Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Σχεδίαση και Ανάπτυξη της Εφαρμογής MyNauticalMap και χρήση των GPS δεδομένων σε αυτή** **Design and Application Development of MyNauticalMap and use of GPS Data on it**
Ονοματεπώνυμο Φοιτητή	**Δημήτριος – Μαρίνος Πλευράκης**
Πατρώνυμο	**Παρθένιος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 14068**
Επιβλέπων	**Ιωάννης Θεοδωρίδης, Καθηγητής**

Ημερομηνία Παράδοσης **Μάϊος 2018**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ιωάννης Θεοδωρίδης Καθηγητής	Άγγελος Πικράκης Επίκουρος Καθηγητής	Νικόλαος Πελέκης Επίκουρος Καθηγητής

Ευχαριστίες

Στην προσπάθεια εκπόνησης της μεταπτυχιακής μου διατριβής συνέβαλαν ορισμένα άτομα, τα οποία με βοήθησαν να ολοκληρώσω αυτή τη διατριβή.

Αρχικά, θα ήθελα να ευχαριστήσω τον καθηγητή Γ. Θεοδωρίδη για την πολύτιμη καθοδήγησή και υποστήριξή του στην πορεία της μεταπτυχιακής μου διατριβής. Επίσης, μου κίνησε το ενδιαφέρον για τον ξεχωριστό κόσμο της Γεωπληροφορικής.

Επιπρόσθετα, ευχαριστώ τον υποψήφιο διδάκτορα Γ. Κοντούλη για την υποστήριξη που μου παρείχε καθ' όλη τη διάρκεια της μεταπτυχιακής μου διατριβής.

Τέλος, ευχαριστώ την κοπέλα μου και τους γονείς μου, που με βοήθησαν ψυχικά προκειμένου να ανταπεξέλθω στο δύσκολο έργο της διατριβής μου.

Πίνακας Περιεχομένων

Περίληψη

Αντικείμενο της παρούσας μεταπτυχιακής διατριβής είναι η ανάπτυξη μιας desktop χαρτογραφικής εφαρμογής με την ονομασία MyNauticalMap, στην οποία υποτυπώνονται χρήσιμες πληροφορίες πάνω σε ένα ναυτιλιακό χάρτη, συνδέεται και αξιοποιείται η τεχνολογία του GPS.

Ειδικότερα, η συγκεκριμένη εφαρμογή δομημένη με το Microsoft Visual Studio και τη βιβλιοθήκη DotSpatial, χρησιμοποιεί ελεύθερα χωρικά δεδομένα ναυτιλιακού περιεχομένου, ενώ παράλληλα παρέχεται στον χρήστη η δυνατότητα αποθήκευσης δεδομένων στο υπάρχον αρχείο shapefile ή σε νέο και της εισαγωγής δεδομένων τύπου εικόνας (π.χ. jpeg, png) σε μία βάση δεδομένων μέσω SQL Server, η οποία βρίσκεται εκτός Διαδικτύου. Τέλος, αξιοποιώντας την τεχνολογία GPS, ο χρήστης έχει τη δυνατότητα να επιβλέπει το στίγμα θέσης του, να εκτελεί καταγραφή και αποθήκευση σημαντικών GPS δεδομένων σε αρχεία shp και csv και να επιβλέπει συνολικά την καμπύλη πορείας της διαδρομής του.

Abstract

The subject of this postgraduate dissertation is the development of a desktop cartographic application called MyNauticalMap, in which useful information is depicted on a nautical map and GPS technology is connected and is exploited.

Particularly, this application developed with Microsoft Visual Studio and Dotspatial library, uses open source spatial data with maritime content, while being provided to the user the ability to store data in the existing shape file or in a new and importing image type data (e.g. jpeg, png) into an SQL database, which is offline. Finally, using GPS technology, the user can see his position, write and save important GPS data in shp and csv files and can see his route overall curve.

1. Εισαγωγή

1.1 Αντικείμενο της μεταπτυχιακής εργασίας

Η παρούσα μεταπτυχιακή διατριβή έχει ως αντικείμενο την ανάπτυξη μιας desktop χαρτογραφικής εφαρμογής, στην οποία θα υποτυπώνονται χρήσιμες πληροφορίες πάνω σε ένα ναυτιλιακό χάρτη και θα αξιοποιείται η τεχνολογία του GPS σε αυτή. Η εφαρμογή ονομάζεται MyNauticalMap και η ανάπτυξή της γίνεται μέσω της προγραμματιστικής πλατφόρμας Microsoft Visual Studio 2017 σε γλώσσα C# καθώς και της βιβλιοθήκης Dotspatial, η οποία συνδέεται άμεσα με το .Net Framework 4 της Microsoft.

Η συγκεκριμένη εφαρμογή χρησιμοποιεί ελεύθερα χωρικά δεδομένα ναυτιλιακού περιεχομένου, ενώ παράλληλα παρέχεται στον χρήστη η δυνατότητα αποθήκευσης δεδομένων σε ένα υπάρχον αρχείο shapefile ή σε νέο και της εισαγωγής δεδομένων τύπου εικόνας (π.χ. jpeg, png) σε μία βάση δεδομένων μέσω SQL Server, η οποία βρίσκεται εκτός Διαδικτύου. Επιπρόσθετα, αξιοποιώντας την τεχνολογία GPS, ο χρήστης έχει τη δυνατότητα να επιβλέπει το στίγμα θέσης του, να εκτελεί καταγραφή και αποθήκευση σημαντικών GPS δεδομένων σε αρχεία shp και csv και να επιβλέπει συνολικά την καμπύλη πορείας της διαδρομής του.

Κίνητρο για την παρούσα μεταπτυχιακή διατριβή αποτελεί η αξιοποίηση και η οργάνωση πληροφοριών ναυτιλιακής φύσεως του Πολεμικού μας Ναυτικού, για την οποία απαιτείται η δημιουργία μιας διαδραστικής εφαρμογής όπου θα περιλαμβάνει κατά το ελάχιστο τα κάτωθι:

- Ναυτιλιακές πληροφορίες λιμένων και λοιπών περιοχών.
- Τυχόν Φωτογραφικό υλικό.
- Βαθυμέτρηση θαλάσσης.

Στόχος της μεταπτυχιακής εργασίας είναι η ανάδειξη πρωτίστως όλων εκείνων των θεωρητικών εννοιών που αφορούν στα συστήματα γεωγραφικών πληροφοριών (GIS) και στο παγκόσμιο σύστημα στιγματοθέτησης (GPS), αλλά κυρίως είναι η υλοποίηση της χαρτογραφικής εφαρμογής MyNauticalMap, από την οποία ο τελικός χρήστης θα έχει τη δυνατότητα να προσθέτει, να τροποποιεί και να αφαιρεί γεωγραφικές πληροφορίες σε ένα ναυτιλιακό χάρτη. Σκοπός της εφαρμογής, είναι να αποτελέσει ένα ισχυρό βοήθημα ή οδηγό για τον εκάστοτε υπεύθυνο ναυτιλίας στη γέφυρα ενός πλοίου.

1.2 Δομή της μεταπτυχιακής εργασίας

Η δομή της μεταπτυχιακής εργασίας αποτελείται από εννιά κεφάλαια τα οποία αναλύονται παρακάτω:

Στο πρώτο κεφάλαιο, γίνεται η εισαγωγή της μεταπτυχιακής εργασίας. Συγκεκριμένα, παρουσιάζεται το αντικείμενο της εργασίας, οι στόχοι της, το κίνητρο για τη διεξαγωγή της και η δομή της.

Στο δεύτερο κεφάλαιο, περιγράφονται τα Γεωγραφικά Συστήματα Πληροφοριών (GIS) και αναλύεται το Παγκόσμιο σύστημα στιγματοθέτησης (GPS). Συγκεκριμένα για τα γεωγραφικά συστήματα πληροφοριών περιγράφονται οι δύο εφαρμογές που κυκλοφορούν στο Διαδίκτυο το QGIS και το ArcGIS. Επίσης, γίνεται αναφορά στους μορφοτύπους χωρικών δεδομένων και στα χωρικά συστήματα αναφοράς. Επιπρόσθετα, στην τεχνολογία του GPS αναλύεται το πρωτόκολλο επικοινωνίας NMEA-0183, παρατίθενται οι κύριοι ανταγωνιστές του GPS και γίνεται περιγραφή των γεωγραφικών συντεταγμένων και των τύπων αυτών.

Στο τρίτο κεφάλαιο, γίνεται αναφορά στα εργαλεία ανάπτυξης της εφαρμογής MyNauticalMap. Συγκεκριμένα, περιγράφονται το .Net Framework, το Visual Studio, ο SQL Server, το DotSpatial και η γλώσσα μοντελοποίησης UML, ενώ όσον αφορά τη μελλοντική επέκταση του MyNauticalMap, περιγράφεται η PostgreSQL και το PostGIS.

Στο τέταρτο κεφάλαιο, αναλύεται ο σχεδιασμός της εφαρμογής και η ανάλυση απαιτήσεων αυτής.

Στο πέμπτο κεφάλαιο, παρουσιάζεται η δόμηση της εφαρμογής ως προς το σχεδιαστικό και λειτουργικό της μέρος.

Στο έκτο κεφάλαιο, αναπτύσσεται η περιγραφή λειτουργικότητας της εφαρμογής, δηλαδή περιγράφονται αναλυτικά οι γενικές λειτουργίες της εφαρμογής, η λειτουργία του GPS στην εφαρμογή και οι λειτουργίες της καρτέλας των εικόνων της εφαρμογής.

Στο έβδομο κεφάλαιο, αναφέρονται οι μελλοντικές επεκτάσεις της εφαρμογής.

Στο όγδοο κεφάλαιο, γίνεται αναφορά στα συμπεράσματα της μεταπτυχιακής διατριβής.

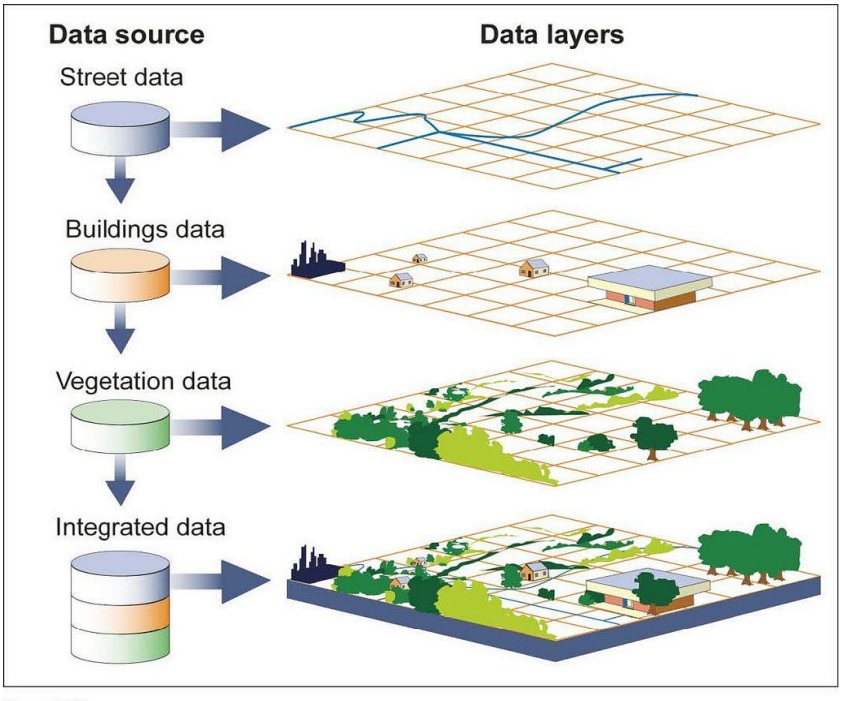
Στο ένατο κεφάλαιο, παρατίθενται οι βιβλιογραφικές αναφορές μέσω των οποίων αντλήθηκαν θεωρητικές και πρακτικές γνώσεις και που συνέβαλαν τόσο στην κατανόηση των διαφόρων θεωριών όσο και στην ανάπτυξη της συγκεκριμένης εφαρμογής.

Στο δέκατο κεφάλαιο, παρουσιάζεται ο κώδικας της εφαρμογής MyNauticalMap ως παράρτημα.

2. Γεωγραφικά συστήματα πληροφοριών (GIS) και Παγκόσμιο Σύστημα Στιγματοθέτησης (GPS)

2.1 Γεωγραφικά Συστήματα Πληροφοριών (GIS)

Το γεωγραφικό σύστημα πληροφοριών (Geographical Information System, GIS) είναι ένα σύστημα που διαχειρίζεται γεωγραφικές πληροφορίες, χωρικές και περιγραφικές. Έχει ως αντικείμενο τη διαχείριση, ανάλυση, απεικόνιση και τη διάχυση των γεωγραφικών πληροφοριών έτσι ώστε να επιλυθούν διάφορα προβλήματα και να ληφθούν αποφάσεις.



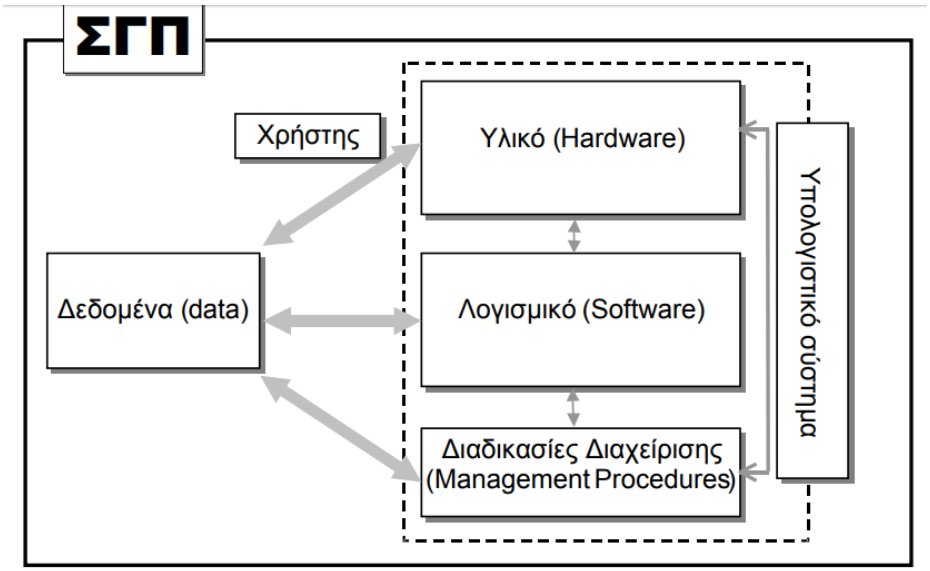
Εικόνα 1- Περιεχόμενο Γεωγραφικών Συστημάτων Πληροφοριών.

Ένα γεωγραφικό σύστημα πληροφοριών περιέχει τα κάτωθι:

- Τεχνικές για εισαγωγή γεωγραφικής πληροφορίας σε ψηφιακή μορφή
- Τεχνικές για αποθήκευση της πληροφορίας σε συμπιεσμένη μορφή
- Μεθόδους αυτοματοποιημένης ανάλυσης των γεωγραφικών δεδομένων
- Μεθόδους πρόβλεψης των αποτελεσμάτων των πιθανών σεναρίων
- Τεχνικές αναπαράστασης των δεδομένων σε μορφή χαρτών και εικόνων
- Δυνατότητα για έξοδο των αποτελεσμάτων σε μορφή αριθμών και πινάκων

Τα συστατικά ενός Γεωγραφικού συστήματος πληροφορίας είναι:

- Υλικό
- Λογισμικό
- Υπολογιστικό σύστημα
- Δεδομένα
- Χρήστες
- Διαδικασίες Διαχείρισης



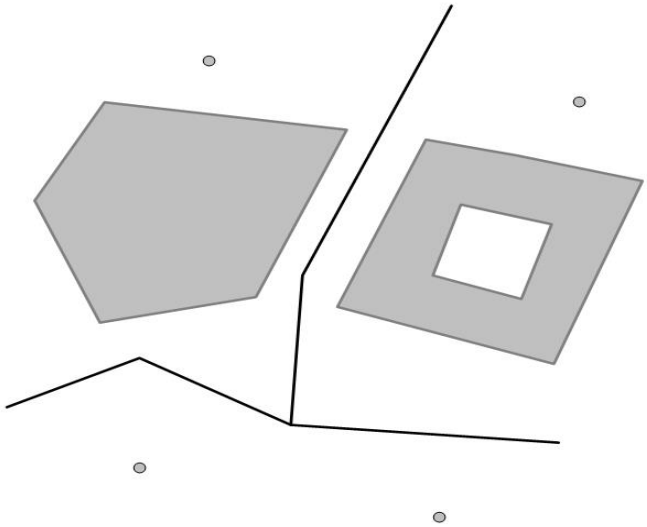
Εικόνα 2- Δομή ενός Γεωγραφικού Συστήματος Πληροφορίας.

Ορισμένα πλεονεκτήματα των ψηφιακών χαρτών που προέκυψαν από ένα Γεωγραφικό σύστημα πληροφορίας είναι:

- Δυναμικός Συμβολισμός σε ένα χάρτη
- Ευκολία και ακρίβεια στον εντοπισμό χωρικών δεδομένων
- Συνοδευτική περιγραφική πληροφορία σε συνδυασμό με τα χωρικά δεδομένα
- Εύκολη αποθήκευση και επαναφορά χαρτών

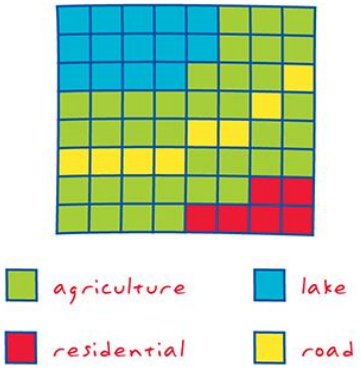
Όσον αφορά την αναπαράσταση των χωρικών δεδομένων στα γεωγραφικά συστήματα πληροφοριών, αυτά διακρίνονται σε δύο βασικές δομές: τη διανυσματική και την ψηφιδωτή μορφή.

Διανυσματική μορφή (vector): Τα δεδομένα σε αυτή τη μορφή μπορούν να αναπαρασταθούν με τρεις βασικές γεωμετρίες: τα σημεία (points), τις γραμμές (lines) και τα πολύγωνα (polygons). Σημείο μπορεί να θεωρηθεί η απεικόνιση μίας πόλης σε ένα χάρτη, γραμμή το οδικό δίκτυο μιας χώρας και πολύγωνο μια έκταση ενός οικοπέδου.



Εικόνα 3- Οι βασικές γεωμετρίες των διανυσματικών δεδομένων.

Ψηφιδωτή μορφή (raster): Αυτή η μορφή χρησιμοποιείται περισσότερο στην αναπαράσταση συνεχών δεδομένων του χώρου (π.χ. θερμοκρασία, πυκνότητα). Η εικόνα ενός χάρτη κατανέμεται σε κελιά και κάθε χαρακτηριστικό παίρνει μια τιμή που αντιστοιχεί στο χαρακτηριστικό αυτό.

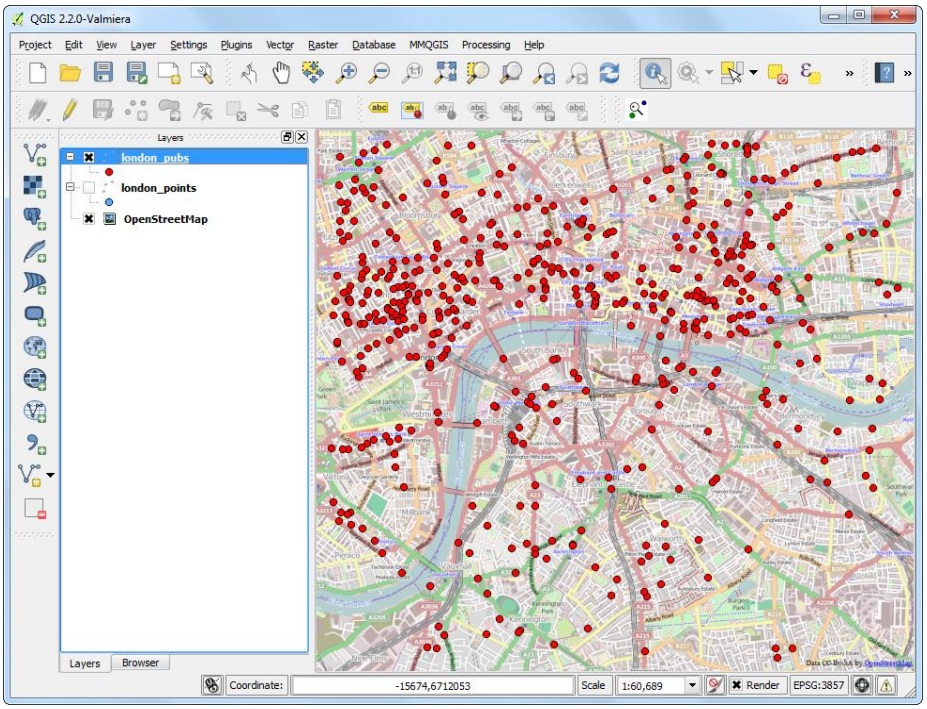


Εικόνα 4- Ψηφιδωτή μορφή ενός χάρτη.

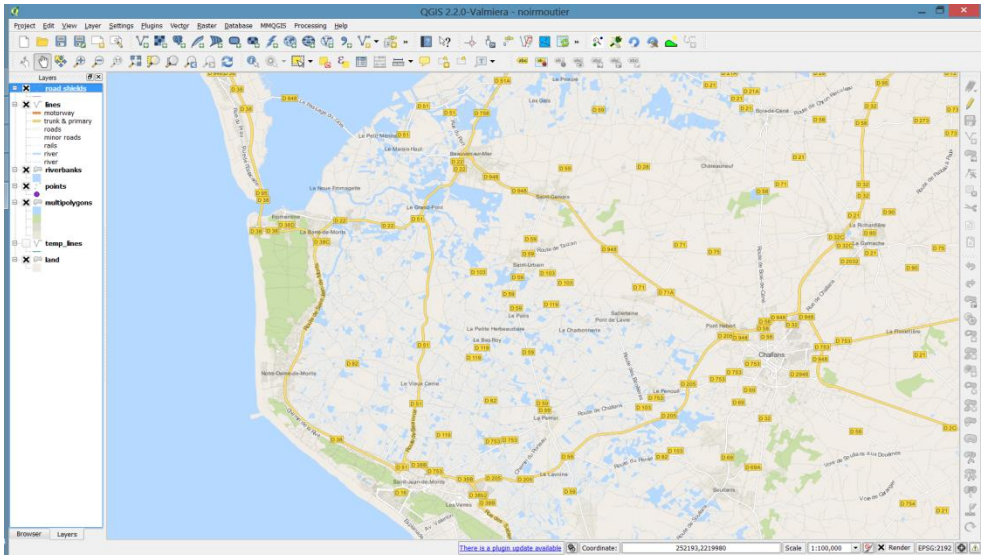
2.1.1 Το QGIS και το ArcGIS

Το QGIS πρωτοεμφανίστηκε το 2002 ως Quantum GIS και είναι ευρέως γνωστή ανοιχτού λογισμικού εφαρμογή που έχει ως αντικείμενο την προβολή, την επεξεργασία και την ανάλυση χωρικών δεδομένων. Το QGIS είναι μια ‘desktop’ εφαρμογή η οποία είναι συμβατή με διάφορα λειτουργικά συστήματα, όπως τα Windows, τα Linux και τα macOS ενώ έχει αναπτυχθεί με γλώσσα C++ χρησιμοποιώντας μια βιβλιοθήκη που ονομάζεται Qt. Όσον αφορά τη λειτουργικότητά αυτού του γεωγραφικού συστήματος πληροφοριών, υποστηρίζει και τα δύο μοντέλα των χωρικών δεδομένων, διανυσματικά (vector) και ψηφιδωτά (raster). Το QGIS επιτρέπει στους χρήστες να επεξεργαστούν χωρικά δεδομένα, να διαχειριστούν ψηφιακούς χάρτες και να τους εξάγουν ως εκτύπωση ή ως φωτογραφία. Επίσης, το QGIS, είναι ικανό να ενσωματώσει διάφορα plugins πχ το ‘OpenStreetMap’ και το ‘Google Maps’. Τέλος, το QGIS συνδέεται με διάφορα σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων, όπως η

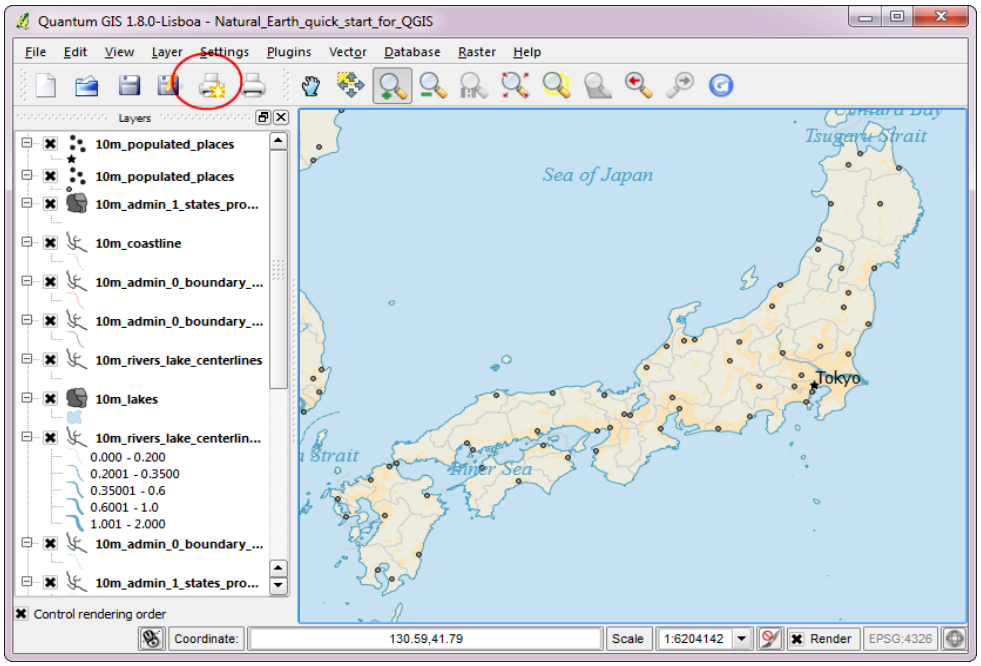
PostgreSQL και η επέκτασή της για χωρικά δεδομένα PostGIS, το Spatial Lite και η MySQL. Μερικές εικόνες από τη λειτουργικότητα του προγράμματος αυτού απεικονίζονται παρακάτω:



Εικόνα 5- Επεξεργασία OpenStreetMap χάρτη στο QGIS.



Εικόνα 6- Προβολή Google Map στο QGIS.



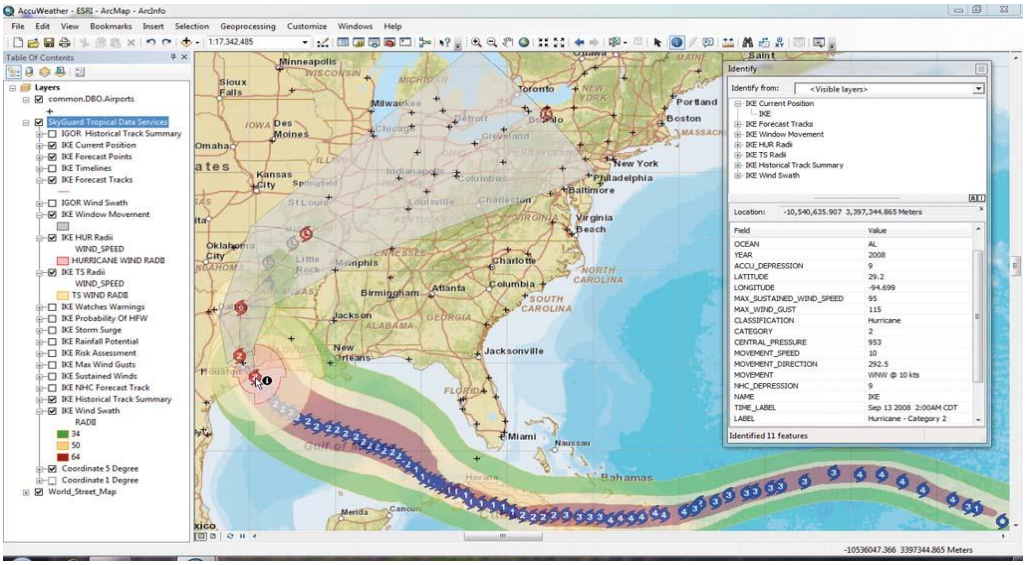
Εικόνα 7- Δημιουργία χάρτη στην εφαρμογή QGIS.

Το ArcGIS αντίστοιχα με το QGIS αλλά εμπορικό είναι ένα γεωγραφικό σύστημα πληροφοριών (GIS) το οποίο δημιουργεί χάρτες και επεξεργάζεται χωρικά δεδομένα. Δημιουργήθηκε το 1999 από την εταιρεία Esri, απευθύνεται κυρίως σε Microsoft Windows λειτουργικά συστήματα και αποτελεί μια ολοκληρωμένη συλλογή από προϊόντα λογισμικού GIS. Συγκεκριμένα, αποτελείται από τις κάτωθι εφαρμογές:

- ArcGIS Desktop
- ArcGIS Explorer
- ArcReader
- ArcGIS Workflow Manager
- ArcMap

Εικόνα 8- Οι εφαρμογές της σουίτας λογισμικού ArcGIS.

Ειδικότερα για το λογισμικό ArcGIS Desktop, υπάρχουν τρεις εκδόσεις το ArcView, το ArcEditor και το ArcInfo. Το ArcView, είναι το πρώτο επίπεδο της εφαρμογής ArcGIS Desktop και επιτρέπει στον χρήστη να διαβάζει χωρικά δεδομένα, να δημιουργεί χάρτες και να διεξάγει βασικές χωρικές αναλύσεις. Το ArcEditor, δίνει μεγαλύτερες δυνατότητες στον χρήστη σε σχέση με το ArcView, όπως τον χειρισμό shapefile αρχείων και χωρικών βάσεων δεδομένων. Τέλος, το ArcInfo περιλαμβάνει ακόμη περισσότερες δυνατότητες σε σχέση με τις δύο άλλες εκδόσεις για χειρισμό, ανάλυση και επεξεργασία χωρικών δεδομένων.



Εικόνα 9- Άποψη της εφαρμογής ArcGIS Desktop.

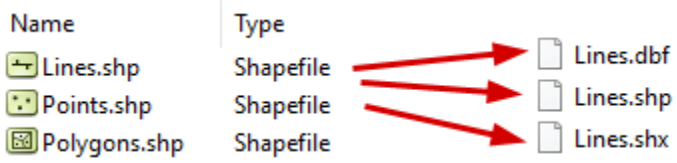
2.1.2 Μορφότυποι χωρικών δεδομένων (Geospatial File Formats)

Ανάλογα το μοντέλο των χωρικών δεδομένων, δηλαδή διανυσματικό (vector) ή ψηφιδωτό (raster) υπάρχει ένα μεγάλο εύρος μορφοτύπων (geospatial file formats) που εξυπηρετούν διάφορες συγκεκριμένες εφαρμογές στο Διαδίκτυο.

Για τα διανυσματικά μοντέλα χωρικών δεδομένων, οι πιο διαδεδομένοι μορφότυποι είναι οι εξής:

SHP (Shapefile): Ο μορφότυπος αυτός δημιουργήθηκε από την εταιρεία Esri και είναι ο πιο διαδεδομένος για τα διανυσματικά χωρικά δεδομένα. Υποστηρίζεται από όλες τις εμπορικές και ανοιχτού κώδικα εφαρμογές. Ένα αρχείο shapefile περιέχει υποχρεωτικά τρείς μορφότυπους:

- .shp Το είδος της διανυσματικής γεωμετρίας (σημείο, γραμμή ή πολύγωνο)
- .shx Ο δείκτης της θέσης της διανυσματικής γεωμετρίας
- .dbf Τα attribute δεδομένα



Εικόνα 10- Η δομή ενός αρχείου shapefile.

Προαιρετικά, το αρχείο shapefile μπορεί να περιέχει και άλλους μορφότυπους επιπλέον των τριών υποχρεωτικών όπως:

- .prj Τα μεταδεδομένα (metadata) του συστήματος προβολής

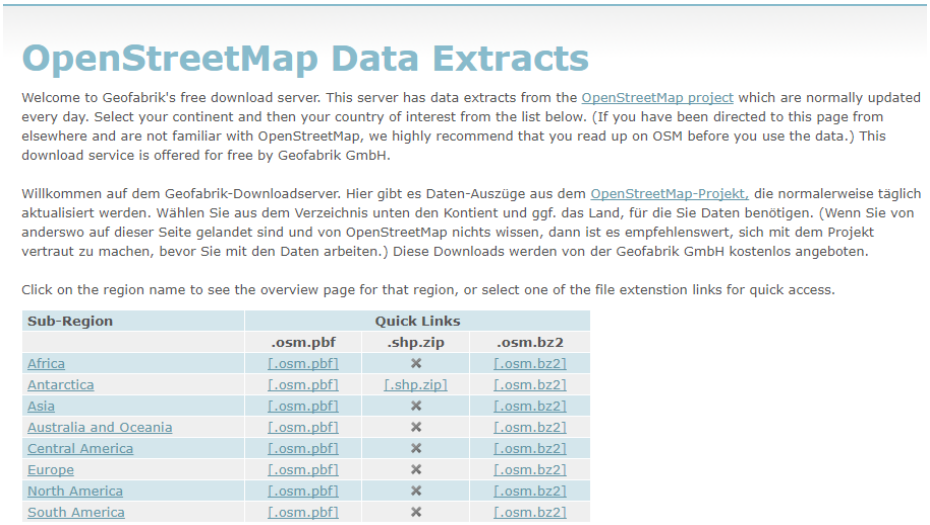
- .shp.xml Χωρικά μεταδεδομένα σε xml μορφή
- .sbn Χωρικός δείκτης για τη βελτιστοποίηση των ερωτημάτων
- .sbx Χωρικός δείκτης που βοηθά στην φόρτωση του αρχείου σε ένα GIS πρόγραμμα.

OSM (Openstreetmap): Το Openstreetmap είναι ένας χάρτης που δημιουργήθηκε από εθελοντές που συλλέγουν χωρικά δεδομένα μέσω αεροφωτογραφιών, συσκευών GPS και τοπικών χαρτών χαμηλής τεχνολογίας. Τα δεδομένα αυτά είναι ελεύθερα για να χρησιμοποιηθούν βάσει της άδειας Open Database License (ODbL). Τα .osm αρχεία, τα οποία είναι βασισμένα σε XML μορφή, είναι αυτά που παράγονται από την εφαρμογή Openstreetmap και περιέχουν χωρική πληροφορία. Εναλλακτική μορφή της xml του .osm αρχείου είναι το μορφότυπο .pbf, το οποίο είναι πιο αποδοτικό και πιο μικρό σε μέγεθος σε σχέση με το xml. Τέλος, υπάρχει και το μορφότυπο osm.bz2 το οποίο είναι η συμπιεσμένη μορφή του xml osm αρχείου.

example.osm

```
<?xml version='1.0' encoding='UTF-8'?>
<osm version='0.6' upload='true' generator='JOSM'>
  <node id='-5' action='modify' visible='true' lat='0.012935739981419907'
lon='-0.001167809869355378'>
    <tag k='name' v='Jeff&apos;s Clothes' />
    <tag k='shop' v='clothes' />
  </node>
</osm>
```

Εικόνα 11- Η XML μορφή ενός .osm αρχείου.



OpenStreetMap Data Extracts

Welcome to Geofabrik's free download server. This server has data extracts from the OpenStreetMap project which are normally updated every day. Select your continent and then your country of interest from the list below. (If you have been directed to this page from elsewhere and are not familiar with OpenStreetMap, we highly recommend that you read up on OSM before you use the data.) This download service is offered for free by Geofabrik GmbH.

Willkommen auf dem Geofabrik-Downloadserver. Hier gibt es Daten-Auszüge aus dem OpenStreetMap-Projekt, die normalerweise täglich aktualisiert werden. Wählen Sie aus dem Verzeichnis unten den Kontient und ggf. das Land, für die Sie Daten benötigen. (Wenn Sie von anderswo auf dieser Seite gelandet sind und von OpenStreetMap nichts wissen, dann ist es empfehlenswert, sich mit dem Projekt vertraut zu machen, bevor Sie mit den Daten arbeiten.) Diese Downloads werden von der Geofabrik GmbH kostenlos angeboten.

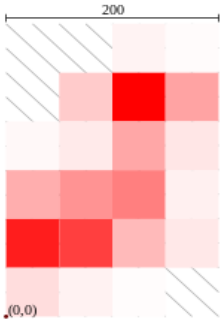
Click on the region name to see the overview page for that region, or select one of the file extenstion links for quick access.

Sub-Region	Quick Links		
	.osm.pbf	.shp.zip	.osm.bz2
Africa	[.osm.pbf]	×	[.osm.bz2]
Antarctica	[.osm.pbf]	[.shp.zip]	[.osm.bz2]
Asia	[.osm.pbf]	×	[.osm.bz2]
Australia and Oceania	[.osm.pbf]	×	[.osm.bz2]
Central America	[.osm.pbf]	×	[.osm.bz2]
Europe	[.osm.pbf]	×	[.osm.bz2]
North America	[.osm.pbf]	×	[.osm.bz2]
South America	[.osm.pbf]	×	[.osm.bz2]

Εικόνα 12- Οι τρείς μορφότυποι osm αρχείων για κατέβασμα από την ιστοσελίδα Geofabrik.

Για τα ψηφιδωτά μοντέλα χωρικών δεδομένων, ένας διαδεδομένος μορφότυπος είναι ο Esri Grid. Δημιουργήθηκε από την εταιρεία Esri και περιέχει δύο μορφές, μία ιδιόκτητη δυαδική γνωστή ως ARC/INFO GRID, ARC GRID και άλλων πολλών παραλλαγών και μία μη-ιδιόκτητη ASCII μορφή γνωστή ως ARC/INFO ASCII GRID. Η δυαδική μορφή χρησιμοποιείται κυρίως στην εμπορική εφαρμογή ARCGIS, ενώ η ASCII μορφή για εξαγωγή αρχείου.

Grid image



Values

	25	75	125	175
275	NA	NA	5	2
225	NA	20	100	36
175	3	8	35	10
125	32	42	50	6
75	88	75	27	9
25	13	5	1	NA

ASCII grid format

```
ncols         4
nrows         6
xllcorner     0.0
yllcorner     0.0
cellsize      50.0
NODATA_value  -9999
-9999 -9999 5 2
-9999 20 100 36
3 8 35 10
32 42 50 6
88 75 27 9
13 5 1 -9999
```

Εικόνα 13- Η ASCII μορφή ενός Esri Grid αρχείου.

2.1.3 Χωρικά Συστήματα Αναφοράς

Το χωρικό σύστημα αναφοράς ή spatial reference system (SRS) μπορεί να είναι ένα τοπικό ή περιφερειακό ή παγκόσμιο σύστημα, το οποίο βασίζεται στις συντεταγμένες και χρησιμοποιείται για τον εντοπισμό χωρικών οντοτήτων. Τα χωρικά συστήματα αναφοράς, αναφέρονται με έναν ακέραιο SRID αριθμό, συμπεριλαμβανομένων και των EPSG κωδικών που ορίζονται από τη Διεθνή Ένωση παραγωγών πετρελαίου και αερίου. Υπάρχουν χιλιάδες συστήματα διαθέσιμα εκ των οποίων αξιοσημείωτα είναι αυτά με τους κωδικούς ESPG: 4326, 2100 και 3857.

Το παγκόσμιο σύστημα αναφοράς (WGS84) με κωδικό EPSG: 4326 καλύπτει τα όρια (-180.0000, -90.0000, 180.0000, 90.0000) και χρησιμοποιείται από το δορυφορικό σύστημα πλοήγησης GPS και για τη γεωδετική χωρομέτρηση του NATO.

Το ελληνικό σύστημα αναφοράς (GGRS87) με κωδικό EPSG: 2100 καλύπτει τα όρια (18.2700, 33.2300, 29.9700, 41.7700).

Το σφαιρικό καρτεσιανό σύστημα αναφοράς με κωδικό EPSG: 3857 χρησιμοποιείται ευρέως διαδικτυακά από το Google Map και από το OpenStreetMap.

2.2 Παγκόσμιο Σύστημα Στιγματοθέτησης (GPS)

Το GPS (Global Positioning System) παγκόσμιο σύστημα στιγματοθέτησης είναι ένα παγκόσμιο σύστημα εντοπισμού γεωγραφικής θέσης ενός ακίνητου ή κινητού χρήστη το οποίο αποτελείται από είκοσι τέσσερις δορυφόρους, εφοδιασμένους με ειδικούς πομποδέκτες για τον υπολογισμό της θέσης του χρήστη. Οι πομποδέκτες αυτοί παρέχουν ακριβείς πληροφορίες σχετικές με τη θέση, το υψόμετρο, την κατεύθυνση και την ταχύτητα ενός σημείου.

Το GPS πρωτοξεκίνησε από το Υπουργείο Άμυνας των ΗΠΑ και ονομάστηκε NAVSTAR GPS (Navigation Signal Timing and Ranging Global Positioning System). Η ρύθμιση του συστήματος GPS πραγματοποιείται καθημερινά από τη βάση της Πολεμικής Αεροπορίας Σρίβερ.

Το GPS αποτελεί ένα παγκόσμιο δίκτυο το οποίο καλύπτει ξηρά, θάλασσα και αέρα. Διαχωρίζεται σε τρία τμήματα τα οποία είναι:

- Διαστημικό τμήμα
- Επίγειο τμήμα ελέγχου
- Τμήμα τελικού χρήστη

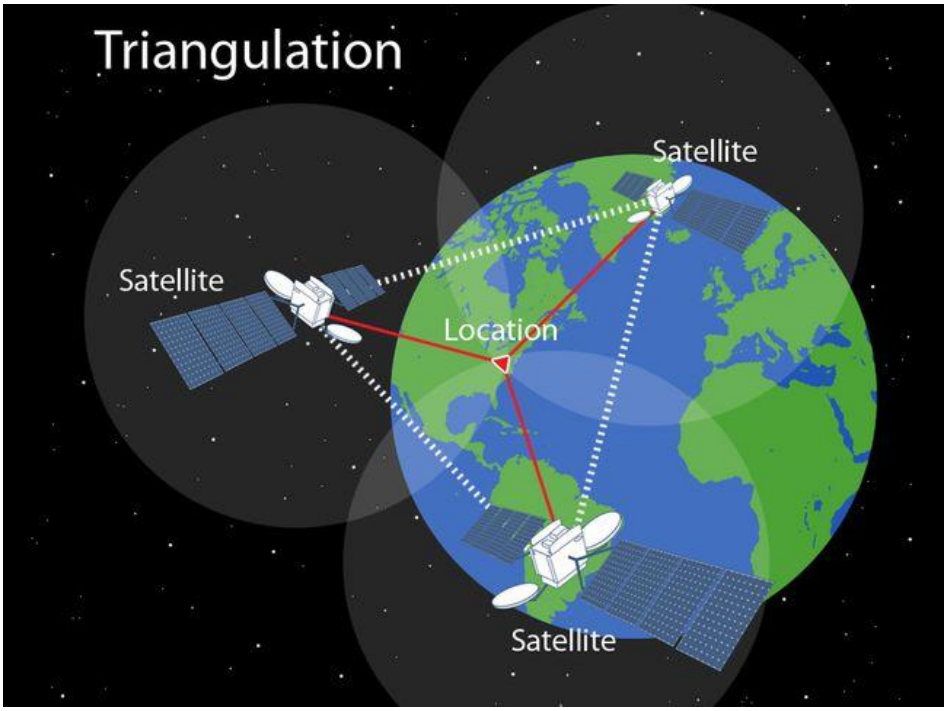
Το διαστημικό τμήμα αποτελείται από το δίκτυο των 24-32 δορυφόρων, οι οποίοι καλύπτουν ομοιόμορφα με το σήμα τους ολόκληρο τον πλανήτη. Αυτοί οι δορυφόροι βρίσκονται σε ύψος 20.200 χιλιομέτρων πάνω από την επιφάνεια της θάλασσας και εκτελούν δύο περιστροφές γύρω από τη Γη κάθε 24ωρο.

Το επίγειο τμήμα ελέγχου αποτελείται από πέντε σταθμούς με κυριότερο αυτό του Κολοράντο, οι οποίοι ελέγχουν τους δορυφόρους ως προς την ταχύτητα, το υψόμετρο και την

επάρκεια σε ηλιακή ενέργεια. Επιπρόσθετα, εφαρμόζουν όλες τις διορθωτικές ενέργειες που αφορούν στο σύστημα χρονομέτρησης των δορυφόρων, ώστε να αποτρέπεται η παροχή λανθασμένων πληροφοριών στους χρήστες του συστήματος.

Το τμήμα τελικού χρήστη αποτελείται από όλους αυτούς τους δέκτες, οι οποίοι λαμβάνουν την πληροφορία του στίγματος από τους δορυφόρους και τη μετατρέπουν σε μορφή κατανοητή προς τον άνθρωπο που μπορεί να αναπαρασταθεί μέσω μιας γεωγραφικής εφαρμογής. Δέκτες GPS αποτελούν συσκευές που χρησιμοποιούνται στη ναυτιλία καθώς και όλα τα έξυπνα κινητά τηλέφωνα (smartphones).

Για την επικοινωνία των GPS δεκτών (GPS Receivers) με τους ηλεκτρονικούς υπολογιστές, χρησιμοποιείται ένα πρωτόκολλο επικοινωνίας το οποίο ονομάζεται NMEA 0183. Οι δέκτες GPS παίρνοντας τις πληροφορίες θέσης του σημείου τους από τους δορυφόρους, μπορούν να αποστείλουν σε κάποια άλλη ναυτιλιακή συσκευή ή σε ένα ηλεκτρονικό υπολογιστή μέσω του πρωτοκόλλου NMEA 0183 τις πληροφορίες αυτές, οι οποίες μπορούν να επεξεργαστούν έτσι ώστε η τελική πληροφορία να είναι κατανοητή για τον άνθρωπο.



Εικόνα 14- Αναπαράσταση εύρεσης σημείου στη Γη από τους δορυφόρους.

2.2.1 Το πρωτόκολλο επικοινωνίας NMEA-0183

Το πρωτόκολλο επικοινωνίας NMEA-0183 ορίστηκε και ελέγχεται από τον μη κερδοσκοπικό οργανισμό NMEA (National Marine Electronics Association), το οποίο εμφανίστηκε τον Μάρτιο του 1983. Το NMEA-0183 είναι ένα πρωτόκολλο επικοινωνίας μεταξύ των ναυτιλιακών ηλεκτρονικών συσκευών (π.χ. γυροπυξίδων, ανεμομέτρων, σόναρ, δεκτών GPS), στο οποίο υπάρχει μεταφορά δεδομένων. Προγενέστερα πρωτόκολλα επικοινωνίας του NMEA-0183 ήταν τα NMEA-0180 και 0182, τα οποία χρησιμοποιούνται σήμερα πολύ περιορισμένα για συγκεκριμένες ναυτιλιακές συσκευές. Το NMEA-2000 είναι η πιο εξελιγμένη έκδοση του NMEA-0183 και η αρχιτεκτονική του βασίζεται στη σύνδεση πολλών ναυτιλιακών κυρίως συσκευών σε ένα κεντρικό καλώδιο γνωστό ως backbone. Το πρωτόκολλο αυτό είναι plug and play δηλαδή

επιτρέπει στις ηλεκτρονικές συσκευές διαφορετικού κατασκευαστή που είναι συνδεδεμένες στο δίκτυο του NMEA-2000 να επικοινωνούν και να μοιράζουν δεδομένα μεταξύ τους.

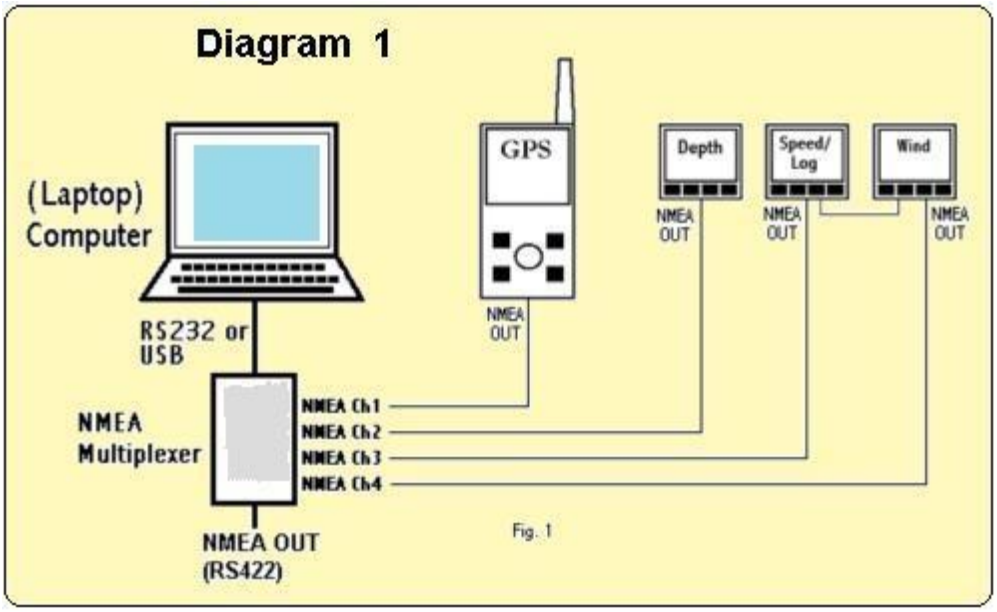
	NMEA 2000	NMEA 0183
Connector	Standard connectors (Plug and play)	Different connectors of each manufacturer
Data rate	250k bits/second	4.8k (38.4k) bits/second
Protocol	Compact binary message	ASC II serial communication
	Multi-talker, multi-listener	Single-talker, multi-listener
	Network	Serial communication (Point to point communication)

Εικόνα 15- Διαφορές μεταξύ πρωτοκόλλων NMEA-2000 και 0183.

Οι ηλεκτρονικές συσκευές που υποστηρίζουν το πρωτόκολλο NMEA-0183 ορίζονται ως ομιλητές (talkers) ή ακροατές (listeners) και χρησιμοποιούν ασύγχρονη σειριακή διεπαφή με τις ακόλουθες παραμέτρους:

- Ρυθμός Μετάδοσης (Baud Rate): 4800
- Αριθμός Δεδομένων σε bits: 8
- Stop bits: 1 (ή περισσότερα)
- Parity: Καμία
- Handshake: Κανένα

Το πρωτόκολλο NMEA-0183 επιτρέπει σε ένα ομιλητή (talker) και πολλούς ακροατές (listeners) να επικοινωνούν μεταξύ τους. Παρακάτω απεικονίζεται η λήψη GPS δεδομένων από ένα δέκτη GPS και η μεταφορά των δεδομένων αυτών μέσω του πρωτοκόλλου NMEA-0183 σε ένα φορητό υπολογιστή.



Εικόνα 16- Αναπαράσταση μεταφοράς δεδομένων μέσω του NMEA-0183.

Η μεταφορά του πρωτοκόλλου NMEA-0183 μπορεί να εκτελεσθεί από διάφορες ηλεκτρονικές συσκευές. Η ηλεκτρονική συσκευή που θα χρησιμοποιηθεί ως ομιλητής (talker) έχει ένα αναγνωριστικό (ID) για τον ακροατή (listener) το οποίο ονομάζεται Talker ID. Π.χ. το talker id ενός GPS δέκτη είναι η λέξη $GP.

Για τη μεταφορά δεδομένων που αφορά GPS δέκτες, χρησιμοποιούνται διάφορα μηνύματα τα οποία έχουν ξεχωριστούς κωδικούς. Τα πιο ευρέως γνωστά NMEA μηνύματα που χρησιμοποιούνται από GPS δέκτες είναι τα κάτωθι:

NMEA Message	UTC date/time	Position	Course	Speed
RMC	+	+	+	+
GGA	+	+		
GLL	+	+		
ZDA	+			
GNS	+	+		
HDT,HDG,HMR			+	
VBW,VHW,VTG			+	+
BEC,BWC,BWR			+	

Εικόνα 17- NMEA μηνύματα που χρησιμοποιούνται από GPS δέκτες.

Τα NMEA μηνύματα ακολουθούν τους εξής κανόνες:

- Κάθε πρόταση ξεκινά με το σύμβολο $ και τελειώνει με το CRLF (CR: Carriage Return, LF: Line Feed).
- Όλα τα δεδομένα μεταδίδονται σε μορφή προτάσεων.
- Επιτρέπονται μόνο ASCII χαρακτήρες.

Υπάρχουν τρία βασικά είδη προτάσεων:

- Talker Sentences
- Proprietary Sentences
- Query Sentences

Η γενική σύνταξη των talker sentences είναι η εξής:

$ttsss, d1, d2, …..<CR><LF> όπου η πρόταση ξεκινά πάντα με το σύμβολο $, στη θέση των tt εισάγεται το talker id π.χ. GP, στη θέση των sss εισάγεται το NMEA μήνυμα π.χ. RMC. Ακολουθούν τα μοναδικά πεδία δεδομένων d1, d2 και η πρόταση ολοκληρώνεται με το <CR><LF>. Παρακάτω απεικονίζονται δύο γνωστά NMEA μηνύματα το RMC και το HDT:

RMC Recommended Minimum Navigation Information

```
                                                           12
       1         2 3       4 5        6 7   8   9    10  11|
       |         | |       | |        | |   |   |    |   | |
$--RMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,xxxx,x.x,a*hh

 1) Time (UTC)
 2) Status, V = Navigation receiver warning
 3) Latitude
 4) N or S
 5) Longitude
 6) E or W
 7) Speed over ground, knots
 8) Track made good, degrees true
 9) Date, ddmmyy
10) Magnetic Variation, degrees
11) E or W
12) Checksum
```

Εικόνα 18- Το RMC μήνυμα.

HDT Heading – True

```
       1   2 3
       |   | |
$--HDT,x.x,T*hh

1) Heading Degrees, true
2) T = True
3) Checksum
```

Εικόνα 19- Το HDT μήνυμα.

Το RMC μήνυμα (Recommended Minimum Navigation Information) κατέχει διάφορες σημαντικές ναυτιλιακές πληροφορίες, οι οποίες γνωστοποιούνται στον ακροατή (listener) και αυτές είναι ο χρόνος σε UTC ώρα, το στίγμα της θέσης ενός σημείου, η ταχύτητα σε knots κ.α.

Το HDT μήνυμα δίνει πληροφορίες σχετικά με την αληθή πορεία ενός σημείου στη Γη σε μοίρες.

Ένας GPS δέκτης λαμβάνει όλες εκείνες τις πληροφορίες που χρειάζεται από τους δορυφόρους και μέσω του πρωτοκόλλου NMEA-0183 μπορεί να τις αποστείλει σε ένα χρήστη ενός φορητού υπολογιστή. Τα μηνύματα που λαμβάνει ο χρήστης του φορητού υπολογιστή είναι πολλά και απεικονίζονται παρακάτω:

```
$GPGGA,225942,3803.043483,N,02344.746799,E,1,10,,146.4,M,3
8.6,M,,*6E
$GPVTG,,T,,M,0.0,N,0.0,K,A*23
$GPRMC,225942,A,3803.043483,N,02344.746799,E,0.0,,010218,
0.0,W,A*24
$GPGSA,A,3,10,25,,,,,,,,,,,4.8,2.5,4.2*39
$GNGSA,A,3,10,25,,,,,,,,,,,4.8,2.5,4.2*27
$GNGSA,A,3,71,85,,,,,,,,,,,4.8,2.5,4.2*2A
$BDGSA,A,3,214,,,,,,,,,,,,4.8,2.5,4.2*19
$GPGSV,3,1,10,10,21,181,25,15,04,092,00,16,40,312,00,20,41,06
4,00*7A
$GPGSV,3,2,10,21,74,012,00,25,10,151,20,26,65,275,00,27,15,29
5,00*73
$GPGSV,3,3,10,29,30,085,00,31,12,220,00*7C
```

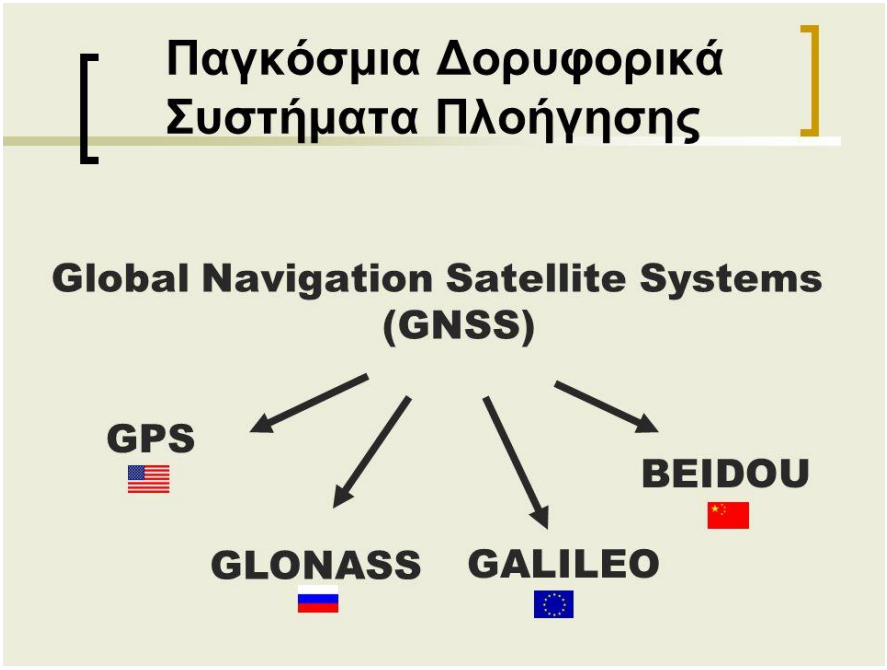
Εικόνα 20- Αποστολή μηνυμάτων σε χρήστη φορητού υπολογιστή μέσω NMEA-0183.

Από τα διάφορα μηνύματα που αποστέλλονται, ο τελικός χρήστης μπορεί να διαλέξει τα δεδομένα που χρειάζεται και να τα αξιοποιήσει αναλόγως.

2.2.2 Οι κύριοι ανταγωνιστές του GPS

Οι κύριοι ανταγωνιστές του GPS που εμφανίστηκαν αργότερα στον χρόνο είναι:

- Galileo
- Beidou
- Glonass



Εικόνα 21- Παγκόσμια δορυφορικά συστήματα πλοήγησης.

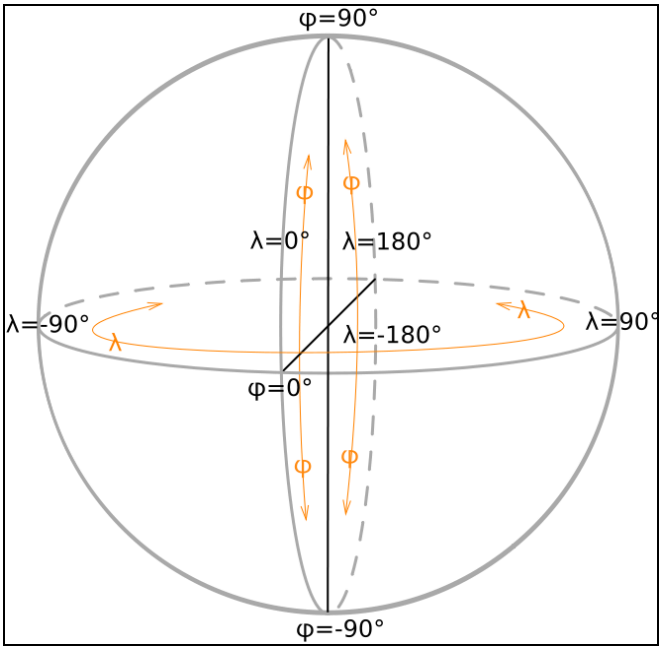
Το δορυφορικό σύστημα πλοήγησης Galileo είναι ένα ευρωπαϊκό σύστημα που δημιουργήθηκε με σκοπό την αποφυγή εξάρτησής του από το Αμερικανικό σύστημα GPS. Το σύστημα Galileo σκοπεύει να προσφέρει δωρεάν υπηρεσίες πλοήγησης και προορίζεται να παρέχει ακρίβεια εντοπισμού μια θέσης της τάξεως του ενός μέτρου. Ο εντοπισμός μια θέσης βορειότερα στην Γη, έχει μεγαλύτερη ακρίβεια σε αυτό το σύστημα.

Αντίστοιχα, το Beidou είναι ένα κινέζικο σύστημα το οποίο λειτουργεί με δεκαέξι δορυφόρους και εξυπηρετεί την περιοχή της Ασίας και του Ειρηνικού ωκεανού. Από δοκιμές που έχουν εκτελεσθεί, το Beidou έχει ακρίβεια στον εντοπισμό μιας θέσης μέσα στα πέντε μέτρα. Το Beidou εκτός από το να προσφέρει πληροφορίες μιας θέσης, είναι και ικανό να προσφέρει υπηρεσίες τηλεπικοινωνιών, όπου επιτρέπει στους χρήστες να επικοινωνούν μεταξύ τους με χρήση μηνυμάτων.

Τέλος, το σύστημα Glonass δημιουργήθηκε από τη Ρωσία, αρχικά για στρατιωτικό σκοπό που όμως αργότερα άλλαξε και πλέον χρησιμοποιείται για εμπορικούς σκοπούς και για τους πολίτες. Χρησιμοποιεί είκοσι τέσσερις δορυφόρους και προσφέρει καλύτερη ακρίβεια εντοπισμού θέσης στα βορειότερα της Γης.

2.2.3. Γεωγραφικές συντεταγμένες και οι τύποι αυτών

Γεωγραφικές συντεταγμένες είναι το σύστημα συντεταγμένων αποτελούμενο από δύο μεγέθη, αυτά του γεωγραφικού μήκους και πλάτους για τον εντοπισμό μιας θέσης στην επιφάνεια της Γης εκτός των πόλων. Βάση για τις γεωγραφικές συντεταγμένες αποτελούν ο Ισημερινός και ο πρώτος Μεσημβρινός.



Εικόνα 22- Οι γεωγραφικές συντεταγμένες στην σφαίρα της Γης.

Γεωγραφικό πλάτος (latitude) (φ) ενός σημείου που βρίσκεται στην επιφάνεια της γης είναι η γωνία που σχηματίζει η κατακόρυφος του τόπου με το επίπεδο του ισημερινού. Το γεωγραφικό πλάτος χαρακτηρίζεται Βόρειο Β (North N) ή Νότιο Ν (South S). Το γεωγραφικό πλάτος του ισημερινού είναι 0°. Η μέτρηση του γεωγραφικού πλάτους γίνεται σε μοίρες και υποδιαιρέσεις αυτών (πρώτα και δεύτερα) από 0°- 90° Β ή 0°- 90° Ν.

Γεωγραφικό μήκος (longitude) (λ) ενός σημείου στην επιφάνεια της γης είναι η στερεή γωνία που σχηματίζεται από το επίπεδο του μεσημβρινού που διέρχεται από το εν λόγω σημείο με το επίπεδο του πρώτου μεσημβρινού. Το γεωγραφικό μήκος χαρακτηρίζεται Ανατολικό Α (East E) ή Δυτικό Δ (West W). Το γεωγραφικό μήκος το πρώτου μεσημβρινού είναι 0°. Η μέτρηση του γεωγραφικού μήκους γίνεται σε μοίρες και υποδιαιρέσεις αυτών (πρώτα και δεύτερα) από 0°- 180° Α ή 0°- 180° Δ.

Οι μορφές στις οποίες οι γεωγραφικές συντεταγμένες μπορούν να αναπαρασταθούν είναι τρεις:

- Degrees, Minutes, Seconds (DMS)
- Degrees Decimal Minutes (DDM)
- Decimal Degrees (DD)

Η μορφή DMS σημαίνει ότι το γεωγραφικό μήκος ή πλάτος αναπαρίσταται σε μοίρες (°), λεπτά (′) και δευτερόλεπτα (″) π.χ. 40° 26′ 46″ N 79° 58′ 56″ W.

Η μορφή DDM σημαίνει ότι το γεωγραφικό μήκος ή πλάτος αναπαρίσταται σε μοίρες και δεκαδικά λεπτά π.χ. 40° 26.767′ N 79° 58.933′ W.

Η μορφή DD σημαίνει ότι το γεωγραφικό μήκος ή πλάτος αναπαρίσταται σε δεκαδικές μοίρες π.χ. 40.446° N 79.982° W.

Για τη μετατροπή από τη μορφή DMS σε DD λαμβάνουμε υπόψη ότι ένα λεπτό ισούται με εξήντα δευτερόλεπτα και 1 μοίρα ισούται με 1 ώρα δηλαδή 60 λεπτά ή 3600 δευτερόλεπτα. Για τον υπολογισμό των DD χρησιμοποιούμε τον τύπο: DD= d + (min/60) + (sec/3600) όπου DD: Δεκαδικές μοίρες, d: μοίρες, min: λεπτά και sec: δευτερόλεπτα.

Αντίθετα, για την μετατροπή από τη μορφή DD σε DMS, έχουμε τα εξής:

- D°= Int (D)
- M′= Int ((D - Int (D)) * 60)
- S″= (((D - Int (D)) * 60) - Int ((D - Int (D)) * 60)) * 60

Π.χ. το γεωγραφικό πλάτος των 45.781111 μοιρών, το οποίο είναι σε δεκαδικές μοίρες (decimal degrees), σε DMS μορφή ισχύει ότι Int (45.781111) και Int (45.781111 - Int (45.781111)) * 60 και (((45.781111 - Int (45.781111))* 60) - Int ((45.781111 - Int (45.781111)) * 60)) * 60 = 45° 46' 52".

Για τη μετατροπή από τη μορφή DMS σε DDM εφαρμόζουμε ότι DDM= D° και M+ (S/60) π.χ. 45 μοίρες 46 λεπτά και 53 δευτερόλεπτα μετατρέπονται 45 και 46 + (53/60)= 45° 46.86666'.

Για τη μετατροπή από DD σε DDM ισχύει ότι:

- D°= Int (D)
- M.M′= (D - Int (D)) * 60

Π.χ. το γεωγραφικό πλάτος των 45.781111 μοιρών μετατρέπεται σε Int (45.781111) και (45.781111 - Int (45.781111)) * 60 = 45° 46.86666'.

Στο κεφάλαιο 2.2.1 στο οποίο έγινε περιγραφή των NMEA μηνυμάτων, όσον αφορά τα δεδομένα που λαμβάνουμε από τους δορυφόρους του GPS και αφορούν το RMC μήνυμα, στο μήνυμα αυτό οι γεωγραφικές συντεταγμένες (latitude, longitude) είναι στη μορφή DDMM.mm και DDDMM.mm αντίστοιχα. Δηλαδή, οι γεωγραφικές συντεταγμένες βρίσκονται στη μορφή DDM που αναφέρθηκε πιο πάνω (μοίρες και δεκαδικά λεπτά). Αν επιθυμούμε να μετατρέψουμε τη μορφή αυτή των γεωγραφικών συντεταγμένων μας σε DD (δεκαδικές μοίρες), διότι είναι προτιμότερο για πολλές χαρτογραφικές εφαρμογές, τότε D°=D και .d=(DM/60) π.χ.45°46.86666' μετατρέπονται σε δεκαδικές μοίρες D=45°, .d=(46.86666/60)=0.781111°, DD= 45 + 0.781111 = 45.781111°.

Συνοψίζοντας, στον κάτωθι πίνακα απεικονίζονται οι μορφές των γεωγραφικών συντεταγμένων.

Πίνακας 1- Μορφές γεωγραφικών συντεταγμένων

Μορφή Γεωγρ. Συντ.	Περιγραφή	Παράδειγμα
DMS	Μοίρες, Λεπτά, Δευτερόλεπτα	45° 46' 52"
DDM	Μοίρες και δεκαδικά λεπτά	45° 46.86666'
DD	Δεκαδικές Μοίρες	45.781111°

3. Εργαλεία Ανάπτυξης και Υλοποίησης

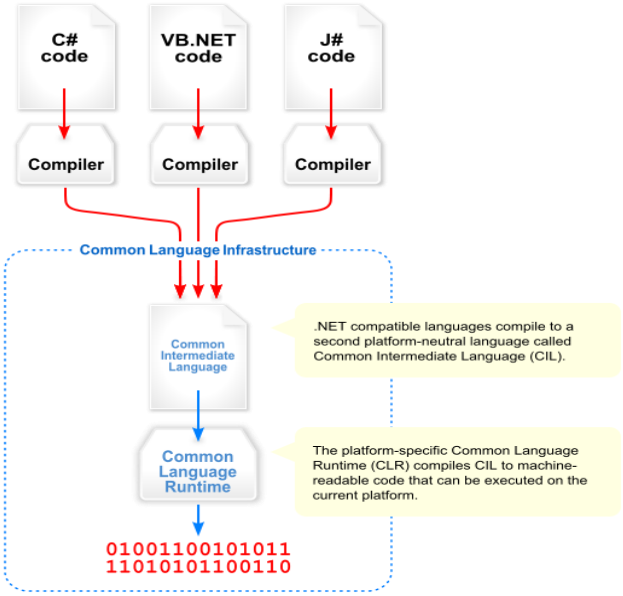
3.1. Το .Net Framework

Το .Net Framework αποτελεί πλατφόρμα ανάπτυξης λογισμικού για τα Microsoft Windows, το Διαδίκτυο, τα κινητά και τις υπηρεσίες cloud όπως το Microsoft Azure. Δημιουργήθηκε από την εταιρεία Microsoft και κυκλοφόρησε στην αγορά πρώτη φορά το 2002.

Εικόνα 23- Το λογότυπο του .Net Framework.

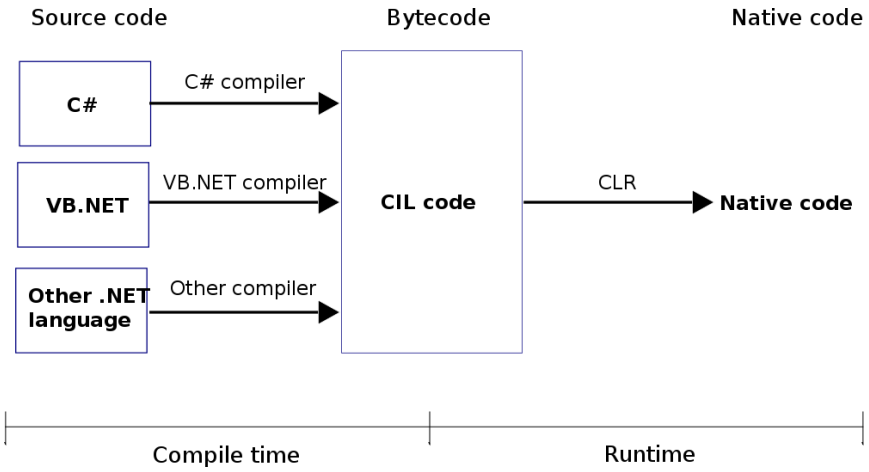
Η αρχιτεκτονική του .Net Framework αποτελείται από τα εξής στοιχεία:

Common Language Runtime (CLR)
Βιβλιοθήκες Class Libraries (FCL)
Γλώσσες Προγραμματισμού (C#, VC++, VB.NET, Jscript.NET)
ASP.NET



Εικόνα 24- Το Common Language Infrastructure.

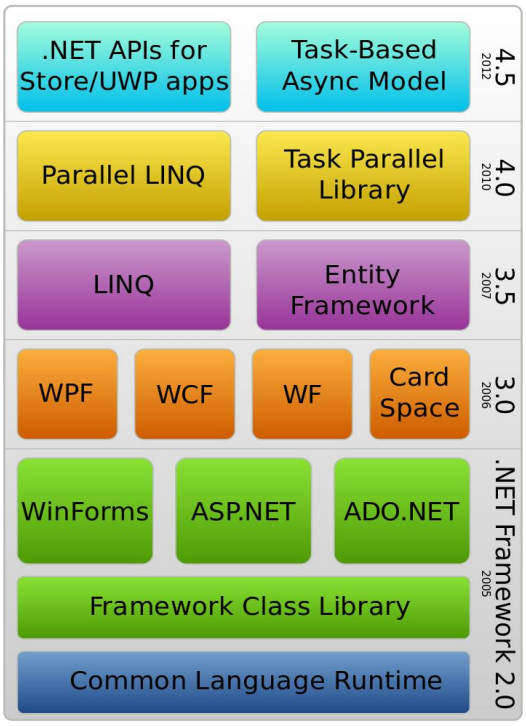
Το Common Language Runtime (CLR) είναι μια εικονική μηχανή (Virtual Machine, VM) η οποία διαχειρίζεται την εξαγωγή των .Net Framework προγραμμάτων. Ο εκάστοτε μεταφραστής (compiler) της γλώσσας προγραμματισμού που υποστηρίζεται από το .Net Framework μεταφράζει τον πηγαίο κώδικα και εξάγει ένα bytecode αρχείο ή αλλιώς το CIL κώδικα (Common Intermediate Language). Στη συνέχεια, ο CLR μεταφράζει το CIL κώδικα και εξάγει το εκτελέσιμο αρχείο. Μία αναπαράσταση της λειτουργίας του CLR απεικονίζεται παρακάτω:



Εικόνα 25- Μετατροπή πηγαίου κώδικα σε εκτελέσιμο μέσω του CLR.

Ο CLR παρέχει δυνατότητες όπως διαχείριση μνήμης, ασφάλεια, διαχείριση λαθών, διαχείριση νημάτων για κάθε γλώσσα προγραμματισμού που συνεργάζεται με το .NET Framework.

Το Framework Class Library είναι μια μεγάλη βιβλιοθήκη που περιέχει κώδικα για προγραμματιστικά θέματα όπως νήματα, εγγραφή/ανάγνωση αρχείων, υποστήριξη βάσεων δεδομένων, μετατροπή σε XML, δομές δεδομένων όπως στοίβες και ουρές κλπ. Η βιβλιοθήκη αυτή είναι διαθέσιμη σε κάθε γλώσσα που συνεργάζεται με το .NET Framework.



Εικόνα 26- Η στοίβα των στοιχείων της .Net Framework.

3.2. Το Visual Studio

Το Microsoft Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης προγραμμάτων (IDE) από τη Microsoft. Χρησιμοποιείται για την ανάπτυξη προγραμμάτων ηλεκτρονικών υπολογιστών για τα Microsoft Windows, για ιστοσελίδες, για διαδικτυακές εφαρμογές, για υπηρεσίες web και για εφαρμογές σε smartphones. Το Visual Studio χρησιμοποιεί πλατφόρμες ανάπτυξης λογισμικού της Microsoft, όπως Windows API, Windows Forms, Windows Presentation Foundation, Windows Store και το Microsoft Silverlight. Υποστηρίζει τριάντα έξι διαφορετικές γλώσσες προγραμματισμού από τις οποίες ενσωματωμένες στο IDE είναι οι C, C++, C++/CLI, Visual Basic .NET, C#, F#, Javascript, Typescript, XML, XSLT, HTML και CSS, ενώ οι Python, Ruby, Node.js, M και άλλες μπορούν να ενσωματωθούν στο IDE ως plug-in. Επίσης, το Visual Studio περιλαμβάνει επεξεργαστή κώδικα (code editor), ο οποίος υποστηρίζει:

- Επισήμανση σύνταξης και ολοκλήρωση κώδικα (syntax highlighting and code completion) χρησιμοποιώντας το IntelliSense.
- Τοποθέτηση σελιδοδεικτών στον κώδικα για γρηγορότερη και ευκολότερη αναζήτηση ενός σημείου σε αυτόν.
- Αναδρομολόγηση κώδικα (code refactoring).

Τέλος, περιλαμβάνει έναν debugger, ο οποίος λειτουργεί ταυτόχρονα ως source-level debugger και ως machine-level debugger.

Το πιο πρόσφατο σε έκδοση Visual Studio 2017 διανέμεται στις εξής κατηγορίες:

- Community
- Professional
- Enterprise

Η κατηγορία Community είναι δωρεάν για τον κάθε χρήστη που σκοπεύει να ασχοληθεί με τον προγραμματισμό σε .Net Framework περιβάλλον και είναι διαθέσιμη στο https://www.visualstudio.com/downloads/. Οι κατηγορίες Professional και Enterprise είναι επί πληρωμή και απευθύνονται κυρίως για επιχειρήσεις που θέλουν να προωθήσουν μια εμπορική εφαρμογή.

Εικόνα 27- Το λογότυπο του Microsoft Visual Studio 2017.

Ένα μεγάλο πλεονέκτημα για τον προγραμματιστή που θέλει να αναπτύξει μια παραθυρική εφαρμογή χρησιμοποιώντας το Visual Studio και μια βάση δεδομένων είναι ότι το Visual Studio ενσωματώνει πλήρως τον Microsoft SQL Server, όλο το περιβάλλον ανάπτυξης μιας βάσης δεδομένων. Επιπρόσθετα, στο Visual Studio μπορεί να προστεθεί η PostgreSQL, η MySQL, η Oracle, η SQLite και η Firebird. Ειδικότερα για την ανάπτυξη μιας χαρτογραφικής εφαρμογής, η PostgreSQL μέσω του extension PostGIS μαζί με το Visual Studio είναι δύο δυνατά εργαλεία που μπορούν να δώσουν πολλές δυνατότητες σε ένα προγραμματιστή.

3.3. SQL Server

Ο SQL Server της Microsoft είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (DBMS), το οποίο πρωτοκυκλοφόρησε στην αγορά το 1989 σε συνεργασία με την Sybase. Οι κύριες γλώσσες που χρησιμοποιούνται στον SQL Server είναι η T-SQL και η ANSI SQL. Η πιο πρόσφατη έκδοση του SQL Server είναι αυτή του 2017. Ο SQL Server 2017 κυκλοφορεί στις εξής κατηγορίες:

- Enterprise
- Standard
- Developer
- Web

- Express

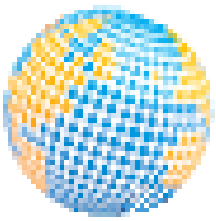
Οι κατηγορίες Developer και Express διανέμονται δωρεάν στον χρήστη ενώ οι υπόλοιπες είναι επί πληρωμή. Το θετικό στην έκδοση του 2017 είναι ότι ο SQL Server μπορεί να χρησιμοποιηθεί εκτός από Microsoft Windows λειτουργικά συστήματα και σε Linux.

3.4. Το DotSpatial

Το DotSpatial είναι μία δωρεάν βιβλιοθήκη γεωγραφικών συστημάτων πληροφοριών (GIS), η οποία εφαρμόζεται στο Microsoft .Net Framework 4. Επιτρέπει στους προγραμματιστές να ενσωματώνουν χωρικά δεδομένα, να αναλύουν και να επεξεργάζονται χάρτες. Διατίθεται μέσω της διαδικτυακής σελίδας GitHub και η πιο πρόσφατη έκδοση είναι η 1.9.

Το DotSpatial παρέχει τα κάτωθι:

- Απεικόνιση ενός χάρτη σε Windows Form του Microsoft Visual Studio
- Άνοιγμα αρχείων shp, grids, rasters και φωτογραφιών
- Ανάγνωση GPS δεδομένων
- Επιστημονική Ανάλυση
- Χειρισμός και απεικόνιση των χωρικών δεδομένων
- Αναπαράσταση συμβόλων και ετικετών
- Άμεση επαναπροβολή



Εικόνα 28- Το λογότυπο του DotSpatial.

3.5. Η UML

Η ενοποιημένη γλώσσα σχεδιασμού UML (Unified Modeling Language) είναι μια γραφική γλώσσα για την οπτική αναπαράσταση, τη διαμόρφωση προδιαγραφών και την τεκμηρίωση συστημάτων που βασίζονται σε λογισμικό. Η UML στοχεύει στο σχεδιασμό αντικειμενοστραφών συστημάτων. Το σχέδιο είναι μια απλοποιημένη αναπαράσταση της πραγματικότητας.

Δημιουργώντας ένα σχέδιο επιτυγχάνουμε τέσσερεις στόχους:

1. Παριστάνουμε οπτικά το σύστημα που έχουμε ή θέλουμε να κατασκευάσουμε,
2. Προσδιορίζουμε τη δομή και τη συμπεριφορά του συστήματος,
3. Δημιουργούμε ένα πρότυπο για να βασίσουμε την κατασκευή του συστήματος,
4. Τεκμηριώνουμε τις αποφάσεις που λάβαμε.

Σε όλους τους τεχνολογικούς τομείς ο σχεδιασμός βασίζεται σε τέσσερεις βασικές αρχές:

1. Η επιλογή του είδους του σχεδίου έχει επίπτωση στον τρόπο και τη μορφή επίλυσης του προβλήματος,
2. Όλα τα σχέδια εκφράζονται σε διαφορετικές βαθμίδες ακρίβειας,
3. Τα καλύτερα σχέδια σχετίζονται με την πραγματικότητα,
4. Ένα είδος σχεδίων δεν είναι ποτέ αρκετό.

Η UML περιλαμβάνει τρία βασικά στοιχεία:

- Οντότητες
- Σχέσεις
- Διαγράμματα

3.5.1. Διαγράμματα UML

Η UML ορίζει εννιά είδη διαγραμμάτων για την αναπαράσταση των διαφορετικών απόψεων μοντελοποίησης και είναι τα εξής:

- Διάγραμμα τάξεων (Class Diagram)
- Διάγραμμα αντικειμένων (Object Diagram)
- Διαγράμματα αλληλεπίδρασης
  - Διάγραμμα συνεργασίας (Collaboration Diagram)
  - Διάγραμμα σειράς (Sequence Diagram)
- Διάγραμμα καταστάσεων (Statechart Diagram)
- Διάγραμμα δραστηριοτήτων (Activity Diagram)
- Διάγραμμα εξαρτημάτων (Component Diagram)
- Διάγραμμα διανομής (Deployment Diagram)
- Διάγραμμα περιπτώσεων χρήσης (Use Case Diagram)

Το διάγραμμα τάξεων αναπαριστά τη στατική δομή όσον αφορά στις τάξεις και τις σχέσεις τους.
Το διάγραμμα αντικειμένων αναπαριστά αντικείμενα και τις σχέσεις τους και αντιστοιχούν σε απλοποιημένα διαγράμματα συνεργασίας που δεν αναπαριστούν μετάδοση μηνυμάτων.
Το διάγραμμα συνεργασίας είναι η αναπαράσταση των αντικειμένων, συνδέσεων και αλληλεπιδράσεων.
Το διάγραμμα σειράς είναι η χρονική αναπαράσταση των αντικειμένων και των αλληλεπιδράσεών τους.
Το διάγραμμα καταστάσεων αναπαριστά τη συμπεριφορά της τάξης όσον αφορά στις καταστάσεις.
Το διάγραμμα δραστηριοτήτων αναπαριστά τη συμπεριφορά μιας λειτουργίας ως σύνολο ενεργειών.
Το διάγραμμα εξαρτημάτων αναπαριστά τα φυσικά εξαρτήματα μιας εφαρμογής.
Το διάγραμμα διανομής αναπαριστά τη διανομή των εξαρτημάτων σε συγκεκριμένα τεμάχια του hardware (υλικού).
Το διάγραμμα περιπτώσεων χρήσης αναπαριστά τις λειτουργίες ενός συστήματος από την οπτική γωνία του χρήστη.

3.5.2. Διάγραμμα Περιπτώσεων χρήσης (Use Case Diagram)

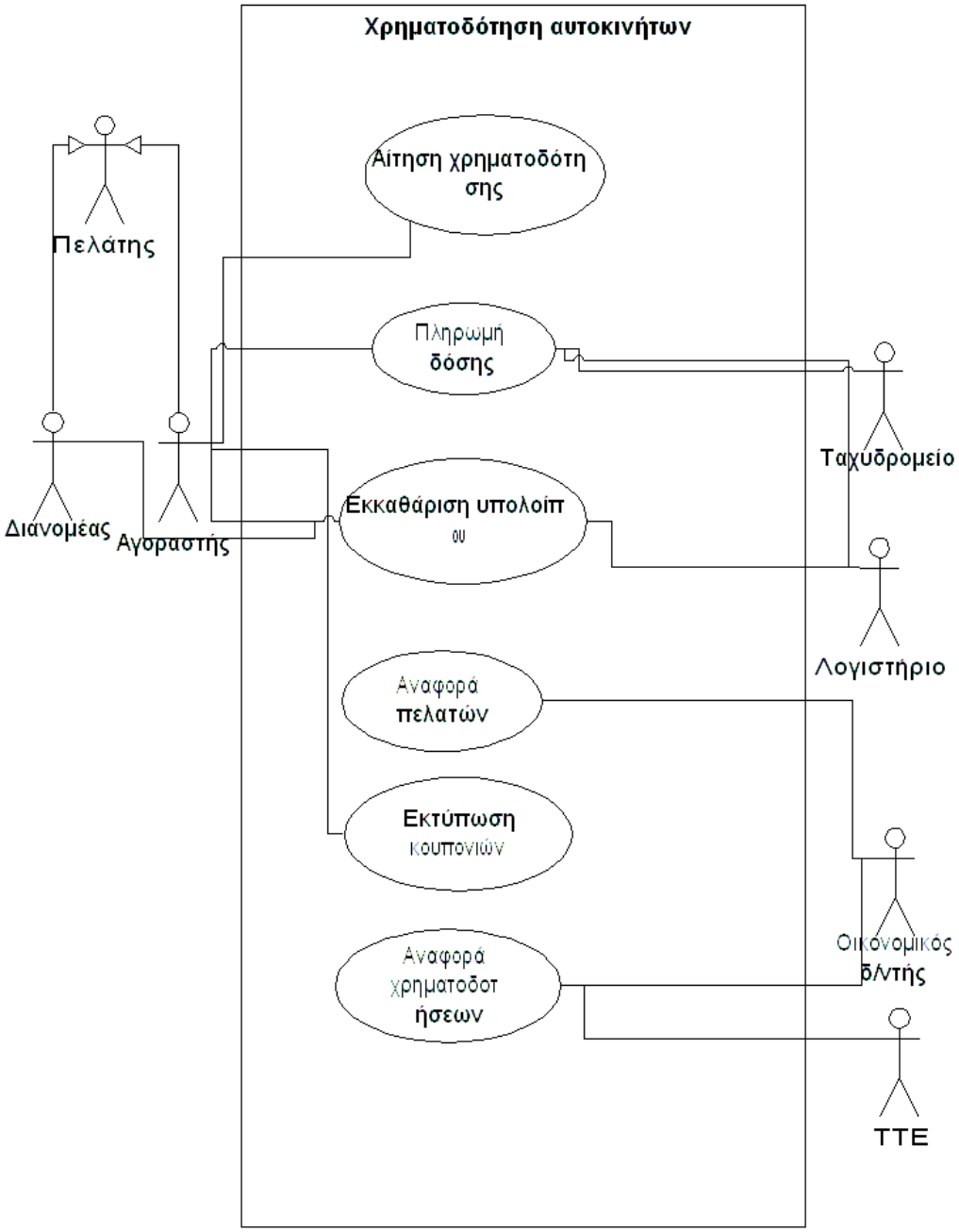
Το διάγραμμα περιπτώσεων χρήσης χρησιμοποιείται για να μοντελοποιήσει το πλαίσιο λειτουργίας του συστήματος, καθώς και τις προδιαγραφές του από την οπτική γωνία ενός χρήστη. Περιλαμβάνει:

- Τι ίδιες τις περιπτώσεις χρήσης
- Τους ενεργοποιούς (*actors*)
- Σχέσεις εξάρτησης, γενίκευσης, σύνδεσης
- Τα όρια του συστήματος

Ο ενεργοποιός (actor) αναπαριστά ένα ρόλο που παίζεται από ένα άτομο ή πράγμα, για τον οποίο υπάρχει αλληλεπίδραση με το σύστημα. Ο προσδιορισμός ενός ενεργοποιού γίνεται παρατηρώντας τους άμεσους χρήστες του συστήματος, οι οποίοι το χρησιμοποιούν και το συντηρούν και παρατηρώντας κάθε άλλο σύστημα που αλληλεπιδρά με αυτό που αναπτύσσεται. Το ίδιο φυσικό πρόσωπο μπορεί να παίξει το ρόλο πολλών ενεργοποιών ενώ πολλοί άνθρωποι μπορεί να παίζουν τον ίδιο ρόλο.

Οι ενεργοποιοί χωρίζονται σε τέσσερις κατηγορίες:

- Κύριοι Ενεργοποιοί: Άνθρωποι που χρησιμοποιούν τις κύριες λειτουργίες ενός συστήματος.
- Δευτερεύοντες Ενεργοποιοί: Άνθρωποι που εκτελούν εργασίες συντήρησης ή διοίκησης.
- Εξωτερικό hardware: Συσκευές hardware που αποτελούν μέρος της εφαρμογής χωρίς να είναι ο κύριος υπολογιστής.
- Άλλα συστήματα που αλληλοεπιδρούν με το σύστημα



Εικόνα 29- Αναπαράσταση διαγράμματος περιπτώσεων χρήσης (Use Case Diagram).

Για τα διαγράμματα περιπτώσεων χρήσης χρησιμοποιούνται σχέσεις μεταξύ περιπτώσεων χρήσης και ενεργοποιών. Η UML ορίζει τρείς σχέσεις για τη σύνδεση των περιπτώσεων χρήσης και ενεργοποιών:

- Σχέση «επικοινωνεί» (communicates): Η σχέση μεταξύ ενεργοποιού και περίπτωσης χρήσης αναπαρίσταται με μία γραμμή και είναι μοναδική.

Εικόνα 29- Αναπαράσταση σχέσης communicates.

- Σχέση «χρησιμοποιεί» (uses): Η σχέση αυτή ορίζεται μεταξύ περιπτώσεων χρήσης και σημαίνει ότι ένα στιγμιότυπο της πηγής συμπεριλαμβάνει τη συμπεριφορά του στόχου. Χρησιμοποιείται σε περιπτώσεις όπου περισσότερες από μία περιπτώσεις χρήσης μοιράζονται στοιχεία από την ίδια λειτουργικότητα.

Εικόνα 30- Αναπαράσταση σχέσης uses.

- Σχέση «επεκτείνει» (extends): Η σχέση αυτή μεταξύ δύο περιπτώσεων χρήσης αναπαριστά την επέκταση μιας περίπτωσης χρήσης σε μία άλλη. Η σχέση extends χρησιμοποιείται για να δείξει προαιρετική συμπεριφορά, συμπεριφορά που υπάρχει μόνο κάτω από ορισμένες συνθήκες και αρκετές διαφορετικές ροές που μπορεί να υπάρχουν βάσει των επιλογών των ενεργοποιών.

Εικόνα 30- Αναπαράσταση σχέσης extends.

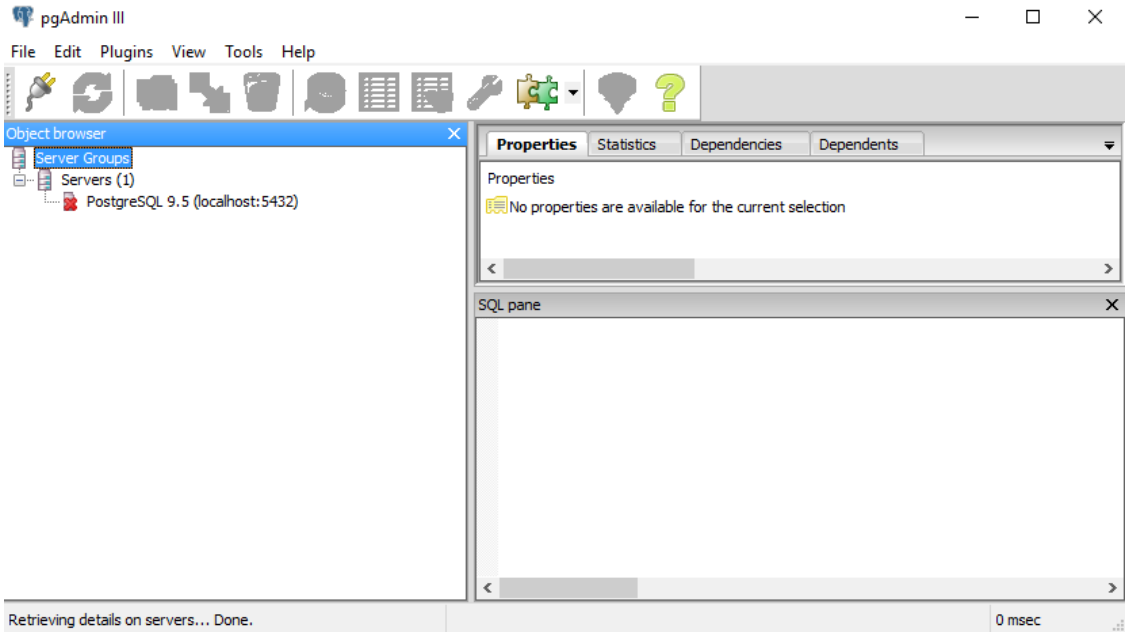
3.6. PostgreSQL και PostGIS

Η PostgreSQL είναι ένα ανοιχτού κώδικα αντικείμενο-σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (Open source object-relational database management system-ORDBMS) πολλών δυνατοτήτων. Η ανάπτυξη της διαρκεί ήδη πάνω από δύο δεκαετίες και βασίζεται σε μια αποδεδειγμένα καλή αρχιτεκτονική η οποία έχει δημιουργήσει μια ισχυρή αντίληψη των χρηστών της γύρω από την αξιοπιστία, την ακεραιότητα δεδομένων και την ορθή λειτουργία. Η PostgreSQL τρέχει σε όλα τα βασικά λειτουργικά συστήματα Linux, UNIX και Windows. Υποστηρίζει αποθήκευση μεγάλων δυαδικών αντικειμένων (binary), όπως εικόνες, ήχοι ή βίντεο. Διαθέτει επίσης περιβάλλοντα προγραμματισμού για τις γλώσσες προγραμματισμού C, C++, Java, Perl, Python, Ruby, Tcl, και διαθέτει υποστήριξη για την πλατφόρμα .NET και το πρότυπο ODBC.

Το PostGIS αποτελεί επέκταση της PostgreSQL για χωρικά δεδομένα. Η πρώτη έκδοσή του έγινε το 2005 ενώ η πιο πρόσφατη έκδοσή είναι το 2.4. Το PosGIS υποστηρίζει διανυσματικά (vector) και ψηφιδωτά (raster) δεδομένα, χωρικά ευρετήρια R-Trees και μετατροπή δεδομένων μεταξύ συστημάτων γεωγραφικών συντεταγμένων.

Για την εγκατάσταση της PostgreSQL στον υπολογιστή μας ακολουθούμε το σύνδεσμο της EnterpriseDB ως [18] ενώ την επέκταση του PosGIS την εισάγουμε στην PostgreSQL μέσω του εργαλείου αυτής Application Stack Builder. Αφού εγκαταστήσουμε τη PostgreSQL με το

PostGIS, εγκαθίσταται επίσης και το pgAdmin, το οποίο αποτελεί το γραφικό περιβάλλον της PostgreSQL για την εκτέλεση ερωτημάτων.



Εικόνα 31- Το pgAdmin.

Βασικές συναρτήσεις του PostGIS για τον υπολογισμό διαφόρων αποτελεσμάτων είναι οι εξής:

- ST_AsText(geometry g1): Προβολή του γεωμετρικού αντικειμένου στη μορφή WKT (Well Known Text).
- ST_GeomFromText(text WKT, integer srid): Επιστροφή γεωμετρικής τιμής από ένα γεωμετρικό αντικείμενο σε μορφή WKT.
- ST_SRID(geometry g1): Επιστροφή αναγνωριστικού γεωμετρικής αναφοράς μιας γεωμετρίας.
- ST_Npoints(geometry g1): Επιστροφή αριθμού σημείων γεωμετρίας.
- ST_Transform(geometry g1, integer srid): Μετατροπή νέας γεωμετρίας σε άλλο σύστημα γεωγραφικών συντεταγμένων.
- ST_Distance(geometry g1, geometry g2): Επιστροφή ελάχιστης απόστασης μεταξύ δύο γεωμετριών.
- ST_Length(geometry a_2dlinestring): Επιστροφή μήκους μιας γραμμής.
- ST_Area(geometry g1): Επιστροφή της περιοχής μιας επιφάνειας αν είναι πολύγωνο.
- ST_Contains(geometry geomA, geometry geomB): Επιστροφή αληθούς τιμής αν τουλάχιστον ένα σημείο της γεωμετρίας B ακουμπά τη γεωμετρία του A.

4. Ανάλυση Απαιτήσεων και Σχεδιασμός Εφαρμογής

4.1. Ανάλυση Απαιτήσεων

Η ανάπτυξη της εφαρμογής ‘MyNauticalMap’ στην παρούσα μεταπτυχιακή διατριβή θα εκτελεστεί μέσω της προγραμματιστικής πλατφόρμας Microsoft Visual Studio σε γλώσσα προγραμματισμού C# με ενσωματωμένη τη βιβλιοθήκη του DotSpatial στον πηγαίο κώδικά της. Η εφαρμογή ‘MyNauticalMap’ θα περιλαμβάνει κατά το ελάχιστο:

- Ναυτιλιακές πληροφορίες λιμένων και λοιπών περιοχών
- Τυχόν φωτογραφικό υλικό
- Βαθυμέτρηση θαλάσσης
- Ημερομηνία λήψης των ανωτέρω στοιχείων

Οι ναυτιλιακές πληροφορίες λιμένων και λοιπών περιοχών π.χ. οι διάφορες μετρήσεις βαθυμέτρησης θαλάσσης θα εισάγονται από τον ίδιο τον χρήστη από τις μετρήσεις που έκανε και θα ενσωματωθούν στον αρχικό χάρτη της εφαρμογής ή σε νέο επίπεδο (layer), ενώ το τυχόν φωτογραφικό υλικό θα μπορεί να εισαχθεί σε μια βάση δεδομένων με χρήση SQL Server.

Ως προς τις λειτουργικές απαιτήσεις, θα υπάρχει συγκεκριμένος χρήστης ανά πλοίο που θα χειρίζεται την εφαρμογή με τον ρόλο του χρήστη (user). Για τη σύνδεση (login) των χρηστών θα τηρείται πίνακας με το όνομα «Login» που θα υπάρχει στη βάση δεδομένων της εφαρμογής. Για τον οποιοδήποτε χρήστη θα πρέπει να ισχύουν τα ακόλουθα:

- Να έχει τη δυνατότητα να συνδεθεί με ένα username και password.
- Να μπορεί να εισάγει και να διαγράφει ναυτιλιακές πληροφορίες
- Να μπορεί να εισάγει και να τροποποιεί οποιαδήποτε δεδομένα της βάσης δεδομένων.
- Να μπορεί να αποσυνδεθεί.

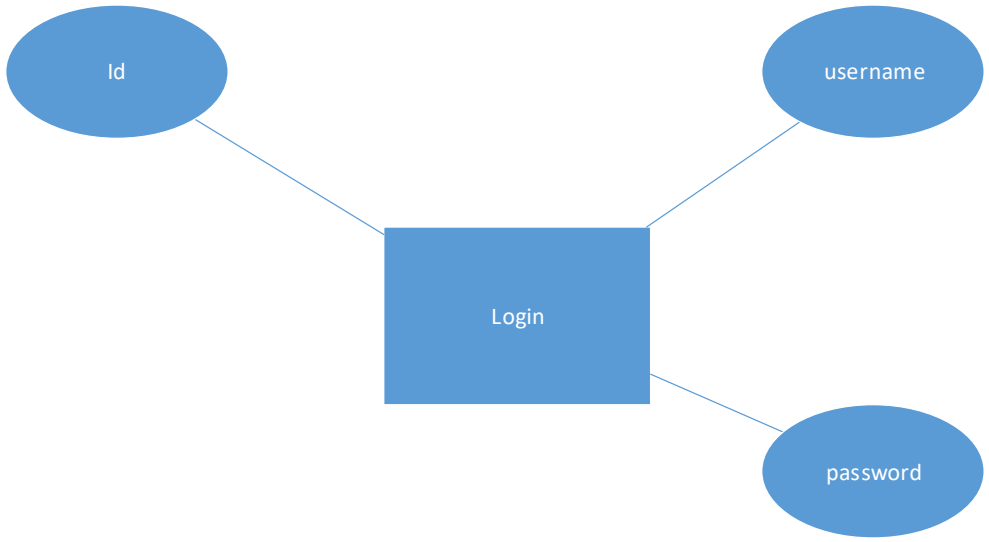
Ως προς τις μη λειτουργικές απαιτήσεις, η εφαρμογή θα πρέπει να πληροί τα κάτωθι:

- Να λειτουργεί αδιάλειπτα.
- Να μπορεί να εκτελείται σε λειτουργικό Windows XP και άνω, ενώ υπολογιστές χωρίς μεγάλη υπολογιστική ισχύ, να μπορούν να εκτελούν την εφαρμογή με ικανοποιητικό χρόνο απόκρισης.
- Η βάση δεδομένων να βρίσκεται εκτός Διαδικτύου. Κάθε πλοίο που φορτώνει πληροφορίες στη βάση δεδομένων του, για κάποιο λιμένα που επισκέφτηκε ή κάποιο αγκυροβόλι που έκανε, είναι αναγκαίο να μπορεί να τα μεταφορτώνει σε μία κεντρική βάση δεδομένων, ώστε οι πληροφορίες αυτές να μπορούν στη συνέχεια να μεταφορτώνονται σε όλα τα πλοία με κάποιο φορητό μέσο αποθήκευσης.
- Θα περιέχει ενδεικτικά δεδομένα – πληροφορίες, τα οποία έχουν παρθεί από διαδικτυακές σελίδες με ελεύθερα χωρικά datasets.

4.2. Σχεδιασμός Εφαρμογής

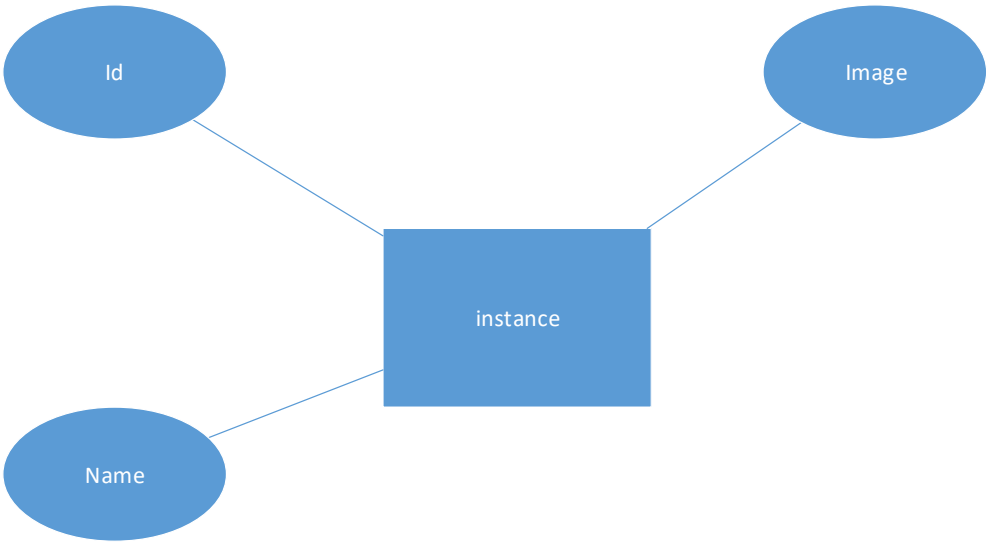
Για τον σχεδιασμό της εφαρμογής ‘MyNauticalMap’ θα απαιτηθεί να περιγραφούν οι δύο βάσεις δεδομένων των στοιχείων των χρηστών (user_id, password) και των εικόνων, όπως επίσης και οι λειτουργίες ενός χρήστη στην εφαρμογή μέσω διαγράμματος περιπτώσεων χρήσης (use case diagram - UML).

Κάθε υπεύθυνος ναυτιλίας ενός πλοίου θα έχει ένα username και ένα password. Με αυτά τα στοιχεία θα μπορεί να συνδεθεί στο σύνολο των λειτουργιών της εφαρμογής.
Όσον αφορά τη βάση δεδομένων των στοιχείων των χρηστών της εφαρμογής, αυτή θα περιέχει τον πίνακα ‘Login’. Ο πίνακας ‘Login’ θα περιέχει τα πεδία id, ship_name, username και password. Το διάγραμμα ER απεικονίζεται παρακάτω:



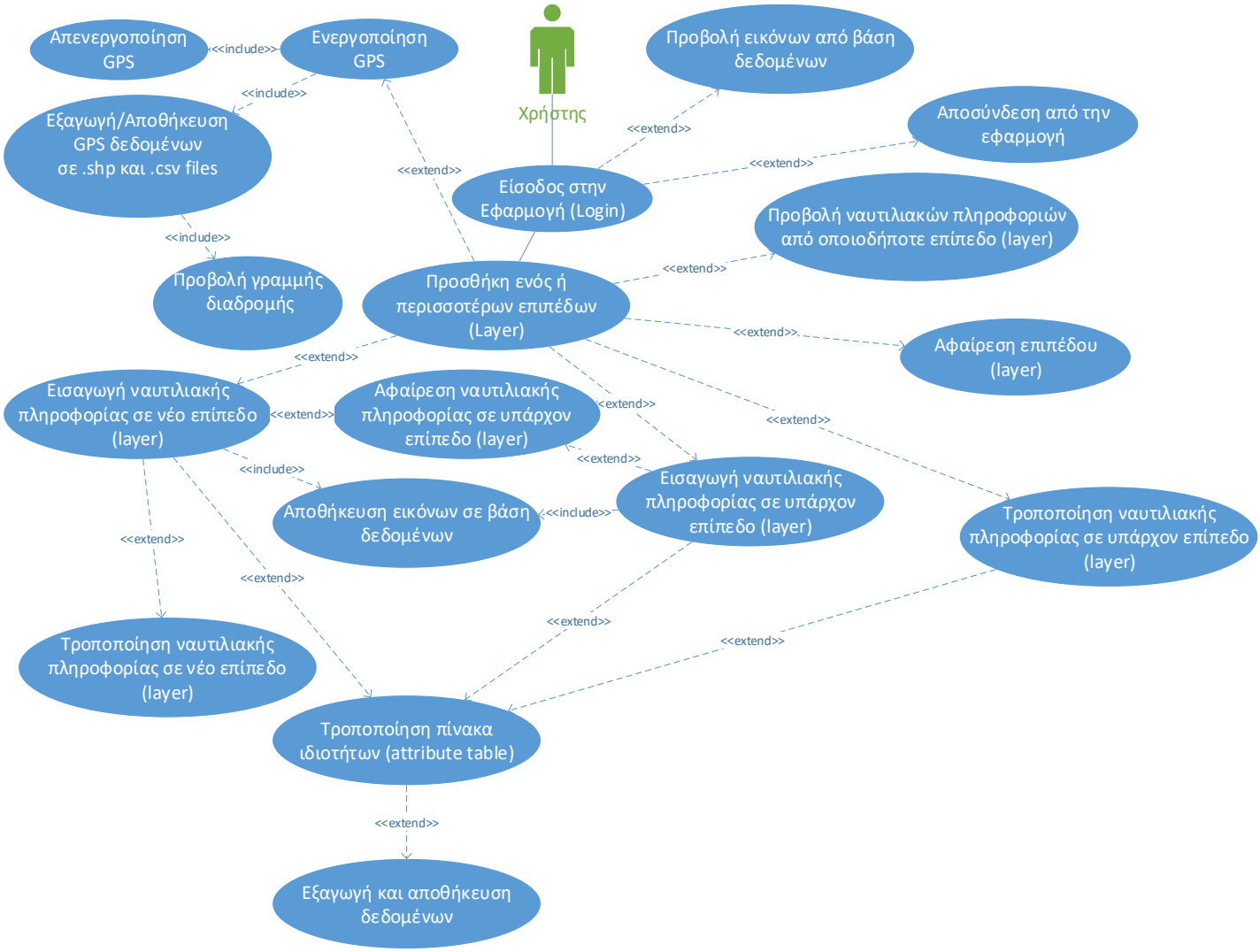
Εικόνα 32 – Λογικό διάγραμμα ER ΒΔ Στοιχεία χρήστη.

Η βάση δεδομένων για την εισαγωγή εικόνων περιέχει τον πίνακα CIW με τα πεδία id, Name και Image. Το διάγραμμα ER απεικονίζεται παρακάτω:



Εικόνα 33 – Λογικό διάγραμμα ER ΒΔ Εικόνων χρήστη.

Για τις λειτουργίες ενός χρήστη πάνω στην εφαρμογή ‘MyNauticalMap’, αυτές μπορούν να απεικονιστούν με τη βοήθεια του διαγράμματος περιπτώσεων χρήσης (use case diagram) της γλώσσας UML. Ο οποιοσδήποτε χρήστης θεωρείται ως actor στο σύστημα – εφαρμογή ενώ οι διάφορες λειτουργίες που μπορεί να εκτελέσει σε αυτή ονομάζονται περιπτώσεις χρήσης (use case). Μεταξύ των πολλών περιπτώσεων χρήσης είναι δυνατό να υπάρχει μια σχέση. Το διάγραμμα περιπτώσεων χρήσης ενός χρήστη απεικονίζεται παρακάτω:



Εικόνα 34 – Διάγραμμα Περιπτώσεων Χρήσης ενός χρήστη (Use case diagram).

Όπως απεικονίζεται παραπάνω, ο χρήστης αφού συνδεθεί στην εφαρμογή με τη χρήση μοναδικού username και password που του αντιστοιχεί θα μπορεί:

- Να προσθέσει ένα ή περισσότερα επίπεδα (layer)
- Να προβάλει εικόνες από τη βάση δεδομένων
- Να αποσυνδεθεί από την εφαρμογή

Στη συνέχεια αφού προσθέσει ένα ή περισσότερα επίπεδα (layers) θα υπάρχει η δυνατότητα για:

- Προβολή ναυτιλιακών πληροφοριών από οποιοδήποτε επίπεδο
- Ενεργοποίηση GPS
- Αφαίρεση επιπέδου (layer)
- Τροποποίηση ναυτιλιακής πληροφορίας σε υπάρχον επίπεδο (layer)
- Εισαγωγή ναυτιλιακής πληροφορίας σε υπάρχον επίπεδο (layer)
- Εισαγωγή ναυτιλιακής πληροφορίας σε νέο επίπεδο (layer)

Με την εισαγωγή των δεδομένων ο χρήστης πλέον θα είναι δυνατό να εκτελείται:

- Αφαίρεση ναυτιλιακής πληροφορίας σε υπάρχον επίπεδο (layer)
- Αποθήκευση εικόνων σε βάση δεδομένων
- Τροποποίηση πίνακα ιδιοτήτων (attribute table)
- Τροποποίηση ναυτιλιακής πληροφορίας σε νέο επίπεδο (layer)

Με την ενεργοποίηση του GPS ο χρήστης πλέον θα είναι δυνατό να εκτελεί:

- Εξαγωγή και αποθήκευση GPS δεδομένων σε αρχεία .shp και .csv
- Απενεργοποίηση του GPS
- Προβολή γραμμής διαδρομής

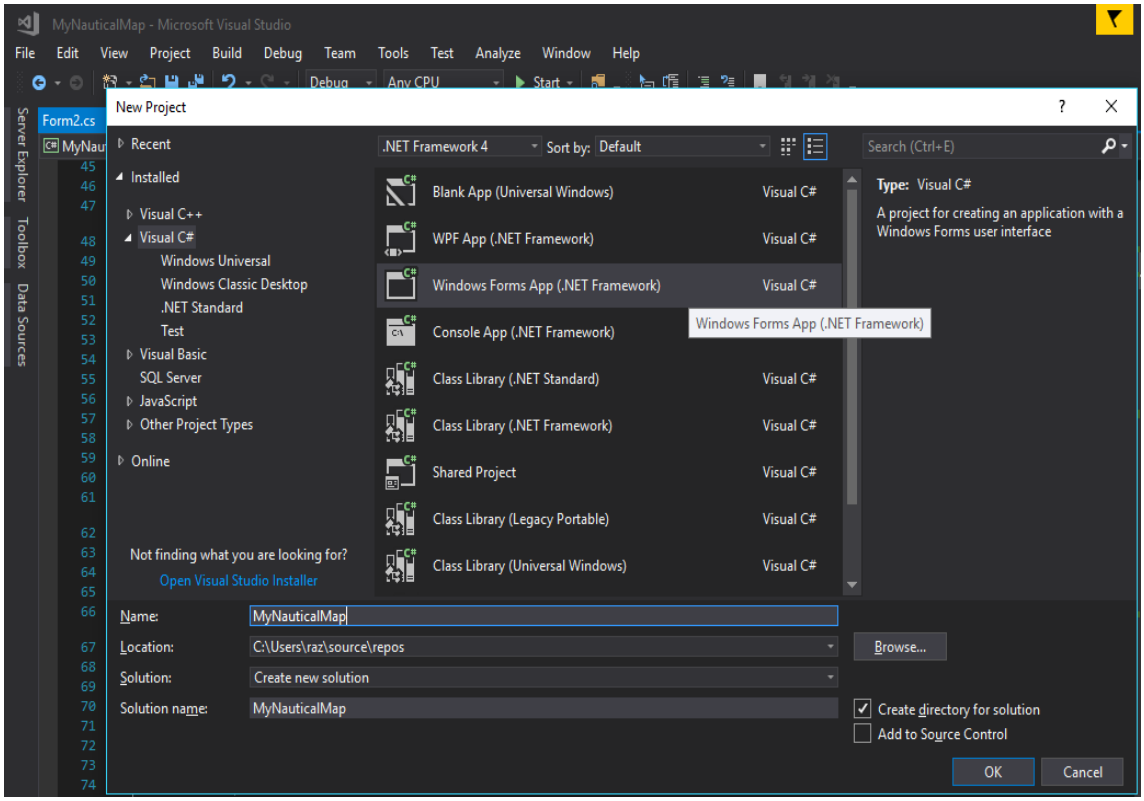
Τέλος, ο χρήστης μπορεί να αποθηκεύσει και να εξάγει τα δεδομένα του ώστε να είναι διαθέσιμα στον ίδιο αλλά και σε άλλους χρήστες της εφαρμογής.

5. Δόμηση Εφαρμογής

5.1. Σχεδιαστικό μέρος

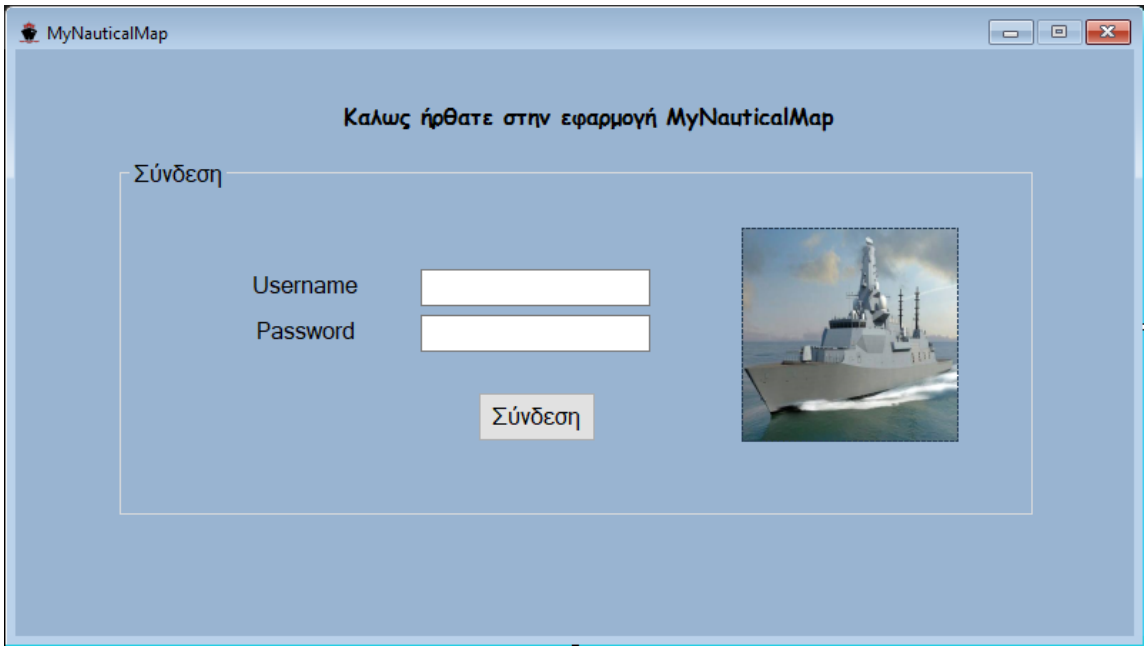
Η εφαρμογή MyNauticalMap αναπτύχθηκε στην πλατφόρμα του Microsoft Visual Studio 2017 σε γλώσσα C#. Τα βήματα ανάπτυξης της εφαρμογής παρατίθενται παρακάτω:

Αρχικά, ανοίγουμε το πρόγραμμα του Visual Studio 2017 και επιλέγουμε File-> New Project.



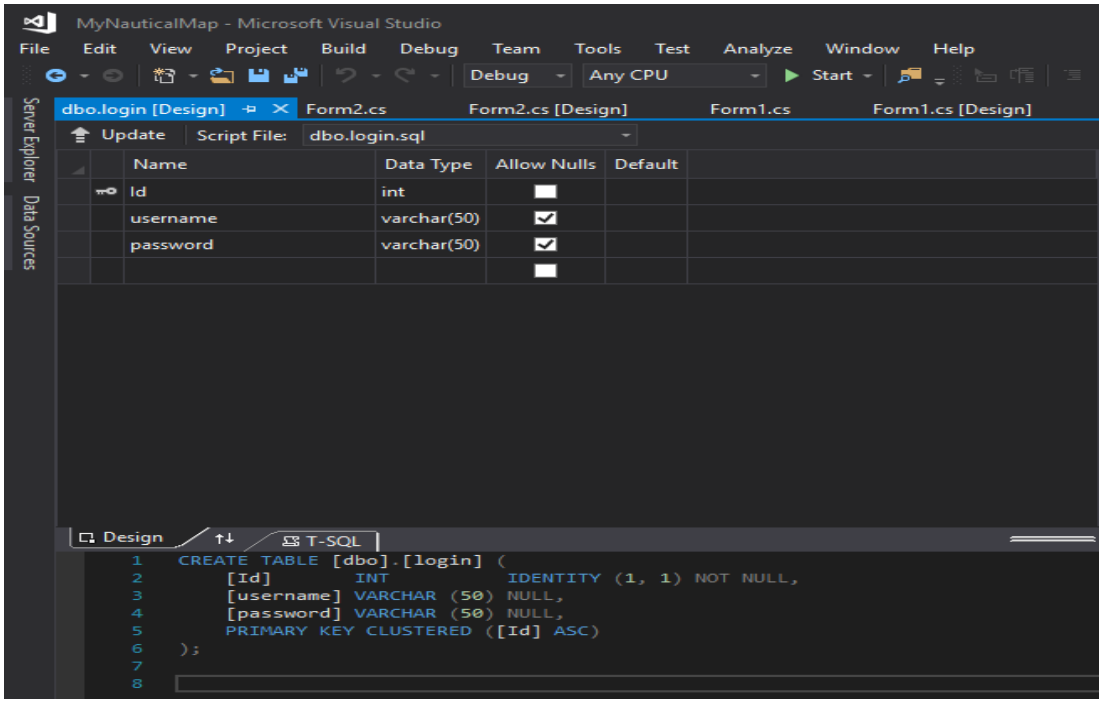
Εικόνα 35- Δημιουργία μίας νέας Windows Form εφαρμογής στο Visual Studio.

Στη συνέχεια, αφού έχει δημιουργηθεί η εργασία μας, στο σχεδιαστικό κομμάτι (design) της πλατφόρμας του Visual Studio, μπορούμε να επεξεργαστούμε το παράθυρο (Form1), το οποίο δημιουργήθηκε αυτόματα με τη δημιουργία της εργασίας. Στο form1 σχεδιάζουμε και αναπτύσσουμε το παράθυρο της εισαγωγής των στοιχείων μας και της σύνδεσής μας στην εφαρμογή MyNauticalMap. Σχεδιαστικά το τελικό form1 απεικονίζεται παρακάτω:



Εικόνα 36- Το αρχικό παράθυρο της εφαρμογής MyNauticalMap.

Ως προς το λειτουργικό κομμάτι του ανωτέρω παραθύρου, για τη σύνδεση ενός χρήστη στην εφαρμογή απαιτείται η δημιουργία μιας βάσης δεδομένων με το όνομα personel. Στη βάση δεδομένων personel δημιουργούμε τον πίνακα login, ο οποίος θα χρησιμοποιηθεί για την αποθήκευση των πλοίων που θα έχουν πρόσβαση στην εφαρμογή. Η βάση δεδομένων δημιουργείται μέσω του Microsoft SQL Server επιλέγοντας την ετικέτα Server Explorer του Visual Studio.



Εικόνα 37- Το σχεδιαστικό μέρος της βάσης δεδομένων personel στο Visual Studio.

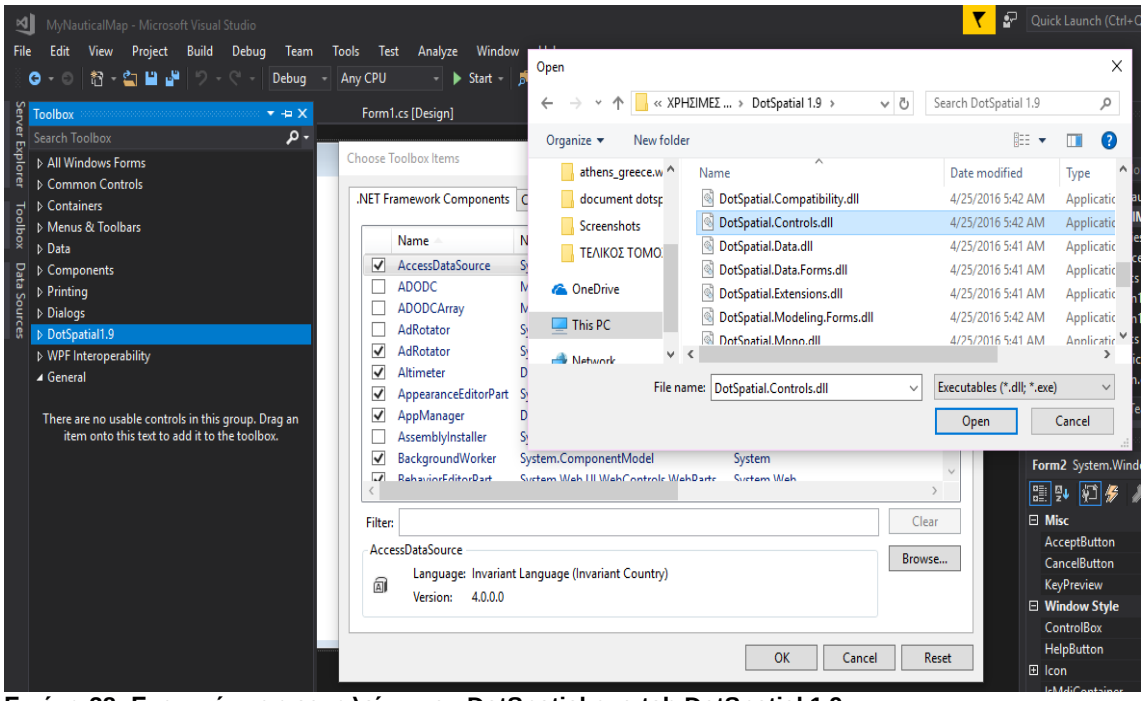
Για τη λειτουργία του κουμπιού ‘Σύνδεση’ χρειάζεται να γίνει σύνδεση στη βάση δεδομένων personel, να ελεγχθούν τα εισαγόμενα στοιχεία και αν αυτά είναι ορθά, να κλείσει προσωρινά το παράθυρο της σύνδεσης για να ανοίξει το νέο παράθυρο της εφαρμογής (Form2). Παρακάτω απεικονίζεται ο κώδικας για το κουμπί της σύνδεσης:

```
//Σύνδεση στην εφαρμογή MyNauticalMap
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "" || textBox2.Text == "")
    {
        MessageBox.Show("Παρακαλώ εισάγετε τα στοιχεία σας και στα δύο πεδία");
        textBox1.Text = "";
        textBox2.Text = "";
    }
    else
    {
        SqlConnection con = new SqlConnection(@"Data Source=
        (LocalDB)\MSSQLLocalDB;AttachDbFilename=
        C:\Users\raz\Desktop\ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ\MyNauticalMap\personel.mdf;
        Integrated Security=True;Connect Timeout=30;");
        SqlDataAdapter sda = new SqlDataAdapter("Select Count(*) From login where username='"
        + textBox1.Text + "' and password='" + textBox2.Text + "'", con);
        DataTable dt = new DataTable();
        sda.Fill(dt);
        if (dt.Rows[0][0].ToString() == "1")
        {
            glob = textBox1.Text;
            this.Hide();
            Form2 form2 = new Form2();
            form2.Show();
        }
        else
        {
            MessageBox.Show("Λάθος Στοιχεία!Παρακαλώ ξαναεισάγετε τα στοιχεία σας");
            textBox1.Text = "";
            textBox2.Text = "";
        }
    }
}
```

Παρατηρούμε ότι από τον ανωτέρω κώδικα, για τη σύνδεση ενός χρήστη στην κύρια εφαρμογή, ελέγχεται αρχικά η υποχρεωτική συμπλήρωση και των δύο πεδίων (username και password). Εάν έχουμε συμπληρώσει και τα δύο πεδία τότε ελέγχεται η ορθότητα των στοιχείων αυτών με τα στοιχεία που είναι κατοχυρωμένα στη βάση δεδομένων personel. Η ορθή συμπλήρωση των στοιχείων ενός χρήστη μεταφέρει αυτόν τον χρήστη στην κύρια εφαρμογή (Form2). Επίσης, η glob μεταβλητή είναι μια global μεταβλητή, η οποία μεταφέρει το username του χρήστη στο Form2.

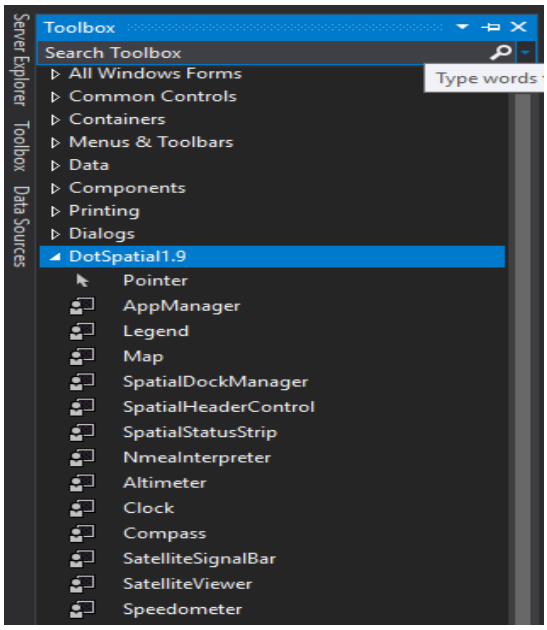
Στη συνέχεια, δημιουργούμε το παράθυρο της κύριας εφαρμογής (Form2) επιλέγοντας από το Solution Explorer δεξί κλικ στο MyNauticalMap-> Add-> Windows Form. Αφού δημιουργηθεί το Form2, προχωράμε στη σχεδίαση του Form2 με τα εργαλεία του DotSpatial. Για τη σχεδίαση αυτή θα επιλέξουμε να δημιουργήσουμε στην καρτέλα Tools του Visual Studio ένα νέο tab με την ονομασία DotSpatial 1.9 και αφού δημιουργηθεί αυτό το tab με δεξί κλικ επιλέγουμε Choose Items-> .Net Framework Components-> Browse-> DotSpatial.Controls.dll με σκοπό να εμφανιστούν τα διάφορα εργαλεία που μας προσφέρει το DotSpatial στο tab

Dotspatial 1.9. Ο φάκελος που περιέχει τα διάφορα dlls για την έκδοση 1.9 του DotSpatial βρίσκεται διαθέσιμος για δωρεάν κατέβασμα από τη διαδικτυακή σελίδα του GitHub ως αναγράφεται στο [21].



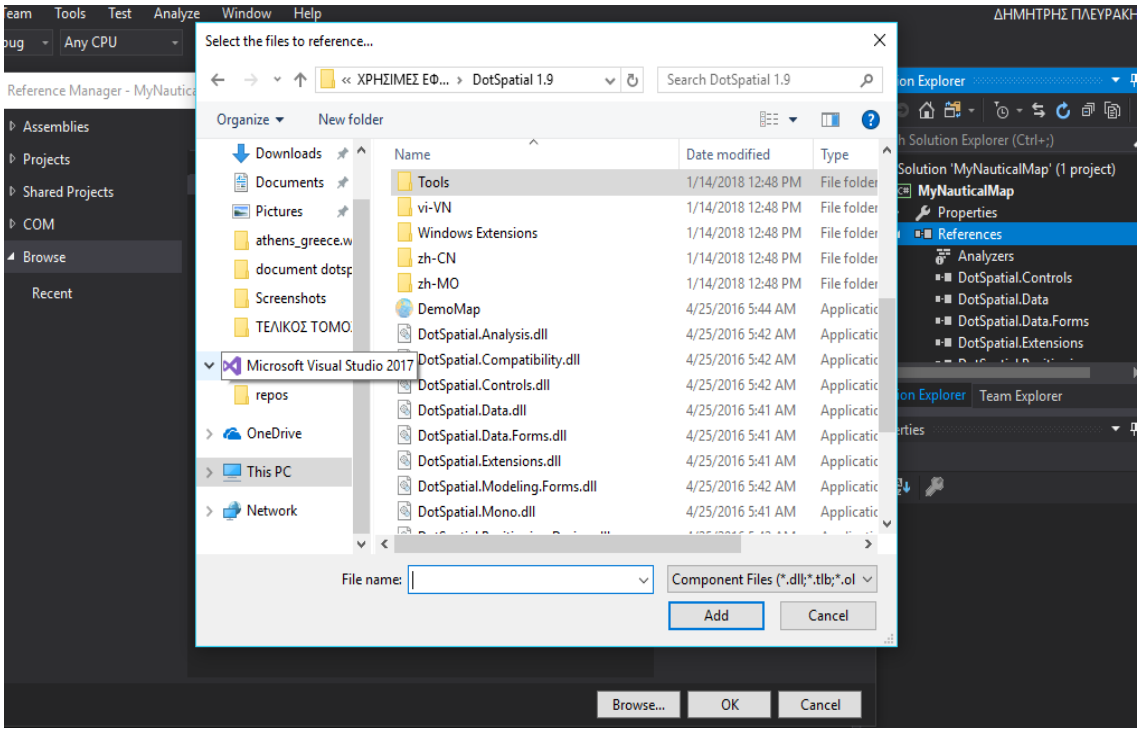
Εικόνα 38- Ενσωμάτωση εργαλείων του DotSpatial στο tab DotSpatial 1.9.

Επιλέγοντας τα αρχεία dll DotSpatial.Controls, DotSpatial.Positioning και DotSpatial.Positioning.Forms εμφανίζονται στην καρτέλα DotSpatial1.9 των Tools τα εργαλεία που απεικονίζονται παρακάτω:



Εικόνα 39- Εργαλεία του DotSpatial 1.9.

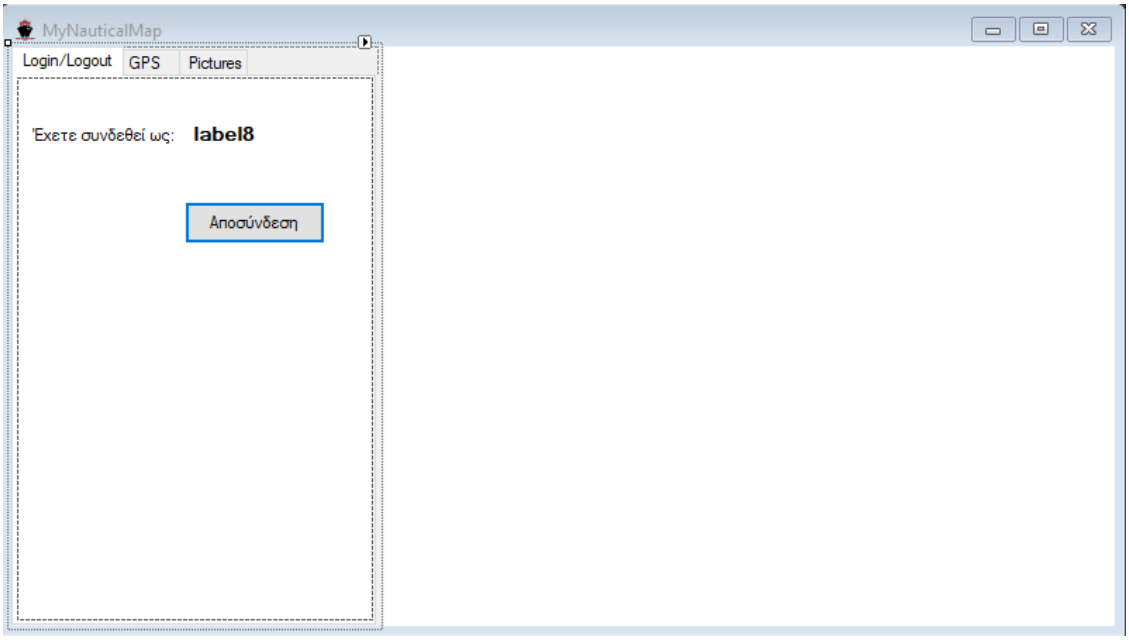
Επιπρόσθετα, ενσωματώνουμε αρχεία dll του DotSpatial 1.9 φακέλου απαραίτητα για τη λειτουργικότητα της εφαρμογής MyNauticalMap, επιλέγοντας με δεξί κλικ στα References του Solution Explorer του Visual Studio Add Reference-> Browse όλα τα αρχεία dll του φακέλου εκτός του αρχείου DotSpatial.Controls.



Εικόνα 40- Προσθήκη των DotSpatial References στην εφαρμογή.

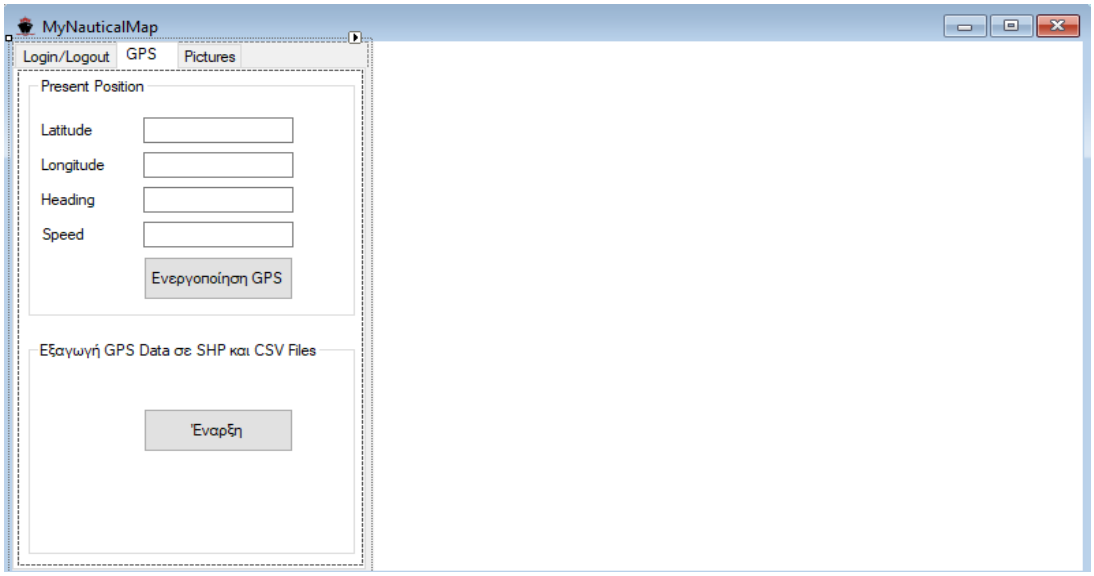
Αφού ολοκληρώσουμε τη διαδικασία με την προσθήκη των DotSpatial References και προσθέσουμε το System.ComponentModel.Composition από το παράθυρο των References στην καρτέλα Assemblies-> Framework, ξεκινάμε να σχεδιάζουμε το Form2.

Ως προς τη σχεδίαση του παραθύρου της εφαρμογής (Form2), αρχικά προσθέτουμε από το DotSpatial 1.9 του Toolbox του Visual Studio το Spatial Dock Manager, το οποίο διαιρεί το παράθυρο σε δύο μέρη (panels) για την προσθήκη διαφόρων λειτουργιών του DotSpatial σε αυτά. Έπειτα, αφού έχει χωριστεί το Form2 σε δύο μέρη, στο δεξί panel προσθέτουμε το map από το DotSpatial 1.9 του Toolbox, στο οποίο θα προβάλλεται οποιοσδήποτε χάρτης που φορτώνεται στην εφαρμογή, ενώ στο αριστερό panel προσθέτουμε το tab control από το Toolbox, το οποίο θα διαχειρίζεται τις καρτέλες που θα αναπτυχθούν όπως η καρτέλα του GPS κ.α. Για τα δύο panels του Spatial Dock Manager που προσθέσαμε στο παράθυρο της εφαρμογής, ρυθμίζουμε στις ιδιότητες (Properties) το Dock να είναι Fill. Όσον αφορά το αριστερό panel προσθέσαμε σε αυτό το tab control και για αυτό ρυθμίζουμε στις ιδιότητες το Dock να είναι Fill. Στη συνέχεια, δημιουργούμε τρία tab controls το Login/Logout, το GPS και το Pictures. Το Login/Logout είναι η καρτέλα για τη δυνατότητα ενός χρήστη που έχει ήδη συνδεθεί στην κύρια εφαρμογή να αποσυνδεθεί. Για την αποσύνδεση του χρήστη από την εφαρμογή, εισάγουμε ένα button και δύο labels από το Toolbox. Το δεύτερο label θα απεικονίζει τον εκάστοτε χρήστη που είναι συνδεδεμένος στην εφαρμογή. Η καρτέλα Login/Logout παρουσιάζεται παρακάτω:



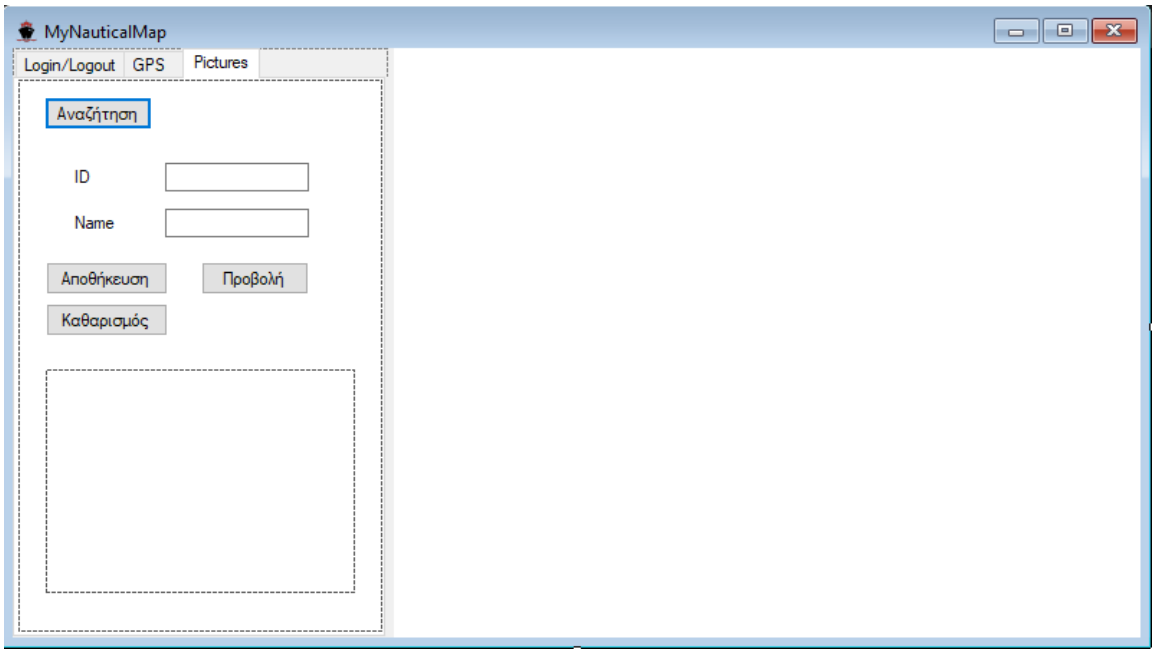
Εικόνα 41- Η καρτέλα Login/Logout.

Για την καρτέλα του GPS, θέλουμε να εμφανίσουμε την τρέχουσα τοποθεσία μας στον χάρτη που εισάγουμε στην εφαρμογή, να έχουμε τη δυνατότητα ενεργοποίησης και απενεργοποίησης του GPS καθώς και εξαγωγή των GPS δεδομένων σε αρχεία shapefile και csv όταν το GPS είναι ενεργοποιημένο. Η απεικόνιση της τρέχουσας τοποθεσίας μας και η εξαγωγή των GPS δεδομένων ομαδοποιούνται σε δύο ξεχωριστά groupboxes στην καρτέλα του GPS. Στο πάνω groupbox που θα απεικονίζεται η τρέχουσα τοποθεσία μας, εισάγουμε ένα button, το οποίο έχει το ρόλο της ενεργοποίησης/απενεργοποίησης του GPS και εισάγουμε labels και textboxes για την απεικόνιση του γεωγραφικού μήκους, πλάτους (latitude, longitude), αληθούς πορείας (heading true) και ταχύτητας (speed). Αντίθετα, στο κάτω groupbox για την εξαγωγή των GPS δεδομένων προσθέτουμε ένα button για την έναρξη και τη λήξη καταγραφής των δεδομένων αυτών. Η καρτέλα για το GPS απεικονίζεται παρακάτω:



Εικόνα 42- Η καρτέλα του GPS.

Για την καρτέλα απεικόνισης διαφόρων εικόνων, τις οποίες έχουμε την επιλογή να εισάγουμε στη βάση δεδομένων content, προσθέτουμε ένα button για αναζήτηση εικόνων από τον τοπικό μας υπολογιστή, δύο labels με τα textboxes για εισαγωγή από τον χρήστη ενός ID ή ονόματος της φωτογραφίας και τρία buttons για λειτουργίες όπως η προβολή, η αποθήκευση και ο καθαρισμός των στοιχείων και της φωτογραφίας που προβλήθηκε. Επιπρόσθετα, για την προβολή μιας επιλεγμένης φωτογραφίας, προσθέτουμε ένα picturebox. Η καρτέλα για τις εικόνες απεικονίζεται παρακάτω:



Εικόνα 43- Η καρτέλα των εικόνων.

5.2. Λειτουργικό μέρος

Ως προς τη λειτουργία της κύριας εφαρμογής, για να εμφανίσουμε τις διάφορες δυνατότητες που θα μας προσφέρει η βιβλιοθήκη του DotSpatial, θα πρέπει να εισάγουμε στο παράθυρο της εφαρμογής (Form2) το AppManager από το φάκελο DotSpatial 1.9 του Toolbox. Το AppManager (διαχειριστής εφαρμογής) είναι ένα εργαλείο, το οποίο διαχειρίζεται τις δυνατότητες που έχει το DotSpatial. Αφού εισάγουμε το AppManager, στις ιδιότητές του επιλέγουμε ως DockManager το spatialDockManager1 και ως Map το map1. Στον κώδικα για να δώσουμε λειτουργικότητα στο AppManager εισάγουμε το κάτωθι μετά το public partial class Form2: Form:

```
[Export("Shell", typeof(ContainerControl))]
private static ContainerControl Shell;
```

Επίσης, μετά το Public Form2() γράφουμε τον κάτωθι κώδικα:

```
public Form2()
{
    InitializeComponent();

    if (DesignMode) return;
    Shell = this;

    appManager1.LoadExtensions();

}
```

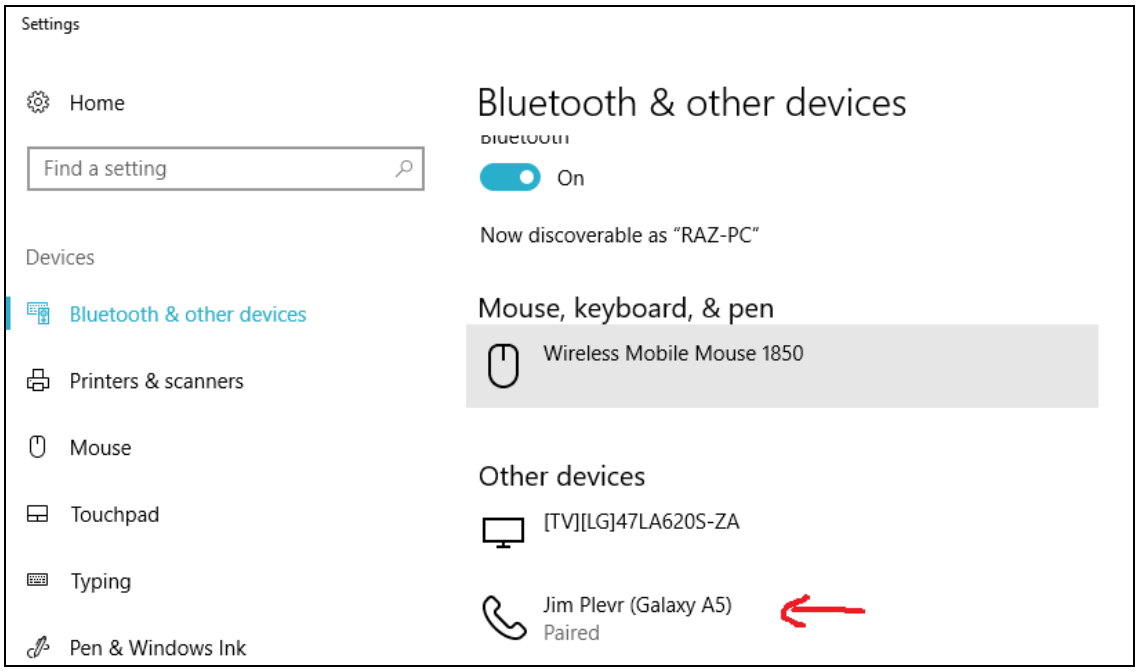
Αφού προσθέσαμε τον παραπάνω κώδικα στο πρόγραμμά μας, συνεχίζουμε με τη δημιουργία ενός φακέλου Plugins στην διαδρομή C:\Users\raz\Desktop\ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ\MyNauticalMap\MyNauticalMap\bin\Debug εκεί όπου βρίσκεται η εφαρμογή μας. Στον φάκελο Plugins επικολλούμε αρχεία dll χρήσιμα για τη λειτουργικότητα της εφαρμογής μας, τα οποία τα αντιγράψαμε από τον φάκελο DotSpatial 1.9 που κατεβάσαμε από το GitHub και είναι:

- DotSpatial.Plugins.About
- DotSpatial.Plugins.FindFeature
- DotSpatial.Plugins.Measure
- DotSpatial.Plugins.ShapeEditor
- DotSpatial.Plugins.SimpleLegend
- DotSpatial.Plugins.StatusBarImprovements
- DotSpatial.Plugins.TableEditor

Τα παραπάνω δίνουν λειτουργικότητες στην εφαρμογή όπως τη δημιουργία υπομνήματος (legend) για τον εκάστοτε χάρτη που φορτώνουμε στην εφαρμογή, εκτέλεση μετρήσεων απόστασης διανυσματικών δεδομένων, επεξεργασία πίνακα ιδιοτήτων (attribute table) και άλλα.

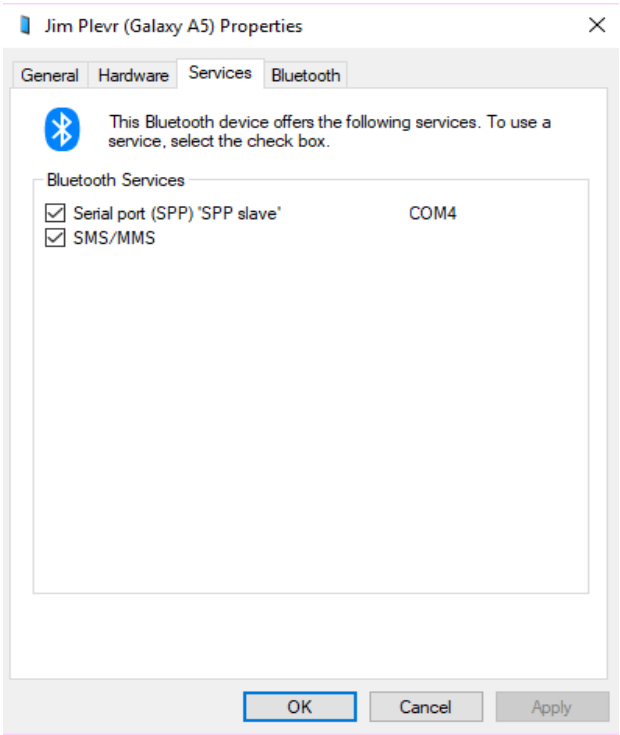
5.2.1. Η καρτέλα GPS

Για την ενεργοποίηση και τη σύνδεση του GPS στην εφαρμογή μας, απαιτείται η ύπαρξη ενός δέκτη GPS (GPS Receiver), ο οποίος θα λαμβάνει τα GPS δεδομένα και θα μπορεί να τα μεταφέρει σε έναν υπολογιστή. Στην περίπτωσή μας, ως δέκτη GPS χρησιμοποιούμε ένα κινητό android smartphone, το οποίο υποστηρίζει λήψη GPS δεδομένων. Μέσω της εφαρμογής Bluetooth GPS Output της Meowsbox, την οποία μπορούμε να κατεβάσουμε δωρεάν από το Playstore της Google [22], μπορούμε με την τεχνολογία Bluetooth να μεταφέρουμε τα GPS δεδομένα στον υπολογιστή μας. Για να είναι εφικτή η σύνδεση του smartphone με τον υπολογιστή μας πρέπει αρχικά να ενεργοποιηθεί η τεχνολογία Bluetooth και στις δύο συσκευές.



Εικόνα 44- Σύνδεση μεταξύ κινητού τηλεφώνου και υπολογιστή μέσω Bluetooth.

Έπειτα, αφού ανοίξουμε την εφαρμογή Bluetooth GPS Output από το κινητό μας τηλέφωνο και ξεκινήσουμε την αποστολή GPS δεδομένων προς τον υπολογιστή μας, στις υπηρεσίες του κινητού τηλεφώνου εμφανίζεται η υπηρεσία SPP (Serial Port), η οποία λειτουργεί στη θύρα COM4.



Εικόνα 45- Η υπηρεσία SPP στις ιδιότητες του συνδεδεμένου μέσω Bluetooth smartphone.

Η υπηρεσία SPP στην ουσία προσομοιώνει τη σύνδεση ενός δέκτη GPS με υπολογιστή μέσω καλωδίου στη σειριακή θύρα αυτού του υπολογιστή. Με άλλα λόγια, η μεταφορά των GPS δεδομένων από το πρόγραμμα Bluetooth GPS Output που βρίσκεται στο smartphone γίνεται προς την εικονική θύρα του υπολογιστή COM4 και μέσω αυτής με κατάλληλες ρυθμίσεις μπορούν να διανεμηθούν αυτά τα GPS δεδομένα σε οποιαδήποτε χαρτογραφική εφαρμογή. Για να επιτευχθεί η σύνδεση και η μεταφορά των GPS δεδομένων στην εφαρμογή MyNauticalMap πρέπει να εισάγουμε το εργαλείο serialport στο Form2 και να ρυθμίσουμε στις ιδιότητές του το BaudRate: 4800 και PortName: COM4. Τα δεδομένα που αποστέλλονται από το smartphone μέσω Bluetooth στον υπολογιστή μας και ειδικότερα στην εφαρμογή μας είναι NMEA δεδομένα, δηλαδή δεδομένα με μια συγκεκριμένη μορφή που στη συνέχεια μέσω κώδικα ή ενός διερμηνέα να αποσαφηνιστούν και να αξιοποιηθούν. Ο κώδικας για την επεξεργασία των NMEA δεδομένων απεικονίζεται παρακάτω:

```
if (serialPort1.IsOpen)
{
    string data = serialPort1.ReadExisting();
    string[] strArr = data.Split('$');
    for (int i = 0; i < strArr.Length; i++)
    {
        string strTemp = strArr[i];
        string[] lineArr = strTemp.Split(',');
        if (lineArr[0] == "GPRMC")
        {

            try
            {
                //date
                date = lineArr[9];

                //time
                time = lineArr[1];
```

Όπως είχε αναφερθεί στο κεφάλαιο 2.2.1 και αποτυπώθηκε η δομή ενός RMC μηνύματος στην εικόνα 18, ο παραπάνω κώδικας λαμβάνει τα διάφορα NMEA μηνύματα και τα αποθηκεύει σε μια μεταβλητή data. Έπειτα, τα διαχωρίζει ως προς το σύμβολο $ αφού όλα τα NMEA μηνύματα ξεκινούν πάντα με το σύμβολο $. Όλα αυτά τα μηνύματα χωρίς το σύμβολο $ αποθηκεύονται σε ένα πίνακα strArr όπου στη συνέχεια το κάθε μήνυμα του πίνακα strArr διαχωρίζεται ανά κόμματα και γίνεται έλεγχος αν η πρώτη λέξη του μηνύματος αυτού είναι ίση με το GPRMC (είναι δηλαδή RMC μήνυμα).

Για την προβολή των στοιχείων τοποθεσίας ενός χρήστη στην καρτέλα GPS της εφαρμογής, δηλαδή για την προβολή των τιμών των Latitude, Longitude, Heading και Speed, αφού ελεγχθούν τα μηνύματα που λαμβάνονται από την εφαρμογή και είναι GPRMC και GPHDT, τότε επιλέγονται οι κατάλληλες θέσεις ενός μηνύματος για την απόδοση των τιμών αυτών. Ειδικότερα, οι τιμές Latitude, Longitude και Speed προκύπτουν από το GPRMC μήνυμα ενώ το Heading προκύπτει από το GPHDT μήνυμα. Επιπρόσθετα, τα Latitude και Longitude όταν λαμβάνονται ως NMEA δεδομένα από την εφαρμογή Bluetooth GPS Output, αυτά βρίσκονται σε μορφή DDMM.mm και DDDMM.mm αντίστοιχα. Αυτό σημαίνει ότι πρέπει να τροποποιηθούν στη μορφή D.D° (Decimal Degrees). Για τη μετατροπή των Longitude και Latitude σε decimal degrees υλοποιείται ο παρακάτω κώδικας:

```
//Latitude
Double dLat = Convert.ToDouble(lineArr[3]);
dLat = dLat / 100;
string[] lat = dLat.ToString().Split('.');
string M = lat[1].Substring(0, 2);
string m = lat[1].Substring(2);
string Mm = M + "." + m;
double d = Math.Round((Convert.ToDouble(Mm) / 60), 5);
string ds = d.ToString().Substring(2);



latitude = lat[0].ToString() + "." + ds + lineArr[4].ToString();
plat = Double.Parse(lat[0].ToString() + "." + ds);



//Longitude
Double dLon = Convert.ToDouble(lineArr[5]);
dLon = dLon / 100;
string[] lon = dLon.ToString().Split('.');
string Mlon = lon[1].Substring(0, 2);
string mlon = lon[1].Substring(2);
string Mmlon = Mlon + "." + mlon;
double dlon = Math.Round((Convert.ToDouble(Mmlon) / 60), 5);
string dslon = dlon.ToString().Substring(2);

longitude = lon[0].ToString() + "." + dslon + lineArr[6].ToString();

plon = Double.Parse(lon[0].ToString() + "." + dslon);
```

Παρατηρούμε στον παραπάνω κώδικα ότι για το latitude αρχικά μετατρέπεται η τιμή από το RMC μήνυμα σε δεκαδικό αριθμό και αποθηκεύεται στη μεταβλητή dLat. Έπειτα, η dLat διαιρείται με το 100 για να μετατοπιστεί η υποδιαστολή δύο θέσεις αριστερότερα. Οπότε η αρχική μορφή του latitude που ήταν DDMM.mm μετατρέπεται σε DD.MMmm. Στη συνέχεια, μετατρέποντας το dLat σε αλφαριθμητικό το διασπάμε σε δύο λέξεις το DD και το MMmm και τα αποθηκεύουμε σε έναν αλφαριθμητικό πίνακα. Το lat[0] είναι οι συντεταγμένες σε μοίρες DD και το lat[1] η λέξη MMmm. Το MM το αποθηκεύουμε σε μια μεταβλητή M ενώ το mm, το οποίο είναι μεταβλητό το αποθηκεύουμε στη μεταβλητή m. Στην Mm μεταβλητή αποθηκεύεται η ένωση των M με το m με τελεία. Στη δεκαδική μεταβλητή d αποθηκεύεται η διαίρεση του Mm με το 60 και το αποτέλεσμα στρογγυλοποιείται σε πέντε δεκαδικά. Στο τέλος στην ds αλφαριθμητική μεταβλητή αποθηκεύεται το κομμάτι μετά την υποδιαστολή της μεταβλητής d. Η τελική τιμή του latitude προκύπτει από το lat[0] βάζοντας την υποδιαστολή και την τιμή ds. Αντίστοιχα υπολογίζεται η τιμή του longitude με τη διαφορά ότι η αρχική μορφή του είναι DDDMM.mm.

Για να υπάρχει ανανέωση των GPS δεδομένων, απαιτείται η χρησιμοποίηση ενός timer, ο οποίος ανά κάποια χρονική στιγμή θα χτυπάει και θα εμφανίζονται οι ανανεωμένες τιμές της τοποθεσίας ενός χρήστη. Οπότε, επιλέγουμε από το Toolbox του Visual Studio την προσθήκη ενός timer στο Form2 και ρυθμίζουμε στις ιδιότητες του timer το Interval: 2000, δηλαδή ανά δύο δευτερόλεπτα να έχουμε timer1_Tick().

Η ενεργοποίηση ή η απενεργοποίηση του GPS εκτελείται με το πάτημα του button και έχουμε τους παρακάτω κώδικες αντίστοιχα:

```
bool btnclicked2 = true;
private void button1_Click(object sender, EventArgs e)
{
    //Enable GPS
    if (btnclicked2)
    {
        // Try to open the serial port
        try
        {
            serialPort1.Open();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            return;
        }
        btnclicked2 = false;
        button1.Text = "Απενεργοποίηση GPS";
        timer1.Enabled = true;
    }
```

```
//Disable GPS
else
{
    map1.Refresh();
    try
    {
        serialPort1.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return;
    }
    btnclicked2 = true;
    button1.Text = "Ενεργοποίηση GPS";
    timer1.Enabled = false;
    txtHead.Text = "";
    txtLong.Text = "";
    txtLat.Text = "";
    txtSpeed.Text = "";
}
```

Στην ενεργοποίηση του GPS ανοίγει η εικονική θύρα COM4 για τη σύνδεση της εφαρμογής με την εφαρμογή που αποστέλλει τα GPS δεδομένα από το κινητό τηλέφωνο και τίθεται ο timer1 ενεργός για να εκτελεί ανά δύο δευτερόλεπτα που είναι ρυθμισμένος ανανέωση της τοποθεσίας

ενός χρήστη. Αντίθετα, στην απενεργοποίηση του GPS η ανοιχτή θύρα COM4 κλείνει, ο timer1 απενεργοποιείται και οι τιμές στα στοιχεία τοποθεσίας του χρήστη απαλείφονται.

Για την εξαγωγή των GPS δεδομένων σε αρχεία shapefile και csv, χρησιμοποιούμε ένα button για την έναρξη και τη λήξη καταγραφής των GPS δεδομένων. Ο κώδικας για την εκτέλεση αυτών των λειτουργιών είναι ο εξής:

```
if (btnclicked)
{

    btnclicked = false;
    button2.Text = "Stop";
    timer1.Enabled = false;
    start_record = true;

    //FeatureSet fs = new FeatureSet(FeatureType.Line);
    fs.DataTable.Columns.Add(new DataColumn("Datetimestamp", typeof(string)));
    fs.DataTable.Columns.Add(new DataColumn("Longitude", typeof(double)));
    fs.DataTable.Columns.Add(new DataColumn("Latitude", typeof(double)));
    fs.DataTable.Columns.Add(new DataColumn("Heading", typeof(string)));
    fs.DataTable.Columns.Add(new DataColumn("Speed", typeof(string)));


    sourceTable.Columns.AddRange(new DataColumn[] {
        new DataColumn("Datetimestamp", typeof(string)),
        new DataColumn("Longitude", typeof(double)),
        new DataColumn("Latitude", typeof(double)),
        new DataColumn("Heading", typeof(string)),
        new DataColumn("Speed", typeof(string))
        });
    timer1.Enabled = true;
```

Σύμφωνα με το παραπάνω κώδικα, όταν πατηθεί το button της ενεργοποίησης των GPS δεδομένων ο timer1 απενεργοποιείται προσωρινά για να δημιουργηθεί ο πίνακας sourcetable, ο οποίος θα αποθηκεύει τις τιμές των Datetimestamp, Longitude, Latitude, Heading και Speed. Επίσης, δημιουργούνται τα πεδία του αρχείου shapefile που θα αποθηκεύει στη συνέχεια. Αφού εκτελεστούν όλα τα προαναφερθέντα, ο timer1 ενεργοποιείται για να καταγράψει τα GPS δεδομένα. Στην επανάληψη εκτέλεσης διαφόρων ενεργειών του timer1 ανά δύο δευτερόλεπτα, αν η μεταβλητή start_record είναι αληθής εκτελείται η καταγραφή των δεδομένων. Ο κώδικας για αυτή την υλοποίηση είναι ο κάτωθι:

```
//Fill sourcetable GPS DATA
if (start_record == true)
{


    sourceTable.Rows.Add(date + "-" + time, plon, plat, heading, speed);


    vertices.Add(new Coordinate(plon, plat));
    vertices.Add(new Coordinate(plon, plat));
    LineString geom = new LineString(vertices);
    // add the geometry to the featureset.
    IFeature feature = fs.AddFeature(geom);
    feature.DataRow.BeginEdit();
    feature.DataRow["Datetimestamp"] = date + "-" + time;
    feature.DataRow["Longitude"] = plon;
    feature.DataRow["Latitude"] = plat;
    feature.DataRow["Heading"] = heading;
    feature.DataRow["Speed"] = speed;
    feature.DataRow.EndEdit();


}
```

Στον παραπάνω κώδικα, παρατηρούμε ότι στον πίνακα sourceTable αποθηκεύονται εγγραφές για τις διάφορες μεταβλητές που αφορούν την τοποθεσία ενός χρήστη. Επιπρόσθετα, παρατηρούμε ότι δημιουργείται μια διανυσματική γεωμετρία γραμμής με σημεία τα vertices στον κώδικα και για το αρχείο shapefile που θα δημιουργηθεί, εγγράφονται οι διάφορες τιμές στα αντίστοιχα πεδία.

Στην περίπτωση λήξης της καταγραφής των GPS δεδομένων, ο timer1 απενεργοποιείται και εν συνεχεία έχουμε δύο αποθηκεύσεις των δεδομένων, πρώτα για αρχείο shapefile και έπειτα για αρχείο csv. Ο κώδικας για την αποθήκευση αρχείου shapefile σε μια διαδρομή της επιλογής μας στον υπολογιστή είναι ο εξής:

```
timer1.Enabled = false;
btnclicked = true;
button2.Text = "Start";
try
{
    using (SaveFileDialog save1 = new SaveFileDialog() { Filter = "SHP|*.shp", ValidateNames = true })
    {
        if (save1.ShowDialog() == DialogResult.OK)
        {
            fs.SaveAs(save1.FileName, true);
        }
    }
    MessageBox.Show("Το αρχείο shp αποθηκεύτηκε με επιτυχία");
}
catch (Exception msg)
{
    MessageBox.Show(msg.ToString());
    throw;
}
```

Ενώ για την αποθήκευση σε αρχείο csv με τη βοήθεια του Rfc4180Writer ως [24] έχουμε τον κάτωθι κώδικα:

```
try
{
    using (SaveFileDialog save2 = new SaveFileDialog() { Filter = "CSV|*.csv", ValidateNames = true })
    {
        if (save2.ShowDialog() == DialogResult.OK)
        {
            using (StreamWriter writer = new StreamWriter(save2.FileName))
            {
                Rfc4180Writer.WriteDataTable(sourceTable, writer, true);
            }
        }
    }
    MessageBox.Show("Το αρχείο csv αποθηκεύτηκε με επιτυχία");
}
catch (Exception msg)
{
    MessageBox.Show(msg.ToString());
    throw;
}
timer1.Enabled = true;
```

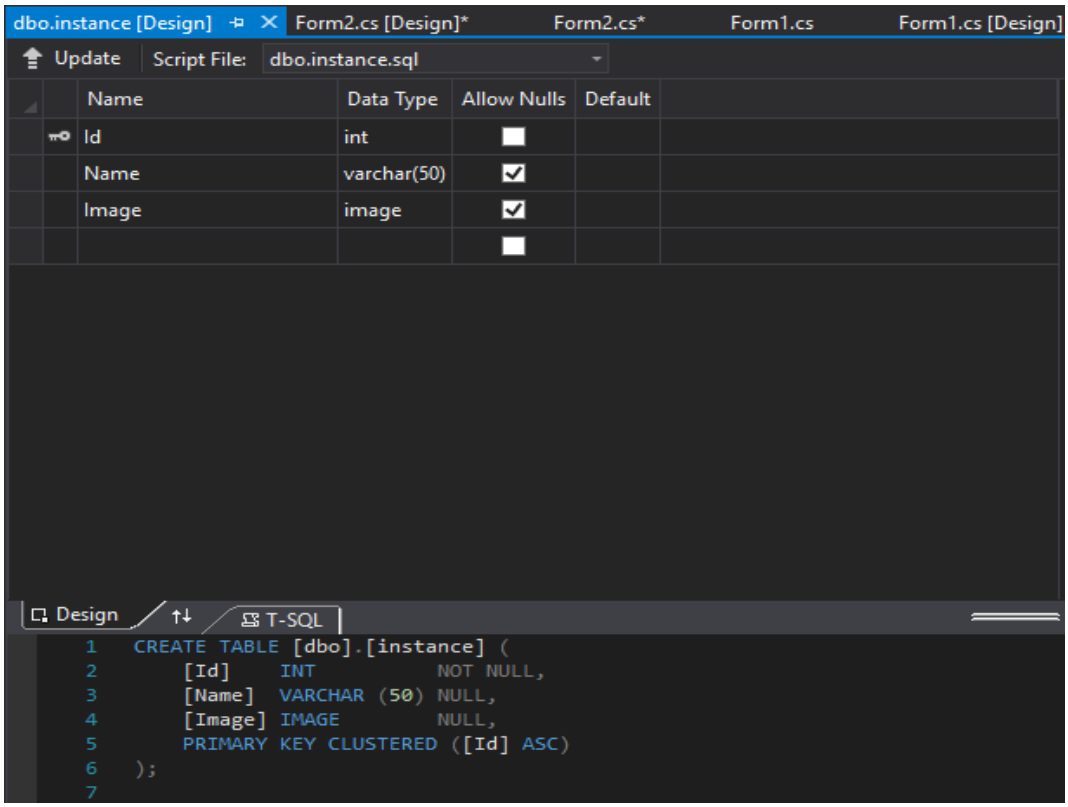
Για την απεικόνιση του στίγματος τοποθεσίας ενός χρήστη στον εκάστοτε χάρτη, απαιτείται το στίγμα θέσης να ανανεώνεται καθώς εκτελείται το timer1_Tick() ανά δύο δευτερόλεπτα. Στην ανανέωση αυτή όμως πρέπει να έχουμε παράλληλα και διαγραφή του σημείου αυτού από τη μνήμη του υπολογιστή έτσι ώστε να αποφευχθεί μία διαρροή μνήμης (memory leak) την οποία δεν θα μπορεί να διαχειριστεί η εφαρμογή μας. Ο κώδικας για την υλοποίηση απεικόνισης του στίγματος θέσης ενός χρήστη διαφαίνεται παρακάτω:

```
//Απεικόνιση στίγματος GPS
_markers = new FeatureSet(FeatureType.Point);
_markerLayer = new DotSpatial.Controls.MapPointLayer(_markers);
_markerLayer.Symbolizer = new PointSymbolizer(Color.Red, DotSpatial.Symbology.PointShape.Diamond, 10);
map1.MapFrame.DrawingLayers.Add(_markerLayer);
Coordinate c = new Coordinate(plon, plat);
_markers.AddFeature(new DotSpatial.Topology.Point(c));
map1.MapFrame.Invalidate();
_markers.Dispose();
```

Όπως απεικονίζεται ο παραπάνω κώδικας για το στίγμα σε ένα οποιοδήποτε χάρτη, δημιουργείται ένα διανυσματικό σημείο (point), το οποίο μορφοποιούμε ως κόκκινο διαμάντι και το προσθέτουμε στον χάρτη που είναι φορτωμένος στην εφαρμογή μας. Η μεταβλητή c λαμβάνει ως καρτεσιανά μεγέθη το longitude και latitude, τα οποία είχαν υπολογισθεί προγενέστερα και το σημείο που δημιουργήσαμε με τιμή την τιμή της μεταβλητής c εμφανίζεται στο map1. Στο τέλος, το σημείο αυτό διαγράφεται από τη μνήμη αλλιώς θα είχαμε αμέτρητα σημεία στον χάρτη τα οποία θα είχαν δημιουργηθεί και η μνήμη του υπολογιστή μας κάποια στιγμή θα γέμιζε από τις τιμές αυτές.

5.2.2. Η καρτέλα Pictures

Για την καρτέλα των εικόνων, υπάρχουν τέσσερα buttons, τα οποία εξυπηρετούν ξεχωριστές λειτουργίες το καθένα. Συγκεκριμένα, είναι η αναζήτηση εικόνων από τον υπολογιστή μας, η αποθήκευση καινούργιων εικόνων στη βάση δεδομένων content, η προβολή ήδη αποθηκευμένων εικόνων από τη βάση δεδομένων content και ο καθαρισμός των στοιχείων που έχουν εισαχθεί στο ID ή το Name της εικόνας. Η βάση δεδομένων content αποτελείται από τον πίνακα instance και ο πίνακας αυτός εμπεριέχει το ID, το Name και την Image. Ο πίνακας της βάσης απεικονίζεται παρακάτω:



Εικόνα 46- Δημιουργία του πίνακα instance στο Visual Studio.

Για την αναζήτηση των εικόνων από τον τοπικό μας υπολογιστή υλοποιείται ο παρακάτω κώδικας στον οποίο ανοίγει το παράθυρο από τα Windows για την επιλογή ενός αρχείου εικόνας (jpeg, png) και αφού επιλεχθεί φορτώνεται στο picturebox της καρτέλας εικόνων.

```
//Αναζήτηση εικόνων
private void button3_Click(object sender, EventArgs e)
{
    OpenFileDialog dialog = new OpenFileDialog();
    dialog.Filter = "png files(*.png)|*.png|jpg files(*.jpg)|*.jpg|All files(*.*)|*.*";
    if (dialog.ShowDialog() == DialogResult.OK)
    {
        imgLocation = dialog.FileName.ToString();
        pictureBox1.ImageLocation = imgLocation;
    }
}
```

Για την αποθήκευση των εικόνων στον πίνακα instance της βάσης δεδομένων content υλοποιείται ο παρακάτω κώδικας:

```
FileStream Streem = new FileStream(imgLocation, FileMode.Open, FileAccess.Read);
BinaryReader brs = new BinaryReader(Streem);
images = brs.ReadBytes((int)Streem.Length);
SqlCommand cmdCount = new SqlCommand("SELECT count(*) from instance WHERE Id = '" + textID.Text + "'", conection);
int count = (int)cmdCount.ExecuteScalar();
if (count > 0)
{
    string sqlQuery = "UPDATE instance SET  Name=@NAME , Image=@images WHERE Id = '" + textID.Text + "'";
    cmd = new SqlCommand(sqlQuery, conection);
    cmd.Parameters.Add(new SqlParameter("@images", images));
    cmd.Parameters.Add(new SqlParameter("@NAME", textName.Text));
    cmd.ExecuteNonQuery();
    conection.Close();
    MessageBox.Show(" Data Saved Successfully on your previous photo.....!");
    imgLocation = "";

}
else
{
    string sqlQuery = "INSERT INTO instance (Id,Name,Image) VALUES ('" + textID.Text + "','" + textName.Text + "', @images)";
    cmd = new SqlCommand(sqlQuery, conection);
    cmd.Parameters.Add(new SqlParameter("@images", images));
    int N = cmd.ExecuteNonQuery();
    conection.Close();
    MessageBox.Show(N.ToString() + " Data Saved Successfully.....!");
    imgLocation = "";
}
```

Από τον παραπάνω κώδικα διαπιστώνουμε ότι μια φωτογραφία που εισαγάγαμε από την αναζήτηση, έχουμε τη δυνατότητα να την αποθηκεύσουμε με νέο ID ή να ανανεώσουμε μια ήδη υπάρχουσα φωτογραφία με μια νέα.

Για την προβολή μιας εικόνας από τον πίνακα της βάσης δεδομένων content ο κώδικας παρουσιάζεται παρακάτω:

```
//Προβολή εικόνων
private void button5_Click(object sender, EventArgs e)
{
    conection.Open();
    string sqlQuery = "select Name,Image from instance where Id='" + textID.Text + "'";

    cmd = new SqlCommand(sqlQuery, conection);
    SqlDataReader Dataread = cmd.ExecuteReader();
    Dataread.Read();
    if (Dataread.HasRows)
    {
        textName.Text = Dataread[0].ToString();
        byte[] images = (byte[])Dataread[1];

        if (images == null)
        {
            pictureBox1.Image = null;
        }

        else
        {
            MemoryStream mstream = new MemoryStream(images);
            pictureBox1.Image = Image.FromStream(mstream);
        }
    }
    else
    {
        MessageBox.Show("This Data Not Available");
    }
    conection.Close();
}
```

Παρατηρούμε από τον παραπάνω κώδικα ότι προβάλουμε μία εικόνα με βάση το ID που έχουμε πληκτρολογήσει ως χρήστες και στην περίπτωση που το ID αυτό δεν είναι καταχωρημένο, μας ειδοποιεί η εφαρμογή ότι τα δεδομένα αυτά δεν υπάρχουν.

Τέλος, για τον καθαρισμό των δεδομένων που έχουν εισαχθεί στα πεδία ID και Name της καρτέλας ή τον καθαρισμό μιας εικόνας που έχει φορτωθεί στην καρτέλα Pictures της εφαρμογής, υλοποιείται ο παρακάτω κώδικας κατά τον οποίο απαλείφονται όλες οι τιμές στα πεδία ID, Name και imgLocation.

```
//Καθαρισμός
private void button6_Click(object sender, EventArgs e)
{
    textID.Text = null;
    pictureBox1.Image = null;
    textName.Text = null;
    imgLocation = "";
}
```

5.2.3. Η καρτέλα Login/Logout

Όσον αφορά την αποσύνδεση του χρήστη στην καρτέλα Login/Logout, πατώντας το button της αποσύνδεσης κλείνει η εικονική θύρα COM4 στην περίπτωση που ήταν ανοιχτή, συλλέγονται τυχόν άχρηστα δεδομένα ή δεδομένα που δεν χρησιμοποιούνται και μεταφερόμαστε στο παράθυρο της Σύνδεσης ενός χρήστη. Ο κώδικας για τα προαναφερθέντα είναι:

```
//Αποσύνδεση χρήστη
private void button7_Click(object sender, EventArgs e)
{
    conection.Close();
    if (serialPort1.IsOpen)
    {
        serialPort1.Close();
    }
    this.Dispose();
    GC.Collect();
    GC.WaitForPendingFinalizers();
    GC.Collect();

    Form1 form1 = new Form1();
    form1.Show();

}
```

6. Περιγραφή Λειτουργικότητας

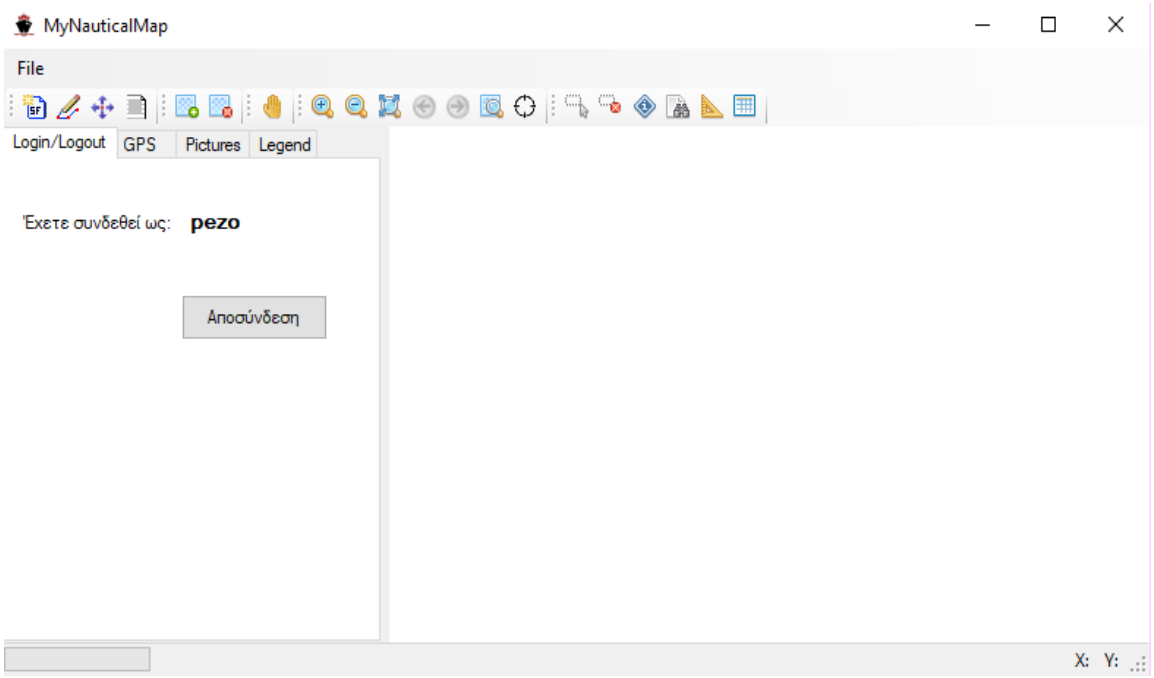
Στο κεφάλαιο αυτό θα γίνει εκτενής ανάλυση των λειτουργιών της εφαρμογής MyNauticalMap, θα παρουσιαστούν οι δυνατότητές της και θα επαληθευθούν διάφορα αποτελέσματά της ως προς την ορθότητα. Όπως αναφέρθηκε και στην ανάλυση απαιτήσεων της εφαρμογής, αρχικά με το άνοιγμα του εκτελέσιμου αρχείου της, θα εμφανιστεί το παράθυρο εισαγωγής των στοιχείων σύνδεσης του χρήστη (ενός πλοίου). Αφού τα στοιχεία ελεγχθούν σύμφωνα με τα στοιχεία, τα οποία είναι εγγεγραμμένα στη βάση δεδομένων personel, τότε ο χρήστης μεταφέρεται στο κύριο παράθυρο της εφαρμογής. Για την είσοδο στο παράθυρο της κύριας εφαρμογής έχουν καταχωρηθεί δύο χρήστες, οι οποίοι μπορούν να τη διαχειριστούν. Οι χρήστες απεικονίζονται στον παρακάτω πίνακα:

Πίνακας 2- Χρήστες εφαρμογής

Id	username	password
1	pezo	12345
2	bles	123456

Έστω ότι ανοίγουμε το εκτελέσιμο αρχείο της εφαρμογής και εισάγουμε τα στοιχεία με username: pezo. Τα παράθυρα που θα εμφανιστούν προβάλλονται παρακάτω:

Εικόνα 47- Το παράθυρο σύνδεσης ενός χρήστη στην εφαρμογή.



Εικόνα 48- Το παράθυρο της κύριας εφαρμογής MyNauticalMap.

Παρατηρούμε στο παράθυρο της κύριας εφαρμογής ότι προβάλλεται αρχικά η καρτέλα Login/Logout, στην οποία φαίνεται το username με το οποίο συνδεθήκαμε στο παράθυρο της σύνδεσης. Παράλληλα σε αυτή την καρτέλα έχουμε τη δυνατότητα να αποσυνδεθούμε από την κύρια εφαρμογή πατώντας το button της αποσύνδεσης.

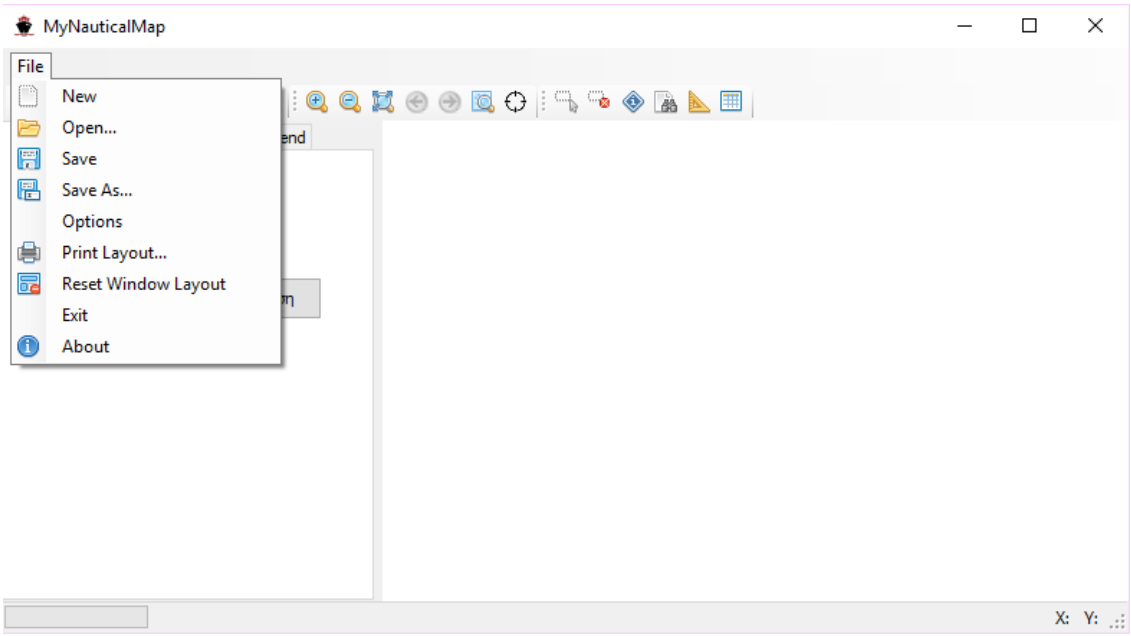
Επιπρόσθετα, παρατηρούμε άλλες τρεις καρτέλες GPS, Pictures και Legend. Η καρτέλα του GPS είναι για την ενεργοποίηση/απενεργοποίηση του GPS, τη λήψη GPS δεδομένων, την

αποθήκευση αυτών σε αρχεία shapefile ή/και csv και την απεικόνιση του στίγματος θέσης σε ένα χάρτη που φορτώθηκε από εμάς στο δεξί μέρος της εφαρμογής. Η καρτέλα Pictures περιλαμβάνει τις λειτουργίες αναζήτησης μιας εικόνας από τον τοπικό μας υπολογιστή, την αποθήκευση και την προβολή εικόνων σε και από μια βάση δεδομένων content και καθαρισμό των εικόνων που έχουν φορτωθεί στο πρόγραμμά μας. Η καρτέλα Legend αποτελεί το υπόμνημα των επιπέδων (layers) που φορτώνονται στο δεξί μέρος της εφαρμογής (map1).

Στο κατώτατο σημείο της εφαρμογής δεξιά έχουμε τις συντεταγμένες X, Y που απεικονίζονται βάζοντας τον κέρσορα στον χάρτη στο σημείο που δείχνει. Οι συντεταγμένες αυτές προβάλλονται σε δεκαδικές μοίρες (decimal degrees). Αντίθετα, στην αριστερή μεριά απεικονίζεται η μπάρα φόρτωσης επιπέδων (layers) στην εφαρμογή καθώς και η δημιουργία του σημείου του στίγματος από το GPS ανά δύο δευτερόλεπτα.

Στο πάνω μέρος της εφαρμογής απεικονίζεται η γραμμή εργαλείων η οποία αποτελείται από πέντε ομαδοποιημένες λειτουργίες. Από αριστερά προς τα δεξιά έχουμε πρώτα την ομάδα με τη δημιουργία διανυσματικών δεδομένων (point, line, polygon) πάνω σε ένα χάρτη, στη δεύτερη ομάδα έχουμε την εισαγωγή/αφαίρεση επιπέδων στην εφαρμογή, στην τρίτη ομάδα έχουμε τη διαχείριση ενός χάρτη (pan), στην τέταρτη ομάδα έχουμε τις λειτουργίες zoom in/zoom out ενώ στην τελευταία ομάδα έχουμε την προβολή του πίνακα ιδιοτήτων ενός επιπέδου (attribute table), τη μέτρηση αποστάσεων στον χάρτη και την εύρεση ενός ID στον χάρτη.

Λίγο πιο πάνω από τη γραμμή εργαλείων έχουμε το μενού File που αν το αναπτύξουμε έχουμε τις κάτωθι επιλογές όπως διαφαίνονται παρακάτω:

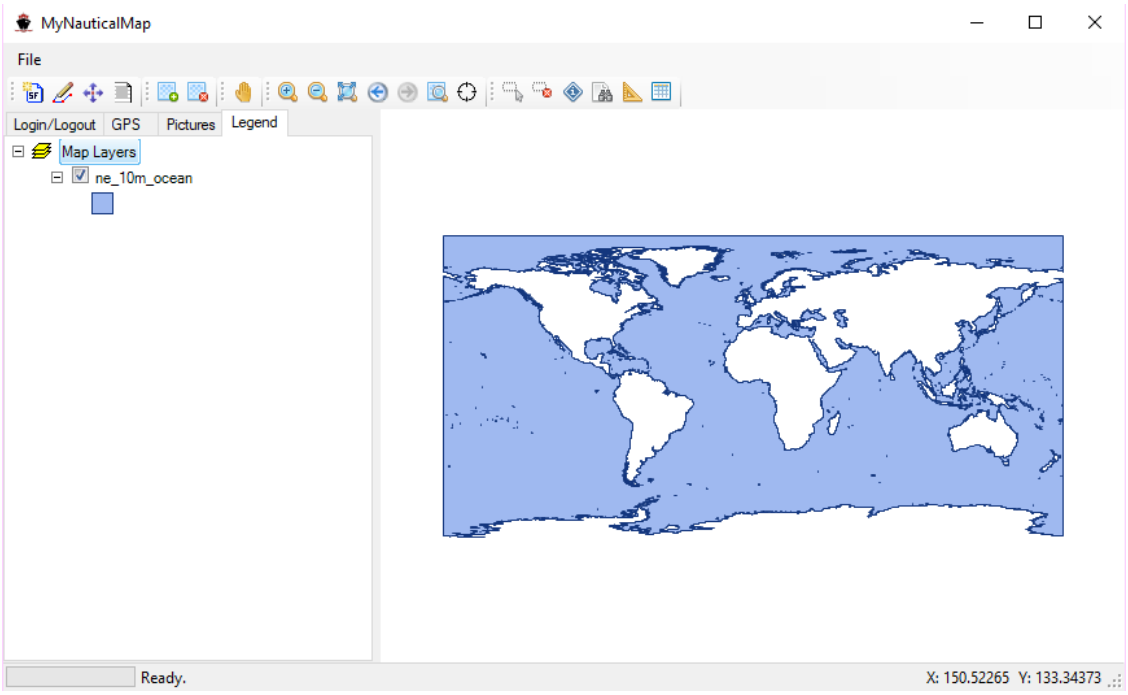


Εικόνα 49- Εξερεύνηση επιλογών μενού File.

Όπως παρατηρούμε παραπάνω, έχουμε τη δυνατότητα δημιουργίας μιας νέας εργασίας, ανοίγματος μιας υπάρχουσας εργασίας, αποθήκευσης μιας εργασίας που κάναμε σε αρχείο dspx, προβολή ιδιοτήτων, εκτύπωση ενός χάρτη, επανεκκίνηση του παραθύρου, έξοδος από την εφαρμογή και προβολή πληροφοριών σχετικά με την εφαρμογή.

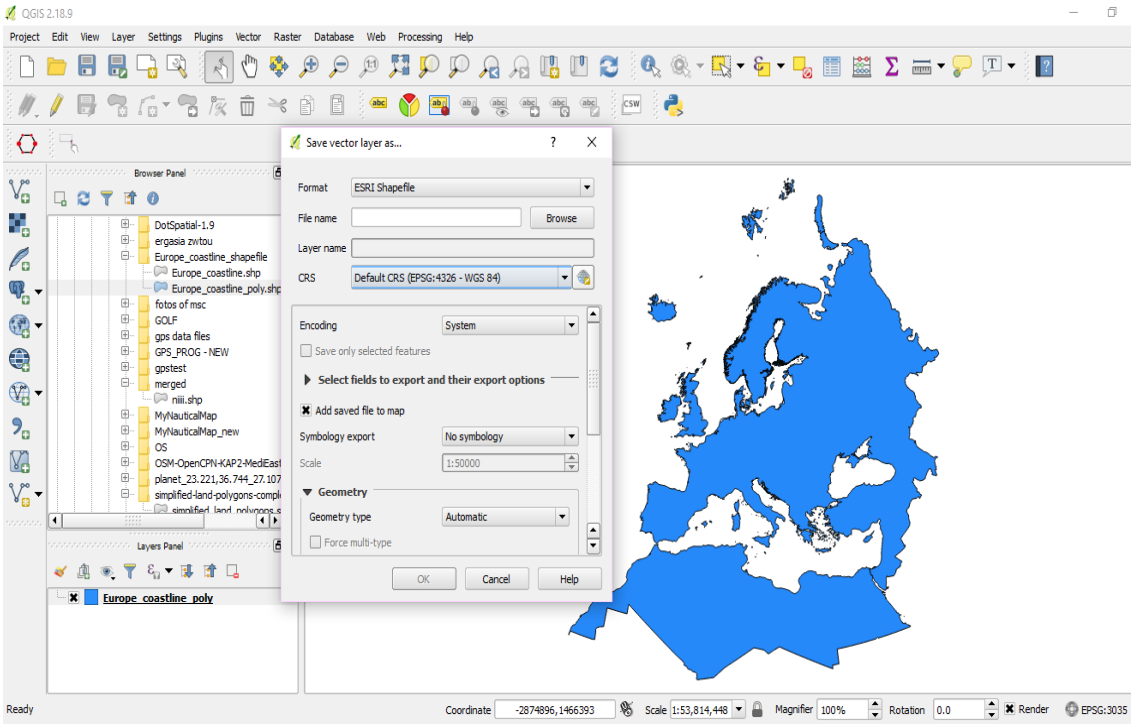
6.1. Γενικές λειτουργίες εφαρμογής

Έστω τώρα ότι θέλουμε να εισάγουμε ένα χάρτη στην εφαρμογή μας. Ο χάρτης που θα εισάγουμε είναι παγκόσμιος και απεικονίζει τους ωκεανούς σε όλο τον κόσμο. Διανέμεται δωρεάν από την ιστοσελίδα της Natural Earth ως [25] και υποστηρίζεται από την NACIS (North American Cartographic Information Society). Η προβολή του χάρτη αυτού γίνεται σε σύστημα συντεταγμένων με κωδικό EPSG:4326. Αφού πατήσουμε Add Layer και επιλέξουμε το αρχείο shapefile που κατεβάσαμε από την ιστοσελίδα εμφανίζεται το παρακάτω παράθυρο.



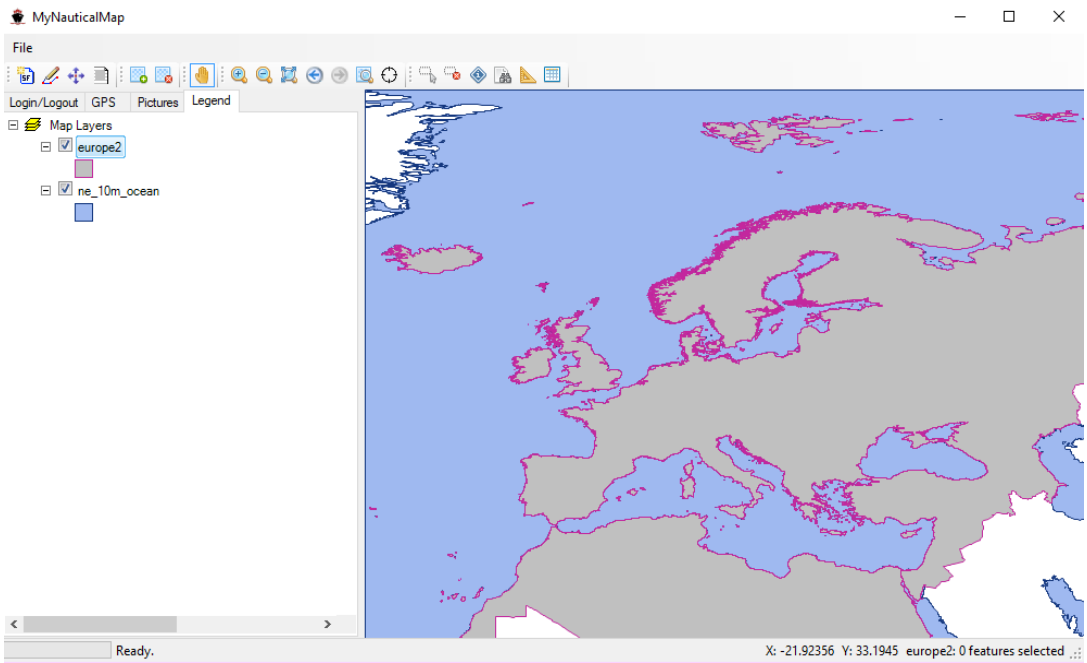
Εικόνα 50- Φόρτωση παγκόσμιου χάρτη για τους ωκεανούς στην εφαρμογή.

Στη συνέχεια, θα εισαγάγουμε έναν ευρωπαϊκό χάρτη, ο οποίος έχει μεγαλύτερη ακρίβεια για τη μελέτη περιοχών στην συγκεκριμένη ήπειρο και διανέμεται δωρεάν από τη European Environment Agency σε ευρωπαϊκό σύστημα συντεταγμένων με κωδικό EPSG:3035 ως [26]. Επειδή το GPS χρησιμοποιεί κατ' εξοχήν δεκαδικές μοίρες (decimal degrees) στο παγκόσμιο σύστημα συντεταγμένων (EPSG:4326), ο ευρωπαϊκός χάρτης θα μετατραπεί από ευρωπαϊκό σύστημα συντεταγμένων σε παγκόσμιο μέσω της εφαρμογής QGIS. Η διαδικασία έχει ως εξής: αφού ανοίξουμε το QGIS, φορτώνουμε το αρχείο shapefile στο πρόγραμμα και αποθηκεύουμε το αρχείο με σύστημα συντεταγμένων το παγκόσμιο. Το QGIS μετατρέπει το παλιό αρχείο shapefile με EPSG:3035 σε νέο αρχείο με EPSG:4326. Η διαδικασία απεικονίζεται παρακάτω:



Εικόνα 51- Μετατροπή συστήματος συντεταγμένων του χάρτη στο QGIS.

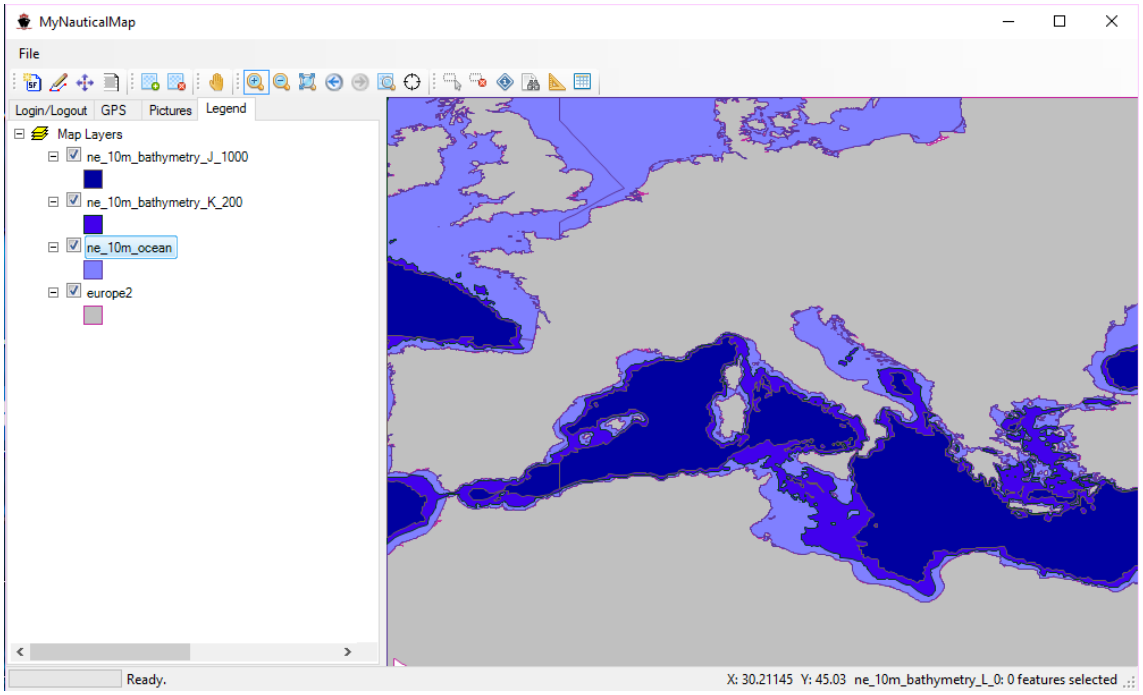
Αφού εκτελέσουμε την ανωτέρω διαδικασία για τον ευρωπαϊκό χάρτη, τον εισάγουμε στην εφαρμογή μας ως δεύτερο επίπεδο (layer), πατώντας Add Layer στη γραμμή εργαλείων.



Εικόνα 52- Φόρτωση ευρωπαϊκού χάρτη στην εφαρμογή.

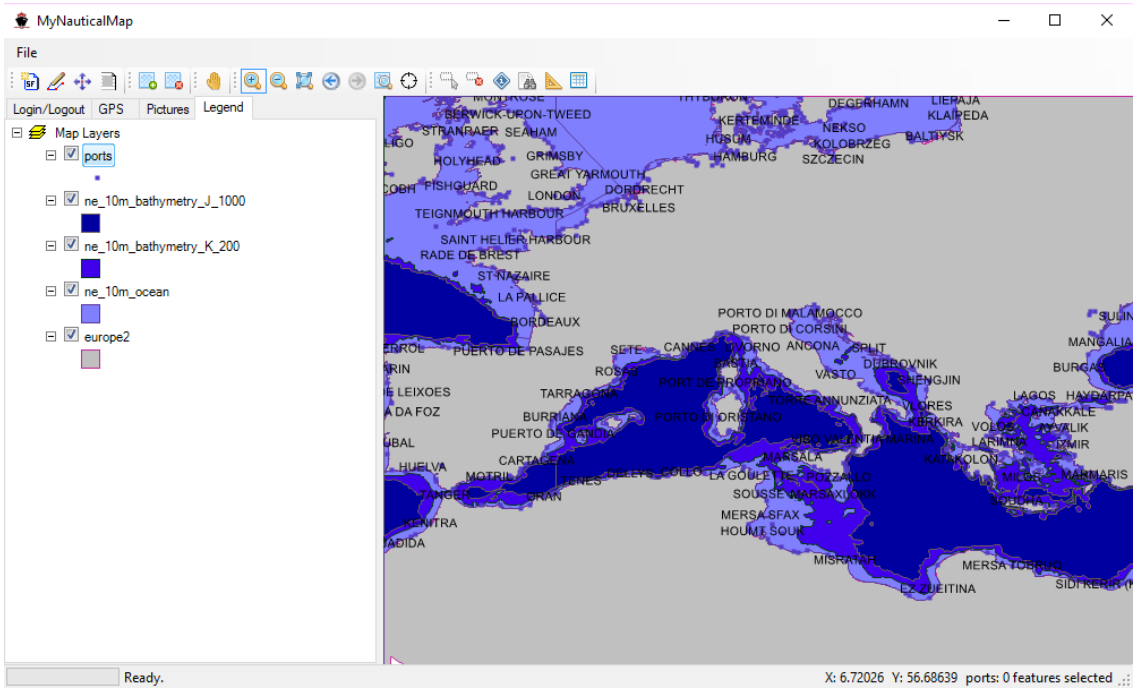
Διακρίνουμε από την παραπάνω εικόνα την ακρίβεια του ευρωπαϊκού χάρτη σε σχέση με τον παγκόσμιο και η διαφορά φαίνεται καλύτερα στη μεγαλύτερη μεγέθυνση διότι το επίπεδο του ευρωπαϊκού χάρτη καλύπτει μια συγκεκριμένη περιοχή του επιπέδου του παγκόσμιου χάρτη. Η επιλογή ενός παγκόσμιου χάρτη με πολύ μεγάλη ακρίβεια αυτομάτως βελτιώνει πάρα πολύ την ακρίβεια ειδικά για πολύ μεγεθυμένες περιοχές, όμως αυξάνει σε μεγάλο ποσοστό την μνήμη του υπολογιστή, την οποία διαχειρίζεται με αποτέλεσμα η εφαρμογή MyNauticalMap να καθυστερεί στην εκτέλεση εντολών από τον χρήστη.

Έπειτα, θα εντάξουμε γεωγραφικά δεδομένα που σχετίζονται με το θαλάσσιο περιβάλλον. Αρχικά, θα λάβουμε υπόψιν τα βάθη των θαλασσών. Το dataset για τα βάθη των θαλασσών αντλήθηκαν από το Natural Earth ως [25] και αφορά βάθη από 0 έως 10.000 μέτρα. Θα εισαγάγουμε τα επίπεδα (layers) για τα βάθη των 200 και 1.000 μέτρων. Η απεικόνιση των συγκεκριμένων βαθών της θάλασσας φαίνεται παρακάτω:



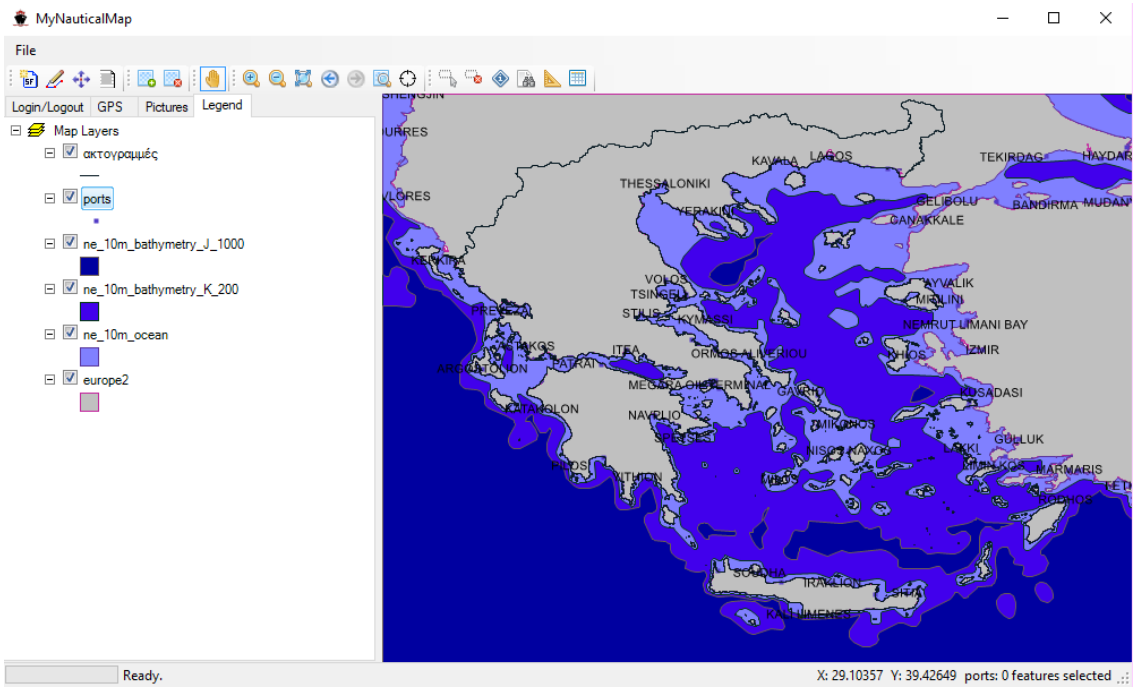
Εικόνα 53- Απεικόνιση βαθών της θάλασσας.

Στον παραπάνω χάρτη, θα προσθέσουμε παγκοσμίως τα κυριότερα λιμάνια, τα οποία μπορούμε να τα κατεβάσουμε δωρεάν ως αρχείο shapefile από την ιστοσελίδα της NGA (National Geospatial-Intelligence Agency) ως [27]. Αφού προσθέσουμε το αρχείο shapefile στην εφαρμογή, για να εμφανιστούν τα ονόματα των λιμανιών στον χάρτη, πρέπει με δεξί κλικ στο επίπεδο ports να επιλέξουμε Labeling-> Label Setup και αφού μας εμφανιστεί το μενού για τις ιδιότητες του Label Setup να επιλέξουμε στο Field Names τη μεταβλητή PORT_NAME. Ο χάρτης θα μετατραπεί όπως απεικονίζεται παρακάτω:



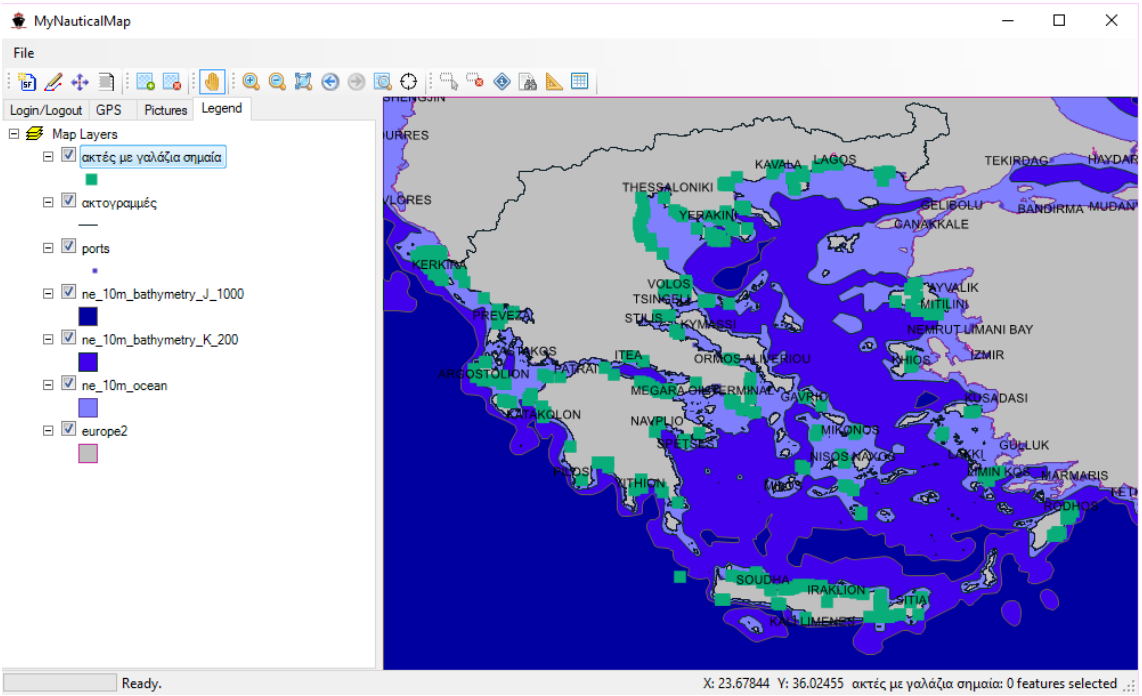
Εικόνα 54- Απεικόνιση κυριότερων λιμανιών στον χάρτη της εφαρμογής.

Συνεχίζοντας στον παραπάνω χάρτη, θα εστιάσουμε στην Ελλάδα όπου θα εισαγάγουμε δεδομένα από την ιστοσελίδα του GEODATA.gov.gr ως [28] και αφορούν την ακτογραμμή και τις ακτές με γαλάζια σημαία που υπάρχουν στην Ελλάδα. Αφού εισαγάγουμε το shapefile με EPSG:4326 στην εφαρμογή μας έχουμε το κάτωθι αποτέλεσμα:



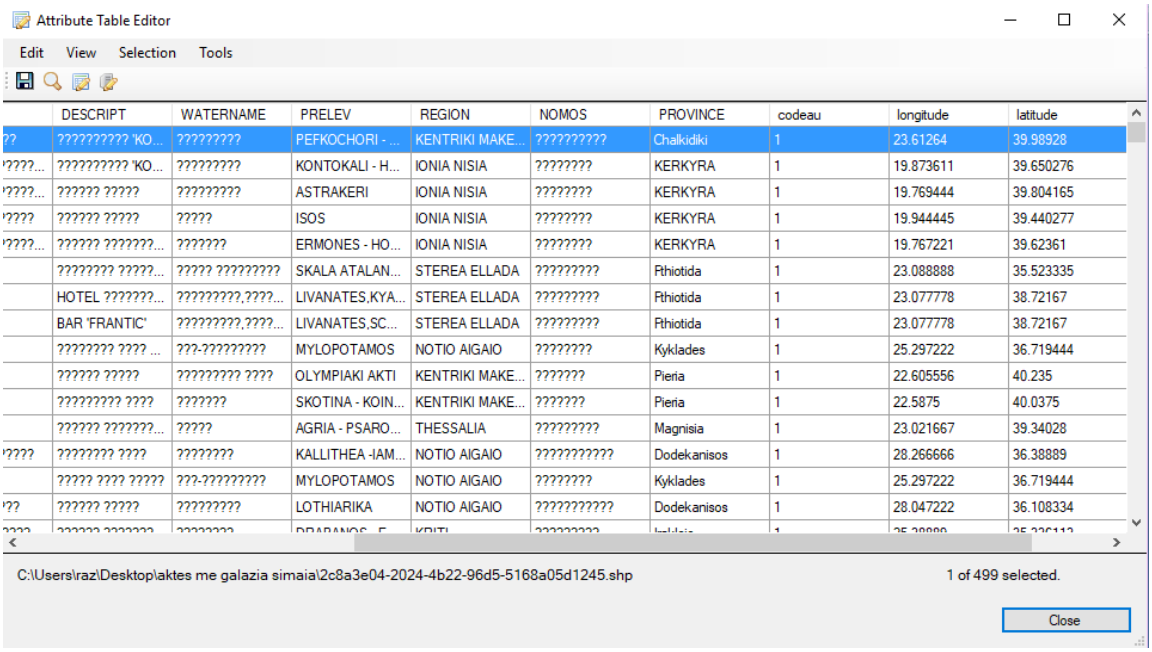
Εικόνα 55- Απεικόνιση ακτογραμμής Ελλάδας στον χάρτη της εφαρμογής.

Προσθέτουμε επίσης και τις ακτές με τη γαλάζια σημαία στην Ελλάδα και έχουμε το εξής αποτέλεσμα:



Εικόνα 56- Απεικόνιση ακτών με γαλάζια σημαία στην Ελλάδα.

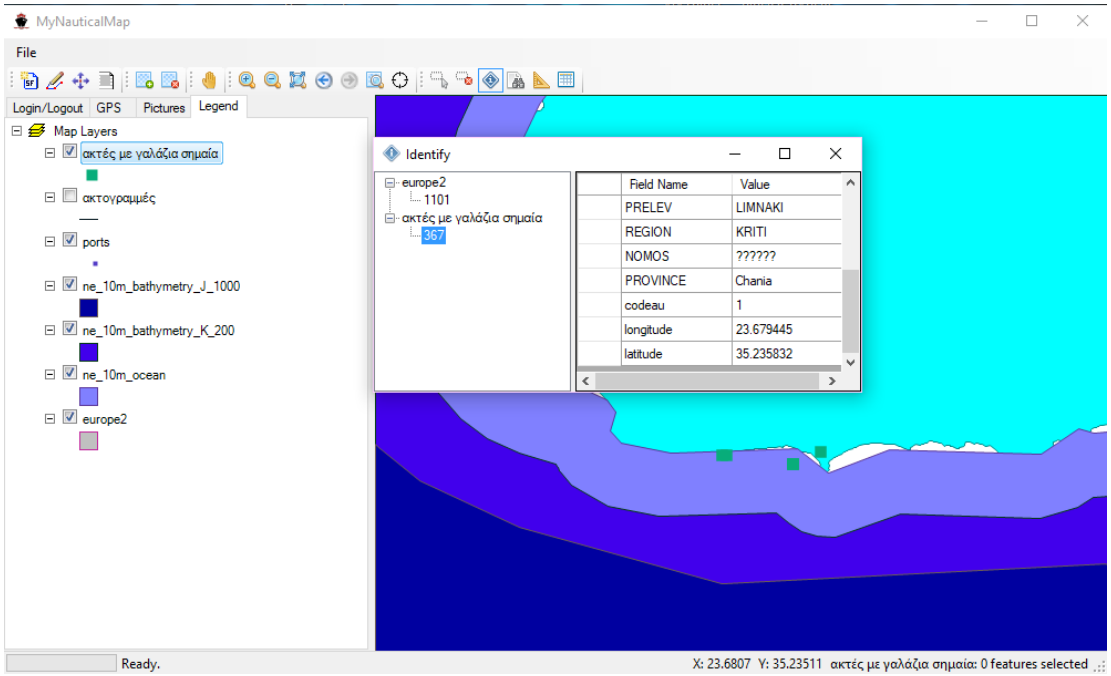
Το σύνολο των δεδομένων για τις ακτές με γαλάζιες σημαίες στην Ελλάδα, μπορούμε να το δούμε επιλέγοντας το συγκεκριμένο επίπεδο (layer) και πατώντας το View Attribute Table στη γραμμή εργαλείων. Το αποτέλεσμα απεικονίζεται παρακάτω:



	DESCRIPT	WATERNAME	PRELEV	REGION	NOMOS	PROVINCE	codeau	longitude	latitude
??	?????????? 'KO...	?????????	PEFKOCHORI - ...	KENTRIKI MAKE...	??????????	Chalkidiki	1	23.61264	39.98928
'????...	?????????? 'KO...	?????????	KONTOKALI - H...	IONIA NISIA	????????	KERKYRA	1	19.873611	39.650276
'????...	?????? ?????	?????????	ASTRAKERI	IONIA NISIA	????????	KERKYRA	1	19.769444	39.804165
'????	?????? ?????	?????	ISOS	IONIA NISIA	????????	KERKYRA	1	19.944445	39.440277
'????...	?????? ??????...	???????	ERMONES - HO...	IONIA NISIA	????????	KERKYRA	1	19.767221	39.62361
	???????? ?????...	????? ?????????	SKALA ATALAN...	STEREA ELLADA	?????????	Fthiotida	1	23.088888	35.523335
	HOTEL ???????...	?????????,????...	LIVANATES,KYA...	STEREA ELLADA	?????????	Fthiotida	1	23.077778	38.72167
	BAR 'FRANTIC'	?????????,????...	LIVANATES,SC...	STEREA ELLADA	?????????	Fthiotida	1	23.077778	38.72167
	???????? ???? ...	???-?????????	MYLOPOTAMOS	NOTIO AIGAIO	????????	Kyklades	1	25.297222	36.719444
	?????? ?????	????????? ????	OLYMPIAKI AKTI	KENTRIKI MAKE...	???????	Pieria	1	22.605556	40.235
	????????? ????	???????	SKOTINA - KOIN...	KENTRIKI MAKE...	???????	Pieria	1	22.5875	40.0375
	?????? ??????...	?????	AGRIA - PSARO...	THESSALIA	?????????	Magnisia	1	23.021667	39.34028
'????	???????? ????	????????	KALLITHEA -IAM...	NOTIO AIGAIO	?????????????	Dodekanisos	1	28.266666	36.38889
	????? ???? ?????	???-?????????	MYLOPOTAMOS	NOTIO AIGAIO	????????	Kyklades	1	25.297222	36.719444
'??	?????? ?????	?????????	LOTHIARIKA	NOTIO AIGAIO	?????????????	Dodekanisos	1	28.047222	36.108334

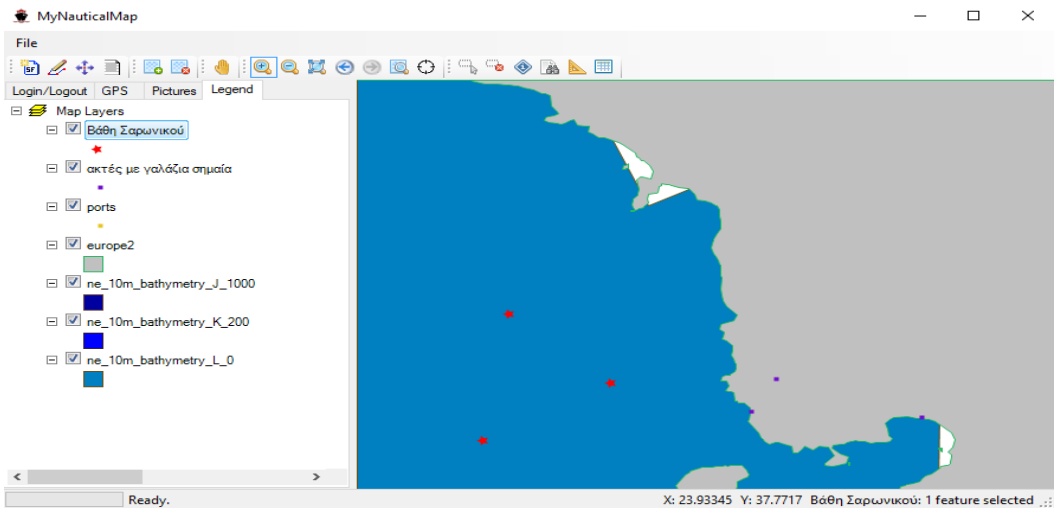
Εικόνα 57- Πίνακας ιδιοτήτων (attribute table) των ακτών με γαλάζιες σημαίες Ελλάδος.

Ακόμη, έχουμε τη δυνατότητα να επιλέξουμε μια ακτή με γαλάζια σημαία πάνω στον χάρτη της εφαρμογής και να αναγνωρίσουμε ποια είναι. Παραδείγματος χάριν, εστιάζουμε στο Νομό Χανίων νότια και αφού επιλέξουμε το εργαλείο Identify από τη γραμμή εργαλείων, πατάμε στην ακτή που επιθυμούμε. Αυτομάτως η εφαρμογή μας παρουσιάζει με FID0: 367 τα στοιχεία για αυτή την ακτή. Το αποτέλεσμα παρουσιάζεται παρακάτω:



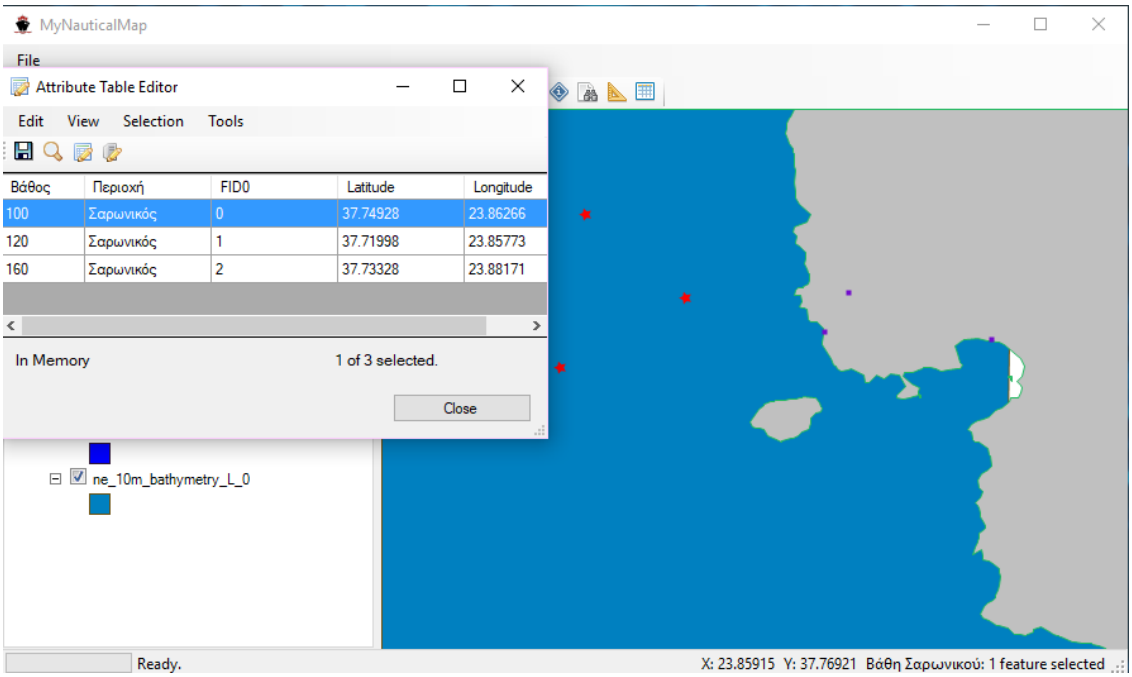
Εικόνα 58- Στοιχεία επιλεγμένης ακτής με γαλάζια σημαία.

Έστω ότι είμαστε ένα ερευνητικό πλοίο το οποίο καταγράφει τα βάθη του Σαρωνικού κόλπου. Για την εισαγωγή σημείων στο χάρτη, επιλέγουμε στην γραμμή εργαλείων New-> Point και στη συνέχεια πατάμε Add Shape όπου επιλέγουμε πάνω στον χάρτη που θέλουμε να τοποθετήσουμε τη μέτρησή μας. Το επίπεδο (layer) που δημιουργήσαμε το μετονομάζουμε ‘Βάθη Σαρωνικού’. Το αποτέλεσμα φαίνεται παρακάτω:



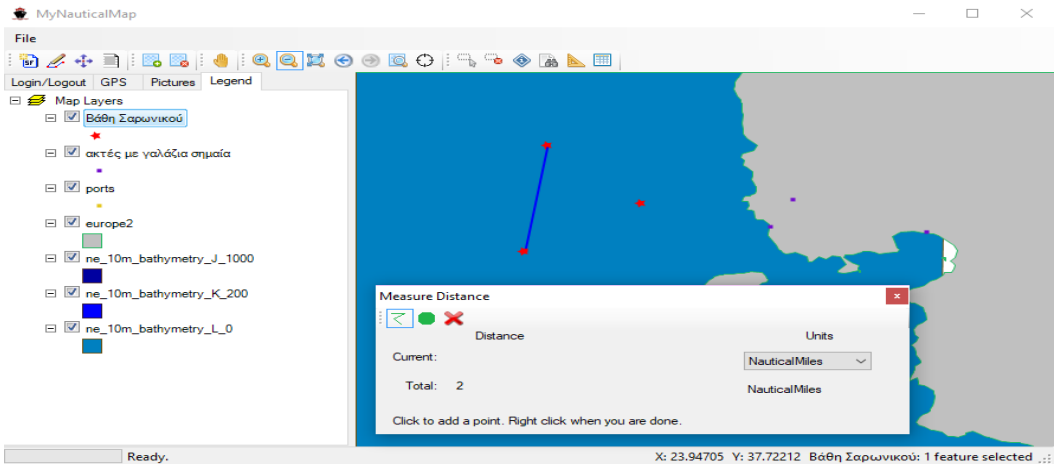
Εικόνα 59- Εισαγωγή σημείων στον χάρτη.

Έχουμε τη δυνατότητα να αλλάξουμε τη μορφή των σημείων σε αστέρια πατώντας δεξί κλικ στο επίπεδο Βάθη Σαρωνικού και επιλέγοντας το Properties. Πατώντας δύο φορές στο Symbol ανοίγει ένα νέο παράθυρο στο οποίο επιλέγουμε ως style το αστέρι. Επίσης, μπορούμε για το σημείο που εισαγάγαμε να επεξεργαστούμε τον πίνακα ιδιοτήτων του (attribute table). Παραδείγματος χάριν, να εισάγουμε νέα πεδία ή να διαγράψουμε υπάρχοντα πεδία και να ανανεώσουμε υπάρχουσες τιμές στα πεδία. Για τα τρία σημεία στα οποία έχει καταγραφεί το βάθος της θάλασσας απεικονίζεται παρακάτω:



Εικόνα 60- Ο πίνακας ιδιοτήτων των τριών σημείων.

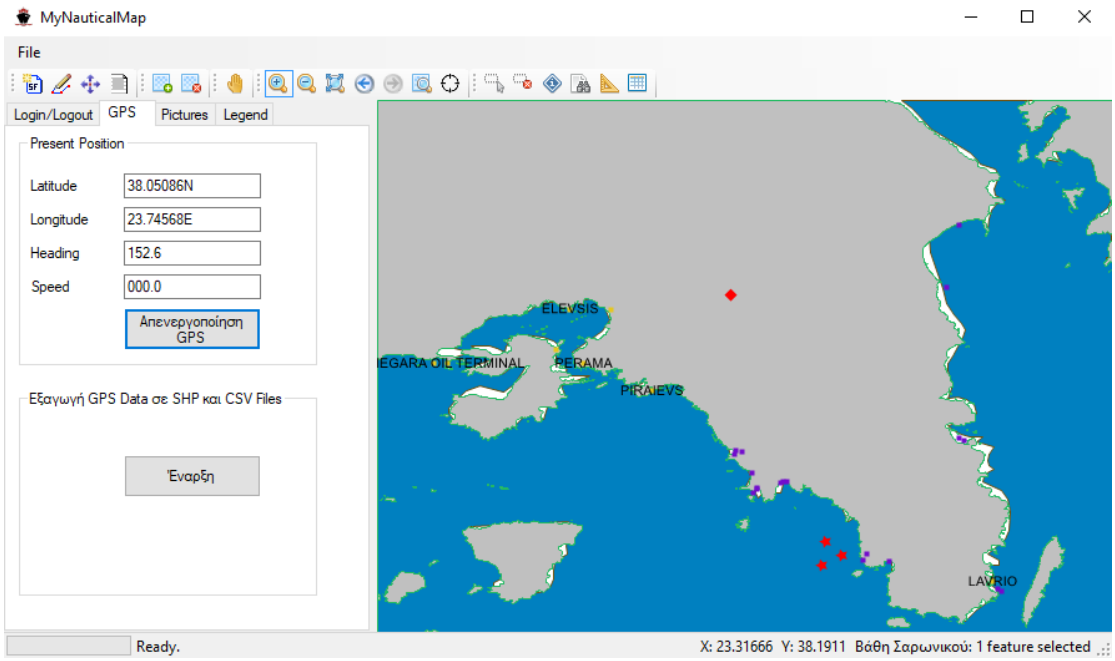
Τέλος, έχουμε τη δυνατότητα να μετρήσουμε μια απόσταση π.χ. μεταξύ δύο σημείων όπως στην παραπάνω εικόνα σε διάφορες μονάδες μέτρησης. Η απόσταση δύο σημείων σε ναυτικά μίλια υπολογίζεται πατώντας το Measure στη γραμμή εργαλείων και αφού επιλεγεί η μονάδα μέτρησης που επιθυμούμε, δημιουργούμε μια γραμμή από το ένα σημείο στο άλλο. Το συγκεκριμένο παράδειγμα απεικονίζεται παρακάτω:



Εικόνα 61- Ο πίνακας ιδιοτήτων των τριών σημείων.

6.2. Λειτουργία GPS

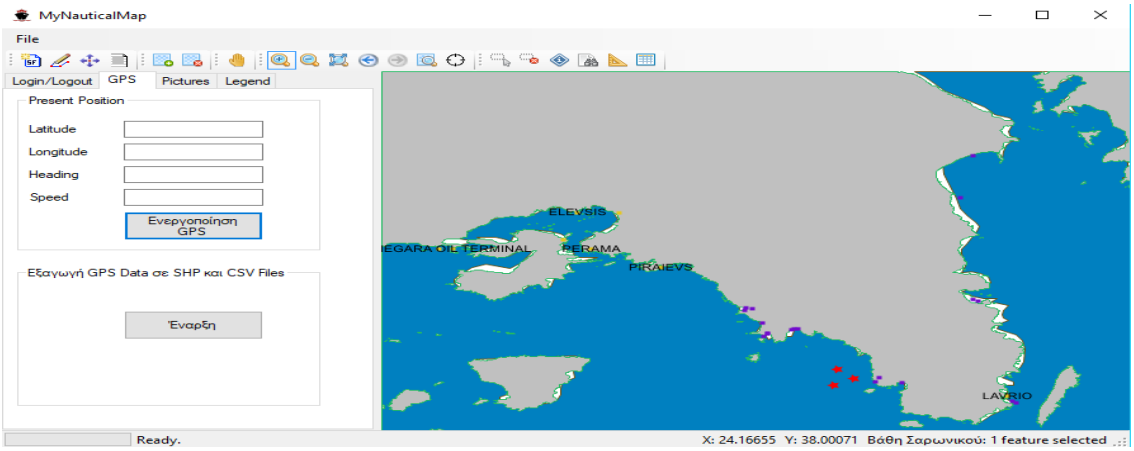
Όσον αφορά τη λειτουργία του GPS, επιλέγοντας την καρτέλα του GPS, όπως αναφέρθηκε στο κεφάλαιο 5.2.1 αφού ενεργοποιήσουμε τη λήψη GPS δεδομένων και ανοίξουμε την εφαρμογή Bluetooth GPS Output από το κινητό μας τηλέφωνο, πατάμε το button ενεργοποίησης στην εφαρμογή μας και με δεδομένο τον χάρτη στο κεφάλαιο 6.1 έχουμε το παρακάτω αποτέλεσμα:



Εικόνα 62- Ενεργοποίηση GPS.

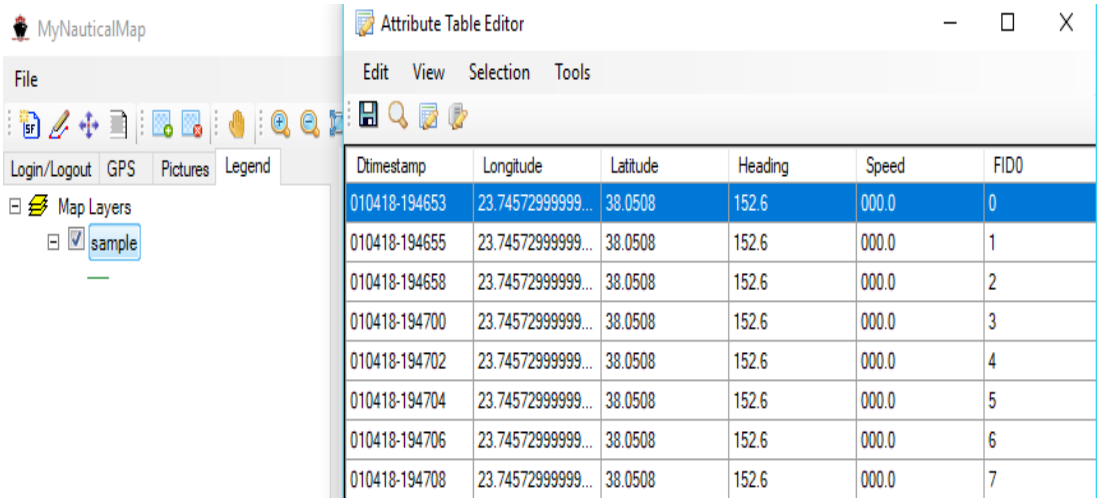
Παρατηρούμε ότι με την ενεργοποίηση του GPS, εμφανίστηκε στο χάρτη το στίγμα της θέσης μας (κόκκινο διαμάντι) καθώς και τα στοιχεία της θέσης μας στην αριστερή μεριά της εφαρμογής (Latitude, Longitude, Heading, Speed). Το στίγμα και τα στοιχεία της θέσης μας ανανεώνονται ανά δύο δευτερόλεπτα.

Αντίθετα, αν θέλουμε να απενεργοποιήσουμε το GPS, πατάμε το button της απενεργοποίησης και το στίγμα χάνεται από τον χάρτη καθώς και τα στοιχεία της θέσης μας απαλείφονται.



Εικόνα 63- Απενεργοποίηση GPS.

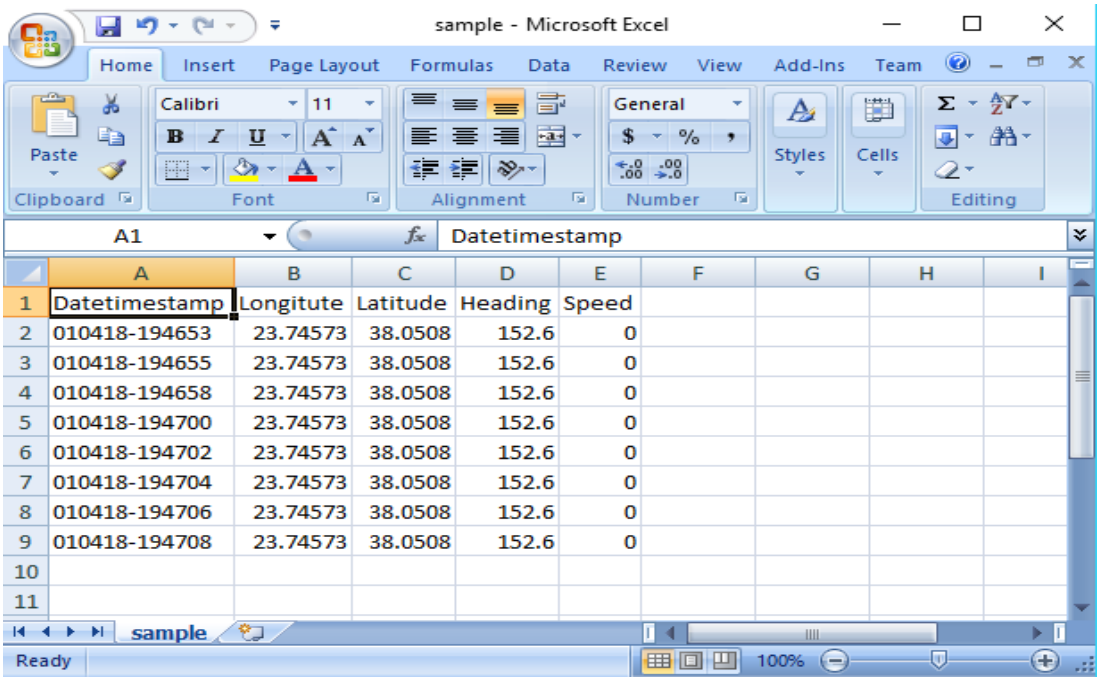
Στην περίπτωση που έχουμε ενεργοποιημένο το GPS και θελήσουμε να καταγράψουμε τα GPS δεδομένα που λαμβάνουμε ανά δύο δευτερόλεπτα, τότε πατάμε το button της έναρξης καταγραφής. Όταν θελήσουμε να σταματήσουμε να καταγράφουμε GPS δεδομένα, τότε πατάμε Λήξη καταγραφής. Αφού σταματήσει η καταγραφή, η εφαρμογή μας ζητά να αποθηκευτούν τα δεδομένα πρώτα σε αρχείο shapefile και στη συνέχεια σε αρχείο csv. Αν ανοίξουμε το αρχείο shapefile από την εφαρμογή πατώντας Add Layer θα πάρουμε το παρακάτω στιγμιότυπο:



Dtimestamp	Longitude	Latitude	Heading	Speed	FID0
010418-194653	23.74572999999...	38.0508	152.6	000.0	0
010418-194655	23.74572999999...	38.0508	152.6	000.0	1
010418-194658	23.74572999999...	38.0508	152.6	000.0	2
010418-194700	23.74572999999...	38.0508	152.6	000.0	3
010418-194702	23.74572999999...	38.0508	152.6	000.0	4
010418-194704	23.74572999999...	38.0508	152.6	000.0	5
010418-194706	23.74572999999...	38.0508	152.6	000.0	6
010418-194708	23.74572999999...	38.0508	152.6	000.0	7

Εικόνα 64- Άνοιγμα αρχείου shp από την εφαρμογή.

Παρατηρούμε από την παραπάνω εικόνα ότι εκτός των τιμών που βλέπουμε στην εφαρμογή μας, στην καταγραφή έχουμε και το Datetimestamp, δηλαδή καταγράφεται και η ημερομηνία με την ώρα λήψης των GPS δεδομένων από τους δορυφόρους. Επίσης, το csv αρχείο μπορούμε να το ανοίξουμε για να δούμε τα δεδομένα που καταγράψαμε μέσω του Microsoft Excel.

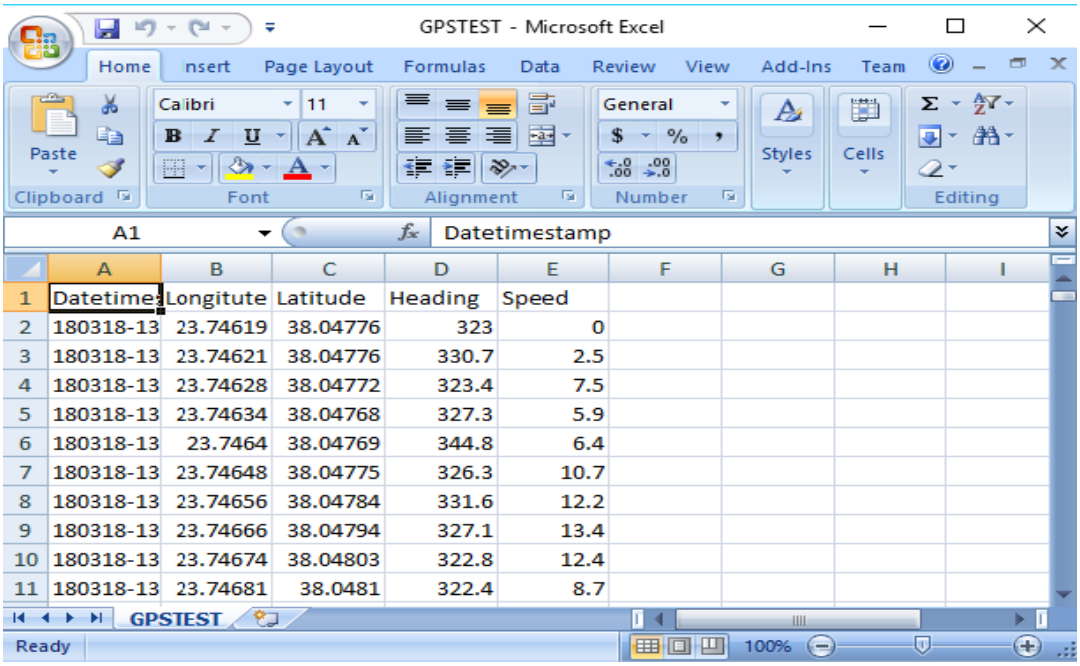


	A	B	C	D	E
1	Datetimestamp	Longitute	Latitude	Heading	Speed
2	010418-194653	23.74573	38.0508	152.6	0
3	010418-194655	23.74573	38.0508	152.6	0
4	010418-194658	23.74573	38.0508	152.6	0
5	010418-194700	23.74573	38.0508	152.6	0
6	010418-194702	23.74573	38.0508	152.6	0
7	010418-194704	23.74573	38.0508	152.6	0
8	010418-194706	23.74573	38.0508	152.6	0
9	010418-194708	23.74573	38.0508	152.6	0
10					
11					

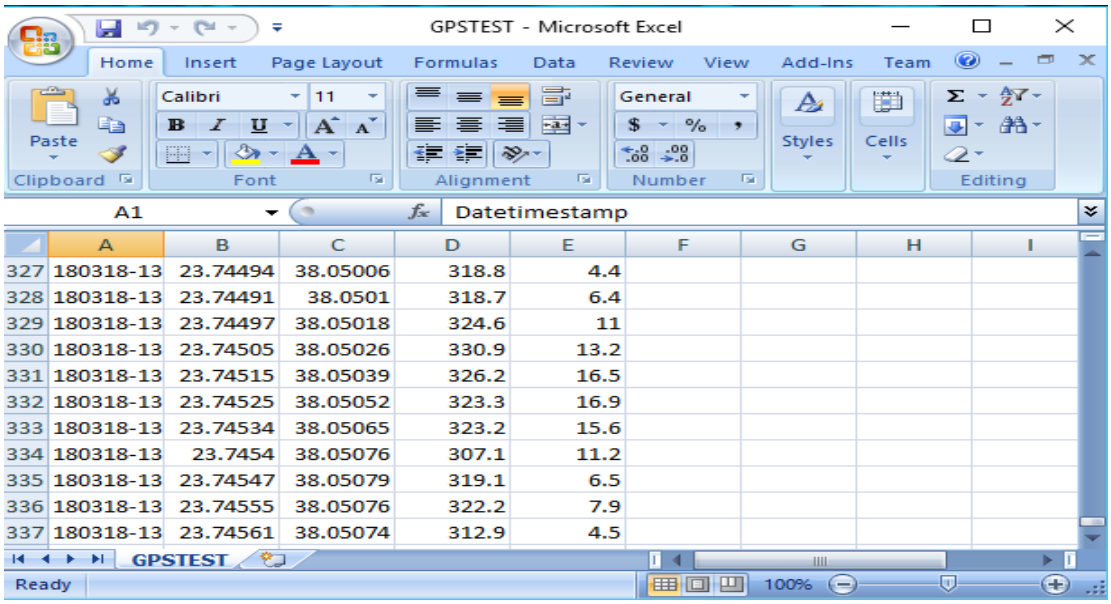
Εικόνα 65- Άνοιγμα αρχείου csv στο Microsoft Excel.

6.2.1. Δοκιμή Ορθότητας Στίγματος GPS

Για τη δοκιμή ορθής προβολής του στίγματος GPS στο χάρτη της εφαρμογής μας, αλλά και της ορθής καταγραφής μιας καμπύλης ενός σημείου που βρίσκεται σε κίνηση και ακολουθεί μια διαδρομή, εκτελέστηκε ένα πείραμα. Χρησιμοποιώντας το ιδιωτικό μας όχημα, καταγράψαμε μια διαδρομή με αφετηρία τη διεύθυνση Λαχανά 2-4 στη Νέα Φιλαδέλφεια και τερματισμό τη διεύθυνση Λασιθίου 7-9 πάλι στη Ν. Φιλαδέλφεια. Χρησιμοποιήσαμε τον υπολογιστή και το κινητό μας τηλέφωνο μέσα στο ιδιωτικό μας όχημα και καταγράψαμε τα GPS δεδομένα αυτής της διαδρομής. Αφού εκτελέστηκε η καταγραφή, αποθηκεύσαμε τα δεδομένα σε αρχεία shp και csv με το όνομα GPSTEST. Το αποτέλεσμα καταγραφής στο csv αρχείο απεικονίζεται παρακάτω:

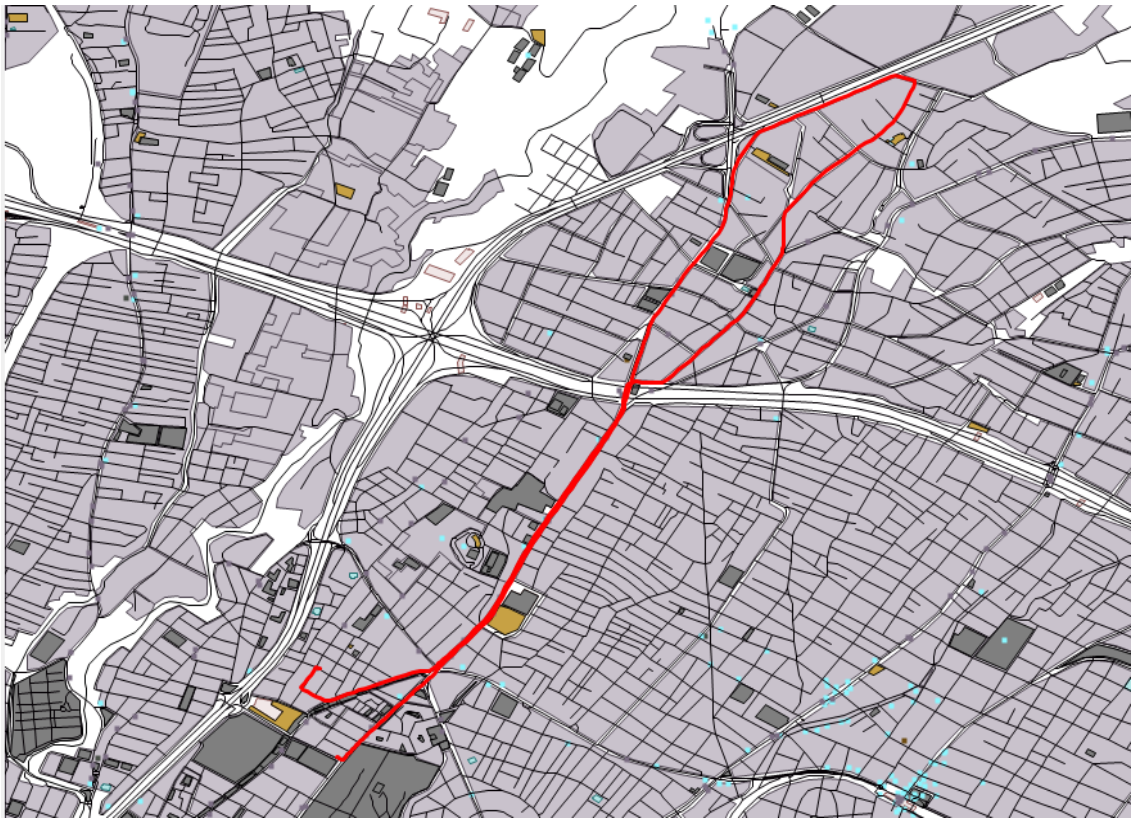


Εικόνα 66- GPS δεδομένα στην αρχή της διαδρομής.



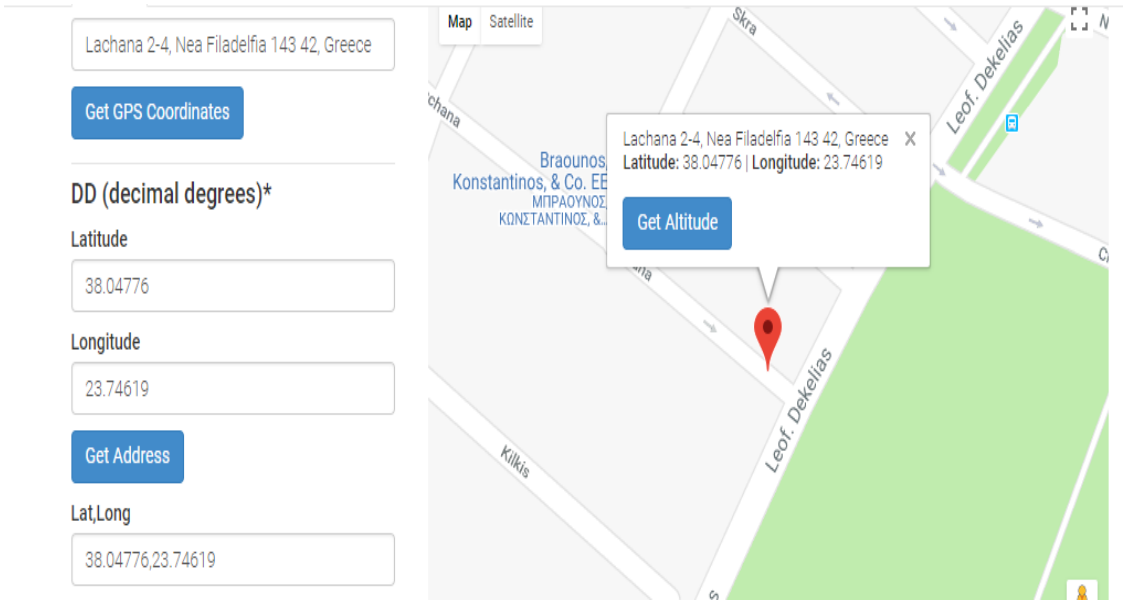
Εικόνα 67- GPS δεδομένα στο τέρμα της διαδρομής.

Η ταχύτητα μετρήθηκε σε κόμβους (knots) ενώ παρατηρούμε και τις άλλες τιμές που μεταβάλλονται με το χρόνο στις παραπάνω καταγραφές. Η αλλαγή θέσης ενός σημείου με την πάροδο του χρόνου σχηματίζει την καμπύλη διαδρομής του χρήστη στην εφαρμογή μας. Φορτώνοντας το χάρτη της Ελλάδας που κατεβάσαμε δωρεάν από το Geofabrik ως [29] και το αρχείο shapefile που δημιουργήσαμε από την καταγραφή της διαδρομής μας με όνομα GPSTEST έχουμε το κάτωθι αποτέλεσμα:

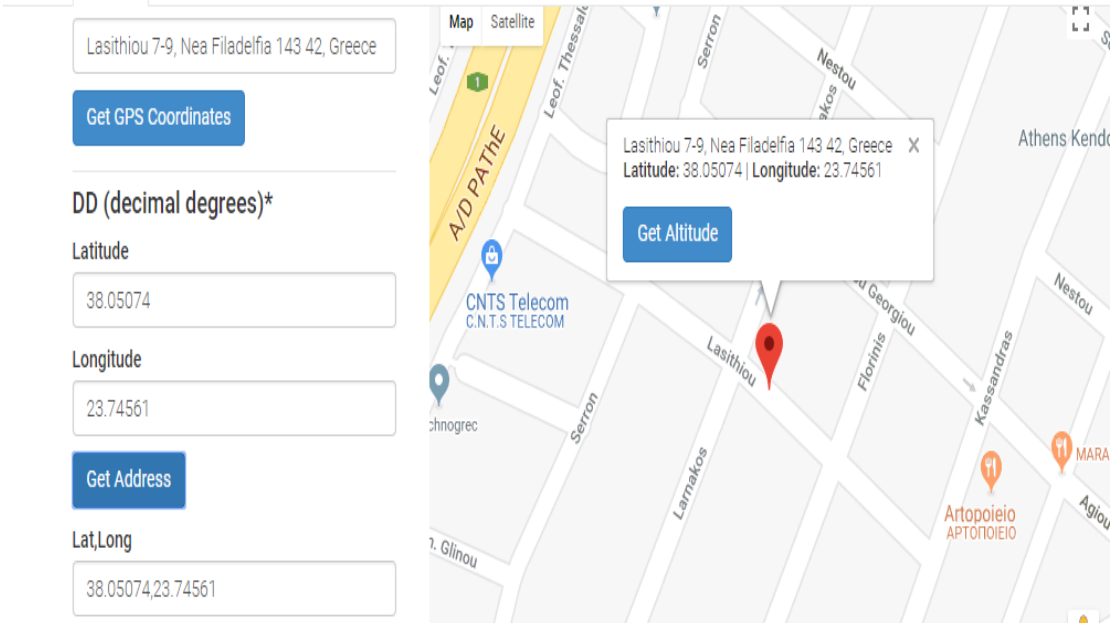


Εικόνα 68- Απεικόνιση διαδρομής αρχείου GPSTEST.

Παρατηρώντας την παραπάνω εικόνα και με δεδομένο τη διαδρομή που εκτελέσαμε με το ιδιωτικό μας όχημα διαπιστώνουμε ότι είναι ορθή. Για να ενισχύσουμε όμως περισσότερο το συγκεκριμένο πείραμα, με τη βοήθεια της εισαγωγής συντεταγμένων σε χάρτες της Google ως [30], εισαγάγαμε τις συντεταγμένες της αφετηρίας και του τερματισμού από το GPSTEST.csv και οι διευθύνσεις που πήραμε από την εφαρμογή της Google ήταν ορθές.



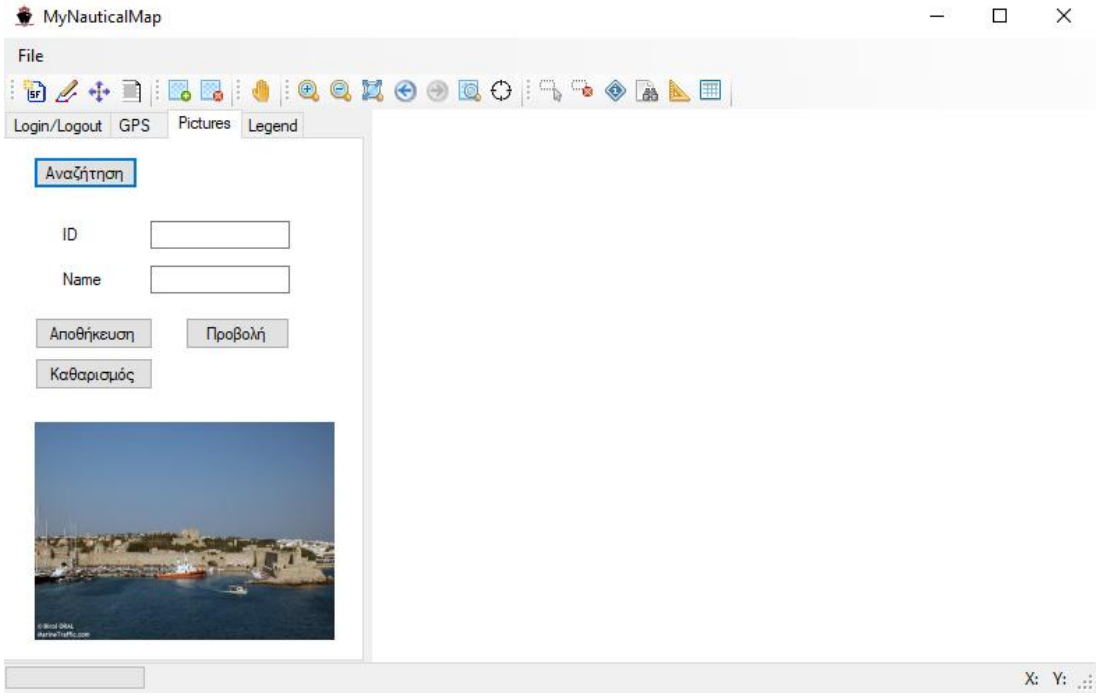
Εικόνα 69- Απεικόνιση διεύθυνσης αφετηρίας διαδρομής.



Εικόνα 70- Απεικόνιση διεύθυνσης τερματισμού διαδρομής.

6.3. Λειτουργίες εικόνων

Αν επιλέξουμε την καρτέλα Pictures, αυτή περιλαμβάνει τέσσερις λειτουργίες, την αναζήτηση για την εισαγωγή νέας εικόνας, την αποθήκευση μιας εικόνας που φορτώθηκε στην εφαρμογή στη βάση δεδομένων content, την προβολή μιας εικόνας από τη βάση δεδομένων content με την εισαγωγή του ID από το χρήστη και τον καθαρισμό οποιαδήποτε φορτωμένης εικόνας από το πρόγραμμα. Έστω ότι θέλουμε να εισάγουμε μια νέα εικόνα στην εφαρμογή μας. Αφού πατήσουμε το button της αναζήτησης μας αναδύεται ένα άλλο παράθυρο για να αναζητήσουμε την εικόνα από τον υπολογιστή μας. Κατεβάζοντας από το Marine Traffic τη φωτογραφία του λιμανιού της Ρόδου στον υπολογιστή μας, τη φορτώνουμε στην εφαρμογή μας.



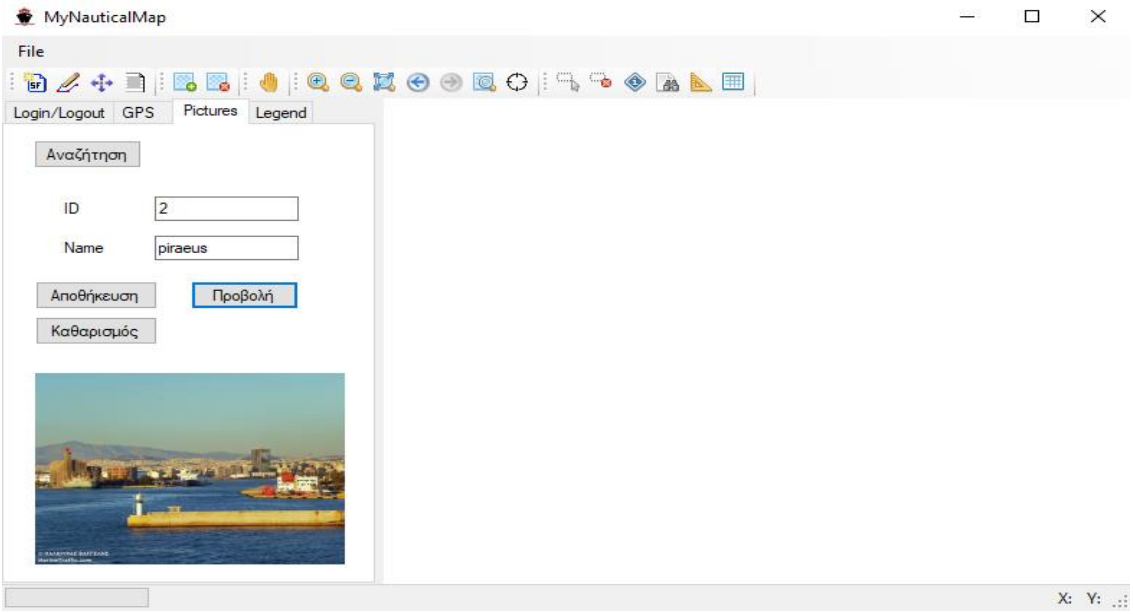
Εικόνα 71- Φόρτωση φωτογραφίας στην εφαρμογή.

Αν επιθυμούμε να αποθηκεύσουμε την παραπάνω φωτογραφία στη βάση δεδομένων content ώστε μελλοντικά να έχουμε τη δυνατότητα να προβάλουμε τη φωτογραφία ξανά, τότε πρέπει να εισάγουμε ένα ID και ένα Name στα πεδία και να πατήσουμε το button της αποθήκευσης. Έστω ότι στη βάση δεδομένων content έχουμε αποθηκεύσει τις κάτωθι εικόνες:

Πίνακας 3- Εικόνες με βάσει το ID

ID	Name	Image
1	Rhodes	Ρόδος
2	Piraeus	Πειραιάς

Αν εισάγουμε στο πεδίο π.χ. ID: 2 και πατήσουμε προβολή τότε μας εμφανίζεται η φωτογραφία του λιμανιού του Πειραιά:



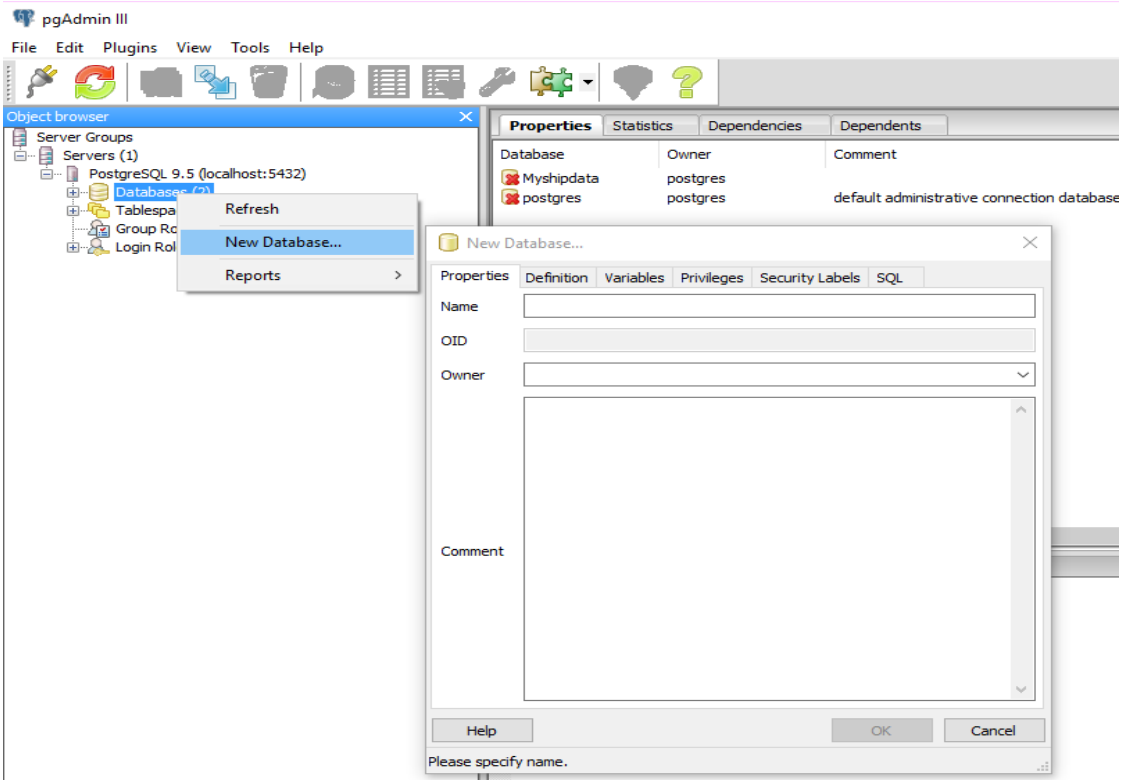
Εικόνα 72- Προβολή εικόνας με βάσει το ID εισαγωγής από το χρήστη.

Τέλος, αν πατήσουμε το button για καθαρισμό των εικόνων που έχουν φορτωθεί, η εικόνα και τα στοιχεία που είχαν εισαχθεί, θα απαλειφθούν.

7. Μελλοντικές επεκτάσεις εφαρμογής

Με βάση τις δυνατότητες που προσφέρει η εφαρμογή MyNauticalMap όπως αναπτύχθηκαν στο κεφάλαιο 6, μελλοντική επέκτασή της που θα μπορούσε να υλοποιηθεί είναι η ενσωμάτωση της PostgreSQL στην εφαρμογή με χρήση του Postgis με σκοπό να αξιοποιηθούν σε μεγαλύτερο βαθμό τα GPS δεδομένα και να εξαχθούν πολλά αποτελέσματα, τα οποία θα είναι χρήσιμα για το χρήστη.

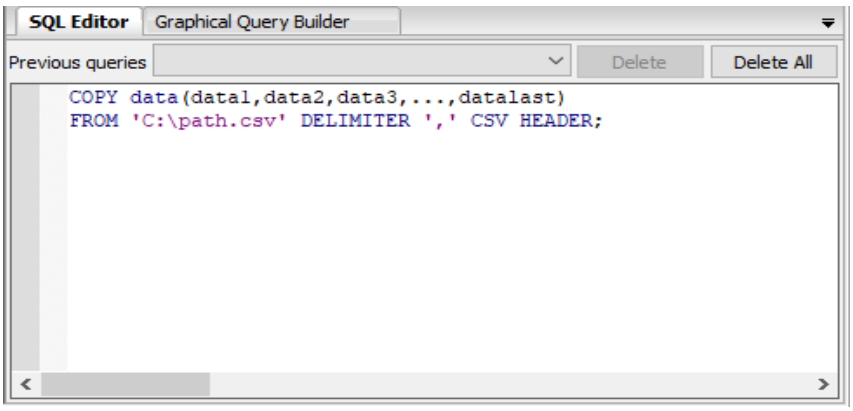
Συγκεκριμένα, για την ανωτέρω επέκταση θα ήταν αναγκαίο να δημιουργηθεί αρχικά μια βάση δεδομένων μέσω του pgAdmin της PostgreSQL. Η διαδικασία για τη δημιουργία της βάσης δεδομένων στο pgAdmin απεικονίζεται παρακάτω:



Εικόνα 73- Δημιουργία βάσης δεδομένων στη PostgreSQL.

Η βάση δεδομένων, η οποία θα δημιουργούνταν, θα έπρεπε να επικοινωνεί με την εφαρμογή MyNauticalMap. Η σύνδεση της PostgreSQL με μία οποιαδήποτε εφαρμογή Windows Form Application από το Visual Studio γίνεται μέσω του Npgsql αρχείου dll. Το Npgsql διατίθεται δωρεάν για κατέβασμα από το Nuget ως [32] και μπορεί να ενσωματωθεί στο Visual Studio μέσω της γραμμής εντολών του Nuget Package Manager. Έχοντας ενσωματώσει το αρχείο αυτό στην εφαρμογή μας, μπορούμε να συνδεθούμε στη βάση δεδομένων που δημιουργήσαμε αρχικά και να εκτελέσουμε πλήθος ερωτημάτων για την εξαγωγή αποτελεσμάτων.

Επίσης, επειδή τα GPS δεδομένα εξάγονται σε shapefile και csv αρχεία αποθήκευσης και δεδομένου ότι η PostgreSQL υποστηρίζει την εισαγωγή csv αρχείων σε μια βάση δεδομένων της, τα csv αρχεία που παράγονται μπορούν με μια εντολή να αποθηκευτούν στη βάση δεδομένων της PostgreSQL.



Εικόνα 74- Αντιγραφή δεδομένων από CSV αρχείο σε βάση δεδομένων PostgreSQL.

Από τα GPS δεδομένα που εξάγονται σε αρχεία shapefile και csv, δηλαδή από τις τιμές των μεταβλητών Datetimestamp, Longitude, Latitude, Heading και Speed, μπορούν να εκτελεστούν διάφορα ερωτήματα με σκοπό να εξαχθούν κάποια αποτελέσματα. Μερικά ερωτήματα είναι δυνατό να είναι:

- Ποια ήταν η μεγαλύτερη/ελάχιστη ταχύτητα που ανέπτυξε ο χρήστης;
- Πέρασε ο χρήστης από μια συγκεκριμένη περιοχή του χάρτη και αν ναι ποια χρονική στιγμή;
- Υπολογισμός πλήθους φορών που βρέθηκε ο χρήστης σε μια περιοχή του χάρτη
- Υπολογισμός αριθμού τροχιών του χρήστη.

Τα παραπάνω ερωτήματα είναι πολύ εύκολο να υλοποιηθούν με τις συναρτήσεις που υπάρχουν στο PostGIS και να εξαχθούν στην εφαρμογή MyNauticalMap.

8. Συμπεράσματα

Συνοψίζοντας, η υλοποίηση ναυτιλιακών εφαρμογών στις μέρες μας, όπως του MyNauticalMap που αναπτύχθηκε και αναλύθηκε στη παρούσα μεταπτυχιακή διατριβή, είναι αναγκαία για το εμπορικό και το πολεμικό ναυτικό της χώρας μας. Οι απαιτήσεις των πλοίων είναι τεράστιες και χωρίς τη βοήθεια των εργαλείων αυτών των εφαρμογών, οι κίνδυνοι θα ήταν μεγάλοι με ολέθρια αποτελέσματα. Επιπρόσθετα, η συμβολή του GPS στη ναυτιλία είναι αδιαμφισβήτητη. Μέρα με τη μέρα, υπάρχει εξέλιξη για την τεχνολογία του GPS όπως η ακρίβεια στο στίγμα που προσφέρει και αυτό καθιστά το GPS κυρίαρχο όχι μόνο για το παρόν άλλα και για το μέλλον. Η εφαρμογή MyNauticalMap αναπτύχθηκε για να προσφέρει στο χρήστη ενός πλοίου απλοϊκά διάφορες δυνατότητες ως προς την πλεύση του και σύμφωνα με τις μελλοντικές επεκτάσεις αυτής, μπορεί να αποτελέσει ένα ισχυρό εργαλείο για το ναυτιλλομένο.

8. Βιβλιογραφικές Αναφορές

1. Γεωγραφικά συστήματα πληροφοριών (GIS), https://eclass.hua.gr/modules/document/file.php/GEO174/GIS/%CE%A0%CE%91%CE%A1%CE%9F%CE%A5%CE%A3%CE%99%CE%91%CE%A3%CE%95%CE%99%CE%A3/%CE%A4%CE%B9%20%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9%20%CF%84%CE%B1%20GIS.pdf, https://el.wikipedia.org/wiki/%CE%A3%CF%8D%CF%83%CF%84%CE%B7%CE%BC%CE%B1_%CE%93%CE%B5%CF%89%CE%B3%CF%81%CE%B1%CF%86%CE%B9%CE%BA%CF%8E%CE%BD_%CE%A0%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%B9%CF%8E%CE%BD

2. QGIS, https://en.wikipedia.org/wiki/QGIS

3. ArcGIS, https://en.wikipedia.org/wiki/ArcGIS, http://www.marathondata.gr/pdfs/arcgis_desktop_products.pdf

4. Μορφότυποι χωρικών δεδομένων, https://en.wikipedia.org/wiki/GIS_file_formats, http://gisgeography.com/gis-formats/, http://wiki.openstreetmap.org/wiki/PBF_Format, https://en.wikipedia.org/wiki/Shapefile

5. Χωρικά συστήματα αναφοράς, http://spatialreference.org/ref/epsg/4326/, http://spatialreference.org/ref/epsg/2100/, https://wiki.openstreetmap.org/wiki/EPSG:3857

6. Σύστημα στιγματοθέτησης (GPS), https://el.wikipedia.org/wiki/Global_Positioning_System, http://dasodata.gr/index.php/systimata-entopismoy-thesis

7. NMEA 0183, http://freenmea.net/docs/nmea0183, http://www.tronico.fi/OH6NT/docs/NMEA0183.pdf

8. Δορυφορικά συστήματα πλοήγησης, https://www.liveviewgps.com/blog/gps-main-competitors-galileo-beidou-glonass/

9. Γεωγραφικές Συντεταγμένες, https://el.wikipedia.org/wiki/%CE%93%CE%B5%CF%89%CE%B3%CF%81%CE%B1%CF%86%CE%B9%CE%BA%CE%AD%CF%82_%CF%83%CF%85%CE%BD%CF%84%CE%B5%CF%84%CE%B1%CE%B3%CE%BC%CE%AD%CE%BD%CE%B5%CF%82

10. Μορφές γεωγραφικών συντεταγμένων και μετατροπές αυτών, https://tnp.uservoice.com/knowledgebase/articles/172110-latitude-longitude-formats-and-conversion

11. Μορφή γεωγραφικών συντεταγμένων στο RMC μήνυμα, http://docs.novatel.com/OEM7/Content/Logs/GPRMC.htm

12. Το .Net Framework, https://en.wikipedia.org/wiki/.NET_Framework, http://www.digitalnews.gr/2037/net-framework

13. Το Visual Studio, https://www.visualstudio.com/downloads/, http://estia.hua.gr/file/lib/default/data/16070/theFile

14. SQL Server, https://www.microsoft.com/en-us/sql-server/sql-server-2017-pricing

15. Το DotSpatial, https://github.com/DotSpatial/DotSpatial

16. Η UML, https://www2.dmst.aueb.gr/dds/ism/oo/indexw.htm

17. Βίρβου Μαρία, Σημειώσεις μαθήματος Τεχνολογίας Λογισμικού, Η γλώσσα μοντελοποίησης UML και μια διαδικασία εφαρμογής, Πανεπιστήμιο Πειραιώς, Τμήμα Πληροφορικής

18. PostgreSQL, https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

19. Θεοδωρίδης Γιάννης, Πελέκης Νίκος (2017) Γεωγραφικά Πληροφοριακά Συστήματα (GIS) & Βάσεις Γεωγραφικών Δεδομένων, Πανεπιστήμιο Πειραιώς, Τμήμα Πληροφορικής

20. Μάριος Βόντας, Παναγιώτης Ταμπάκης, Εργαστηριακή Διάλεξη στα Γεωγραφικά Πληροφοριακά Συστήματα- PostGIS, InfoLab, Τμήμα Πληροφορικής, Πανεπιστήμιο Πειραιώς

21. DotSpatial 1.9, https://github.com/DotSpatial/DotSpatial/tree/1.9

22. Bluetooth GPS Output, https://play.google.com/store/apps/details?id=com.meowsbox.btgps&hl=el

23. Μετατροπή Longitude και Latitude, https://www.directionsmag.com/site/latlong-converter/
24. Rfc4180Writer, https://www.codeproject.com/Tips/665519/Writing-a-DataTable-to-a-CSV-File
25. Παγκόσμιος Χάρτης, http://www.naturalearthdata.com/downloads/10m-physical-vectors/
26. Ευρωπαϊκός χάρτης, https://www.eea.europa.eu/data-and-maps/data/eea-coastline-for-analysis-1/gis-data/europe-coastline-shapefile
27. Κυριότερα λιμάνια παγκοσμίως, https://msi.nga.mil/NGAPortal/MSI.portal?_nfpb=true&_pageLabel=msi_portal_page_62&pubCode=0015
28. Ακτογραμμή και ακτές με γαλάζια σημαία Ελλάδας, http://geodata.gov.gr/
29. Χάρτης Ελλάδος, https://download.geofabrik.de/europe/greece.html
30. Προβολή συντεταγμένων σε χάρτες Google, https://www.gps-coordinates.net/
31. Marine Traffic, https://www.marinetraffic.com/
32. Npgsql, https://www.nuget.org/packages/Npgsql/

Παράρτημα

Κώδικας εφαρμογής MyNauticalMap

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading.Tasks;
using System.Data.SqlClient;

namespace MyNauticalMap
{
    public partial class Form1 : Form
    {
        public static string glob="";
        public Form1()
        {
```

```
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {

    }

    //Σύνδεση στην εφαρμογή MyNauticalMap
    private void button1_Click(object sender, EventArgs e)
    {
        if (textBox1.Text == "" || textBox2.Text == "")
        {
            MessageBox.Show("Παρακαλώ εισάγετε τα στοιχεία σας και στα δύο πεδία");
            textBox1.Text = "";
            textBox2.Text = "";
        }
        else
        {
            SqlConnection con = new SqlConnection(@"Data Source=
           (LocalDB)\MSSQLLocalDB;AttachDbFilename=
C:\Users\raz\Desktop\ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ\MyNauticalMap\personel.mdf;
Integrated Security=True;Connect Timeout=30;");
            SqlDataAdapter sda = new SqlDataAdapter("Select Count(*) From login where
username='" + textBox1.Text + "' and password='" + textBox2.Text + "'", con);
            DataTable dt = new DataTable();
            sda.Fill(dt);
            if (dt.Rows[0][0].ToString() == "1")
            {
                glob = textBox1.Text;
                this.Hide();
                Form2 form2 = new Form2();
                form2.Show();
            }
            else
```

```
            {
                MessageBox.Show("Λάθος Στοιχεία!Παρακαλώ ξαναεισάγετε τα στοιχεία σας");
                textBox1.Text = "";
                textBox2.Text = "";
            }
          }
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Form2.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.ComponentModel.Composition;
using System.Windows;
using System.IO;
using System.IO.Ports;
using DotSpatial.Positioning;
using DotSpatial.Positioning.Design;
using DotSpatial.Positioning.Forms;
using System.Globalization;
using DotSpatial.Projections;
using DotSpatial.Topology;
using DotSpatial.Data;
using DotSpatial.Symbology;
using DotSpatial.Controls;
using System.Data.SqlClient;
```

```
namespace MyNauticalMap
{
   public partial class Form2 : Form
   {
      public FeatureSet _markers;
      public DotSpatial.Controls.MapPointLayer _markerLayer;
      public string date;
      public string time;
      public string latitude;
      public string longitude;
      public string heading;
      public string speed;
      public double plon;
      public double plat;
      public DataTable sourceTable = new DataTable();
      public bool start_record = false;
      public FeatureSet fs = new FeatureSet(FeatureType.Line);
      public List<Coordinate> vertices = new List<Coordinate>();



      [Export("Shell", typeof(ContainerControl))]
      private static ContainerControl Shell;
      public Form2()
      {
         InitializeComponent();

         if (DesignMode) return;
         Shell = this;

         appManager1.LoadExtensions();

      }

      SqlConnection conection = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\raz\Desktop\ΜΕΤΑΠΤΥΧΙΑΚΗ
ΔΙΑΤΡΙΒΗ\MyNauticalMap\content.mdf;Integrated Security=True;Connect Timeout=30;");
      string imgLocation = "";
      SqlCommand cmd;
      private void Form2_Load(object sender, EventArgs e)
      {
         labelLog.Text = Form1.glob;
      }
```

```
private void timer1_Tick(object sender, EventArgs e)
{
   if (serialPort1.IsOpen)
   {
      string data = serialPort1.ReadExisting();
      string[] strArr = data.Split('$');
      for (int i = 0; i < strArr.Length; i++)
      {
         string strTemp = strArr[i];
         string[] lineArr = strTemp.Split(',');
         if (lineArr[0] == "GPRMC")
         {

            try
            {
               //date
               date = lineArr[9];

               //time
               time = lineArr[1];

               //Latitude
               Double dLat = Convert.ToDouble(lineArr[3]);
               dLat = dLat / 100;
               string[] lat = dLat.ToString().Split('.');
               string M = lat[1].Substring(0, 2);
               string m = lat[1].Substring(2);
               string Mm = M + "." + m;
               double d = Math.Round((Convert.ToDouble(Mm) / 60), 5);
               string ds = d.ToString().Substring(2);




               latitude = lat[0].ToString() + "." + ds + lineArr[4].ToString();
               plat = Double.Parse(lat[0].ToString() + "." + ds);



               //Longitude
               Double dLon = Convert.ToDouble(lineArr[5]);
               dLon = dLon / 100;
               string[] lon = dLon.ToString().Split('.');
```

```
            string Mlon = lon[1].Substring(0, 2);
            string mlon = lon[1].Substring(2);
            string Mmlon = Mlon + "." + mlon;
            double dlon = Math.Round((Convert.ToDouble(Mmlon) / 60), 5);
            string dslon = dlon.ToString().Substring(2);

            longitude = lon[0].ToString() + "." + dslon + lineArr[6].ToString();

            plon = Double.Parse(lon[0].ToString() + "." + dslon);


            //Speed
            speed = lineArr[7];


            //Display
            txtLat.Text = latitude;
            txtLong.Text = longitude;
            txtSpeed.Text = speed;



        }
        catch
        {
            //Cannot Read GPS values
            txtLat.Text = "GPS Unavailable";
            txtLong.Text = "GPS Unavailable";

        }
    }
    if (lineArr[0] == "GPHDT")
    {
        try
        {
            heading = lineArr[1];


            //Display
            txtHead.Text = heading;


        }
        catch
```

```
                {
                    //Cannot Read GPS values
                    txtLat.Text = "GPS Unavailable";
                    txtLong.Text = "GPS Unavailable";

                }
            }

        }

        //Απεικόνιση στίγματος GPS
        _markers = new FeatureSet(FeatureType.Point);
        _markerLayer = new DotSpatial.Controls.MapPointLayer(_markers);
        _markerLayer.Symbolizer = new PointSymbolizer(Color.Red,
DotSpatial.Symbology.PointShape.Diamond, 10);
        map1.MapFrame.DrawingLayers.Add(_markerLayer);
        Coordinate c = new Coordinate(plon, plat);
        _markers.AddFeature(new DotSpatial.Topology.Point(c));
        map1.MapFrame.Invalidate();
        _markers.Dispose();


        //Fill sourcetable GPS DATA
        if (start_record == true)
        {


            sourceTable.Rows.Add(date + "-" + time, plon, plat, heading, speed);


            vertices.Add(new Coordinate(plon, plat));
            vertices.Add(new Coordinate(plon, plat));
            LineString geom = new LineString(vertices);
            // add the geometry to the featureset.
            IFeature feature = fs.AddFeature(geom);
            feature.DataRow.BeginEdit();
            feature.DataRow["Dtimestamp"] = date + "-" + time;
            feature.DataRow["Longitude"] = plon;
            feature.DataRow["Latitude"] = plat;
            feature.DataRow["Heading"] = heading;
            feature.DataRow["Speed"] = speed;
            feature.DataRow.EndEdit();
```

```
                }


            }
            else
            {
                txtLat.Text = "COM Port Closed";
                txtLong.Text = "COM Port Closed";
            }
        }

        //Εξαγωγή GPS Δεδομένων σε shp και csv file
        bool btnclicked = true;
        private void button2_Click(object sender, EventArgs e)
        {
            //Έλεγχος για το αν είναι ενεργοποιημένα τα GPS DATA
            if (timer1.Enabled == false)
            {
                MessageBox.Show("Παρακαλώ ενεργοποιήστε πρώτα το GPS για λήψη
δεδομένων");
                return;
            }
            else
            {
                if (btnclicked)
                {

                    btnclicked = false;
                    button2.Text = "Stop";
                    timer1.Enabled = false;
                    start_record = true;

                    //FeatureSet fs = new FeatureSet(FeatureType.Line);
                    fs.DataTable.Columns.Add(new DataColumn("Dtimestamp", typeof(string)));
                    fs.DataTable.Columns.Add(new DataColumn("Longitude", typeof(double)));
                    fs.DataTable.Columns.Add(new DataColumn("Latitude", typeof(double)));
                    fs.DataTable.Columns.Add(new DataColumn("Heading", typeof(string)));
                    fs.DataTable.Columns.Add(new DataColumn("Speed", typeof(string)));


                    sourceTable.Columns.AddRange(new DataColumn[] {
                        new DataColumn("Datetimestamp", typeof(string)),
                        new DataColumn("Longitute", typeof(double)),
                        new DataColumn("Latitude", typeof(double)),
```

```
                    new DataColumn("Heading", typeof(string)),
                    new DataColumn("Speed", typeof(string))
                    });
                timer1.Enabled = true;



            }
            else
            {
                timer1.Enabled = false;
                btnclicked = true;
                button2.Text = "Start";

                try
                {
                    using (SaveFileDialog save1 = new SaveFileDialog() { Filter = "SHP|*.shp",
ValidateNames = true })
                    {
                        if (save1.ShowDialog() == DialogResult.OK)
                        {


                            fs.SaveAs(save1.FileName, true);

                        }

                    }

                    MessageBox.Show("Το αρχείο shp αποθηκεύτηκε με επιτυχία");
                }
                catch (Exception msg)
                {
                    MessageBox.Show(msg.ToString());
                    throw;

                }

                try
                {
                    using (SaveFileDialog save2 = new SaveFileDialog() { Filter = "CSV|*.csv",
ValidateNames = true })
                    {
                        if (save2.ShowDialog() == DialogResult.OK)
```

```
                    {
                        using (StreamWriter writer = new StreamWriter(save2.FileName))
                        {
                            Rfc4180Writer.WriteDataTable(sourceTable, writer, true);
                        }
                    }

                }
                MessageBox.Show("Το αρχείο csv αποθηκεύτηκε με επιτυχία");

            }
            catch (Exception msg)
            {
                MessageBox.Show(msg.ToString());
                throw;

            }


            timer1.Enabled = true;


        }
    }
}

    public static class Rfc4180Writer
    {
      public static void WriteDataTable(DataTable sourceTable, TextWriter writer, bool
includeHeaders)
        {
         if (includeHeaders)
         {
            IEnumerable<String> headerValues = sourceTable.Columns
               .OfType<DataColumn>()
               .Select(column => QuoteValue(column.ColumnName));

            writer.WriteLine(String.Join(",", headerValues));
         }

         IEnumerable<String> items = null;

         foreach (DataRow row in sourceTable.Rows)
         {
```

```
            items = row.ItemArray.Select(o => QuoteValue(o?.ToString() ?? String.Empty));
            writer.WriteLine(String.Join(",", items));
        }

        writer.Flush();
       }

       private static string QuoteValue(string value)
       {
        return String.Concat("\"",
        value.Replace("\"", "\"\""), "\"");
       }
      }

bool btnclicked2 = true;
private void button1_Click(object sender, EventArgs e)
{
   //Enable GPS
   if (btnclicked2)
   {
      // Try to open the serial port
      try
      {
         serialPort1.Open();
      }
      catch (Exception ex)
      {
         MessageBox.Show(ex.Message);
         return;
      }
      btnclicked2 = false;
      button1.Text = "Απενεργοποίηση GPS";
      timer1.Enabled = true;
   }
   //Disable GPS
   else
   {
      map1.Refresh();
      try
      {
         serialPort1.Close();
      }
      catch (Exception ex)
      {
```

```
                MessageBox.Show(ex.Message);
                return;
            }
            btnclicked2 = true;
            button1.Text = "Ενεργοποίηση GPS";
            timer1.Enabled = false;
            txtHead.Text = "";
            txtLong.Text = "";
            txtLat.Text = "";
            txtSpeed.Text = "";
        }
    }

    //Αναζήτηση εικόνων
    private void button3_Click(object sender, EventArgs e)
    {
        OpenFileDialog dialog = new OpenFileDialog();
        dialog.Filter = "png files(*.png)|*.png|jpg files(*.jpg)|*.jpg|All files(*.*)|*.*";
        if (dialog.ShowDialog() == DialogResult.OK)
        {
            imgLocation = dialog.FileName.ToString();
            pictureBox1.ImageLocation = imgLocation;
        }
    }

    //Αποθήκευση εικόνων στην βάση δεδομένων
    private void button4_Click(object sender, EventArgs e)
    {
        conection.Open();
        byte[] images = null;
        if (textID.Text == "" || textName.Text == "")
        {
            MessageBox.Show("Κάποιο/Κάποια από τα πεδία είναι κενά");
            conection.Close();
        }
        else if (imgLocation == "")
        {
            MessageBox.Show("Παρακαλώ εισάγετε νέα εικόνα");
            conection.Close();
        }
        else
        {
            FileStream Streem = new FileStream(imgLocation, FileMode.Open,
FileAccess.Read);
```

```
            BinaryReader brs = new BinaryReader(Streem);
            images = brs.ReadBytes((int)Streem.Length);
            SqlCommand cmdCount = new SqlCommand("SELECT count(*) from instance
WHERE Id = '" + textID.Text + "'", conection);
            int count = (int)cmdCount.ExecuteScalar();
            if (count > 0)
            {
                string sqlQuery = "UPDATE instance SET  Name=@NAME , Image=@images
WHERE Id = '" + textID.Text + "'";
                cmd = new SqlCommand(sqlQuery, conection);
                cmd.Parameters.Add(new SqlParameter("@images", images));
                cmd.Parameters.Add(new SqlParameter("@NAME", textName.Text));
                cmd.ExecuteNonQuery();
                conection.Close();
                MessageBox.Show(" Data Saved Successfully on your previous photo.....!");
                imgLocation = "";

            }
            else
            {
                string sqlQuery = "INSERT INTO instance (Id,Name,Image) VALUES ('" +
textID.Text + "','" + textName.Text + "', @images)";
                cmd = new SqlCommand(sqlQuery, conection);
                cmd.Parameters.Add(new SqlParameter("@images", images));
                int N = cmd.ExecuteNonQuery();
                conection.Close();
                MessageBox.Show(N.ToString() + " Data Saved Successfully.....!");
                imgLocation = "";
            }


        }
    }

    //Προβολή εικόνων
    private void button5_Click(object sender, EventArgs e)
    {
        conection.Open();
        string sqlQuery = "select Name,Image from instance where Id='" + textID.Text + "'";

        cmd = new SqlCommand(sqlQuery, conection);
        SqlDataReader Dataread = cmd.ExecuteReader();
        Dataread.Read();
        if (Dataread.HasRows)
```

```
        {
            textName.Text = Dataread[0].ToString();
            byte[] images = (byte[])Dataread[1];

            if (images == null)
            {
                pictureBox1.Image = null;
            }

            else
            {
                MemoryStream mstream = new MemoryStream(images);
                pictureBox1.Image = Image.FromStream(mstream);
            }
        }
        else
        {
            MessageBox.Show("This Data Not Available");
        }
        conection.Close();
    }

    //Καθαρισμός
    private void button6_Click(object sender, EventArgs e)
    {
        textID.Text = null;
        pictureBox1.Image = null;
        textName.Text = null;
        imgLocation = "";
    }

    private void Form2_FormClosing(object sender, FormClosingEventArgs e)
    {
        Application.Exit();
    }

    //Αποσύνδεση χρήστη
    private void button7_Click(object sender, EventArgs e)
    {
        conection.Close();
        if (serialPort1.IsOpen)
        {
            serialPort1.Close();
        }
```

```
            this.Dispose();
            GC.Collect();
            GC.WaitForPendingFinalizers();
            GC.Collect();

            Form1 form1 = new Form1();
            form1.Show();

        }
    }
}
```