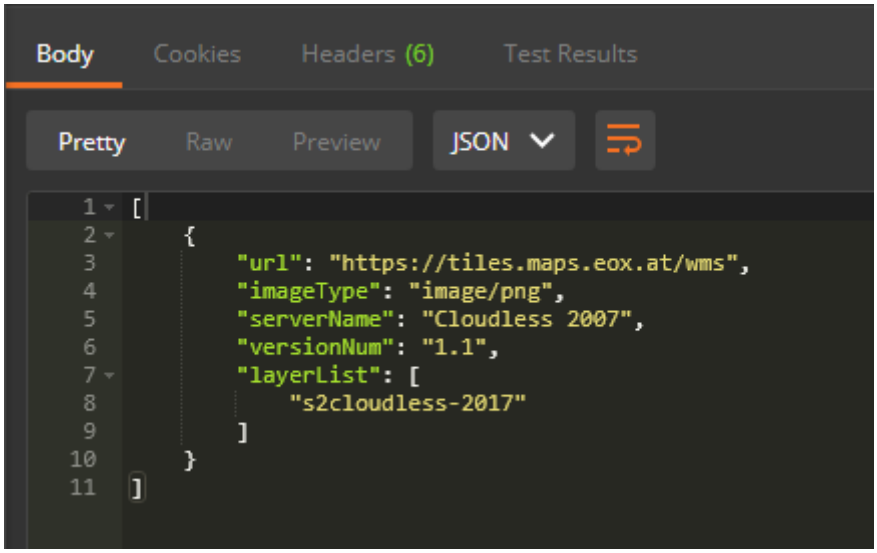
3.5 Υλοποίηση λειτουργίας Χάρτη

3.5.1 Επιλογή προκαθορισμένου Χάρτη Υποβάθρου

Η εφαρμογή επικοινωνεί με τον Εξυπηρετητή και λαμβάνει τα στοιχεία του προκαθορισμένου εξυπηρετητή χαρτών του χρήστη.

```
@POST("v1/map/getUserDefaultMapServer")
Observable<List<MapServer>> getUserDefaultMapServer(@Header("Content-Type") String content_type,
                                                    @Body WSMessageBody WSMessageBody);
```

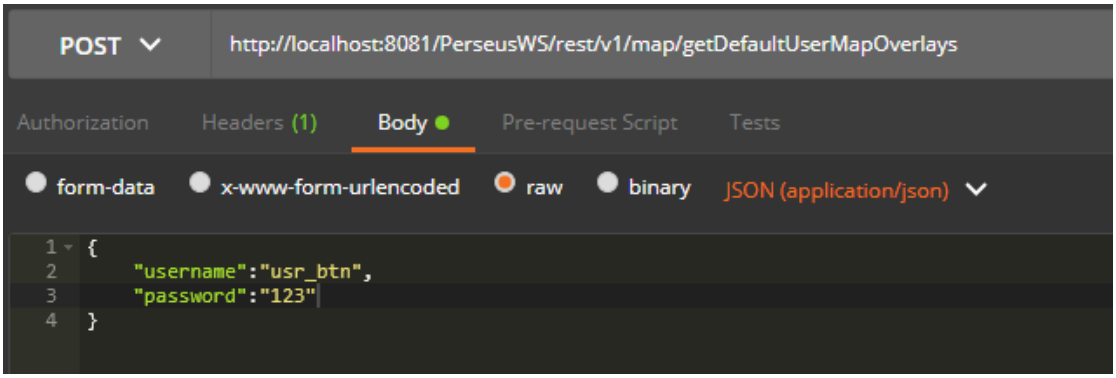




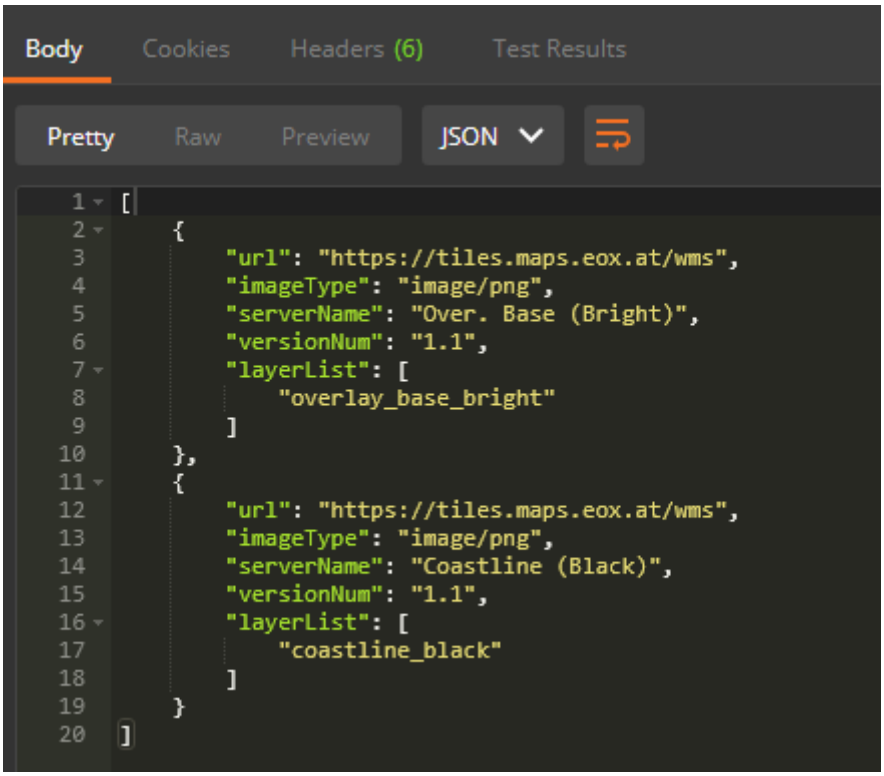
Εικόνα 32 getDefaultMapServer Webservice

3.5.2 **Επιλογή προκαθορισμένων Διαφανών Χάρτη**

Η εφαρμογή επικοινωνεί με τον Εξυπηρετητή και λαμβάνει τα στοιχεία των προκαθορισμένων εξυπηρετητών διαφανών του χρήστη.



```
@POST("v1/map/getDefaultUserMapOverlays")
Observable<List<MapServer>> getDefaultUserMapOverlays(
        @Header("Content-Type") String content_type,
        @Body WSMessageBody WSMessageBody);
```



Εικόνα 33 getDefaultMapOverlays Webservice

3.5.3 **Διαθεσιμότητα επιλογών**

Ανάλογα της κατάστασης λειτουργίας της εφαρμογής (offline ή online) καθώς και των δικαιωμάτων του Χρήστη, η εφαρμογή μπορεί να μην δίνει πρόσβαση σε μία ή περισσότερες λειτουργίες της:

```
private void checkUserPermissions(UserPermissions permissions, boolean offlineMode) {
    HashMap<String, Boolean> permissionMap = permissions.getUserPermissions();
    if (offlineMode || permissionMap == null || permissionMap.isEmpty()
            || !permissionMap.containsKey("GET_DATA") || (!permissionMap.get("GET_DATA"))) {
        for (int i = 0; i < MapUIConstants.SWITCHES_COUNT; i++) {
            if (i != MapUIConstants.LIVE_AIR_SWITCH && i != MapUIConstants.LIVE_FRIEND_SWITCH
                    && i != MapUIConstants.LIVE_HOSTILE_SWITCH
                    && i != MapUIConstants.LIVE_NAVAL_SWITCH) {
                getNavigationDrawerSwitch(i).setVisibility(View.INVISIBLE);
            }
        }
    }

    if (offlineMode || permissionMap == null || permissionMap.isEmpty()
            || !permissionMap.containsKey("GET_LIVE_DATA")
            || (!permissionMap.get("GET_LIVE_DATA"))) {
        for (int i = 0; i < MapUIConstants.SWITCHES_COUNT; i++) {
            if (i == MapUIConstants.LIVE_AIR_SWITCH || i == MapUIConstants.LIVE_FRIEND_SWITCH
                    || i == MapUIConstants.LIVE_HOSTILE_SWITCH
                    || i == MapUIConstants.LIVE_NAVAL_SWITCH) {
                getNavigationDrawerSwitch(i).setVisibility(View.INVISIBLE);
            }
        }
    }

    if (offlineMode || permissionMap == null || permissionMap.isEmpty()
            || !permissionMap.containsKey("DOWNLOAD_OVERLAYS")
            || (!permissionMap.get("DOWNLOAD_OVERLAYS"))) {
        NavigationView navigationView = findViewById(R.id.nav_view);
        navigationView.getMenu().findItem(R.id.download_unit_overlay).setVisible(false);
        navigationView.getMenu().findItem(R.id.subordinate_unit_overlays).setVisible(false);
        navigationView.getMenu().findItem(R.id.superior_unit_overlay).setVisible(false);
    }

    if (offlineMode || permissionMap == null || permissionMap.isEmpty()
            || !permissionMap.containsKey("UPLOAD_UNIT_OVERLAY")
            || (!permissionMap.get("UPLOAD_UNIT_OVERLAY"))) {
        NavigationView navigationView = findViewById(R.id.nav_view);
        navigationView.getMenu().findItem(R.id.upload_unit_overlay).setVisible(false);
    }
}
```

Εικόνα 34 Κώδικας Προσαρμογής Επιλογών

3.5.4 **Επιλογές Ανάκτησης Στατικών Δεδομένων**

Η Εφαρμογή δίνει την δυνατότητα στον χρήστη να ανακτήσει τα παρακάτω στατικά δεδομένα επιχειρησιακού ενδιαφέροντος:

- Airports (Αεροδρόμια)
- Harbours (Λιμάνια)
- Bridges (Γέφυρες)
- Hospitals (Νοσοκομεία)

- Friendly Military Bases (Στρατιωτικές Βάσεις)

Τα παραπάνω δεδομένα καλούνται με παρόμοιο τρόπο, αλλάζοντας μόνο τον τύπο της εγκατάστασης (facType). Για αυτό θα επιδειχθεί μόνο η λειτουργία για τα αεροδρόμια:

```
@POST("v1/operationalData/getFacilities")
Observable<SymbolList> getFacilities(@Header("Content-Type") String content_type,
                                     @Body WSMessageBody WSMessageBody);
```

POST http://localhost:8081/PerseusWS/rest/v1/operationalData/getFacilities

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "username":"testUser",
    "password":"123456!a",
    "unitId":1,
    "planId":1,
    "facType":"AIRPRT"
}
```

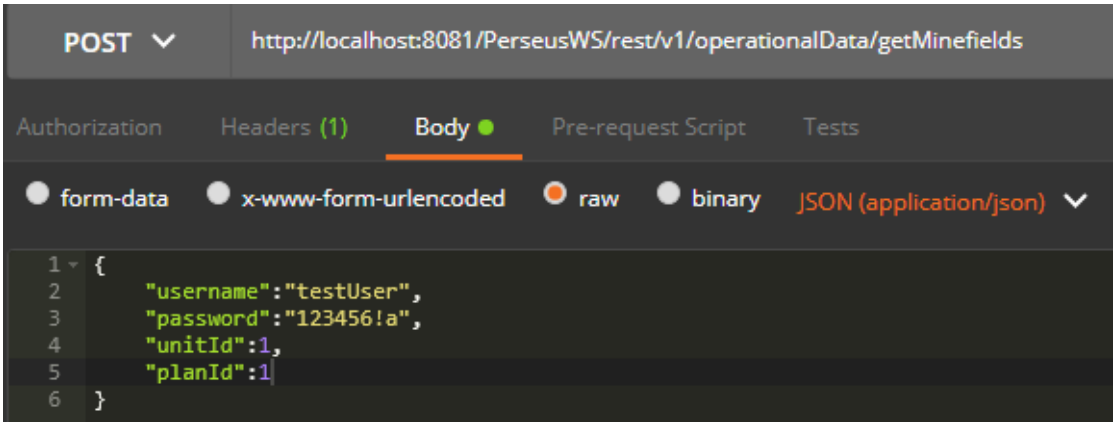
Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON


```
{
    "symbolLst": [
        {
            "higherFormationName": "1st Brigade",
            "uniqueDesignator1": "Airport  0",
            "uniqueDesignator2": null,
            "reducedOrReinforced": null,
            "symbolCode": "SFGPIBA---H----",
            "position": {
                "lat": 38.530899,
                "lon": 12.976832,
                "altitude": 0
            },
            "positions": null,
            "featureType": null
        },
        {
            "higherFormationName": "1st Brigade",
            "uniqueDesignator1": "Airport  1",
            "uniqueDesignator2": null,
            "reducedOrReinforced": null,
            "symbolCode": "SFGPIBA---H----",
            "position": {
                "lat": 39.700586,
                "lon": 14.22815,
                "altitude": 0
            },
            "positions": null,
            "featureType": null
        },
```


Εικόνα 35 getFacilites Webservice

- Minefields (Ναρκοπέδια)



Εικόνα 36 getMinefields Response

3.5.5 **Επιλογές Ανάκτησης Δυναμικών Δεδομένων**

Η εφαρμογή παρέχει την δυνατότητα στον χρήστη να παρακολουθεί την εξέλιξη των επιχειρήσεων μέσω της απεικόνισης δυναμικών δεδομένων που περιλαμβάνουν:

- Friendly Ground Picture (Φίλιες Επίγειες Δυνάμεις)
- Hostile Ground Picture (Εχθρικές Επίγειες Δυνάμεις)
- Air Picture (Αεροπορική Εικόνα)
- Naval Picture (Ναυτική Εικόνα)

Για αυτά τα δεδομένα χρησιμοποιείται η δυνατότητα που μας δίνει η RxJava 2 να παρακολουθούμε ένα Observalbe για καθορισμένο χρονικό διάστημα, σε κάθε πέρας του οποιου μπορούμε να λαμβάνουμε την τρέχουσα εικόνα των παραπάνω δεδομένων.

Επιδεικνύεται η παραπάνω λειτουργία για τα αεροπορικά δεδομένα (Air Picture).

```
public void requestLiveData(int overlayId) {
    Overlay overlay = new Overlay();
    overlay.setName(getOverlayNameByID(overlayId));
    addOverlayToMap(overlay, overlayId);

    int count = 60;

    switch (overlayId) {
        case MapUIConstants.LIVE_AIR_OVERLAY:
            count = 6;
            break;
        case MapUIConstants.LIVE_FRIEND_SWITCH:
            count = 20;
            break;
        case MapUIConstants.LIVE_NAVAL_OVERLAY:
            count = 30;
            break;
    }

    Disposable live = Observable.interval(count, TimeUnit.SECONDS)
            .startWith(11)
            .doOnNext(l -> {
                if (overlayId == MapUIConstants.LIVE_AIR_OVERLAY
                        && getNavigationDrawerSwitch(MapUIConstants.LIVE_AIR_SWITCH)
                        .isChecked()) {
                    presenter.requestLiveData(overlayId);
                } else if (overlayId == MapUIConstants.LIVE_NAVAL_OVERLAY
                        && getNavigationDrawerSwitch(MapUIConstants.LIVE_NAVAL_SWITCH)
                        .isChecked()) {
                            presenter.requestLiveData(overlayId);
                } else if (overlayId == MapUIConstants.LIVE_FRIEND_OVERLAY
                        && getNavigationDrawerSwitch(MapUIConstants.LIVE_FRIEND_SWITCH)
                        .isChecked()) {
                            presenter.requestLiveData(overlayId);
                } else if (overlayId == MapUIConstants.LIVE_HOSTILE_OVERLAY
                        && getNavigationDrawerSwitch(MapUIConstants.LIVE_HOSTILE_SWITCH)
                        .isChecked()) {
                            presenter.requestLiveData(overlayId);
                        }
                    }
            )
            .doOnError(throwable -> {
                Log.e(TAG, throwable.getMessage());
            })
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe();


    disposableArray[overlayId] = live;
}
```

```
@POST("v1/liveData/getAirData")
Observable<LiveData> getAirData(@Header("Content-Type") String content_type,
                                @Body WSMessageBody WSMessageBody);
```

POST http://localhost:8081/PerseusWS/rest/v1/liveData/getAirData

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "username":"testUser",
    "password":"123456!a"
}
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON


```
{
    "unitLst": [
        {
            "unitID": "295f9040-ffb3-475e-b3cd-c85f957cc015",
            "currentCoord": {
                "lon": 24.988190184966754,
                "altitude": 0,
                "lat": 37.98652685609551
            },
            "direction": 122,
            "speed": 10889,
            "modifierOne": "ultrices",
            "modifierTwo": "cursus",
            "country": "NR",
            "symbolCode": "SHAPMFF--------",
            "type": "AIR"
        },
        {
            "unitID": "bc5761b7-d740-4401-ae3e-5fd5719f8b09",
            "currentCoord": {
                "lon": 23.428449462782172,
                "altitude": 0,
                "lat": 41.04855711306897
            },
            "direction": 20,
            "speed": 4413,
            "modifierOne": "maiorum",
            "modifierTwo": "legimus",
            "country": "SO",
            "symbolCode": "SFAPMFF--------",
            "type": "AIR"
        },
        {
            "unitID": "81559e6f-d35a-468c-8b29-9a153197c84f",
            "currentCoord": {
                "lon": 23.760114662200074,
                "altitude": 0,
                "lat": 37.207051069806276
            },
            "direction": 54,
            "speed": 8811,
            "modifierOne": "efficiantur",
            "modifierTwo": "quod",
            "country": "NR",
            "symbolCode": "SHAPMFF--------",
            "type": "AIR"
        },
```


Εικόνα 37 requestLiveData Webservice

3.5.6 **Επιλογές Επιχειρησιακών Δεδομένων**

Παρέχονται στον χρήστη η δυνατότητα να ανακτήσει δεδομένα άμεσου ενδιαφέροντος για την επιχειρησιακή σχεδίαση. Αυτά τα δεδομένα αφορούν:

- Boundaries (Όρια Μονάδων)

```
@POST("v1/operationalData/getBoundaries")
Observable<SymbolList> getBoundaries(@Header("Content-Type") String content_type,
                                     @Body WSMessageBody WSMessageBody);
```

POST http://localhost:8081/PerseusWS/rest/v1/operationalData/getBoundaries

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "username":"testUser",
    "password":"123456!a",
    "unitId":1,
    "planId":1
}
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON


```
{
    "symbolLst": [
        {
            "higherFormationName": null,
            "uniqueDesignator1": "1 BTN",
            "uniqueDesignator2": "1 BRIG",
            "reducedOrReinforced": null,
            "symbolCode": "GFG*GLB------X",
            "position": null,
            "positions": [
                {
                    "lat": 38.284354,
                    "lon": 22.837394,
                    "altitude": 0
                },
                {
                    "lat": 38.286236,
                    "lon": 22.861253,
                    "altitude": 0
                },
                {
                    "lat": 38.288986,
                    "lon": 22.873453,
                    "altitude": 0
                },
                {
                    "lat": 38.28871,
                    "lon": 22.906651,
                    "altitude": 0
                },
                {
                    "lat": 38.287907,
                    "lon": 22.939682,
                    "altitude": 0
                }
            ],
            "featureType": null
        },
```


Εικόνα 38 getBoundaries Webservice

- Weather (Καιρός)

```
@POST("v1/operationalData/getWeatherData")
Observable<WeatherData> getWeatherData(@Header("Content-Type") String content_type,
                                       @Body WSMessageBody WSMessageBody);
```

POST http://localhost:8081/PerseusWS/rest/v1/operationalData/getWeatherData

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "username":"testUser",
    "password":"123456!a",
    "lat":38.22,
    "lon":27.35
}
```

Body Cookies Headers (6) Test Results

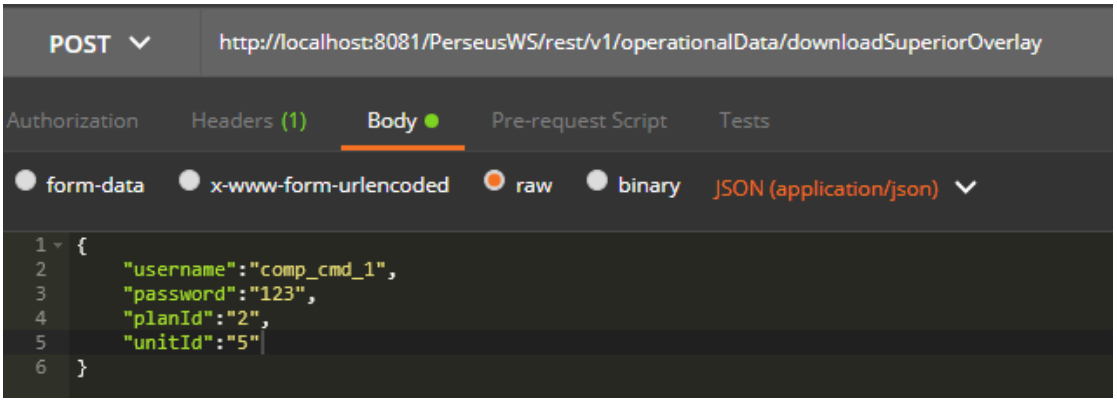
Pretty Raw Preview JSON


```
{
    "coord": {
        "lon": 38.22,
        "lat": 38.22
    },
    "weather": [
        {
            "id": 800,
            "main": "Clear",
            "description": "clear sky",
            "icon": "01n"
        }
    ],
    "base": "stations",
    "main": {
        "temp": -1,
        "pressure": 1020,
        "humidity": 86,
        "temp_min": -1,
        "temp_max": -1
    },
    "visibility": 10000,
    "wind": {
        "speed": 2.1,
        "deg": 220
    },
    "clouds": {
        "all": 0
    },
    "dt": 1550780400,
    "sys": {
        "type": 1,
        "id": 6961,
        "message": 0.0036,
        "country": "TR",
        "sunrise": 1550722166,
        "sunset": 1550761945
    },
    "id": 296791,
    "name": "Yesilyurt",
    "cod": 200
}
```


Εικόνα 39 getWeatherData Webservice

- Superior Unit Overlay (Σχέδιο Επιχειρήσεων Προϊστάμενου)

```
@POST("v1/operationalData/downloadSuperiorOverlay")
Observable<String> downloadSuperiorOverlay(@Header("Content-Type") String content_type,
                                           @Body WSMessageBody WSMessageBody);
```



Εικόνα 40 downloadSuperiorOverlay Webservice

- Subordinate Unit Overlays (Σχέδια Επιχειρήσεων Άμεσα Υφισταμένων)

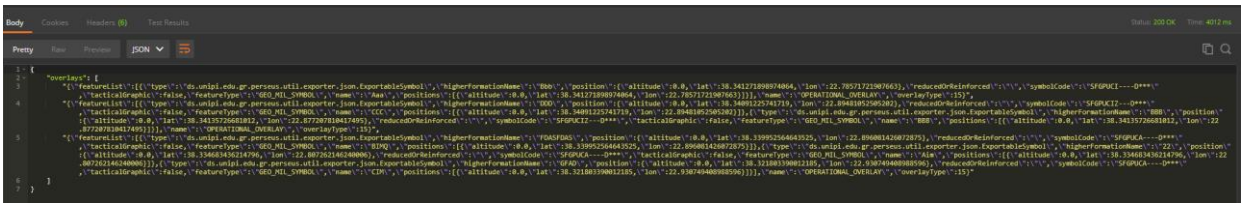
```
@POST("v1/operationalData/downloadSubordinateOverlays")
Observable<OverlayList> downloadSubordinateOverlays(@Header("Content-Type") String content_type,
                                                    @Body WSMessageBody WSMessageBody);
```

POST http://localhost:8081/PerseusWS/rest/v1/operationalData/downloadSubordinateOverlays

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "username":"usr_btn",
    "password":"123",
    "permissionKey":"GET_DATA",
    "unitId":4,
    "planId":2
}
```



Εικόνα 41 downloadSubordinateOverlays Webservice

3.5.7 Επιλογές Τοπικής Αποθήκευσης και Ανάκτησης Διαφανών Επιχειρήσεων

Η εφαρμογή τόσο στον online τρόπο λειτουργίας όσο και στον offline τρόπο λειτουργίας δίνει στον χρήστη τις παρακάτω δυνατότητες:

- Δυνατότητα να αποθηκεύσει το διαφανές επιχειρήσεων στο οποίο εργάζεται

```java
@Override
public Completable saveOverlay(String fileName, String username, int planId,
                               String planName, long unitId, DateTime date,
                               String overlayJSON) {
    return Completable.fromAction(() -> {
        MutableDocument overlay = new MutableDocument()
                .setDouble("app_version", Constants.APP_VERSION)
                .setString("filename", fileName)
                .setString("username", username)
                .setString("date", date.toString())
                .setInt("planId", planId)
                .setString("planName", planName)
                .setLong("unitId", unitId)
                .setString("json", overlayJSON);
        getDatabase().save(overlay);
    });
}
```

Εικόνα 42 Κώδικας αποθήκευσης Διαφανούς

- Δυνατότητα να ανακτήσει παλαιότερα αποθηκευμένο διαφανές, από μια λίστα αποθηκευμένων διαφανών.

```java
@Override
public Observable<OverlayPOJO> getUserOverlayList(String username, long unitId) {
    Query query = QueryBuilder
            .select(SelectResult.property("filename"),
                    SelectResult.property("planId"),
                    SelectResult.property("planName"),
                    SelectResult.property("unitId"),
                    SelectResult.property("date"))
            .from(DataSource.database(getDatabase()))
            .where(Expression.property("username")
                    .equalTo(Expression.string(username))
                    .and(Expression.property("unitId")
                            .equalTo(Expression.longValue(unitId))
                            .or(Expression.property("unitId")
                                    .equalTo(Expression.longValue(-1)))));

    List<OverlayPOJO> overlayList = new ArrayList<>();
    try {
        ResultSet rs = query.execute();
        for (Result r : rs) {
            OverlayPOJO opj = new OverlayPOJO();
            opj.setUsername(username);
            opj.setUnitId(unitId);
            opj.setPlandId(r.getInt( key: "planId"));
            opj.setPlanName(r.getString( key: "planName"));
            opj.setFileName(r.getString( key: "filename"));
            opj.setDate(r.getString( key: "date"));
            opj.setUnitId(r.getLong( key: "unitId"));
            overlayList.add(opj);
        }
    } catch (CouchbaseLiteException ex) {
        Log.e(TAG, ex.getMessage());
    }

    return Observable.fromIterable(overlayList);
}
```

Εικόνα 43 Κώδικας λήψης Διαφανών

```
@Override
public Observable<ExportableOverlay> loadOverlay(String fileName, String username,
                                                 long unitId, int planId) {
    Query query = QueryBuilder
            .select(SelectResult.property("json"))
            .from(DataSource.database(getDatabase()))
            .where(Expression.property("username")
                    .equalTo(Expression.string(username))
                    .and(Expression.property("unitId")
                            .equalTo(Expression.longValue(unitId)))
                    .and(Expression.property("filename")
                            .equalTo(Expression.string(fileName))));

    String overlayJSON = "";
    try {
        ResultSet rs = query.execute();
        for (Result r : rs) {
            overlayJSON = r.getString( key: "json");
        }

    } catch (CouchbaseLiteException ex) {
        Log.e(TAG, ex.getMessage());
    }

    return Observable.just(JSONUtil.convertJSONToOverlay(overlayJSON));
}
```

Εικόνα 44 Κώδικας Φόρτωσης Διαφανούς

- Εκκαθάριση όλων των στοιχείων από το τρέχων διαφανές επιχειρήσεων.

```
public void clearOperationalOverlay() {
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Clear Operational Overlay");
    builder.setMessage("This will remove all symbols" +
            " from the Current operational Overlay. Do you want to proceed?");

    builder.setPositiveButton( text: "OK",
            (dialog, which) -> replaceOverlay(new Overlay(),
                    MapUIConstants.OPERATIONAL_OVERLAY));

    builder.setNegativeButton( text: "Cancel", (dialog, which) -> dialog.cancel());

    builder.show();
}
```

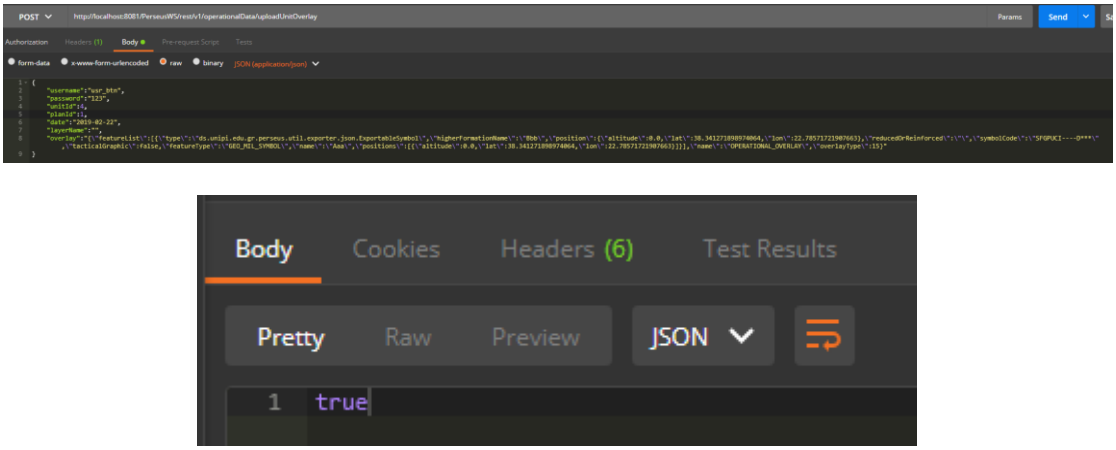
Εικόνα 45 Κώδικας Εκκαθάρισης Διαφανούς

3.5.1 **Επιλογές Δικτυακού Ορισμού και Λήψης Διαφανούς Επιχειρήσεων Μονάδας**

Η εφαρμογή στον online τρόπο λειτουργίας δίνει στον χρήστη τις δυνατότητες:

- Upload Unit Overlay (Ορισμός τρέχοντος διαφανούς ως Διαφανές Επιχείρησης Μονάδας)

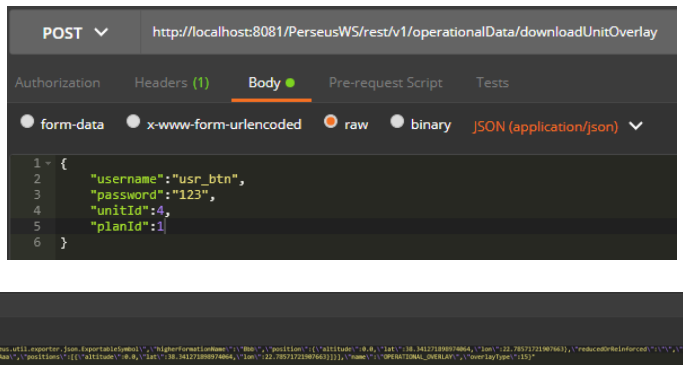
```
@POST("v1/operationalData/uploadUnitOverlay")
Observable<Boolean> uploadUnitOverlay(@Header("Content-Type") String content_type,
                                      @Body WSMessageBody WSMessageBody);
```



Εικόνα 46 uploadUnitOverlay Webservice

- Download Unit Overlay (Λήψη Διαφανούς Επιχείρησης Μονάδας ως τρέχων Διαφανές)

```
@POST("v1/operationalData/downloadUnitOverlay")
Observable<String> downloadUnitOverlay(@Header("Content-Type") String content_type,
                                       @Body WSMessageBody WSMessageBody);
```



Εικόνα 47 downloadUnitOverlay Webservice

3.5.2 **Επιλογές Χαρτών Χρήστη**

Η εφαρμογή δίνει την δυνατότητα στον Χρήστη να επιλέξει:

- Επιθυμητό Χάρτη Υποβάθρου

```
@POST("v1/map/getUserBaseMapServers")
Observable<List<MapServer>> getUserBaseMapServers(@Header("Content-Type") String content_type,
                                                  @Body WSMessageBody WSMessageBody);
```

POST http://localhost:8081/PerseusWS/rest/v1/map/getUserBaseMapServers

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "username":"usr_btn",
    "password":"123"
}
```

Body Cookies Headers (6) Test Results

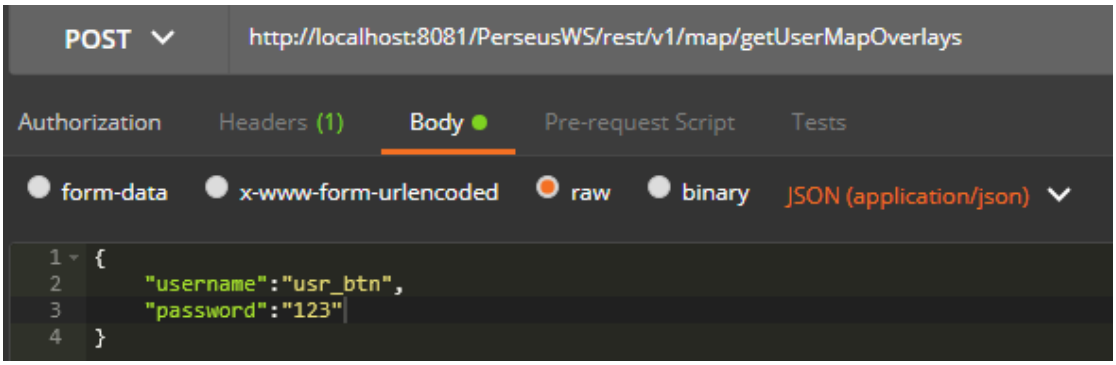
Pretty Raw Preview JSON

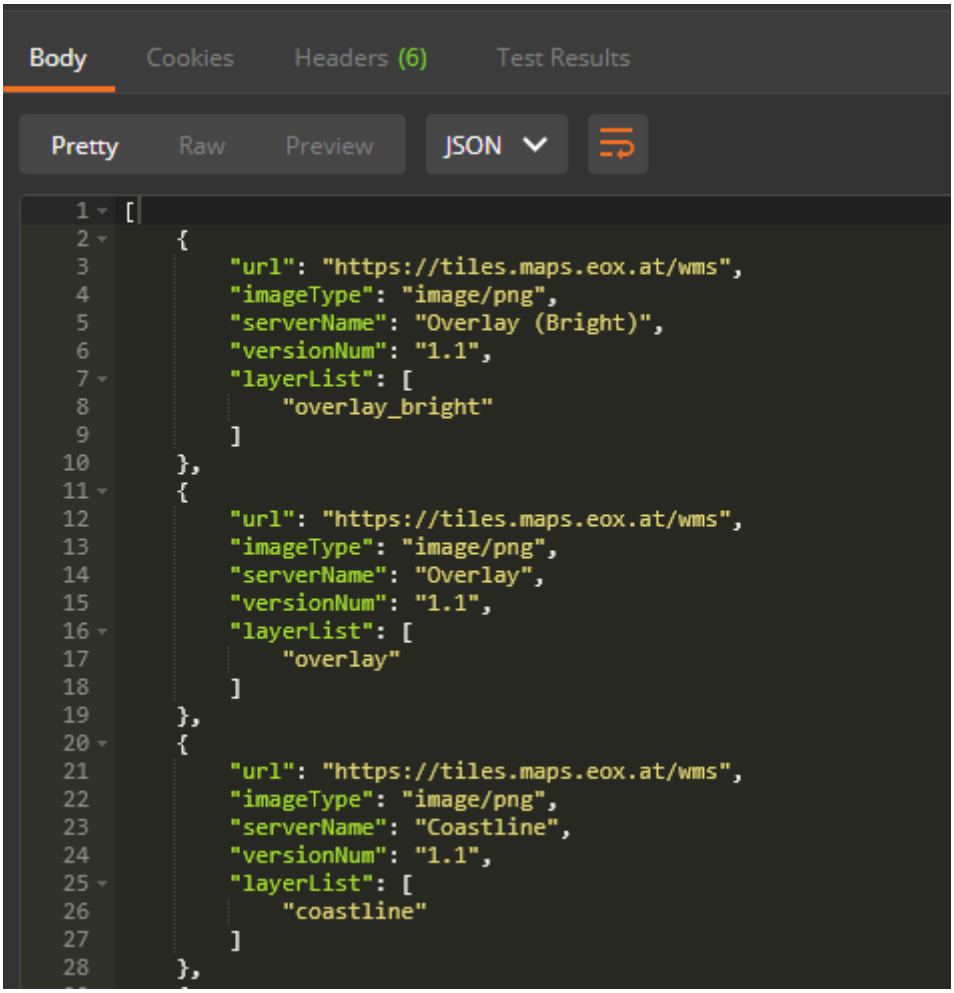
```
[
    {
        "url": "https://tiles.maps.eox.at/wms",
        "imageType": "image/png",
        "serverName": "Cloudless 2007",
        "versionNum": "1.1",
        "layerList": [
            "s2cloudless-2017"
        ]
    },
    {
        "url": "https://tiles.maps.eox.at/wms",
        "imageType": "image/png",
        "serverName": "Black Marble",
        "versionNum": "1.1",
        "layerList": [
            "blackmarble"
        ]
    },
    {
        "url": "https://tiles.maps.eox.at/wms",
        "imageType": "image/png",
        "serverName": "Terrain",
        "versionNum": "1.1",
        "layerList": [
            "terrain"
        ]
    },
```

Εικόνα 48 getUserBaseMapServers Webservice

- Επιθυμητά Διαφανή Χάρτη

```
@POST("v1/map/getUserMapOverlays")
Observable<List<MapServer>> getUserMapOverlays(@Header("Content-Type") String content_type,
                                               @Body WSMessageBody WSMessageBody);
```





Εικόνα 49 getUserMapOverlays Webservice

3.5.3 **Επιλογές Βοηθητικών Εργαλείων Χρήστη**

Η εφαρμογή παρέχει τις παρακάτω λειτουργίες, σε offline και online τρόπο λειτουργίας, για την υποβοήθηση του χρήστη:

- Default Zoom (Προκαθορισμένη Εστίαση)

```
public void goToDefaultZoom() {
    Config appConfig = presenter.getCurrentAppState().config();
    goToZoom(appConfig.getDefaultLatitude(), appConfig.getDefaultLongitude(), appConfig.getDefaultAltitude(), animate: false);
}
```

Εικόνα 50 Κώδικας Προκαθορισμένης Εστίασης

- Show All Symbols (Εστίαση σε όλα τα σύμβολα)

```
public void zoomToAllIcons() {
    mapView.zoomTo(getOverlay(MapUIConstants.OPERATIONAL_OVERLAY), animate: false);
}
```

Εικόνα 51 Κώδικας Εστίασης σε όλα τα σύμβολα

- Area Calculation (Υπολογισμός Περιοχής)
- Distance Calculation (Υπολογισμός Απόστασης)

```
private void showShapeAreaAndPerimeter() {
    if (getCurrentSelectedFeature() instanceof Polygon) {
        Polygon p = (Polygon) getCurrentSelectedFeature();
        showShapeAreaDialog(GeoUtils.calculatePolygonArea(p.getPositions()),
                GeoUtils.calculateDistanceOfPath(p.getPositions(), isPolygon: true),
                shape: "polygon");
    } else if (getCurrentSelectedFeature() instanceof Circle) {
        Circle c = (Circle) getCurrentSelectedFeature();
        showShapeAreaDialog(GeoUtils.calculateCircleAreaInKM(c.getRadius()),
                GeoUtils.calculateCirclePerimeterInKM(c.getRadius()), shape: "circle");
    } else if (getCurrentSelectedFeature() instanceof Path) {
        Path p = (Path) getCurrentSelectedFeature();
        showShapeAreaDialog( area: 0, GeoUtils.calculateDistanceOfPath(p.getPositions(),
                isPolygon: false), shape: "path");
    }
}
```

Εικόνα 52 Κώδικας Υπολογισμού Απόστασης

- Change Coordinate System (Αλλαγή Συστήματος Συντεταγμένων)

```
private void changeCoordSystemAction() {
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Change Map Coordinate System");

    String items[] = {"DMS", "UTM", "MGRS", "DD"};

    builder.setTitle("Select Coordinates system");
    builder.setItems(items, (dialog, item) -> {
        String coord = items[item];
        MapGridTypeEnum coordSystem;
        switch (coord) {
            case "DMS":
                coordSystem = MapGridTypeEnum.DMS;
                break;
            case "MGRS":
                coordSystem = MapGridTypeEnum.MGRS;
                break;
            case "DD":
                coordSystem = MapGridTypeEnum.DD;
                break;
            case "UTM":
            default:
                coordSystem = MapGridTypeEnum.UTM;
                break;
        }
        changeCoordSystem(coordSystem);
    });
    AlertDialog alert = builder.create();
    alert.show();
}
```

Εικόνα 53 Κώδικας Αλλαγής Συστήματος Συντεταγμένων

- Show Grid (Εμφάνιση Διαφανούς Γραμμών Συντεταγμένων)

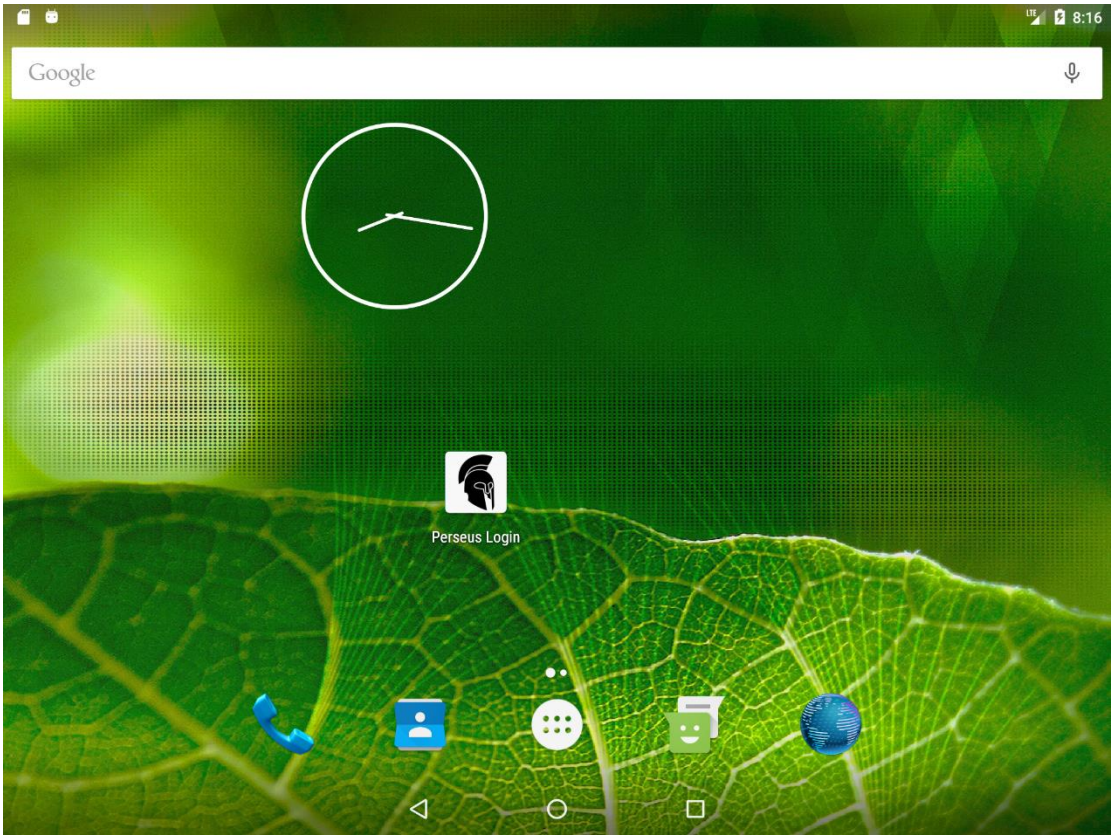
```
private void showMapGridAction() {
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Show Map Grid");

    String items[] = {"DMS", "UTM", "MGRS", "DD", "NONE"};

    builder.setTitle("Select Grid system");
    builder.setItems(items, (dialog, item) -> {
        String coord = items[item];
        MapGridTypeEnum coordSystem;
        switch (coord) {
            case "DMS":
                coordSystem = MapGridTypeEnum.DMS;
                break;
            case "MGRS":
                coordSystem = MapGridTypeEnum.MGRS;
                break;
            case "DD":
                coordSystem = MapGridTypeEnum.DD;
                break;
            case "NONE":
                coordSystem = MapGridTypeEnum.NONE;
                break;
            case "UTM":
            default:
                coordSystem = MapGridTypeEnum.UTM;
                break;
        }
        showGridSystem(coordSystem);
    });
    AlertDialog alert = builder.create();
    alert.show();
}
```

4. Παρουσίαση του Συστήματος

4.1 Εισαγωγή

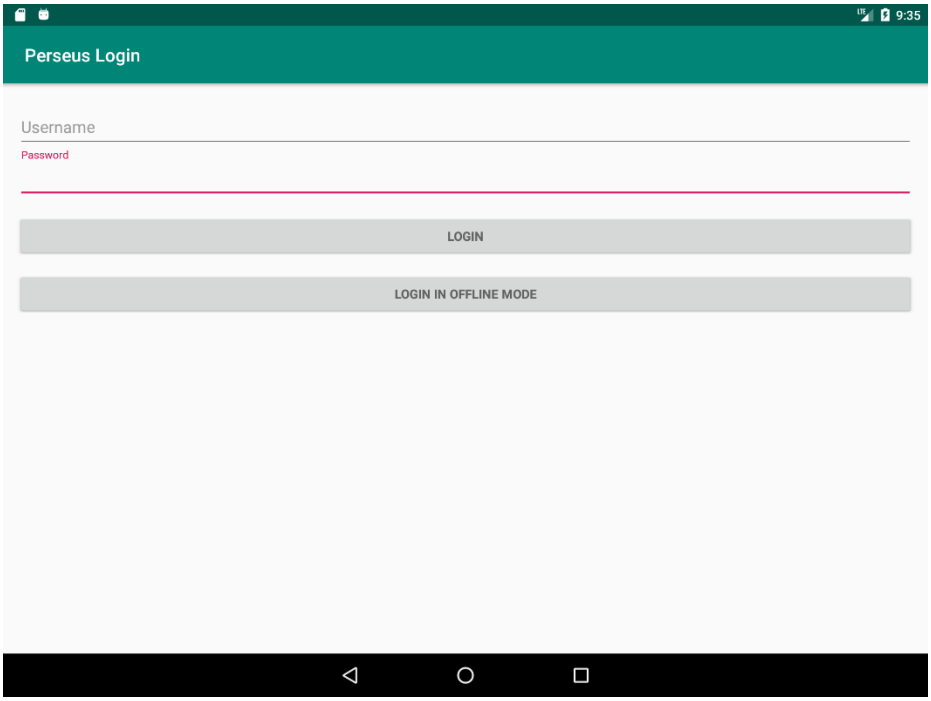
4.2 Είσοδος στην Εφαρμογή (Online)

Η είσοδος στην εφαρμογή γίνεται με πάτημα του κομβίου της εφαρμογής «Perseus Login» από το λειτουργικό Android.



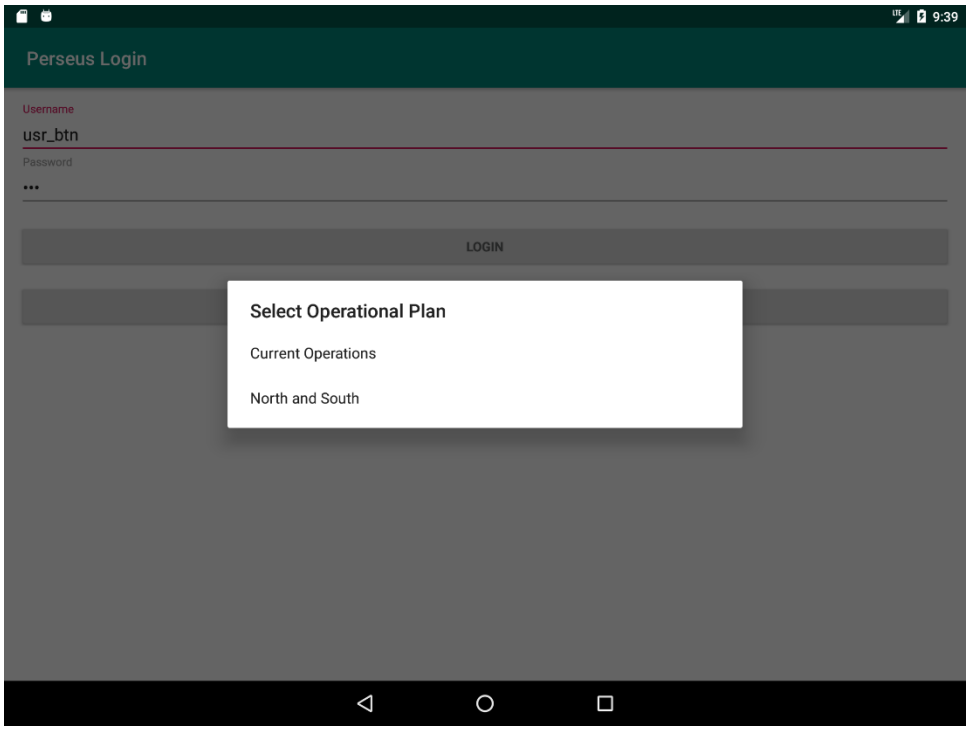
Εικόνα 54 Επιφάνεια Android

Μετά την φόρτωση της εφαρμογής, εμφανίζεται η οθόνη Login, στην οποία ο χρήστης μπορεί να εισάγει το username και password του, και να κάνει Login.



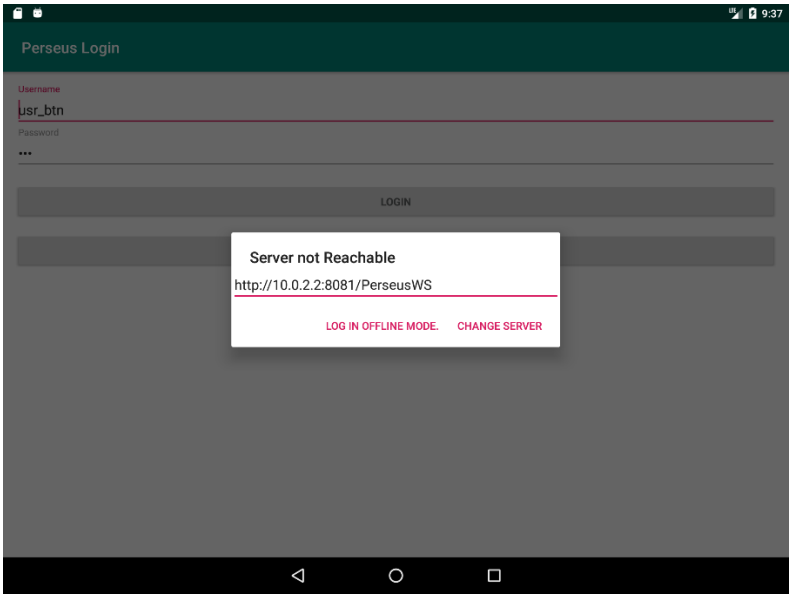
Εικόνα 55 Οθόνη Login

Αν η αυθεντικοποίηση είναι επιτυχής, ο χρήστης θα κληθεί να διαλέξει ποιο σχέδιο επιχειρήσεων επιθμεί να χρησιμοποιεί η εφαρμογή.



Εικόνα 56 Οθόνη Σχεδίων

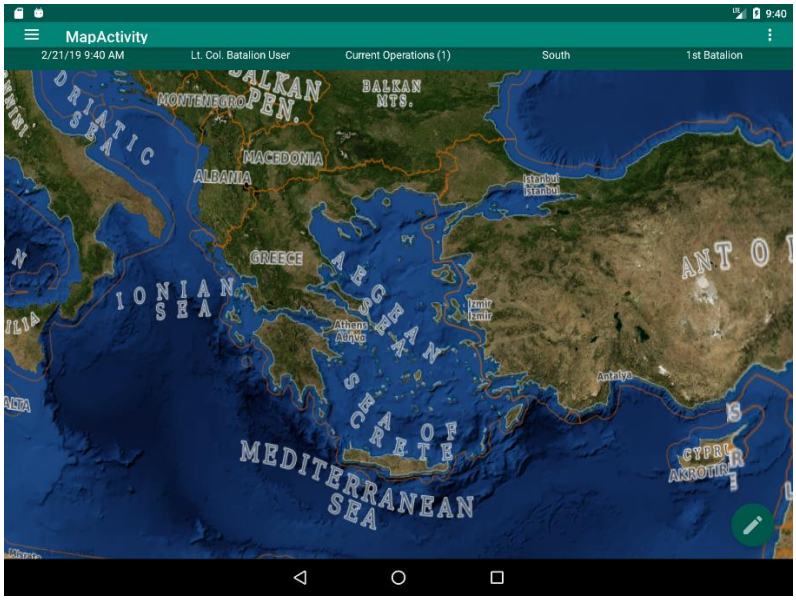
Αν ο Εξυπηρετητής της εφαρμογής δεν είναι διαθέσιμος, δίνεται η δυνατότητα στον χρήστη να αλλάξει χειροκίνητα εξυπηρετητή ή να εισέλθει στην εφαρμογή σε κατάσταση λειτουργίας Offline.



Εικόνα 57 Οθόνη Αλλαγής Εξυπηρετητή

4.3 Κατάσταση Λειτουργίας Online

Μετά την είσοδο στην εφαρμογή, εμφανίζεται η κεντρική οθόνη, που περιλαμβάνει τις λειτουργίες και τον χάρτη.

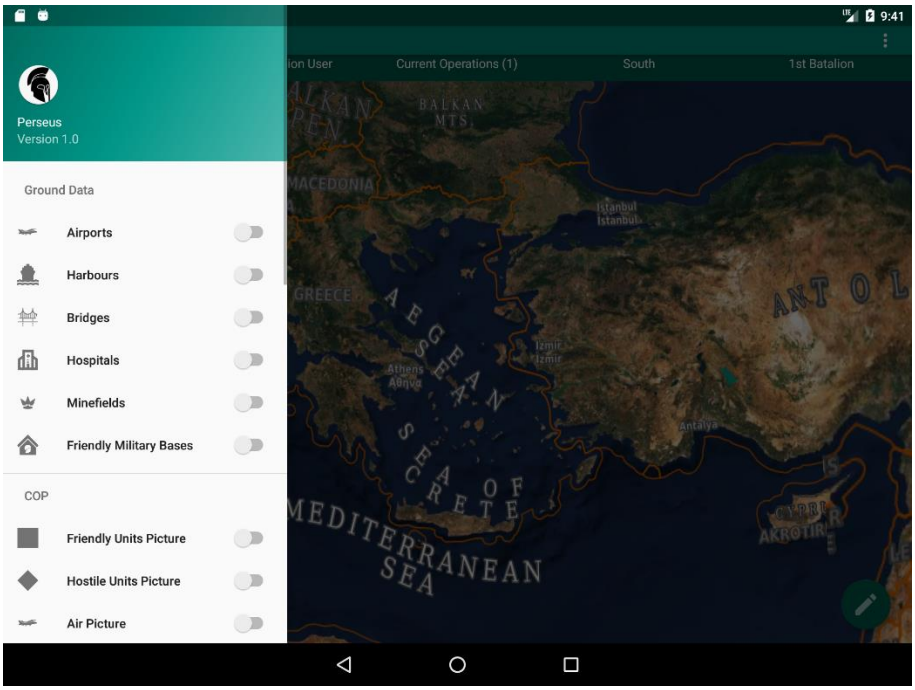


Εικόνα 58 Οθόνη Χάρτη

Ο χρήστης μπορεί να ελέγξει τον χάρτη με τα δάχτυλά του, με την διεπαφή χρήστη να είναι παρόμοια με αυτή που χρησιμοποιούν και άλλες εφαρμογές χαρτών σε κινητά. Με σύρσιμο ο χάρτης μετακινείται προς εκείνη την κατεύθυνση, με τα δυο δάχτυλα γίνεται εστίαση προς και από το κέντρο της κίνησης, ενώ με τρία δάκτυλα αλλάζει η γωνία υπό την οποία βλέπουμε τον χάρτη. Τέλος, με απλό πάτημα πάνω στον χάρτη βλέπουμε συντεταγμένες του σημείου, και αν υπάρχει στοιχείο, πληροφορίες για το στοιχείο.

4.3.1 **Λειτουργίες Δεδομένων**

Με πάτημα στις τρείς οριζόντιες γραμμές πάνω αριστερά, ο χρήστης έχει πρόσβαση στις λειτουργίες δεδομένων της εφαρμογής, αν έχει τα δικαιώματα για αυτές και βρίσκεται στον κατάλληλο τρόπο λειτουργίας.

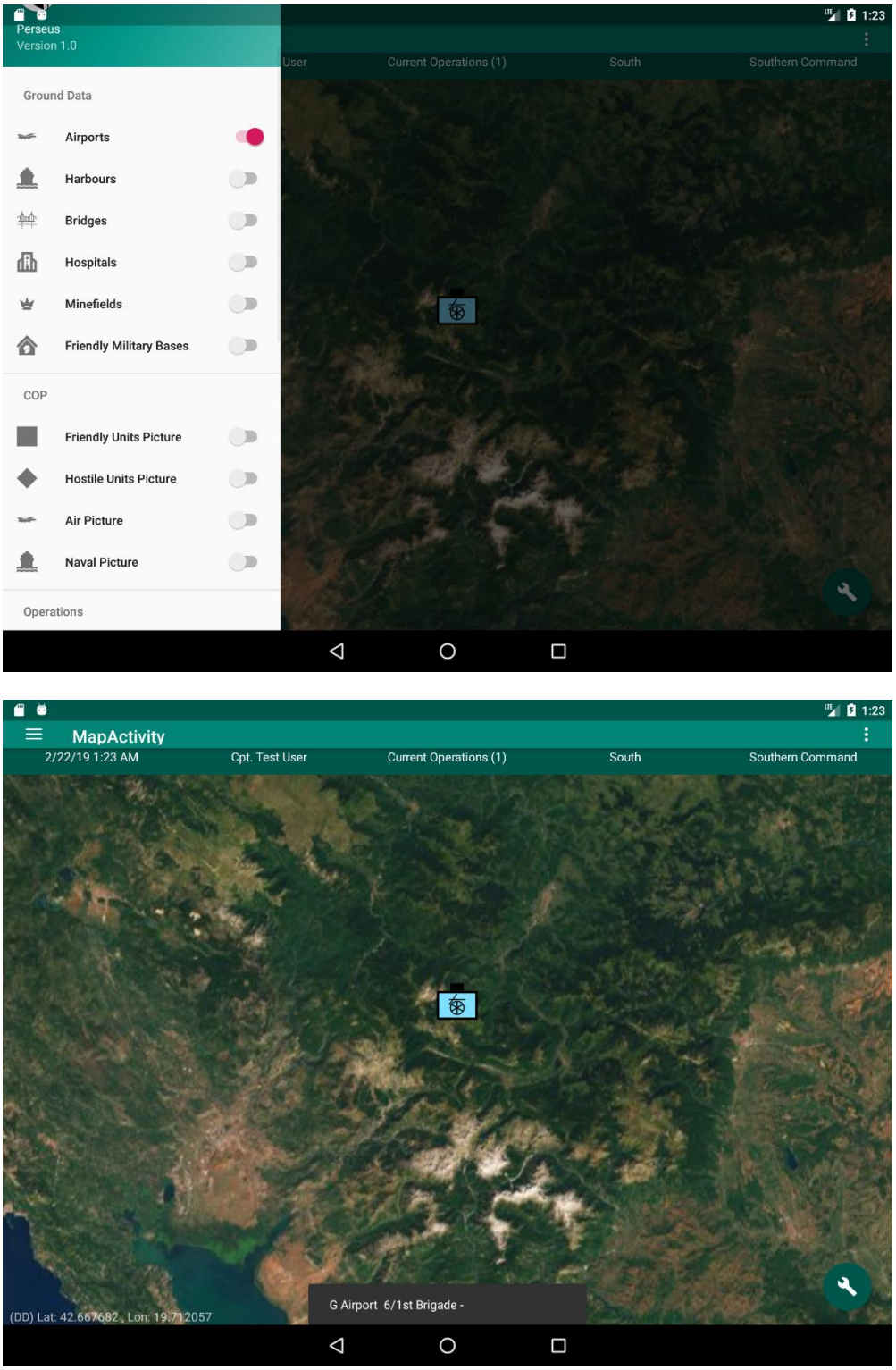


Εικόνα 59 Λειτουργίες Δεδομένων

4.3.1.1 Λειτουργίες Στατικών Δεδομένων

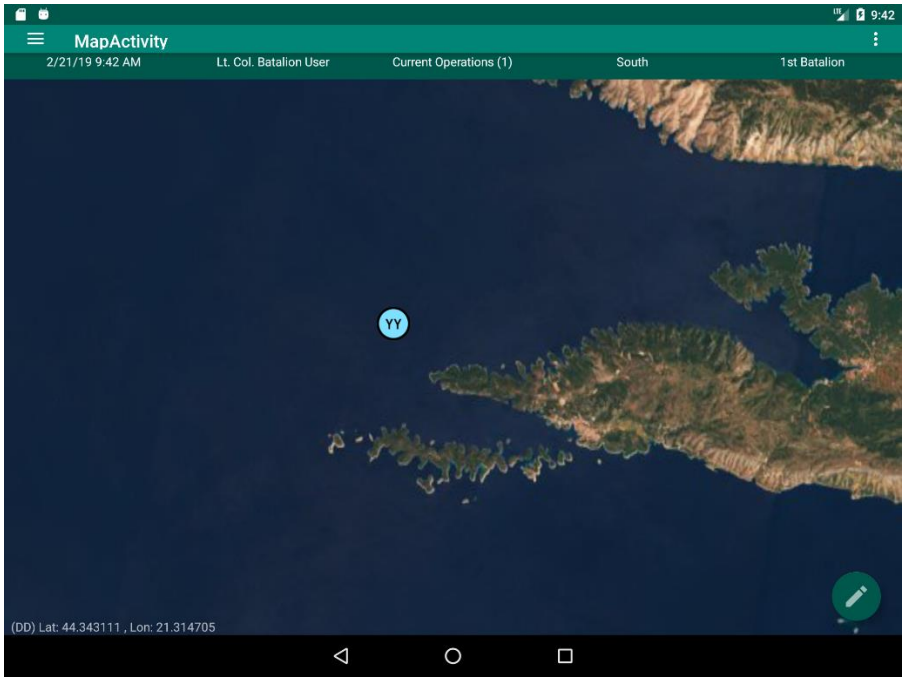
Ο χρήστης μπορεί να επιλέξει να λάβει δεδομένα για τις ακόλουθες κατηγορίες στατικών δεδομένων:

- Airports



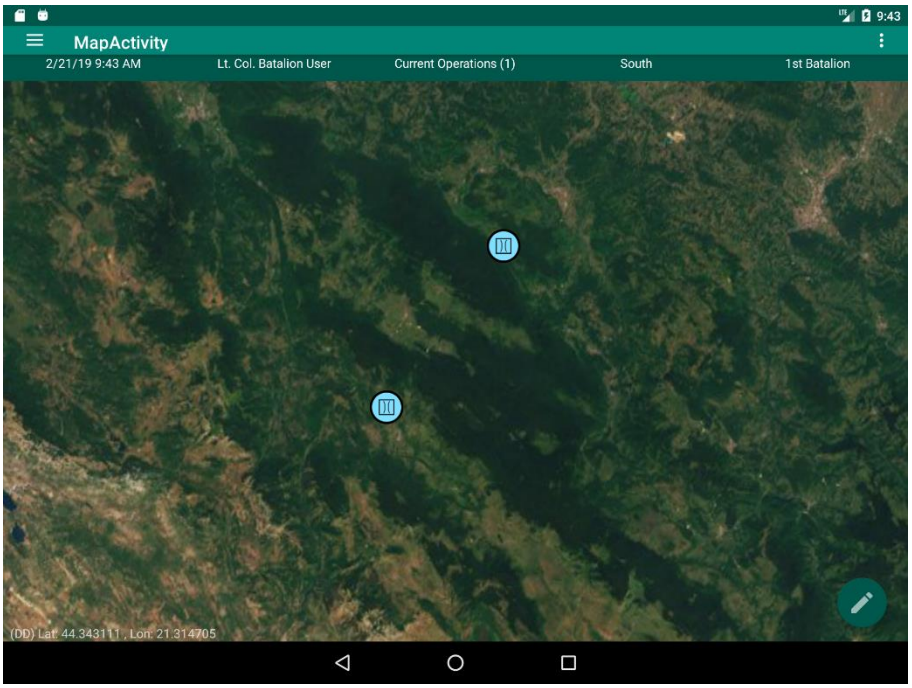
Εικόνα 60 Αεροδρόμια

- Harbors



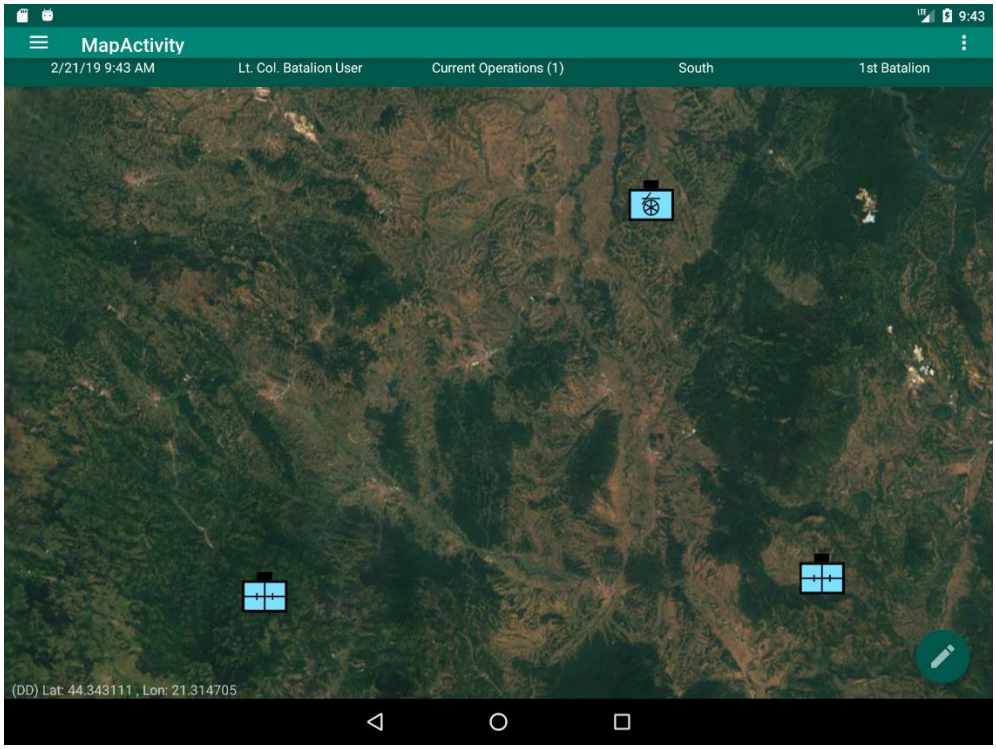
Εικόνα 61 Λιμάνια

- Bridges



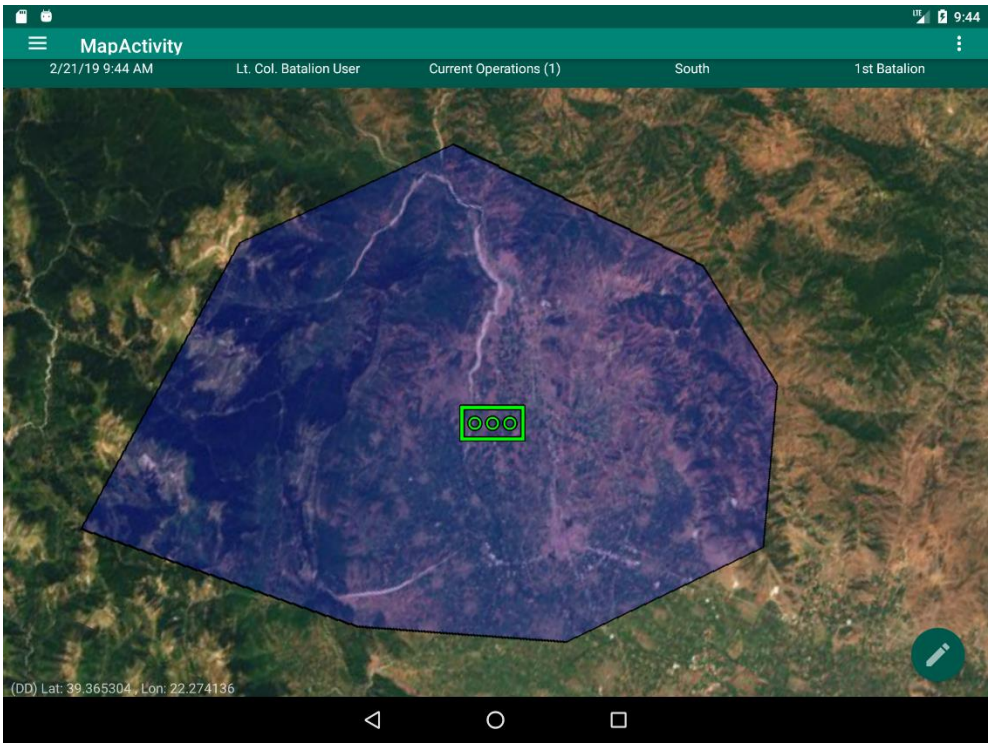
Εικόνα 62 Γέφυρες

- Hospital



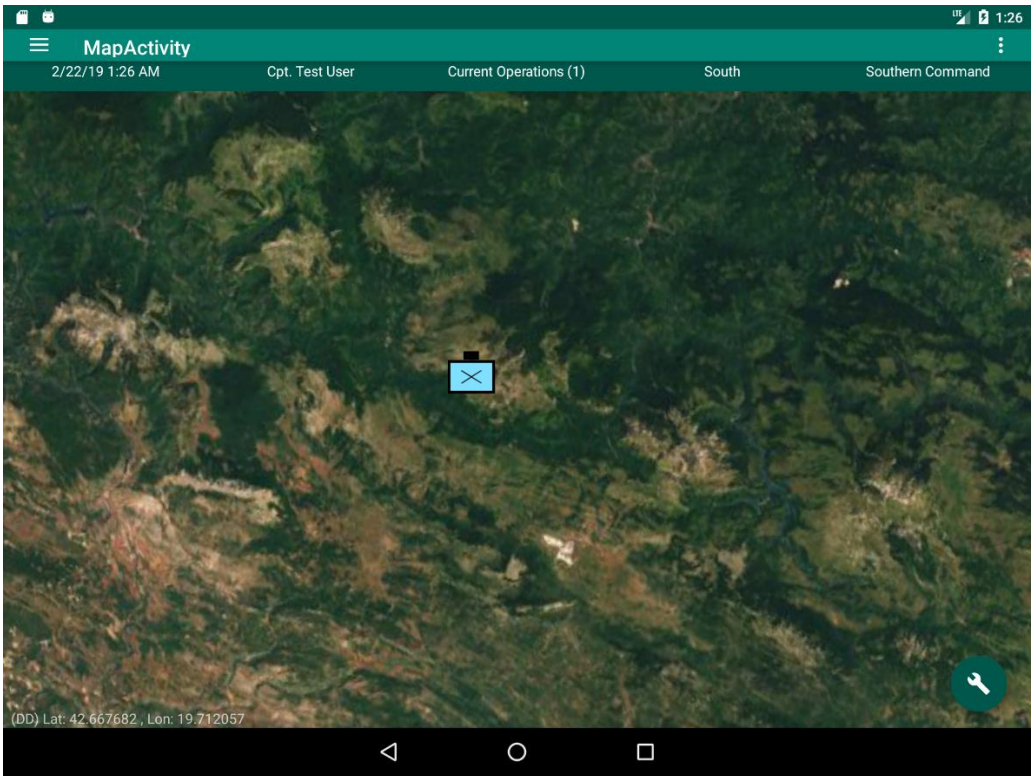
Εικόνα 63 Νοσοκομεία

- Minefields



Εικόνα 64 Ναρκοπέδια

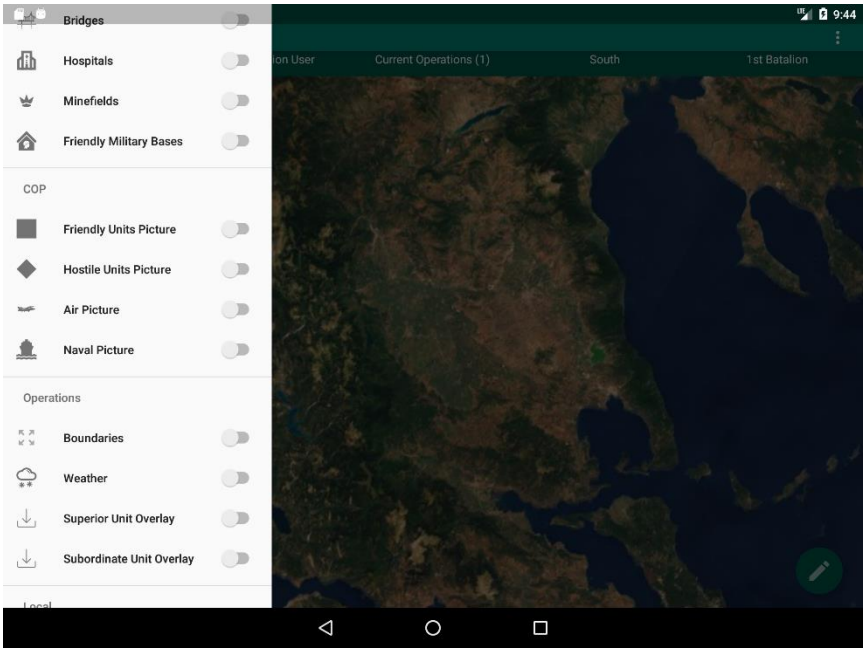
- Friendly Military Bases



Εικόνα 65 Στρατιωτικές Βάσεις

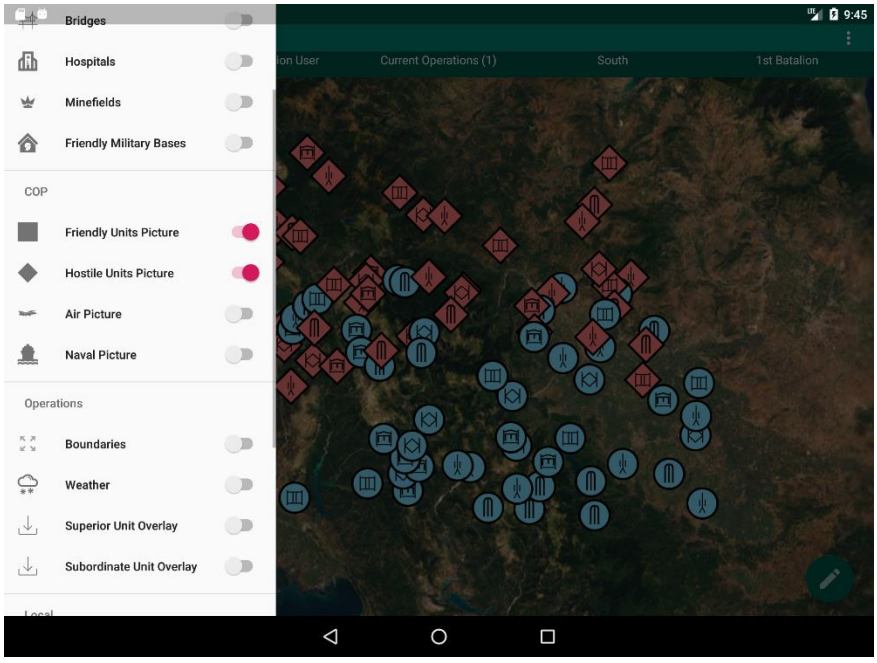
4.3.2 **Λειτουργίες Δυναμικών Δεδομένων**

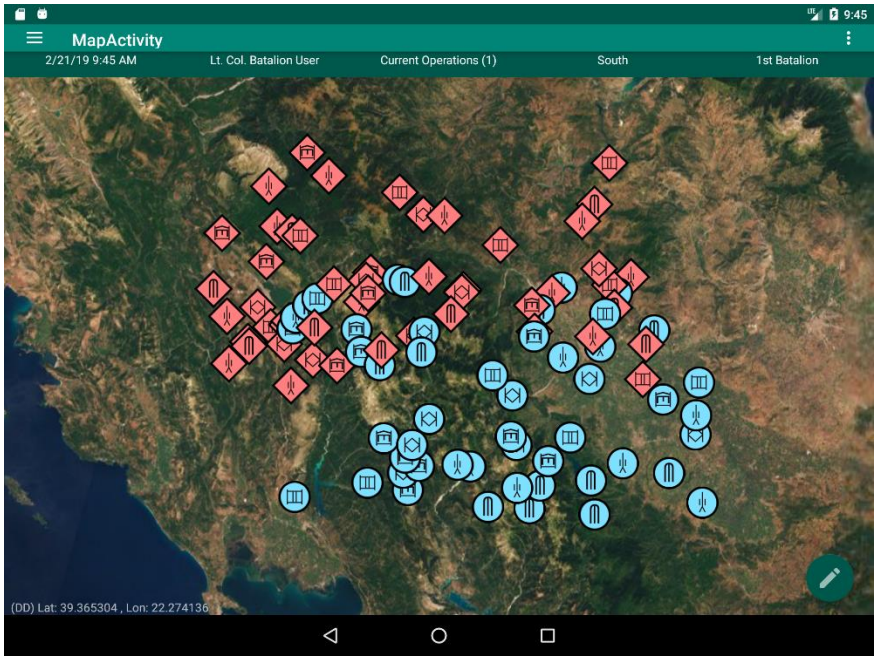
Ο Χρήστης έχει την δυνατότητα να επιλέξει για απεικόνιση δυναμικά δεδομένα, τα οποία μεταβάλλονται διαρκώς:



Εικόνα 66 Λειτουργίες Δυναμικών Δεδομένων

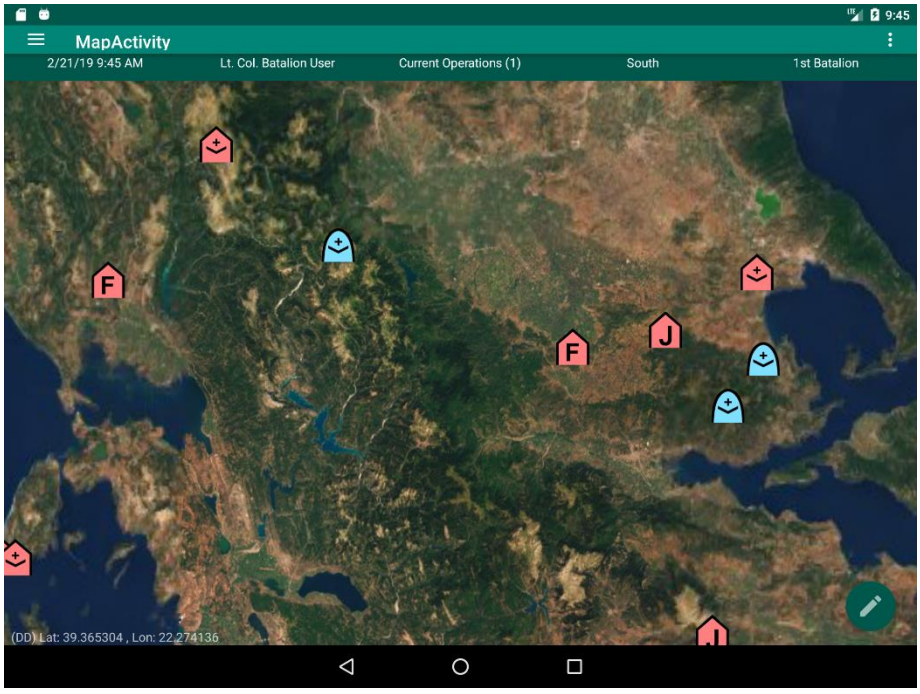
- Friendly Unit Picture
- Hostile Unit Picture





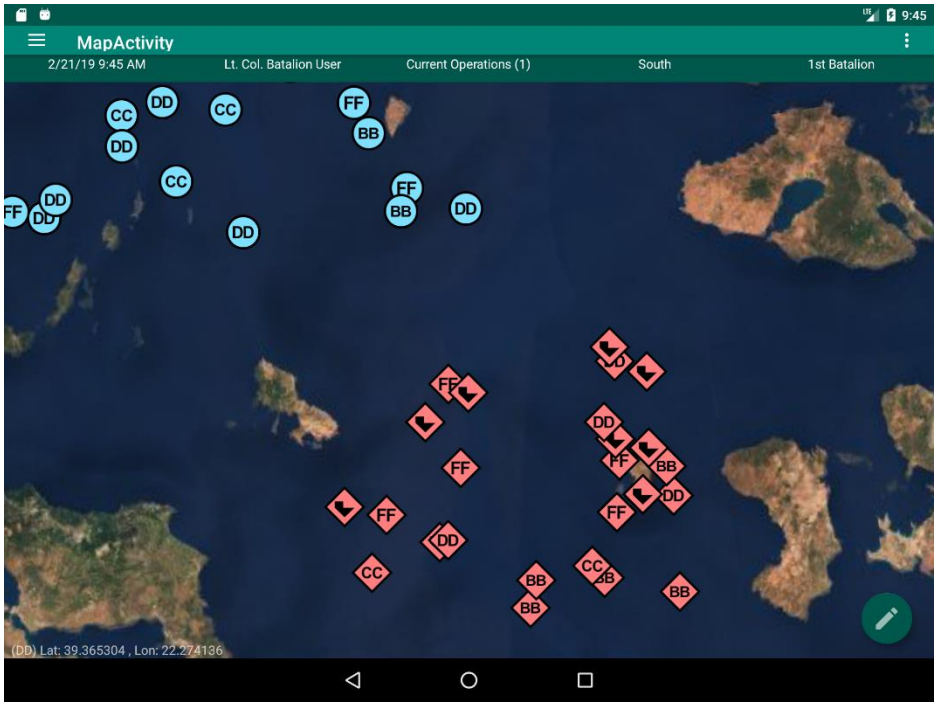
Εικόνα 67 Εικόνα Επίγειων Δυνάμεων

- Air Picture



Εικόνα 68 Αεροπορική Εικόνα

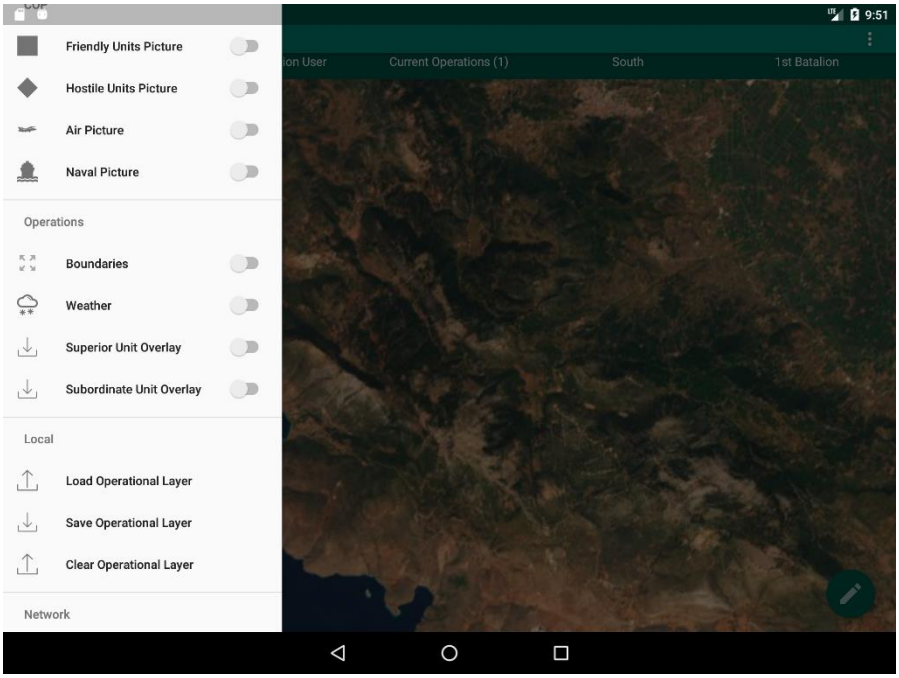
- Naval Picture



Εικόνα 69 Ναυτική Εικόνα

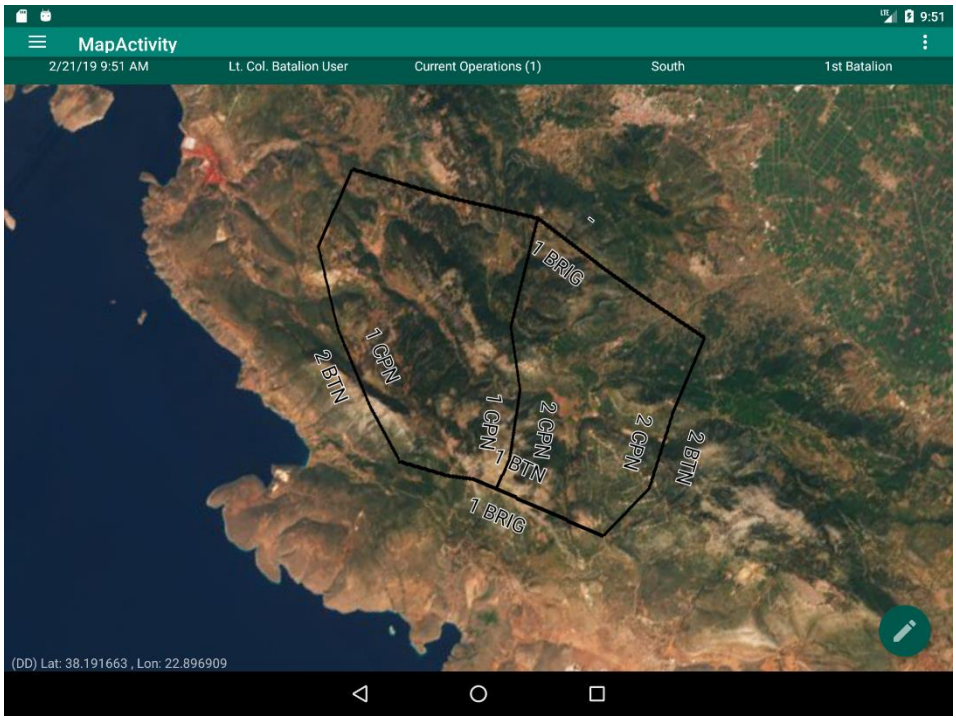
4.3.3 **Λειτουργίες Επιχειρησιακών Δεδομένων**

Ο χρήστης έχει την δυνατότητα να λάβει δεδομένα άμεσου επιχειρησιακού ενδιαφέροντος:



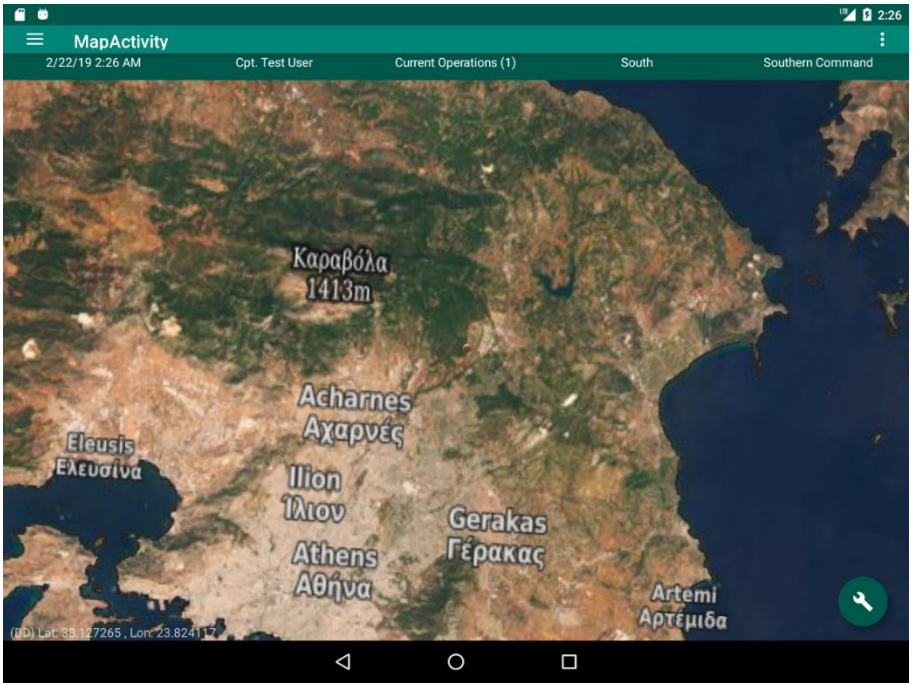
Εικόνα 70 Λειτουργίες Επιχειρησιακού Ενδιαφέροντος

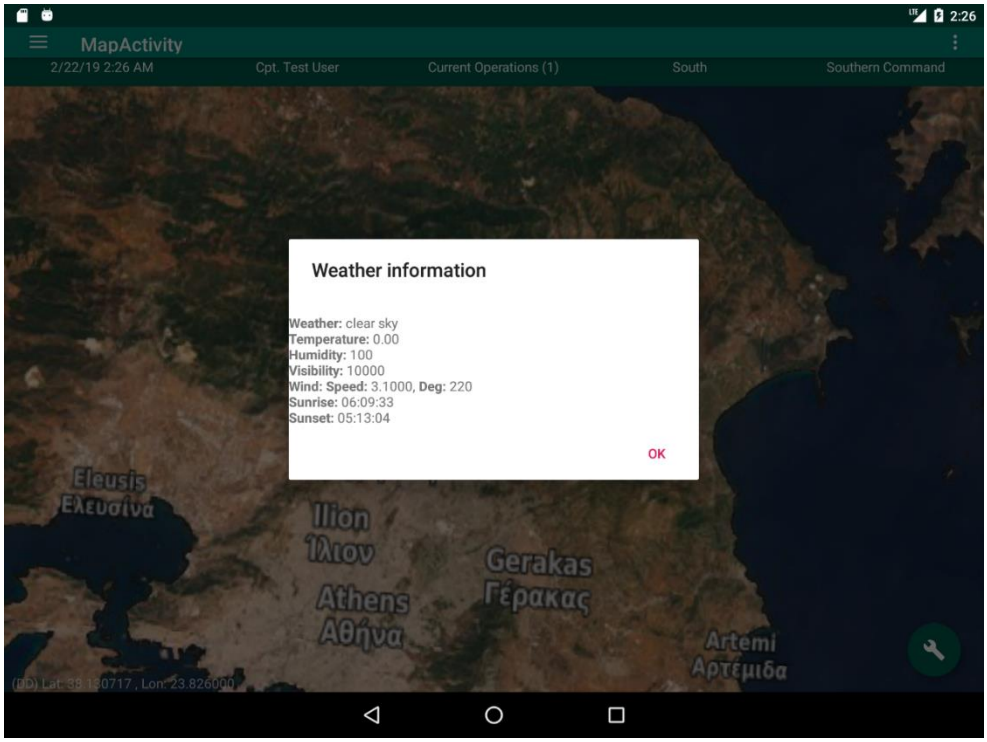
- Boundaries



Εικόνα 71 Όρια

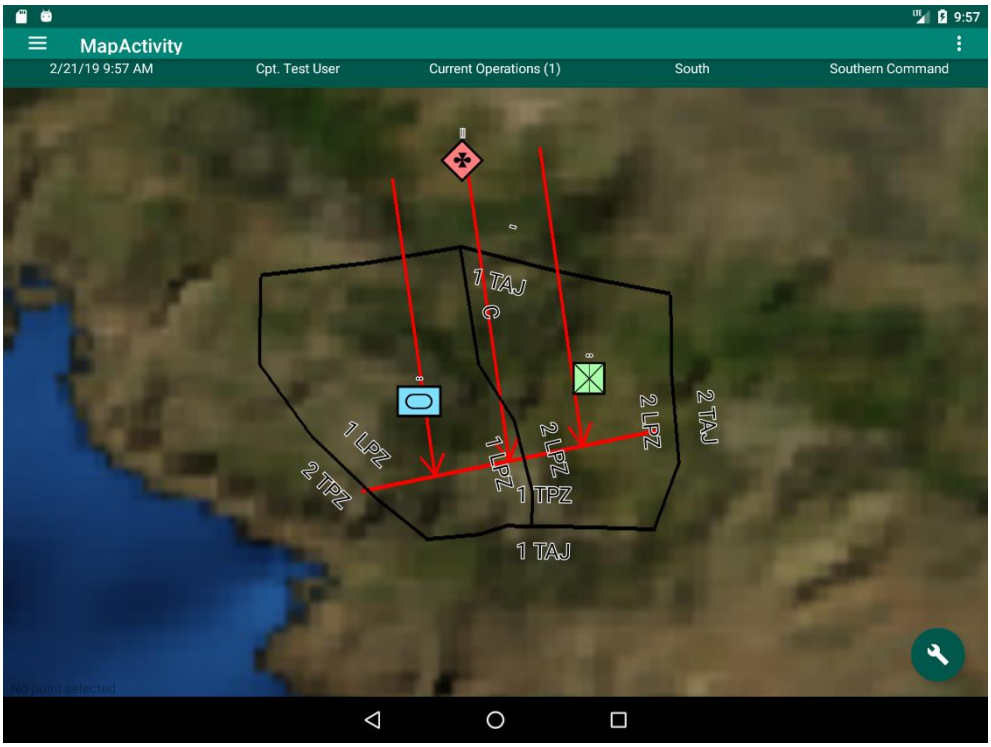
- Weather





Εικόνα 72 Καιρός

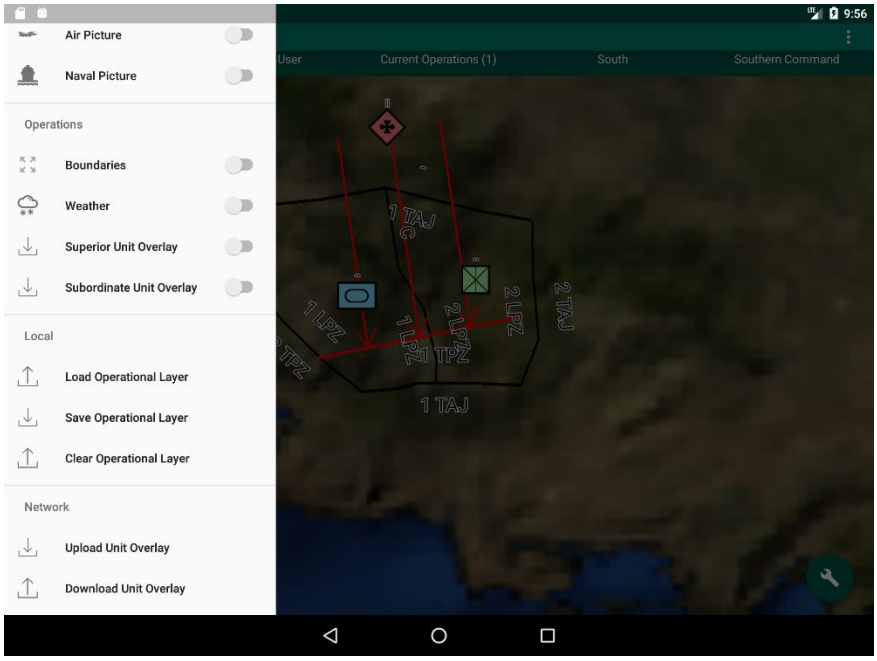
- Superiod Unit Overlay
- Subordinate Unit Overlay



Εικόνα 73 Διαφανή Ετέρων Σχηματισμών

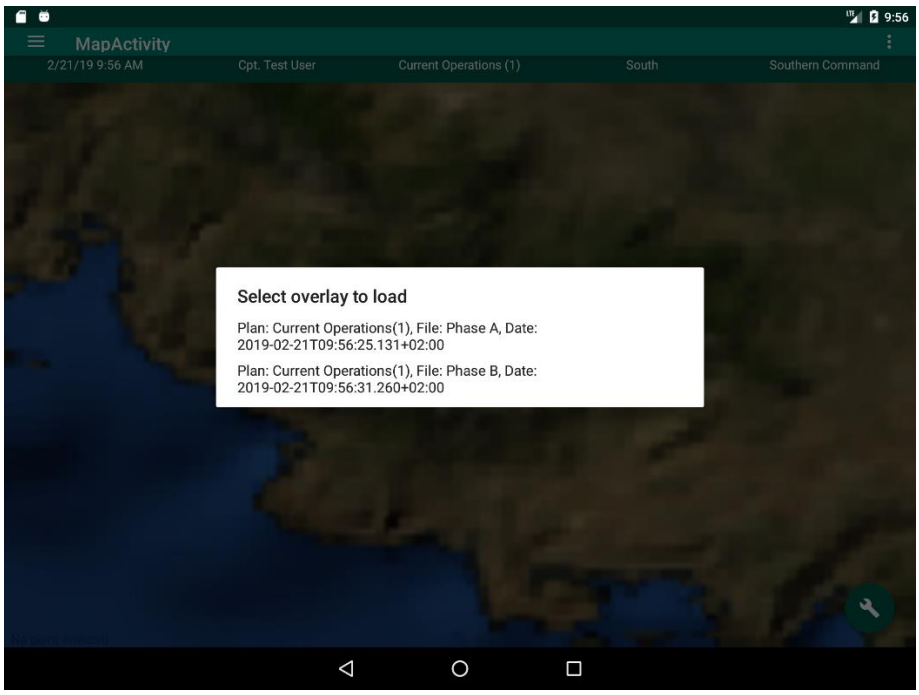
4.3.4 **Λειτουργίες Τοπικής Αποθήκευσης και Ανάκτησης Διαφανών Επιχειρήσεων**

Ο Χρήστης έχει την δυνατότητα τοπικά να αποθηκεύσει, ανακτήσει και εκκαθαρίσει το διαφανές επιχειρήσεων:



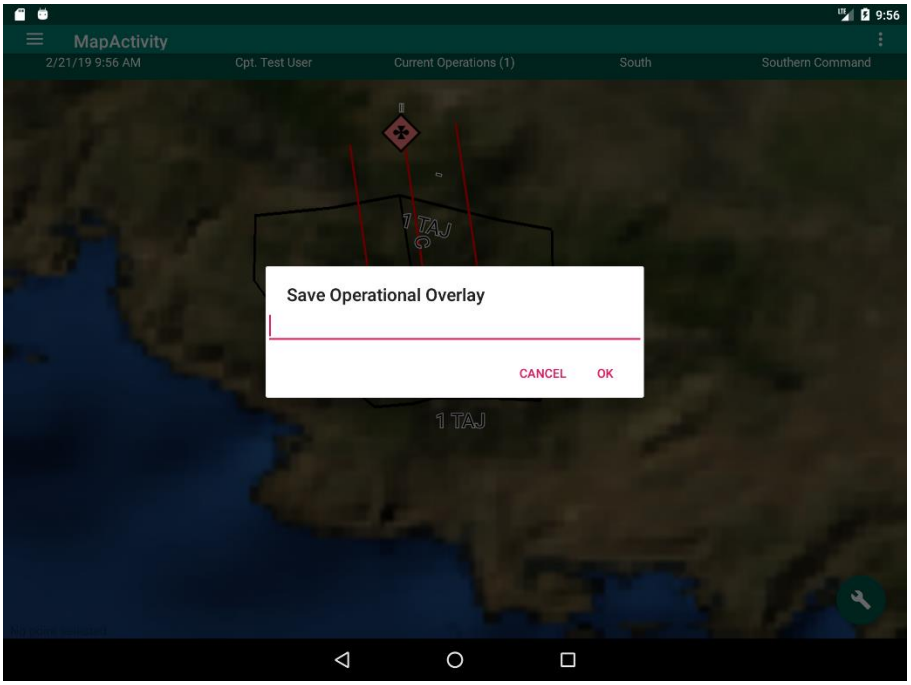
Εικόνα 74 Λειτουργίες Τοπικής Αποθήκευσης

- Load Operational Overlay



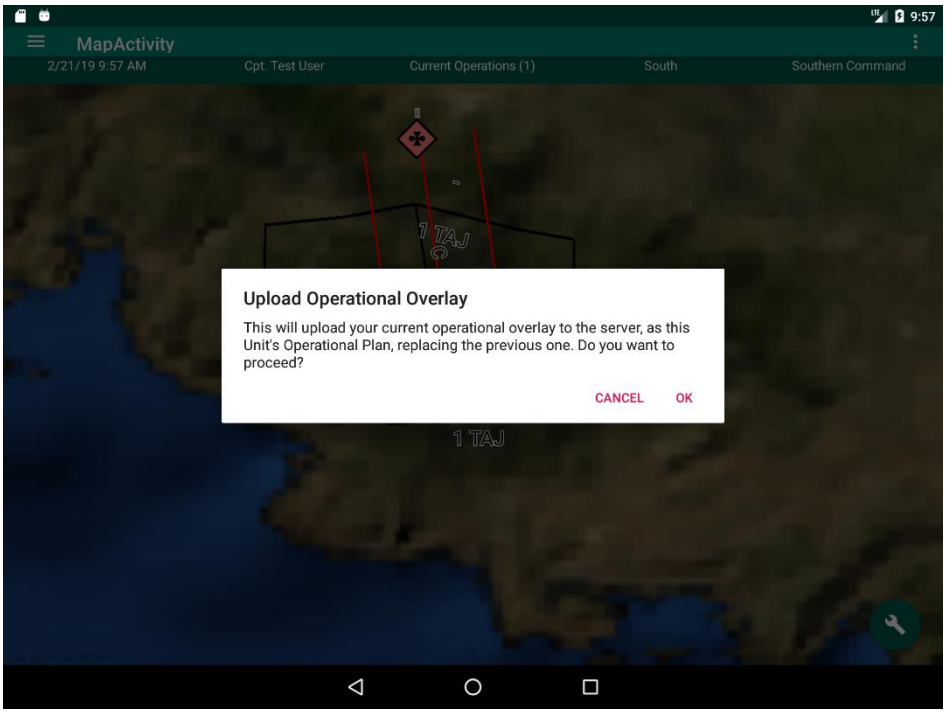
Εικόνα 75 Φόρωση Τοπικού Διαφανούς

- Save Operational Overlay



Εικόνα 76 Αποθήκευση Τοπικού Διαφανούς

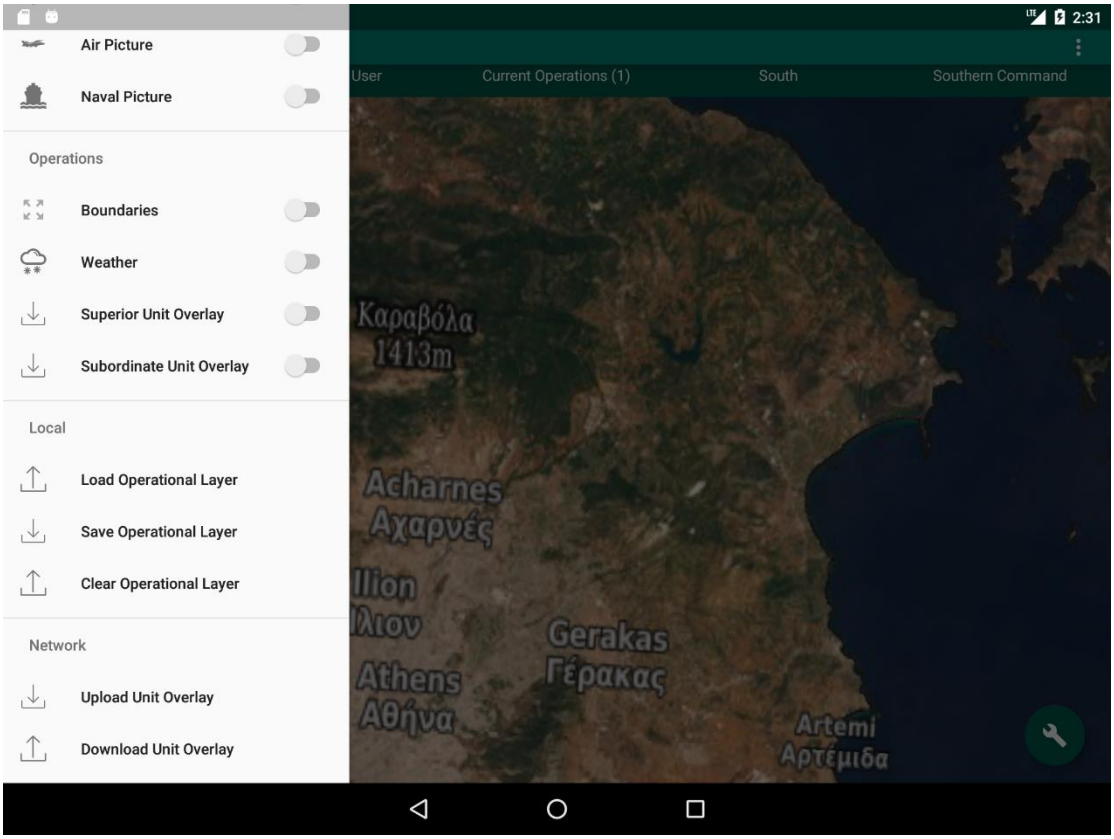
- Clear Operational Overlay



Εικόνα 77 Εκκαθάριση Διαφανούς

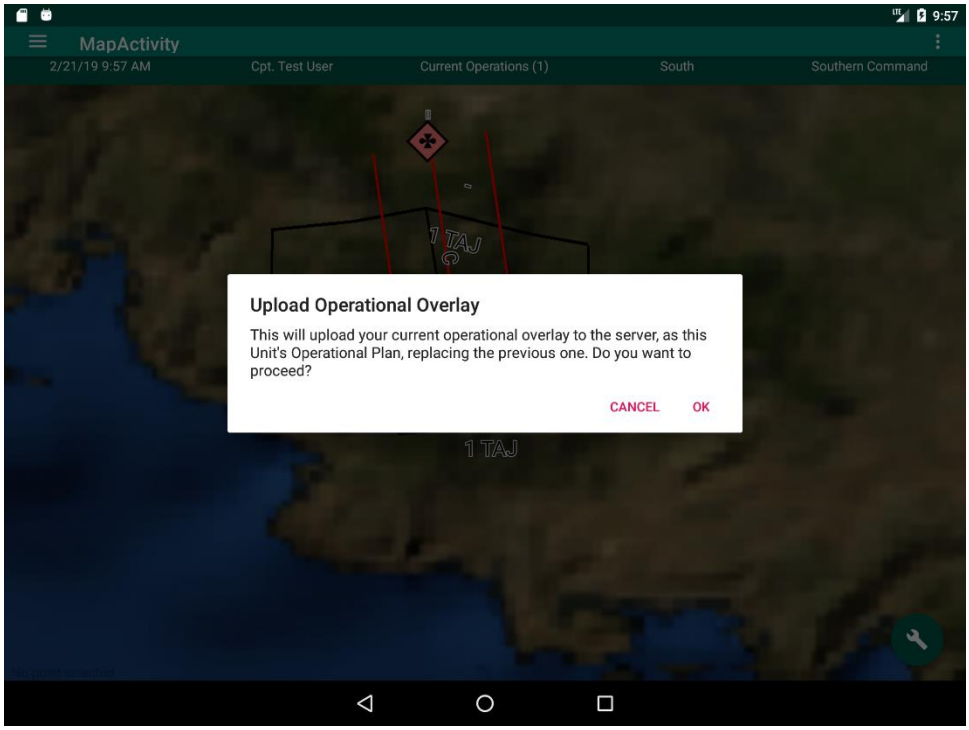
4.3.5 **Λειτουργείες Ορισμού και Λήψης Διαφανούς Επιχειρήσεων Μονάδας**

Η εφαρμογή δίνει την δυνατότητα στον χρήστη, αν έχει τα κατάλληλα δικαιώματα, να αποθηκεύσει και να ορίσει το διαφανές του ως το διαφανές επιχειρήσεων της Μονάδας και την δυνατότητα να λάβει το διαφανές της Μονάδας ως τρέχων διαφανές:



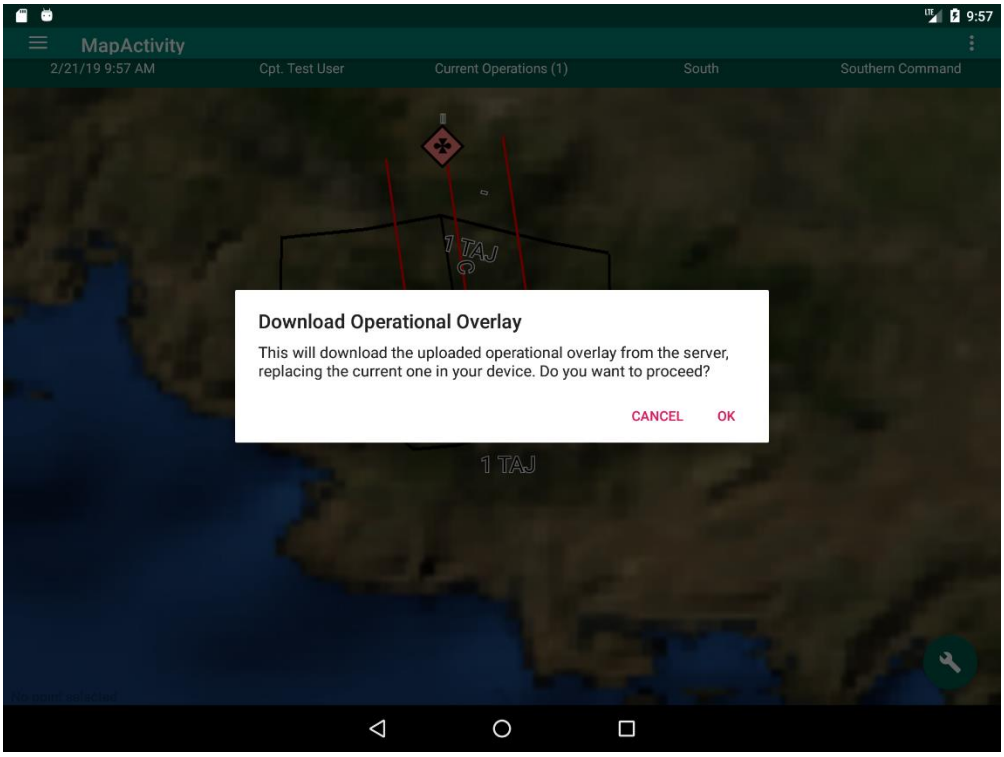
Εικόνα 78 Λειτουργίες Διαφανούς Επιχειρήσεων Μονάδας

- Upload Unit Overlay



Εικόνα 79 Ορισμός Διαφανούς Μονάδας

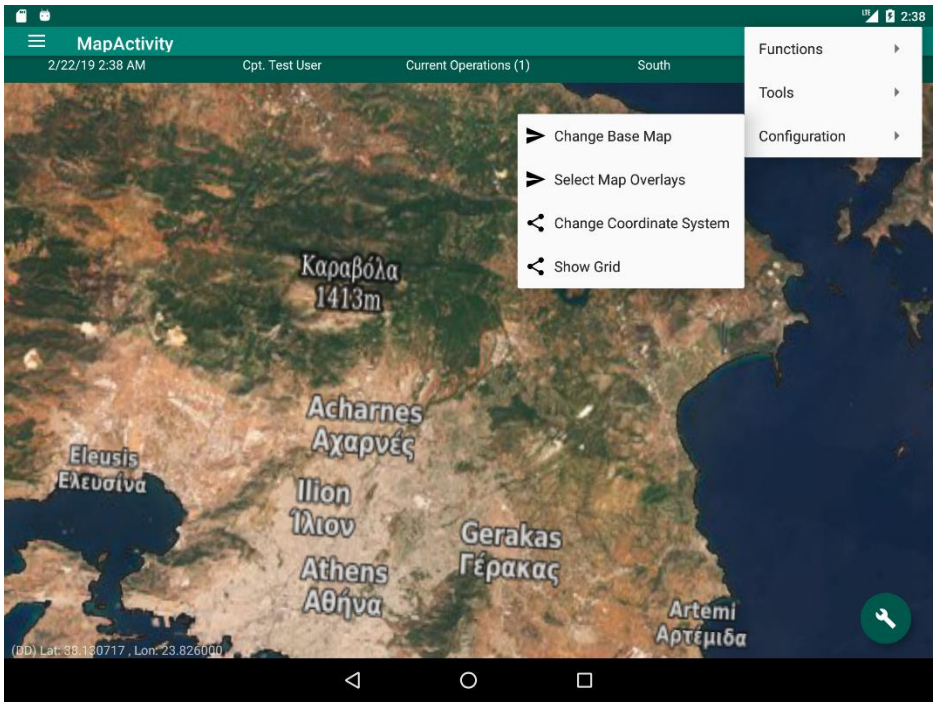
- Download Unit Overlay



Εικόνα 80 Λήψη Διαφανούς Μονάδας

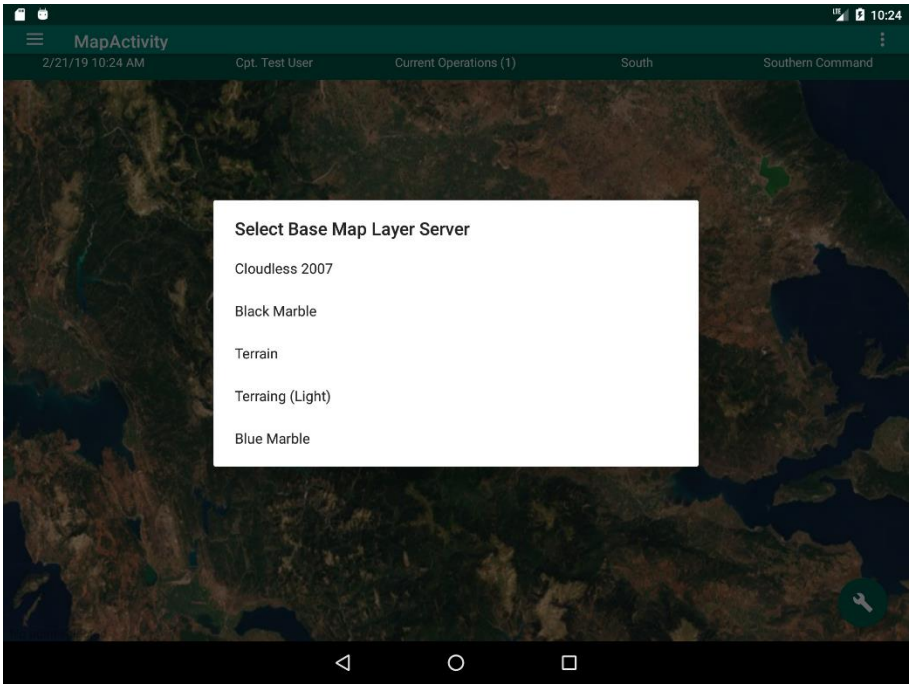
4.3.6 **Επιλογές Χαρτών Υποβάθρου και Χαρτών Διαφανών**

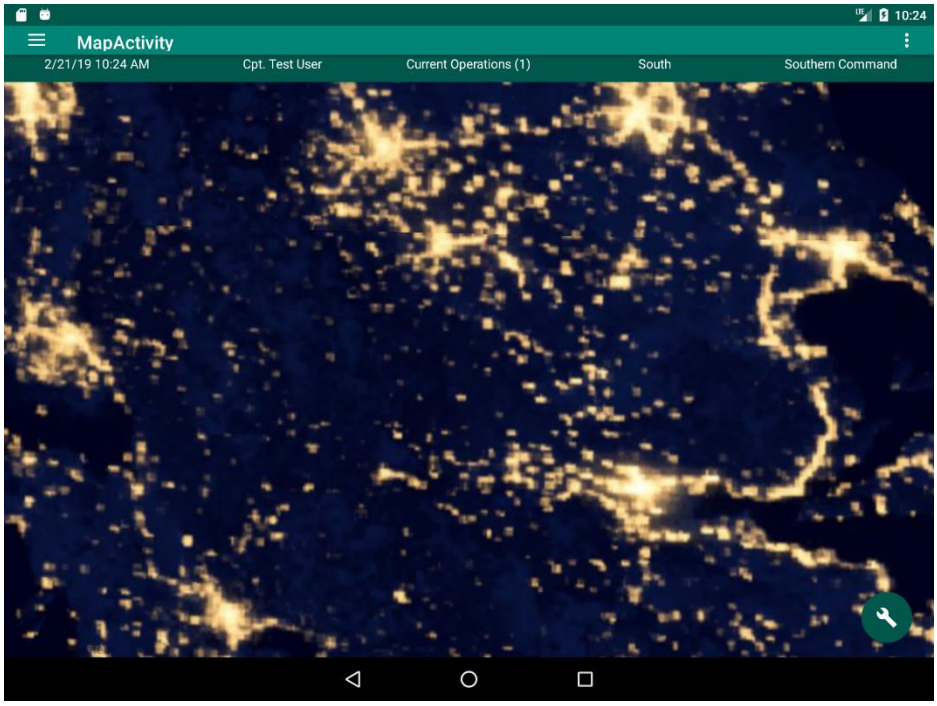
Η Εφαρμογή δίνει την δυνατότητα στον Χρήστη αλλαγής χάρτη υποβάθρου και διαφανών όπως παρακάτω:



Εικόνα 81 Επιλογές Χαρτών

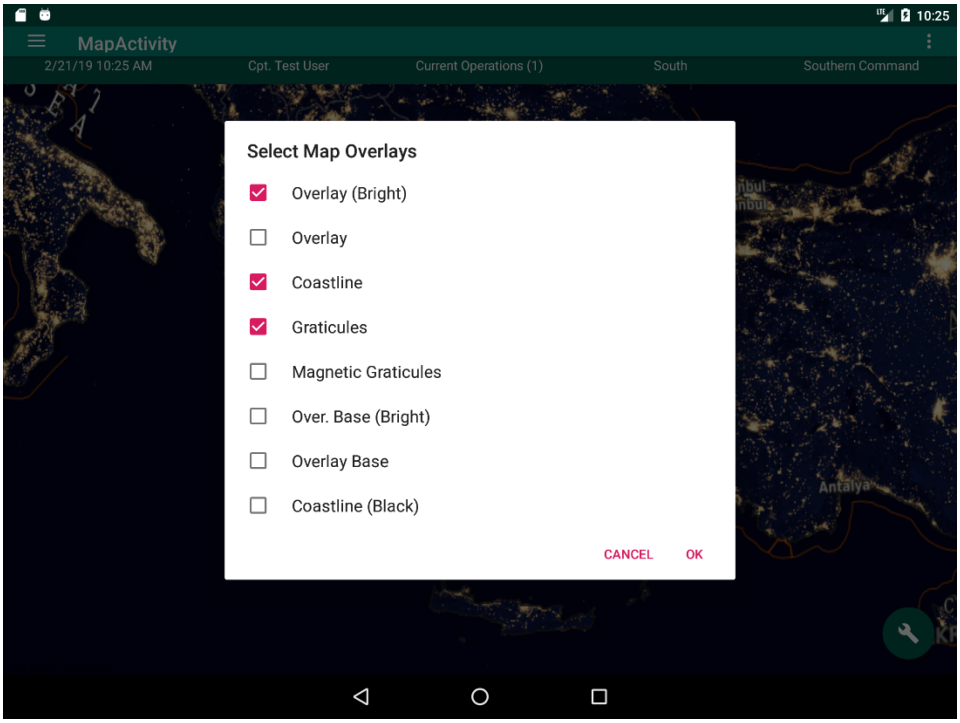
- Change Base Map

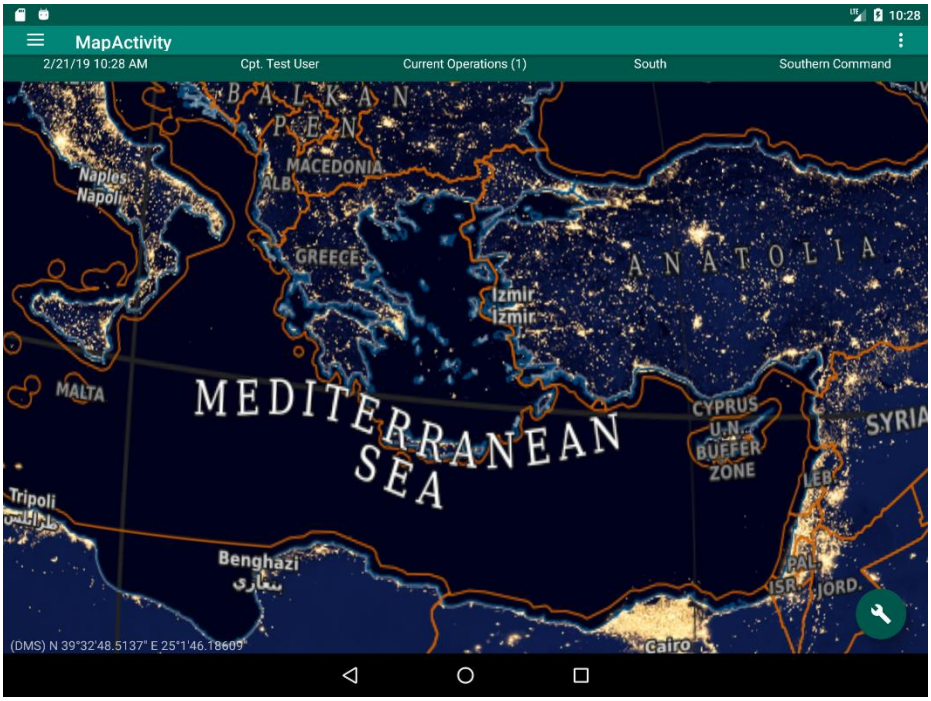




Εικόνα 82 Αλλαγή Χάρτη Υποβάθρου

- Select Map Overlays

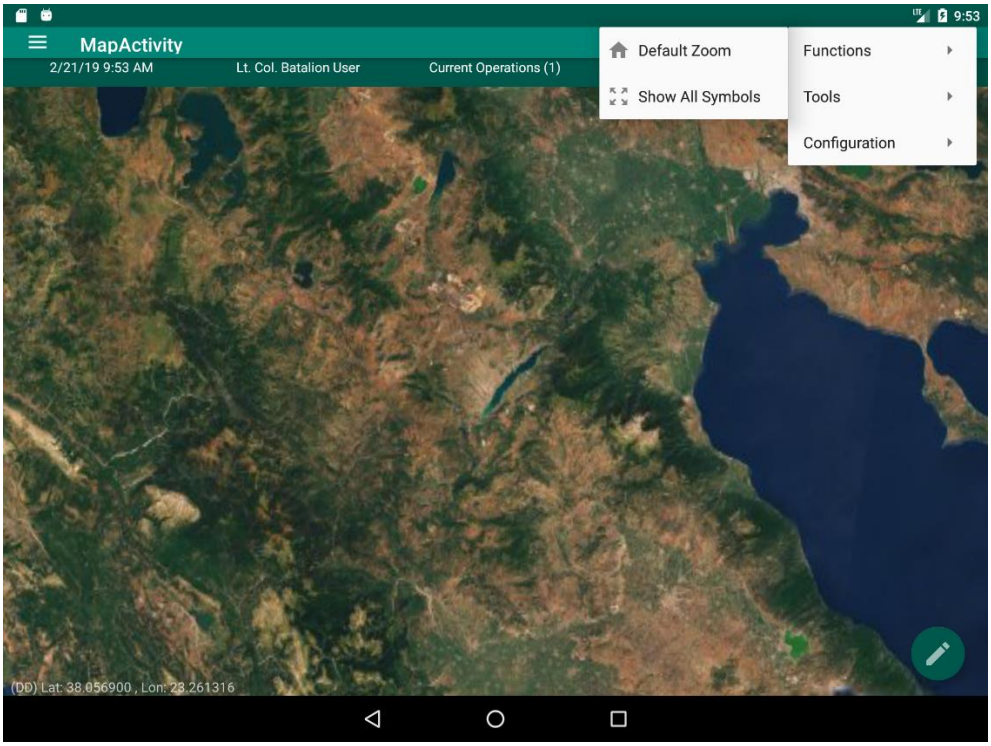


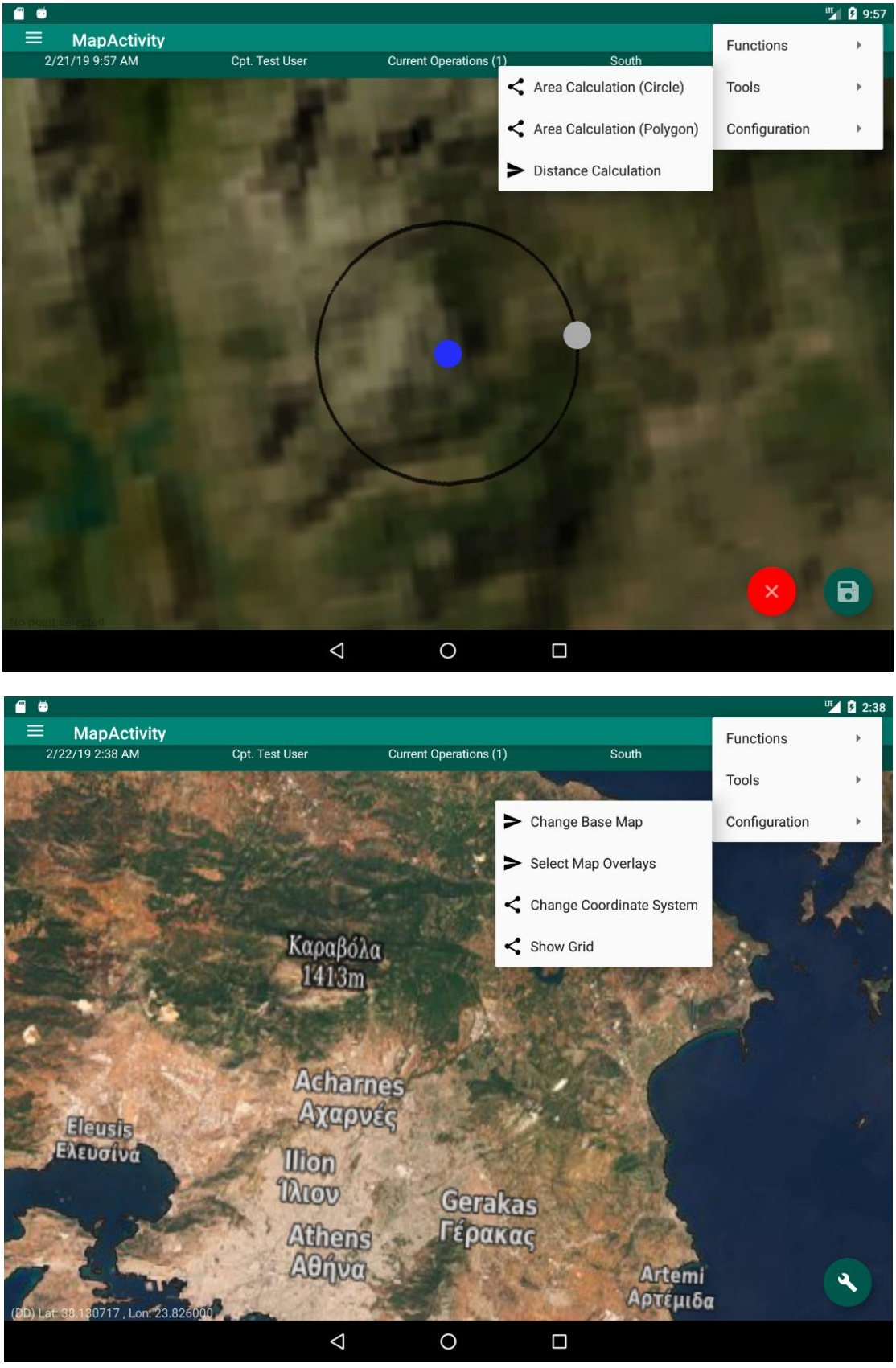


Εικόνα 83 Επιλογή Διαφανών Χαρτών

4.3.7 **Βοηθητικά Εργαλεία**

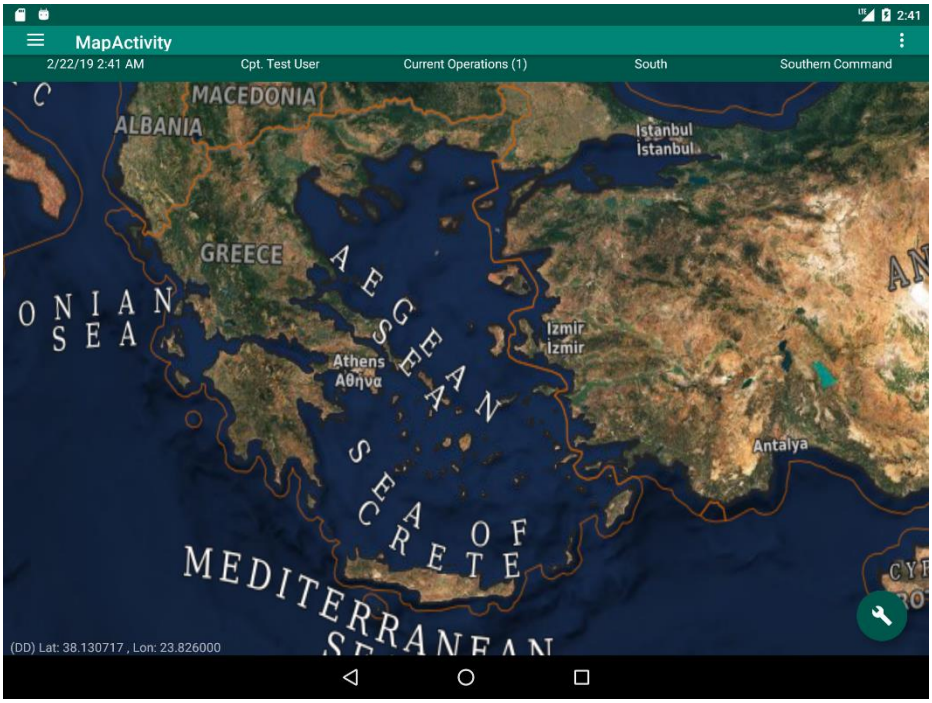
Η Εφαρμογή παρέχει ορισμένα βοηθητικά εργαλεία για την υποβοήθηση του Χρήστη κατά την χρήση της:





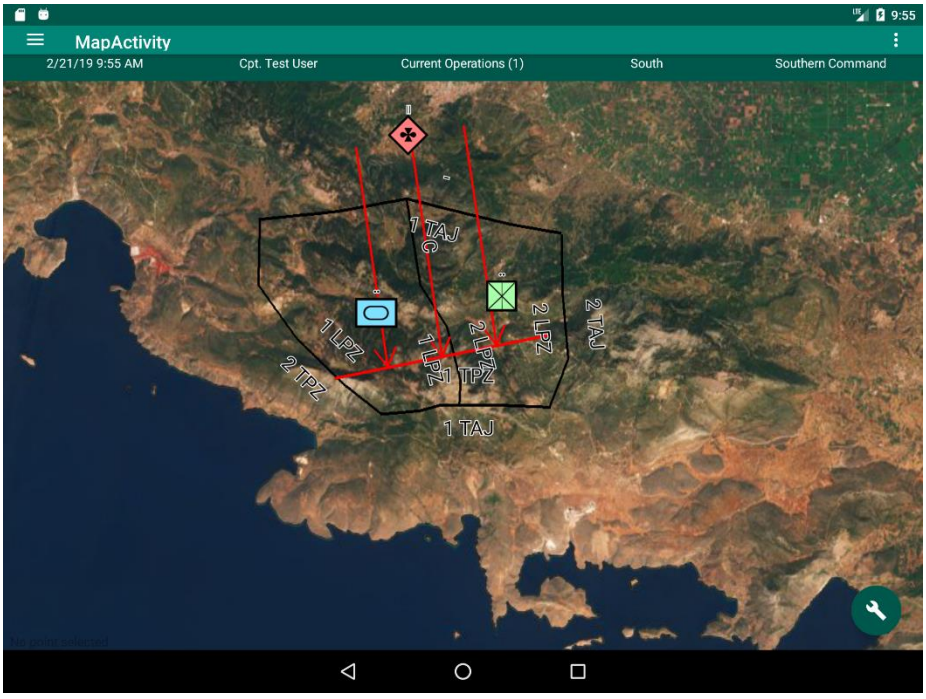
Εικόνα 84 Βοηθητικά Εργαλεία

- Default Zoom



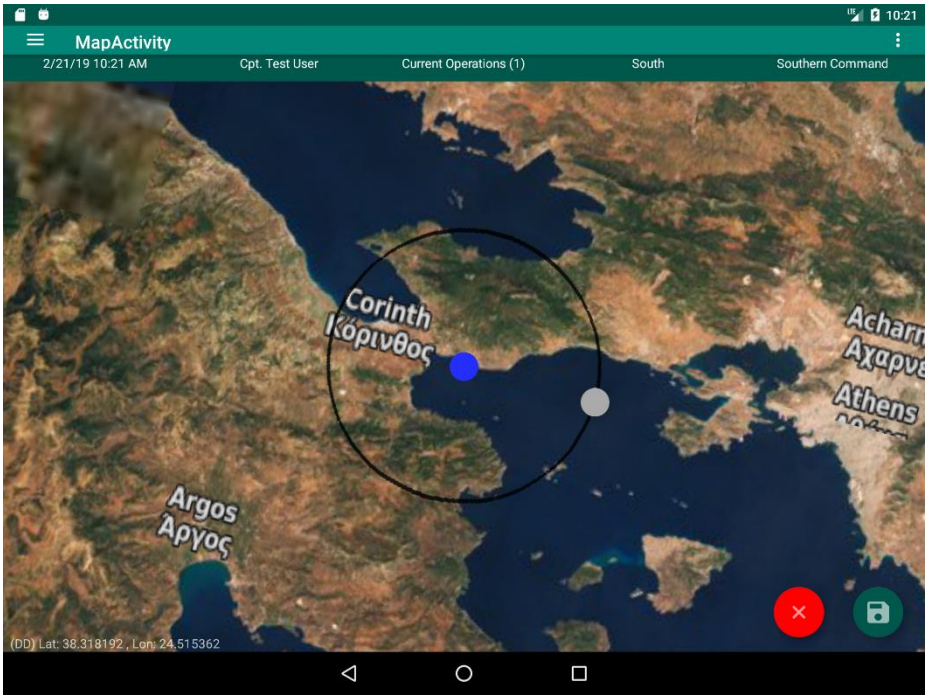
Εικόνα 85 Προκαθορισμένη Εστίαση

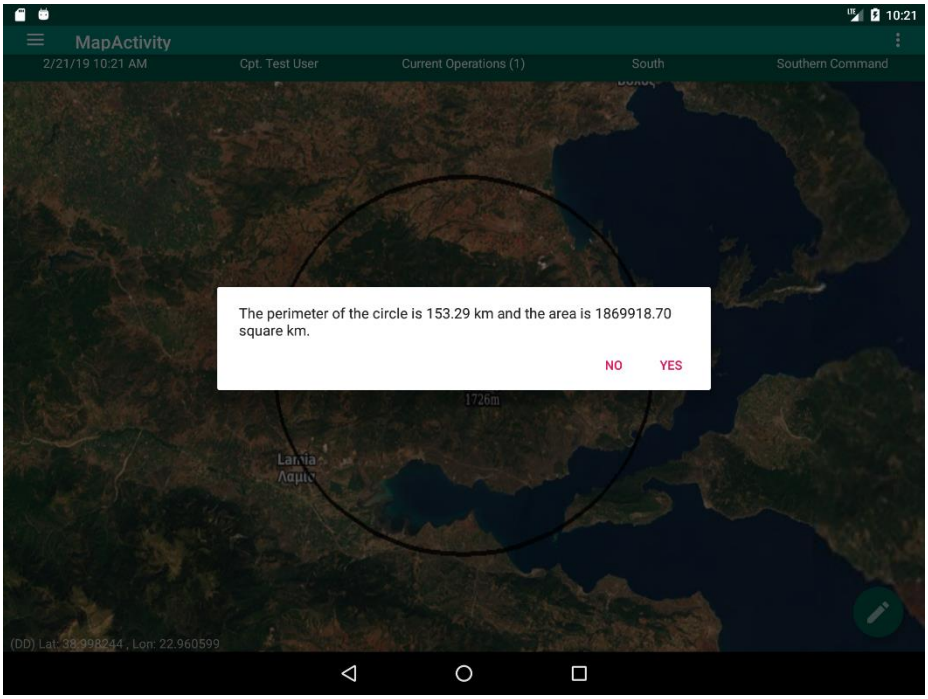
- Zoom to All Icons



Εικόνα 86 Εστίαση σε όλα τα εικονίδια

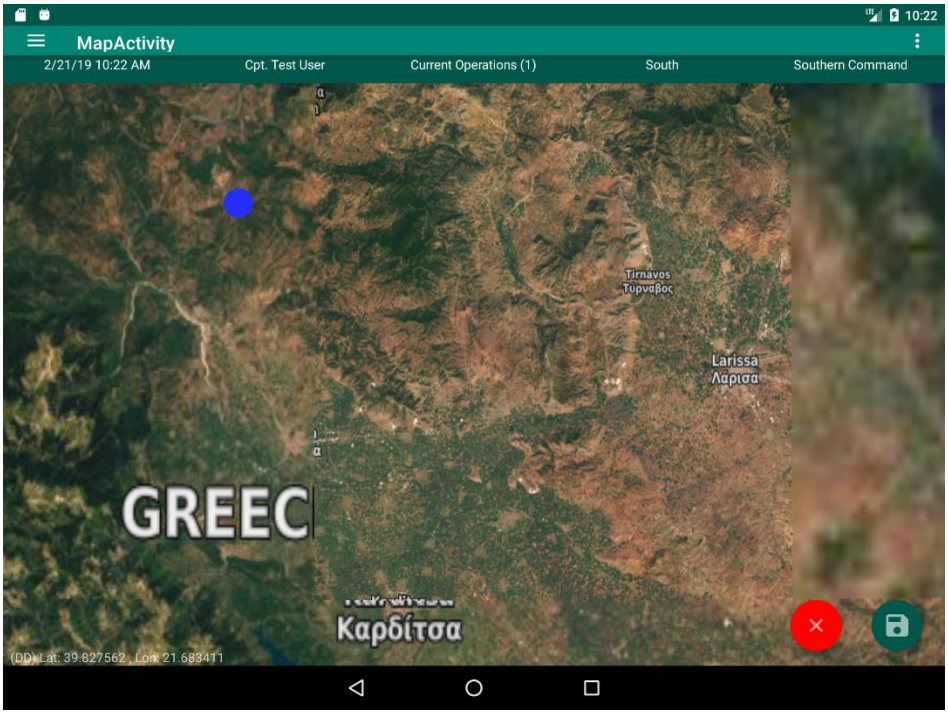
- Area Calculation (Circle)

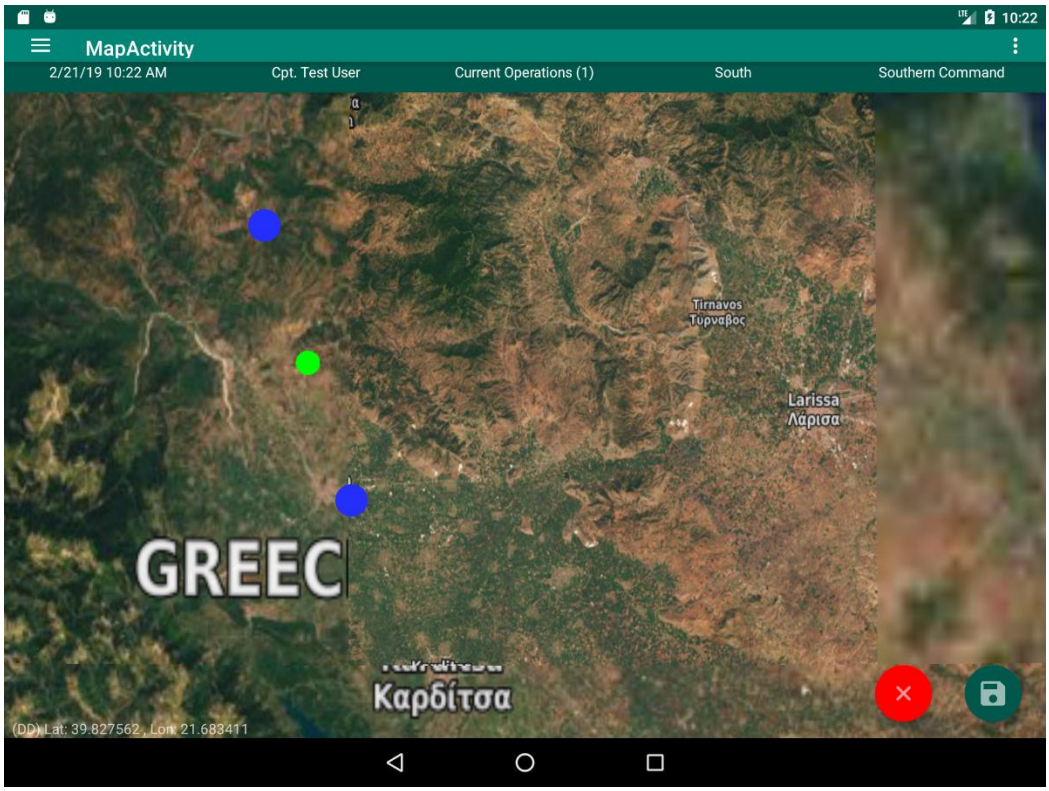


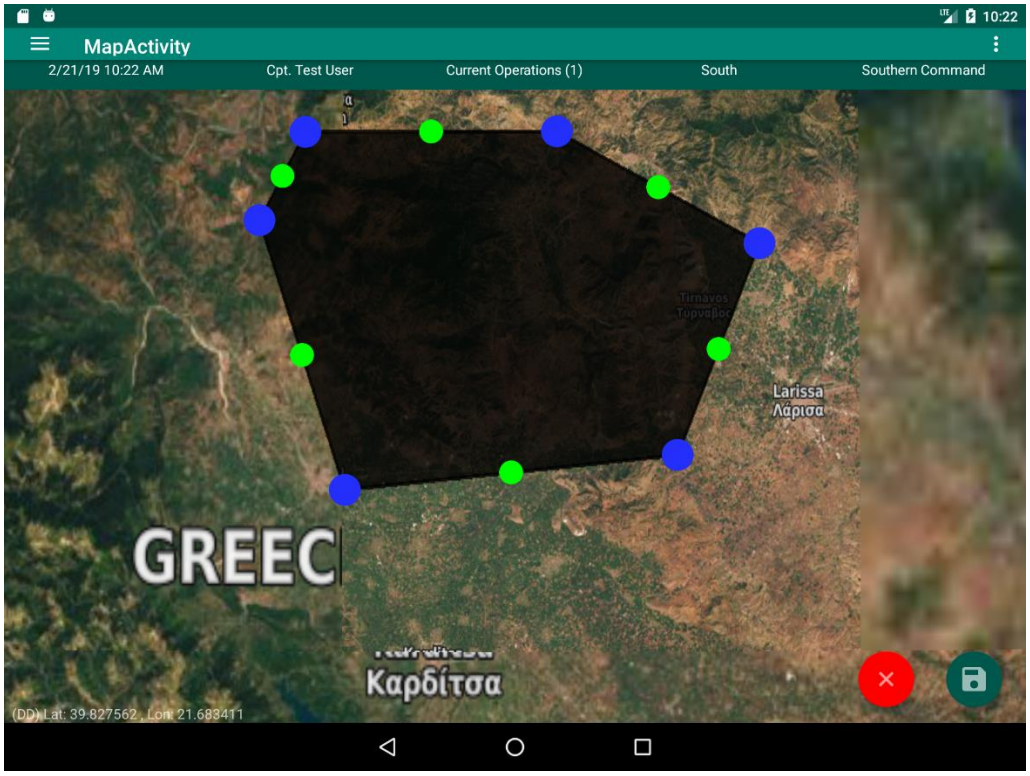


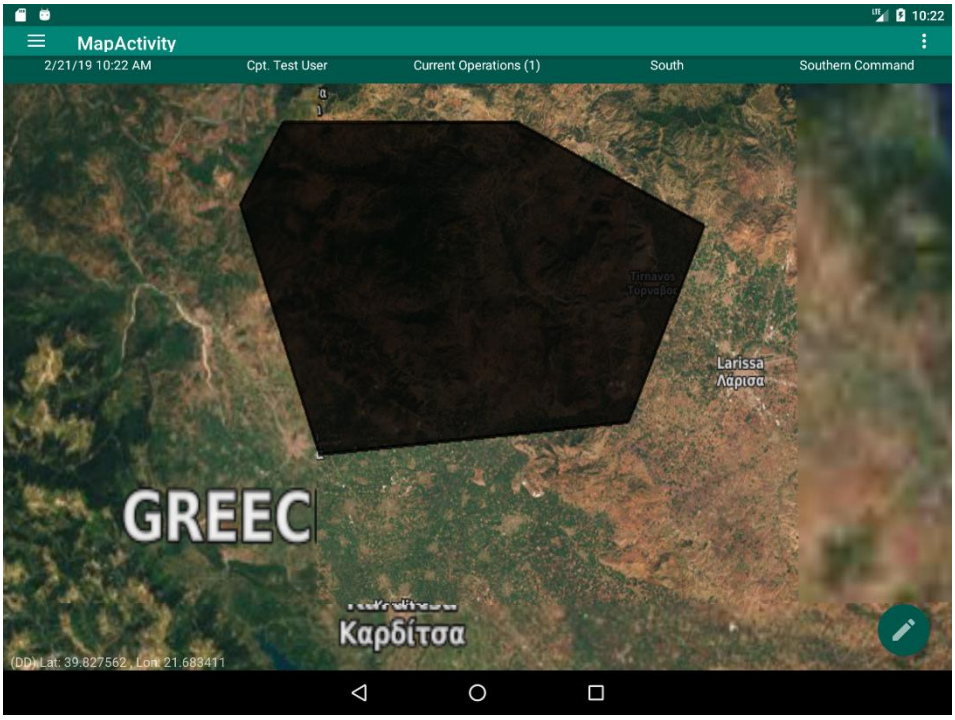
Εικόνα 87 Υπολογισμός Περιοχής Κύκλου

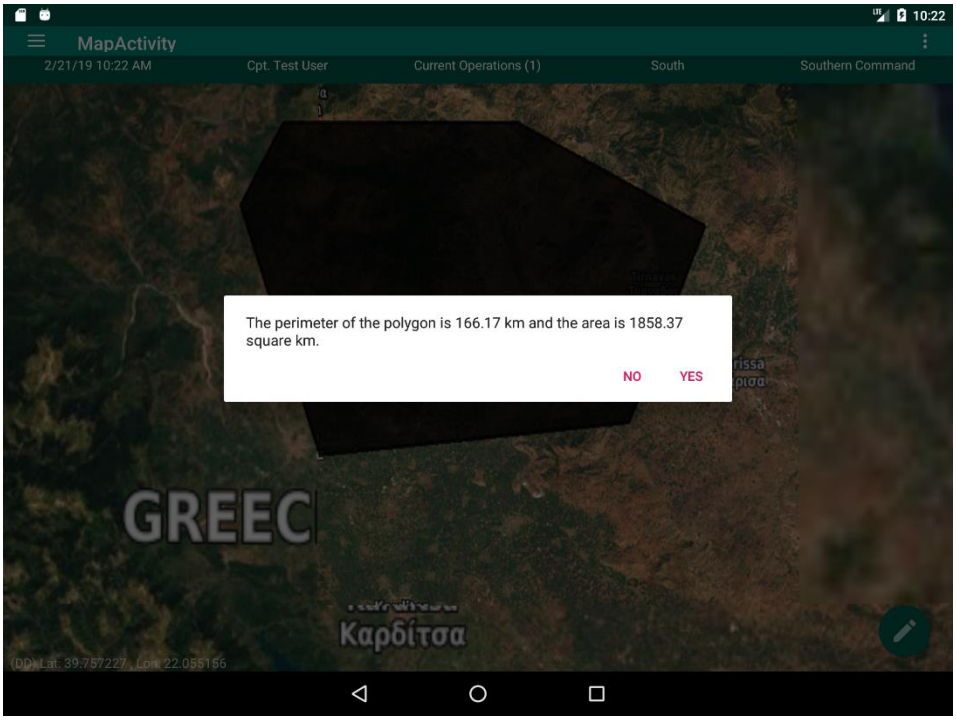
- Area Calculation (Polygon)





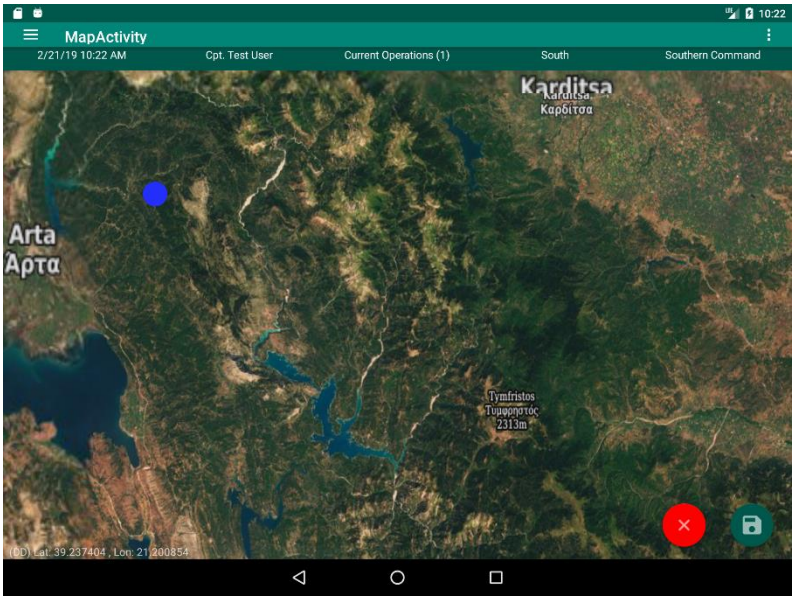

10:22
MapActivity
2/21/19 10:22 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Larissa
Λάρισα
GREEC
Καρδίτσα
(DD) Lat: 39.827562 , Lon: 21.683411

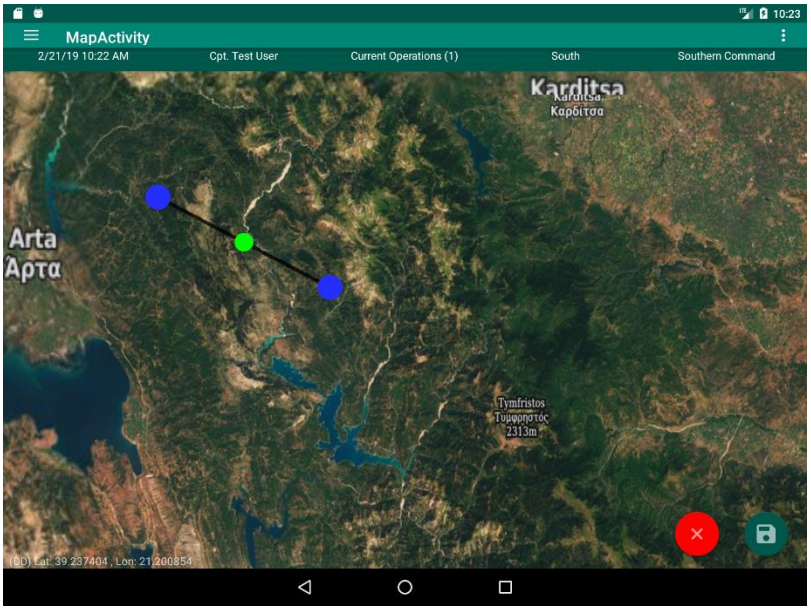

10:22
MapActivity
2/21/19 10:22 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Larissa
Λάρισα
GREEC
Καρδίτσα
(DD) Lat: 39.827562 , Lon: 21.683411

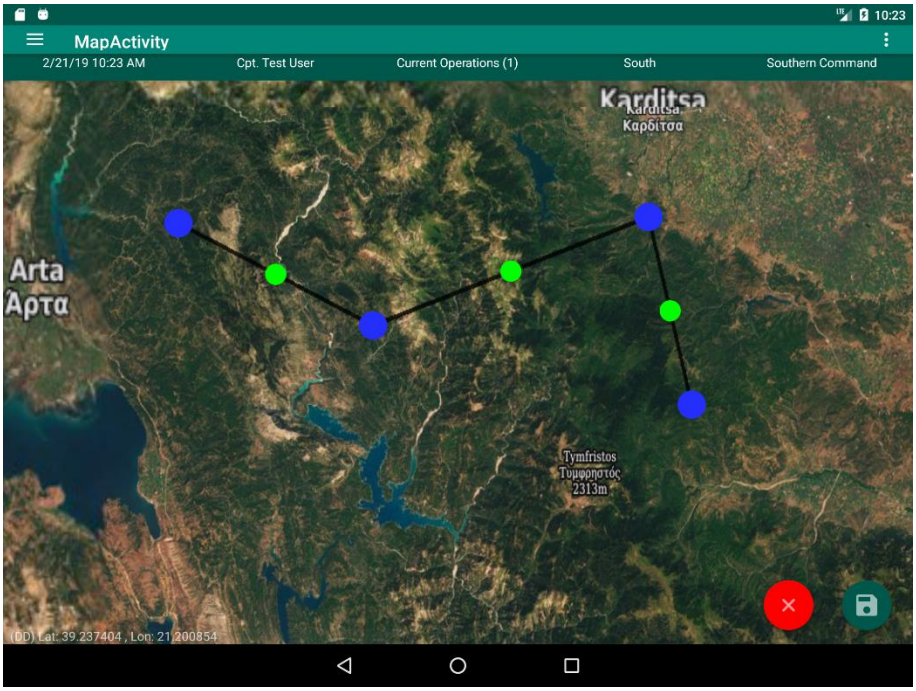


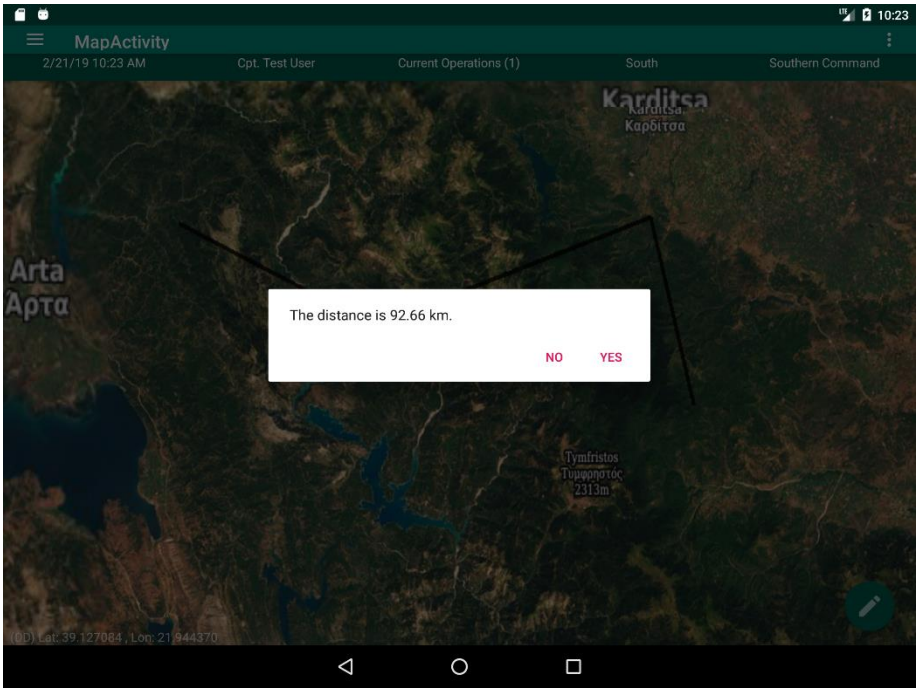
Εικόνα 88 Υπολογισμός Περιοχής Πολυγώνου

- Distance Calculation



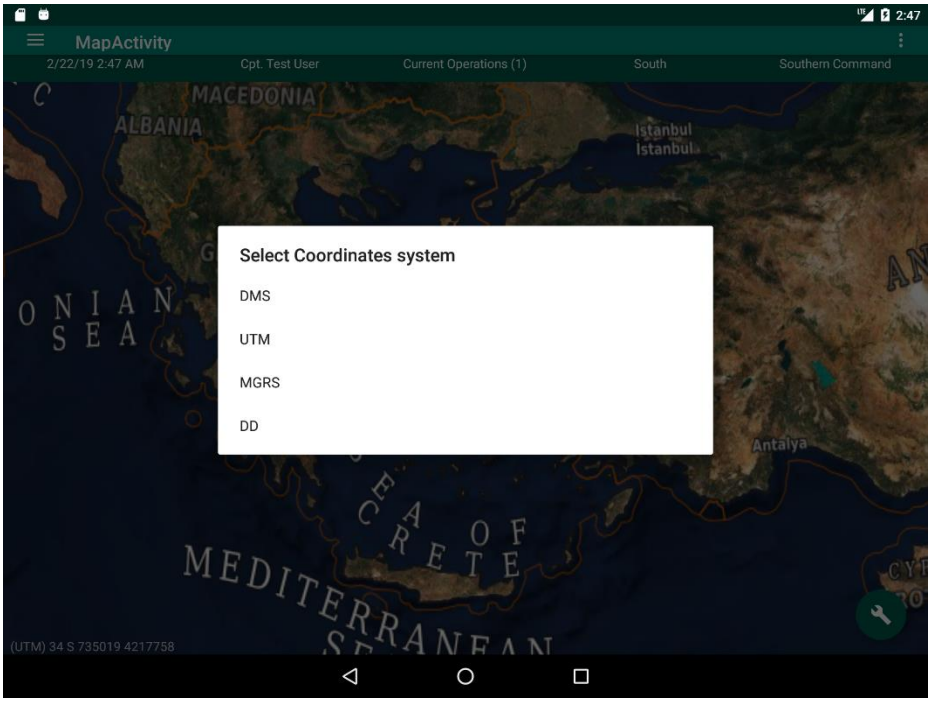

10:23
MapActivity
2/21/19 10:22 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Karditsa
Καρδίτσα
Arta
Άρτα
Tymfristos
Τυμφρηστός
2313m
(DD) Lat: 39.237404 , Lon: 21.200854

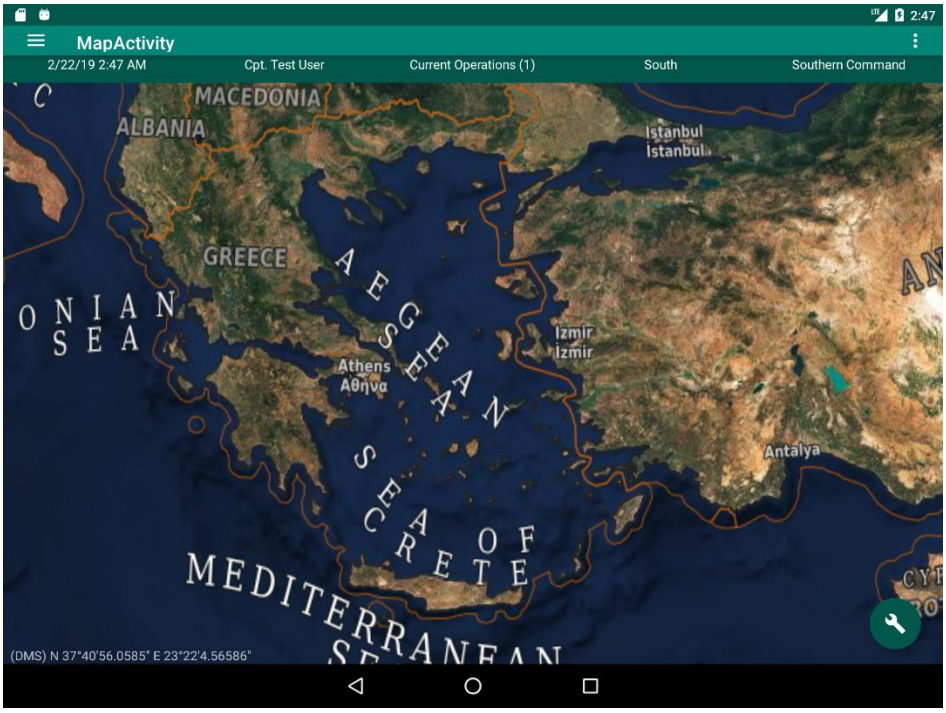

10:23
MapActivity
2/21/19 10:23 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Karditsa
Καρδίτσα
Arta
Άρτα
Tymfristos
Τυμφρηστός
2313m
(DD) Lat: 39.237404 , Lon: 21.200854



Εικόνα 89 Υπολογισμός Απόστασης

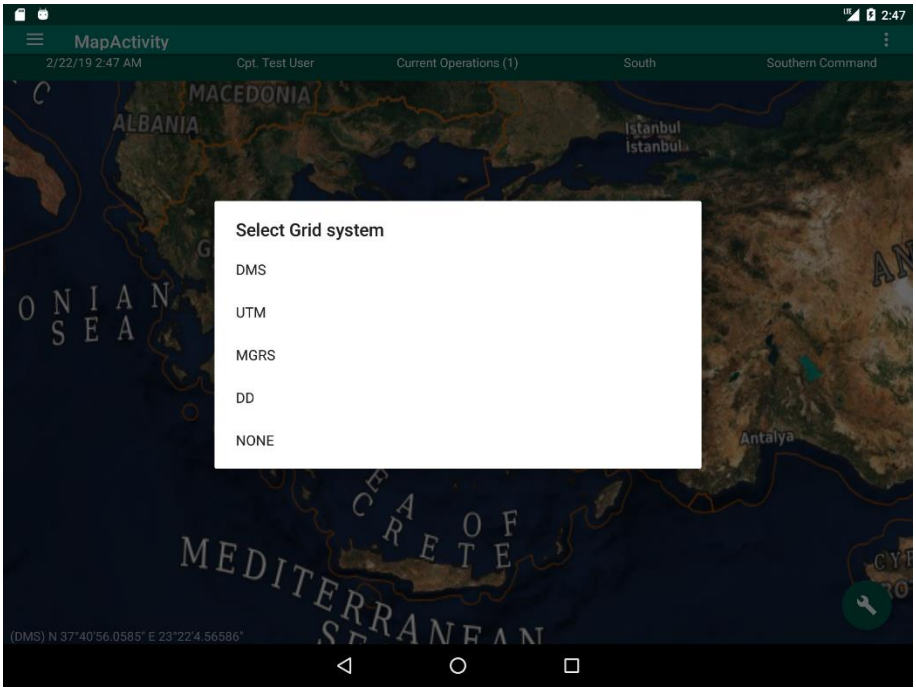
- Change Map Coordinate System

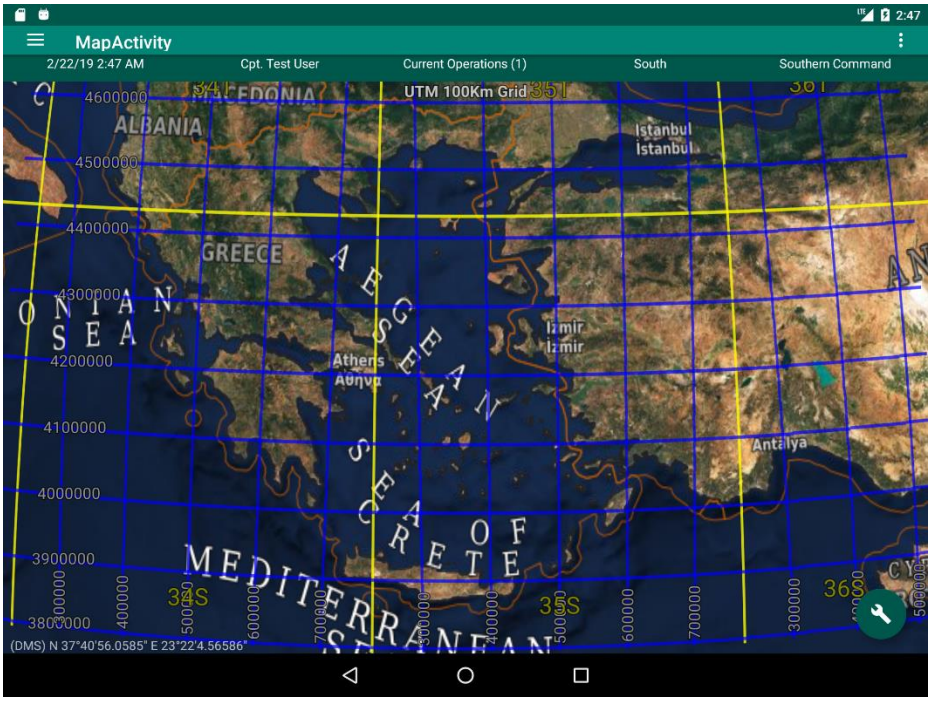




Εικόνα 90 Αλλαγή Συστήματος Συντεταγμένων

- Show Grid

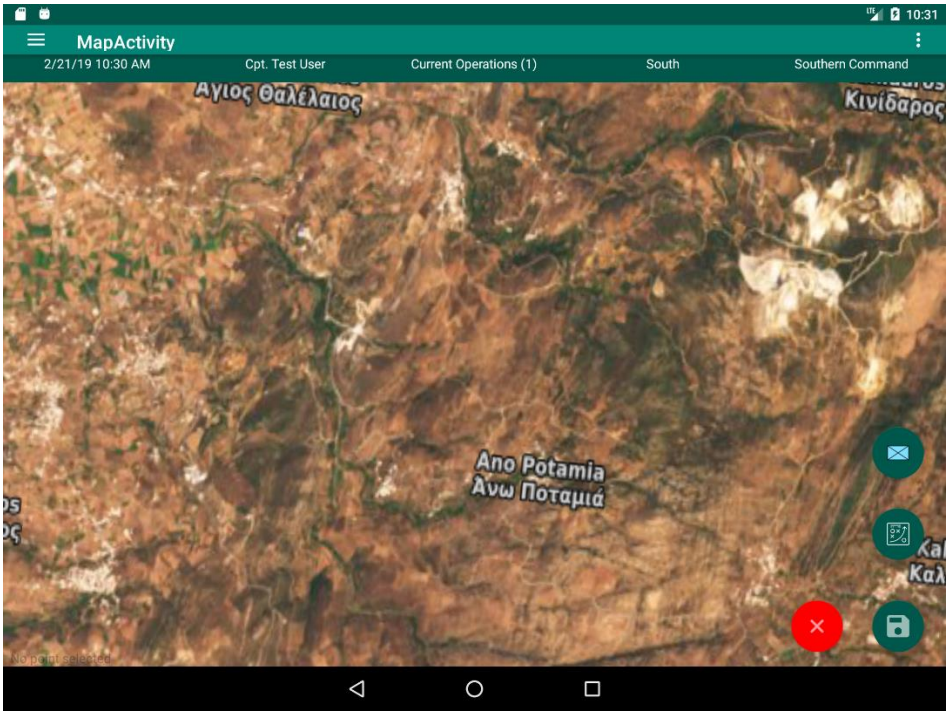




Εικόνα 91 Χρήση Γραμμών Τετραγωνισμού

4.3.8 **Σχεδίαση Διαφανούς Επιχειρήσεων**

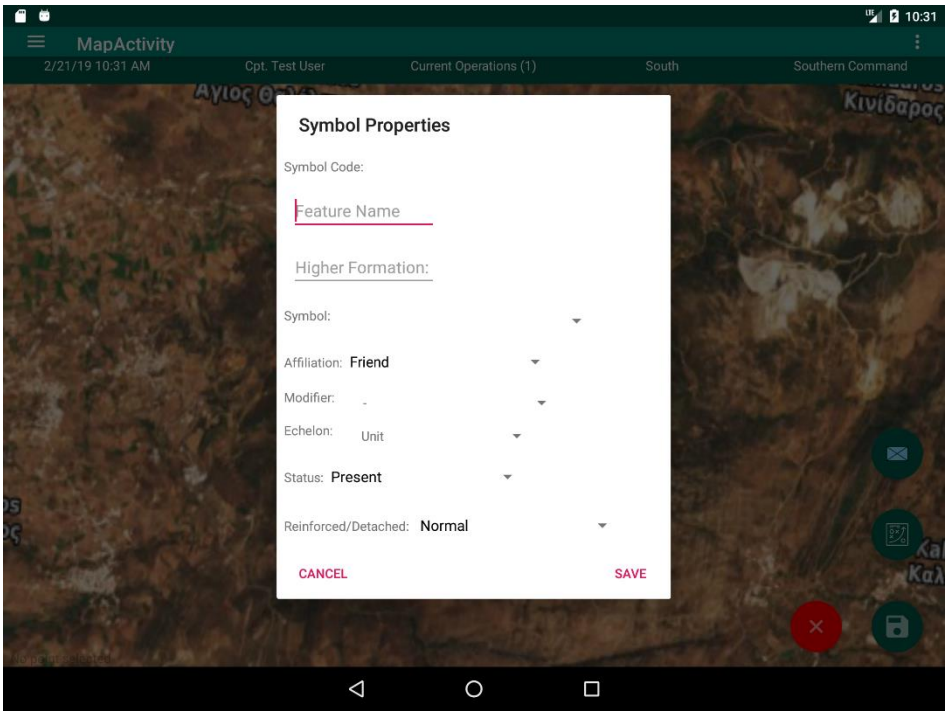
Για την σχεδίαση διαφανούς επιχειρήσεων, ο Χρήστης πατάει πάνω στο κάτω δεξιά κομβίο για να μεταβεί σε κατάσταση «NEW».

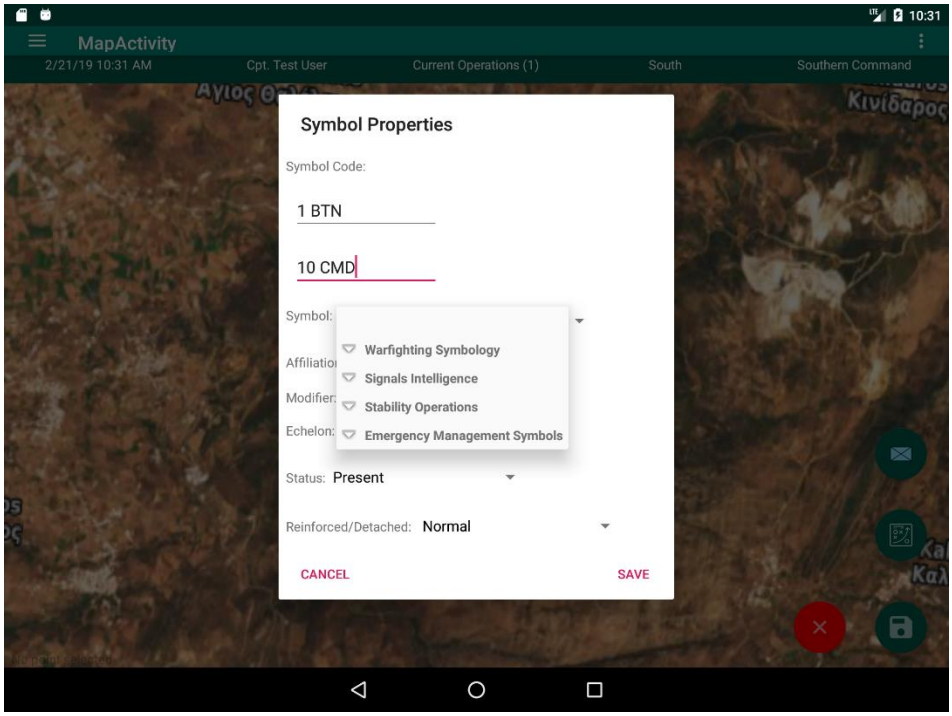


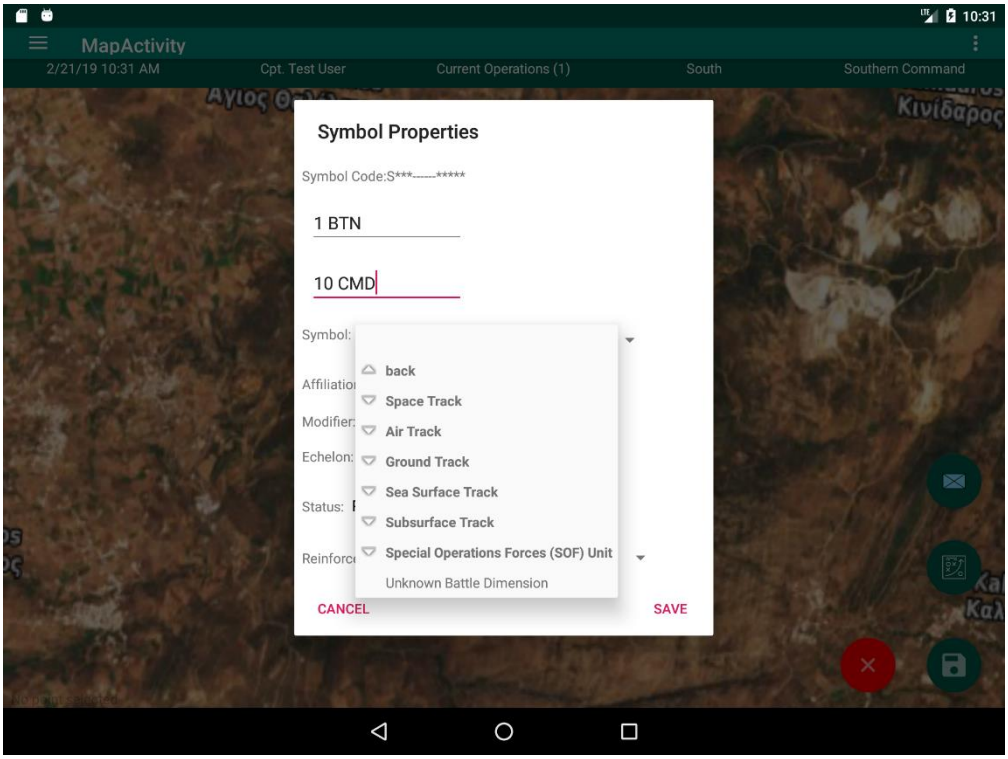
Εικόνα 92 Μενού Εισαγωγής Γραφικών

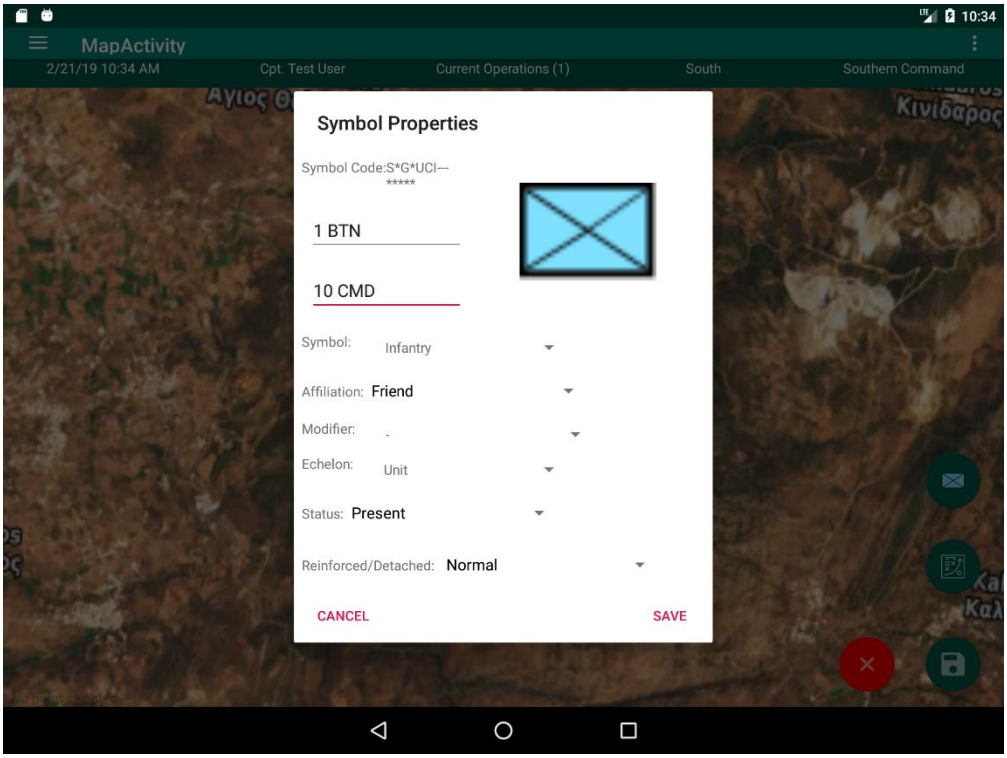
Σε αυτήν την κατάσταση μπορεί να εισάγει νέα εικονίδια ή γραφικά τακτικής, και να τα αποθηκεύσει πατώντας πάνω στο κομβίο κάτω δεξιά. Οποιαδήποτε στιγμή θέλει μπορεί να ακυρώσει την τρέχουσα διαδικασία και να επιστρέψει στην αρχική κατάσταση πατώντας το κόκκινο κομβίο κάτω αριστερά.

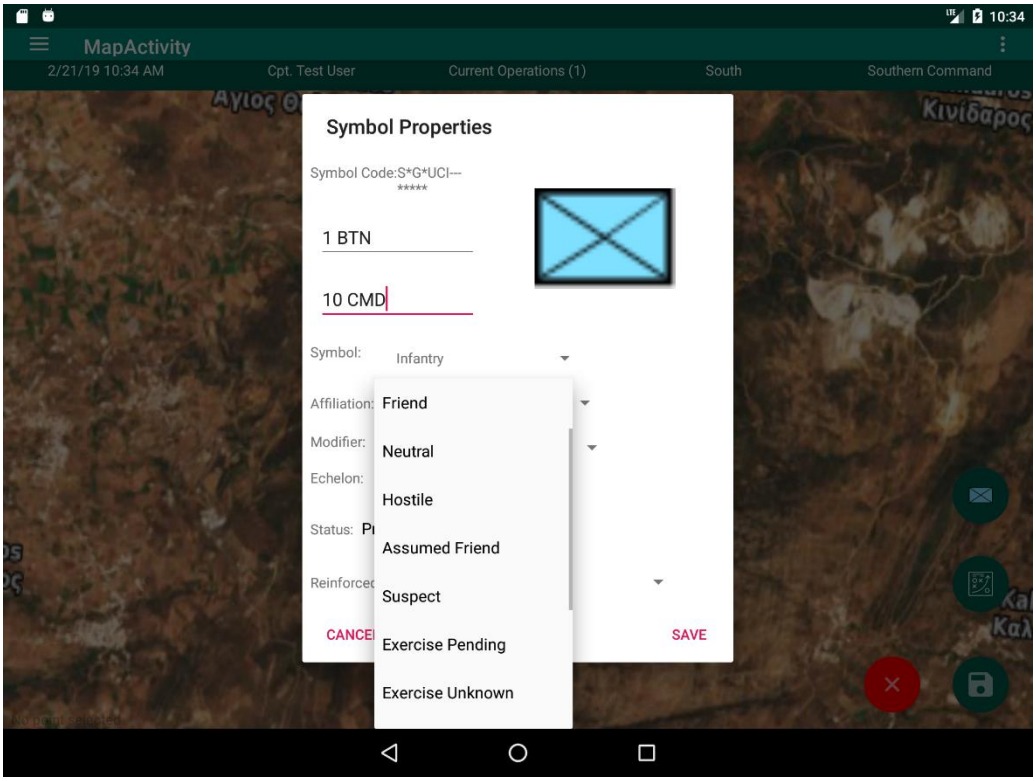
Αν επιθυμεί να εισάγει νέο εικονίδιο Μονάδας / Εξοπλισμό / Εγκατάστασης, πρέπει να πιέσει το πάνω κομβίο, που απεικονίζει μια μονάδα. Αυτό θα ανοίξει το μενού από το οποίο θα επιλέξει τα στοιχεία για την νέα μονάδα, όπως όνομα Μονάδας, όνομα προϊσταμένου, τύπος Μονάδας, Επίπεδο Μονάδας κ.ο.κ.

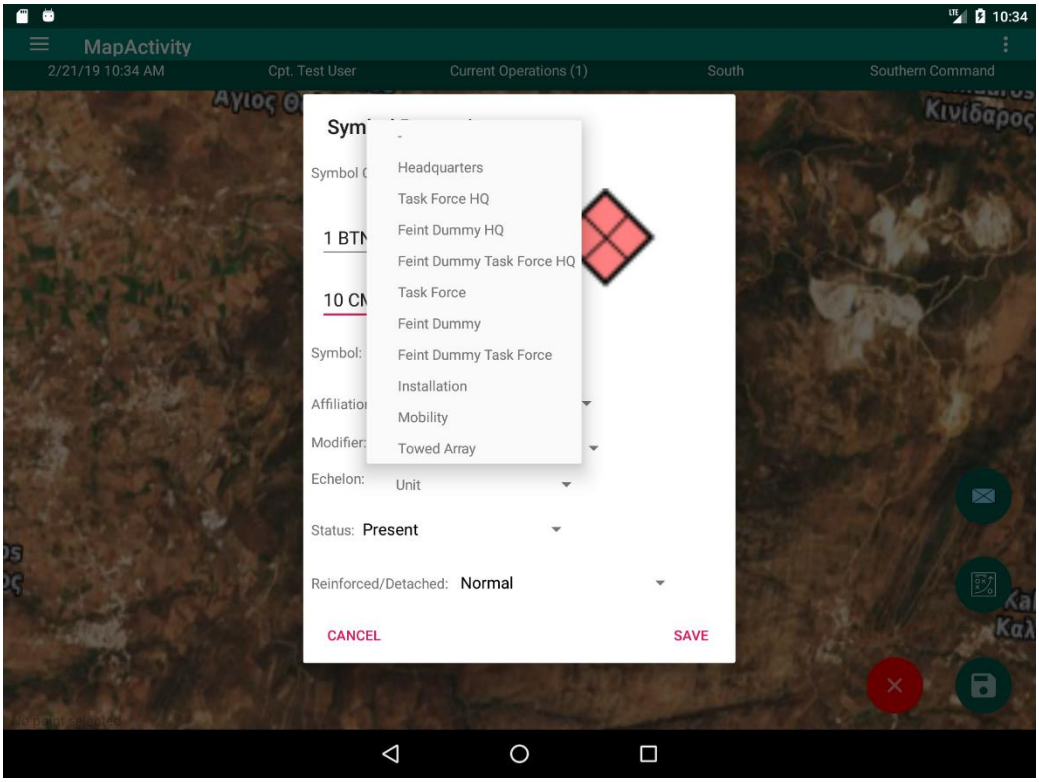


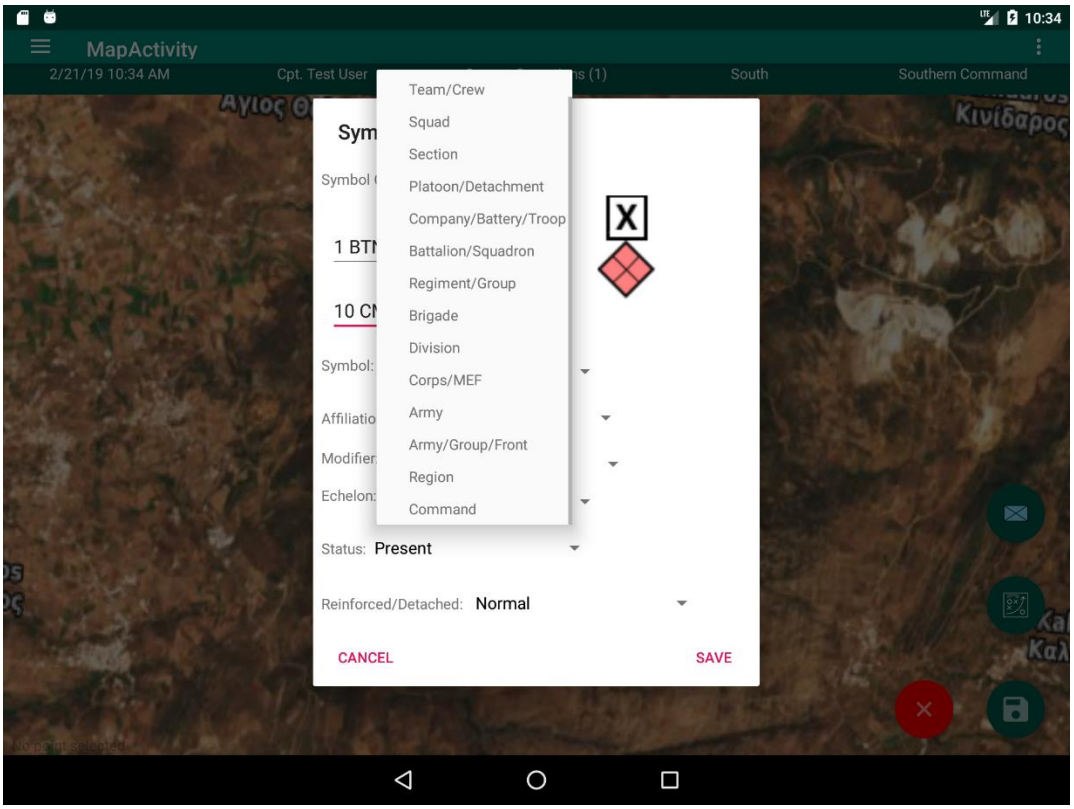

10:31
MapActivity
2/21/19 10:31 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Symbol Properties
Symbol Code:
1 BTN
10 CMD
Symbol:
Warfighting Symbology
Signals Intelligence
Stability Operations
Emergency Management Symbols
Status: Present
Reinforced/Detached: Normal
CANCEL
SAVE

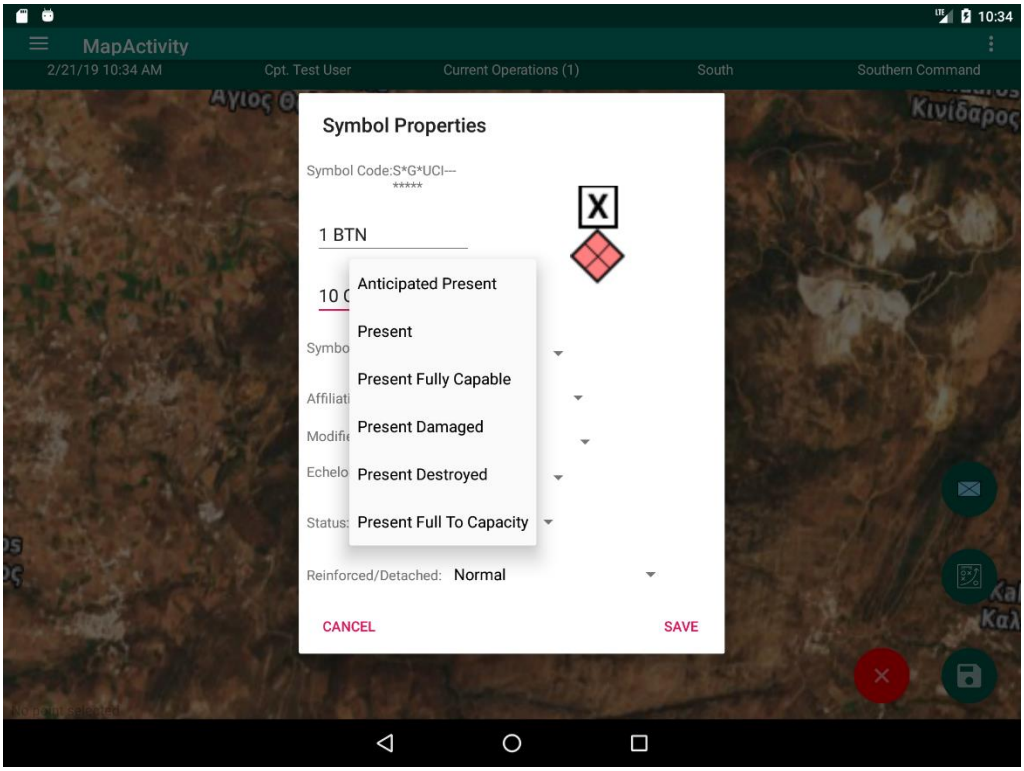

10:31
MapActivity
2/21/19 10:31 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Symbol Properties
Symbol Code:S***------*****
1 BTN
10 CMD
Symbol:
back
Space Track
Air Track
Ground Track
Sea Surface Track
Subsurface Track
Special Operations Forces (SOF) Unit
Unknown Battle Dimension
CANCEL
SAVE

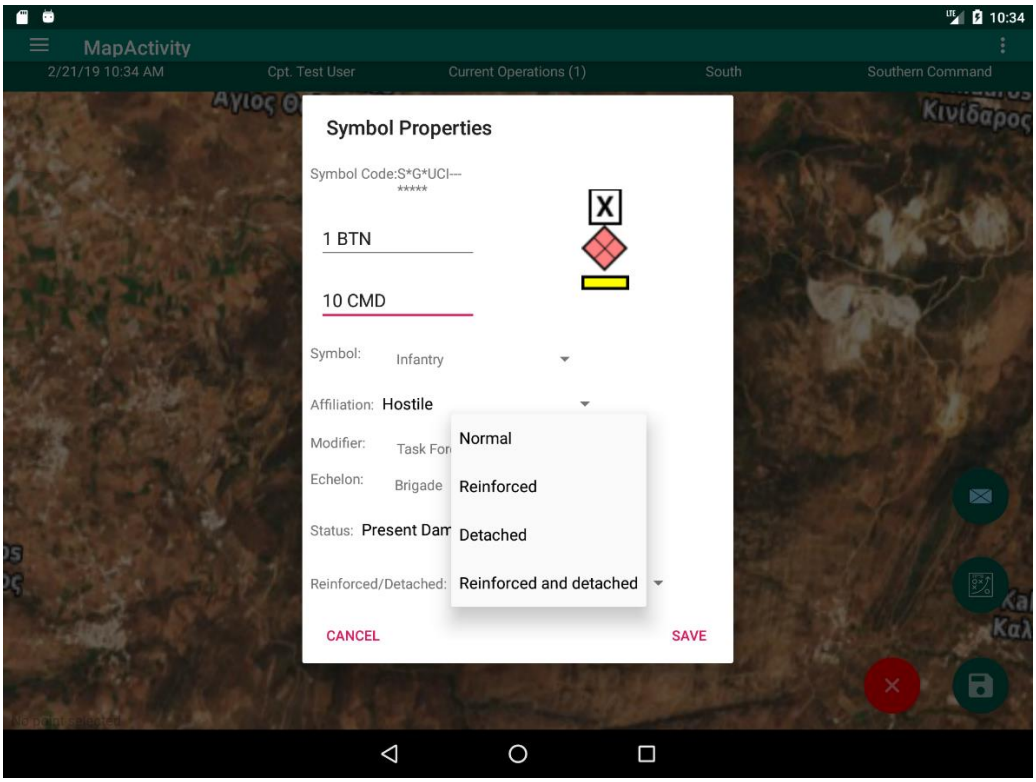

10:34
MapActivity
2/21/19 10:34 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Symbol Properties
Symbol Code:S*G*UCI---*****
1 BTN
10 CMD
Symbol: Infantry
Affiliation: Friend
Modifier:
Echelon: Unit
Status: Present
Reinforced/Detached: Normal
CANCEL
SAVE


10:34
MapActivity
2/21/19 10:34 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Symbol Properties
Symbol Code:S*G*UCI---*****
1 BTN
10 CMD
Symbol: Infantry
Affiliation:
Modifier:
Echelon:
Status:
Reinforced
CANCEL
SAVE
Friend
Neutral
Hostile
Assumed Friend
Suspect
Exercise Pending
Exercise Unknown


10:34
MapActivity
2/21/19 10:34 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Headquarters
Task Force HQ
Feint Dummy HQ
Feint Dummy Task Force HQ
Task Force
Feint Dummy
Feint Dummy Task Force
Installation
Mobility
Towed Array
Echelon:
Unit
Status:
Present
Reinforced/Detached:
Normal
CANCEL
SAVE


10:34
MapActivity
2/21/19 10:34 AM
Cpt. Test User
South
Southern Command
Team/Crew
Squad
Section
Platoon/Detachment
Company/Battery/Troop
Battalion/Squadron
Regiment/Group
Brigade
Division
Corps/MEF
Army
Army/Group/Front
Region
Command
Status:
Present
Reinforced/Detached:
Normal
CANCEL
SAVE

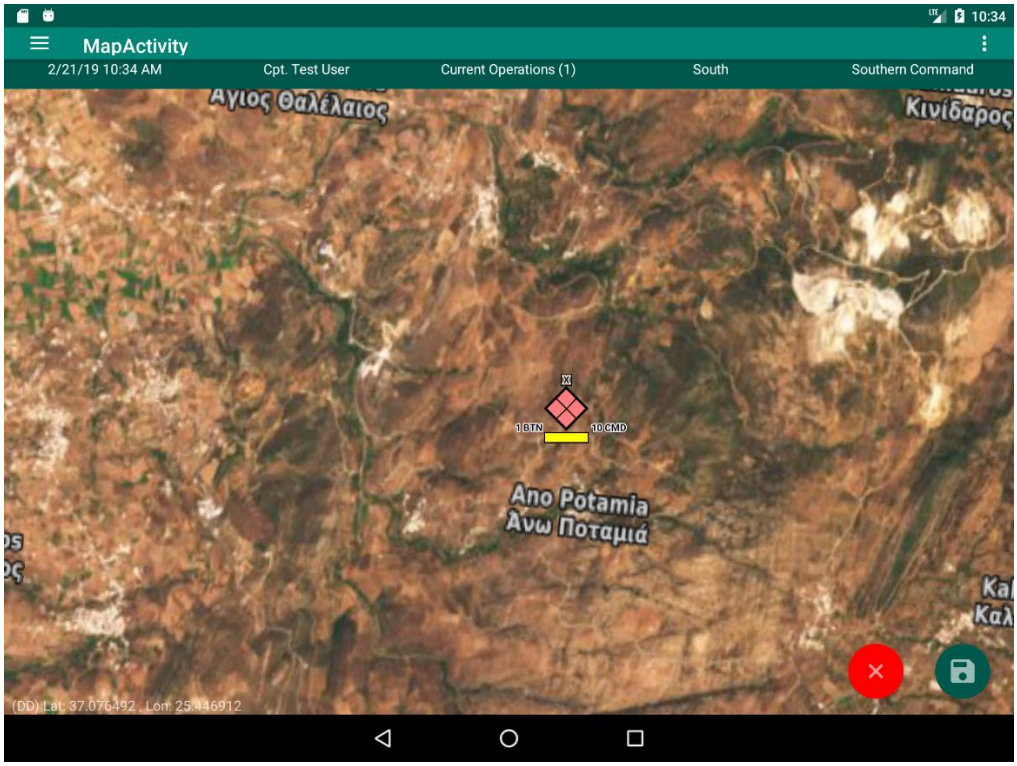




Εικόνα 93 Εισαγωγή Εικονιδίου Μονάδας

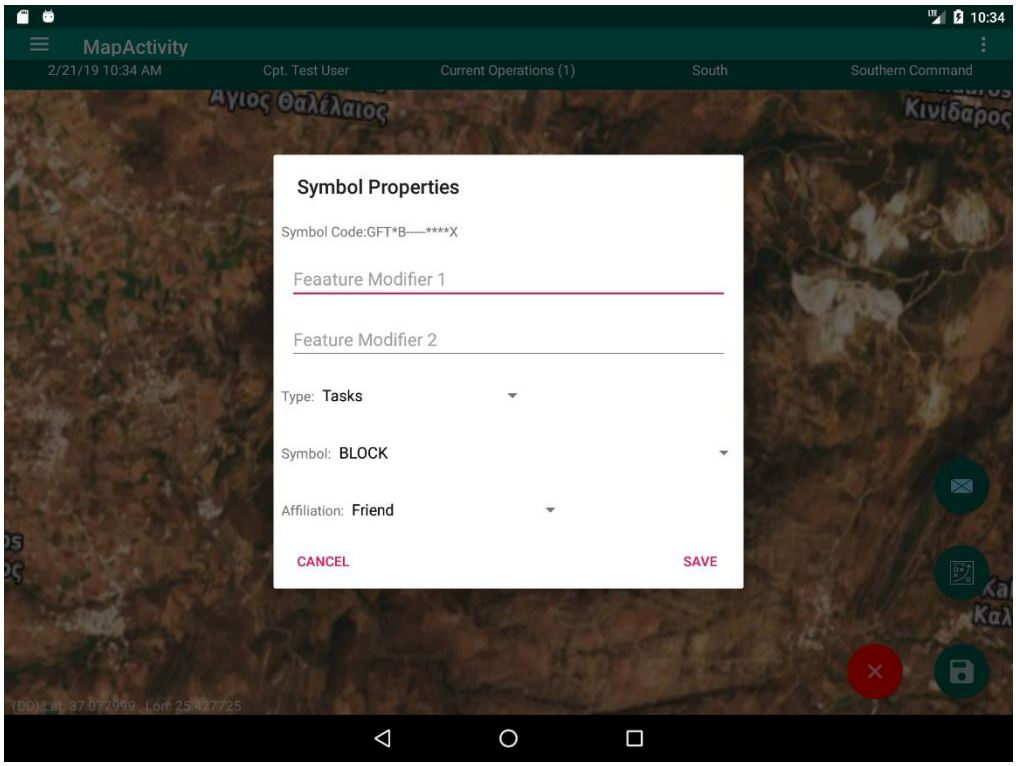
Μόλις έχει σχηματίσει την Μονάδα που επιθυμεί, πατάει την επιλογή «Save» για να τοποθετήσει την Μονάδα στον Χάρτη. Σε

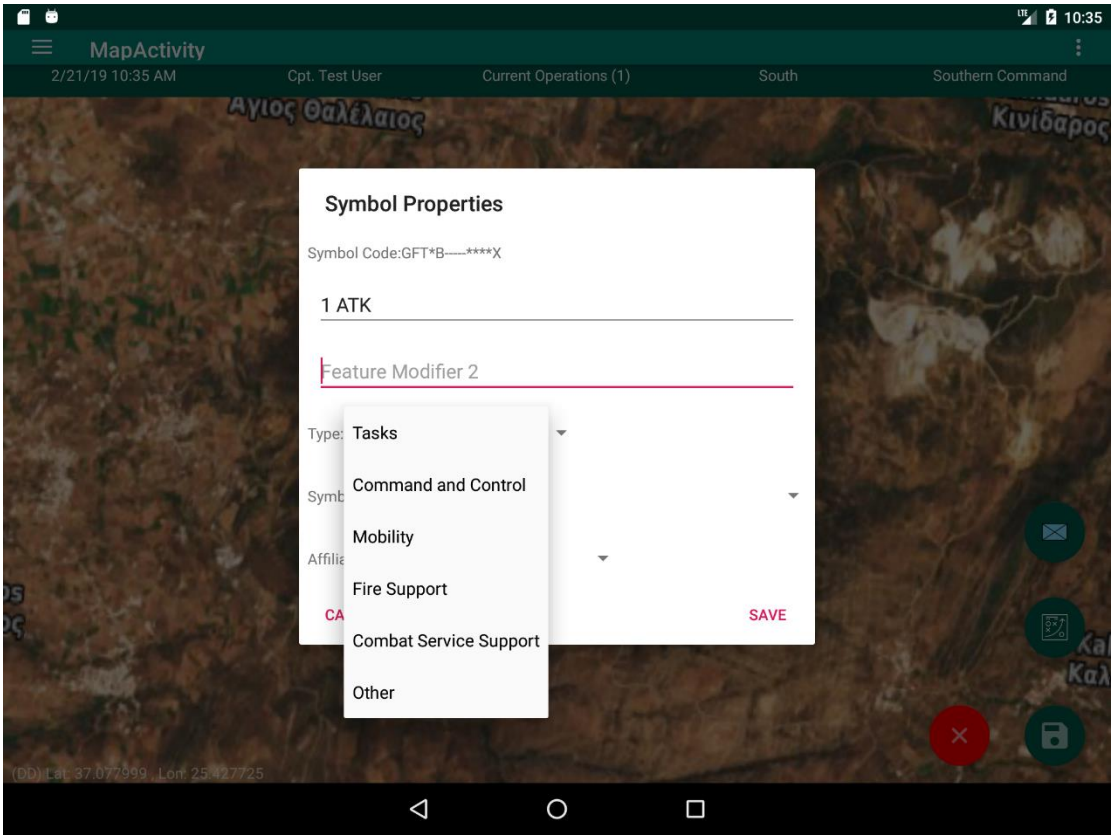
οποιαδήποτε φάση της δημιουργίας, η επιλογή «Cancel» κλείνει το παράθυρο χωρίς καμιά αλλαγή.

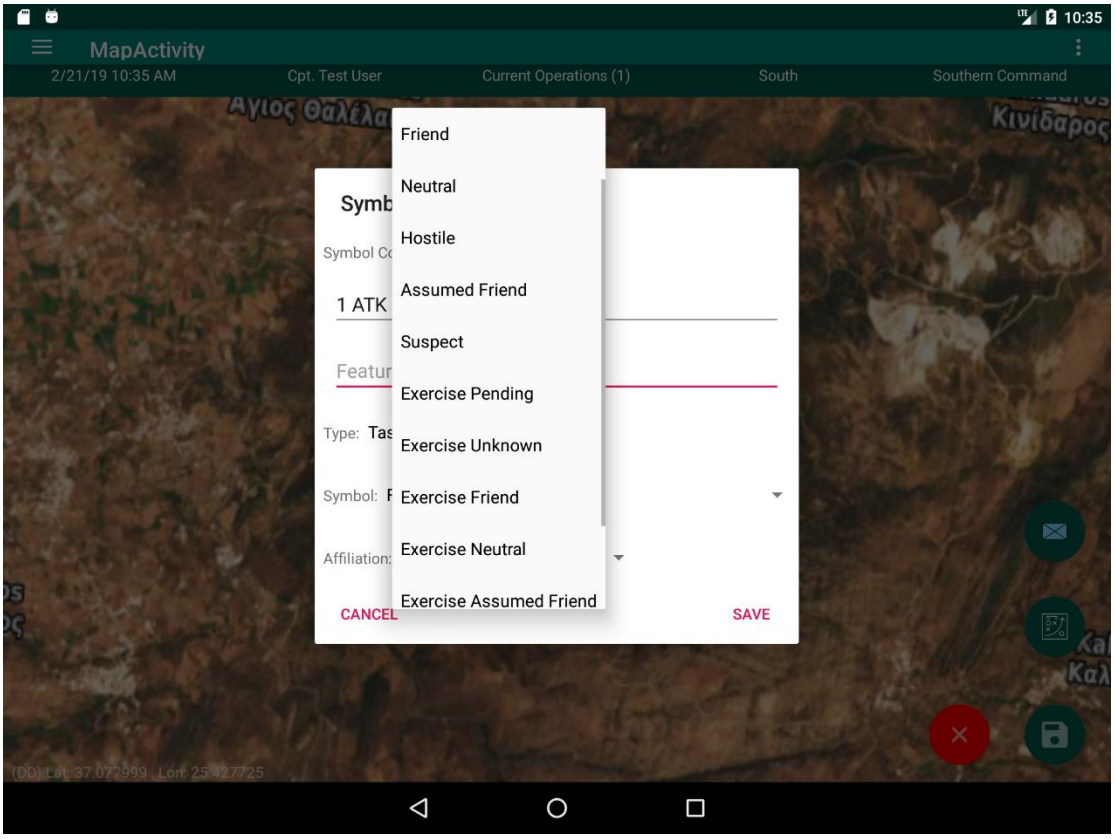


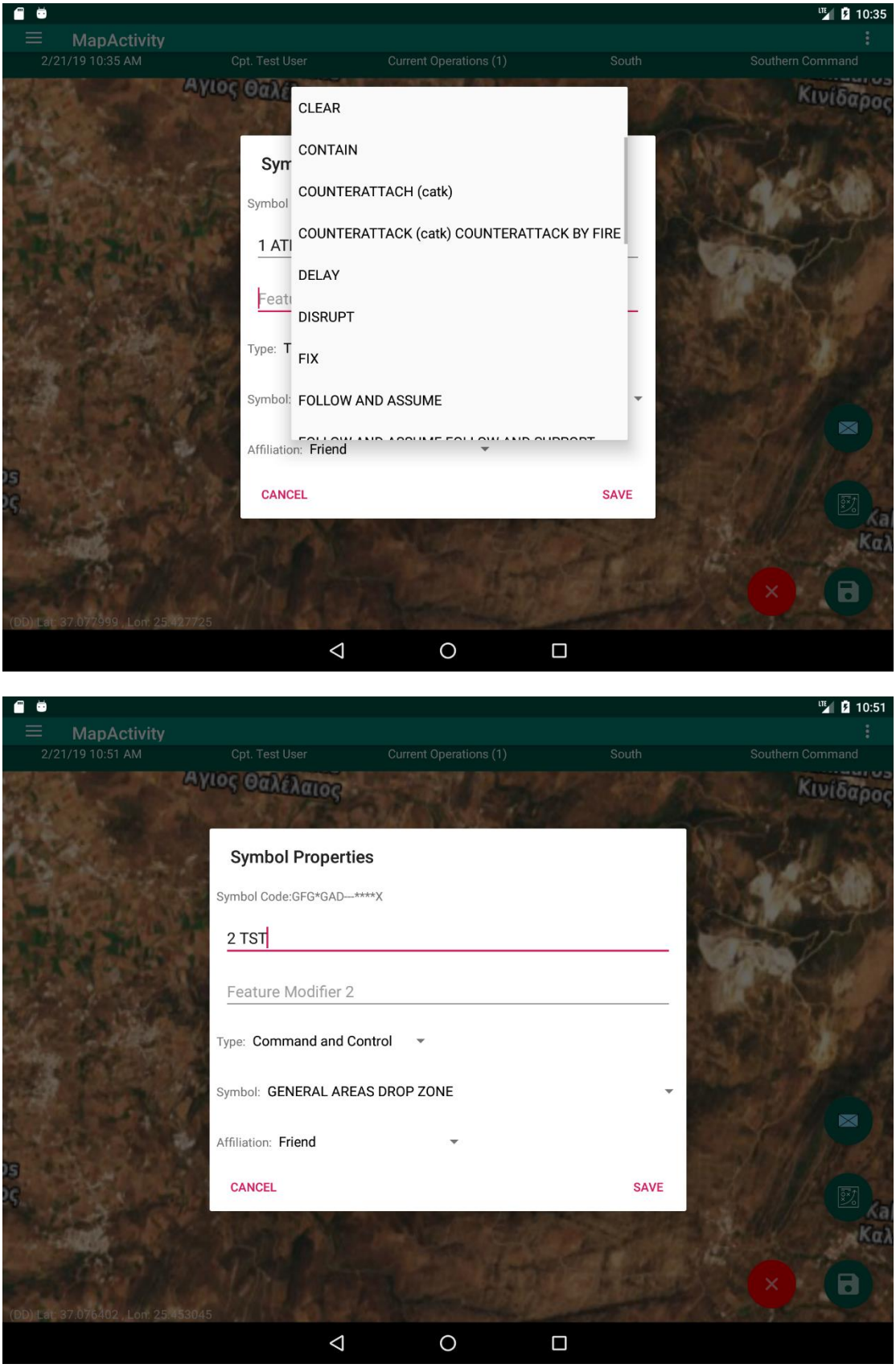
Εικόνα 94 Εικονίδιο Μονάδας

Παρόμοια διαδικασία εκτελείται και για τα γραφικά τακτικής:




10:35
MapActivity
2/21/19 10:35 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Symbol Properties
Symbol Code:GFT*B-----****X
1 ATK
Feature Modifier 2
Type:
Tasks
Command and Control
Mobility
Fire Support
Combat Service Support
Other
SAVE

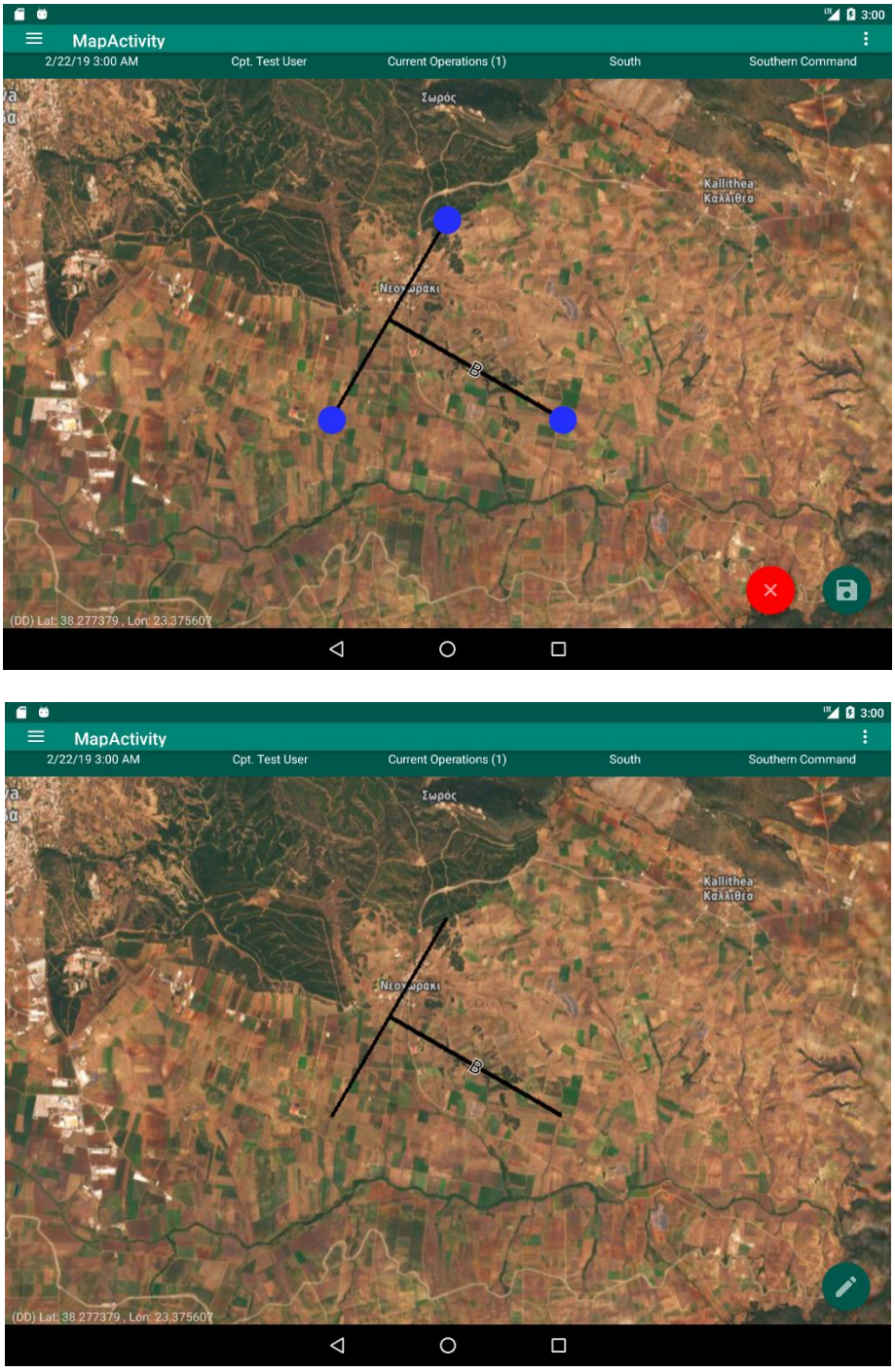

10:35
MapActivity
2/21/19 10:35 AM
Cpt. Test User
Current Operations (1)
South
Southern Command
Friend
Neutral
Hostile
Assumed Friend
Suspect
Exercise Pending
Exercise Unknown
Exercise Friend
Exercise Neutral
Exercise Assumed Friend
1 ATK
CANCEL
SAVE



Εικόνα 95 Εισαγωγή Γραφικού Τακτικής

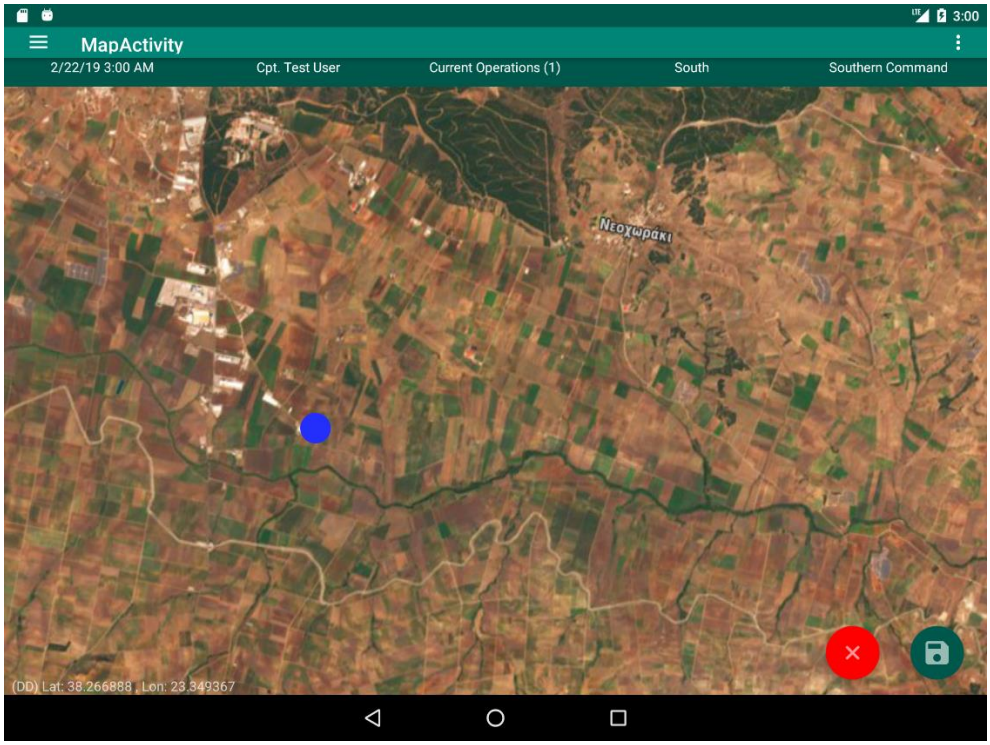
Βασική διαφορά με τα εικονίδια Μονάδων είναι ότι τα γραφικά τακτικής μπορούν να αποτελούνται από προκαθορισμένα σχήματα ή από σχήματα ελεύθερης μορφής.

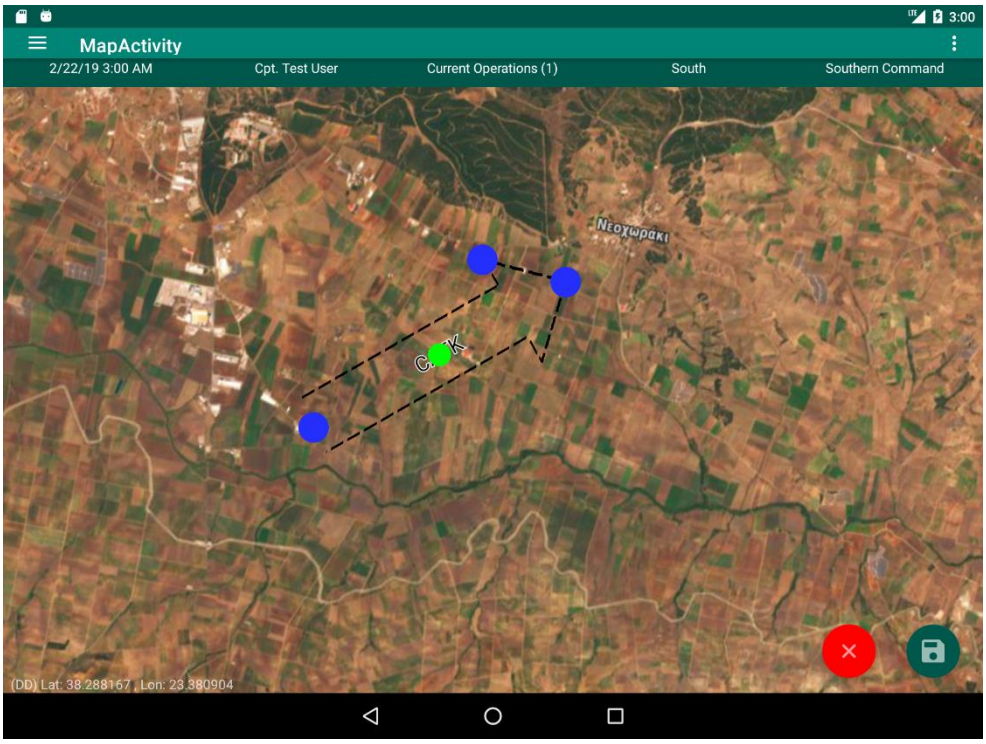
Τα προκαθορισμένα σχήματα τοποθετούνται στον χάρτη, και ο χρήστης μπορεί να πειράξει τα σημεία τους με συγκεκριμένο τρόπο, ώστε να μην χαλάσει το σχήμα τους:



Εικόνα 96 Προκαθορισμένα Γραφικά Τακτικής

Τα ελεύθερα σχήματα απαιτούν από τον χρήστη να καθορίσει ορισμένα σημεία, πριν σχηματιστούν, και επιτρέπουν μεγαλύτερη ελευθερία στην σχεδίαση τους:

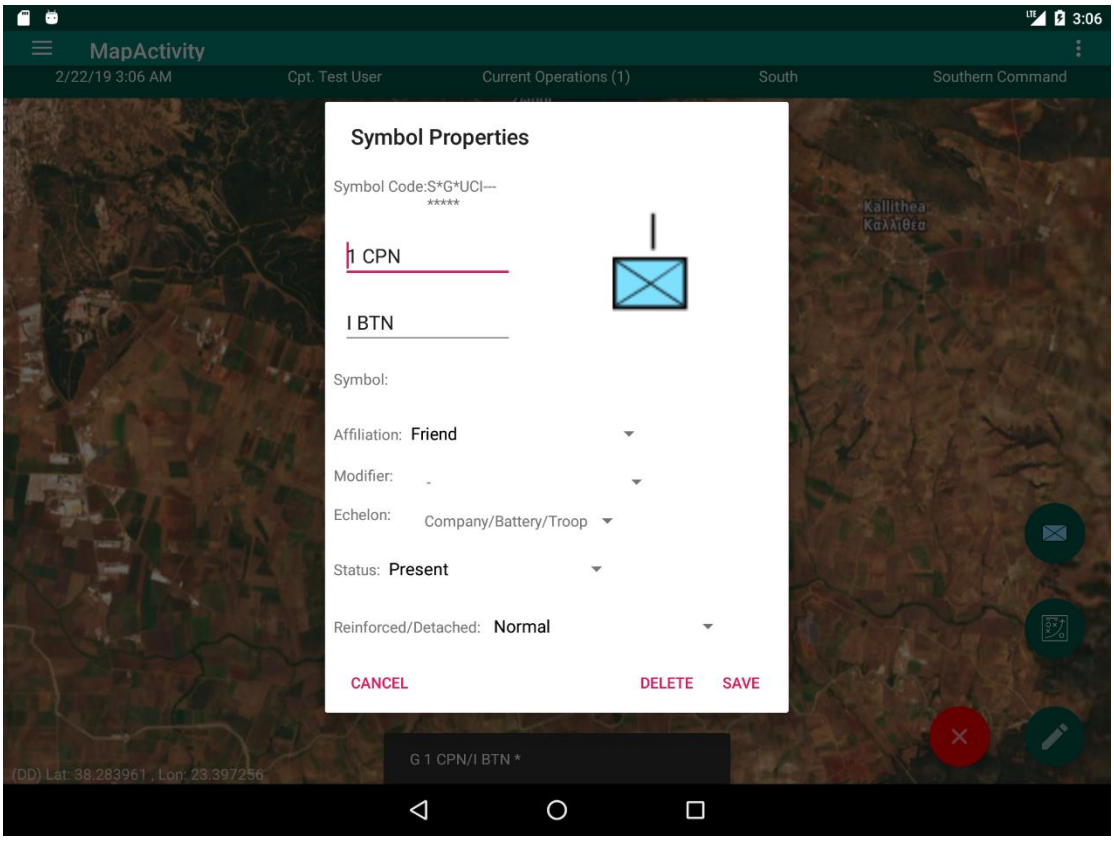




Εικόνα 97 Γραφικά Τακτικής Ελεύθερης Μορφής

Όταν ο Χάρτης δεν είναι σε κατάσταση εισαγωγής νέου σχήματος, μπορεί να μπει σε κατάσταση επεξεργασίας, με διπλό πάτημα πάνω σε ένα εικονίδιο. Ο χρήστης μπορεί να μεταφέρει τότε το εικονίδιο σε άλλο σημείο, και μόλις πατήσει το πράσινο εικονίδιο κάτω αριστερά, η νέα τοποθεσία θα αποθηκευτεί.

Τέλος, μπορούμε να αλλάξουμε κάποια στοιχεία μιας μονάδας ή ενός γραφικού τακτικής, πατώντας πάνω στο εικονίδιο παρατεταμένα. Αυτό θα ανοίξει το παράθυρο τροποποίησης, επιτρέποντας αλλαγές σε κάποια δεδομένα της Μονάδας, έως και πλήρους διαγραφής του γραφικού.



Εικόνα 98 Μενού Επεξεργασίας Γραφικού

5. Επίλογος

5.1 Σύνοψη

Στα πλαίσια της διπλωματικής μου εργασίας, όπως αυτή καθορίστηκε από την ΓΕΣ/ΔΕΠΛΗ, αναπτύχθηκε η εφαρμογή «Perseus», εφαρμογή σχεδιασμού και παρακολούθησης στρατιωτικών επιχειρήσεων σε πραγματικό χρόνο. Τα σημαντικότερα χαρακτηριστικά της είναι τα παρακάτω:

- Είναι πλήρως συμβατή με τις τρέχουσες εφαρμογές σχεδίασης και παρακολούθησης επιχειρήσεων του Στρατού Ξηράς.
- Μπορεί να λάβει δεδομένα από όλες τις εφαρμογές που ακολουθούν το πρότυπο ανταλλαγής δεδομένων του ΝΑΤΟ.
- Μπορεί να χρησιμοποιηθεί για εθνικές ή διεθνείς ασκήσεις, καθώς χρησιμοποιεί συμβατές στρατιωτικές συνθηματικές παραστάσεις.
- Έχει εύκολη χειρισμό που δεν απαιτεί τεχνικές γνώσεις, ενώ η υλοποίηση της διεπαφής χρήστη βρίσκεται κοντά στις μοντέρνες εφαρμογές του Στρατού.
- Είναι δημιουργημένη με αρχιτεκτονικές που επιτρέπουν εύκολη αναβάθμιση όπου και όταν απαιτηθεί.
- Παρέχει τα απαραίτητα δεδομένα στο πεδίο της μάχης.
- Αποτελεί ένα εργαλείο που διευκολύνει τον σχεδιασμό στην διάρκεια επιχειρήσεων.
- Παρέχει γρήγορο τρόπο ανταλλαγής σχεδίων ακόμα και σε μεγάλες αποστάσεις.

5.2 Μελλοντικές επεκτάσεις

Παρακάτω θα γίνει μια αναφορά στις αναμενόμενες επεκτάσεις – βελτιώσεις της εφαρμογής στο μέλλον:

- Υλοποίηση εναπομενόντων λειτουργιών δεδομένων.

- Δυνατότητα άμεσης ανταλλαγής δεδομένων μεταξύ συσκευών της ίδιας Μονάδας, χωρίς την ανάγκη χρήσης Βάσης Δεδομένων. Αυτά τα δεδομένα μπορούν να χρησιμοποιηθούν αντί για τα δεδομένα του Εξυπηρετητή.

- Δυνατότητα καταγραφής και ανταλλαγής στιγμάτων με τον προϊστάμενο, τις γειτονικές μονάδες και τους υφισταμένους.

- Εξιδεικευμένη λειτουργία καταγραφής και παρακολούθησης στόχων.

- Δυνατότητα λήψης προϊόντων επεξεργασμένων πληροφοριών.

- Διασύνδεση της εφαρμογής με λοιπές επιχειρησιακές εφαρμογές του Στρατού Ξηράς

- Μετάφραση στα Ελληνικά όλης της Εφαρμογής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Clausewitz, On War, https://www.gutenberg.org/files/1946/1946-h/1946-h.htm

[2] World War I, https://en.wikipedia.org/wiki/World_War_I

[3] World War II, https://en.wikipedia.org/wiki/World_War_II

[4] Technology during World War II, https://en.wikipedia.org/wiki/Technology_during_World_War_II

[5] How smartphone will reshape the Modern Battlefield, https://mwi.usma.edu/smartphones-will-reshape-modern-battlefield/

[6] Στρατιωτικός Κανονισμός 31-14, Δόγμα Πληροφοριών Στρατού Ξηράς

[7] Στρατιωτικός Κανονισμός 101-1, Η Σχεδίαση στον Στρατό Ξηράς

[8] Android (operating system), https://en.wikipedia.org/wiki/Android_(operating_system)

[9] Android Marshmallow, https://en.wikipedia.org/wiki/Android_Marshmallow

[10] Java (programming language), https://en.wikipedia.org/wiki/Java_(programming_language)

[11] SQL, https://en.wikipedia.org/wiki/SQL

[12] MariaDB, https://en.wikipedia.org/wiki/MariaDB

[13] NoSQL, https://en.wikipedia.org/wiki/NoSQL

[14] Couchbase Lite 2.1, https://en.wikipedia.org/wiki/Couchbase_Server

[15] Rest Model, https://www.codecademy.com/articles/what-is-rest

[16] JSON, https://en.wikipedia.org/wiki/JSON

[17] Retrofit, https://www.baeldung.com/retrofit

[18] EMP3 Android Development Kit, https://github.com/missioncommand/emp3-android

[19] Dagger 2, https://github.com/google/dagger

[20] RxJava 2, https://github.com/ReactiveX/RxJava

[21] Wildfly 11, https://en.wikipedia.org/wiki/WildFly

[22] Android Studio 3.1, https://en.wikipedia.org/wiki/Android_Studio

[23] IntelliJJ, https://en.wikipedia.org/wiki/IntelliJ_IDEA

[24] Github, https://en.wikipedia.org/wiki/GitHub