ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Διαδικτυακή Συλλογή και Χαρτογράφηση Γεωχωρικών και Θαλάσσιων Μεγάλων Δεδομένων προς Ανάπτυξη Εφαρμογών στην Επιστήμη Δεδομένων

Διονύσιος Κοτζαΐτσης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υπεύθυνος

Θεόδωρος Τζουραμάνης,

Μόνιμος Επίκουρος Καθηγητής

Λαμία, 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Διαδικτυακή Συλλογή και Χαρτογράφηση Γεωχωρικών και Θαλάσσιων Μεγάλων Δεδομένων προς Ανάπτυξη Εφαρμογών στην Επιστήμη Δεδομένων

Διονύσιος Κοτζαΐτσης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υπεύθυνος

Θεόδωρος Τζουραμάνης,

Μόνιμος Επίκουρος Καθηγητής

Λαμία, 2021

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. *Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί* ***χωρίς να τα περικλείω σε εισαγωγικά*** *και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

2. *Δέχομαι ότι η αυτολεξεί* ***παράθεση χωρίς εισαγωγικά****, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. *Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*

4. *Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: 25/9/2021

Ο Δηλών

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.

Διαδικτυακή Συλλογή και Χαρτογράφηση
Γεωχωρικών και Θαλάσσιων Μεγάλων Δεδομένων
προς Ανάπτυξη Εφαρμογών στην Επιστήμη Δεδομένων

Διονύσιος Κοτζαΐτσης

Τριμελής Επιτροπή:

Θεόδωρος Τζουραμάνης, Μόνιμος Επίκουρος Καθηγητής.

Ιωάννης Αναγνωστόπουλος, Καθηγητής.

Αθανάσιος Λουκόπουλος, Αναπληρωτής Καθηγητής.

ΕΥΧΑΡΙΣΤΗΡΙΟ ΣΗΜΕΙΩΜΑ

Αρχικά, θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα καθηγητή Θεόδωρο Τζουραμάνη για την πολύτιμη καθοδήγησή του, τις εύστοχες επισημάνσεις του και την εξαιρετική συνεργασία σε όλα τα στάδια εκπόνησης της πτυχιακής μου εργασίας.

Επιπλέον, θεωρώ υποχρέωσή μου να ευχαριστήσω θερμά όλους τους οργανισμούς, επιχειρήσεις και ιδιώτες, για την βοήθεια που μου προσέφεραν με την αποδοχή χρήσης στοιχείων και δεδομένων.

Επιπρόσθετα, οφείλω να αφιερώσω την πτυχιακή μου εργασία τόσο στην οικογένειά μου, όσο και στο φιλικό μου περιβάλλον, που με υποστήριξαν και μου συμπαραστάθηκαν κατά τη διάρκεια της φοίτησης μου στο Πανεπιστήμιο Θεσσαλίας, στο εξαιρετικό τμήμα Πληροφορικής με Εφαρμογές στην Βιοϊατρική.

ΠΕΡΙΛΗΨΗ – ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ.

Η εργασία έχει ως αντικείμενο την εφαρμογή της Γεωπληροφορικής στο θαλάσσιο περιβάλλον, καθώς και τη διαχείριση δεδομένων βασισμένων στις ανάγκες του προαναφερθέντα κλάδου της Πληροφορικής μέσω συστημάτων βάσεων δεδομένων, αλλά και την παρουσίαση τους σε διαδικτυακή εφαρμογή, με τη χρήση σύγχρονων τεχνολογιών διαδικτύου.

Η παρούσα πτυχιακή εργασία χωρίζεται σε δύο σκέλη. Αρχικά, στο πρώτο μέρος της εργασίας γίνεται μια γενική αναφορά στα συστήματα Γεωγραφικών και Θαλάσσιων δεδομένων, καθώς και μια βαθύτερη μελέτη των δεδομένων αυτών. Επιπλέον, αναλύονται Συστήματα Διαχείρισης Βάσεων Δεδομένων με σκοπό την κατάλληλη χρήση τους στην διαχείριση Γεωχωρικών δεδομένων. Παράλληλα, πραγματοποιείται μια εισαγωγή στις σύγχρονες τεχνολογίες εφαρμογών διαδικτύου και στον τρόπο με τον οποίο αυτές χρησιμοποιούνται στα Γεωγραφικά Πληροφοριακά (ή εν συντομία Γεωπληροφοριακά) Συστήματα.

Στο δεύτερο σκέλος θα παρουσιαστούν οι μέθοδοι που αξιοποιήθηκαν για την ανάπτυξη της εφαρμογής, η ενσωμάτωση των δεδομένων σε αυτή, αλλά και τα δεδομένα που χρησιμοποιήθηκαν αυτά καθ’ αυτά σε συσχέτιση πάντα με την βάση δεδομένων.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ, ΓΕΩΓΡΑΦΙΚΑ ΔΕΔΟΜΕΝΑ, ΘΑΛΑΣΣΙΑ ΔΕΔΟΜΕΝΑ, ΜΕΓΑΛΑ ΔΕΔΟΜΕΝΑ, ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ, ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ, ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ, ΔΙΑΔΙΚΤΥΑΚΟΣ ΧΑΡΤΗΣ, ΓΕΩΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ.

SUMMARY - KEYWORDS.

The topic of this thesis is about Geospatial and Maritime Informatics, as well as the administration and study of the data based in these two fields, using databases. In addition this data will be used to create a web-map using state of the art web development technologies.

The thesis is pivoted between two main work streams. On one hand, we study about the geospatial and maritime data and systems, in general, as well as the ways that we could use some database management systems for administrating geospatial data. In addition, we examine the usage of state of the art web development techniques and technologies for developing Geographic Information Systems.

On the second leg of the thesis, we will discuss the development of our system, the data we are using and how we are going to build our database and app.

KEYWORDS: DATABASES, GEOSPATIAL DATA, MARITIME DATA, BIG DATA, DATA SCIENCE, WEB MAP, WEB DEVELOPMENT, GEOGRAPHIC INFORMATION SYSTEMS, APACHE TOMCAT, OPEN LAYERS, MAPPING .

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. ΕΙΣΑΓΩΓΗ - ΣΚΟΠΟΣ ΤΗΣ ΕΡΓΑΣΙΑΣ.

Η Γη αποτελεί «σπίτι» του ανθρώπου και όχι μόνο, κάθε κίνηση συμβαίνει και κάθε αντικείμενο υπάρχει πάνω σε αυτή, κάθε ζωντανός οργανισμός ζει σε κάποια γωνία της, κάθε συμβάν λαμβάνει χώρα στην επιφάνεια της ή κοντά σε αυτή. Σκάβουμε, για να χρησιμοποιήσουμε τους φυσικούς πόρους, κατασκευάζουμε λιμάνια και αεροδρόμια, για να καλύψουμε τις ανάγκες μας, πλέουμε σε πελάγη και ωκεανούς με τα πλοία μας, αλλά και παρακολουθούμε τις καιρικές συνθήκες. Μελετούμε τις φυσικές καταστροφές, που έχουν σχέση με το ανάγλυφο της, π.χ. σεισμούς και ηφαίστεια και παράλληλα μελετάμε βιότοπους άλλων οργανισμών με τους οποίους συμβιώνουμε.

Όπως είναι αναμενόμενο, μέσω της μελέτης της Γης -αλλά και της ανθρώπινης δραστηριότητας πάνω σε αυτή- δημιουργείται ένας ολοένα και αυξανόμενος όγκος δεδομένων, σε βαθμό τέτοιο που η διαχείριση και μελέτη του καθίσταται αδύνατη χωρίς την χρήση της υπολογιστικής ισχύος των ηλεκτρονικών υπολογιστών.

Έτσι, με σκοπό την επίλυση του παραπάνω δημιουργήθηκε ο κλάδος της Γεωπληροφορικής (Geoinformatics), ο οποίος έχει εφαρμογές σε διάφορους τομείς του ανθρώπινου βίου, από την επιχειρηματική ζωή, την επιστημονική μελέτη, μέχρι και την απλή εύρεση μιας διαδρομής. Ταυτόχρονα, με την ανάπτυξη της Γεωπληροφορικής προέκυψε ένας ακόμη κλάδος εφαρμογής της Πληροφορικής -πολύ σημαντικός για την χώρα μας- η Γεωπληροφορική για το θαλάσσιο περιβάλλον (Maritime Informatics), με σκοπό την μελέτη των δεδομένων από πλοία και των θαλάσσιων μεταφορών.

Η εργασία λοιπόν βασίζεται στις παραπάνω επιστήμες και έχει ως θέμα «Διαδικτυακή συλλογή και χαρτογράφηση γεωχωρικών και θαλάσσιων μεγάλων δεδομένων προς ανάπτυξη εφαρμογών στην Επιστήμη Δεδομένων», ή στα Αγγλικά «Collecting and mapping geospatial and maritime big data on the web for Data Science applications».

Στόχος είναι η ανάπτυξη ενός συστήματος το οποίο θα έχει ως θεμέλιο του μία βάση δεδομένων με την χρήση PostgreSQL και του εργαλείου της PostGIS και το οποίο διευκολύνει το έργο της διαχείρισης των γεωγραφικών δεδομένων. Βασιζόμενοι σε αυτό το σύστημα διαχείρισης βάσεων δεδομένων, χαρτογραφούμε

και αναδιανέμουμε τα δεδομένα αυτά με χρήση τεχνολογιών web development, όπως JavaScript, CSS, HTML, OpenLayers, Apache Tomcat και GeoServer για την εξυπηρέτηση της χαρτογράφησης προς τον χρήστη, jQuery για την μεταφορά δεδομένων από και προς τον εξυπηρετητή, χρήση API's για συγκεκριμένα δεδομένα και άλλα, όπως θα αναλυθούν στη συνέχεια. Να σημειωθεί ότι τα δεδομένα μας αφορούν την Ελλάδα.

Κατά την διάρκεια της ανάπτυξης του συστήματος αυτού, υπήρχαν πολλές δυσκολίες που έπρεπε να ξεπεραστούν. Αρχικά, τα δεδομένα αρκετές φορές ήταν μη διαθέσιμα για αναδιανομή ή πολλές φορές ακόμα και για δωρεάν χρήση, καθιστώντας δύσκολο έως αδύνατο το έργο της πλήρους ολοκλήρωσης του συνόλου δεδομένων που χαρτογραφήθηκαν. Επιπλέον, τα δεδομένα ήταν διαθέσιμα σε διαφορετικές μορφές και διαφορετικά είδη αρχείων. Έτσι, για να μπορέσουμε να υλοποιήσουμε το σύστημα μας, έπρεπε να επινοήσουμε διάφορες διαδικασίες επεξεργασίας και εισαγωγής των δεδομένων μας.

Η υλοποίηση του συστήματος αυτού έχει γίνει για τους σκοπούς της εκπόνησης της διπλωματικής εργασίας και μελέτης, παρ' όλα αυτά δεν σημαίνει ότι δεν μπορούν να υπάρξουν βελτιώσεις. Για παράδειγμα, η μεταφορά των δεδομένων σε έναν Server και η αγορά ενός Domain, θα μπορούσε να οδηγήσει στην ανάπτυξη ενός κανονικού προϊόντος. Επιπλέον, η αγορά κάποιων αδειών χρήσης μπορεί να βοηθήσει στην ολοκλήρωση της συλλογής των δεδομένων και να βελτιώσει το σύστημα.

2. ΤΑ ΓΕΩΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ Η ΕΦΑΡΜΟΓΗ ΤΟΥΣ ΣΤΟ ΘΑΛΑΣΣΙΟ ΠΕΡΙΒΑΛΛΟΝ

I. Γεωπληροφοριακά Συστήματα

Αρχικά, οφείλουμε να δώσουμε έναν ορισμό για τα Γεωγραφικά και Χωρικά Δεδομένα. Με τον όρο Γεωγραφικά Δεδομένα, αναφερόμαστε στα δεδομένα που έχουν να κάνουν με περιοχές στην επιφάνεια της Γης ή κοντά σε αυτή. Με τον όρο χωρικά δεδομένα, από την άλλη, αναφερόμαστε σε δεδομένα που έχουν να κάνουν με τη γενική έννοια του χώρου, αν και συχνά θα αναφερόμαστε στα χωρικά δεδομένα ως Γεωγραφικά ή Γεωχωρικά Δεδομένα [1].

Τα δεδομένα αυτά, όπως είναι λογικό, είναι άκρως σημαντικά ως προς τη μελέτη και την ανάλυση τους, καθότι χρησιμεύουν σε κάθε πτυχή της ανθρώπινης ύπαρξης. Παρ' όλα αυτά η διαχείρισή τους είναι μια δύσκολη και απαιτητική εργασία και αυτό διότι:

- Τα γεωγραφικά δεδομένα είναι πολυδιάστατα, δηλαδή, κατά την μελέτη μας απαιτείται η χρήση του Γεωγραφικού μήκους, Γεωγραφικού πλάτους, αλλά και πολλές φορές του ύψους και του βάθους, εάν μελετούμε το ανάγλυφο του εδάφους.
- Επειδή τα γεωγραφικά δεδομένα βρίσκονται υπό διαρκή ανανέωση και αύξηση δημιουργείται το πρόβλημα της διαχείρισης χώρου, αφού τα δεδομένα μπορούν να αυξηθούν πολύ γρήγορα σε μέγεθος.
- Η χαρτογράφηση αυτών των δεδομένων μπορεί να γίνει με διαφορετική κλίμακα και ανάλυση ανάλογα με τις ανάγκες του χρήστη και της εφαρμογής.
- Τα δεδομένα παράγονται σε διαφορετικές μορφές αρχείων.
- Επειδή η επιφάνεια της Γης είναι κυρτή και οι χάρτες μας στις περισσότερες περιπτώσεις είναι επίπεδοι, χρειαζόμαστε αρκετές μετατροπές σε διάφορα συστήματα συντεταγμένων, όπως το WGS 84, κάτι που καθιστά τα δεδομένα μας κάπως «θορυβώδη» σε σχέση με τις πραγματικές συντεταγμένες πάνω στην Γη.

- Επιπλέον, τα συστήματα συντεταγμένων που χρησιμοποιούνται σε διάφορες περιοχές του πλανήτη διαφέρουν, με αποτέλεσμα να είναι αναγκαίο να γίνουν ακόμη περισσότερες μετατροπές στις συντεταγμένες των δεδομένων μας [Πίνακας α].
- Ένα ακόμη πρόβλημα με τα δεδομένα αυτά, είναι ότι χρησιμοποιούνται για πολλούς και διαφορετικούς λόγους, και η ερμηνεία τους αλλάζει από χρήση σε χρήση. Παραδείγματος χάριν, αλλιώς θα χρησιμοποιούνται τα δεδομένα για τη δημιουργία μιας εφαρμογής πλοήγησης και αλλιώς, για να παρακολουθήσουμε περιβαλλοντικές αλλαγές.
- Τέλος, τα γεωγραφικά δεδομένα δεν αποτελούν απλά ακατέργαστα δεδομένα, τα οποία μπορούμε να διαχειριστούμε. Κάθε συντεταγμένη και κάθε σύνολο δεδομένων αποτελεί και δίνει στον αναγνώστη μία πολύ συγκεκριμένη γνώση. Ως εκ τούτου, είναι πολύ ευαίσθητα σε λάθη, κάτι που κάνει τη διαχείρισή τους εξόχως δύσκολη.

Χώρα	Σύστημα Συντεταγμένων
Ελλάδα	Hellenic Geodetic Reference System 1987
Κίνα	Lambert Conformal Conic
Ηνωμένο Βασίλειο	British National Grid
Η.Π.Α.	United States National Grid
Γαλλία	Lambert-93
Ισραήλ	Israeli Transverse Mercator
Ολλανδία	Stelsel van de Rijksdriehoeksmeting

Πίνακας α: Ενδεικτικός πίνακας χωρών με διαφορετικά συστήματα συντεταγμένων.

Για τους παραπάνω λοιπόν λόγους, αλλά και με την ραγδαία ανάπτυξη της τεχνολογίας, η επιστήμη οδηγήθηκε στο συμπέρασμα ότι η καλύτερη λύση για τη μελέτη και επεξεργασία των δεδομένων αυτών θα είναι με τη χρήση συστημάτων βασισμένων στους ηλεκτρονικούς υπολογιστές. Με τη χρήση υπολογιστικών εργαλείων μπορεί να γίνει αποθήκευση, επεξεργασία, ανάλυση και χαρτογράφηση των δεδομένων αυτών δημιουργώντας έτσι τον κλάδο των Γεωπληροφοριακών Συστημάτων.

Κατά μία γενική άποψη τα σύγχρονα Γεωπληροφοριακά Συστήματα αποτελούνται από πέντε κύριους άξονες:

a. *Το δίκτυο*. Όλα τα μοντέρνα Γεωπληροφοριακά Συστήματα βασίζονται στο διαδίκτυο χρησιμοποιώντας Servers και τεχνολογίες Cloud Computing, ώστε να δώσουν στον χρήστη τα δεδομένα του άμεσα.
b. *Κατάλληλο λογισμικό,* για ανάγνωση και ανάλυση των δεδομένων. Είτε αυτό είναι ανοιχτού λογισμικού είτε επί πληρωμή.
c. *Κατάλληλο υλικό,* που θα μπορέσει να αντέξει τον υπολογιστικό φόρτο της επεξεργασίας και αποθήκευσης των δεδομένων, αλλά και να λειτουργήσει παράλληλα με το λογισμικό.
d. *Τα δεδομένα*. Ίσως το πιο σημαντικό κομμάτι της Γεωπληροφορικής. Η ποσότητα και η ποιότητα των δεδομένων παίζει πολύ σημαντικό ρόλο στην εύρυθμη λειτουργία του υπολογιστικού συστήματος μας. Η δημιουργία και διαχείριση μιας βάσης δεδομένων -σχεσιακής ή μη- και η σωστή διακυβέρνηση και ανανέωση της είναι κλειδί στην λειτουργία του συστήματος, αφού ο όγκος και η πολυπλοκότητα των δεδομένων απαιτεί την ύπαρξη ενός πιο αναπτυγμένου προγράμματος από ένα απλό σύστημα αποθήκευσης.
e. *Τα άτομα*. Τα άτομα που θα κληθούν να επιλέξουν, να προγραμματίσουν, να χαρτογραφήσουν, αλλά και να μελετήσουν τα δεδομένα πρέπει, πέρα από καλοί γνώστες προγραμματισμού, να έχουν μια διεξοδική γνώση πάνω στα Γεωχωρικά δεδομένα, αλλά και στη λειτουργία των υπολογιστικών συστημάτων γενικότερα.

Έτσι, εάν ακολουθηθούν οι παραπάνω άξονες, θα παραχθεί εύχρηστο και λειτουργικό προς τον χρήστη σύστημα.

Όσον αφορά τη δομή ενός Γεωπληροφοριακού Συστήματος, διαφέρει από χρήση σε χρήση διατηρώντας κάποια βασικά χαρακτηριστικά σταθερά. Αρχικά, θα πρέπει να κάνουμε τον διαχωρισμό του επιπέδου υλοποίησης του συστήματος. Ένα σύστημα-εφαρμογή μπορεί να είναι προσωπικό πρότζεκτ ή να ανήκει σε επίπεδο εφαρμογής τμήματος ή εταιρείας -δηλαδή να χρησιμοποιείται για εμπορικούς σκοπούς. Παρά τον διαχωρισμό, βέβαια, υπάρχει ένα κλασικό αρχιτεκτονικό μοντέλο

κατασκευής ενός τέτοιου πληροφοριακού συστήματος [Σχήμα a], με βάση το οποίο μπορεί να γίνει εύκολη επέκταση του συστήματος αυτού, ανάλογα με το επίπεδο υλοποίησης. Το παρακάτω μοντέλο ισχύει και για συστήματα τα οποία τρέχουν τοπικά, αλλά και για συστήματα που στηρίζονται στις τεχνολογίες δικτύων -δηλαδή τρέχουν σε κάποιο LAN ή WAN μέσω ενός GIS Server. Περισσότερα για την υποδομή θα συζητηθούν σε επόμενα κεφάλαια.

Σχήμα a: Κλασικό μοντέλο κατασκευής ενός Γεωπληροφοριακού Συστήματος σύμφωνα με πηγή [1]. Με bold γραφή αποτελούν τα στοιχεία του σχήματος.

Ακολουθώντας το παραπάνω αρχιτεκτονικό μοντέλο, υπάρχουν διαθέσιμα αρκετά συστήματα και εφαρμογές εστιασμένες στη Γεωπληροφορική. Αρχικά, εφαρμογές web map, όπως το Open Street Map (http://www.openstreetmap.org) που είναι εφαρμογή χαρτογράφησης ανοιχτού λογισμικού, αλλά και οι εμπορικές εφαρμογές Google Maps (https://www.google.com/maps), Apple Maps (https://www.apple.com/maps/) κ.λ.π. δημιούργησαν ένα νέο κεφάλαιο στη διαδικτυακή χαρτογράφηση και πλοήγηση.

Επιπρόσθετα, οι εφαρμογές ArcGIS και QGIS αποτελούν σημαντικά συστήματα κάθε επιστήμονα που ασχολείται με τη Γεωπληροφορική, καθώς δίνουν στον χρήστη εργαλεία και υποπρογράμματα, έτσι ώστε να μπορέσει ο ίδιος να δημιουργήσει χάρτες και συστήματα. Επιπλέον, για τη μελέτη των δεδομένων αυτών εξαιρετική βοήθεια αποτελεί η βιβλιοθήκη GDAL –η οποία χρησιμοποιήθηκε για τις ανάγκες της εργασίας, όπως θα δούμε παρακάτω–, η οποία μέσω API μπορεί να χρησιμοποιηθεί σε όλες τις σύγχρονες γλώσσες προγραμματισμού (Python, R, MATLAB, C κλπ.).

Κλείνοντας λοιπόν αυτό το κεφάλαιο, οφείλουμε για άλλη μια φορά να τονίσουμε την σημασία των δεδομένων και των συστημάτων αυτών στην ζωή του ανθρώπου. Τα Γεωπληροφοριακά Συστήματα μπορούν να χρησιμοποιηθούν από τον πιο απλό άνθρωπο για τη δημιουργία ενός χάρτη και την πλοήγηση σε έναν οποιοδήποτε χώρο, μέχρι και από κυβερνήσεις για σχέδια διαχείρισης κρίσεων.

Επαγγελματίες πολλών χώρων, όπως για παράδειγμα στρατιωτικοί, επαγγελματίες της ευρείας οικονομικής ζωής -εστίαση, ξενοδοχεία, μεταφορές– χρησιμοποιούν καθημερινά εφαρμογές πλοήγησης και αναζήτησης, βασισμένες σε συντεταγμένες, ενώ πολυεθνικές εταιρίες καυσίμων και τηλεπικοινωνιών ελέγχουν υποθαλάσσια για την εξεύρεση κατάλληλων περιοχών για ανάπτυξη δραστηριοτήτων.

Περιβαλλοντικές οργανώσεις, αλλά και επαγγελματίες αλιείς, γεωργοί και κτηνοτρόφοι παρακολουθούν τις μετεωρολογικές και περιβαλλοντικές μεταβολές ανά τοποθεσία. Εκατοντάδες εκατομμύρια χρήματα έχουν δαπανηθεί από τεχνολογικούς κολοσσούς για την ανάπτυξη αντίστοιχων εφαρμογών. Πολλές επιστήμες, όπως της Γεωλογίας, της Μετεωρολογίας, της Πληροφορικής και φυσικά της Γεωγραφίας έχουν αναπτυχθεί παράλληλα με τα Γεωπληροφοριακά Συστήματα.

Επιπροσθέτως, ο κλάδος της Ναυτιλίας, αλλά και των θαλάσσιων μεταφορών πλέον εξαρτάται από τέτοια συστήματα για τον συντονισμένο απόπλου των χιλιάδων πλοίων που σαλπάρουν καθημερινά από λιμάνια σε όλη την υφήλιο, δημιουργώντας τον άκρως αναπτυσσόμενο κλάδο της εφαρμογής της Γεωπληροφορικής στο θαλάσσιο περιβάλλον, ο οποίος θα αναλυθεί στο επόμενο υποκεφάλαιο.

II. Γεωπληροφοριακά Συστήματα για το Θαλάσσιο Περιβάλλον

Περίπου το 72% της επιφάνειας της Γης, σύμφωνα με το USGS (https://www.usgs.gov/), καλύπτεται από νερό. Ωκεανοί, πελάγη, ποτάμια και λίμνες, δημιουργούνται από κοιλότητες του εδάφους και είναι ζωτικής σημασίας για την ύπαρξη ζωής στον πλανήτη.

Τη ζωτική αυτή σημασία του νερού και των θαλάσσιων περιοχών την αντιλήφθηκε το ανθρώπινο είδος από αρχαιοτάτων χρόνων. Δεν είναι τυχαίο, άλλωστε, ότι οι μεγαλύτεροι πολιτισμοί και οι μεγαλύτερες μητροπόλεις βρίσκονται σε παράκτιες περιοχές. Αυτή η μετοίκιση των ανθρώπων στις περιοχές με θαλάσσια πρόσβαση, αλλά και η τεράστια έκταση των υδάτινων περιοχών μετέτρεψε τη θάλασσα σε ένα πρόσφορο χώρο αρχικά για εύρεση τροφής (μέσω της αλιείας), για επικοινωνία με άλλες χώρες, αλλά και στη συνέχεια για την ανάπτυξη του εμπορίου και των μεταφορών.

Από την αρχαιότητα, λοιπόν, οι άνθρωποι κατασκεύασαν πλοία για να μεταφέρουν πληθυσμούς και αγαθά από και προς αποικίες και συμμάχους. Ενώ, με την πάροδο των ετών, η ναυτιλία χρησιμοποιήθηκε επιπλέον και για στρατιωτικούς και οικονομικούς σκοπούς, αλλά και για λόγους εποικισμού.

Παραδοσιακές δυνάμεις στον χώρο της ναυτιλίας αναπτύχθηκαν λόγω της στενής σχέσης, αλλά και μικρής απόστασης από την θάλασσα. Από την Αρχαία Ελλάδα και τους Βίκινγκ, τη Μεγάλη Βρετανία, τις ΗΠΑ και τη Γερμανία -μέσω της στρατιωτικής ναυτιλίας- κατά τον Α’ και Β’ Παγκόσμιο Πόλεμο, έως και τεράστιες σύγχρονες οικονομικές δυνάμεις του χώρου του εμπορίου (π.χ. Ελλάδα, Ολλανδία) και της αλιείας (π.χ. Ιαπωνία), η Ναυτιλία εξελίσσεται και αναδιαμορφώνεται ως κυρίαρχη ανθρώπινη δραστηριότητα.

Παρά, λοιπόν, την μακρόχρονη ιστορία του ανθρώπου στη θάλασσα, υπήρχαν και υπάρχουν δυο προβλήματα–χαρακτηριστικά που κάνουν δύσκολο το θαλάσσιο ταξίδι. Αυτά είναι η εύρεση αξιόπιστης πηγής συντεταγμένων εν πλω και αξιόπιστων χαρτών (βαθυμετρικών και πλοήγησης) της θάλασσας [2].

Δημιουργήθηκαν έτσι, πολλές τεχνολογίες και πολλές υπηρεσίες που έχουν οδηγήσει στη σημερινή άνθηση της Ναυτιλίας αλλά και παράλληλων κλάδων, όπως η Ναυτιλιακή και Θαλάσσια Πληροφορική. Από τον μηχανισμό των Αντικυθήρων και τα Sonar, μέχρι και τη σημερινή εποχή του AIS (Automatic Identification System ή

Αυτόματο Σύστημα Αναγνώρισης) και του διαδικτύου, η διαχείριση και μελέτη του τεράστιου όγκου δεδομένων απαιτεί εξειδικευμένο λογισμικό και προσωπικό, αλλά και οργάνωση από κεντρικές οντότητες όπως ο IMO (International Maritime Organization) με συγκεκριμένες τυποποιήσεις.

Η ύπαρξη, η παρατήρηση και η καταγραφή του τεράστιου όγκου πληροφορίας για τη θάλασσα και τις θαλάσσιες μετακινήσεις δεν είναι κάτι νέο. Παρ' όλα αυτά, σήμερα απαιτείται η ακριβής πληροφορία με τον μικρότερο δυνατό θόρυβο. Επιπλέον, η ύπαρξη νέων τεχνολογιών επιτρέπει την καταγραφή, ανάλυση, επεξεργασία και αναδιανομή αυτού του μεγάλου όγκου πληροφορίας χρησιμοποιώντας τα Πληροφοριακά Συστήματα. Έτσι, δημιουργήθηκε ο κλάδος της Ναυτιλιακής Πληροφορικής, τομέας παραγωγής των Ναυτιλιακών Πληροφοριακών Συστημάτων, ένας κλάδος–κλειδί για την τεράστια και ανατροφοδοτούμενη οικονομία της Ναυτιλίας.

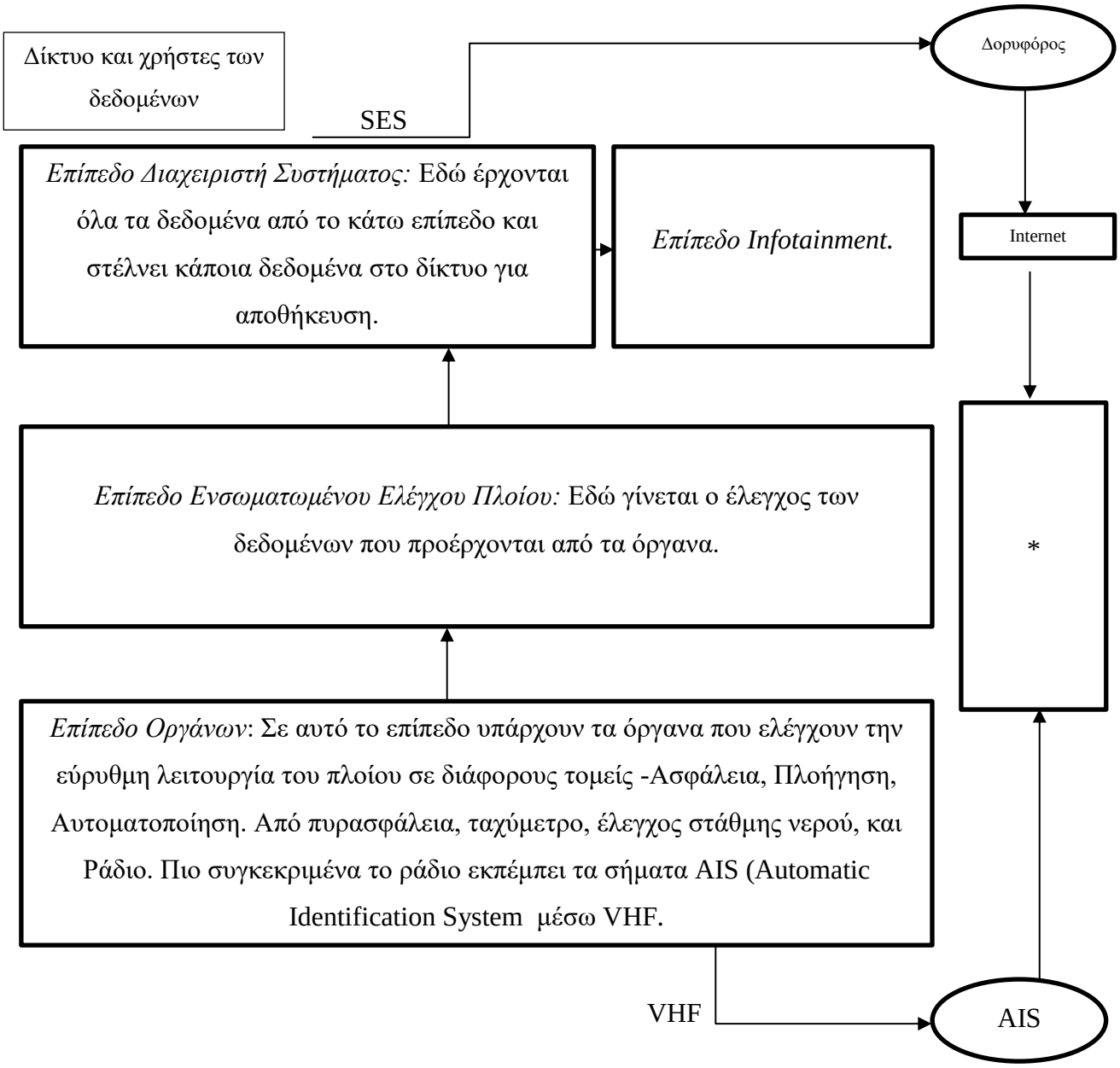
Πέραν όμως των Ναυτιλιακών Πληροφοριακών Συστημάτων, με χρήσεις όπως για την πλοήγηση στη θάλασσα, υπάρχουν και τα ευρύτερου αντικειμένου Γεωπληροφοριακά Συστήματα για το Θαλάσσιο Περιβάλλον, τα οποία έχουν και άλλες εφαρμογές στο χώρο. Σημαντικά δεδομένα που εξετάζονται και σχετίζονται με την πλοήγηση, αποτελούν τα βαθυμετρικά, τα οποία ανήκουν στον ευρύτερο τομέα της εφαρμογής της Γεωπληροφορικής στο θαλάσσιο περιβάλλον, και θα αναλυθούν στην συνέχεια. Επιπλέον, δεδομένα για στρατιωτικές ασκήσεις, βιότοπους, καιρικά φαινόμενα αλλά και άμεσους κινδύνους προς τα πλοία -π.χ. πειρατικές ενέργειες– είναι και αυτά άκρως χρήσιμα. Τέλος, ένα Γεωπληροφοριακό Σύστημα για το Θαλάσσιο Περιβάλλον μπορεί να μη συνοδεύεται απαραίτητα και με χαρτογράφηση των δεδομένων του, όμως δεν θα ασχοληθούμε με τέτοιες περιπτώσεις στο πλαίσιο της εργασίας μας.

Όπως βέβαια σε κάθε επιστημονικό τομέα, για να προχωρήσουμε στην έρευνα και στην σωστή μελέτη των αποτελεσμάτων, πρέπει να ορίσουμε κάποιες σταθερές. Αυτές προέκυψαν μέσα από διάφορα προβλήματα τα οποία η εφαρμογή της επιστήμης της Γεωπληροφορικής στο θαλάσσιο περιβάλλον χρειάστηκε να επιλύσει [2]:

- *Λάθος υπολογισμοί*
- *Ασυμμετρία Δεδομένων*

- *Σύγχυση*
- *Εσκεμμένη Αμέλεια*
- *Μη αναμενόμενα αποτελέσματα από την εφαρμογή του Γεωπληροφοριακού Συστήματος.*

Έτσι, τελικά, προκύπτει ένα γενικό πολυεπίπεδο αρχιτεκτονικό μοντέλο Πληροφοριακών Συστημάτων. Αυτά τα επίπεδα συνεργάζονται με σκοπό την ολοκληρωμένη και ασφαλή λειτουργία κάθε πλοίου, στο οποίο χρησιμοποιούνται. [Σχήμα 2].



Σχήμα b: Γενική εικόνα τωρινής και μελλοντικής αρχιτεκτονικής Ναυτιλιακών Υπολογιστικών Συστημάτων σε πλοία [2].

**Στο επίπεδο αυτό υπάρχουν οι πλοιοκτήτες, εταιρείες, ρυθμιστικές αρχές, μάνατζερ κλπ.*

Κατά τη δημιουργία ενός Γεωπληροφοριακού Συστήματος για το Θαλάσσιο Περιβάλλον, το σημαντικότερο, ίσως, κομμάτι είναι αυτό της εύρεσης, φιλτραρίσματος και χρήσης των δεδομένων. Τα στάδια, από την εύρεση μέχρι την χρήση των δεδομένων, είναι τα εξής [3] [2]:

1. *Εύρεση Δεδομένων:* Τα δεδομένα πρέπει να προέρχονται από αξιόπιστες πηγές και να ακολουθούν συγκεκριμένα frameworks και προδιαγραφές, έτσι ώστε να είναι αξιοποιήσιμα. Σημαντική σημείωση είναι ότι τα δεδομένα μας μπορεί να είναι σε διαφορετικές μορφές, όπως XML ή GeoJSON (γεωχωρική πληροφορία μέσω αρχείου Javascript Object Notataion), NetCDF (πολυδιάστατα αρχεία που εμπεριέχουν επιστημονική πληροφορία), CSV (Comma Separated Files) αρχεία, GeoTiff (Tag Image File Format που εμπεριέχει γεωχωρική πληροφορία) αρχεία που παρουσιάζουν εικόνες, ShapeFiles και άλλα.

   Τα δεδομένα που μελετώνται στη Ναυτιλιακή Πληροφορική μπορούμε να τα βρούμε σε σταθμούς ελέγχου Λιμανιών, σταθμούς που λαμβάνουν τα σήματα AIS, μετεωρολογικούς και ωκεανογραφικούς σταθμούς και αισθητήριες συσκευές που ελέγχουν διάφορες παραμέτρους του σκάφους.
2. *Προεπεξεργασία Δεδομένων:* Σε αυτό το στάδιο τα δεδομένα διαβάζονται, μελετώνται και καθαρίζονται από οποιονδήποτε θόρυβο μέσω τεχνικών εξομάλυνσης ή regression, διορθώνονται. αν υπάρχουν λάθη που χρήζουν επιδιόρθωσης. Τα θαλάσσια δεδομένα, όπως και τα γεωγραφικά, απαιτούν εξαιρετική ακρίβεια και καθαρότητα, έτσι ώστε να χρησιμοποιηθούν, πολύ δε μάλλον στον τομέα της ναυτιλίας, στον οποίο διακυβεύονται χρήματα και ανθρώπινες ζωές.
3. *Αποθήκευση δεδομένων:* Τα δεδομένα μετά την αρχική επεξεργασία πρέπει να αποθηκευτούν. Η αποθήκευση αυτή μπορεί να γίνει σε μια σχεσιακή ή μη Βάση Δεδομένων (PostgreSQL/MongoDB), ένα Σύστημα Κατανεμημένων Αρχείων (Hadoop), ή ένα γεωχωρικό σύστημα, όπως η PostGIS.

4. *Χρήση Δεδομένων:* Αυτό μπορεί να συμβεί είτε με απλή μελέτη των ερωτημάτων της βάσης δεδομένων, ανάλυση δεδομένων με λογισμικά, εύρεση μοτίβων στα δεδομένα, ομαδοποίηση των αποτελεσμάτων αναζήτησης, αλγορίθμους μηχανικής μάθησης, απεικόνιση, χαρτογράφηση είτε με άλλα, ανάλογα με την εφαρμογή.

Συμπερασματικά λοιπόν, και κλείνοντας αυτή την υποενότητα μπορούμε να δούμε τη στενή σχέση που έχουν τα Γεωγραφικά με τα Θαλάσσια δεδομένα. Η χρήση, διαχείριση και επεξεργασία των δεδομένων και των δύο κλάδων γίνονται με παρόμοιο τρόπο, ενώ και τα δύο είδη δεδομένων είναι άκρως ευαίσθητα σε λάθη και θορύβους. Κατά συνέπεια, η παραγωγή Πληροφοριακών Συστημάτων, που να συνδυάζει τους κλάδους, κρίνεται αναγκαία και απολύτως χρήσιμη.

3. ΓΕΩΓΡΑΦΙΚΑ ΜΟΝΤΕΛΑ ΔΕΔΟΜΕΝΩΝ

Στα προηγούμενα κεφάλαια είδαμε ότι τα Γεωγραφικά και Θαλάσσια δεδομένα είναι από τα πιο σημαντικά και μελετημένα είδη δεδομένων στην ιστορία της ανθρωπότητας. Αυτό, φυσικά, δεν θα μπορούσε να λείψει από τη σημερινή εποχή της τεχνολογικής άνθησης και ψηφιοποίησης.

Οι ηλεκτρονικοί υπολογιστές ως μηχανήματα είναι ιδανικοί για την αποθήκευση και επεξεργασία δεδομένων σε μεγάλη κλίμακα. Για την αποθήκευση αυτή, αλλά και τη μετέπειτα ανάγνωση των δεδομένων χρησιμοποιούν το δυαδικό σύστημα αρίθμησης, δηλαδή αποθηκεύουν οποιαδήποτε πληροφορία ως σειρά από μηδέν και ένα.

Έτσι και τα υπό μελέτη δεδομένα μας για την κατασκευή ενός Πληροφοριακού Συστήματος δεν θα μπορούσαν να αποτελέσουν εξαίρεση στον κανόνα. Κάθε χρήσιμη πληροφορία για την επιφάνεια της Γης μετατρέπεται τελικά σε στοιχείο από μια βάση δεδομένων γεωγραφικής πληροφορικής και εν τέλει σε κάποιο συνδυασμό από μηδενικά και ένα.

Για να γίνει αυτή η μετατροπή δεδομένων και στην συνέχεια η τελική παρουσίαση στο χρήστη, έχουν παραχθεί διάφορα πρότυπα, όπως είναι τα JPEG, PNG, TIFF για εικόνες, MPEG για βίντεο και MP3 για ήχο. Κάποια από αυτά τα πρότυπα χρησιμοποιούνται και για τη μοντελοποίηση γεωγραφικών δεδομένων, όπως θα δούμε στην συνέχεια.

Η ψηφιοποίηση, λοιπόν, αυτών των δεδομένων σε δυαδικό σύστημα είναι χρήσιμη για πολλούς λόγους:

- Τα δεδομένα είναι σε ένα κοινό πρότυπο και έτσι δεν χρειάζονται πολύπλοκες «μεταφράσεις», που μπορούν να οδηγήσουν σε λάθη.
- Τα δεδομένα μπορεί να κβαντιστούν, δηλαδή να δημιουργηθούν πακέτα δεδομένων, τα οποία είναι εύχρηστα και εύκολο να αποσταλούν.
- Μπορεί να γίνει εύκολη κωδικοποίηση και αποκωδικοποίηση των δεδομένων, έτσι ώστε ευαίσθητα δεδομένα να μεταφέρονται και να διαβάζονται από συγκεκριμένα άτομα.
- Εύκολα διαβάζονται από ένα μεγάλο εύρος συσκευών που λειτουργούν χρησιμοποιώντας το δυαδικό σύστημα.

- Τέλος, κυρίως για τα γεωγραφικά δεδομένα, η ψηφιοποίηση βοηθάει στον ευκολότερο χειρισμό, ανάλυση και μελέτη των δεδομένων μας. Είναι η πρώτη φορά στην μακρόχρονη ιστορία των γεωγραφικών δεδομένων που μπορούμε να αλλάξουμε τόσο εύκολα κλίμακα, να κάνουμε συγκρίσεις και να λάβουμε ζωντανά δεδομένα, χωρίς την ανάγκη δημιουργίας νέου χάρτη από την αρχή.

Για να μελετήσουμε, βέβαια, τα γεωγραφικά μοντέλα δεδομένων, χρειαζόμαστε ένα κοινό σημείο αναφοράς, το οποίο είναι η λεγόμενη γεωαναφορά. Η γεωαναφορά (αγγλ. Georeferencing), αποτελεί τον ακρογωνιαίο λίθο κάθε Γεωγραφικού Συστήματος είτε αυτό είναι Πληροφορικής, είτε μιλάμε απλά για εκτυπωμένους χάρτες και οδηγεί στην συνέχεια στη δημιουργία των γεωγραφικών μοντέλων δεδομένων.

Η γεωαναφορά είναι τόσο σημαντική για τρεις συγκεκριμένους λόγους. Από τη μία, αποτελεί ακριβέστατη πληροφορία και είναι συσχετισμένη με μια συγκεκριμένη τοποθεσία πάνω στην Γη. Από την άλλη, τα δεδομένα αυτά είναι κοινά και προσιτά σε όποιον επιλέξει να τα μελετήσει Τέλος, σημαντικότατο στοιχείο της γεωαναφοράς είναι ότι παραμένει σταθερή στο πέρασμα του χρόνου, χωρίς να υπάρχουν σημαντικές αλλαγές που θα καταστήσουν τα δεδομένα μας αναξιόπιστα στο μέλλον.

Παραδείγματα γεωαναφοράς αποτελούν το σύστημα αναφοράς με το γεωγραφικό πλάτος και το γεωγραφικό μήκος, που αντιστοιχίζει επ’ ακριβώς όλα τα σημεία με συντεταγμένες, αλλά και το σύστημα οδών-αριθμών σε μία πόλη.

Μπορούμε να έχουμε διάφορα είδη γεωαναφοράς, ανάλογα την εφαρμογή και τα δεδομένα μας.

- *Τοπογραφική ονομασία.* Αποτελεί τον πιο απλό τρόπο γεωαναφοράς που μπορούμε να έχουμε. Μπορούμε να δώσουμε όνομα σε οποιοδήποτε τοπογραφικό χαρακτηριστικό, από ένα βουνό, έναν ωκεανό, έναν δρόμο, ένα χωριό, αξιοθέατα μέχρι και ένα δέντρο. Δεν αποτελεί τον βέλτιστο τρόπο γεωαναφοράς, καθότι μπορεί να υπάρξουν γεωγραφικοί τόποι οι οποίοι έχουν την ίδια επονομασία. Επιπλέον, το όνομα ενός τόπου μπορεί να διαφέρει από

ομάδα ατόμων σε ομάδα ατόμων –π.χ. το όνομα μίας οροσειράς που βρίσκεται ανάμεσα σε δύο χώρες.

- *Ταχυδρομικοί κώδικές και ταχυδρομικές διευθύνσεις.* Δημιουργήθηκαν κατά την κατασκευή των πρώτων ταχυδρομείων και μπορούν να είναι άκρως αξιόπιστος τρόπος γεωαναφοράς για κατοικίες και γραφεία, αλλά όχι για γεωγραφικά χαρακτηριστικά, αφού δεν έχουμε χωρικές συντεταγμένες.
- *IP διευθύνσεις.* Κάθε υπολογιστής έχει μια IP διεύθυνση που τον καθιστά μοναδικό. Μπορεί να γίνει άμεση συσχέτιση με μια κατά προσέγγιση τοποθεσία. Πρόβλημα παρουσιάζεται σε ομαδοποιημένους υπολογιστές τοπικών δικτύων, αλλά και σε υπολογιστές που χρησιμοποιούν ιδιωτικό εικονικό δίκτυο.
- *Γραμμικά συστήματα αναφοράς.* Από ένα δίκτυο σημείων και χαρακτηριστικών, μετράει την απόσταση ενός τόπου από ένα σημείο αναφοράς του δικτύου (π.χ. μία διασταύρωση). Χρησιμοποιούνται ευρέως στον χώρο των μεταφορών και στα GPS (Global Positioning System).
- *Γεωγραφικό Πλάτος και Γεωγραφικό Μήκος.* Ίσως ο πιο διαδεδομένος, αλλά και ο πιο ακριβής τρόπος αναπαράστασης και αναφοράς γεωγραφικών τόπων. Για να υπολογιστεί, χρησιμοποιούμε το σχήμα της Γης και μετράμε τις αποστάσεις από τον ισημερινό και τον μεσημβρινό του Greenwich. Από εκεί και πέρα κάθε τοποθεσία πάνω στην επιφάνεια της Γης μπορεί να αναπαρασταθεί από τα δύο αυτά δεδομένα. Μπορούμε, έτσι, να βρούμε σημεία, να μετρήσουμε αποστάσεις και να πλοηγηθούμε σε κάθε μέρος της επιφάνειας της Γης και στη θάλασσα, με ακρίβεια. Το πρόβλημα του συστήματος είναι ότι το σχήμα της Γης δεν επιτρέπει την ακριβή μεταφορά των δεδομένων αυτών σε χάρτες και άλλα δεδομένα (πχ raster εικόνες και φωτογραφίες) που πρέπει να είναι επίπεδα. Αυτό οδηγεί σε θορυβώδη δεδομένα και σε πολλά διαφορετικά συστήματα συντεταγμένων, όπως αναφέραμε και στην εισαγωγή.

  Η λύση του προβλήματος εμπίπτει στην δημιουργία προβολών που διατηρούν συγκεκριμένες πληροφορίες –διατήρηση σχημάτων και αντίστοιχο μέγεθος επιφανειών Γης με χάρτη- ανέπαφες. Οι προβολές μπορεί να είναι κυλινδρικές, κωνικές και αζιμουθιακές και πρακτικά αποτελούν αντιστοίχιση

των συντεταγμένων σε ένα «χαρτί» τοποθετημένο (αντίστοιχα με το όνομα της προβολής) σε σχέση με τη Γη.

Όπως αναφέραμε, λοιπόν, η γεωαναφορά αποτελεί το βασικότερο, ίσως, κομμάτι για την δημιουργία μοντέλων γεωγραφικών δεδομένων, τα οποία με τη σειρά τους θα οδηγήσουν στην παραγωγή του Γεωπληροφοριακού Συστήματος Δεδομένων.

Αρχικά, πρέπει να δώσουμε έναν ορισμό στον όρο γεωγραφικά μοντέλα δεδομένων. Μοντέλα γεωγραφικών δεδομένων αποτελούν συγκεκριμένες δομές, οι οποίες χρησιμοποιούνται στα Συστήματα Διαχείρισης Γεωγραφικών Βάσεων Δεδομένων και μέσω αυτών γίνεται αναπαράσταση αντικειμένων και διαδικασιών σε ψηφιακό περιβάλλον. Αυτά αποτελούν, από την πλευρά του χρήστη, την περιγραφή του συστήματος ή των δεδομένων, ενώ, από την πλευρά του προγραμματιστή του συστήματος, μια δομή του συστήματος που θα υλοποιηθεί στην εφαρμογή.

Παρ' όλα αυτά, τα μοντέλα δεν αποτελούν εύκολη και άμεση πανάκεια για το ζήτημα της παραγωγής Γεωπληροφοριακού Συστήματος για το Θαλάσσιο Περιβάλλον. Τα δεδομένα είναι πολύ μεγάλων διαστάσεων, ενώ παράλληλα οι εφαρμογές εξυπηρετούν πολλούς χρήστες με διαφορετικά επίπεδα χρήσης. Ως εκ τούτου, ο δημιουργός του συστήματος καλείται να λάβει κάποιες αποφάσεις για τον τρόπο και το επίπεδο μοντελοποίησης των δεδομένων του πραγματικού κόσμου στο σύστημα διαχείρισης της βάσης δεδομένων του.

Μια εκ των αποφάσεων αποτελεί η επιλογή του κατάλληλου επιπέδου αναπαράστασης των δεδομένων από την κανονική ζωή σε διάφορα μοντέλα και τελικά σε στοιχεία των πινάκων της βάσης δεδομένων. Για να γίνει αυτό, η μοντελοποίηση περνάει από 3 στάδια:

- *Ορισμός Δεδομένων.* Σε αυτό το σημείο ορίζουμε τα δεδομένα μας, αλλά και τους τύπους αυτών.
- *Λογική Μοντελοποίηση.* Εδώ, τα δεδομένα από το πρώτο στάδιο συνδέονται λογικά με διαγράμματα και λίστες. Αυτά υποδεικνύουν τη συμπεριφορά των οντοτήτων, αλλά και τις μεταξύ τους σχέσεις.
- *Φυσική Μοντελοποίηση.* Σε αυτό το στάδιο, τα δεδομένα μετατρέπονται σε πίνακες (ή αρχεία, αν έχουμε μη σχεσιακή βάση δεδομένων) βάσης

δεδομένων, όπου αποθηκεύονται ψηφιακά οι σχέσεις των οντοτήτων μεταξύ τους.

Το κάθε δεδομένο, έτσι, μοντελοποιείται με βάση κάποιο από τα παρακάτω μοντέλα γεωγραφικών δεδομένων και δημιουργεί μια οντότητα. Οντότητες που ανήκουν στην ίδια κατηγορία-μοντέλο (πχ όλα τα vectors που είναι σημεία), μπορούν να ομαδοποιηθούν και να δημιουργήσουν ένα επίπεδο (*αγγλ. Layer*).

Μοντέλο Δεδομένων	*Επεξήγηση*
Raster/Grid	Για χωρική ανάλυση δεδομένων.
Image	Απλή ανάλυση εικόνας και grid.
Vector	Εμφάνιση γεωγραφικών χαρακτηριστικών στον χάρτη, σύμφωνα με τις γεωμετρικές ιδιότητες της κάθε οντότητας.
Object	Συνδυασμός των παραπάνω.
Network	Κυρίως για ανάλυση δικτύων μεταφορών.
CAD (Computer-aided Design)	Αυτοματοποιημένα μηχανολογικά σχέδια.

Πίνακας b: Οι πιο γνωστές μοντελοποιήσεις γεωγραφικών δεδομένων.

Τα δεδομένα, λοιπόν, που χρησιμοποιούνται για την παραγωγή ενός Γεωπληροφοριακού Συστήματος για το Θαλάσσιο Περιβάλλον ακολουθούν συγκεκριμένα πρότυπα μοντελοποίησης [Πίνακας b], κάποια εκ των οποίων χρησιμοποιήσαμε στην εργασία μας και θα αναλύσουμε παρακάτω.

1. *Μοντέλα Εικόνων.* Χρησιμοποιούνται εναέριες φωτογραφίες και φωτογραφίες από δορυφόρους, οι οποίες δέχονται επεξεργασία και μετά χρησιμοποιούνται. Συνήθως δεν περιέχουν από μόνες τους γεωγραφική πληροφορία, οπότε είτε χρησιμοποιούνται ως σημεία αναφοράς για μία συγκεκριμένη εφαρμογή -π.χ. έξυπνη γεωργία και μελέτη αγροκτήματος- είτε τοποθετούνται σε πλέγμα γεωγραφικών συντεταγμένων για γεωγραφική χρήση.
2. *Γραφικά μοντέλα.* Τα μοντέλα αυτά προέρχονται από τη χαρτογράφηση. Αυτό μπορεί να προκύπτει είτε από την δημιουργία και παραγωγή ψηφιακών

χαρτών από άλλα δεδομένα είτε από σκανάρισμα και ψηφιοποίηση φυσικών χαρτών.

3. *Raster μοντέλα.* Μοιάζει αρκετά με το Image model, παρ' όλα αυτά έχει λειτουργικές διαφορές. Αρχικά, αποτελεί το ίδιο μια εικόνα, η οποία εμπεριέχει επιπλέον πληροφορία (metadata), χωρίς να στηρίζεται σε μία εικόνα. Δηλαδή, χρησιμοποιεί έναν πίνακα από εικονοστοιχεία, σε κάθε εγγραφή του οποίου αποθηκεύονται συγκεκριμένα δεδομένα, όπως από 0 ή 1, αν θέλουμε να αποθηκεύσουμε δυαδικά δεδομένα, μέχρι και αριθμούς διπλής ακρίβειας, αν θέλουμε να αποθηκεύσουμε άλλα δεδομένα (πχ βαθυμετρικά στοιχεία).

   Αρνητικό χαρακτηριστικό αυτού του τύπου μοντελοποίησης για κάποιες εκ των εφαρμογών, στις οποίες χρησιμοποιείται, αποτελεί η μεγάλη απαίτηση που έχει για μνήμη υπολογιστή. Πρόβλημα το οποίο αντιμετωπίζεται, με τη χρήση συμπιεσμένων μορφών του αρχείου.

   Επιπλέον χαρακτηριστικό, σε σχέση με τα μοντέλα εικόνων, είναι ότι αυτόματα τα raster μοντέλα τοποθετούνται σε γεωγραφικό πλέγμα και έτσι διατηρείται η χρήσιμη πληροφορία του γεωγραφικού πλάτους και μήκους.

   Τέλος, τα συγκεκριμένα δεδομένα μπορούν να αναλυθούν και να ομαδοποιηθούν πολύ εύκολα, καθώς ο μεγάλος τους όγκος και η ακρίβεια τους επιτρέπει τεχνικές ανάλυσης δεδομένων.

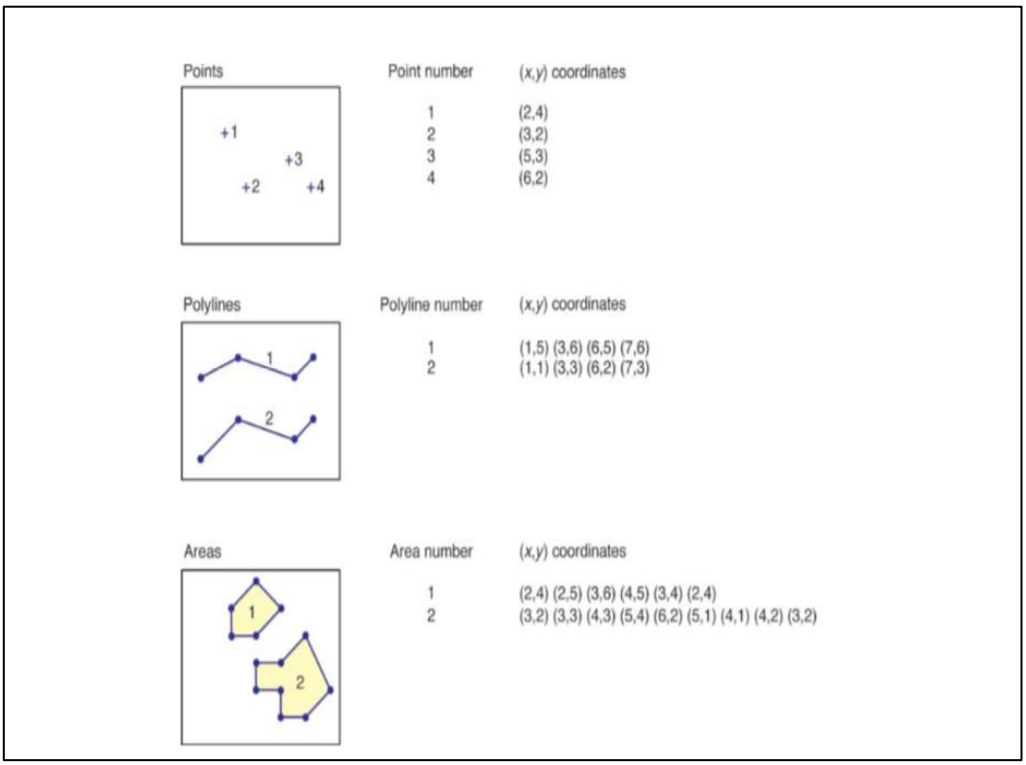
4. *Μοντέλα vector (διανυσμάτων).* Για δεδομένα στατικά που έχουν να κάνουν γενικά με την αντίληψη του χώρου και των χαρακτηριστικών του. Έχουν κάποιες συγκεκριμένες ιδιότητες που τα καθιστούν ιδανικά για τη χρήση σε συστήματα γεωγραφικών βάσεων δεδομένων.

   a. Είναι εξαιρετικά ακριβή στις μετρήσεις τους.
   b. Είναι εξαιρετικά αποδοτικά από θέμα μνήμης.
   c. Είναι εύκολα διαχειρίσιμα με οποιοδήποτε εργαλείο ανάλυσης και επεξεργασίας γεωγραφικών δεδομένων.
   d. Είναι εύκολο να παραχθούν μέσα από κάποια άλλη πληροφορία (π.χ. έχοντας τα γεωγραφικά μήκη και πλάτη μπορούμε να μοντελοποιήσουμε ένα σημείο).

e. Μπορούν να είναι δισδιάστατα (x,y συντεταγμένες), τρισδιάστατα (x,y,z: ύψος) ή και τετραδιάσταστα (x,y,z και w μια άλλη παράμετρος) ανάλογα με την πληροφορία που θέλουμε να μοντελοποιήσουμε.

Τα δεδομένα που μοντελοποιούνται με την χρήση vector δημιουργούν τα λεγόμενα χαρακτηριστικά. Σε μία βάση γεωγραφικών δεδομένων, σε κάθε σειρά του πίνακα υπάρχει ένα χαρακτηριστικό, ενώ σε κάθε στήλη υπάρχουν ιδιότητες του χαρακτηριστικού. Τα χαρακτηριστικά, με την σειρά τους, χωρίζονται και αυτά σε κατηγορίες ανάλογα με τις ιδιότητές τους, όπως παρακάτω:

i. *Απλά χαρακτηριστικά.* Μπορεί να μοντελοποιούν απλά χωρικά σημεία, γραμμές (δηλ. σύνολο σημείων ή ένωση σημείων με βέλτιστο χωρικά τρόπο) ή πολύγωνα (δηλ. ένωση σημείων μέχρι να δημιουργηθεί «κλειστό» σχήμα). Επιπλέον, μπορούν να δημιουργηθούν πολύ-πολύγωνα (δηλ. σύνολο πολυγώνων που δημιουργεί μια ένωση) ή και πολυ-γραμμές (δηλ. σύνολο γραμμών που δημιουργούν μια ένωση).



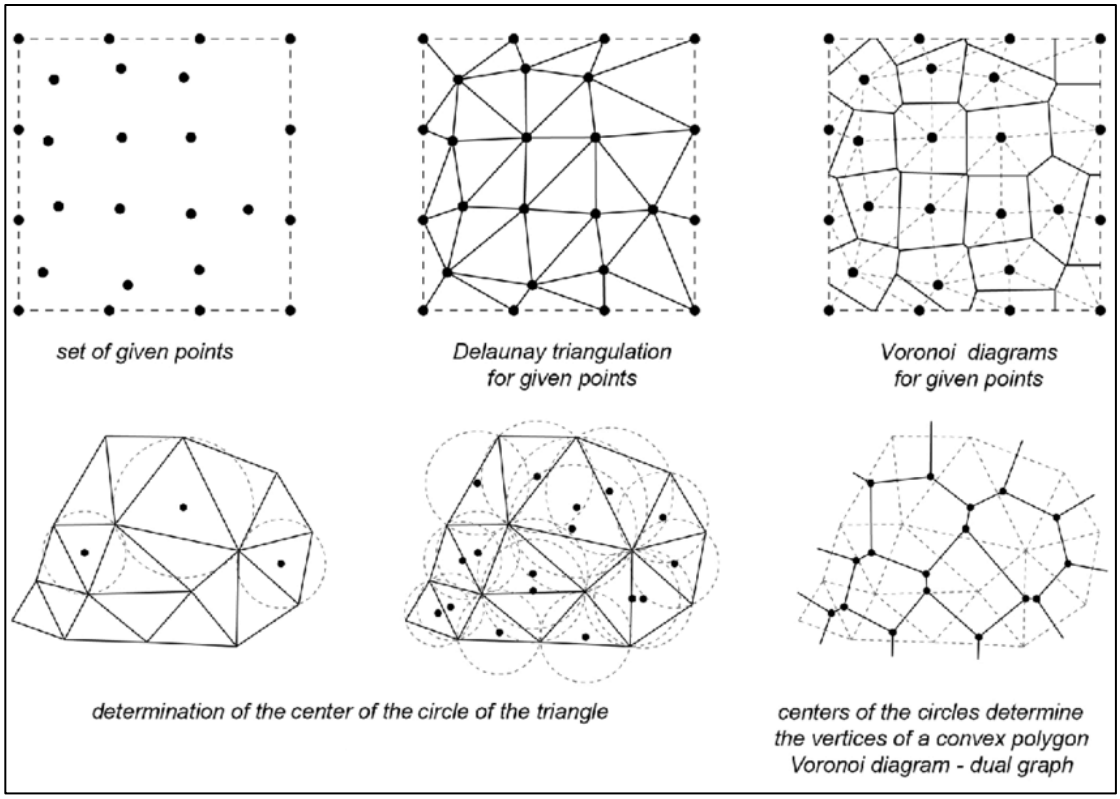
Εικόνα 1: Παράδειγμα σημείων, πολύ-γραμμών και πολυγώνων σε μοντέλα γεωγραφικών δεδομένων. Βλέπουμε την αντιστοιχία σημείων με ένα αφηρημένο σύστημα συντεταγμένων. Εικόνα από [1] (pp. 159).

ii. *Τοπολογικά χαρακτηριστικά.* Δεν είναι κάτι πέρα από απλά χαρακτηριστικά που είναι μοντελοποιημένα χρησιμοποιώντας απλώς τοπολογικούς κανόνες. Τοπολογικοί κανόνες είναι μη γραμμικές, μαθηματικές ή γεωμετρικές σχέσεις, που παραμένουν σταθερές ακόμα και αν ο χώρος παραμορφωθεί – πχ γειτνίαση ακμών σε αντίθεση με τις αποστάσεις που αλλάζουν.

Η τοπολογική αυτή προσέγγιση παίζει σημαντικό ρόλο στα παραγόμενα γεωγραφικά συστήματα πληροφορικής, αφού βοηθάει στην επικύρωση της πληροφορίας (τεστάροντας την τοπολογική συνάφεια, ελέγχουμε και την ακρίβεια των δεδομένων), στην μοντελοποίηση συμπεριφοράς εσωτερικών χαρακτηριστικών (περιοχές με κοινές ιδιότητες πχ με νερό, μπορούν να μοντελοποιηθούν μαζί σε κοινό χαρακτηριστικό), αλλά και στη βελτιστοποίηση των ερωτημάτων προς τη βάση δεδομένων μας (μπορούμε να κάνουμε βελτιστοποιημένα ερωτήματα με βάση την τομή, την ένωση, τη γειτνίαση ή και την ύπαρξη συγκεκριμένων χαρακτηριστικών μεταξύ τους).

iii. *Δικτυακά χαρακτηριστικά.* Αποτελούν χαρακτηριστικά που χρησιμοποιούνται κυρίως για δίκτυα (μεταφορών, αγωγών, οδικό κλπ.). Χρησιμοποιούν σημεία ως κόμβους, και γραμμές ως ακμές του δικτύου. Μπορεί να αναλυθεί με κλασικές μεθόδους ανάλυσης δικτύων-γράφων.

iv. *TIN χαρακτηριστικά.* Τα χαρακτηριστικά δημιουργούν ένα τριγωνικό ακανόνιστο δίκτυο. Τα δεδομένα είναι τρισδιάστατα και χρησιμοποιούνται, για να δείξουν την επιφάνεια του εδάφους. Πρακτικά, μοντελοποιούν το πλέγμα του χώρου σε τρίγωνα, με το μέγεθος και την τοπολογία τους να εξαρτάται από το λεγόμενο Delaunay triangulation [Εικόνα 2]. Αυτά τα χαρακτηριστικά, από την μία περιέχουν τα αρχικά σημεία, χρήσιμα για την τοποθεσία, αλλά και την τριγωνοποίηση του πλέγματος με διάφορη πυκνότητα, η οποία βοηθάει στην παρατήρηση του ανάγλυφου του εδάφους και των τοπικών χωρικών ιδιοτήτων.



Εικόνα 2 Delaunay triangulation και διαγράμματα Voronoi για την τριγωνοποίηση ενός χώρου. Πηγή: Rokicki, Wiesław & Gawell, Ewelina. (2016). Voronoi diagrams – architectural and structural rod structure research model optimization. MAZOWSZE Studia Regionalne. 2016. 155-164. 10.21858/msr.19.10.

Γενικά, η αποθήκευση και δημιουργία των χαρακτηριστικών μπορεί να γίνει με διάφορους τρόπους. Για παράδειγμα, μπορούμε να δημιουργήσουμε πολύγωνα από πολυ-γραμμές ή από σημεία. Επιπλέον, ο τρόπος αποθήκευσης μπορεί να διαφέρει, καθώς μπορούμε να χρησιμοποιήσουμε λίστες και πίνακες γειτνίασης για κάθε χαρακτηριστικό, ανάλογα με τα δεδομένα μας και τη διαχείριση του χώρου αποθήκευσης μας.

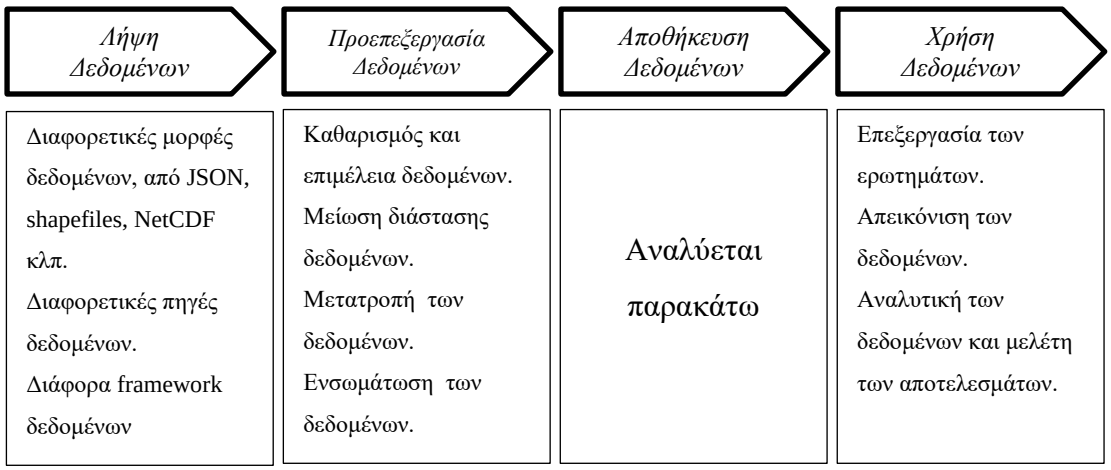
Εν κατακλείδι, λοιπόν, για την παραγωγή μιας αξιόπιστης εφαρμογής πληροφοριακών συστημάτων που έχει να κάνει με γεωγραφικά δεδομένα, αρχικά οφείλουμε να έχουμε ένα στιβαρό και αξιόπιστο σύστημα γεωαναφοράς, έτσι ώστε τα δεδομένα μας να είναι γεωγραφικώς «επικυρωμένα». Κατά δεύτερον, και ίσως σημαντικότερο σημείο σε όλη την διαδικασία υλοποίησης του συστήματος, είναι η σωστή μοντελοποίηση των δεδομένων του πραγματικού κόσμου, με σκοπό την ορθή χρήση στο σύστημα βάσεων δεδομένων μας και μετέπειτα στην εφαρμογή μας.

4. ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΓΕΩΧΩΡΙΚΩΝ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΤΟ ΘΑΛΑΣΣΙΟ ΠΕΡΙΒΑΛΛΟΝ

Τα θαλάσσια δεδομένα, όπως αναφέραμε αποτελούν αναπόσπαστο κομμάτι για την κατασκευή οποιουδήποτε ναυτιλιακού και θαλάσσιου υπολογιστικού συστήματος, ιδιαίτερα δε όταν αναφερόμαστε σε συστήματα πλοήγησης πλοίων που περιλαμβάνουν από βαθυμετρικά στοιχεία, καιρικά δεδομένα, μέχρι και ζωντανή πληροφορία για τις θέσεις των σκαφών κάθε χρονική στιγμή.

Ο μεγάλος όγκος των δεδομένων αυτών, σε συνδυασμό με την ύπαρξη γεωγραφικής πληροφορίας με την οποία συσχετίζεται, δημιουργούν δυσκολίες κατά την διάρκεια της αποθήκευσης και της μελέτης τους [4], παρά την σύνθετη διαδικασία προεπεξεργασίας, η οποία περιλαμβάνει τη λήψη και τον καθαρισμό των δεδομένων.

Τη λύση στο παραπάνω πρόβλημα έρχεται να δώσει ο κλάδος της Πληροφορικής που σχετίζεται με την ανάπτυξη και διαχείριση συστημάτων διαχείρισης δεδομένων. Αυτός μας επιτρέπει εύκολα να αποθηκεύουμε δεδομένα, αλλά και να λαμβάνουμε μέσω ερωτημάτων συγκεκριμένες απαντήσεις, οι οποίες θα χρησιμοποιηθούν στην συνέχεια για κατάλληλη χρήση των δεδομένων αυτών.



Εικόνα 3: Η ακολουθία χρήσης των θαλάσσιων δεδομένων [2](pp.314).

Τα δεδομένα μας, συνήθως, έρχονται σε διαφορετικές μορφές οι οποίες είναι ιδανικές για απλή καταγραφή, χωρίς την δυνατότητα μελέτης [5]. Αρχεία CSV, excel αλλά και αρχεία καταγραφής δεδομένων MatLab και R, αποτελούν ιδανικά περιβάλλοντα για την αποθήκευση και μελέτη των απλών σχετικά δεδομένων. Πέρα όμως από αυτά, μπορούμε να έχουμε πιο σύνθετα συστήματα καταγραφής για τα θαλάσσια-γεωγραφικά δεδομένα μας, όπως κατανεμημένα συστήματα υπολογιστών, σχεσιακές ή μη βάσεις δεδομένων και χωροχρονικά συστήματα αποθήκευσης, τα οποία συνήθως συνδυάζονται με κάποια από τα προηγούμενα, με σκοπό την ολοκληρωμένη μελέτη των δεδομένων που εμπεριέχουν γεωχωρική πληροφορία.

Παρακάτω, λοιπόν θα κάνουμε μια μικρή θεωρητική ανάλυση των συστημάτων αυτών [2]:

- *Κατανεμημένα Συστήματα Υπολογιστών:* Οι χρήστες, αλλά και τα δεδομένα είναι διαμοιρασμένα σε ένα σύστημα υπολογιστών, το οποίο συνδέεται μέσω του διαδικτύου. Κατασκευάζεται ένα ιεραρχικό δίκτυο από υπολογιστές-εξυπηρετητές, παρ' όλα αυτά η προσπέλαση των δεδομένων γίνεται ακριβώς όπως η τοπική αποθήκευση. Τα δεδομένα αποθηκεύονται σε διάφορες μορφές και σκοπός του συστήματος είναι να δώσει στον χρήστη μια εικόνα ενός ενωμένου και ολιστικού μοντέλου συστήματος αποθήκευσης, παρά τον διαμοιρασμό των πόρων.
    - Παραδείγματα Συστημάτων: HDFS, NetWare, NFS, Hadoop.
    - Πλεονεκτήματα: Τα δεδομένα είναι διαθέσιμα και αποθηκευμένα για όλους τους χρήστες, υπάρχει καταμερισμός φόρτου συστήματος και εύκολη επεκτασιμότητά του. Επιπλέον, είναι αρκετά ευέλικτο ως προς την αποθήκευση διαφορετικών μορφών δεδομένων.
    - Μειονεκτήματα: Μη δεικτοδεικτούμενο σύστημα, το οποίο είναι μη αποδοτικό για κάποιες αναζητήσεις.
- *NoSQL Key-Value Stores:* Τα δεδομένα είναι αποθηκευμένα ως συλλογή από τιμές και κλειδιά. Έχουμε τις κλασικές μορφές ερωτημάτων διαχείρισης βάσης δεδομένων –CREATE, READ, UPDATE, SELECT– με χρήση στηλών-κλειδιών.

- Παραδείγματα Συστημάτων: Dynamo, Voldemort, Riak.
- Πλεονεκτήματα: Ευκολία χρήσης, χαμηλό κόστος, ευκολία στην επέκταση του συστήματος και μικρή καθυστέρηση στις αναζητήσεις.
- Μειονεκτήματα: Μπορούν να γίνουν μόνο απλές αναζητήσεις και ολιστικές ανανεώσεις τιμών στους πίνακες.

- *NoSQL Document Stores:* Βασισμένη όχι σε σχέσεις μεταξύ αντικειμένων, αλλά σε αρχεία οργανωμένα με την χρήση κλειδιών. Αποθηκεύουν δεδομένα σε μορφές τύπου JSON χρησιμοποιώντας διάφορα API's για την ανάκτηση δεδομένων.
  - Παραδείγματα συστημάτων: MongoDB, SimpleDB, CouchDB.
  - Πλεονεκτήματα: Μπορούμε να έχουμε πολυεπίπεδη δεικτοδότηση και αναζήτηση στη βάση. Δουλεύει το σύστημα για ημιδομημένα δεδομένα. Τα δεδομένα μπορεί να είναι εμφωλευμένα ή και σαν λίστες. Αποδοτικό για πολλά δεδομένα.
  - Μειονεκτήματα: Τα δεδομένα είναι μεγάλα σε αριθμό bit λόγω της μη κανονικοποιημένης μορφής. Δεν υποστηρίζονται τομές δεδομένων. Δουλεύει καλύτερα για ημιδομημένα δεδομένα.
- *NoSQL wide-columnar Stores:* Τα δεδομένα είναι οργανωμένα σε πίνακες και οι ιδιότητες αποτελούν στήλες και οικογένειες στηλών. Οι οικογένειες στηλών θυμίζουν οργάνωση αρχείων.
  - Παραδείγματα Συστημάτων: BigTable, Cassandra.
  - Πλεονεκτήματα: Βελτιστοποίηση τοπικού caching, αποδοτικό σύστημα σε περίπτωση συμπίεσης δεδομένων, απόδοση συστήματος σε αθροιστικές αναζητήσεις, αρκετά καλό σε επεκτασιμότητα.
  - Μειονεκτήματα: Αυξημένο κόστος νέας εγγραφής, το σύστημα δεν μπορεί να κάνει συναλλαγές δεδομένων, αυξημένο κόστος ανακατασκευής πλειάδων.
- *NoSQL graph-based Stores:* Εξειδικευμένα και πολύ αποδοτικά συστήματα, αν τα δεδομένα μας είναι οργανωμένα σε γράφους. Δηλαδή, υπάρχει η παρουσίαση των δεδομένων ως κόμβοι, με τις μεταξύ τους σχέσεις να είναι οι ακμές.
  - Παραδείγματα Συστημάτων: Neo4j, ArangoDB, TitanDB.

- Πλεονεκτήματα: Δείχνει μόνο τις σημαντικές πληροφορίες των δεδομένων, αλλά και τις σχέσεις μεταξύ τους. Άκρως αποδοτικά για διασυνδεδεμένα συστήματα. Γρήγορη σχεσιακή αναζήτηση.
- Μειονεκτήματα: Βελτιστοποιημένα μόνο για γραφοκεντρικά συστήματα οργάνωσης δεδομένων. Μη αποδοτικά για μεγάλα ερωτήματα.

- *Χωροχρονικά (Spatiotemporal) Συστήματα:* Αποτελούν συστήματα διαχείρισης βάσεων δεδομένων -σχεσιακών ή μη- απόλυτα εξειδικευμένα για την αποθήκευση αντικειμένων σχετιζόμενων με τον χώρο και τον χρόνο. Τα συστήματα αυτά μπορούν να παρέχουν εργαλεία για την ανάλυση δεδομένων όπως μοντελοποιήσεις χαρακτηριστικών, αλλά και εργαλεία για γεωμετρική και γεωγραφική ανάλυση και τροποποίηση των δεδομένων. Επιπλέον, πολλές φορές λειτουργούν ως επεκτάσεις για άλλες βάσεις δεδομένων (βλ. PostGIS και PostgreSQL). Η χωροχρονική πληροφορία, που εμπεριέχουν τα αντικείμενα αυτά, μπορεί αυτόματα να ανανεωθεί. Αυτά τα συστήματα χρησιμοποιήσαμε και εμείς για τον σκοπό της εργασίας μας, όπως θα αναλυθεί σε επόμενα κεφάλαια.
  - Παραδείγματα Συστημάτων: PostGIS, PostGIS-T, GeoMESA.
  - Πλεονεκτήματα: Είναι βελτιστοποιημένα, για να κάνουν συχνές ανανεώσεις των δεδομένων και μάλιστα σε μεγάλο αριθμό. Επίσης, παρέχουν στον χρήστη αποδοτικότατες μεθόδους για την απόκτηση και αποθήκευση χωρικών δεδομένων.
  - Μειονεκτήματα: Είναι κεντρικοποιημένα και με μικρή επεκτασιμότητα.

Στη συνέχεια, μετά την χρήση ενός από τα παραπάνω συστήματα διαχείρισης βάσεων θαλάσσιων δεδομένων, έρχεται η διαδικασία των ερωτημάτων στο σύστημα, για να λάβουμε πίσω την απαραίτητη πληροφορία για την εφαρμογή μας. Η διαδικασία των ερωτημάτων αλλά και της επεξεργασίας τους γίνεται με την χρήση συστημάτων επεξεργασίας query. Αυτά αποτελούν μια διεπαφή χρήστη, η οποία μας δίνει την δυνατότητα χρησιμοποιώντας SQL ερωτήματα να λάβουμε πίσω τα δεδομένα, αλλά και να τα αναλύσουμε. Παραδείγματα τέτοιων συστημάτων είναι τα: Apache Hive, SparkSQL, Presto, Apache Impala, GeoMesa και GeoSpark.

Συμπερασματικά λοιπόν, από την μελέτη μας πάνω στα συστήματα διαχείρισης βάσεων δεδομένων -και κυρίως πάνω στα θαλάσσια δεδομένα- συμπεραίνουμε ότι για τα δεδομένα μας, τα οποία πληρούν συγκεκριμένα κριτήρια, απαιτείται διεξοδική γνώση για τις νέες τεχνολογίες διαχείρισης βάσεων δεδομένων. Ενώ, κατά την διάρκεια της κατασκευής του πληροφοριακού συστήματός μας, πρέπει να αποφασίσουμε την κατάλληλη τεχνολογία, ανάλογα με τις συνθήκες που απαιτούν οι χρήστες της εφαρμογής μας.

5. ΔΙΑΔΙΚΤΥΑΚΗ ΧΑΡΤΟΓΡΑΦΗΣΗ

I. Εργαλεία και τεχνικές για κατασκευή διαδικτυακού χάρτη

Σχεδόν όλα τα Γεωπληροφοριακά Συστήματα βασίζονται στην κατασκευή χαρτών. Κατά τα τελευταία χρόνια, με την ραγδαία ανάπτυξη του διαδικτύου ως κυρίαρχου μέσου επικοινωνίας και δικτύωσης, η επιστήμη της χαρτογράφησης εκσυγχρονίστηκε και κινήθηκε ανάλογα με τις τάσεις της εποχής. Αυτή η τάση οδήγησε στην ανάπτυξη του κλάδου της διαδικτυακής χαρτογράφησης.

Με τον όρο διαδικτυακή χαρτογράφηση (web mapping), εννοούμε την χρήση τεχνολογιών διαδικτύου, με σκοπό την δημιουργία ενός δικτυακά προσβάσιμου λογισμικού, το οποίο εμπεριέχει γεωγραφική πληροφορία πάνω σε χάρτη. [1] (pp. 142)

Τα συστήματα αυτά συνήθως βασίζονται σε μια client–server αρχιτεκτονική, χρησιμοποιώντας έναν GIS (Geographic Information System) Server, για να μεταδώσει στο χρήστη τα δεδομένα που χρειάζεται κάθε στιγμή μέσω διάφορων υπηρεσιών. Τα δεδομένα αυτά είναι αποθηκευμένα σε μια βάση δεδομένων, δισδιάστατη ή τρισδιάστατη ως προς τις γεωγραφικές προβολές, και η απεικόνιση τους προς τον χρήστη γίνεται μέσα σε έναν χάρτη κατασκευασμένο με τη χρήση τεχνολογιών διαδικτύου, όπως API, frameworks και άλλα. Αλλά, ας δούμε αναλυτικά τις τεχνικές και τα εργαλεία που χρησιμοποιούν οι προγραμματιστές διαδικτυακών χαρτών, για να κατασκευάσουν τα συστήματα τους.

- *Λογισμικά GIS:* Μπορεί η κατηγορία να μην εμπίπτει αμιγώς στην κατασκευή ενός διαδικτυακού χάρτη, καθώς χρησιμοποιούνται και για άλλες χρήσεις, αλλά τα λογισμικά GIS, όπως QGIS (https://qgis.org/en/site/) και το ArcGIS (https://www.arcgis.com/index.html), είναι εξαιρετικά χρήσιμα για κάθε είδους πρότζεκτ στον τομέα των Γεωπληροφοριακών Συστημάτων. Ο χρήστης-χειριστής του Γεωπληροφοριακού Συστήματος έχει την δυνατότητα να κατασκευάσει δικό του τοπικό χάρτη, να χειριστεί και να μελετήσει τα γεωγραφικά δεδομένα, να εξάγει διαφορετικές μορφές δεδομένων, αλλά και να τα επεξεργαστεί. Επιπλέον, μπορεί να κατασκευάσει επίπεδα (layers) πάνω στον χάρτη του, για να απεικονίσει την πληροφορία με άμεσο τρόπο, ενώ τα

συστήματα αυτά του δίνουν εν δυνάμει άπειρες δυνατότητες αναλυτικής δεδομένων και απεικόνισης. Παράλληλα, σε διάφορα αντίστοιχα λογισμικά παρέχεται η δυνατότητα δημοσιοποίησης του κατασκευασμένου χάρτη ως διαδικτυακή εφαρμογή.

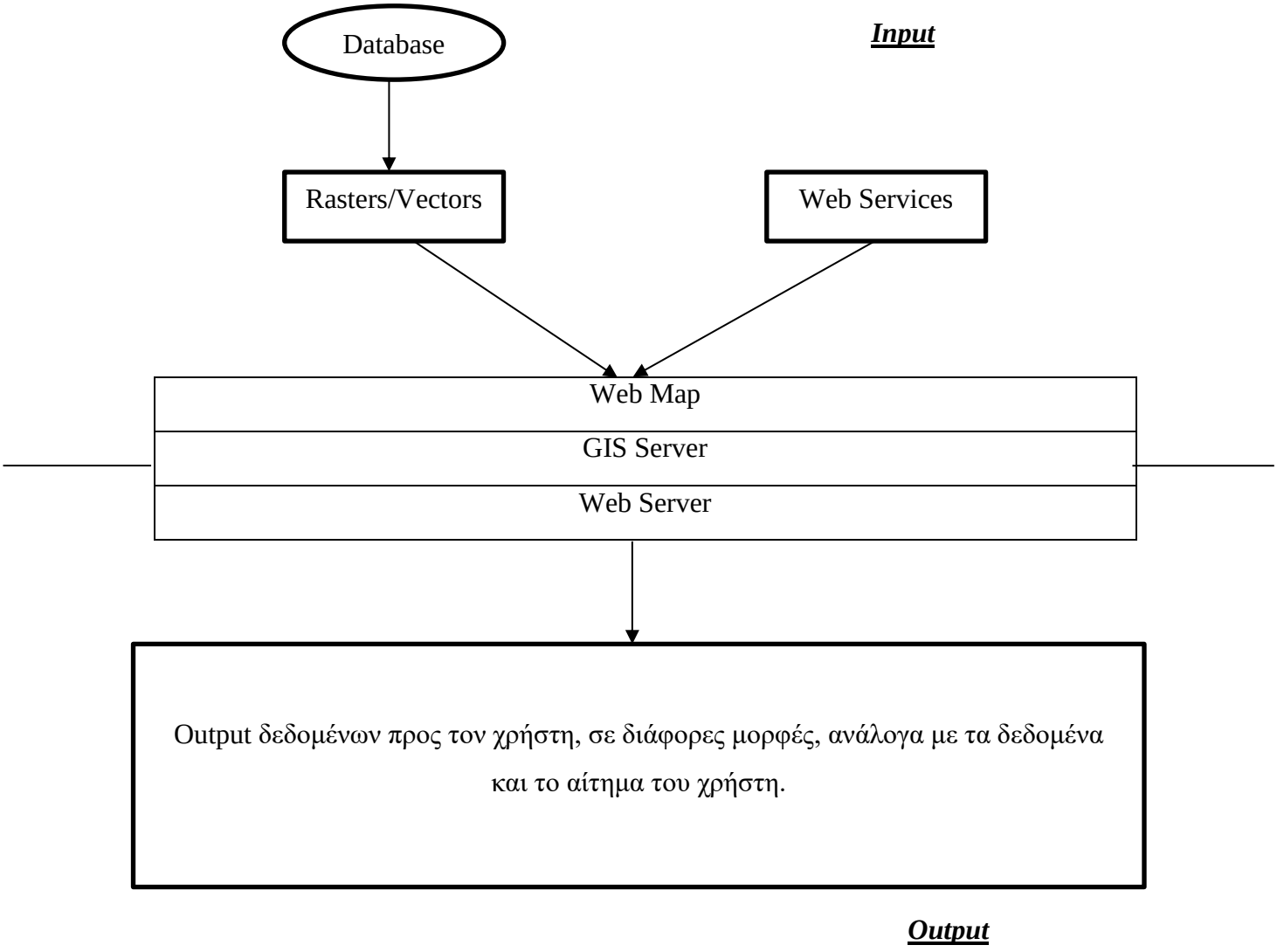
- *Βασικός χάρτης:* Για την κατασκευή ενός web map συστήματος, προφανώς, βασική προϋπόθεση είναι η ύπαρξη ενός χάρτη. Ο λεγόμενος βασικός χάρτης είναι μια δισδιάστατη απεικόνιση της Γης, σε οποιαδήποτε προβολή επιλέξει ο χρήστης. Χωρίζεται σε πλέγμα και περιέχει τη βασική γεωγραφική πληροφορία του γεωγραφικού πλάτους και μήκους προβαλλόμενη ανάλογα. Οι χάρτες αυτοί είναι εύκολα προσβάσιμοι μέσω APIs, με πολλές διαφορετικές επιλογές να είναι διαθέσιμες. Παραδείγματα base maps APIs αποτελούν τα OpenStreetMap API (https://www.openstreetmap.org), Google Maps API (https://www.google.com/maps) και άλλα.
- *Γλώσσες προγραμματισμού διαδικτύου και κατάλληλες βιβλιοθήκες:* Η δημιουργία ενός διαδικτυακού χάρτη, αποτελεί μια εργασία παρόμοια με την κατασκευή μιας ιστοσελίδας. Όπως είναι γνωστό, για την κατασκευή ιστοσελίδας χρειαζόμαστε κάποιες γλώσσες προγραμματισμού -με τις κατάλληλες βιβλιοθήκες-, αλλά και κάποιες γλώσσες σήμανσης και εμφάνισης. Έτσι, για την υλοποίηση της εφαρμογής χρειαζόμαστε [6]:
    - HTML (Hyper Text Markup Language), που είναι η βασική γλώσσα για την κατασκευή ιστοσελίδων. Περιγράφει την δομή μιας σελίδας, αλλά και πληροφορία σχετικά με την εμφάνιση. Η σελίδα αποτελείται από HTML στοιχεία και μέσω των εντολών γίνεται στοιχειοθέτηση.
    - CSS (Cascading Style Sheets), η οποία είναι η βασική γλώσσα που περιγράφει το πώς τα HTML στοιχεία θα εμφανιστούν στην οθόνη. Δουλεύει σε συνεργασία με την HTML. Μπορεί να έχουμε εξωτερικά αρχεία CSS ή και εμφωλευμένα μέσα σε HTML ή JavaScript αρχεία.
    - JavaScript, η οποία είναι η γλώσσα με την οποία ελέγχουμε την λειτουργία της ιστοσελίδας μας. Η JavaScript, είναι η πιο διαδεδομένη γλώσσα αυτή την στιγμή, καθώς αποτελεί την «καρδιά» της κατασκευής ιστοσελίδων. Χρησιμοποιεί όλα τα στοιχεία των παραπάνω γλωσσών, αλλά και επιπλέον στοιχεία, τα συνδέει με τρόπο

που θέλει ο προγραμματιστής και έτσι κατασκευάζεται η ιστοσελίδα μας.

- Βιβλιοθήκες και frameworks. Οι βιβλιοθήκες αποτελούν σημαντικό στοιχείο για την δομή μιας ιστοσελίδας, αφού μπορούν με κλήσεις API να καλέσουν, εύκολα και γρήγορα, αντικείμενα που χρειάζεται ο προγραμματιστής. Σημαντικότατες είναι και στην λειτουργία των web maps, καθότι για την κατασκευή του χάρτη απαιτείται η κλήση βιβλιοθηκών JavaScript -όπως η OpenLayers (https://openlayers.org/) και η Leaflet (https://leafletjs.com/)- για την μορφοποίηση του χάρτη. Επιπλέον, υπάρχουν τα frameworks. Τα frameworks, βοηθάνε τον προγραμματιστή να δομήσει σωστά την ιστοσελίδα του, χωρίς να χρειάζεται να δημιουργήσει τα πάντα από το μηδέν. Το bootstrap (https://getbootstrap.com/), αποτελεί ένα τέτοιο framework για δομές CSS, ενώ τα ReactJS (https://reactjs.org/) και το AngularJS (https://angularjs.org/) αποτελούν νέες αποδοτικότατες τεχνολογίες JavaScript framework.

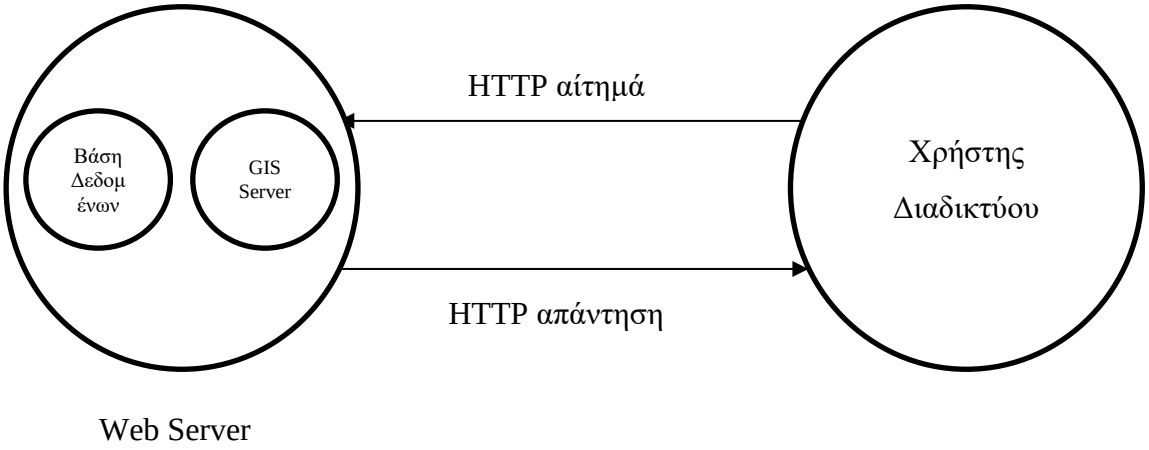
- *Βάση δεδομένων:* Για την κατασκευή ενός διαδικτυακού χάρτη πρέπει να υφίστανται δεδομένα. Η αποθήκευση αυτών των δεδομένων γίνεται σε μια δομή βάσης δεδομένων είτε σχεσιακή είτε μη σχεσιακή. Οι βάσεις δεδομένων πρέπει να είναι βελτιστοποιημένες για γεωγραφικά δεδομένα και οι επιλογές, αναφέρονται στο κεφάλαιο με τίτλο «Συστήματα Διαχείρισης Γεωχωρικών Βάσεων Δεδομένων για το Θαλάσσιο Περιβάλλον», καθώς και τα θαλάσσια δεδομένα εμπεριέχουν γεωγραφική πληροφορία. Εμείς, στην εργασία μας, χρησιμοποιήσαμε την PostgreSQL, με χρήση του εργαλείου διαχείρισης γεωγραφικών βάσεων δεδομένων, PostGIS.
- *GIS Server:* Αποτελεί σύστημα το οποίο δέχεται αιτήσεις από διάφορους χρήστες του υπολογιστικού μας συστήματος, κατασκευάζει επίπεδα χάρτη από τα δεδομένα και επιστρέφει τα γεωγραφικά δεδομένα που ο χρήστης χρειάζεται. Τα δεδομένα που αποθηκεύονται στην Βάση δεδομένων της επιλογής μας, σε συνδυασμό με διάφορες υπηρεσίες του γεωγραφικού εξυπηρετητή, όπως WMS (Web Map Service: αποτελεί HTTPS διεπαφή χρήστη, η οποία χρησιμοποιείται, για να αποσταλούν στον χρήστη γεωαναφορικές εικόνες και επίπεδα) και το WFS (Web Feature Service:

αποτελεί και αυτό HTTPS διεπαφή, με την οποία είναι δυνατόν να γίνουν συναλλαγές γεωχωρικών δεδομένων ανάμεσα στους Client και Server και μ' αυτό τον τρόπο ο χρήστης μπορεί, ανάλογα με τις προδιαγραφές του συστήματος, να δημιουργήσει, να αναγνώσει, να ανανεώσει, να διαγράψει και να κατεβάσει συγκεκριμένα μεμονωμένα χαρακτηριστικά) και να παρέχονται οι πληροφορίες μέσω αιτήματος του χρήστη. Γνωστά συστήματα GIS Server αποτελούν τα MapServer (https://mapserver.org/), ArcGIS WebServer και ο GeoServer (http://geoserver.org/).



Εικόνα 4: Γενικότερη αρχιτεκτονική συστημάτων web mapping.

- *Web Server:* Τα συστήματα GIS Server μπορούν να χρησιμοποιηθούν για την επιλογή και επεξεργασία των γεωγραφικών δεδομένων, πριν αποσταλούν στον χρήστη. Για την αποστολή αυτή, συνήθως, χρησιμοποιούμε έναν web server, που λειτουργεί σαν «κέλυφος» για τον GIS Server. Αυτός δέχεται τις αιτήσεις και χρησιμοποιεί τον GIS Server σαν HTTP Server, για να επιλέξει τα δεδομένα που θα επιστραφούν. Συνήθως στα συστήματα μας, χρησιμοποιούμε Apache (https://apache.org/) ή Apache Tomcat (https://tomcat.apache.org/).



Εικόνα 5: Σχήμα λειτουργίας χρήστη-web server, με ενσωματωμένο GIS Server και βάση δεδομένων

.

- *JQuery, AJAX και REST API:* Αυτά τα τρία αποτελούν τον «συνδετικό ιστό» μιας εφαρμογής web mapping και, γενικότερα, όλων των δυναμικών ιστοσελίδων, καθώς συνδυάζονται και συνδέουν το frontend με το backend της εφαρμογής μας.
    - *JQuery:* Αποτελεί μια βιβλιοθήκη της JavaScript, η οποία μας βοηθάει σε διαφορετικές περιπτώσεις δημιουργώντας εύκολα single-line commands. Χρησιμοποιείται για χειρισμό HTML αρχείων, CSS αρχείων, χειρισμό εντολών AJAX και άλλα.
    - *AJAX (Asynchronous JavaScript and XML):* Με το AJAX μπορούμε να φορτώσουμε δεδομένα στο background και να τα μεταφέρουμε στην ιστοσελίδα μέσω XML, χωρίς να χρειαστεί επαναφόρτωση. Έτσι

έχουμε τη δυνατότητα, μέσω jQuery εντολής, να κάνουμε ένα HTTP αίτημα από τον GIS Server με τη χρήση AJAX.

- *REST API:* Αποτελεί σχεδιαστικό μοτίβο σε επίπεδο λογισμικού. Αυτό θεωρεί τα αντικείμενα, από την πλευρά του server, ως οντότητες, οι οποίες μπορούν να προστεθούν, να διαγραφούν, να ανανεωθούν και να ληφθούν από τον εξυπηρετητή από και προς τον χρήστη του λογισμικού. Αντικείμενα, από την πλευρά του server, μπορούν να χαρακτηριστούν, για παράδειγμα, οι πίνακες μιας βάσης δεδομένων ή ακόμα και μια εγγραφή ενός πίνακα.

Γενικότερα, λοιπόν, τα παραπάνω αποτελούν τα βασικά εργαλεία, αλλά και την βασική αρχιτεκτονική για όποιον ενδιαφέρεται να σχεδιάσει και να υλοποιήσει ένα γεωγραφικό πληροφοριακό σύστημα, που βασίζεται στην διαδικτυακή χαρτογράφηση. Πρέπει, με γνώμονα πάντα το σχεδιαστικό μοντέλο που έχει επιλέξει, αλλά και την λογική της εφαρμογής του, να επιλέξει τα κατάλληλα εργαλεία, με σκοπό την αποδοτική και λειτουργική κατασκευή του συστήματος.

Πέρα από αυτά τα εργαλεία, ο δημιουργός του υπολογιστικού συστήματος οφείλει να γνωρίζει και κάποιες τεχνικές για την κατασκευή του. Αυτές, κυρίως, εμπίπτουν σε δυο κατηγορίες. Σε τεχνικές για τη λήψη και την επεξεργασία των δεδομένων και σε τεχνικές για την κατασκευή του κορμού του υπολογιστικού συστήματος. Κάποιες από τις τεχνικές είναι:

- *Χειρισμός API από διάφορες πηγές για την χρήση συγκεκριμένων δεδομένων.*
- *Χειρισμός αρχείων σημασιολογικού ιστού πχ JSON, XML*. Σε αυτά ο προγραμματιστής οφείλει να βρει την πληροφορία που χρειάζεται και να την απομονώσει.
- *Αποστολή δεδομένων από και προς μια βάση δεδομένων*. Με οποιαδήποτε γλώσσα και οποιαδήποτε βιβλιοθήκη.
- *Χειρισμός γεωγραφικών δεδομένων με βάση την τοποθεσία και την μορφή τους*. Ο προγραμματιστής μπορεί να χρειαστεί, είτε μέσω GIS εφαρμογών είτε μετά από ερωτήματα στη βάση δεδομένων, να κρατήσει συγκεκριμένη πληροφορία, αλλά και να κάνει τομές και επερωτήσεις στα δεδομένα.

- *Χειρισμός πινάκων βάσης δεδομένων*. Μπορεί να χρειαστεί να αναπτύξει συγγενικές σχέσεις μεταξύ των πινάκων ή να κάνει ενώσεις, τομές και joins σε διάφορους πίνακες.
- *Μετατροπή δεδομένων raster σε vector μέσω της χρήσης της βιβλιοθήκης GDAL* και GIS εφαρμογής ή οποιασδήποτε γλώσσας προγραμματισμού.
- *POST, GET, DELETE εντολές μέσω του REST API και AJAX*. Αυτά τα χρησιμοποιούμε για την αποστολή δεδομένων από και προς τον εξυπηρετητή, αλλά και την ορθή λειτουργία των Web υπηρεσιών του server. Αυτές μας επιστρέφουν από επίπεδα χάρτη, μέχρι και την πληροφορία του κάθε επιπέδου από την βάση δεδομένων.
- *Ευχέρεια με SQL ερωτήματα, για την εύρεση και ανάκτηση της κατάλληλης πληροφορίας.*
- *Εύρεση δεδομένων μέσα σε ιστοσελίδες*. Πολλές φορές ο χρήστης μπορεί να χρειαστεί να κάνει καταγραφή δεδομένων «με το χέρι», αλλαγή στη μορφή των δεδομένων ή και web scrapping, αν τα δεδομένα που θέλει είναι διαθέσιμα προς χρήση, αλλά σε μη επιθυμητή μορφή.
- *Και άλλα.*

Συμπερασματικά λοιπόν, μέσα από την μελέτη και την εμπειρία μας και έχοντας απομονώσει τα παραπάνω, ως κάποιες από τις βασικές τεχνικές και εργαλεία για την κατασκευή τέτοιων εφαρμογών, μπορούμε να παρατηρήσουμε ένα συγκεκριμένο πρότυπο κατασκευής, το οποίο αποτυπώνεται εξαιρετικά στα Σχήματα 4 και 5.

Έτσι, αφού ο προγραμματιστής βρει, επεξεργαστεί και αποθηκεύσει τα δεδομένα του σε μια βάση δεδομένων, δύναται να κατασκευάσει μέσω του GIS Server επίπεδα, τα οποία μέσω JavaScript βιβλιοθηκών μπορούν να προβληθούν πάνω στον βασικό μας χάρτη. Από εκεί και έπειτα, ανάλογα με τις συνθήκες των εφαρμογών, ο χρήστης έχει τη δυνατότητα να λάβει την κατάλληλη πληροφορία.

II. Παρόμοιες εφαρμογές διαδικτυακής χαρτογράφησης

Στην συνέχεια της πτυχιακής εργασίας, θα αναλυθεί η κατασκευή της εφαρμογής μας, η οποία είναι βασισμένη στα πρότερα θεωρητικά κομμάτια, αλλά σε εις βάθος μελέτη. Παρ' όλα αυτά στο συγκεκριμένο σημείο αξίζει να εξετάσουμε διάφορες παρόμοιες εφαρμογές που υπάρχουν στο διαδίκτυο και αποτέλεσαν έμπνευση, αλλά και πρότυπο για την εκπόνηση της εργασίας μας.

- *Google Maps (https://www.google.com/maps) :* Αποτελεί, ίσως, την πιο γνωστή εφαρμογή τύπου web map που κυκλοφορεί. Ο τεχνολογικός κολοσσός της Google κατασκεύασε την εφαρμογή το 2005 και από τότε αποτελεί μια τεράστια επιτυχία. Μέσα από την εφαρμογή ο χρήστης μπορεί να πλοηγηθεί, να κατασκευάσει πλάνο ταξιδιού και να βρει από εστιατόρια μέχρι ξενοδοχεία. Αρχαιολογικοί χώροι και μουσεία αποτελούν επίσης layers που είναι δυνατόν να επιλέξει να δει ο χρήστης, όπως και πολλές άλλες επιλογές. Παράλληλα, τα δεδομένα είναι συνδεδεμένα με δρομολόγια πλοίων και αεροπλάνων, ακόμα και με τιμές διοδίων. Τέλος, δίνεται η δυνατότητα στον χρήστη για επιλογή διαφορετικών βασικών χαρτών, ενώ με το Street View Project έχουν ψηφιοποιηθεί σε τρισδιάστατη μορφή ολόκληρες πόλεις.
- *OpenStreetMap (https://www.openstreetmap.org):* Κατασκευάστηκε το 2004 και αποτελεί μια από τις πρώτες, αν όχι η πρώτη, προσπάθεια κατασκευής εφαρμογής διαδικτυακής χαρτογράφησης, ανοιχτού λογισμικού. Στη σημερινή διαδικτυακή μορφή αποτελείται από έξι στρώματα-επίπεδα βασικών χαρτών, που περιέχουν διάφορες πληροφορίες και μπορούν να εναλλαχθούν, ενώ οι βασικές πληροφορίες όπως γεω-ονόματα και συγκοινωνιακό δίκτυο εμφανίζονται σε όλες. Σημαντικό χαρακτηριστικό του OpenStreetMap αποτελεί ότι παρέχει στους χρήστες μέσω API τους βασικούς χάρτες του δωρεάν, έτσι ώστε να μπορούν οι ίδιοι να δημιουργήσουν τους δικούς τους χάρτες.
- *MarineTraffic (https://www.marinetraffic.com/) και FlightRadar24 (https://www.flightradar24.com/):* Αποτελούν εφαρμογές πρότυπο για την δυναμική διαδικτυακή χαρτογράφηση και μελέτη δεδομένων μεγάλου όγκου. Από την μία το MarineTraffic αποτελεί την κυρίαρχη διαδικτυακή πλατφόρμα

για την παρακολούθηση πλοίων σε ζωντανή κάλυψη. Τα πλοία εμφανίζονται στον χάρτη ως επίπεδο και έχουν διαφορετικό χρωματισμό ανάλογα με το είδος. Επιλέγοντας το πλοίο ο χρήστης έχει τη δυνατότητα να δει διαθέσιμες πληροφορίες για αυτό, όπως η χώρα προέλευσης, η ταχύτητα αλλά και λιμάνια αναχώρησης και προορισμού. Επιπλέον πληροφορίες -όπως ιστορικά δεδομένα ταξιδιών κλπ.- μπορούν να εμφανιστούν στον χάρτη ως στρώματα, ενώ υπάρχει και η δυνατότητα κατασκευής ταξιδιού. Αντίστοιχο κρηπίδωμα, αποτελεί το FlightRadar24 για τις αεροπορικές πτήσεις, με τα δεδομένα να είναι παρόμοια (για αεροπλάνα προφανώς), όπως και οι λειτουργίες. Σε αυτή την περίπτωση δίνεται η δυνατότητα δωρεάν στατιστικών στοιχείων, όπως οι πιο συχνά αναζητούμενες πτήσεις. Και τα δύο συστήματα παρέχονται στην βασικότατη έκδοσή τους δωρεάν, ενώ επιπλέον λειτουργίες αλλά και λήψη δεδομένων μέσω API μπορούν να γίνουν επί πληρωμή.

- *European Marine Observation and Data Network(https://www.emodnet-humanactivities.eu/view-data.php):* Δημιουργήθηκε υπό την αιγίδα της Ευρωπαϊκής Ένωσης βασισμένο στην κοινή ναυτιλιακή πολιτική της. Αποτελεί στην ουσία ένα δίκτυο οργανισμών, το οποίο παρατηρεί, συγκεντρώνει, προβάλλει και αναδιανέμει τα δεδομένα της δωρεάν προς τους ενδιαφερόμενους. Τα δεδομένα ανήκουν σε μία εκ των επτά κατηγοριών, οι οποίες είναι: Βαθυμετρικά δεδομένα, δεδομένα Βιολογίας Θάλασσας, δεδομένα Χημείας Θάλασσας, Γεωλογικά Δεδομένα, Δεδομένα Ανθρώπινης Δραστηριότητας, Δεδομένα Θαλάσσιας Φυσικής και Περιβαλλοντικά Δεδομένα. Η υπηρεσία αυτή παρέχει διαδικτυακό χάρτη γι' αυτά τα δεδομένα, όπου ο χρήστης μπορεί να τα μελετήσει επί τόπου ή να τα κατεβάσει.
- *OpenWeatherMap (https://openweathermap.org/):* Το Open Weather Map αποτελεί εφαρμογή χαρτογράφησης για δεδομένα καιρού. Ο διαδικτυακός τους χάρτης περιέχει ιστορικά, αλλά και ζωντανά καιρικά δεδομένα σχετικά με την ατμοσφαιρική πίεση, την θερμοκρασία, τον αέρα, τα σύννεφα και γενικά τα καιρικά φαινόμενα. Τα δεδομένα αυτά μπορούν να εμφανιστούν ως στρώματα πάνω στον βασικό χάρτη με διαφορετικούς χρωματισμούς και εμφανίσεις, ανάλογα με τις συνθήκες. Επιπλέον, η πλατφόρμα παρέχει δεδομένα μέσω API κλήσεων στους χρήστες δωρεάν για κάποια συγκεκριμένα πακέτα χρήσης και επί πληρωμή για κάποια άλλα.

- *OpenSeaMap (https://map.openseamap.org/):* Αποτελεί την εφαρμογή που μας έδωσε περισσότερη έμπνευση, όσον αφορά παρόμοια πληροφοριακά συστήματα με το δικό μας. Αποτελεί έναν δωρεάν ναυτικό χάρτη, στον οποίο λαμβάνονται και χαρτογραφούνται ως layers πάνω στον βασικό χάρτη -τύπου Open Street Map- δεδομένα καιρού, δεδομένα από το Marine Traffic, δεδομένα από την Wikipedia, λιμάνια, βαθυμετρικά στοιχεία και άλλα. Επιπλέον, δίνεται η δυνατότητα στον χρήστη για δημιουργία πλάνου ταξιδιού, ενώ μπορεί να κατεβεί η εφαρμογή και να εγκατασταθεί σε διάφορες συσκευές για χρήση εκτός σύνδεσης.

6. ΤΟ ΠΡΟΤΕΙΝΟΜΕΝΟ ΓΕΩΠΛΗΡΟΦΟΡΙΑΚΟ ΣΥΣΤΗΜΑ ΓΙΑ ΤΟ ΘΑΛΑΣΣΙΟ ΠΕΡΙΒΑΛΛΟΝ - ΔΕΔΟΜΕΝΑ

Όπως πολλάκις έχουμε αναφέρει κατά την διάρκεια των προηγούμενων κεφαλαίων, για να μπορέσουμε να δημιουργήσουμε ένα οποιοδήποτε πληροφοριακό σύστημα και εφαρμογή, το κυριότερο που απαιτείται είναι η εύρεση κατάλληλων δεδομένων.

Έτσι, σημαντικό κομμάτι της έρευνάς μας αποτέλεσε η αναζήτηση δεδομένων τα οποία, αφενός, εμπίπτουν αμιγώς στο κομμάτι της δημιουργίας μιας εφαρμογής διαδικτυακού χάρτη της Ελληνικής επικράτειας για γεωγραφικά και θαλάσσια δεδομένα, αφετέρου, τα δεδομένα να είναι διαθέσιμα προς χρήση, να είναι δωρεάν, και, όσο το δυνατόν, διαθέσιμα για αναδιανομή στους χρήστες του συστήματος.

Παρ' όλα αυτά, στο κεφάλαιο αυτό δεν θα συζητηθεί μόνο η εύρεση των κατάλληλων δεδομένων. Θα μελετήσουμε τη διαδικασία που ακολουθείται για την τελική χρήση των δεδομένων που αναφέρεται στο κεφάλαιο «Συστήματα Διαχείρισης Γεωχωρικών Βάσεων Δεδομένων για το Θαλάσσιο Περιβάλλον», στη σελίδα 31-εικόνα 3 και, κυρίως, τα τρία πρώτα στάδια, δηλαδή της συλλογής, της (προ)επεξεργασίας και της αποθήκευσης των δεδομένων μας.

Η εικόνα 3 της σελίδας 31.

I. Αποθήκευση δεδομένων

Αρχικά πρέπει να αναφερθούμε στο σύστημα διαχείρισης βάσεων δεδομένων που χρησιμοποιήσαμε. Η βάση δεδομένων της επιλογής μας ήταν η PostgreSQL, με την χρήση του εργαλείου διαχείρισης γεωγραφικών βάσεων δεδομένων PostGIS για τη βέλτιστη διαχείριση δεδομένων με γεωγραφική πληροφορία. Αυτή η επιλογή δεν έγινε τυχαία, καθώς η PostgreSQL αποτελεί μία από τις πιο ισχυρές βάσεις δεδομένων «ανοιχτού κώδικα».

Η PostgreSQL επιλέχθηκε, καθώς αποτελεί μια ισχυρή και αξιόπιστη σχεσιακή βάση δεδομένων. Επιπλέον, μας δίνει την δυνατότητα να χειριστούμε περίπλοκες γεωμετρικές δομές, όπως σημεία, γραμμές, πολύγωνα, πολυ-γραμμές και πολυ-πολύγωνα, ενώ μπορεί να χειριστεί εξαιρετικά αρχεία, όπως τα JSON, GeoJSON και τα XML. Τέλος, αυτό το σύστημα διαχείρισης βάσεων δεδομένων μας δίνει την δυνατότητα αποδοτικής κληρονομικότητας μεταξύ πινάκων, αλλά και διάσπασης πινάκων με βάση συγκεκριμένα χαρακτηριστικά, μία ιδιότητα την οποία χρησιμοποιήσαμε για κάποια δεδομένα μεγάλου όγκου.

Από την άλλη, η επέκταση του συστήματος διαχείρισης βάσεων δεδομένων, «PostGIS» μας έδωσε την δυνατότητα να χειριστούμε την γεωγραφική πληροφορία μας βέλτιστα, αφού μπορούμε να κάνουμε συγκεκριμένα ερωτήματα στη βάση, τα οποία σχετίζονται με την γεωγραφική τοποθεσία, αλλά και την ιδιότητα της γεωμετρίας που εμπεριέχεται στην PostgreSQL.

Η ολική παρακολούθηση και διοίκηση του συστήματος μας γίνεται με τη χρήση του προγράμματος PgAdmin. Το PgAdmin λειτουργεί σε έναν τοπικό server στη «θύρα» 5432 και αποτελεί ένα σύστημα, στο οποίο μπορούμε να παρακολουθήσουμε τις συνδέσεις που γίνονται, να ελέγξουμε τα σχήματα βάσεων δεδομένων μας, τους πίνακες μέσα σε αυτές, αλλά και τις συναρτήσεις (triggers και trigger functions) και τις σχέσεις μεταξύ των πινάκων -π.χ. σε περίπτωση κληρονομικότητας μεταξύ δυο ή παραπάνω πινάκων- και πολλά άλλα.

Η εισαγωγή των δεδομένων στην βάση μας, κατά την διάρκεια παραγωγής του συστήματος μας, έγινε με τέσσερις τρόπους:

- *Εισαγωγή δεδομένων με χρήση QGIS.* Το QGIS, όπως αναφέραμε και σε προηγούμενο κεφάλαιο, αποτελεί ένα σημαντικότατο εργαλείο στα χέρια

οποιουδήποτε προγραμματιστή και κατασκευαστή Γεωπληροφοριακών Συστημάτων. Συνδέσαμε απευθείας τη βάση δεδομένων μας στο QGIS, με το οποίο είναι δυνατόν, αφού γίνει επεξεργασία των δεδομένων με οποιονδήποτε τρόπο, τα δεδομένα να εξαχθούν σαν shapefile (μορφή αρχείων τα οποία αποθηκεύουν γεωχωρική πληροφορία σε μορφή διανύσματος, με κάθε γεωγραφική πληροφορία να αποθηκεύεται ως γεωμετρία και να αποτελεί στοιχείο του διανύσματος αυτού) και να εισαχθούν σε ένα σχήμα της Βάσης Δεδομένων μας ως πίνακας.

- *Εισαγωγή με PostGIS shapefile insertion γραφικής διεπαφής χρήστη (GUI).* Κατ' αντιστοιχία ανάλογος τρόπος με τον προηγούμενο, μόνο που σε αυτή τη διεπαφή χρήστη απλώς επιλέγουμε το shapefile που θέλουμε και στη συνέχεια, αφού προσθέσουμε το SRID –κωδικός που αντιστοιχεί στην προβολή που θέλουμε τα δεδομένα μας να εισαχθούν στη βάση δεδομένων, στη βάση μας χρησιμοποιούμε το σύστημα συντεταγμένων WGS84, το οποίο χρησιμοποιείται και στα GPS – κάνουμε Import κατευθείαν τα δεδομένα.
- *Εισαγωγή με ερώτημα- εισαγωγή μέσω python script.* Στην περίπτωση αυτή έχουμε συνήθως ζωντανή ροή δεδομένων που έρχεται από κάποιο API ή δεδομένα, τα οποία προσθέτουμε εμείς «χειροκίνητα» στην βάση δεδομένων μας. Τα δεδομένα συνήθως έρχονται σε διάφορες μορφές από κάποια πηγή και προστίθενται στη βάση με κάποιο SQL ερώτημα INSERT ή UPDATE. Αν χρησιμοποιήσουμε python script, κάνουμε χρήση της βιβλιοθήκης-αντάπτορα psycopg2 (https://www.psycopg.org/), για να κάνουμε σύνδεση/αποσύνδεση, να ελέγξουμε τα δεδομένα μέσα στην βάση μας μέσω κάποιων κέρσορων εισαγωγής (cursors), οι οποίοι δίνονται μέσω της βιβλιοθήκης, αλλά και να δημιουργήσουμε συναρτήσεις trigger για διάφορους πίνακες του σχήματος μας.
- *Εντολές της βιβλιοθήκης GDAL, raster2pgsql:* Με τη χρήση της εντολής raster2pgsql της βιβλιοθήκης GDAL (βιβλιοθήκη που χρησιμοποιείται για μετατροπές δεδομένων σε διάφορες γεωχωρικές μορφές, είτε raster είτε vector, https://gdal.org/index.html), μπορούμε να εισάγουμε ένα raster αρχείο -οποιασδήποτε μορφής, που περιέχει εικόνες με γεωγραφικά μεταδεδομένα- και να δημιουργήσει έναν πίνακα raster που θα περιέχει πρακτικά τη γεωγραφική πληροφορία της εικόνας.

II. Συλλογή και επεξεργασία δεδομένων

Σε αυτό το σημείο θα αναλύσουμε το συνολικό dataset που χρησιμοποιήσαμε για την εφαρμογή μας. Θα αναφέρουμε τί είναι το κάθε δεδομένο, την πηγή, πιθανή επεξεργασία που χρειάστηκε για να απομονώσουμε τον επιθυμητό τύπο για εισαγωγή στη βάση και τις κατάλληλες συντεταγμένες, σχέσεις με άλλους πίνακες-οντότητες, αλλά και τον τρόπο εισαγωγής στο σύστημα διαχείρισης βάσεων δεδομένων [7] [8] [9].

- *Γεωγραφικά ονόματα.* Πηγή: [10]. Τα δεδομένα μας σε αυτή την περίπτωση ήταν σε CSV -αρχείο που η πληροφορία είναι χωρισμένη με κόμματα-, η οποία περιέχει τη χώρα προέλευσης με ISO ονομασία, συντεταγμένες και όνομα. Αρχικά, μέσω QGIS μετατρέψαμε το αρχείο csv σε shapefile και το εισαγάγαμε στη βάση δεδομένων. Για να μπορέσουμε να κρατήσουμε τα γεωονόματα για τον Ελλαδικό χώρο, κάναμε SELECT * INTO geonames_grc FROM geonames WHERE iso3=’GRC’

geonames_grc

gid	iso3	latitude	longitude	label	sublabel	geom
integer	varchar	numeric	numeric	varchar	varchar	geometry

- *Περιφέρειες, δήμοι και αποκεντρωμένες διοικήσεις.* Πηγή: [11]. Αποτελούν και αυτά αμιγώς γεωγραφικά δεδομένα που δείχνουν τις αυτόνομες διοικητικές μονάδες της χώρας. Τα αρχεία από την πηγή μας ερχόταν ως shapefile, το οποίο γίνεται import μέσω της διεπαφής που αναφέραμε νωρίτερα. Στην συνέχεια προχωρήσαμε σε έναν καθαρισμό στη βάση δεδομένων, μέσω της χρήσης του iso ονόματος της χώρας, όπως στα γεωονόματα.

greece_states (κάποιες από τις στήλες)

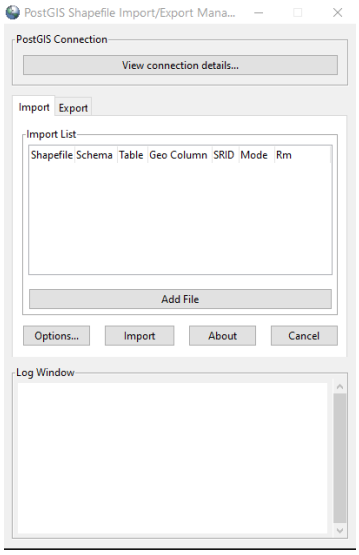
gid	adm1_code	latitude	longitude	name	Geom
integer	varchar	double precision	double precision	varchar	Geometry

apokentromenes_dioikiseis

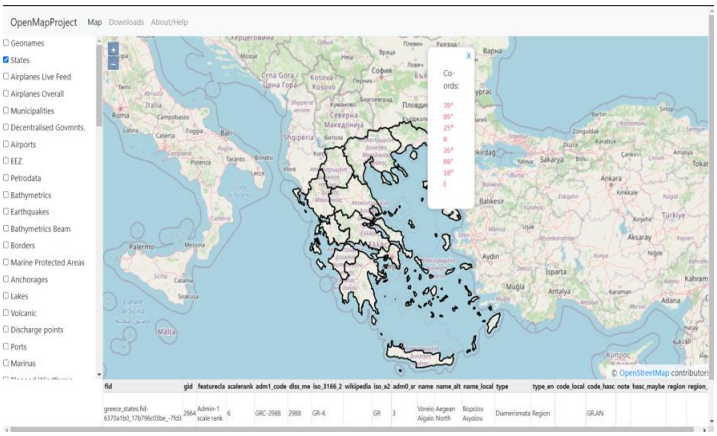
gid	gid_0	name_0	gid_1 (UNIQUE)	name_1	varname_1	nl_name	type_1	geom
integer	varchar	varchar	varchar	varchar	varchar	varchar	Varchar	geometry

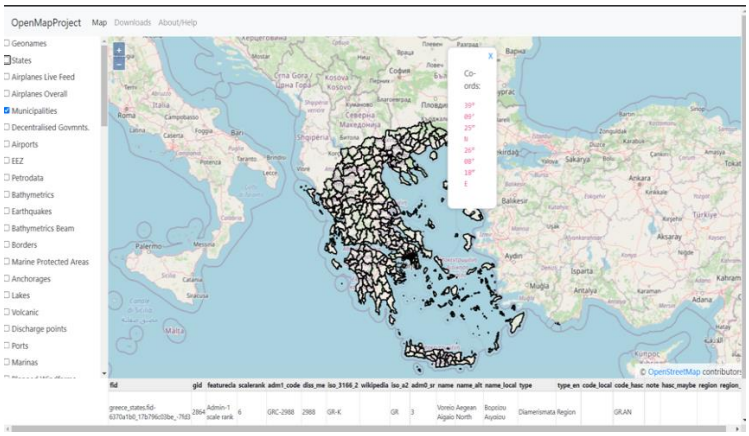
dimoi (Κάποιες στήλες)

Gid	gid_1 (FK apokentromenes_dioikiseis gid_1)	name_1	gid_0	name_0	geom
Integer	varchar	varchar	varchar	varchar	geometry



Εικόνα 6: Γραφικό περιβάλλον χρήστη για εισαγωγή shapefile. Κάνουμε την σύνδεση με τα στοιχεία της βάσης. Add file-> το αρχείο, και στην στήλη SRID βάζω το id της προβολής μου. Εμάς, ισούται με 4326 που αντιστοιχεί στην WG84. Import.

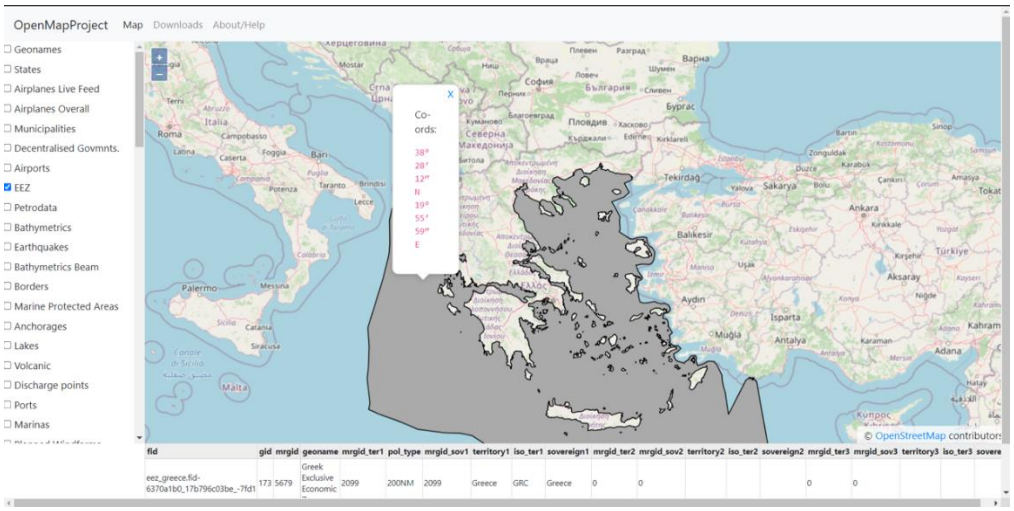




- *Αποκλειστική οικονομική ζώνη.* Πηγή: [11]. Η αποκλειστική οικονομική ζώνη της χώρας θεωρείται η θαλάσσια έκταση, εντός της οποίας ένα κράτος έχει δικαίωμα έρευνας ή άλλης εκμετάλλευσης των θαλασσίων πόρων, συμπεριλαμβανομένης της παραγωγής ενέργειας από το νερό και τον άνεμο. Είναι από την ίδια πηγή με τις περιφέρειες, δήμους και αποκεντρωμένες Διοικήσεις και εισάγεται στη βάση δεδομένων με παρόμοιο τρόπο.

eez_greece (Κάποιες στήλες)

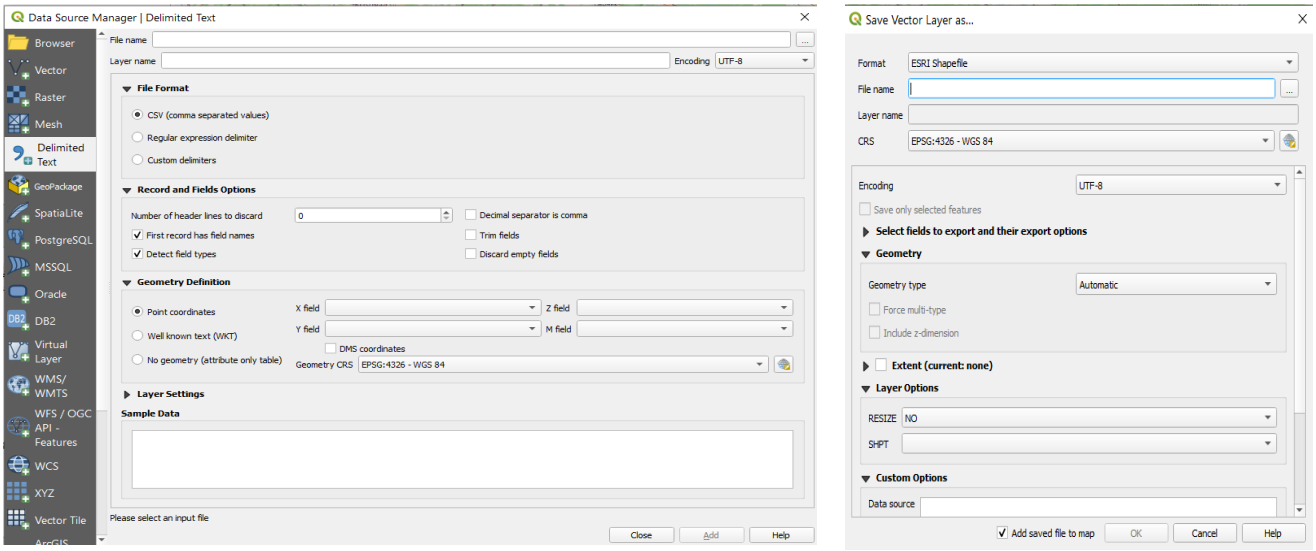
<u>Geoname</u>	territory1 (UNIQUE)	iso_ter1 (UNIQUE)	sovereign1	mrgid_ter1	geom
Varchar	varchar	varchar	varchar	numeric	geometry



- *Αεροδρόμια.* Πηγή [12]. Τα δεδομένα για τα αεροδρόμια, σύμφωνα με χάρτες του open street map. Για να κρατήσουμε τα δεδομένα που είναι σε μορφή csv file, κάνουμε την μετατροπή μέσω QGIS σε ESRI shapefile, και στη συνέχεια, αφού γίνει εισαγωγή στη βάση, κάνουμε πάλι καθαρισμό μέσω οποιασδήποτε SQL εντολής, μέσω ISO ονομασίας χώρας.

airp_grc (Κάποιες στήλες)

id	ident	name	type	iso_countr	geom
bigint	varchar	varchar	varchar	varchar	geometry



Εικόνα 7: Σύστημα QGIS, εισαγωγή αρχείου csv αριστερά [Layer->Add Layer->Add Delimited Text Layer], μετατροπή σε shapefile [Export Layer->Save Feature As…->ESRI shapefile] δεξιά. Βλέπουμε ότι μπορούν να μελετηθούν και άλλες μορφές αρχείων στο σύστημα, ενώ δεξιά βλέπουμε τις ρυθμίσεις για shapefile, όπως επιλογή προβολής, επιλογή γεωμετρίας, κλπ.

- *Πετρελαϊκά δεδομένα.* Πηγή: [13]. Τα δεδομένα για τις πετρελαϊκές αποθέσεις σε στεριά και θάλασσα. Τα δεδομένα έρχονται σε δύο dataset, ξεχωριστά για τη στεριά και τη θάλασσα. Αυτά είναι ήδη σε μορφή shapefile και αποτελούν γεωμετρίες πολυγώνων. Για να μπορέσουμε να ενώσουμε τα δεδομένα, κάνουμε ένα καθαρισμό μεταξύ των πινάκων, σύμφωνα με το όνομα της χώρας (δηλ. SELECT * FROM πίνακα WHERE country=’Greece’) και κάνουμε ένα UNION μεταξύ των αποτελεσμάτων των SELECT.

petrodatagrc (Κάποιες στήλες)

primkey	country (FK eez territory1)	fipscode	name	res	resinfo	geom
varchar	varchar	varchar	varchar	varchar	varchar	geometry

- *Βαθυμετρικά δεδομένα- 1.* Πηγή [14]. Το Nippon foundation, το οποίο σκοπεύει μέχρι το 2030 να χαρτογραφήσει το σύνολο του ωκεανού, δίνει τα δεδομένα σε πολλές διαφορετικές μορφές, από shapefile, μέχρι και raster εικόνες. Τα συγκεκριμένα βαθυμετρικά δεδομένα είναι κατασκευασμένα ως βαθυμετρικά chart. Σε αυτή τη κατηγορία δεδομένων, τα βαθυμετρικά κατασκευάζονται από ισομετρικές καμπύλες ενώνοντας πρακτικά καταγεγραμμένα σημεία (σύμφωνα με το βάθος) και με αυτό τον τρόπο δημιουργείται ένας χάρτης βαθυμετρικών δεδομένων εύκολα και με αρκετή ακρίβεια. Τα δεδομένα μας τα λάβαμε σε raster αρχείο. Για να κρατήσουμε τα δεδομένα της Ελλάδας, χρησιμοποιήσαμε το QGIS, το οποίο μας δίνει τη δυνατότητα να κάνουμε clip μια εικόνα raster, με τα γεωεπεξεργαστικά εργαλεία του. Στην συνέχεια, φορτώσαμε στη βάση τα δεδομένα, με τη χρήση της βιβλιοθήκης GDAL και την εντολή raster2pgsql.

bathymetrics

Rid	raster
Integer	raster

- *Βαθυμετρικά δεδομένα- 2.* Πηγή: [15]. Πέρα από τα βαθυμετρικά δεδομένα από ισομετρικές καμπύλες, μπορούμε να έχουμε και βαθυμετρικά δεδομένα από άμεση καταγραφή, μέσω ακουστικών ακτινών. Αυτές αντανακλούν στον πυθμένα του χώρου που μελετάμε και ο χρόνος επιστροφής μας καταδεικνύει το βάθος [16]. Οι μελέτες αυτές, παρότι είναι ακριβέστατες, είναι πολύ πιο δύσκολο να γίνει καταγραφή, καθώς πρέπει να υπάρχει συνεχής ιχνηλάτηση του βυθού. Στην συγκεκριμένη περίπτωση, λάβαμε από την πηγή μας πολλές διαφορετικές καταγραφές για τον χώρο της Μεσογείου μέσω διαφορετικών raster αρχείων τύπου GeoTiff. Για να απομονώσουμε την επιθυμητή

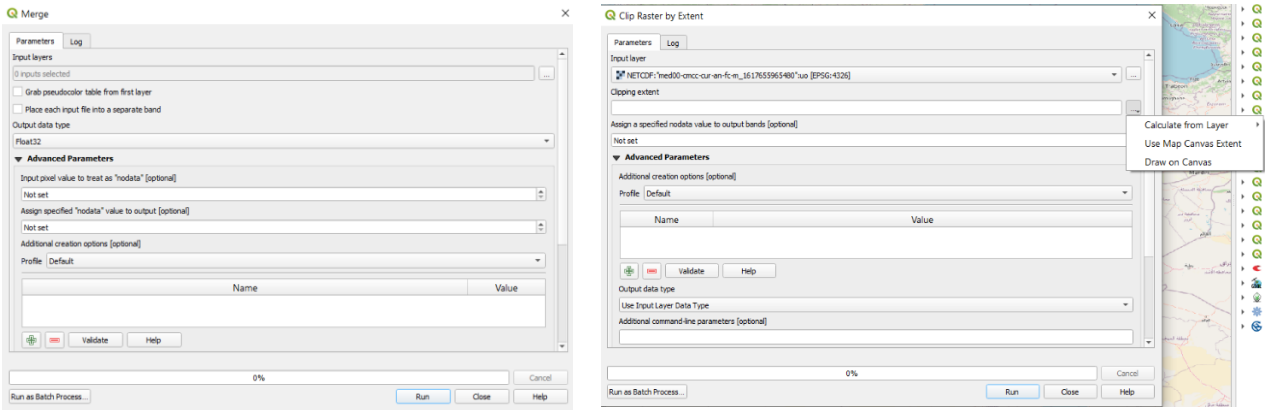
πληροφορία, κάναμε, μέσω του QGIS και των εργαλείων γεωεπεξεργασίας raster αρχείων, merge τις εικόνες σύμφωνα με την προβολή WG84 και στη συνέχεια κάναμε clip για έναν συγκεκριμένο γεωγραφικό χώρο και δημιουργήσαμε ένα νέο geoTiff αρχείο, που στη συνέχεια έγινε εισαγωγή με το raster2pgsql. Επειδή η ποιότητα της εικόνας των δεδομένων μας δεν ήταν καλή, μετά την επεξεργασία αποφασίσαμε να χρησιμοποιήσουμε την εντολή ST_PixelAsPoints της PostGIS. Με τη χρήση της εντολής αυτής γίνεται μετατροπή κάθε pixel της εικόνας σε γεωμετρικό σημείο, το οποίο περιλαμβάνει ούτως ή άλλως μεταδεδομένα (γεωγραφικά και μη) από το GeoTiff αρχείο.

Η εντολή που χρησιμοποιήσαμε:

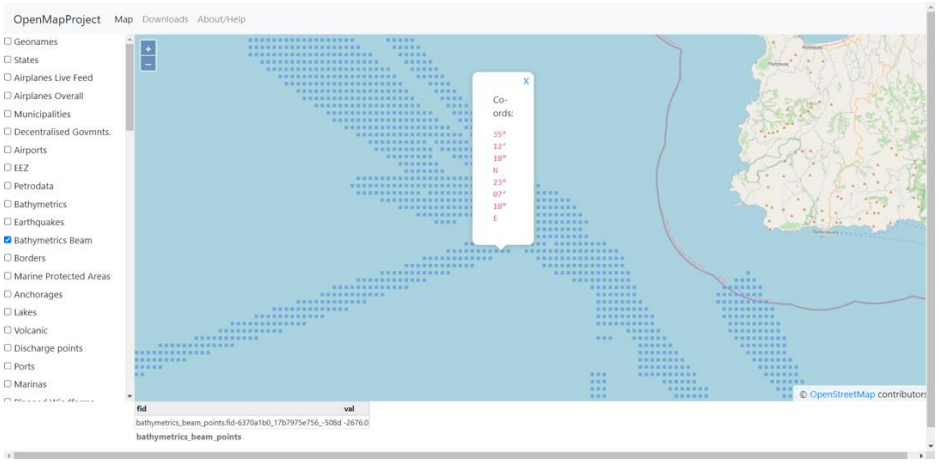
SELECT val, geom INTO public.bathymetrics_beam FROM (SELECT (ST_PixelAsPoints(rast,1)).* FROM raster.bathymetrics_beam WHERE rid=2) foo;

bathymetrics_beam_points

id	geom	val
integer	geometry	integer



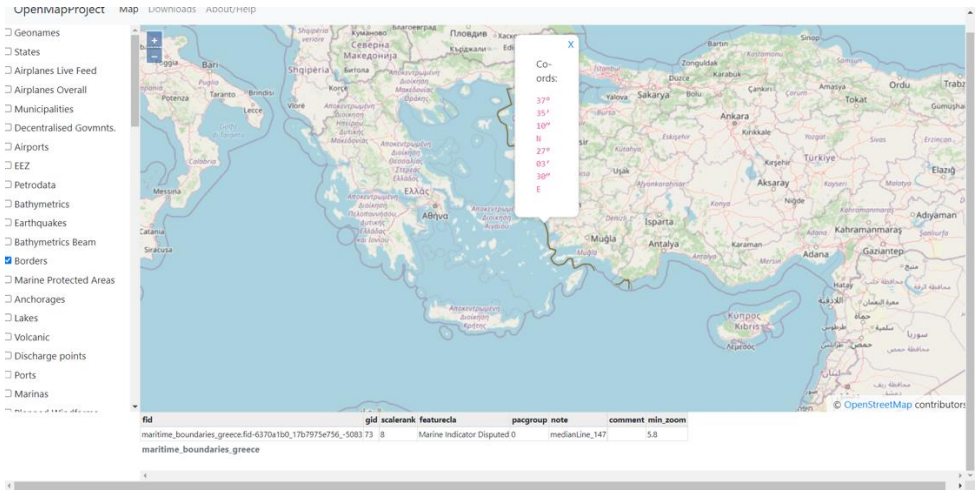
Εικόνα 8: Αριστερά, ένωση raster αρχείων, επιλέγουμε input layers με τις παραμέτρους που επιθυμούμε και run. Δεξιά, περικοπή ενός raster αρχείου, πατάμε την περικοπή ανάλογα με τις συνθήκες που επιθυμούμε και τρέχουμε το πρόγραμμα. Σημαντικό είναι ότι δεν χάνεται ούτε πληροφορία για τις προβολές, ούτε μεταδεδομένα του raster αρχείου.



- *Σύνορα.* Πηγή: [11]. Τα σύνορα της χώρας μας. Τα δεδομένα ήταν σαν shapefile και η επιλογή έγινε στη βάση με καθαρισμό μέσω iso ονόματος της χώρας.

maritime_boundaries_greece

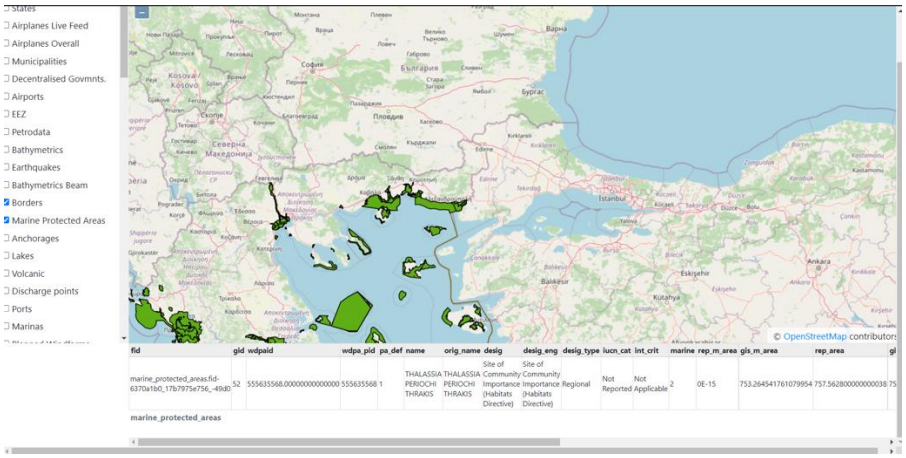
gid	Scalerank	featurecla	pacgroup	country (FK απο eez_greece iso_ter1)	note	min_zoom	geom
integer	Bigint	varchar	smallint	varchar	varchar	double precision	geometry



- *Προστατευόμενες περιβαλλοντικές περιοχές θάλασσας*. Πηγή: [17]. Τα δεδομένα μας δίνουν τις περιοχές, στις οποίες ζουν προστατευόμενα θαλάσσια είδη, όπως και περιοχές του δικτύου Natura. Τα δεδομένα λαμβάνονται σε διάφορες (τα λάβαμε σε shapefile) μορφές και είναι διαχωρισμένα ήδη για κάθε χώρα, οπότε γίνεται εισαγωγή μέσω της διεπαφής για το σύστημα συντεταγμένων WG84.

marine_protected_areas (Κάποιες στήλες)

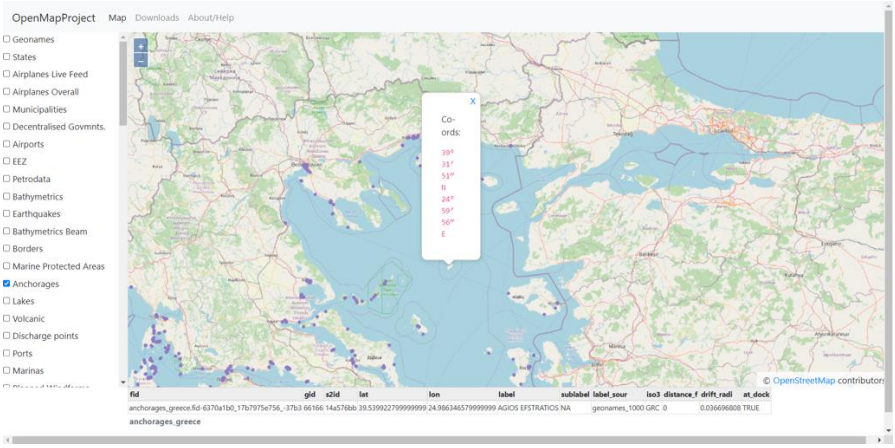
wdpa_pid	name	orig_name	iso3 (FK eez_greece iso_ter1)	design_eng	geom
varchar	varchar	varchar	varchar	varchar	geometry



- *Σημεία αγκυροβόλησης.* Πηγή: [18]. Τα δεδομένα αυτά έχουν να κάνουν με τα σημεία, στα οποία τα πλοία αγκυροβολούν κοντά σε λιμάνια. Αυτά λαμβάνονται σε αρχείο csv, το οποίο μετατρέψαμε σε shapefile μέσω του QGIS – ως προς το σύστημα συντεταγμένων WG84, πάντα – και το εισαγάγαμε στη βάση δεδομένων. Για να κρατήσουμε τα δεδομένα για την Ελλάδα, χρησιμοποιήσαμε ως σημείο αναφοράς το iso όνομα της χώρας, δηλαδή το ‘GRC’, μέσα σε ένα ερώτημα SQL.

  SELECT * INTO anchorages_grc FROM anchorages WHERE iso3=’GRC’;

  DROP TABLE anchorages;



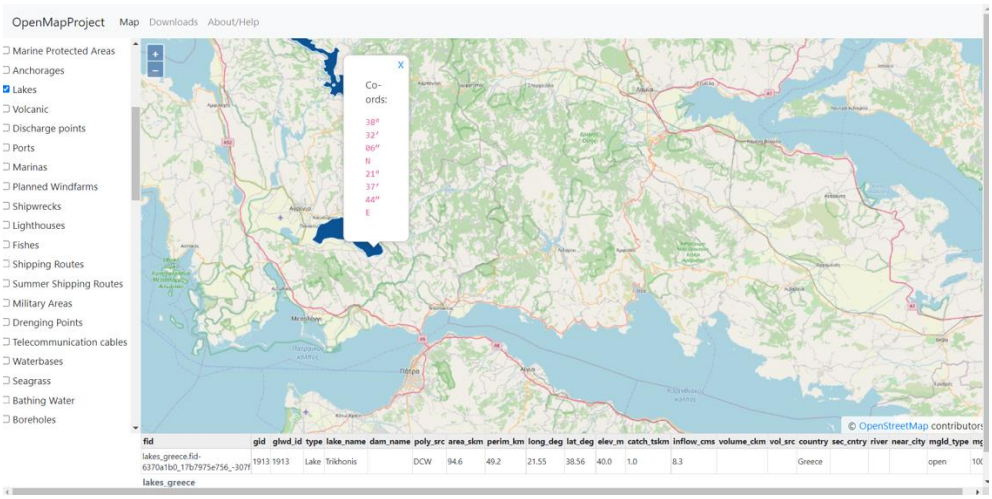
anchorages_greece (Κάποιες στήλες)

s2id	lat	lon	iso3 (FK eez_greece iso_ter1)	geom
varchar	numeric	numeric	varchar	geometry

- *Λίμνες και υγρά στοιχεία.* Πηγή: [19]. Αυτά τα δεδομένα μας δίνουν τις σημαντικές λίμνες και υγρά στοιχεία που υπάρχουν. Τα δεδομένα είναι σε μορφή shapefile και η επιλογή για την Ελλάδα έγινε με SQL ερώτημα στη βάση, DELETE * FROM lakes WHERE country NOT LIKE 'Greece'. Αυτό αποκλείει όλα τα ονόματα χωρών που δεν είναι Greece και τα διαγράφει.

lakes_greece (Κάποιες στήλες)

glwd_id	lake_name	dam_name	county	area_skm	geom
integer	varchar	varchar	varchar	double precision	geometry



- *Ηφαίστεια.* Πηγή: [20]. Τα δεδομένα μας σε αυτή την περίπτωση έχουν να κάνουν με την καταγεγραμμένη ηφαιστειακή δραστηριότητα. Αφού αναζητήσουμε τα ηφαίστεια για την Ελλάδα, τα δεδομένα μας δίνονται σε μορφή XML αρχείου. Για να μπορέσουμε να κάνουμε καθαρισμό και εισαγωγή στη βάση, διαβάζουμε το αρχείο με Microsoft Excel, έτσι ώστε να κρατήσουμε την κατάλληλη γεωγραφική πληροφορία. Τέλος, μέσω του QGIS, μετατρέψαμε τα δεδομένα σε shapefile και τέλος κάναμε εισαγωγή στην βάση δεδομένων.

volcanic_greece

gid	volcano_nu	volcano_na	country (FK eez_greece territory1)	last_known	region	geom
integer	varchar	varchar	varchar	varchar	varchar	geometry

- *Σημεία εκφόρτωσης λυμάτων.* Πηγή: [21]. Αυτό το σετ δεδομένων μας καταδεικνύει σημεία στα οποία γίνεται η εναπόθεση των λυμάτων είτε σε στεριά είτε σε θάλασσα. Τα δεδομένα μας είναι διαθέσιμα σε διάφορες μορφές, παρόλα αυτά εμάς μας ευνοεί η χρήση shapefile. Έτσι, κάνουμε εισαγωγή το αρχείο με SRID = 4326, το οποίο αντιστοιχεί σε WG84 σύστημα συντεταγμένων και στη συνέχεια καθαρισμό μέσω SQL ερώτημα.

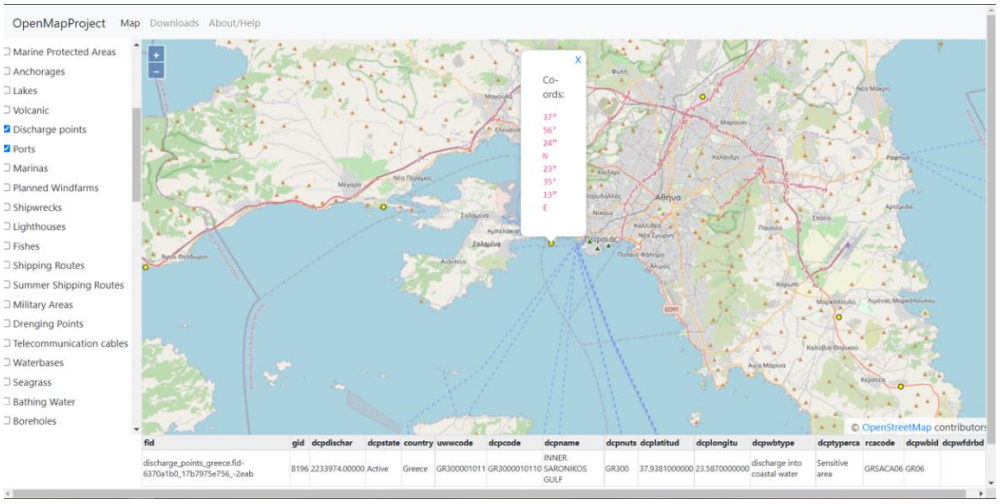
discharge_points (Κάποιες στήλες)

gid	dcpdischar	dcpstate	country (FK eez_greece territory1)	uwwcode	dcpname	geom
integer	numeric	Varchar	varchar	varchar	varchar	geometry

- *Λιμάνια.* Πηγή: [22]. Η πηγή που χρησιμοποιούμε είναι η πιο πλήρης πηγή για πληροφορίες λιμανιών από τις διαθέσιμες δωρεάν πηγές. Τα λιμάνια φαίνονται σαν γεωμετρία σημείου και περιέχουν πολλές πληροφορίες για το κάθε λιμάνι, συμπεριλαμβανομένων της χώρας και του ονόματος. Το αρχείο μας είναι shapefile και γίνεται απομόνωση των Ελληνικών λιμανιών με την πληροφορία του ονόματος χώρας.

portsgreece (Κάποιες στήλες)

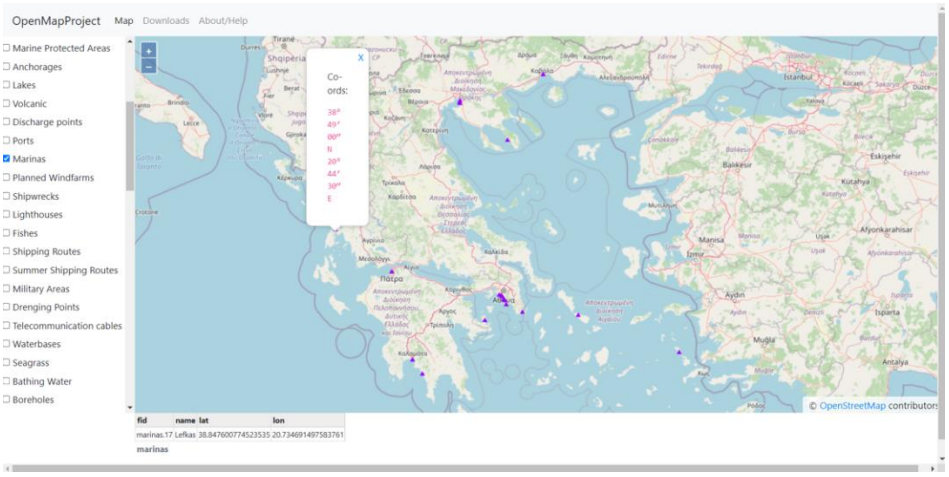
index_no	region_no	port_name	country	latitude	longitude	geom
double precision	double precision	Varchar	varchar	double precision	double precision	geometry



- *Μαρίνες*. Πηγή: [23]. Τα δεδομένα για τις Μαρίνες αποδείχτηκαν από τα πιο δύσκολα για εύρεση και χειρισμό. Τα δεδομένα αυτά ήταν δυσεύρετα δωρεάν και οι πηγές που τα παρείχαν -όπως και η Wikipedia- τα έδιναν σε μορφές αρχείων που δεν μπορούσαν να χρησιμοποιηθούν άμεσα, όπως στις προηγούμενες περιπτώσεις. Έτσι, χρειάστηκε να καταγραφούν οι Μαρίνες που αναγράφονται στην πηγή μας σε ένα αρχείο csv, μετά από αναζήτηση των συντεταγμένων. Για κάθε εγγραφή καταγράφονται το όνομα της μαρίνας, αλλά και τα γεωγραφικά πλάτη και μήκη. Τέλος, έγινε μετατροπή σε shapefile μέσω του QGIS, και μετέπειτα εισαγωγή στην βάση δεδομένων.

marinas

gid	name	lat	lon	Geom
integer	varchar	numeric	numeric	Geometry

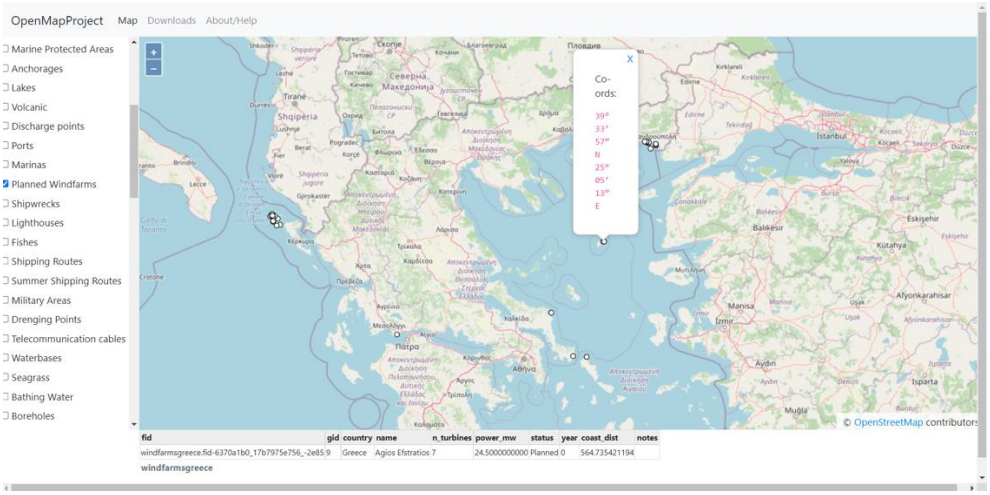


- *Ανεμογεννήτριες*. Πηγή: [21] Χρησιμοποιήσαμε ξανά την πηγή EMOD Human Activities, όπως και σε άλλες περιπτώσεις. Το σετ δεδομένων μας δείχνει τις

ανεμογεννήτριες που σχεδιάζονται να κατασκευαστούν τα επόμενα χρόνια σε παραθαλάσσια ή και θαλάσσια σημεία. Επιλογή δεδομένων μετά την εισαγωγή στην βάση, με χρήση SQL ερωτήματος ως προς το χαρακτηριστικό country.

windfarmsgreece

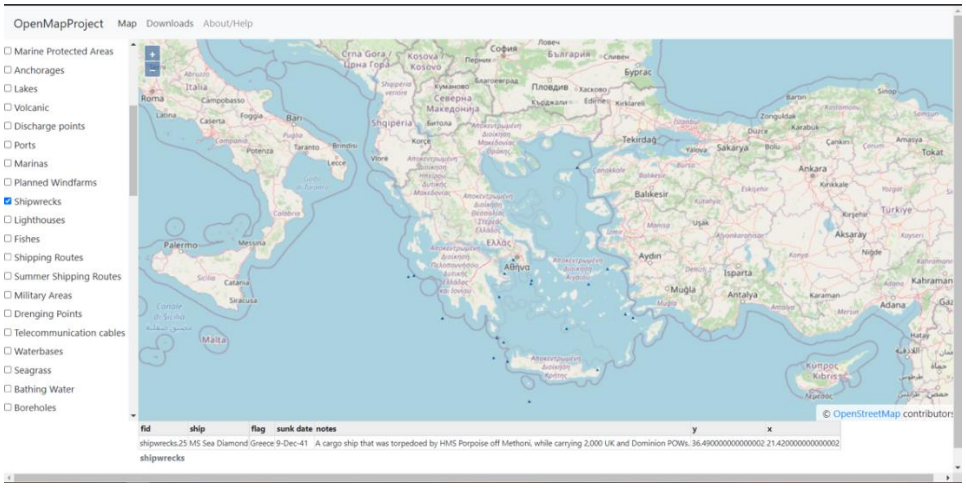
gid	country (FK eez_greece territory1)	name	n_turbines	power_mw	status	year	coast_dist	geom
integer	varchar	varchar	bigint	numeric	varchar	integer	numeric	geometry



- *Ναυάγια.* Πηγή: [24]. Η Wikipedia διαθέτει μία εμπεριστατωμένη λίστα ναυαγίων για όλες τις χώρες, μεταξύ των οποίων και η Ελλάδα. Ακολουθήσαμε ίδια τεχνική με τις Μαρίνες, καθώς τα δεδομένα δεν ήταν διαθέσιμα από άλλες πηγές και σε άλλες μορφές αρχείων.

shipwrecks

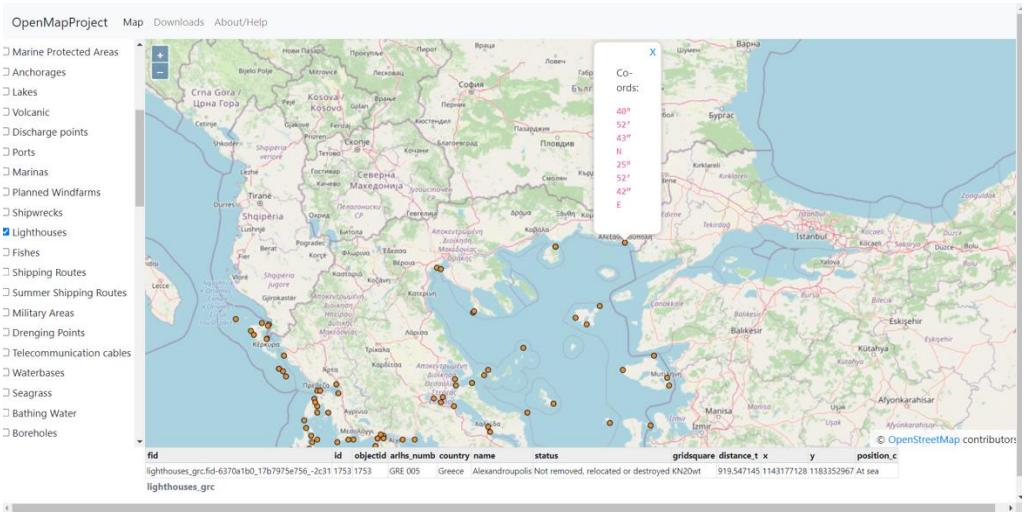
gid	ship	flag	sunk_date	notes	X	y	geom
integer	varchar	varchar	Varchar	varchar	numeric	numeric	geometry



- *Φάροι.* Πηγή: [25]. Οι φάροι αποτελούν πολύ σημαντικό κομμάτι της ναυτιλίας, οπότε τα συγκεκριμένα δεδομένα δεν ήταν δυνατόν να λείψουν από το συνολικό μας σετ δεδομένων. Στην πηγή δίνεται η δυνατότητα αναζήτησης των δεδομένων για συγκεκριμένες χώρες και περιοχές. Αφού αναζητήσαμε τα δεδομένα μας, στη συνέχεια τα περάσαμε σε έναν πίνακα Excel. Αποθηκεύσαμε το αρχείο μας ως comma separated file (csv), για να είναι εύκολα διαχειρίσιμο, και κάναμε μετατροπή σε shapefile, έτσι ώστε να το περάσουμε στην βάση δεδομένων.

lighthouses_grc (Κάποιες στήλες)

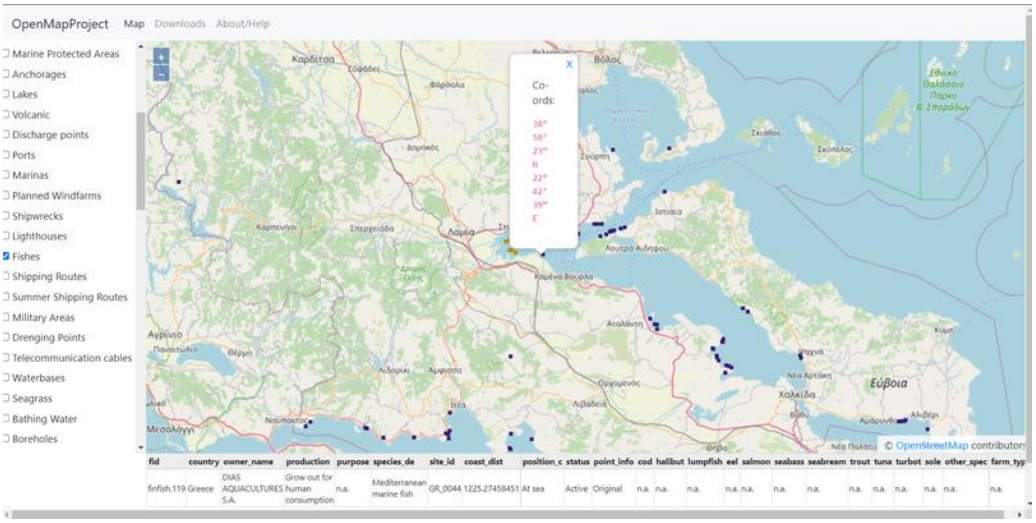
Id	country (FK eez_greece territory1)	arhls_numb	name	status	gridsquare	distance_t	position	geom
Integer	varchar	varchar	varchar	varchar	varchar	numeric	varchar	geometry



- *Ψάρια και οστρακοειδή.* Πηγή: [21] Τα δύο αυτά δεδομένα προέρχονται από την ίδια πηγή ως shapefile και υποδεικνύουν γεωγραφικά σημεία στα οποία υπάρχουν ιχθυοκαλλιέργειες συγκεκριμένων ειδών ψαριών και οστρακοειδών. Επιλογή, μέσω SQL ερωτήματος, για χώρα. Στον χάρτη φαίνονται ομαδοποιημένα, καθώς κατασκευάσαμε layer group, από τον GeoServer και περάσαμε τους δύο πίνακες της βάσης σε ένα στρώμα χάρτη.

Δυο αντίστοιχοι πίνακες, οι finfish, shellfish (Παραθέτουμε έναν πίνακα με κάποιες κοινές στήλες των δύο).

gid	country (FK eez_greece territory1)	owner_name	status	geom
integer	varchar	varchar	varchar	geometry



- *Ναυτιλιακές διαδρομές- 1.* Πηγή: [26]. Τα δεδομένα αυτά προέρχονται από συγκεκριμένα AIS δεδομένα πλοίων και στη συνέχεια προσομοιώσεις που έγιναν με μηχανική μάθηση. Δηλαδή κάποια από τα data του σετ δεν είναι πραγματικά, αλλά προβλέψεις διαδρομής. Αποφασίσαμε να χειριστούμε αυτά τα σημεία, καθώς θα μας έδινε μια ευκαιρία να ενώσουμε σημεία μιας διαδρομής, ως γραμμή, παρά την ανακρίβεια των δεδομένων. Τα δεδομένα λαμβάνονταν σε csv αρχείο ως σημεία και στη συνέχεια έγινε η μετατροπή σε shapefile, αφού πρώτα κάνουμε clip τα δεδομένα μας, για Ελλάδα. Τέλος, χρησιμοποιούμε την εξής εντολή:

SELECT prev_port, next_port, frequency,ST_MakeLine(geom ORDER BY trip_count ASC) as geom INTO shproutesgrc FROM shproutespoints GROUP BY prev_port,next_port, frequency,trip_count

Όπου παίρνει τον πίνακα με τα σημεία, και δημιουργεί από αυτά γραμμές ενώνοντας με αύξοντα τρόπο το trip_count που πρακτικά λειτουργεί σαν μετρητής του ταξιδιού.

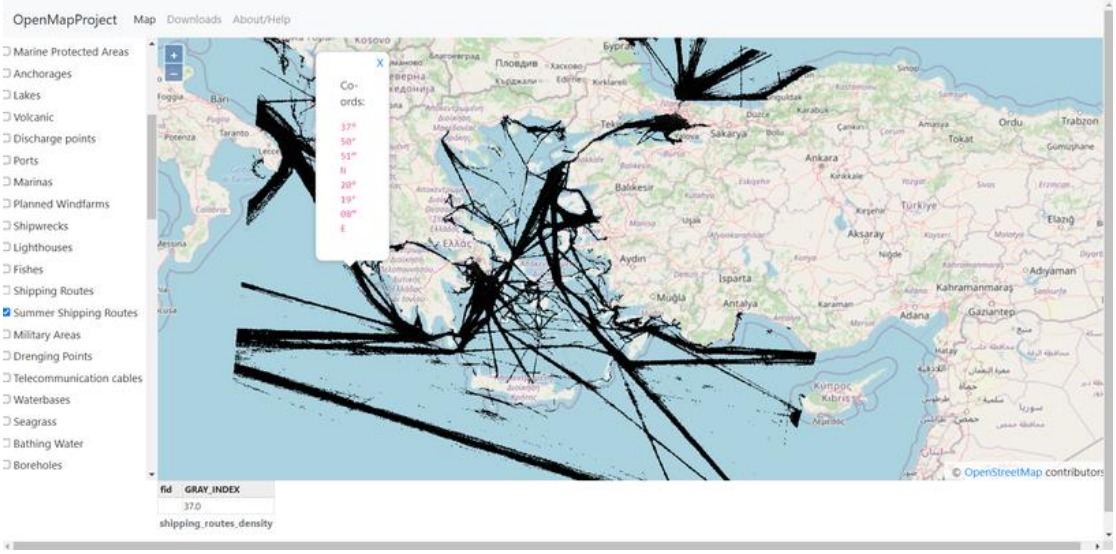
shproutes_grc

id	prev_port	next_port	frequency	trip_count	geom
integer	bigint	bigint	numeric	bigint	geometry

- *Ναυτιλιακές διαδρομές- 2.* Πηγή: [21]. Η συγκεκριμένη πηγή δίνει τις κύριες ναυτιλιακές διαδρομές της Μεσογείου ως προς την πυκνότητα των πλοίων σαν GeoTiff, raster εικόνες. Έτσι, λαμβάνουμε την raster εικόνα, κάνουμε clip μέσω του προγράμματος QGIS και στη συνέχεια το εισάγουμε στο σύστημά μας με την GDAL εντολή raster2pgsql. Η χρήση της εντολής ST_PixelAsPoint ή της ST_PixelAsLine, κρίνεται μη αποδοτική λόγω του μεγέθους και της λεπτομέρειας του raster αρχείου.

routes

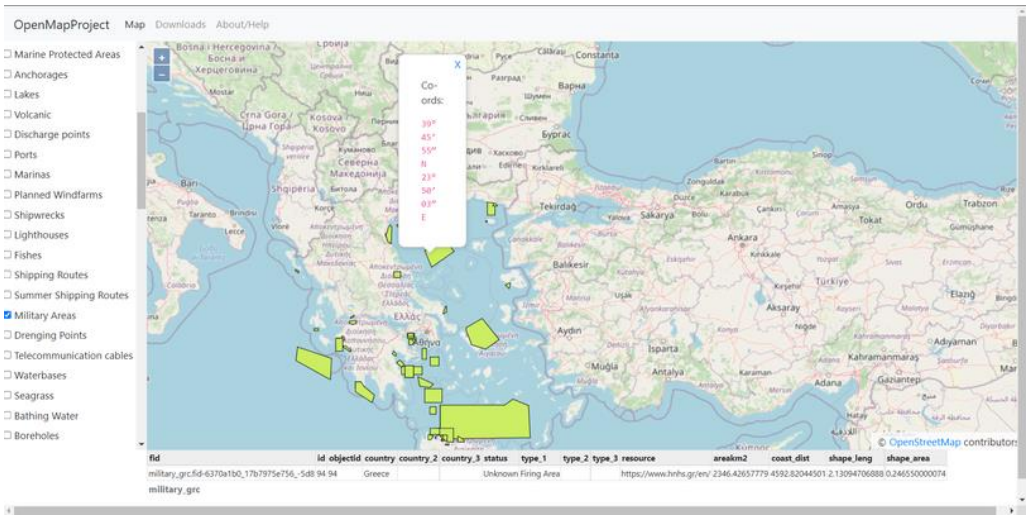
rid	rast
integer	raster



- *Περιοχές καταγεγραμμένων στρατιωτικών ασκήσεων.* Πηγή: [21]. Περιέχει σε shapefile τις ασκήσεις που έχουν γίνει σε διάφορα σημεία.

military_grc (Κάποιες στήλες)

id	country (FK eez_greece territory1)	status	type_1	resource	areakm2	geom
integer	Varchar	varchar	varchar	varchar	double precision	geometry



- *Σημεία απορρόφησης.* Πηγή: [21]. Σε shapefile -σαν σημεία εμφανίζονται- και η επιλογή έγινε με SQL ερώτημα, με βάση την χώρα.

drenching_points_grc

id	country (FK eez_greece territory1)	posinfo	seabasin	extraction	sea	geom
integer	varchar	varchar	varchar	varchar	varchar	geometry

- *Καλώδια τηλεπικοινωνιών.* Πηγή: [21] Σε αυτή την πηγή υπάρχουν δύο είδη δεδομένων, τα καλώδια, τα οποία εμφανίζονται στο shapefile ως πολυγραμμές και τα σημεία στα οποία ξεκινάει η γραμμή (landing stations), τα οποία είναι γεωμετρίας σημείου, σε διαφορετικό shapefile. Αφού κάνουμε εισαγωγή των shapefile στη βάση δεδομένων, κάνουμε για τα καλώδια και τα σημεία

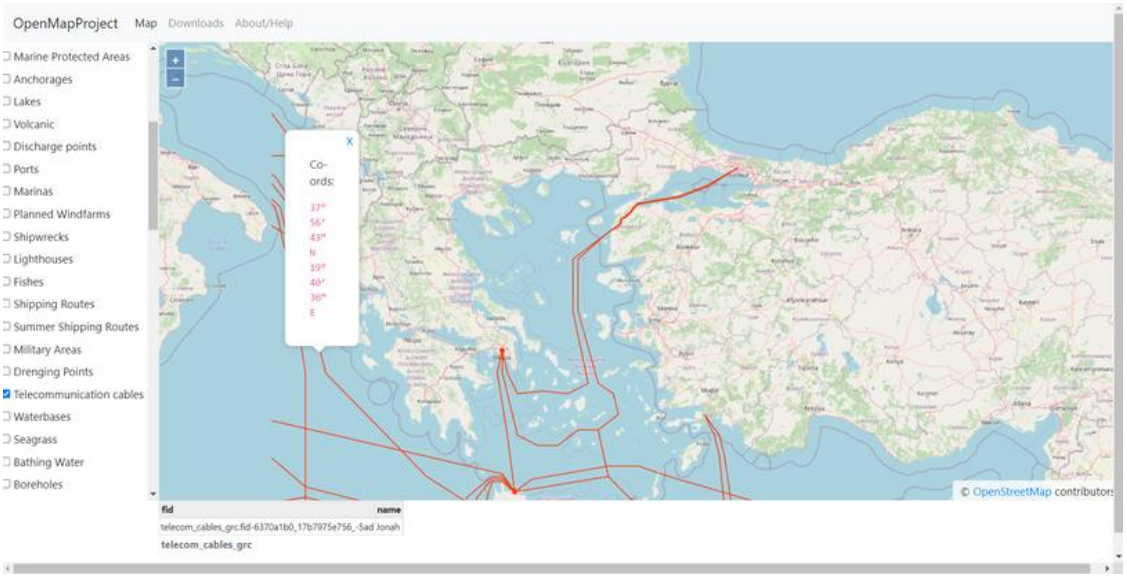
```
SELECT *
INTO cables_grc
FROM
((SELECT cables.name AS name, ST_Intersection(cables.geom, gr.geom) AS
geom
FROM cables cables, greece gr)
UNION
(SELECT name,geom
FROM stations
WHERE Country='Greece')) AS foo;
```

cables

gid	Name	owners	landing_po	length_km	source	geom
integer	Varchar	varchar	varchar	numeric	varchar	geometry

landing_stations_grc

Gid	Name	source	geom
Integer	Varchar	varchar	geometry

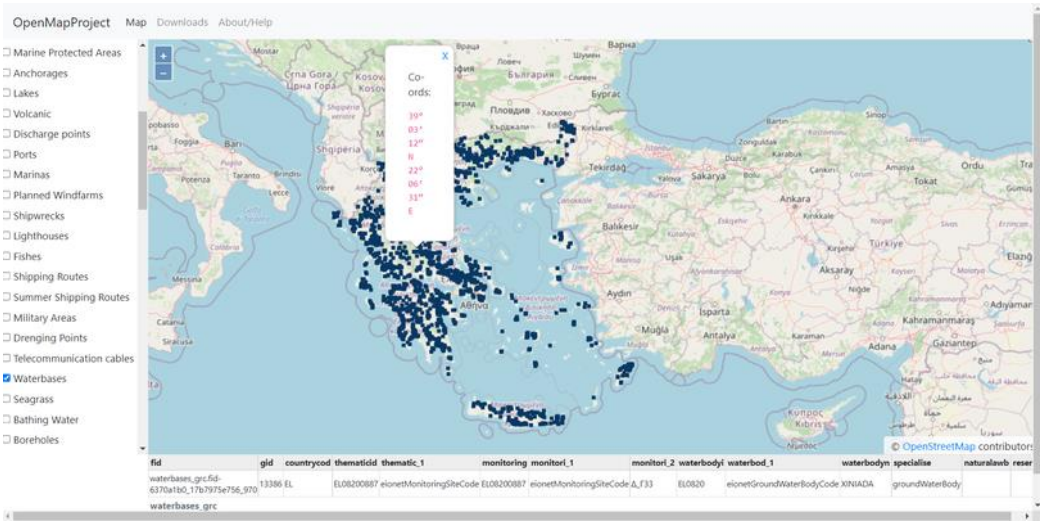


- *Βάσεις νερού.* Πηγή: [27]. Η πηγή αυτή μας δίνει δεδομένα από σταθμούς-βάσεις νερού, στις οποίες γίνεται μελέτη της ποιότητας των υδάτων. Τα δεδομένα παρέχονται σε csv αρχεία, τα οποία εμείς μετατρέψαμε σε ESRI shapefile με τον γνωστό τρόπο που αναφέρθηκε και παραπάνω. Στη συνέχεια,

διατηρήσαμε την πληροφορία για την Ελλάδα χρησιμοποιώντας το δεδομένο countrycod=’EL’, που παρέχεται στο σετ. Τα csv αρχεία είναι δύο, οπότε ο συνολικός πίνακας για τα waterbases προκύπτει από UNION δεδομένων.

waterbases_grc

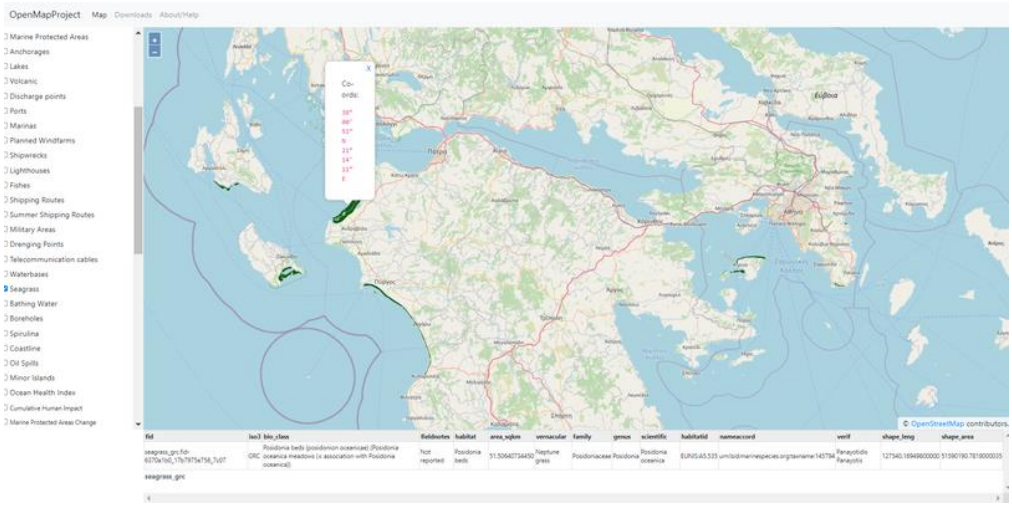
gid	country (FK eez_greece territory1)	Countrycod	monitoring	waterbodyn	geom
integer	varchar	Varchar	varchar	varchar	geometry



- *Θαλάσσια φενερόγαμα.* Πηγή: [28]. Η πηγή αυτή μας παρέχει την παγκόσμια κατανομή υδάτινου γρασιδιού. Τα δεδομένα παρέχονται σε shapefile και η επιλογή γίνεται με βάση το iso3 όνομα ‘GRC’. Τα δεδομένα εμφανίζονται στον χάρτη ως γεωμετρία πολυγώνου.

seagrass_grc

id	iso3 (FK eez_greece iso_ter1)	habitat	area_sqkm	shape_leng	shape_area	scientific	geom
integer	varchar	varchar	varchar	numeric	numeric	Varchar	geometry

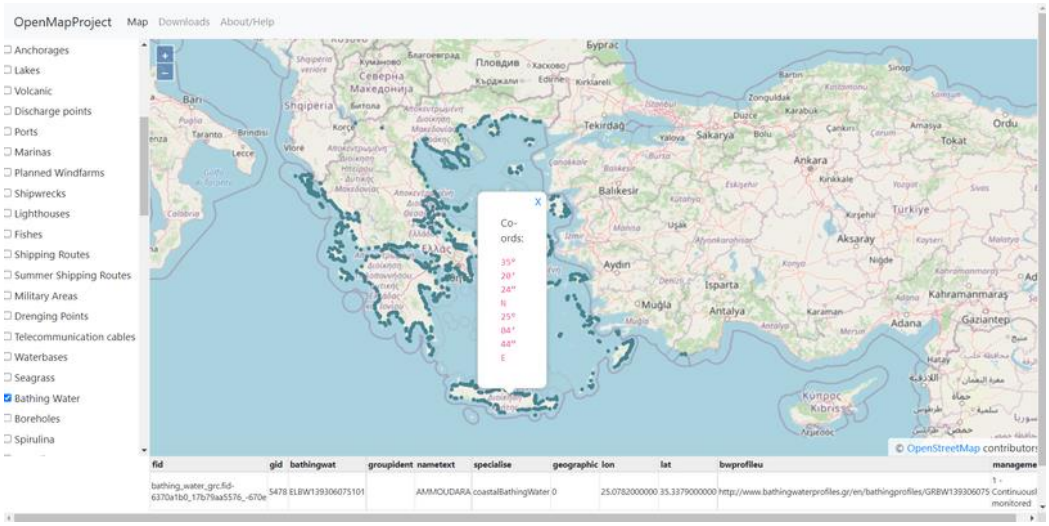


- *Παραλίες*. Πηγή: [21]. Το EMODHuman Activities μας παρέχει ένα σύνολο δεδομένων, το οποίο μας εμφανίζει σημεία, όπου το νερό είναι κατάλληλο για κολύμβηση. Τα δεδομένα παρέχονται σε ESRI shapefile και είναι πανευρωπαϊκά. Για την διατήρηση της Ελληνικής πληροφορίας, κάνουμε επιλογή των δεδομένων από τον κωδικό bathingwat.

  DELETE FROM bath_water WHERE bathingwat NOT LIKE ‘EL%’

bathing_water_grc

gid	bathingwat	nametext	specialize	bwprofileu	geom
integer	varchar	varchar	Varchar	varchar	geometry



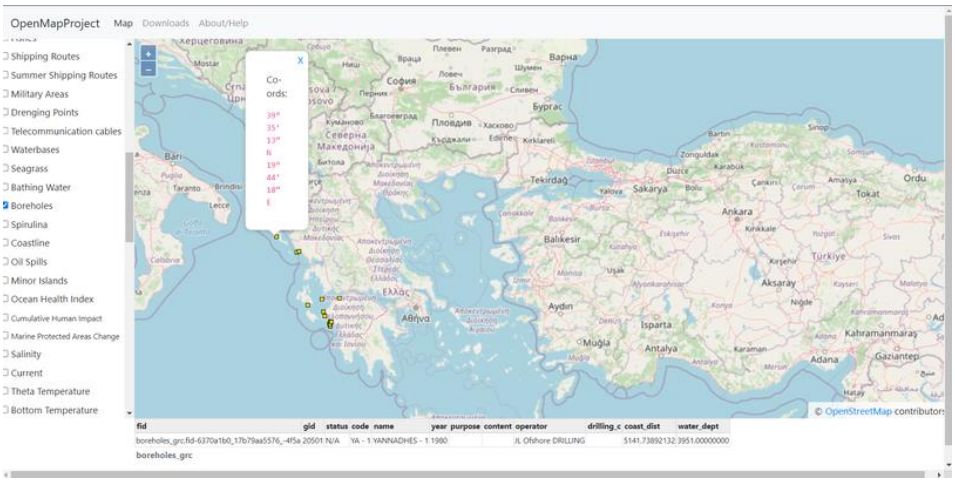
- *Γεωτρήσεις.* Πηγή: [21]. Σε αυτό το σύνολο δεδομένων μας δίνονται τα δεδομένα για την ύπαρξη γεωτρήσεων, οι οποίες εμφανίζονται ως γεωμετρία σημείου. Για να κρατήσουμε τα δεδομένα που μας ενδιαφέρουν, μετά την εισαγωγή του shapefile, κάνουμε:

  SELECT * INTO boreholes_grc FROM boreholes WHERE ((ST_X(geom)>=18.731 AND ST_X(geom)<=30.033) AND (ST_Y(geom)>=33.582) AND ST_Y(geom)<=41.9)

  Όπου κρατάμε τις γεωγραφικές συντεταγμένες X και Y από τη γεωμετρία σημείου και ελέγχουμε αν ανήκουν σε ένα bounding box που καλύπτει την Ελλάδα.

boreholes_grc

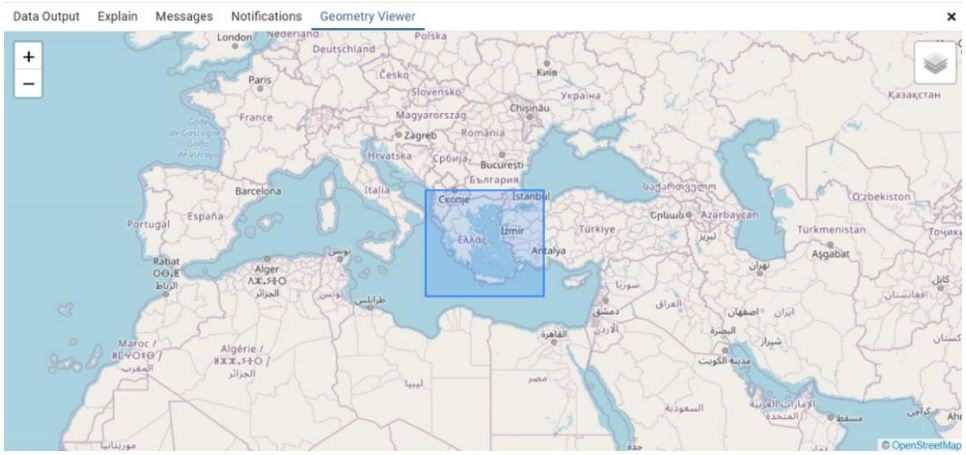
gid	name	code	Year	Operator	coast_dist	water_dept	geom
integer	varchar	varchar	Integer	Varchar	numeric	numeric	geometry



- *Περιοχές παραγωγής Σπιρουλίνας.* Πηγή: [21]. Εμφανίζει τις περιοχές, όπου παράγεται σπιρουλίνα. Τα δεδομένα παρέχονται σε shapefile με γεωμετρία σημείου.

spirulina_grc

Gid	id_site	owner_name	country (FK eez_greece territory1)	Lat	lon	pos_info	geom
Integer	varchar	varchar	varchar	Numeric	numeric	varchar	geometry



Εικόνα 9: Bounding Box με γεωμετρία πολυγώνου, που περιλαμβάνει την Ελλάδα. Το χρησιμοποιήσαμε για ST_Intersections και ST_Intersects.

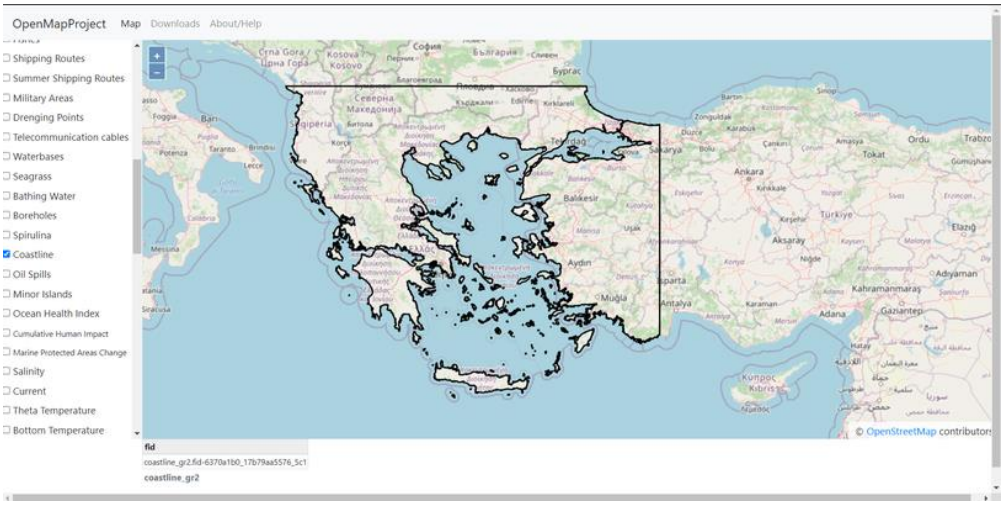
- *Ακτογραμμή.* Πηγή: [11]. Η πηγή αυτή μας παρέχει ως πολύγωνο την συνολική ακτογραμμή. Αφού κάνουμε εισαγωγή του shapefile, κάνουμε αποκοπή με bounding box, χρησιμοποιώντας:

  SELECT ST_Intersection(coastline.geom,gr.geom) AS geom INTO coastline_grc FROM coastline, greece gr;

  Για το bounding box βλ. [Εικόνα 9]. Έτσι, γίνεται τομή των δύο γεωμετριών πολυγώνου και κρατάμε την ακτογραμμή.

coastline_gr2

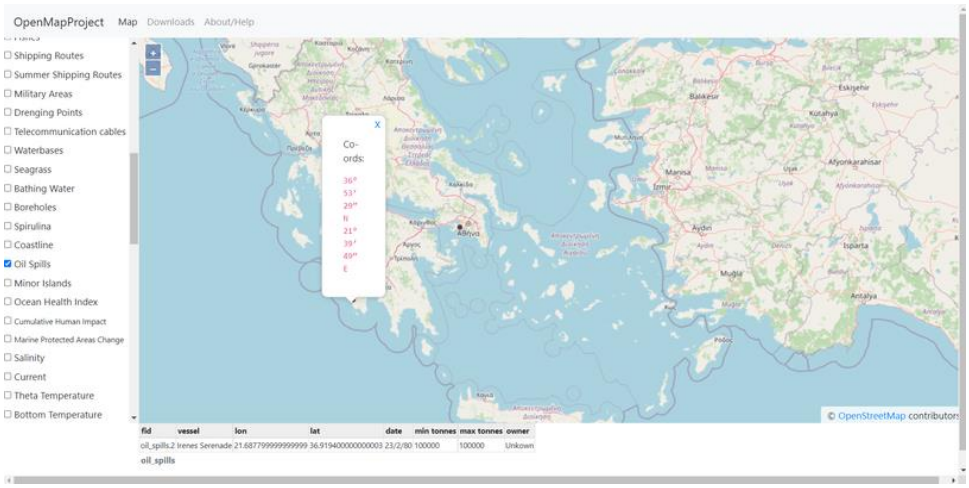
id	geom
integer	geometry



- *Πετρελαιοκηλίδες.* Πηγή [29]. Η πηγή μας παρέχει ιστορικά δεδομένα για πετρελαιοκηλίδες στον Ελλαδικό χώρο. Για να κρατήσουμε τα δεδομένα, δημιουργήσαμε csv αρχείο, το οποίο περιέχει τα γεωγραφικά δεδομένα και τα δεδομένα της κηλίδας. Στη συνέχεια, κάναμε μετατροπή δεδομένων σε shapefile και εισαγωγή στη βάση δεδομένων.

oil_spills

Gid	Vessel	date	lon	lat	min_tonnes	max_tonnes	owner	geom
Integer	varchar	varchar	numeric	numeric	bigint	bigint	varchar	geometry



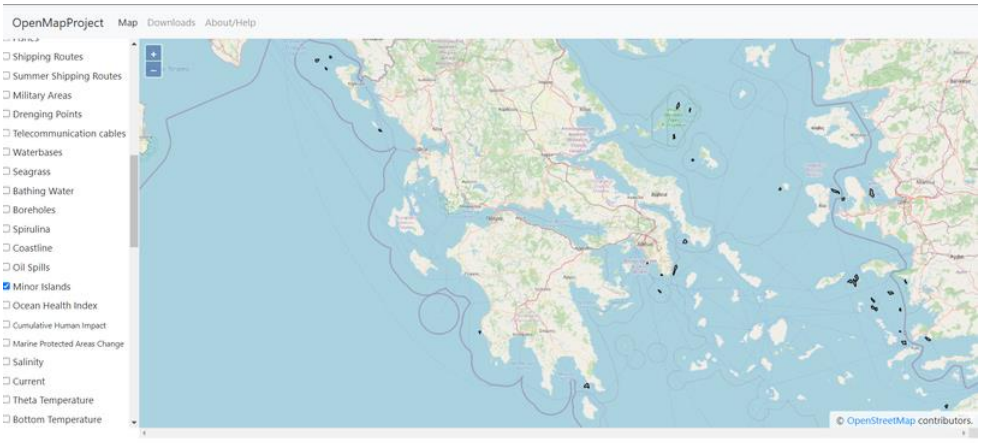
- *Μικρά νησιά, μικρότερα των 2 τετραγωνικών χιλιομέτρων.* Πηγή: [11]. Τα δεδομένα μας παρουσιάζουν όλα τα μικρά κατοικήσιμα νησιά, τα οποία είναι μικρότερα των δύο τετραγωνικών χιλιομέτρων. Τα δεδομένα παρέχονται σε shapefile και μας δίνουν μόνο γεωγραφικές συντεταγμένες. Έτσι, για την απομόνωση των δεδομένων ακολουθούμε την τακτική του:

  DELETE FROM minor_islands
  WHERE
  ((ST_X(ST_Centroid(geom))<=18.731
  AND ST_X(ST_Centroid(geom)) >=30.033)
  AND ((ST_Y((ST_Centroid(geom))<=33.582)
  AND ST_Y((ST_Centroid(geom))>=41.9))

  Εδώ, κρατάμε το κεντροειδές σημείο της κάθε γεωμετρίας πολυγώνου και ελέγχουμε αν αυτό δεν ανήκει στο bounding box σύμφωνα με τις συντεταγμένες.

minor_islands_greece

gid	featurecla	minzoom	scalerank	geom
integer	varchar	double precision	bigint	geometry



- *Πειρατικές ενέργειες και απειλές προς πλοία.* Πηγή: [30]. Στο συγκεκριμένο σύνολο δεδομένων παρουσιάζονται οι πειρατικές ενέργειες και οι απειλητικές ενέργειες στα πλοία. Τα δεδομένα εμφανίζονται ως σημεία και δίνονται σε shapefile. Χρησιμοποιούμε την ίδια λογική με τις γεωτρήσεις.

Piracy_greece

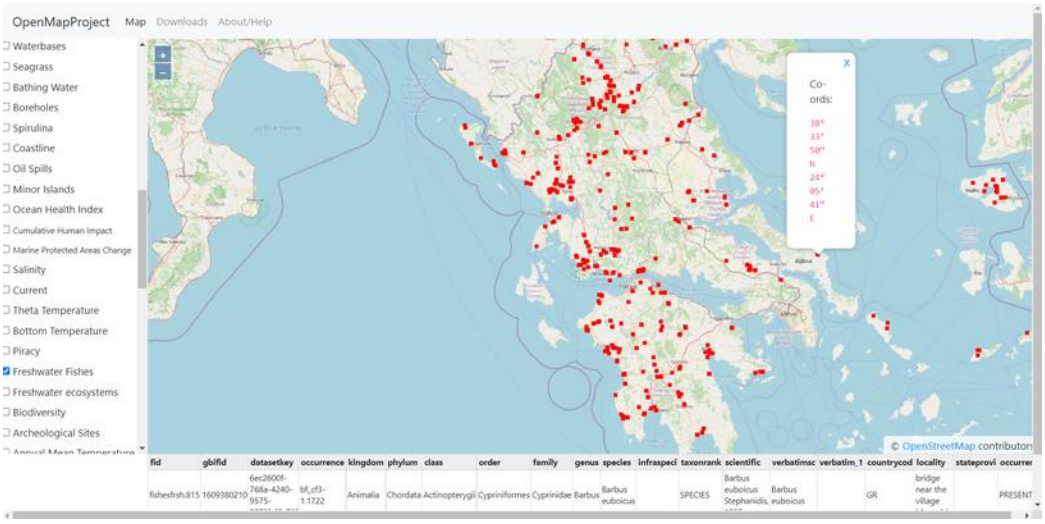
gid	reference	dataeofocc	subreg	hostility_	victim_d	descriptio	geom
integer	varchar	Date	varchar	varchar	varchar	varchar	geometry

- *Είδη ψαριών φρέσκου νερού και βιοποικιλότητα.* Πηγή: [31]. Από την πηγή μας καταγράψαμε δύο σύνολα δεδομένων, για την ποικιλότητα ειδών ψαριών και για την γενική βιοποικιλότητα διάφορων ζωικών πληθυσμών. Τα δεδομένα μας ήταν σε csv αρχεία, τα οποία τα λάβαμε, τα επεξεργαστήκαμε - κρατώντας τα στοιχεία με countrycod=’GR’- και τα μετατρέψαμε σε shapefiles, τα οποία μετατράπηκαν σε πίνακες της βάσης δεδομένων.

fishesfrsh

gid	gbifid	datasetkey	occurrence	kingdom	phylum	class	geom
integer	bigint	varchar	varchar	varchar	varchar	varchar	geometry

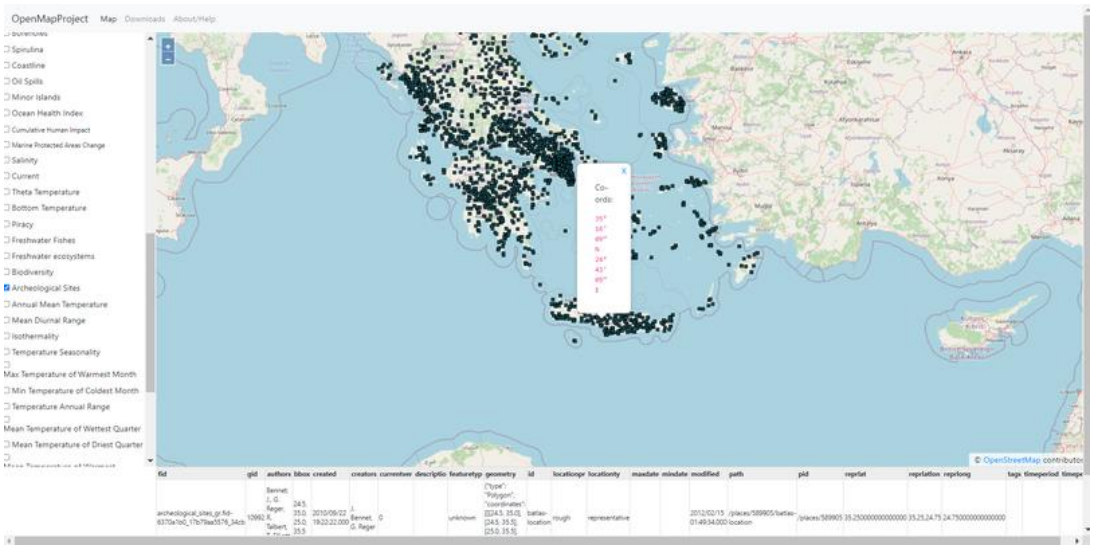
Ενώ ο πίνακας freshwater_ecosystems αποτελείται μόνο από στήλη geom (geometry).



- *Αρχαιολογικοί χώροι-χώροι με γεωγραφικά μνημεία.* Πηγή: [32]. Λαμβάνουμε τα δεδομένα σε μορφή csv αρχείου. Το επεξεργαζόμαστε, το μετατρέπουμε και το περνάμε ως shapefile στην βάση δεδομένων.

archeological_sites_gr

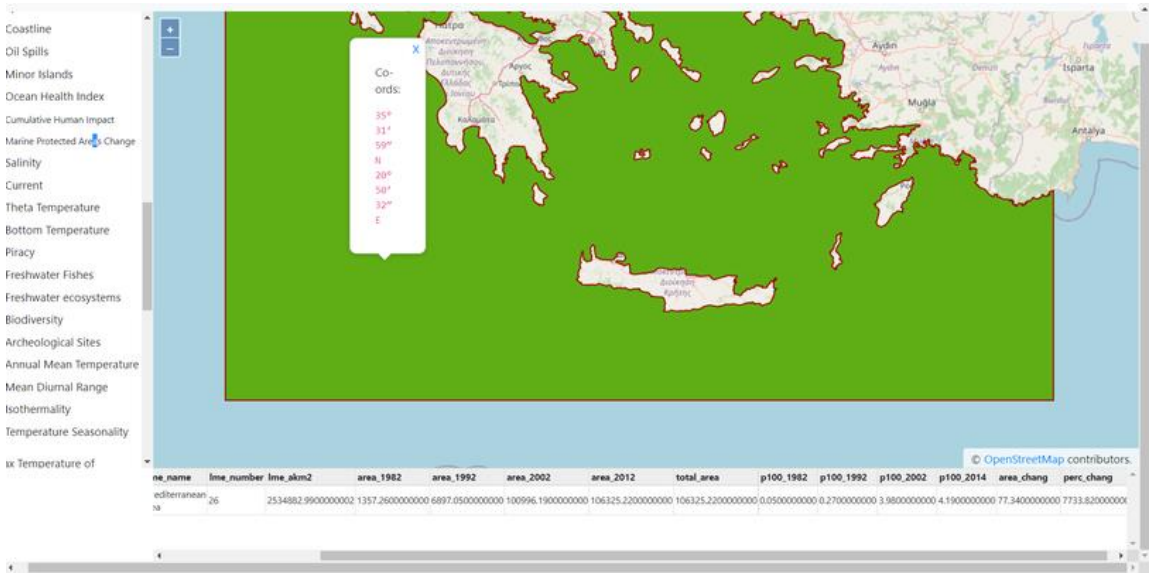
gid	Authors	bbox	created	creators	descriptio	geom
integer	Varchar	varchar	varchar	varchar	varchar	geometry



- *Περιβαλλοντικές θαλάσσιες πληροφορίες.* Πηγή: [33]. Από τη συγκεκριμένη πηγή λάβαμε πληροφορίες, οι οποίες έχουν να κάνουν με την «υγεία» των υδάτων, αλλά και την περιβαλλοντική κατάσταση της θάλασσας. Πιο συγκεκριμένα, έχουμε το Ocean Health Index, το οποίο μας δίνει διάφορες μετρικές για την υγεία του νερού και βαθμολογεί την κατάσταση από το 1-100 [34]. Επιπλέον, έχουμε το Marine Protected Areas Change, το οποίο μας δείχνει με μετρική το μέγεθος της περιοχής ανά τα χρόνια, αλλά και τα ποσοστά μεταβολής του μεγέθους της προστατευόμενης περιοχής. Τέλος, έχουμε το δεδομένο της αθροιστικής ανθρώπινης επίπτωσης στους ωκεανούς, με διάφορες μετρικές για ανθρώπινες δραστηριότητες, όπως ρύπανση και αλιεία. Τα δεδομένα αυτά έρχονται σαν ESRI shapefiles, τα οποία σηματοδοτούν ενιαία πολύγωνα για μεγάλες θαλάσσιες μάζες, πχ τη Μεσόγειο.

mpa_change_gr, ocean_health_index_gr, cumulative_human_impact_gr

gid	lme_name	Άλλα χαρακτηριστικά ανάλογα τον πίνακα	geom
integer	Varchar		geometry



- *Βιοκλιματικά δεδομένα.* Πηγή: [35]. Η πηγή World Clim μας παρέχει GeoTiff raster δεδομένα υψηλής ποιότητας για βιοκλιματικές μετρικές. Τα δεδομένα αυτά χρειάστηκε να τα προεπεξεργαστούμε με την χρήση του QGIS, έτσι ώστε να απομονώσουμε τα δεδομένα της χώρας μας. Στην συνέχεια, αφού κάναμε το clipping, μέσω των εργαλείων του QGIS, εισαγάγαμε τα δεδομένα μας στο raster σχήμα που είχαμε ήδη κατασκευάσει στη βάση δεδομένων μας, με την εντολή της βιβλιοθήκης GDAL, raster2pgsql, όπως παρακάτω.

  raster2pgsql -s 4326 -c -I raster_name.tif -F raster.raster_name | psql -d openMap

  όπου το -s: δείχνει το SRID της προβολής των γεωγραφικών δεδομένων στο raster αρχείο,

  το -c: δημιουργεί νέο πίνακα στον οποίο θα βάλει το raster

  το -I: δημιουργεί ένα index για τα γεωγραφικά δεδομένα μας

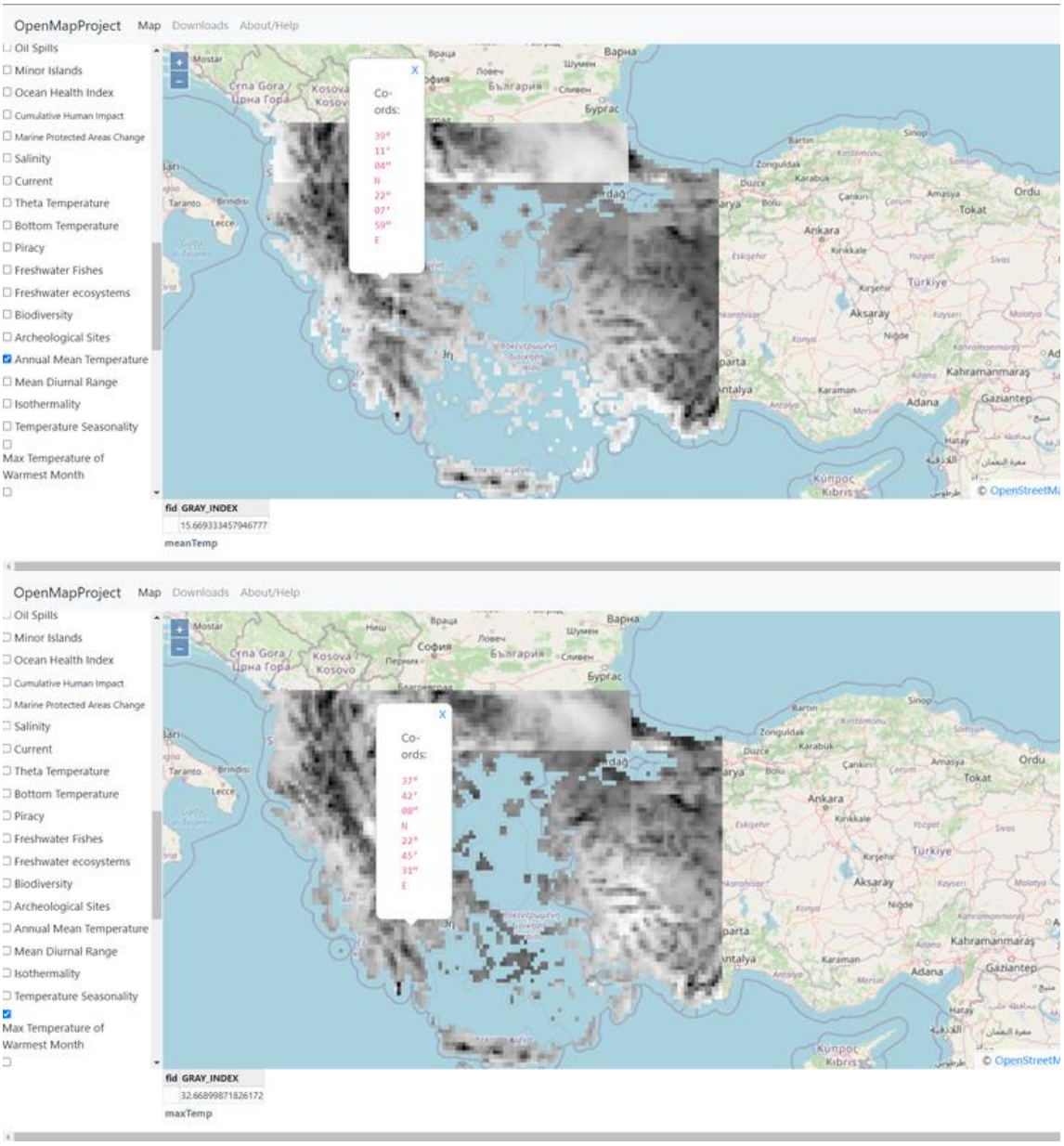
  το -F: βάζει στο πίνακα στήλη με το όνομα του αρχείου.

  Βάζουμε τα δεδομένα του tiff αρχείου που θέλουμε να εισάγουμε και το πού θα εισαχθεί στη βάση με συγκεκριμένο όνομα (raster.raster_name). Τέλος, κάνουμε pipe τα δεδομένα στη βάση μας.

  Τα δεδομένα της πηγής μας, περιλαμβάνουν 19 βιοκλιματικά στοιχεία, τα οποία είναι ισοθερμικότητα, εποχικότητα θερμοκρασίας, μέγιστη θερμοκρασία, ελάχιστη θερμοκρασία, ετήσια διαφορά θερμοκρασίας, μέση θερμοκρασία υγρότερου τριμήνου, μέση θερμοκρασία στεγνότερου τριμήνου, μέση θερμοκρασία ψυχρότερου τριμήνου, μέση θερμοκρασία θερμότερου τριμήνου, ετήσια κατακρήμνιση, κατακρήμνιση των αντίστοιχων μηνών (θερμότερου, ψυχρότερου, στεγνότερου, υγρότερου) και των αντίστοιχων τριμήνων (Παραδείγματα παρακάτω: Μέση θερμοκρασία και μέγιστη θερμοκρασία).

Για όλους τους πίνακες που δημιουργούνται σε αυτό το σύνολο δεδομένων (αναφέρονται παραπάνω), έχουμε:

Rid	geom
Integer	geometry



- *Θαλάσσια περιβαλλοντικά δεδομένα.* Πηγή: [36]. Τα δεδομένα μας αφορούν συγκεκριμένα περιβαλλοντικά δεδομένα σχετικά με τη θάλασσα. Τα δεδομένα αυτά έρχονται σε μορφή netCDF αρχείου και είναι πολυδιάστατα. Περιέχουν πολλές «μπάντες» δεδομένων -η μία πάνω στην άλλη- και έτσι περιλαμβάνουν όλη την πληροφορία. Για να μπορέσουμε να εισάγουμε την πληροφορία στη βάση δεδομένων μας, θα χρησιμοποιήσουμε την εντολή της GDAL βιβλιοθήκης gdal_translate [37]:

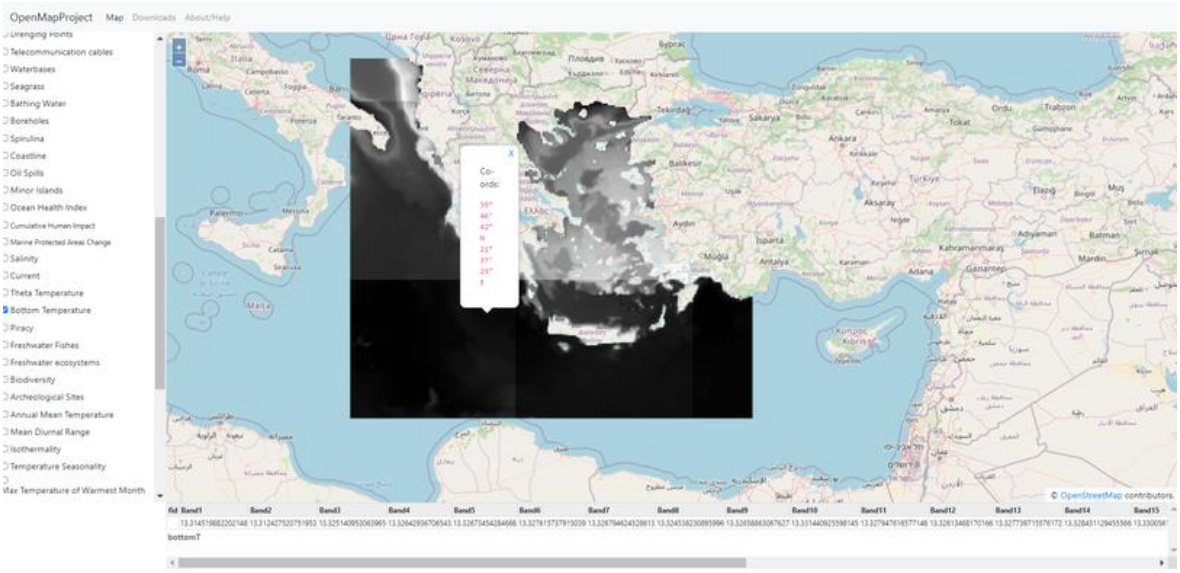
  gdal_translate NETCDF:"name_of_file.nc":band name.tiff ,

  και μετά κάνουμε raster2pgsql, όπως παραπάνω.

Στο σύνολο εμπεριέχονται δεδομένα για την ελάχιστη θερμοκρασία της θάλασσας, τη μέση θαλάσσια θερμοκρασία, την «αλμυρότητα» και το ρεύμα (Παράδειγμα παρακάτω: Ελάχιστη θερμοκρασία).

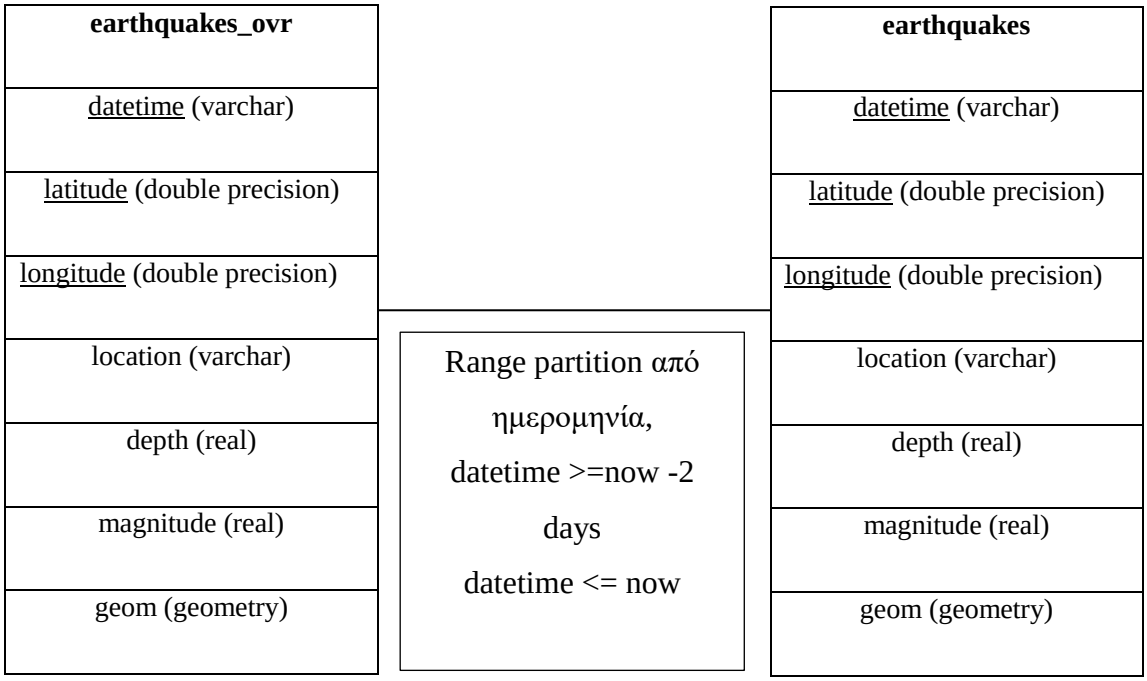
Αντίστοιχα, όπως παραπάνω:

Rid	geom
Integer	geometry



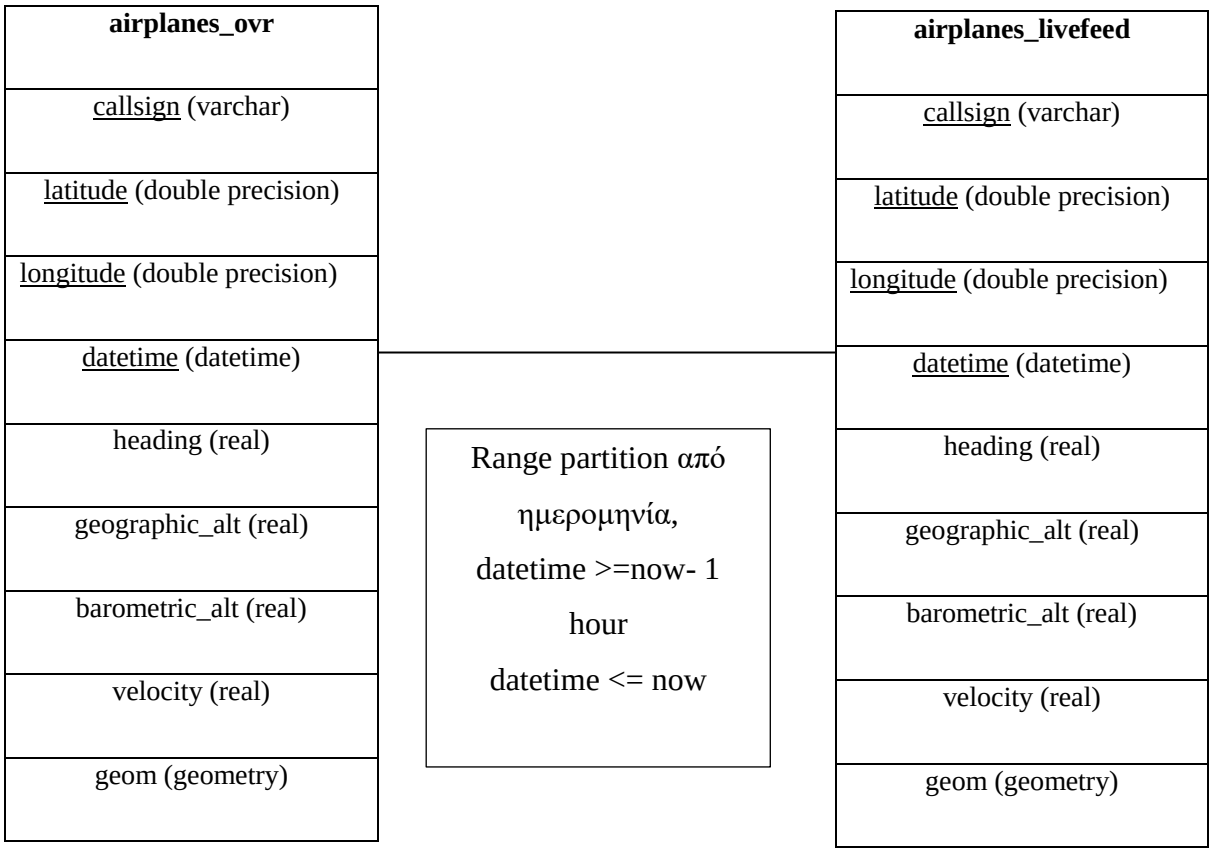
Όλα τα προηγούμενα δεδομένα αποτελούσαν είτε ιστορικά δεδομένα είτε στατικά δεδομένα τα οποία λαμβάναμε σε μορφή αρχείων shapefile, csv, geoTiff και netCDF. Παρόλα αυτά, δεν είναι όλα τα δεδομένα μας στατικά. Πολλά από αυτά, όπως οι σεισμοί, είναι δυναμικά δεδομένα που συμβαίνουν συνεχώς. Έτσι, πολλές υπηρεσίες, που παρέχουν στο χρήστη τα δεδομένα τους, στέλνουν συνεχή ροή δεδομένων σε διαφορετικές μορφές, όπως JSON αρχεία, XML αρχεία, είτε αρχεία σε πίνακες HTML που ανανεώνονται. Έτσι, χρειάστηκε να επινοήσουμε «έξυπνες» λύσεις για το πρόβλημα και να διαχειριστούμε τα δεδομένα αυτά με κάποιο τρόπο, ώστε η ζωντανή ροή πληροφορίας να εισέρχεται απευθείας στο σύστημά μας μέσω κώδικα Python. Παρακάτω θα δούμε τις τρεις περιπτώσεις που μελετήσαμε. Τα δεδομένα σεισμών, τα δορυφορικά δεδομένα θέσης και τα καιρικά δεδομένα.

- *Ζωντανά δεδομένα σεισμών.* Πηγή: [38]. Οι σεισμοί αποτελούν φυσικό φαινόμενο, το οποίο είναι απρόβλεπτο και άμεσο. Η Ελλάδα, αποτελεί μία εκ των πιο σεισμογενών περιοχών στον κόσμο, καθότι βρίσκεται ανάμεσα σε δύο ρήγματα των τεκτονικών πλακών, στο ελληνικό τόξο και στο ρήγμα της Βόρειας Ανατολίας. Αυτή η ιδιαίτερη σεισμικότητα μπορεί να αποτελέσει πρόβλημα σε στεριά και θάλασσα και έτσι η καταγραφή και μελέτη των σεισμών αποτέλεσε σημαντικό κομμάτι του συνόλου δεδομένων μας. Γι' αυτό τον λόγο, κατασκευάσαμε έναν κώδικα python [βλ. Παράρτημα κωδίκων-Κώδικας Σεισμών], ο οποίος διαβάζει την ζωντανή ροή δεδομένων από το πίνακα HTML του γεωδυναμικού ινστιτούτου. Για να το κάνουμε αυτό, κατασκευάζουμε δύο πίνακες, στους οποίους θα αποθηκεύουμε τα δεδομένα. Ο ένας θα αποθηκεύει τα δεδομένα των τελευταίων δύο ημερών και ο άλλος θα αποθηκεύει το σύνολο των δεδομένων. Οι πίνακες θα έχουν ως κυρίως κλειδιά το γεωγραφικό πλάτος, το γεωγραφικό μήκος και τη χρονοσφραγίδα του σεισμού. Αρχικά, το πρόγραμμα ελέγχει την ύπαρξη των πινάκων και, αν δεν υπάρχουν, κατασκευάζει τους πίνακες και δημιουργεί στον πίνακα με τα συνολικά δεδομένα ένα trigger function, το οποίο θα προσθέτει τα δεδομένα ταυτόχρονα στον άλλο πίνακα και θα διαγράφει τα δεδομένα που είναι μεγαλύτερα των δυο ημερών. Όλες αυτές οι διαδικασίες πάνω στη βάση δεδομένων γίνονται χρησιμοποιώντας την βιβλιοθήκη psycopg2. Για να λάβει τα δεδομένα, το πρόγραμμα χρησιμοποιεί την βιβλιοθήκη requests, για να συνδεθεί στην ιστοσελίδα και να διαβάσει τον HTML πίνακα, ενώ χρησιμοποιεί την pandas (https://pandas.pydata.org/), για να διαχειριστεί τα δεδομένα του πίνακα και να τα επεξεργαστεί. Στην συνέχεια, αφού λάβει τα δεδομένα, τα περνάει σε ένα λεξικό και μετέπειτα βάζει τα δεδομένα στους πίνακες με την συνάρτηση insertAndUpdate, προσθέτοντας τις γραμμές -μέσω cursor- για κάθε εγγραφή του λεξικού και κάνοντας update τη στήλη της γεωμετρίας, για να δημιουργήσει σημείο με τις συντεταγμένες μας μέσω του ST_GeomFromText. Έτσι κατασκευάζουμε τους δυναμικούς πίνακες μας, τους οποίους έχουμε ρυθμίσει (μέσω της βιβλιοθήκης schedule, https://schedule.readthedocs.io/en/stable/) να λαμβάνουν δεδομένα δύο φορές την ημέρα, με την επιλογή να ληφθούν δεδομένα για περισσότερες ώρες, σε περίπτωση που κάποιος επιθυμεί να αξιοποιήσει τον κώδικα για προσωπική χρήση.

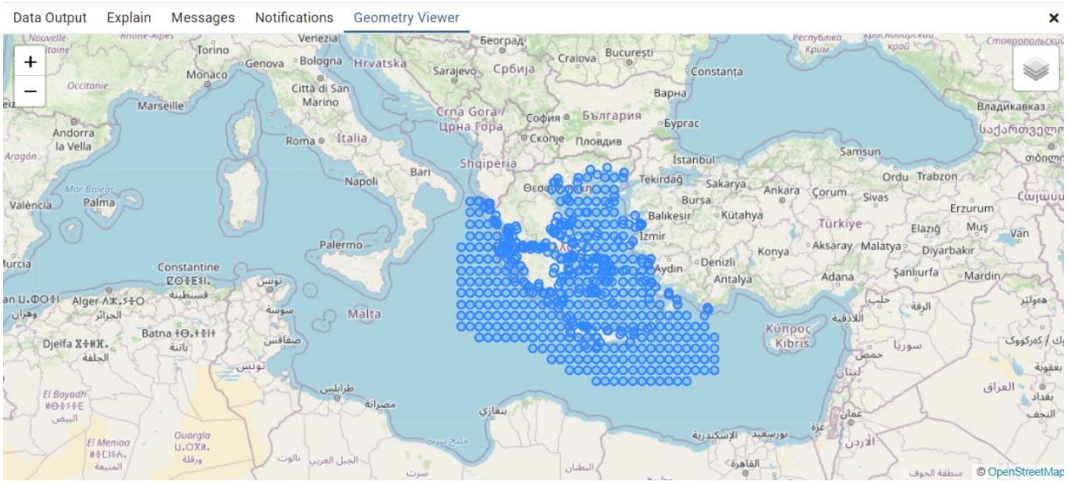


- *Δεδομένα θέσης από δορυφορικές συντεταγμένες.* Πηγή: [39]. Αρχικός σκοπός μας ήταν η χρήση δεδομένων AIS από τα πλοία, καθότι τα δεδομένα αυτά χρησιμοποιούνται κατά κόρον στην ναυτιλία και έτσι θα ήταν άκρως χρήσιμη για την εφαρμογή μας. Το AIS (ή αλλιώς Automatic Identification System) αποτελεί σύστημα, με το οποίο ένας πομπός σημάτων επικοινωνεί με δορυφορικό ή επίγειο δέκτη και αποστέλλει συνεχώς πληροφορίες για το πλοίο και τη θέση του. Παρ' όλα αυτά, τα AIS δεδομένα δεν παρέχονται δωρεάν και δεν είναι ανοιχτά για χρήση και αναδιανομή, γι' αυτό το λόγο χρησιμοποιήσαμε δεδομένα θέσης αεροπλάνων (που μας ενδιαφέρουν εφόσον περιγράφουν κίνηση η οποία πραγματοποιείται πάνω από το θαλάσσιο χώρο που αποτελεί το αντικείμενο της πτυχιακής), τα οποία λειτουργούν με αντίστοιχο τρόπο και έχουν παρόμοιο τρόπο διαχείρισης, ενώ ταυτόχρονα ο όγκος τους είναι αντίστοιχα μεγάλος. Για να λύσουμε και εδώ το πρόβλημα, χρησιμοποιούμε ένα python script, το οποίο διατίθεται στο παράρτημα στο σημείο Κώδικας για δεδομένα θέσης. Ο κώδικας αυτός λειτουργεί με παρόμοιο τρόπο με αυτόν των σεισμικών δεδομένων, με κάποιες βέβαια λειτουργικές διαφορές. Αρχικά, τα δεδομένα μας έρχονται σε μορφή JSON αρχείων μέσω του OpenSky API. Στον κώδικα αυτό λαμβάνουμε τα δεδομένα κάθε λεπτό και διαβάζουμε τα JSON αρχεία σαν αρχεία που μοιάζουν με λεξικό, περνώντας τα παράλληλα σε ένα δικό μας λεξικό για καλύτερη

διαχείριση και αποσφαλμάτωση. Εν συνεχεία, κάνουμε insertAndUpdate, όπως ακριβώς πάνω. Η μόνη επιπλέον διαφορά είναι ότι δεν κρατάμε συνολικό πίνακα για τις εγγραφές, αλλά κρατάμε έναν πίνακα της ζωντανής ροής και έναν πίνακα της τελευταίας ώρας χρησιμοποιώντας το: DELETE FROM ONLY airplanes_ovr WHERE datetime < (now() –'1 hour'::interval);. Οι πίνακες αυτοί χρησιμοποιούν ως πρωτεύον κλειδί τον συνδυασμό των γεωμετρικών συντεταγμένων, την ώρα καταγραφής του σήματος, αλλά και το ιδιαίτερο σήμα-όνομα του αεροσκάφους. Τέλος, κατά την κλήση της συνάρτησης getAPI, κάνουμε διαχείριση timeout exception, καθώς ο εξυπηρετητής της υπηρεσίας εμφανίζει συχνά λάθη στην αποστολή δεδομένων λόγω φόρτου.



- *Δεδομένα καιρού.* Πηγή: [40]. Τα δεδομένα καιρού αποτελούν κάποια εκ των πιο σημαντικών δεδομένων στον ειδικότερο χώρο της εφαρμογής της Γεωπληροφορικής στο θαλάσσιο περιβάλλον. Τα δεδομένα αυτά είναι μεγάλου όγκου και αλλάζουν πολλές φορές, ακόμα και στη διάρκεια μίας μέρας. Αντίστοιχα με τις προηγούμενες περιπτώσεις δυναμικών δεδομένων, για να κάνουμε την διαχείριση, αναπτύξαμε κώδικα γραμμένο σε python (Παράρτημα- Κώδικας καιρού). Τα δεδομένα μας τα λαμβάνουμε από την πηγή, η οποία παρέχει τα δεδομένα μέσω API κλήσεων, σε μορφή JSON αρχείων, με την δωρεάν έκδοση της υπηρεσίας να παρέχει τα δεδομένα για 60 κλήσεις ανά λεπτό. Για να καταπολεμήσουμε αυτό το πρόβλημα, χρησιμοποιούμε ένα flag στη συνάρτηση current που κάνει τις κλήσεις API, το οποίο αυξάνεται κατά μία μονάδα σε κάθε κλήση, και χρονοπρογραμματίζουμε τις κλήσεις αφαιρώντας από τον χρόνο 60 δευτερολέπτων τα δευτερόλεπτα υλοποίησης των 60 κλήσεων. Επειδή οι κλήσεις API μέσω του openWeatherMap απαιτούν συγκεκριμένα σημεία, υλοποιήσαμε ένα έξυπνο σύστημα, στο οποίο χρησιμοποιούμε ένα κατασκευασμένο από εμάς πίνακα της βάσης δεδομένων, ο οποίος περιέχει 500 σημεία ανά την Ελλάδα, συμπεριλαμβανομένων λιμανιών, αεροδρομίων, σημείων φάρων, αλλά και σημείων μέσα στην ανεξάρτητη οικονομική ζώνη, έτσι ώστε να λάβουμε διάσπαρτα δεδομένα για όλη την επικράτεια (βλ. εικόνα 10). Η λογική παραγωγής πινάκων βάσης δεδομένων μέσω κώδικα παραμένει αρκετά παρόμοια με τις άλλες πληροφορίες που λαμβάνουμε δυναμικά, παρ' όλα αυτά, στη συγκεκριμένη περίπτωση χρησιμοποιούμε trigger συνάρτηση στα συνολικά δεδομένα καιρού, η οποία κατασκευάζει διασπασμένους πίνακες δεδομένων, για κάθε μήνα του χρόνου και κάθε χρονιάς μέσω της συνάρτησης triggerFun().



Εικόνα 10: Σημεία χρήσης, για κλήση API.

Δηλαδή, κάθε πρώτη του μήνα κατασκευάζει νέο πίνακα-παιδί του συνολικού, μέσω του tableCreation(). Τέλος, ο κώδικας, ο οποίος αναδιανέμεται στον χρήστη, του δίνει την δυνατότητα να κρατήσει πληροφορία για έναν συγκεκριμένο χώρο της Ελλάδας μέσω περιβάλλοντος κυτίου χρησιμοποιώντας την συνάρτηση dbBringBBOX(). Τα δεδομένα είναι προγραμματισμένα, μέσω της schedule, να έρχονται κάθε μισή ώρα, ενώ γίνεται έλεγχος για διάσπαση πινάκων κάθε μέρα στις 00:00, για λόγους πληρότητας κώδικα. Τέλος, δίνεται στον χρήστη, ως σχόλιο, ο τρόπος διαχείρισης JSON αρχείων μέσα στη βάση. Η PostgreSQL μας δίνει τη δυνατότητα να έχουμε στήλη στον πίνακα, η οποία αποτελείται από JSON αρχεία και να διαβάσουμε τα δεδομένα χρησιμοποιώντας το σύμβολο ->> μετά το όνομα της στήλης μας και μετά να βάλουμε ποιο από τα εμφωλευμένα δεδομένα του JSON θέλουμε να λάβουμε πίσω.

weather_data (Παράδειγμα για 3 μήνες)

lat	lon	dt	weather	temp	feels_like	temp_min	temp_max	pressure	…	geom
double precision	double precison	timestamp	json	real	real	real	real	real	…	geometry

weather2020_12

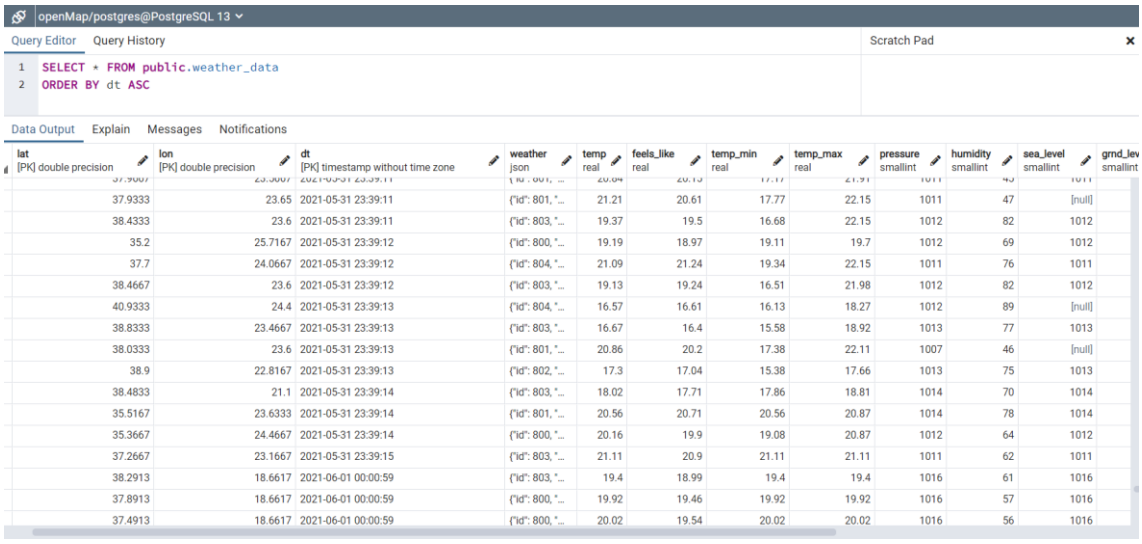
Range partition με το dt,
από dt,
dt >= 1/12/2020
&&
dt<1/1/2021

weather2021_1

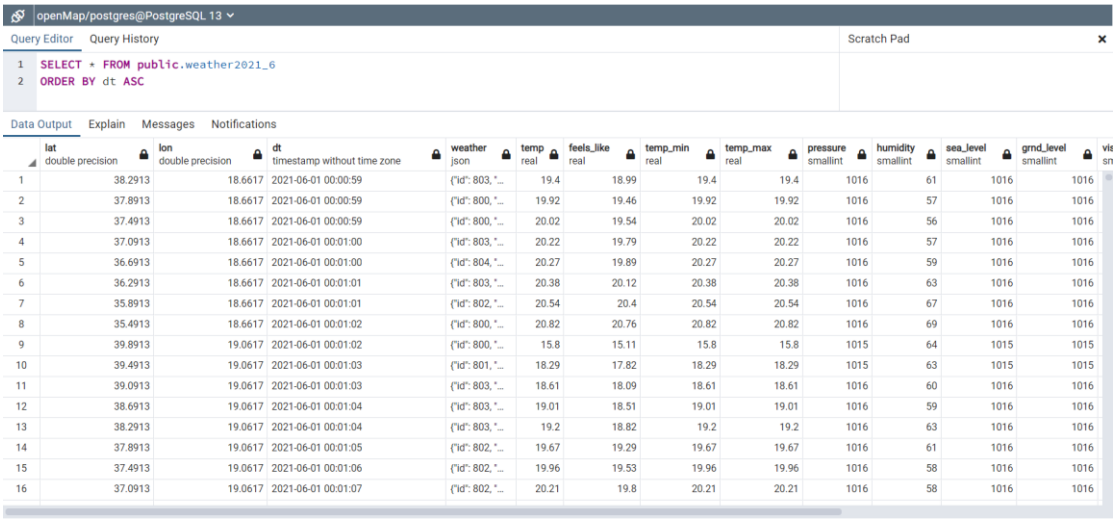
Range partition με το dt,
από dt,
dt >= 1/1/2021
&&
dt<1/2/2021

weather2021_2

Range partition με το dt,
από dt,
dt >= 1/2/2021
&&
dt<1/3/2021



Τα δεδομένα από τον συνολικό πίνακα για Μάιο, Ιούνιο.



openMap/postgres@PostgreSQL 13

Query Editor | Query History | Scratch Pad

```
SELECT * FROM public.weather2021_6
ORDER BY dt ASC
```

Data Output | Explain | Messages | Notifications

	lat double precision	lon double precision	dt timestamp without time zone	weather json	temp real	feels_like real	temp_min real	temp_max real	pressure smallint	humidity smallint	sea_level smallint	grnd_level smallint	vis sn
1	38.2913	18.6617	2021-06-01 00:00:59	{"id": 803, "...	19.4	18.99	19.4	19.4	1016	61	1016	1016	
2	37.8913	18.6617	2021-06-01 00:00:59	{"id": 800, "...	19.92	19.46	19.92	19.92	1016	57	1016	1016	
3	37.4913	18.6617	2021-06-01 00:00:59	{"id": 800, "...	20.02	19.54	20.02	20.02	1016	56	1016	1016	
4	37.0913	18.6617	2021-06-01 00:01:00	{"id": 803, "...	20.22	19.79	20.22	20.22	1016	57	1016	1016	
5	36.6913	18.6617	2021-06-01 00:01:00	{"id": 804, "...	20.27	19.89	20.27	20.27	1016	59	1016	1016	
6	36.2913	18.6617	2021-06-01 00:01:01	{"id": 803, "...	20.38	20.12	20.38	20.38	1016	63	1016	1016	
7	35.8913	18.6617	2021-06-01 00:01:01	{"id": 802, "...	20.54	20.4	20.54	20.54	1016	67	1016	1016	
8	35.4913	18.6617	2021-06-01 00:01:02	{"id": 800, "...	20.82	20.76	20.82	20.82	1016	69	1016	1016	
9	39.8913	19.0617	2021-06-01 00:01:02	{"id": 800, "...	15.8	15.11	15.8	15.8	1015	64	1015	1015	
10	39.4913	19.0617	2021-06-01 00:01:03	{"id": 801, "...	18.29	17.82	18.29	18.29	1015	63	1015	1015	
11	39.0913	19.0617	2021-06-01 00:01:03	{"id": 803, "...	18.61	18.09	18.61	18.61	1016	60	1016	1016	
12	38.6913	19.0617	2021-06-01 00:01:04	{"id": 803, "...	19.01	18.51	19.01	19.01	1016	59	1016	1016	
13	38.2913	19.0617	2021-06-01 00:01:04	{"id": 803, "...	19.2	18.82	19.2	19.2	1016	63	1016	1016	
14	37.8913	19.0617	2021-06-01 00:01:05	{"id": 802, "...	19.67	19.29	19.67	19.67	1016	61	1016	1016	
15	37.4913	19.0617	2021-06-01 00:01:06	{"id": 802, "...	19.96	19.53	19.96	19.96	1016	58	1016	1016	
16	37.0913	19.0617	2021-06-01 00:01:07	{"id": 802, "...	20.21	19.8	20.21	20.21	1016	58	1016	1016	

Τα δεδομένα από τον πίνακα του Ιουνίου.

7. ΤΟ ΠΡΟΤΕΙΝΟΜΕΝΟ ΓΕΩΠΛΗΡΟΦΟΡΙΑΚΟ ΣΥΣΤΗΜΑ ΓΙΑ ΤΟ ΘΑΛΑΣΣΙΟ ΠΕΡΙΒΑΛΛΟΝ – ΚΑΤΑΣΚΕΥΗ ΚΑΙ ΧΡΗΣΗ

Στο προηγούμενο κεφάλαιο αναλύσαμε και μελετήσαμε τα δεδομένα που αποτελούν τον κορμό του πληροφοριακού μας συστήματος. Παρόλα αυτά, τα δεδομένα από μόνα τους δεν είναι αρκετά, έτσι ώστε να κατασκευαστεί ολοκληρωτικά η εφαρμογή διαδικτυακής χαρτογράφησης.

Πέρα λοιπόν από τη βάση δεδομένων, υλοποιήσαμε, με την χρήση εργαλείων ανοιχτού λογισμικού, αλλά και δικού μας κώδικα, ένα σύστημα, το οποίο διαχειρίζεται τα ευαίσθητα γεωγραφικά δεδομένα και δημιουργεί στρώματα χάρτη, τα οποία στη συνέχεια εμείς προσθέτουμε, με την χρήση κώδικα, στην ιστοσελίδα-εφαρμογή μας. Αυτό, λοιπόν, θα εξετάσουμε στο παρόν κεφάλαιο, συζητώντας δύο άξονες κατασκευής, το frontend και το backend, ενώ ταυτόχρονα θα παρουσιάσουμε και τις λειτουργίες που δίνουμε στο χρήστη μέσω της εφαρμογής.

- *Backend*.

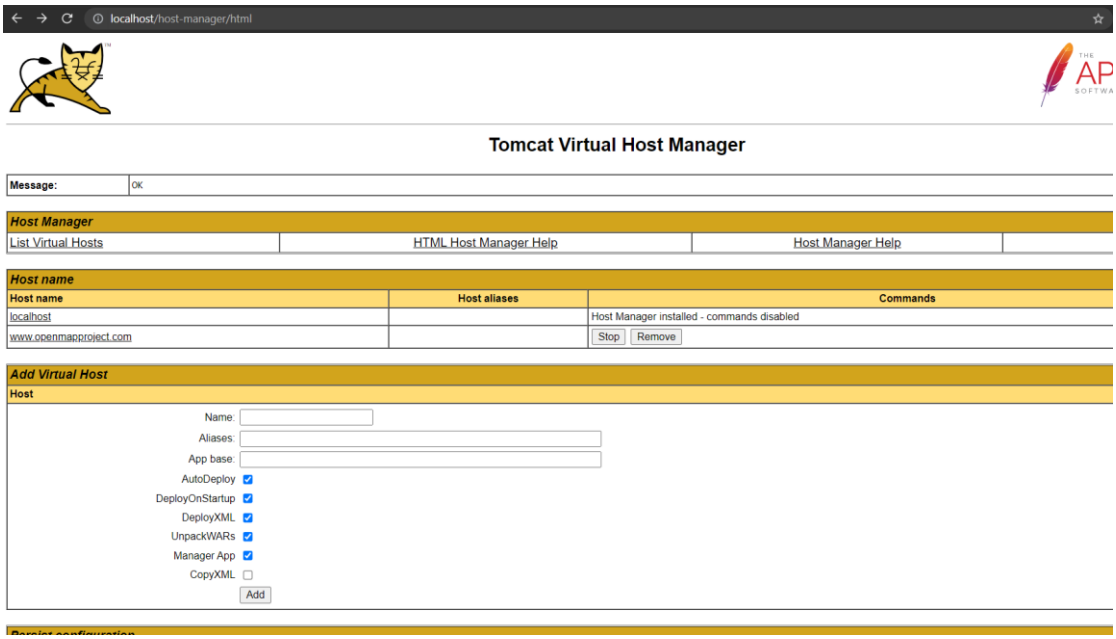
Για την κατασκευή μιας εφαρμογής διαδικτύου απαιτείται να υπάρχει η κατάλληλη υποδομή, η οποία χρησιμοποιείται για αποθήκευση, διαχείριση και μετάδοση των δεδομένων. Η υποδομή αυτή αποτελείται από την βάση δεδομένων (για την οποία μιλήσαμε στο προηγούμενο κεφάλαιο) που είναι υπεύθυνη για την αποθήκευση της πληροφορίας, αλλά και από τον server του συστήματος σε επίπεδο υλικού και λογισμικού. Αυτός είναι υπεύθυνος για την διαχείριση και μετάδοση της πληροφορίας και είναι άκρως σημαντικός στα Γεωπληροφοριακά Συστήματα, καθότι επεξεργάζονται την ευαίσθητη γεωγραφική πληροφορία.

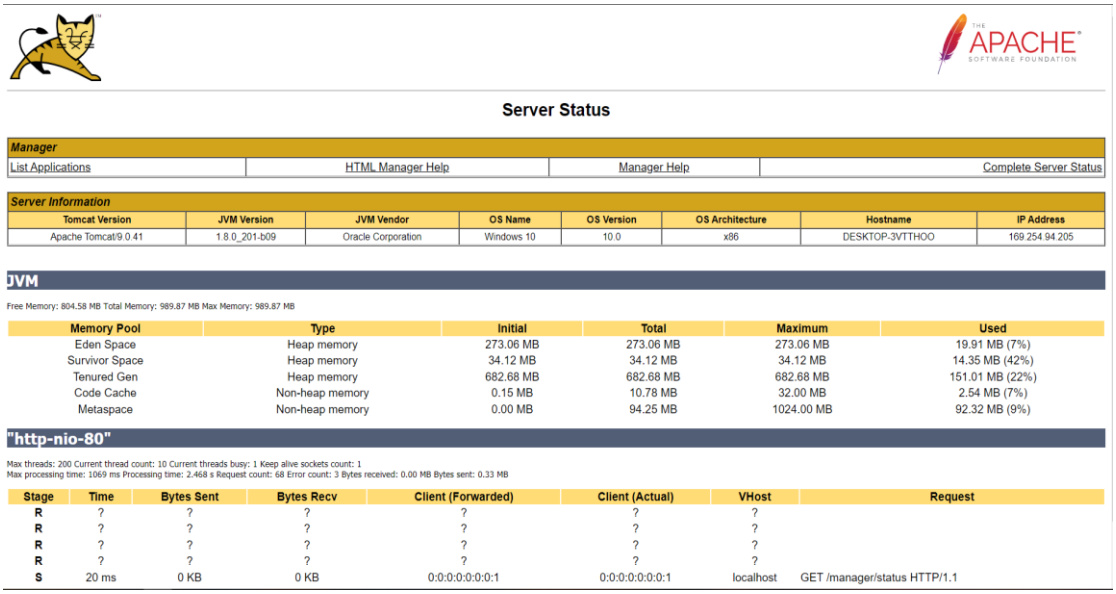
Για την κατασκευή, λοιπόν, της εφαρμογής μας, αρχικά έπρεπε να επιλέξουμε έναν web server, ο οποίος θα δέχεται, θα διαχειρίζεται και θα απαντάει στα αιτήματα των χρηστών. Ταυτόχρονα, θα κάνει κι άλλες διαδικασίες, όπως διαχείριση μνήμης, συγχρονισμός μεταξύ των αιτημάτων χρηστών και caching κάποιων δεδομένων, έτσι ώστε να υπάρχει ταχύτερη ανταπόκριση.

Για τον web server μας, λοιπόν, χρησιμοποιήσαμε το λογισμικό Apache Tomcat (https://tomcat.apache.org/). Μπορούμε να κατεβάσουμε το λογισμικό του Apache Tomcat από τον ιστότοπο, είτε να το χρησιμοποιήσουμε μέσω της εφαρμογής XAMPP. Το Tomcat, αποτελεί πρακτικά ένα servlet, το οποίο χρησιμοποιείται, για να διαχειριστούμε διαδικτυακές εφαρμογές και HTTP αιτήματα που γίνονται σε αυτές (όπως εμφανίζεται σχηματικά στην Εικόνα 5- σελίδα 40), ενώ επιπλέον μπορούμε να εξυπηρετήσουμε πολλούς host, οι οποίοι χρησιμοποιούν τον υπολογιστή στον οποίο τρέχει το Tomcat σαν server.

Για να ξεκινήσουμε τον web server, ακολουθούμε την διαδρομή που έχουμε αποθηκεύσει τα αρχεία του->bin->και τρέχουμε το αρχείο./startup.bat. Έτσι εκκινεί και τρέχει στη θύρα 8080 του υπολογιστή μας, αν δεν υπάρξουν αλλαγές ρυθμίσεων. Εμείς, στην περίπτωσή μας, αλλάξαμε μέσω του αρχείου server.xml, που βρίσκεται στον φάκελο config, τη θύρα στην οποία τρέχει βάζοντας την καθολική θύρα επικοινωνίας του HTTP πρωτοκόλλου, την θύρα 80, ενώ ταυτόχρονα προσθέσαμε σαν host τον ιστότοπο www.openmapproject.com, για να τρέχει το πρότζεκτ μας σε αυτή την εικονική διεύθυνση.

Τέλος, στον φάκελο webapps, κατασκευάσαμε έναν φάκελο που θα περιλαμβάνει τα αρχεία της εφαρμογής μας.





Server Status

Manager

List Applications	HTML Manager Help	Manager Help	Complete Server Status

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture	Hostname	IP Address
Apache Tomcat/9.0.41	1.8.0_201-b09	Oracle Corporation	Windows 10	10.0	x86	DESKTOP-3VTTHOO	169.254.94.205

JVM

Free Memory: 804.58 MB Total Memory: 989.87 MB Max Memory: 989.87 MB

Memory Pool	Type	Initial	Total	Maximum	Used
Eden Space	Heap memory	273.06 MB	273.06 MB	273.06 MB	19.91 MB (7%)
Survivor Space	Heap memory	34.12 MB	34.12 MB	34.12 MB	14.35 MB (42%)
Tenured Gen	Heap memory	682.68 MB	682.68 MB	682.68 MB	151.01 MB (22%)
Code Cache	Non-heap memory	0.15 MB	10.78 MB	32.00 MB	2.54 MB (7%)
Metaspace	Non-heap memory	0.00 MB	94.25 MB	1024.00 MB	92.32 MB (9%)

"http-nio-80"

Max threads: 200 Current thread count: 10 Current threads busy: 1 Keep alive sockets count: 1
Max processing time: 1069 ms Processing time: 2.468 s Request count: 68 Error count: 3 Bytes received: 0.00 MB Bytes sent: 0.33 MB

Stage	Time	Bytes Sent	Bytes Recv	Client (Forwarded)	Client (Actual)	VHost	Request
R	?	?	?	?	?	?	
R	?	?	?	?	?	?	
R	?	?	?	?	?	?	
R	?	?	?	?	?	?	
S	20 ms	0 KB	0 KB	0:0:0:0:0:0:0:1	0:0:0:0:0:0:0:1	localhost	GET /manager/status HTTP/1.1

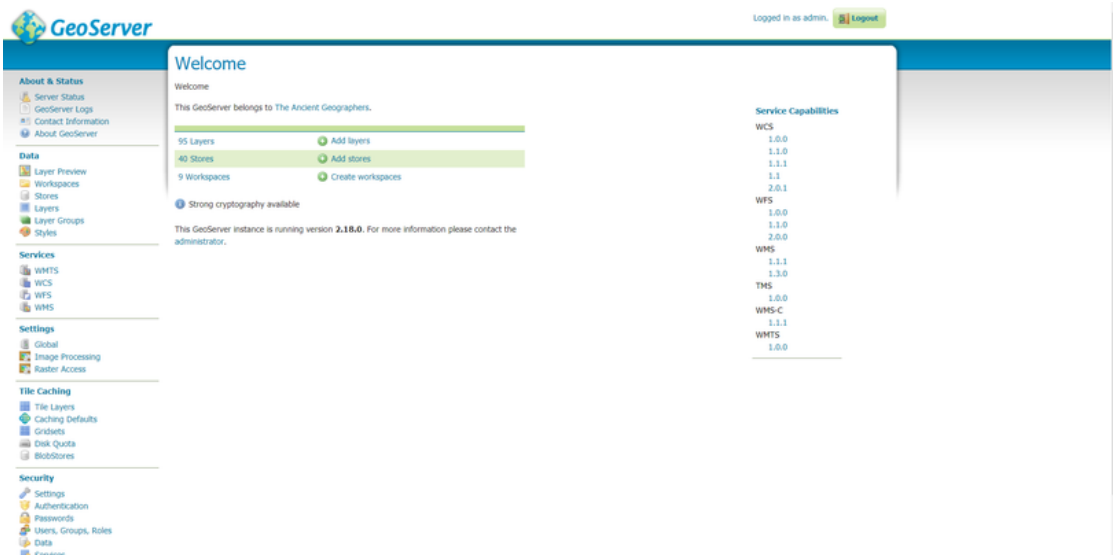
Εικόνα 11: Παρουσιάζουμε στις παραπάνω εικόνες μία εικόνα του Apache web server, από την διεπαφή χρήστη, για την διαχείριση του συστήματος εξυπηρέτησης. Στην πρώτη εικόνα, παρουσιάζεται ο διαχειριστής των εικονικών host που διαχειρίζεται το σύστημα. Στην κάτω εικόνα φαίνεται το σύστημα διαχείρισης του server, με όλα τα δεδομένα όπως διαθέσιμη μνήμη και τύπο μνήμης.

Όπως αναφέραμε και σε προηγούμενα κεφάλαια, πέρα από τον web server, ο οποίος δέχεται και απαντάει (ή και απορρίπτει) HTTP αιτήσεις δικτύου, στα συστήματα γεωγραφικής πληροφορικής υπάρχει και ο GIS server, ο οποίος πρακτικά αποτελεί διαδικτυακή εφαρμογή, η οποία εξυπηρετείται επίσης από τον web server.

Στην περίπτωση της εφαρμογής μας χρησιμοποιούμε τον GeoServer (http://geoserver.org/) ως GIS server. Ο GeoServer συνδέεται με τη βάση δεδομένων μας και ελέγχει την σύνθετη γεωγραφική πληροφορία μέσω των υπηρεσιών που διαθέτει.

Αρχικά, λοιπόν, ο GeoServer προστίθεται ως webapp (εφαρμογή διαδικτύου) πάνω στον ήδη υπάρχοντα web server, τον Apache Tomcat. Ο διαχειριστής του συστήματος προσθέτει τα στοιχεία σύνδεσής του, ενώ τα δεδομένα είναι κρυπτογραφημένα, για να διασφαλιστεί η ασφάλεια του συστήματος.

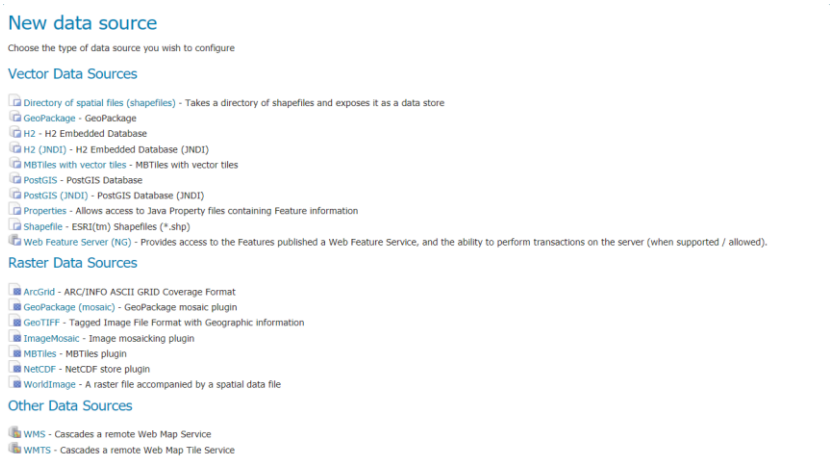
Αφού γίνει η σύνδεση με το σύστημα του GeoServer, ξεκινάμε να κάνουμε ρυθμίσεις, έτσι ώστε να χρησιμοποιήσουμε βέλτιστα τον server για τα γεωγραφικά μας δεδομένα.

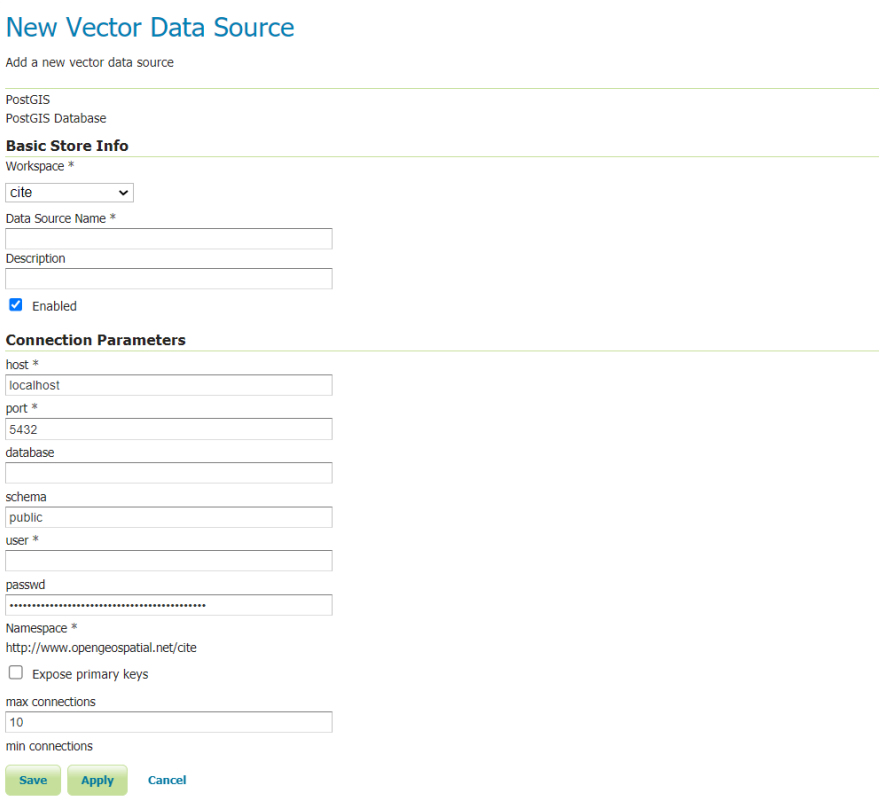


Εικόνα 12: Κεντρική οθόνη της διεπαφής του GeoServer. Βλέπουμε στην αριστερή στήλη τις υπηρεσίες οι οποίες παρέχονται από τον GeoServer, αλλά και τις ρυθμίσεις που μπορούμε να κάνουμε για να διαμορφώσουμε τη χρήση και την ασφάλεια του συστήματος. Επιπλέον, δίνονται επιλογές για το caching των γεωγραφικών δομών.

Πρωτίστως, αυτό που απαιτείται να κάνουμε για τη χρήση του GIS server είναι η δημιουργία ενός χώρου εργασίας (workspace). Εκεί κατασκευάζουμε μία δομή, η οποία μας βοηθάει να διαχειριστούμε καλύτερα τα στρώματα που θα δημιουργήσουμε και στη συνέχεια θα προσθέσουμε στον χάρτη. Πρακτικά αποτελεί μια δομή παρόμοια ενός φάκελου μέσα στο σύστημα διαχείρισης των δεδομένων.

Αφού πραγματοποιήσουμε την κατασκευή του χώρου εργασίας, πρέπει με κάποιο τρόπο να περάσουμε-συνδέσουμε τα δεδομένα μας από τη βάση στον server. Έτσι, πηγαίνουμε στην επιλογή add new store [βλ. Εικόνα 12] και επιλέγουμε το PostGIS Database, για να συνδέσουμε τη βάση δεδομένων μας, όπως φαίνεται από την εικόνα 13.

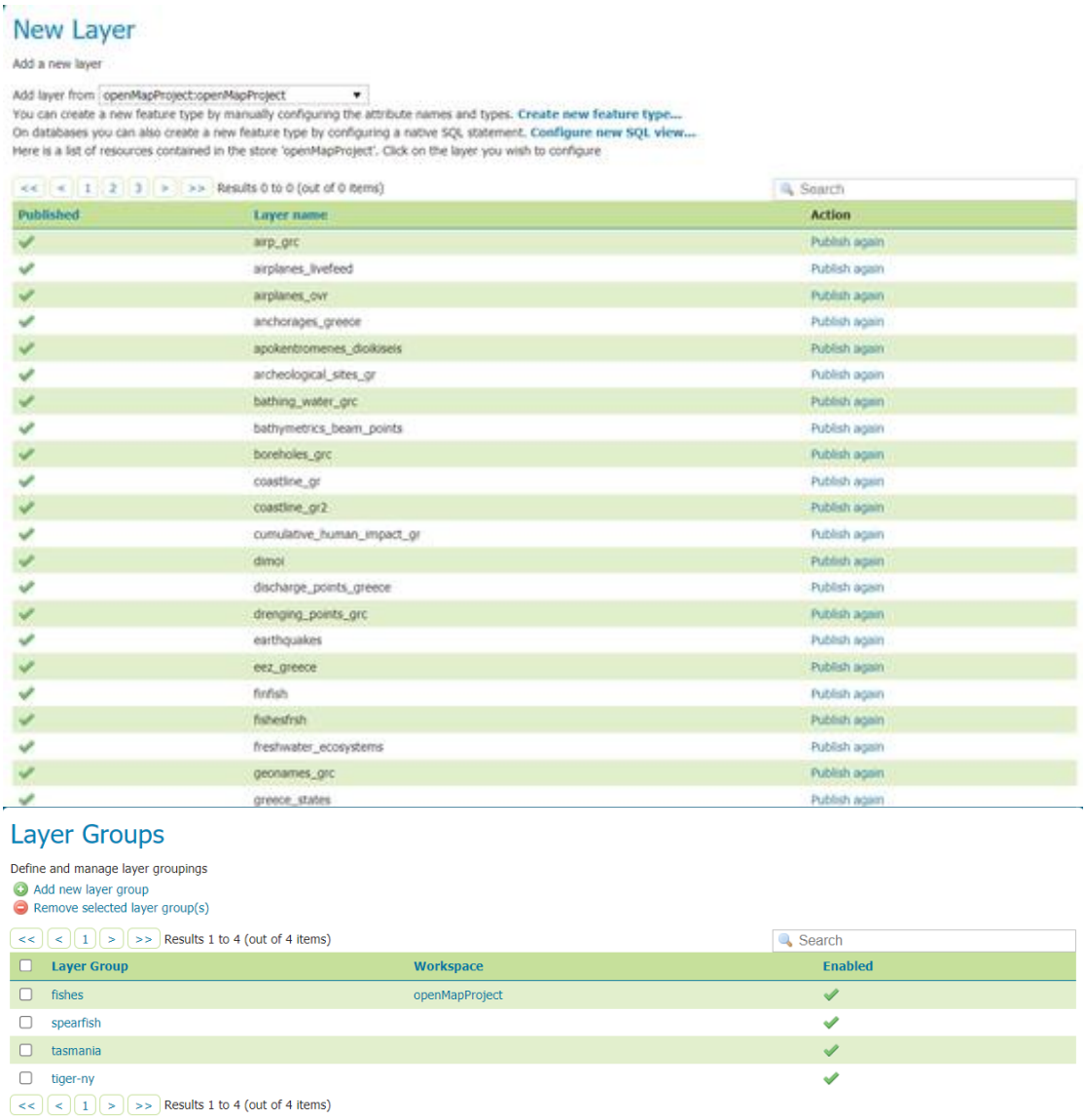




Εικόνα 13: Στην πρώτη εικόνα βλέπουμε την δημιουργία ενός store δεδομένων και τις επιλογές που μας δίνει ο GeoServer. Στην δεύτερη εικόνα έχουμε την επιλογή για τη σύνδεση της βάσης δεδομένων, που περιέχει PostGIS πληροφορία.

Αφού, λοιπόν, δημιουργήσαμε ένα workspace με όνομα openMapProject, στη συνέχεια συνδέσαμε τη βάση δεδομένων μας σε ένα νέο store με ίδιο όνομα. Κατά τη σύνδεση της βάσης δεδομένων, προσθέτουμε βασικές πληροφορίες, όπως σε ποια διεύθυνση και σε ποια θύρα «τρέχει» η βάση δεδομένων, κωδικούς πρόσβασης, αλλά και επιπλέον πληροφορίες για τη σύνδεση, όπως πόσες συνδέσεις μπορούν να γίνουν ταυτόχρονα, τον ρυθμό ανανέωσης των δεδομένων και άλλα. Οι ρυθμίσεις αυτές εναλλάσσονται ανάλογα με τις ανάγκες του εκάστοτε πληροφοριακού συστήματος.

Εν συνεχεία, μετά την κατασκευή του store δεδομένων, στο οποίο συνδέσαμε τη βάση μας, έπρεπε να μετατρέψουμε τα δεδομένα από PostgreSQL πίνακες σε στρώματα χάρτη. Η διαδικασία αυτή έγινε με την επιλογή Layers, και Layer Groups, όπως φαίνεται στην εικόνα 14.



Εικόνα 14: Οι επιλογές New Layer, Layer Groups.

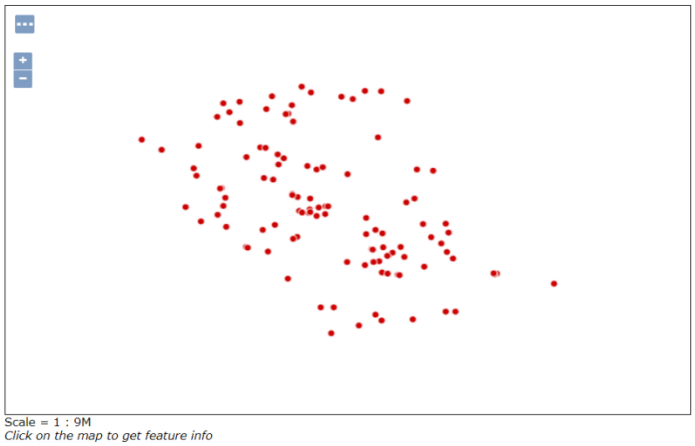
Μεταβαίνοντας, λοιπόν, στην λίστα New Layer, το λογισμικό διαβάζει όλους τους πίνακες της βάσης δεδομένων και τους εμφανίζει ως πιθανά επίπεδα για δημοσίευση. Αφού επιλέξουμε το επίπεδο που θέλουμε, κάνουμε publish. Εκεί εμφανίζεται το layer και οι ρυθμίσεις του. Αλλάζουμε το όνομα, την προβολή που θέλουμε να έχει -αν το επίπεδο έχει ήδη περασμένη προβολή από την εισαγωγή στην βάση, διαβάζεται αυτόματα- και το περιβάλλον κυτίο, στο οποίο εμφανίζονται τα δεδομένα μας. Επιπλέον, επιλέγουμε στυλ εμφάνισης από την καρτέλα publishing. Τα

στυλ αποτελούνται από XML αρχεία εμφάνισης και μπορούμε να τα τροποποιήσουμε από την επιλογή Styles. Τέλος, μπορούμε να ρυθμίσουμε επιλογές ασφαλείας επιλέγοντας ποιοι θα έχουν πρόσβαση στο δεδομένο για read/write -από όλους, μόνο από τον διαχειριστή, μόνο τους αυθεντικοποιημένους χρήστες και συνδυασμοί αυτών-, αλλά και επιλογές για caching.

Με την επιλογή Layer Groups, από την άλλη, μπορούμε να ομαδοποιήσουμε επίπεδα του συστήματος -όπως είχαμε αναφέρει στις ιχθυοκαλλιέργειες για τα οστρακοειδή και τα ψάρια- επιλέγοντας δύο ή παραπάνω επίπεδα της βάσης. Τα ομαδοποιημένα αυτά επίπεδα δημιουργούν ένα νέο επίπεδο.

Για να προσθέσουμε raster αρχεία -όπως GeoTiff και NetCDF αρχεία-, ο GeoServer μας δίνει την επιλογή για δημιουργία store ανά αρχείο raster. Έτσι, από την εικόνα 13 επιλέγουμε από τις επιλογές Raster Data Sources, τις επιλογές GeoTIFF ή NetCDF, ή οποιαδήποτε άλλη έχουμε στην διάθεσή μας. Από εκεί, η διαδικασία δεν διαφέρει σε σχέση με τα δεδομένα της βάσης, καθώς η διαδικασία δημοσίευσης του επιπέδου είναι ακριβώς ίδια. Layers-> Add a new Layer -> Select Store-> Publish Layer-> Ρυθμίσεις -> Publish.

Σημαντική καρτέλα του συστήματος αποτελεί η καρτέλα Layer Preview. Στην καρτέλα αυτή, μπορούμε να δούμε το πώς φαίνεται το στρώμα μας σε ένα χάρτη, αλλά, κυρίως, μας δίνεται η δυνατότητα να δούμε το URI του στρώματος. Αυτό θα χρειαστεί κατά την παραγωγή του frontend της εφαρμογής, κυρίως κατά την κλήση του REST API, μέσω jQuery και AJAX.



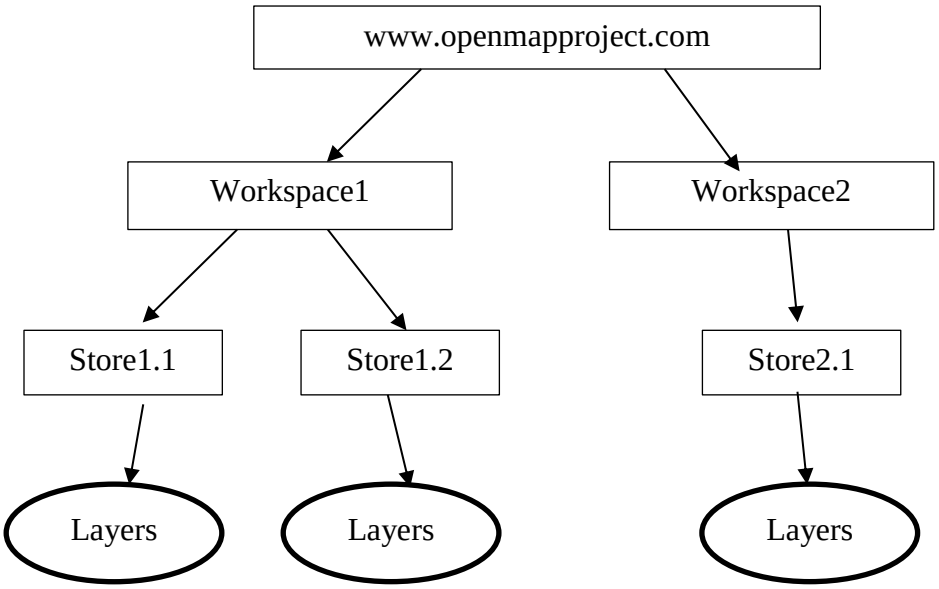
Εικόνα 15: Εικόνα από το Layer Preview του layer των αεροδρομίων. Στο χάρτη βλέπουμε όλα τα σημεία, όπως είναι αποθηκευμένα στην βάση μας.

Το URL για το συγκεκριμένο στρώμα το παρακάτω:

http://www.openmapproject.com/geoserver/openMapProject[1]/wms?service=WMS[2]&version=1.1.0&request=GetMap&layers=openMapProject%3Aairp_grc[3]&bbox=19.42056%2C34.86012%2C29.5764007568%2C41.227475&width=768&height=481&srs=EPSG%3A4326[4]&styles=&format=application/openlayers

Όπου αναλύοντας το βλέπουμε όλη την διαδρομή στο file system του GeoServer, και επιπλέον πληροφορίες για το layer. Αναλύοντας, λοιπόν, τα σημαντικά σημεία του συνδέσμου έχουμε:

1. Το αρχικό path του layer, δηλαδή, σε ποιο ιστότοπο και σε ποιο workspace είμαστε στο "file system" του server. Αυτό μας υποδεικνύει πού είναι αποθηκευμένο και πού θα το αναζητήσει το σύστημα σε πιθανή κλήση.



Εικόνα 16: Εικόνα συστήματος φακέλων (file system).. Το μονοπάτι ξεκινάει από το root του GeoServer και προχωράει μέσα στους χώρους εργασίας, στα stores και μετέπειτα στα κατασκευασμένα στρώματα.

2. Η υπηρεσία που κάλεσε το στρώμα. Οι υπηρεσίες του GeoServer, είναι πολύ σημαντικές για την λειτουργία της εφαρμογής, καθώς αυτές ευθύνονται στην ουσία για την ορθή απάντηση στα requests του χρήστη.
3. Εμφανίζεται η συνέχεια του μονοπατιού, δηλαδή σε ποιο store και ποιο layer έχουμε.

4. Στο σημείο 4 έχουμε την επιλογή της προβολής του στρώματος στον χάρτη. Όπως έχουμε αναφέρει επανειλημμένα, η προβολή είναι πολύ σημαντική, καθότι, αν η προβολή του στρώματος δεν ταυτίζεται με αυτή του χάρτη, τα σημεία θα εμφανίζονται σε λάθος συντεταγμένες.

Με αυτόν τον τρόπο λοιπόν, έχουμε κατασκευάσει τα στρώματα, τα οποία αντιστοιχούν στα δεδομένα που αναφερθήκαμε στο προηγούμενο κεφάλαιο. Αυτά τα δεδομένα, λοιπόν, είναι αποθηκευμένα στη βάση δεδομένων και τροποποιημένα κατάλληλα στον GeoServer. Για να μπορέσουμε να μοιραστούμε αυτά τα δεδομένα, αλλά και να δώσουμε στον χρήστη την δυνατότητα να τα εμφανίσει, να λάβει πληροφορίες γι' αυτά, αλλά και να τα «κατεβάσει», θα χρησιμοποιήσουμε τις υπηρεσίες που μας προσφέρει ο GIS Server μας. Αυτές είναι:

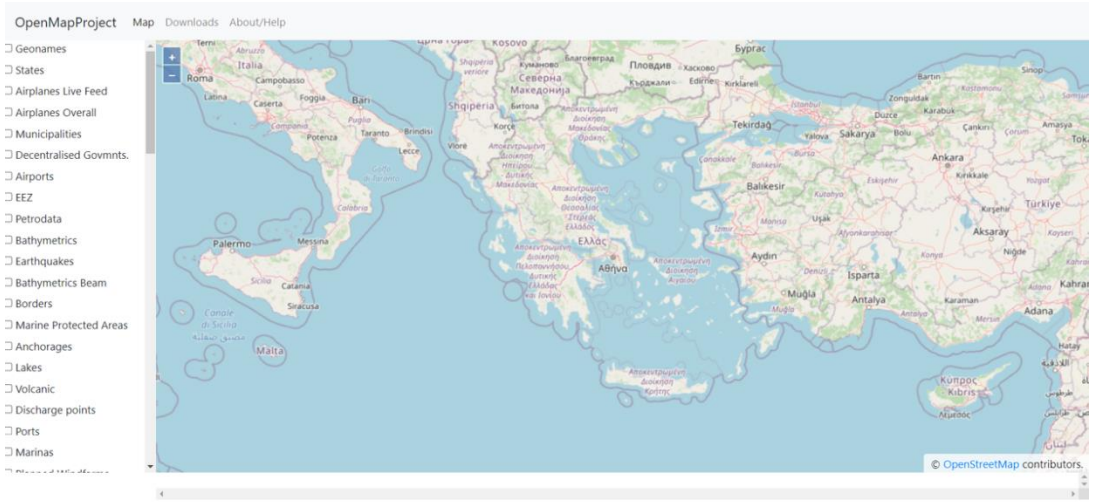
- *WMS (Web Map Services):* Πρακτικά αποτελεί μια HTTP διεπαφή, η οποία επιτρέπει στον χρήστη να ζητάει γεωαναφορικά δεδομένα από τον διαδικτυακό χάρτη. Επιπλέον, δίνει την δυνατότητα να ληφθούν δεδομένα από πολλούς GIS servers και να συνδυαστούν σε μια κοινή εικόνα. Κάποιες από τις διεργασίες που μπορεί να εκτελέσει η υπηρεσία είναι: GetMap –επιστρέφει έναν χάρτη από τον εξυπηρετητή, για μια συγκεκριμένη περιοχή -, GetCapabilities –επιστρέφει μεταδεδομένα για την υπηρεσία WMS, μεταξύ των οποίων και οι διεργασίες- και GetFeatureInfo –που επιστρέφει τα δεδομένα για τις τιμές που έχουν αποθηκευτεί στην βάση δεδομένων για το συγκεκριμένο χαρακτηριστικό. Τα δεδομένα μπορούν να επιστραφούν στον χρήστη σε πολλές μορφές, από XML, JSON, μέχρι και εικόνες.
- *WFS (Web Feature Services):* Μια υπηρεσία κατασκευασμένη από την παγκόσμια κοινοπραξία «ανοιχτών» γεωχωρικών δεδομένων, με σκοπό την παραγωγή, ανταλλαγή και επεξεργασία γεωχωρικών δεδομένων, μέσω HTTP requests. Το WFS είναι επιπλέον υπεύθυνο για την μετατροπή των δεδομένων σε GML γλώσσα, μία υπογλώσσα της XML για γεωγραφικά δεδομένα. Έτσι, μέσα από μία σειρά από ερωτήματα –μέσω REST API ερωτημάτων, GET, POST, DELETE-, κλειδώματα κοινών δεδομένων και μεταφοράς δεδομένων, οι χρήστες έχουν πρόσβαση στα γεωχωρικά δεδομένα της βάσης με διεργασίες, όπως η GetFeature –μοιάζει με την GetFeatureInfo της WMS,

μόνο που εδώ έχουμε δυνατότητα να λάβουμε συγκεκριμένα δεδομένα μέσω Bounding Box και υπάρχουν ερωτήματα τύπου τομής και ένωσης για δεδομένα.

Με τον παραπάνω τρόπο, λοιπόν, έχουμε κατασκευάσει το backend του πληροφοριακού συστήματος. Τα στρώματα του χάρτη μας είναι έτοιμα για προβολή και αναδιανομή. Για να μπορέσει, βέβαια, να γίνει αυτή η διαδικασία, απαιτείται η σωστή κατασκευή του frontend της ιστοσελίδας, όσον αφορά τη μορφή και τη λειτουργικότητα.

- *Frontend- Λειτουργικότητα.*

Σε αυτό το σημείο της εργασίας, θα παραθέσουμε τμήματα κώδικα από το frontend της εφαρμογής.



Εικόνα 17: Κύρια οθόνη εφαρμογής.

Ξεκινάμε από την κυρίως καρτέλα της εφαρμογής ή αλλιώς το index αρχείο html. Πάνω έχουμε τις καρτέλες, χρησιμοποιώντας navigation bar από το bootstrap. Στο κεντρικό τμήμα της οθόνης έχουμε ένα 2x2 πλέγμα, όπου χωρίζεται στο αριστερό πάνω τμήμα, όπου έχουμε τα διαθέσιμα στρώματα, ενώ στο δεξί πάνω τμήμα έχουμε τον χάρτη. Στο κάτω τμήμα έχουμε τη γραμμή, όπου θα εμφανίζονται οι πληροφορίες του layer.

```
<html>
<head>
   <title>OpenMapProject</title>
   <!--openLayers-->
   <script src="https://cdn.jsdelivr.net/gh/openlayers/openlayers.github.io@master/en/v6.5.0/build/ol.js"
crossorigin="anonymous"></script>
   <script src="https://code.jquery.com/jquery-3.5.1.min.js" crossorigin="anonymous"></script>
   <link rel="stylesheet" href="http://openlayers.org/en/v6.5.0/css/ol.css" type="text/css">
   <!-- bootsrap-->
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
   <link rel="stylesheet" href="./styles.css">
   <link rel="stylesheet" href="./popup.css">
   <!--ShapeFileWriter -->
   <script src="./myshpwrite.js"></script>
</head>
<body style="overflow-y: auto; overflow-x: auto;">
   <nav class="navbar navbar-expand-lg navbar-light bg-light">
      <a class="navbar-brand" >OpenMapProject</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-
controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
          <a class="nav-item nav-link active" href="">Map <span class="sr-only">(default)</span></a>
          <a class="nav-item nav-link" href="downloads.html" >Downloads</a>
          <a class="nav-item nav-link" href="about.html" >About/Help</a>
        </div>
      </div>
    </nav>
   <div class="grid-container-fluid">
      <div class="grid-fluid" name="grid" style=" overflow-y: scroll;" >
         <input type="checkbox" id="layerve" name="layerve" value="layerve">
         <label for="layerve"> Layer Vector </label><br>
         <input type="checkbox" id="layerra" name="layerra" value="layerra">
         <label for="layerra"> Layer Raster </label><br>
      </div>
      <div class="grid" name="central">
         <div id="map" class="map"></div>
         <div id="popup" class="ol-popup">
           <a href="#" id="popup-closer" class="ol-popup-closer"></a>
           <div id="popup-content"></div>
      </div>
      <div class='grid' name="info" id="info" style=" overflow-x: scroll;  overflow-y: scroll;">
        
      </div>
   </div>

   <div  name="box" id="box">
     
   </div>

   <script src="main.js"></script>
   <script src="layers.js"></script>
   <script src="controlbox.js"></script>
   <script src="switcher.js"></script>
   <script src="eventlisteners.js"></script>
</body>
</html>
```

Κώδικας 1: index.html

Κάθε επίπεδο σηματοδοτείται ως ένα checkbox στην αριστερή λίστα. Ο χρήστης πιέζει το κατάλληλο πλήκτρο, για να εμφανίσει το επίπεδο που θέλει. Επίσης, στο index.html κάνουμε κλήση στα αρχεία js και css, που έχουμε για εμφάνιση και λειτουργικότητα της εφαρμογής μας.

```
.ol-popup {
    position: absolute;
    background-color: white;
    -webkit-filter: drop-shadow(0 1px 4px rgba(0,0,0,0.2));
    filter: drop-shadow(0 1px 4px rgba(0,0,0,0.2));
    padding: 35px;
    border-radius: 10px;
    border: 1px solid #cccccc;
    bottom: 8px;
    left: -50px;
  }
  .ol-popup:after,.ol-popup:before {
    top: 100%;
    border: solid transparent;
    content: " ";
    height: 0;
    width: 0;
    position: absolute;
    pointer-events: none;
  }
  .ol-popup:after {
    border-top-color: white;
    border-width: 10px;
    left: 48px;
    margin-left: -10px;
  }
  .ol-popup:before {
    border-top-color: #cccccc;
    border-width: 11px;
    left: 48px;
    margin-left: -11px;
  }
  .ol-popup-closer {
    text-decoration: none;
    position: absolute;
    top: 2px;
    right: 8px;
  }
  .ol-popup-closer:after {
    content: "X";
  }
```

```
*{
    box-sizing: border-box;
}

body{
    margin: 0;
    padding: 0;
}

.grid-container-fluid{
    display: grid;
    grid-template-columns: 14vw 86vw ;
    grid-template-rows: 80vh 12vh;


}
.sidebar{
    margin-left: 15px;
}

.ol-dragbox {
    background-color: rgba(18, 15, 218, 0.4);
    border-color: rgb(249, 250, 247);
  }
```

Κώδικας 2: Έχουμε τα css αρχεία για το pop-up παράθυρο συντεταγμένων και την εμφάνιση της κυρίως καρτέλας.

Στα css αρχεία έχουμε κατασκευάσει ένα παράθυρο, στο οποίο ο χρήστης μπορεί να πατήσει στον χάρτη και να εμφανιστούν οι συντεταγμένες. Αυτό εμφανίζεται στο popup.css (Κώδικας 2), ενώ το παραθυρικό αυτό περιβάλλον είναι διαθέσιμο μέσω της βιβλιοθήκης openLayers. Η γενικότερη λειτουργικότητα του παραθύρου εμφανίζεται στο main.js (Κώδικας 3).

```
var container = document.getElementById('popup');
var content = document.getElementById('popup-content');
var closer = document.getElementById('popup-closer');

/**
 * Add a click handler to hide the popup.
 * @return {boolean} Don't follow the href.
 */
closer.onclick = function() {
  overlay.setPosition(undefined);
  closer.blur();
  return false;
};

/**
 * Create an overlay to anchor the popup to the map.
 */
var overlay = new ol.Overlay(/** @type {olx.OverlayOptions} */ ({
  element: container,
  autoPan: true,
  autoPanAnimation: {
    duration: 250
  }
}));

var view= new ol.View({
  center: ol.proj.fromLonLat([24.0, 38.0]),
  zoom: 6.25,
  minZoom:6.25,

})

var map = new ol.Map({
    target: 'map',
    overlays:[overlay],
    projection: 'EPSG:4326',
    layers: [
      new ol.layer.Tile({
        source: new ol.source.OSM(),
      })
    ],
    view:view
  });
  map.on('singleclick', function(evt) {
    var coordinate = evt.coordinate;
    var hdms = ol.coordinate.toStringHDMS(ol.proj.transform(
        coordinate, 'EPSG:3857', 'EPSG:4326'));

    content.innerHTML = '<p>Co-ords:</p><code>' + hdms +
        '</code>';
    overlay.setPosition(coordinate);
  });
```

Κώδικας 3: Το main.js

Στο JavaScript αρχείο χρησιμοποιούμε τη βιβλιοθήκη openLayers και τον Open Street Map API, για να καλέσουμε τον βασικό χάρτη που θα εμφανίζεται στον χρήστη. Ο χάρτης είναι λεγόμενος “slippy“ Tiled map, δηλαδή αποτελείται από τμήματα-πλακίδια δεδομένων. Αυτά φορτώνονται αρχικά σε ένα βάθος πληροφορίας, και, όσο κάνουμε ζουμ σε ένα σημείο, γίνεται περισσότερο φόρτωμα πληροφορίας, χωρίζοντας εκείνο το τμήμα του χάρτη σε πιο μικρά πλακίδια. Η τακτική αυτή χρησιμοποιήθηκε και στα raster στρώματα. Επιπλέον, στο main.js κάνουμε

αρχικοποίηση του ζουμ, της προβολής του χάρτη και της θέσης που ξεκινάμε να βλέπουμε τον χάρτη.

Τέλος, χρησιμοποιούμε Event Listeners της JavaScript, ώστε να εντοπίζουμε το κλικ του χρήστη στον χάρτη. Εφόσον εντοπιστεί κλικ, διαβάζουμε τις συντεταγμένες από τον χάρτη και η πληροφορία εμφανίζεται στο HTML στοιχείο του pop-up παραθύρου (Κώδικας 2).

```
var layer_vector= new ol.layer.Image({
   target: 'map',
   visible:false,
   source:new ol.source.ImageWMS({
      url:'http://www.openmapproject.com/geoserver/openMapProject/wms',
      params:{'LAYERS':'openMapProject:layer1'},
      serverType:'geoserver'
   })
});

var layer_raster= new ol.layer.Tile({
   target: 'map',
   visible:false,
   source:new ol.source.TileWMS({
      url:'http://www.openmapproject.com/geoserver/rasterData/wms',
      params:{'LAYERS':'rasterData:layer2'},
      serverType:'geoserver'
   })
});

map.addLayer(layer_vector);
map.addLayer(layer_raster);
```

Κώδικας 4: Ο κώδικας για το layer.js, όπου τα layers προθέτονται στον χάρτη. Χρησιμοποιούμε δύο ενδεικτικά στρώματα στο παράδειγμα. Κανονικά η διαδικασία γίνεται για κάθε νέο layer.

Στην συνέχεια, αφού έχουμε την αρχική υποδομή του χάρτη προσθέτουμε τα στρώματα, όπως ενδεικτικά καταδεικνύεται στον κώδικα 4. Στο παράδειγμα έχουμε 2 στρώματα, ένα εικονικό vector επίπεδο, το οποίο εμπεριέχει δεδομένα από την βάση δεδομένων και ένα raster επίπεδο. Τα vector επίπεδα τα φορτώνουμε ως Image επίπεδα του OpenLayers, καθώς είναι μικρά σε μέγεθος και η φόρτωσή τους άμεση. Τα raster στρώματα τα εμφανίζουμε ως Tiles, όπως αναφέραμε και νωρίτερα για τον χάρτη. Αυτό γίνεται και για την αποσυμφόρηση του συστήματος από την άμεση προσπέλαση πολλών «μη χρήσιμων» δεδομένων, την πιο γρήγορη φόρτωση, αλλά και την πληρέστερη πληροφορία, ανάλογα με το ζουμ.

Ο κώδικας 4 φορτώνει το επίπεδο με βάση το URL που είναι αποθηκευμένο στο GeoServer, μέσω της υπηρεσίας WMS, όπως αναφέραμε προηγουμένως. Επιπλέον, «στοχεύει» το επίπεδο στον χάρτη σύμφωνα με το id του και κάνει αρχικά το layer μη εμφανές στον χρήστη. Δηλαδή, είναι σαν να καρφιτσώνει το επίπεδο στον χάρτη και να το κρύβει, μέχρι ο χρήστης να πατήσει το αντίστοιχο κουμπί στη λίστα, όπως θα δούμε στον κώδικα 5.

```
function switcher(button,layer){
   button.addEventListener('change', function() {
      if (this.checked) {
        layer.setVisible(true);
      }
      else{
        layer.setVisible(false);
      }
     });
}

switcher(layerve,layer_vector);
switcher(layerra,layer_raster);
```

Κώδικας 5: Ο κώδικας switcher.js

Στο επόμενο κομμάτι, πρέπει να κάνουμε κάθε επίπεδο εμφανές, κατά βούληση του χρήστη του συστήματος. Κάθε επίπεδο είναι εξ αρχής μη εμφανές στον χρήστη, αλλά φορτωμένο στο background. Επιπλέον, κάθε κουμπί της λίστας σηματοδοτείται από ένα μοναδικό id. Έτσι χρησιμοποιούμε την συνάρτηση switcher, με ορίσματα το id του κουμπιού και το όνομα του επιπέδου και λειτουργεί για όλα τα επίπεδα του συστήματος. Όταν ένα checkbox είναι ενεργοποιημένο, η κίνηση εντοπίζεται και γίνεται ταυτόχρονα εμφανές το επίπεδο στον χρήστη.

```
function changer(button,layer){
  map.on("click", function (evt) {
    if(button.checked){
      var viewResolution = view.getResolution();
      var url = layer.getSource().getFeatureInfoUrl(
        evt.coordinate, viewResolution, view.getProjection(),
        { 'INFO_FORMAT': 'text/html' ,format_options: 'callback: getHtml'});
      if(url){
        $.ajax({
          type: "GET",
          url:url,
          dataType: 'html',
```

```
        callback: 'geHtml',
        success: function (data) {
          document.getElementById('info').innerHTML =data;
       }
      });
     }
    }
  });
};
changer(layerve,layer_vector)
changer(layerra,layer_raster);
```

Κώδικας 6: Ο κώδικας eventlisteners.js

Πέρα όμως από την εμφάνιση του layer στον χρήστη, πρέπει τα αποθηκευμένα μεταδεδομένα να γίνουν και αυτά άμεσα προσπελάσιμα. Για να το πετύχουμε αυτό, κατασκευάσαμε την συνάρτηση changer. Η συνάρτηση αυτή δέχεται τα ίδια ορίσματα με την switcher και στην συνέχεια ελέγχει για «κλικ» πάνω στον χάρτη. Αν υπάρχει κάποιο ή κάποια επίπεδα ενεργοποιημένα, τότε διαβάζει το url του ενεργοποιημένου επιπέδου, υπολογίζει τις συντεταγμένες και την προβολή από το χάρτη και κάνει AJAX με την χρήση jQuery. Το ερώτημα AJAX γίνεται μέσω REST API στον GeoServer και χρησιμοποιούμε το GET, για να λάβουμε τα δεδομένα του επιθυμητού layer. Αν το ερώτημα είναι έγκυρο, τότε τα δεδομένα επιστρέφονται με τη μορφή πίνακα html και εμφανίζονται στο κάτω κομμάτι της κυρίως καρτέλας, στο πλαίσιο με id “info”.

```
var select = new ol.interaction.Select();
map.addInteraction(select);

var selectedFeatures = select.getFeatures();

var dragBox = new ol.interaction.DragBox({
  condition: ol.events.condition.shiftKeyOnly,
  style: new ol.style.Style({
    stroke: new ol.style.Stroke({
      color: [0, 0, 255, 1]
    })
  })
});

map.addInteraction(dragBox);

var infoBox = document.getElementById('box');

    dragBox.on('boxend', function() {

        var lonList=[]
        var latList=[]

        var string = dragBox.getGeometry().getCoordinates();
```

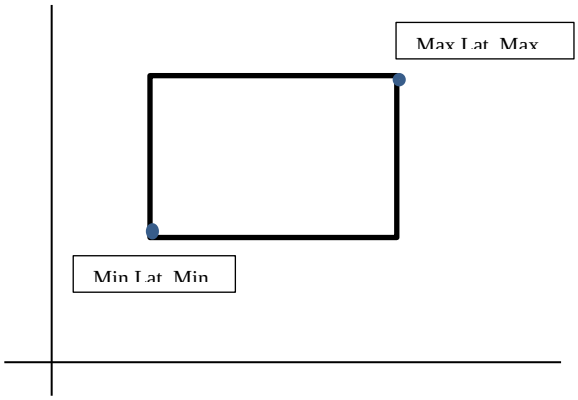
```
      for(i=0;i<4;i++){
        string[0][i]=ol.proj.transform(string[0][i],'EPSG:3857', 'EPSG:4326');
        lonList.push(string[0][i][0])
        latList.push(string[0][i][1])
      }

      var minLon=Math.min.apply(Math,lonList)
      var minLat=Math.min.apply(Math,latList)
      var maxLon=Math.max.apply(Math,lonList)
      var maxLat=Math.max.apply(Math,latList)


      window.alert("Min Longitude:  "+minLon+"\n"+"Max Longitude:  "+maxLon+"\n"+"Min
Latitude:  "+minLat+"\n"+"Max Latitude:  "+maxLat)
      var param;
      function getFeature(button,layer){
        if(typeof(layer)!="string"){
          param=(layer.getSource()['params_']['LAYERS'])
        }
        else{
          param=layer
        }

        if(button.checked){
            var url=
"http://www.openmapproject.com/geoserver/wfs?service=WFS&version=2.0.0&request=GetFea
ture&typeNames="+ param
+"&srsName=EPSG:4326&maxFeatures=1000000&outputFormat=json&format_options=callba
ck:getJson&bbox="+minLon + "," + minLat + "," + maxLon + "," + maxLat+",EPSG:4326"  ;
          if(url){
            $.ajax({
              type:"GET",
              url: url,
              dataType: 'json',
              jsonpCallback: 'getJson',
              contentType: 'application/json',
              success: function(data){
                if(data['features']!=0){
                  shpwrite.download(data)
                  //console.log(data)
                  }
                }
            });
          }
        };
      }
      getFeature(layerve,layer_vector)
      getFeature(fishes,'openMapProject:shellfish')
      getFeature(fishes,'openMapProject:finfish')
```

Κώδικας 7: Ο κώδικας ελέγχου του bounding box. controlbox.js

Εν συνεχεία, θέλαμε να προσθέσουμε έναν τρόπο με τον οποίο ο χρήστης θα μπορεί να λάβει τα δεδομένα δυναμικά, δηλαδή, πέρα από την καρτέλα για τα download, ο χρήστης να μπορεί να λάβει και μεμονωμένα δεδομένα και να τα κατεβάζει σε μορφή shapefile, για να τα προσθέσει στην βάση δεδομένων του. Για να το πετύχουμε αυτό, κατασκευάσαμε το js αρχείο control box, το οποίο ελέγχει την κατασκευή περιβάλλοντος κυτίου μέσα στον χάρτη, μια επιλογή που είναι διαθέσιμη μέσω του OpenLayers API.

Για να κατασκευαστεί το bounding box, ο χρήστης πιέζει το shift και «τραβάει» τον κέρσορα του μέχρι το σημείο που τον ενδιαφέρει. Οι συντεταγμένες των 4 σημείων αποθηκεύονται σε μία λίστα, μετατρέπονται στο κατάλληλο σύστημα συντεταγμένων και αποθηκεύονται σε δύο λίστες για γεωγραφικό πλάτος και γεωγραφικό μήκος. Μετά, με την χρήση της βιβλιοθήκης Math, κρατάμε τις ελάχιστες και τις μέγιστες τιμές από κάθε λίστα και τις εμφανίζουμε στον χρήστη.

Σε περίπτωση που υπάρχει ενεργοποιημένο επίπεδο στον χάρτη, τότε χρησιμοποιούμε την υπηρεσία WFS του GeoServer και πιο συγκεκριμένα της διεργασίας GetFeature. Για να πετύχουμε το σωστό αίτημα στον GeoServer και για να μην εμφανιστεί Error, χρησιμοποιούμε ως παραμέτρους το όνομα του layer, το σύστημα συντεταγμένων και τις τέσσερις γωνίες του περιβάλλοντος κυτίου -με χρήση ελάχιστης και μέγιστης γεωγραφικής συντεταγμένης, όπως φαίνεται στην εικόνα 18. Σημαντική σημείωση αποτελεί ότι η χρήση της συνάρτησης δεν μπορεί να γίνει σε raster επίπεδα. Αυτά μπορούν να κατέβουν μόνο ολόκληρα. Τέλος, βλέπουμε την διαφορά στον κώδικα 7 σε περίπτωση Group Layer. Επειδή η πληροφορία είναι διαφορετική, απλά ομαδοποιημένη, πρέπει να «σπάσουμε» την κλήση της συνάρτησης στα υποστρώματα του ομαδοποιημένου layer.



Εικόνα 18

Το παράθυρο που εμφανίζει τις συντεταγμένες διαφαίνεται ανεξάρτητα με την ύπαρξη ενεργού επιπέδου στον χάρτη. Σε περίπτωση ενεργού επιπέδου, γίνεται πάλι μέσω AJAX και JQuery, κλήση GET στον REST API της υπηρεσίας WFS και επιστρέφεται, αρχικά, Json αρχείο. Τέλος, χρησιμοποιούμε την βιβλιοθήκη shp-write από την πλευρά του client *[41]*, για να μετατρέψουμε τα δεδομένα σε shapefile και στην συνέχεια χρησιμοποιούμε το download, για να γίνει κατέβασμα του zip αρχείου.

Σημαντικό για την ορθή λειτουργία του συστήματος αποτελεί η σωστή ρύθμιση του CORS (Cross- Origin Resource Sharing), έτσι ώστε οι πληροφορίες να είναι διαθέσιμες από τον εξυπηρετητή σε πιθανές κλήσεις μέσω AJAX. Για να γίνει η ρύθμιση αυτή, μεταβαίνουμε στο Apache Tomcat *[42]* στο web.xml, του φακέλου config και αλλάζουμε στο σημείο CORS access control, βάζοντας * που σημαίνει ότι επιτρέπονται τα CORS requests από όλους τους χρήστες.

```
<filter>
  <filter-name>CorsFilter</filter-name>
  <filter-class>org.apache.catalina.filters.CorsFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>CorsFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

```
<html>
   <head>
      <title>Downloads</title>
      <!—bootsrap→
      <link r”l="stylesh”et"
hr”f="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.”ss"
integri”y="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6”Xm"
crossorig”n="anonym”us">
      <link r”l="stylesh”et" hr”f="./downloads.”ss">
   </head>

   <body>
      <nav cla”s="navbar navbar-expand-lg navbar-light bg-li”ht">
         <a cla”s="navbar-br”nd" >OpenMapProject</a>
         <button cla”s="navbar-togg”er" ty”e="but”on" data-togg”e="colla”se" data-
targ”t="#navbarNavAltMar”up" aria-contro”s="navbarNavAltMar”up" aria-expand”d="fa”se"
```

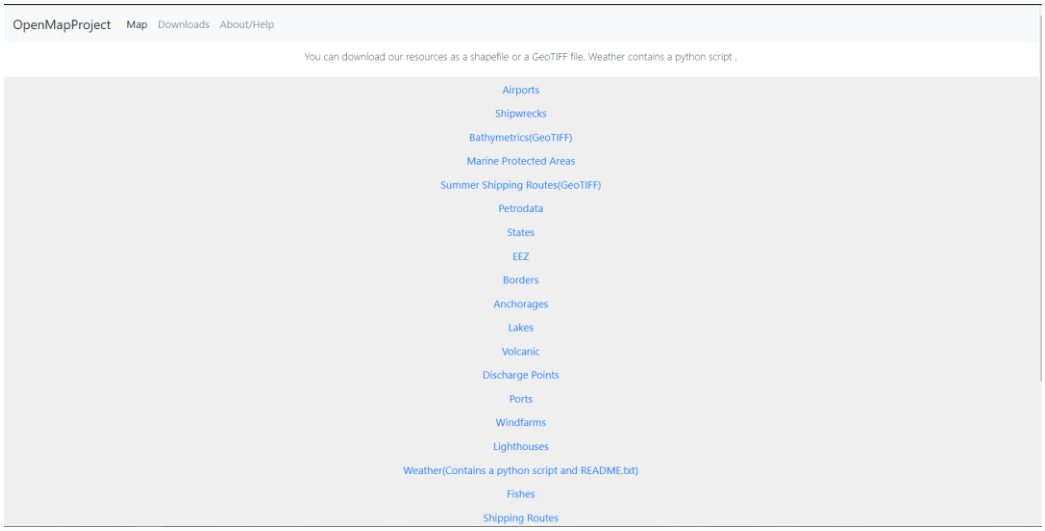
```
aria-lab”l="Toggle navigat”on">
            <span cla”s="navbar-toggler-i”on"></span>
          </button>
          <div cla”s="collapse navbar-colla”se" ”d="navbarNavAltMar”up">
            <div cla”s="navbar-”av">
              <a cla”s="nav-item nav-link act”ve" hr”f="index.h”ml">Map <span cla”s="sr-
o”ly">(default)</span></a>
              <a cla”s="nav-item nav-l”nk" hr”f="downloads.h”ml" >Downloads</a>
              <a cla”s="nav-item nav-l”nk" hr”f="about.h”ml" >About/Help</a>
            </div>
          </div>
        </nav>
        <p cla”s="font-weight-li”ht" sty”e="margin-top: 10”x;">You can download our resources
as a shapefile or a GeoTIFF file. Weather contains a python script.</p>
        <div cla”s="btn-group-verti”al" sty”e="margin-top: 10”x;">
          <button cla”s=””tn"><i cla”s="fa fa-downl”ad"></i> <a
hr”f="download_resources\layer_vector.”ip" downlo”d="layer_vec”or">Layer
Vector</a></button>
          <button cla”s=””tn"><i cla”s="fa fa-downl”ad"></i> <a
hr”f="download_resources\layer_raster.”ip" downlo”d="layer_ras”er">Layer
Raster</a></button>
        </div>
    </body>
</html>
```

Κώδικας 8: Κώδικας download.html. Εδώ χρησιμοποιούμε το download και κατεβάζουμε ολόκληρα τα αρχεία από τον φάκελο που είναι αποθηκευμένα. Η μορφοποίηση έγινε με χρήση στοιχείων HTML και Bootstrap.

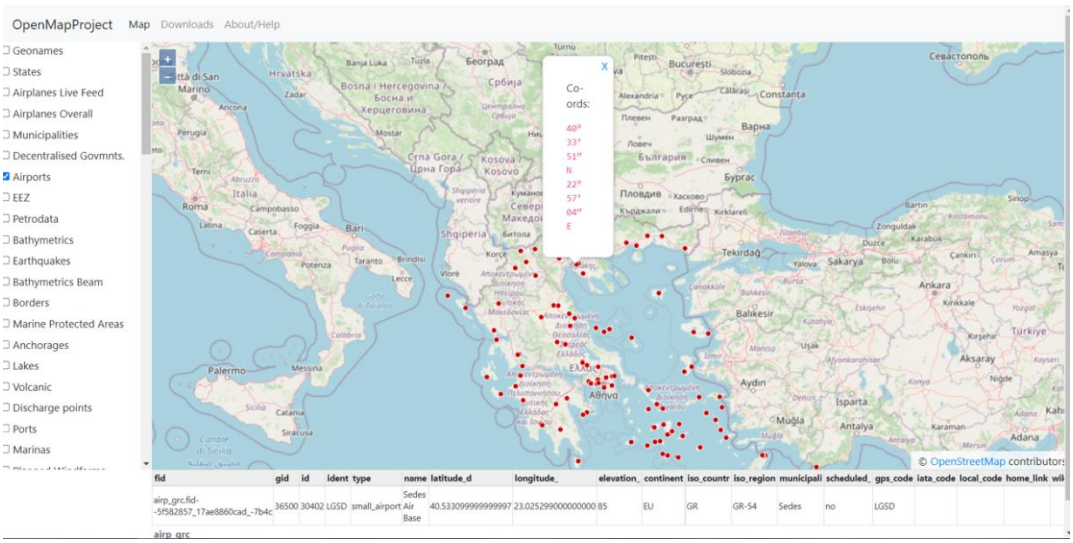
Γενικά, λοιπόν, ο χρήστης μπορεί:

- Να ενεργοποιήσει/απενεργοποιήσει στρώματα χάρτη, για να τα δει.
- Να λάβει συγκεκριμένη τοποθεσία.
- Να λάβει τις γεωγραφικές συντεταγμένες ενός περιβάλλοντος κυτίου.
- Να λάβει τις πληροφορίες ενός στρώματος.
- Να κάνει ζουμ σε ένα raster αρχείο, ώστε να δει περισσότερη πληροφορία.
- Να κατεβάσει ολόκληρα layers.
- Να κατεβάσει τμήματα των επιπέδων χρησιμοποιώντας το πλήκτρο “shift” και σχηματίζοντας ένα περιβάλλον κυτίο.
- Να κατεβάσει source codes που περιέχονται και στο παράρτημα.
- Να κατεβάσει μεμονωμένα δεδομένα ενός στρώματος

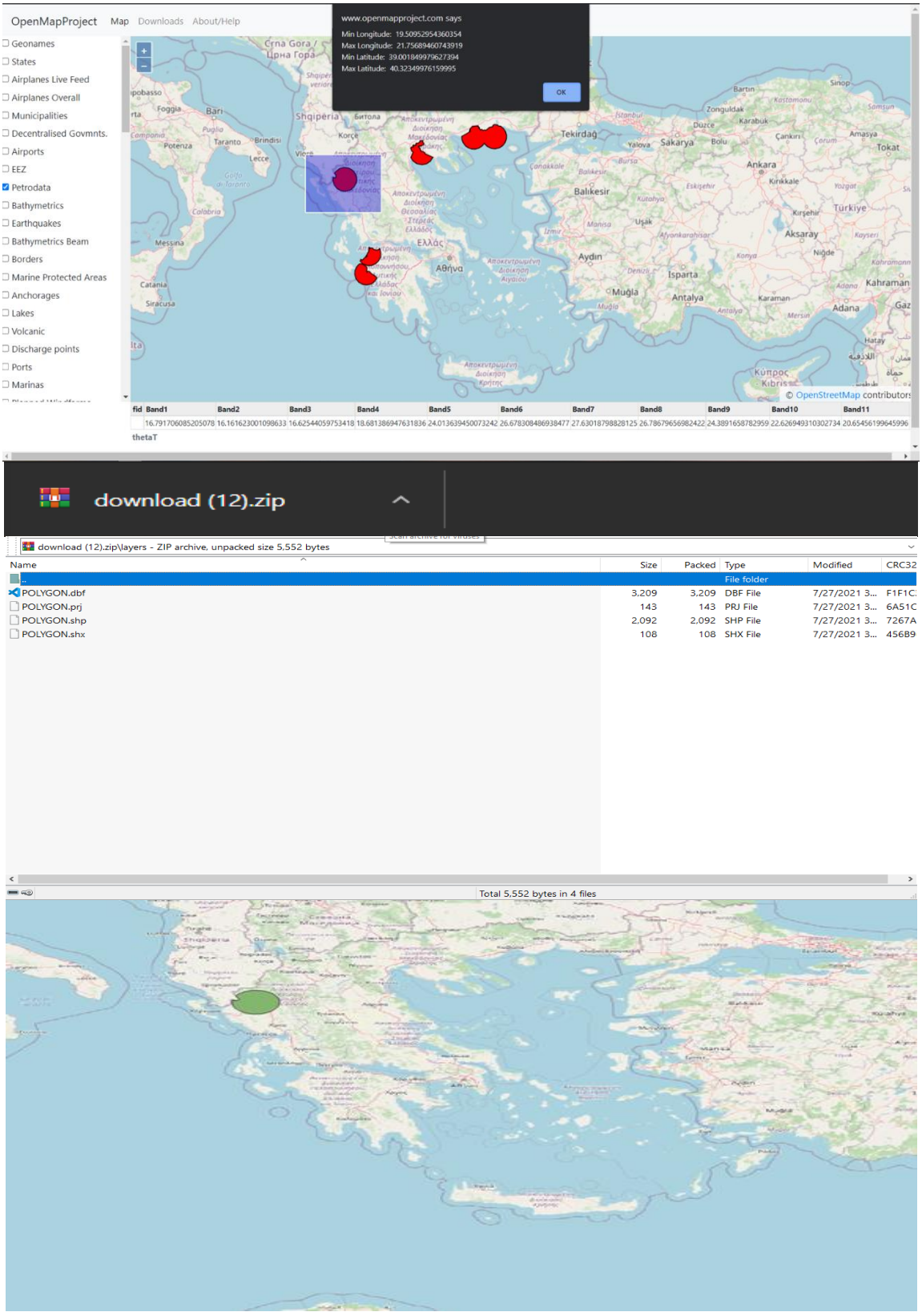
Τα δεδομένα που παρέχονται για αναδιανομή, αλλά και ολόκληροι οι κώδικες του frontend της εφαρμογής διατίθενται στον χώρο https://github.com/dkotzaitsis/openMapProject.



Εικόνα 19: Η καρτέλα downloads.

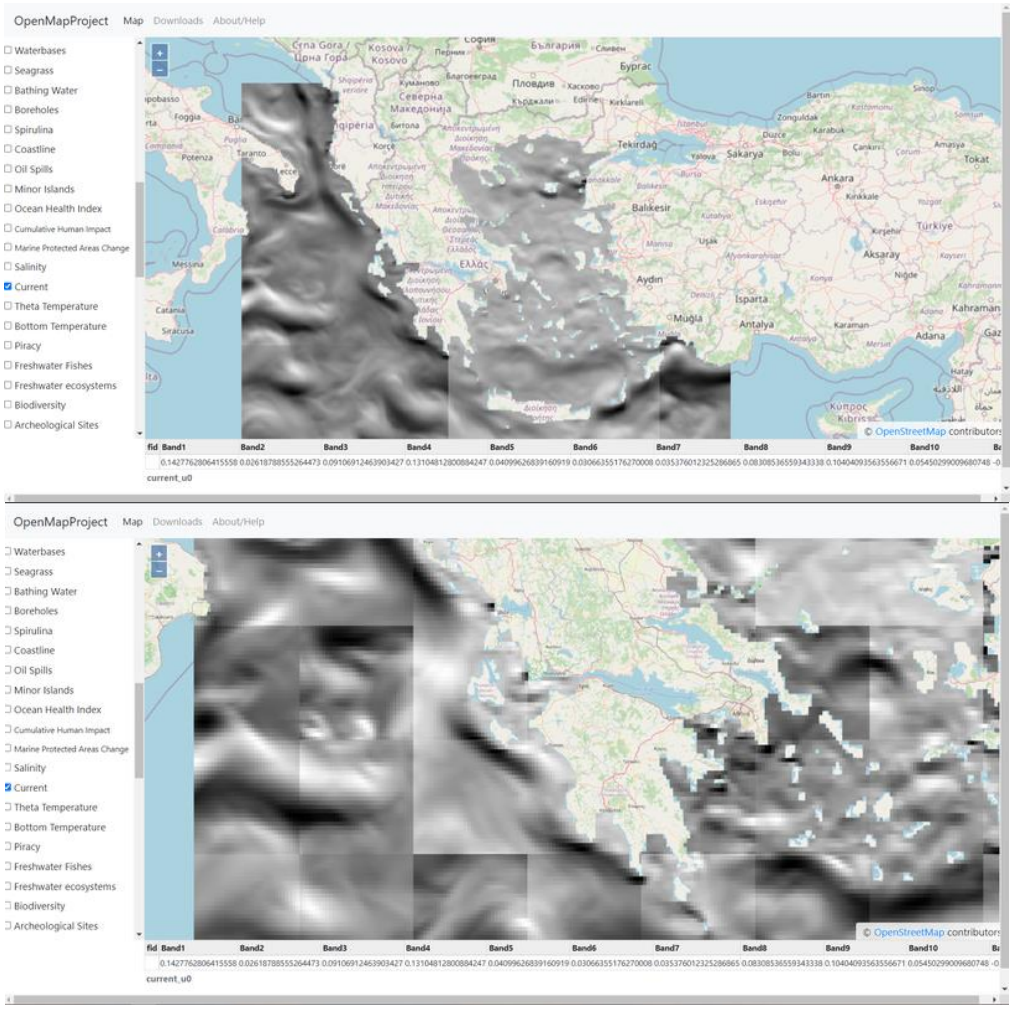


Εικόνα 20: Στην εικόνα αυτή βλέπουμε την επιλογή ενός vector στρώματος που είναι γεωμετρίας σημείου. Ο χρήστης, αφού ενεργοποίησε το επίπεδο, έχει πατήσει πάνω σε ένα σημείο και κάτω εμφανίζονται οι πληροφορίες για το σημείο αυτό. Επιπλέον, εμφανίζεται το pop-up παράθυρο με τις συντεταγμένες.

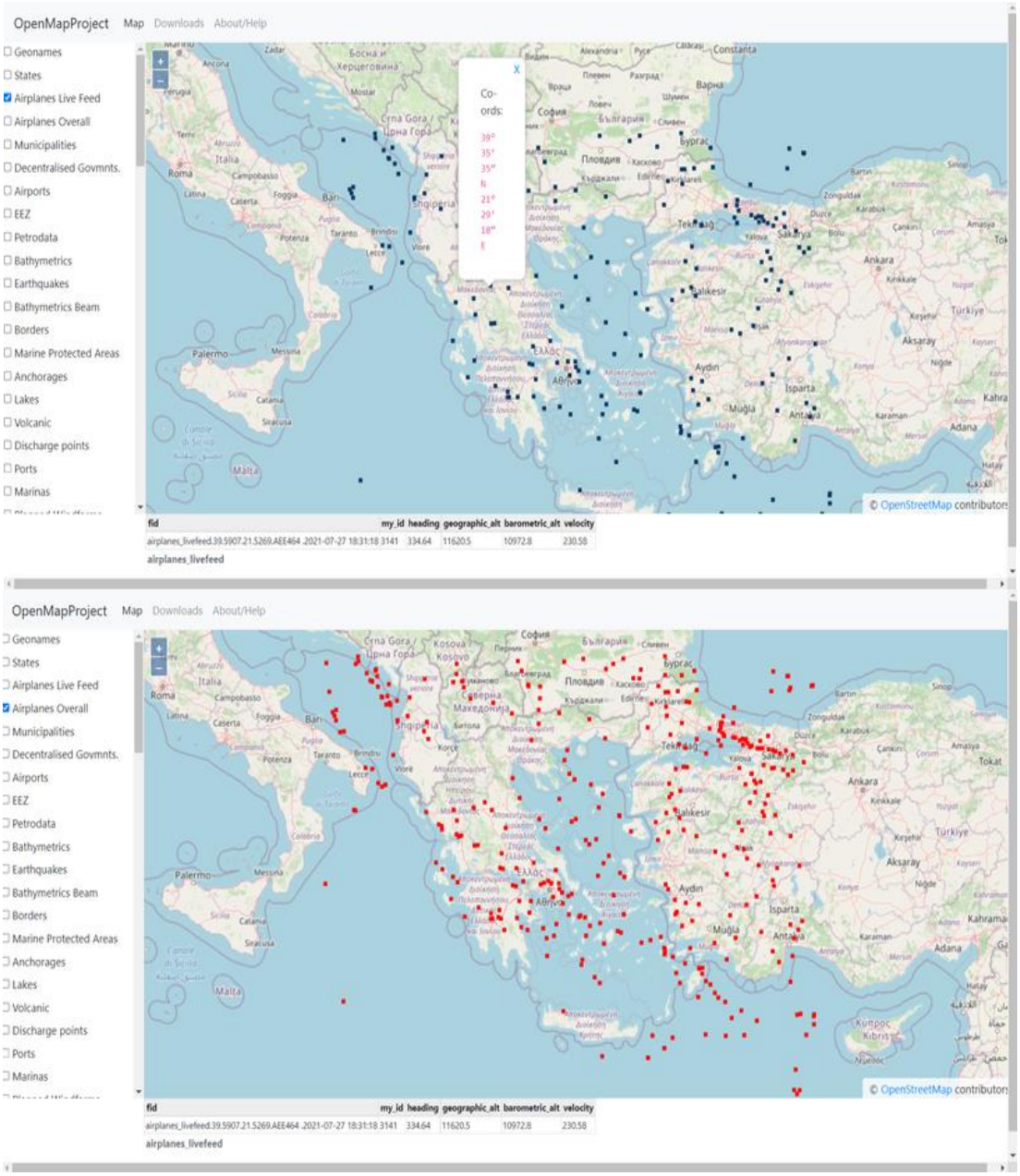


Εικόνα 21: Εδώ βλέπουμε την ενεργοποίηση ενός vector επιπέδου, γεωμετρίας πολυγώνου στην πρώτη εικόνα. Στην εικόνα επίσης, χρησιμοποιούμε την λειτουργία του bounding box. *Βλέπουμε το παραθυρικό περιβάλλον που εμφανίζει τις συντεταγμένες και ταυτόχρονα, επειδή είναι ενεργοποιημένο το στρώμα, γίνεται* download. *Αυτό φαίνεται στην δεύτερη και στην τρίτη εικόνα. Τέλος, βλέπουμε το layer στο QGIS.*

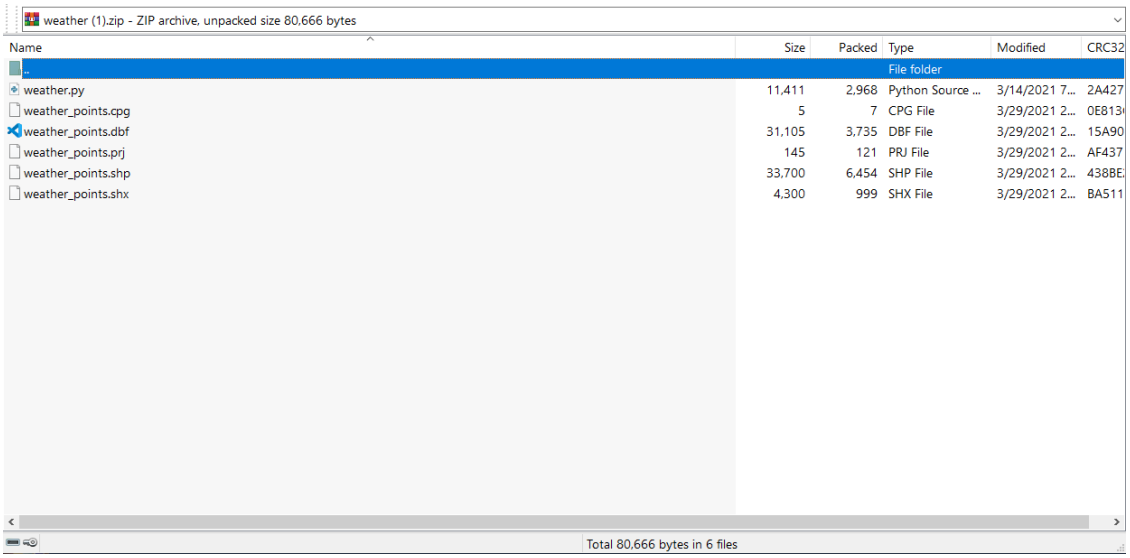
Εικόνα 22: Raster GeoTIFF αρχείο για τα βαθυμετρικά- υψομετρικά δεδομένα μέσω δορυφόρου. Με την κλήση των δεδομένων να φαίνεται κάτω και την ανάγνωση συντεταγμένων του σημείου που πιέσαμε.



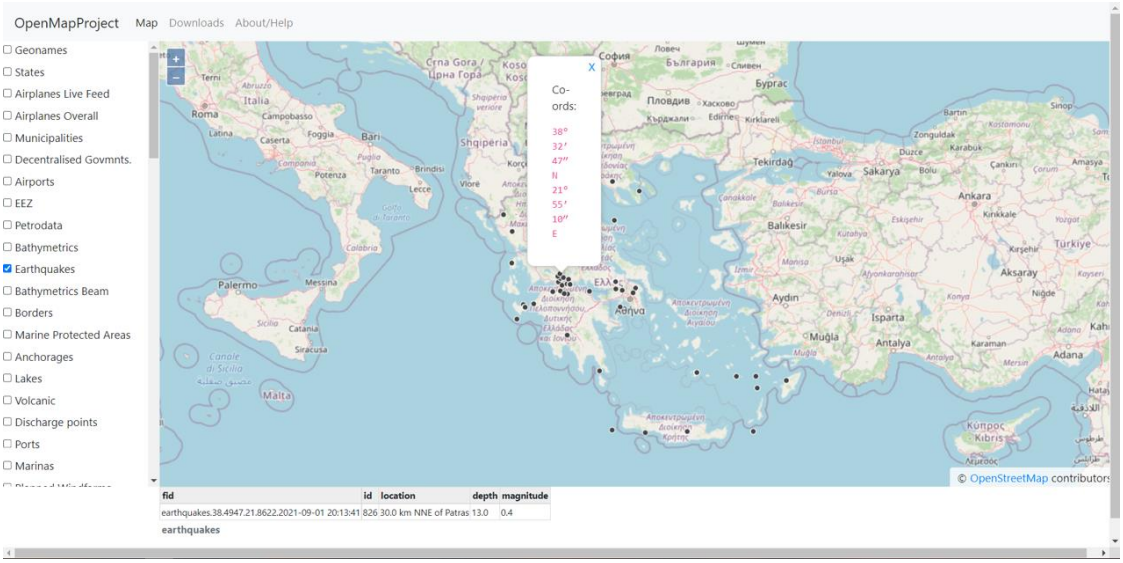
Εικόνα 23: Στις παραπάνω εικόνες βλέπουμε raster αρχεία, τύπου NetCDF. Οι μπάντες του αρχείου φαίνονται κάτω. Επιπλέον, στη δεύτερη εικόνα δείχνουμε τη σταδιακή φόρτωση των πλακιδίων που αναφέραμε προηγουμένως.



Εικόνα 24: Εδώ βλέπουμε τη λειτουργία των δυναμικών layers. Στην πρώτη εικόνα έχουμε για μία κλήση δεδομένων το live feed. Ενώ στη δεύτερη έχουμε τα συνολικά δεδομένα μετά από δύο κλήσεις του Open Sky API.



Εικόνα 25: Αποτέλεσμα download, χρησιμοποιώντας την καρτέλα Downloads. Στη συγκεκριμένη περίπτωση, ο χρήστης κατέβασε τα δεδομένα για τον καιρό. Επειδή τα δεδομένα από το Open Weather Map δεν είναι διαθέσιμα για αναδιανομή, δίνεται στον χρήστη η δυνατότητα να κατεβάσει το python source code μας. Επιπλέον, δίνεται στον χρήστη το αρχείο με τα σημεία αφοράς, ώστε να χρησιμοποιήσει το API του κώδικα.



Εικόνα 26: Εικόνα πρόσφατων σεισμών στην εφαρμογή μας, κατά την διάρκεια της συγγραφής.

8. ΣΥΝΟΨΗ, ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΒΕΛΤΙΩΣΕΙΣ

Συμπερασματικά, αφού αναλύσαμε το θεωρητικό επίπεδο της ανάπτυξης πληροφοριακών συστημάτων, αλλά και τον ειδικότερο χώρο της εφαρμογής της Γεωπληροφορικής στο θαλάσσιο περιβάλλον, ξεκινήσαμε την ανάπτυξη του δικού μας πληροφοριακού συστήματος μέσω εργαλείων και τεχνικών, ανοιχτού λογισμικού. Μέσω διαρκούς αναζήτησης στο διαδίκτυο, εξερευνήσαμε πηγές γεωχωρικών δεδομένων που μας επέτρεπαν την χρήση και αναδιανομή, ενώ απαιτήθηκε όχι μόνο η χρήση ήδη γνωστών τεχνικών για την επεξεργασία, απομόνωση και διαχείριση των δεδομένων στην βάση δεδομένων μας, αλλά και κατασκευή εξατομικευμένων τρόπων διαχείρισης της γεωχωρικής πληροφορίας.

Μελετήσαμε και αναλύσαμε, επιπλέον, γεωχωρικά δεδομένα -κυρίως θαλάσσιου χαρακτήρα- στατικά και δυναμικά, αφού ελέγξαμε γνωστά σύνολα δεδομένων από διάφορες επιστημονικές πηγές, και αποκτήσαμε γνώσεις για τεχνικές χαρτογράφησης και γενικότερα της παραγωγής συστημάτων γεωχωρικού χαρακτήρα.

Βέβαια, παρά το εύρος της μελέτης μας και την κατασκευή μιας σύνθετης εφαρμογής στην Επιστήμη Γεωχωρικών Δεδομένων, το παραγόμενο πληροφοριακό σύστημα επιδέχεται βελτιώσεων και τροποποιήσεων, ώστε να γίνει απόλυτα πλήρες και να μπορεί να αποτελέσει πραγματικά προϊόν που θα έχει χρήση πέρα από εκπαιδευτικούς και ερευνητικούς σκοπούς, για παράδειγμα ως εξής:

- Αρχικά, η χρήση ενός φυσικού server και η μετακίνηση όλης της υποδομής εκεί, θα επέτρεπε την μετατροπή της ερευνητικής αυτής εφαρμογής σε προϊόν παραγωγής, αφού, στα πλαίσια της εργασίας, δουλέψαμε τοπικά με τον υπολογιστή μας ως εξυπηρετητή (localhost). Αρκετά καλές λύσεις για χρήση cloud service εξυπηρετητών αποτελούν τα Amazon Web Services (AWS, https://aws.amazon.com/?nc2=h_lg), ενώ απαιτείται η αγορά ενός web domain για να φιλοξενήσει την εφαρμογή μας.
- Προσθήκη νέων λειτουργιών. Μελετώντας παρόμοιες εφαρμογές, εντοπίσαμε εξαιρετικές λειτουργίες. Αυτές, ναι μεν, στα πλαίσια της εργασίας μας ήταν «εκτός θέματος», παρ' όλα αυτά σε περίπτωση κυκλοφορίας της εφαρμογής

στην αγορά, θα αποτελούν ένα συγκριτικό πλεονέκτημα σε σχέση με άλλες εμπορικές εφαρμογές. Κάποιες από αυτές είναι: διαχειριστής ταξιδιού, επιπλέον βασικοί χάρτες και εμφάνιση ωρών δρομολογίων ανάλογα με τις προτιμήσεις του χρήστη κ.τ.λ.

- Προσθήκη επιπλέον δεδομένων. Για την παραγωγή της εφαρμογής μας, όπως αναφέραμε, αναζητήσαμε στο διαδίκτυο και σε άλλα σύνολα δεδομένων για συγκεκριμένου τύπου πληροφορία, που σχετίζεται με τα θαλάσσια γεωγραφικά δεδομένα. Μπορούμε να διευρύνουμε περαιτέρω τη λειτουργικότητα της εφαρμογής αν προσθέσουμε και άλλα δεδομένα από άλλες επιστήμες, τα οποία εμπεριέχουν γεωγραφική πληροφορία.
- Αγορά δεδομένων. Ένα από τα μεγάλα εμπόδια που είχαμε να αντιμετωπίσουμε, ήταν ότι σε πολλές περιπτώσεις τα δεδομένα είναι μη διαθέσιμα για αναδιανομή ή/και υπάρχουν περιορισμοί στη χρήση τους (π.χ. αριθμός κλήσεων στο API), αν δεν υπάρχει άδεια επί πληρωμή. Για να λυθεί αυτό το ζήτημα, θα μπορούσαμε σε μελλοντική ανάπτυξη της εφαρμογής να αγοράσουμε κάποιες άδειες χρήσης εφαρμογών.

Κλείνοντας, λοιπόν, αυτή την εργασία, μετά από μήνες προσπάθειας, μελέτης και προγραμματισμού, θα μπορούσαμε να πούμε ότι ο σκοπός της εργασίας επιτεύχθηκε σε μεγάλο βαθμό. Αναλύσαμε και μελετήσαμε τον τεράστιο χώρο των θαλάσσιων γεωχωρικών δεδομένων, επεξεργαστήκαμε μεγάλη ροή δεδομένων, διαφόρων τύπων και μορφών. Επιπλέον, διαχειριστήκαμε μια δική μας βάση δεδομένων, μαζί με όλα τα εργαλεία που μας βοήθησαν να χρησιμοποιήσουμε τα γεωγραφικά δεδομένα, αλλά και κατασκευάσαμε μια σύγχρονη εφαρμογή διαδικτύου, βασισμένοι σε σύγχρονες τεχνολογίες διαδικτυακού προγραμματισμού και client- server μοντέλων επικοινωνίας.

Έτσι, φτάσαμε σε ένα αρκετά υψηλό ερευνητικό επίπεδο, που μας καθιστά γνώστες των αντικειμένων της διαχείρισης και ανάλυσης γεωχωρικών δεδομένων μεγάλου όγκου, αλλά και της κατασκευής διαδικτυακών εφαρμογών χαρτογράφησης από την υποδομή - βάση δεδομένων και server- μέχρι και την εμφάνιση. Βέβαια, λόγω του εύρους και των δυνατοτήτων που παρέχονται στο συγκεκριμένο ερευνητικό χώρο, θα μπορούσε η εργασία αυτή να αποτελέσει το έναυσμα για την ενασχόληση και την περαιτέρω μελέτη από μελλοντικούς φοιτητές.

9. ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΩΝ

I. Κώδικας Δεδομένων Σεισμών

```
import pandas as pd
import lxml.html as lh
import requests
import  json,sys
import datetime
import schedule
from datetime import date,timedelta,timezone,datetime
from geopy.geocoders import Nominatim
import psycopg2
from psycopg2.extensions import AsIs
from psycopg2.extras import Json,DictCursor
import pytz



def dbOpen():
    try:
        connection=psycopg2.connect(host="localhost",
                                    database="openMap",
                                    user="your_username",
                                    password="your_password")
        cursor=connection.cursor()
    except (Exception, psycopg2.Error) as error:
        print("Error while fetching data from PostgreSQL", error)
    return connection,cursor

#closing db fun
def dbClose(connection,cursor):
    if connection:
        cursor.close()
        connection.close()
        print("PostgreSQL connection is closed")


def createCurrentTable(cursor,connection):
    commands=["""
        CREATE TABLE earthquakes_ovr (
            id serial,
            datetime TIMESTAMP UNIQUE,
            location VARCHAR,
            latitude double precision NOT NULL,
            longitude double precision NOT NULL,
            depth REAL,
            magnitude REAL,
            geom GEOMETRY,
            PRIMARY KEY (latitude,longitude,datetime)
        )
        """,
        """
        CREATE UNLOGGED TABLE earthquakes (
            id serial,
            datetime TIMESTAMP UNIQUE,
            location VARCHAR,
```

```
            latitude double precision NOT NULL,
            longitude double precision NOT NULL,
            depth REAL,
            magnitude REAL,
            geom GEOMETRY,
            PRIMARY KEY (latitude,longitude,datetime)
        )
        """]
    for command in commands:
        cursor.execute(command)
    connection.commit()



def check(connection,cursor):
    cursor.execute("select exists(select * from information_schema.tables where
table_name=%s)", ('earthquakes_ovr',))
    cursor.execute("select exists(select * from information_schema.tables where
table_name=%s)", ('earthquakes_ovr',))
    if cursor.fetchone()[0]==False:
        createCurrentTable(cursor,connection)
    else:
        cursor.execute("DROP TRIGGER IF EXISTS earthquakes_trigger ON earthquakes_ovr;")
        connection.commit()


def createDict(entry):

    vdict={}
    vdict['datetime']=datetime.strptime(entry['Origin Time(GMT)'],'%d/%m/%Y
%H:%M:%S').strftime('%Y/%m/%d %H:%M:%S')
    vdict['location']=entry['Epicentral Location']
    vdict['latitude']=entry['Latitude(°N)']
    vdict['longitude']=entry['Longitude(°E)']
    vdict['depth']=entry['Depth(km)']
    vdict['magnitude']=entry['Mag.']
    lat=(vdict['latitude'])
    lon=(vdict['longitude'])
    return vdict,lat,lon


def triggerFun(cursor,connection):
    name="earthquakes"
    commands=["""
        CREATE OR REPLACE FUNCTION earthquakes_trigger()
        RETURNS TRIGGER AS $$
        BEGIN
            INSERT INTO %s VALUES (NEW.*);
            DELETE FROM ONLY earthquakes WHERE datetime < (now() - '2 days'::interval);
            RETURN NEW;
        END;
        $$
        LANGUAGE plpgsql;
        """%(name),
        """CREATE TRIGGER earthquakes_trigger
                BEFORE INSERT ON earthquakes_ovr
                FOR EACH ROW EXECUTE PROCEDURE earthquakes_trigger();"""]
```

```
    for i in commands:
        cursor.execute(i)
    connection.commit()


def insertAndUpdate(vdict,lat,lon,connection):
    cur=connection.cursor(cursor_factory=DictCursor)
    insert = "insert into earthquakes_ovr (%s) values %s;"
    l = [(c, v) for c, v in vdict.items()]
    columns = ','.join([t[0] for t in l])
    values = tuple([t[1] for t in l])
    sql=cur.mogrify(insert, ([AsIs(columns)] + [values]))
    geom="update earthquakes_ovr SET geom=(ST_GeomFromText('POINT(%s %s)',4326))
WHERE (latitude=%s AND longitude=%s);" %(str(lon),str(lat),lat,lon)
    geom2="update earthquakes SET geom=(ST_GeomFromText('POINT(%s %s)',4326))
WHERE (latitude=%s AND longitude=%s);" %(str(lon),str(lat),lat,lon)
    try:
        cur.execute(sql)
        cur.execute(geom)
        cur.execute(geom2)
    except psycopg2.Error:
        connection.rollback()
        flag=0
        return flag
    else:
        connection.commit()
        flag=1
    cur.close()
    return flag


def scrapper(flag,mode):
    connection,cursor=dbOpen()
    check(connection,cursor)
    #inheritTable(connection,cursor)
    triggerFun(cursor,connection)
    if flag==0:
        dbClose(connection,cursor)
    else:
        URL = "http://www.geophysics.geol.uoa.gr/stations/maps/recent.html"
        tables = pd.read_html(URL,attrs={'class':'sortable'})
        table=tables[0]
        if mode==1:
            hourstime=int(input("How many hours back?\n"))
            past2=((datetime.now()-
timedelta(hours=hourstime)).astimezone(pytz.timezone('GMT'))).strftime('%d-%m-%Y
%H:%M:%S')
            for i in range(len(table)):
                entry=table.iloc[i]
                mydatetime=((datetime.strptime(entry['Origin Time(GMT)'],'%d/%m/%Y
%H:%M:%S')).astimezone(pytz.timezone('Europe/Athens'))).strftime('%d-%m-%Y
%H:%M:%S')
                if mydatetime>past2:
                    vdict,lat,lon=createDict(entry)
                    print(vdict)
                    fl=insertAndUpdate(vdict,lat,lon,connection)
                    if fl==0:
```

```
                dbClose(connection,cursor)
                break
        else:
            past1=((datetime.now()-
timedelta(hours=13)).astimezone(pytz.timezone('GMT'))).strftime('%d-%m-%Y %H:%M:%S')
            past2=((datetime.now()-
timedelta(hours=1)).astimezone(pytz.timezone('GMT'))).strftime('%d-%m-%Y %H:%M:%S')
            #for i in range(len(table)):
            for i in range(len(table)):
                entry=table.iloc[i]
                mydatetime=((datetime.strptime(entry['Origin Time(GMT)'],'%d/%m/%Y
%H:%M:%S')).astimezone(pytz.timezone('Europe/Athens'))).strftime('%d-%m-%Y
%H:%M:%S')
                if mydatetime>past1 and mydatetime<past2:
                    vdict,lat,lon=createDict(entry)
                    print(vdict)
                    fl=insertAndUpdate(vdict,lat,lon,connection)
                    if fl==0:
                        dbClose(connection,cursor)
                        break
        dbClose(connection,cursor)
        return 0



def main():
    try:
        mode=int(input("Give me your mode. If you want to collect old data press 1 else press any
key.\n"))
        if mode==1:
            scrapper(1,mode)
        schedule.every().day.at("12:00").do(scrapper,1,0)
        schedule.every().day.at("00:00").do(scrapper,1,0)

        #earthquakes(connection,cursor)
        while True:
            schedule.run_pending()
    except KeyboardInterrupt:
        print("Closing")
        scrapper(0,mode)
        sys.exit(0)

main()
```

II. Κώδικας δεδομένων για σήματα θέσης αεροσκαφών

```
import requests
import  json,sys
import datetime
import schedule
import time
from datetime import date,timedelta,timezone,datetime
from geopy.geocoders import Nominatim
import psycopg2
from psycopg2.extensions import AsIs
from psycopg2.extras import Json,DictCursor
import uuid
import pytz
from opensky_api import OpenSkyApi


ϊδια dbOpen() και dbClose() με earthquakes.py


def createCurrentTable(cursor,connection):
    commands=["""
        CREATE TABLE airplanes_ovr (
            my_id serial,
            callsign VARCHAR(30),
            heading REAL,
            geographic_alt REAL,
            barometric_alt REAL,
            datetime TIMESTAMP,
            longitude double precision NOT NULL,
            latitude double precision NOT NULL,
            velocity REAL,
            geom GEOMETRY,
            PRIMARY KEY (latitude,longitude,callsign,datetime)
        )
        """,
        """
        CREATE UNLOGGED TABLE airplanes_livefeed (
            my_id serial,
            callsign VARCHAR(30),
            heading REAL,
            geographic_alt REAL,
            barometric_alt REAL,
            datetime TIMESTAMP,
            longitude double precision NOT NULL,
            latitude double precision NOT NULL,
            velocity REAL,
            geom GEOMETRY,
            PRIMARY KEY (latitude,longitude,callsign,datetime)
        )
        """]
    for command in commands:
        cursor.execute(command)
    connection.commit()
```

```
def check(connection,cursor):
   cursor.execute("select exists(select * from information_schema.tables where
table_name=%s)", ('airplanes_ovr',))
   cursor.execute("select exists(select * from information_schema.tables where
table_name=%s)", ('airplanes_ovr',))
   if cursor.fetchone()[0]==False:
      createCurrentTable(cursor,connection)
   else:
      cursor.execute("DROP TRIGGER IF EXISTS airplanes_trigger ON airplanes_ovr;")
      connection.commit()


def triggerFun(cursor,connection):
   name="airplanes_livefeed"
   commands=["""
      CREATE OR REPLACE FUNCTION airplanes_trigger()
      RETURNS TRIGGER AS $$
      BEGIN
         INSERT INTO %s VALUES (NEW.*);
         DELETE FROM ONLY airplanes_livefeed WHERE datetime < (now() - '1
minutes'::interval);
         DELETE FROM ONLY airplanes_ovr WHERE datetime < (now() –‘1 hour’::interval);
         RETURN NEW;
      END;
      $$
      LANGUAGE plpgsql;
      """%(name),
      """CREATE TRIGGER airplanes_trigger
            BEFORE INSERT ON airplanes_ovr
            FOR EACH ROW EXECUTE PROCEDURE airplanes_trigger();"""]
   for i in commands:
      cursor.execute(i)
   connection.commit()


def createDict(s):
   vdict={}
   vdict['callsign']=s.callsign
   vdict['heading']=s.heading
   vdict['geographic_alt']=s.geo_altitude
   vdict['barometric_alt']=s.baro_altitude
   vdict['datetime']=datetime.now().strftime('%Y/%m/%d %H:%M:%S')
   vdict['longitude']=s.longitude
   vdict['latitude']=s.latitude
   vdict['velocity']=s.velocity
   lat,lon=vdict['latitude'],vdict['longitude']
   return vdict,lat,lon

def insertAndUpdate(vdict,lat,lon,connection):
   cur=connection.cursor(cursor_factory=DictCursor)
   insert = "insert into airplanes_ovr (%s) values %s;"
   l = [(c, v) for c, v in vdict.items()]
   columns = ','.join([t[0] for t in l])
   values = tuple([t[1] for t in l])
```

```
    sql=cur.mogrify(insert, ([AsIs(columns)] + [values]))
    geom="update airplanes_ovr SET geom=(ST_GeomFromText('POINT(%s %s)',4326))
WHERE (latitude=%s AND longitude=%s);" %(str(lon),str(lat),lat,lon)
    geom2="update airplanes_livefeed SET geom=(ST_GeomFromText('POINT(%s %s)',4326))
WHERE (latitude=%s AND longitude=%s);" %(str(lon),str(lat),lat,lon)
    try:
        cur.execute(sql)
        cur.execute(geom)
        cur.execute(geom2)
    except psycopg2.Error:
        connection.rollback()
        flag=0
        return flag
    else:
        connection.commit()
        flag=1
    cur.close()
    return flag

def getAPI(flag):
    connection,cursor=dbOpen()
    check(connection,cursor)
    triggerFun(cursor,connection)
    ########################################
    if flag==0:
        dbClose(connection,cursor)
        return 0
    else:
        api = OpenSkyApi(username="username",password="Pass")
            # bbox = (min latitude, max latitude, min longitude, max longitude)
        try:
            states = api.get_states(bbox=(33, 42.5, 17.2, 31.5))
            for s in states.states:
                vdict,lat,lon=createDict(s)
                print(vdict)
                flag=insertAndUpdate(vdict,lat,lon,connection)
                if flag==0:
                    dbClose(connection,cursor)
                    return 0
        except:
            print("Timeout exception occured\n")
            return 0
    dbClose(connection,cursor)
    return 0


def main():
    try:
        schedule.every(62).seconds.do(getAPI,1)
        while True:
            schedule.run_pending()
    except KeyboardInterrupt:
        print("Closing")
        getAPI(0)
        sys.exit(0)

main()
```

III. Κώδικας καιρικών δεδομένων

```
import requests
import  json,sys
import datetime
import schedule
import time
from datetime import date,timedelta,timezone,datetime
from geopy.geocoders import Nominatim
import psycopg2
from psycopg2.extensions import AsIs
from psycopg2.extras import Json,DictCursor
import uuid



#################### Postgres ########################
#TO SELECT ANYHING FROM JSON COLUMNS USE
# SELECT col.name->> 'json_key' AS my_column_name FROM weather_data

#WHERE col.name ->>'json_key'='whatever';
#########################################################
```

Ίδια dbopen, dbClose με earthquakes

```
#bringing the points table
def dbBring(cursor):
    latitude=[]
    longitude=[]
    postgreSQL_query="SELECT latitude,longitude FROM grc_weather_checks"
    cursor.execute(postgreSQL_query)
    records=cursor.fetchall()
    for column in records:
        latitude.append(column[0])
        longitude.append(column[1])

    return latitude,longitude

#if BBOX same as dbBring/
def dbBringBBOX(cursor,latmin,latmax,lonmin,lonmax):
    latitude=[]
    longitude=[]
    postgreSQL_query="SELECT latitude,longitude FROM grc_weather_checks WHERE
((latitude>=%f) AND (latitude<=%f) AND(longitude>=%f) AND (longitude<=%f));"
%(latmin,latmax,lonmin,lonmax)
    cursor.execute(postgreSQL_query)
    records=cursor.fetchall()
    for column in records:
        latitude.append(column[0])
        longitude.append(column[1])

    return latitude,longitude


#creating partional tables
def tableCreation(cursor,connection,flag):
```

```
    name="weather"+str(date.today().year)+"_"+str(date.today().month)
    if flag==1:
        commands=["""CREATE TABLE %s () INHERITS (weather_data);""" %(name),
                """CREATE INDEX %s_dt ON %s (dt);""" %(name,name)]
        for i in commands:
            cursor.execute(i)
        connection.commit()
    else:
        if date.today().day!=1:
            return
        else:
            commands=["""CREATE TABLE %s () INHERITS (weather_data);""" %(name),
                    """CREATE INDEX %s_dt ON %s (dt);""" %(name,name)]
            for i in commands:
                cursor.execute(i)
            connection.commit()

#creating main table
def createCurrentTable(cursor,connection):
    commands="""
        CREATE TABLE weather_data (
            lat double precision NOT NULL ,
            lon double precision NOT NULL ,
            dt TIMESTAMP,
            weather json,
            temp REAL,
            feels_like REAL,
            temp_min REAL,
            temp_max REAL,
            pressure SMALLINT,
            humidity SMALLINT,
            sea_level SMALLINT,
            grnd_level SMALLINT,
            visibility SMALLINT,
            wind json,
            clouds SMALLINT,
            sys json,
            geom GEOMETRY,
            PRIMARY KEY (lat,lon,dt)
        )
        """
    #for command in commands:
    cursor.execute(commands)
    connection.commit()

#insert data from dict and creating geometry col
def insertAndUpdate(lati,longi,vdict,connection):#,cur,connection):
    cur=connection.cursor(cursor_factory=DictCursor)
    insert = "insert into weather_data (%s) values %s;"
    l = [(c, v) for c, v in vdict.items()]
    columns = ','.join([t[0] for t in l])
    values = tuple([t[1] for t in l])
    sql=cur.mogrify(insert, ([AsIs(columns)] + [values]))
    geom="update weather_data SET geom=(ST_GeomFromText('POINT(%s %s)',4326))
WHERE (lat=%s AND lon=%s);" %(str(longi),str(lati),lati,longi)
    cur.execute(sql)
    cur.execute(geom)
```

```
    connection.commit()
    cur.close()

#data dictionary
def createDict(datacurrent):
    vdict={}
    lati=datacurrent['coord']['lat']
    longi=datacurrent['coord']['lon']
    vdict['lat']=lati
    vdict['lon']=longi
    vdict['dt']=datetime.fromtimestamp(datacurrent['dt'])
    vdict['weather']=Json(datacurrent['weather'][0])
    for i in datacurrent['main']:
        vdict[i]=datacurrent['main'][i]
    vdict['visibility']=datacurrent['visibility']
    vdict['wind']=Json(datacurrent['wind'])
    vdict['clouds']=datacurrent['clouds']['all']
    vdict['sys']=Json(datacurrent['sys'])
    return lati,longi,vdict



#creating postgres Trig Fun
def triggerFun(cursor,connection,flag):
    name="weather"+str(date.today().year)+"_"+str(date.today().month)
    if flag==1:
        commands=["""
            CREATE OR REPLACE FUNCTION weather_data_insert_trigger()
            RETURNS TRIGGER AS $$
            BEGIN
                IF ( NEW.dt >= TIMESTAMP '%s-%s-01 00:00:00' AND
                    NEW.dt < TIMESTAMP '%s-%s-01 00:00:00' ) THEN
                    INSERT INTO %s VALUES (NEW.*);
                ELSE
                    RAISE EXCEPTION 'Date out of range.  Fix the weather_data_insert_trigger()
function!';
                END IF;
                RETURN NULL;
            END;
            $$
            LANGUAGE plpgsql;

"""%(str(date.today().year),str(date.today().month),str(date.today().year),str(int(date.today().mo
nth)+1),name),
            """CREATE TRIGGER insert_weather_data_trigger
                    BEFORE INSERT ON weather_data
                    FOR EACH ROW EXECUTE PROCEDURE weather_data_insert_trigger();"""]
        for i in commands:
            cursor.execute(i)
        connection.commit()
    else:
        if date.today().day!=1:
            return
        else:
            if date.today().month!=12:
                commands=["""
                    CREATE OR REPLACE FUNCTION weather_data_insert_trigger()
```

```
                RETURNS TRIGGER AS $$
                BEGIN
                   IF ( NEW.dt >= TIMESTAMP '%s-%s-01 00:00:00' AND
                      NEW.dt < TIMESTAMP '%s-%s-01 00:00:00' ) THEN
                      INSERT INTO %s VALUES (NEW.*);
                   ELSE
                      RAISE EXCEPTION 'Date out of range.  Fix the weather_data_insert_trigger()
function!';
                   END IF;
                   RETURN NULL;
                END;
                $$
                LANGUAGE plpgsql;

"""%(str(date.today().year),str(date.today().month),str(int(date.today().year)+1),str(int(date.toda
y().month)+1),name)]
             for i in commands:
                cursor.execute(i)
             connection.commit()
             return
          else:
             commands=["""
                CREATE OR REPLACE FUNCTION weather_data_insert_trigger()
                RETURNS TRIGGER AS $$
                BEGIN
                   IF ( NEW.dt >= TIMESTAMP '%s-%s-01 00:00:00' AND
                      NEW.dt < TIMESTAMP '%s-%s-01 00:00:00' ) THEN
                      INSERT INTO %s VALUES (NEW.*);
                   ELSE
                      RAISE EXCEPTION 'Date out of range.  Fix the weather_data_insert_trigger()
function!';
                   END IF;
                   RETURN NULL;
                END;
                $$
                LANGUAGE plpgsql;

"""%(str(date.today().year),str(date.today().month),str(int(date.today().year)+1),str(1),name)]
             for i in commands:
                cursor.execute(i)
             connection.commit()



def current(lat,lon,api_key,connection):
  flag=0
  #cur=connection.cursor(cursor_factory=DictCursor)
  dt=time.time()
  t=time.time()
  for i in range(len(lat)):

current="https://api.openweathermap.org/data/2.5/weather?lat=%s&lon=%s&appid=%s&units=
metric"% (
            str(lat[i]), str(lon[i]), api_key)
     responsecurrent = requests.get(current)
     datacurrent = json.loads(responsecurrent.text)
     lati,longi,vdict=createDict(datacurrent)
```

```
      insertAndUpdate(lati,longi,vdict,connection)#cur,connection)
      flag=flag+1
      if flag==60:
        print("60")
        t1=time.time()
        dt=t1-t
        if dt>60:
           continue
        else:
           print('wait')
           t1=0
           time.sleep(60-dt)
           flag=0
           t=0
           t=time.time()
           print('start again')
      else:
         continue
  print('finished')
  #cur.close()




def main():
   try:
      connection,cursor=dbOpen()
      lat,lon=dbBring(cursor)
      ############### FOR BBOX DATA ########################
      #lat,lon=dbBringBBOX(cursor,37.0,40.4,18.501,22.430)
      ######################################################
      name="weather"+str(date.today().year)+"_"+str(date.today().month)



      cursor.execute("select exists(select * from information_schema.tables where
table_name=%s)", ('weather_data',))
      if cursor.fetchone()[0]==False:
         createCurrentTable(cursor,connection)
      else:
         cursor.execute("DROP TRIGGER IF EXISTS insert_weather_data_trigger ON
weather_data;")
         connection.commit()

      api_key = "API key"

      schedule.every().day.at("00:00").do(tableCreation,cursor,connection,2)
      schedule.every().day.at("00:00").do(triggerFun,cursor,connection,2)

      schedule.every().hour.at(":01").do(current,lat,lon,api_key,connection)
      schedule.every().hour.at(":31").do(current,lat,lon,api_key,connection)



      cursor.execute("select exists(select * from information_schema.tables where
table_name=%s)", (name,))
      if cursor.fetchone()[0]==False:
```

```
            tableCreation(cursor,connection,1)
        triggerFun(cursor,connection,1)


        if ((int(datetime.now().minute)>=1 and int(datetime.now().minute)<=20) or
(int(datetime.now().minute)>=31 and int(datetime.now().minute)<50)):
            current(lat,lon, api_key,connection)



        while True:
            schedule.run_pending()

    except KeyboardInterrupt:
        print("Closing")
        dbClose(connection,cursor)
        sys.exit(0)


main()
```

ΒΙΒΛΙΟΓΡΑΦΙΑ / ΠΗΓΕΣ

[1] P. A. Longley, M. F. Goodchild and D. W. R. David J. Maguire, Geographic Information Science And Systems, Wiley, 2015.

[2] M. Lind, M. Michaelides, R. Ward and R. T. Watson, Maritime Informatics, Springer, 2021.

[3] A. Artikis and D. Zissis, Guide to Maritime Informatics, Springer, 2021.

[4] J. Ferreira, C. Agostinho, R. Lopes, K. Chatzikokolakis, D. Zissis, M.-E. Vidal and et al., Maritime data technology landscape and value chain exploiting oceans of data for maritime applications. In Proceedings of the 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), 2017, p. 1113–1122.

[5] H. Wang, X. Zhuge, G. Strazdins, Z. Wei, G. Li and H. Zhang, Data integration and visualisation for demanding marine operations. In Proceedings of the MTS/IEEE OCEANS 2016 Conference, 2016, pp. 1-7.

[6] w3schools, «www.w3schools.com,» [Ηλεκτρονικό]. Available: https://www.w3schools.com/.

[7] R. Cyril, D. Richard, C. Elena, J. Anne-Laure and I. Clément, «Heterogeneous integrated dataset for Maritime Intelligence, surveillance, and reconnaissance,» *ELSEVIER*, 2019.

[8] C. Kalyvas, A. Kokkos and T. Tzouramanis, «A survey of official online sources of high-quality free-of-charge geospatial data for maritime geographic information systems applications,» *ELSEVIER*, 2016.

[9] T. Tzouramanis, «Navigating the ocean of publicly available maritime data», Chapter 2, In A.Artikis and D.Zissis, «Guide to Maritime Informatics», 2021, pp. 31-69.

[10] The GlobalFishingWatch 1. database. [Ηλεκτρονικό]. Available: https://github.com/GlobalFishingWatch/anchorages_pipeline/blob/master/pipe_anchorages/data/port_lists/geonames_1000.csv.

[11] Natural Earth, «Natural Earth,» [Ηλεκτρονικό]. Available: https://www.naturalearthdata.com/.

[12] OpenDataSoft, «OpenDataSoft,» [Ηλεκτρονικό]. Available: https://data.opendatasoft.com/explore/dataset/osm-world-airports@babel/export/.

[13] prio.org, «prio.org,» [Ηλεκτρονικό]. Available: https://www.prio.org/Data/Geographical-and-Resource-Datasets/Petroleum-Dataset/.

[14] The Nippon Foundation. [Ηλεκτρονικό]. Available: https://seabed2030.org/.

[15] P.-E. I. F. O. &. M. D. MANAGEMENT. [Ηλεκτρονικό]. Available: https://www.seadatanet.org/.

[16] USGS. [Ηλεκτρονικό]. Available: https://www.usgs.gov/centers/oki-water/science/bathymetric-surveys.

[17] Protected Planet. [Ηλεκτρονικό]. Available: https://www.protectedplanet.net/en.

[18] Global Fishing Watch. [Ηλεκτρονικό]. Available: https://globalfishingwatch.org/datasets-and-code-anchorages/.

[19] WWF, «Global Lakes and WetLands Database,» [Ηλεκτρονικό]. Available: https://www.worldwildlife.org/pages/global-lakes-and-wetlands-database.

[20] Smithsonian Institution. [Ηλεκτρονικό]. Available: https://volcano.si.edu/list_volcano_pleistocene.cfm.

[21] EMODnet Human Activities [Ηλεκτρονικό]. Available: https://www.emodnet-humanactivities.eu/view-data.php.

[22] The Humanitarian Data Exchange, «World Port Index,» [Ηλεκτρονικό]. Available: https://data.humdata.org/dataset/world-port-index.

[23] WikiPedia, «List of marinas - Greece,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/List_of_marinas#Greece.

[24] Wikipedia, «List of shipwrecks of Europe - Greece,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/List_of_shipwrecks_of_Europe#Greece.

[25] ARLHS LLC, «ARLHS World List of Lights,» [Ηλεκτρονικό]. Available: http://wlol.arlhs.com/index.php.

[26] A. Novikov, «Towards Data Science,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/creating-sea-routes-from-the-sea-of-ais-data-30bc68d8530e.

[27] European Environmental Agency. [Ηλεκτρονικό]. Available:

https://www.eea.europa.eu/data-and-maps/data/waterbase-water-quality-icm.

[28] Oceam Data Viewer. [Ηλεκτρονικό]. Available: https://data.unep-wcmc.org/datasets/7.

[29] Wikipedia, «List of oil spills,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/List_of_oil_spills.

[30] National Geospatial-Intelligence Agency. [Ηλεκτρονικό]. Available: https://msi.nga.mil/Piracy.

[31] Global Biodiversity Information Facility. [Ηλεκτρονικό]. Available: https://www.gbif.org/.

[32] Ancient World Mapping Center Search for:. [Ηλεκτρονικό]. Available: http://awmc.unc.edu/wordpress/.

[33] Global Environment Facility. [Ηλεκτρονικό]. Available: https://www.thegef.org/topics/large-marine-ecosystems.

[34] Ocean Health Index. [Ηλεκτρονικό]. Available: https://ohi-science.org/.

[35] WorldClim. [Ηλεκτρονικό]. Available: https://www.worldclim.org/data/bioclim.html.

[36] Copernicus Institute of European Union. [Ηλεκτρονικό]. Available: https://www.copernicus.eu/el.

[37] National Snow and Ice Data Center. [Ηλεκτρονικό]. Available: https://nsidc.org/.

[38] Γεωδυναμικό Ινστιτούτο - Πανεπιστήμιο Αθηνών. [Ηλεκτρονικό]. Available: http://www.geophysics.geol.uoa.gr/stations/maps/recent.html.

[39] OpenSky Network. [Ηλεκτρονικό]. Available: https://opensky-network.org/.

[40] Open Weather Map. [Ηλεκτρονικό]. Available: https://openweathermap.org/.

[41] shp-write, «GitHub,» [Ηλεκτρονικό]. Available: https://github.com/mapbox/shp-write.

[42] Apache Tomcat Documentation. [Ηλεκτρονικό]. Available: https://tomcat.apache.org/tomcat-7.0-doc/config/filter.html#CORS_Filter.

[43] M. J. d. Smith, M. F. Goodchild and P. A. Longley, «Geospatial Analysis: A Comprehensive Guide,» The Winchelsea Press, 2018.

[44] S. Davis, «GIS for Web Developers, Adding Where to Your Web Applications», The Pragmatic Programmers, 2007.

[45] PostgreSQL, «PostgreSQL,» [Ηλεκτρονικό]. Available: https://www.postgresql.org/about/.

[46] PostGIS, «PostGIS,» [Ηλεκτρονικό]. Available: https://postgis.net/.