ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΝΣΩΜΑΤΩΣΗ ΣΤΟΙΧΕΙΩΝ ΓΕΩΓΡΑΦΙΚΩΝ ΔΕΔΟΜΕΝΩΝ (GIS) ΣΕ ΔΙΔΑΚΤΙΚΗ ΕΦΑΡΜΟΓΗ ΒΑΣΙΣΜΕΝΗ ΣΕ ΠΛΑΤΦΟΡΜΑ REACT

Πτυχιακή Εργασία

Εμμανουήλ Χατζηδάκη

Επιβλέπων Καθηγητής: Αθανάσιος Μαλάμος

Ηράκλειο, Δεκέμβριος 2023

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον καθηγητή μου κ. Μαλάμο για τη συνεχή υποστήριξή του, και την καθοδήγησή κατά τη διάρκεια της πτυχιακής μου εργασίας. Οι σε βάθος συζητήσεις μας πάνω στο θέμα με βοήθησαν να καταλάβω καλύτερα τις αρχές της βιβλιοθήκης React, και να μπορέσω να εξασκήσω τις δεξιότητές μου πάνω σε αυτή. Τέλος, θα ήθελα επίσης να ευχαριστήσω την οικογένεια μου που ήταν το στήριγμά μου σε όλα τα χρόνια των σπουδών μου.

Περίληψη

Η παρούσα πτυχιακή εργασία βασίζεται στην βιβλιοθήκη React & React Leaflet, και αποσκοπεί στην αποτύπωση και διαχείριση γεωγραφικών δεδομένων. Εν προκειμένω, θα αναπτύξουμε την απεικόνιση ορίων γεωγραφικού διαμερίσματος πάνω στο χάρτη με τη μορφή πολυγώνου. Μέσω αυτής της εφαρμογής θα δίνεται η δυνατότητα στον χρήστη η προβολή και αναζήτηση των συγκεκριμένων σημείων πάνω στο χάρτη (markers), ενώ ο admin-χρήστης θα έχει επιπλέον τη δυνατότητα να επεξεργάζεται τα ήδη υπάρχοντα πολύγωνα και σημεία αλλά και να προσθέτει καινούργια στοιχεία.

Abstract

The present bachelor thesis is based on the React & React Leaflet library and aims in the capture and management of geographic data. In more detail, we are going to develop the display of the boundaries of a geographical division on a map in the form of polygon. Via this application, the user will have the ability to view and search these particular points (markers) on the map, while the admin-user will also have the extra ability of editing the already existing polygons and markers, but at the same time to add new.

Περιεχόμενα

Πίνακας Εικόνων

Κεφάλαιο 1: Εισαγωγή

Η πορεία εξέλιξης της τεχνολογίας έχει απομυθοποιήσει σταδιακά την πολυπλοκότητα του χειρισμού γεωγραφικών δεδομένων, επιτρέποντας την εκτέλεση εξελιγμένων χειρισμών χωρικών πληροφοριών όχι μόνο από ειδικούς, αλλά και από τον γενικό πληθυσμό. Η ανάπτυξη μιας διαισθητικής εφαρμογής για την αλληλεπίδραση με γεωγραφικά δεδομένα βρίσκεται στη συμβολή της τεχνολογικής προόδου και του εκδημοκρατισμού των γεωχωρικών πληροφοριών. Αυτή η εργασία περιγράφει τη δημιουργία μιας τέτοιας εφαρμογής - κατασκευασμένης με την χρήση των framework React και React-Leaflet - η οποία παρέχει στους χρήστες μια διαδραστική και ευέλικτη διεπαφή βασισμένη σε χάρτη. Αυτή η διεπαφή επιτρέπει την εξέταση και τον χειρισμό γεωγραφικών σημείων ενδιαφέροντος που αντιπροσωπεύονται ως σημεία (markers) και πολύγωνα, ανοίγοντας δρόμους τόσο για γενικές περιπτώσεις όσο και για εξειδικευμένες χρήσεις.

1.1 Πλαίσιο

Τα συστήματα γεωγραφικών πληροφοριών (Geographic Information Systems - GIS) έχουν εξελιχθεί για να γίνουν βασικό στοιχείο σε πολλές εφαρμογές, που κυμαίνονται από τον πολεοδομικό σχεδιασμό και τη διαχείριση του περιβάλλοντος έως τις μεταφορές και τη δημόσια ασφάλεια. Οι παραδοσιακές εφαρμογές GIS, ενώ είναι ισχυρές, συχνά παρουσιάζουν μια απότομη καμπύλη μάθησης (learning curve) και απαιτούν ουσιαστικές τεχνικές γνώσεις για να λειτουργήσουν αποτελεσματικά. Η εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας πτυχιακής εργασίας παρέχει μια φιλική προς τον χρήστη και βασισμένη στο διαδίκτυο λύση που απλοποιεί την αλληλεπίδραση με γεωγραφικά δεδομένα. Αξιοποιώντας τις σύγχρονες τεχνολογίες Web, απαλλάσσει την προϋπόθεση της τεχνογνωσίας στα GIS.

Μέσω της χρήσης διαδραστικών χαρτών, η εφαρμογή διευκολύνει ένα πλήθος ενεργειών:

Προβολή χάρτη: Κατά την είσοδο, οι χρήστες θα βλέπουν έναν χάρτη με διάφορα σημεία και περιοχές ενδιαφέροντος. Τα σημεία αυτά και οι περιοχές εμφανίζονται με τη χρήση σημείων(markers) και πολυγώνων.

Ανάκτηση πληροφοριών: Μια λειτουργία αναζήτησης δίνει τη δυνατότητα στους χρήστες να περιηγηθούν και να ανακαλύψουν λεπτομερείς πληροφορίες σχετικά με τα εμφανιζόμενα δεδομένα.

Προσαρμογή εμφάνισης δεδομένων: Τα φίλτρα είναι στη διάθεση των χρηστών για να προσαρμόσουν την εμφάνιση του χάρτη, διασφαλίζοντας τη συνάφεια και την εστίαση στα δεδομένα που σχετίζονται με τις ατομικές ανάγκες. Ο χρήστης έχει τη δυνατότητα να πραγματοποιήσει αναζήτηση μέσα από τη λίστα με τα δεδομένα που υπάρχουν πάνω στο χάρτη και να ανακαλύψει περισσότερες πληροφορίες σχετικά με αυτά.

Διαχείριση επιπέδου διαχειριστή: Ο διαχειριστής(Admin) διαθέτει επιπλέον δυνατότητες περιλαμβανομένης της:

- **Προσθήκης και τροποποίηση σημείων:** Οι διαχειριστές μπορούν να εισάγουν νέα σημεία ή να αλλάξουν τα υπάρχοντα χρησιμοποιώντας μια ειδική φόρμα.

- **Προσθήκη και επεξεργασία πολυγώνου**: Εκτός από τη διαχείριση σημείων, οι διαχειριστές έχουν τη δυνατότητα να ορίζουν και να αναθεωρούν περιοχές ενδιαφέροντος—πολύγωνα—τόσο μέσω της φόρμας όσο και απευθείας μέσα στο περιβάλλον του διαδραστικού χάρτη.

1.2 Περιγραφή του Προβλήματος

Η χρήση των γεωγραφικών πληροφοριών περιορίζεται παραδοσιακά από τα εμπόδια του πολύπλοκου λογισμικού και της εξειδικευμένης γνώσης. Παρά τις τεράστιες δυνατότητες των χωρικών δεδομένων να βελτιώσουν την κατανόηση και τη λήψη αποφάσεων σε διάφορους τομείς, το εύρος της εφαρμογής τους έχει περιοριστεί σημαντικά από τέτοια εμπόδια. Το κεντρικό πρόβλημα που προσπαθεί να αντιμετωπίσει αυτή η εργασία είναι η αποσύνδεση μεταξύ των ισχυρών δυνατοτήτων των συστημάτων γεωγραφικών πληροφοριών και της προσβασιμότητάς τους σε χρήστες χωρίς τεχνικό υπόβαθρο στο GIS.
Το κενό είναι ιδιαίτερα εμφανές σε καταστάσεις όπου οι μη ειδικοί χρήστες, όπως οι σχεδιαστές κοινοτήτων, οι τοπικές αρχές και οι εκπαιδευτικοί, απαιτούν διαδραστική και σε πραγματικό χρόνο πρόσβαση σε γεωχωρικά δεδομένα. Το πρόβλημα επεκτείνεται στη δυναμική της διαχείρισης χωρικών δεδομένων, όπου η εργασία προσθήκης, ενημέρωσης και διαχείρισης γεωγραφικών σημείων και πολυγώνων είναι συχνά δύσκολη, επιρρεπής σε σφάλματα και χρονοβόρα όταν εκτελείται με ακατάλληλα εργαλεία.
Μια άλλη διάσταση του προβλήματος είναι η στατική φύση της παρουσίασης γεωγραφικών δεδομένων σε υπάρχουσες πλατφόρμες. Αυτή η ακαμψία περιορίζει την ικανότητα του χρήστη να φιλτράρει και να χειρίζεται δεδομένα δυναμικά, οδηγώντας σε μια στατική και λιγότερο ελκυστική εμπειρία χρήστη. Ουσιαστικά, η ανάγκη για μια λύση που μπορεί να προσφέρει διαδραστικές δυνατότητες χαρτογράφησης σε πραγματικό χρόνο, σε συνδυασμό με φιλικά προς τον χρήστη εργαλεία διαχείρισης δεδομένων, είναι το πρόβλημα στο επίκεντρο αυτής της διατριβής.
Η συγκεκριμένη εφαρμογή η οποία έχει υλοποιηθεί, αφορά εκκλησίες και ενορίες όπου τα markers αντιπροσωπεύουν εκκλησίες και τα πολύγωνα αντιπροσωπεύουν ενορίες.

1.3 Στόχοι και Κίνητρα της Εργασίας

Ο πρωταρχικός στόχος αυτής της εργασίας είναι να σχεδιάσει και να εφαρμόσει μια διαδικτυακή εφαρμογή που υπερβαίνει τους περιορισμούς που εντοπίζονται στο σημερινό τοπίο αλληλεπίδρασης και διαχείρισης γεωγραφικών δεδομένων χρησιμοποιώντας σύγχρονες τεχνολογίες Web. Οι στόχοι είναι οι εξής:

Ανάπτυξη μιας διαδραστικής διεπαφής χάρτη: Για τη δημιουργία μιας ανταποκρινόμενης και διαισθητικής διεπαφής χάρτη που επιτρέπει στους χρήστες να βλέπουν και να αλληλεπιδρούν με γεωγραφικά σημεία και πολύγωνα.

Διευκόλυνση αναζήτησης και ανάκτησης δεδομένων: Ενσωμάτωση μιας λειτουργίας αναζήτησης που επιτρέπει στους χρήστες να εντοπίζουν αποτελεσματικά και να έχουν πρόσβαση σε λεπτομερείς πληροφορίες σχετικά με τις γεωγραφικές οντότητες που εμφανίζονται στο χάρτη.

Ενεργοποίηση προσαρμοσμένης οπτικοποίησης δεδομένων: Παροχή στους χρήστες εργαλεία φιλτραρίσματος για την προσαρμογή της οπτικής αναπαράστασης γεωγραφικών δεδομένων σύμφωνα με τα συγκεκριμένα ενδιαφέροντα ή τις απαιτήσεις τους.

Απλοποίηση της διαχείρισης δεδομένων για χρήστες διαχειριστή: Ενδυνάμωση διαχειριστών με τη δυνατότητα προσθήκης, επεξεργασίας και διαχείρισης γεωγραφικών σημείων και πολυγώνων με φιλικό προς τον χρήστη τρόπο, διευκολύνοντας έτσι τις ενημερώσεις και τις επεξεργασίες σε πραγματικό χρόνο απευθείας μέσα στη διεπαφή χάρτη.

Σχεδιασμός με επίκεντρο τον χρήστη: Για να δοθεί προτεραιότητα στην εμπειρία του χρήστη στη σχεδίαση της εφαρμογής, διασφαλίζοντας ότι η διεπαφή είναι προσβάσιμη σε χρήστες με διάφορους βαθμούς τεχνικής εξειδίκευσης.

Το κίνητρο της παρούσας πτυχιακής εργασίας είναι να δημιουργήσει μια εφαρμογή που θα προσφέρει τη προβολή και επεξεργασία γεωγραφικών δεδομένων, καθώς και την προβολή τους πάνω σε ένα χάρτη. το παρών πηγάζει από τη σημαντική ανάγκη για μια εύκολη και πρακτική λύση στο τομέα της διαχείρισης γεωγραφικών πληροφοριών. Η δυνατότητα προβολής γεωγραφικών δεδομένων πάνω σε ένα χάρτη κάνει την κατανόηση των πληροφοριών ευκολότερη, καθώς οπτικοποιωντας τις πληροφορίες γίνονται πιο κατανοητές στο χρήστη.

Κεφάλαιο 2: Τεχνολογίες

2.1 Εισαγωγή

Τα Γεωγραφικά Συστήματα Πληροφοριών (GIS) τοποθετούνται ως πολύπλευρες δομές σχεδιασμένες για τη σχολαστική συλλογή, διαχείριση και εξέταση πληροφοριών που είναι εγγενώς γεωγραφικές. Τα GIS βοηθούν στη συγχώνευση διαφορετικών δεδομένων, παρέχοντας έτσι τα απαραίτητα εργαλεία για τον έλεγχο της χωρικής τοποθέτησης και τη διαστρωμάτωση των πληροφοριακών βαθμίδων σε ενδεικτικές αποδόσεις με τη μορφή χαρτογραφικών εξόδων και τρισδιάστατων σκηνών. Στο πλαίσιο αυτής της εργασίας, το GIS δεν είναι απλώς μια υποστηρικτική δομή, αλλά η ουσία στην ανάπτυξη μιας εφαρμογής που είναι ικανή για τις εξελιγμένες λειτουργίες διαχείρισης και αναπαράστασης γεωγραφικών δεδομένων σε έναν διαδραστικό τομέα που διαμεσολαβείται από το διαδίκτυο.

Στον τομέα του GIS, το οικοσύστημα της React δίνει τη δυνατότητα στους προγραμματιστές να δημιουργήσουν διεπαφές χαρτών υψηλής απόκρισης και διαισθητικής, με δυνατότητες όπως ζωντανή ροή δεδομένων, περιεχόμενο που δημιουργείται από τον χρήστη και σύνθετη χωρική ανάλυση που παρουσιάζονται με φιλικό προς τον χρήστη τρόπο.

Η ενσωμάτωση της MongoDB σε εφαρμογές GIS προσφέρει μια ισχυρή λύση στο back-end που υποστηρίζει ενημερώσεις δεδομένων σε πραγματικό χρόνο, επεκτασιμότητα για τη διαχείριση υψηλής κίνησης και όγκου δεδομένων και την προσαρμοστικότητα που απαιτείται για τις ποικίλες απαιτήσεις της γεωχωρικής ανάλυσης. Τα γεωχωρικά ερωτήματα στη MongoDB επιτρέπουν την αναζήτηση βάσει τοποθεσίας, η οποία είναι απαραίτητη για εφαρμογές GIS που απαιτούν τη δυνατότητα εύρεσης δεδομένων σε μια ορισμένη απόσταση από ένα σημείο, μέσα σε μια καθορισμένη περιοχή ή κοντά σε μια καθορισμένη διαδρομή. Οι δυνατότητες δημιουργίας ευρετηρίου της βάσης δεδομένων ενισχύουν την απόδοση, ειδικά όταν ασχολούνται με μεγάλους όγκους δεδομένων, διασφαλίζοντας ότι οι χρήστες αντιμετωπίζουν ελάχιστο λανθάνοντα χρόνο κατά την ανάκτηση δεδομένων.

Προκειμένου να επιτευχθούν οι στόχοι της παρούσας πτυχιακής έγινε χρήση της **JavaScript** όπως και πολλές σύγχρονες τεχνολογίες τόσο για το front-end όσο και για το back-end. Για το front-end χρησιμοποιήθηκαν η βιβλιοθήκη React μαζί με τη React-Leaflet για τους χάρτες, καθώς και η γλώσσα CSS μαζί με την βιβλιοθήκη η Ant Design για την εμφάνιση. Επιπλέον έγινε χρήση του React Leaflet Draw για την επεξεργασία δεδομένων πάνω στο χάρτη. Στο back-end για το server χρησιμοποιήθηκε το Node.js με το framework Express και για την αποθήκευση των δεδομένων έγινε η χρήση της βάσης δεδομένων MongoDB σε συνδυασμό με τη Mongoose.

2.2 JavaScript

Η άνοδος της JavaScript ως η κορυφαία γλώσσα για τον Web είναι απόδειξη της ευελιξίας και στο πεδίο. Αρχικά σχεδιασμένο το 1995 από την Netscape Communications, η JavaScript έχει υποστεί μια αξιοσημείωτη εξέλιξη, ειδικά αφού τυποποιήθηκε με την προδιαγραφή ECMAScript. Ξεκίνησε ως ένα μέσο για τη σύνταξη απλών αλληλεπιδράσεων σε ένα πρόγραμμα περιήγησης ιστού και έχει εξελιχθεί σε μια ολοκληρωμένη σειρά εργαλείων που αψηφούν τους αρχικούς περιορισμούς δημιουργίας του.

Η ευελιξία της JavaScript στις λειτουργίες από την πλευρά του πελάτη και του διακομιστή

Η JavaScript ενσωματώνει το βασικό εργαλείο για προγραμματιστές, με την εφαρμογή της να εκτείνεται πέρα από τις απλές ενέργειες από την πλευρά του πελάτη στην ενορχήστρωση λειτουργιών από την πλευρά του διακομιστή. Αυτό αποδεικνύεται από περιβάλλοντα όπως το Node.js, τα οποία χρησιμοποιούμε σε αυτήν την εργασία. Ο βαθύς αντίκτυπος της JavaScript είναι εμφανής στην ικανότητά της να διευκολύνει την τροποποίηση περιεχομένου σε πραγματικό χρόνο, να ζωντανεύει τα γραφικά με λεπτότητα και να χειρίζεται ασύγχρονες επικοινωνίες με διακομιστές. Τέτοιες δυνατότητες είναι ζωτικής σημασίας για την καλλιέργεια μιας διαδραστικής εμπειρίας, ενός προτύπου στη σύγχρονη σχεδίαση ιστοσελίδων και μοτίβα αλληλεπίδρασης με τους χρήστες.

Frameworks και βιβλιοθήκες: Ενίσχυση των δυνατοτήτων της JavaScript

Η επιρροή της JavaScript στην τροχιά ανάπτυξης του Web έχει ενισχυθεί περαιτέρω με την εμφάνιση ενός οικοσυστήματος πλούσιου σε frameworks και βιβλιοθήκες. Αυτά τα abstractions έδωσαν τη δυνατότητα στους προγραμματιστές να δημιουργήσουν πολύπλοκες λειτουργίες με μεγαλύτερη ευκολία. Frameworks όπως το React, που χρησιμοποιούμε σε αυτήν την εργασία, αποτελούν παράδειγμα αυτής της τάσης. Η React, ειδικότερα, φέρνει επανάσταση στον τρόπο κατασκευής των διεπαφών χρήστη, εισάγοντας μια αρχιτεκτονική βασισμένη σε widgets που ενισχύει την επαναχρησιμοποίηση και τη συντηρησιμότητα του κώδικα.

Ο ρόλος της JavaScript στις σύγχρονες εφαρμογές GIS

Στο πλαίσιο των συστημάτων γεωγραφικών πληροφοριών (GIS), η ευελιξία της JavaScript γίνεται ακόμη πιο έντονη. Οι εφαρμογές GIS που βασίζονται στο Web αξιοποιούν τη JavaScript για τη διαχείριση χωρικών δεδομένων, την απόδοση χαρτών διαδραστικά και την αποτελεσματική επεξεργασία γεωχωρικών πληροφοριών. Frameworks όπως το React επιτρέπουν τη δημιουργία ανταποκρινόμενων και φιλικών προς το χρήστη διεπαφών GIS, ενισχύοντας την ικανότητα του χρήστη να αλληλεπιδρά με πολύπλοκα χωρικά δεδομένα.

Συμπερασματικά, το ταξίδι της JavaScript από μια απλή γλώσσα scripting στη ραχοκοκαλιά των σύγχρονων διαδικτυακών εφαρμογών είναι αξιοσημείωτο. Η συνεχής εξέλιξή της, που υποστηρίζεται από ένα ακμάζον οικοσύστημα πλαισίων και βιβλιοθηκών, διασφαλίζει τη διαρκή συνάφειά του στον ταχέως εξελισσόμενο τομέα της ανάπτυξης Ιστού. Αυτή η εργασία, μέσω της χρήσης JavaScript και της React, παρουσιάζει την ευελιξία της γλώσσας και τον κεντρικό της ρόλο στη δημιουργία εξελιγμένων εφαρμογών ιστού.

2.3 Τεχνολογίες Front-End

2.3.1 React

Η React είναι μια βιβλιοθήκη JavaScript ανοιχτού κώδικα, γνωστή για τη δημιουργία διεπαφών χρήστη, ιδιαίτερα γνωστή για την αποτελεσματική ενημέρωση και απόδοση των components κατά τις αλλαγές δεδομένων. Αναπτύχθηκε από το Facebook, η React βασίζεται σε πολλές βασικές αρχές που επαναπροσδιορίζουν τον τρόπο με τον οποίο οι προγραμματιστές προσεγγίζουν τη σχεδίαση διεπαφής χρήστη.

Αρχιτεκτονική που βασίζεται σε components: Μια modular προσέγγιση

Στο επίκεντρο της σχεδιαστικής φιλοσοφίας της React είναι η αρχιτεκτονική της που βασίζεται σε components. Αυτή η modular προσέγγιση περιστρέφεται γύρω από components, αυτόνομες μονάδες που διαχειρίζονται τη δική τους κατάσταση (state) και συνδυάζονται για να σχηματίσουν σύνθετες διεπαφές χρήστη. Αυτό όχι μόνο προάγει την επαναχρησιμοποίηση και τον διαχωρισμό των λειτουργιών, αλλά οδηγεί επίσης σε πιο διαχειρίσιμο και επεκτάσιμο κώδικα. Τα στοιχεία στη React λειτουργούν ως αυτοτελή στοιχεία που ενσωματώνουν τόσο τη λογική όσο και την παρουσίαση, καθιστώντας τα εξαιρετικά επαναχρησιμοποιήσιμα σε διαφορετικά μέρη μιας εφαρμογής.

State and Props: Διαχείριση δεδομένων στο React

Η διάκριση της React μεταξύ κατάστασης και αμετάβλητων props υποστηρίζει τον μηχανισμό χειρισμού δεδομένων του. Ενώ η κατάσταση είναι εσωτερική και ελέγχει τη συμπεριφορά ενός στοιχείου, τα props μεταβιβάζονται σε στοιχεία από τον γονέα τους, παρόμοια με παραμέτρους συνάρτησης. Αυτός ο διαχωρισμός απλοποιεί τη ροή δεδομένων και τη διαχείριση κατάστασης, επιτρέποντας πιο προβλέψιμες και ευκολότερες διεπαφές σφαλμάτων.

Μονόδρομη ροή δεδομένων: Απλοποίηση πολυπλοκότητας

Ένα από τα βασικά χαρακτηριστικά της React είναι η μονοκατευθυντική ροή δεδομένων, η οποία μειώνει την πολυπλοκότητα και ενισχύει την προβλεψιμότητα στις εφαρμογές. Αυτή η αρχιτεκτονική απλοποιεί τον εντοπισμό σφαλμάτων και τον συλλογισμό σχετικά με τη δομή της εφαρμογής, καθώς τα δεδομένα ρέουν προς μια ενιαία κατεύθυνση από τα γονικά προς τα θυγατρικά στοιχεία, διευκολύνοντας την παρακολούθηση των αλλαγών και την κατανόηση του τρόπου με τον οποίο τα δεδομένα επηρεάζουν τη διεπαφή χρήστη.

Virtual DOM

Η React χρησιμοποιεί ένα εικονικό DOM για τη βελτίωση της απόδοσης της εφαρμογής. Συμβιβάζοντας τις αλλαγές σε μια εικονική αναπαράσταση της διεπαφής χρήστη, ελαχιστοποιεί τον άμεσο χειρισμό DOM, βελτιστοποιώντας τους χρόνους απόδοσης. Αυτή η προσέγγιση επιτρέπει στη React να υπολογίσει το ελάχιστο σύνολο αλλαγών που απαιτούνται για την ενημέρωση του πραγματικού DOM, οδηγώντας σε σημαντικά κέρδη απόδοσης, ειδικά σε εφαρμογές μεγάλης κλίμακας.

JSX: Γεφύρωση HTML και JavaScript

Το JSX, μια επέκταση σύνταξης που μοιάζει με HTML, επιτρέπει στους προγραμματιστές να γράφουν δομές διεπαφής χρήστη σε μια οικεία μορφή εντός JavaScript. Αυτό οδηγεί σε πιο ευανάγνωστο κώδικα και μια βελτιωμένη διαδικασία ανάπτυξης. Το JSX κάνει τον κώδικα οπτικά πιο παρόμοιο με το τελικό αποτέλεσμα, μειώνοντας το γνωστικό φορτίο στους προγραμματιστές καθώς σχεδιάζουν τη διεπαφή χρήστη.

Οικοσύστημα και Κοινότητα της React

Πέρα από τα βασικά χαρακτηριστικά του, η δύναμη της React βρίσκεται στο ζωντανό οικοσύστημα και την κοινότητά του. Η δημοτικότητα της βιβλιοθήκης έχει ωθήσει την ανάπτυξη πολλών εργαλείων, επεκτάσεων και συμπληρωματικών βιβλιοθηκών, όπως το Redux για διαχείριση κατάστασης και το React Router για πλοήγηση σε εφαρμογές web. Αυτό το οικοσύστημα εμπλουτίζει την εμπειρία της React, παρέχοντας στους προγραμματιστές μια σειρά από ισχυρά εργαλεία για τη δημιουργία ισχυρών, δυναμικών εφαρμογών.

Η εισαγωγή της React σηματοδότησε ένα σημαντικό ορόσημο στην ανάπτυξη Web apps. Η αρχιτεκτονική της που βασίζεται σε components , σε συνδυασμό με τους αποτελεσματικούς μηχανισμούς χειρισμού και απόδοσης δεδομένων, το έχουν καταστήσει δημοφιλή επιλογή για προγραμματιστές σε όλο τον κόσμο. Σε αυτήν την εργασία, αξιοποιούμε τη δύναμη της React για τη δημιουργία εξελιγμένων και διαδραστικών διεπαφών χρήστη, επιδεικνύοντας τις δυνατότητές του στο χειρισμό πολύπλοκων εφαρμογών Ιστού.

2.3.2 React-Leaflet

Η βιβλιοθήκη React-Leaflet είναι ένα ισχυρό εργαλείο που συνδυάζει τα δυνατά σημεία του React με την ευελιξία του Leaflet, με στόχο τη δημιουργία διαδραστικών χαρτών σε εφαρμογές React. Συγκεκριμένα, το React-Leaflet είναι μια βιβλιοθήκη JavaScript που επιτρέπει την ενσωμάτωση του Leaflet σε ένα πρόγραμμα React. Προσφέρει επίσης στοιχεία που μπορούν να χρησιμοποιηθούν για τη δημιουργία χαρτών, όπως το '<Map>' για την εμφάνιση του χάρτη και το '<Marker>' για τα σημεία ενδιαφέροντος.

Βελτιωμένη προσαρμογή διεπαφής χρήστη και δυναμική αλληλεπίδραση χάρτη

Ένα βασικό χαρακτηριστικό του React-Leaflet είναι η ικανότητά του να αναδιαμορφώνει τη διεπαφή χρήστη των χαρτών χρησιμοποιώντας στοιχεία και καταστάσεις React. Αυτό επιτρέπει τον έλεγχο και την ανανέωση του χάρτη, βελτιώνοντας την εμπειρία του χρήστη μέσω δυναμικής αλληλεπίδρασης. Η βιβλιοθήκη υποστηρίζει το χειρισμό συμβάντων όπως κλικ και αλλαγές στην κατάσταση του χάρτη, επιδεικνύοντας τη διαδραστικότητά του.

Προηγμένες δυνατότητες GIS με React-Leaflet

Στο πλαίσιο του GIS, το React-Leaflet ξεχωρίζει για την ικανότητά του να ενσωματώνει διάφορα σχήματα όπως γραμμές, κύκλους, πολύγωνα και άλλα, χρησιμοποιώντας αντίστοιχα στοιχεία και παραμέτρους. Αυτή η δυνατότητα είναι ζωτικής σημασίας για την οπτικοποίηση πολύπλοκων χωρικών δεδομένων και γεωγραφικών χαρακτηριστικών. Το React-Leaflet διευκολύνει την αναπαράσταση αυτών των στοιχείων με τρόπο αποτελεσματικό και οπτικά ελκυστικό, καθιστώντας το ένα ανεκτίμητο εργαλείο για εφαρμογές GIS.

Προσαρμογή χρήστη και επεκτάσεις κοινότητας

Το React-Leaflet επιτρέπει στους χρήστες να προσαρμόζουν τους χάρτες με συγκεκριμένα στοιχεία, προσφέροντας υψηλό βαθμό ευελιξίας. Επιπλέον, η βιβλιοθήκη επωφελείται από μια προσέγγιση που βασίζεται στην κοινότητα, με διάφορα πακέτα επέκτασης που παρέχονται από άλλους χρήστες. Αυτές οι επεκτάσεις ενισχύουν περαιτέρω την αποτελεσματικότητα και τη χρηστικότητα της βιβλιοθήκης, ειδικά σε εξελιγμένες εφαρμογές GIS όπου απαιτούνται προηγμένα χαρακτηριστικά και λειτουργίες χαρτογράφησης.

Συνοπτικά, το React-Leaflet αντιπροσωπεύει μια σημαντική πρόοδο στην ενσωμάτωση διαδραστικών χαρτών σε διαδικτυακές εφαρμογές, ιδιαίτερα για σκοπούς GIS. Η ικανότητά του να συνδυάζει τις δυνατότητες reactive UI του React με τις ισχυρές λειτουργίες χαρτογράφησης του Leaflet το καθιστά ιδανική επιλογή για την ανάπτυξη εξελιγμένων, φιλικών προς το χρήστη εφαρμογών GIS. Η υποστήριξη της βιβλιοθήκης για χειρισμό συμβάντων, διάφορα στοιχεία χαρτογράφησης και επεκτάσεις που βασίζονται στην κοινότητα την καθιστούν μια ισχυρή λύση για σύγχρονα έργα GIS που βασίζονται στον ιστό.

2.3.3 CSS

Η CSS (Cascading Style Sheets) είναι η γλώσσα για τη δημιουργία στυλ στο Web, η οποία δείχνει την οπτική αισθητική των στοιχείων HTML. Οι προδιαγραφές του περιλαμβάνουν ένα ευρύ φάσμα δυνατοτήτων styling, καθοριστικής σημασίας για τη σύγχρονη σχεδίαση ιστοσελίδων.

Ο πυρήνας του CSS βρίσκεται στη δομή κανόνων του, η οποία αποτελείται από επιλογείς που στοχεύουν στοιχεία HTML και μπλοκ δηλώσεων που ορίζουν τα στυλ. Αυτή η δομή επιτρέπει τον ακριβή έλεγχο της εμφάνισης των στοιχείων Ιστού, από τη βασική μορφοποίηση κειμένου έως τα πολύπλοκα σχέδια διάταξης. Κάθε κανόνας στο CSS είναι ένα σαφές σύνολο εντολών, που καθορίζει τον τρόπο με τον οποίο θα πρέπει να εμφανίζονται συγκεκριμένα στοιχεία ή ομάδες στοιχείων σε μια ιστοσελίδα.

Οπτικές ιδιότητες: Βελτίωση διεπαφών χρήστη

Το CSS ελέγχει βασικές οπτικές ιδιότητες όπως διάταξη, χρώμα, τυπογραφία και άλλα. Αυτός ο έλεγχος είναι σημαντικός για τη βελτίωση των διεπαφών χρήστη, διασφαλίζοντας ότι οι ιστοσελίδες δεν είναι μόνο λειτουργικές αλλά και αισθητικά ευχάριστες. Το CSS παρέχει μια ποικιλία ιδιοτήτων που διαχειρίζονται κάθε πτυχή της εμφάνισης ενός στοιχείου, από το μέγεθος και το σχήμα του έως τη θέση του στη σελίδα, το χρωματικό του συνδυασμό και το στυλ γραμματοσειράς.

Μέθοδοι ενσωμάτωσης

Τα στυλ CSS μπορούν να συμπεριληφθούν ως εξωτερικά css αρχεία ή ενσωματωμένα στυλ (με το <style> tag της html, προσφέροντας ευελιξία στον τρόπο χρήσης του CSS σε ένα έργο. Τα εξωτερικά αρχεία προάγουν τη συνέπεια σε πολλές html σελίδες, ενώ τα ενσωματωμένα στυλ επιτρέπουν συγκεκριμένες, εφάπαξ προσαρμογές. Αυτή η ευελιξία επιτρέπει στους προγραμματιστές να επιλέξουν την πιο αποτελεσματική και κατάλληλη μέθοδο στυλ για κάθε έργο, διασφαλίζοντας ότι το CSS μπορεί να προσαρμοστεί σε ένα ευρύ φάσμα αναγκών ανάπτυξης ιστού.

Συνδυασμοί επιλογής

Οι σύνθετες στρατηγικές επιλογής στο CSS επιτρέπουν στοχευμένο στυλ, διευκολύνοντας προηγμένα σχέδια σχεδίασης. Οι συνδυασμοί επιλογέων επιτρέπουν στους προγραμματιστές να εφαρμόζουν στυλ σε στοιχεία υπό συγκεκριμένες συνθήκες ή σε συγκεκριμένα περιβάλλοντα. Αυτή η δυνατότητα είναι σημαντική για τον αποκριτικό σχεδιασμό, επιτρέποντας στα στυλ να προσαρμόζονται βάσει παραγόντων όπως το μέγεθος της οθόνης, ο τύπος συσκευής ή η αλληλεπίδραση με τον χρήστη.

Media Queries

Μια ουσιαστική πτυχή του σύγχρονου CSS είναι η χρήση Media Queries, τα οποία επιτρέπουν τη δημιουργία responsive σελίδων. Τα Media Queries επιτρέπουν στα στυλ να προσαρμόζονται με βάση τα χαρακτηριστικά της συσκευής που εμφανίζει το περιεχόμενο, όπως το μέγεθος της οθόνης, την ανάλυση και τον προσανατολισμό. Αυτή η προσαρμοστικότητα είναι κρίσιμη για τη διασφάλιση ότι οι ιστότοποι παρέχουν βέλτιστες εμπειρίες χρήστη σε όλες τις συσκευές, από επιτραπέζιους υπολογιστές έως smartphone.

Μεταβάσεις και κινούμενα σχέδια

Το CSS υποστηρίζει επίσης μεταβάσεις και κινούμενα σχέδια, προσθέτοντας ένα επίπεδο διαδραστικότητας και δυναμισμού στις ιστοσελίδες. Αυτές οι δυνατότητες επιτρέπουν ομαλούς μετασχηματισμούς ιδιοτήτων με την πάροδο του χρόνου, δημιουργώντας ελκυστικές και διαδραστικές εμπειρίες χρήστη. Διαδραματίζουν σημαντικό ρόλο στη σύγχρονη σχεδίαση ιστοσελίδων, όπου η αφοσίωση και η αλληλεπίδραση των χρηστών είναι πρωταρχικής σημασίας.

Συμπερασματικά, το CSS είναι ένα απαραίτητο εργαλείο στην ανάπτυξη ιστού, που προσφέρει μια σειρά λειτουργιών για τη δημιουργία οπτικά συναρπαστικών και διαδραστικών ιστοσελίδων. Η ικανότητά του να ελέγχει τη διάταξη, την εμφάνιση και την ανταπόκριση το καθιστά μια ισχυρή γλώσσα για τη διαμόρφωση της εμπειρίας χρήστη στον ιστό. Η εξέλιξη του CSS συνεχίζει να οδηγεί την καινοτομία στον σχεδιασμό ιστοσελίδων, επιτρέποντας στους προγραμματιστές και τους σχεδιαστές να δημιουργούν διαισθητικές, αισθητικά ευχάριστες και φιλικές προς το χρήστη διεπαφές.

2.3.4 Ant-Design

Αυτή η συγκεκριμένη component library και σύστημα σχεδίασης χρησιμοποιείται ευρέως για τη δημιουργία διεπαφών χρήστη σε εφαρμογές Web. Προσφέρει μια σειρά από τυποποιημένα και προκαθορισμένα στοιχεία διεπαφής χρήστη που βασίζονται σε σύγχρονες αρχές σχεδίασης, με αποτέλεσμα έναν συνδυασμό ωραίας εμφάνισης και ευκολίας χρήσης. Τα βασικά χαρακτηριστικά του περιλαμβάνουν:

Components διεπαφής χρήστη: Η βιβλιοθήκη παρέχει ένα ευρύ φάσμα έτοιμων προς χρήση components, όπως γραφήματα, φόρμες, κουμπιά, πίνακες κ.λπ. Αυτά τα components μπορούν να εφαρμοστούν απευθείας σε εφαρμογές, διασφαλίζοντας συνέπεια και ποιότητα στη διεπαφή χρήστη. Είναι σχεδιασμένα να είναι ευέλικτα και προσαρμόσιμα, καλύπτοντας ένα ευρύ φάσμα σχεδιαστικών απαιτήσεων και βελτιώνοντας τη συνολική εμπειρία χρήστη.

Diverse Toolkit: Παρόμοια με τα components διεπαφής χρήστη, αυτή η βιβλιοθήκη περιλαμβάνει διάφορα εργαλεία όπως ένα σύστημα διαχείρισης κατάστασης και εργαλεία ανάπτυξης. Αυτά τα εργαλεία διευκολύνουν την ευκολότερη και πιο αποτελεσματική ανάπτυξη εφαρμογών. Το σύστημα διαχείρισης κατάστασης εξασφαλίζει απρόσκοπτη ροή δεδομένων και αλληλεπίδραση εντός της εφαρμογής, ενώ τα εργαλεία ανάπτυξης παρέχουν βασικές λειτουργίες για τον εξορθολογισμό της διαδικασίας ανάπτυξης, όπως βοηθήματα εντοπισμού σφαλμάτων, εργαλεία βελτιστοποίησης απόδοσης και δυνατότητες ενοποίησης με άλλα πλαίσια και βιβλιοθήκες.

Επαναχρησιμοποίηση κώδικα: Τα προκαθορισμένα components αυτής της βιβλιοθήκης μπορούν να ενσωματωθούν στον κώδικα και να επαναχρησιμοποιηθούν, εξοικονομώντας σημαντικό χρόνο ανάπτυξης. Αυτή η πτυχή της επαναχρησιμοποίησης όχι μόνο επιταχύνει τη διαδικασία ανάπτυξης αλλά προάγει επίσης τη συνέπεια σε διάφορα μέρη της εφαρμογής. Επιτρέπει στους προγραμματιστές να επικεντρωθούν σε πιο σύνθετες πτυχές της εφαρμογής, γνωρίζοντας ότι τα τυπικά στοιχεία διεπαφής χρήστη είναι άμεσα διαθέσιμα και διατηρούμενα.

Επιπτώσεις στη ροή εργασιών και την αποτελεσματικότητα της ανάπτυξης

Η υιοθέτηση αυτού του συστήματος βιβλιοθήκης και σχεδιασμού συστατικών στην ανάπτυξη εφαρμογών Ιστού αποφέρει πολλά οφέλη:

Αυξημένη απόδοση: Παρέχοντας ένα ολοκληρωμένο σύνολο έτοιμων προς χρήση components και εργαλείων, η βιβλιοθήκη μειώνει τον χρόνο και την προσπάθεια που απαιτείται για τη δημιουργία και τη διατήρηση στοιχείων διεπαφής χρήστη.

Συνέπεια και επεκτασιμότητα: Τα τυποποιημένα components εξασφαλίζουν συνεπή εμφάνιση και αίσθηση σε όλη την εφαρμογή, η οποία είναι σημαντική για την ταυτότητα της επωνυμίας και την εμπειρία του χρήστη. Η επεκτασιμότητα ενισχύεται επίσης, καθώς τα εξαρτήματα μπορούν εύκολα να τροποποιηθούν ή να επεκταθούν για να καλύψουν τις εξελισσόμενες ανάγκες.

Συνεργασία και ομοιομορφία: Η χρήση μιας βιβλιοθήκης κοινού εξαρτήματος ενισχύει την καλύτερη συνεργασία μεταξύ σχεδιαστών και προγραμματιστών. Δημιουργεί μια ενοποιημένη γλώσσα για την ανάπτυξη διεπαφής χρήστη, διευκολύνοντας την επικοινωνία ιδεών σχεδιασμού και λεπτομερειών υλοποίησης.

Εστίαση στην εμπειρία χρήστη: Με τη φροντίδα των θεμελιωδών στοιχείων διεπαφής χρήστη, οι προγραμματιστές και οι σχεδιαστές μπορούν να επικεντρωθούν περισσότερο στη βελτίωση της εμπειρίας χρήστη και στην προσθήκη μοναδικών λειτουργιών στην εφαρμογή.

Συμπερασματικά, η χρήση αυτής της συγκεκριμένης βιβλιοθήκης συστατικών και συστήματος σχεδίασης αποτελεί απόδειξη της εξελισσόμενης φύσης της ανάπτυξης ιστού, όπου η αποτελεσματικότητα, η συνέπεια και η εμπειρία χρήστη είναι πρωταρχικής σημασίας. Η βιβλιοθήκη αποτελεί παράδειγμα των πλεονεκτημάτων μιας δομημένης προσέγγισης για την ανάπτυξη διεπαφής χρήστη, βελτιστοποιώντας τη διαδικασία και δίνοντας τη δυνατότητα στους προγραμματιστές να δημιουργούν υψηλής ποιότητας εφαρμογές web με επίκεντρο τον χρήστη.

2.3.5 React-Leaflet-Draw

Το React-Leafle-Draw plugin είναι μια σημαντική προσθήκη στο οικοσύστημα React-Leaflet, επιτρέποντας δυνατότητες σχεδίασης σε χάρτες που έχουν δημιουργηθεί με τις βιβλιοθήκες που αναφέρθηκαν προηγουμένως. Αυτό το plugin ενισχύει σημαντικά τις διαδραστικές πτυχές του χειρισμού χαρτών σε εφαρμογές web. Τα χαρακτηριστικά και τα οφέλη του περιλαμβάνουν:

Σχέδιο και δημιουργία σχήματος: Οι χρήστες μπορούν να δημιουργήσουν αβίαστα διάφορα σχήματα στο χάρτη, όπως πολύγωνα, γραμμές, κύκλους και άλλα. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη σε εφαρμογές GIS όπου απαιτείται προσαρμοσμένη αναπαράσταση χωρικών δεδομένων. Για παράδειγμα, οι χρήστες μπορούν να οριοθετήσουν συγκεκριμένες περιοχές, να σχεδιάσουν διαδρομές ή να επισημάνουν ζώνες απευθείας στον χάρτη, κάνοντας την οπτικοποίηση δεδομένων πιο διαδραστική και φιλική προς τον χρήστη.

Προσαρμογή στυλ και στοιχείων ελέγχου: Το plugin επιτρέπει την προσαρμογή του στυλ αυτών των σχημάτων, συμπεριλαμβανομένου του χρώματος, του πάχους και της αδιαφάνειας. Οι χρήστες μπορούν επίσης να τροποποιήσουν τα εικονίδια ελέγχου και άλλες ρυθμίσεις, προσαρμόζοντας την εμφάνιση και τη λειτουργικότητα του χάρτη σε συγκεκριμένες ανάγκες. Αυτό το επίπεδο προσαρμογής διασφαλίζει ότι ο χάρτης όχι μόνο εξυπηρετεί τον λειτουργικό του σκοπό, αλλά και ευθυγραμμίζεται με τη συνολική σχεδίαση και την αισθητική της εφαρμογής.

Επεξεργασία και τροποποίηση: Ένα βασικό χαρακτηριστικό του React-Leaflet-Draw είναι η δυνατότητα των χρηστών να επεξεργάζονται τα δημιουργημένα σχήματα. Αυτό περιλαμβάνει την αλλαγή του μεγέθους, του σχήματος και της θέσης τους ή ακόμη και την πλήρη διαγραφή τους. Τέτοιες δυνατότητες επεξεργασίας είναι απαραίτητες σε δυναμικές εφαρμογές όπου τα δεδομένα χάρτη μπορεί να χρειάζεται να ενημερώνονται συχνά ή ως απόκριση στις αλληλεπιδράσεις των χρηστών.

Ενσωμάτωση με τις Γεωχωρικές Δυνατότητες του Leaflet: Το React-Leaflet-Draw αξιοποιεί τις ισχυρές δυνατότητες γεωχωρικής επεξεργασίας του Leaflet, επιτρέποντας περισσότερα από το απλό σχέδιο. Μπορεί να ενσωματωθεί με άλλα πρόσθετα Leaflet για βελτιωμένη λειτουργικότητα, όπως γεωκωδικοποίηση, δρομολόγηση και χωρική ανάλυση. Αυτή η ενοποίηση το καθιστά ένα ισχυρό εργαλείο για την ανάπτυξη ολοκληρωμένων εφαρμογών GIS.

Αλληλεπίδραση: Επιτρέποντας στους χρήστες να αλληλεπιδρούν απευθείας με τον χάρτη, το React-Leaflet-Draw ενθαρρύνει μια πιο ελκυστική και διαδραστική εμπειρία χρήστη. Είτε πρόκειται για εισαγωγή δεδομένων, οπτικοποίηση ή ανάλυση, η δυνατότητα σχεδίασης και επεξεργασίας στο χάρτη καθιστά την εφαρμογή πιο διαισθητική και χρηστοκεντρική.

Συνοπτικά, το React-Leaflet-Draw είναι μια πολύτιμη επέκταση για εφαρμογές web που απαιτούν δυνατότητες διαδραστικής χαρτογράφησης. Τα χαρακτηριστικά του για σχεδίαση, προσαρμογή, επεξεργασία και ενσωμάτωση με τις γεωχωρικές λειτουργίες του Leaflet το καθιστούν απαραίτητο εργαλείο για σύγχρονες εφαρμογές GIS. Το plugin ενισχύει την αλληλεπίδραση και την αφοσίωση των χρηστών, επιτρέποντας μια πιο δυναμική και αποκριτική εμπειρία χαρτογράφησης.

2.4 Τεχνολογίες Back-End

Το back-end κομμάτι της εργασίας εστιάζει στις τεχνολογίες από την πλευρά του server που αποτελούν τη βάση των διαδικτυακών εφαρμογών, επιτρέποντας τη δημιουργία, διαχείριση και κλιμάκωση δυναμικών υπηρεσιών. Το stack που επιλέχθηκε για αυτό το project περιλαμβάνει σύγχρονες, βασισμένες σε JavaScript τεχνολογίες που εξασφαλίζουν ένα αποτελεσματικό περιβάλλον ανάπτυξης.

2.4.1 Node.js

Το Node.js είναι ένα runtime environment που επιτρέπει την εκτέλεση JavaScript από την πλευρά του server, επεκτείνοντας την εμβέλεια του JavaScript πέρα από τον browser. Η non-blocking αρχιτεκτονική του το καθιστά ιδανικό για επεκτάσιμες εφαρμογές δικτύου.

Το non-blocking αναφέρεται σε μια προσέγγιση προγραμματισμού όπου ένα σύστημα μπορεί να συνεχίσει να εκτελεί άλλες εργασίες ενώ περιμένει να ολοκληρωθούν άλλες λειτουργίες, συνήθως εργασίες εισόδου/εξόδου (I/O). Σε ένα μοντέλο χωρίς αποκλεισμό, οι εργασίες που συνήθως προκαλούν το σύστημα σε αναμονή, όπως η ανάγνωση από ένα αρχείο, η πρόσβαση σε μια βάση δεδομένων ή η ανάκτηση δεδομένων μέσω ενός δικτύου, εκτελούνται με τρόπο που να μην εμποδίζουν την εκτέλεση επακόλουθων λειτουργιών . Αυτό επιτυγχάνεται με τη χρήση callbacks, promises ή άλλων ασύγχρονων μηχανισμών. Ως αποτέλεσμα, ένα non-blocking σύστημα μπορεί να χειρίζεται περισσότερες εργασίες ταυτόχρονα και να χρησιμοποιεί πόρους πιο αποτελεσματικά, οδηγώντας σε βελτιωμένη απόδοση εφαρμογών, ειδικά σε περιβάλλοντα που ασχολούνται με μεγάλο όγκο λειτουργιών I/O, όπως οι διακομιστές Ιστού.

Ακολουθούν τα βασικά χαρακτηριστικά του Node.js:

Event Loop: Στην καρδιά του non-blocking μοντέλου εισόδου/εξόδου του Node.js βρίσκεται το event loop. Αυτή η βασική δυνατότητα επιτρέπει στο Node.js να χειρίζεται πολλές λειτουργίες ταυτόχρονα χωρίς την επιβάρυνση της διαχείρισης νημάτων (threads). Το event loop διαχειρίζεται αποτελεσματικά τις ασύγχρονες λειτουργίες, επιτρέποντας γρήγορες και αποκριτικές εφαρμογές από την πλευρά του διακομιστή, καθιστώντας τον ιδιαίτερα κατάλληλο για το χειρισμό εργασιών υψηλής επισκεψιμότητας και έντασης δεδομένων.

NPM Ecosystem: Το Node.js μπορεί να υπερηφανεύεται για ένα τεράστιο αποθετήριο βιβλιοθηκών ανοιχτού κώδικα, προσβάσιμες μέσω npm (Node Package Manager), το οποίο βελτιώνει σημαντικά τις δυνατότητές του και μειώνει τον χρόνο ανάπτυξης. Αυτό το οικοσύστημα παρέχει στους προγραμματιστές μια ευρεία γκάμα εργαλείων και λειτουργικών μονάδων, εκσυγχρονίζοντας την ενσωμάτωση διαφόρων λειτουργιών σε εφαρμογές και ενισχύοντας την καινοτομία.

Ενιαία γλώσσα σε όλο το stack: Η χρήση JavaScript για ανάπτυξη τόσο στο front-end όσο και στο back-end βελτιστοποιεί τη διαδικασία ανάπτυξης. Αυτή η ενοποιημένη γλωσσική προσέγγιση επιτρέπει κοινό κώδικα και πόρους, ενισχύοντας έτσι μια πιο ολοκληρωμένη και αποτελεσματική ροή εργασίας. Οι προγραμματιστές επωφελούνται από το μειωμένο γνωστικό φορτίο και την αυξημένη παραγωγικότητα χρησιμοποιώντας μια συνεπή γλώσσα και τεχνολογική στοίβα σε ολόκληρη την εφαρμογή.

Microservices Oriented: Η ελαφριά φύση του Node.js το καθιστά ιδανικό υποψήφιο για αρχιτεκτονικές microservices. Προωθεί τις αρθρωτές και διατηρούμενες βάσεις κωδικών, επιτρέποντας την ανάπτυξη εφαρμογών ως μια σουίτα μικρών, διασυνδεδεμένων υπηρεσιών. Αυτή η αρχιτεκτονική είναι ολοένα και πιο δημοφιλής για την επεκτασιμότητα, την ευκολία ανάπτυξης και την ικανότητα απομόνωσης και αντιμετώπισης ζητημάτων σε συγκεκριμένα μέρη μιας εφαρμογής χωρίς να επηρεάζει το σύνολο.

Οφέλη του Node.js στην ανάπτυξη Ιστού

Οι μοναδικές δυνατότητες του Node.js προσφέρουν πολλά οφέλη στην ανάπτυξη ιστού:

Επεκτασιμότητα: Η αρχιτεκτονική του που βασίζεται σε events επιτρέπει τον χειρισμό μεγάλου αριθμού ταυτόχρονων συνδέσεων, καθιστώντας το εξαιρετικά επεκτάσιμο.

Απόδοση: Το μοντέλο I/O χωρίς αποκλεισμό εξασφαλίζει αποτελεσματική επεξεργασία, μειώνοντας τον χρόνο που απαιτείται για την εκτέλεση λειτουργιών και βελτιώνοντας τη συνολική απόδοση της εφαρμογής.

Υποστήριξη κοινότητας: Μια ισχυρή και ενεργή κοινότητα σημαίνει συνεχείς βελτιώσεις, πλήθος κοινών γνώσεων και άφθονους πόρους για τους προγραμματιστές.

Ευελιξία: Το Node.js χρησιμοποιείται σε διάφορες εφαρμογές, από εφαρμογές Ιστού έως REST API, εφαρμογές σε πραγματικό χρόνο όπως εφαρμογές συνομιλίας και παιχνιδιών, ακόμη και συσκευές Internet of Things (IoT).

Συμπερασματικά, το Node.js έχει επηρεάσει βαθιά τη σύγχρονη ανάπτυξη ιστού, προσφέροντας μια ευέλικτη, αποτελεσματική και επεκτάσιμη λύση για προγραμματισμό από την πλευρά του διακομιστή. Η ικανότητά του να χειρίζεται ασύγχρονες λειτουργίες, μαζί με την εκτεταμένη αρχιτεκτονική φιλική προς το οικοσύστημα και τις μικροϋπηρεσίες, το καθιστά μια προτιμώμενη επιλογή για προγραμματιστές που θέλουν να δημιουργήσουν γρήγορες, επεκτάσιμες και αποτελεσματικές εφαρμογές Ιστού.

2.4.2 Express

Το Express είναι ένα μινιμαλιστικό και ευέλικτο framework εφαρμογών ιστού Node.js που προσφέρει ένα ισχυρό σύνολο δυνατοτήτων για την ανάπτυξη εφαρμογών ιστού μιας σελίδας, πολλών σελίδων και υβριδικών εφαρμογών. Τα βασικά του οφέλη περιλαμβάνουν:

Middleware: Το Express χρησιμοποιεί ενδιάμεσο λογισμικό για να ανταποκρίνεται σε αιτήματα HTTP, καθιερώνοντας μια μέθοδο για την εκτέλεση κοινών εργασιών στον διακομιστή, όπως ελέγχους ελέγχου ταυτότητας, επικύρωση δεδομένων και πολλά άλλα. Οι συναρτήσεις Middleware στο Express έχουν πρόσβαση στο αντικείμενο αιτήματος (req), στο αντικείμενο απόκρισης (res) και στην επόμενη λειτουργία ενδιάμεσου λογισμικού στον κύκλο αίτησης-απόκρισης της εφαρμογής, επιτρέποντας στους προγραμματιστές να δημιουργήσουν αποτελεσματική και αρθρωτή λογική από την πλευρά του διακομιστή.

Δρομολόγηση: Περιλαμβάνει μια βιβλιοθήκη δρομολόγησης για τη διαχείριση αιτημάτων σε διαφορετικές διαδρομές URL, διευκολύνοντας την οργάνωση των τελικών σημείων της εφαρμογής. Αυτό το σύστημα δρομολόγησης επιτρέπει στους προγραμματιστές να ορίζουν διαδρομές με βάση μεθόδους HTTP και διευθύνσεις URL, επιτρέποντας τη δημιουργία ενός δομημένου και διατηρήσιμου API από την πλευρά του διακομιστή.

Απόδοση: Το Express έχει σχεδιαστεί για υψηλή απόδοση, ικανό να χειρίζεται πολλές ταυτόχρονες συνδέσεις με ελάχιστο κόστος. Η ελαφριά φύση του το καθιστά εξαιρετική επιλογή για εφαρμογές που απαιτούν γρήγορους χρόνους απόκρισης και αποτελεσματικό χειρισμό των λειτουργιών I/O.

Απλότητα και έλεγχος: Ενώ παρέχουν ένα λεπτό στρώμα βασικών χαρακτηριστικών εφαρμογών ιστού, οι προγραμματιστές διατηρούν τον έλεγχο, αποφεύγοντας τους περιορισμούς μεγαλύτερων πλαισίων. Η Express δίνει στους προγραμματιστές την ελευθερία να δομούν την εφαρμογή τους όπως τους βολεύει, επιτρέποντάς τους να χρησιμοποιούν όσες ή λιγότερες λειτουργίες Express χρειάζονται.

Πρόσθετα χαρακτηριστικά και ενσωμάτωση οικοσυστήματος

Template Engines: Το Express υποστηρίζει διάφορες μηχανές προτύπων, επιτρέποντας στους προγραμματιστές να αποδίδουν δυναμικό HTML στον διακομιστή. Αυτή η δυνατότητα είναι ζωτικής σημασίας για εφαρμογές που απαιτούν απόδοση από την πλευρά του διακομιστή για σκοπούς SEO ή για την αποστολή πλήρως διαμορφωμένων σελίδων στον πελάτη.

Κοινότητα και οικοσύστημα: Όντας ένα από τα πιο δημοφιλή Node.js framework, το Express διαθέτει μια μεγάλη και ενεργή κοινότητα. Αυτή η κοινότητα συμβάλλει σε ένα τεράστιο οικοσύστημα ενδιάμεσων λογισμικών και βοηθητικών μονάδων, παρέχοντας λύσεις σε κοινά προβλήματα ανάπτυξης ιστού και επεκτείνοντας τις δυνατότητες των εφαρμογών Express.

Ευελιξία και Επεκτασιμότητα: Η άγνωστη φύση του Express το καθιστά εξαιρετικά ευέλικτο, επιτρέποντάς του να χρησιμοποιείται ως η ραχοκοκαλιά οποιασδήποτε διαδικτυακής εφαρμογής, από έργα μικρής κλίμακας έως μεγάλα, πολύπλοκα συστήματα. Η επεκτασιμότητα του σημαίνει ότι μπορεί να ενσωματωθεί με άλλες λειτουργικές μονάδες και πλαίσια Node.js, όπως εκείνα για τη δημιουργία RESTful API ή υπηρεσίες επικοινωνίας σε πραγματικό χρόνο.

Συμπερασματικά, το Express αποτελεί βασικό framework στο οικοσύστημα Node.js, απλοποιώντας τη διαδικασία ανάπτυξης διαδικτυακών εφαρμογών ενώ προσφέρει την απόδοση, την απλότητα και την ευελιξία που απαιτούνται για τις σύγχρονες εφαρμογές Ιστού. Η μινιμαλιστική προσέγγισή του, σε συνδυασμό με τις ισχυρές του δυνατότητες δρομολόγησης και ενδιάμεσου λογισμικού, το καθιστούν ιδανική επιλογή για προγραμματιστές που θέλουν να δημιουργήσουν αποτελεσματικές, επεκτάσιμες και διατηρούμενες εφαρμογές web.

2.4.3 MongoDB

Η MongoDB είναι μια ισχυρή βάση δεδομένων NoSQL που έχει σχεδιαστεί για εύκολη ανάπτυξη και επεκτασιμότητα. Τα βασικά χαρακτηριστικά του περιλαμβάνουν:

Μοντέλα δεδομένων χωρίς σχήμα (Schema-less): Σε αντίθεση με τις σχεσιακές βάσεις δεδομένων, το MongoDB χρησιμοποιεί ένα ευέλικτο μοντέλο εγγράφου που μπορεί να αποθηκεύσει δεδομένα σε έγγραφα που μοιάζουν με JSON. Αυτή η ευελιξία επιτρέπει τον εύκολο χειρισμό πολύπλοκων και ετερογενών δεδομένων. Τα έγγραφα στη MongoDB είναι ελεύθερα να περιέχουν διαφορετικά πεδία και δομές, επιτρέποντας την αποθήκευση διαφορετικών τύπων δεδομένων χωρίς την ανάγκη για προκαθορισμένο σχήμα. Αυτή η προσέγγιση είναι ιδιαίτερα επωφελής για εφαρμογές που ασχολούνται με εξελισσόμενες απαιτήσεις δεδομένων ή μη δομημένα δεδομένα.

Επεκτασιμότητα: Η αρχιτεκτονική της MongoDB επιτρέπει την οριζόντια επεκτασιμότητα, διευκολύνοντας τη διανομή δεδομένων σε πολλούς διακομιστές καθώς αυξάνονται οι ανάγκες των εφαρμογών. Αυτή η επεκτασιμότητα επιτυγχάνεται μέσω λειτουργιών όπως η κοινή χρήση, όπου τα δεδομένα κατανέμονται σε ένα σύμπλεγμα μηχανών, βελτιώνοντας την απόδοση ανάγνωσης/εγγραφής και επιτρέποντας το χειρισμό λειτουργιών δεδομένων μεγάλης κλίμακας χωρίς συμβιβασμούς στην απόδοση.

Ευρετηρίαση (indexing): Οι αποτελεσματικές δυνατότητες ευρετηρίασης της MongoDB εξασφαλίζουν υψηλή απόδοση σε μεγάλα σύνολα δεδομένων, επιτρέποντας ταχύτερες απαντήσεις ερωτημάτων. Τα ευρετήρια στη MongoDB μπορούν να δημιουργηθούν σε οποιοδήποτε πεδίο ενός εγγράφου, συμπεριλαμβανομένων πεδίων μέσα σε ένθετα έγγραφα και πίνακες, βελτιώνοντας σημαντικά την ταχύτητα και την αποτελεσματικότητα της ανάκτησης δεδομένων.

Πρόσθετα χαρακτηριστικά του MongoDB:

Aggregation Framework: Η MongoDB περιλαμβάνει ένα ισχυρό Aggregation Framework για πολύπλοκη επεξεργασία δεδομένων και μετασχηματισμούς, παρόμοιο με τη λειτουργικότητα GROUP BY της SQL, αλλά πιο ευέλικτο και ισχυρό.

Αναπαραγωγή και υψηλή διαθεσιμότητα: Η MongoDB υποστηρίζει την αναπαραγωγή, επιτρέποντας πλεονασμό δεδομένων και υψηλή διαθεσιμότητα. Αυτή η δυνατότητα διασφαλίζει ότι τα δεδομένα αντιγράφονται σε πολλούς διακομιστές, προστατεύοντας από αστοχίες υλικού και διασφαλίζοντας συνεχή διαθεσιμότητα δεδομένων.

Ευέλικτο και ποικίλο Querying: Η MongoDB υποστηρίζει ένα πλούσιο σύνολο δυνατοτήτων ερωτημάτων, συμπεριλαμβανομένων των ερωτημάτων πεδίου, εύρους, αναζητήσεων κανονικών εκφράσεων και πολλά άλλα. Αυτή η ευελιξία τη καθιστά κατάλληλη για ένα ευρύ φάσμα εφαρμογών, από απλές λειτουργίες αναζήτησης έως σύνθετες αναλύσεις δεδομένων.

Έγγραφα τύπου JSON και μορφή BSON: Η MongoDB αποθηκεύει δεδομένα σε μια μορφή που ονομάζεται BSON (Binary JSON), η οποία επεκτείνει το μοντέλο JSON για να παρέχει πρόσθετους τύπους δεδομένων και να είναι αποτελεσματική για κωδικοποίηση και αποκωδικοποίηση σε διαφορετικές γλώσσες.

Συμπερασματικά, η MongoDB προσφέρει μια επεκτάσιμη λύση υψηλής απόδοσης για τις σύγχρονες ανάγκες εφαρμογών. Ο No-SQL σχεδιασμος, η ισχυρή ευρετηρίαση και οι ισχυρές δυνατότητες αναζήτησης ερωτημάτων το καθιστούν κατάλληλο για το χειρισμό ενός ευρέος φάσματος τύπων και δομών δεδομένων, από απλές εφαρμογές έως πολύπλοκα συστήματα μεγάλης κλίμακας. Η αρχιτεκτονική της MongoDB είναι ιδιαίτερα ευνοϊκή για εφαρμογές που απαιτούν γρήγορη επανάληψη, ευέλικτα μοντέλα δεδομένων και δυνατότητα κλιμάκωσης χωρίς κόπο.

2.4.4 Mongoose

Το Mongoose είναι μια ισχυρή βιβλιοθήκη Object Data Modeling (ODM) για MongoDB και Node.js που διευκολύνει τη διαχείριση των σχέσεων δεδομένων, την επικύρωση σχήματος και τη μετάφραση μεταξύ αντικειμένων στον κώδικα και την αναπαράστασή τους στη MongoDB. Τα κύρια χαρακτηριστικά του περιλαμβάνουν:

Σχήματα: Το Mongoose επιβάλλει τη δομή στις συλλογές ορίζοντας μοντέλα σχημάτων. Αυτή η επιβολή δομής καθιστά τον χειρισμό δεδομένων πιο προβλέψιμο και αξιόπιστο. Τα σχήματα στο Mongoose περιγράφουν το σχήμα των εγγράφων σε μια συλλογή MongoDB, συμπεριλαμβανομένων των τύπων πεδίων, των προεπιλεγμένων τιμών και άλλων. Αυτή η δομή παρέχει ένα σαφές σχέδιο για τα δεδομένα, διασφαλίζοντας ότι όλα τα έγγραφα σε μια συλλογή τηρούν μια καθορισμένη μορφή.

Επικύρωση: Το Mongoose παρέχει ενσωματωμένη επικύρωση, διασφαλίζοντας την ακεραιότητα των δεδομένων επιβάλλοντας συγκεκριμένους κανόνες και τύπους δεδομένων πριν από την εισαγωγή τους στη βάση δεδομένων. Αυτή η δυνατότητα είναι ζωτικής σημασίας για τη διατήρηση της ποιότητας και της συνέπειας των δεδομένων, αποτρέποντας την αποθήκευση μη έγκυρων ή εσφαλμένων δεδομένων.

Query Building: Το API της Mongoose προσφέρει ένα πλούσιο σύνολο δυνατοτήτων αναζήτησης, απλοποιώντας το έργο της κατασκευής και εκτέλεσης ερωτημάτων βάσης δεδομένων. Το API παρέχει μια ευχάριστη και ευέλικτη διεπαφή για την υποβολή ερωτημάτων στη MongoDB, επιτρέποντας πολύπλοκες λειτουργίες φιλτραρίσματος, ταξινόμησης και συγκέντρωσης, γεγονός που καθιστά την ανάκτηση δεδομένων αποτελεσματική και απλή.

Middleware: Παρόμοια με το Express, το Mongoose χρησιμοποιεί ενδιάμεσο λογισμικό για να εφαρμόσει άγκιστρα στον κύκλο ζωής του εγγράφου, επιτρέποντας την εκτέλεση λογικής πριν και μετά τις λειτουργίες. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη για εργασίες όπως η επικύρωση δεδομένων, η τροποποίηση εγγράφων πριν από την αποθήκευση ή οι λειτουργίες καταγραφής κατά τη διάρκεια του κύκλου ζωής του εγγράφου.

Πρόσθετα πλεονεκτήματα της χρήσης Mongoose

Αντιστοίχιση αντικειμένων: Η Mongoose αντιστοιχίζει έγγραφα σε αντικείμενα, παρέχοντας μια οικεία διεπαφή για προγραμματιστές που έχουν συνηθίσει να εργάζονται με αντικείμενα σε JavaScript. Αυτή η αφαίρεση απλοποιεί τις αλληλεπιδράσεις με τη MongoDB, καθιστώντας πιο διαισθητική την εργασία με δεδομένα.

Χειρισμός σχέσεων: Το Mongoose επιτρέπει τον ορισμό σχέσεων μεταξύ διαφορετικών σχημάτων χρησιμοποιώντας αναφορές ή ενσωματωμένα έγγραφα, διευκολύνοντας την αναπαράσταση πολύπλοκων, σχεσιακών δομών δεδομένων σε μια βάση δεδομένων NoSQL.

Οικοσύστημα Plugin: Το Mongoose υποστηρίζει πρόσθετα, επιτρέποντας την επέκταση της λειτουργικότητάς του. Αυτό το οικοσύστημα προσθηκών μπορεί να προσθέσει πρόσθετες δυνατότητες σε σχήματα ή να παρέχει επαναχρησιμοποιήσιμα στοιχεία, ενισχύοντας τις δυνατότητες των εφαρμογών Mongoose.

Συμπερασματικά, το Mongoose είναι ένα απαραίτητο εργαλείο στη στοίβα Node.js και MongoDB, προσφέροντας μια εξελιγμένη προσέγγιση στη διαχείριση δεδομένων. Τα σχήματα, η επικύρωση, η δημιουργία ερωτημάτων και οι λειτουργίες του ενδιάμεσου λογισμικού απλοποιούν τη διαδικασία εργασίας με τη MongoDB, καθιστώντας το πιο αποτελεσματικό, ισχυρό και φιλικό προς τους προγραμματιστές. Το Mongoose γεφυρώνει το χάσμα μεταξύ της ευελιξίας των βάσεων δεδομένων NoSQL και της ανάγκης για δομημένη διαχείριση δεδομένων, διαδραματίζοντας κρίσιμο ρόλο στην ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών.

Κεφάλαιο 3: Υλοποίηση Backend

Το backend ενός συστήματος λογισμικού είναι το θεμελιώδες μέρος του που ασχολείται την επεξεργασία, την αποθήκευση και τη διαχείριση των δεδομένων. Λειτουργεί στα παρασκήνια, μακριά από το γραφικό περιβάλλον των τελικών χρηστών, και διασφαλίζει την ανταπόκριση, την αποτελεσματικότητα και την ασφάλεια του συστήματος. Σε αυτό το κεφάλαιο θα δούμε το backend σύστημα που υλοποιήθηκε για τη συγκεκριμένη εργασία.

3.1 Setup Περιβάλλοντος

Όταν ξεκινάμε να εργαζόμαστε σε ένα back-end για μια εφαρμογή με το Node.js, πρέπει πρώτα να δημιουργήσουμε το package.json που είναι ένα json αρχείο με τις βιβλιοθήκες που χρειάζονται για το project. Το αρχείο αυτό δημιουργείται με την εντομή

```
npm init
```

3.1.1 Μεταβλητές Περιβάλλοντος

Στη συνέχεια, δεν θέλουμε να έχουμε κωδικούς πρόσβασης ή μυστικά κλειδιά απευθείας στον κώδικα της εφαρμογής σας, επειδή αυτό δεν είναι ασφαλές. Αντίθετα, χρησιμοποιούμε κάτι που λέγεται μεταβλητές περιβάλλοντος (environment variables) για να διατηρήσουμε αυτές τις πληροφορίες κρυφές,

Στο πλαίσιο αυτού του project, το nodemon.json χρησιμοποιείται κατά τη φάση ανάπτυξης για την εισαγωγή μεταβλητών περιβάλλοντος στην εφαρμογή Node.js. Αυτό το αρχείο αναφέρεται από το nodemon, ένα βοηθητικό πρόγραμμα που διευκολύνει την αυτόματη επανεκκίνηση του διακομιστή κατά τον εντοπισμό αλλαγών στη βάση κώδικα. Είναι ανάγκη να σημειωθεί ότι το nodemon.json δεν προορίζεται για την αποθήκευση ευαίσθητων πληροφοριών αλλά για τον καθορισμό μεταβλητών που εξορθολογίζουν τη διαδικασία ανάπτυξης. Για περιβάλλοντα παραγωγής, απαιτείται μια πιο ασφαλής προσέγγιση, όπως η χρήση ενός αποκλειστικού συστήματος διαμόρφωσης περιβάλλοντος ή μιας υπηρεσίας που ειδικεύεται στη μυστική διαχείριση.

3.1.2 app.js

Το entry point μιας εφαρμογής Node.js είναι συνήθως ένα αρχείο JavaScript που προετοιμάζει τον διακομιστή εφαρμογών και ρυθμίζει αρχικό ενδιάμεσο λογισμικό, διαδρομές και συνδέσεις βάσης δεδομένων. Στην περίπτωση αυτού του project, το app.js είναι το script αυτό. Είναι υπεύθυνο για την εγκατάσταση της εφαρμογής Express, την ενσωμάτωση του ενδιάμεσου λογισμικού και τη σύνδεση στη βάση δεδομένων MongoDB μέσω του Mongoose. Η καθιέρωση του app.js ως αρχικού σημείου επαφής για το σύστημα υποστήριξης είναι μια τυπική σύμβαση που ενισχύει την κατανόηση και την πλοήγηση του κώδικα.

3.2 Δομή του backend

3.2.1 Επισκόπηση δομής Backend:

app.js: Αυτό είναι το entry point της εφαρμογής. Ρυθμίζει τον διακομιστή Express, το ενδιάμεσο λογισμικό, τις διαδρομές, τον χειρισμό σφαλμάτων και τη σύνδεση της βάσης δεδομένων.

controllers: Αυτός ο φάκελος περιέχει data-controller.js, το οποίο χειρίζεται την επιχειρηματική λογική για τις διαδρομές. Ο ελεγκτής αλληλεπιδρά με τα μοντέλα για τη δημιουργία, ανάγνωση, ενημέρωση και διαγραφή πόρων (λειτουργίες CRUD) και αποστολή απαντήσεων στον πελάτη.

models: Αυτός ο φάκελος περιέχει σχήματα Mongoose (user.js, marker.js, polygon.js, http-error.js) που ορίζουν τη δομή των δεδομένων και τα μοντέλα που αλληλεπιδρούν με τη βάση δεδομένων.

routes: Αυτός ο φάκελος περιέχει data-routes.js που ορίζει τα endpoints του API και τα συσχετίζει με τις αντίστοιχες λειτουργίες του ελεγκτή.

package.json & package-lock.json: Αυτά τα αρχεία παρακολουθούν τις εξαρτήσεις του έργου και τις συγκεκριμένες εκδόσεις τους.

node_modules: Ένας φάκελος που δημιουργήθηκε από το npm για να κρατήσει τα εγκατεστημένα πακέτα.

nodemon.json: Διαμόρφωση για το Nodemon, ένα βοηθητικό πρόγραμμα που παρακολουθεί τυχόν αλλαγές στην πηγή και επανεκκινεί αυτόματα τον διακομιστή.

3.2.2 Σκοπός κάθε φακέλου και αρχείου

Controllers:

data-controller.js: Περιέχει λειτουργίες που σχετίζονται με κάθε διαδρομή για τη διαχείριση αιτημάτων για χειρισμό marker και πολυγώνων, καθώς και έλεγχο ταυτότητας χρήστη.

Models:

http-error.js: Ορίζει μια προσαρμοσμένη κλάση σφάλματος για την τυποποίηση των απαντήσεων σφαλμάτων.

user.js: Ορίζει ένα σχήμα για δεδομένα χρήστη και ένα μοντέλο για λειτουργίες βάσης δεδομένων που σχετίζονται με το χρήστη.

marker.js: Ορίζει ένα σχήμα για δεδομένα δείκτη (σημεία σε χάρτη) και ένα μοντέλο για λειτουργίες βάσης δεδομένων που σχετίζονται με δείκτη.

polygon.js: Ορίζει ένα σχήμα για δεδομένα πολυγώνου (γεωμετρικά σχήματα σε χάρτη) και ένα μοντέλο για λειτουργίες βάσης δεδομένων που σχετίζονται με πολύγωνα.
Διαδρομές

data-routes.js: Αυτό το αρχείο θα καθορίζει τα endpoints διαδρομής και τις μεθόδους HTTP (GET, POST, PATCH, DELETE) και θα τα συνδέει με τις κατάλληλες λειτουργίες ελεγκτή.

3.3 Schemas βάσης δεδομένων

Μέσα στο περιβάλλον node.js, η mongoose λειτουργεί ως η βιβλιοθήκη Μοντελοποίησης Δεδομένων Αντικειμένων (ODM), προσφέροντας μια λύση βασισμένη σε σχήματα για τη μοντελοποίηση των δεδομένων της εφαρμογής. Οι ορισμοί του υπό εξέταση σχήματος είναι αναπόσπαστοι στον μηχανισμό αποθήκευσης γεωχωρικών χαρακτηριστικών, αντανακλώντας μια καλά δομημένη προσέγγιση για την αναπαράσταση δεδομένων στο MongoDB.

3.3.1 Marker

Το Schema του Marker ορίζεται ως το σημείο στο χάρτη και αποτελείται από τα παρακάτω χαρακτηριστικά:

Type: Το πεδίο type είναι προεπιλεγμένο σε "Feature", ευθυγραμμιζόμενο με την προδιαγραφή GeoJSON, η οποία είναι μια τυπική μορφή για την κωδικοποίηση μιας ποικιλίας δομών γεωγραφικών δεδομένων.
Geometry: Στο πλαίσιο της γεωμετρίας, η δομή των δεδομένων είναι διπλή. Πρώτον, καθορίζει τον δικό του τύπο (από προεπιλογή "Point") για να οριοθετήσει τη φύση των γεωχωρικών δεδομένων, σε αυτήν την περίπτωση, υποδεικνύοντας ένα ζεύγος συντεταγμένων. Δεύτερον, ο πίνακας συντεταγμένων ορίζεται για να διατηρούν τις τιμές γεωγραφικού μήκους και γεωγραφικού πλάτους, και ορίζεται όπως απαιτείται, διασφαλίζοντας ότι δεν μπορεί να υπάρξει δείκτης χωρίς μια καθορισμένη θέση.
Properties: Το properties document ενσωματώνει πρόσθετες πληροφορίες σχετικά με το Point, όπως όνομα και ενορία, με τα δύο πεδία να επισημαίνονται ως απαιτούμενα. Αυτό υποδηλώνει ότι κάθε Point δεν είναι μόνο μια γεωγραφική οντότητα αλλά περιέχει επίσης σημαντικές πληροφορίες που σχετίζονται με τον τομέα εφαρμογής, όπως διοικητικές διαιρέσεις ή σημεία πολιτιστικού ενδιαφέροντος.

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const markerSchema = new Schema({
  type: {
    type: String,
    default: "Feature",
  },
  geometry: {
    type: {
      type: String,
      default: "Point",
    },
    coordinates: [
      { type: Number, required: true },
      { type: Number, required: true },
    ],
  },
  properties: {
    name: { type: String, required: true },
    parish: { type: String, required: true },
  },
});
const MarkerModel = mongoose.model("MarkerDocument", markerSchema, "Markers");
module.exports = {
  MarkerModel,
};
```

Εικόνα 1: Το Schema του Marker

3.3.2 Polygon

Το polygon Schema ορίζει μια πολυγωνική περιοχή, που χρησιμοποιείται συνήθως για την αναπαράσταση ορίων ή ζωνών σε έναν χάρτη.

Έχει τα παρακάτω χαρακτηριστικά:

Type: Ομοίως με το marker, ο τύπος έχει προκαθοριστεί σε "Feature" για να συμμορφώνεται με τη μορφή GeoJSON,

Geometry: Το τμήμα Geometry εδώ διαφοροποιείται δηλώνοντας τον τύπο του ως "Polygon". Το πεδίο συντεταγμένων είναι δομημένο ως ένας πίνακας πινάκων για να συλλάβει μια ακολουθία σημείων που, όταν συνδέονται, σχηματίζουν την περιφέρεια του πολυγώνου. Αυτή η ένθεση είναι απαραίτητη για την αναπαράσταση της πολυπλοκότητας ενός πολυγώνου που μπορεί να περιέχει πολλαπλούς γραμμικούς δακτυλίους - ο πρώτος αντιπροσωπεύει το εξωτερικό όριο, με οποιεσδήποτε επόμενες συστοιχίες οριοθετούν οπές εντός του πολυγώνου.

Properties: Σε αυτό το σχήμα, το πεδίο ιδιοτήτων περιέχει μόνο ένα χαρακτηριστικό όνομα. Αυτό υποδηλώνει ότι τα πολύγωνα στην εφαρμογή χρησιμοποιούνται για μια απλούστερη κατηγοριοποίηση του χώρου, χωρίς την ανάγκη για εκτενή περιγραφικά δεδομένα και συγκεκριμένα για τα πολύγωνα.

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const polygonSchema = new Schema({
  type: {
    type: String,
    default: "Feature",
  },
  geometry: {
    type: {
      type: String,
      default: "Polygon",
    },
    coordinates: [
      [
        { type: Number, required: true },
        { type: Number, required: true },
      ],
    ],
  },
  properties: {
    name: { type: String, required: true },
  },
});
const PolygonModel = mongoose.model(
  "PolygonDocument",
  polygonSchema,
  "Polygons"
);
module.exports = { PolygonModel };
```

Εικόνα 2: Το Schema του Polygon

3.3.3 User

Το user Schema είναι θεμελιωδώς διαφορετικό από τα δύο προηγούμενα γεωχωρικά σχήματα, εστιάζοντας αντ' αυτού στο επίπεδο διαχείρισης της εφαρμογής.

Χαρακτηριστικά χρήστη: Εδώ, παρατηρούμε τον ορισμό του ονόματος χρήστη και του κωδικού πρόσβασης, και τα δύο επισημαίνονται ως απαιτούμενα. Αυτό τονίζει έναν σχεδιασμό συστήματος που δίνει προτεραιότητα στον έλεγχο ταυτότητας, αναγνωρίζοντας έτσι την κρισιμότητα της ακεραιότητας και της ασφάλειας των δεδομένων σε γεωχωρικές εφαρμογές.

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const userSchema = new Schema({
  username: { type: String, required: true },
  password: { type: String, required: true },
});
const UserModel = mongoose.model("UserDocument", userSchema, "Admins");
module.exports = {
  UserModel,
};
```

Εικόνα 3: Το Schema του User

Για κάθε schema, ένα αντίστοιχο μοντέλο δημιουργείται χρησιμοποιώντας mongoose.model(), δεσμεύοντας αποτελεσματικά το σχήμα σε μια συλλογή MongoDB. Η ρητή αναφορά των ονομάτων συλλογών - "Markers", "Polygons" και "Admins" - διευκολύνει την άμεση αντιστοίχιση μεταξύ του μοντέλου Mongoose και της αντίστοιχης συλλογής στη βάση δεδομένων. Αυτή η δέσμευση είναι ουσιαστική, καθώς επιτρέπει τον επακόλουθο χειρισμό των εγγράφων σε αυτές τις συλλογές μέσω των κατασκευασμένων μοντέλων.

3.4 app.js

Το αρχείο app.js σε μια εφαρμογή Node.js, όπως αναφέρθηκε και νωρίτερα, χρησιμεύει ως η καρδιά της διαμόρφωσης διακομιστή. Διευκολύνει τις αλληλεπιδράσεις μεταξύ της πλευράς του πελάτη και του διακομιστή μέσω μιας σειράς ενδιάμεσων προγραμμάτων και διαδρομών. Ο υπό ανάλυση κώδικας περιγράφει λεπτομερώς τη ρύθμιση ενός διακομιστή Express.js που είναι ενσωματωμένος σε μια βάση δεδομένων MongoDB, παρέχοντας πληροφορίες για τον τρόπο με τον οποίο ο διακομιστής χειρίζεται αιτήματα, απαντήσεις και συνδεσιμότητα βάσης δεδομένων.

Αρχικοποίηση Express

```
const express = require("express");
const app = express();
```

Εικόνα 4: Αρχικοποίηση Express

Body Parsing Middleware:

Το Body-Parser είναι μια λειτουργική μονάδα ενδιάμεσου λογισμικού για το Express.js που αναλύει τα body των εισερχόμενων αιτημάτων σε ένα ενδιάμεσο λογισμικό πριν από τους handlers, που διατίθεται στο req.body. Η χρήση του bodyParser.json() είναι μια δήλωση ότι ο διακομιστής αναμένει να λάβει και να αναλύσει ωφέλιμα φορτία JSON, μια κοινή μορφή ανταλλαγής δεδομένων στις σύγχρονες υπηρεσίες Ιστού.

```
const bodyParser = require("body-parser");
app.use(bodyParser.json());
```

Εικόνα 5: Δήλωση body-parser

CORS Configuration:

Εδώ, ο διακομιστής διαμορφώνει την πολιτική CORS. Αυτό είναι ένα κρίσιμο στοιχείο στην ασφάλεια, το οποίο περιορίζει τα προγράμματα περιήγησης ιστού από το να υποβάλλουν αιτήματα σε έναν τομέα διαφορετικό από αυτόν που εξυπηρετούσε την ιστοσελίδα. Η ρύθμιση των κεφαλίδων CORS όπως παραπάνω επιτρέπει σε οποιαδήποτε προέλευση να έχει πρόσβαση στους πόρους, οι οποίοι είναι κατάλληλοι για δημόσια API ή κατά τη φάση ανάπτυξης.

```
app.use((req, res, next) => {
  res.setHeader("Access-Control-Allow-Origin", "*");
  res.setHeader(
    "Access-Control-Allow-Headers",
    "Origin, X-Requested-With, Content-Type, Accept, Authorization"
  );
  res.setHeader("Access-Control-Allow-Methods", "GET, POST, PATCH, DELETE");

  next();
});
```

Εικόνα 6: CORS Configuration:

Routing Middleware:

```
const dataRoutes = require("./routes/data-routes");
app.use("/api", dataRoutes);
```

Εικόνα 7: Δρομολόγηση με το dataRoutes

Ο διακομιστής οριοθετεί τη λογική δρομολόγησης που σχετίζεται με τα δεδομένα σε μια ξεχωριστή μονάδα, βελτιώνοντας τη σπονδυλωτότητα. Τα αιτήματα για διαδρομές που ξεκινούν με /api ανατίθενται σε dataRoutes.

Σύνδεση με τη βάση δεδομένων

```
mongoose
  .connect(`mongodb+srv://...`)
  .then(() => {
    app.listen(process.env.PORT);
  })
  .catch((err) => {
    console.log(err);
  });
```

Εικόνα 8: Σύνδεση με τη βάση δεδομένων

Ο διακομιστής δημιουργεί μια σύνδεση με το MongoDB χρησιμοποιώντας το Mongoose. Η συμβολοσειρά σύνδεσης, που περιέχει ευαίσθητα credentials, είναι εύστοχα κρυμμένη χρησιμοποιώντας μεταβλητές περιβάλλοντος, ευθυγραμμισμένη με τις βέλτιστες πρακτικές ασφάλειας. Μετά την επιτυχή σύνδεση, ο διακομιστής αρχίζει να ακούει στη θύρα που καθορίζεται από τη μεταβλητή περιβάλλοντος process.env.PORT, σηματοδοτώντας την ετοιμότητα του διακομιστή να χειριστεί τα εισερχόμενα αιτήματα.

3.5 data-controllers.js

Το αρχείο data-controller.js είναι σχεδιασμένο για να χειρίζεται διάφορα αιτήματα HTTP που σχετίζονται με γεωγραφικά δεδομένα, όπως Point και πολύγωνα. Το αρχείο εξάγει ένα σύνολο ασύγχρονων συναρτήσεων (middleware) που αλληλεπιδρούν με μια βάση δεδομένων MongoDB χρησιμοποιώντας μοντέλα Mongoose. Ακολουθεί μια ανάλυση κάθε ενότητας και λειτουργίας στο αρχείο:

Εξαρτήσεις και αρχικοποίηση δεδομένων:

HttpError: Προσαρμοσμένη μονάδα χειρισμού σφαλμάτων.

MarkerModel, PolygonModel, UserModel: Mongoose μοντέλα για τη δομή δεδομένων της εφαρμογής.

admins: Μια σειρά από χρήστες διαχειριστή για σκοπούς σύνδεσης.

data: HardCoded δεδομένα που αντιπροσωπεύουν μια συλλογή γεωγραφικών χαρακτηριστικών, που πιθανότατα χρησιμοποιούνται πριν από την ενοποίηση της βάσης δεδομένων.

HTTP Request Handlers:

getAllData: Ανακτά ασύγχρονα όλα τα δεδομένα πολυγώνων και σημείων από τη βάση δεδομένων, συνενώνει τα αποτελέσματα, τα μετατρέπει σε μορφή φιλική προς το JSON χρησιμοποιώντας το toObject και τα στέλνει πίσω στον πελάτη με status code 200

.

```
const getAllData = async (req, res, next) => {
  let  features=[] ;
  try {
    const polygons = await PolygonModel.find({});
    const markers = await MarkerModel.find({});
console.log(polygons)
console.log(markers)
    features  = polygons.concat(markers)
  } catch (err) {
    const error = new HttpError("Σφάλμα κατά την ανάκτηση των δεδομένων:", 500);
    return next(error);
  }
  features =  features.map((p) =>
    p.toObject({ getters: true })
  );
  res.status(200).json({ features});
};
```

Εικόνα 9: Κώδικας του ανάκτησης δεδομένων

createMarker: Δημιουργεί ασύγχρονα ένα νέο point στη βάση δεδομένων με τα δεδομένα που παρέχονται στο body του αιτήματος, συμπεριλαμβανομένου ενός νέου μοναδικού αναγνωριστικού, και στέλνει πίσω τον δείκτη που δημιουργήθηκε με έναν κωδικό κατάστασης 201 που υποδεικνύει την επιτυχή δημιουργία.

```
const createMarker = async (req, res, next) => {
  const { name, coordinates, parish } = req.body;
  const createdMarker = new MarkerModel({
    properties: { name: name, id: uuidv4(), parish: parish },
    geometry: { coordinates: coordinates },
  });
  try {
    await createdMarker.save();
  } catch (err) {
    const error = new HttpError(
      "Creating marker failed, please try again",
      500
    );
    return next(error);
  }
  res.status(201).json({ marker: createdMarker });
};
```

Εικόνα 10: Κώδικας δημιουργίας νέου σημείου

updateMarker: Ενημερώνει ασύγχρονα έναν υπάρχοντα Point που προσδιορίζεται από το req.params.mid με τα νέα δεδομένα που παρέχονται στο σώμα του αιτήματος και στέλνει πίσω τα ενημερωμένα δεδομένα δείκτη.

```
const updateMarker = async (req, res, next) => {
  const { name, coordinates, parish } = req.body;
  const id = req.params.mid;
  let marker;
  try {
    marker = await MarkerModel.findById(id);
  } catch (err) {
    const error = new HttpError(
      "Something went wrong,could not update the marker",
      500
    );
    return next(error);
  }
  marker.properties.name = name;
  marker.geometry.coordinates = coordinates;
  marker.properties.parish = parish;
  console.log(marker);
  try {
    await marker.save();
  } catch (err) {
    const error = new HttpError(
      "Something went wrong,could not update the marker",
      500
    );
    return next(error);
  }
  res.status(200).json({ marker: marker.toObject({ getters: true }) });
};
```

Εικόνα 11: Κώδικας επεξεργασίας σημείου

deleteMarker: Διαγράφει ασύγχρονα ένα Point από τη βάση δεδομένων που προσδιορίζεται από το req.params.mid και στέλνει ένα μήνυμα επιτυχίας κατά τη διαγραφή.

```
const deleteMarker = async (req, res, next) => {
  const id = req.params.mid;
  let marker;
  try {
    marker = await MarkerModel.findById(id);
  } catch (err) {
    const error = new HttpError(
      "Something went wrong,could not delete the marker",
      500
    );
    return next(error);
  }
  try {
    await marker.deleteOne();
  } catch (err) {
    const error = new HttpError(
      "Something went wrong,could not delete the marker",
      500
    );
    return next(error);
  }
  res.status(200).json({ message: "deleted place" });
};
```

Εικόνα 12: Κώδικας διαγραφής σημείο

createPolygon: Δημιουργεί ασύγχρονα ένα νέο πολύγωνο στη βάση δεδομένων με τα δεδομένα που παρέχονται στο σώμα του αιτήματος, παρόμοιο με το createMarker, και στέλνει πίσω το δημιουργημένο πολύγωνο.

```
const createPolygon = async (req, res, next) => {
  const { name, coordinates } = req.body;
  console.log(coordinates)
  const createdPolygon = new PolygonModel({
    properties: { name: name, id: uuidv4() },
    geometry: { coordinates: coordinates },
  });
  try {
    await createdPolygon.save();
  } catch (err) {
    const error = new HttpError(
      "Creating polygon failed, please try again",
      500
    );
    return next(error);
  }
  res.status(201).json({ polygon: createdPolygon });
};
```

Εικόνα 13: Κώδικας δημιουργίας νέου πολυγώνου

updatePolygon: Ενημερώνει ασύγχρονα ένα υπάρχον πολύγωνο στη βάση δεδομένων, όπως το updateMarker, και στέλνει πίσω τα ενημερωμένα δεδομένα πολυγώνου.

```
const updatePolygon = async (req, res, next) => {
  const { name, coordinates } = req.body;
  const id = req.params.pid;
  let polygon;
  try {
    polygon = await PolygonModel.findById(id);
  } catch (err) {
    const error = new HttpError(
      "Something went wrong,could not update the polygon",
      500
    );
    return next(error);
  }
  polygon.properties.name = name;
  polygon.geometry.coordinates = coordinates;
  try {
    await polygon.save();
  } catch (err) {
    const error = new HttpError(
      "Something went wrong,could not update the polygon",
      500
    );
    return next(error);
  }
  res.status(200).json({ feature: polygon });
};
```

Εικόνα 14: Κώδικας επεξεργασίας πολυγώνου

deletePolygon: Διαγράφει ασύγχρονα ένα πολύγωνο που προσδιορίζεται από το req.params.pid από τη βάση δεδομένων και στέλνει ένα μήνυμα επιτυχίας κατά τη διαγραφή.

```
const deletePolygon = async (req, res, next) => {
  const id = req.params.pid;
  let polygon;
  try {
    polygon = await PolygonModel.findById(id);
  } catch (err) {
    const error = new HttpError(
      "Something went wrong,could not delete the polygon",
      500
    );
    return next(error);
  }
  try {
    await polygon.deleteOne();
  } catch (err) {
    const error = new HttpError(
      "Something went wrong,could not delete the polygon",
      500
    );
    return next(error);
  }
  res.status(200).json({ message: "deleted polygon" });
};
```

Εικόνα 15: Κώδικας διαγραφής πολυγώνου

Έλεγχος ταυτότητας χρήστη:

login: Ελέγχει ασύγχρονα εάν ο χρήστης υπάρχει και εάν ο κωδικός πρόσβασης ταιριάζει με την εγγραφή στη βάση δεδομένων. Εάν ο έλεγχος ταυτότητας είναι επιτυχής, στέλνει πίσω ένα "Logged in!" μήνυμα, διαφορετικά, επιστρέφει σφάλμα.

```
const login = async (req, res, next) => {
  const { username, password } = req.body;
  let existingUser;
  try {
    existingUser = await UserModel.findOne({ username: username });
  } catch (err) {
    const error = new HttpError("Logging failed ,please try again later", 500);
    return next(error);
  }
  if (!existingUser || existingUser.password !== password) {
    const error = new HttpError("Invalid credentials,could not log in", 401);
    return next(error);
  }
  res.json({ message: "Logged in!" });
};
```

Εικόνα 16: Κώδικας ελέγχου login

Γενικά, το αρχείο data-controller.js είναι ένα κρίσιμο μέρος του backend της εφαρμογής, που έχει ως αποστολή τον χειρισμό της επιχειρηματικής λογικής που σχετίζεται με τη διαχείριση γεωχωρικών δεδομένων και τον έλεγχο ταυτότητας χρήστη.

3.6 Εκτέλεση

Το backend εκτελείται με την εντολή “npm start” και τρέχει στο port 5000 στο localhost (http://localhost:5000/).

Μπορούμε να δοκιμάσουμε το backend στον browser για να διαπιστώσουμε τη σωστή λειτουργία του. Για παράδειγμα μεταβούμε στο http://localhost:5000/api/data ενώ το backend λειτουργεί, θα δούμε όλα τα δεδομένα των σημείων και των πολυγώνων που έχουμε στη βάση.

Κεφάλαιο 4: Υλοποίηση Frontend

Το front end κομμάτι της εφαρμογής είναι ένα React Application, το οποίο εμφανίζει την διεπαφή στον χρήστη και κάνει δυνατή την αλληλεπίδραση με το σύστημα.

Στο αρχείο .env ορίζεται το URL του backend σαν μεταβλητή περιβάλλοντος

```
REACT_APP_BACKEND_URL=http://localhost:5000/api
```

Εικόνα 17: Δήλωση URL του backend σαν μεταβλητή περιβάλλοντος

Ο φάκελος src περιέχει τα αρχεία και καταλόγους που είναι τυπικοί για μια εφαρμογή React:

App.css: Το αρχείο CSS για το στυλ του στοιχείου App.

App.js: Αυτό είναι το κύριο στοιχείο React που χρησιμεύει ως σημείο εισόδου για την ιεραρχία στοιχείων React. Περιέχει τον Provider για το Authentication και κάνει render το <Map>

components: Ένας κατάλογος που περιέχει επαναχρησιμοποιήσιμα components React.

index.css: Το global στυλ για την εφαρμογή.

index.js: Το αρχείο καταχώρισης για την εφαρμογή React όπου το ριζικό στοιχείο αποδίδεται συνήθως στο DOM.

Layers: Περιέχει το MarkerLayer και το PolygonLayer.

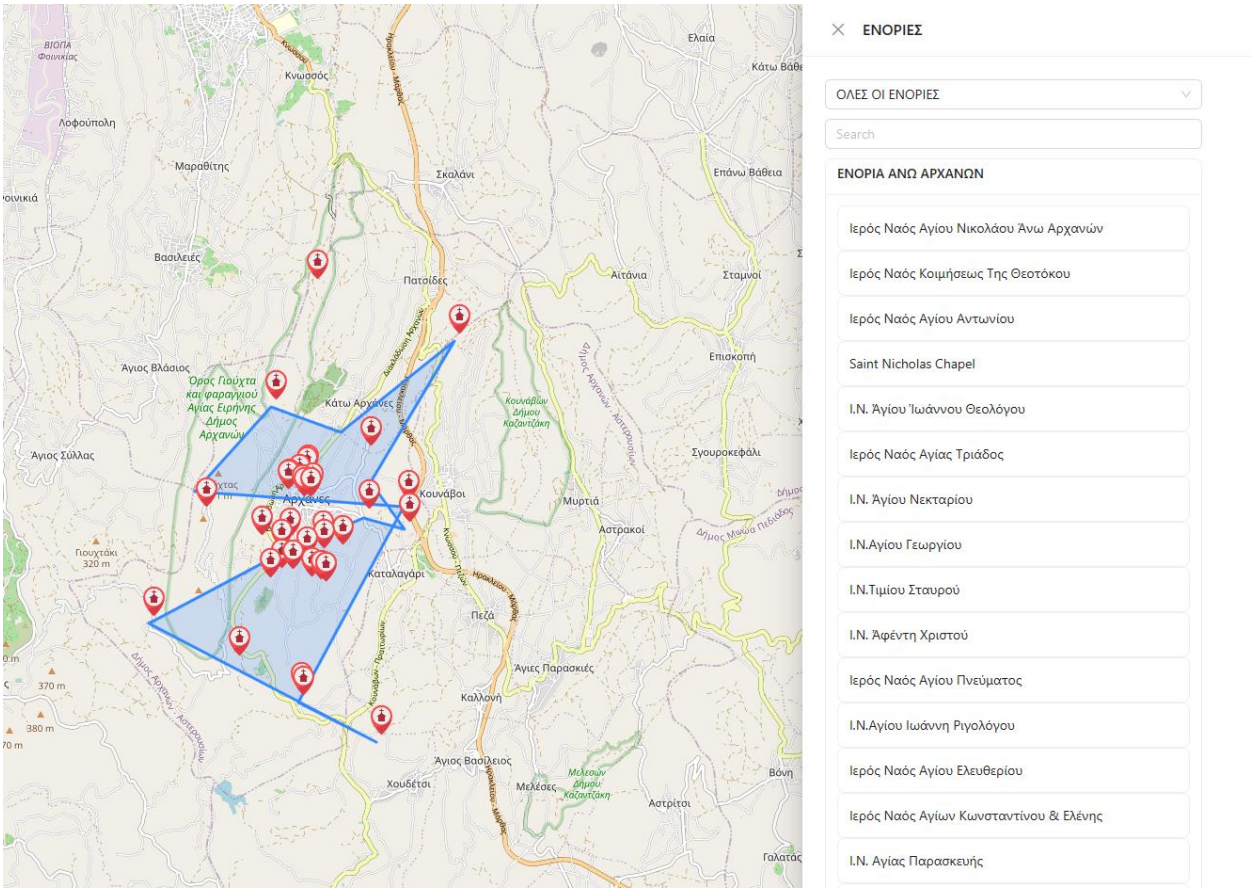
Η αρχιτεκτονική frontend της εφαρμογής αξιοποιεί τη δομή του React που βασίζεται σε component, προσφέροντας μια modular και επεκτάσιμη διεπαφή χρήστη. Η εφαρμογή χωρίζεται σε διακριτούς καταλόγους που αντικατοπτρίζουν τη λογική της δομή: components και layers.Ο κατάλογος components φιλοξενεί αυτόνομες λειτουργικές μονάδες που είναι υπεύθυνες για συγκεκριμένες λειτουργίες διεπαφής χρήστη, όπως έλεγχο ταυτότητας (LogInDrawer) και παρουσίαση δεδομένων (ListDrawer). Ο κατάλογος layers περιέχει αφαιρέσεις για λειτουργίες που σχετίζονται με χάρτη, διαχωρισμό σημείων (MarkerLayer) και πολύγωνων (PolygonLayer) σε ξεχωριστά επίπεδα.

4.1 Components

Τα Components είναι τα δομικά στοιχεία της διεπαφής χρήστη της εφαρμογής.

ListDrawer Component

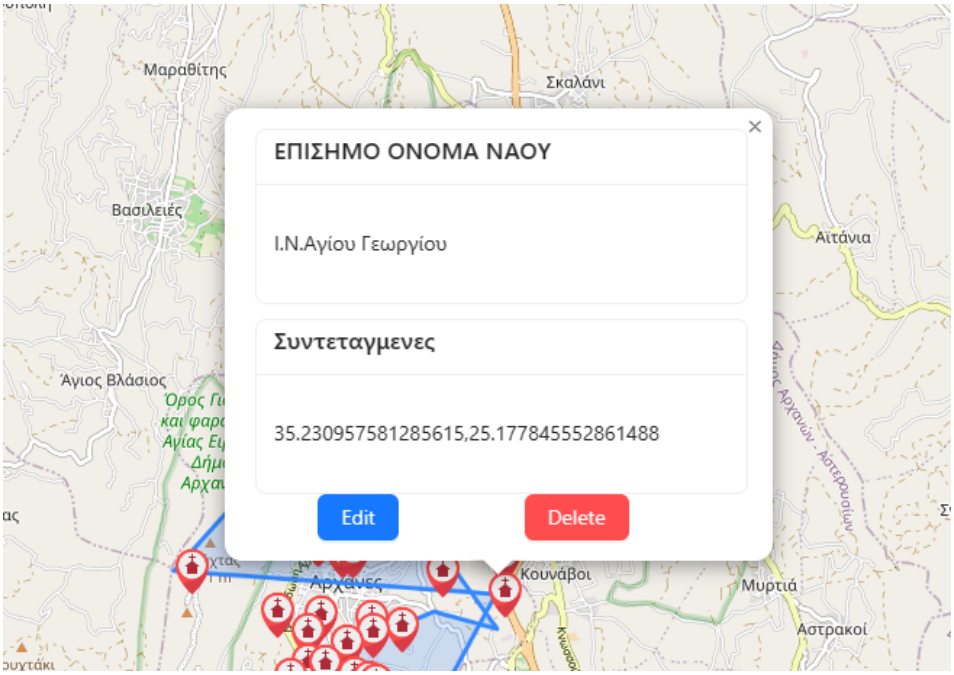
Το ListDrawer Component είναι μια πλαϊνή γραμμή που παραθέτει στοιχεία με γεωγραφικές οντότητες— με τα οποία μπορούν να αλληλεπιδράσουν οι χρήστες. Το στοιχείο εισάγει στοιχεία διεπαφής χρήστη από το antd, μια βιβλιοθήκη διεπαφής χρήστη, και χρησιμοποιεί το leaflet για λειτουργίες που σχετίζονται με χάρτη.



Εικόνα 18: Το ListDrawer Component

MarkerLayer Component

Το MarkerLayer είναι υπεύθυνο για την απόδοση σημείων χάρτη και το χειρισμό των αλληλεπιδράσεων των χρηστών με αυτούς τους δείκτες, όπως η επεξεργασία ή η διαγραφή τους.



Εικόνα 19: Το popup επεξεργασίας ενός marker

Για παράδειγμα, στην περίπτωση της διαγραφής ενός Marker, η συνάρτηση που εκτελείται είναι η παρακάτω:

```
  const onDelete = async () => {
    try {
      props.setRender(null);
      const response = await
fetch(process.env.REACT_APP_BACKEND_URL+`/marker/${id}`, {
        method: "DELETE",
        headers: {
          "Content-Type": "application/json",
        },
      });
      const responseData = await response.json();
      if (!response.ok) {
        throw new Error(response.message);
      }
      console.log(responseData);
    } catch (err) {
      console.log(err);
    }
  };
```

Εικόνα 20: Συνάρτηση διαγραφής ενός Marker

Η συνάρτηση αυτή καλεί το backend με την εντολή fetch έτσι ώστε αυτό να κάνει πράξη τη διαγραφή του Marker (process.env.REACT_APP_BACKEND_URL+`/marker/${id})

Map Component

Ένα κρίσιμο Component είναι το Map , το οποίο ενσωματώνει διάφορες πτυχές των λειτουργιών χαρτογράφησης της εφαρμογής. Χρησιμοποιεί react-leaflet και react-leaflet-draw για δυνατότητες διαδραστικής χαρτογράφησης.

Το Map είναι το Component της εφαρμογής που συγκεντρώνει πολλές λειτουργίες που επικεντρώνονται γύρω από τη διαδραστική διεπαφή χάρτη. Είναι υπεύθυνο για την οπτικοποίηση και την αλληλεπίδραση με τα γεωγραφικά δεδομένα, τα στοιχεία ελέγχου χαρτών και την ενοποίηση με άλλα στοιχεία διεπαφής χρήστη. Το Component χρησιμοποιεί τη βιβλιοθήκη react-leaflet.

Το Component ξεκινά με ένα σύνολο import για τις απαραίτητες βιβλιοθήκες και τα στυλ.

Χρησιμοποιεί το useState για να διατηρήσει το state από τα παρακάτω:

type: Ένα flag για τον προσδιορισμό του τύπου της αλληλεπίδρασης που εκτελεί ο χρήστης, όπως η δημιουργία ενός νέου δείκτη ή πολυγώνου.

open, openList, openLogIn: Booleans για τον έλεγχο της ορατότητας διαφόρων drawers και παραθύρων.

feature: Για να διατηρήσουμε το τρέχον επιλεγμένο γεωγραφικό χαρακτηριστικό.

parishes: Για φιλτράρισμα ή επιλογή συγκεκριμένων γεωγραφικών περιοχών.

fly: Συντεταγμένες προς τις οποίες πρέπει να "πετάξει" ο χάρτης όταν επιλέγεται μια τοποθεσία.

Οι επιλογές στο Map, εμφανίζονται μόνο στους συνδεδεμένους χρήστες.
Αυτό επιτυγχάνεται με τη χρήση του authContext.

```
const auth = useContext(AuthContext);
```

Εικόνα 21: Δήλωση AuthContext

Με τον τρόπος αυτό διαβάσουμε το AuthContext το οποίο περιέχει τη μεταβλητή isLoggedIn ο οποία μας λέει ανά πάσα στιγμή αν ο χρήστης είναι συνδεδεμένος.
Το auth-context ορίζεται σε ξεχωριστό αρχείο ως εξής:

```
import { createContext } from 'react';

export const AuthContext = createContext({
  isLoggedIn: false,
  login: () => {},
  logout: () => {}
});
```

Εικόνα 22: Κώδικας createContext

Στο Map, όταν ο χρήστης πατήσει το κουμπί για το Login, η μεταβλητή openLogIn γίνεται true με αποτέλεσμα να γίνει render to LoginDrawer. Εκεί, όταν ο χρήστης συμπληρώσει το στοιχεία του και πατήσει το κουμπί, γίνεται η επικοινωνία με το backend. Το backend επιστρέφει την απάντηση. Αν είναι θετική, τότε ολοκληρώνεται το login όπως φαίνεται παρακάτω:

```
const onFinish = async (event) => {
  try {
    const response = await fetch(
      process.env.REACT_APP_BACKEND_URL + `/login`,
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          username: event.username,
          password: event.password,
        }),
      }
    );
    const responseData = await response.json();
    if (!response.ok) {
      throw new Error(response.message);
    }
    auth.login();
    props.setOpenLogIn(false);
    console.log("loginggh");
  } catch (err) {
    console.log(err);
  }
};
```

Εικόνα 23: Κώδικας αποστολής στοιχείων για login

Στο Map, τώρα μέσω του auth γνωρίζουμε εάν ο χρήστης έχει συνδεθεί. Εάν έχει, εμφανίζουμε τα κουμπιά προσθήκης και επεξεργασίας, όπως φαίνεται παρακάτω για τη δημιουργία Marker:

```
{auth.isLoggedIn &&<button
  title="Create a marker"
  style={{
    position: "absolute",
    left: "10px",
    top: " 100px",
    width: "32px",
    height: "32px",
    zIndex: 1000,
    display: "flex",
    alignItems: "center",
    justifyContent: "center",
  }}
  onClick={() => {
    setOpen(true);
    setType("newMarker");
  }}
 >
  <EnvironmentOutlined style={{ fontSize: "18px" }} />
</button>}
```

Εικόνα 24: Έλεγχος σύνδεσης για εμφάνιση button

Επεξεργασία και Ομαδοποίηση

Το EditControl από το react-leaflet-draw χρησιμοποιείται για την προσθήκη δυνατοτήτων σχεδίασης και επεξεργασίας στον χάρτη, με προγράμματα χειρισμού συμβάντων όπως το onEdited για να χειρίζονται τη διατήρηση των αλλαγών.

Το MarkerClusterGroup από το react-leaflet-cluster ομαδοποιεί σημεία που βρίσκονται γεωγραφικά κοντά μεταξύ τους για να βελτιώσουν την αναγνωσιμότητα.

4.2 Λειτουργία

Συμπερασματικά το entry point και ο ενορχηστρωτής της εφαρμογής είναι το App.js, λειτουργεί ως η main από τον οποίο περιστρέφεται ο ιστός της λειτουργικότητας.

Κεντρικό στοιχείο της λειτουργικότητας της εφαρμογής είναι το στοιχείο <Map>, όπου τα γεωγραφικά δεδομένα αποδίδονται και επεξεργάζονται. Το στοιχείο <Map>, επιτρέπει στους χρήστες να πλοηγούνται στο χαρτογραφικό έδαφος, εναλλάσσοντας μεταξύ Points και πολυγώνων που αντιπροσωπεύουν τα σημεία γεωγραφικών δεδομένων.

Ενσωματωμένα στο στοιχείο <Map> είναι τα στοιχεία <PolygonLayer> και <MarkerLayer>, το καθένα αφιερωμένο στον χειρισμό ενός ξεχωριστού τύπου οντότητας χάρτη. Αυτά τα επίπεδα παρέχουν στην εφαρμογή ένα σύνολο λειτουργιών και αλληλεπιδράσεων, όπως η προσθήκη, η επεξεργασία και η διαγραφή γεωγραφικών χαρακτηριστικών.

Το στοιχείο <ListDrawer> αυξάνει τη λειτουργικότητα του χάρτη παρέχοντας μια πλοηγήσιμη λίστα γεωγραφικών οντοτήτων, με τις οποίες οι χρήστες μπορούν να αλληλεπιδράσουν για να πραγματοποιήσουν αλλαγές στη θύρα προβολής του χάρτη. Αυτό το στοιχείο αποτελεί απόδειξη της δέσμευσης της εφαρμογής να παρέχει μια διεπαφή με επίκεντρο τον χρήστη που συνδυάζει την ορατότητα των δεδομένων με την προσβασιμότητα.

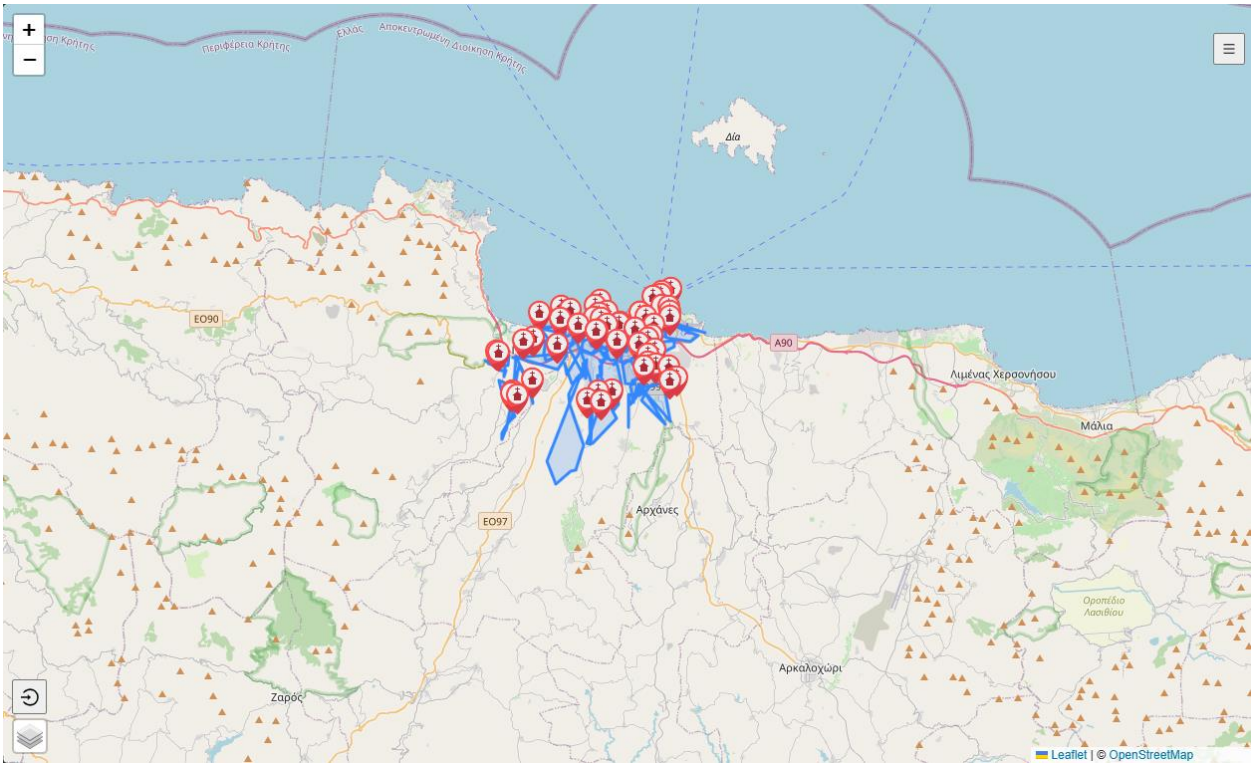
Η επεξεργασία και η διαγραφή των σημείων αντιμετωπίζονται στο <MarkerLayer>. Ενσωματώνοντας τη λογική για αυτές τις αλληλεπιδράσεις στα αναδυόμενα στοιχεία διεπαφής χρήστη, η εφαρμογή διατηρεί έναν καθαρό διαχωρισμό των ανησυχιών, διασφαλίζοντας ότι κάθε στοιχείο αντιμετωπίζει μια ενιαία ευθύνη. Αυτή η προσέγγιση όχι μόνο βελτιστοποιεί την εμπειρία του χρήστη, αλλά διατηρεί επίσης την αρχιτεκτονική ακεραιότητα της εφαρμογής.

Η ροή δεδομένων του frontend έχει σχεδιαστεί με ακρίβεια, τηρώντας ένα μοντέλο μονής κατεύθυνσης που διασφαλίζει ότι τα δεδομένα μεταβιβάζονται συστηματικά από τα γονικά στοιχεία στα παιδιά τους. Η διαχείριση κατάστασης εντοπίζεται εκεί που εξυπηρετεί καλύτερα, με hooks όπως το useState και το useEffect που επιτρέπουν responsive και side-effect-laden λειτουργίες.

Κεφάλαιο 5: Σενάρια Χρήσης Εφαρμογής

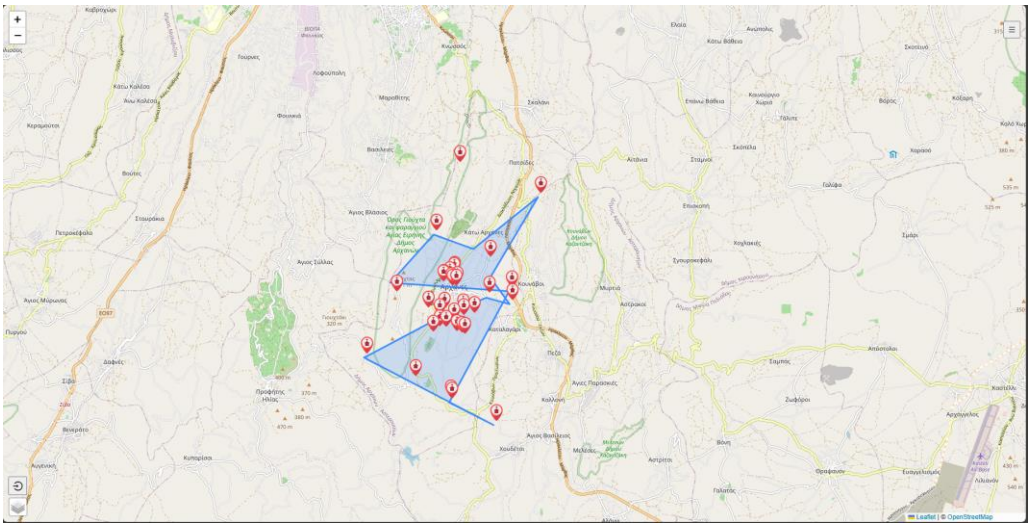
Στο κεφάλαιο αυτό θα δούμε συνοπτικά πως λειτουργεί η εφαρμογή που υλοποιήθηκε στα πλαίσια αυτής της πτυχιακής εργασίας.

Στην αρχική οθόνη της εφαρμογής βλέπουμε τα δεδομένα τα οποία έχουν γίνει fetch από τον server παρουσιασμένα πάνω σε έναν χάρτη.



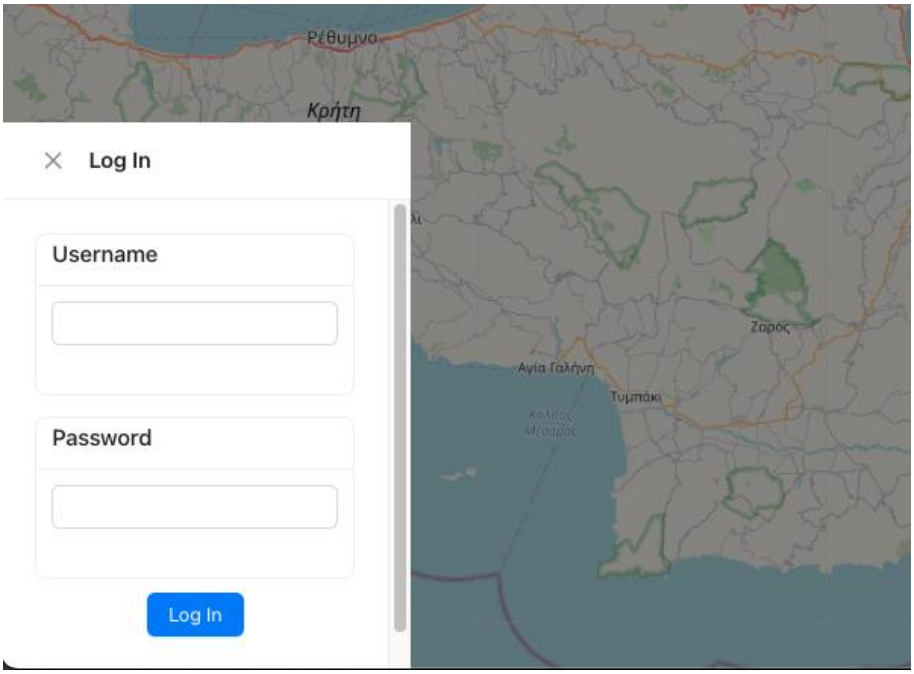
Εικόνα 25: Παρουσίαση των δεδομένων πάνω στο χάρτη

Κάνοντας zoom μπορεί να δει καλύτερα τα σημεία και τα πολύγωνα πάνω στο χάρτη.



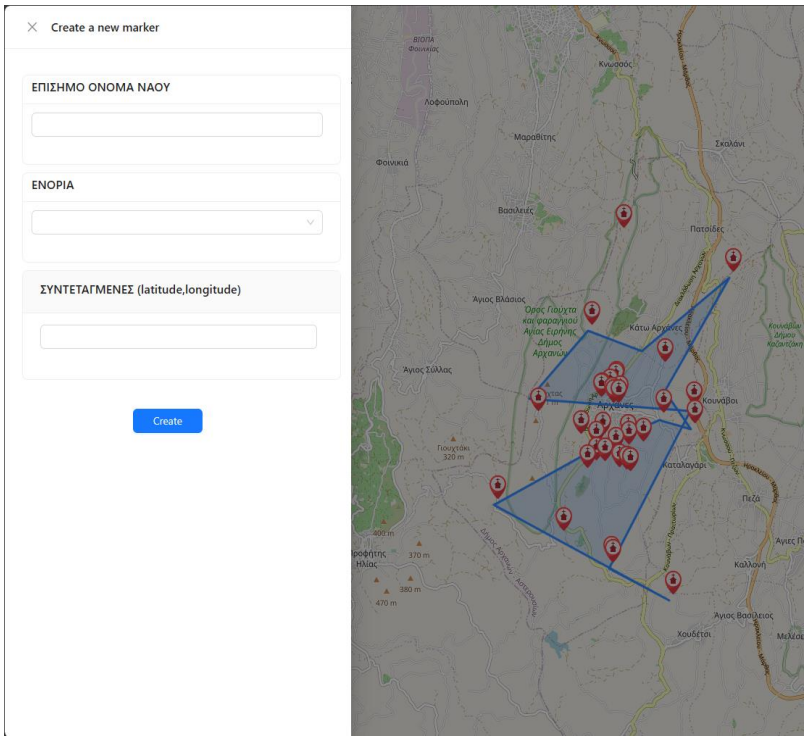
Εικόνα 26: Zoom σε περιοχή με πολύγωνο

Ο χρήστης πατώντας το κουμπί στο κάτω αριστερά μέρος της οθόνης μπορεί να κάνει login με τα στοιχεια του(username,password). Κάνοντας login μπορεί πλέον να προσθέσει, να αφαιρέσει και να επεξεργαστεί σημεία και πολύγωνα.



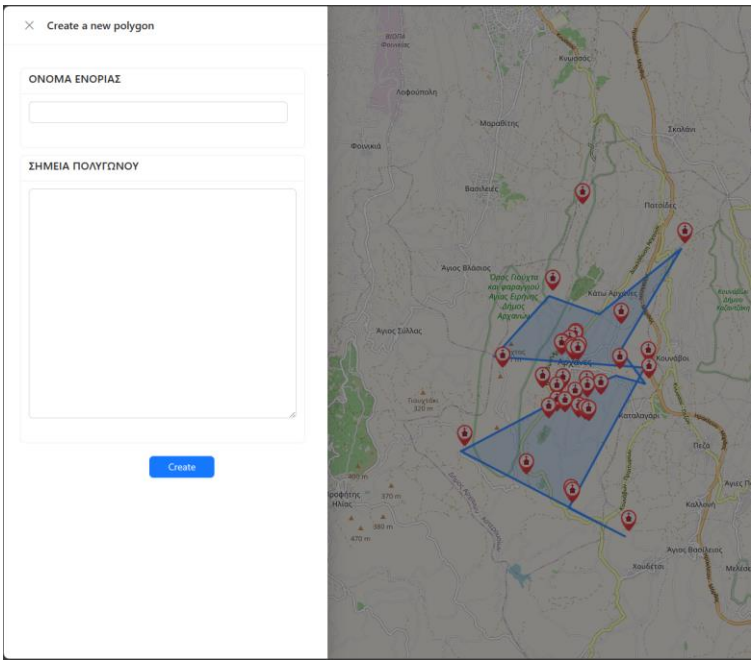
Εικόνα 27: Το μενού σύνδεσης

Πατώντας το εικονίδιο με το Marker στην αριστερή μπάρα μπορεί να προσθέσει νέο σημείο επιλέγοντας το πολύγωνο (ενορία) και δίνοντας όνομα και συντεταγμένες.



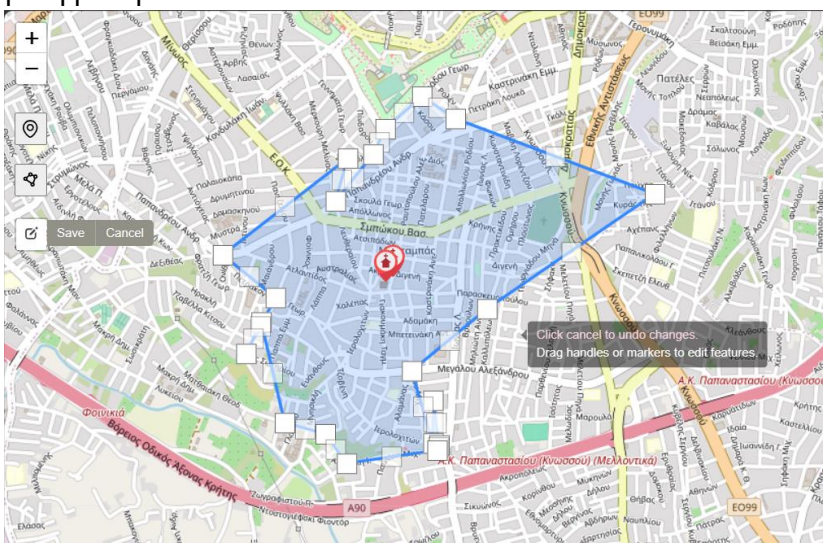
Εικόνα 28: Το μενού δημιουργίας νέου marker

Από την δεύτερη επιλογή, μπορεί να δημιουργήσει ένα νέο Πολύγωνο (ενορία) δίνοντας το όνομα και συντεταγμένες του.



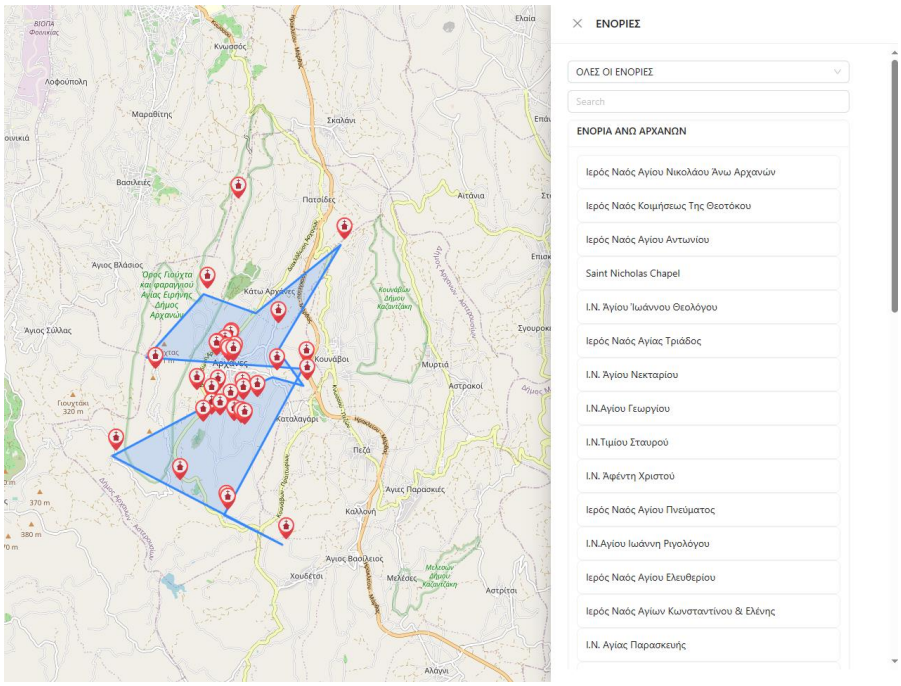
Εικόνα 29 :Το μενού δημιουργίας νέου πολυγώνου

Από το τελευταίο εικονίδιο ο χρήστης μπορεί να κάνει επεξεργασία απευθείας πάνω στο χάρτη και αποθήκευση στη βάση.



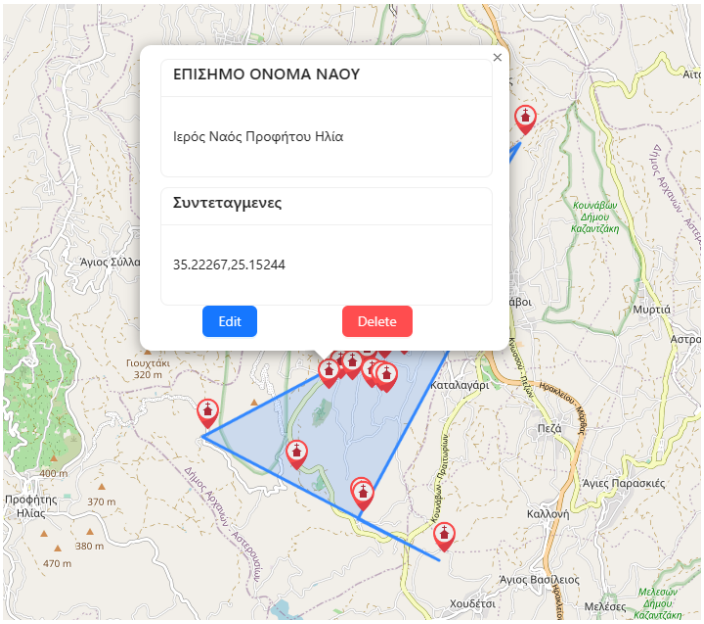
Εικόνα 30: Επεξεργασία πολυγώνου απευθείας πάνω στο χάρτη

Από το Drawer στα δεξιά δίνεται η δυνατότητα να γίνει αναζήτηση στα δεδομένα. Όταν επιλέξουμε κάποια ο χάρτης θα ζουμάρει αυτόματα προς αυτό.



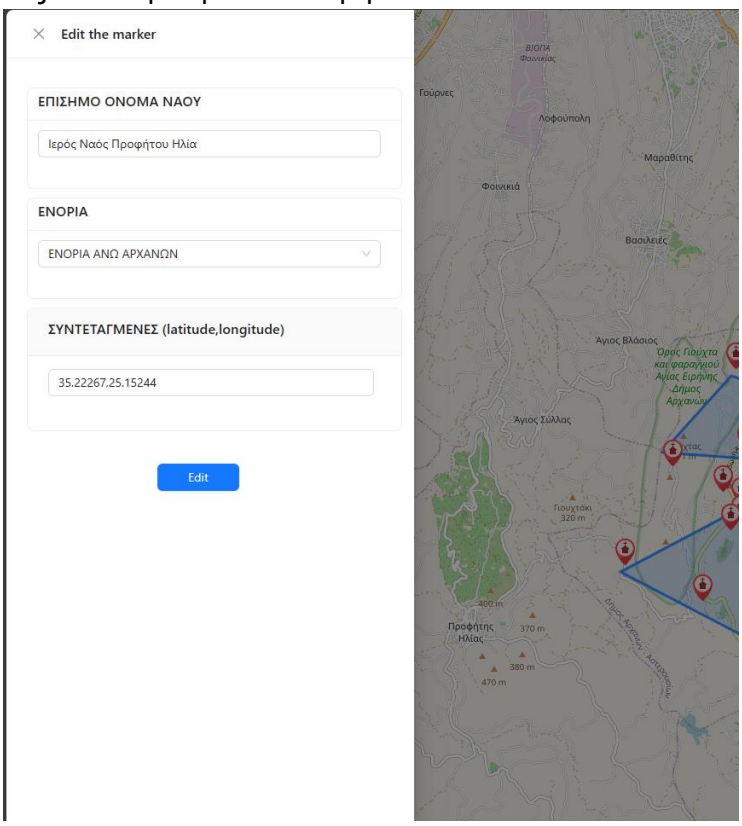
Εικόνα 31: Αναζήτηση στα δεδομένα

Πατώντας σε κάποιο σημείο ή πολύγωνο στο χάρτη μπορεί να δει τα στοιχεία του.



Εικόνα 32: Προβολή στοιχείων σημειου

Πατώντας “Edit” ανοίγει μια φόρμα στα αριστερά οπού μπορεί να γίνει αλλαγή στα στοιχεία του σημείου ενώ πατώντας Delete μπορεί να το σβήσει.



Εικόνα 33: Επεξεργασία στοιχείων σημείου

Κεφάλαιο 6: Συμπεράσματα και Μελλοντικές Εργασίες

Η εργασία που παρουσιάσαμε έχει ξεκινήσει την προσπάθεια δημιουργίας ενός ισχυρού και ανταποκρινόμενου συστήματος για τη διαχείριση των γεωχωρικών πληροφοριών. Αξιοποιώντας τη δύναμη του Node.js και του MongoDB, σχεδιάσαμε μια δομή υποστήριξης ικανή να χειρίζεται σύνθετα γεωγραφικά δεδομένα, υποστηρίζοντας σημεία και πολύγωνα, τα οποία μπορούν να αντιπροσωπεύουν μια ποικιλία χαρακτηριστικών σε έναν χάρτη, όπως ιστορικές τοποθεσίες, διοικητικά όρια ή σημεία ενδιαφέροντος.

Η προσέγγισή μας για τη δημιουργία ενός φιλικού και ασφαλούς περιβάλλοντος τόσο για τους διαχειριστές όσο και για τους χρήστες περιλάμβανε σχολαστικό σχεδιασμό και εκτέλεση. Ενσωματώσαμε ένα αξιόπιστο σύστημα ελέγχου ταυτότητας, επιτρέποντας ασφαλείς συνδέσεις χρηστών και λειτουργίες εντός του συστήματος. Μέσω της διαδικασίας ανάπτυξης, εξασφαλίσαμε ότι κάθε λειτουργικότητα, από την ανάκτηση δεδομένων έως τις ενημερώσεις και τις διαγραφές, υλοποιήθηκε με γνώμονα τη χρηστικότητα στον πραγματικό κόσμο.

Όσον αφορά τις μελλοντικές εργασίες, μπορούν να διερευνηθούν διάφορες διαδρομές για να αναβαθμιστούν οι δυνατότητες του συστήματος και η εμπειρία χρήστη. Η μελλοντική εργασία μπορεί να επικεντρωθεί στους ακόλουθους τομείς:

Επεκτασιμότητα: Καθώς η βάση των χρηστών αυξάνεται, το σύστημα θα πρέπει να διατηρεί την απόδοση. Η εξισορρόπηση φορτίου και η αρχιτεκτονική μικροϋπηρεσιών θα μπορούσαν να διερευνηθούν για την υποστήριξη περισσότερων ταυτόχρονων χρηστών και μεγαλύτερων συνόλων δεδομένων.

Advanced Analytics: Η εφαρμογή αλγορίθμων μηχανικής μάθησης για προγνωστική ανάλυση και ανάλυση χωρικών δεδομένων μπορεί να προσφέρει στους χρήστες πληροφορίες για τάσεις και μοτίβα στα γεωχωρικά δεδομένα.

Απόκριση σε κινητές συσκευές: Η ανάπτυξη μιας responsive εφαρμογής για κινητά ή η βελτιστοποίηση της εφαρμογής Web για χρήση σε κινητές συσκευές θα ήταν αρκετά σημαντική.

Επέκταση API: Η δημιουργία ενός ολοκληρωμένου API που θα επιτρέπει σε τρίτους προγραμματιστές να δημιουργούν εφαρμογές που μπορούν να αλληλεπιδρούν με το σύστημα θα επεκτείνει τη λειτουργικότητά του και την εμβέλειά του.

Βελτιώσεις ασφαλείας: Η συνεχής βελτίωση των μέτρων ασφαλείας, όπως η ενσωμάτωση πιο ισχυρής κρυπτογράφησης για τη μετάδοση και αποθήκευση δεδομένων, καθώς και οι τακτικοί έλεγχοι, μπορούν να προστατεύσουν από τις εξελισσόμενες απειλές στον κυβερνοχώρο.

Βιβλιογραφία

Goodchild, M.F., 1991. Geographic information systems. Progress in Human geography, 15(2), pp.194-200.

Wieczorek, W.F. and Delmerico, A.M., 2009. Geographic information systems. Wiley Interdisciplinary Reviews: Computational Statistics, 1(2), pp.167-186.

Gackenheimer, C., 2015. Introduction to React. Apress.

Makris, A., Tserpes, K., Spiliopoulos, G. and Anagnostopoulos, D., 2019, March. Performance Evaluation of MongoDB and PostgreSQL for Spatio-temporal Data. In EDBT/ICDT Workshops.

Fedosejev, A., 2015. React. js essentials. Packt Publishing Ltd.

https://react-leaflet.js.org/

https://ant.design/

https://www.npmjs.com/package/react-leaflet-draw

Tilkov, S. and Vinoski, S., 2010. Node. js: Using JavaScript to build high-performance network programs. IEEE Internet Computing, 14(6), pp.80-83.

Hahn, E., 2016. Express in Action: Writing, building, and testing Node. js app

https://mongoosejs.com/