ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΚΑΤΕΥΘΥΝΣΗ 1^{η} : Τεχνολογίες και Εφαρμογές Ιστού

Τίτλος Εργασίας

ΠΡΟΒΟΛΗ ΤΡΟΧΙΩΝ ΚΑΙ ΚΑΤΗΓΟΡΙΑΣ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ ΠΛΟΙΩΝ ΣΕ ΔΙΑΔΡΑΣΤΙΚΟΥΣ, ΔΙΑΔΙΚΤΥΑΚΟΥΣ ΧΑΡΤΕΣ

Διπλωματική Εργασία

Όνομα φοιτητή

ΧΑΡΑ ΜΠΟΥΛΟΥΓΑΡΗ

Αθήνα, Ιούλιος 2021

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΚΑΤΕΥΘΥΝΣΗ 1^{η} : Τεχνολογίες και Εφαρμογές Ιστού

Τριμελής Εξεταστική Επιτροπή

Τσερπές Κωνσταντίνος, Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής & Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Κουσιουρής Γεώργιος, Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής & Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Ξύδης Σωτήριος, Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής & Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Η Μπουλούγαρη Χαρά,

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία υλοποιήθηκε στα πλαίσια του μεταπτυχιακού προγράμματος σπουδών του τμήματος Πληροφορικής και Τηλεματικής στο Χαροκόπειο Πανεπιστήμιο το Νοέμβριο του 2020, αρχές Μαΐου και Ιουνίου του 2021.

Θα ήθελα να ευχαριστήσω όλους όσους με υποστήριξαν καθ' όλη τη διάρκεια αυτής της εργασίας και κατ' επέκταση και όλων των σπουδών μου. Πιο συγκεκριμένα θα ήθελα να ευχαριστήσω τον κ. Κωνσταντίνο Τσερπέ, επίκουρο καθηγητή στο Χαροκόπειο Πανεπιστήμιο, που υπήρξε ο επιβλέπων καθηγητής της παρούσας διπλωματικής για όλη την καθοδήγηση, βοήθεια που μου έδωσε με κάθε τρόπο και υποστήριξη που μου παρείχε και χάρης τον οποίο κατάφερα να μην το βάλω κάτω και να φτάσω έως αυτό το σημείο των σπουδών μου με υπομονή και μελέτη. Έτσι μου δόθηκε η ευκαιρία να γνωρίσω και να εξερευνήσω τεχνολογίες που ποτέ δεν μου πέρασε από το μυαλό πως θα κατάφερνα να διαχειριστώ! Επίσης τον ευχαριστώ και για το γεγονός ότι πίστεψε σε μένα όταν εγώ δεν πίστευα και στο ότι αναγνώρισε και εκτίμησε όλη μου την προσπάθεια που κατέβαλα ώστε να καταφέρω να φτάσω έως εδώ, που και πάλι ποτέ μου δεν φανταζόμουν πως θα πετύχαινα στο παρελθόν!

Επίσης, ευχαριστώ από την καρδιά μου τον παππού μου και τη γιαγιά μου για όλη την αγάπη και φροντίδα που μου έδωσαν ως παιδί και βοήθεια που ποτέ δεν μου στέρησαν σε όλα τα χρόνια σπουδών μου, μαθητικών και ακαδημαϊκών. Ελπίζω να με βλέπουν από εκεί ψηλά και να τους κάνω περήφανους κάθε στιγμή!

Τέλος θα ήθελα να επιβραβεύσω και να ευχαριστήσω και εμένα για την υπομονή που δείχνω σε αυτόν τον δύσκολο αγώνα της κατάκτησης της γνώσης όλα αυτά τα χρόνια των σχολικών αλλά και ακαδημαϊκών μου σπουδών !

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ

Αντικείμενο της παρούσας εργασίας είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής διαδραστικών χαρτών με σκοπό την αναπαράσταση των τροχιών πλοίων καθώς και της δραστηριότητας πλοήγησης την οποία έχουν σε κάθε θέση τους πάνω στον χάρτη ώστε να γίνει εφικτή η μετέπειτα επεξεργασία των δεδομένων θέσεών τους και να παραχθούν οι υπηρεσίες που διατίθενται στον πελάτη μετά την επεξεργασία αυτή. Η υλοποίηση της εφαρμογής είχε δύο στάδια τα οποία έγιναν με χρήση των τεχνολογιών React *(μία javascript βιβλιοθήκη ιδιαίτερα αποδοτική και εύκολη για την ανάπτυξη πολύπλοκων διαδραστικών διεπαφών που συνθέτονται από μικρά και απομονωμένα κομμάτια τα αποκαλούμενα components, μόνο στην έννοια του View, που επιτρέπει τη διαχείριση της κατάστασης των μεταβλητών σε συνεργασία ακόμα και με άλλες βιβλιοθήκες μονοδρομικά.)*, επίσης συνδυάστηκε σε συνεργασία με τη βιβλιοθήκη χαρτών leaflet για το πρώτο στάδιο του frontend και python *(μια γλώσσα κατάλληλη για επεξεργασία ολόκληρων datasets με χρήση ειδικών βιβλιοθηκών για το σκοπό αυτό, όπως η pandas και η numpy)* για το στάδιο της επεξεργασίας του dataset με τις ιστορικές θέσεις των πλοίων και τον υπολογισμό ειδικής στήλης ετικετών πλοήγησης. Επίσης, χρησιμοποιήθηκε και ένα backend το οποίο ανοίγει ένα websocket για την παροχή μιας συνεχούς, αμφίδρομης και ταυτόχρονης επικοινωνίας και τροφοδότησης της εφαρμογής μας με δεδομένα που προέρχονται από την ανάγνωση του αρχείου με τις ιστορικές αυτές θέσεις των πλοίων

Η υλοποίηση μιας τέτοιας εφαρμογής στοχεύει στην απεικόνιση και γραφική αναπαράσταση των παρακολουθούμενων πλοίων για την επεξεργασία και ανάλυση των θέσεων στις οποίες αυτά βρίσκονται με αποτέλεσμα την παραγωγή γνώσης και προβλέψεων μελλοντικών καταστάσεων σχετικά με την κίνηση των πλοίων αυτών προς τον τελικό προορισμό τους, καθώς και πολλή ακόμα γνώση σε περίπτωση επέκτασής της στο προσεχές μέλλον που θα μπορούσε να γίνει διαθέσιμη ως υπηρεσία για το ευρύ κοινό. Πιο αναλυτικά, η εφαρμογή δέχεται ένα dataset μορφής csv από το οποίο λαμβάνει πληροφορίες για κάποιες ιστορικές θέσεις διαφόρων πλοίων και τις στέλνει για γραφική αναπαράσταση στη διεπαφή της εφαρμογής όπου και φορτώνει πάντα την τελευταία πιο πρόσφατη θέση του κάθε πλοίου κάνοντας μη εμφανή την προηγούμενη. Αν ο χρήστης θελήσει να πατήσει πάνω σε κάποια θέση κάποιου πλοίου που υπάρχει ήδη στον χάρτη τότε ενεργοποιείται ολόκληρο το δρομολόγιο που ακολούθησε το πλοίο αυτό και εμφανίζεται με χρώμα κάθε κομμάτι στο οποίο ανήκουν ίδιας κατηγορίας δραστηριότητας θέσεις καθώς και κατά το πέρασμα του δείκτη του ποντικιού από πάνω από αυτά τα κομμάτια. Επίσης με το που ανοίξει το δρομολόγιο στα δεξιά κρατείται σε μια καρτέλα η τελευταία θέση του πλοίου η οποία έχει διαβαστεί μέχρι στιγμής από το dataset και φαινόταν ως κουκκίδα πάνω στον χάρτη πριν την πατήσουμε. Τέλος ο χρήστης μπορεί να διαγράψει τα ενεργοποιημένα δρομολόγια από τον χάρτη ή και να αλλάξει το χρώμα θέματος όλης της εφαρμογής.

Λέξεις Κλειδιά: React Interactive Web Maps, Leaflet, Ships' Latest Positions, View Full Ship's Trajectory, Historic vessels' positions

ABSTRACT

The subject of this dissertation is the development of a web application based on interactive maps with the aim of visualizing the trajectories of ships and the navigation activity that they have on each position that they are upon the map as well for the one purpose of analyzing all this positional data so at the end all the related services can be produced and then be provided to the customer afterwards. The development of the application had two stages that were accomplished by the use of technologies like React *(a javascript library which is extremely effective and flexible for the development of complex interactive user interfaces that are composed from small and isolated pieces of code called components, only under the concept of the View, allowing the management of the state with other libraries working with one-way data binding to achieve a unidirectional data flow)*, furthermore React was combined with another library for the part of the interactive maps called Leaflet these two make the perfect match together for the first stage of the frontend development and then python *(an appropriate language for the manipulation of whole datasets with the use of expertised libraries for this purpose, such as pandas and numpy)* for the stage of the historical positions of the ships (our dataset) manipulation and the calculation of a specific column which stores the navigation labels inside. There was also the need of creating a backend which opens a websocket connection for the supply of a persistent bidirectional and full-duplex communication with the frontend of the application serving the application with the necessary data that comes from the reading of the file with our dataset of all this historical data of the ships.

The development of such an application focuses on the visualization of the tracked ships for the purpose of analyzing their positions at which they are navigating from as said before but with the aim of producing knowledge and other predictions of their future path and movements till their final destination in case this application gets expanded someday in order to be available to customers. In detail, this application takes a dataset of a csv format from which it receives information for some historical positions of several different ships and it sends it to the frontend to be depicted on the map where it loads only the latest received position of each ship making invisible the previous one. If the user wants to click upon a point, representing a position of a ship that already exists on the map, then it reveals the whole trajectory that this ship followed and it colors every segment in which some of the trajectory points belong and are labeled under the same navigation category not only when the trajectory opens up but also when the user just hovers upon the segment while the trajectory is open. Furthermore, when the trajectory gets opened the latest position that has been read, up to the moment, is getting stored in the first right panel. Finally, the user can also remove from the map the opened trajectories if he wishes to or even change the whole theme color of the application.

Keywords: React Interactive Web Maps, Leaflet, Ships' Latest Positions, View Full Ship's Trajectory, Historic vessels' positions

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΑΠΟΣΠΑΣΜΑΤΩΝ ΚΩΔΙΚΑ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

SPA	Single Page Application
MVC	Model View Controller
HTTP	Hypertext Transfer Protocol
APP	Application
CLI	Command Line Interface
NPM	Node Package Manager
DOM	Document Object Model
HTML5	HyperText Markup Language
UML	Unified Modeling Language
URL	Universal Resource Locator (aka Uniform Resource Locator)
URI	Uniform Resource Identifier
TCP / IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
WS	WebSocket
WSS	WebSocketSecure
API	Application Programming Interface
OBJ	Object
AIS	Automatic Identification System
RAM	Random Access Memory
GB	GigaByte
CPU	Central Processing Unit
I/O	Input/Output
VMS	Vessel Monitoring Systems
.csv	Comma-separated values Microsoft excel file
JSX	JavaScript XML
GeoJSON	Geographical Javascript Object Notation
JSON	JavaScript Object Notation
SEO	search engine optimization
UI	User interface
GPS	Global Positioning System

1 ΕΙΣΑΓΩΓΗ

1.1 Η σημασία της ενασχόλησης με την παρακολούθηση πλοίων

Η ραγδαία ανάπτυξη της τεχνολογίας έχει σημαντικά βελτιώσει τις μεθόδους παρακολούθησης και ελέγχου των πλοίων τα τελευταία χρόνια. Οι γρήγοροι ρυθμοί βελτίωσης και κατασκευής δορυφορικών υπηρεσιών ολοένα και αναβαθμίζουν την παρακολούθηση πλοίων σε όλο τον κόσμο διασφαλίζοντας την ασφάλεια των πλοίων από διάφορες απειλές.

Με τη βοήθεια συστημάτων βασισμένα σε δορυφορικές υπηρεσίες VMS πολλά πλοία διαφορετικών τύπων ελέγχονται ακόμα και από τους ιδίους τους ιδιοκτήτες τους εδώ και πάρα πολύ καιρό αλλά και με τη βοήθεια πλέον ειδικών πομπών με τους οποίους εξοπλίζονται τα πλοία, σήματα στέλνονται προς τους δορυφόρους ώστε να επιτευχθεί η συλλογή δεδομένων όπως η θέση, το δρομολόγιο, η ταχύτητα, ο τύπος πλοίου και άλλα για την μετέπειτα παροχή τους σε διάφορους φορείς εξειδικευμένους στην ανάλυση των δεδομένων αυτών ώστε να επιτευχθεί ο στόχος της προστασίας των πλοίων μέσω της μελέτης των δεδομένων για αυτά. Οι φορείς αυτοί συγκεκριμένα χρησιμοποιούν το σύστημα AIS, (Kaushik, 2021) το οποίο επωφελείται τις δυνατότητες του GPS (Raunek, 2021) κι έτσι καταφέρνουν να παρέχουν την πληροφορία που παράγουν σε πραγματικό χρόνο απεικονίζοντας κάθε πλοίο πάνω σε διαδραστικούς χάρτες πριν την διάθεση της τελικής πληροφορίας στον ενδιαφερόμενο πελάτη.

Η ανάλυση αυτών των δεδομένων των πλοίων μπορεί να φανεί χρήσιμη πέρα από την εξασφάλιση της ασφάλειας των πλοίων και στο γεγονός οι πελάτες τέτοιων υπηρεσιών να μπορούν να ελέγχουν και πιο γενικά το θαλάσσιο χώρο γύρω από τον οποίο κινείται το πλοίο τους, όπως δηλαδή να μπορούν να βρίσκουν τους κοντινότερους φάρους, λιμάνια ή άλλα πλοία που πλησιάζουν, μπορούν να υπολογίζουν τον χρόνο άφιξης σε ένα λιμάνι ή τις μελλοντικές θέσεις στις οποίες θα βρίσκονται στο μέλλον του δρομολογίου τους, ή να έχουν πληροφορίες σχετικά με τις καιρικές συνθήκες στην περιοχή όπου βρίσκονται, ή να βλέπουν τη δραστηριότητα του πλοίου τους στο σημείο της τοποθεσίας τους, ή ακόμα και να έχουν πρόσβαση σε διάφορα πιθανολογικά ή και στατιστικά μοντέλα που υπολογίζονται με βάση τα δεδομένα των θέσεών τους για ανάλυση της κίνησης στη θάλασσα. (Kaushik, 2021) Όλα αυτά εξυπηρετούν στο να προσανατολιστούν καλύτερα και με μεγαλύτερη ευκολία στο δρομολόγιο που ακολουθούν ή θα αποφασίσουν να ακολουθήσουν. Επίσης, η ανάλυση των δεδομένων αυτή προσφέρει την βοήθεια στην αποφυγή συγκρούσεων πλοίων και στην ενημέρωση

των ακτοφυλάκων στον να εντοπίσουν πιο πλοίο παραβαίνει τους νόμους θαλάσσης ή παρεμβαίνει σε λάθος δρομολόγιο. (Raunek, 2021)

Παρόλο που πολλοί φορείς υπόσχονται παρακολούθηση πλοίων σε πραγματικό χρόνο στην πραγματικότητα ο πραγματικός χρόνος διαφέρει αρκετά από αυτόν που απεικονίζεται στους διαδραστικούς τους χάρτες κατά αρκετά λεπτά ή ακόμα και ώρες οπότε σίγουρα δεν θα πρέπει να παρθούν αυτές οι πληροφορίες που διατίθενται για χρήση πραγματικής πλοήγησης. (Kaushik, 2021)

1.2 Αντικείμενο διπλωματικής

Αντικείμενο της παρούσας διπλωματικής είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής διαδραστικών χαρτών με σκοπό την αναπαράσταση των τροχιών πλοίων καθώς και της δραστηριότητας πλοήγησης την οποία έχουν σε κάθε θέση τους πάνω στον χάρτη ώστε να γίνει εφικτή η μετέπειτα επεξεργασία των δεδομένων θέσεών τους και να παραχθούν οι υπηρεσίες που διατίθενται στον πελάτη μετά την επεξεργασία αυτή.

1.2.1 Ορισμός προβλήματος

Καθώς η τεχνολογία μπορεί να γίνει αρωγός των ανθρώπων σχεδόν στα πάντα στην καθημερινότητά τους από τα πιο απλά έως τα πιο πολύπλοκα προβλήματα που έχουν να αντιμετωπίσουν έτσι λοιπόν και στο θαλάσσιο χώρο όπου οι μετακινήσεις είτε ανθρώπων είτε φορτίων γίνονται με πλοία εμφανίστηκε η ανάγκη τεχνικών που θα μπορούσαν να εξυπηρετήσουν τις μετακινήσεις των πλοίων με ασφάλεια προς τον προορισμό τους, η ανάγκη για εντοπισμό των πλοίων ανά πάσα ώρα και στιγμή, η ανάγκη παρακολούθησης των κοντινότερων πλοίων στην περιοχή και η ανάγκη ανάλυσης των δεδομένων δραστηριότητας πλοήγησης στην κάθε θέση που έχει το κάθε πλοίο και για πόσες συνεχόμενες θέσεις βρίσκεται κάτω από την ίδια δραστηριότητα, καθώς και ποια είναι η τελευταία ληφθείσα θέση μέχρι στιγμής στην οποία βρίσκεται.

1.2.2 Τρόπος επίλυσης προβλήματος

Το πρόβλημα που προαναφέρθηκε προσεγγίστηκε με την υλοποίηση μιας εφαρμογής η οποία σκοπό έχει να απεικονίσει πάνω σε διαδραστικούς διαδικτυακούς χάρτες τις θέσεις πλοίων που κατέχουν το σύστημα AIS το οποίο χρησιμοποιούν για να αποστείλουν τη θέση τους και άλλες πληροφορίες μαζί. Η συγκεκριμένη εφαρμογή χρησιμοποιεί ιστορικές τέτοιες θέσεις που προσεγγίζουν την αναπαράστασή τους πάνω στο χάρτη σε ψευδο-πραγματικό χρόνο και η ουσία είναι μετά την αναπαράσταση αυτή να γίνουν παρατηρήσεις και συμπεράσματα προσεγγιστικά για την ιστορική κίνηση των πλοίων, ώστε να λυθεί το πρόβλημα του εντοπισμού κίνησης των πλοίων στην γύρω περιοχή αναζήτησης. Επίσης έχει γίνει χρωματική αναπαράσταση των ιδίων δραστηριοτήτων πλοήγησης που βρίσκονται σε

συνεχόμενη εμφάνιση για πιο εύκολο εντοπισμό τους πάνω στο δρομολόγιο ενός επιλεγμένου πλοίου, ως λύση στο πρόβλημα των δραστηριοτήτων. Έτσι ο τελικός χρήστης μπορεί να έχει εικόνα και για τα πλοία που βρίσκονται στην περιοχή που εξετάζει και συγκεκριμένα για τη δραστηριότητα του κάθε πλοίου σε κάθε μία από τις θέσεις του ώστε να μπορέσει να βγάλει τα συμπεράσματά του για την κίνηση των πλοίων.

Στο χρήστη επίσης προσφέρεται ενημέρωση για την ώρα που το πλοίο περνάει από κάποια θέση το γεωγραφικό μήκος και πλάτος τον τύπο του πλοίου και άλλα σχετικά με το πλοίο δεδομένα. Τέλος, στο χρήστη εμφανίζεται με την είσοδό του στην εφαρμογή μόνο η πιο τελευταία θέση του πλοίου πάνω στον χάρτη με ένα αναδυόμενο μικρό παραθυράκι που αναφέρει το όνομα του πλοίου καθώς ο χρήστης περνά το δείκτη του ποντικιού από πάνω. Για την προσωπική διευκόλυνση του χρήστη κρατείται η τελευταία αυτή θέση που έχει ληφθεί μόνιμα σε ειδική καρτέλα εντός της εφαρμογής, έτσι ανά πάσα ώρα και στιγμή του δίνεται η δυνατότητα να κάνει επί τόπου σύγκριση των δεδομένων της τρέχουσας θέσεις που έχει επιλέξει να μελετήσει με την τελευταία που έχει ληφθεί από τη συλλογή ιστορικών δεδομένων που έχουμε.

1.2.3 Σύντομη περιγραφή λογικής της υλοποίησης

Η λογική που ακολουθήθηκε για την υλοποίηση και ανάπτυξη της εφαρμογής αποδίδεται με τα παρακάτω βήματα.

- Αρχικά υλοποιήθηκε ο χάρτης ο οποίος θα δέχεται τα δεδομένα ώστε να γίνει η αναπαράστασή τους σε ψευδο-πραγματικό χρόνο καθώς αυτά είναι ιστορικά!
- Μετά υλοποιήθηκε επιλογή αλλαγής χρώματος του θέματος της εφαρμογής ώστε να υπάρξει μια επιπλέον αλληλεπίδραση και εξατομίκευση της εφαρμογής με τον χρήστη και προς τον χρήστη.
- Στη συνέχεια προστέθηκαν καρτέλες στα δεξιά της εφαρμογής οι οποίες κρατούν δεδομένα που επιλέγονται από τις διάφορες αλληλεπιδράσεις του χρήστη με τον χάρτη.
- Και τέλος η βασική λογική πέρα από το εμφανισιακό κομμάτι που περιγράφηκε στα προηγούμενα βήματα είναι ότι έρχονται τα δεδομένα με τις θέσεις διαφόρων πλοίων από μια πηγή στο backend της εφαρμογής η οποία στόχο έχει απλώς να διαβάζει αυτά τα δεδομένα από το αρχείο με τη συλλογή αυτών κι έπειτα να τα στέλνει στον χάρτη στο frontend όπου και απεικονίζονται με μια κουκκίδα η οποία παίρνει χρώμα αν ο χρήστης περάσει από πάνω της εμφανίζοντας το όνομα του πλοίου και σε περίπτωση που η θέση ενημερωθεί από το

backend, δηλαδή έρθει μια πιο καινούρια θέση η προηγούμενη εξαφανίζεται και αποτυπώνεται εκ νέου η καινούρια. Αν ο χρήστης επιλέξει να πατήσει μια κουκκίδα ένα μονοπάτι/δρομολόγιο με θέσεις απ' τις οποίες πέρασε το πλοίο εμφανίζονται και ένα ενημερωτικό αναδυόμενο παράθυρο με πληροφορίες για την επιλεγμένη θέση ανοίγει.

- Έχει γίνει υπολογισμός των κατηγοριών δραστηριότητας στις διάφορες αυτές θέσεις και στην εφαρμογή έχει γίνει ομαδοποίηση ανάλογα με τις συνεχόμενες κοινές αυτές ετικέτες που έχουν προστεθεί.
- Συνοψίζοντας η λογική είναι να μπορεί ο χρήστης να αλληλεπιδράσει πλήρως με τον χάρτη ο οποίος λαμβάνει τα ιστορικά δεδομένα του σε ψευδο-πραγματικο χρόνο από το backend και τα φορτώνει με χρωματικούς κανόνες ομαδοποιώντας τα σε δρομολόγια ανά πλοίο και κατ' επέκταση έχουμε άλλη μια ομαδοποίηση εκείνη των ετικετών δραστηριότητας ανά συνεχόμενες κοινές/ίδιες ετικέτες.

1.2.4 Στόχοι υλοποίησης και στόχοι διπλωματικής

Η υλοποίηση μιας τέτοιας εφαρμογής στοχεύει στην απεικόνιση και γραφική αναπαράσταση των παρακολουθούμενων πλοίων για την επεξεργασία και ανάλυση των θέσεων στις οποίες αυτά βρίσκονται με αποτέλεσμα την παραγωγή γνώσης σχετικά με την κίνηση των πλοίων αυτών εντός του θαλάσσιου χώρου στον οποίο κινούνται.

Στόχος της υλοποίησης είναι η απλή, οικεία, εύκολη και εύχρηστη εμφάνιση της τελικής εφαρμογής η οποία εμφάνιση θα είναι και προσαρμόσιμη στα προσωπικά γούστα του χρήστη αλλά και καθαρή ώστε να μην προκαλεί σύγχυση στο χρήστη ο οποίος στόχο έχει την εύκολη διαχείρισή της και αλληλεπίδραση με αυτή ώστε να μπορέσει να κάνει παρατηρήσεις πάνω στα δεδομένα του χάρτη και για να επιτευχθεί κάτι τέτοιο θα πρέπει η εμφάνιση της εφαρμογής να επιτρέπει την άμεση εξοικείωση του χρήστη με αυτήν και αυτό είναι ένα πολύ σημαντικό στοιχείο που λαμβάνεται υπόψη κατά την υλοποίηση της εφαρμογής αυτής. Επίσης, στόχος της υλοποίησης είναι και το πρακτικό κομμάτι δηλαδή η παροχή σωστών δεδομένων και τεχνικών για χρωματικούς σχεδιασμούς για την πιο εύκολη ανάγνωση των δεδομένων πάνω στο χάρτη.

Στόχος της όλης διπλωματικής είναι η παραγωγή μιας εφαρμογής που θα βοηθήσει στην παρακολούθηση ιστορικών θέσεων διαφόρων πλοίων για καταφυγή σε διάφορα συμπεράσματα που θα παραχθούν από αυτήν την μελέτη των κινήσεων και δραστηριοτήτων τους ανά θέση κίνησης που είχαν

αυτά τα πλοία σε παρελθοντικούς χρόνους για την μετέπειτα επεξεργασία αυτών των παρατηρήσεων ώστε να βγουν στο μέλλον σε κάποια προσπάθεια επέκτασης της παρούσας εφαρμογής πιθανολογικά μοντέλα ή και στατιστικά μοντέλα μελλοντικής πιθανής κίνησης και δρομολογίων για αποφυγή συγκρούσεων ή και διαφόρων άλλων απειλών και παγίδων που κρύβονται στη θάλασσα.

1.3 Οργάνωση κειμένου

Η συνέχεια της παρούσας διπλωματικής εργασίας ολοκληρώνεται με βάση την ακόλουθη δομή.

Στο κεφάλαιο 2 αναλύονται οι απαιτήσεις λειτουργικές και μη λειτουργικές που απαιτούνται να ακολουθούνται στην εφαρμογή, καθώς γίνεται αναφορά και στην αρχιτεκτονική της εφαρμογής και στα στυλ αλληλεπίδρασης που υπάρχουν με τον χρήστη από τη μεριά του frontend.

Στο κεφάλαιο 3 παρουσιάζεται η ανασκόπηση της τρέχουσας κατάστασης της εφαρμογής ή αλλιώς το γνωστό σε όλους State Of The Art και αναφέρονται επίσης και άλλες εφαρμογές που ήδη υπάρχουν στην αγορά. Επίσης, κάνουμε μια γνωριμία με τις τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ολοκλήρωση και τελειοποίηση της παρούσας εργασίας και επισημαίνουμε τους λόγους για τους οποίους επιλέχθηκαν αυτές οι τεχνολογίες και όχι κάποιες άλλες με παρόμοιες δυνατότητες.

Στο κεφάλαιο 4 δίνουμε έμφαση στις λεπτομέρειες της υλοποίησης και δείχνουμε εν συντομία την εγκατάσταση των τεχνολογιών που απαιτήθηκαν και του περιβάλλοντος που δουλέψαμε και αναφέρουμε χρήσιμα σημεία του κώδικα όπως αλγορίθμους μεθόδους συναρτήσεις παραθέτοντας και σχετικές εικόνες (snippets) που βοηθούν στην επεξήγηση του κειμένου.

Στο κεφάλαιο 5 γίνεται μια σύντομη αξιολόγηση της εφαρμογής, με σενάρια εκτέλεσης που τρέξαμε ώστε να εντοπιστούν πιθανά λάθη ή και αν ικανοποιούνται πλήρως οι αρχικοί στόχοι και απαιτήσεις που είχαμε.

Στο κεφάλαιο 6 παραθέτουμε μια σύνοψη των όσων έγιναν και διάφορα συμπεράσματα, μια πιθανή μελλοντική επέκταση που θα μπορούσε να εφαρμοστεί στην εφαρμογή και τέλος το κοινό στο οποίο απευθύνεται η παρούσα διπλωματική εργασία.

2 ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ ΣΥΣΤΗΜΑΤΟΣ

2.1 Ανάλυση απαιτήσεων συστήματος

Σε αυτήν την ενότητα θα αναλύσουμε τις λειτουργικές και μη λειτουργικές απαιτήσεις της εφαρμογής μας. Σε γενικές γραμμές θα δώσουμε δύο σύντομους ορισμούς για τους παραπάνω δύο όρους!

- Λειτουργικές Απαιτήσεις καλύπτουν την προσδοκούμενη συμπεριφορά της εφαρμογής. Αποτελούνται από τις υπηρεσίες, τις διεργασίες ή τις μεθόδους που πρέπει να ακολουθήσουμε κατά την υλοποίηση (Malan, 1999) Είναι εκείνες που αποτελούν το κανάλι επικοινωνίας μεταξύ των πελατών και των προγραμματιστών ανάπτυξης συστημάτων ή και εφαρμογών. Π.χ. αν ένας χρήστης κάνει κλικ σε μια ιστοσελίδα μια ενέργεια πρέπει να γίνει όπου θα του φέρνει κάποια αποτελέσματα (qracorp, n.d.)

- Μη λειτουργικές Απαιτήσεις είναι απαιτήσεις που δεν έχουν να κάνουν τόσο με τη συμπεριφορά και τις λειτουργίες που θα πρέπει να επιτελεί η εφαρμογή αλλά έχουν να κάνουν σχέση με την επεκτασιμότητα, τη συμβατότητα της εφαρμογής, την ασφάλεια, απόδοση, συντήρηση, στυλ αλληλεπίδρασης, διαθεσιμότητα, αξιοπιστία, χωρητικότητα, ανάκτηση, διαχειρισιμότητα, ακεραιότητα δεδομένων, κ.α. (Malan, 1999; Systemation, 2015) Είναι εκείνες που αποτελούν την ποιότητα του προϊόντος. Π.χ. αν κάποιος κάνει κλικ σε μια ιστοσελίδα και η ενέργεια που θα γίνει του φέρει πίσω κάποια αποτελέσματα πόσο γρήγορα αυτά τα αποτελέσματα θα επιστραφούν ? χωρίς αυτό το μέτρο της απόδοσης στην επιστροφή δεδομένων η εμπειρία χρήστη μειώνεται και η ποιότητα της εφαρμογής είναι σε ρίσκο! (qracorp, n.d.)

Για να αποφασίσουμε ποιες είναι οι λειτουργικές και ποιες οι μη λειτουργικές μας απαιτήσεις μια καλή τεχνική είναι να χωρίσουμε σε ομάδες τις ανάγκες που έχουμε να ικανοποιήσουμε ! Καλό είναι να θυμόμαστε ότι κάθε λειτουργική απαίτηση συνήθως έχει και μια μη λειτουργική συμπληρωματική, οπότε για τις λειτουργικές απαιτήσεις θα χωρίζαμε αυτές τις ανάγκες σε :

2.1.1 Λειτουργικές Απαιτήσεις

- Επιχειρησιακές Ανάγκες = είναι ο απώτερός μας στόχος θα μπορούσε να είναι είτε ένα σύστημα παραγγελιών, είτε ένα φυσικό προϊόν, (qracorp, n.d.) αλλά στη δική μας περίπτωση είναι η δημιουργία μιας εφαρμογής διαδραστικών διαδικτυακών χαρτών όπου ο χρήστης θα μπορεί να αλληλεπιδρά με τον χάρτη και να κάνει παρατηρήσεις πάνω στα απεικονιζόμενα δεδομένα του χάρτη.

- Διαχειριστικές Ανάγκες = θα μπορούσε να είναι ένα σύνολο μεθόδων για αναφορές (reporting), (qracorp, n.d.) στην δική μας περίπτωση δεν υπάρχει κάποια τέτοια απαίτηση στο στάδιο που είναι ακόμα η εφαρμογή

- Ανάγκες Χρήστη = είναι οι ανάγκες που η εφαρμογή μας θα πρέπει να διεκπεραιώνει για να μπορεί να τη χειριστεί ο χρήστης και να του παρέχει τα αποτελέσματα για τα οποία τη χρησιμοποιεί, (qracorp, n.d.) στην εφαρμογή που έχουμε αυτές οι ανάγκες είναι οι υπάρχουσες λειτουργίες της εφαρμογής όπως η δυνατότητα ο χρήστης να μπορεί να ενεργοποιήσει ένα trajectory πάνω στο χάρτη ανοίγοντάς το ώστε να εμφανιστούν όλες οι θέσεις του επιλεγμένου πλοίου, η δυνατότητα να μπορεί να του εμφανιστούν σχετικές πληροφορίες για την εκάστοτε επιλεγμένη θέση του πλοίου του οποίου τις θέσεις μελετά, η δυνατότητα να μπορεί να περνάει το δείκτη του ποντικιού του πάνω από αυτές τις θέσεις και να του επισημαίνονται με ειδικό χρώμα οι συνεχόμενα ίδιες ετικέτες κατηγορίας δραστηριότητας στις συνεχόμενες θέσεις στις οποίες βρίσκονται και τέλος η δυνατότητα να μπορεί να αλλάξει το χρώμα φόντου της εφαρμογής και η αποθήκευση της τελευταίας κάθε φορά ληφθείσας θέσης του κάθε επιλεγμένου πλοίου σε ειδική καρτέλα εντός της εφαρμογής όπως και η δυνατότητα διαγραφής των επιλεγμένων ανοιχτών trajectories.

- Ανάγκες Συστήματος = έχουν να κάνουν σχέση με λεπτομέρειες του λογισμικού ή του υλικού ενός συστήματος ή και με την ανταπόκριση του συστήματος (qracorp, n.d.) που στην δική μας περίπτωση αυτές οι ανάγκες θα μπορούσαν να προσδιοριστούν ως η προδιαγραφή υπάρξεως μιας μνήμης RAM αρκετών GB ώστε να μπορέσουν να φορτωθούν επιτυχώς όλα τα δεδομένα στον χάρτη καθώς γνωρίζουμε ότι γραφικές αναπαραστάσεις στο frontend απαιτούν αυτό το χαρακτηριστικό από το υλικό του μηχανήματος στο οποίο πάνω τρέχουμε την εφαρμογή μας, επίσης θα χρειαστεί από πλευράς υλικού και μια αρκετών δυνατοτήτων CPU για τους υπολογισμούς και I/O λειτουργιών του backend. Από άποψη λογισμικού πρέπει να

έχουμε σίγουρα έναν περιηγητή της επιλογής μας ώστε να δούμε εκεί το τελικό προϊόν πως εκτελείται.

Και για τις μη λειτουργικές θα χωρίζαμε αυτές τις ανάγκες σε :

2.1.2 Μη λειτουργικές απαιτήσεις

- Χρηστικότητα = έχει να κάνει με την εμφάνιση και με τη διεπαφή χρήστη και την αλληλεπίδρασή του με αυτήν, τι χρώμα έχει η εφαρμογή, πόσο μεγάλα είναι τα κουμπιά ενεργειών κλπ. (qracorp, n.d.) η εφαρμογή αυτό το κομμάτι το έχει σε μεγάλο ποσοστό τηρήσει με την άπλετη αλληλεπίδραση και εξατομίκευση που προσφέρει στον χρήστη καθώς επιτρέπει όπως έχει ήδη προαναφερθεί την αλλαγή χρώματος φόντου και τις διάφορες ενέργειες εντός του χάρτη πάνω σε κάποιο επιλεγμένο πλοίο του οποίου έχουν εμφανιστεί οι θέσεις και οι δραστηριότητες πλοήγησης με ειδικό χρώμα τονισμού. Επίσης η δυνατότητα διαγραφής των ήδη ανοιχτών δρομολογίων και η αλληλεπίδραση με τα αναδυόμενα παράθυρα με τις σχετικές πληροφορίες για την επιλεγμένη θέση του πλοίου είναι κάποιες από τις υπάρχουσες επιλογές αλληλεπίδρασης της εφαρμογής με το χρήστη!

- Διαθεσιμότητα = έχει να κάνει με το χρόνο που η εφαρμογή είναι προσβάσιμη, χρειάζεται να είναι 24 ώρες το 24ωρο όλες τις μέρες της βδομάδας για όλο το έτος ή είναι διαθέσιμη και προσβάσιμη με συγκεκριμένες μέρες και ώρες στον τελικό χρήστη? (qracorp, n.d.) η εφαρμογή θα μπορούσε να είναι διαθέσιμη 24/7/365 από τη στιγμή που γίνει deploy σε κάποιο cloud provider

- Επεκτασιμότητα = αν χρειαστεί επέκταση μπορεί το σύστημα να το αντέξει π.χ. για φυσικές εγκαταστάσεις επιπλέον υλικού? (qracorp, n.d.), στο προσεχές μέλλον σε περίπτωση επέκτασης της εφαρμογής με την προσθήκη περαιτέρω λειτουργιών και δυνατοτήτων που καθίσταται απαραίτητη η επέκτασή της ο κώδικας με τον οποίο έχει υλοποιηθεί δίνει τη δυνατότητα αυτή διότι κάνει χρήση της javascript βιβλιοθήκης react που θα αναφέρουμε και σε επόμενο κεφάλαιο πιο αναλυτικά κι έτσι επιτυγχάνει τη λογική των components από τη μεριά του view που την κάνει ευκολοδιαχειρίσιμη σε τέτοια περίπτωση!

- Απόδοση = πόσο γρήγορα λειτουργεί? (qracorp, n.d.) η απόδοση είναι συνήθως ανάλογη και των πόρων που διαθέτει το σύστημα στο οποίο τρέχουμε την εφαρμογή πολλές φορές! Ανάλογα λοιπόν και με το σύστημα που έχουμε η εφαρμογή ανταποκρίνεται γρήγορα χωρίς καθυστερήσεις και φορτώνει έναν όγκο δεδομένων που λαμβάνει από το dataset που διαθέτουμε!

- Δυνατότητα Υποστήριξης = η υποστήριξη που έχει είναι με φυσική παρουσία ή εξ' αποστάσεως? (qracorp, n.d.)

- Ασφάλεια = τι ασφάλεια παρέχει και από την οπτική των εγκαταστάσεων και από την οπτική του cyber security. (qracorp, n.d.)

2.1.3 Use Case UML Diagram

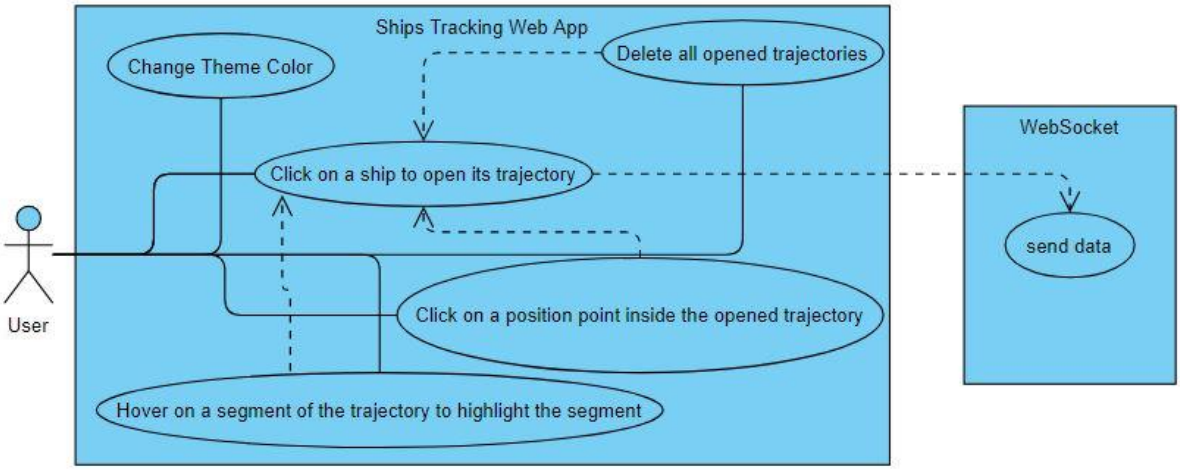
Ένα UML use case διάγραμμα είναι ο κύριος και βασικός τρόπος απεικόνισης των απαιτήσεων ενός νέου υπό υλοποίηση συστήματος/λογισμικού. Επικεντρώνονται στην προσδοκώμενη συμπεριφορά (στο τι θέλουμε) και όχι στον ακριβή τρόπο με τον οποίο αυτό θα συμβεί/υλοποιηθεί (όχι στο πως). Μας βοηθάει στο να κατασκευάσουμε ένα σύστημα από άποψη πλευράς χρήστη.

- Συνοψίζει κάποιες από τις σχέσεις μεταξύ των cases, των actors και του συτήματος.

- Δεν εστιάζει στη σειρά την οποία ακολουθούν τα διάφορα βήματα που εκτελούνται για να επιτύχουν τους στόχους του κάθε case.

Στη UML ανήκει στην κατηγορία των Behavioral Diagrams (Visual Paradigm, n.d.)

Στο παρακάνω use case διάγραμμα φαίνονται οι αρχικές βασικές απαιτήσεις του συστήματος που υλοποιήθηκαν και οι σχέσεις που έχουν μεταξύ τους.



Εικ.1. Use Case Diagram

Στο παραπάνω use case διάγραμμα παρατηρούμε όλες τις ενέργειες που μπορεί να κάνει ο χρήστης με την εφαρμογή ώστε στη συνέχεια να του επιστραφούν τα αποτελέσματα των ενεργειών αυτών. Με άλλα λόγια βλέπουμε όλες τις λειτουργικές απαιτήσεις που έχει η εφαρμογή μας και τις εξαρτήσεις κάποιων από αυτών (μη συνεχόμενα βελάκια / dashed arrows) που δηλώνουν ότι για να γίνει μια συγκεκριμένη ενέργεια κάποια άλλη πρέπει να έχει προηγηθεί. Επίσης, βλέπουμε και την εξάρτηση της κυριότερης ενέργειας από την πηγή που στέλνει τα δεδομένα.

2.2 Κατηγορίες χρηστών της εφαρμογής

Στον παρακάτω πίνακα βλέπουμε διάφορες κατηγορίες χρηστών και με σήμανση επιλογής όλους όσους απευθύνεται η εφαρμογή της διπλωματικής αυτής εργασίας. Συνήθως οι κατηγορίες χρηστών προέρχονται από τις ιδιότητες/δικαιώματα (priviledges) που έχει κάποιος στην εφαρμογή και από τους ρόλους που υπηρετεί καθώς και από τη σχέση που έχει με τη βάση δεδομένων της εφαρμογής. Επίσης τα δικαιώματα μπορούν να διαφέρουν ανάλογα εάν ο χρήστης χρειάζεται να κάνει εγγραφή στην εφαρμογή για να τα αποκτήσει ή όχι. (Ion et al., 2011)

ΚΑΤΗΓΟΡΙΕΣ ΧΡΗΣΤΩΝ	**ΕΠΙΛΟΓΗ**
Απλοί χρήστες / τυχαίοι επισκέπτες	✓
Ιδιοκτήτες Πλοίων	✓
Παρατηρητές στατιστικών	✓
Ακτοφύλακες (Coast Guards)	✓
Αναλυτές	✓
Διαχειριστές (Administration Users)	Όχι ακόμα
Επιχειρησιακούς Χρήστες (Business Users)	Όχι ακόμα
Σχεδιαστές (Designers)	Όχι ακόμα
Διευθυντικούς Χρήστες (Managers)	Όχι ακόμα
Χρήστες Οπτικών Γραφικών (Visualization Users)	Όχι ακόμα
Εγγεγραμμένους Χρήστες	Όχι ακόμα
Μη εγγεγραμμένους Χρήστες	✓
Χρήστες με δυνατότητα CRUD λειτουργιών προς τη βάση δεδομένων	Όχι ακόμα

2.3 Αρχιτεκτονική

Σκοπός αυτής της παραγράφου είναι να αναλυθεί και να περιγραφεί η αρχιτεκτονική την οποία ακολουθεί η εφαρμογή, ο τρόπος συσχέτισης των επιμέρους συστατικών της με τις αρχικές απαιτήσεις και αντικειμενικούς στόχους που είχαν τεθεί πριν την υλοποίηση της καθώς επίσης και η λειτουργικότητα καθενός συστατικού από αυτά.

2.3.1 Ανάθεση απαιτήσεων σε συστατικά λογισμικού (components)

Προηγουμένως αναφερθήκαμε στην συσχέτιση των συστατικών μερών της εφαρμογής με τις απαιτήσεις δηλώνοντας κάτι τέτοιο με χρήση ενός use case διαγράμματος. Σε αυτήν την παράγραφο θα αναθέσουμε κάθε μία από αυτές τις απαιτήσεις που αναφέρθηκαν προηγουμένως σε κάποιο συστατικό μέρος της εφαρμογής ώστε να τεθεί προς υλοποίηση. Η ανάθεση αυτή και ο διαχωρισμός των μερών της εφαρμογής μπορεί να αναπαρασταθεί από ένα άλλο διάγραμμα το γνωστό component diagram.

Εδώ θα αναλύσουμε λίγο τις λεπτομέρειες της ανάθεσης αυτής ανά συστατικό και παρακάτω θα δείξουμε την αναπαράσταση των συστατικών σε διάγραμμα από το οποίο δημιουργήθηκε η εφαρμογή. Στο σημείο αυτό αξίζει να τονίσουμε ότι η υλοποίηση της εφαρμογής έγινε με τη βιβλιοθήκη react που σημαίνει ότι τα παρακάτω components απευθύνονται μόνο στο κομμάτι του view καθώς η react δεν υποστηρίζει τις έννοιες του model και του controller από το μοντέλο MVC, όπως θα δούμε και σε επόμενο κεφάλαιο.

- **App Component** = λειτουργεί ως wrapper γύρω από τα δύο βασικά συστατικά μέρη της εφαρμογής και πιο συγκεκριμένα υλοποιεί ένα συστατικό router με μέσα ένα άλλο συστατικό switch με δυο routes όπου το ένα route είναι το LoadingScreen συστατικό με μονοπάτι στο url το → path="/", και το MainScreen με μονοπάτι το → path="/view".

- **LoadingScreen Component** = είναι το πρώτο συστατικό το οποίο εμφανίζεται στην οθόνη από το οποίο αν πατηθεί το ειδικό κουμπί σε παραπέμπει στο άλλο συστατικό της εφαρμογής το MainScreen. Με άλλα λόγια λειτουργεί ως ένας δρομολογητής από το συστατικό εισόδου στην εφαρμογή στο συστατικό της ίδιας της εφαρμογής! Είναι ο λόγος για τον οποίο η εφαρμογή μπορεί να θεωρηθεί και ως SPA εφαρμογή διότι η δρομολόγηση γίνεται από το ένα συστατικό στο άλλο χωρίς την ανάγκη επαναφόρτωσης της σελίδας!

- **MainScreen Component** = εμφωλεύει όλα τα υπόλοιπα συστατικά όπως το TheMap, Info και TheNavBar στα οποία τεμαχίσαμε στην ουσία το ui της εφαρμογής μας. Ο λόγος υπάρξεώς του και κατ' επέκταση και της εν λόγου εμφωλεύσεως είναι για να μπορούμε να μεταφέρουμε πληροφορία στα εμφωλευμένα συστατικά από μια ανώτερη ιεραρχία συστατικών προς την εμφωλευμένη μέσω της χρήσης ενός συστατικού που λειτουργεί ως provider. Συγκεκριμένα μέσω αυτού του συστατικού διαμοιράζουμε σε όλα τα υπόλοιπα συστατικά μέρη της εφαρμογής το χρώμα επιλογής του χρήστη για εξατομίκευση του χρώματος φόντου.

- **TheMap Component** = είναι το πιο σπουδαίο συστατικό καθώς αυτό είναι υπεύθυνο για την δημιουργία του χάρτη και την απεικόνιση των δεδομένων (πάνω στον χάρτη) που έρχονται από το backend! Κάθε στρώμα του χάρτη δημιουργείται και προστίθεται σε αυτόν χάρης αυτό το συστατικό όπως επίσης και η οποιαδήποτε ομαδοποίηση με βάση κάποια κοινή τιμή όπως οι συνεχόμενα ίδιες τιμές στις ετικέτες πλοήγησης και η απεικόνισή του με χρώμα τονισμού εντός του trajectory γίνεται σε αυτό το συστατικό με μεθόδους που θα αναλυθούν παρακάτω σε επόμενο κεφάλαιο!

- **ChangeThemeColor Component** = είναι το συστατικό εκείνο που είναι υπεύθυνο να δημιουργήσει το createContext hook της react ώστε να αποθηκεύει το state των χρωμάτων σε διαφορετικά useContext hooks κάθε φορά που ο χρήστης το αλλάζει

- **Info Component** = είναι υπεύθυνο για τη δημιουργία των καρτελών στα δεξιά της εφαρμογής και σε αυτό περνάει μέσω του useContext hook όλο το αλφαριθμητικό με τα δεδομένα της τελευταίας θέσης που έχει ληφθεί από το backend μέχρι στιγμής και ο χρήστης έχει πατήσει εντός του χάρτη. Η πληροφορία αυτή περνάει σε αυτό το συστατικό από το TheMap συστατικό και χρησιμοποιείται από αυτό με τη useContext

- **TheNavBar Component** = είναι υπεύθυνο για τη δημιουργία των δύο navigation bar της εφαρμογής και τη λειτουργία τους. Λαμβάνει είσοδο από το συστατικό ChangeThemeColor ώστε να δώσει χρώμα επιλογής του χρήστη στο navigation bar,

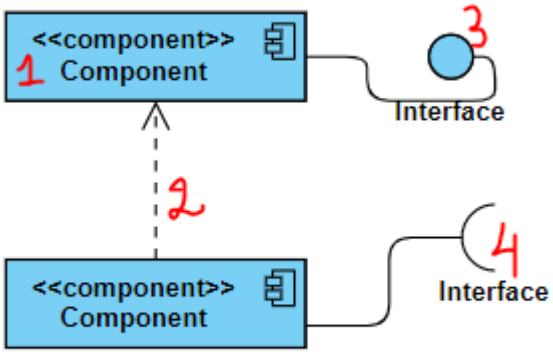
όπως για τον ίδιο λόγο λαμβάνουν και τα υπόλοιπα συστατικά μέρη την ίδια πληροφορία!

2.3.1.1 UML UI Component model Diagram

Για την αναπαράσταση της αρχιτεκτονικής καθώς και την επεξήγηση αυτής χρησιμοποιήθηκε λοιπόν το component diagram. Αφού δώσουμε μια μικρή επεξήγηση των εννοιών αυτού θα αποτυπώσουμε τις αλληλοεξαρτήσεις των παραπάνω προαναφερθέντων συστατικών με βάση αυτό το διάγραμμα. Αυτές οι αλληλοεξαρτήσεις μπορούν να φανούν και από την παραπάνω ανάλυση των λειτουργιών του κάθε συστατικού!

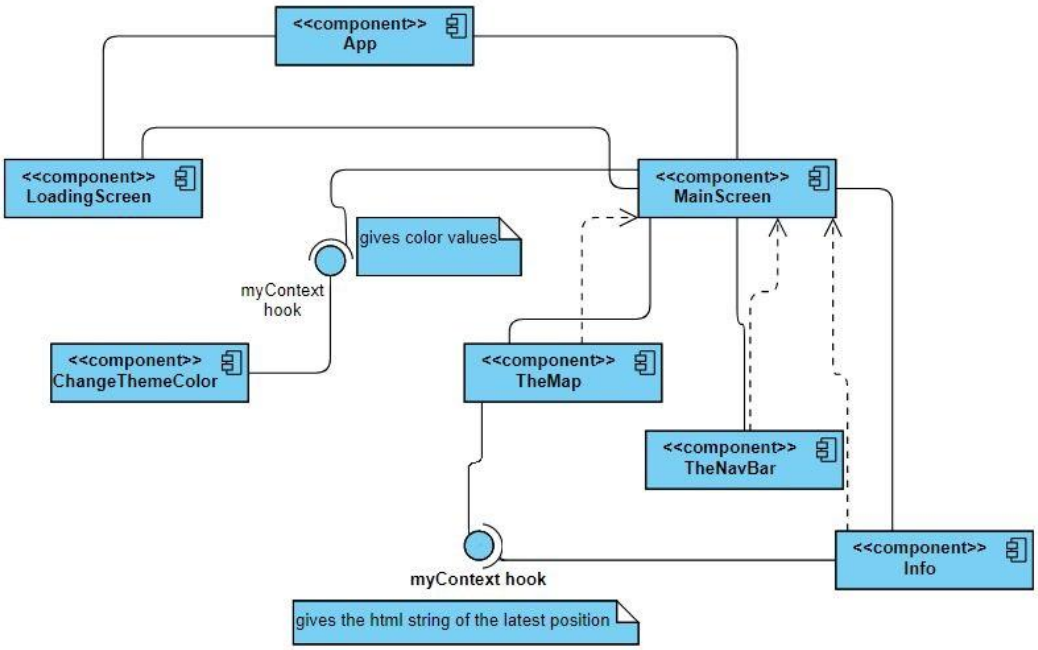
Ένα component diagram είναι ένα αναπόσπαστο κομμάτι για την ανάπτυξη ενός λογισμικού συστήματος. Σχεδιάζεται με τη βοήθεια ενός λογισμικού σχεδίασης σε UML και είναι οι αρωγοί στην κατανόηση της δομής των ήδη υπαρχόντων συστημάτων με σκοπό την επιτυχή δημιουργία νέων. Σκοπός τους είναι να τονίσουν τη σχέση μεταξύ διαφορετικών components του συστήματος. Στη UML κατατάσσεται στην κατηγορία των Structural Diagrams (*Component Diagram Tutorial*, n.d.)

Πριν εξηγήσουμε πως σχετίζονται τα components στο παραπάνω διάγραμμα, θα εξηγήσουμε πρώτα με λίγα λόγια τα σύμβολα τα οποία εμφανίζονται στο διάγραμμα. Η παρακάτω εικόνα έχει αριθμημένα όλα τα σύμβολα που χρησιμοποιήθηκαν και επεξηγούνται παρακάτω.



Εικ.2. Επεξήγηση συμβόλων Component διαγράμματος

1. Component	Μία οντότητα που απαιτείται για να εκτελεστεί μια λειτουργία. Προσφέρει και χρησιμοποιεί λειτουργικότητα από τα interfaces και άλλα components. (*Component Diagram Tutorial*, n.d.)
2. Dependency	Δείχνει ότι ένα κομμάτι του συστήματος εξαρτάται από κάποιο άλλο. (*Component Diagram Tutorial*, n.d.)
3. Provided Interface	Αναπαριστά όλα τα inputs και τα materials που παρέχει ή λαμβάνει ένα component. Αντιπροσωπεύει τα interfaces τα οποία ένα component παρέχει ως δεδομένα στο required interface ενός άλλου. (*Component Diagram Tutorial*, n.d.) Αντιπροσωπεύει και τα services τα οποία το component χρειάζεται ώστε να φέρει εις πέρας τις υποχρεώσεις του(Visual Paradigm, n.d.)
4. Required Interface	Αναπαριστά όλα τα inputs και τα materials που παρέχει ή λαμβάνει ένα component. Αντιπροσωπεύει τα interfaces από τα οποία το component χρειάζεται δεδομένα ώστε να φέρει εις πέρας τις λειτουργίες του. (*Component Diagram Tutorial*, n.d.) Αντιπροσωπεύει και τα services τα οποία το component χρειάζεται ώστε να φέρει εις πέρας τις υποχρεώσεις του(Visual Paradigm, n.d.)

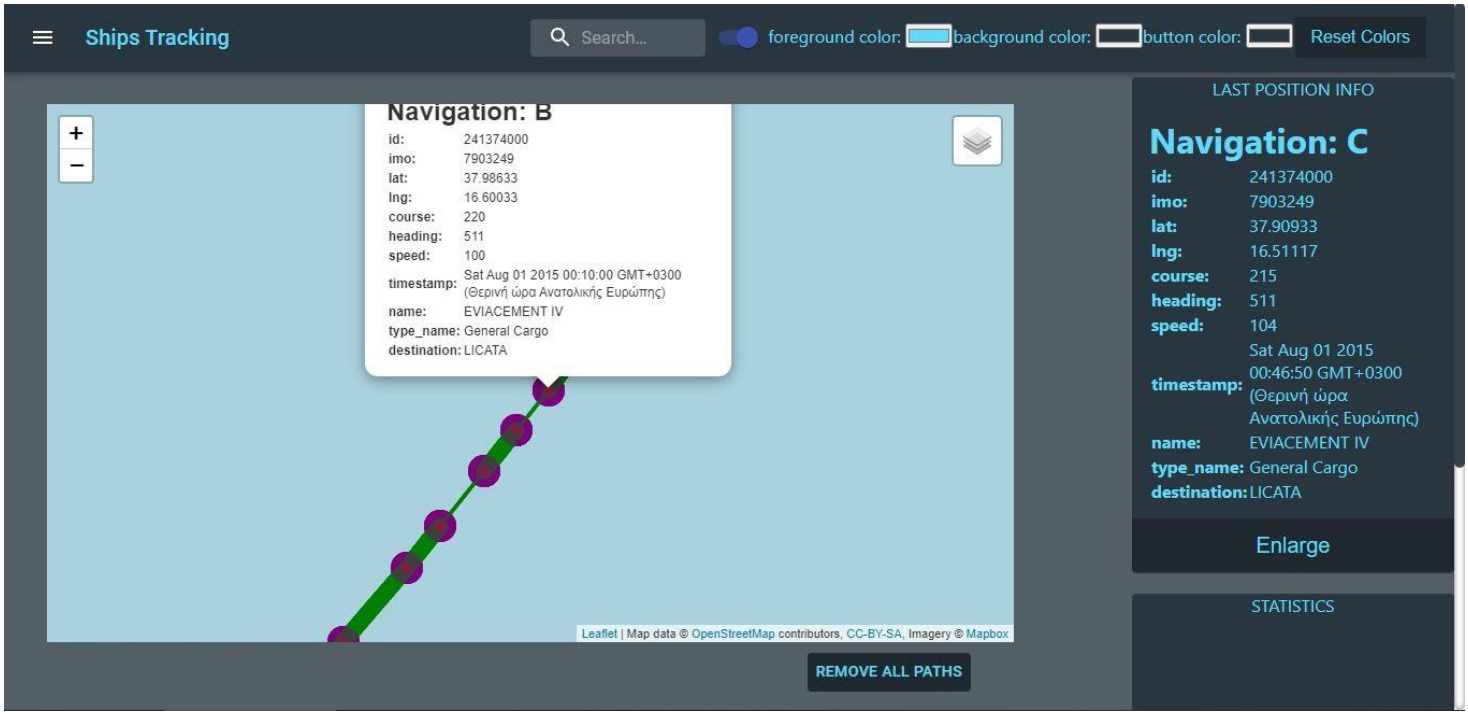


Εικ.3. Component Diagram

Εν συντομία επισημαίνεται πως το σύμβολο που στα αγγλικά ονομάζεται ως 'lollipop' στο παραπάνω διάγραμμα δηλώνει τη μεταφορά της πληροφορίας του χρώματος στην μία περίπτωση και τη μεταφορά του αλφαριθμητικού με τα μεταδεδομένα της τελευταίας θέσης του επιλεγμένου πλοίου στην άλλη περίπτωση από το ένα συστατικό στο άλλο! Το ανοιχτό ημισφαίριο δηλώνει την λήψη της πληροφορίας από το συστατικό που κάνει τη λήψη ενώ το κλειστό μπλε κυκλάκι του συμβόλου αυτού δηλώνει την αποστολή της πληροφορίας από το συστατικό αποστολής της. Τα TheMap, TheNavBar και Info components εξαρτώνται από το MainScreen για να λάβουν αυτήν την πληροφορία του χρώματος στην προκειμένη περίπτωση ενώ Info επικοινωνεί απευθείας με τον αποστολέα οπότε δεν συμπεριελήφθη το βελάκι με τις παύλες ώστε να δηλώσει εξάρτηση!

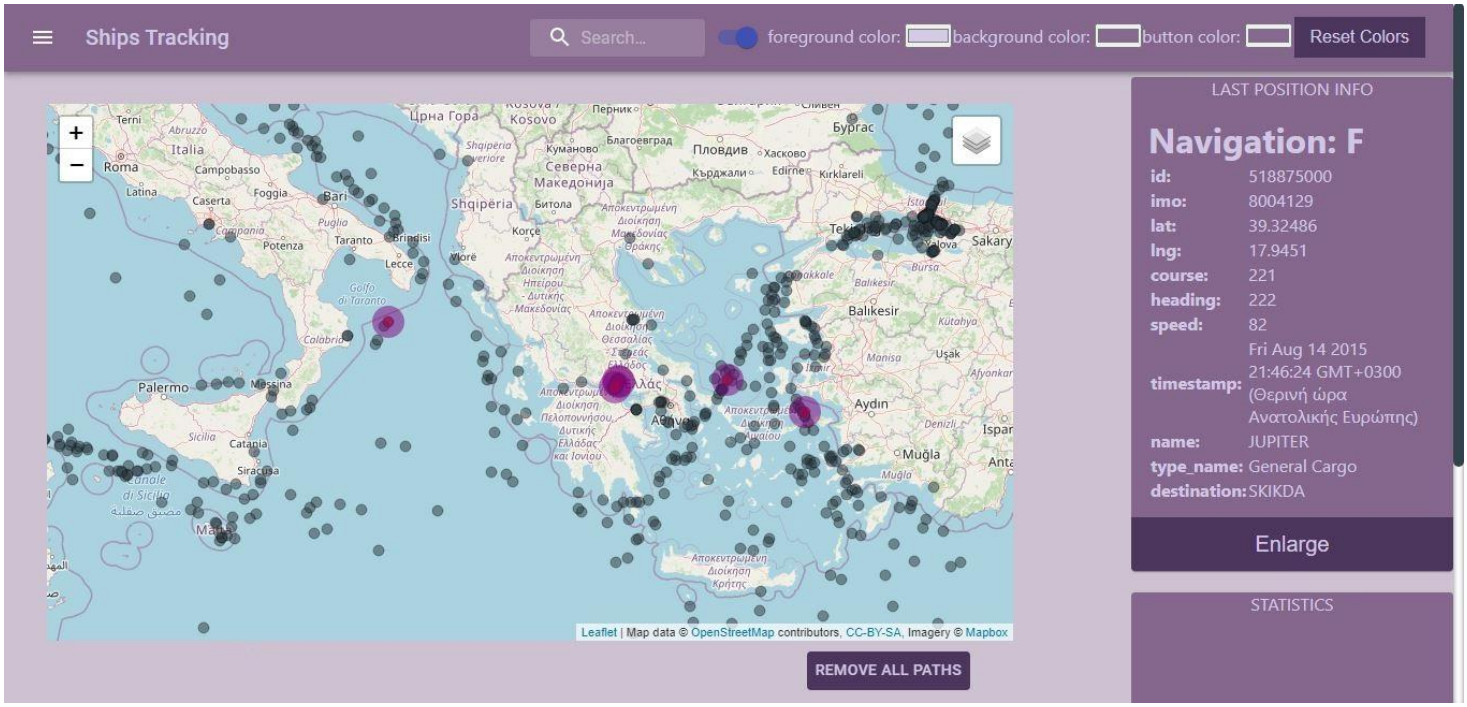
2.3.2 Περιγραφή Αλληλεπιδράσεων (Interfaces)

Στις παρακάτω τρεις εικόνες έχουμε μία οπτική επαφή με τη διεπαφή χρήστη της εφαρμογής και τις αλληλεπιδράσεις που μπορεί ο χρήστης να έχει με αυτή.



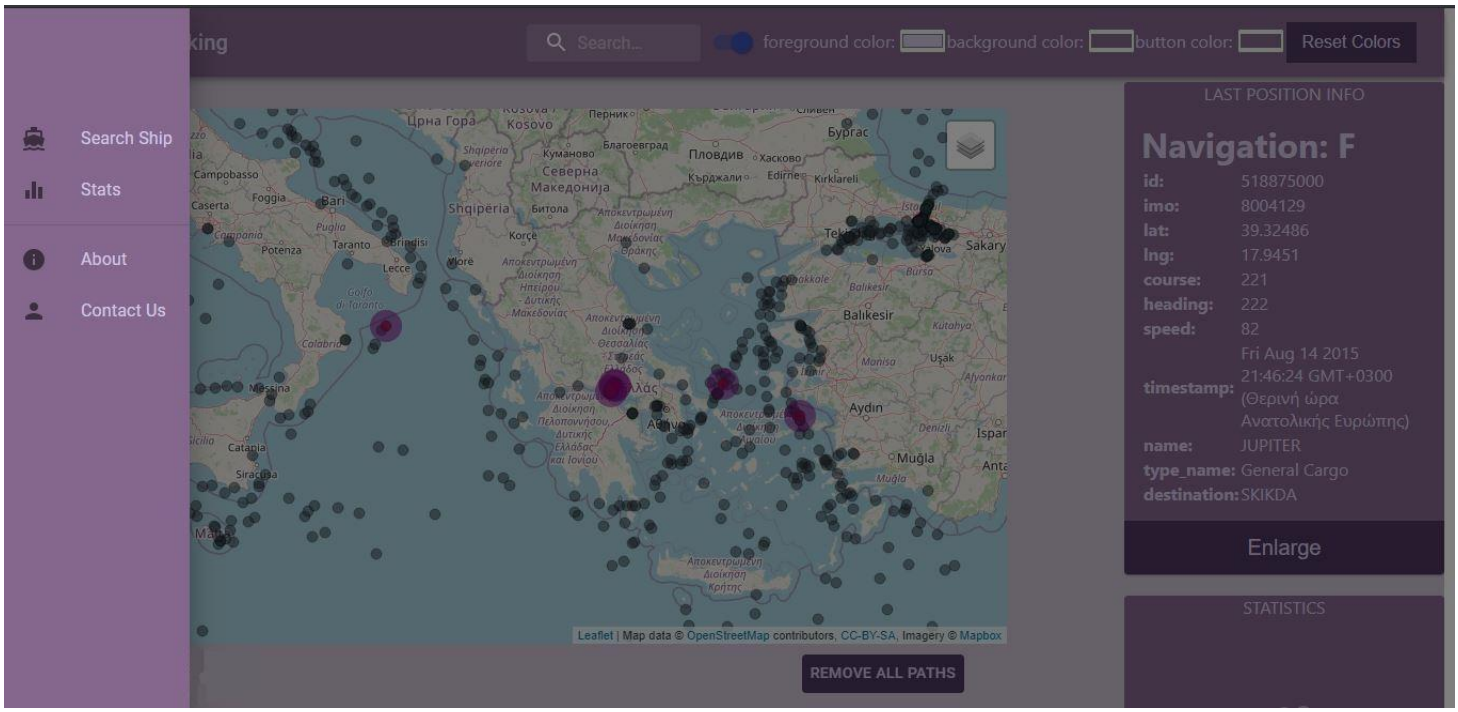
Εικ.4. Στυλ Αλληλεπίδρασης εφαρμογής 1

Στην εικόνα βλέπουμε ότι ο χρήστης καθώς πατήσει σε ένα πλοίο ανοίγει ολόκληρο το δρομολόγιο θέσεων που έχει αυτό ακολουθήσει και η αλληλεπίδραση που μπορεί να έχει ο χρήστης πέρα από το αρχικό κλικ στο πλοίο που έκανε είναι να περάσει το δείκτη του ποντικιού του πάνω από μια έντονη πράσινη γραμμή χωρίς να κάνει κλικ και θα διαπιστώσει ότι η εφαρμογή ανταποκρίνεται στην κίνησή του χρωματίζοντας με κίτρινο τονισμό το έντονο πράσινο μονοπάτι το οποίο στην ουσία είναι αυτό που έχουμε ήδη προαναφέρει ότι δηλώνει τη συνεχόμενη εμφάνιση ιδίων ετικετών πλοήγησης ή αλλιώς κατηγοριών δραστηριότητας. Επίσης πατώντας μεμονωμένα πάνω από κάποιο σημείο θέσης ένα αναδυόμενο παράθυρο σαν και αυτό που φαίνεται στην εικόνα θα εμφανιστεί έχοντας πληροφορίες μεταδεδομένων σχετικά με το πλοίο. Στα δεξιά της εικόνας φαίνεται η καρτέλα που αποθηκεύει για όσο χρονικό διάστημα ο χρήστης έχει ενεργοποιημένο αυτό το δρομολόγιο αυτού του πλοίου η τελευταία ληφθείσα θέση ακόμα και αν ο χρήστης πατήσει να μάθει πληροφορίες σε κάποιο άλλο ενδιάμεσο σημείο θέσης (τα σημεία θέσης απεικονίζονται με μωβ χρώμα πάνω στο μονοπάτι). Κάτω από τον χάρτη υπάρχει ένα κουμπί διαγραφής όλων των επιλεγμένων δρομολογίων με το οποίο μπορεί εξίσου να αλληλεπιδράσει ο χρήστης εκτελώντας την ενέργεια της διαγραφής. Εντός του χάρτη υπάρχει επιλογή εμφάνισης συγκεκριμένης κατηγορίας πλοίων που στην προκειμένη είναι ενεργοποιημένη η κατηγορία general cargo να εμφανίζεται στον χάρτη. Τέλος ο χρήστης μπορεί να δοκιμάσει να κάνει ζουμ μέσα ή έξω από τον χάρτη και να δει μεγαλύτερη περιοχή κάλυψης!



Εικ.5. Στυλ Αλληλεπίδρασης εφαρμογής 2

Στην δεύτερη εικόνα που παραθέτουμε έχουμε σε ζουμ μακρινό την αλληλεπίδραση με τον χάρτη που έχει ο χρήστης όταν έχει ενεργοποιήσει αρκετά trajectories τα οποία μπορεί να διαγράψει πατώντας το κουμπί κάτω δεξιά στον χάρτη! Επίσης γίνεται εμφανής σε αυτή την εικόνα και η αλλαγή χρώματος στο φόντο!



Εικ.6. Στυλ Αλληλεπίδρασης εφαρμογής 3

Στην τελευταία εικόνα βλέπουμε μια ακόμα αλληλεπίδραση που έχουμε με μια navigation bar στα αριστερά της εφαρμογής

3 ΑΝΑΣΚΟΠΗΣΗ ΤΡΕΧΟΥΣΑΣ ΚΑΤΑΣΤΑΣΗΣ (State Of The Art) ΚΑΙ ΠΑΡΟΜΟΙΕΣ ΑΛΛΕΣ ΕΦΑΡΜΟΓΕΣ (Related Work)

3.1 Παρόμοιες εφαρμογές στην αγορά

Στην αγορά πολλές είναι οι φορείς υπηρεσιών παρακολούθησης πλοίων που παρέχουν πολλές δυνατότητες στους πελάτες τους συγκεκριμένα παρακάτω παρατίθενται οι 3 πιο κορυφαίοι φορείς και μια μικρή σύντομη ανάλυση των υπηρεσιών τους.

1. Marine-Traffic (προσφέρει διαδραστικούς διαδικτυακούς χάρτες για παρακολούθηση πλοίων σε πραγματικό χρόνο και με αρκετά έξυπνα φίλτρα αναζήτησης, AIS σύστημα παρακολούθησης, όπως και εμπλουτισμένη δορυφορική κάλυψη, χάρτες πυκνότητας, ναυτικά γραφήματα, εικόνες των διαφόρων πλοίων και πολλά άλλα, όπως υπολογισμός καιρικών συνθηκών, πιθανού δρομολογίου έως τον τελικό προορισμό, παρακολούθηση λιμανιών κλπ)
2. FleetMon (προσφέρει παρακολούθηση των θέσεων των πλοίων με χρήση του συτήματος AIS, τεχνικές πληροφορίες πλοίων, αφίξεις στα λιμάνια, μοτίβα συναλλαγών, έλεγχο στόλου, ανάλυση κίνησης κλπ)
3. Shipfinder (προσφέρει παρακολούθηση πλοίων μέσω φίλτρων αναζήτησης όπως ο αριθμός πλοίου, η χωρητικότητα, ο ιδιοκτήτης κλπ κρατά και ιστορικό αναζήτησης για καλύτερη εμπειρία χρήστη) (Kaushik, 2021)

3.1.1 Ανάλυση προσέγγισης ως προς την ικανοποίηση απαιτήσεων

Η εφαρμογή μας εστιάζει προς το παρόν να ικανοποιήσει τις Ανάγκες του χρήστη από τη μεριά των λειτουργικών και τις ανάγκες χρηστικότητας από τη μεριά των μη λειτουργικών. Η προσέγγιση που ακολουθήθηκε για καθεμιά από τις απαιτήσεις περιγράφεται παρακάτω δίπλα από κάθε μία απαίτηση

- η δυνατότητα ο χρήστης να μπορεί να ενεργοποιήσει ένα trajectory πάνω στο χάρτη ανοίγοντάς το ώστε να εμφανιστούν όλες οι θέσεις του επιλεγμένου πλοίου

  ΠΡΟΣΕΓΓΙΣΗ ΙΚΑΝΟΠΟΙΗΣΗΣ ΑΠΑΙΤΗΣΗΣ

  - Χρησιμοποιώντας τη βιβλιοθήκη Leaflet δημιουργήσαμε το χάρτη στον οποίο μέσω της δομής GeoJSON αποτυπώσαμε σε ξεχωριστό στρώμα του χάρτη τις τελευταίες ληφθείσες θέσεις των πλοίων. Έπειτα καθώς ο χρήστης πατάει σε κάποια από αυτές ενεργοποιεί άλλο ένα στρώμα ξεχωριστό στον χάρτη το οποίο υλοποιεί έναν αλγόριθμο ομαδοποίησης των ίδιων εγγραφών που εντοπίζονται ως προς τον αριθμό ταυτοποίησης του πλοίου με αποτέλεσμα να επιτρέπει την εμφάνιση όχι μόνο του τελευταίου σημείου θέσεως αλλά και όλων των υπολοίπων. Το στρώμα αυτό δημιουργείται σε ειδική μέθοδο με υπογραφή whenClicked(). Ο αλγόριθμος ομαδοποίησης θα επεξηγηθεί με λεπτομέρεια σε επόμενο κεφάλαιο καθώς χρησιμοποιεί τη συνάρτηση reduce για την αναζήτηση όλων των εγγραφών του ιδίου πλοίου!

- η δυνατότητα να μπορεί να του εμφανιστούν σχετικές πληροφορίες για την εκάστοτε επιλεγμένη θέση του πλοίου του οποίου τις θέσεις μελετά

  ΠΡΟΣΕΓΓΙΣΗ ΙΚΑΝΟΠΟΙΗΣΗΣ ΑΠΑΙΤΗΣΗΣ

  - Το στρώμα που δημιουργείται με το πάτημα σε κάποιο πλοίο και αναφέρθηκε ακριβώς πιο πάνω υλοποιεί σε ειδική μέθοδο εντός του GeoJSON στρώματος και συγκεκριμένα της μεθόδου onEachFeature την εμφάνιση ειδικού παραθύρου με πληροφορίες σχετικά με το πλοίο το οποίο κάνει bindPopup στο στρώμα για το επιλεγμένο σημείο θέσης. Εντός του αναδυόμενου αυτού παραθύρου ενσωματώνει ένα html αλφαριθμητικό με το περιεχόμενο που πρέπει να διαβαστεί ώστε να εκτυπωθεί στην οθόνη προς ενημέρωση του χρήστη που έκανε την ενέργεια!

```
onEachFeature:  function (feature, layer) {
                var popupContent = '<h1><b>Navigation: </b>'+ feature.p
roperties['navigation'] +'</h1><table>';
                for (var p in feature.properties) {
                    if(p !== "show_on_map" && p !== "navigation"){
                    popupContent += "<tr><td><b>" + p + "<b>:</b>" + "<
/b></td><td>"+ feature.properties[p] + "</td></tr>";
                    }
```

```
                }
                popupContent += "</table>";
                    layer.bindPopup(popupContent);
```

Απόσπασμα 1: onEachFeature bind popup code snippet

Όπου p είναι ένα-ένα τα features ενός obj με όλα τα μεταδεδομένα/features ενός πλοίου π.χ. speed, imo, id, course, lat, lon κλπ..

- η δυνατότητα να μπορεί να περνάει το δείκτη του ποντικιού του πάνω από αυτές τις θέσεις και να του επισημαίνονται με ειδικό χρώμα οι συνεχόμενα ίδιες ετικέτες κατηγορίας δραστηριότητας στις συνεχόμενες θέσεις στις οποίες βρίσκονται

ΠΡΟΣΕΓΓΙΣΗ ΙΚΑΝΟΠΟΙΗΣΗΣ ΑΠΑΙΤΗΣΗΣ

  - Δημιουργούμε έναν segments array όπου τον γεμίζουμε με polylines οι οποίες διέρχονται από σημεία ίδιας κατηγορίας δραστηριότητας κάνοντας τους παρακάτω ελέγχους ξεκινώντας από μία τυχαία αρχική τιμή ως current_label = 'A' την οποία συγκρίνουμε με την ετικέτα του τρέχοντος σημείου στη σειρά/ακολουθία των ομαδοποιημένων εγγραφών

```
if(elmt.properties.navigation === current_label){

 c.push([x, y]); // holds the coordinates in a form that a polyline nee
ds in order to be drawn
```

Απόσπασμα 2: if current_label check code snippet

όπου χ και y το γεωγραφικό μήκος και πλάτος του κάθε σημείου με ίδια ετικέτα..

```
        var x = elmt.properties.lat;
         var y = elmt.properties.lng;
```

Απόσπασμα 3: variables of lon lat code snippet

δημιουργούμε το segment polyline και το προσθέτουμε στον segmentsarray όπως παρακάτω μηδενίζοντας τον πίνακα c ώστε να δεχτεί εκ νέου το επόμενο segment ίδιων ετικετών!

```
segment = L.polyline(c).setStyle({
                    color: 'green',
                    weight: 15
```

```
                }).addTo(mymap);
                segmentsArray.push(segment);
                c.length = 0;
```

Απόσπασμα 4: create segment polyline code snippet

αλλιώς θέτουμε το current_label με του τρέχοντος σημείου και ακολουθούμε παρόμοια βήματα όπως απεικονίζεται παρακάτω

```
else {
            segment = L.polyline(c).setStyle({
                color: 'green',
                weight: 15
            }).addTo(mymap);
            segmentsArray.push(segment);
            c.length = 0; // c array is getting filled with all points
under same label which form the whole segment
            // and when we find the whole segment we need to empty this
 array so it can store the next segment
            // before emptying it we save its segment inside the polyLi
neArray and slowly slowly each single segment
            // that we form will end up form the whole polyline at the
end !
            current_label = elmt.properties.navigation;
            c.push([x, y]);

        }
```

Απόσπασμα 5: else if not current_lanel check code snippet

Τέλος, στον τελικό segments array που δημιουργείται καλούμε την forEach item μέσα στον segments array όπως παρακάτω για να εφαρμόσουμε κίτρινο τονισμό σε κάθε segment κατά το hover

```
segmentsArray.forEach(function (segment, index) {
        segment.on('mouseover', function(e) {
            var layer = e.target;
            layer.setStyle({
                color: 'yellow',
                weight: 15,
                opacity: 0.5
            });
        });
```

Απόσπασμα 6: final segments array code snippet

- η δυνατότητα να μπορεί να αλλάξει το χρώμα φόντου της εφαρμογής

  ΠΡΟΣΕΓΓΙΣΗ ΙΚΑΝΟΠΟΙΗΣΗΣ ΑΠΑΙΤΗΣΗΣ

  - Δημιουργήσαμε έναν myContext hook εντός του συστατικού ChangeThemeColor και κάναμε χρήση αυτού εντός του συστατικού για ενημέρωση με νέες τιμές αυτού του myContext

```
export const myContext = createContext();
```

```
const {usercolor, setUserColorValue} = useContext(myC
ontext);
```

    υλοποιήσαμε και τις μεθόδους όπως παραδειγματικά παραθέτουμε μία στο παραπάνω snippet με το όνομα setUserColorValue οι οποίες ενημερώνουν το state της σταθεράς usercolor. Επίσης μπορούμε πλέον να κάνουμε χρήση με τη usecontext σε όποιο άλλο συστατικό θέλουμε να ενημερώσουμε την τιμή του χρώματος με τον παρακάτω τρόπο που φαίνεται στο παρακάτω snippet

```
<Card style={{backgroundColor: usercolor, color: userlettercolor}}>
```

Απόσπασμα 7: createContext useContext hooks code snippet

Όπου card είναι σαν ένα νέο html element που ορίζει η react με το materialUI στο οποίο element περνάμε με το style attribute το χρώμα του φόντου όπου μεταφέραμε στο component αυτό μέσω του παραπάνω hook όπως ορίσαμε!

- η αποθήκευση της τελευταίας κάθε φορά ληφθείσας θέσης του κάθε επιλεγμένου πλοίου σε ειδική καρτέλα εντός της εφαρμογής

  ΠΡΟΣΕΓΓΙΣΗ ΙΚΑΝΟΠΟΙΗΣΗΣ ΑΠΑΙΤΗΣΗΣ

  - Με την ίδια λογική των παραπάνω προαναφερθέντων hooks περνάμε και την πληροφορία της τελευταίας ληφθείσας θέσης στο συστατικό όπου την χρειάζεται ως είσοδο, οπότε η απαίτηση αυτή ικανοποιήθηκε με την ίδια τεχνική !

- η δυνατότητα διαγραφής των επιλεγμένων ανοιχτών trajectories.

  ΠΡΟΣΕΓΓΙΣΗ ΙΚΑΝΟΠΟΙΗΣΗΣ ΑΠΑΙΤΗΣΗΣ

- Η προσέγγιση έχει την εξής λογική εδώ κάνοντας χρήση ενός state με τον usestate hook χρησιμοποιούμε μια μεταβλητή ως ‘διακόπτη’ όπου όταν πατιέται το κουμπί γίνεται true, η ανανέωση αυτή της τιμής φέρει ως αποτέλεσμα εντός του useeffect hook να γίνει καθαρισμός των επιπλέον στρωμάτων που έχουν προστεθεί στον χάρτη ένα εκ των οποίων αποτελεί και το στρώμα των ομαδοποιημένων εγγραφών δηλαδή τα ανοιχτά ενεργοποιημένα trajectories !

3.1.2 Σύγκριση προσέγγισης με άλλες εφαρμογές ως προς τις απαιτήσεις

Οι εφαρμογές που υπάρχουν ήδη στην αγορά επειδή απευθύνονται σε αληθινό κοινό με μεγάλο αριθμό πελατών προσφέρουν πολύ περισσότερες υπηρεσίες και αλληλεπιδράσεις σε σύγκριση με την εφαρμογή της παρούσας εργασίας, όμως η παρούσα εργασία ικανοποιεί πλήρως τις απαιτήσεις για τις οποίες αρχικά υπολογίστηκε να ικανοποιεί ! Θα μπορούσαμε να κάνουμε μια σύντομη σύγκριση της δικής μας εφαρμογής με τις ήδη υπάρχουσες εκεί έξω αναφέροντας τα σημεία όπου υπερτερούν και οι μεν και οι δε..

<u>Πλεονεκτήματα εφαρμογών της αγοράς:</u>

1. Πληθώρα φίλτρων αναζήτησης
2. Μεγάλη βάση δεδομένων
3. Πραγματικός χρόνος παρακολούθησης
4. Υπολογισμοί στατιστικών μοντέλων
5. Υπολογισμοί μελλοντικών trajectories ή και routes
6. Εικόνες πλοίων
7. Υπολογισμός αγνώστων τιμών όπως άγνωστου προορισμού
8. Παρακολούθηση λιμανιών και φάρων
9. Κάλυψη περιοχής από κοντινά πλοία ή και κοντινούς φάρους
10. Υπολογισμός άφιξης σε λιμάνι
11. Διαγραφή επιλογών από το χάρτη
12. Επιλογή τύπων πλοίων προς αναπαράσταση στον χάρτη π.χ. μόνο general cargo, fishing, passenger κ.α.
13. Επιπλέον στρώματα χάρτη για καιρικά φαινόμενα
14. Σύγχρονη διεπαφή χρήστη και εμπειρία χρήστη

Πλεονεκτήματα εφαρμογής διπλωματικής:

1. Ιστορικά δεδομένα μη πραγματικός χρόνος παρακολούθησης αλλά δυνατότητα παρατηρήσεων από ιστορικές κινήσεις των πλοίων και συμπεράσματα της γενικής εικόνας των κινήσεών τους και των ρουτινών τους
2. Παρακολούθηση ιστορικής κατηγορίας δραστηριότητας που βοηθάει στα συμπεράσματα για τον τρόπο πλοήγησης των πλοίων
3. Αλλαγή χρώματος φόντου κάτι που δεν παρέχεται στις υπάρχουσες. Αυτό επιτυγχάνει την εξατομίκευση και την ευκολία προσαρμογής του χρήστη σε χρώματα της αρεσκείας του για πιο άμεση οικειοποίηση με το περιβάλλον
4. Αποθήκευση τελευταίας ληφθείσας θέσης του πλοίου στη μνήμη για σύγκριση με την τρέχουσα θέση
5. Διαγραφή επιλογών από το χάρτη
6. Επιλογή τύπων πλοίων προς αναπαράσταση στον χάρτη π.χ. μόνο general cargo
7. Σύγχρονη διεπαφή χρήστη και εμπειρία χρήστη

Παρατηρούμε ότι υπάρχουν μερικά κοινά της εφαρμογής που περιγράφουμε με τις ήδη υπάρχουσες στην αγορά με τη μόνη διαφορά ότι οι εφαρμογές της αγοράς είναι πιο εξειδικευμένες ακόμα και στα κοινά χαρακτηριστικά που εντοπίζονται από την παραπάνω καταμέτρηση των πλεονεκτημάτων των δύο περιπτώσεων εφαρμογών

3.2 Τεχνολογίες που χρησιμοποιήθηκαν και εργαλεία

Στην ενότητα που ακολουθεί θα αναφερθούμε στις τεχνολογίες που χρησιμοποιήθηκαν και στα εργαλεία που τις συνόδεψαν

3.2.1 Γνωριμία με τη React

Η react είναι μια javascript βιβλιοθήκη και όχι framework. Έχει αναπτυχθεί αρχικά το 2013 και συντηρείται από την ομάδα του facebook και το instagram, το ίδιο το facebook και το Netflix είναι κάποια από τα παραδείγματα εφαρμογών που έχουν αναπτυχθεί σε react.

Η φιλοσοφία της βιβλιοθήκης αυτής θέλει τα δεδομένα να μην πηγαίνουν προς δύο κατευθύνσεις (2-way data binding) παρά προς μία (unidirectional data flow). Αυτό σε γενικές γραμμές σημαίνει ότι

frameworks όπως η angular που βασίζονται στη λογική της αμφίδρομης επικοινωνίας από το view στο model και αντίστροφα (δηλαδή η ενημέρωση μιας τιμής στο view ενημερώνει ταυτόχρονα την τιμή και στο model και αντίστροφα) η react αυτό το αντικρούει και προτιμά την μονοδρομική επικοινωνία των δεδομένων δηλαδή από το model προς το View του οποίου αναλαμβάνει να ενημερώνει το state των μεταβλητών διαμέσου ενός virtual dom σε αντίθεση με την angular που ενημερώνει απευθείας το dom για τις οποιεσδήποτε αλλαγές! Το πλεονέκτημα σε αυτόν τον τρόπο επικοινωνίας είναι ότι κάθε φορά που θα ενημερωθεί το state μόνο τα συστατικά μέρη κάτω από εκείνο στην ιεραρχία όπου επηρεάστηκε θα επηρεαστούν και τα υπόλοιπα όπως τα sibling ή parent components δεν θα επηρεαστούν από τις αλλαγές! Η react έχει τη δυνατότητα να συνδυάζεται τέλεια και με άλλες βιβλιοθήκες για να μπορέσει να ολοκληρωθεί και να αποτελέσει μια δελεαστική εξίσου επιλογή όπως ένα framework! Εν ολίγοις αγκαλιάζει την έννοια των συστατικών μερών (components) αλλά μόνο στη μεριά του View σε αντίθεση με την angular ή άλλα frameworks..

Υποστηρίζει τη μορφή jsx αρχείων που δεν είναι τίποτα άλλο παρά αρχεία υβριδικά με html και javascript, οπότε όλο το component στην ουσία δημιουργείται εντός ενός αρχείου μορφής jsx.

Ορίζει την έννοια των functional (stateless) components που στην ουσία δίνει τη δυνατότητα δημιουργίας components βασισμένα σε απλές functions που απλώς βασίζονται στα expressions και declarations και όχι στην έννοια του διαμοιρασμού κοινού state όπως στον αντικειμενοστραφή προγραμματισμό.

Η react χρησιμοποιεί μια καινούρια έννοια την έννοια των hooks για τη διαχείριση των τιμών των μεταβλητών και χρειάζεται επιπλέον τεχνολογίες για να υποστηρίξει την έννοια του dependency injection που με λίγα λόγια είναι μια τεχνική όπου ένα αντικείμενο λαμβάνει συνήθως μέσω των constructors ένα άλλο αντικείμενο από το οποίο βασίζεται. Η react επίσης ορίζει άλλη μία έννοια για τη μεταφορά δεδομένων προς άλλα συστατικά μέρη κυρίως και αυτή είναι το context api που παρέχει και έχουμε πάρει μια ιδέα πως περίπου δημιουργείται και χρησιμοποιείται στα snippets κώδικα που παραθέσαμε προηγουμένως ! Το virtual dom που έχει την κάνει τον νικητή σε θέματα απόδοσης (performance) μεταξύ άλλων τεχνολογιών για την ανάπτυξη frontend εφαρμογών. Ο λόγος είναι διότι το visrtual αυτό dom φορτώνει συγκεκριμένα κομμάτια της εφαμογής. Επίσης επειδή είναι μια βιβλιοθήκη και το μόνο που χρειάζεται κανείς να γνωρίζει για να αρχίσει να δημιουργεί εφαρμογές σε αυτήν είναι html και javascript είναι μια καλή αρχή ! (Yoshitaka Shiotsu, 2020)

3.2.1.1 Γιατί React?

Με βάση την πρώτη μας γνωριμία με τις δεξιότητες της react από την προηγούμενη παράγραφο, σε αυτήν την παράγραφο θα εστιάσουμε περισσότερο στο γιατί έγινε η επιλογή αυτής της βιβλιοθήκης και όχι κάποιου άλλου γνωστού framework όπως angular ή vue..

Πρέπει να χρησιμοποιούμε τη react όταν θέλουμε να …

Έχουμε βαριά γραφικά περιβάλλοντα στη μεριά του χρήστη ή και χειρισμούς του dom που απαιτούν βαρύς υπολογισμούς, ή όταν προτιμούμε να έχουμε functional προγραμματισμό στην υλοποίηση των εφαρμογών μας και επικοινωνία δεδομένων προς μία μόνο κατεύθυνση σε περίπτωση που θέλουμε απομόνωση κάποιων συστατικών μερών από την ενημέρωση του state των μεταβλητών.

Πλεονεκτήματα της react που μας κάνουν να την επιλέγουμε για τους παραπάνω λόγους:

- Flexibility: Η React δίνει το στρώμα του view και αφήνει την επιλογή σε μας να αποφασίσουμε τι θα κάνουμε με την επιλογή router και σύστημα διαχείρησης του state όπου θέλουμε να χρησιμοποιήσουμε.
- Virtual DOM: Κάνει τις αλλαγές πρώτα σε μια αναπαράσταση του αυθεντικού DOM ώστε να γίνουν οι αλλαγές μόνο στα κομμάτια του αυθεντικού DOM που χρειάζονται αλλαγή (αντί του να ενημερώνει ολόκληρη την σελίδα).
- Functional components: Απομονώνει τη διαχείρηση του state από το όλο component για να δημιουργήσει απλές functions (που συχνά αποκαλούνται και ως stateless components).
- Component reuse: Της React η μείξη από κλάσεις και functional components δίνει περισσότερο flexibility στο βαθμό στον οποίο τα components μπορούν να ξαναχρησιμοποιηθούν.
- Downward data flow: Απλοποιεί τη διαχείρηση του state ή αλλιώς της κατάστασης των μεταβλητών στις εφαρμογές που απαιτούν βαριές λειτουργίες στο DOM με τη φιλοσοφία του one-way data binding.
- Thriving community: Μεγάλη κοινότητα με open-source developers και projects ώστε να γίνει πιο εύκολη η επιλογή βιβλιοθηκών και εργαλείων για κάθε ανάγκη της εφαρμογής που θα προκύψει ή ακόμα και για εφαρμογές κινητών.
- React Developer Tools: Έχει μία εύχρηστη Chrome DevTools επέκταση (Yoshitaka Shiotsu, 2020)

Σε αντίθεση η angular κάνει τις εφαρμογές πιο βαριές λόγω των πολλών features που έχει και τα οποία μπορούν να εμποδίσουν την απόδοση, καθώς γνωρίζουμε πως βαριές εφαρμογές = αργή απόδοση σε σύγκριση με τη react ή τη vue ! Επίσης αναβαθμίζεται συνεχώς και το

γεγονός αυτό την καθιστά πιο δύσκολη στο να την ακολουθήσει κανείς και να τη μάθει αν θέλει να ξεκινήσει με angular (Piero Borrelli, 2021)

Αυτό δε σημαίνει ότι δεν αξίζει σαν framework διότι η επιλογή τεχνολογίας κάθε φορά πρέπει να γίνεται με βάση και άλλα πιο τεχνικά ή και πιο αρχιτεκτονικά δεδομένα !

Σε αντίθεση η react είναι προορισμένη για να είναι γρήγορη, απλή και επεκτάσιμη, είναι μια καλή επιλογή αν θέλουμε ταχύτητα και απόδοση στις εφαρμογές μας, επίσης δεν χρειάζεται κάτι ιδιαίτερο για να προσαρμοστεί σε μια νέα πλατφόρμα, σημαντικό παράγοντα παίζει και το ότι έχει καλές επιδόσεις σε κατατάξεις SEO της google παρέχοντας έτσι καλύτερη εμπειρία χρήστη. Ο λόγος της καλής κατάταξης της εφαρμογής μας είναι επειδή αν η εφαρμογή ή η ιστοσελίδα μας είναι αποδοτική και γρήγορη αυτό παίρνει υψηλούς βαθμούς κατάταξης και κάνει την εφαρμογή μας να εμφανίζεται στα πρώτα αποτελέσματα του google ! (Piero Borrelli, 2021)

3.2.2 Leaflet

Το Leaflet είναι μια βιβλιοθήκη JavaScript ανοιχτού κώδικα που χρησιμοποιείται για τη δημιουργία εφαρμογών χαρτογράφησης ιστού. Κυκλοφόρησε για πρώτη φορά το 2011, υποστηρίζει τις περισσότερες πλατφόρμες για κινητά και επιτραπέζιους υπολογιστές, υποστηρίζοντας HTML5 και CSS3. (wikipedia, 2021b)

3.2.3 Python

Η Python είναι διερμηνευόμενη (interpreted), γενικού σκοπού (general-purpose) και υψηλού επιπέδου, γλώσσα προγραμματισμού. Ανήκει στις γλώσσες προστακτικού προγραμματισμού (Imperative programming) και υποστηρίζει τόσο το διαδικαστικό (procedural programming) όσο και το αντικειμενοστραφές (object-oriented programming) προγραμματιστικό υπόδειγμα (programming paradigm). Είναι δυναμική γλώσσα προγραμματισμού (dynamically typed) και υποστηρίζει συλλογή απορριμμάτων (garbage collection ή GC). Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της. Το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα από ότι θα ήταν δυνατόν σε γλώσσες όπως η C++ ή η Java.

Διακρίνεται λόγω του ότι έχει πολλές βιβλιοθήκες που διευκολύνουν ιδιαίτερα αρκετές συνηθισμένες εργασίες και για την ταχύτητα εκμάθησης της. Μειονεκτεί στο ότι επειδή είναι διερμηνευόμενη είναι πιο αργή από τις μεταγλωττιζόμενες (compiled) γλώσσες όπως η C και η C++. Για αυτόν τον λόγο δεν είναι κατάλληλη για γραφή λειτουργικών συστημάτων.(wikipedia, 2021)

3.2.4 WebSockets

Το WebSocket είναι μια μόνιμη/συνεχής σύνδεση μεταξύ πελάτη και διακομιστή. Τα WebSockets παρέχουν ένα αμφίδρομο, πλήρες διπλό/ταυτόχρονο κανάλι επικοινωνίας που λειτουργεί πάνω στο HTTP μέσω μιας σύνδεσης TCP / IP. Στον πυρήνα του, το πρωτόκολλο WebSocket διευκολύνει τη μετάδοση μηνυμάτων μεταξύ πελάτη και διακομιστή. (Kevin Sookocheff, 2019)

Τα WebSockets δεν χρησιμοποιούν το σχήμα http: // ή https: // (επειδή δεν ακολουθούν το πρωτόκολλο HTTP). Αντίθετα, τα URI των WebSocket χρησιμοποιούν ένα νέο σχήμα ws: (ή wss: για ένα ασφαλές κανάλι WebSocket). Το υπόλοιπο URI είναι το ίδιο με το HTTP URI (Kevin Sookocheff, 2019)

Τα WebSockets ξεκινούν τη ζωή ως ένα τυπικό αίτημα/απάντηση HTTP. Μέσα σε αυτήν την ακολουθία αίτησης – απόκρισης, ο πελάτης ζητά να ανοίξει μια σύνδεση WebSocket και ο διακομιστής αποκρίνεται (εάν είναι σε θέση). Εάν αυτή η αρχική χειραψία είναι επιτυχής, ο πελάτης και ο διακομιστής έχουν συμφωνήσει να χρησιμοποιήσουν την υπάρχουσα σύνδεση TCP / IP που δημιουργήθηκε για το αίτημα HTTP ως σύνδεση WebSocket. Τα δεδομένα μπορούν τώρα να ρέουν μέσω αυτής της σύνδεσης. Μόλις αμφότερα οι δύο πλευρές αναγνωρίσουν ότι η σύνδεση WebSocket θα πρέπει να κλείσει, η σύνδεση TCP διαλύεται. (Kevin Sookocheff, 2019)

Οι συνδέσεις WebSocket δημιουργούνται αναβαθμίζοντας ένα ζεύγος αιτήσεων / απόκρισης HTTP Ένας πελάτης που υποστηρίζει WebSockets και θέλει να δημιουργήσει μια σύνδεση θα στείλει ένα αίτημα HTTP που περιλαμβάνει μερικές απαιτούμενες κεφαλίδες:

Μαζί, αυτές οι κεφαλίδες θα είχαν ως αποτέλεσμα ένα αίτημα HTTP GET από τον πελάτη σε ένα ws: // URI όπως στο ακόλουθο παράδειγμα:

```
GET ws://example.com:8181/ HTTP/1.1
Host: localhost:8181
Connection: Upgrade
Pragma: no-cache
```

Cache-Control: no-cache
Upgrade: websocket
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: q4xkcO32u266gldTuKaSOw==
(Αυτή είναι μια εφάπαξ τυχαία τιμή (nonce) που δημιουργείται από τον πελάτη.)
Sec-WebSocket-Version: 13

Μόλις ένας πελάτης στείλει το αρχικό αίτημα για άνοιγμα μιας σύνδεσης WebSocket, περιμένει την απάντηση του διακομιστή. Η απάντηση πρέπει να έχει έναν κωδικό απόκρισης HTTP 101 Switching Protocols.
(Kevin Sookocheff, 2019)

3.2.4.1 Γιατί WebSockets?

Η ιδέα του WebSocket γεννήθηκε από τους περιορισμούς της τεχνολογίας που βασίζεται το HTTP. Με το HTTP, ένας πελάτης ζητά έναν πόρο και ο διακομιστής αποκρίνεται με τα ζητούμενα δεδομένα. Το HTTP είναι ένα αυστηρά μονοκατευθυντικό πρωτόκολλο - τυχόν δεδομένα που αποστέλλονται από τον διακομιστή στον πελάτη πρέπει πρώτα να ζητηθούν από τον πελάτη. Το Long-polling λειτούργησε παραδοσιακά ως λύση για αυτόν τον περιορισμό. Με το Long-polling, ένας πελάτης υποβάλλει ένα αίτημα HTTP με μεγάλο χρονικό διάστημα (timeout) και ο διακομιστής χρησιμοποιεί αυτό το μεγάλο χρονικό όριο για να προωθήσει δεδομένα στον πελάτη. Το Long-polling λειτουργεί πολύ καλά, αλλά συνοδεύεται από το εξής μειονέκτημα : οι πόροι στο διακομιστή δεσμεύονται καθ 'όλη τη διάρκεια του Long-polling, ακόμη και όταν δεν υπάρχουν διαθέσιμα δεδομένα για αποστολή! (Kevin Sookocheff, 2019)

Τα WebSockets, από την άλλη πλευρά, επιτρέπουν την αποστολή δεδομένων βάσει μηνυμάτων, παρόμοια με το UDP, αλλά με την αξιοπιστία του TCP. Το WebSocket χρησιμοποιεί το HTTP ως τον αρχικό μηχανισμό μεταφοράς, αλλά διατηρεί τη σύνδεση TCP ζωντανή μετά τη λήψη της απόκρισης του HTTP, ώστε να μπορεί να χρησιμοποιηθεί για την αποστολή μηνυμάτων μεταξύ πελάτη και διακομιστή. Τα WebSockets μας επιτρέπουν να δημιουργούμε εφαρμογές "σε πραγματικό χρόνο" χωρίς τη χρήση long-polling .(Kevin Sookocheff, 2019)

3.2.5 VsCode

Για την ανάπτυξη οποιασδήποτε εφαρμογής, λογισμικού ή συστήματος βασικό ρόλο έχει και η επιλογή ενός κατάλληλου editor. Για την ανάπτυξη λοιπόν της παρούσας εφαρμογής ως editor επιλέχτηκε το visual studio code της Microsoft.

Το visual studio code είναι ένας editor ανοικτού λογισμικού που μπορεί να χρησιμοποιηθεί με μια μεγάλη ποικιλία γλωσσών προγραμματισμού και έχει δημιουργηθεί από τη Microsoft για πλατφόρμες Windows, Linux και macOS και είναι γραμμένο σε typescript, javascript και css. (“Visual Studio Code,” 2019) Υποστηρίζει δικό του debugger με breakpoints, call stacks και διαδραστική κονσόλα. Υποστηρίζει επίσης snippets, code refactoring, syntax highlighting, autocomplete και με χρήση IntelliSense πηγαίνει ένα βήμα παραπάνω στο να παρέχει έξυπνες συμπληρώσεις κώδικα βασισμένες σε τύπους μεταβλητών, ορισμούς μεθόδων και προστιθέμενων modules. (Microsoft, 2019) Επιπλέον μπορεί να προσθέσει κι άλλη λειτουργικότητα με την εγκατάσταση διαφόρων extensions. (“Visual Studio Code,” 2019) Στα πλαίσια της εφαρμογής αυτής έχει γίνει εγκατάσταση ενός extension και συγκεκριμένα του tslint ενός extension που βοηθάει για τη διόρθωση του συντακτικού, προτείνοντας λύσεις. Τέλος παρέχει και ενσωματωμένο terminal για άμεση εκτέλεση εντολών του node το οποίο χρησιμοποιήθηκε για την εφαρμογή αυτή. Σε έρευνα που διεξάγει στο StackOverflow.com το 2019 το visual studio code ψηφίστηκε ως το πιο δημοφιλές εργαλείο περιβάλλον ανάπτυξης κώδικα με 50.7% στους 87.317 χρήστες να ισχυρίζονται ότι το χρησιμοποιούν. (“Visual Studio Code,” 2019)

3.2.6 Our Browser Google Chrome and its debug tools

Το frontent της εφαρμογής έτρεχε σε έναν browser και γι’ αυτό το σκοπό χρησιμοποιήθηκε το google chrome. Επίσης το debugging του κώδικα από μεριάς frontent έγινε στον εξαιρετικά γραμμένο και ικανό debugger του chrome.

4 ΛΕΠΤΟΜΕΡΕΙΕΣ ΣΥΣΤΗΜΑΤΟΣ (Υλοποίηση)

4.1 Εγκατάσταση απαραίτητων τεχνολογιών και εργαλείων

4.1.1 Εγκατάσταση NodeJS και React

Για να εγκαταστήσουμε το NodeJS επισκεπτόμαστε την ιστοσελίδα του https://nodejs.org/en/download/ και κατεβάζουμε την έκδοση "Recommended For Most Users". Έπειτα ανοίγουμε το κατεβασμένο αρχείο και ακολουθούμε τα βήματα στην οθόνη για την εγκατάσταση. Μόλις ο installer τελειώσει την εγκατάσταση ανοίγουμε το Node.js command prompt που θα βρούμε στον φάκελο εγκατάστασης του NodeJS. Σε αυτόν τον command prompt μπορούμε γράφοντας και τρέχοντας την εντολή

node –v

να δούμε την τρέχουσα εγκατεστημένη έκδοση που μόλις εγκαταστήσαμε.

Σε περίπτωση που έχουμε το NodeJS αυτό που πρέπει να κάνουμε είναι να ελέγξουμε την εγκατεστημένη έκδοση με την εντολή που προαναφέρθηκε και αν θέλουμε τα εγκαταστήσουμε μια νεότερη θα πρέπει να διαγράψουμε πλήρως την ήδη εγκατεστημένη. Για χρήστες των Windows η διαγραφή θα πρέπει να γίνει πέρα από τον πίνακα ελέγχου και από τα μονοπάτια:

C:\Users\User_Name\AppData\Roaming\npm
C:\Users\ User_Name \ AppData \Roaming\npm-cache

Επόμενο βήμα η εγκατάσταση του editor…

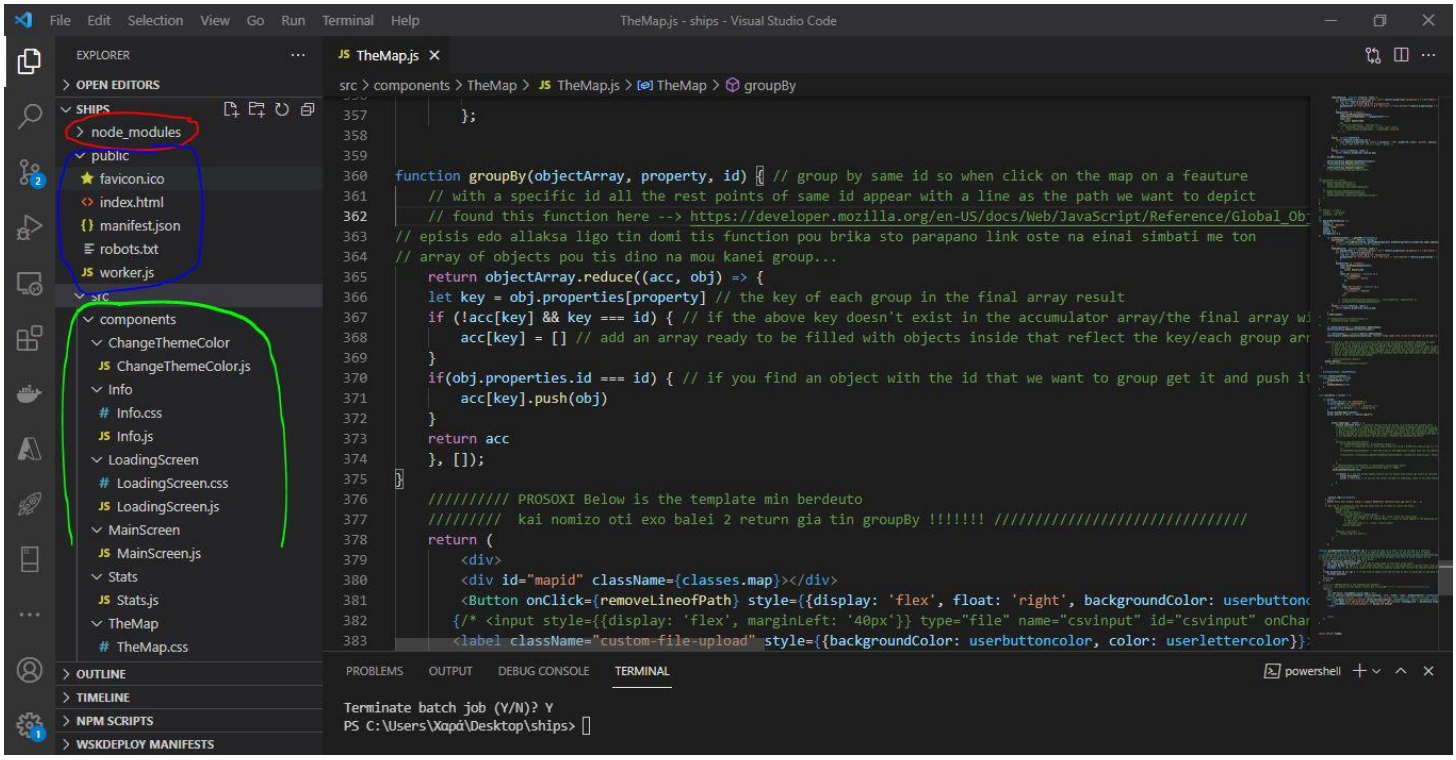
4.1.2 Εγκατάσταση Visual Studio

Για την εγκατάσταση του Visual Studio Code πηγαίνουμε στην ιστοσελίδα της Microsoft https://code.visualstudio.com/download

και από εκεί κατεβάζουμε την έκδοση που μας ενδιαφέρει. Στη συνέχεια εγκαθιστούμε από το εκτελέσιμο αρχείο που κατεβάσαμε τον editor ακολουθώντας τα βήματα στην οθόνη. Μόλις η εγκατάσταση ολοκληρωθεί μπορούμε να ανοίξουμε τον editor μας και να αρχίσουμε να εξοικειωνόμαστε μαζί του.

4.2.2.1 Εξοικείωση και γνωριμία με τον editor

Η δημιουργία νέου project που θα δούμε παρακάτω Βήμα – βήμα επιφέρει την τελική εμφάνιση του editor μας να μοιάζει με αυτήν της εικόνας.



Η react λοιπόν δημιουργεί την παραπάνω δομή του project μας εντός του editor μετά την εντολή δημιουργίας του στο cli του node

Στα αριστερά της εικόνας με κόκκινη υπογράμμιση βλέπουμε τον φάκελο με όλα τα επιπλέον modules που έχουμε συμπεριλάβει στο project μας ώστε να λειτουργεί η εφαρμογή μας ! αυτός ο φάκελος πρέπει να δημιουργείται πάντα πριν ξεκινήσουμε την εφαρμογή μας όπως θα αναλύσουμε και στην ακριβώς επόμενη παράγραφο ! Η επόμενη ακριβώς υπογράμμιση που αποτυπώνεται με μπλε χρώμα δείχνει τον public φάκελό μας τον οποίο μπορούμε να κάνουμε deploy σε κάποια υπηρεσία υπολογιστικού νέφους αφού πρώτα κάνουμε build το project μας και τέλος η πράσινη υπογράμμιση δείχνει τη δομή των συστατικών μερών της εφαρμογής μας που από όσο βλέπουμε η react έχει μια

εύκολη φιλοσοφία στη δομή αυτή καθώς κάθε φάκελος αντιπροσωπεύει και από ένα συστατικό μέρος της εφαρμογής μας όπου τα αρχεία μέσα σε κάθε φάκελο είναι της μορφής jsx και περιέχουν και τον κώδικα του view και του model εντός των javascript αρχείων ! τα css αρχεία γίνονται αν χρειαστούν import στα javascript αρχεία κανονικά !

4.2 Δημιουργία νέου Project

Αφού έχουμε στη διάθεσή μας εγκατεστημένα τα node + npm μπορούμε να τρέξουμε την παρακάτω εντολή

```
npx create-react-app shipstracking
```

έπειτα θα πρέπει να μεταφερθούμε μέσα στο φάκελο που δημιουργήθηκε για το project με την εντολή:

```
cd shipstracking
```

επίσης επειδή χρησιμοποιούμε το material UI που είναι ειδικά για την react ανεπτυγμένο πρέπει να εκτελέσουμε και τις παρακάτω δύο εντολές ! Μπορούμε να εκτελούμε αυτές τις εντολές και στον ενσωματωμένο terminal εντός του editor !

```
npm install @material-ui/core --save
```

```
npm install @material-ui/icons –save
```

κάθε φορά που εγκαθιστάμε κάτι μέσα από το πακέτο διαχείρισης του node (npm) καλό είναι να προσθέτουμε και το flag –save όπως φαίνεται και στις παραπάνω εντολές ώστε να γίνεται μια προσθήκη του τι μόλις εγκαταστήσαμε στο αρχείο package.json που μπορεί να φανεί χρήσιμο εάν θέλουμε να μεταφέρουμε την εφαρμογή μας σε άλλο μηχάνημα και πρέπει να εγκαταστήσουμε ξανά όλα τα πακέτα από την αρχή ! σε τέτοια περίπτωση με την εντολή npm install και αν είμαστε εντός του φακέλου όπου τρέχουμε το project θα ξεκινήσει να διαβάζει αυτό το αρχείο και αυτόματα θα αρχίσει να εγκαθιστά όλα τα πακέτα που έχουμε εγκαταστήσει μεμονωμένα και έχουμε καταχωρήσει σε αυτό το αρχείο !

όταν ολοκληρώσουμε την υλοποίηση της εφαρμογής σε ένα καλό σημείο μπορούμε να την τρέξουμε με την παρακάτω εντολή και να την δούμε στο browser στη διεύθυνση http://localhost:3000/

```
npm start
```

4.3 Περιγραφή Αλγορίθμων που χρησιμοποιήθηκαν

4.3.1 Η συνάρτηση reduce

Η μέθοδος reduce() εκτελεί μια function reducer (που παρέχετε) σε κάθε στοιχείο του πίνακα, με αποτέλεσμα μια τιμή εξόδου.

Η συνάρτηση reduce() παίρνει τέσσερα ορίσματα:

- Accumulator
- Current Value
- Current Index
- Source Array

Η επιστρεφόμενη τιμή της συνάρτησης reduce() εκχωρείται στον accumulator, του οποίου η τιμή απομνημονεύεται σε κάθε επανάληψη σε ολόκληρο τον πίνακα και τελικά γίνεται η τελική, μεμονωμένη τιμή που προκύπτει. (developer.mozilla, n.d.)

4.3.2 Η συνάρτηση find

Η μέθοδος find() επιστρέφει την τιμή του στοιχείου του πίνακα που περνά έναν έλεγχο (που παρέχεται από μια συνάρτηση).

Η μέθοδος εκτελεί τη συνάρτηση μία φορά για κάθε στοιχείο που υπάρχει στον πίνακα:

Εάν εντοπίσει ένα στοιχείο του πίνακα όπου η συνάρτηση επιστρέφει μια λογική τιμή αληθείας ‘true’, η find() επιστρέφει την τιμή αυτού του στοιχείου του πίνακα (και δεν ελέγχει τις υπόλοιπες τιμές)

Διαφορετικά επιστρέφει ‘undefined’ ! (w3schools, n.d.)

4.4 Σημεία κώδικα και λογικής που αξίζουν να αναφερθούν

Σε αυτήν την παράγραφο θα αναφερθούμε σε διάφορα σημεία του κώδικα που αξίζει να επισημάνουμε τις επιλογές αλγορίθμων ή και τεχνικών, μεθόδων που επιλέχθηκαν και θα εξηγήσουμε εν συντομία λίγα πράγματα για το κάθε σημείο που φαίνεται χρήσιμο !

4.4.1 Η υλοποίηση της reduce

```
function groupBy(objectArray, property, id) {
    return objectArray.reduce((acc, obj) => {
    let key = obj.properties[property]

    if (!acc[key] && key === id) {
       acc[key] = []
    }
    if(obj.properties.id === id) {
        acc[key].push(obj)
    }
    return acc
    }, []);
}
```

Απόσπασμα 8: υλοποίηση της reduce code snippet

Η group by παίρνει όλες τις εγγραφές ως $1^{ο}$ της όρισμα παίρνει δηλαδή έναν array of objects ως $2^{ο}$ όρισμα παίρνει ένα αλφαριθμητικό συγκεκριμένα την τιμή 'id' ως τιμή string ώστε να δηλώσουμε ότι θέλουμε να κάνουμε ομαδοποίηση γύρω από τα id και ως $3^{ο}$ όρισμα την τιμή του id σε μορφή αριθμού όπου έχει το πλοίο το οποίο έγινε κλικ στον χάρτη ώστε να εμφανίσει όλες του τις θέσεις. Η μέθοδος αυτή θα επιστρέψει το αποτέλεσμα της reduce η οποία εφαρμόζεται πάνω στον array of objects που έχει όλες τις εγγραφές ώστε να ψάξει να βρει τις υπόλοιπες εγγραφές με ίδιο id με αυτό που στάλθηκε στην group by και να επιτευχθεί η ομαδοποίηση που θέλουμε με όλες τις θέσεις ενός πλοίου ! Μετά από κάθε εγγραφή από την οποία περνάει η reduce αποθηκεύει στην μεταβλητή key την τιμή που έχει ως id η εγγραφή που κοιτά εκείνη τη στιγμή και εάν δεν υπάρχει τέτοιο στοιχείο με τέτοια τιμή δείκτη στον πίνακα acc τότε δημιουργεί αυτόν τον πίνακα με αυτό τον δείκτη ο οποίος θα συσσωρεύσει όλες τις θέσεις σιγά σιγά που θα βρεθούν για το συγκεκριμένο id πλοίου ! έτσι μετά πάει και βάζει το στοιχείο με την push στη θέση όπου ο δείκτης του acc ταυτίζεται με το id της εγγραφής η οποία ερευνάται από τη reduce αυτή τη στιγμή και ταυτίζεται κιόλας και με το εισερχόμενο id του οποίου ψάχνουμε όλες τις εγγραφές ώστε να γίνει η ομαδοποίηση και να αναπαρασταθεί αυτός ο πίνακας ως ένα ενιαίο polyline σε ξεχωριστό στρώμα πάνω στον χάρτη !

4.4.2 Η υλοποίηση της find

```
var foundfeaturewithsameid = batch_features.find(f => {
                    return (f.properties.id === parseInt(row_values[0
]) && f.properties.show_on_map === true);
                });
```

Απόσπασμα 9: υλοποίηση της find code snippet

Ψάχνοντας μέσα σε ένα κομμάτι με αρκετές εγγραφές η find δέχεται μια arrow function ως όρισμα η οποία αν βρει σε ψάχνοντας σε κάθε εγγραφή f (η κάθε εγγραφή φέρει πάνω της όλα τα properties που έχουμε για μια συγκεκριμένη θέση του πλοίου) το id της θέσης f είναι ίδιο με το 1^{ο} στοιχείο το οποίο είναι το id της τρέχουσας εγγραφής που διαβάζεται και το πλοίο στη θέση f έχει ήδη ενεργοποιημένη την ιδιότητα/property show_on_map σε τιμή 'true' τότε μιας που βρήκαμε νέα θέση του ίδιου πλοίου (αφού ισχύει η συνθήκη των ίδιων id που εξετάστηκε προηγουμένως) κάνει στην πορεία σε κώδικα που δεν παραθέτουμε false την παλιά τιμή ώστε να ανανεωθεί και στον χάρτη η νέα ληφθείσα θέση !

5 ΑΞΙΟΛΟΓΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αξιολογήσουμε την εφαρμογή αναφερόμενοι σε σενάρια εκτέλεσης. Θα αποτυπώσουμε την επιτυχία ικανοποίησης των απαιτήσεων καθώς και αντικειμενικών στόχων που είχαν τεθεί πριν την υλοποίηση

5.2 Σενάρια Εκτέλεσης

5.2.1 Αρχικές συνθήκες και στόχοι σεναρίων εκτέλεσης

Μόλις η εφαρμογή είχε φτάσει σε τελικό στάδιο υλοποιήθηκε 1 σενάριο εκτέλεσης, λόγω των χαμηλών εξ αρχής απαιτήσεων που χρειάστηκε να υλοποιηθούν, οπότε δεν υπήρχε επιρρέπεια στα λάθη.

Οι στόχοι των σεναρίων είναι να ελεγχθεί η ανταπόκριση σε πραγματικά δεδομένα που είχε η εφαρμογή και κατά πόσο τα εξυπηρετεί. Επίσης να φανερωθούν πιθανά κενά στη λογική του κώδικα και να διορθωθούν. Επιπλέον να δούμε κατά πόσο αυτή η εφαρμογή ικανοποιεί τις απαιτήσεις που τέθηκαν στην αρχή λειτουργικές καθώς και μη λειτουργικές.

Στο πρώτο και μοναδικό σενάριο εκτέλεσης παρατηρήθηκε ότι υπήρχε καθυστέρηση ανταπόκρισης του χάρτη από τον τρόπο υλοποίησης της μεταφοράς δεδομένων από την τότε επιλεγμένη πηγή μεταφοράς δεδομένων κι έτσι επιλέχθηκαν τα websockets ως εναλλακτική για τους λόγους που αναφέρθηκαν στην αντίστοιχη παράγραφο. Η αρχική συνθήκη που θα έπρεπε να εφαρμοστεί είναι η απόκριση της εφαρμογής σε αυτήν την περίπτωση και το πρώτο σενάριο εκτέλεσης εντόπισε αυτήν την ανάγκη αλλαγής τρόπου υλοποίησης της πηγής.

5.2.2 Ικανοποίηση απαιτήσεων και αντικειμενικών στόχων από το σύστημα

Στις λειτουργικές απαιτήσεις που εφαρμόστηκαν επιτυχώς στην εφαρμογή μας συνοψίζοντας έχουμε:

- Διαγραφή Επιλεγμένων ανοιχτών θαλάσσιων μονοπατιών πάνω στο χάρτη
- Πάτημα σε θέση πλοίου και εμφάνιση λεπτομερειών πλοίου σε αναδυόμενο παράθυρο
- Χρωματικός τονισμός ιδίων συνεχόμενων ετικετών δραστηριότητας
- Επιλογή τύπου πλοίων προς εμφάνιση στον χάρτη από μενού εντός του χάρτη
- Δυνατότητα ζουμ στον χάρτη
- Ψευδοπραγματικός χρόνος αναπαράστασης δεδομένων
- Αποθήκευση τελευταίας ληφθείσας θέσης επιλεγμένου πλοίου σε ειδική περιοχή εντός της εφαρμογής

Στις μη λειτουργικές/δευτερεύουσες έχουμε:

- Σύγχρονο UI
- Αλληλεπίδραση με το θέμα φόντου
- Αλληλεπίδραση με το χάρτη

Η εφαρμογή καταφέρνει να ικανοποιήσει όλες τις παραπάνω απαιτήσεις

5.2.3 Σύνοψη ικανοποίησης αντικειμενικών στόχων (checklist)

Ανταπόκριση σε ιστορικά δεδομένα σε ψευδοπραγματικό χρόνο	**ΝΑΙ**
Σύγχρονη εμφάνιση	ΝΑΙ
Απλό design	ΝΑΙ
Αλληλεπίδραση με το χρήστη	ΝΑΙ
Χρηστικότητα	ΝΑΙ

6 ΣΥΜΠΕΡΑΣΜΑΤΑ

6.1 Σύνοψη της όλης υλοποίησης και αποτελέσματα επιτυχίας

Με λίγα λόγια φτάνοντας στο τέλος αυτής της διπλωματικής θα αναφερθούμε στο τι υλοποιήθηκε και αν κατάφερε να ικανοποιήσει τις απαιτήσεις με επιτυχία.

Οι απαιτήσεις χωρίζονται σε δύο επίπεδα τις λειτουργικές και τις μη λειτουργικές όσο οι πρώτες τόσο και οι δεύτερες είναι εξίσου σημαντικές και η εφαρμογή κατάφερε να εκπληρώσει όλες τις απαραίτητες απαιτήσεις που έχουν προαναφερθεί και σε προηγούμενη ενότητα.

Πιο συγκεκριμένα, η εφαρμογή επιτρέπει αλληλεπίδραση με το χρήστη τόσο σε επίπεδο χάρτη (ενέργειες όπως hover πάνω από τα πλοία και εμφάνιση του ονόματός τους, κλικ σε κάποια θέση και εμφάνιση των μεταδεδομένων του πλοίου και εμφάνιση όλων των θέσεων, χρωματικός τονισμός ιδίων συνεχόμενων θέσεων με ίδιες συνεχόμενες ετικέτες δραστηριότητας πλοίου, κλπ) όσο και σε επίπεδο εκτός χάρτη (ενέργειες όπως άνοιγμα αριστερής navigation bar, επιλογή χρώματος φόντου για εξατομίκευση, πέρασμα τελευταίας θέσης πλοίου σε ειδική καρτέλα στα δεξιά της εφαρμογής)

Οι παραπάνω ενέργειες έχουν υλοποιηθεί με αλγορίθμους και μεθόδους που βοηθούν στην άμεση απόκριση της εφαρμογής, και απόδοση

Οι τεχνολογίες που χρησιμοποιήθηκαν εξασφαλίζουν μια ομαλή εκτέλεση της εφαρμογής για απαιτητικές γραφικές αναπαραστάσεις στον client κάνοντας την εφαρμογή ελαφριά και αποδοτική.

6.2 Μελλοντική αναβάθμιση της εφαρμογής (επιπλέον λειτουργικότητα)

Όπως κάθε νέο λογισμικό έτοιμο προς διάθεση στην αγορά ποτέ δε σταματά να εξελίσσεται και να αναβαθμίζεται από τους δημιουργούς του έτσι και στην περίπτωση αυτής της εφαρμογής σκοπός είναι η εξέλιξή της με την προσθήκη μελλοντικών προσδοκώμενων χαρακτηριστικών περισσότερων επιλογών λειτουργικότητας.

Τα χαρακτηριστικά αυτά τα οποία προβλέπεται να προστεθούν στο μέλλον είναι :

- Προσθήκη υπολογισμού μελλοντικού θαλάσσιου μονοπατιού έως τον τελικό προορισμό του πλοίου

- Προσθήκη επιπλέον στρώματος γραφικής αναπαράστασης κύματος τριών διαστάσεων σε πανοραμική ημιτονοειδή κάτοψη με χρωματικό τονισμό σε σημεία όπου το κύμα είναι πιο μεγάλο
- Υπολογισμός και πρόβλεψη τιμών όπου δεν έχει γίνει σωστή λήψη του σήματός τους από το σύστημα όπως π.χ. τιμές unknown destination και unknown heading κλπ..
- Επιπλέον στρώμα εμφάνισης ανέμων
- Περισσότερα φίλτρα αναζήτησης
- Δυνατότητα παροχής πιθανολογικών μοντέλων για πρόβλεψη μελλοντικών καιρικών συνθηκών

6.3 Συνεισφορά στο ευρύ κοινό

Σε αυτήν την παράγραφο θα αναφερθούμε σε πόσα επίπεδα μια τέτοια εφαρμογή θα μπορούσε να φανεί χρήσιμη και πως. Αρχικά σε ατομικό επίπεδο και στη συνέχεια θα αναφερθούμε στις θετικές συνέπειες που η χρήση της θα μπορούσε να επιφέρει σε ένα ευρύτερο φάσμα όπως η κοινωνία.

Από άποψη προσωπικής χρήσης η συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί για την παρακολούθηση των ιστορικών θέσεων ενός ιδιοκτήτη πλοίου και να μπορέσει να βγάλει συμπέρασμα για την κίνηση του πλοίου του. Επίσης μπορεί να τη χρησιμοποιήσει και για να μπορέσει να δει σε σύνοψη όλα τα μεταδεδομένα τα οποία το πλοίο του στέλνει σε φορείς όπου λαμβάνουν αυτά τα δεδομένα.

Από άποψη μη προσωπικής χρήσης σε ένα πιο ευρύ κοινό αυτή η εφαρμογή θα μπορούσε να χρησιμοποιηθεί για λόγους ανάλυσης και παρακολούθησης όλων των πλοίων τύπου general cargo στην περιοχή της μεσογείου και να μπορέσουν ίσως να καταλήξουν σε διάφορα συμπεράσματα που ίσως μπορεί να χρειάζονται να έχουν και μια εικόνα πιο γενική και όχι απαραίτητα πραγματικού χρόνου για τους ήδη υπό επίλυση υπολογισμούς τους. Αυτή η εικόνα ίσως μπορεί να βοηθήσει σε υποθέσεις στατιστικές με άλλα λόγια πολύ πιθανόν.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Component Diagram Tutorial. (n.d.). [Computer Science]. https://www.lucidchart.com/pages/uml-component-diagram

developer.mozilla. (n.d.). Array.prototype.reduce(). https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce#grouping_objects_by_a_property

Ion, I., PALAGHITA Dragoş, & Sorin, V. (2011). User Types in Online Applications. Volume XVIII, 31–52.

Kaushik, M. (2021, April 7). Top 8 Ship Tracking Websites To Find Your Ship Accurately [Marine insight]. https://www.marineinsight.com/know-more/top-8-websites-to-track-your-ship/

Kevin Sookocheff. (2019, April 4). How Do Websockets Work? https://sookocheff.com/post/networking/how-do-websockets-work/

Malan, R. (1999). Functional Requirements and Use Cases. 8. https://doi.org/10.1.1.436.4773

Microsoft. (2019). Visual Studio Code [Computer Science]. https://code.visualstudio.com/

Piero Borrelli. (2021, January 11). Angular vs. Vue vs. React: Comparing frameworks by performance. https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/

qracorp. (n.d.). Functional vs Non-Functional Requirements: The Definitive Guide. https://qracorp.com/functional-vs-non-functional-requirements/

Raunek, K. (2021, June 25). The Importance of Vessel Tracking System [Marine navigation, marine safety]. https://www.marineinsight.com/marine-safety/the-importance-of-vessel-tracking-system/

Systemation. (2015, September 28). Functional vs. Non-Functional: What's the Difference? www.syste- mation.com/functional-and-non-functional-require- ments-a-primer/

Visual Paradigm. (n.d.). What is Use Case Diagram? [Computer Science]. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

Visual Studio Code. (2019). In Wikipedia (Online). https://en.wikipedia.org/wiki/Visual_Studio_Code

w3schools. (n.d.). JavaScript Array find().

wikipedia. (2021a). Python.

wikipedia. (2021b). Leaflet (software). https://en.wikipedia.org/wiki/Leaflet_(software)

Yoshitaka Shiotsu. (2020, July 2). Angular vs. React: Which is Better for Web Development? https://www.upwork.com/resources/angular-vs-react?utm_source=google&utm_campaign=SEM_GGL_INTL_NonBrand_Marketplace_DSA&utm_medium=cpc&utm_content=113089129402&utm_term=&campaignid=11384804789&matchtype=b&device=c&gclid=EAIaIQobChMI-ZuD4ZHN8QIVmLd3Ch264AhOEAAYAiAAEgJ2zPD_BwE

ΕΠΙΠΛΕΟΝ ΣΥΝΔΕΣΜΟΙ ΑΠΟ ΔΙΑΦΟΡΕΣ ΠΗΓΕΣ

- **Codedamn**

React Crash Course 2020 - Learn React in 1 video + Projects
https://www.youtube.com/watch?v=15YB__vYpuA

- **WebDev Simplified**

Learn useMemo In 10 Minutes
https://www.youtube.com/watch?v=5LrDIWkK_Bc&list=RDCMUCFbNIlppjAuEX4znoulh0Cw&index=8

Learn useContext In 13 Minutes
https://www.youtube.com/watch?v=THL1OPn72vo&list=RDCMUCFbNIlppjAuEX4znoulh0Cw&index=7