Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών

Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων

Ανάπτυξη διαδικτυακής εφαρμογής για τη βελτιστοποίηση της διαδικασίας φόρτισης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΝΤΑΝΙΕΛΑ ΣΤΟΓΙΑΝ

Επιβλέπων Καθηγητής: Χρυσόστομος Δούκας

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών

Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων

Ανάπτυξη διαδικτυακής εφαρμογής για τη βελτιστοποίηση της διαδικασίας φόρτισης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΝΤΑΝΙΕΛΑ ΣΤΟΓΙΑΝ

Εγκρίθηκε από την τριμελή επιτροπή την 12/07/2023

Χρυσόστομος Δούκας Ιωάννης Ψαρράς Ευάγγελος Μαρινάκης

Ντανιέλα Στογιάν

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

Copyright © Ντανιέλα Στογιάν, 2023

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας Εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της Εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η παρούσα διπλωματική στοχεύει στην ανάπτυξη μιας διαδικτυακής εφαρμογής που θα αναπαριστά τους σταθμούς φόρτισης στην Ελλάδα και μέσω ενός αλγορίθμου βελτιστοποίησης θα προτείνει στον χρήστη τον βέλτιστο σταθμό. Τα τελευταία χρόνια υπάρχει ολοένα και μεγαλύτερη αύξηση της ηλεκτροκίνησης με αποτέλεσμα να είναι πλέον απαραίτητος ο έξυπνος προγραμματισμός των διαδικασιών φόρτισης. Λόγω των πολλών παρόχων σταθμών φόρτισης αλλά και των ξεχωριστών εφαρμογών που διαθέτουν, συχνά οι χρήστες οδηγούνται στην χρήση συγκεκριμένων σταθμών φόρτισης, έναντι των πιο κοντινών, με αποτέλεσμα να δημιουργείται συνωστισμός και να βρίσκουν τους σταθμούς αυτούς μη διαθέσιμους. Η διαδικτυακή εφαρμογή που σχεδιάστηκε στο πλαίσιο της διπλωματικής, έχει ως σκοπό να οδηγήσει τον χρήστη στην κατάλληλη επιλογή σταθμού με εύκολο και γρήγορο τρόπο.

Για την υλοποίηση της εφαρμογής αρχικά συλλέγονται τα δεδομένα των σταθμών φόρτισης μέσω των εφαρμογών των παρόχων, ώστε να απεικονιστούν στη συνέχεια στον χάρτη στην διεπαφή. Στη συνέχεια συλλέγονται τα δεδομένα φόρτισης ηλεκτρικών οχημάτων από σταθμούς του εξωτερικού, ώστε να χρησιμοποιηθούν για να παράγουν εικονικά δεδομένα φόρτισης στους σταθμούς της εφαρμογής. Τα δεδομένα αυτά είναι απαραίτητα καθώς δεν υπάρχει αρκετή χρήση των σταθμών της Ελλάδας για να παρθούν ζωντανά πραγματικά δεδομένα. Έτσι με την κατάλληλη επεξεργασία τους δημιουργούνται εικονικές αφίξεις στον κάθε σταθμό φόρτισης, που αντιπροσωπεύουν την μελλοντική αναμενόμενη συμπεριφορά.

Έπειτα υλοποιείται ο αλγόριθμος βελτιστοποίησης, ο οποίος είναι το κύριο συστατικό της εφαρμογής καθώς μέσω των δεδομένων που συλλέχθηκαν και των εισόδων του χρήστη, ο αλγόριθμος έχει την δυνατότητα να προτείνει στον χρήστη τους πιο βολικούς σταθμούς για αυτόν. Ο αλγόριθμος για την απόφασή του λαμβάνει υπόψη κυρίως την τοποθεσία του τελικού προορισμού του οδηγού, την ώρα άφιξης και την πρόβλεψη της διαθεσιμότητας των σταθμών που βρίσκονται σε κοντινή απόσταση.

Στη συνέχεια παρουσιάζεται αναλυτικά η υλοποίηση του συστήματος της εφαρμογής, μέσω της αρχιτεκτονικής που ορίστηκε. Η διεπαφή της εφαρμογής φαίνεται αναλυτικά, και έπειτα παρουσιάζονται τα πιθανά σενάρια χρήσης της.

Τέλος προτείνονται επεκτάσεις που μπορούν να προστεθούν μελλοντικά στην εφαρμογή, οι οποίες δεν μπόρεσαν να πραγματοποιηθούν στα πλαίσια της διπλωματικής εργασίας.

Λέξεις Κλειδιά: Σταθμοί Φόρτισης, Επαύξηση Δεδομένων, Αλγόριθμος Βελτιστοποίησης, Ηλεκτροκίνηση, Διαδικτυακή Εφαρμογή

Abstract

The present thesis aims to develop a web application that represents charging stations in Greece and, through an optimization algorithm, suggests the optimal station to the user. In recent years, there has been a significant increase in use of electric vehicles, making smart scheduling of charging processes essential. Due to the numerous providers of charging stations and the separate applications they offer, users often end up using specific stations rather than the closest ones, leading to the congestion and unavailability of those stations. The web application designed as part of this thesis, aims to guide users to the appropriate station in an easy and fast manner.

To implement the application, data from charging stations are initially collected through the providers' applications, to be subsequently displayed on the map of the interface. Then, charging data from foreign stations are collected to generate virtual charging data for the application's stations. This data is necessary as there is not enough usage of Greek stations to obtain real-time data. Thus, with the appropriate processing, virtual arrivals are created for each charging station, representing the expected future behavior.

Next, the optimization algorithm is implemented, which is the main component of the application. Using the collected data and user inputs, the algorithm can suggest the most convenient stations to the user. The algorithm mainly considers the user's final destination, the estimated time of arrival, and the prediction of station availability in close proximity.

The implementation of the application system is then presented in detail, following the defined architecture. The application's interface is thoroughly demonstrated, followed by the presentation of possible usage scenarios.

Finally, extensions that could be added to the application in the future are proposed, which could not be realized within the scope of this thesis.

Key Words: Charging Stations, Data Augmentation, Optimization Algorithm, Electromobility, Web Application

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Χρυσόστομο Δούκα για την ευκαιρία που μου έδωσε να εκπονήσω την διπλωματική μου εργασία σε ένα τόσο ενδιαφέρον, όσο και χρήσιμο θέμα στον τομέα της ηλεκτρικής φόρτισης. Επίσης θα ήθελα να ευχαριστήσω τους καθηγητές κ. Ιωάννη Ψαρρά και Ευάγγελο Μαρινάκη για την συμμετοχή τους στην επιτροπή εξέτασης της διπλωματικής εργασίας.

Ιδιαίτερες ευχαριστίες θα ήθελα να δώσω στον Βαγγέλη Σπηλιώτη και στον Παναγιώτη Σκαλούμπακα για την συνεχή τους βοήθεια και συνεργασία καθ' όλη την διάρκεια της διπλωματικής. Επίσης θα ήθελα να ευχαριστήσω και την κ. Ιωάννα Μακαρούνη για τις πολύτιμες συμβουλές της. Τέλος η διπλωματική εργασία δεν θα είχε επιτευχθεί χωρίς την στήριξη της οικογένειας και των φίλων μου.

Περιεχόμενα

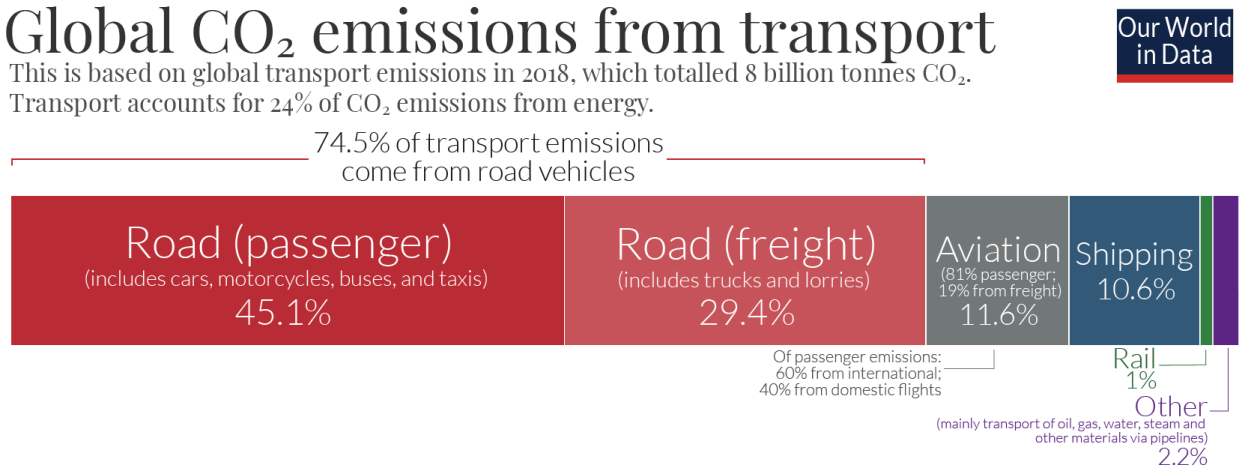
Κατάλογος εικόνων

Κατάλογος πινάκων

Κεφάλαιο 1ο. Εισαγωγή

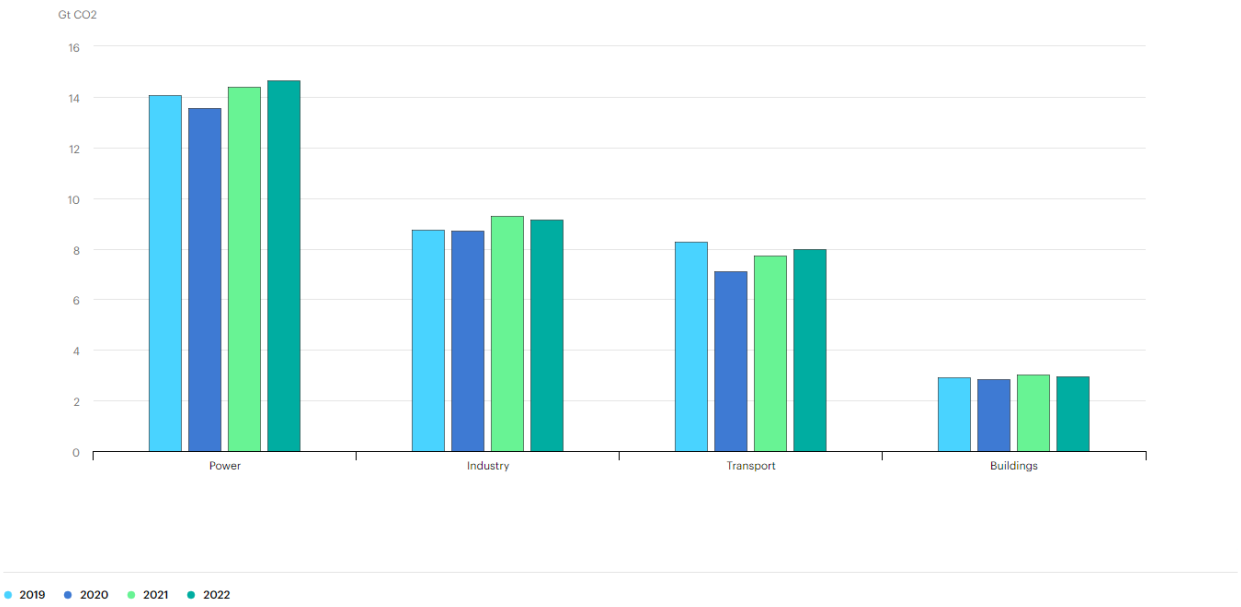
1.1 Εισαγωγή

Η αύξηση των ρύπων τα τελευταία χρόνια αποτελεί μία από τις πιο σοβαρές προκλήσεις που αντιμετωπίζει η σημερινή κοινωνία. Ήδη η συγκέντρωση διοξειδίου του άνθρακα έχει φτάσει τα 419 ppm ανά mole ξηρού αέρα [1] ενώ ο μέσος όρος θερμοκρασίας παγκοσμίως έχει αυξηθεί κατά 0.7°C τα τελευταία 30 χρόνια [2]. Ένας μεγάλος παράγοντας σε αυτό είναι και ο τομέας μεταφοράς, καθώς ευθύνεται για πάνω από 20% των συνολικών εκπομπών διοξειδίου παγκοσμίως [3]. Από αυτό το 74.5% προέρχεται από την οδική μεταφορά και πιο συγκεκριμένα το 45.1% προκαλείται από επιβατικά οχήματα [4].



Εικόνα 1.1.1: Ποσοστά CO2 από διάφορα είδη μεταφορικών μέσων [4]

Παρακάτω αναπαρίστανται γραφικά οι εκπομπές διοξειδίου του άνθρακα, μετρημένες σε gigatons, παγκοσμίως για τα τελευταία τέσσερα έτη. Η χρήση κινητήρων εσωτερικής καύσης όπως φαίνεται έχουν μεγάλη επίδραση στο περιβάλλον καθώς παράγουν περίπου το 20% [3] των συνολικών ρύπων στην ατμόσφαιρα.



Εικόνα 1.1.2: Εκπομπή διοξειδίου του άνθρακα τα τελευταία τέσσερα χρόνια [3]

Εξαιτίας του φαινομένου αυτού γίνονται μεγάλες προσπάθειες παγκοσμίως ώστε να ελαχιστοποιηθεί όσο το δυνατόν περισσότερο η παραγωγή ρύπων από τις οδικές μετακινήσεις. Η επικρατέστερη λύση σε αυτό είναι η χρήση ηλεκτρικών οχημάτων, για την τροφοδοσία των οποίων θα χρησιμοποιούνται κυρίως πηγές ανανεώσιμης ενέργειας. Υπάρχουν διάφοροι τύποι ηλεκτρικών οχημάτων, με τους κυριότερους να είναι: αυτοκίνητα με μπαταρία (Battery Electric Vehicles - BEVs), υβριδικά ηλεκτρικά αυτοκίνητα (Hybrid Electric Vehicles - HEVs) και υβριδικά αυτοκίνητα φόρτισης (Plug-in Hybrid Electric Vehicles - PHEVs).

Τα BEVs είναι τα επικρατέστερα ηλεκτρικά οχήματα στην αγορά παγκοσμίως [5] και λειτουργούν αποκλειστικά με ηλεκτρική ενέργεια που αποθηκεύεται σε μπαταρίες. Τα οχήματα αυτά απαιτούν φόρτιση από εξωτερικούς σταθμούς φόρτισης ή οικιακούς φορτιστές και δεν διαθέτουν κινητήρα εσωτερικής καύσης, με αποτέλεσμα να παράγουν μηδενικές εκπομπές. Τα BEVs είναι πολύ οικολογικά οχήματα όχι μόνο όσον αφορά τις εκπομπές ρύπων, αλλά και λόγω της μειωμένης ηχορύπανσης, καθώς είναι σχεδόν αθόρυβα. Ωστόσο, επειδή λειτουργούν μόνο με ηλεκτροκινητήρα, διαθέτουν περιορισμένη αυτονομία και χρειάζονται περισσότερες φορτίσεις σε μακρινές διαδρομές από τα συμβατικά οχήματα, ενώ παράλληλα ο χρόνος φόρτισης είναι αρκετά μεγάλος.

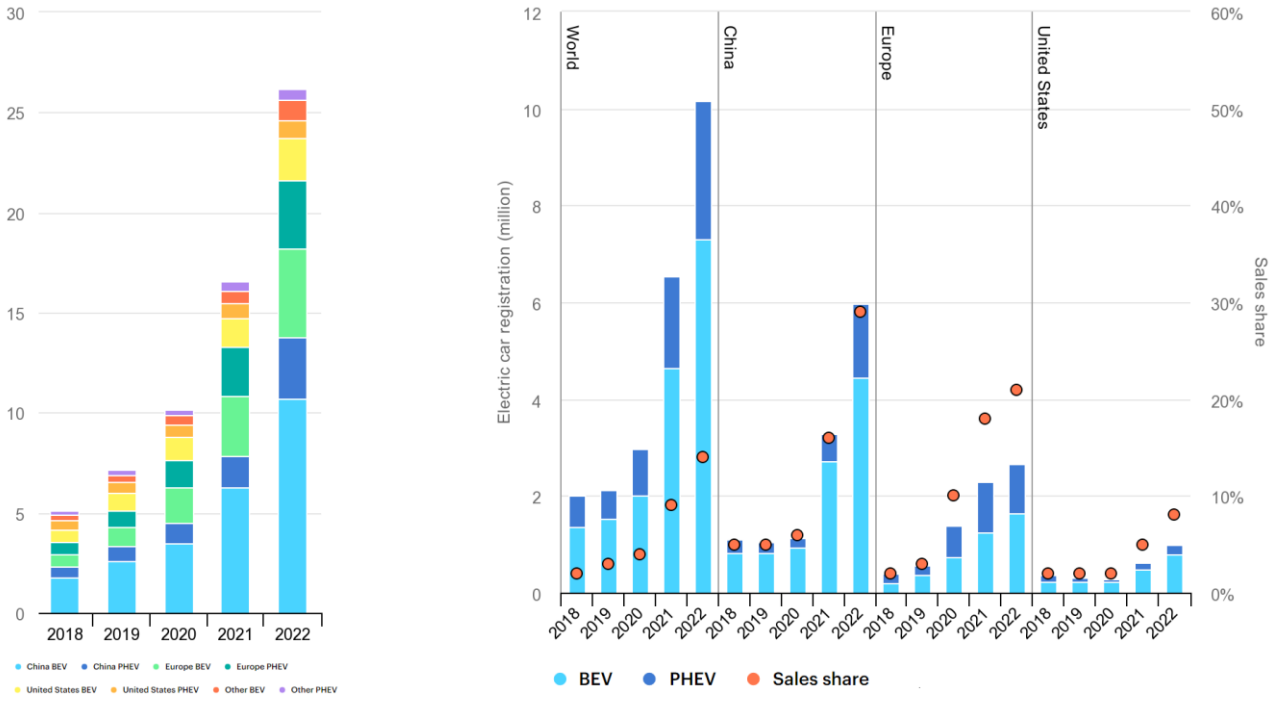
Τα HEV αντίθετα συνδυάζουν ηλεκτρική ενέργεια από μπαταρίες με έναν κινητήρα εσωτερικής καύσης. Ο ηλεκτροκινητήρας βοηθά τον κινητήρα καύσης κατά την επιτάχυνση και την οδήγηση με χαμηλές ταχύτητες, βελτιώνοντας έτσι την αποδοτικότητα χρήσης καυσίμων και κατά συνέπεια μειώνοντας και τους ρύπους. Δεν απαιτούν εξωτερική φόρτιση, καθώς οι μπαταρίες φορτίζονται μέσω της ανάκτησης ενέργειας κατά την πέδηση, και έτσι διαθέτουν μεγαλύτερη αυτονομία από

τα BEVs. Γενικά ωστόσο έχουν υψηλότερο κόστος λόγω της τεχνολογίας και των επιπλέον συστημάτων που χρησιμοποιούν.

Τέλος τα PHEV, που επίσης διαθέτουν σημαντικό μερίδιο στην αγορά, όπως και τα HEV διαθέτουν δύο κινητήρες, έναν ηλεκτρικό και έναν καύσης, και έχουν τη δυνατότητα να φορτίσουν τις μπαταρίες από εξωτερική πηγή. Με τον τρόπο αυτό επιτρέπουν την οδήγηση σε καθαρά ηλεκτρική λειτουργία για μικρές αποστάσεις, ενώ δεν περιορίζουν την αυτονομία του αυτοκινήτου καθώς υπάρχει και ο κινητήρας εσωτερικής καύσης και έτσι όταν η ενέργεια των μπαταριών εξαντλείται, το αυτοκίνητο λειτουργεί ως συμβατικό υβριδικό αυτοκίνητο. Το αυτοκίνητο αυτό συνδυάζει τα πλεονεκτήματα των BEVs και συμβατικών οχημάτων, έχοντας ωστόσο υψηλότερο κόστος.

Η ποικιλία των ηλεκτρικών οχημάτων ανταποκρίνεται σε διάφορες ανάγκες, με αποτέλεσμα η αγορά τους να απευθύνεται σε ένα ευρύτερο κοινό. Τα τελευταία χρόνια μάλιστα, η χρήση τους έχει αρχίσει να μεγαλώνει ολοένα και περισσότερο και ήδη το έτος 2022 τα ηλεκτρικά αυτοκίνητα συνολικά έχουν ξεπεράσει τα 25 εκατομμύρια παγκοσμίως [6].

Στο επόμενο διάγραμμα φαίνεται το πλήθος ηλεκτρικών αυτοκινήτων παγκοσμίως από το 2018 και μετά [6] [5], καθώς και το μερίδιο που έχει το κάθε είδος στην αγορά. Όπως φαίνεται, η αύξηση των συνολικών πωλήσεων είναι αξιοσημείωτη τα τελευταία χρόνια, και ήδη αναμένεται να αυξηθεί κατά 35% παραπάνω έτος 2023 [7].

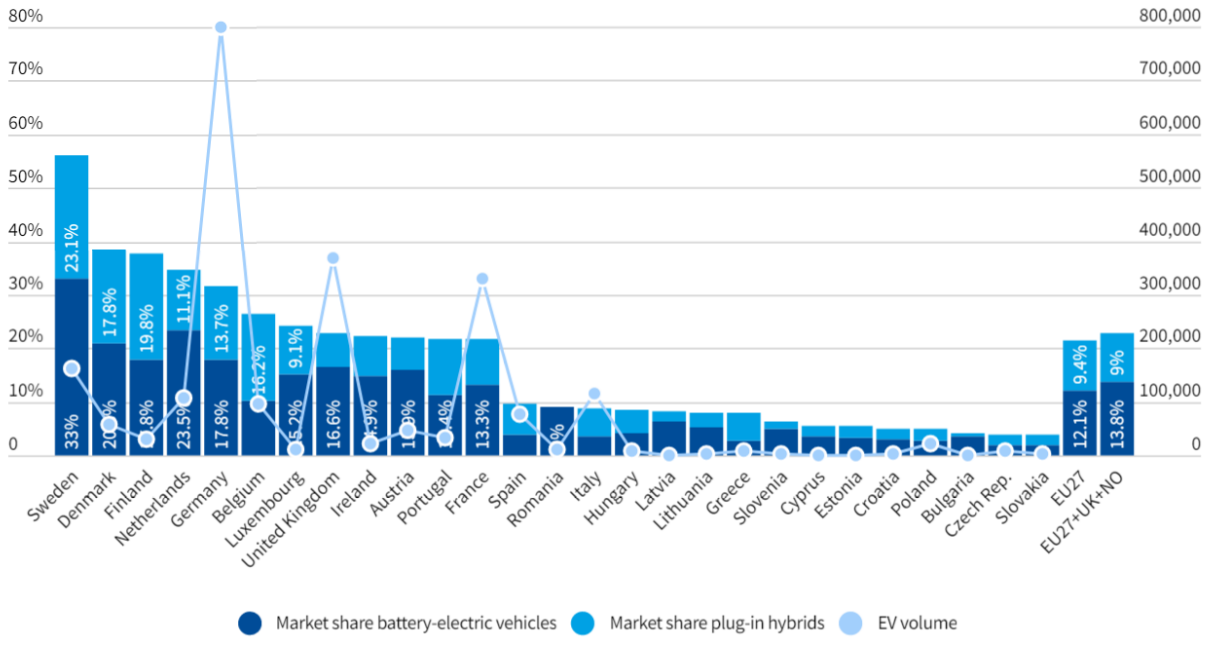


Εικόνα 1.1.3: Αριθμός ηλεκτρικών οχημάτων κατά την περίοδο 2018-2022 [6] [5]

Η Ευρώπη συμμετέχει ενεργά στην προσπάθεια της προώθησης της ηλεκτροκίνησης και ήδη έχει θέσει ως στόχο την ελάττωση των εκπομπών ρύπων από νέα ιδιωτικά οχήματα κατά 55% μέχρι το 2030 και κατά 100% μέχρι το 2035 [8]. Για να μπορέσει να εκπληρωθεί ο στόχος αυτός έχει θέσει αρκετά μέτρα, ώστε να αυξηθεί η χρήση ηλεκτρικών οχημάτων το γρηγορότερο δυνατό. Αρχικά η Ευρωπαϊκή Ένωση έχει θέσει αυστηρούς κανονισμούς για τις εκπομπές CO_2 των αυτοκινήτων που πωλούνται στην ευρωπαϊκή αγορά. Από το 2020, ο μέσος όρος των εκπομπών CO_2 αυτοκινήτων πρέπει να είναι 95 γραμμάρια CO_2 ανά χιλιόμετρο και οι κατασκευαστές που δεν τηρούν το όριο αυτό υφίστανται κυρώσεις. Επίσης η ΕΕ έχει επενδύσει στην ανάπτυξη υποδομών φόρτισης για ηλεκτρικά οχήματα σε ολόκληρη την Ευρώπη, ώστε να προστεθούν 5.700 σταθμοί φόρτισης [9]. Τέλος η ΕΕ παρέχει διάφορα οικονομικά κίνητρα για την αγορά ηλεκτρικών οχημάτων όπως επιδοτήσεις και φοροαπαλλαγές [10] ενώ ταυτόχρονα συνεχίζει να επενδύει σημαντικούς πόρους στην έρευνα και ανάπτυξη νέων τεχνολογιών ηλεκτροκίνησης.

Τα μέτρα αυτά ωστόσο είναι πρόσφατα και ακόμα η πρόοδός είναι σχετικά αργή, καθώς το 2023 στην Ευρωπαϊκή Ένωση μόλις το 2.1% [11] του συνολικού αριθμού αυτοκινήτων είναι ηλεκτρικά. Ωστόσο τα BEV και PHEV αυτοκίνητα πλέον αποτελούν το 12.1% και 9.4% των συνολικών πωλήσεων αντίστοιχα το οποίο υποδεικνύει σημαντική πρόοδο [12].

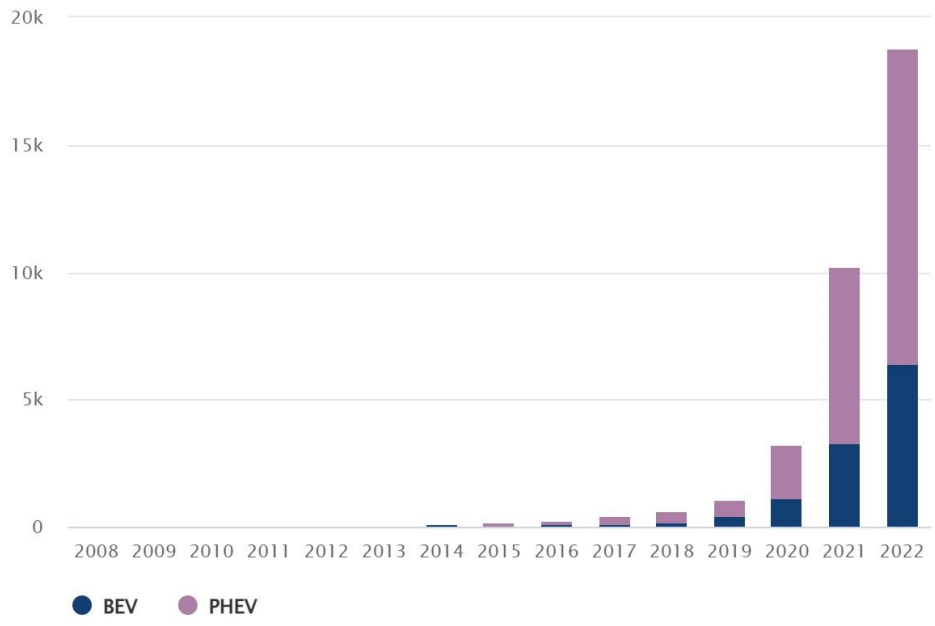
Στη συνέχεια φαίνεται το μερίδιο αγοράς ηλεκτρικών οχημάτων στην Ευρώπη για κάθε χώρα.



Εικόνα 1.1.4: Μερίδιο αγοράς ηλεκτρικών οχημάτων για κάθε χώρα της Ευρωπαϊκής ένωσης [12]

Στην Ελλάδα, ο στόλος ηλεκτροκίνητων οχημάτων ανέρχεται στα 18.717 [13], το οποίο αποτελεί λιγότερο από 1% του συνολικού στόλου οχημάτων. Η χρήση τους είναι ακόμα αρκετά περιορισμένη και πολύ χαμηλή σε σχέση με τα παραδοσιακά οχήματα με κινητήρα εσωτερικής καύσης, όμως οι προσπάθειες για την προώθησή τους είναι υπό εξέλιξη, καθώς ήδη οι αγορές ηλεκτρικών οχημάτων έχουν εξαπλασιαστεί από το 2020 [13].

Παρακάτω φαίνεται ο συνολικός αριθμός των επιβατικών οχημάτων που χρησιμοποιούν εναλλακτικά καύσιμα στην Ελλάδα.

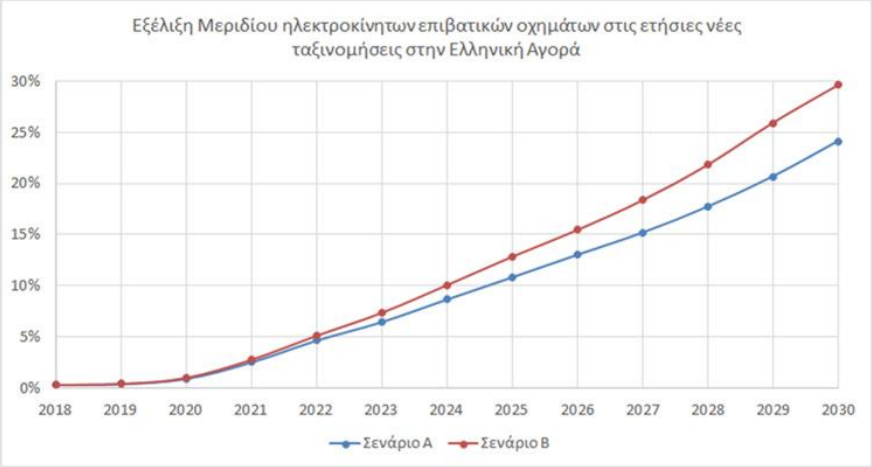


Εικόνα 1.1.5: Επιβατικά ηλεκτρικά οχήματα στην Ελλάδα στο διάστημα 2008-2022 [13]

Για την πρόσφατη αύξηση της χρήσης ηλεκτρικών αυτοκινήτων έχουν συμβάλλει και ορισμένα προγράμματα που επιβλήθηκαν στη χώρα. Ένα από αυτά είναι το κρατικό πρόγραμμα «Κινούμαι Ηλεκτρικά 2», το οποίο έχει ως σκοπό την προώθηση της ηλεκτροκίνησης μέσω επιδοτήσεων αγοράς ηλεκτρικών οχημάτων και φορτιστών. Το πρόγραμμα αυτό καλύπτει ένα μέρος του κόστους του ηλεκτρικού αυτοκινήτου και ταυτόχρονα προσφέρει φορολογικές ελαφρύνσεις για αυτό [14]. Πέρα από την πλευρά του αγοραστή, το πρόγραμμα προσφέρει επιδότηση και σε επιχειρήσεις για την αγορά οχημάτων μαζικά, ενώ και στις δύο περιπτώσεις επιβραβεύει την απόσυρση οχημάτων με χρηματική αποζημίωση. Παράλληλα υπάρχει πρόγραμμα επιδότησης για την ανάπτυξη δημοσίως προσβάσιμων σταθμών φόρτισης με την συνεισφορά της Ευρωπαϊκής Ένωσης, το οποίο στοχεύει στην δημιουργία ενός βασικού δικτύου δημόσιας φόρτισης ηλεκτρικών οχημάτων σε όλη την χώρα, ώστε να υπάρχει η δυνατότητα φόρτισης σε κάθε περιοχή. Με το πρόγραμμα αυτό δίνονται οικονομικά κίνητρα ώστε να τοποθετηθούν νέοι σταθμοί και έτσι

να υπάρχει προσβασιμότητα στη φόρτιση σε όλη την Ελλάδα. Πέρα από την οικονομική συνεισφορά, το πρόγραμμα επίσης προσφέρει ανοιχτούς διαγωνισμούς παραχώρησης δημοσίων χώρων σε κάθε δήμο, για την τοποθέτηση επιπλέον σταθμών. Τα μέτρα αυτά, σε συνδυασμό με την συνεχώς φθίνουσα τιμή των ηλεκτρικών αυτοκινήτων και της συντήρησής τους, προβλέπεται ότι θα προωθήσουν περαιτέρω την αύξηση της χρήσης ηλεκτροκίνητων οχημάτων στην Ελλάδα.

Παρακάτω φαίνεται η προβλεπόμενη συμπεριφορά χρήσης ηλεκτρικών οχημάτων σύμφωνα με το Υπουργείο Περιβάλλοντος και Ενέργειας μέχρι το έτος 2030 [15]. Το σενάριο Α αναφέρεται στην διείσδυση της ηλεκτροκίνησης στην χώρα σύμφωνα με τις τωρινές συνθήκες εξέλιξης της χώρας, ενώ το σενάριο Β προϋποθέτει η Ελλάδα να βρίσκεται σε οικονομική ανάπτυξη και να επιβάλει αυξημένα μέτρα πολιτικής.



Εικόνα 1.1.6: Ποσοστό ηλεκτρικών επιβατικών οχημάτων κατά τα έτη 2018-2030 [15]

Ένα σημαντικό βήμα για να προωθηθούν τα ηλεκτρικά αυτοκίνητα περαιτέρω είναι η εύκολη πρόσβαση σε σταθμούς φόρτισης, ανεξαρτήτως της περιοχής και της ώρας. Ιδιαίτερη ανάγκη εμφανίζεται στα αστικά κέντρα, όπου σε πολλές περιπτώσεις δεν υπάρχει η δυνατότητα φόρτισης του οχήματος με οικιακό φορτιστή, λόγω έλλειψης ιδιωτικής στάθμευσης. Είναι σημαντικό επομένως να υπάρχουν διαθέσιμοι δημόσιοι σταθμοί φόρτισης, σε μία κοντινή εμβέλεια, ώστε να μπορεί να εξυπηρετηθεί κάθε πολίτης όταν το χρειαστεί. Ένας ακόμα λόγος να δοθεί βαρύτητα στο ζήτημα αυτό, είναι η περιορισμένη αυτονομία που διαθέτουν τα ηλεκτρικά οχήματα, το οποίο καθιστά αναγκαίο να υπάρχουν σταθμοί φόρτισης σε κάθε περιοχή της χώρας, ώστε να είναι δυνατή η διάνυση μεγάλων αποστάσεων. Ταυτόχρονα είναι σημαντικό να υπάρχουν και σταθμοί ταχείας φόρτισης, καθώς τα ηλεκτρικά οχήματα χρειάζονται πολύ χρόνο για να φορτίσουν πλήρως.

Όσον αφορά τους σταθμούς φόρτισης, υπάρχουν διάφορα είδη φορτιστών που μπορούν να διαθέτουν. Ο πιο διαδεδομένος τύπος φορτιστή είναι ο τύπου 2 (type 2), ο οποίος χρησιμοποιείται

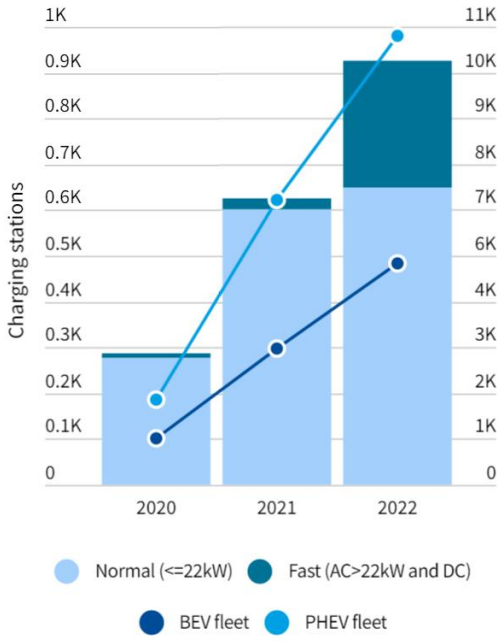
για AC φόρτιση (με εναλλασσόμενο ρεύμα) και παρέχει μέγιστη ισχύ 22kW. Σε σταθμούς όπου προσφέρεται ταχεία φόρτιση, ο φορτιστής αυτός μπορεί να φτάσει έως 43kW [16]. Ο φορτιστής συνδέεται με το αυτοκίνητο μέσω ενός βύσματος Mennekes, το οποίο είναι το πρότυπο φόρτισης για ηλεκτρικά οχήματα στην Ευρώπη.

Ένας ακόμα διαθέσιμος φορτιστής είναι ο τύπου CCS 2 (Combined Charging System 2) ο οποίος παρέχει DC φόρτιση (με συνεχές ρεύμα) και χρησιμοποιείται για γρήγορη φόρτιση. Ο τύπος CCS 2 συνδυάζει ένα βύσμα τύπου 2 για AC φόρτιση και ένα επιπλέον βύσμα για DC γρήγορη φόρτιση. Στην Ελλάδα ο φορτιστής έχει συνήθως ισχύ 50kW ενώ στο εξωτερικό έχει πλέον φτάσει στα 700kW [16] [17].

Τέλος, στην Ελλάδα συναντάται και ο φορτιστής CHAdeMO (CHArge de MOve) ο οποίος επίσης παρέχει DC φόρτιση. Για τον CHAdeMO χρησιμοποιείται ένα ειδικό βύσμα CHAdeMO για τη σύνδεση με το αυτοκίνητο. Αυτό το βύσμα διαθέτει τρεις ακίδες για τη μεταφορά του DC ρεύματος, καθώς και επιπλέον ακίδες για τη μεταφορά δεδομένων και ελέγχου. Παρέχει ταχεία φόρτιση με ισχύ 50kW αν και οι πιο πρόσφατες εκδόσεις μπορούν να φτάσουν έως και 400kW [16] [18].

Τα τελευταία έτη, χάρη στην άνοδο που σημειώθηκε στην αγορά ηλεκτρικών οχημάτων, οι σταθμοί φόρτισης έχουν πλέον αυξηθεί σημαντικά. Έχουν εμφανιστεί αρκετοί πάροχοι, όπου ο καθένας συνεισφέρει στην ποικιλία φορτιστών αλλά και τιμών, ενώ ήδη το 27% των φορτιστών είναι ταχυφορτιστές [12].

Στο παρακάτω διάγραμμα φαίνεται το πλήθος σταθμών φόρτισης [12]. Οι σταθμοί φόρτισης έχουν τριπλασιαστεί σε αριθμό μέσα σε 2 χρόνια με τους ταχυφορτιστές να έχουν μόλις ξεπεράσει το 30%.



Εικόνα 1.1.7: Συνολικό πλήθος σταθμών φόρτισης [12]

1.2 Στόχος Διπλωματικής Εργασίας

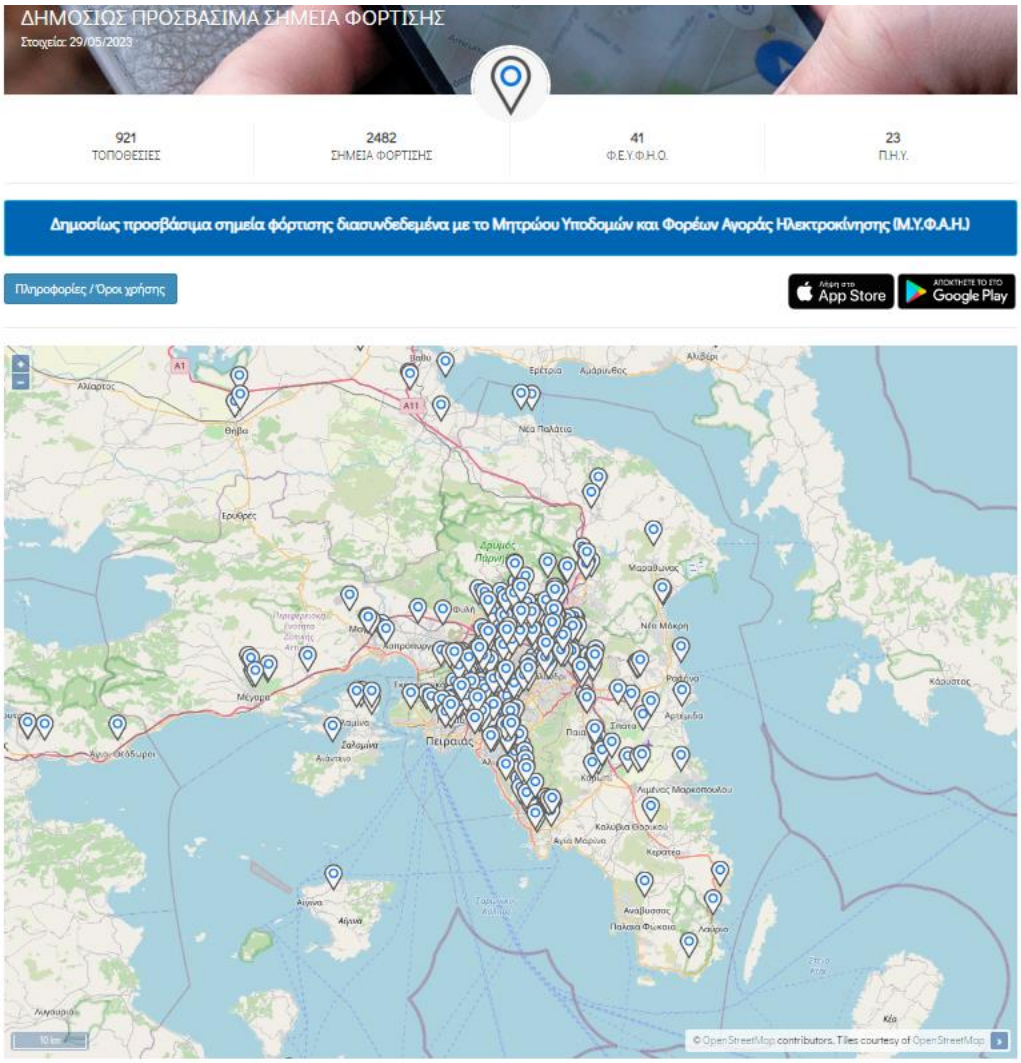
Με την άνοδο της ηλεκτροκίνησης, υπάρχει και αύξηση των σταθμών φόρτισης αλλά και των εταιριών που τους παρέχει. Όπως ήδη έχει αναφερθεί, η Ελλάδα έως το 2022 διέθετε λιγότερο από χίλιους σταθμούς φόρτισης, οι οποίοι ωστόσο ανήκουν σε πολλούς διαφορετικούς παρόχους. Κάθε πάροχος συνήθως διαθέτει δική του ιστοσελίδα ή εφαρμογή, με αποτέλεσμα να υποβαθμίζεται η εμπειρία του χρήστη κατά την αναζήτηση σταθμού φόρτισης. Ο χρήστης αναγκάζεται να ενημερώνεται από πολλές εφαρμογές για τους σταθμούς της περιοχής που επιθυμεί να φορτίσει, το οποίο οδηγεί σε δυσκολία σύγκρισης των σταθμών ως προς τα βασικά χαρακτηριστικά τους, όπως η τιμή, η απόσταση και ο τύπος φορτιστών. Παρακάτω φαίνονται οι μεγαλύτερες εταιρίες παροχής σταθμών φόρτισης στην Ελλάδα με τις τιμές φόρτισης που προσφέρουν και τον αριθμό των σταθμών που έχουν εγκαταστήσει μέχρι τον Μάιο του 2023.

Πίνακας 1.2.1: Αριθμών σταθμών φόρτισης και τιμών χρέωσης για τους μεγάλους παρόχους της Ελλάδας

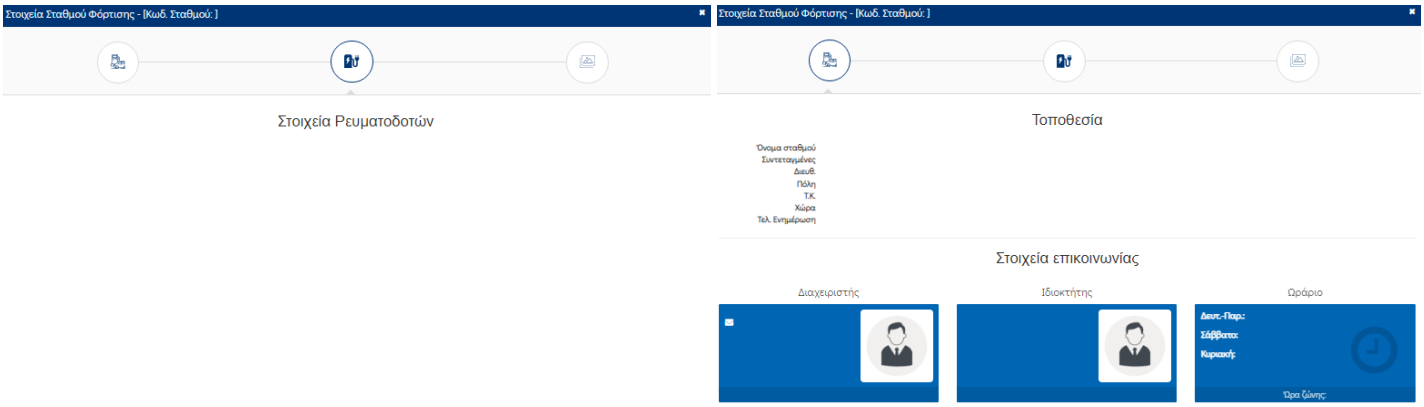
Πάροχοι	**Αριθμός σημείων φόρτισης**	**Χρέωση Φόρτισης**
ChargeSpot [19]	559	0,45-0,60€ / kWh
DEI blue [20]	427	0,50-0,70€ / kWh
Blink Charging [21]	214	0,60€ / kWh
ProtergiaCharge [22]	141	NA
ElpeFuture [23]	62	0,50-0,65€ / kWh
Fortisis [24]	50	NA

Οι διαφορές στις τιμές φόρτισης αλλά και οι διαφορετικές εφαρμογές που πρέπει να εγκαταστήσει ο χρήστης για να μπορεί να χρησιμοποιήσει τον σταθμό ή για να δει πληροφορίες που τον αφορούν, καθιστούν πολύ δύσκολη την διαδικασία της φόρτισης. Ο χρήστης χρειάζεται να επισκέπτεται όλες τις εφαρμογές και να κάνει μόνος του τις συγκρίσεις για να βρει τον κατάλληλο για αυτόν σταθμό.

Προς αντιμετώπιση του προβλήματος αυτού, έχουν γίνει κρατικές προσπάθειες να γίνει η συλλογή των σταθμών φόρτισης στη χώρα σε μία εφαρμογή [25]. Η εφαρμογή δίνει την δυνατότητα στον χρήστη να βλέπει όλους τους διαθέσιμους σταθμούς και τις πληροφορίες τους. Το αποτέλεσμα φαίνεται παρακάτω, ωστόσο είναι προφανές πως βρίσκεται ακόμα σε αρχικά στάδια καθώς δεν διαθέτει όλους τους σταθμούς φόρτισης και σε πολλούς από αυτούς ακόμα δεν υπάρχουν οι βασικές πληροφορίες όπως οι τύποι φορτιστών ή οι τιμές φόρτισης.



Εικόνα 1.2.1: Κρατική εφαρμογή συλλογής σταθμών φόρτισης [25]



Εικόνα 1.2.2: Παράδειγμα έλλειψης στοιχείων από την κρατική εφαρμογή [25]

Δεδομένων των προβλημάτων που αναφέρθηκαν, η παρούσα διπλωματική έχει ως στόχο την δημιουργία μιας διαδικτυακής εφαρμογής, στην οποία θα δίνεται η δυνατότητα στον χρήστη να έχει στη διάθεσή του σταθμούς από πολλούς παρόχους και με την χρήση κατάλληλου αλγορίθμου να του προτείνονται οι καταλληλότεροι για αυτόν σταθμοί. Η εφαρμογή αυτή θα προσφέρει στον χρήστη βασικές πληροφορίες για όλους τους σταθμούς που έχουν συλλεχθεί, ενώ παράλληλα με την επιλογή κατάλληλων φίλτρων ο χρήστης θα μπορεί να βρίσκει τους κοντινότερους σε αυτόν σταθμούς οι οποίοι είναι διαθέσιμοι. Αυτό που διαφοροποιεί ωστόσο την εφαρμογή είναι ο αλγόριθμος συστάσεων που διαθέτει, για να προτείνει στον χρήστη σταθμούς φόρτισης. Με την ραγδαία αύξηση των σταθμών φόρτισης, είναι ευκολότερο για τον χρήστη να έχει στη διάθεσή του ένα αυτόματο σύστημα συστάσεων, από την χειροκίνητη σύγκριση και επιλογή σταθμών. Ως προτεραιότητα για τον αλγόριθμο τέθηκε η απόσταση και η διαθεσιμότητα των σταθμών. Η επιλογή αυτή στηρίχθηκε στο γεγονός ότι η απόσταση είναι ιδιαίτερα σημαντική όταν ο χρήστης αφήνει το όχημά του για αρκετή ώρα και δεν έχει μέσο μεταφοράς στο διάστημα αυτό. Ιδιαίτερα στα αστικά κέντρα όπου η δυνατότητα οικιακής φόρτισης είναι περιορισμένη και η ανάγκη για φόρτιση σε δημόσιο σταθμό είναι πιο συχνή, είναι σημαντικό ο χρήστης να βρίσκει έναν κοντινό σταθμό. Ταυτόχρονα, στον αλγόριθμο δίνεται ίση βαρύτητα και στην διαθεσιμότητα των σταθμών, ώστε οι χρήστες να βρίσκουν χώρο στους σταθμούς φόρτισης που τους προτάθηκαν και ταυτόχρονα να μην δημιουργείται συνωστισμός σε ορισμένους μόνο σταθμούς.

Το πρώτο ζήτημα της εφαρμογής είναι να είναι συγκεντρωμένες οι πληροφορίες των σταθμών, ώστε ο χρήστης να έχει πρόσβαση σε αυτές ανεξαρτήτως του παρόχου. Για κάθε σταθμό θα υπάρχουν διαθέσιμα η τιμή του, οι τύποι φορτιστών, η θέση του στον χάρτη και η διαθεσιμότητα του σταθμού κάθε ώρα. Με τις πληροφορίες αυτές ο χρήστης μπορεί να συγκρίνει και να αποφασίσει τον σταθμό που θέλει να επισκεφτεί με τα δικά του κριτήρια.

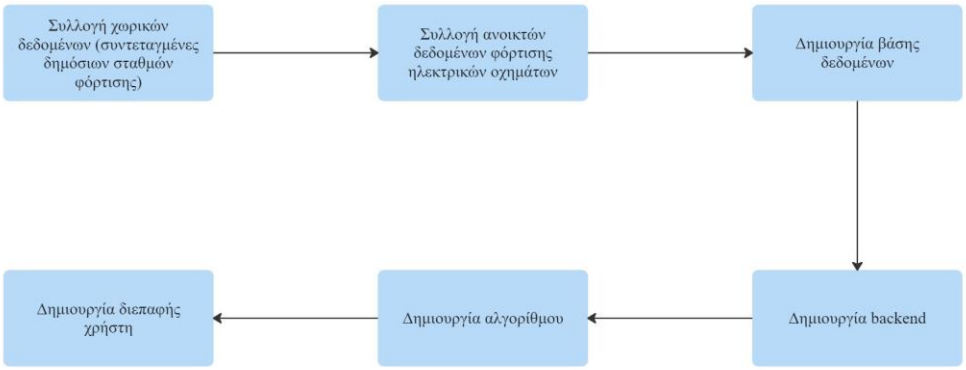
Το δεύτερο ζήτημα είναι ο αλγόριθμος συστάσεων, που θα προτείνει σταθμούς φόρτισης στον χρήστη ανάλογα με τα φίλτρα που έχει επιλέξει. Τα φίλτρα αρχικά περιλαμβάνουν την περιοχή και την ακτίνα στην οποία ο χρήστης ενδιαφέρεται να φορτίσει το αυτοκίνητό του, ώστε να προτείνει μόνο κοντινούς σταθμούς. Έπειτα χρειάζεται να επιλεχθεί η ώρα επίσκεψης του σταθμού ώστε να επιλεχθούν μόνο όσοι σταθμοί είναι διαθέσιμοι την ώρα εκείνη. Ο χρήστης μπορεί να επιλέξει και τις ώρες που θα παραμείνει στον σταθμό, ώστε αν θέλει στη συνέχεια να κάνει κράτηση και να αποθηκευτεί η επίσκεψή του ώστε να ληφθεί υπόψη για τους επόμενους ενδιαφερόμενους. Τέλος, ο χρήστης αν θέλει μπορεί να επιλέξει τον τύπο φορτιστή που θέλει να χρησιμοποιήσει. Ο αλγόριθμος στη συνέχεια προτείνει στον χρήστη τους κοντινότερους σταθμούς, ταξινομημένους ως προς το πόσο κατάλληλοι είναι. Για κάθε σταθμό φαίνεται η τιμή φόρτισης και η απόσταση από το σημείο αναφοράς. Ο χρήστης έτσι μπορεί να επιλέξει από την λίστα αυτόν που θεωρεί πιο κατάλληλο και αν θέλει να κάνει κράτηση ώστε να αποθηκευτεί η επίσκεψή του.

Με την εφαρμογή αυτή επομένως, σκοπός είναι να βελτιωθεί η συνολική εμπειρία του χρήστη και να προωθηθεί η χρήση ηλεκτρικών οχημάτων, ενώ ταυτόχρονα να επωφελούνται όλοι οι σταθμοί φόρτισης χωρίς να δημιουργούνται συνωστισμοί.

1.3 Δομή Διπλωματικής Εργασίας

Η υλοποίηση της εφαρμογής βασίστηκε σε μια μελλοντική κατάσταση στην οποία η ηλεκτροκίνηση θα έχει αυξηθεί σε βαθμό, τέτοιο ώστε να είναι χρήσιμος και απαραίτητος ένας αλγόριθμος συστάσεων. Στην πραγματικότητα η Ελλάδα ακόμα βρίσκεται σε αρχικά στάδια όσον αφορά την ηλεκτροκίνηση, με αποτέλεσμα οι σταθμοί φόρτισης ακόμα να είναι λίγοι και με ελάχιστη χρήση. Έτσι, με τα σημερινά δεδομένα ο αλγόριθμος συστάσεων θα μπορούσε να υποθέσει πώς οι σταθμοί φόρτισης είναι πάντα διαθέσιμοι. Ωστόσο η εφαρμογή αυτή καλείται να λάβει υπόψη την κατάσταση της χώρας μελλοντικά, όπου η ηλεκτροκίνηση θα είναι πλέον ευρέως διαδεδομένη και θα είναι απαραίτητο να συμπεριληφθεί στον αλγόριθμο ο συνωστισμός που θα προκαλείται, ιδιαίτερα στα αστικά κέντρα.

Παρακάτω φαίνεται το διάγραμμα ροής της διπλωματικής εργασίας με τα βήματα που ακολουθήθηκαν και στη συνέχεια ακολουθεί η επεξήγησή τους.



Εικόνα 1.3.1: Διάγραμμα ροής διπλωματικής εργασίας

Το πιο σημαντικό βήμα για την υλοποίηση της εφαρμογής είναι η συγκέντρωση των σταθμών φόρτισης και των πληροφοριών τους. Για κάθε σταθμό είναι απαραίτητες οι χωρικές του συντεταγμένες ώστε να απεικονιστεί στον χάρτη, αλλά και επιπλέον πληροφορίες όπως οι τύποι φορτιστών που διαθέτει, η τιμές φόρτισής και ο πάροχός του. Έτσι, αρχικά έγινε συλλογή και η επεξεργασία των σταθμών, μέσω του scrapping των εφαρμογών του κάθε παρόχου. Για τους σταθμούς έγιναν scraped οι εξής ιστοσελίδες: Chargespot, Fortisis, ProtergiaCharge, PlugShare, ElpeFuture και Blink Charging.

Επόμενο βήμα είναι η συλλογή δεδομένων όσον αφορά τις αφίξεις που γίνονται σε κάθε σταθμό. Μία σημαντική ιδιότητα της εφαρμογής είναι η δυνατότητα προβολής της διαθεσιμότητας των

σταθμών ανά ώρα, στην οποία στηρίζεται και ο αλγόριθμος συστάσεων. Όπως προαναφέρθηκε όμως, στην Ελλάδα οι σταθμοί ακόμα έχουν ελάχιστες αφίξεις, με αποτέλεσμα το φαινόμενο αυτό να οδηγεί στην έλλειψη δεδομένων. Εφόσον δεν είναι δυνατή η ζωντανή λήψη δεδομένων από τους σταθμούς των παρόχων, αποφασίστηκε πως η καλύτερη λύση είναι η δημιουργία ενός προσεγγιστικού προφίλ για κάθε σταθμό, στο οποίο θα φαίνεται η συνήθης συμπεριφορά αφίξεων. Το προφίλ αυτό πρέπει να αντιπροσωπεύει μια ρεαλιστική συμπεριφορά, η οποία εξαρτάται από την περιοχή και το μέγεθος του σταθμού, και έτσι για την δημιουργία του χρησιμοποιήθηκαν κατάλληλα δεδομένα από σταθμούς εκτός Ελλάδας με μεγαλύτερη επισκεψιμότητα,. Τα δεδομένα αυτά αν και αποτελούνται από περισσότερες αφίξεις ανά σταθμό, χρειάζονται να επεξεργαστούν κατάλληλα ώστε να προκύψει ένα διαφορετικό προφίλ για κάθε σταθμό, το οποίο να αντιπροσωπεύει την συμπεριφορά της περιοχής στην οποία βρίσκεται. Για την δημιουργία των προφίλ επομένως επιλέχθηκαν δεδομένα από σταθμούς φόρτισης του εξωτερικού τα οποία παρουσιάζονται αναλυτικότερα στον παρακάτω πίνακα [26] [27] [28]:

Πίνακας 2.3.1: Δεδομένα σταθμών φόρτισης που επιλέχθηκαν για την δημιουργία προφίλ

Δεδομένα	Τοποθεσία σταθμών	Αριθμός δεδομένων	Χρονικό διάστημα
Α	Λονδρέζικο Προάστιο του Μπαρνέτ	5500	31/03/2021-31/03/2022
Β	Μπόλντερ, Κολοράντο	43660	20/01/2018-31/07/2022
Γ	Λονδρέζικο Προάστιο του Μπαρνέτ	2400	01/01/2020-15/06/2021

Έπειτα δημιουργήθηκε η βάση δεδομένων και το μοντέλο της και έπειτα έγινε η τοποθέτηση των δεδομένων που συλλέχθηκαν σε αυτή. Το μοντέλο της βάσης είναι απλό καθώς οι σταθμοί είναι λίγοι και δεν χρειάζονται πολλές πληροφορίες για αυτούς.

Στη συνέχεια δημιουργήθηκαν το backend ώστε να μπορεί να υποστηρίξει τις λειτουργίες της εφαρμογής και να διασφαλίσει την σωστή επικοινωνία με την βάση δεδομένων. Στο backend έχει υλοποιηθεί και ο αλγόριθμος συστάσεων, ο οποίος με τα φίλτρα που έχουν δοθεί από τον χρήστη, βρίσκει και επιστρέφει τους κατάλληλους σταθμούς.

Τέλος δημιουργήθηκε η διεπαφή χρήστη, στην οποία ο χρήστης μπορεί να δει και να κάνει κράτηση στους σταθμούς. Στο τμήμα αυτό απεικονίζονται τα δεδομένα και οι λειτουργίες των προηγούμενων σταδίων, με τα βασικότερα να είναι ο χάρτης που περιέχει τους σταθμούς φόρτισης και τις αντίστοιχες πληροφορίες τους και τα φίλτρα με τα οποία στη συνέχεια μπορεί να χρησιμοποιηθεί ο αλγόριθμος συστάσεων.

1.4 Βιβλιογραφική Ανασκόπηση

Οι αλγόριθμοι βελτιστοποίησης έχουν κρίσιμο ρόλο στον τομέα της φόρτισης ηλεκτρικών οχημάτων. Οι αλγόριθμοι αυτοί σχεδιάζονται για να διαχειρίζονται αποδοτικά τη διαδικασία φόρτισης, λαμβάνοντας υπόψη παράγοντες όπως η διαθεσιμότητα ενέργειας, η χωρητικότητα των σταθμών φόρτισης, οι προτιμήσεις των χρηστών και οι περιορισμοί του δικτύου. Στόχος τους είναι η μέγιστη αξιοποίηση της υποδομής φόρτισης, η ελαχιστοποίηση των φορτίων φόρτισης και η διασφάλιση αξιόπιστων και βολικών υπηρεσιών φόρτισης για τους παρόχους σταθμών φόρτισης.

Ένα σημαντικό τμήμα της σχεδίασης ενός αλγορίθμου βελτιστοποίησης είναι τα κριτήριά του και τι στοχεύει να βελτιστοποιήσει. Στον τομέα της ηλεκτρικής φόρτισης έχουν ήδη γίνει σχετικές μελέτες από διαφορετικές οπτικές γωνίες. Μία αξιοσημείωτη μελέτη έχει γίνει από το Πολιτειακό Πανεπιστήμιο του Οχάιο και έχει ως σκοπό την μελέτη της σχέσης της τοποθεσίας των σταθμών φόρτισης με την μέγιστη χρήση τους [29].

Το σχετικό άρθρο ερευνά πιθανές τοποθεσίες στις οποίες μπορούν να τοποθετηθούν σταθμοί φόρτισης και προσομοιώνει τις επισκέψεις και τις ώρες που απασχολούνται οι σταθμοί. Οι τοποθεσίες που ερευνά ανήκουν σε τρείς διακριτές κατηγορίες: πανεπιστήμια, εμπορικές περιοχές και εργασιακούς χώρους. Οι περιοχές αυτές επιλέχθηκαν να μελετηθούν καθώς διαφέρουν μεταξύ τους ως προς την συμπεριφορά των αφίξεων και της διάρκειας της διαμονής σε αυτές. Για παράδειγμα μία εμπορική περιοχή χαρακτηρίζεται από πολλές αφίξεις και μικρότερη διάρκεια παραμονής στον χώρο, σε αντίθεση με έναν πανεπιστημιακό χώρο, όπου οι αφίξεις είναι λιγότερες αλλά οι ώρες παραμονής είναι μεγάλες και έχουν μέση διάρκεια 8-9 ώρες. Με τα δεδομένα αυτά ο αλγόριθμος έχει ως σκοπό να προσομοιώσει και να βρει τον αριθμό αλλά και τον τύπο φορτιστών που θα τοποθετηθούν σε κάθε περιοχή. Οι τύποι φορτιστών είναι δύο, τύπου 1, ο οποίος είναι οικονομικός αλλά χρειάζεται πολλές ώρες για να επιτύχει πλήρη φόρτιση της μπαταρίας, και τύπου 2, οποίος είναι ακριβός αλλά προσφέρει ταχεία φόρτιση. Η προσομοίωση έπειτα γίνεται με δύο διαφορετικούς προϋπολογισμούς, των ενός και τριών εκατομμυρίων δολαρίων.

Τα αποτελέσματα στη συνέχεια δείχνουν πόσοι σταθμοί τύπου 1 και 2 θα τοποθετηθούν σε κάθε περιοχή ανάλογα με τον διαθέσιμο προϋπολογισμό. Στην περίπτωση των 3 εκατομμυρίων οι ταχυφορτιστές που τοποθετούνται είναι πολύ περισσότεροι στις εργασιακές και εμπορικές περιοχές, καθώς με τον τρόπο αυτό επιτυγχάνεται μεγαλύτερη φόρτιση της μπαταρίας σε περιορισμένο χρονικό διάστημα, το οποίο είναι ιδανικό όταν υπάρχουν πολλές επισκέψεις περιορισμένης διάρκειας.

Charging stations and chargers built at location types under energy-maximization optimization criterion.

	Charging stations				Chargers			
	$1 million		$3 million		$1 million		$3 million	
	Level-1	Level-2	Level-1	Level-2	Level-1	Level-2	Level-1	Level-2
Working	172	9	82	193	778	13	279	912
University	12	0	3	22	55	0	9	72
Shopping	0	1	25	109	0	1	25	128

Charging stations and chargers built at location types under EV-maximization optimization criterion.

	Charging stations		Chargers	
	$1 million	$3 million	$1 million	$3 million
Working	136	275	731	1198
University	19	25	74	81
Shopping	48	145	45	165

Note: All are level-one.

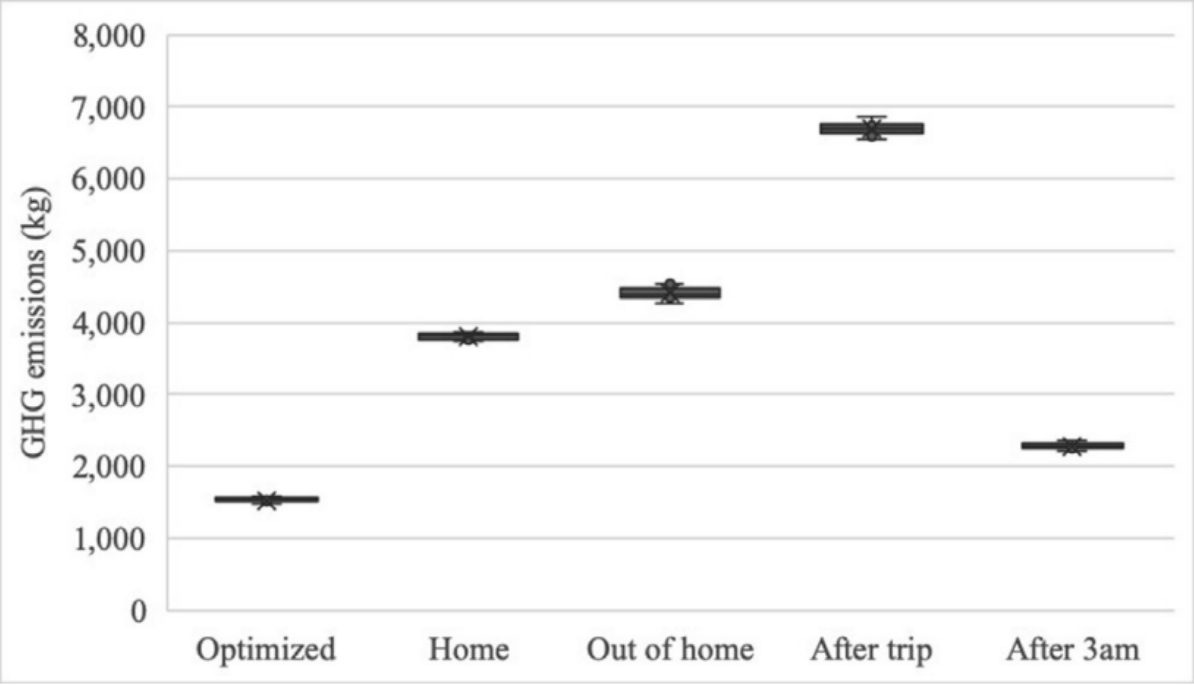
Εικόνα 1.4.1: Αποτελέσματα της μελέτης τοποθέτησης σταθμών [29]

Με την έρευνα αυτή επιτυγχάνεται η βελτιστοποίηση κυρίως από την πλευρά του επενδυτή. Με τις προσομοιώσεις αυτές είναι δυνατό να καθοριστεί η βέλτιστη τοποθέτηση και η μέγιστη απόδοση των σταθμών φόρτισης δεδομένου του οικονομικού προϋπολογισμού. Το αποτέλεσμα επομένως βοηθάει στην σωστή επένδυση στους σταθμούς φόρτισης ώστε να μεγιστοποιηθεί η απόδοση του κάθε σταθμού και κατ' επέκταση το κέρδος.

Μια διαφορετική οπτική φαίνεται στην μελέτη του Πανεπιστημίου του Τορόντο, του οποίου η έρευνα βασίζεται στην βελτιστοποίηση των ωραρίων φόρτισης των αυτοκινήτων με στόχο την ελαχιστοποίηση των εκπομπών αερίων του θερμοκηπίου [30]. Αν και τα ηλεκτρικά αυτοκίνητα παράγουν μηδενικές εκπομπές, η παραγωγή ηλεκτρικού ρεύματος για την φόρτισή τους επιβαρύνει το περιβάλλον. Ιδιαίτερα σε περιοχές όπου η ηλεκτρική ενέργεια παράγεται κυρίως από καύση ορυκτών, οι εκπομπές αερίων μπορεί να είναι υψηλότερες κατά τις περιόδους αυξημένης ζήτησης, όταν ενεργοποιούνται επιπλέον μονάδες παραγωγής ηλεκτρικής ενέργειας για να καλύψουν την αυξημένη ζήτηση. Για τον λόγο αυτό η μελέτη αυτή αποσκοπεί στον σχεδιασμό ενός ευριστικού αλγορίθμου για τη βελτιστοποίηση των χρονοπρογραμμάτων φόρτισης, με στόχο την ελαχιστοποίηση των εκπομπών αερίων του θερμοκηπίου από την παραγωγή ηλεκτρικής ενέργειας.

Ο αλγόριθμος βρίσκει την βέλτιστη χωροχρονική κατανομή των δραστηριοτήτων φόρτισης, η οποία μπορεί να χρησιμοποιηθεί για την ανάθεση θέσεων φόρτισης και έτσι να οδηγήσει σε πιο οικολογικές συμπεριφορές φόρτισης. Ο αλγόριθμος μπορεί επίσης να λάβει υπόψη περιορισμούς

όπως η μέγιστη διαθέσιμη ηλεκτρική ενέργεια, η ηλεκτρική χωρητικότητα του δικτύου και τα χαρακτηριστικά της περιοχής όπως ο πληθυσμός, τοποθεσία σταθμών φόρτισης κ.α. και έτσι μπορεί να λειτουργήσει και για περιοχές πέρα από αυτήν που μελετήθηκε. Στη συνέχεια φαίνονται τα αποτελέσματα της μελέτης και οι εκπομπές αέριων ρύπων σε κάθε περίπτωση. Ο αλγόριθμος βελτιστοποίησης συγκρίθηκε με τέσσερις άλλες περιπτώσεις οι οποίες περιλαμβάνουν με την σειρά που φαίνονται: την φόρτιση μόνο στο σπίτι, την φόρτιση μόνο σε σταθμούς φόρτισης, της φόρτιση μετά από την χρήση του αυτοκινήτου και την φόρτιση μόνο μετά τις 3 π.μ.



Εικόνα 2.4.2: Αποτελέσματα ρύπων ανά περίπτωση της έρευνας δημιουργίας προγράμματος φόρτισης [30]

Με τον αλγόριθμο αυτό τίθεται ως προτεραιότητα η μείωση των περιβαλλοντικών ρύπων, έναντι της οικονομικής μεγιστοποίησης από την πλευρά του επενδυτή. Πέρα από τις δύο μελέτες που αναφέρθηκαν, έχουν δημιουργηθεί αλγόριθμοι βελτιστοποίησης της φόρτισης για να εξυπηρετηθούν πολλοί διαφορετικοί σκοποί. Ο Andrenacci et al [31] κατένειμαν σταθμούς φόρτισης σε υποπεριοχές στη Ρώμη και έτσι τα ταξίδια ομαδοποιήθηκαν ανά ζώνες προορισμού και οι εκτιμήσεις των απαιτήσεων ηλεκτρικής ενέργειας έγιναν με βάση την ενεργειακή κατανάλωση από ένα κινηματικό μοντέλο. Ο Tu et al [32] σχεδίασαν έναν αλγόριθμο βελτιστοποίησης για την ανάθεση των σταθμών φόρτισης στη Σενζέν, προκειμένου να μεγιστοποιήσουν την κάλυψη της υπηρεσίας ηλεκτρικού ταξί και το επίπεδο της υπηρεσίας φόρτισης. Ο Alhazmi et al [33] βελτίωσαν την προσβασιμότητα στους σταθμούς φόρτισης μέσω

της βελτιστοποίησης των τοποθεσιών φόρτισης, με βάση την απόσταση του ταξιδιού και την αβεβαιότητα της ηλεκτρικής αυτονομίας. Πέρα από την ατομική αυτονομία των οχημάτων, ο He et al [34] μεγιστοποίησαν τη συνολική ροή κυκλοφορίας στο οδικό δίκτυο μέσω της βελτιστοποίησης των τοποθεσιών φόρτισης.

Ανάλογα με τον τελικό σκοπό, υπάρχουν πολλές βελτιστοποιήσεις που μπορούν να γίνουν στην διαδικασία φόρτισης. Στην διπλωματική εργασία η βελτιστοποίηση γίνεται αρχικά προς όφελος του χρήστη και κατ' επέκταση προς όφελος των παρόχων των σταθμών. Ο αλγόριθμος έχει ως κύριο στόχο την εύρεση του σταθμού που είναι κοντά στον χρήστη και ταυτόχρονα έχει μεγάλη διαθεσιμότητα ώστε να μειωθεί η πιθανότητα να είναι γεμάτος την ώρα επίσκεψης του χρήστη. Η απόσταση είναι ένα ιδιαίτερα σημαντικό κριτήριο στην επιλογή σταθμού όταν δεν υπάρχει διαθέσιμος σταθμός ταχείας φόρτισης και χρειάζεται να αφήσει το όχημά του για αρκετές ώρες ο χρήστης. Παράλληλα, το πρόβλημα του συνωστισμού σε ορισμένους μόνο σταθμούς, υποβαθμίζει την εμπειρία του χρήστη, καθώς χρειάζεται να περιμένει ή να επισκεφτεί άλλον σταθμό, αλλά και των παρόχων, καθώς δεν υπάρχει ισοκατανομή του φόρτου επισκέψεων με αποτέλεσμα να μην επωφελούνται όλοι οι σταθμοί. Έτσι ο αλγόριθμος καλείται να λύσει αυτό το πρόβλημα και να καθοδηγήσει του χρήστες της εφαρμογής να χρησιμοποιούν τους κατάλληλους σταθμούς. Επιπρόσθετα, μέσω της δυνατότητας κράτησης της εφαρμογής, ο αλγόριθμος προσαρμόζεται στις ενδεχόμενες επισκέψεις στους σταθμούς και αποτρέπει τον συνωστισμό.

Επομένως, μέσω της διπλωματικής εργασίας στόχος είναι να βελτιωθεί η εμπειρία του χρήστη και να παροτρυνθούν και νέοι χρήστες να μεταβούν στην ηλεκτροκίνηση, καθώς θα έχουν τα απαραίτητα εργαλεία στη διάθεσή τους για να τους βοηθήσουν να βρίσκουν διαθέσιμους σταθμούς όπου και αν βρίσκονται.

Κεφάλαιο 2ο. Αρχιτεκτονική και εργαλεία συστήματος

2.1 Αρχιτεκτονική συστήματος

Η επιλογή της αρχιτεκτονικής του συστήματος είναι ένας σημαντικός και αποφασιστικός παράγοντας κατά τη διαδικασία ανάπτυξης λογισμικού. Η αρχιτεκτονική καθορίζει τον τρόπο με τον οποίο η εφαρμογή θα οργανωθεί, θα διαχειριστεί τα δεδομένα, θα αλληλεπιδρά με τους χρήστες και θα εξυπηρετεί τις λειτουργικές απαιτήσεις και επομένως κατά την επιλογή της, πρέπει να ληφθούν υπόψη πολλοί παράγοντες. Ένας από τους πιο σημαντικούς είναι η λειτουργικότητα της εφαρμογής και οι απαιτήσεις της. Ανάλογα με τον τύπο της εφαρμογής, μπορεί να απαιτείται μια αρχιτεκτονική που επιτρέπει την αποτελεσματική διαχείριση των δεδομένων, την ανταλλαγή πληροφοριών μεταξύ συστημάτων, την ανταπόκριση σε υψηλό φορτίο εργασίας ή την επεκτασιμότητα για μελλοντικές ανάγκες. Επιπλέον, παράγοντες όπως η απόδοση, η ασφάλεια, η ευελιξία, η συντηρησιμότητα και η διαθεσιμότητα πρέπει επίσης να ληφθούν υπόψη. Για παράδειγμα, αν η εφαρμογή απαιτεί υψηλή απόδοση, μπορεί να είναι απαραίτητη η χρήση μιας αρχιτεκτονικής που επιτρέπει την παράλληλη εκτέλεση ή τη χρήση κατανεμημένων συστημάτων. Η επιλογή της αρχιτεκτονικής συστήματος πρέπει επίσης να λαμβάνει υπόψη τους περιορισμούς και τους πόρους που είναι διαθέσιμοι, όπως οι οικονομικοί πόροι, ο χρόνος ανάπτυξης και η εμπειρογνωμοσύνη της ομάδας ανάπτυξης. Τελικά, η επιλογή αυτή είναι μια πολυσύνθετη διαδικασία που απαιτεί αξιολόγηση πολλών παραγόντων και την εύλογη εξισορρόπηση μεταξύ των ανταγωνιστικών απαιτήσεων της εφαρμογής. Οι αποφάσεις που λαμβάνονται σε αυτήν τη φάση έχουν σημαντικές επιπτώσεις στην απόδοση, την ανάπτυξη και τη μελλοντική επέκταση του συστήματος.

Υπάρχουν πολλές αρχιτεκτονικές συστήματος που μπορούν να χρησιμοποιηθούν στην ανάπτυξη λογισμικού και ορισμένα παραδείγματα περιλαμβάνουν:

- Αρχιτεκτονική επιπέδων (Layered Architecture): Η εφαρμογή οργανώνεται σε διάφορα επίπεδα, όπως το επίπεδο της παρουσίασης, της λογικής και των δεδομένων. Κάθε επίπεδο αναλαμβάνει συγκεκριμένες λειτουργίες, με την επικοινωνία να γίνεται μεταξύ των γειτονικών επιπέδων.
- Μικροσυστήματα (Microservices): Η εφαρμογή αναπτύσσεται ως σύνολο ανεξάρτητων μικρών υπηρεσιών που λειτουργούν μαζί για να παρέχουν την συνολική λειτουργικότητα. Κάθε μικροϋπηρεσία είναι αυτόνομη και μπορεί να αναπτυχθεί, να κατανεμηθεί και να λειτουργήσει ανεξάρτητα.
- Αρχιτεκτονική MVC (Model-View-Controller): Η εφαρμογή διαιρείται σε τρία κύρια συστατικά. Το μοντέλο (Model) αναλαμβάνει τη διαχείριση των δεδομένων, η προβολή (View) αναλαμβάνει την παρουσίαση των δεδομένων στους χρήστες, και ο ελεγκτής (Controller) διαχειρίζεται τις αλληλεπιδράσεις μεταξύ του μοντέλου και της προβολής.

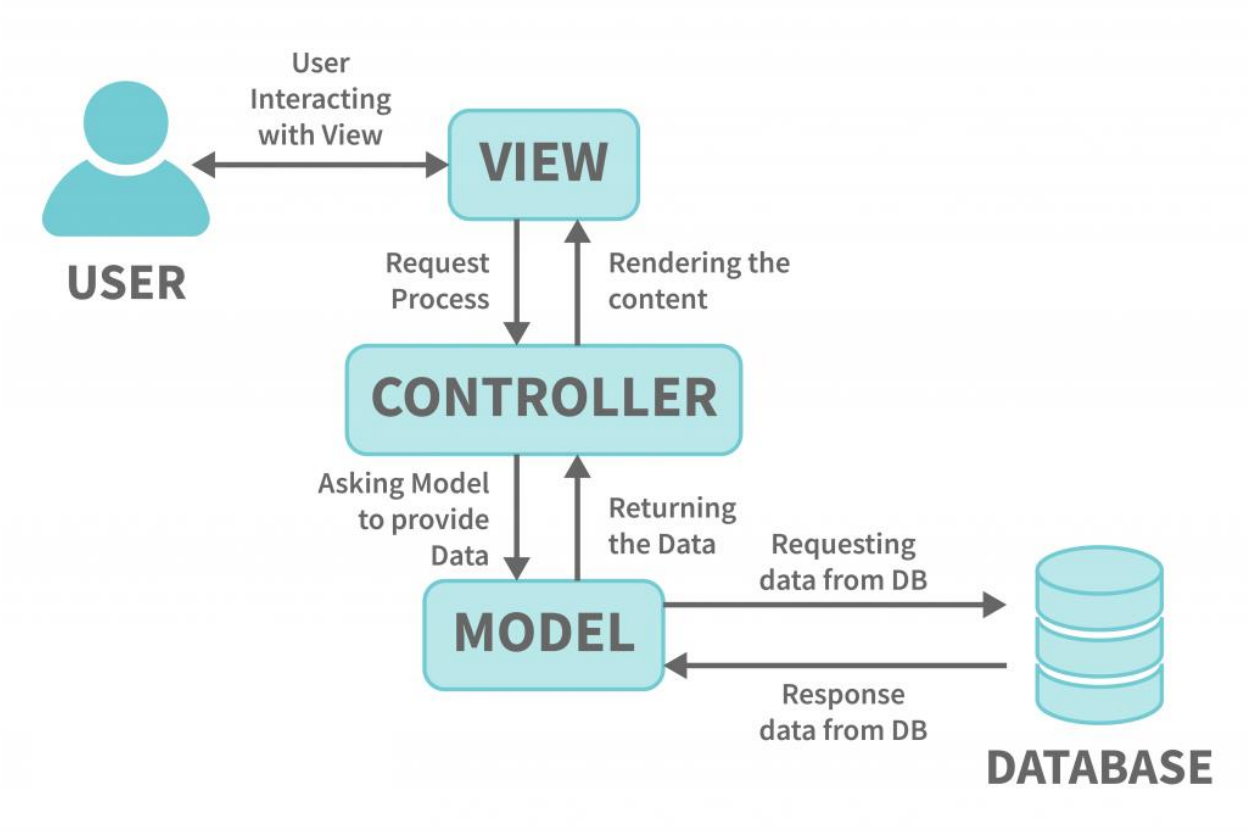
- Αρχιτεκτονική SOA (Service-Oriented Architecture): Η εφαρμογή αναπτύσσεται ως ένα σύνολο ανεξάρτητων υπηρεσιών που παρέχουν λειτουργίες μέσω διασυνδέσεων. Οι υπηρεσίες μπορούν να επαναχρησιμοποιηθούν και να συνδυαστούν για να παρέχουν συνολική λειτουργικότητα.

Στην συγκεκριμένη περίπτωση όπου η εφαρμογή δεν έχει μεγάλη πολυπλοκότητα και ο χρόνος αλλά και το κόστος είναι περιορισμένο, μια καλή επιλογή είναι η MVC αρχιτεκτονική. Η αρχιτεκτονική αυτή είναι σχετικά απλή στην υλοποίηση ενώ με τη διαχωρισμένη δομή του μοντέλου, της προβολής και του ελεγκτή, ο κώδικας μπορεί να οργανωθεί με σαφήνεια και ευκολία. Επίσης η αρχιτεκτονική MVC ενθαρρύνει την επαναχρησιμοποίηση του κώδικα καθώς το μοντέλο και η προβολή μπορούν να επαναχρησιμοποιηθούν σε μελλοντικές αναβαθμίσεις της εφαρμογής, μειώνοντας έτσι τον χρόνο και τις προσπάθειες ανάπτυξης. Τέλος διευκολύνει τη συντήρηση και την επέκταση της εφαρμογής στο μέλλον αφού χάρη στη διαχωρισμένη δομή, μπορούν να προστεθούν εύκολα νέες λειτουργίες ή να τροποποιηθεί η λειτουργικότητα χωρίς να επηρεαστεί ολόκληρη η εφαρμογή.

Στην αρχιτεκτονική MVC, ο κύριος ρόλος διαδραματίζεται από τρεις συνιστώσες:

- Το μοντέλο (Model): Το μοντέλο αναπαριστά τα δεδομένα και τη λογική της εφαρμογής. Αναλαμβάνει την αποθήκευση, την επεξεργασία και την ανάκτηση των δεδομένων, καθώς και την υλοποίηση των κατάλληλων λειτουργιών και μεθόδων. Το μοντέλο δεν γνωρίζει τίποτα για την παρουσίαση ή τον τρόπο που επικοινωνεί με τους χρήστες.
- Η προβολή (View): Η προβολή αναλαμβάνει την απεικόνιση των δεδομένων στους χρήστες. Παρουσιάζει την πληροφορία σε μια μορφή που είναι κατανοητή και ευανάγνωστη για τους αυτούς. Η προβολή δεν περιέχει λογική, αλλά εξυπηρετεί απλώς την παρουσίαση των δεδομένων που παρέχονται από το μοντέλο.
- Ο ελεγκτής (Controller): Ο ελεγκτής διαχειρίζεται τις αλληλεπιδράσεις μεταξύ του μοντέλου και της προβολής. Ανταποκρίνεται στις ενέργειες των χρηστών και ενημερώνει το μοντέλο ή την προβολή ανάλογα. Ο ελεγκτής είναι υπεύθυνος για την εκτέλεση της κατάλληλης λογικής για την ανταπόκριση σε ενέργειες όπως κλικ κουμπιού, αίτηση φόρμας κ.λπ.

Η αρχιτεκτονική MVC παρέχει μια διαχωρισμένη δομή για τον κώδικα και διευκολύνει τη συντήρηση, τη διαχείριση σφαλμάτων και την επαναχρησιμοποίηση των συνιστωσών. Επίσης, δίνει τη δυνατότητα της υλοποίησης διάφορες πτυχές της εφαρμογής ταυτόχρονα, αυξάνοντας την αποτελεσματικότητα της ανάπτυξης.



Εικόνα 2.1.1: Διάγραμμα λειτουργία MVC Αρχιτεκτονικής [35]

2.2 Εργαλεία ανάπτυξης εφαρμογής

Για την ανάπτυξη μιας εφαρμογής, χρειάζεται ένα σύνολο εργαλείων που θα βοηθήσουν στην υλοποίηση των λειτουργιών και τη διαχείριση του κώδικα, τα οποία περιλαμβάνουν αναπτυξιακά πλαίσια (frameworks), βιβλιοθήκες και προγράμματα που επιτρέπουν την αποτελεσματική ανάπτυξή της. Στη συνέχεια περιγράφονται αναλυτικά τα αντίστοιχα εργαλεία που χρησιμοποιήθηκαν και τον σκοπό που εξυπηρετούν.

2.2.1 Αναπτυξιακά πλαίσια (Frameworks)

Τα αναπτυξιακά πλαίσια είναι ένα σύνολο προκαθορισμένων λειτουργιών, βιβλιοθηκών και εργαλείων που βοηθούν στην ανάπτυξη εφαρμογών. Αυτά τα πλαίσια παρέχουν δομές και συμβάσεις για την οργάνωση του κώδικα και την ανάπτυξη λειτουργιών, όπως η διαχείριση δικτύων, η ανταλλαγή δεδομένων, η ασφάλεια κ.α.

Ένα από τα σημαντικότερα frameworks που χρησιμοποιήθηκαν είναι το Django, στο οποίο βασίστηκε η υλοποίηση του backend. Το Django είναι ένα ανοικτού κώδικα πλαίσιο που βασίζεται στη γλώσσα προγραμματισμού python. Αποτελεί ένα από τα πιο δημοφιλή πλαίσια ανάπτυξης για την κατασκευή ιστοσελίδων και εφαρμογών. Ο στόχος του Django είναι να προσφέρει μια υψηλής ποιότητας ανάπτυξη ιστοσελίδων, γρήγορα και με ασφάλεια.

Η αρχιτεκτονική του Django βασίζεται στο μοντέλο ανάπτυξης MVC και στην αρχή του DRY (Don't Repeat Yourself με αποτέλεσμα να διευκολύνει την οργάνωση, τη συντήρηση και την επέκταση του κώδικα. Το μοντέλο ανάπτυξης MVC του Django αποτελείται από τα εξής συστατικά:

- Μοντέλα (Models): Τα μοντέλα αντιπροσωπεύουν τη δομή και την αλληλεπίδραση με τα δεδομένα της εφαρμογής. Ορίζονται ως Python κλάσεις και περιέχουν τις περιγραφές των πινάκων της βάσης δεδομένων και τις σχέσεις μεταξύ τους.
- Προβολές (Views): Οι προβολές είναι υπεύθυνες για τον χειρισμό των αιτημάτων των χρηστών και την επεξεργασία των δεδομένων. Αποτελούν το μέρος του κώδικα που ανταποκρίνεται στα URLs και παράγει τα απαραίτητα δεδομένα για τις προβολές.
- Πρότυπα (Templates): Τα πρότυπα είναι αρχεία που καθορίζουν την εμφάνιση και τη δομή των δεδομένων που παρουσιάζονται στον χρήστη. Χρησιμοποιούν τη γλώσσα προτύπων του Django για τη δημιουργία δυναμικών προβολών.

Επιπλέον το Django περιλαμβάνει δυνατότητες όπως αυτόματη διαχείριση των βάσεων δεδομένων, διαχείριση χρηστών, αυθεντικοποίηση, δρομολόγηση URLs, αναγνώριση αλλαγών στο μοντέλο και πολλά άλλα.

Το framework αυτό επιλέχτηκε για την εφαρμογή καθώς συμβαδίζει με την αρχιτεκτονική του συστήματος αλλά και με τις απαιτήσεις της εφαρμογής. Οι ενσωματωμένες λειτουργίες που παρέχει βοηθάει στην ταχεία ανάπτυξη της εφαρμογής και χάρη στην αρχιτεκτονική του διευκολύνεται η οργάνωση και η διαχείριση του κώδικα. Επίσης, ένα σημαντικό πλεονέκτημά του είναι η ενεργή και ανεπτυγμένη κοινότητά του, η οποία παρέχει υποστήριξη, τεκμηρίωση και πληθώρα παραδειγμάτων και εργαλείων. Αυτό διευκολύνει την επίλυση προβλημάτων, την εύρεση πόρων και την εκμάθηση της πλατφόρμας. Για τους λόγους αυτούς επομένως, το Django θεωρήθηκε ως το πιο κατάλληλο πλαίσιο για την υλοποίηση του backend.

Ένα ακόμα αξιοσημείωτο πλαίσιο που χρησιμοποιήθηκε είναι το ReactJS, μια δημοφιλής βιβλιοθήκη JavaScript που χρησιμοποιείται για τη δημιουργία διεπαφών χρήστη (UI) για ιστοσελίδες και εφαρμογές. Η αρχιτεκτονική του ReactJS βασίζεται σε κομμάτια κώδικα (components), τα οποία είναι αυτόνομα και αναπαριστούν μια συγκεκριμένη λειτουργικότητα ή παρουσίαση στη διεπαφή του χρήστη.

Το ReactJS λειτουργεί με τη χρήση ενός εικονικού DOM (Μοντέλο Αντικειμένου Εγγράφου), το οποίο είναι μια αναπαράσταση του πραγματικού DOM. Αντί να αλληλεπιδρά απευθείας με τον πραγματικό DOM κάθε φορά που γίνονται αλλαγές στην κατάσταση της εφαρμογής, το ReactJS χρησιμοποιεί το εικονικό DOM για να υπολογίσει και να απεικονίσει μόνο τις αλλαγές που απαιτούνται. Αυτή η προσέγγιση επιτρέπει στο ReactJS να είναι αποδοτικό και γρήγορο στην ανανέωση της διεπαφής χρήστη.

Η ανάπτυξη γίνεται με την κατασκευή συστατικών (components) που αντιπροσωπεύουν τμήματα της διεπαφής του χρήστη. Αυτά τα συστατικά μπορούν να επαναχρησιμοποιηθούν και να συνδυαστούν για τη δημιουργία πολύπλοκων διεπαφών. Το ReactJS χρησιμοποιεί την αρχή της ανταπόκρισης (reactivity) για να ενημερώνει αυτόματα τη διεπαφή όταν αλλάζει η κατάσταση των συστατικών.

Μια σημαντική έννοια είναι τα "props" (properties), που είναι παράμετροι που μεταβιβάζονται στα components για να τα διαμορφώσουν. Τα props μπορούν να περιέχουν δεδομένα ή συναρτήσεις και μπορούν να περάσουν από ένα component σε ένα άλλο, δημιουργώντας ένα δέντρο σύνθετων components.

Το ReactJS προσφέρει ένα απλό και ευέλικτο μοντέλο ανάπτυξης, με αποτέλεσμα να διευκολύνει την δημιουργία της διεπαφής χρήστη. Επιλέχτηκε αντί για άλλες παραδοσιακές μεθόδους υλοποίησης της διεπαφής, κυρίως διότι η χρήση του εικονικού DOM και της ανταπόκρισης (reactivity) το καθιστούν γρήγορο και αποδοτικό στην ανανέωση της διεπαφής χρήστη. Έτσι ο χρήστης μπορεί να αλληλεπιδρά με την εφαρμογή πιο φυσικά, αφού οι αλλαγές εμφανίζονται ταχύτατα. Επιπλέον το ReactJS έχει μια μεγάλη και ενεργή κοινότητα

Το τελευταίο framework που χρησιμοποιήθηκε είναι το Scrapy, ένα πλαίσιο ανάπτυξης σε Python που χρησιμοποιείται για τον αυτοματισμό της ανάκτησης δεδομένων από ιστοσελίδες (web

scraping). Το Scrapy επιτρέπει τη δημιουργία ενός προσαρμόσιμου και αποδοτικού συστήματος για την εξαγωγή δεδομένων από διάφορες πηγές, όπως ιστοσελίδες, αρχεία JSON ή XML κ.α.

Το Scrapy χρησιμοποιεί «αράχνες» (Spiders) για το web crawling, οι οποίες είναι κλάσεις που καθορίζουν τη δομή των ιστοσελίδων που θα ανακτηθούν, τις πληροφορίες που θα εξαχθούν και τον τρόπο ανάκτησης των δεδομένων. Το Scrapy επίσης παρέχει πολλά εργαλεία και μηχανισμούς που διευκολύνουν τον αυτοματισμό της ανάκτησης δεδομένων, όπως την παράλληλη επεξεργασία, τη διαχείριση αυτόματων ερωτημάτων και την αποθήκευση των δεδομένων σε διάφορες μορφές (π.χ. CSV, JSON) .Στην περίπτωση της εφαρμογής, όπου θα χρησιμοποιηθούν πολλές ιστοσελίδες για την συλλογή σταθμών φόρτισης, το Scrapy επιτρέπει την κατασκευή διαφορετικού spider για κάθε ιστοσελίδα ώστε το scrapping να είναι προσαρμοσμένο στα χαρακτηριστικά της. Στη συνέχεια, τα δεδομένα μπορούν να αποθηκευτούν σε μια βάση δεδομένων ή να εξαχθούν σε άλλα αρχεία για περαιτέρω επεξεργασία μέσω ειδικών pipelines, δηλαδή ειδικοί αγωγοί οι οποίοι λαμβάνουν τα δεδομένα από τα spiders και ορίζουν πού θα τοποθετηθούν.

Το framework αυτό αποτελεί ισχυρή επιλογή καθώς είναι βασισμένο στην αρχιτεκτονική του Django και είναι πολύ εύκολο να ενσωματωθεί σε αυτό. Επίσης είναι σχεδιασμένο για να είναι γρήγορο και αποδοτικό στην ανάκτηση δεδομένων και ταυτόχρονα προσφέρει αξιοπιστία, ευελιξία και απόδοση, καθώς και μια ενεργή κοινότητα για υποστήριξη και ανάπτυξη.

2.2.2 Βιβλιοθήκες (Libraries)

Οι βιβλιοθήκες είναι συλλογές προκαθορισμένου κώδικα που παρέχουν έτοιμες λειτουργίες για συγκεκριμένες ανάγκες. Μπορούν να χρησιμοποιηθούν για να επιταχύνουν την ανάπτυξη και να παρέχουν πρόσθετες δυνατότητες. Στη συνέχεια αναφέρονται μερικές από τις σημαντικότερες βιβλιοθήκες που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής.

NumPy: Η βιβλιοθήκη NumPy παρέχει υψηλής απόδοσης πίνακες και λειτουργίες που επιτρέπουν την εκτέλεση γρήγορων αριθμητικών υπολογισμών στην Python. Είναι ιδιαίτερα χρήσιμη για επιστημονικούς υπολογισμούς, επεξεργασία εικόνων, μοντελοποίηση δεδομένων και άλλες αριθμητικές εργασίες.

Pandas: Η βιβλιοθήκη Pandas παρέχει δομές δεδομένων υψηλής απόδοσης όπως τα DataFrames, τα οποία επιτρέπουν την ευέλικτη και αποτελεσματική ανάλυση, μετασχηματισμό και επεξεργασία δεδομένων στην Python. Είναι ιδιαίτερα χρήσιμη για την εξερεύνηση και ανάλυση δομημένων δεδομένων.

Django REST framework: Το Django REST framework είναι μια παράταξη βιβλιοθηκών για την ανάπτυξη Web API με το Django framework. Παρέχει εργαλεία για την εύκολη δημιουργία και αυτοματοποίηση των API, καθώς και υποστήριξη για αυθεντικοποίηση, αδειοδότηση και άλλες λειτουργίες.

Material-UI: Το Material-UI είναι μια opensource βιβλιοθήκη για τη δημιουργία διεπαφών χρήστη (UI) σε React εφαρμογές. Βασισμένο στο σχεδιασμό της Google με το όνομα Material Design, το Material-UI παρέχει ένα σύνολο προκαθορισμένων στοιχείων UI, όπως κουμπιά, φόρμες, πίνακες, γραφήματα, και πολλά άλλα, τα οποία ακολουθούν τις αρχές του Material Design.

Statsmodels.tsa.seasonal: Η υποβιβλιοθήκη statsmodels.tsa.seasonal παρέχει μοντέλα για την ανάλυση και πρόβλεψη χρονοσειρών με εποχιακή συμπεριφορά. Περιλαμβάνει μεθόδους για την ανίχνευση εποχιακών μοτίβων, την εκτίμηση μοντέλων και την πρόβλεψη με βάση αυτά.

Matplotlib.pyplot: Το Matplotlib.pyplot είναι ένα υπο-πακέτο του Matplotlib που παρέχει συναρτήσεις για την οπτικοποίηση δεδομένων. Χρησιμοποιείται συχνά για τη δημιουργία γραφημάτων και αναπαραστάσεων δεδομένων.

Seaborn: Η βιβλιοθήκη Seaborn παρέχει επιπλέον γραφικά για την οπτικοποίηση δεδομένων στην Python. Βασίζεται στη βιβλιοθήκη Matplotlib και παρέχει εύκολες συναρτήσεις για τη δημιουργία θεματικών γραφημάτων και επιλογών.

Math: Η βιβλιοθήκη Math παρέχει μαθηματικές συναρτήσεις και εργαλεία για αριθμητικούς υπολογισμούς στην Python. Περιλαμβάνει συναρτήσεις όπως trigonometric, logarithmic, exponential, κ.ά.

Shapely.geometry: Η βιβλιοθήκη Shapely.geometry παρέχει γεωμετρικά αντικείμενα και λειτουργίες για γεωμετρικές επεξεργασίες, όπως σημεία, γραμμές, πολύγωνα, κ.ά. Χρησιμοποιείται συχνά σε εφαρμογές GIS (Συστήματα Πληροφοριών Γεωγραφικού Χώρου).

Geopandas: Η βιβλιοθήκη Geopandas παρέχει επέκταση της βιβλιοθήκης Pandas για εργασία με χωρικά δεδομένα. Παρέχει δομές δεδομένων όπως τα GeoDataFrame και λειτουργίες για την ανάλυση, επεξεργασία και οπτικοποίηση χωρικών δεδομένων.

Shapefile: Η βιβλιοθήκη Shapefile παρέχει λειτουργίες για την ανάγνωση, εγγραφή και επεξεργασία αρχείων shapefile, που είναι ένα δημοφιλές αρχείο χωρικών δεδομένων.

Datetime: Η βιβλιοθήκη Datetime παρέχει κλάσεις και λειτουργίες για ημερομηνίες και ώρες στην Python. Χρησιμοποιείται για τη δημιουργία, μετατροπή και ανάλυση ημερομηνιών.

OS: Η βιβλιοθήκη OS παρέχει λειτουργίες για τη διαχείριση του λειτουργικού συστήματος, όπως πρόσβαση σε αρχεία και φακέλους, εκτέλεση εντολών και άλλες λειτουργίες.

Sys: Η βιβλιοθήκη Sys παρέχει λειτουργίες για την αλληλεπίδραση με το σύστημα και την παραμετροποίηση της εκτέλεσης ενός προγράμματος. Χρησιμοποιείται για την πρόσβαση σε ορίσματα γραμμής εντολών και άλλες συστημικές πληροφορίες.

Requests: Η βιβλιοθήκη Requests παρέχει απλές και ευέλικτες λειτουργίες για την αποστολή αιτημάτων HTTP/1.1. Χρησιμοποιείται για την ανάκτηση και αποστολή δεδομένων μέσω δικτύου.

JSON: Η βιβλιοθήκη JSON παρέχει λειτουργίες για την κωδικοποίηση και αποκωδικοποίηση δεδομένων σε μορφή JSON (JavaScript Object Notation), το οποίο είναι ένα δημοφιλές μορφότυπο ανταλλαγής δεδομένων.

Googlemaps: Η βιβλιοθήκη Googlemaps παρέχει πρόσβαση στις υπηρεσίες του Google Maps. Χρησιμοποιείται για την ανάκτηση γεωγραφικών δεδομένων, την αναζήτηση τοποθεσιών, τη δρομολόγηση και άλλες λειτουργίες σχετικές με τον χάρτη και τις τοποθεσίες.

2.2.3 Επιπλέον εργαλεία (Extra tools)

Επιπλέον αξιοσημείωτα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής είναι αρχικά το Git για την καταγραφή και την αποθήκευση των αλλαγών στον κώδικα της εφαρμογής, καθώς και για την διαχείριση και τον συγχρονισμό του κώδικα με το GitHub. Επίσης χρησιμοποιήθηκε το PostgreSQL ως βάση δεδομένων το οποίο παρέχει και την αντίστοιχη εφαρμογή, η οποία έκανε πιο εύκολη την δημιουργία, την τροποποίηση και τη διαγραφή πινάκων αλλά και γενικά τη διαχείριση δεδομένων. Επιπλέον χρησιμοποιήθηκε το Postman για την δοκιμή του API κατά την ανάπτυξή του. Το Google Maps API επίσης έχει σημαντική θέση στα εργαλεία, καθώς χρειάστηκε στο frontend και στο backend για την διαχείριση του χάρτη αλλά και για τις λειτουργίες που παρέχει όσον αφορά τα γεωγραφικά δεδομένα. Τέλος η ανάπτυξη του κώδικα έγινε εξολοκλήρου στο Visual Studio Code, ένα ενσωματωμένο περιβάλλον ανάπτυξης (IDE) που παρέχει πληθώρα λειτουργιών και επεκτάσεων για την ανάπτυξη λογισμικού.

Κεφάλαιο 3ο . Συλλογή και επεξεργασία δεδομένων φόρτισης

3.1 Συλλογή και επαύξηση δεδομένων

Ένα σημαντικό πρώτο βήμα για την ανάπτυξη της εφαρμογής είναι η εύρεση δεδομένων όσον αφορά τις αφίξεις στους σταθμούς και τον χρόνο φόρτισης σε αυτούς. Η ιδανική περίπτωση θα ήταν να υπήρχαν πραγματικά δεδομένα αφίξεων για κάθε σταθμό της εφαρμογής, ώστε στη συνέχεια να εφαρμόζεται ο αλγόριθμος συστάσεων και να προτείνει τον πιο κατάλληλο σταθμό στον χρήστη. Ωστόσο αυτό θα απαιτούσε την συνεργασία από πολλούς παρόχους σταθμών φόρτισης, το οποίο είναι δύσκολα εφικτό, αλλά και την ύπαρξη αρκετών επισκέψεων ανά σταθμό ώστε να έχει νόημα ένας αλγόριθμος συστάσεων.

Όπως ήδη αναφέρθηκε παραπάνω, η Ελλάδα ακόμα βρίσκεται σε στάδιο ανάπτυξης σε αυτό το θέμα, με αποτέλεσμα οι σταθμοί που υπάρχουν να μην χρησιμοποιούνται ακόμα σε μεγάλο βαθμό αλλά να παραμένουν άδειοι το μεγαλύτερο μέρος της ημέρας. Αυτό συνεπάγεται τα δεδομένα αφίξεων να μην είναι αρκετά πυκνά ώστε να δημιουργείται συνωστισμός στους σταθμούς και έτσι να μην χρειάζεται ένα αλγόριθμος συστάσεων.

Το πρόβλημα αυτό οδήγησε στην ανάγκη εύρεσης και τροποποίησης δεδομένων ώστε να περιγράφουν μια μελλοντική κατάσταση στην οποία θα υπάρχει υψηλή χρήση των σταθμών φόρτισης και έτσι θα χρειάζεται μια εφαρμογή που να βοηθάει και να προτείνει στον κάθε χρήστη τον βέλτιστο σταθμό φόρτισης. Αρχικό βήμα επομένως ήταν η εύρεση δεδομένων φόρτισης τα οποία θα χρησιμοποιηθούν ως βάση για την δημιουργία προφίλ, που θα περιγράφουν τον μέσο όρο επισκέψεων σε κάθε σταθμό. Αυτό σημαίνει πως θα χρειαστούν πολλά δεδομένα τα οποία να είναι αρκετά πυκνά ώστε να είναι δυνατή η εξαγωγή ενός μοτίβου αφίξεων από το οποίο στη συνέχεια θα προκύψει ένα προφίλ. Η διαδικασία που ακολουθεί επομένως χωρίζεται σε δύο βασικά τμήματα: την επιλογή των δεδομένων που θα χρησιμοποιηθούν με βάση τα κατάλληλα κριτήρια και την επαύξησή τους στη συνέχεια ώστε να περιγράφουν την μελλοντική συμπεριφορά των συχνών αφίξεων.

3.2 Συλλογή δεδομένων

Αρχικά πρέπει να συλλεχθούν τα δεδομένα με σκοπό την δημιουργία ενός προφίλ επισκέψεων για κάθε σταθμό φόρτισης. Τα προφίλ που προκύψουν πρέπει να είναι ρεαλιστικά ως προς την συμπεριφορά της κάθε περιοχής στην οποία βρίσκεται ο κάθε σταθμός και να αντιπροσωπεύουν τις ώρες αιχμής που θα είχε. Για τον λόγο αυτό οι σταθμοί χωρίστηκαν σε τρείς κατηγορίες, ανάλογα με την τοποθεσία και τον πληθυσμό της περιοχής στην οποία βρίσκονται. Οι κατηγορίες αυτές είναι:

1. Πρώτη Κατηγορία: Περιλαμβάνει σταθμούς οι οποίοι ανήκουν σε δήμους στο πολεοδομικό συγκρότημα της Αθήνας ή της Θεσσαλονίκης και στις 20 μεγαλύτερες πόλεις της Ελλάδας[1]. Οι σταθμοί αυτής της κατηγορίας αυτής αφορούν περιοχές με μεγάλο πληθυσμό και αναμένεται να έχουν τη μεγαλύτερη κίνηση καθώς και διακριτές ώρες αιχμής, στις οποίες είναι γεμάτοι. Για την κατηγορία αυτή το σύνολο δεδομένων πρέπει να είναι σχετικά πυκνό, με πολλαπλές αφίξεις κατά την διάρκεια της ημέρας και με ώρες αιχμής όπου είναι διακριτός ο συνωστισμός.
2. Δεύτερη Κατηγορία: Περιλαμβάνει σταθμούς οι οποίοι ανήκουν σε πρωτεύουσες των νομών και δήμους με πάνω από 30.000 κατοίκους. Και σε αυτήν την κατηγορία αναμένεται το σύνολο δεδομένων να έχει πολλές αφίξεις και ώρες αιχμής, σε μικρότερο ωστόσο ποσοστό και έκταση.
3. Τρίτη Κατηγορία: Περιλαμβάνει σταθμούς οι οποίοι ανήκουν σε δήμους μεσαίων και απομακρυσμένων περιοχών καθώς και μικρών νησιών. Στην κατηγορία αυτή προβλέπεται μικρή κίνηση σε σταθμούς επομένως το σύνολο δεδομένων αναμένεται να είναι αραιό με λίγες αφίξεις την μέρα και πολύ σπάνιο έως και καθόλου συνωστισμό.

Μετά την κατηγοριοποίηση των σταθμών, χρειάζεται να γίνει η συλλογή των δεδομένων που θα περιγράφουν όσο το δυνατόν καλύτερα τις κατηγορίες αυτές και τα χαρακτηριστικά τους. Όπως ήδη έχει προαναφερθεί στο προηγούμενο κεφάλαιο, εφόσον στους σταθμούς της Ελλάδας δεν υπάρχουν αρκετά πυκνά δεδομένα, χρειάστηκε να χρησιμοποιηθούν δεδομένα από περιοχές του εξωτερικού που έχουν αυξημένη χρήση των σταθμών φόρτισης. Πέρα από την πυκνότητα ένα ακόμα κριτήριο για επιλογή των δεδομένων φόρτισης είναι και η τοποθεσία των σταθμών. Σταθμοί που δεν ήταν προσβάσιμοι σε όλο το κοινό, αλλά βρίσκονταν εντός ιδιωτικού χώρου περιορισμένης πρόσβασης ή εντός πανεπιστημίων, δεν χρησιμοποιήθηκαν καθώς η συμπεριφορά αφίξεων στους σταθμούς αυτούς είναι βασισμένη στα ωράρια των εργαζομένων και δεν είναι αντιπροσωπευτικά του γενικού κοινού της περιοχής. Τα δεδομένα που χρησιμοποιήθηκαν περιγράφονται πιο αναλυτικά παρακάτω:

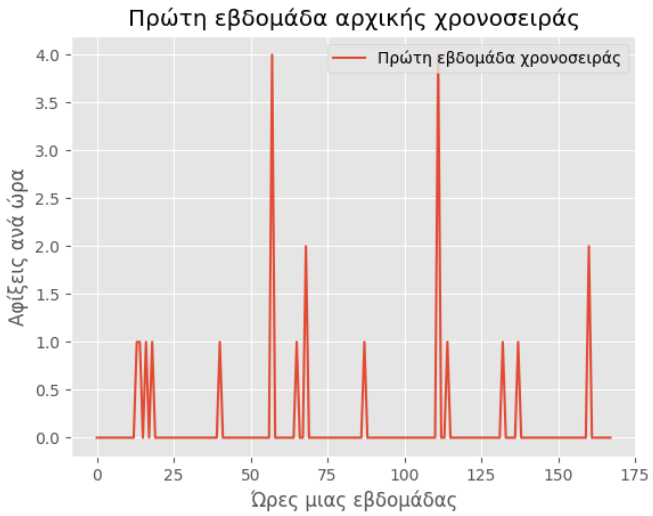
[1] Αγρίνιο, Αλεξανδρούπολη, Βόλος, Δράμα, Ηράκλειο, Ιωάννινα, Καβάλα, Καλαμάτα, Κατερίνη, Κοζάνη, Κομοτηνή, Λαμία, Λάρισα, Ξάνθη, Πάτρα, Ρόδος, Σέρρες, Τρίκαλα, Χαλκίδα, Χανιά

3.2.1 Κατηγορία 1: Πυκνοκατοικημένες περιοχές

Για την κατηγορία αυτή, χρησιμοποιήθηκαν δεδομένα από σταθμούς του London Borough of Barnet [26], μια περιοχή με περίπου 400,000 κατοίκους. Το συγκεκριμένο σύνολο δεδομένων αποτελούταν από πολλούς σταθμούς, με αποτέλεσμα να είναι πολύ πυκνό σαν σύνολο και να έχει ψηλές αιχμές σε ορισμένες ώρες της εβδομάδας. Λόγου αυτού, αλλά και του μεγάλου πληθυσμού της πόλης, χρησιμοποιήθηκε για να αντιπροσωπεύσει την συμπεριφορά σε μεγάλες πόλεις. Ένα πρόβλημα που αντιμετωπίστηκε σε όλους σχεδόν τους σταθμούς της περιοχής είναι η μη περιοδικότητα των δεδομένων, με αποτέλεσμα να υπάρχουν πολλά διαστήματα με μηδενικές αφίξεις. Αυτό, σε συνδυασμό με την έλλειψη πυκνών δεδομένων ανά σταθμό, οδήγησε στο να επιλεχθούν οι τέσσερις σταθμοί με τις περισσότερες φορτίσεις στο ίδιο διάστημα και να προστεθούν, ώστε φτιαχτεί μία βασική χρονοσειρά χωρίς κενές περιόδους, την οποία μπορούμε να μεταχειριστούμε στη συνέχεια.

Παρακάτω φαίνεται η χρονοσειρά που προέκυψε και ένα τμήμα της με διάρκεια μιας εβδομάδας:

Εικόνα 3.2.1.1: Αρχική χρονοσειρά αφίξεων κατηγορίας 1



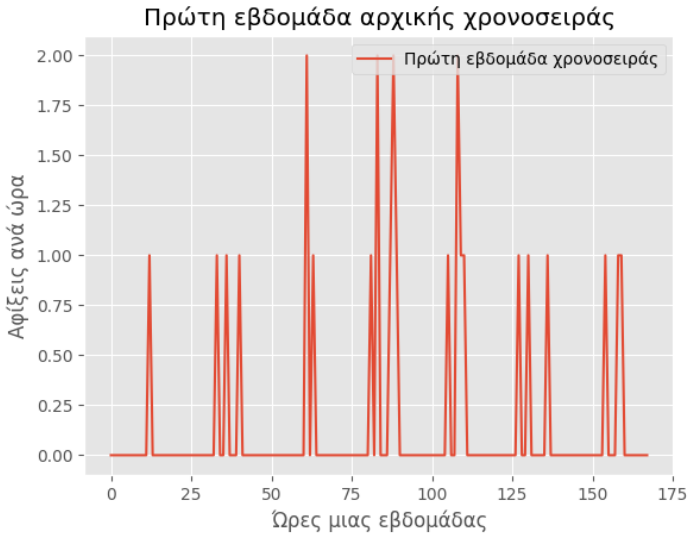
Εικόνα 3.2.1.2: Πρώτη εβδομάδα της αρχικής χρονοσειράς αφίξεων κατηγορίας 1

3.2.2 Κατηγορία 2: Λοιπές αστικές περιοχές

Στην κατηγορία αυτή χρησιμοποιήθηκαν δεδομένα από σταθμούς του Boulder, Colorado [27], μια ημιαστική περιοχή με 100,000 κατοίκους. Η χρονοσειρά που προέκυψε από αυτό το σύνολο δεδομένων, ήταν πυκνή με μικρότερες κορυφές φανερώνοντας έτσι αρκετή κίνηση στη διάρκεια της ημέρας αλλά χωρίς τόσο συνωστισμό όπως στο προηγούμενο σύνολο δεδομένων. Αυτό το καθιστά κατάλληλο για να αντιπροσωπεύσει μικρότερες αλλά αρκετά μεγάλες πόλεις όπως αυτές που περιλαμβάνει αυτή η κατηγορία. Λόγω μη περιοδικότητας και πάλι των σταθμών δεν γινόταν να χρησιμοποιηθεί ο Μ.Ο. πολλών σταθμών για να φτιαχτεί μια βασική χρονοσειρά. Βρέθηκε ωστόσο ένας σταθμός με ικανοποιητικό αριθμό αφίξεων χωρίς κενά, και έτσι χρησιμοποιήθηκε ως σταθμός αναφοράς για τη συνέχεια.

Παρακάτω φαίνεται η χρονοσειρά που προέκυψε και ένα τμήμα της με διάρκεια μιας εβδομάδας:

Εικόνα 3.2.2.1: Αρχική χρονοσειρά αφίξεων κατηγορίας 2



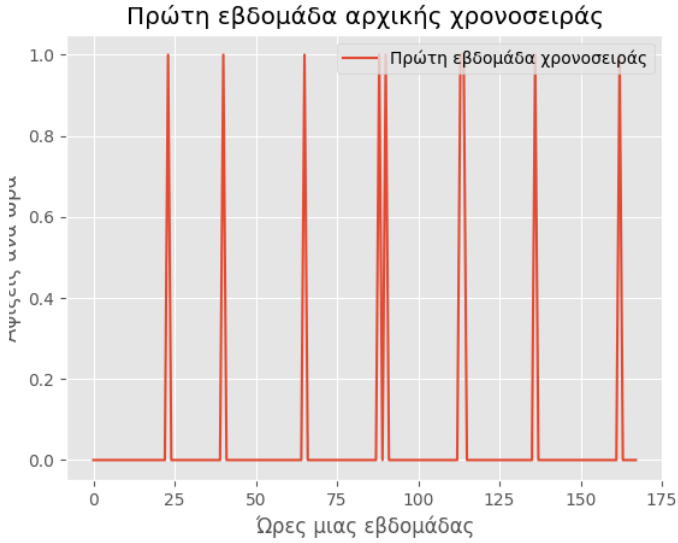
Εικόνα 3.2.2.2: Πρώτη εβδομάδα της αρχικής χρονοσειράς αφίξεων κατηγορίας 2

3.2.3 Κατηγορία 3: Αραιοκατοικημένες περιοχές

Τέλος στην κατηγορία αυτή χρησιμοποιήθηκαν σταθμοί από το προάστιο του Λονδίνου, Borough of Barnet [28]. Αν και η περιοχή αυτή βρίσκεται κοντά στο Λονδίνο, που είναι μεγάλη πόλη, οι αφίξεις του συνόλου δεδομένων ήταν πολύ αραιές και η κίνηση στους συγκεκριμένους σταθμούς ήταν πολύ περιορισμένη. Για τον λόγο αυτό το εν λόγω σύνολο δεδομένων αποτέλεσε κατάλληλη επιλογή να για χρησιμοποιηθεί σε σταθμούς μικροκατοικημένων περιοχών. Συγκεκριμένα επιλέχθηκαν 2 σταθμοί με τον περισσότερο αριθμό αφίξεων και την μικρότερη επικάλυψη, καθώς πολλοί από τους σταθμούς είχαν δεδομένα για συγκεκριμένα χρονικά διαστήματα.

Παρακάτω φαίνεται η χρονοσειρά που προέκυψε και ένα τμήμα της με διάρκεια μιας εβδομάδας:

Εικόνα 3.2.3.1: Αρχική χρονοσειρά αφίξεων κατηγορίας 3



Εικόνα 3.2.3.2: Πρώτη εβδομάδα της αρχικής χρονοσειράς αφίξεων κατηγορίας 3

3.3 Επαύξηση δεδομένων

Παρά την επιλογή δεδομένων με βάση την κατηγορία στην οποία ανήκει ο σταθμός, η μορφή τους δεν είναι ακόμα ιδανική για την δημιουργία προφίλ. Στην αρχική κατάσταση των δεδομένων, οι αφίξεις δεν είναι αρκετές καθώς υπάρχουν πολλές ώρες μέσα στη μέρα όπου δεν υπάρχει καμία άφιξη. Τελικός σκοπός είναι να υπάρχουν αρκετές αφίξεις ώστε ο μέσος όρος του συνόλου δεδομένων ανά ώρα να περιγράφει την συμπεριφορά αφίξεων του σταθμού ανάλογα με την περιοχή όπου βρίσκεται, και φυσικά να διαφέρει να σταθμό, κρατώντας όμως εμφανές ένα λογικό μοτίβο αφίξεων.

Για να μπορεί να συμβεί αυτό, τα δεδομένα χρειάζεται να είναι ακόμα πιο πυκνά και επίσης χρειάζεται να είναι σχετικά διαφορετικά για κάθε σταθμό. Συνεπώς χρειάζεται να γίνει επαύξηση των δεδομένων με τρόπο τέτοιο ώστε να προκύπτει μια διαφορετική χρονοσειρά κάθε φορά. Η διαδικασία αυτή αποφασίστηκε να πραγματοποιηθεί με την εφαρμογή bootstrapping, γνωστό και ως επαναληπτική δειγματοληψία. Το bootstrapping είναι μια στατιστική τεχνική που χρησιμοποιείται για γίνει η εκτίμηση της αβεβαιότητας ή της διακύμανσης μιας στατιστικής παραμέτρου, όπως ο μέσος όρος ή ο συντελεστής διάσπασης, χωρίς να είναι γνωστή η ακριβής κατανομή των δεδομένων. Ο τρόπος που λειτουργεί περιλαμβάνει την δημιουργία νέων δεδομένων μέσω επαναληπτικής δειγματοληψίας από το αρχικό δείγμα. Συχνά χρησιμοποιείται σε διάφορα στατιστικά προβλήματα, όπως η εκτίμηση μέσου όρου, η πρόβλεψη μιας χρονοσειράς κλπ, καθώς είναι αρκετά ευέλικτο και χρησιμοποιεί μόνο διαθέσιμα δεδομένα ακόμα και αν είναι ελλιπή. Στην συγκεκριμένη περίπτωση το bootstrapping χρησιμοποιήθηκε για την κατασκευή νέων χρονοσειρών από μία αρχική χρονοσειρά, με σκοπό να έχουν ίδια δομικά χαρακτηριστικά αλλά διαφορετική τυχαία διακύμανση [36].

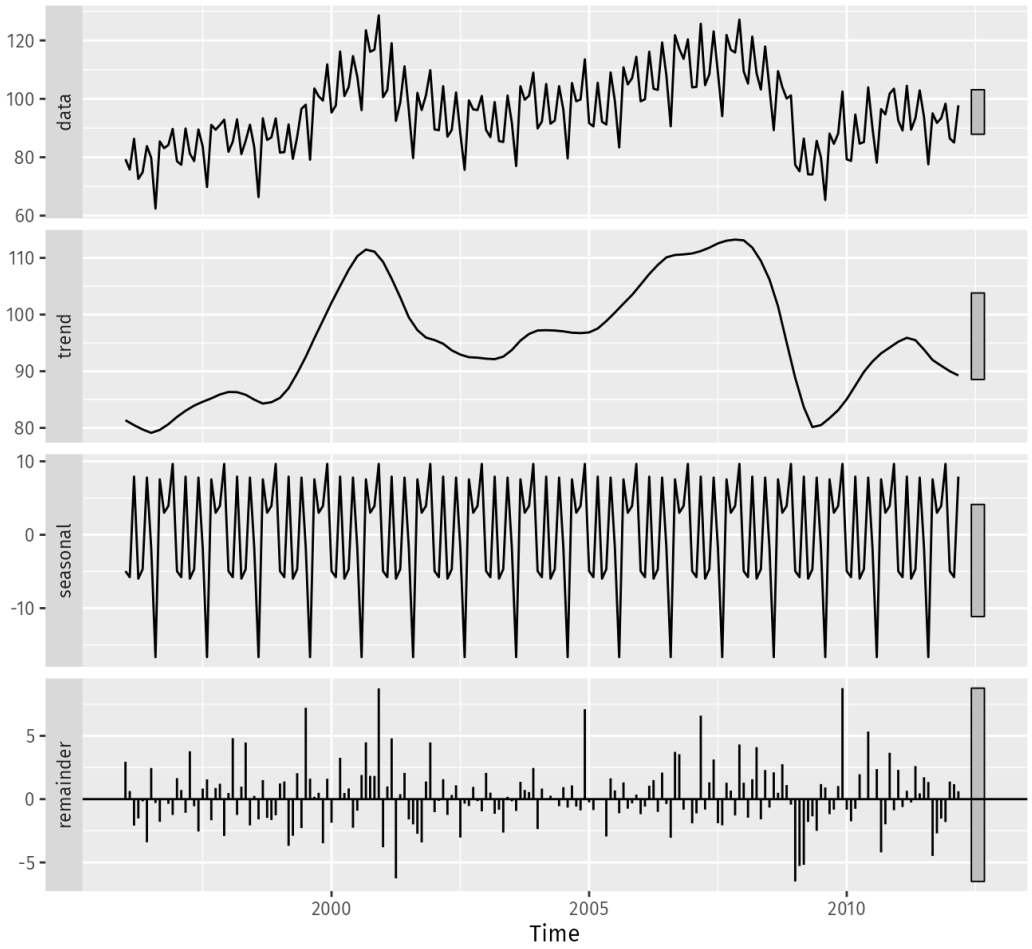
3.3.1 Μετασχηματισμός δεδομένων Box-Cox

Η διαδικασία που ακολουθήθηκε για την επαύξηση της χρονοσειράς κάθε κατηγορίας αρχικά περιλαμβάνει την εφαρμογή Box-Cox μετασχηματισμού. Ο μετασχηματισμός αυτός είναι ένα χρήσιμο εργαλείο για την εξερεύνηση και την επεξεργασία μη κανονικών δεδομένων πριν από την εφαρμογή στατιστικών αναλύσεων, καθώς σκοπός του είναι να προσεγγίσει μια κανονική κατανομή και να μειώσει την ασυμμετρία και την ανισομορφία των δεδομένων. Η τεχνική αυτή χρησιμεύει επομένως στο να κανονικοποιηθεί η χρονοσειρά και να σταθεροποιηθεί η διασπορά της, με αποτέλεσμα να αποκαλυφθεί το μοτίβο εποχικότητας. Ο μετασχηματισμός αυτός φαίνεται παρακάτω για θετικές τιμές:

$$y(\lambda) = \begin{cases} \dfrac{y^{\lambda} - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \log\, y, & \text{if } \lambda = 0 \end{cases}$$

3.3.2 Αποσύνθεση χρονοσειρών STL

Μετά την εφαρμογή του Box-Cox μετασχηματισμού ακολουθεί η αποσύνθεση της χρονοσειράς χρησιμοποιώντας Seasonal and Trend decomposition using Loess (STL), μια διαδικασία κατά την οποία η χρονοσειρά αποσυντίθεται σε τρείς κύριες συνιστώσες: την εποχιακή (seasonal), την τάση (trend) και το υπόλοιπο (residual). Η εποχικότητα αναπαριστά την κυκλική ή επαναλαμβανόμενη παραλλαγή που παρατηρείται στη χρονοσειρά σε συγκεκριμένα χρονικά διαστήματα, η τάση αναπαριστά την μακροπρόθεσμη αύξηση ή μείωση της χρονοσειράς και τέλος το υπόλοιπο αναπαριστά την τυχαιότητα ή την ασυσχέτιστη παραλλαγή που δεν εξηγείται από την εποχικότητα και την τάση. Ο μετασχηματισμός STL αναλύει τη χρονοσειρά σε επιμέρους συνιστώσες, καθιστώντας ευκολότερη την επεξεργασία και την ανάλυση της. Επίσης, μπορεί να χρησιμοποιηθεί για την απομόνωση της εποχικότητας και της τάσης από την αρχική χρονοσειρά, προκειμένου να εξεταστούν παραλλαγές της ή να γίνει καλύτερη πρόβλεψή της. Ένα παράδειγμα εφαρμογής της stl σε μία χρονοσειρά φαίνεται παρακάτω [37] :

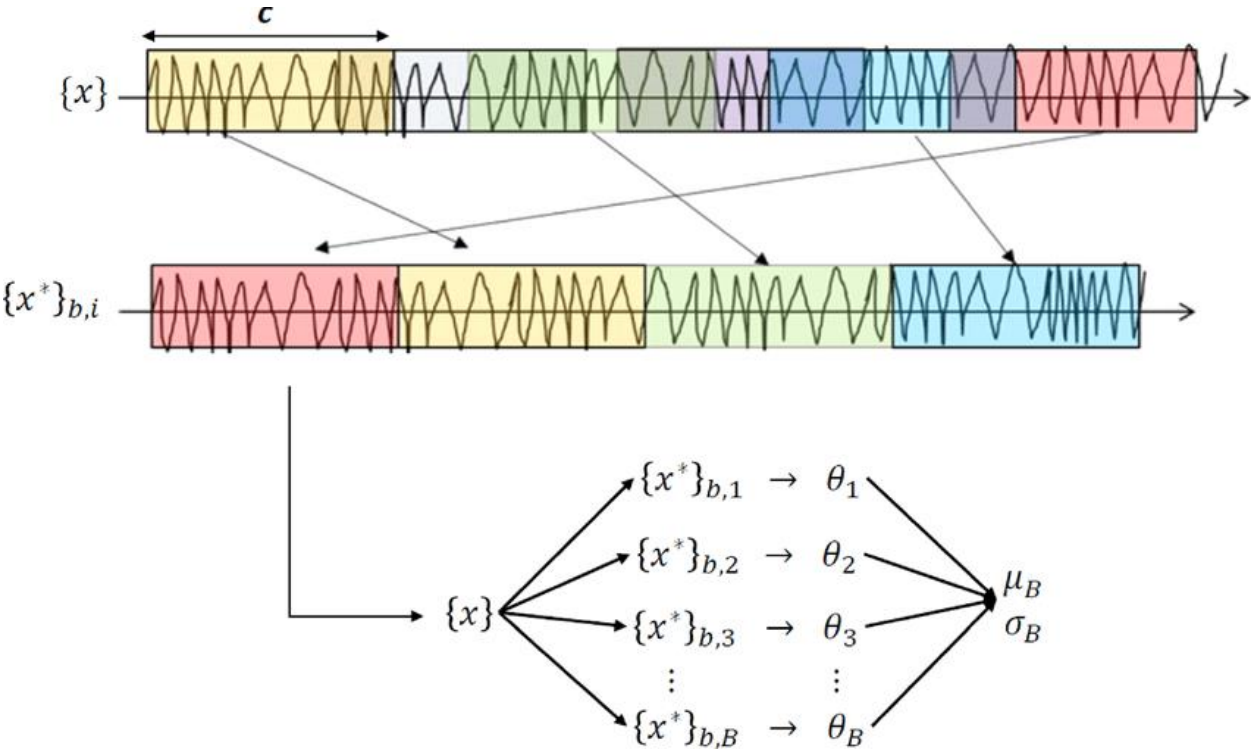


Εικόνα 3.3.2.1: Παράδειγμα αποσύνθεσής χρονοσειράς σε τάση εποχικότητα και υπόλοιπο με της stl μέθοδο [37]

Με την χρήση της μεθόδου STL είναι επομένως δυνατό να απομονωθούν οι τρεις συνιστώσες της χρονοσειράς και να χρησιμοποιηθούν στη συνέχεια. Από αυτές η εποχικότητα είναι ιδιαίτερα χρήσιμη στην αναγνώριση του μοτίβου συμπεριφοράς των αφίξεων, όπου αναμένεται να έχουμε εποχικότητα και μάλιστα σε ημερήσια αλλά και εβδομαδιαία βάση, η οποίο πρέπει να διατηρηθεί σε κάθε σταθμό της ίδιας κατηγορίας.

3.3.3 Moving Block Bootstrap

Επόμενο βήμα είναι η χρήση του υπολοίπου που απομονώθηκε με την STL μέθοδο και η εφαρμογή του αλγορίθμου Moving Block Bootstrap (MBB). Ο αλγόριθμος αυτός είναι μια τεχνική ανακατασκευής που εφαρμόζεται σε μη εξαρτημένα δεδομένα και λειτουργεί με την επανατοποθέτηση και τη μετακίνηση τμημάτων δεδομένων στην αρχική χρονοσειρά. Πιο συγκεκριμένα, ο αλγόριθμος χωρίζει την αρχική χρονοσειρά σε τμήματα σταθερού μήκους και στη συνέχεια τα τμήματα δεδομένων επανατοποθετούνται με επαναλαμβανόμενο τρόπο στην αρχική χρονοσειρά. Η διαδικασία αυτή επαναλαμβάνεται πολλές φορές και τελικά δημιουργείται μια ανακατασκευασμένη χρονοσειρά. Ένα παράδειγμα εφαρμογής του MBB φαίνεται παρακάτω [38]:



Εικόνα 3.3.3.1: Απεικόνιση εφαρμογής του αλγορίθμου MBB σε μια χρονοσειρά [38]

Με την εφαρμογή του αλγορίθμου αυτού επομένως δημιουργούνται νέα σύνολα δεδομένων που είναι παρόμοια με το αρχικό, μετατοπίζοντας τμήματά του στοχαστικά κατά μήκος του χρόνου. Το πλεονέκτημα που προσφέρει ο MBB είναι πως δημιουργεί μια νέα τυχαία χρονοσειρά, χωρίς να προσθέτει θόρυβο αλλά κρατώντας τα χαρακτηριστικά των δεδομένων που διατίθονται. Έτσι αφού χρησιμοποιηθεί ο MBB δημιουργείται ένα νέο υπόλοιπο το οποίο θα χρησιμοποιηθεί στη συνέχεια για την δημιουργία της νέας ανακατασκευασμένης χρονοσειράς. Ο λόγος που ο

αλγόριθμος αυτός χρησιμοποιείται μόνο στο υπόλοιπο, είναι διότι δεν είναι επιθυμητή η αλλοίωση της τάσης ή της εποχικότητας. Αντίθετα το υπόλοιπο μπορεί να χρησιμοποιηθεί για να προσδώσει τυχαιότητα στο σήμα, χωρίς να επηρεάσει το μοτίβο συμπεριφοράς των αφίξεων.

3.3.4 Σύνθεση χρονοσειρών

Στο νέο υπόλοιπο που δημιουργήθηκε χρειάζεται να προστεθεί η τάση και η εποχικότητα για να ξαναδημιουργηθεί η χρονοσειρά. Στο στάδιο αυτό, στην χρονοσειρά έχει προστεθεί τυχαιότητα μέσω του MBB, και στη συνέχει χρειάζεται να γίνει επαύξηση. Η επαύξηση εξαρτάται από τον αριθμό φορτιστών που διαθέτει ο κάθε σταθμός, και από την κατηγορία στην οποία ανήκει ο σταθμός. Οι σταθμοί της κατηγορίας 1 για παράδειγμα, που αφορούν πυκνοκατοικημένες περιοχές, χρειάζονται πολλές αφίξεις την ημέρα για να είναι ρεαλιστικοί, και έτσι αναμένεται να έχουν πολλές αιχμές στις χρονοσειρές τους όπου είναι γεμάτοι, και σχετικά μεγάλη κίνηση όλη τη μέρα. Αντίθετα σταθμοί της κατηγορίας 3 χρειάζονται μικρότερη επαύξηση καθώς έχουν μικρότερο πληθυσμό και επομένως μικρότερη κίνηση.

Πρώτο βήμα για να επιτευχθεί η επαύξηση, είναι να πολλαπλασιαστεί η εποχικότητα, ώστε ενισχυθεί περισσότερο το μοτίβο που ακολουθούν οι αφίξεις και να αυξηθεί περισσότερο η χρονοσειρά. Αυτό βοηθά κυρίων στην αύξηση των αιχμών της χρονοσειράς. Ο παράγοντας πολλαπλασιασμού που χρησιμοποιείται για την εποχικότητα, αυξάνεται σταδιακά ώσπου το σήμα να έχει την επιθυμητή αύξηση. Τέλος εφαρμόζεται αντίστροφος Box-Cox μετασχηματισμός και προστίθεται στην αρχική χρονοσειρά η νέα που κατασκευάστηκε. Η προσθήκη αυτή γίνεται τρεις φορές και στην συνέχεια διαιρείται δια τρία ώστε να παραμείνει ο μέσος όρος των χρονοσειρών, με σκοπό να προκύψει μία χρονοσειρά κοντά στην αρχική η οποία διατηρεί τα χαρακτηριστικά της.

Αφού κατασκευάστηκε η νέα χρονοσειρά, γίνεται έλεγχος αν έχει αυξηθεί όσο χρειάζεται και αν όχι επαναλαμβάνεται η διαδικασία. Ο έλεγχος επαύξησης γίνεται διαιρώντας το άθροισμα της νέα χρονοσειράς με την αρχική, και στη συνέχει το αποτέλεσμα στρογγυλοποιείται και συγκρίνεται με τον επιθυμητό λόγο αύξησης. Ο λόγος αύξησης της χρονοσειράς επιλέγεται τυχαία σε ένα μικρό διάστημα, το οποίο εξαρτάται από την κατηγορία και από των αριθμό θέσεων φόρτισης του κάθε σταθμού, ώστε να υπάρχει τυχαιότητα μεταξύ των σταθμών και να μην έχουν όλοι την ίδια συγκέντρωση κάθε στιγμή. Η τυχαιότητα επίσης προκύπτει χάρη στον MBB ο οποίος επιλέγει τυχαία τα τμήματα και την τοποθέτησή τους και έτσι προκύπτει διαφορετικό αποτέλεσμα κάθε φορά. Έτσι με αυτόν τον τρόπο να προκύπτει μια διαφορετική τελική χρονοσειρά για κάθε σταθμό, η οποία είναι διαμορφωμένη στα χαρακτηριστικά του.

3.4 Υλοποίηση μεθοδολογίας επαύξησης δεδομένων

Αρχικά για κάθε σταθμό επιλέγεται η αντίστοιχη χρονοσειρά αναφοράς ανάλογα με την κατηγορία στην οποία ανήκει. Στο στάδιο αυτό καθορίζεται το ποσοστό επαύξησης της χρονοσειράς που είναι επιθυμητό. Το ποσοστό αυτό εξαρτάται από τον αριθμό ρευματοδοτών που διαθέτει ο σταθμός φόρτισης, ώστε οι κορυφές της χρονοσειράς να μην το ξεπερνάνε, και την κατηγορία στην οποία ανήκει, καθώς σε πιο πυκνοκατοικημένες περιοχές αναμένεται περισσότερη κίνηση. Στην συνέχεια ξεκινάει η διαδικασία της επαύξησης με bootstrap η οποία φαίνεται παρακάτω:

```
def bootstrap(arrivals,num,mu):
    # Box-Cox transformation
    if np.min(arrivals['days']) > 1e-6:
        box_cox, lambda_ = stats.boxcox(arrivals['days'], lmbda=None)
        box_cox = pd.DataFrame(box_cox)
    else:
        box_cox, lambda_ = stats.boxcox(arrivals['days'], lmbda=1)
        box_cox = pd.DataFrame(box_cox)

    # Decomposition
    stl=sm(box_cox, model='additive', period=168)

    # Bootstrap
    # window_size = block_size = 2*freq
    mbb = MBB(stl.resid, window_size=2*168)
    for i in range(0,len(mbb)):
        mbb[i] += stl.trend[i] + stl.seasonal[i] * mu
    xs = []
    mbb =  np.nan_to_num(mbb)
    xs.append(arrivals['days'])
    for i in range(1,num):
        tmp = invboxcox(mbb,lambda_)
        tmp =  np.nan_to_num(mbb)
        xs.append(tmp)
```

Η συνάρτηση δέχεται ως παραμέτρους την αρχική χρονοσειρά με τις αφίξεις (arrivals), πόσες φορές θα προστεθεί η ανακατασκευασμένη χρονοσειρά στην αρχική (num) και τον πολλαπλασιαστή της εποχικότητας (mu). Αρχικά εφαρμόζεται ο Box-Cox μετασχηματισμός με την stats.boxcox που είναι ήδη υλοποιημένη στην βιβλιοθήκη scipy.stats. Στη συνέχεια η πλέον κανονικοποιημένη χρονοσειρά αποσυντίθεται με την STL μέθοδο με χρήση της βιβλιοθήκης statsmodels.tsa.seasonal. Στην STL χρησιμοποιείται additive μοντέλο, καθώς μελετώντας τις αρχικές χρονοσειρές, φαίνεται πως έχουν σταθερή εποχικότητα και τάση χωρίς μεγάλες αλλαγές

στο πλάτος. Αυτό φαίνεται και στο επόμενο διάγραμμα αποσύνθεσης της πρώτης χρονοσειράς, όπου η τάση φαίνεται να έχει διακυμάνσεις τάξης 0.1 που είναι αμελητέες. Η περίοδος για την αποσύνθεση είναι 168 ώρες, δηλαδή μία εβδομάδα, καθώς αυτή θα είναι η διάρκεια του προφίλ που θα δημιουργηθεί τελικά. Για τα προφίλ επιλέχτηκε μήκος μίας εβδομάδας καθώς εκεί είναι περισσότερο εμφανής η εποχικότητα, ενώ δεν εντοπίστηκαν σημαντικές αλλαγές σε αυτήν ανάμεσα σε διαφορετικούς μήνες.

Εικόνα 3.4.1: Παράδειγμα αποσύνθεσης STL της πρώτης χρονοσειράς αναφοράς

Στη συνέχεια απομονώνεται το υπόλοιπο από την χρονοσειρά και του εφαρμόζεται ο MBB αλγόριθμος. Ο αλγόριθμος αυτός χρησιμοποιείται για να προσθέσει τυχαιότητα στην τελική χρονοσειρά, χρησιμοποιώντας πραγματικά δεδομένα αντί για τυχαίο θόρυβο. Με τον τρόπο αυτό κάθε χρονοσειρά που προκύπτει θα είναι μοναδική, και θα ταυτόχρονα θα διατηρεί τα χαρακτηριστικά της αρχικής. Ο αλγόριθμος φαίνεται αναλυτικά παρακάτω:

```
def MBB(x, window_size):
    bx = np.zeros(int(np.floor(len(x) / window_size + 2) * window_size))
    for i in range(1, int(np.floor(len(x) / window_size)) + 2):
        c = np.random.randint(1, len(x) - window_size + 1)
        bx[((i - 1) * window_size + 1):(i * window_size)] = x[c:c +
window_size - 1]
```

```
    start_from = np.random.randint(0, window_size-1) + 1
    return bx[start_from :start_from - 1 + len(x)]
```

Οι παράμετροι της συνάρτησης είναι το υπόλοιπο της χρονοσειράς (x) και το μέγεθος του παραθύρου (window size), το οποίο είναι το διπλάσιο της συχνότητας δηλαδή 168 * 2. Πρώτα αρχικοποιείται ο πίνακας bx, στον οποίο θα ανακατασκευαστεί το υπόλοιπο. Στη συνέχεια ξεκινάει η επαναληπτική διαδικασία του ανακατασκευασμού. Συγκεκριμένα, επιλέγεται ένα τυχαίο σημείο της χρονοσειράς στο οποίο θα ξεκινάει το τυχαίο παράθυρο. Έπειτα το τμήμα αυτό τοποθετείται στον πίνακα bx. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να γεμίσει ο bx πίνακας. Έτσι προκύπτει ένα νέο ανακατασκευασμένο υπόλοιπο, το οποίο και επιστρέφεται.

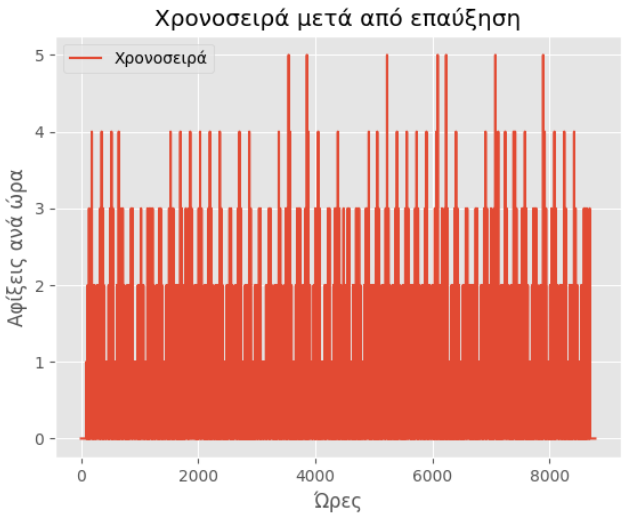
Έπειτα στο νέο υπόλοιπο προστίθενται ξανά η τάση και ο εποχικός παράγοντας πολλαπλασιασμένος, καθώς αποτελεί σημαντικός στην ανάδειξη του μοτίβου αφίξεων. Ο παράγοντας πολλαπλασιασμού αυξάνεται κάθε φορά που επαναλαμβάνεται η συνάρτηση bootstrap, δηλαδή έως ότου η τελική χρονοσειρά φτάσει το προκαθορισμένο ποσοστό επαύξησης. Τέλος, αφού χρησιμοποιηθεί αντίστροφος Box-Cox μετασχηματισμός, η νέα ανακατασκευασμένη χρονοσειρά τοποθετείται σε έναν νέο πίνακα num φορές και επιστρέφεται.

Μετά της ολοκλήρωση της συνάρτησης η τελική χρονοσειρά προκύπτει προσθέτοντας την αρχική και την ανακατασκευασμένη num φορές. Εάν η χρονοσειρά δεν έχει επαυξηθεί αρκετά, επαναλαμβάνεται η διαδικασία με μεγαλύτερο πολλαπλασιαστή mu. Έτσι τελικά προκύπτει μία νέα πυκνή χρονοσειρά, η οποία ωστόσο διατηρεί τα χαρακτηριστικά και το μοτίβο της αρχικής.

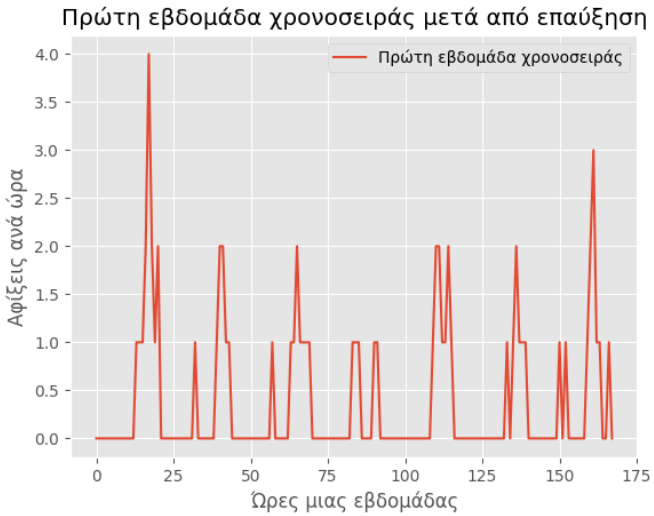
Για να εξάγουμε το τελικό προφίλ του σταθμού, επιλέγουμε ένα τυχαίο τμήμα της χρονοσειράς , μεγέθους 30%, 20%, 50% της αρχικής για τις τρεις κατηγορίες αντίστοιχα, και σχηματίζουμε τον μέσο όρο του για μία εβδομάδα. Ο λόγος που διαλέγουμε τυχαία διαστήματα είναι γιατί αν χρησιμοποιήσουμε ολόκληρη την χρονοσειρά, λόγο του μεγέθους της, θα προκύψουν παρόμοια αποτελέσματα όταν υπολογίζουμε τον μέσο όρο της και έτσι χάνεται η διαφοροποίηση μεταξύ των σταθμών. Τα ποσοστά των τυχαίων διαστημάτων διαφέρουν λόγω των διαφορετικών μεγεθών των χρονοσειρών. Σε κάθε περίπτωση για να μην χαθεί η διαφοροποίηση αλλά για να επικρατούν τα χαρακτηριστικά της χρονοσειράς, μέσω πειραματικών δοκιμών αποδείχθηκε πως χρειάζονται περίπου 5000 ώρες για προκύψει ένα προφίλ με το επιθυμητό αποτέλεσμα. Στη συνέχεια φαίνονται τα αποτελέσματα της επαύξησης για κάθε κατηγορία.

3.4.1 Κατηγορία 1: Πυκνοκατοικημένες περιοχές

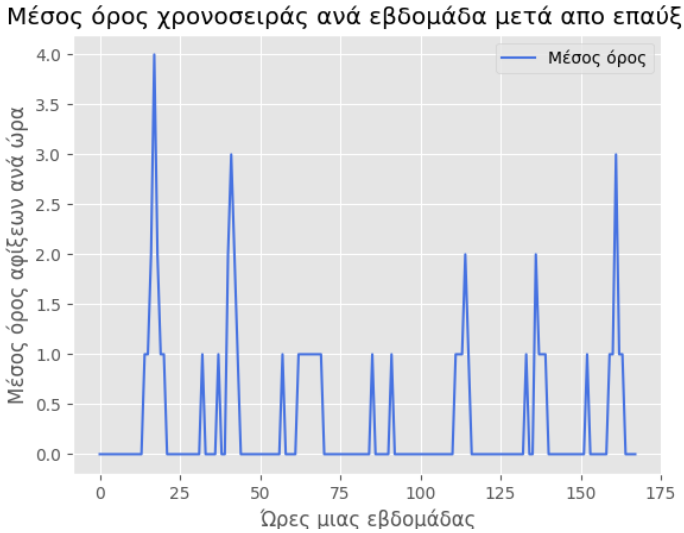
Αριστερά φαίνεται η τελική χρονοσειρά για έναν σταθμό στην πρώτη κατηγορία. Η χρονοσειρά έχει πλέον πολλές αφίξεις τη μέρα, το οποίο είναι αναμενόμενο για μια πυκνοκατοικημένη περιοχή. Δεξιά της φαίνεται μια εβδομάδα της ίδια χρονοσειράς. Τέλος φαίνεται και η μέση τιμή από την οποία φαίνεται η αύξηση των φορτίσεων το σαββατοκύριακο όπου συνήθως γίνονται πολλές έξοδοι καθώς και τη δευτέρα που έχουν προηγηθεί έξοδοι. Η μέση τιμή έχει διάρκεια μια εβδομάδα και αποτελεί το προφίλ αφίξεων του σταθμού.



Εικόνα 3.4.1.1: Αρχική χρονοσειρά αφίξεων κατηγορίας 1 μετά από επαύξηση

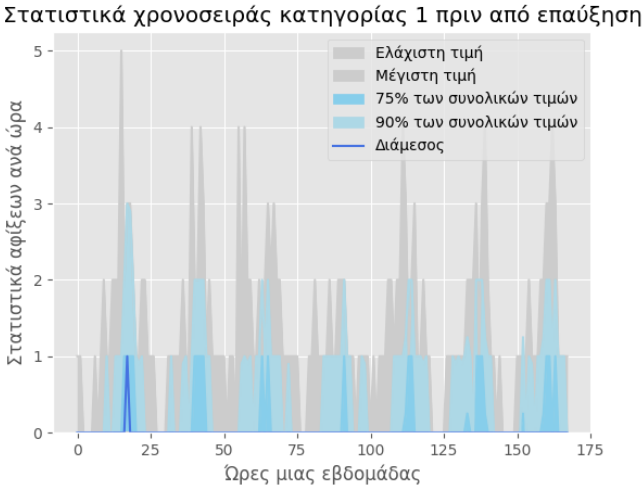


Εικόνα 3.4.1.2: Πρώτη εβδομάδα της αρχικής χρονοσειράς αφίξεων κατηγορίας 1 μετά από επαύξηση

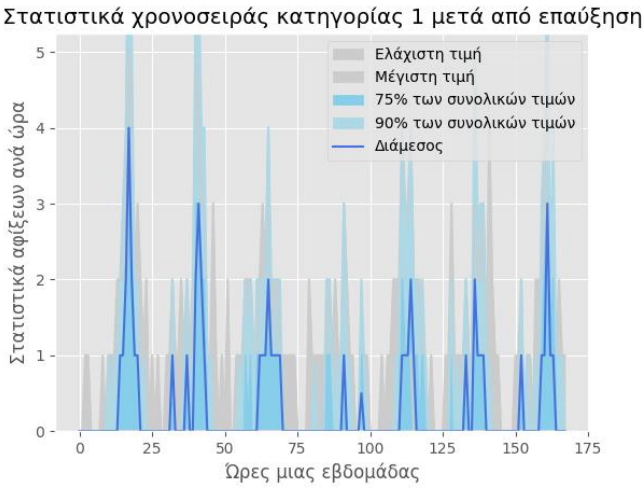


Εικόνα 3.4.1.3: Μέσος όρος χρονοσειράς κατηγορίας 1 ανά εβδομάδα μετά την επαύξηση

Παρακάτω παρουσιάζονται ορισμένα βασικά στατιστικά πριν και μετά την επαύξηση της χρονοσειράς. Με γκρι χρώμα φαίνονται οι ελάχιστες και οι μέγιστες τιμές της χρονοσειράς, με γαλάζιο φαίνεται που κυμαίνονται το 75% και 90% των τιμών της και τέλος με μπλε φαίνεται το median της. Η επαύξηση είναι ιδιαίτερα αισθητή στο διάγραμμα αυτό καθώς οι περισσότερες τιμές έχουν αυξηθεί και το median δεν είναι πλέον παντού μηδενικό. Επίσης φαίνεται πως το μοτίβο της χρονοσειράς έχει διατηρηθεί.



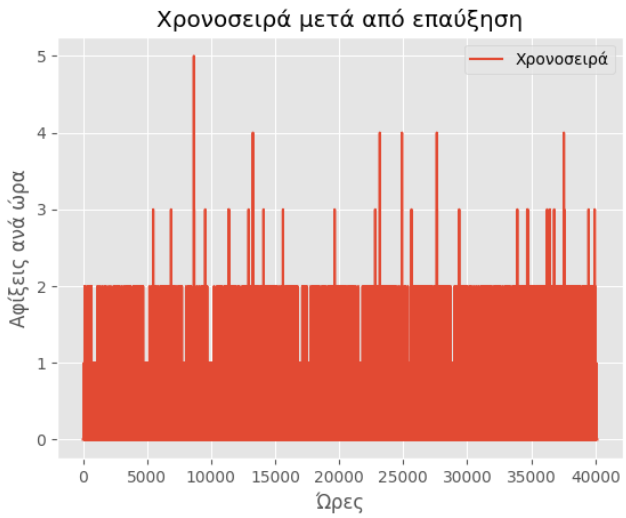
Εικόνα 3.4.1.4: Στατιστικά της χρονοσειράς κατηγορίας 1 πριν την επαύξηση τα οποία περιλαμβάνουν την μέγιστη-ελάχιστη τιμή, που κυμαίνονται το 75% και 90% των συνολικών τιμών και την διάμεσο



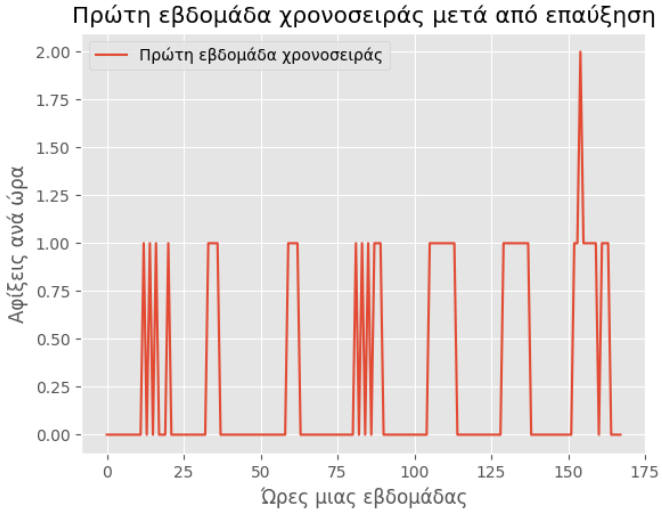
Εικόνα 3.4.1.5: Στατιστικά της χρονοσειράς κατηγορίας 1 μετά την επαύξηση τα οποία περιλαμβάνουν την μέγιστη-ελάχιστη τιμή, που κυμαίνονται το 75% και 90% των συνολικών τιμών και την διάμεσο

3.4.2 Κατηγορία 2: Λοιπές αστικές περιοχές

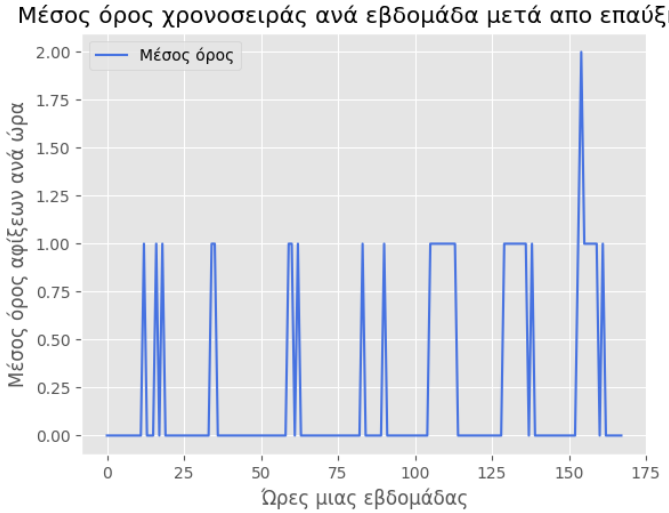
Αριστερά φαίνεται η τελική χρονοσειρά για έναν σταθμό στην δεύτερη κατηγορία. Η χρονοσειρά έχει περισσότερες αφίξεις τη μέρα, ωστόσο είναι λιγότερο επαυξημένη από μια χρονοσειρά της πρώτης κατηγορίας. Δεξιά της φαίνεται μια εβδομάδα της ίδια χρονοσειράς. Τέλος από τη μέση τιμή φαίνεται πως οι αφίξεις είναι αρκετές, αλλά πιο αραιές, και πως είναι περισσότερο συγκεντρωμένες τις μέρες παρασκευή και σαββατοκύριακο, το οποίο ταιριάζει και πάλι με την αναμενόμενη συμπεριφορά.



Εικόνα 3.4.2.1: Αρχική χρονοσειρά αφίξεων κατηγορίας 2 μετά από επαύξηση

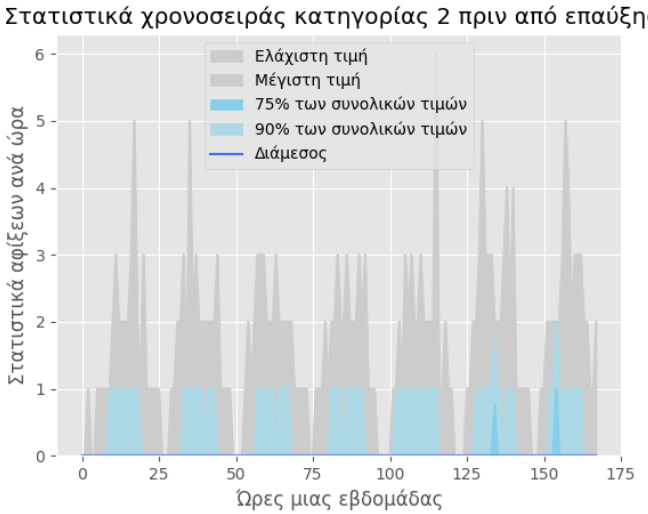


Εικόνα 3.4.2.2: Πρώτη εβδομάδα της αρχικής χρονοσειράς αφίξεων κατηγορίας 2 μετά από επαύξηση

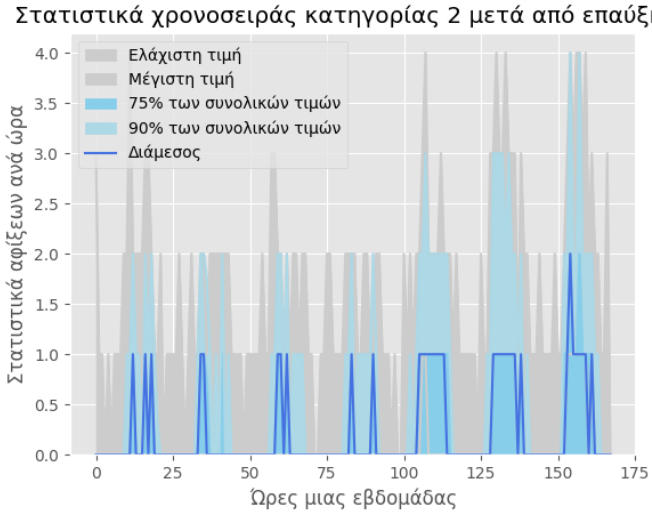


Εικόνα 3.4.2.3: Μέσος όρος χρονοσειράς κατηγορίας 2 ανά εβδομάδα μετά την επαύξηση

Ομοίως παρακάτω παρουσιάζονται ορισμένα βασικά στατιστικά πριν και μετά την επαύξηση της χρονοσειράς. Η επαύξηση είναι λιγότερο αισθητή στο διάγραμμα αυτό, ωστόσο το 90% των τιμών είναι πλέον πιο υψηλό και το median έχει αυξηθεί τις ημέρες που υπάρχει μεγαλύτερος συνωστισμός. Και πάλι το μοτίβο της χρονοσειράς έχει διατηρηθεί.



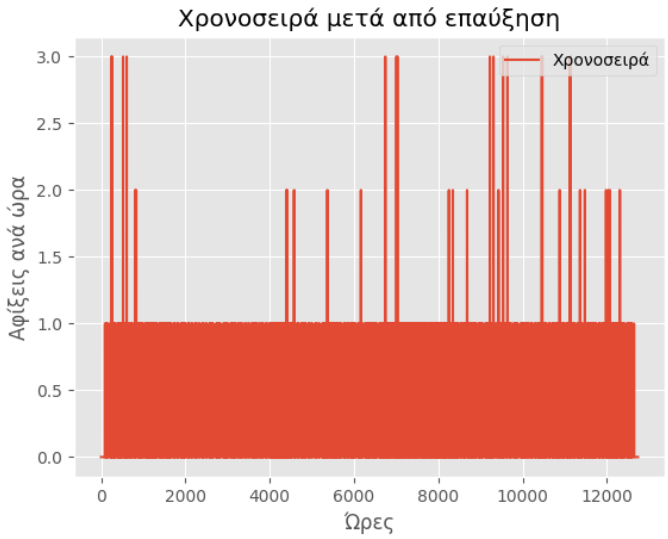
Εικόνα 3.4.2.4: Στατιστικά της χρονοσειράς κατηγορίας 2 πριν την επαύξηση τα οποία περιλαμβάνουν την μέγιστη-ελάχιστη τιμή, που κυμαίνονται το 75% και 90% των συνολικών τιμών και την διάμεσο



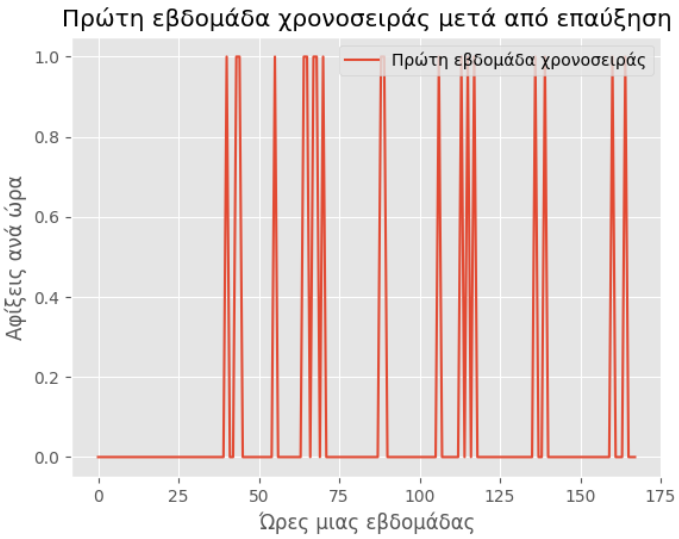
Εικόνα 3.4.2.5: Στατιστικά της χρονοσειράς κατηγορίας 1 μετά την επαύξηση τα οποία περιλαμβάνουν την μέγιστη-ελάχιστη τιμή, που κυμαίνονται το 75% και 90% των συνολικών τιμών και την διάμεσο

3.4.3 Κατηγορία 3: Αραιοκατοικημένες περιοχές

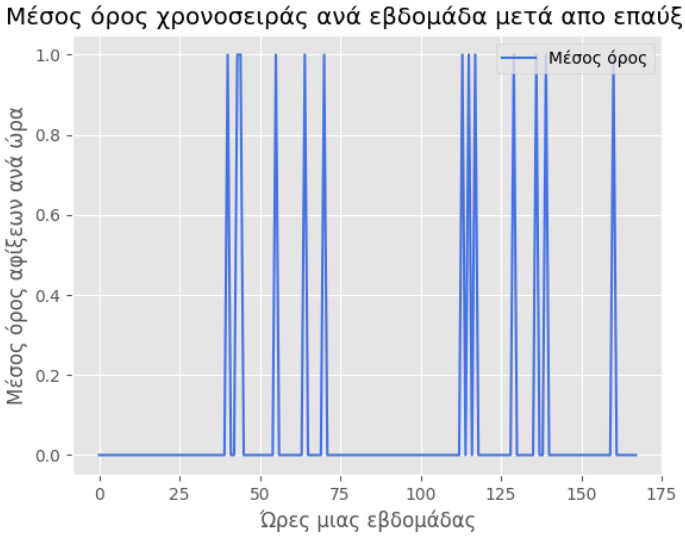
Αριστερά φαίνεται η τελική χρονοσειρά για έναν σταθμό στην τρίτη κατηγορία. Η χρονοσειρά έχει επαυξηθεί και από τη μέση τιμή φαίνεται πως οι αφίξεις είναι περισσότερες αλλά είναι ακόμα πιο αραιές και πέρα από την εποχικότητα, δεν παρατηρείται κάποια ιδιαίτερη πύκνωση σε συγκεκριμένες μέρες, λόγω των πολύ αραιών δεδομένων. Αυτό φυσικά δεν αποτελεί πρόβλημα καθώς αναφέρεται σε αραιοκατοικημένες περιοχές όπου δεν υπάρχει συνωστισμός.



Εικόνα 3.4.3.1: Αρχική χρονοσειρά αφίξεων κατηγορίας 3 μετά από επαύξηс

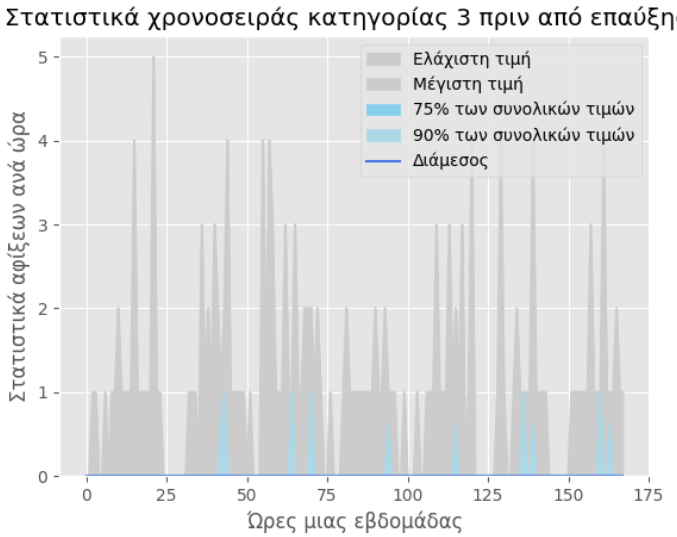


Εικόνα 3.4.3.2: Πρώτη εβδομάδα της αρχικής χρονοσειράς αφίξεων κατηγορίας 3 μετά από επαύξηση

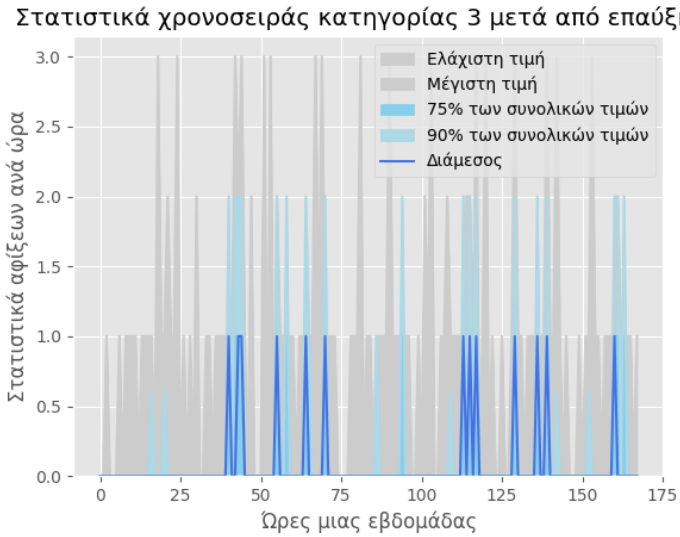


Εικόνα 3.4.3.3: Μέσος όρος χρονοσειράς κατηγορίας 3 ανά εβδομάδα μετά την επαύξηση

Τέλος παρακάτω φαίνονται ορισμένα βασικά στατιστικά πριν και μετά την επαύξηση της χρονοσειράς. Το median έχει αυξηθεί, όπως και το 90% των τιμών που πλέον έχουν υψηλότερη τιμή.



Εικόνα 3.4.3.4: Στατιστικά της χρονοσειράς κατηγορίας 3 πριν την επαύξηση τα οποία περιλαμβάνουν την μέγιστη-ελάχιστη τιμή, που κυμαίνονται το 75% και 90% των συνολικών τιμών και την διάμεσο

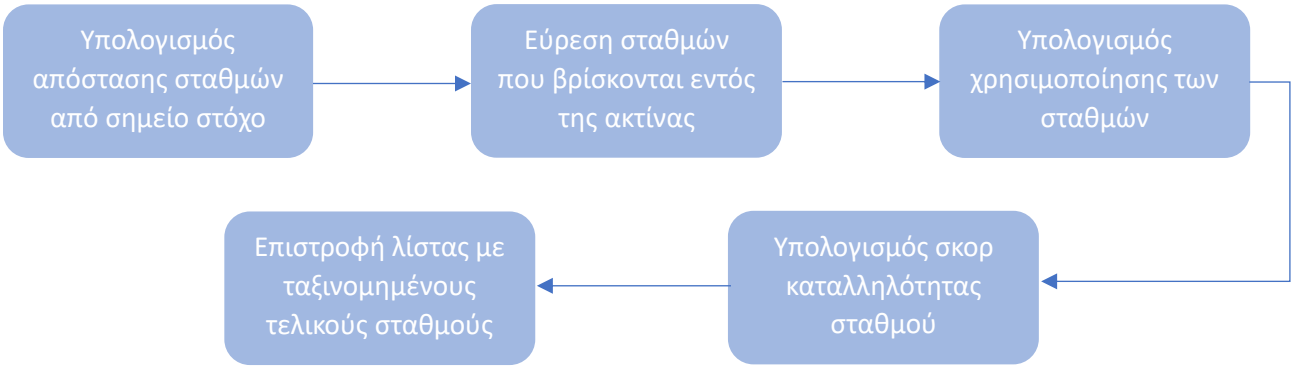


Εικόνα 3.4.3.5: Στατιστικά της χρονοσειράς κατηγορίας 3 μετά την επαύξηση τα οποία περιλαμβάνουν την μέγιστη-ελάχιστη τιμή, που κυμαίνονται το 75% και 90% των συνολικών τιμών και την διάμεσο

Κεφάλαιο 4ο. Αλγόριθμος βελτιστοποίησης

4.1 Αλγόριθμος συστάσεων

Ο αλγόριθμος συστάσεων σταθμών φόρτισης είναι το κύριο κομμάτι της εφαρμογής, καθώς αυτός θα χρησιμοποιήσει τα δεδομένα της εφαρμογής για να δώσει το τελικό αποτέλεσμα στους χρήστες. Στόχος του αλγορίθμου είναι, αφού του δοθούν οι κατάλληλες παράμετροι από τον χρήστη μέσω των φίλτρων, αυτός να του επιστρέψει τους καλύτερους σταθμούς φόρτισης που θα τον εξυπηρετήσουν. Ο αλγόριθμος αυτός είναι ένας αλγόριθμος βελτιστοποίησης της φόρτισης που χρησιμεύει στην εύρεση σταθμών με κύρια προτεραιότητα την εξυπηρέτηση του χρήστη. Συγκεκριμένα δίνει έμφαση στο να ελαχιστοποιήσει την απόσταση που θα διανύσει ο χρήστης, αφού αφήσει το όχημά του στον σταθμό φόρτισης, και παράλληλα φροντίζει ώστε ο σταθμός αυτός να έχει μεγάλη πιθανότητα να είναι διαθέσιμος της ώρα επίσκεψης. Ταυτόχρονα, ο αλγόριθμος δουλεύει και υπέρ των παρόχων των σταθμών φόρτισης, διότι ευνοεί τους σταθμούς με την μικρότερη χρήση και έτσι ισομοιράζει την κίνηση σε όλους τους σταθμούς. Με τον τρόπο αυτό αποφεύγεται ο συνωστισμός σε ορισμένους μόνο σταθμούς και έτσι βελτιώνεται η εμπειρία του χρήστη ενώ επωφελούνται περισσότεροι σταθμοί. Για να μπορεί να λειτουργήσει σωστά είναι απαραίτητο να υπάρχουν τα δεδομένα αφίξεων ώστε να μπορεί να υπολογιστεί η χρησιμοποίηση του κάθε σταθμού. Επίσης είναι σημαντικές και οι παράμετροι που θέτει ο χρήστης μέσω των φίλτρων, οι οποίες αναλύονται στη συνέχεια. Παρακάτω φαίνεται το διάγραμμα ροής με τα βήματα που ακολουθεί ο αλγόριθμος.



Εικόνα 4.1.1: Διάγραμμα ροής αλγορίθμου βελτιστοποίησης

Ο αλγόριθμος ακολουθεί την εξής διαδικασία: βρίσκει τους ταιριαστούς στον χρήστη σταθμούς ανάλογα με τα φίλτρα που έχει επιλέξει και στη συνέχεια τους βαθμολογεί με βάση την καταλληλότητά τους και τους επιστρέφει ταξινομημένους στον χρήστη. Ο αλγόριθμος είναι υλοποιημένος στο API του backend της εφαρμογής και έχει τις εξής εισόδους:

- radius: η ακτίνα χιλιομέτρων μέσα στην οποία ο χρήστης ενδιαφέρεται να βρε σταθμούς φόρτισης
- lat, long: οι συντεταγμένες του κέντρου του κύκλου στον οποίο θα βρεθούν σταθμοί
- start_time: η ώρα κατά την οποία ο χρήστης ενδιαφέρεται να επισκεφτεί τον σταθμό
- stay_hours: πόσες ώρες εκτιμάται να κάτσει ο χρήστης στον σταθμό.
- type: ο τύπος ρευματοδότη που χρειάζεται ο χρήστης να διαθέτει ο σταθμός. Η επιλογή αυτή μπορεί να είναι κενή και να μην ληφθεί υπόψη ο τύπος ρευματοδότη στην διερεύνηση σταθμών.

Η έξοδος του αλγορίθμου περιλαμβάνει μια ταξινομημένη λίστα με τους προτεινόμενους σταθμούς φόρτισης και το σκορ καταλληλότητας που προέκυψε για τον καθένα.

4.2 Μεθοδολογία Αλγορίθμου

Κατά την εκκίνηση του αλγορίθμου, πρώτο βήμα είναι η εύρεση όλων των σταθμών που βρίσκονται στην ακτίνα που έθεσε ο χρήστης. Αρχικά ζητούνται όλοι οι σταθμοί φόρτισης από τη βάση και τοποθετούνται σε μία λίστα. Σκοπός είναι να απομονωθούν οι συντεταγμένες τους και να υπολογιστεί η απόστασή τους από το κέντρο που έθεσε ο χρήστης, ώστε να παραμείνουν μόνο οι σταθμοί που βρίσκονται εντός της ακτίνας χιλιομέτρων που έχει οριστεί. Για τον λόγο αυτό ορίζεται η συνάρτηση «distance», η οποία υπολογίζει την απόσταση σε χιλιόμετρα μεταξύ δύο γεωγραφικών σημείων, χρησιμοποιώντας τον τύπο Haversine και την προσέγγιση για την ακτίνα της Γης.

Η συνάρτηση λαμβάνει τέσσερις παραμέτρους: lat1, lon1, lat2, lon2, που αναπαριστούν τις γεωγραφικές συντεταγμένες (σε μοίρες) των δύο σημείων. Αρχικά αντιστοιχίζει αυτές τις τιμές σε ακτίνια, χρησιμοποιώντας τη συνάρτηση math.radians, για να μπορέσει να χρησιμοποιήσει τριγωνομετρικές συναρτήσεις σε αυτές. Στη συνέχεια, υπολογίζει τις αποκλίσεις τους σε γεωγραφικό μήκος (dlon) και σε γεωγραφικό πλάτος (dlat). Χρησιμοποιώντας τις αποκλίσεις αυτές, υπολογίζει τον αριθμητή (a) και τον παρονομαστή (c) του τύπου Haversine, ο οποίος χρησιμοποιείται για τον υπολογισμό της απόστασης μεταξύ δύο σημείων σε μια σφαίρα. Τέλος, πολλαπλασιάζει την απόσταση με την ακτίνα της Γης (R = 6373.0) για να τη μετατρέψει σε χιλιόμετρα και επιστρέφει την απόσταση ως αποτέλεσμα της συνάρτησης. Η συνάρτηση αυτή φαίνεται παρακάτω:

```
def distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0
    lat1 = math.radians(lat1)
    lon1 = math.radians(lon1)
    lat2 = math.radians(lat2)
    lon2 = math.radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = math.sin(dlat / 2)**2 + math.cos(lat1) * math.cos(lat2) *
math.sin(dlon / 2)**2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))

    distance = R * c

    return distance
```

Αφού υπολογιστούν όλες οι αποστάσεις, στο επόμενο στάδιο συνεχίζουν οι σταθμοί που έχουν απόσταση από το κέντρο μικρότερο της ακτίνας. Για τους σταθμούς αυτούς γίνεται έλεγχος αν έχουν χώρο την ώρα που ο χρήστης έχει επιλέξει να φορτίσει το όχημά του. Ταυτόχρονα, αν έχει επιλεχτεί συγκεκριμένος τύπος ρευματοδότη, αποκλείονται όσοι σταθμοί δεν τον έχουν. Η διαδικασία αυτή φαίνεται υλοποιημένη παρακάτω:

```
    candidates = []
    dt = datetime.now()
    wd = dt.weekday()
    for station in stations:
        if
(int(station[0].mean_updating[start_time+24*wd])<int(station[0].chargers_nu
m)) and (types in station[0].type):
            candidates.append(station)
    if len(candidates) == 0:
        return 0
```

Πλέον οι σταθμοί έχουν φιλτραριστεί και απομένει να ταξινομηθούν με βάση το σκορ καταλληλότητάς τους. Για το σκορ λαμβάνονται υπόψη η χρήση του κάθε σταθμού (utilization) και η απόστασή του από το κέντρο. Η χρήση των σταθμών υπολογίζεται με βάση τις 2 προηγούμενες ώρες, ώστε να συμπεριλαμβάνονται και χρήστες που βάση προηγούμενων προτάσεων πιθανώς να έχουν επισκεφτεί τον σταθμό, καθώς κατά την δημιουργία των προφίλ επισκέψεων των σταθμών θεωρήθηκε πως κατά μέσο όρο οι χρήστες παραμένουν δύο ώρες σε αυτούς. Έτσι για τον υπολογισμό της χρησιμοποίησης προσθέτουμε τους χρήστες του σταθμού για τις δύο τελευταίες ώρες και έπειτα διαιρούμε με τον αριθμό των διαθέσιμων ρευματοδοτών επί δύο, για να προκύψει το μέσο utilization του σταθμού εκείνη τη στιγμή. Η διαδικασία αυτή φαίνεται παρακάτω:

```
 for st in candidates:
        mean = [int(i) for i in st[0].mean_updating]
        mean = pd.DataFrame(mean)
        tmp_list = (mean[wd*24:(wd+1)*24].shift(0) +
mean[wd*24:(wd+1)*24].shift(1, fill_value=0))/(int(st[0].chargers_num)*2)
        winner_st.append((tmp_list[0][start_time+24*wd],st[0]))
```

Στη συνέχεια, υπολογίζεται το συνολικό σκορ του κάθε σταθμού, δίνοντας ίση βαρύτητα στην απόσταση και στο utilization. Για τον υπολογισμό του σκορ, χρειάζεται να υπολογιστούν τα επιμέρους σκορ της χρήσης και της απόστασης του κάθε σταθμού. Επομένως αρχικά δημιουργούνται δύο ταξινομημένες λίστες util_sort και dist_sort, οι οποίες περιέχουν τους σταθμούς με τα utilizations και τις αποστάσεις αντίστοιχα. Έπειτα για κάθε πίνακα υπολογίζονται τα σχετικά σκορ, με 50 να είναι το υψηλότερο και 0 το χαμηλότερο. Το 50 δίνεται μόνο στον σταθμό με την λιγότερη χρήση ενώ όσον αφορά την απόσταση το 50 δίνεται στην μηδενική απόσταση. Για έχουμε σχετικά σκορ το 0 δίνεται πάντα στον τελευταίο σταθμό, δηλαδή σε αυτόν με την μεγαλύτερη χρήση και σε αυτόν που είναι πιο μακρινός. Τα σχετικά σκορ αποθηκεύονται στις λίστες new_util και new_dist. Τέλος, για κάθε σταθμό προστίθενται τα σκορ των δύο λιστών που αντιστοιχούν στον σταθμό αυτό και προκύπτει το τελικό σκορ. Η λίστα αυτή επιστρέφεται ταξινομημένη ως προς τα σκορ στον χρήστη ο οποίος μπορεί στη συνέχεια να επιλέξει τον σταθμό που του ταιριάζει. Με τον τρόπο αυτό όλοι οι σταθμοί έχουν βαθμολογηθεί ως προς της απόσταση και την διαθεσιμότητά τους και στη συνέχεια ο χρήστης μπορεί να δει πόσο κατάλληλος είναι ο κάθε σταθμός και να επιλέξει αντίστοιχα. Η διαδικασία αυτή φαίνεται παρακάτω:

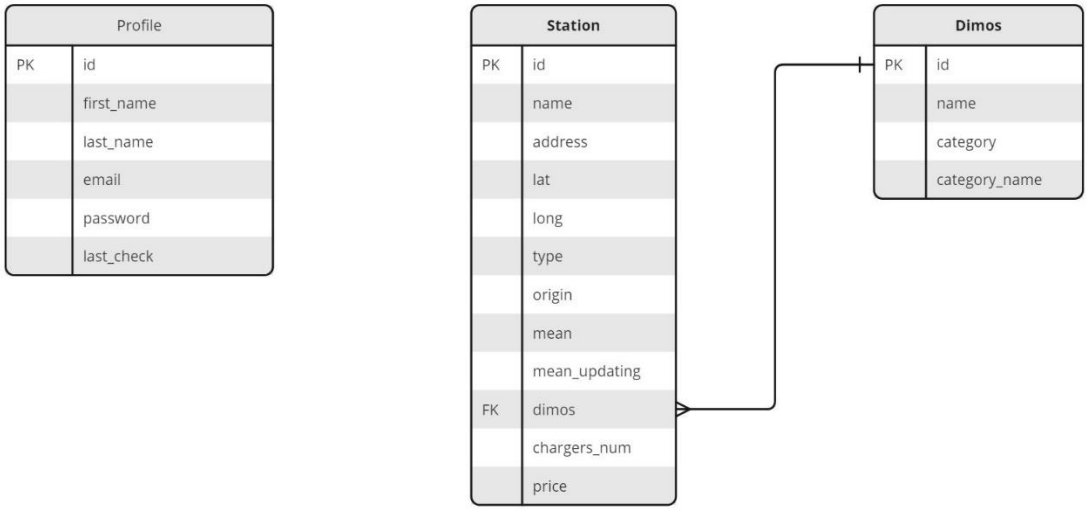
```
    util_sort = sorted(winner_st, key=lambda tup: (tup[0]))
    dist_sort = sorted(candidates, key=lambda tup: (tup[1]))
    if ((util_sort[0][0] >= 1.0) or dist_sort[0][1]>=radius):
        return 0
    util_percent = 50/(1.0 - util_sort[0][0])
    dist_percent = 50/(radius - dist_sort[0][1])
    new_util = [(50.0,util_sort[0][1])]
    new_dist = [(dist_sort[0][0],50.0)]
    dist_diff = dist_sort[0][1]
    new_dist.append((dist_sort[1][0], new_dist[0][1]-
(dist_diff*dist_percent)))
    new_dist.pop(0)

    for i in range(1,len(util_sort)):
        util_diff = util_sort[i][0] - util_sort[i-1][0]
        dist_diff = dist_sort[i][1] - dist_sort[i-1][1]
        new_util.append((new_util[i-1][0]-(util_diff*util_percent),
util_sort[i][1]))
        new_dist.append((dist_sort[i][0], new_dist[i-1][1]-
(dist_diff*dist_percent)))
    final_winner = []
    for i in range(len(new_util)):
        for j in range(len(new_dist)):
            if new_util[i][1] == new_dist[j][0]:
                final_winner.append((int(new_util[i][0] + new_dist[j][1]),
new_util[i][1]))
                break
    return sorted(final_winner, key=lambda tup: (tup[0]), reverse=True)
```

Κεφάλαιο 5ο . Βάση Δεδομένων και backend

5.1 Database

Η επιλογή της βάσης δεδομένων είναι αρκετά ευέλικτη καθώς ο όγκος των δεδομένων είναι μικρός και το μοντέλο της βάσης είναι απλό. Ωστόσο, με δεδομένο ότι το backend είναι υλοποιημένο με Django, επιλέχτηκε η βάση να είναι η PostgreSQL, καθώς προσφέρει συμβατότητα και ευκολία ενσωμάτωσης. Η PostgreSQL είναι μια open source βάση γενικής χρήσης βασισμένη σε sql. Η Django προσφέρει επιπλέον δυνατότητες όσον αφορά την βάση αυτή, όπως η εύκολη δημιουργία σύνθετων queries, το οποίο είναι ιδιαίτερα χρήσιμο στην υλοποίηση του API στη συνέχεια.

Αρχικά φαίνεται το μοντέλο οντοτήτων της βάσης, το οποίο αποτελείται από τρεις πίνακες. Ο πίνακας «Profile» χρησιμοποιείται για να αποθηκεύσει τα βασικά στοιχεία των χρηστών που εγγράφονται στην εφαρμογή. Ο πίνακας «Station» έχει τη μεγαλύτερη σημασία καθώς αποθηκεύει όλους τους σταθμούς φόρτισης. Τέλος, στον πίνακα «Dimos» συγκεντρώνονται όλοι οι δήμοι της Ελλάδας μαζί με την κατηγορία πληθυσμού στην οποία ανήκουν.



Εικόνα 5.1.1: ER διάγραμμα της βάσης δεδομένων

Από το διάγραμμα οντοτήτων φαίνονται τα πεδία του κάθε πίνακα. Τα πεδία αυτά επεξηγούνται στη συνέχεια:

Profile

- first_name: το όνομα του χρήστη
- last_name: το επώνυμο
- email: η διεύθυνση ηλεκτρονικού ταχυδρομείου
- password: ο κωδικός χρήστη
- last_check: η πιο πρόσφατη ημερομηνία και ώρα που ο χρήστης έκανε κράτηση στην εφαρμογή

Station

- name: η ονομασία του σταθμού φόρτισης
- address: η διεύθυνση
- lat: το γεωγραφικό πλάτος
- long: το γεωγραφικό μήκος
- type: οι τύποι ρευματοδοτών που είναι διαθέσιμοι από τον σταθμό
- origin: ο πάροχος του σταθμού
- mean: το προφίλ αφίξεων (μέση τιμή μιας εβδομάδας)
- mean_updating: το προφίλ αφίξεων συμπεριλαμβανομένων των αφίξεων που αποθηκεύτηκαν μέσω της εφαρμογής από τους χρήστες
- dimos: ο δήμος στον οποίο ανήκει ο σταθμός
- chargers_num: ο αριθμός ρευματοδοτών που παρέχει
- price: η τιμή φόρτισης ανά κιλοβατώρα

Dimos

- name: η ονομασία του δήμου
- category: ο αριθμός της κατηγορίας στην οποία ανήκει
- category_name: το όνομα της κατηγορίας στην οποία ανήκει

5.2 Συλλογή δεδομένων

Το πρώτο και κύριος μέρος στην συλλογή των απαραίτητων δεδομένων είναι το scrapping, δηλαδή η συλλογή δεδομένων από ιστοσελίδες με σκοπό την συλλογή σταθμών φόρτισης. Τα δεδομένα αυτά περιλαμβάνουν αρχικά τις γεωγραφικές συντεταγμένες των σταθμών φόρτισης, ώστε να μπορούν να απεικονιστούν στον χάρτη στη συνέχεια. Οι συντεταγμένες επίσης χρειάζονται για την κατηγοριοποίηση των σταθμών σε δήμους, και για τον υπολογισμό αποστάσεων στον αλγόριθμο συστάσεων. Σε ορισμένες ιστοσελίδες που έγιναν scrapped υπάρχουν έτοιμες οι συντεταγμένες ενώ σε άλλες δίνεται μόνο η διεύθυνση, η οποία μετατρέπεται στη συνέχεια σε συντεταγμένες μέσω του google API. Τα δεδομένα επίσης περιλαμβάνουν την ονομασία του σταθμού, τους τύπους φορτιστών και τις τιμές φόρτισης. Τα δεδομένα περιλαμβάνουν τις βασικές μόνο πληροφορίες, καθώς οι εφαρμογές των παρόχων και τα δεδομένα που μπορούν να αποστατούν από κάθε μία έχουν ελάχιστη συνοχή μεταξύ τους. Έτσι επιλέχτηκαν μόνο τα δεδομένα που μπορούσαν να βρεθούν σε όλες τις εφαρμογές ώστε να υπάρχει ομοιογένεια στους σταθμούς της τελικής εφαρμογής.

Για το μέρος αυτό χρησιμοποιήθηκε το Scrapy, ένα open source framework για web scrapping και crawling, δηλαδή για συλλογή πληροφοριών σε ιστοσελίδες. Το framework αυτό αποτελεί κατάλληλη επιλογή καθώς είναι βασισμένο στην αρχιτεκτονική του Django και είναι πολύ εύκολο να ενσωματωθεί σε αυτό. Συγκεκριμένα με τον συνδυασμό των δύο αυτών εργαλείων, τα pipelines συνδέονται με τη βάση μέσω του Django, με αποτέλεσμα να υπάρχει ελεύθερη πρόσβαση στα δεδομένα και στα ειδικά Django queries από το Scrapy. Ταυτόχρονα από το Django είναι δυνατή η υλοποίηση και η χρονοδρομολόγηση scrapping εντολών.

Όσον αφορά το scrapping των δεδομένων το scrapy ακολουθεί την εξής διαδικασία: Αρχικά κάνει ένα HTTP request στο URL της ιστοσελίδας στόχου και αναλύει την HTML ή την XML που έλαβε. Σε περίπτωση που η ιστοσελίδα δεν επιτρέπει τους crawlers, γίνεται να καθοριστεί ο user-agent που θα χρησιμοποιηθεί. Στη συνέχεια χρησιμοποιώντας ειδικούς selectors, επιλέγει και εξάγει τα δεδομένα που του ζητήθηκαν. Έπειτα γίνεται η κατάλληλη επεξεργασία τους και τέλος στέλνονται στη βάση μέσω των pipelines. Η διαδικασία φαίνεται παρακάτω για την ιστοσελίδα του παρόχου Chargespot.

```
class ChargeSpot(scrapy.Spider):
    name = "ChargeSpot"

    start_urls = [
      "https://chargespot.gr/simeia-fortisis/"
    ]

    download_delay = 0.1
```

```
def parse(self, response):
    # define pipeline
    items = StationItem()
    stations = response.css('.col-lg-3')
    for station in stations:
        # data extraction with css selector
        info = station.css('::text').extract()
        coords = station.css('a::attr(href)').extract()
        coords = coords[0].split("/")[-1]
        lat = coords.split(",")[0]
        long = coords.split(",")[1]
        name = info[0]
        address = info[1]
        if len(info) < 4:
          type = info[2]
        else:
          type = info[3]

        # data filtering
        form_type = type.split("x")[1]
        kw = form_type[:2]
        if 'CCS' in form_type:
            items['type'] = 'CCS Type 2 ' + kw + ' KW'
        else:
            items['type'] = 'AC Type 2 ' + kw + ' KW'
        form_type = form_type.split("socket")[0]

        dhmos = add_category(lat,long)
        if dhmos[0]!=0:
            items['dhmos'] = Dhmos.objects.get(name=dhmos[0])

        # sending data to pipeline
        items['chargers_num'] = type[0]
        items['name'] = name
        items['address'] = address
        items['lat'] = lat
        items['long'] = long
        items['origin'] = "ChargeSpot"
        yield items
```

Σε ορισμένες περιπτώσεις που το επέτρεπε η ιστοσελίδα, το scrapping έγινε στέλνοντας request στο API της ιστοσελίδας με κατάλληλους user-agent και headers και έπειτα επιλέγοντας τα

δεδομένα που χρειάζονται από το αντίστοιχο response. Στην συνέχεια τα δεδομένα επιστρέφονται στο αντίστοιχο spider και αποθηκεύονται μέσω του pipeline.

Οι ιστοσελίδες οι χρησιμοποιήθηκαν για scrapping των σταθμών είναι οι Chargespot, Fortisis, ProtergiaCharge, PlugShare, ElpeFuture και Blink Charging.

Κατά τη διάρκεια του scrapping, οι σταθμοί φόρτισης αντιστοιχούνται στον δήμο όπου ανήκουν με βάση τις συντεταγμένες τους. Αυτό είναι δυνατό γιατί για κάθε δήμο υπάρχει το αντίστοιχο γεωγραφικό του πολύγωνο, το οποίο μπορεί να χρησιμοποιηθεί για να βρεθούν οι σταθμοί που περικλείει. Το σύνολο των δήμων καθώς και τα γεωγραφικά του πολύγωνα παρέχονται από το ΕΛΣΤΑΤ [39]. Ο κώδικας που επιτελεί αυτήν την διαδικασία φαίνεται παρακάτω.

```
def add_category(lat,long):
    # load GeoJSON file containing sectors
    with open('...path\\dhmoi.geojson', encoding = 'utf-8') as f:
        js = json.load(f)

    organized_dhmoi = data = pd.read_csv('path\\organized_dhmoi.csv')
    # construct point based on lon/lat returned by geocoder
    point = Point(float(long),float(lat))
    # check each polygon to see if it contains the point
    for feature in js['features']:
        polygon = shape(feature['geometry'])
        if polygon.contains(point):
            name = feature['properties']["LEKTIKO"]
            # ΔΗΜΟΣ
            name = name.replace("ΔΗΜΟΣ ","")
            # spaces
            name = name.replace(" - ","-")
            # ' -> A
            name = name.replace("'","A")
            tmp = 0
            for i in range(0,organized_dhmoi.shape[0]):
                dhmos = organized_dhmoi[data.columns[1]][i]
                if dhmos==name:
                    return [dhmos,organized_dhmoi[data.columns[3]][i]]
                    tmp = 1
            if tmp==0:
                res = add_manually(name)
                if res[0]!=0:
                    return res
                else:
                    return [0,0]
    return [0,0]
```

5.3 Backend

Για την υλοποίηση του backend χρησιμοποιήθηκε το Django, ένα open-source framework σχεδιασμένο για την ανάπτυξη διαδικτυακών εφαρμογών. Η επιλογή αυτή έγινε καθώς το framework αυτό προσφέρει τα απαραίτητα εργαλεία για τον εύκολο χειρισμό της βάσης δεδομένων και του API.

Πρώτο βήμα στην ανάπτυξη του backend είναι η σύνδεση με τη βάση και η εγκατάσταση των βασικών βιβλιοθηκών, τα οποία ρυθμίζονται στο αρχείο setting.py. Εκεί ορίστηκαν και εργαλεία που θα χρησιμοποιηθούν για το middleware και για τα cors headers ώστε να μπορούν να σταλθούν με επιτυχία στη συνέχεια τα requests στο API.

Στη συνέχεια καθορίστηκαν τα μοντέλα των δεδομένων όπως αναγράφονται στο ERD παραπάνω. Μαζί με τα μοντέλα ορίστηκαν και οι αντίστοιχοι serializers, οι οποίοι φέρνουν τα δεδομένα στην σωστή μορφή για να σταλθούν όταν ζητηθούν σε αντίστοιχο API request. Και τα δύο αρχεία φαίνονται παρακάτω.

```
from django.db import models

class Station(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=200)
    address = models.CharField(max_length=200,null=True)
    lat = models.CharField(max_length=50)
    long = models.CharField(max_length=50)
    type = models.CharField(max_length=500,null=True)
    origin = models.CharField(max_length=50)
    mean = models.CharField(max_length=200,null=True)
    mean_updating = models.CharField(max_length=200,null=True)
    chargers_num = models.CharField(max_length=50,null=True)
    dhmos = models.ForeignKey("Dhmos", on_delete=models.CASCADE)
    price = models.CharField(max_length=10, null=True)

class Dhmos(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=200)
    category = models.CharField(max_length=50,null=True)
    category_name = models.CharField(max_length=200,null=True)

class Profile(models.Model):
    id = models.AutoField(primary_key=True)
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    email = models.CharField(max_length=200)
```

```
    password = models.CharField(max_length=200)
    last_check = models.DateTimeField(null=True)
```

```
from rest_framework import serializers

from charging_stations.models import Station, Profile

class StationSerializer(serializers.ModelSerializer):
   class Meta:
       model = Station
       fields = ('id', 'name', 'address', 'lat', 'long', 'type', 'origin',
'dhmos',
        'mean', 'mean_updating', 'chargers_num', 'price')

class ProfileSerializer(serializers.ModelSerializer):
   class Meta:
       model = Profile
       fields = ('id', 'first_name', 'last_name', 'email', 'password',
'last_check')
```

Έπειτα δημιουργήθηκε το API της εφαρμογής στο αρχείο views.py ώστε να έχει πρόσβαση το frontend στις λειτουργίες και στα δεδομένα του backend. Το API χωρίζεται σε δύο κλάσεις, την StationsView για τα requests που αφορούν τους σταθμούς φόρτισης και την ProfilesView για τα requests που αφορούν τους χρήστες. Για τις δύο αυτές κλάσεις δημιουργήθηκαν τα κατάλληλα routes στο urls.py όπως φαίνεται παρακάτω.

```
from django.contrib import admin
from django.urls import path, include
from rest_framework import routers
from charging_stations.views import StationsView, ProfilesView

router = routers.DefaultRouter()
router.register(r'stations', StationsView, basename="stations")
router.register(r'profiles', ProfilesView, basename="profiles")

urlpatterns = [
   path('', include(router.urls)),
]
```

Το API έχει επτά βασικές συναρτήσεις για να επικοινωνεί με το frontend, οι οποίες περιγράφονται αναλυτικά στη συνέχεια.

Η get_dhm_geojson επιστρέφει ένα json με την γεωγραφική περιγραφή των δήμων της Ελλάδας ώστε να μπορούν να αναπαρασταθούν στον χάρτη.

Η get_stations επιστρέφει όλους τους σταθμούς φόρτισης ώστε να τοποθετηθούν έπειτα στον χάρτη.

Η get_radius έχει ως παραμέτρους τα φίλτρα που επέλεξε ο χρήστης στο frontend και με βάση αυτά διαλέγει τους σταθμούς που πληρούν τις προϋποθέσεις. Στη συνέχει τους εφαρμόζει τον αλγόριθμο βελτιστοποίησης και επιστρέφει τους τελικούς σταθμούς στο frontend ταξινομημένους από τον καλύτερο για τον χρήστη μέχρι τον χειρότερο.

Η add_arrival αποσκοπεί στο να κρατήσει την επιλογή του χρήστη να επισκεφτεί κάποιον σταθμό τις ορισμένες από αυτόν ώρες. Έτσι αρχικά ελέγχει αν αυτό είναι πραγματοποιήσιμο και αν ο σταθμός είναι διαθέσιμος και στη συνέχεια αν είναι ενημερώνει την βάση δεδομένων για την αλλαγή αυτή.

Η get_profile χρησιμοποιείται στο sign in και έτσι ελέγχει τον κωδικό και το email που της δόθηκαν σαν παραμέτρους και δίνει την αντίστοιχη απάντηση αν τα στοιχεία αυτά υπάρχουν στη βάση δεδομένων ή όχι.

Η add_profile χρησιμοποιείται στο sign up και καταχωρεί έναν νέο χρήστη στη βάση δεδομένων.

Η get_check_permission χρησιμοποιείται να για δώσει άδεια στον χρήστη να κάνει κράτηση ενός σταθμού. Σκοπός της είναι να σταματήσει τους χρήστες από το να κάνουν πολλαπλές κρατήσεις σε σύντομο χρονικό διάστημα και να γεμίζουν όλους τους σταθμούς.

Κεφάλαιο 6ο . Διεπαφή Χρήστη

6.1 Διεπαφή Χρήστη

Προτού αρχίσει η σχεδίαση της διεπαφής χρήστη, είναι σημαντικό να ληφθεί υπόψη η εμπειρία του χρήστη και πώς θα αλληλεπιδρά με την εφαρμογή. Για να μπορούν οι χρήστες να έχουν μια ικανοποιητική εμπειρία, η διεπαφή πρέπει να είναι ευανάγνωστη και εύκολη στη χρήση. Για τον σκοπό αυτό έχουν καθιερωθεί οι αρχές σχεδιασμού μιας διεπαφής χρήστη, οι οποίες είναι οι εξής:

1. Απλότητα: Η διεπαφή πρέπει να είναι απλή και ευανάγνωστη, με ελάχιστα και κατανοητά στοιχεία. Ο σχεδιασμός πρέπει να αποφεύγει τα περιττά στοιχεία και να προσφέρει μια ομαλή ροή χρήσης.

2. Ομοιομορφία: Ο σχεδιασμός πρέπει να είναι συνεπής σε όλη τη διεπαφή και ανάμεσα σε διάφορες σελίδες ή ενότητες. Οι ίδιες ενέργειες πρέπει να έχουν την ίδια εμφάνιση και λειτουργικότητα για να παρέχεται συνοχή και ευκολία στην πλοήγηση.

3. Ολοκληρωμένη ανταπόκριση: Η διεπαφή πρέπει να ανταποκρίνεται αμέσως στις ενέργειες του χρήστη και να παρέχει οπτική ανατροφοδότηση. Αυτό μπορεί να περιλαμβάνει αλλαγές χρωμάτων, κινήσεις ή μηνύματα επιβεβαίωσης για να ενισχυθεί η αίσθηση αλληλεπίδρασης.

4. Ευκολία πλοήγησης: Η διεπαφή πρέπει να παρέχει ευκολία στην πλοήγηση και στον εντοπισμό πληροφοριών. Κατάλληλα μενού, σύνδεσμοι και ενδείξεις πρέπει να βοηθούν τον χρήστη να περιηγηθεί με άνεση και να εντοπίσει τις επιθυμητές λειτουργίες ή πληροφορίες.

5. Προσαρμοστικότητα: Η διεπαφή πρέπει να είναι προσαρμόσιμη σε διάφορες συσκευές και μεγέθη οθονών, ώστε να παρέχεται μια ομοιόμορφη εμπειρία σε όλες τις πλατφόρμες.

Ένα χρήσιμο εργαλείο για την σχεδίαση επαφής που να ακολουθεί τις αρχές αυτές είναι το Material Design, ένας σχεδιαστικός οδηγός που αναπτύχθηκε από την Google και προσφέρει αρχές και κατευθυντήριες γραμμές για τον σχεδιασμό διεπαφών χρήστη. Οι αρχές του Material Design περιλαμβάνουν:

- Υλικό: Η διεπαφή πρέπει να μιμείται τον υλικό κόσμο, χρησιμοποιώντας πραγματικούς και οπτικά πλούσιους υλικούς και αποτυπώματα. Ο σχεδιασμός πρέπει να προσφέρει πιθανολογούμενες και ευαίσθητες απαντήσεις στις ενέργειες του χρήστη.
- Κίνηση: Η κίνηση παίζει σημαντικό ρόλο στον σχεδιασμό της διεπαφής. Οι κινούμενες μεταβάσεις, οι αλληλεπιδράσεις και οι μετατροπές πρέπει να είναι ομαλές, φυσικές και να προσφέρουν οπτική ανταπόκριση.
- Υπευθυνότητα: Η διεπαφή πρέπει να είναι προσαρμόσιμη και να ανταποκρίνεται σε διάφορες συσκευές και μεγέθη οθονών. Πρέπει να παρέχει μια συνεκτική εμπειρία στους χρήστες ανεξάρτητα από τη συσκευή που χρησιμοποιούν.

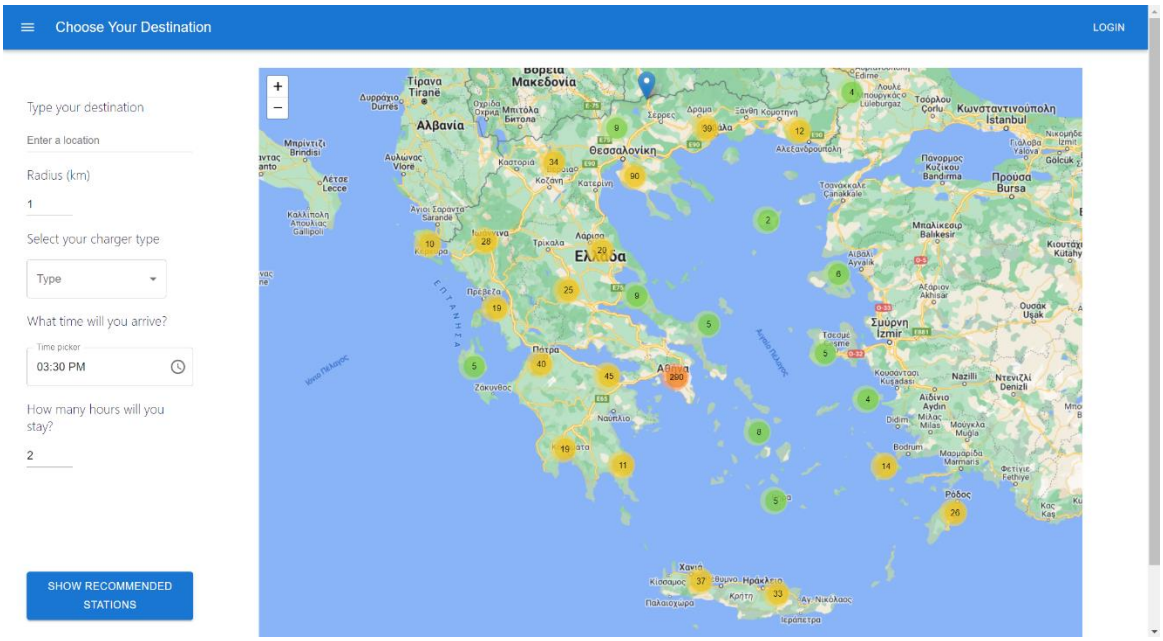
- Αισθητική: Ο σχεδιασμός πρέπει να είναι αισθητικά ελκυστικός και να προσφέρει ευχάριστη εμπειρία στον χρήστη. Πρέπει να δίνει έμφαση στην καθαρότητα, την απλότητα και την ευανάγνωστη διεπαφή.

Με τις αρχές αυτές είναι δυνατή η δημιουργία μιας οπτικά ελκυστικής, λειτουργικής και ευχάριστης διεπαφής χρήστη που παρέχει μια ενιαία εμπειρία χρήσης σε διάφορες πλατφόρμες και συσκευές.

Η διεπαφή χρήστη υλοποιήθηκε επίσης με την χρήση του ReactJS, μία open-source βιβλιοθήκη γραμμένη σε JavaScript η οποία έχει δημιουργηθεί από την Facebook. Είναι γνωστή για την εύκολη και αποδοτική υλοποίηση δυναμικών στοιχείων διεπαφής χρήστη.

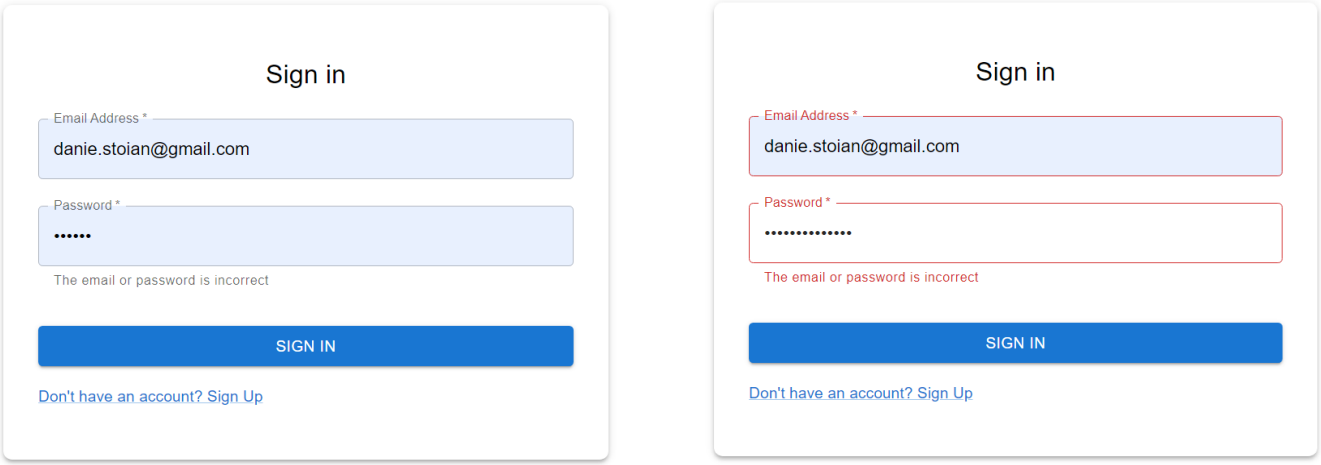
6.2 Σχεδίαση Διεπαφής (UI)

Με την είσοδο στην εφαρμογή φαίνεται αρχικά η κεντρική σελίδα. Αριστερά περιέχει τα φίλτρα που μπορεί να επιλέξει ο χρήστης ώστε να του φανερωθούν οι προτεινόμενοι σταθμοί φόρτισης, ενώ στην υπόλοιπη οθόνη φαίνεται ο χάρτης με όλους τους σταθμούς τοποθετημένους. Πάνω δεξιά ο χρήστης μπορεί να κάνει login.



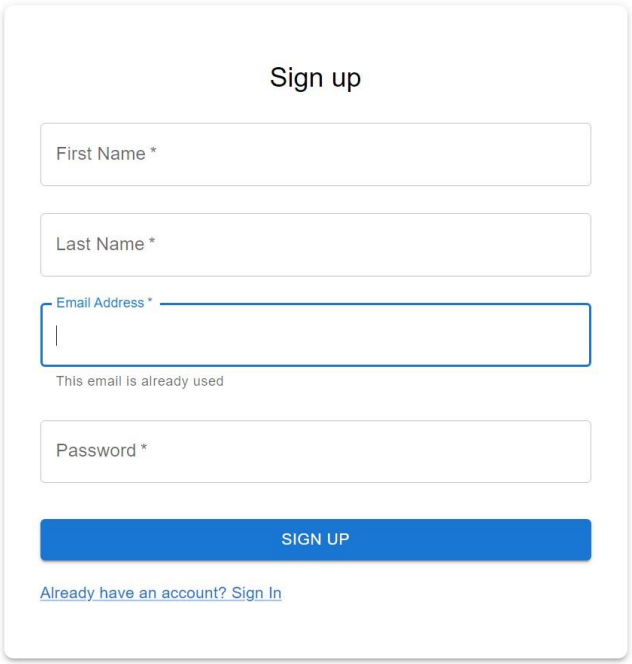
Εικόνα 6.2.1: Κεντρική οθόνη διεπαφής

Πατώντας το login εμφανίζεται η παρακάτω σελίδα, στην οποία ο χρήστης εισάγει τα στοιχεία του και συνδέεται με τον λογαριασμό του στην εφαρμογή. Τα πεδία έχουν αυτόματη συμπλήρωση και είναι υποχρεωτικά για να ολοκληρωθεί η είσοδος του χρήστη. Πατώντας sign in με λάθος στοιχεία τα πεδία κοκκινίζουν και εμφανίζεται κατάλληλο μήνυμα σφάλματος.



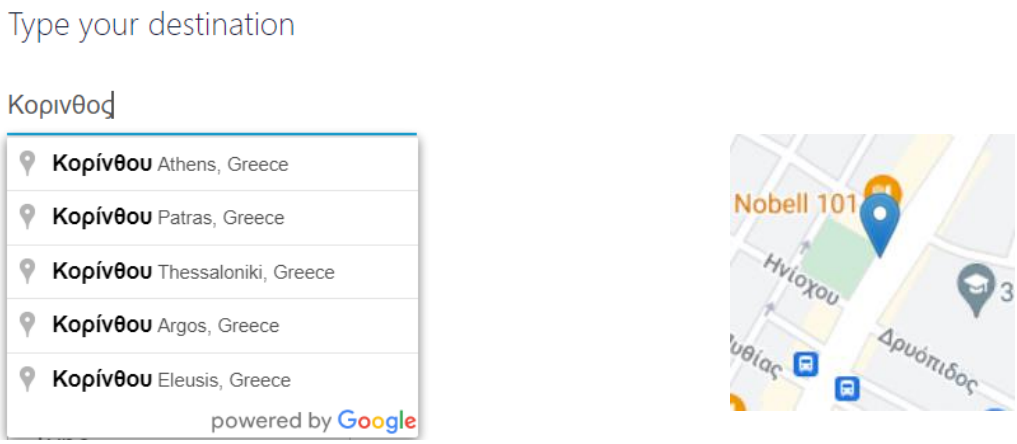
Εικόνα 6.2.2: Σελίδα σύνδεσης χρήστη

Εάν ο χρήστης δεν έχει δημιουργήσει λογαριασμό στην εφαρμογή, μπορεί να το κάνει πατώντας sign up. Στη συνέχεια εμφανίζεται η παρακάτω σελίδα, όπου ο χρήστης χρειάζεται να συμπληρώσει τα βασικά του στοιχεία, δηλαδή το ονοματεπώνυμό του, την διεύθυνση ηλεκτρονικού ταχυδρομείου και τον κωδικό του. Τα πεδία είναι όλα υποχρεωτικά και το πεδίο email απαιτείται να έχει την κατάλληλη μορφή. Μόλις ο χρήστης συμπληρώσει όλα τα πεδία και πατήσει sign up, τα στοιχεία του αποθηκεύονται στην βάση δεδομένων μέσω του API, και στη συνέχεια ο χρήστης ανακατευθύνεται στην σελίδα sign in όπου μπορεί να συνδεθεί με τα στοιχεία του.



Εικόνα 6.2.3: Σελίδα εγγραφής χρήστη

Αφού ο χρήστης κάνει sign in εμφανίζεται και πάλι η κεντρική σελίδα. Για να μπορέσει ο χρήστης να δει τους προτεινόμενους σταθμούς χρειάζεται να προσδιορίσει τα φίλτρα. Αρχικά ο χρήστης πρέπει να πληκτρολογήσει την διεύθυνση στην οποία θέλει να ψάξει για σταθμούς, η εναλλακτικά να επιλέξει το σημείο στον χάρτη, όπου θα εμφανιστεί ένας μπλε δείκτης. Στην συμπλήρωση της διεύθυνσης χρησιμοποιείται το API του google maps ώστε μέσω της διεύθυνσης να επιστρέφονται οι συντεταγμένες της.

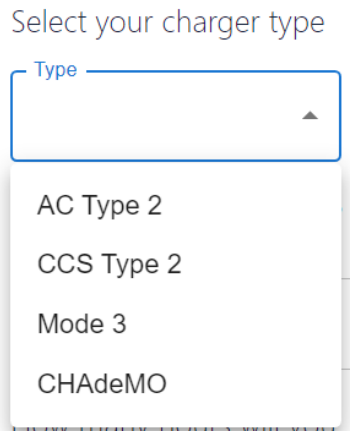


Εικόνα 6.2.4: Παράδειγμα επιλογής προορισμού με δύο τρόπους

Στη συνέχεια ο χρήστης επιλέγει την ακτίνα σε χιλιόμετρα μέσα στην οποία ενδιαφέρεται να βρει σταθμούς φόρτισης. Η μέγιστη επιλογή είναι 20km και η ελάχιστη είναι 1km.

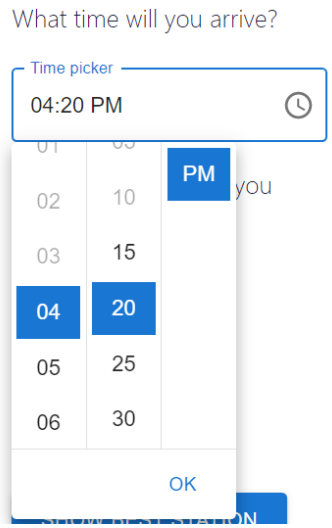
Εικόνα 6.2.5: Παράδειγμα επιλογής ακτίνας χιλιομέτρων

Έπειτα επιλέγει τον τύπο ρευματοδότη που ταιριάζει σε αυτόν. Οι διαθέσιμες επιλογές φαίνονται παρακάτω. Αν η επιλογή παραμείνει κενή, επιστρέφονται όλοι οι σταθμοί ανεξαρτήτως του τύπου ρευματοδότη.



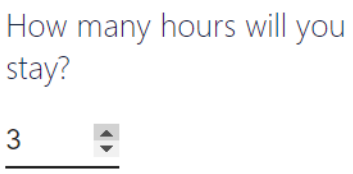
Εικόνα 6.2.6: Παράδειγμα επιλογής τύπου φορτιστή

Ο χρήστης πρέπει επίσης να επιλέξει την ώρα που τον ενδιαφέρει να φτάσει στον σταθμό, ώστε να ελεγχθεί η διαθεσιμότητά του. Οι περασμένες ώρες είναι μη επιλέξιμες και έχουν γκρι χρώμα ώστε να διακρίνονται πιο εύκολα.



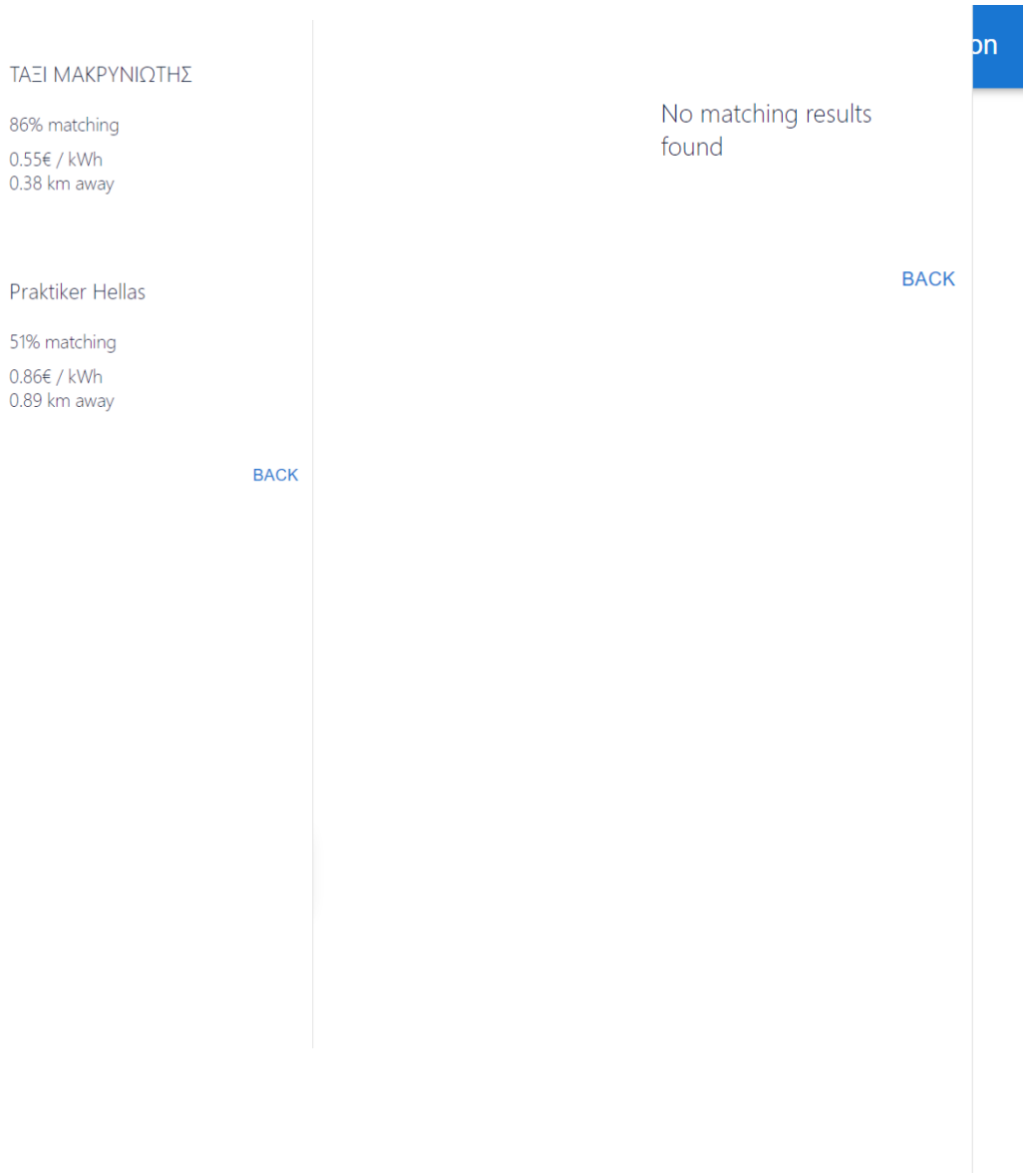
Εικόνα 6.2.7: Παράδειγμα επιλογής ώρας άφιξης στον σταθμό

Τέλος ο χρήστης επιλέγει πόσες ώρες επιθυμεί να παραμείνει στον σταθμό. Με αυτόν τον τρόπο στη συνέχεια μπορεί να κάνει κράτηση και το χρονικό διάστημα που θα παραμείνει αποθηκεύεται στη βάση δεδομένων και λαμβάνεται υπόψη στον επόμενο χρήστη που θα επιλέξει τον ίδιο σταθμό.



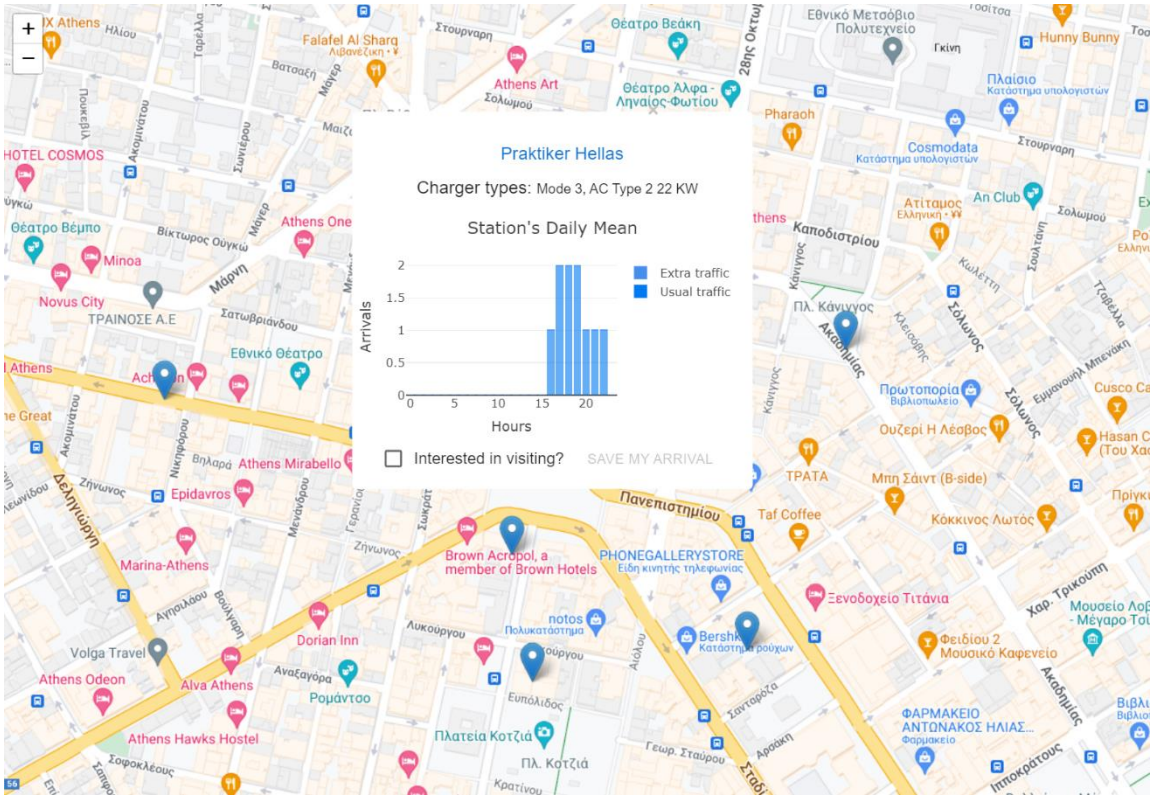
Εικόνα 6.2.8: Παράδειγμα επιλογής ωρών παραμονής στον σταθμό

Αφού ο χρήστης έχει επιλέξει τα κατάλληλα φίλτρα και έχει πατήσει το κουμπί «show recommended stations», εμφανίζονται όλες οι επιλογές που του προτείνει ο αλγόριθμος στο αριστερό τμήμα της οθόνης, σε ένα drawer. Οι σταθμοί εμφανίζονται ταξινομημένοι με βάση το σκορ καταλληλότητας που προέκυψε από τον αλγόριθμο συστάσεων. Αν δεν βρεθεί κανένας σταθμός που να τηρεί τις προϋποθέσεις των φίλτρων, εμφανίζεται το κατάλληλο μήνυμα, όπως φαίνεται παρακάτω.



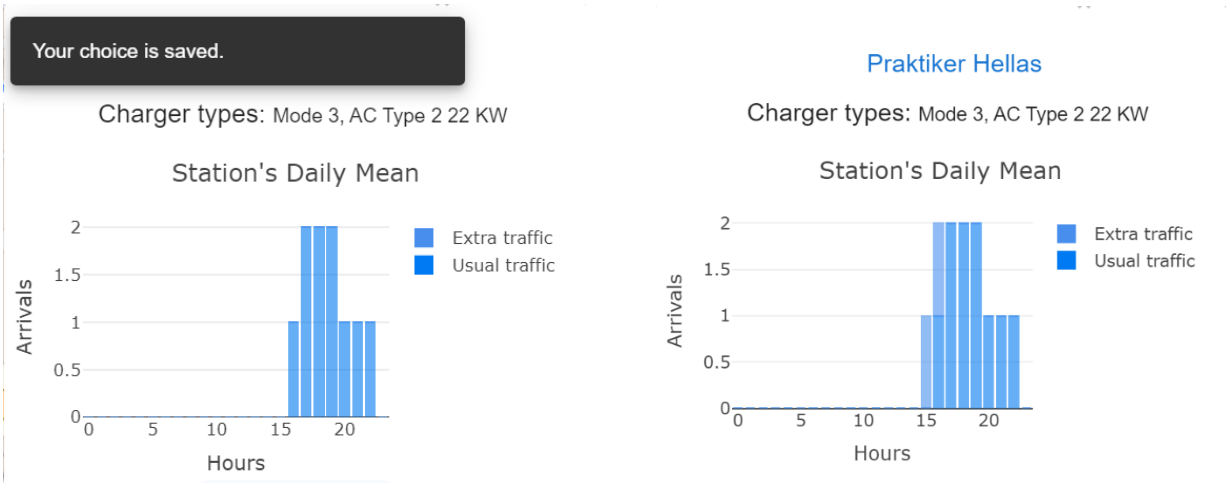
Εικόνα 6.2.9: Εμφάνιση αποτελεσμάτων όταν βρέθηκαν και όταν δεν βρέθηκαν σταθμοί φόρτισης

Αν ο χρήστης επιλέξει κάποιον από τους προτεινόμενους σταθμούς, το drawer κλείνει και ο χάρτης εστιάζει στον σταθμό που επιλέχθηκε. Αφού ο χρήστης πατήσει στον δείκτη του σταθμού πάνω στον χάρτη, εμφανίζεται ένα popup με τις πληροφορίες του σταθμού αυτού. Οι πληροφορίες περιλαμβάνουν την ονομασία του σταθμού, τους τύπους ρευματοδότη που διαθέτει και το διάγραμμα κίνησης που έχει συνήθως ο σταθμός αυτός την αντίστοιχη μέρα, μαζί με τις κρατήσεις που έχουν γίνει από άλλους χρήστες.



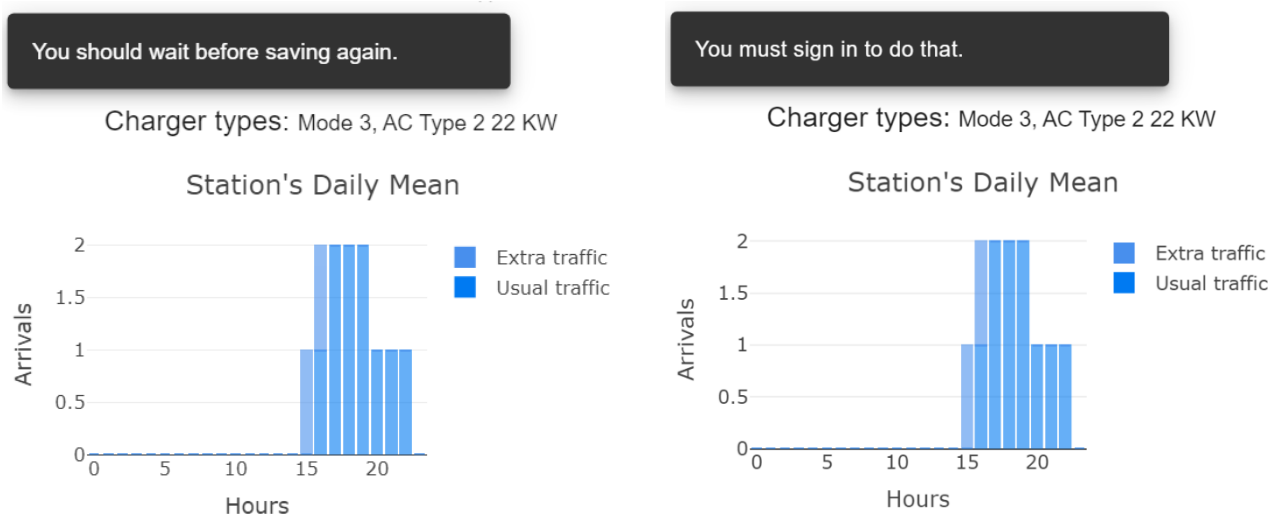
Εικόνα 6.2.10: Διαθέσιμες πληροφορίες του σταθμού

Αν ο χρήστης πατήσει το κουμπί «save my arrival», και έχει ήδη συνδεθεί με τον λογαριασμό του, η επιλογή του αποθηκεύεται, εμφανίζοντας το αντίστοιχο μήνυμα. Στην συνέχεια αν κάποιος χρήστης επιλέξει τον ίδιο σταθμό, η νέα προσθήκη εμφανίζεται στο διάγραμμα του σταθμού και λαμβάνεται υπόψη από τον αλγόριθμο.



Εικόνα 6.2.11: Εμφάνιση μνήματος αποθήκευσης της κράτησης και αλλαγή διαγράμματος σταθμού με την νέα κράτηση

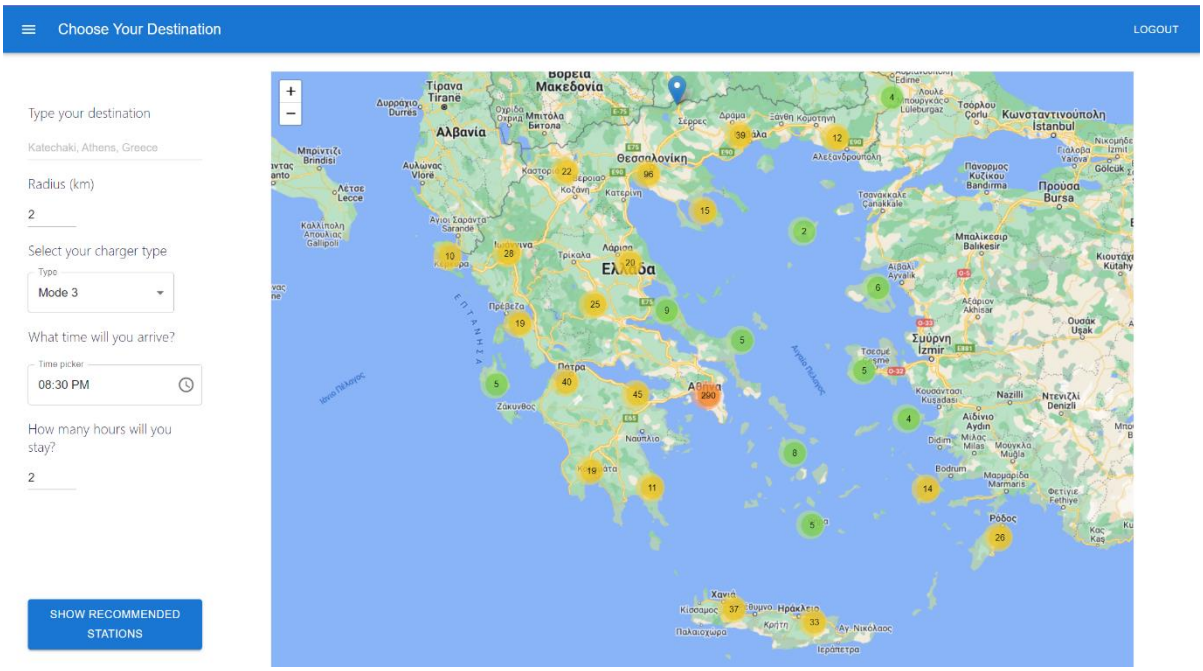
Αν ο χρήστης δεν έχει συνδεθεί με τον λογαριασμό του ή προσπαθήσει να αποθηκεύσει την επιλογή του ξανά σε μικρό χρονικό διάστημα, τότε εμφανίζεται το αντίστοιχο μήνυμα σφάλματος.



Εικόνα 6.2.12: Εμφάνιση μηνυμάτων σφάλματος κατά την κράτηση

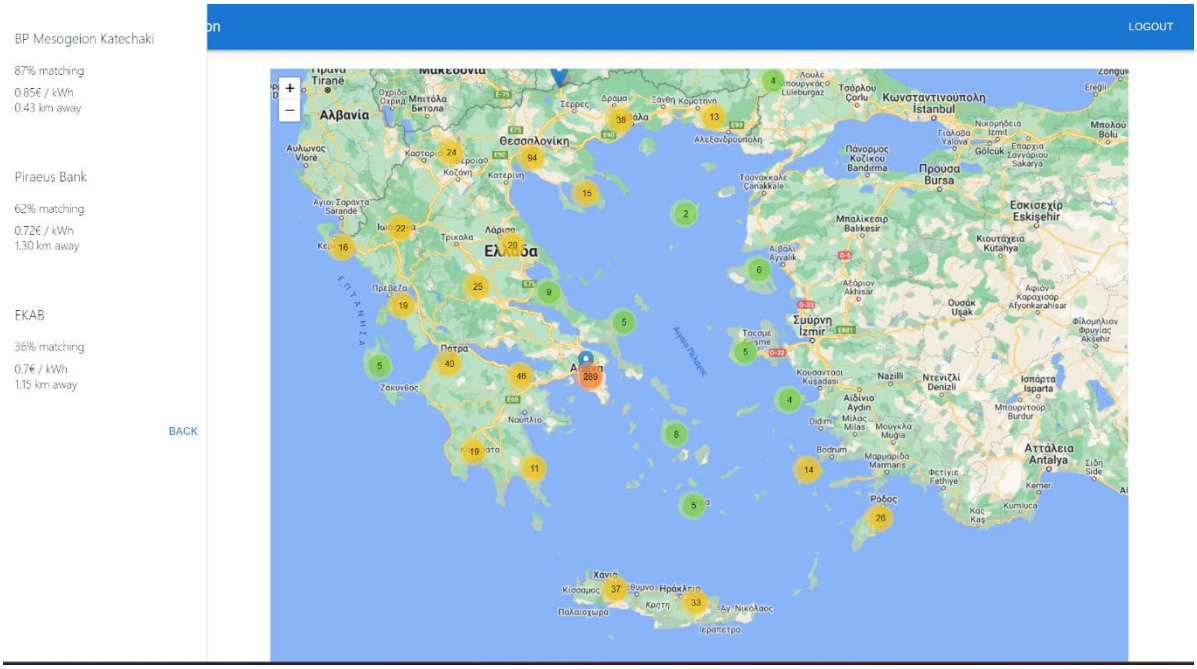
6.3 Παράδειγμα χρήσης της εφαρμογής

Στο παρόν τμήμα παρουσιάζεται ένα παράδειγμα χρήσης της εφαρμογής και οι διάφορες περιπτώσεις που προκύπτουν. Αρχικά υποθέτουμε τρείς διαφορετικούς χρήστες οι οποίοι έχουν ήδη συνδεθεί στην εφαρμογή και μπορούν να κάνουν κράτηση σε αυτή. Επίσης θεωρούμε πως και οι τρείς χρήστες ενδιαφέρονται για την ίδια περιοχή, δηλαδή στο συγκεκριμένο παράδειγμα την Λεωφόρο Κατεχάκη, και πως επιθυμούν να κάνουν την κράτηση την ίδια ώρα και μέρα. Αρχικά συνδέεται ο πρώτος χρήστης και η αρχική του σελίδα με τα επιλεγμένα φίλτρα είναι η εξής:



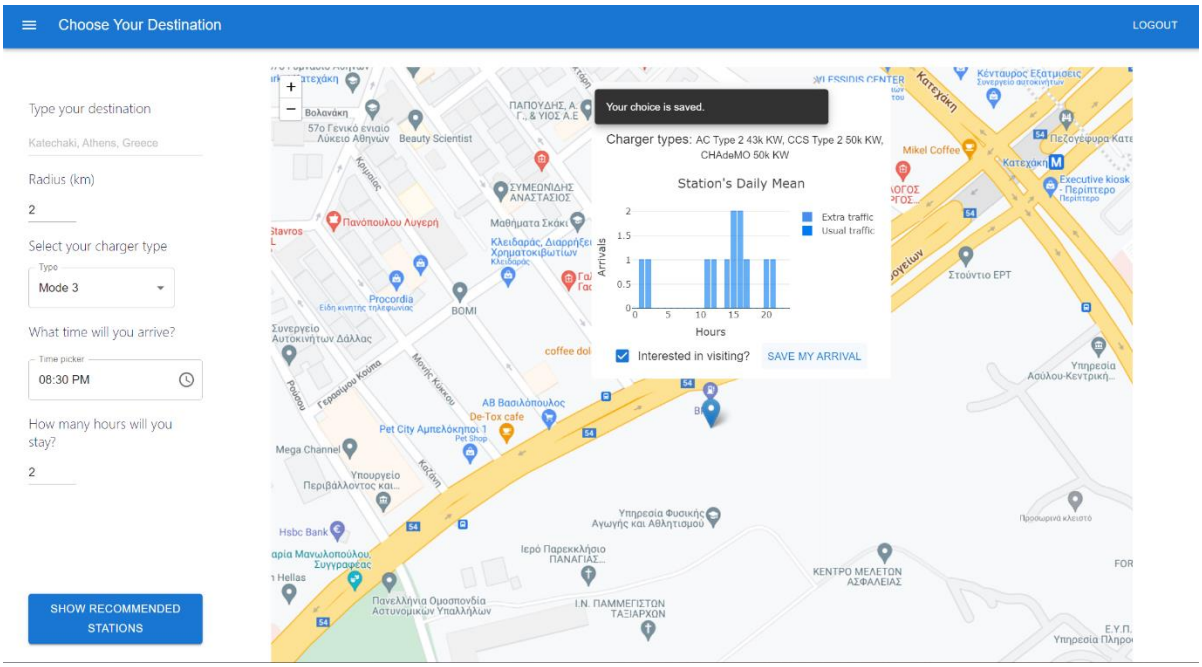
Εικόνα 6.3.1 Αρχική σελίδα εφαρμογής μετά την σύνδεση του πρώτου χρήστη

Αφού ο χρήστης πατήσει το «Show recommended station» του εμφανίζονται οι εξής τέσσερις σταθμοί:



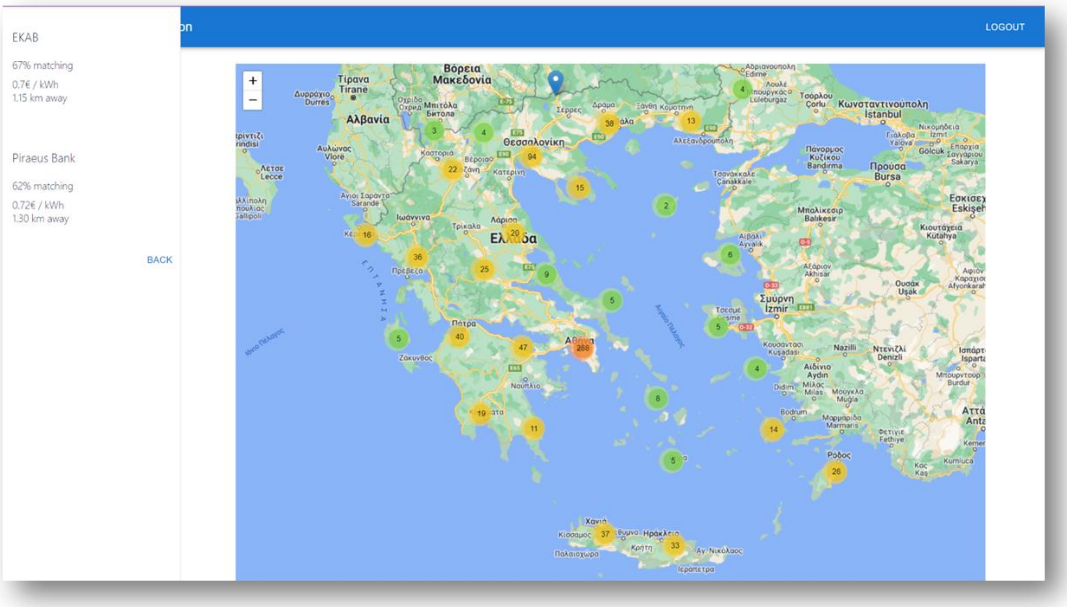
Εικόνα 6.3.2 Εμφάνιση προτεινόμενων σταθμών φόρτισης στον πρώτο χρήστη

Θεωρούμε πως ο πρώτος χρήστης επιλέγει να κάνει κράτηση στον πρώτο σταθμό, δηλαδή στον «BP Mesogeion Katechaki», για τις 8:30 μμ. Η επιλογή του αποθηκεύεται και στη συνέχεια αποσυνδέεται.



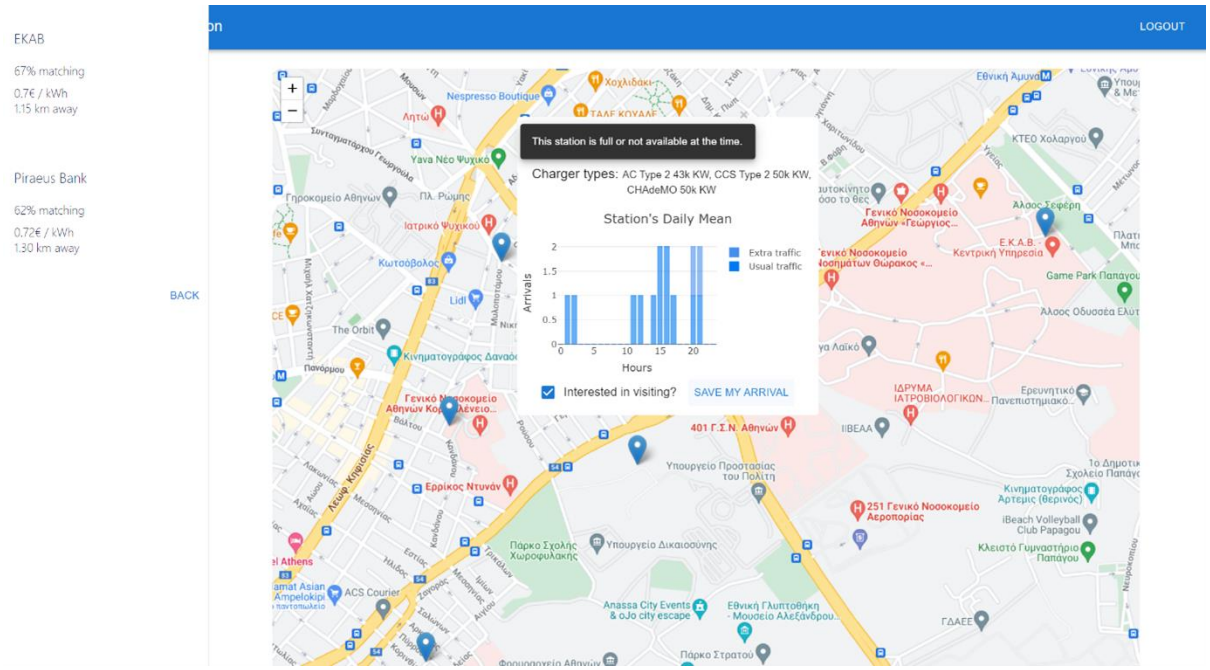
Εικόνα 6.3.3 Επιλογή σταθμού και πραγματοποίηση κράτησης

Έπειτα γίνεται αλλαγή λογαριασμού και συνδέεται ο δεύτερος χρήστης. Επιλέγοντας τα ίδια φίλτρα οι επιλογές που του εμφανίζονται είναι οι εξής:



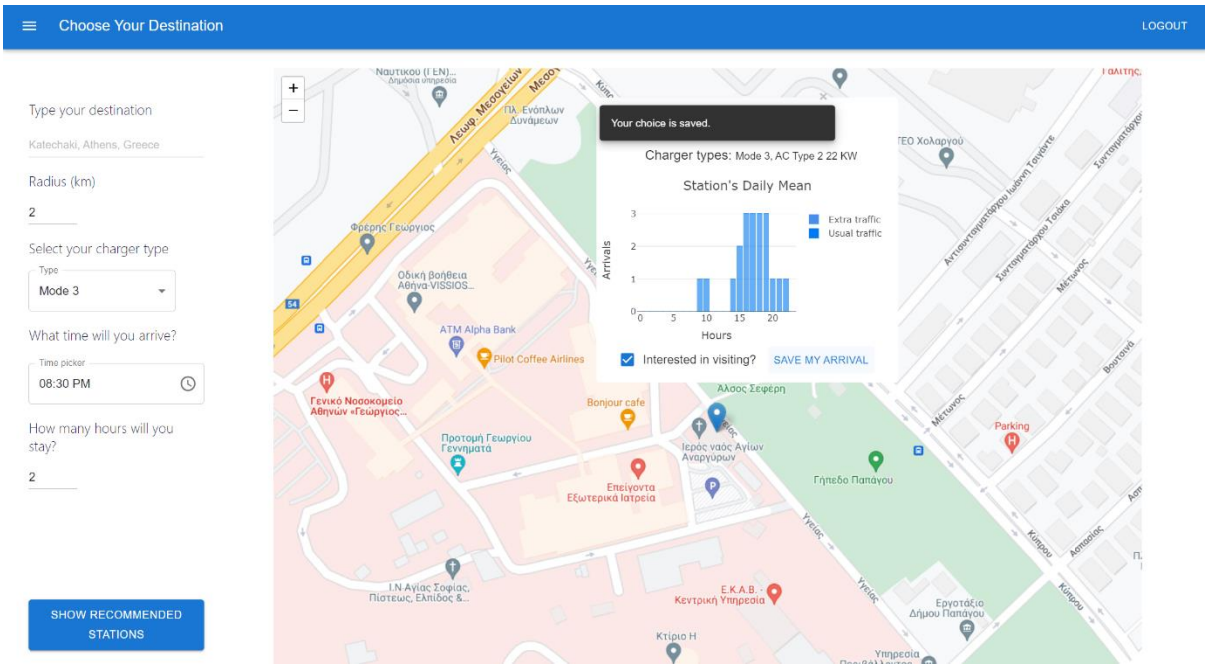
Εικόνα 6.3.4: Εμφάνιση αποτελεσμάτων σταθμών φόρτισης στον δεύτερο χρήστη

Όπως φαίνεται ο σταθμός που επέλεξε ο προηγούμενος χρήστης δεν εμφανίζεται πια σαν επιλογή καθώς εκείνη την ώρα είναι γεμάτος. Στον χάρτη όταν πλέον επιλεχτεί ο συγκεκριμένος σταθμός, φαίνεται η νέα προσθήκη στο διάγραμμά του ενώ αν γίνει προσπάθεια κράτησης εμφανίζεται το αντίστοιχο μήνυμα σφάλματος όπως φαίνεται παρακάτω:



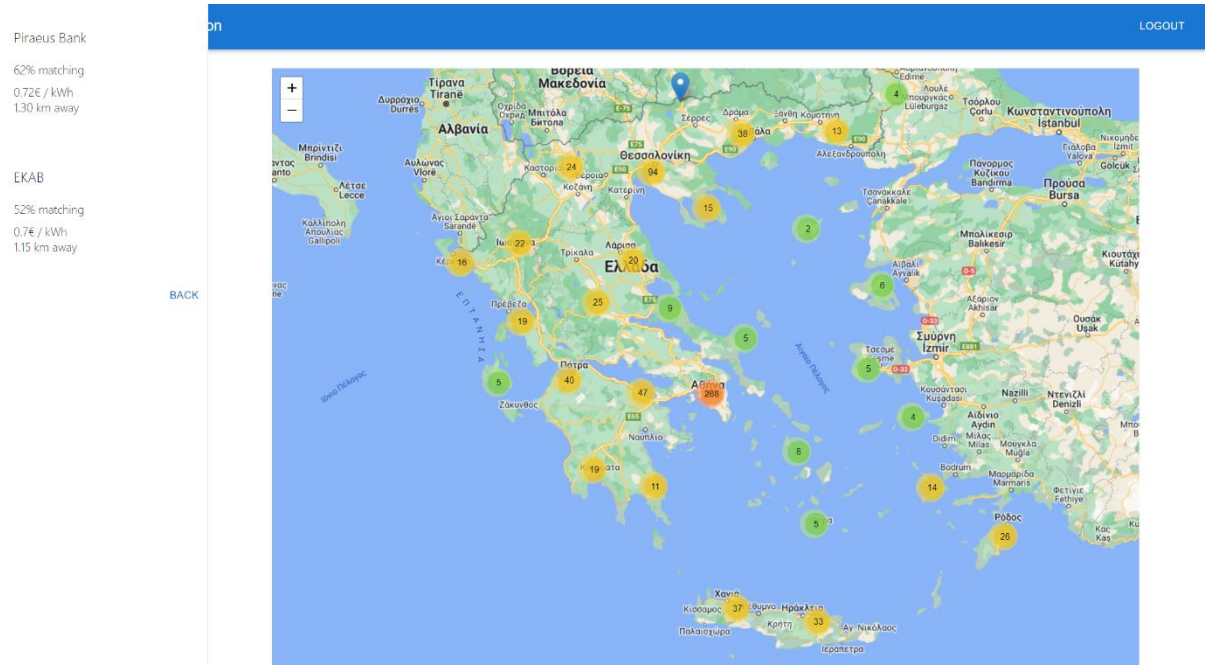
Εικόνα 6.3.5: Επιλογή σταθμού που δεν είναι διαθέσιμος και προσπάθεια κράτησης

Ο δεύτερος χρήστης επομένως επιλέγει τον πρώτο προτεινόμενο σταθμό με το όνομα «EKAB» και η επιλογή του αποθηκεύεται στη βάση.



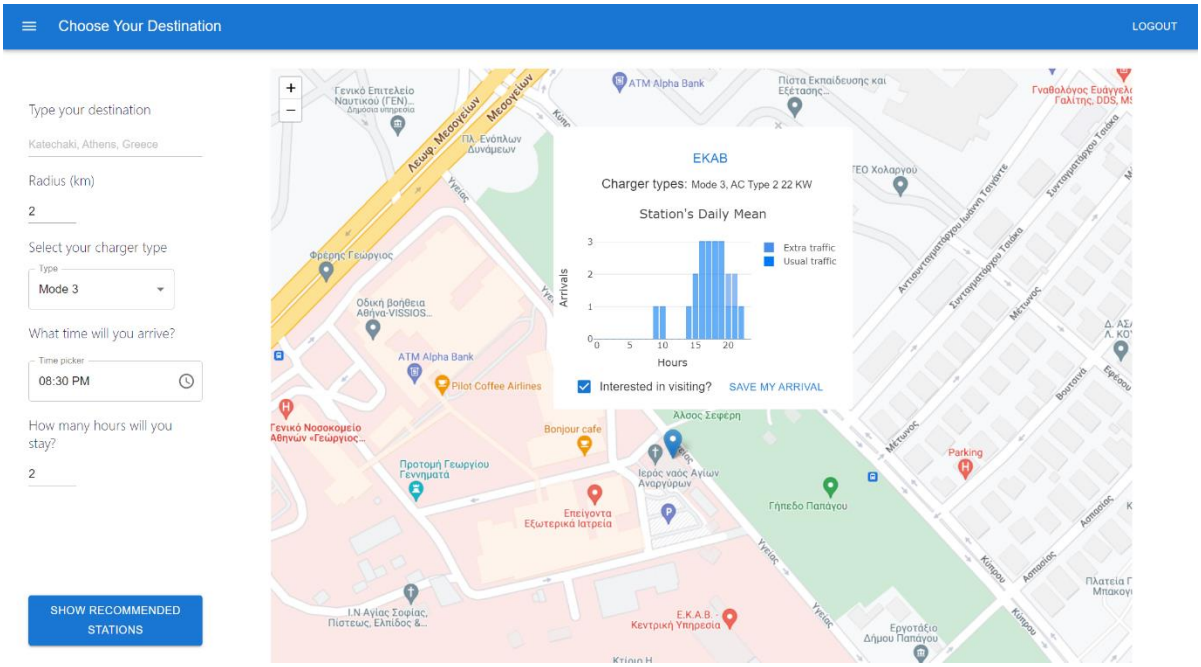
Εικόνα 6.3.6: Επιλογή διαθέσιμου σταθμού και πραγματοποίηση κράτησης από τον δεύτερο χρήστη

Τέλος γίνεται η είσοδος του τρίτου χρήστη και συμπληρώνει τα φίλτρα αντίστοιχα όπως πριν. Οι επιλογές που του προτείνονται είναι οι εξής:



Εικόνα 6.3.7: Εμφάνιση αποτελεσμάτων σταθμών φόρτισης για τον τρίτο χρήστη

Όπως φαίνεται και από τα αποτελέσματα, αν και ο πρώτος προτεινόμενος σταθμός είναι ο ίδιος με πριν, το σκορ του έχει μειωθεί στο 52% από το 67% που είχε προηγουμένως. Αυτό είναι αναμενόμενο καθώς ο σταθμός έχει πλέον μεγαλύτερο utilization, γεγονός που μειώνει το σκορ του. Εάν ο σταθμός αυτός επιλεχθεί και πάλι φαίνεται στο διάγραμμα η νέα κράτηση που προηγήθηκε.



Εικόνα 6.3.8 Εμφάνιση κράτησης προηγούμενου χρήστη στο διάγραμμα

Κεφάλαιο 7ο . Επίλογος

Συμπεράσματα και επεκτάσεις

Στόχος της διπλωματικής εργασίας ήταν η ανάπτυξη μιας φιλικής προς τον χρήστη εφαρμογής, που θα βοηθήσει στην επιλογή σταθμού φόρτισης και θα αποτρέψει τον συνωστισμό. Με τον τρόπο αυτό βελτιώνεται η εμπειρία φόρτισης αυτοκινήτου καθώς υπάρχουν συγκεντρωμένοι όλοι οι σταθμοί σε μία μόνο εφαρμογή, ενώ ταυτόχρονα ο χρήστης δέχεται προτάσεις για να διαλέξει τον κατάλληλο σταθμό χωρίς κόπο. Έμμεσα μέσω της εφαρμογής προωθείται περαιτέρω η ηλεκτροκίνηση, καθώς η δυσκολία εύρεσης κοντινού σταθμού φόρτισης αλλά και η άφιξη σε κατειλημμένο σταθμό είναι σημαντικά μειονεκτήματα για πολλούς, και σκοπός είναι να εξαλειφθούν.

Τα πιο κρίσιμα σημεία της εφαρμογής ήταν αρχικά η συλλογή και κατάλληλη επεξεργασία των δεδομένων αφίξεων, ώστε να προκύψουν τα προφίλ επισκέψεων στους σταθμούς φόρτισης, και στη συνέχεια ο αλγόριθμος βελτιστοποίησης, ο οποίος επιλέγει και ταξινομεί τους σταθμούς που θα προτείνει στον χρήστη. Χωρίς τα κατάλληλα προφίλ δεν θα υπήρχε η ρεαλιστική συμπεριφορά των σταθμών και δεν θα είχε νόημα ένας αλγόριθμος συστάσεων.

Αν και η εφαρμογή επιτελεί τις βασικές λειτουργίες που σχεδιάστηκαν, υπάρχει χώρος για πολλές επεκτάσεις που μπορούν να προστεθούν, οι οποίες ήταν δύσκολο να υλοποιηθούν στο χρονικό πλαίσιο της διπλωματικής εργασίας. Η εφαρμογή έχει θέσει τις βάσεις για την ανάπτυξη μιας ολοκληρωμένης εφαρμογής συστάσεων σταθμών που μπορεί να χρησιμοποιηθεί εμπορικά.

Μια σημαντική προσθήκη θα ήταν η ανανέωση των προφίλ επισκέψεων στο τέλος κάθε εβδομάδας, με βάση τις νέες επισκέψεις που σημειώθηκαν. Η εφαρμογή αποθηκεύει τις κρατήσεις που γίνονται από τους χρήστες και έτσι μπορούν να ληφθούν υπόψη στα προφίλ επισκέψεων ώστε να προσαρμόζονται στις συνήθειες των χρηστών με το πέρασμα του χρόνου.

Μια ακόμα αξιοσημείωτη επέκταση είναι η προσθήκη νέων φίλτρων στην αναζήτηση σταθμών. Ο αλγόριθμος βελτιστοποίησης θέτει ως κριτήρια την απόσταση και την διαθεσιμότητα. Είναι δυνατόν ωστόσο να τροποποιηθεί ώστε να συμπεριλαμβάνει την τιμή φόρτισης στους σταθμούς ή την ταχύτητα φόρτισης των φορτιστών. Με τον τρόπο αυτό ο χρήστης θα μπορεί να βρίσκει σταθμούς ανάλογα με τα δικά του κριτήρια.

Τέλος, αν και οι κρατήσεις που γίνονται μέσω της εφαρμογής δεν είναι δεσμευτικές, αλλά χρησιμεύουν στο να έχει καλύτερη εικόνα τη διαθεσιμότητα των σταθμών ο αλγόριθμος βελτιστοποίησης, μελλοντικά θα μπορούσε μέσω της συνεργασίας των παρόχων να γίνονται πραγματικές κρατήσεις και να υπάρχουν πραγματικά και ζωντανά δεδομένα. Με τον τρόπο αυτό η διαθεσιμότητα των σταθμών θα έχει την μέγιστη ακρίβεια και ο αλγόριθμος συστάσεων θα

λειτουργεί ακόμα πιο ικανοποιητικά. Ταυτόχρονα με την υλοποίηση αυτή, θα μπορεί να δημιουργηθεί μια νέα σελίδα στην εφαρμογή με το ατομικό προφίλ του κάθε χρήστη, το οποίο θα έχει πληροφορίες όπως οι κρατήσεις που έχει κάνει και στατιστικά στοιχεία πάνω σε αυτές.

Βιβλιογραφία

[1] «https://gml.noaa.gov/ccgg/trends/».

[2] «https://ourworldindata.org/co2-and-greenhouse-gas-emissions#co2-and-greenhouse-gas-emissions-country-profiles».

[3] «IEA, Global CO2 emissions by sector, 2019-2022, IEA, Paris https://www.iea.org/data-and-statistics/charts/global-co2-emissions-by-sector-2019-2022, IEA. Licence: CC BY 4.0».

[4] «https://ourworldindata.org/co2-emissions-from-transport».

[5] «IEA, Electric car registrations and sales share in China, United States and Europe, 2018-2022, IEA, Paris https://www.iea.org/data-and-statistics/charts/electric-car-registrations-and-sales-share-in-china-united-states-and-europe-2018-2022, IEA. Licence:».

[6] «IEA, Global electric car stock, 2010-2022, IEA, Paris https://www.iea.org/data-and-statistics/charts/global-electric-car-stock-2010-2022, IEA. Licence: CC BY 4.0».

[7] «IEA (2023), Global EV Outlook 2023, IEA, Paris https://www.iea.org/reports/global-ev-outlook-2023, License: CC BY 4.0».

[8] «https://ec.europa.eu/commission/presscorner/detail/en/ip_22_6462».

[9] «https://cinea.ec.europa.eu/news-events/news/transport-infrastructure-eur-2925-million-eu-funding-projects-contributing-greener-mobility-2022-09-12_en».

[10] «https://alternative-fuels-observatory.ec.europa.eu/transport-mode/road/finland/incentives-legislations».

[11] «https://alternative-fuels-observatory.ec.europa.eu/transport-mode/road/european-union-eu27#skipChart_gb2bsnxls94».

[12] «https://www.transportenvironment.org/discover/co2-targets-propel-european-ev-sales/».

[13] «https://alternative-fuels-observatory.ec.europa.eu/transport-mode/road/greece».

[14] «https://kinoumeilektrika2.gov.gr/».

[15] «https://ypen.gov.gr/».

[16] «https://www.recharge.gr/plug-port/».

[17] «Suarez, Camilo & Martinez, Wilmar. (2019). Fast and Ultra-Fast Charging for Battery Electric Vehicles -A Review. 10.1109/ECCE.2019.8912594.».

[18] «https://web.archive.org/web/20200523044852/https://www.chademo.com/chademo-releases-the-latest-version-of-the-protocol-enabling-up-to-400kw/».

[19] «https://chargespot.gr/simeia-fortisis/».

[20] «https://www.deiblue.gr/».

[21] «https://blinkcharging.gr/en/ev-driver/stations/».

[22] «https://protergiacharge.gr/charge-map/».

[23] «https://elpefuture.gr/location».

[24] «https://www.fortisis.eu/fortizo-points/».

[25] «https://electrokinisi.yme.gov.gr/public/ChargingPoints/».

[26] «ttps://www.data.gov.uk/dataset/16c7326b-57fe-4803-88f8-9286c387f68a/electric-vehicle-charging-transactions».

[27] «https://open-data.bouldercolorado.gov/datasets/39288b03f8d54b39848a2df9f1c5fca2_0».

[28] «https://www.data.gov.uk/dataset/16c7326b-57fe-4803-88f8-9286c387f68a/electric-vehicle-charging-transactions».

[29] «Xi, X., Sioshansi, R., & Marano, V. (2013). Simulation–optimization model for location of a public electric vehicle charging infrastructure. Transportation Research Part D: Transport and Environment, 22, 60–69. https://doi.org/10.1016/j.trd.2013.02.014».

[30] «Tu, R., Gai, Y. (Jessie), Farooq, B.,Posen, D.,&Hatzopoulou,M.(2020). Electric vehicle charging optimization to minimize marginal greenhouse gas emissions from power generation. Applied Energy, 277,115517. https://doi.org/10.1016/j.apenergy.2020.115517».

[31] R. R. V. G. Andrenacci N, «A demand-side approach to the optimal deployment of electric vehicle charging stations in metropolitan areas. Appl Energy 2016;182:39–46. https://doi.org/10.1016/j.apenergy.2016.07.137.».

[32] L. Q. F. Z. L. S. S. Z. B. C. X. Tu W, «Optimizing the locations of electric taxi charging stations: A spatial–temporal demand coverage approach. Transp Res Part C Emerg Technol 2016;65:172–89. https://doi.org/10.1016/j.trc. 2015.10.004.».

[33] M. H. S. M. Alhazmi YA, «Optimal allocation for electric vehicle charging stations using Trip Success Ratio. Int J Electr Power Energy Syst 2017;91:101–16. https://doi.org/10.1016/j.ijepes.2017.03.009.».

[34] Y. H. T. T. H. H. He J, «An optimal charging station location model with the consideration of electric vehicle's driving range. Transp Res Part C Emerg Technol 2018:86. https://doi.org/10.1016/j.trc.2017.11.026.».

[35] «https://www.interviewbit.com/blog/mvc-architecture/».

[36] «Bergmeir, C., Hyndman, Rob J., & Benítez, José M. (2016). Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation. International Journal of Forecasting, 32(2), 303–312. https://doi.org/10.1016/j.ijforecast.2015.07.002».

[37] «(https://otexts.com/fpp2/stl.html)».

[38] E. &. L. S. &. F. F. Boufidi, «A Probabilistic Uncertainty Estimation Method for Turbulence Parameters Measured by Hot Wire Anemometry in Short Duration Wind Tunnels. Journal of Engineering for Gas Turbines and Power. 142. 10.1115/1.4044780,» 2019.

[39] «https://www.statistics.gr/digital-cartographical-data».

[40] «https://www.plugshare.com/».