ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Σχολή Τεχνολογίας

Τμήμα Ψηφιακών Συστημάτων

Web Εφαρμογή με REST API και React

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΧΡΗΣΤΟΣ ΠΟΛΥΖΟΣ (ΑΜ: 3119036)

Επιβλέπων: ΚΟΚΚΟΡΑΣ ΦΩΤΗΣ

ΛΑΡΙΣΑ, 2024

UNIVERSITY OF THESSALY

School of Technology

Digital Systems Department

Web Application with REST API and React

BACHELOR THESIS

CHRISTOS POLYZOS (RN: 3119036)

Supervisor: KOKKORAS FOTIS

LARISSA, 2024

Υπεύθυνη Δήλωση περί Ακαδημαϊκής Δεοντολογίας και Πνευματικών Δικαιωμάτων

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα πτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον, και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο/Η Δηλών/ούσα

(Υπογραφή)

Ονοματεπώνυμο Φοιτητή/-ήτριας

ΧΡΗΣΤΟΣ ΠΟΛΥΖΟΣ

Ημερομηνία

Περίληψη

Η πτυχιακή εργασία με τίτλο "Web Εφαρμογή με REST API και React" αφορά την ανάπτυξη της εφαρμογής FuelConnect, μιας διαδικτυακής πλατφόρμας που επιτρέπει στους χρήστες να εντοπίζουν και να αλληλοεπιδρούν με πρατήρια καυσίμων σε έναν δια δραστικό χάρτη. Η εφαρμογή αξιοποιεί τη βιβλιοθήκη Leaflet για την απεικόνιση των πρατηρίων με δείκτες τοποθεσίας, προσφέροντας στους χρήστες λειτουργίες όπως η προβολή πληροφοριών και η δυνατότητα παραγγελίας καυσίμων.

Οι κύριοι στόχοι ήταν η αποτελεσματική διαχείριση δεδομένων καυσίμων, η δημιουργία μιας εύχρηστης εμπειρίας χρήστη, και η διασφάλιση της ορθής λειτουργίας μέσω σύγχρονων εργαλείων ανάπτυξης και υποδομής. Για την υλοποίηση, χρησιμοποιήθηκαν React για το Frontend, Material UI για την οπτική διαμόρφωση, PHP με το SLIM framework και τον Apache Server για το Backend, και MySQL για τη βάση δεδομένων. Το σύστημα αλληλεπίδρασης βασίζεται σε REST API, ενώ η συνολική αρχιτεκτονική είναι πλήρως dockerized για εύκολη ανάπτυξη και συντήρηση μέσω Docker Compose.

Η εφαρμογή περιλαμβάνει μηχανισμούς σύνδεσης και εγγραφής χρηστών, με διαφοροποίηση δικαιωμάτων μεταξύ ιδιοκτητών πρατηρίων και νέων χρηστών. Οι υπάρχοντες ιδιοκτήτες έχουν τη δυνατότητα να ενημερώνουν τις τιμές των καυσίμων τους, ενώ η λειτουργία αναζήτησης βοηθά στην εύρεση πρατηρίων βάσει κριτηρίων.

Το FuelConnect παρέχει μια ολοκληρωμένη και σύγχρονη λύση, ενσωματώνοντας προηγμένες τεχνολογίες και βέλτιστες πρακτικές ανάπτυξης λογισμικού.

Abstract

The thesis titled "Web Application with REST API and React" presents the development of FuelConnect, an online platform that enables users to locate and interact with fuel stations on an interactive map. The application leverages the Leaflet library to display fuel stations using location markers, providing functionalities such as viewing station information and ordering fuel.

The primary objectives were efficient fuel data management, a user-friendly experience, and reliable performance achieved through modern development tools and infrastructure. The implementation included React for the frontend, Material UI for visual styling, PHP with the SLIM framework and the Apache server for the backend, and MySQL for the database, with interactions facilitated through a REST API. The entire architecture is fully dockerized for easy deployment and maintenance via Docker Compose.

The application features user authentication mechanisms, differentiating permissions between fuel station owners and standard users. Registered owners can update fuel prices at their stations, while the search function assists users in finding fuel stations based on specific criteria.

FuelConnect offers a comprehensive and modern solution, integrating advanced technologies and best practices in software development.

Περιεχόμενα

1 Εισαγωγή

Στη σύγχρονη εποχή, η τεχνολογία των διαδικτυακών εφαρμογών έχει διαμορφώσει νέους τρόπους παροχής υπηρεσιών προς το ευρύ κοινό, καλύπτοντας ανάγκες που σχετίζονται με την ταχύτητα, την ακρίβεια και την ευκολία χρήσης. Ένας από τους τομείς που έχουν επωφεληθεί σημαντικά από αυτές τις τεχνολογικές εξελίξεις είναι η αγορά των καυσίμων. Η βελτίωση της πρόσβασης σε δεδομένα σχετικά με πρατήρια καυσίμων και η δυνατότητα άμεσης παραγγελίας καυσίμων αποτελούν κρίσιμους παράγοντες για τη διευκόλυνση των καταναλωτών.

Η παρούσα πτυχιακή εργασία, με τίτλο "Web Εφαρμογή με REST API και React," περιγράφει την ανάπτυξη της εφαρμογής FuelConnect. Η εφαρμογή αυτή έχει σχεδιαστεί ως μια δια δραστική πλατφόρμα που επιτρέπει στους χρήστες να εντοπίζουν πρατήρια καυσίμων σε έναν χάρτη, να βλέπουν σημαντικές πληροφορίες και να πραγματοποιούν παραγγελίες καυσίμων από επιλεγμένα πρατήρια. Χρησιμοποιώντας σύγχρονες τεχνολογίες όπως το React για το Frontend και το PHP με το SLIM framework για το Backend, ενώ η εφαρμογή υποστηρίζεται από τον Apache Server, η εφαρμογή στοχεύει να προσφέρει μια ολοκληρωμένη και αποδοτική εμπειρία χρήστη.

Η καινοτομία της FuelConnect έγκειται στη χρήση δια δραστικών χαρτών μέσω της βιβλιοθήκης Leaflet, που επιτρέπει στους χρήστες να πλοηγούνται εύκολα στον χάρτη και να βρίσκουν πρατήρια με βάση τα κριτήρια αναζήτησής τους. Παράλληλα, η εφαρμογή ενσωματώνει μηχανισμούς σύνδεσης και εγγραφής χρηστών, προσφέροντας στους ιδιοκτήτες πρατηρίων τη δυνατότητα να διαχειρίζονται τις τιμές των καυσίμων τους. Η υλοποίηση βασίζεται σε ένα REST API που επιτρέπει την αποτελεσματική επικοινωνία μεταξύ του Frontend και του Backend, εξασφαλίζοντας την ομαλή λειτουργία της εφαρμογής.

Η ανάπτυξη της εφαρμογής έγινε με γνώμονα τη βέλτιστη απόδοση και την ευκολία συντήρησης, χρησιμοποιώντας τεχνικές dockerization για την απομόνωση και την

ταυτόχρονη διαχείριση των διαφορετικών στοιχείων του συστήματος. Το τελικό αποτέλεσμα είναι μια αξιόπιστη πλατφόρμα που ανταποκρίνεται στις ανάγκες των χρηστών για γρήγορη και ασφαλή πρόσβαση σε υπηρεσίες καυσίμων.

2 Τεχνολογικό Υπόβαθρο

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν σύγχρονα εργαλεία και τεχνολογίες, όπως το React για το Frontend, το SLIM framework σε PHP για το Backend, και το Material UI για την οπτική διαμόρφωση της διεπαφής. Η βιβλιοθήκη Leaflet αξιοποιήθηκε για τη δια δραστική απεικόνιση των πρατηρίων στον χάρτη, προσφέροντας μια δυναμική εμπειρία πλοήγησης. Στην υποδομή περιλαμβάνονται επίσης η MySQL για τη βάση δεδομένων, ο Apache Server για τη διαχείριση των αιτημάτων του Backend, και το Docker για την αποδοτική διαχείριση και συντήρηση των υπηρεσιών. Αυτά τα εργαλεία συμβάλλουν στη σταθερότητα, την ευελιξία και την ευκολία συντήρησης του συστήματος.

2.1 Frontend

Το Frontend της εφαρμογής FuelConnect υλοποιήθηκε χρησιμοποιώντας την React, μια δημοφιλή βιβλιοθήκη JavaScript για τη δημιουργία εύχρηστων και ευέλικτων διεπαφών χρήστη. Η React επιτρέπει την οργάνωση του κώδικα σε επαναχρησιμοποιήσιμα components, εξασφαλίζοντας την ευελιξία και την ευκολία στη διαχείριση της διεπαφής, ακόμα και όταν αυτή περιέχει πολύπλοκες λειτουργίες.

Για τη διαχείριση της εμφάνισης και της διάταξης του περιεχομένου, χρησιμοποιήθηκε η βιβλιοθήκη Material UI (MUI). Το MUI προσφέρει ένα σύνολο έτοιμων οπτικών στοιχείων που ακολουθούν τις αρχές του Material Design, παρέχοντας μια μοντέρνα και ομοιόμορφη εμφάνιση στην εφαρμογή. Τα στοιχεία του MUI συμβάλλουν στη διαμόρφωση μιας ευχάριστης εμπειρίας χρήστη, ενώ παράλληλα διευκολύνουν τη διαδικασία ανάπτυξης.

Στον τομέα της γεωγραφικής απεικόνισης και της πλοήγησης στον χάρτη, η εφαρμογή βασίζεται στη βιβλιοθήκη Leaflet. Η Leaflet προσφέρει εύκολες μεθόδους διαχείρισης και απόδοσης χαρτών, υποστηρίζοντας τη δημιουργία δια δραστικών χαρτών και την απεικόνιση των πρατηρίων καυσίμων μέσω δεικτών τοποθεσίας (markers). Η βιβλιοθήκη επιλέχθηκε λόγω της αποτελεσματικότητας και της υποστήριξής της σε περιβάλλοντα με

περιορισμένους πόρους, προσφέροντας έτσι άμεση απόκριση και ομαλή εμπειρία περιήγησης.

Η αρχιτεκτονική του Frontend βασίζεται σε μια μονοσέλιδη εφαρμογή (Single-Page Application - SPA), η οποία ενισχύει την ταχύτητα και την απόκριση του συστήματος. Το SPA επιτρέπει στους χρήστες να αλληλοεπιδρούν με την εφαρμογή χωρίς ανανέωση της σελίδας, προσφέροντας μια πιο φυσική και γρήγορη εμπειρία χρήσης.

2.2 Backend

Η υλοποίηση του Backend της εφαρμογής FuelConnect βασίζεται σε PHP και το SLIM framework. Το SLIM είναι ένα ελαφρύ, αλλά ισχυρό framework για την ανάπτυξη RESTful APIs, το οποίο παρέχει όλες τις βασικές δυνατότητες που απαιτούνται για την κατασκευή μίας δυναμικής και αποδοτικής εφαρμογής, όπως routing, middleware και υποστήριξη για τα HTTP request/response.

Η βάση δεδομένων της εφαρμογής διαχειρίζεται με την MySQL, μια από τις πιο δημοφιλείς και αξιόπιστες σχέσεις βάσεων δεδομένων. Η MySQL παρέχει τη δυνατότητα αποθήκευσης και ανακτήσεων δεδομένων με υψηλή ταχύτητα και ασφάλεια. Οι πίνακες της βάσης σχεδιάστηκαν ώστε να υποστηρίζουν την ευέλικτη διαχείριση των δεδομένων των πρατηρίων καυσίμων, των χρηστών, καθώς και των ιστορικών τιμών καυσίμων.

Για την αποδοτική διαχείριση των υπηρεσιών και την εύκολη ανάπτυξη, χρησιμοποιείται το Docker, το οποίο επιτρέπει τη δημιουργία και διαχείριση των υπηρεσιών μέσα σε Containers. Αυτή η προσέγγιση επιτρέπει την απομόνωση του περιβάλλοντος ανάπτυξης και την εύκολη διαχείριση της υποδομής σε διάφορες φάσεις ανάπτυξης, δοκιμών και παραγωγής.

Ο Apache Server χρησιμοποιείται ως Web Server για την εξυπηρέτηση του Backend της εφαρμογής. Ο Apache προσφέρει αξιόπιστο και αποτελεσματικό χειρισμό των

αιτημάτων HTTP, καθώς και την υποστήριξη δυναμικών εφαρμογών μέσω του PHP, διασφαλίζοντας την ομαλή λειτουργία του Backend.

Η χρήση αυτών των τεχνολογιών, συνδυασμένων με το REST API, εξασφαλίζει τη σωστή και αποδοτική επικοινωνία μεταξύ του Frontend και του Backend, προσφέροντας στους χρήστες μια άμεση και αξιόπιστη εμπειρία.

2.3 Βάση Δεδομένων

Για την αποθήκευση και διαχείριση των δεδομένων, η εφαρμογή FuelConnect χρησιμοποιεί την MySQL, μια αξιόπιστη και ευρέως χρησιμοποιούμενη σχεσιακή βάση δεδομένων. Η MySQL επιλέχθηκε για την υποστήριξη της εφαρμογής λόγω της αποτελεσματικότητάς της στην αποθήκευση δεδομένων σε δομημένη μορφή και της δυνατότητας εκτέλεσης σύνθετων queries για την ανάκτηση πληροφοριών.

Η βάση δεδομένων περιλαμβάνει πίνακες για τους χρήστες, τα πρατήρια καυσίμων, τις τιμές καυσίμων και τις παραγγελίες. Για τη διασφάλιση της συνέπειας των δεδομένων, χρησιμοποιούνται σχέσεις μεταξύ των πινάκων, ενώ οι στρατηγικές κανονικοποίησης και χρήση δεικτών (indexes) διασφαλίζουν την απόδοση του συστήματος.

Η επιλογή της MySQL επιτρέπει στην εφαρμογή να χειρίζεται μεγάλες ποσότητες δεδομένων με υψηλή ταχύτητα και ασφάλεια, ενώ υποστηρίζει τη δυνατότητα εξωτερικής ανάκτησης και τροποποίησης δεδομένων μέσω του REST API της εφαρμογής.

2.4 Docker

Το Docker αποτελεί ένα από τα πιο δημοφιλή εργαλεία για τη διαχείριση και την απομόνωση εφαρμογών σε μορφή containers. Τα containers είναι ανεξάρτητα περιβάλλοντα, τα οποία εξασφαλίζουν ότι κάθε εφαρμογή λειτουργεί σε ένα απομονωμένο σύστημα, ανεξαρτήτως της υποδομής του εκάστοτε server.

2.4.1 Πλεονεκτήματα του Docker

Η χρήση του Docker στην ανάπτυξη εφαρμογών έχει φέρει σημαντικές αλλαγές στην παραγωγικότητα και την αξιοπιστία, καθώς προσφέρει:

- **Απομόνωση Περιβαλλόντων**: Με το Docker, η κάθε υπηρεσία μπορεί να λειτουργεί σε ένα απομονωμένο κοντέινερ, εξασφαλίζοντας ότι τυχόν αλλαγές ή προβλήματα σε μία υπηρεσία δεν επηρεάζουν τις υπόλοιπες.
- **Ευκολία στη Διαχείριση Εξαρτήσεων**: Μέσω της διαδικασίας containerization, οι εξαρτήσεις κάθε υπηρεσίας βρίσκονται μέσα στα Docker images, αποτρέποντας προβλήματα συμβατότητας μεταξύ διαφορετικών βιβλιοθηκών ή εκδόσεων λογισμικού.
- **Δια λειτουργικότητα**: Το Docker εξασφαλίζει ότι οι εφαρμογές μπορούν να τρέξουν απρόσκοπτα σε οποιοδήποτε περιβάλλον, είτε πρόκειται για τοπική ανάπτυξη (Local) είτε για ανάπτυξη σε περιβάλλον παραγωγής (Production).
- **Αποδοτικότητα και Κλιμάκωση**: Τα containers καταναλώνουν λιγότερους πόρους συγκριτικά με τις εικονικές μηχανές, επιτρέποντας την ανάπτυξη πολλαπλών containers στον ίδιο server και βελτιώνοντας την απόδοση της εφαρμογής. Επιπλέον, η διαδικασία scaling γίνεται πιο αποδοτική.

2.4.2 Docker Compose

Το Docker Compose είναι ένα εργαλείο που συνοδεύει το Docker και επιτρέπει τη διαχείριση πολλαπλών Containers ως ένα ενιαίο σύστημα. Μέσω του *dockercompose.yml*, οι προγραμματιστές μπορούν να ορίζουν όλες τις υπηρεσίες που χρειάζεται η εφαρμογή, διασφαλίζοντας την εύκολη εκκίνηση και διαχείριση των Containers με εντολές όπως docker-compose up και docker-compose down.

Το Docker και το Docker Compose αποτελούν έτσι βασικά εργαλεία στην ανάπτυξη σύγχρονων εφαρμογών, καθώς διευκολύνουν την ανάπτυξη και την απομόνωση περιβαλλόντων, παρέχοντας παράλληλα τη δυνατότητα εύκολης κλιμάκωσης και διαχείρισης του συστήματος σε πολλαπλά περιβάλλοντα.

3 Αρχιτεκτονική και Ανάλυση Συστήματος

Το κεφάλαιο αυτό περιγράφει την αρχιτεκτονική και τα κύρια συστατικά της εφαρμογής **FuelConnect**, με στόχο την ομαλή συνεργασία του Frontend, του Backend και της βάσης δεδομένων. Η αρχιτεκτονική της εφαρμογής βασίζεται σε σύγχρονες τεχνολογίες και πρακτικές ανάπτυξης, διασφαλίζοντας την απόδοση, τη συντήρηση και την ευκολία ανάπτυξης.

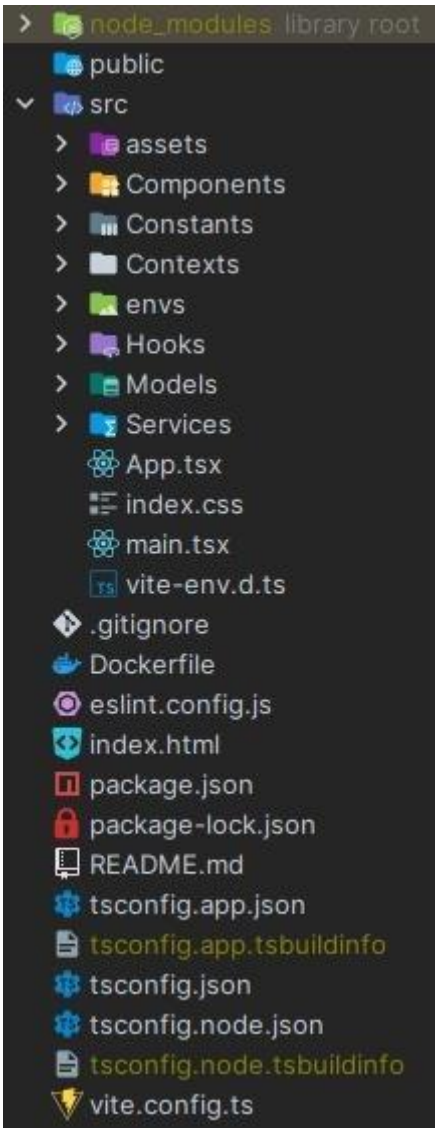
3.1 Αρχιτεκτονική και Δομή του Frontend

Η σχεδίαση του Frontend της εφαρμογής **FuelConnect** βασίζεται στην τεχνολογία React, η οποία επιτρέπει την ανάπτυξη δυναμικών και αλληλεπιδραστικών διεπαφών χρήστη. Η χρήση της TypeScript διασφαλίζει την ασφαλή ανάπτυξη και τη διαχείριση τύπων δεδομένων, μειώνοντας τα λάθη κατά την εκτέλεση του κώδικα. Ο σχεδιασμός της διεπαφής επικεντρώνεται στην εμπειρία του χρήστη, με τη χρήση της βιβλιοθήκης Material UI (MUI) για την ανάπτυξη ενός ελκυστικού και φιλικού περιβάλλοντος. Επιπλέον, η εφαρμογή χρησιμοποιεί τη βιβλιοθήκη Leaflet για την προβολή των σταθμών καυσίμων στον χάρτη, προσφέροντας μια εύχρηστη και δυναμική χαρτογραφική εμπειρία. Το Frontend και το Backend επικοινωνούν μέσω RESTful API, επιτρέποντας την ανταλλαγή δεδομένων μεταξύ των δύο πλευρών.

3.1.1 Δομή Φακέλων

Η δομή του Frontend του FuelConnect ακολουθεί τις βέλτιστες πρακτικές για την ανάπτυξη εφαρμογών React, με τη χρήση σαφούς και λογικής οργάνωσης του κώδικα.

Οι βασικοί φάκελοι περιλαμβάνουν:



- **node_modules**: Ο Φάκελος περιέχει όλες τις βιβλιοθήκες και εξαρτήσεις του έργου που έχουν εγκατασταθεί μέσω του Node.js. Οι εξαρτήσεις αυτές είναι απαραίτητες για τη λειτουργία της εφαρμογής.
- **src**: Ο κύριος φάκελος κώδικα της εφαρμογής. Περιέχει όλα τα αρχεία και τους φακέλους με τον κώδικα της εφαρμογής, όπως Components, hooks, context και Services. Αυτός είναι ο πυρήνας του Frontend και περιέχει τις βασικές λειτουργικότητες και το UI της εφαρμογής.
- **assets**: Φάκελος που αποθηκεύει στατικά αρχεία της εφαρμογής, όπως εικόνες, εικονίδια και ενδεχομένως αρχεία πολυμέσων. Αυτά τα στοιχεία χρησιμοποιούνται από τα components για την ενίσχυση της οπτικής παρουσίασης της διεπαφής χρήστη.
- **Components**: Ο φάκελος αυτός περιέχει όλα τα επαναχρησιμοποιήσιμα κομμάτια της διεπαφής χρήστη (UI) της εφαρμογής FuelConnect. Κάθε υποφάκελος

αντιπροσωπεύει ένα λειτουργικό μέρος ή μια συγκεκριμένη διεπαφή της εφαρμογής, επιτρέποντας έτσι την οργάνωση και την επεκτασιμότητα των στοιχείων.

- **Constants**: Φάκελος που περιλαμβάνει σταθερές τιμές που χρησιμοποιούνται σε όλη την εφαρμογή.
- **envs**: Φάκελος που αποθηκεύει αρχεία περιβάλλοντος, όπως **.env** αρχεία με περιβαλλοντικές μεταβλητές που ρυθμίζουν τη λειτουργία της εφαρμογής για τοπικό ή παραγωγικό περιβάλλον. Αυτές οι μεταβλητές μπορεί να περιλαμβάνουν ευαίσθητες πληροφορίες, όπως κλειδιά API και URLs του Backend.
- **Hooks**: Περιέχει συναρτήσεις που χρησιμοποιούνται για τη διαχείριση της κατάστασης (state) και της λογικής των components. Αυτές οι συναρτήσεις επιτρέπουν την απομόνωση της επιχειρησιακής λογικής από τα components, επιτρέποντας έτσι την επαναχρησιμοποίησή της σε διαφορετικά σημεία της εφαρμογής και τη διευκόλυνση της συντήρησης του κώδικα.
- **Models**: Φάκελος που περιέχει τα αρχεία μοντέλων δεδομένων της εφαρμογής. Αυτά τα μοντέλα ορίζουν τη δομή των δεδομένων που χρησιμοποιεί η εφαρμογή, επιτρέποντας την ασφαλή και συνεπή διαχείριση δεδομένων με TypeScript interfaces.
- **Services**: Περιέχει αρχεία που είναι υπεύθυνα αποκλειστικά για την επικοινωνία της εφαρμογής με το Backend μέσω APIs. Τα Services περιλαμβάνουν συναρτήσεις για την εκτέλεση HTTP αιτήσεων (όπως GET, POST, κ.λπ.), επιτρέποντας έτσι την αποστολή και λήψη δεδομένων από το Backend.
- **App.tsx**: Το κύριο αρχείο της εφαρμογής που περιλαμβάνει τον κεντρικό κώδικα της React. Εδώ γίνεται η διαμόρφωση του γενικού UI και η σύνδεση με τα context providers, ώστε να διαμοιράζεται η κατάσταση της εφαρμογής σε όλα τα components.
- **index.css**: Το βασικό αρχείο CSS που περιέχει γενικά στυλ για την εφαρμογή.
- **main.tsx**: Το αρχείο εισόδου της εφαρμογής, το οποίο αρχικοποιεί το React και το root component (App.tsx) στο DOM. Αυτό το αρχείο αποτελεί το σημείο όπου συνδέεται η εφαρμογή με το HTML της σελίδας και ρυθμίζονται οι αρχικοί context providers.
- **Dockerfile**: Περιλαμβάνει τις οδηγίες για το Docker ώστε να δημιουργηθεί ένα Container που θα φιλοξενεί το Frontend της εφαρμογής.

- **eslint.config.js**: Αρχείο ρυθμίσεων για το ESLint, το οποίο διασφαλίζει τη συμμόρφωση του κώδικα με συγκεκριμένα πρότυπα και κανόνες σύνταξης, μειώνοντας τα πιθανά σφάλματα και βελτιώνοντας τη συντηρησιμότητα.
- **index.html**: Το κύριο αρχείο HTML της εφαρμογής, το οποίο λειτουργεί ως το σημείο εισόδου της εφαρμογής στον Browser. Ο κώδικας της React ενσωματώνεται και εμφανίζεται μέσω αυτού του αρχείου.
- **package.json**: Περιλαμβάνει τις εξαρτήσεις, τα scripts και άλλες βασικές πληροφορίες για το έργο. Είναι το κύριο αρχείο ρυθμίσεων του έργου, από το οποίο εξαρτώνται όλες οι εξωτερικές βιβλιοθήκες και τα εργαλεία ανάπτυξης.
- **package-lock.json**: Αρχείο που διασφαλίζει τη σταθερότητα των εκδόσεων των εξαρτήσεων, διασφαλίζοντας ότι η εφαρμογή θα χρησιμοποιεί τις ίδιες εκδόσεις ανεξαρτήτως περιβάλλοντος.
- **tsconfig.app.json**, **tsconfig.node.json**, και **tsconfig.json**: Αρχεία ρυθμίσεων για την TypeScript. Τα αρχεία αυτά καθορίζουν τη διαμόρφωση του TypeScript compiler, όπως οι διαδρομές και οι τύποι που χρησιμοποιούνται, επιτρέποντας την ασφαλή ανάπτυξη και τη βελτίωση της απόδοσης του κώδικα.
- **vite.config.ts**: Αρχείο ρυθμίσεων για το Vite, το οποίο χρησιμοποιείται ως εργαλείο ανάπτυξης για γρήγορη και αποδοτική ανάπτυξη της εφαρμογής. Το αρχείο αυτό καθορίζει τις παραμέτρους που χρησιμοποιούνται κατά το χτίσιμο και την εκτέλεση της εφαρμογής σε περιβάλλον ανάπτυξης.

3.1.2 Διαχείριση Εντολών API (API Calls)

Σε αυτή την ενότητα, περιγράφεται η διαδικασία μέσω της οποίας το Frontend της εφαρμογής FuelConnect επικοινωνεί με το Backend, στέλνοντας αιτήματα API για την άντληση και αποστολή δεδομένων. Αυτή η διαδικασία διευκολύνεται μέσω της χρήσης των **Services**, που περιλαμβάνουν συγκεκριμένες συναρτήσεις για την εκτέλεση των αιτημάτων και την επεξεργασία των αποκρίσεων από τον διακομιστή.

- **Αποστολή Αιτημάτων**: Όταν απαιτούνται δεδομένα ή κάποια λειτουργία από το Backend (π.χ., σύνδεση χρήστη, ενημέρωση τιμής καυσίμου, αναζήτηση σταθμού), το Frontend στέλνει ένα αίτημα μέσω των Services. Κάθε Service έχει συγκεκριμένες συναρτήσεις για κάθε τύπο αιτήματος (GET, POST, PUT), διευκολύνοντας έτσι την οργάνωση και επαναχρησιμοποίηση του κώδικα.

- **Διαχείριση Αποκρίσεων**: Τα δεδομένα που επιστρέφονται από το Backend λαμβάνονται στα Services, και ανάλογα με την επιτυχία ή αποτυχία του αιτήματος, ο χρήστης ενημερώνεται μέσω Alert. Αυτή η απλή προσέγγιση βελτιώνει την εμπειρία του χρήστη, χωρίς περίπλοκη λογική ελέγχου στο Service.
- **Ενημέρωση των Components**: Τα components λαμβάνουν τα δεδομένα από τα Services και ενημερώνουν την κατάσταση (state) τους ή εμφανίζουν τα δεδομένα στη διεπαφή χρήστη. Έτσι, τα components παραμένουν απλά και επικεντρώνονται μόνο στην παρουσίαση των δεδομένων, ενώ η λογική της αλληλεπίδρασης με το Backend είναι διαχωρισμένη στα Services.

3.1.3 Διαχείριση Καταστάσεων και Ροές Δεδομένων

Στην εφαρμογή FuelConnect, η διαχείριση των καταστάσεων (States) πραγματοποιείται με τη βοήθεια του Context API και custom hooks. Η εφαρμογή ξεκινά με την ανάκτηση των δεδομένων για όλα τα πρατήρια μέσω ενός Context, το οποίο αποθηκεύει και διαχειρίζεται αυτά τα δεδομένα κεντρικά. Ύστερα, τα δεδομένα αυτά περνούν στα διάφορα components μέσω των props, επιτρέποντας τη σωστή ανανέωση και εμφάνιση των πληροφοριών χωρίς να απαιτείται κάθε φορά νέο fetch.

Επιπλέον, η διαχείριση της κατάστασης των modals γίνεται με τη βοήθεια ενός custom hook, το οποίο παρακολουθεί την κατάσταση των διαφόρων modals στο Navigation Bar και διευκολύνει την εναλλαγή μεταξύ τους. Το hook αυτός επιτρέπει την αποδοτική διαχείριση των πολλαπλών καταστάσεων του UI, εξασφαλίζοντας την ομαλή λειτουργία των modals και την εύκολη αλληλεπίδραση με τον χρήστη.

Αυτή η προσέγγιση βοηθά στην αποδοτική διαχείριση της κατάστασης σε μεγάλη κλίμακα και την αποφυγή της αναγκαιότητας για συνεχιζόμενη μεταφορά δεδομένων μεταξύ των components μέσω props.

3.2 Αρχιτεκτονική και Δομή του Backend

Η αρχιτεκτονική του Backend της εφαρμογής FuelConnect βασίζεται στο Slim Framework, ένα lightweight PHP framework που επιτρέπει την εύκολη δημιουργία RESTful API. Ο σχεδιασμός της εφαρμογής εστιάζει στην καθαρή και επεκτάσιμη

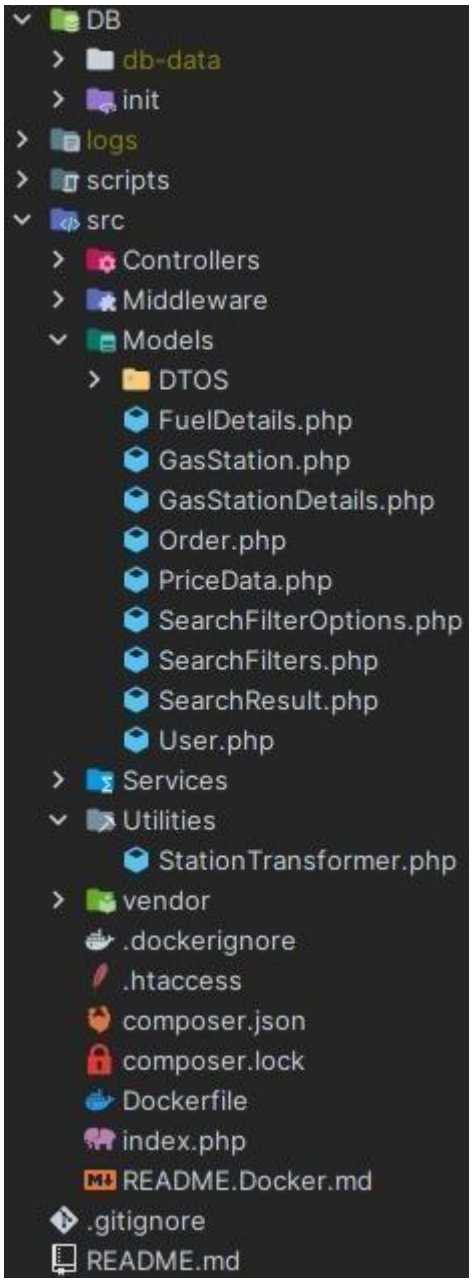
διαχείριση των αιτημάτων και στην αλληλεπίδραση με τη βάση δεδομένων μέσω SQL queries.

Η επικοινωνία μεταξύ του Frontend και του Backend γίνεται μέσω HTTP αιτημάτων, τα οποία διαχειρίζονται από τα αντίστοιχα endpoints της εφαρμογής. Κάθε endpoint έχει σχεδιαστεί έτσι ώστε να εκτελεί συγκεκριμένες λειτουργίες, όπως η ανάκτηση και η ενημέρωση των δεδομένων των χρηστών, των πρατηρίων, η επικύρωση των στοιχείων, και η επεξεργασία αιτημάτων παραγγελιών.

Το Backend χρησιμοποιεί μια RESTful αρχιτεκτονική, όπου κάθε τύπος αίτηματος (GET, POST, PUT, DELETE) συνδέεται με τις αντίστοιχες λειτουργίες της εφαρμογής. Η διαχείριση των αιτημάτων και των δεδομένων γίνεται με σαφήνεια και επαγγελματισμό, προκειμένου να εξασφαλιστεί η βέλτιστη απόδοση και ασφάλεια.

3.2.1 Δομή του Κώδικα του Backend

Η δομή του Backend της εφαρμογής FuelConnect οργανώνεται γύρω από τα βασικά στοιχεία του Slim Framework και τις ανάγκες της εφαρμογής για επεκτασιμότητα και ευκολία συντήρησης. Ο κώδικας του Backend χωρίζεται σε διάφορους φακέλους, καθένας από τους οποίους έχει έναν ξεχωριστό ρόλο στην υποστήριξη των λειτουργιών της εφαρμογής. Η δομή αυτή είναι η εξής:



- **DB**: Περιέχει το volume από το Docker με τα δεδομένα της βάσης δεδομένων. Επίσης, περιλαμβάνει το αρχικό script για τη δημιουργία της βάσης, των tables και την εισαγωγή αρχικών δεδομένων.
- **logs**: Φάκελος όπου αποθηκεύονται τα logs της εφαρμογής για την παρακολούθηση.
- **src**: Ο κύριος φάκελος του κώδικα του Backend, ο οποίος περιλαμβάνει όλα τα αρχεία που σχετίζονται με τη λογική της εφαρμογής (controllers, Models, Services κ.λπ.).
- **Controllers**: Ο φάκελος που περιέχει τους controllers της εφαρμογής, οι οποίοι χειρίζονται τις HTTP αιτήσεις (requests) και επεξεργάζονται τα δεδομένα που επιστρέφονται από το Backend.

- **Middleware**: Περιέχει τα middleware της εφαρμογής, τα οποία εκτελούνται πριν ή μετά την επεξεργασία των αιτημάτων και μπορούν να τροποποιήσουν την αίτηση (request) ή την απάντηση (response).
- **Models**: Περιέχει τα μοντέλα της εφαρμογής, που αντιπροσωπεύουν τα δεδομένα και τις επιχειρησιακές λογικές της εφαρμογής. Στον φάκελο αυτόν περιλαμβάνεται επίσης ο υποφάκελος **DTOS**, ο οποίος περιέχει Data Transfer Objects που χρησιμοποιούνται για την αποστολή δεδομένων μεταξύ των επιπέδων της εφαρμογής.
- **Services**: Ο φάκελος που περιέχει τις υπηρεσίες της εφαρμογής, οι οποίες διαχειρίζονται την επικοινωνία με τη βάση δεδομένων και την επεξεργασία των δεδομένων. Οι υπηρεσίες χρησιμοποιούνται για την εκτέλεση αιτημάτων προς τη βάση, την ανάκτηση και αποθήκευση δεδομένων, καθώς και για την εφαρμογή επιχειρησιακής λογικής σχετικής με τα δεδομένα της εφαρμογής.
- **Utilities**: Ο φάκελος που περιέχει βοηθητικές κλάσεις και συναρτήσεις που χρησιμοποιούνται σε όλη την εφαρμογή.
- **vendor**: Φάκελος που περιέχει τις εξαρτήσεις της εφαρμογής που διαχειρίζεται το Composer.
- **.dockerignore**: Αρχείο που καθορίζει ποια αρχεία ή φάκελοι θα πρέπει να αγνοούνται κατά τη διάρκεια της διαδικασίας δημιουργίας του Docker Image.
- **.htaccess**: Αρχείο ρυθμίσεων για την παραμετροποίηση του Apache Web Server.
- **composer.json**: Αρχείο που καθορίζει τις εξαρτήσεις (dependencies) της εφαρμογής, καθώς και τις σχετικές ρυθμίσεις για το Composer.
- **composer.lock**: Αρχείο που καταγράφει τις ακριβείς εκδόσεις των εξαρτήσεων που χρησιμοποιούνται στην εφαρμογή.
- **Dockerfile**: Ο φάκελος περιέχει το Dockerfile του Backend, το οποίο χρησιμοποιείται για την κατασκευή της εικόνας Docker της εφαρμογής. Το Dockerfile περιλαμβάνει όλα τα απαραίτητα βήματα για την εγκατάσταση των απαιτούμενων εξαρτημάτων και βιβλιοθηκών. Ειδικότερα, το Dockerfile κατεβάζει και εγκαθιστά το Slim Framework, το οποίο είναι το κύριο πλαίσιο εργασίας για την υλοποίηση του Backend.
- **index.php**: Το κύριο αρχείο εισόδου της εφαρμογής, όπου ορίζεται η εκκίνηση της εφαρμογής Slim Framework και η δρομολόγηση των αιτημάτων.

3.2.2 Ροή Δεδομένων και Επεξεργασία Αιτημάτων στο Backend

Η αλληλεπίδραση των διαφόρων στοιχείων στο Backend της εφαρμογής ακολουθεί μια οργανωμένη ροή δεδομένων που ξεκινά από το αρχικό αίτημα του χρήστη και καταλήγει στην επεξεργασία του δεδομένου και την επιστροφή του στην εφαρμογή Frontend.

Αίτημα από το Frontend

Όταν ο χρήστης αλληλοεπιδρά με την εφαρμογή στο Frontend (π.χ., με την αποστολή ενός αιτήματος μέσω ενός API ή την αποθήκευση δεδομένων), το αίτημα αποστέλλεται στον Server.

Το αίτημα αυτό φτάνει στο *index.php*, το οποίο είναι το σημείο εκκίνησης της εφαρμογής στον Server. Από εκεί, το αίτημα προωθείται στην κατάλληλη διαδρομή (route) σύμφωνα με το URL και τη μέθοδο του HTTP (GET, POST, PUT, DELETE κλπ.).

Controllers

Όταν το αίτημα φτάσει στο *index.php*, το Slim Framework θα αναλάβει να κατευθύνει το αίτημα στον αντίστοιχο Controller.

Ο Controller είναι υπεύθυνος για την επεξεργασία του αιτήματος και τη σύνδεση με τη σωστή υπηρεσία (Service). Ανάλογα με τον τύπο του αιτήματος, ο Controller μπορεί να καλέσει υπηρεσίες για να ανακτήσει δεδομένα από τη βάση δεδομένων, να εκτελέσει υπολογισμούς ή άλλες λογικές, ή να επιστρέψει ένα response στο Frontend.

Services

Οι υπηρεσίες (Services) είναι υπεύθυνες για τη λογική πίσω από την επεξεργασία των δεδομένων και την επικοινωνία με τη βάση δεδομένων. Για παράδειγμα, όταν το αίτημα αφορά την ενημέρωση ενός πρατηρίου, το Service θα αναλάβει να επικοινωνήσει με τη βάση δεδομένων για να πραγματοποιήσει την ενημέρωση.

Ένα σημαντικό κομμάτι του Service είναι η επικύρωση (validation) των δεδομένων πριν την εκτέλεση οποιασδήποτε ενέργειας με τη βάση δεδομένων. Η υπηρεσία ελέγχει εάν τα δεδομένα που υποβάλλονται είναι έγκυρα (π.χ., η μορφή ενός email ή επικύρωση του σωστού κωδικού πρόσβασης σε περίπτωση ενημέρωσης στοιχείων χρήστη). Εάν τα δεδομένα δεν είναι σωστά, επιστρέφει ένα σφάλμα και ενημερώνει το Frontend.

Ειδικότερα για ενέργειες όπως η ενημέρωση (update), η υπηρεσία εξετάζει αν τα δεδομένα πληρούν όλες τις απαιτήσεις της εφαρμογής πριν προχωρήσει στην ενημέρωση της βάσης.

Database Interaction:

Μετά την επικύρωση των δεδομένων, η υπηρεσία επικοινωνεί με τη βάση δεδομένων μέσω των μοντέλων (Models). Οι υπηρεσίες χρησιμοποιούν τα μοντέλα για να ανακτήσουν ή να αποθηκεύσουν δεδομένα στην βάση.

Τα μοντέλα οργανώνουν τα δεδομένα με τρόπο που αντανακλά τη δομή της βάσης και επιτρέπουν την εύκολη αλληλεπίδραση με τα δεδομένα.

3.3 Αρχιτεκτονική και Δομή της Βάσης Δεδομένων

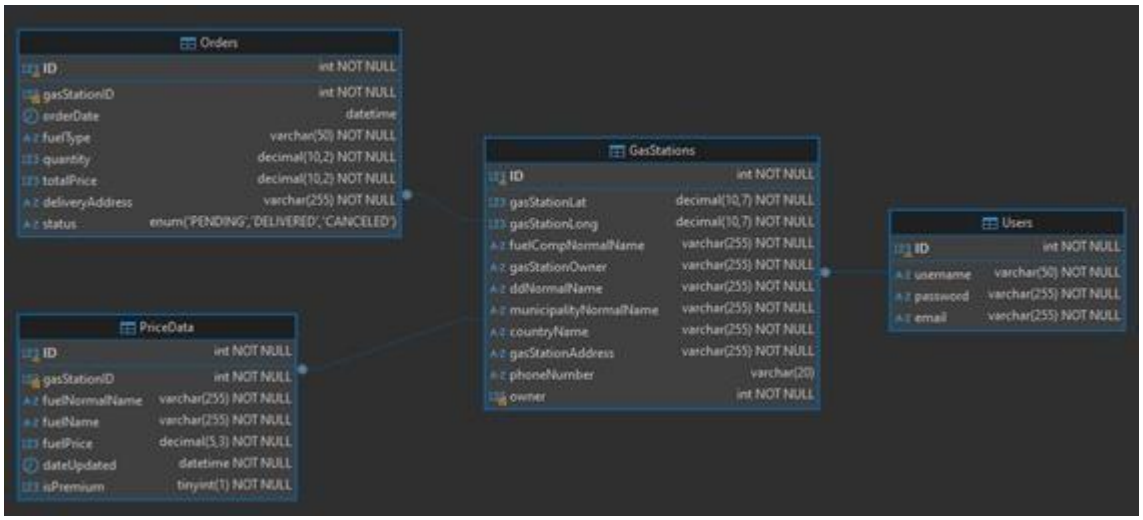
Η βάση δεδομένων της εφαρμογής χρησιμοποιεί MySQL και φιλοξενείται μέσω Docker για ευκολία διαχείρισης. Η δομή της περιλαμβάνει πίνακες που αφορούν χρήστες, πρατήρια και παραγγελίες, με σχέσεις **1:N** και εξωτερικά κλειδιά (foreign keys) για τη διασφάλιση της ακεραιότητας των δεδομένων. Η αρχικοποίηση των πινάκων και η εισαγωγή των πρώτων δεδομένων γίνεται μέσω ενός σεναρίου (script) στον φάκελο *DB* του Backend.

3.3.1 Σχέσεις Στη Βάση Δεδομένων

Οι σχέσεις μεταξύ των πινάκων καθορίζουν την αλληλεπίδραση των δεδομένων στη βάση. Η πιο σημαντική σχέση είναι η σύνδεση μεταξύ των πινάκων **GasStations**, **Orders**, και **PriceData** μέσω του foreign key **gasStationID**. Ο πίνακας **Users** συνδέεται με τον πίνακα **GasStations** μέσω foreign key **owner**, υποδεικνύοντας ποιος είναι ο ιδιοκτήτης κάθε πρατηρίου. Η δομή αυτών των σχέσεων ενισχύει την ακεραιότητα των δεδομένων και επιτρέπει τη σωστή αποθήκευση και ανάκτηση πληροφοριών.

3.3.2 UML Διάγραμμα της Βάσης Δεδομένων

Το παρακάτω UML διάγραμμα απεικονίζει τις σχέσεις και τη δομή των πινάκων της βάσης δεδομένων.



3.3.3 Ανάλυση Πινάκων και Στηλών

Οι πίνακες της βάσης δεδομένων περιλαμβάνουν τα εξής:

GasStations Table

Ο πίνακας αυτός αποθηκεύει τις πληροφορίες των πρατηρίων, όπως τη γεωγραφική θέση, τη διεύθυνση, τον ιδιοκτήτη και άλλες βασικές πληροφορίες. Ο πίνακας συνδέεται με τον πίνακα Users μέσω του πεδίου owner, το οποίο αναφέρεται στο ID του χρήστη.

Στήλες (Columns):

- **ID**: Το μοναδικό αναγνωριστικό για κάθε πρατήριο.
- **gasStationLat**: Η γεωγραφική συντεταγμένη πλάτους του πρατηρίου **gasStationLong**: Η γεωγραφική συντεταγμένη μήκους του πρατηρίου
- **fuelCompNormalName**: Το όνομα της εταιρείας καυσίμων που παρέχει το πρατήριο
- **ddNormalName**: Η Δημοτική Ενότητα στην οποία ανήκει το πρατήριο
- **municipalityNormalName**: Ο Δήμος στον οποίο ανήκει το πρατήριο
- **countryName**: Η χώρα στην οποία βρίσκεται το πρατήριο
- **gasStationAddress**: Η φυσική διεύθυνση του πρατηρίου **phoneNumber**: Προαιρετικός αριθμός τηλεφώνου.
- **owner**: Το ID του ιδιοκτήτη, το οποίο είναι συνδεδεμένο με τον πίνακα Users.

Orders table

Ο πίνακας αυτός αποθηκεύει τις παραγγελίες που γίνονται από τα πρατήρια, με αναφορά στον τύπο καυσίμου, την ποσότητα, την τιμή, τη διεύθυνση παράδοσης και την κατάσταση της παραγγελίας.

Στήλες (Columns):

- **ID**: Το μοναδικό αναγνωριστικό για κάθε παραγγελία **gasStationID**: Το **ID** του πρατηρίου που έκανε την παραγγελία **orderDate**: Η ημερομηνία δημιουργίας της παραγγελίας.
- **fuelType**: Το επίσημο όνομα του καυσίμου
- **quantity**: Η ποσότητα της παραγγελίας σε λίτρα (LT)
- **totalPrice**: Το συνολικό πληρωτέο πόσο της παραγγελίας
- **status**: Η κατάσταση της παραγγελίας

PriceData Table

Ο πίνακας αυτός καταγράφει τις τιμές των καυσίμων που προσφέρονται από τα πρατήρια, με τη χρήση του ξένου κλειδιού (foreign key) **gasStationID**.

Στήλες (Columns):

- **ID**: Το μοναδικό αναγνωριστικό για κάθε εγγραφή τιμής
- **gasStationID**: Το **ID** του πρατηρίου στο οποίο ανήκει η τιμή
- **fuelNormalName**: Το επίσημο όνομα του καυσίμου
- **fuelName**: Ο τύπος του καυσίμου
- **fuelPrice**: Η τιμή του καυσίμου
- **dateUpdated**: Η ημερομηνία που ενημερώθηκε η τιμή
- **isPremium**: Σημαία (flag) που υποδεικνύει αν το καύσιμο είναι premium ή όχι

Users Table

Ο πίνακας αυτός περιέχει τις πληροφορίες των χρηστών της εφαρμογής, όπως όνομα χρήστη, κωδικό πρόσβασης και email.

Στήλες (Columns):

- **ID**: Το μοναδικό αναγνωριστικό για κάθε χρήστη.
- **username**: Το όνομα χρήστη
- **password**: Ο κωδικός του χρήστη
- **email**: Το email του χρήστη

3.4 Διαχείριση Περιβαλλόντων με Docker

Η εφαρμογή FuelConnect χρησιμοποιεί το Docker για τη δημιουργία και διαχείριση απομονωμένων περιβαλλόντων ανάπτυξης και παραγωγής. Η χρήση Docker επιτρέπει την εύκολη διαχείριση των εξαρτημάτων της εφαρμογής, εξασφαλίζοντας συνέπεια και ευκολία στην ανάπτυξη, την υποστήριξη και την εκτέλεση.

3.4.1 Frontend Image

Στο Frontend της εφαρμογής υπάρχει το αρχείο *Dockerfile* το οποίο χτίζει ένα Docker Image. Το Docker image είναι ανεβασμένο στο Docker Hub στο προσωπικό μου repository, επιτρέποντας εύκολη εγκατάσταση και αναβάθμιση σε οποιοδήποτε περιβάλλον.

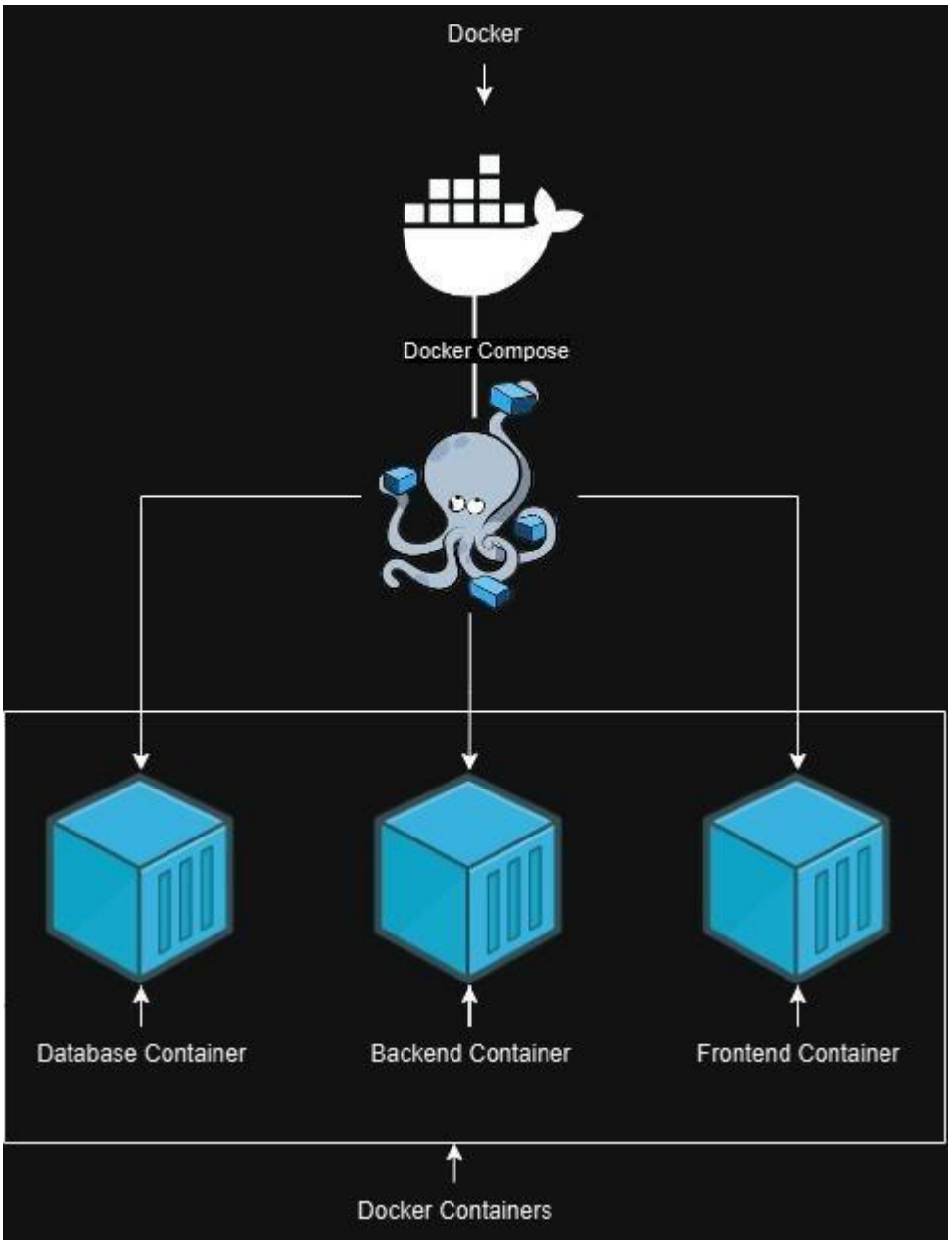
3.4.2 Backend Image

Το Dockerfile για τον Backend δημιουργεί επίσης ένα Docker Image, το οποίο κατεβάζει και εγκαθιστά το Slim Framework μέσα στο container. Αυτό επιτρέπει την απομόνωση της εφαρμογής και διασφαλίζει ότι όλες οι εξαρτήσεις, όπως το Slim Framework, είναι εγκατεστημένες με τη σωστή έκδοση. Το Image του Backend είναι επίσης αποθηκευμένο στο Docker Hub στο προσωπικό μου repository.

3.4.3 Database image

Για τη βάση δεδομένων δεν υπάρχει συγκεκριμένο *Dockerfile* που να χτίζει το container καθώς χρησιμοποιείται το official MySQL Docker Image, το οποίο παρέχει μία πλήρη και αξιόπιστη βάση δεδομένων MySQL σε περιβάλλον container. Η βάση δεδομένων ενσωματώνεται στο σύστημα μέσω του docker-compose, το οποίο διασφαλίζει ότι όλα τα συστατικά της εφαρμογής (Frontend, Backend και Database) είναι συνδεδεμένα και λειτουργούν σωστά μεταξύ τους.

3.4.4 Docker Compose



Η διαχείριση όλων των Containers γίνεται μέσω του Docker Compose, το οποίο συντονίζει τη δημιουργία και εκκίνηση των Containers για το Frontend, Backend και τη βάση δεδομένων. Το αρχείο *docker-compose.yml* περιλαμβάνει τα Images για το Frontend και Backend από το Docker Hub και το MySQL container από το επίσημο MySQL Image. Επιπλέον, το Docker Compose απομονώνει τα Containers σε ένα εσωτερικό δίκτυο, επιτρέποντας την ασφαλή και αξιόπιστη επικοινωνία μεταξύ τους, χωρίς να επηρεάζονται από το εξωτερικό περιβάλλον. Αυτό επιτρέπει την εύκολη εκκίνηση της εφαρμογής σε οποιοδήποτε περιβάλλον με μία μόνο εντολή, διασφαλίζοντας τη σωστή λειτουργία και σύνδεση των υπηρεσιών.

3.4.5 Ανάλυση του αρχείου docker-compose.yml

```yaml
version: '3.1'
services:
  backend:
    image: chrispolgg/fuel-connect-backend:latest
    container_name: backend
    build:
      context: ./backend/src
    ports:
      - "8080:80"
    volumes:
      - ./backend/src:/var/www/html
      - ./backend/logs:/var/log/apache2
    depends_on:
      - mysql
    environment:
      DB_HOST: mysql
      DB_NAME: fuelGr
      DB_USER: admin
      DB_PASS: polpw!
      DB_CHARSET: utf8mb4
  frontend:
    image: chrispolgg/fuel-connect-frontend:latest
    container_name: frontend
    build:
      context: ./frontend
    ports:
      - "15040:15040"
    volumes:
      - ./frontend:/app
      - /app/node_modules
    command: ["npm", "run", "local"]
    depends_on:
      - backend
  mysql:
    image: mysql
    container_name: mysql-db
    restart: always
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: rootpw
      MYSQL_DATABASE: fuelGr
      MYSQL_USER: admin
      MYSQL_PASSWORD: polpw!
    volumes:
      - ./backend/DB/db-data/mysql/logs:/var/log/mysql
      - ./backend/DB/db-data/mysql/data:/var/lib/mysql
      - ./backend/DB/init/db_init.sql:/docker-entrypoint-initdb.d/db_init.sql:ro
```

Στην παραπάνω εικόνα βλέπουμε την δομή του αρχείου *docker-compose.yml*.

Services

1. **backend**

Το service backend αντιστοιχεί στον server της εφαρμογής και βασίζεται σε custom Docker image (chrispolgg/fuel-connect-backend). Αυτό το Service είναι υπεύθυνο για την εξυπηρέτηση των αιτημάτων του Backend της εφαρμογής, χρησιμοποιώντας τον Apache Server και το Slim framework με PHP.

- **Build**: Το build πεδίο ορίζει το context στο ./backend/src, δηλαδή το σημείο που βρίσκονται τα αρχεία του backend
- **Ports**: Εκτίθεται στη θύρα 8080 του host, παρέχοντας έτσι πρόσβαση στον backend Server από τον έξω κόσμο μέσω αυτής της θύρας **Volumes**:
    - Ο φάκελος ./backend/src συνδέεται με τον /var/www/html μέσα στο container, επιτρέποντας στον Apache να έχει πρόσβαση στον πηγαίο κώδικα
    - Ο φάκελος ./backend/logs συνδέεται με τον /var/log/apache2, αποθηκεύοντας τα logs του Apache για διαχείριση και ανάλυση
- **Environment Variables**: Παρέχονται μεταβλητές για τη σύνδεση με τη βάση δεδομένων (DB_HOST, DB_NAME, DB_USER, DB_PASS, DB_CHARSET)
- **depends_on**: Δηλώνει ότι το service mysql πρέπει να έχει ξεκινήσει πρώτα, ώστε το backend να μπορέσει να συνδεθεί στη βάση δεδομένων

2. **Frontend**

   Το service Frontend αντιστοιχεί στο React Frontend της εφαρμογής. Χρησιμοποιεί ένα custom Docker Image (chrispolgg/fuel-connect-frontend) και παρέχει τη διεπαφή χρήστη για το FuelConnect

   - **Build**: Το context έχει οριστεί στο ./frontend, που περιέχει τον κώδικα της React εφαρμογής
   - **Ports**: Εκτίθεται στη θύρα 15040, επιτρέποντας στους χρήστες να αποκτούν πρόσβαση στο frontend μέσω του host port 15040 **Volumes**:
       - Ο φάκελος ./frontend/app συνδέεται με τον φάκελο /app του container, επιτρέποντας συγχρονισμό των αρχείων του frontend
       - Το node_modules του frontend συνδέεται επίσης για να είναι άμεσα διαθέσιμο στο container

- **Command**: Χρησιμοποιείται η εντολή **npm run local**, που πιθανώς εκκινεί την εφαρμογή σε τοπικό περιβάλλον ανάπτυξης
- **depends_on**: Εξαρτάται από το backend service για να εξασφαλίσει ότι ο backend Server είναι διαθέσιμος όταν εκκινεί το frontend

3. mysql

   Το service mysql αντιστοιχεί στη βάση δεδομένων της εφαρμογής και βασίζεται σε επίσημο Docker Image της MySQL

   - **Ports**: Εκτίθεται στη θύρα 3306, που είναι η προεπιλεγμένη θύρα για τη MySQL και επιτρέπει τη σύνδεση με τη βάση δεδομένων
   - **Environment Variables**: Οι μεταβλητές περιβάλλοντος περιλαμβάνουν το root password, το όνομα της βάσης (fuelGr), καθώς και τα credentials για τη σύνδεση (MYSQL_USER και MYSQL_PASSWORD) **Volumes**:
     - Ο φάκελος ./backend/DB/db-data συνδέεται με τους φακέλους var/lib/mysql και var/log/mysql, διασφαλίζοντας την αποθήκευση των δεδομένων και των logs της βάσης
     - Το αρχείο db_init.sql εκτελείται κατά την αρχικοποίηση, εξασφαλίζοντας ότι η βάση δεδομένων θα έχει την αρχική δομή και δεδομένα που απαιτούνται από την εφαρμογή

4 Λειτουργικότητα και Διαχείριση της Εφαρμογής

Το κεφάλαιο αυτό αναλύει τις βασικές λειτουργίες της εφαρμογής, εστιάζοντας στη συμπεριφορά των χρηστών και στην αλληλεπίδραση τους με το σύστημα. Αναφέρονται οι δυνατότητες που παρέχονται στους χρήστες, οι οποίοι διακρίνονται σε παλιούς και νέους, ανάλογα με τα δικαιώματα τους.

4.1 Περιγραφή Χρηστών και Ρόλων

Στην εφαρμογή υπάρχουν δύο κατηγορίες χρηστών: οι παλιοί και οι νέοι χρήστες. Οι παλιοί χρήστες είναι αυτοί που συνδέονται με ήδη καταχωρημένα πρατήρια στην

πλατφόρμα, τα οποία έχουν εισαχθεί μέσω των αρχικών δεδομένων στη βάση. Αυτοί οι χρήστες έχουν τη δυνατότητα να τροποποιούν τις τιμές των καυσίμων στα πρατήρια τους. Από την άλλη, οι νέοι χρήστες, οι οποίοι εγγράφονται στην πλατφόρμα, δεν έχουν τη δυνατότητα να επεξεργαστούν τιμές, καθώς δεν διαθέτουν δικά τους πρατήρια στην πλατφόρμα.

4.2 Λειτουργικότητα εφαρμογής

Η εφαρμογή παρέχει στους χρήστες τη δυνατότητα να περιηγηθούν στον χάρτη και να δουν τα πρατήρια που υπάρχουν στην περιοχή τους, καθώς και να παρακολουθήσουν τις τιμές καυσίμων σε αυτά. Η πλατφόρμα υποστηρίζει την αναζήτηση πρατηρίων μέσω φίλτρων, όπως γεωγραφική θέση, τύπος καυσίμου κ.ά., και επιτρέπει την προβολή περισσότερων λεπτομερειών για κάθε πρατήριο, όπως η διεύθυνση, η τιμή των καυσίμων και οι τρόποι επικοινωνίας.

Επιπλέον, οι χρήστες μπορούν να παραγγείλουν καύσιμα μέσω του μενού λεπτομερειών κάθε πρατηρίου, διευκολύνοντας την παραγγελία και την παραλαβή καυσίμων στον επιθυμητό προορισμό τους.

4.3 Ανάλυση των API Endpoints

Η εφαρμογή αλληλεπιδρά με το backend μέσω διαφόρων API endpoints, τα οποία διαχειρίζονται τις βασικές λειτουργίες της πλατφόρμας. Τα κυριότερα endpoints της εφαρμογής περιλαμβάνουν τα εξής:

Διαχείριση Χρηστών

- **POST /updateUser**: Επιτρέπει την ενημέρωση των στοιχείων του χρήστη. Στο body του request υπάρχουν τα πεδία που θέλει να ενημερώσει ο χρήστης σε μορφή **JSON**.
- **POST /register**: Δημιουργεί έναν νέο χρήστη στην πλατφόρμα με τα απαραίτητα στοιχεία. Στο body του request υπάρχουν τα στοιχεία που πληκτρολόγησε ο χρήστης σε μορφή **JSON**.
- **POST /login**: Επιτρέπει στους χρήστες να συνδεθούν στην εφαρμογή με τα στοιχεία τους. Στο body του request υπάρχουν τα στοιχεία που πληκτρολόγησε ο χρήστης σε μορφή **JSON.**

Διαχείριση Πρατηρίων

- **GET /gasStations**: Επιστρέφει τη λίστα με όλα τα πρατήρια στην πλατφόρμα.
- **GET /gasStationDetails/{id}**: Επιστρέφει τις λεπτομέρειες για ένα συγκεκριμένο πρατήριο (παράμετρος είναι το **ID** του συγκεκριμένου πρατηρίου).

Λειτουργικότητα Αναζητήσεων

- **POST /search**: Επιτρέπει την αναζήτηση πρατηρίων με συγκεκριμένα κριτήρια. Στο body του request υπάρχουν τα φίλτρα που επέλεξε ο χρήστης σε μορφή **JSON**.
- **GET /filtersOptions**: Παρέχει τις διαθέσιμες επιλογές φίλτρων (dropdown options) για την αναζήτηση πρατηρίων.

Διαχείριση Παραγγελιών Καυσίμου

- **POST /order**: Δημιουργεί μια παραγγελία καυσίμου, καθορίζοντας το πρατήριο, τον τύπο και την ποσότητα καυσίμου. Στο body του request υπάρχουν οι πληροφορίες για την παραγγελία σε μορφή **JSON**.

Διαχείριση Τιμών Καυσίμων

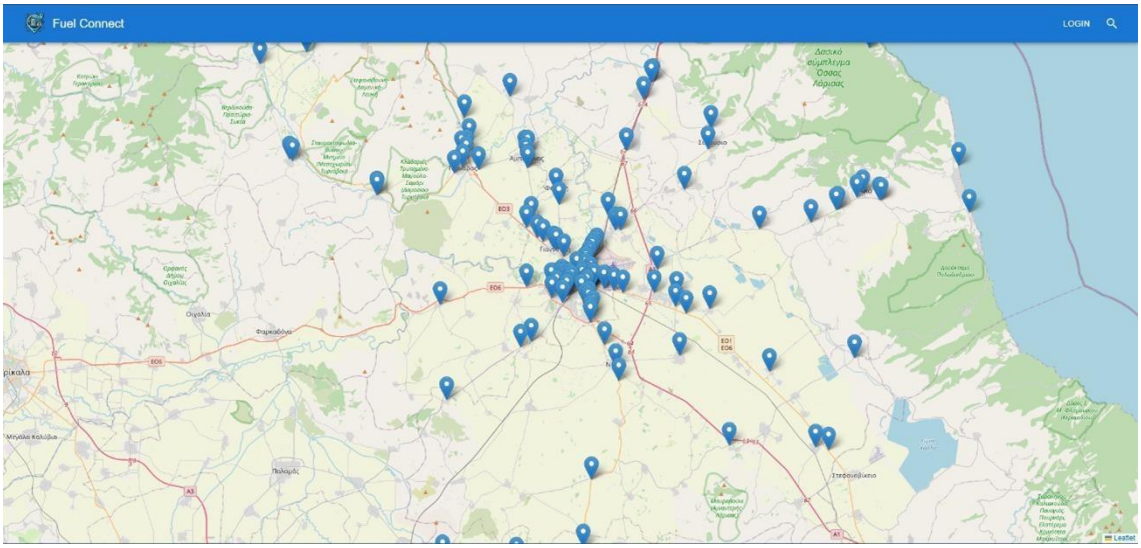
- **GET /fuelTypesAndPrices/{id}**: Επιστρέφει τις τιμές των καυσίμων για ένα συγκεκριμένο πρατήριο (Παράμετρος το **ID** του πρατηρίου).
- **POST /updateFuelsPrices**: Επιτρέπει την ενημέρωση των τιμών καυσίμου για ένα πρατήριο (διαθέσιμο μόνο για ιδιοκτήτες). Στο body του request υπάρχουν οι πληροφορίες για την ενημέρωση του καυσίμου σε μοργή **JSON**.

5 Οπτική Παρουσίαση της Εφαρμογής

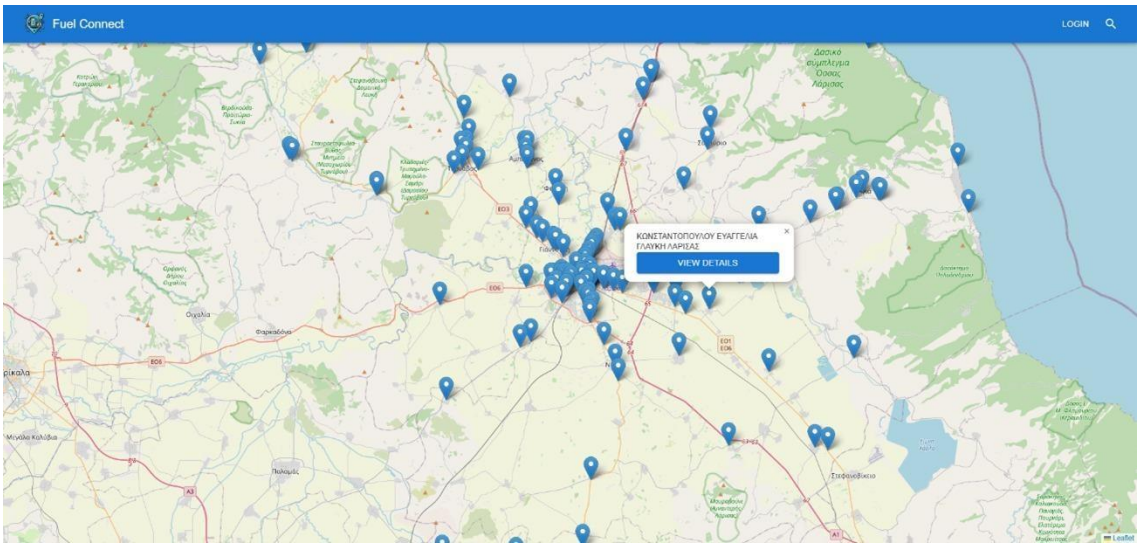
Αυτό το κεφάλαιο παρέχει μια οπτική αναπαράσταση της εφαρμογής, επιδεικνύοντας τις κύριες λειτουργίες και διεπαφές χρήστη μέσω αντιπροσωπευτικών εικόνων. Με τη βοήθεια αυτών, ο αναγνώστης μπορεί να κατανοήσει καλύτερα τη ροή και την αλληλεπίδραση της εφαρμογής, βλέποντας πώς η θεωρητική περιγραφή συνδυάζεται με την πρακτική υλοποίηση. Οι παρακάτω εικόνες δείχνουν τον χάρτη πρατηρίων, την αναζήτηση πρατηρίων, την είσοδο (login) και εγγραφή (register) του χρήστη στην εφαρμογή, την ενημέρωση τιμών καυσίμων καθώς και τη διαδικασία παραγγελίας καυσίμου.

Αρχική Οθόνη

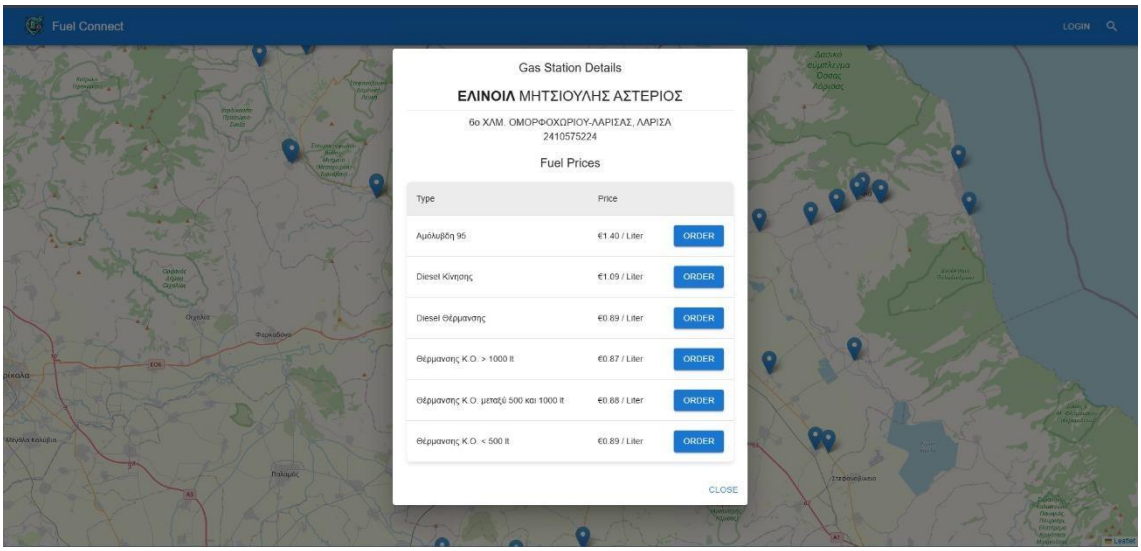
Στην αρχική οθόνη της εφαρμογής, ο χρήστης βλέπει έναν δια δραστικό χάρτη με διάφορους δείκτες τοποθεσίας που αντιπροσωπεύουν ένα πρατήριο. Ο χάρτης επιτρέπει στον χρήστη να κάνει zoom in και zoom out για να εξερευνήσει τις διάφορες τοποθεσίες.



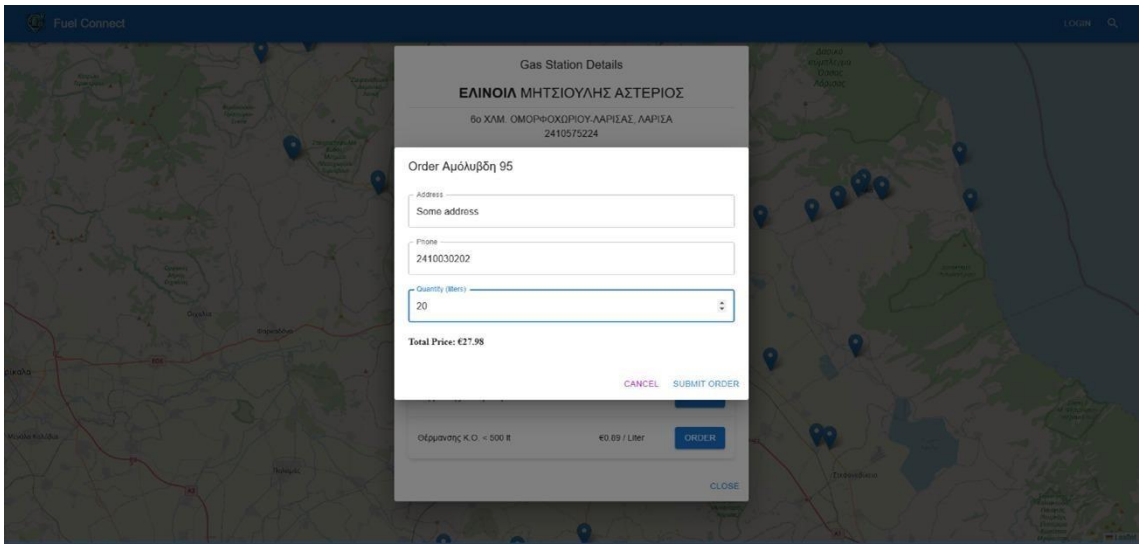
Πατώντας πάνω σε ένα δείκτη τοποθεσίας ανοίγει ένα μικρό modal με τα στοιχεία του ιδιοκτήτη και την διεύθυνση του πρατηρίου και ένα κουμπί **VIEW DETAILS**.



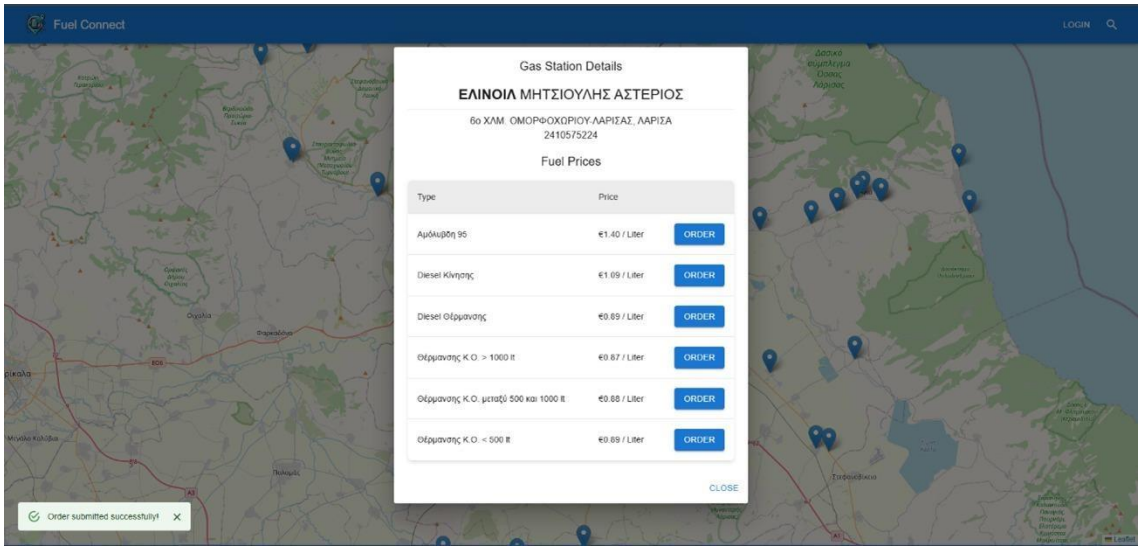
Πατώντας το **VIEW DETAILS** κουμπί, ο χρήστης μπορεί να δει σε ένα Modal τα πλήρη στοιχεία για το πρατήριο, όπως τιμές καυσίμων και τρόπους επικοινωνίας. Από αυτό το Modal του δίνεται επίσης η δυνατότητα να παραγγείλει κάποιο καύσιμο μέσω του κουμπιού **Order**.



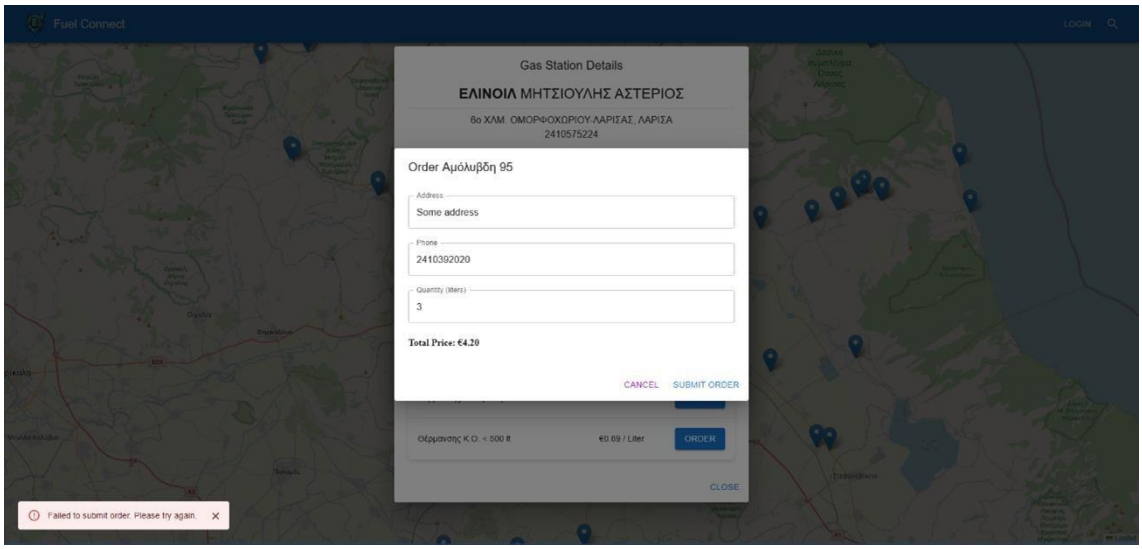
Πατώντας το κουμπί **ORDER** ανοίγει ένα καινούργιο Modal. Εδώ ο χρήστης συμπληρώνει την διεύθυνση του, το τηλέφωνο του και την ποσότητα σε λίτρα, που επιθυμεί. Το συνολικό ποσό εμφανίζεται όσο ο χρήστης πληκτρολογεί την ποσότητα.



Ο χρήστης μπορεί να τοποθετήσει την παραγγελία του πατώντας το **Submit Order** κουμπί. Αν η παραγγελία τοποθετηθεί με επιτυχία τότε το σύστημα ενημερώνει τον χρήστη με ένα success alert στην κάτω αριστερή γωνία.

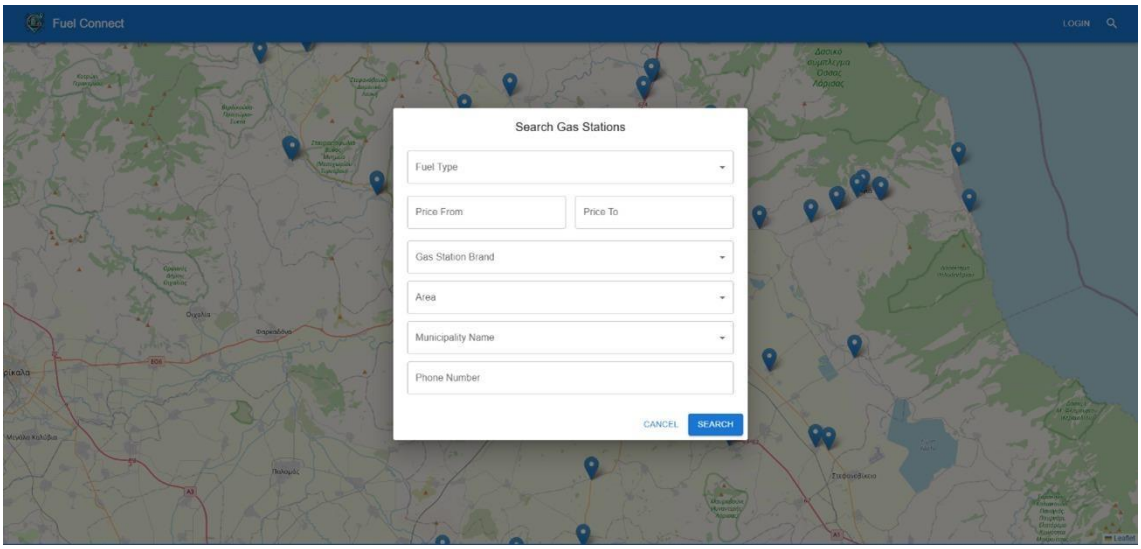


Αν υπήρξε κάποιο πρόβλημα με τον server και η παραγγελία δεν καταχωρήθηκε τότε το σύστημα ενημερώνει τον χρήστη με ένα error alert.

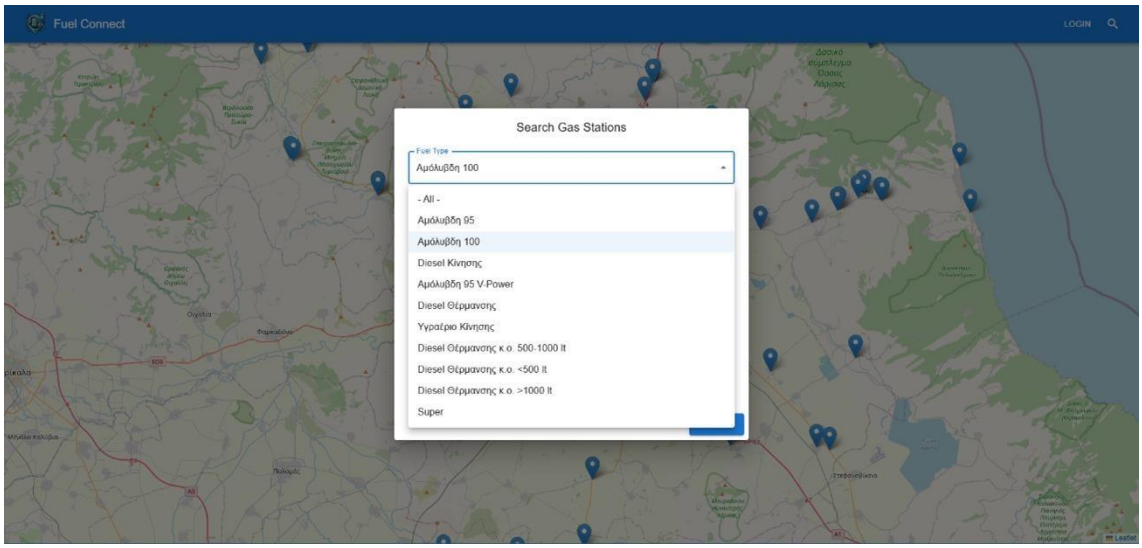


Μηχανισμός Αναζήτησης

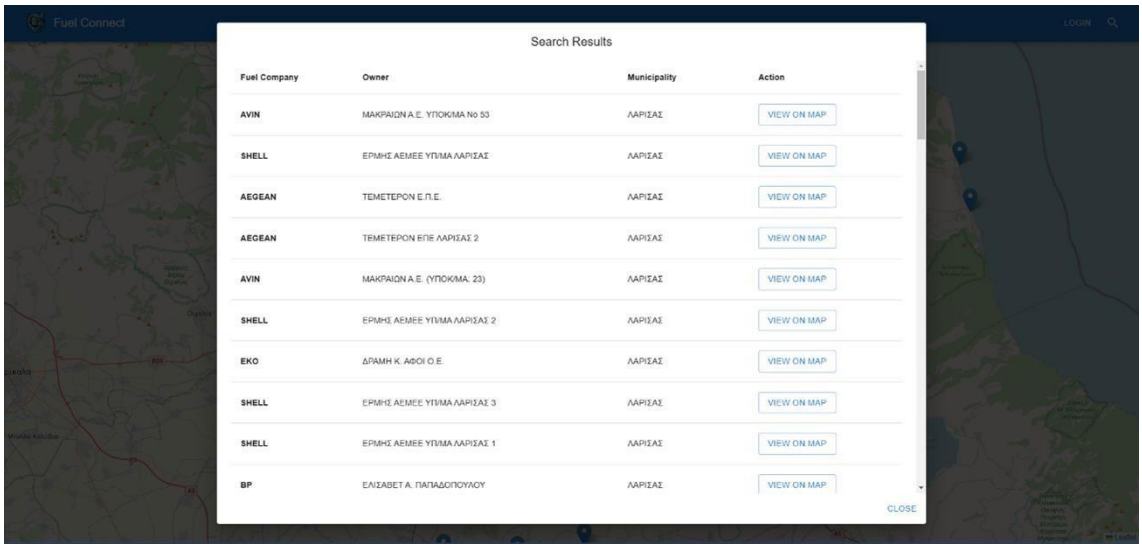
Στο Navigation Bar ο χρήστης μπορεί να κάνει αναζήτηση για κάποιο πρατήριο πατώντας πάνω στον μεγεθυντικό φακό (Search). Πατώντας θα ανοίξει ένα Modal με διάφορα φίλτρα που ο χρήστης μπορεί να επιλέξει για να ψάξει με τις δικές του προτιμήσεις.



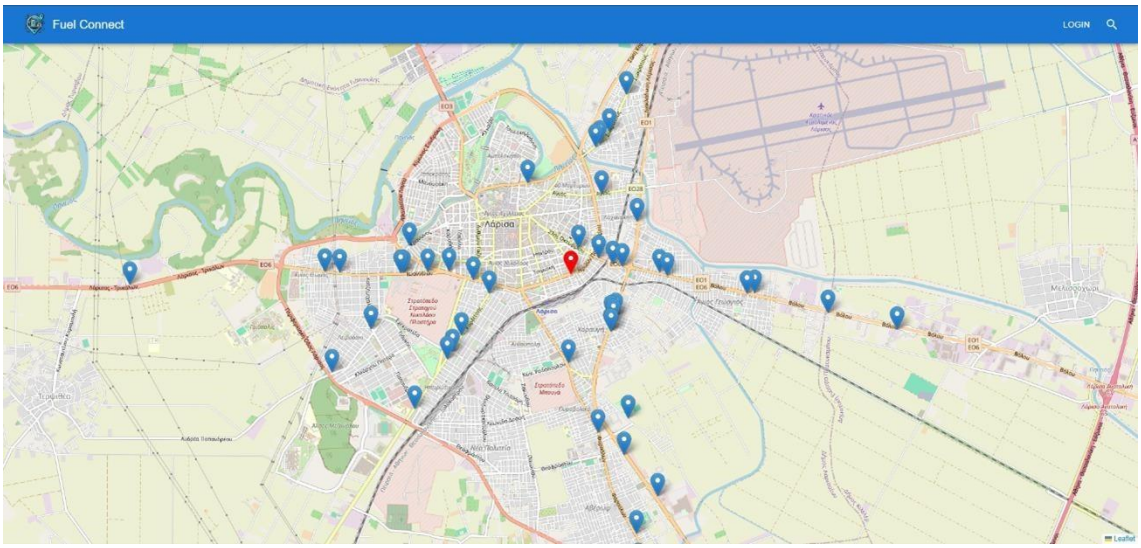
Τα dropdown πεδία περιέχουν επιλογές οι οποίες έρχονται από την βάση δεδομένων.



Πατώντας το κουμπί Search ανοίγει ένα καινούργιο Modal με τα αποτελέσματα της αναζήτησης. Σε αυτό το Modal βλέπουμε πληροφορίες για το πρατήριο όπως την εταιρία, τον ιδιοκτήτη, και τον Νομό στον οποίο βρίσκεται.

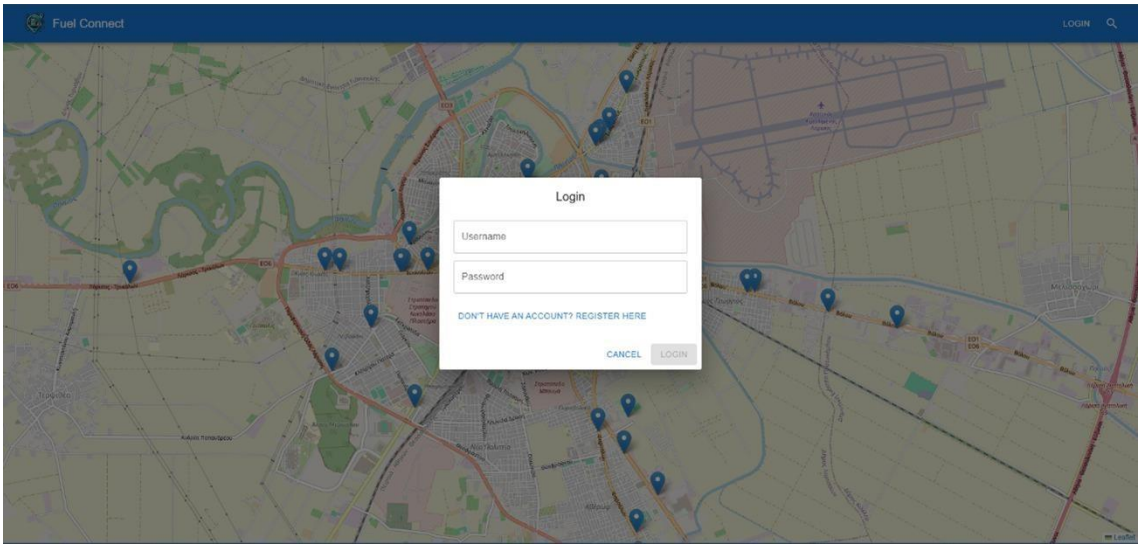


Υπάρχει επίσης ένα κουμπί **VIEW ON MAP** το οποίο πατώντας το εστιάζει στο χάρτη στο συγκεκριμένο πρατήριο το οποίο είναι highlighted με έναν κόκκινο δείκτη τοποθεσίας.

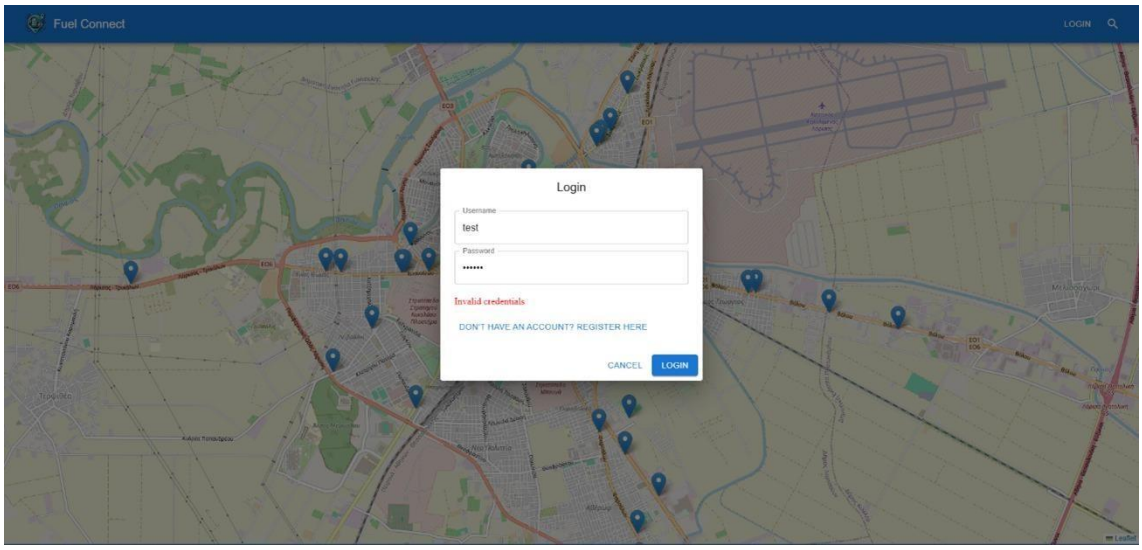


Login/Register

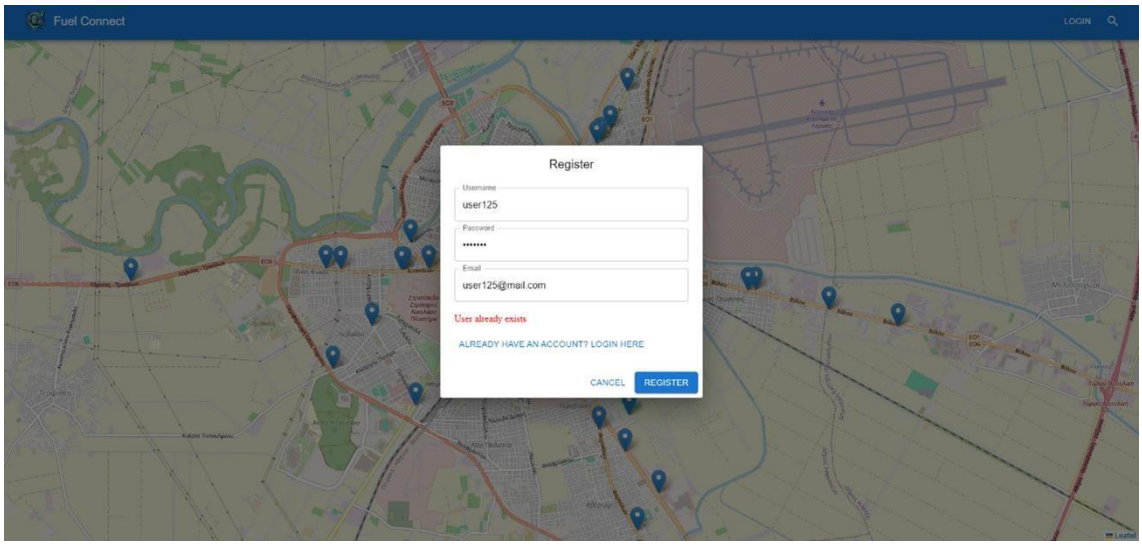
Στο Navigation Bar ο χρήστης μπορεί να βρει και το κουμπί **LOGIN** μέσω του οποίου ανοίγει ένα modal όπου ο χρήστης επιλέγει αν θέλει να κάνει σύνδεση (login) ή εγγραφή (register).



Στην περίπτωση που ο χρήστης επιλέξει να κάνει σύνδεση και δώσει λάθος στοιχεία το σύστημα τον ενημερώνει ότι τα στοιχεία που έδωσε είναι λανθασμένα.



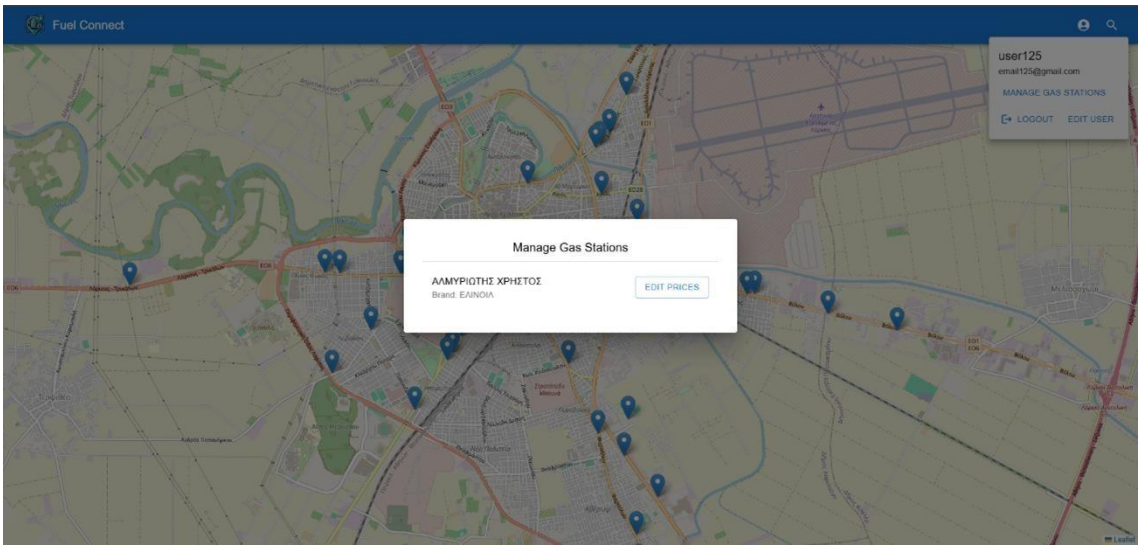
Αν η σύνδεση είναι επιτυχής τότε ο χρήστης αποθηκεύεται στην Local Storage του browser για να μην χρειαστεί να ξανακάνει log in αν τυχόν κάνει ανανέωση την ιστοσελίδα. Στην περίπτωση που ο χρήστης επιλέξει να κάνει εγγραφή και δώσει κάποιον user ο οποίος υπάρχει ήδη τότε το σύστημα τον ενημερώνει ότι υπάρχει ο χρήστης με τα συγκεκριμένα στοιχεία.



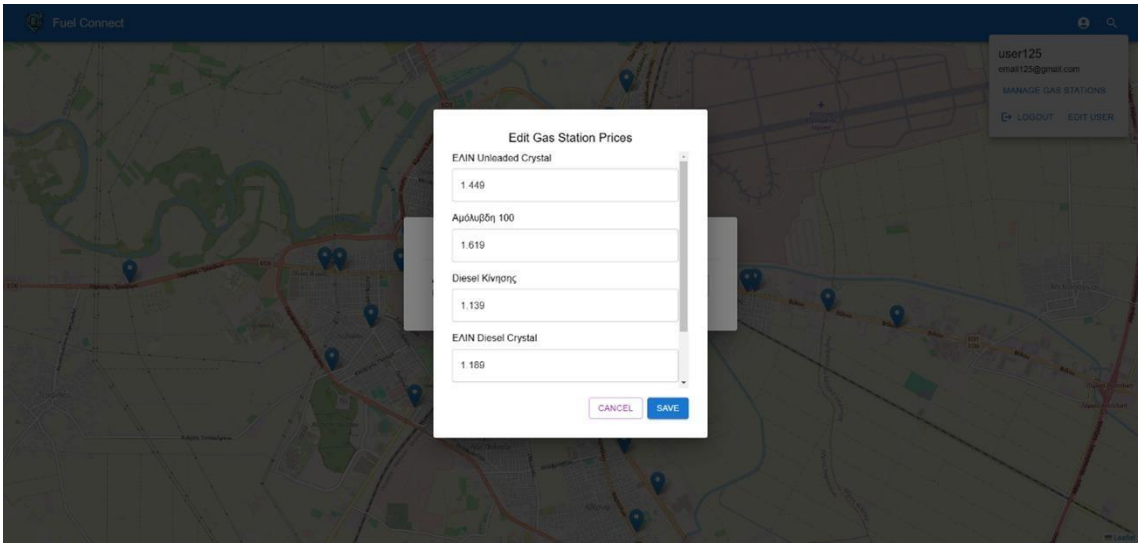
Διαχείριση Τιμών Πρατηρίου

Εφόσον ο χρήστης κάνει login μπορεί να παρατηρήσει ένα εικονίδιο στην πάνω δεξιά γωνία το οποίο αν το πατήσει θα δει πληροφορίες για αυτόν όπως το Username και το email του. Θα δει επίσης κάποιες άλλες επιλογές όπως **MANAGE GAS STATIONS**.

Πατώντας το κουμπί αυτό ανοίγει ένα Modal το οποίο περιέχει μια λίστα με τα πρατήρια του χρήστη.

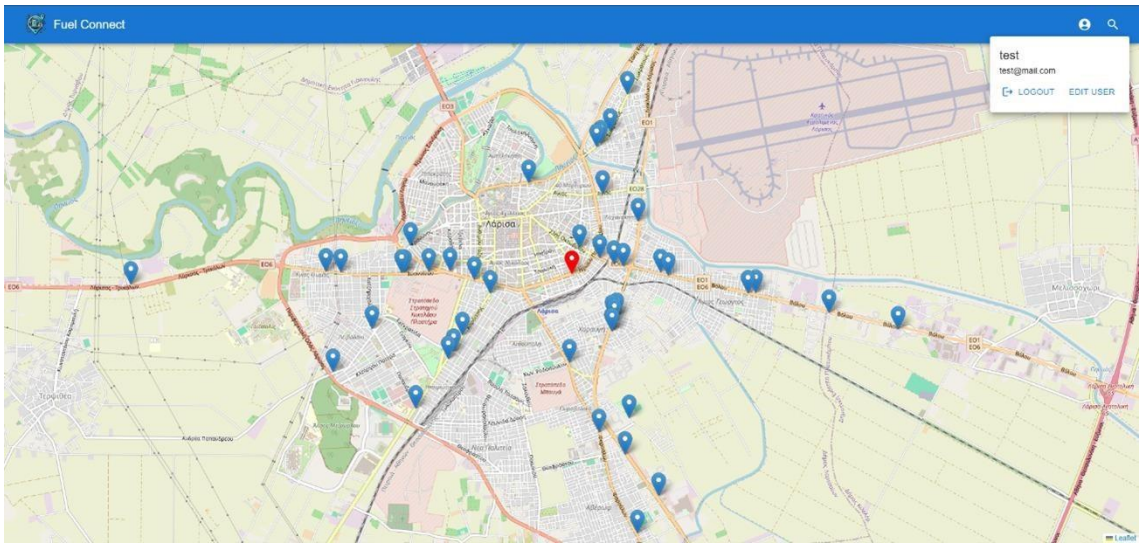


Μέσω του κουμπιού **EDIT PRICES** μπορεί να αλλάξει τις τιμές των καυσίμων στο πρατήριο του.

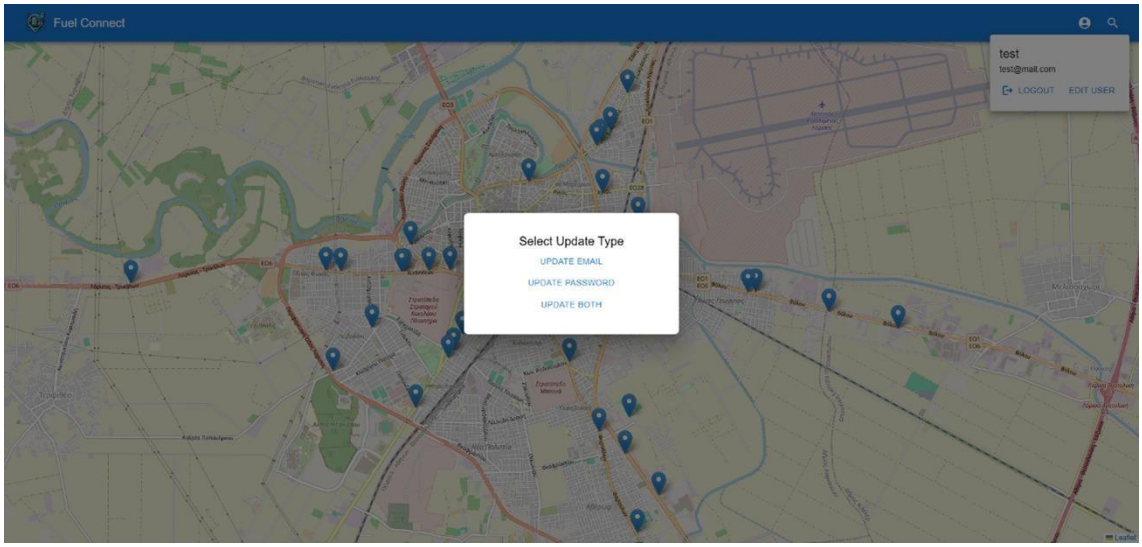


Αντίστοιχα με την παραγγελία καυσίμου, πατώντας το save, αν η ενημέρωση ήταν επιτυχής τότε ο χρήστης θα δει ένα success alert στην κάτω αριστερή γωνία αλλιώς θα δει ένα error alert.

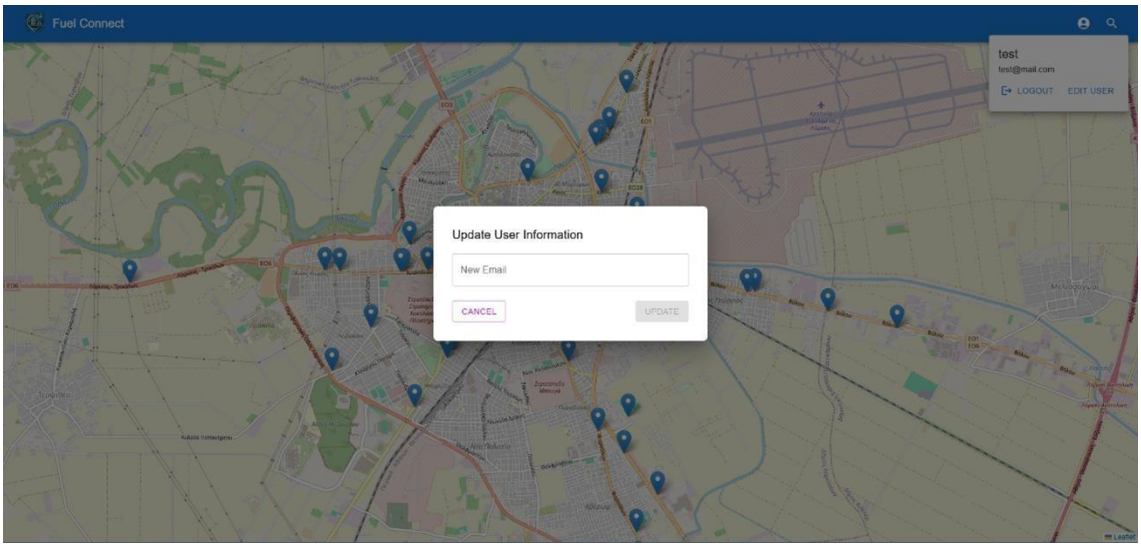
Αν ο επιλέξει να κάνει εγγραφή θα δούμε ότι δεν έχει πρόσβαση σε αυτή την λειτουργιά καθώς δεν είναι ιδιοκτήτης κάποιο πρατηρίου.



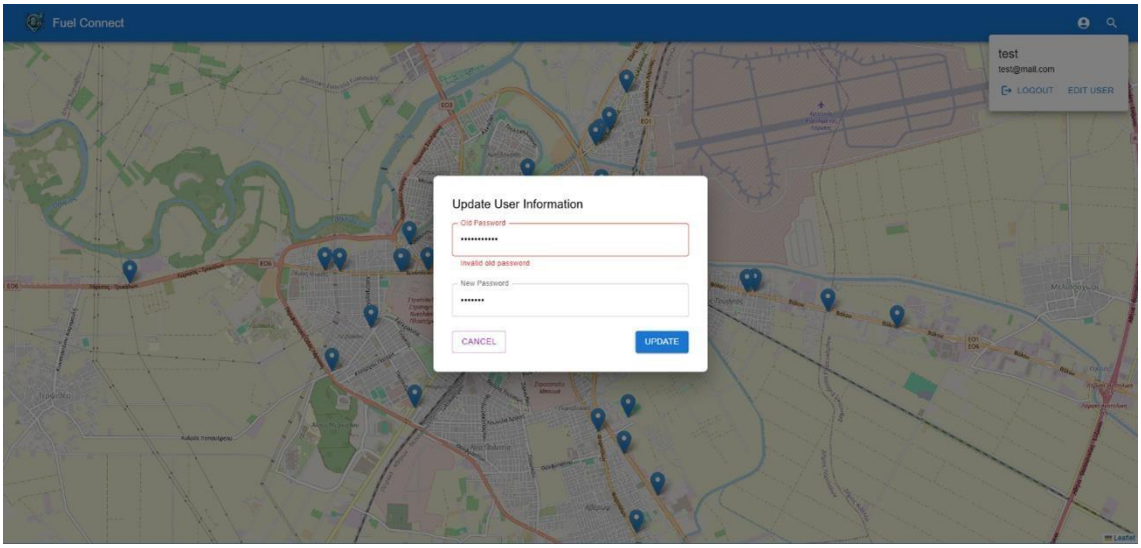
Και στις δύο περιπτώσεις (σύνδεση ή εγγραφή) ο χρήστης έχει την δυνατότητα να κάνει αποσύνδεση μέσω του κουμπιού **LOGOUT** όπου και διαγράφει τα στοιχεία του από την Local Storage του browser. Μπορεί επίσης να ενημερώσει τον κωδικό ή το mail του ή και τα δύο πατώντας στο **EDIT USER**, το οποίο θα ανοίξει ένα modal και ο χρήστη επιλέγει τι επιθυμεί να κάνει ενημέρωση.



Για την ενημέρωση του email δεν απαιτείται ο κωδικός πρόσβασης.



Αν ο χρήστης επιθυμεί να αλλάξει τον κωδικό πρόσβασης τότε θα πρέπει επίσης να συμπληρώσει τον τωρινό κωδικό του. Αν αυτός ο κωδικός είναι λάθος τότε το σύστημα τον ενημερώνει ότι δεν ταιριάζει ο κωδικός που πληκτρολόγησε.



Όπως και στα προηγούμενα requests αν ήταν επιτυχής η ενημέρωση το σύστημα ειδοποιεί τον χρήστη με ένα success alert στην κάτω αριστερή γωνία, και αντίστοιχα ένα error alert αν παρουσιάστηκε κάποιο σφάλμα.

6 Συμπεράσματα

Στο πλαίσιο της πτυχιακής εργασίας, αναπτύχθηκε μια Web εφαρμογή για την αναζήτηση και διαχείριση πρατηρίων, επιτρέποντας στους χρήστες να εντοπίζουν τα πλησιέστερα

πρατήρια καυσίμων, να συγκρίνουν τιμές, και να πραγματοποιούν παραγγελίες καυσίμων από την πλατφόρμα. Η εφαρμογή σχεδιάστηκε και υλοποιήθηκε χρησιμοποιώντας το React και το Material UI για την πλευρά του Frontend, ενώ η Slim PHP χρησιμοποιήθηκε στο Backend για την ανάπτυξη των REST API. Η διαχείριση των Containers με Docker και Docker Compose προσέφερε ευκολία στην εγκατάσταση και λειτουργία της εφαρμογής σε οποιοδήποτε περιβάλλον, ενώ η χρήση του Leaflet για την απεικόνιση των σημείων στον χάρτη προσέφερε μια εύχρηστη εμπειρία πλοήγησης.

Κατά την ανάπτυξη, προέκυψαν ορισμένες προκλήσεις, όπως η βέλτιστη διαχείριση των δεδομένων των πρατηρίων και η σωστή αναπαράσταση της δια δραστικότητας των χαρτών. Ένα ζήτημα ήταν ο συγχρονισμός της επιλογής των πρατήριων με την εστίαση στον χάρτη, το οποίο αντιμετωπίστηκε με την προσθήκη κατάλληλων hooks και λειτουργιών για κεντρική διαχείριση των Modal Views. Επιπλέον, ο σχεδιασμός της αρχιτεκτονικής με βάση τα Containers και τη διασύνδεσή τους με Docker και Docker Compose αρχικά φάνηκε απαιτητικός, ωστόσο κατέστησε το σύστημα πιο σταθερό και ευέλικτο για μελλοντικές επεκτάσεις.

Αν ξεκινούσα την πτυχιακή εργασία από την αρχή, πιθανόν να προσέθετα εξ αρχής περισσότερες δυνατότητες διαχείρισης χρηστών και να έδινα μεγαλύτερη έμφαση στην ανάπτυξη πιο σύνθετων φίλτρων αναζήτησης για τα πρατήρια, καθώς αυτά κρίθηκαν σημαντικά για τη συνολική εμπειρία χρήστη. Συνολικά, η εργασία αυτή αποτέλεσε μια πολύτιμη ευκαιρία εμβάθυνσης σε σύγχρονες τεχνολογίες ανάπτυξης λογισμικού και με ώθησε να ασχοληθώ περαιτέρω με το θέμα της ανάπτυξης εφαρμογών σε containerized περιβάλλον.

Μελλοντική Επέκταση

Για τη συνέχιση και επέκταση της εφαρμογής, προτείνονται αρκετές βελτιώσεις που θα εμπλουτίσουν τη λειτουργικότητά της και θα ενισχύσουν την εμπειρία των χρηστών:

Δυνατότητα προσθήκης πρατηρίων από νέους χρήστες: Θα μπορούσε να δοθεί η επιλογή σε νέους χρήστες να καταχωρούν τα δικά τους πρατήρια, εφόσον είναι ιδιοκτήτες, διευρύνοντας το δίκτυο και τα διαθέσιμα δεδομένα στην πλατφόρμα.

Προσθήκη επιπλέον πρατηρίων για υπάρχοντες χρήστες: Οι χρήστες που ήδη διαθέτουν καταχωρημένα πρατήρια θα μπορούν να προσθέτουν νέα πρατήρια στον λογαριασμό τους, αν είναι επίσης ιδιοκτήτες, διευκολύνοντας έτσι τη διαχείριση πολλαπλών επιχειρήσεων.

Παρακολούθηση παραγγελιών: Η εισαγωγή μιας λειτουργίας παρακολούθησης παραγγελιών θα επιτρέπει στους χρήστες να βλέπουν την κατάσταση κάθε παραγγελίας σε πραγματικό χρόνο, αυξάνοντας τη διαφάνεια και την αμεσότητα της διαδικασίας.

Πρόσθετες διαδικασίες επικύρωσης: Ένας τομέας βελτίωσης είναι η ενίσχυση της ασφάλειας, απαιτώντας από τους χρήστες να είναι συνδεδεμένοι για να παραγγείλουν καύσιμα. Αυτό θα διασφαλίσει τη γνησιότητα των παραγγελιών και την προστασία των χρηστών.

Διαχωρισμός ρόλων σε πελάτες και ιδιοκτήτες: Η δημιουργία διακριτών ρόλων θα επιτρέψει την προσθήκη εξειδικευμένων λειτουργιών και περιορισμών ανάλογα με τον τύπο του χρήστη, βελτιώνοντας την ασφάλεια και την εμπειρία των ιδιοκτητών και πελατών αντίστοιχα.

Αυτές οι προτάσεις μπορούν να προσφέρουν σημαντικές δυνατότητες για την πλατφόρμα, καθιστώντας τη μια ολοκληρωμένη και δυναμική λύση στον τομέα της αναζήτησης και παραγγελίας καυσίμων.

7 Βιβλιογραφία

[1] Welling, L., & Thomson, L., PHP & MySQL: Server-side Web Development, Pearson Education, pp. 159-199, 209-221, 2016.

[2] Rippon, C. Learn React with TypeScript: Develop strongly typed applications using hooks, context, and Redux. Packt Publishing, 2023.

[3] https://docs.docker.com/ Official Documentation about Docker, Docker Overview.

[4] https://docs.docker.com/compose/ Official Documentation about Docker Compose

[5] https://docs.docker.com/docker-hub/ Official Documentation about Docker Hub