Ατομική Διπλωματική Εργασία

Υλοποίηση και Αποτίμηση Συστήματος για την Φόρτιση Ηλεκτρικών Οχημάτων με Χρήση Φωτοβολταϊκών Μικρό-Δικτύων

Ελένη Μιχάλα

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση και Αποτίμηση Συστήματος για την Φόρτιση Ηλεκτρικών Οχημάτων με Χρήση Φωτοβολταϊκών Μικρό-Δικτύων

Ελένη Μιχάλα

Επιβλέπων Καθηγητής
Δρ. Δημήτρης Ζεϊναλιπούρ

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2025

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμότερες ευχαριστίες μου στον επιβλέποντα καθηγητή μου, Δρ. Δημήτρη Ζεϊναλιπούρ, για την εμπιστοσύνη που μου έδειξε αναθέτοντάς μου την παρούσα διπλωματική εργασία, καθώς και για την πολύτιμη καθοδήγηση, τη συνεχή υποστήριξη και την αδιάκοπη μεταλαμπάδευση γνώσεων τόσο κατά την εκπόνηση της εργασίας όσο και καθ' όλη τη διάρκεια των σπουδών μου στο Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου.

Θα ήθελα να ευχαριστήσω θερμά τον διδακτορικό φοιτητή του Εργαστηρίου Συστημάτων Διαχείρισης Δεδομένων, Σωτήρη Κωνσταντίνου, για τη συνεπή καθοδήγηση, τη διαθεσιμότητα και τις χρήσιμες συμβουλές του, οι οποίες συνέβαλαν ουσιαστικά στην πρόοδο και ολοκλήρωση της εργασίας αυτής.

Εκφράζω τις ευχαριστίες μου στον Δρ. Αντρέα Κωνσταντινίδη, καθώς το σύνολο του ερευνητικού του έργου σε συνεργασία με τον επιβλέποντα καθηγητή μου, Δρ. Ζεϊναλιπούρ, συνέβαλε ουσιαστικά στη γενικότερη κατανόηση και διαμόρφωση της παρούσας εργασίας.

Θα ήθελα επίσης να ευχαριστήσω την οικογένειά μου για την αμέριστη στήριξη και ενθάρρυνση καθ' όλη τη διάρκεια των σπουδών μου.

Περίληψη

Η κλιματική αλλαγή και η ανάγκη για μείωση των εκπομπών αερίων του θερμοκηπίου έχουν οδηγήσει σε σημαντική στροφή προς την ηλεκτροκίνηση και τις καθαρές μορφές ενέργειας. Ωστόσο, η μετάβαση στην ηλεκτροκίνηση από μόνη της δεν επαρκεί για την επίτευξη των στόχων βιωσιμότητας, εφόσον η ενέργεια που χρησιμοποιείται για τη φόρτιση των ηλεκτρικών οχημάτων προέρχεται από μη ανανεώσιμες πηγές. Η αξιοποίηση μικροδικτύων που τροφοδοτούνται από φωτοβολταϊκά και άλλες ανανεώσιμες πηγές ενέργειας αποτελεί έναν πολλά υποσχόμενο δρόμο προς την περιβαλλοντικά υπεύθυνη φόρτιση. Η παρούσα διπλωματική εργασία επιχειρεί να ενσωματώσει αυτή την κατεύθυνση σε ένα λειτουργικό, διαδραστικό και υπολογιστικά αποδοτικό σύστημα, το EcoCharge+. Ο στόχος του συστήματος είναι να παρέχει στους χρήστες προτάσεις φόρτισης σε ηλεκτρικά οχήματα που λαμβάνουν υπόψη όχι μόνο την απόσταση και τη διαθεσιμότητα των φορτιστών, αλλά και το ποσοστό ενέργειας που προέρχεται από ανανεώσιμες πηγές, κυρίως ηλιακή. Για την επίτευξη αυτού του στόχου, το σύστημα συλλέγει δεδομένα από εξωτερικές υπηρεσίες όπως το OpenWeather API για την πρόβλεψη ηλιακής ακτινοβολίας και αξιοποιεί δεδομένα ισχύος, διαθεσιμότητας και αποστάσεων. Η αρχιτεκτονική του συστήματος αποτελείται από τρία βασικά επίπεδα. Το επίπεδο δεδομένων βασίζεται σε SQLite βάση δεδομένων και APIs, ενώ το υπολογιστικό επίπεδο υλοποιείται σε Python με χρήση Flask και περιλαμβάνει τον αλγόριθμο υπολογισμού ενός οικολογικού σκορ, ο οποίος λαμβάνει υπόψη την ηλιακή παραγωγή, τη διαθεσιμότητα φορτιστών, και τον χρόνο προσέγγισης. Το επίπεδο διεπαφής χρήστη προσφέρει έναν διαδραστικό χάρτη με χρήση Leaflet.js και Bootstrap, με δυνατότητα εμφάνισης και φιλτραρίσματος φορτιστών, διαδρομών, μικροδικτύων, και δυναμικών καρτελών με πληροφορίες. Η λειτουργία του συστήματος αξιολογήθηκε πειραματικά με έμφαση στην απόδοση κάθε επιπέδου. Στο επίπεδο δεδομένων, συγκρίθηκε η ταχύτητα ανάκτησης δεδομένων από SQLite και JSON αρχεία. Στο υπολογιστικό επίπεδο μετρήθηκε ο χρόνος εκτέλεσης του αλγορίθμου κατάταξης και παρατηρήθηκε η επίδραση της ακτίνας αναζήτησης και της ώρας πρόβλεψης στην οικολογική βαθμολογία. Τα αποτελέσματα έδειξαν ότι για μεγαλύτερες ακτίνες, το σύστημα προτείνει φορτιστές με υψηλότερο οικολογικό σκορ λόγω της μεγαλύτερης διαθέσιμης ποικιλίας. Αντίστοιχα, η χρήση προβλεπόμενων δεδομένων ηλιακής παραγωγής ανέδειξε τη σημασία της χρονικής στιγμής στην απόδοση του συστήματος. Τέλος, στο επίπεδο της διεπαφής, μελετήθηκαν οι χρόνοι απόκρισης για ενέργειες όπως η εμφάνιση όλων των φορτιστών, η αλλαγή φίλτρων πρίζας και η εμφάνιση λεπτομερειών μικροδικτύου, με τα αποτελέσματα να επιβεβαιώνουν την χρηστικότητα και την ομαλή εμπειρία χρήστη.

Περιεχόμενα

Κεφάλαιο 1 Εισαγωγή

1.1 Κίνητρο

Η παγκόσμια τάση προς την πράσινη ενέργεια και τη μείωση του αποτυπώματος άνθρακα έχει οδηγήσει στην ευρεία υιοθέτηση ηλεκτρικών οχημάτων. Σύμφωνα με τον Διεθνή Οργανισμό Ενέργειας (IEA), το 2023 οι πωλήσεις ηλεκτρικών οχημάτων έφτασαν τα 14 εκατομμύρια παγκοσμίως, αντιπροσωπεύοντας το 18% των συνολικών πωλήσεων αυτοκινήτων, με πρόβλεψη να φτάσουν το 35% έως το 2030 [1]. Ακόμη, σύμφωνα με μελέτες, η ζήτηση ενέργειας για ηλεκτρική φόρτιση στις ΗΠΑ αναμένεται να αυξηθεί από 4.7 TWh το 2020 σε 107 TWh μέχρι το 2035, καθιστώντας απαραίτητο τον οικολογικό σχεδιασμό των συστημάτων φόρτισης [3]. Αυτή η ταχεία διείσδυση των ηλεκτρικών οχημάτων φέρνει νέες προκλήσεις και ευκαιρίες στον τομέα των μεταφορών και της ενεργειακής υποδομής.

Η σημαντικότερη πρόκληση αφορά τη φόρτιση των οχημάτων με τρόπο που να είναι ταυτόχρονα βιώσιμος και αποδοτικός. Ενώ η χρήση ηλεκτρικών οχημάτων μειώνει τις εκπομπές ρύπων σε σύγκριση με τα οχήματα εσωτερικής καύσης, η θετική αυτή επίδραση αναιρείται όταν η ενέργεια φόρτισης προέρχεται από μη ανανεώσιμες πηγές. Πολλοί οδηγοί επιλέγουν να φορτίσουν το όχημά τους σε ώρες που δεν συμπίπτουν με μέγιστη ηλιακή παραγωγή, συχνά από συνήθεια ή για λόγους άνεσης. Η ενέργεια αυτή, αν δεν προέρχεται από ανανεώσιμες πηγές, αυξάνει την κατανάλωση από το δίκτυο και εντείνει την πίεση σε ώρες αιχμής.

Η ανάγκη για συντονισμένη, βιώσιμη και "έξυπνη" φόρτιση είναι πλέον επιτακτική. Η έννοια της ανανεώσιμης συσσώρευσης ενέργειας αποκτά σημασία, ειδικά σε περιπτώσεις όπως:

- ηλεκτρικά ταξί (Uber, Bolt) που περιμένουν πελάτες,

- οδηγοί που αφήνουν τα παιδιά σε δραστηριότητες,
- ιδιοκτήτες ηλεκτρικών αυτοκινήτων που πηγαίνουν για ψώνια ή εργάζονται,
- συμμετοχή σε εκδηλώσεις ή συνέδρια.

Στις παραπάνω περιπτώσεις, η εκμετάλλευση του χρόνου στάθμευσης για βιώσιμη φόρτιση μέσω ηλιακής ενέργειας μπορεί να οδηγήσει σε σημαντική μείωση του αποτυπώματος άνθρακα των μετακινήσεων.

Παρότι υπάρχουν εφαρμογές όπως PlugShare, IONITY, Octopus Electroverse, που επιτρέπουν την εύρεση σημείων φόρτισης, δεν παρέχουν δυναμική πληροφόρηση για το αν η ενέργεια προέρχεται από ΑΠΕ ή για το οικολογικό αποτύπωμα της φόρτισης. Επίσης, δεν ενσωματώνουν χωρικά και χρονικά δεδομένα που αφορούν τον καιρό, την παραγωγή ηλιακής ενέργειας ή τη διαθεσιμότητα των φορτιστών. Για την αντιμετώπιση αυτών των ελλείψεων, αναπτύχθηκε το EcoCharge+, ένα ολοκληρωμένο σύστημα που συνδυάζει δεδομένα καιρού, διαθεσιμότητας, απόστασης και παραγωγής ενέργειας, μέσω της ενσωμάτωσης μικροδικτύων, τοπικών ενεργειακών δικτύων με φωτοβολταϊκά συστήματα. Χρησιμοποιεί χωροχρονικά ερωτήματα για την εύρεση και ταξινόμηση των φορτιστών, προσφέρει στον χρήστη εξατομικευμένες προτάσεις βασισμένες στο επιλεγμένο όχημα και επιθυμητό επίπεδο «οικολογικότητας» και ενσωματώνει τεχνικές πρόβλεψης ηλιακής παραγωγής με βάση πραγματικά και προβλεπόμενα δεδομένα OpenWeather [5].

Η επιλογή του βέλτιστου σημείου φόρτισης δεν μπορεί να βασίζεται μόνο σε γεωμετρικά κριτήρια (όπως η απόσταση), αλλά απαιτεί πολυκριτηριακή αξιολόγηση με βάση εκτιμώμενα στοιχεία και προτιμήσεις του χρήστη. Συγκεκριμένα, λαμβάνονται υπόψη:

- το κόστος παρέκκλισης, δηλαδή πόσο μακριά είναι ο φορτιστής από τη διαδρομή του οχήματος,
- η διαθεσιμότητα του φορτιστή, βάσει των εκτιμώμενων ωρών αιχμής,
- τη διαθέσιμη καθαρή ενέργεια, δηλαδή το ποσοστό ηλιακής ενέργειας που μπορεί να χρησιμοποιηθεί τη δεδομένη στιγμή.

Για το σκοπό αυτό, το σύστημα χρησιμοποιεί έναν προσαρμοσμένο αλγόριθμο ταξινόμησης των k πλησιέστερων φορτιστών, ο οποίος, εκτός από την απόσταση, αξιολογεί κάθε φορτιστή βάσει μιας συνολικής βαθμολογίας (EcoScore). Αυτή υπολογίζεται συνδυάζοντας την προβλεπόμενη παραγωγή ηλιακής ενέργειας στο μικροδίκτυο του φορτιστή, τον συγκεκριμένο μήνα και ώρα της ημέρας, την εκτιμώμενη νεφοκάλυψη είτε για την τωρινή ώρα είτε στο κοντινό μέλλον (μέσω OpenWeather API) και τις προτιμήσεις του χρήστη σε κάθε παράγοντα

(π.χ. προτεραιότητα στην καθαρή ενέργεια, στη μικρή απόκλιση, ή στη διαθεσιμότητα). Η χρήση αυτής της προσέγγισης προσφέρει προσωποποιημένες και περιβαλλοντικά φιλικές επιλογές, ξεπερνώντας τους περιορισμούς των απλών χωρικών ερωτημάτων k-NN που δεν λαμβάνουν υπόψη δυναμικά δεδομένα καιρού και ενεργειακής παραγωγής [4].

Η εργασία περιλαμβάνει τον σχεδιασμό και την ανάπτυξη βάσης δεδομένων, τη συλλογή και επεξεργασία δεδομένων καιρού, τη δυναμική βαθμολόγηση φορτιστών, καθώς και μια διαδραστική χαρτογραφική διεπαφή για την πλοήγηση και επιλογή διαδρομής. Η τελική εφαρμογή φιλοδοξεί να αποτελέσει ένα παράδειγμα τού πώς τα δεδομένα και οι αλγόριθμοι μπορούν να υποστηρίξουν την πράσινη μετάβαση στις μεταφορές, μέσω τεχνολογιών που αξιοποιούν ανανεώσιμες πηγές ενέργειας και ευφυείς μεθόδους λήψης αποφάσεων. Η χρήση τεχνολογιών όπως τα μικροδίκτυα, οι έξυπνοι αλγόριθμοι δρομολόγησης, και η χαρτογραφική απεικόνιση με Leaflet επιτρέπουν τη μετάβαση σε μια πράσινη εμπειρία μετακίνησης. Το σύστημα προσφέρει μια νέα προσέγγιση όπου η φόρτιση δεν είναι απλά γεωγραφική επιλογή, αλλά οικολογική απόφαση.

1.2 Πρόβλημα

Η μετάβαση στην ηλεκτροκίνηση αποτελεί κρίσιμο βήμα προς την επίτευξη των στόχων της Ευρωπαϊκής Πράσινης Συμφωνίας για κλιματική ουδετερότητα έως το 2050 [7]. Η χρήση ηλεκτρικών οχημάτων αυξάνεται ραγδαία παγκοσμίως, καθώς αποτελούν ένα σημαντικό βήμα για τη μείωση των εκπομπών CO_2 και την επίτευξη βιώσιμων μεταφορών. Ωστόσο, η πραγματική περιβαλλοντική επίδραση ενός οχήματος εξαρτάται σε μεγάλο βαθμό από την πηγή της ενέργειας φόρτισής του. Όταν η ενέργεια προέρχεται από το κεντρικό δίκτυο ηλεκτρισμού, το οποίο σε πολλές περιπτώσεις τροφοδοτείται από μη ανανεώσιμες πηγές ενέργειας (όπως φυσικό αέριο ή πετρέλαιο), τα οφέλη της ηλεκτροκίνησης μειώνονται σημαντικά. Οι υφιστάμενες λύσεις πλοήγησης και εύρεσης φορτιστών βασίζονται κυρίως σε στατικά γεωγραφικά δεδομένα (γεωγραφική απόσταση) και αγνοούν σημαντικούς δυναμικούς παράγοντες, όπως το πόσο καθαρή είναι η ενέργεια που παρέχεται στο σημείο φόρτισης, τη διαθεσιμότητα του φορτιστής τη δεδομένη στιγμή, και πόσο μακριά βρίσκεται από την κανονική διαδρομή του χρήστη.

Για την προσέγγιση αυτών των ζητημάτων, προτάθηκε στην προηγούμενη ερευνητική εργασία το πλαίσιο EcoCharge [26], το οποίο εισήγαγε ένα συνεχές ερώτημα k πλησιέστερων γειτόνων

με εκτιμώμενα στοιχεία (CkNN-EC). Ο αλγόριθμος του EcoCharge ταξινομεί τους φορτιστές βάσει:

1. Κόστους παρέκκλισης (δηλαδή του χρόνου ή της απόστασης που απαιτείται για να φτάσει κάποιος στον φορτιστή, λαμβάνοντας υπόψη την κυκλοφορία),
2. Διαθεσιμότητας φορτιστή, βάσει προβλέψεων για την πληρότητα σε συγκεκριμένες ώρες,
3. Καθαρής ενέργειας, βάσει πρόβλεψης καιρού.

Οι χρήστες επιλέγουν το βάρος κάθε παράγοντα, και το σύστημα επιστρέφει ταξινομημένες προτάσεις φορτιστών με βάση έναν δείκτη βιωσιμότητας.

Η παρούσα εργασία επεκτείνει ουσιαστικά τον παραπάνω αλγόριθμο, εισάγοντας ένα νέο μοντέλο αξιολόγησης που ενσωματώνει δεδομένα από πολλαπλές πηγές. Αρχικά, λαμβάνει υπόψη προβλεπόμενη ηλιακή παραγωγή από φωτοβολταϊκά συστήματα μικροδικτύων, βασισμένη σε στην ώρα της ημέρας (ωριαίο προφίλ παραγωγής), τον μήνα του έτους (εποχικότητα), και τη νεφοκάλυψη. Ενσωματώνει τα τεχνικά χαρακτηριστικά του μικροδικτύου που τροφοδοτεί τον κάθε φορτιστή, όπως εγκατεστημένη ισχύς (kWp) και απόδοση, ώστε να διαμοιράσει ανάλογα την ενέργεια στου φορτιστές του. Επίσης, υποστηρίζει συμβατότητα με τύπους πρίζας υποδοχής του φορτιστή ανά όχημα, για απόρριψη μη συμβατών φορτιστών. Τέλος, παρέχει τη δυνατότητα για πρόβλεψη των οικολογικά καλύτερων φορτιστών στο κοντινό μέλλον.

Το νέο μοντέλο ταξινομεί δυναμικά τους φορτιστές, αξιολογώντας τους ως προς τη συνολική περιβαλλοντική τους αποδοτικότητα με βάση έναν σύνθετο δείκτη βιωσιμότητας (EcoScore), ο οποίος προσαρμόζεται ανάλογα με τις προτιμήσεις του χρήστη.

Συνοψίζοντας, το πρόβλημα που επιχειρεί να επιλύσει η παρούσα εργασία είναι η έλλειψη ενός ευφυούς, οικολογικά ευαισθητοποιημένου και εξατομικευμένου συστήματος προτάσεων φόρτισης ηλεκτρικών οχημάτων. Το σύστημα αυτό πρέπει να λαμβάνει υπόψη χωροχρονικά δεδομένα, τεχνικά χαρακτηριστικά μικροδικτύων, πρόβλεψη ηλιακής απόδοσης, και προτιμήσεις του χρήστη, ώστε να προσφέρει πραγματικά βιώσιμες και πρακτικές επιλογές φόρτισης.

1.3 Συνεισφορά

Η παρούσα εργασία εισάγει μια νέα μεθοδολογία για την αξιολόγηση και επιλογή σημείων φόρτισης ηλεκτρικών οχημάτων με βάση περιβαλλοντικά και λειτουργικά κριτήρια, αξιοποιώντας χωρικά και χρονικά δεδομένα, πρόβλεψη καιρού, και τεχνικά χαρακτηριστικά μικροδικτύων. Η συνεισφορά της εργασίας εντοπίζεται τόσο στον τεχνολογικό σχεδιασμό και την υλοποίηση, όσο και στην προώθηση της βιώσιμης φόρτισης των ηλεκτρικών οχημάτων.

Πιο συγκεκριμένα, η εργασία επεκτείνει τον αλγόριθμο που χρησιμοποιεί το συνεχές ερώτημα k πλησιέστερων γειτόνων (CkNN) με εκτιμώμενα στοιχεία, ενσωματώνοντας δεδομένα πρόβλεψης ηλιακής ενέργειας σε επίπεδο ώρας και μήνα, καιρικές μεταβλητές όπως η νεφοκάλυψη μέσω του OpenWeather API, καθώς και τεχνικά χαρακτηριστικά των μικροδικτύων όπως η εγκατεστημένη ισχύς και η απόδοση. Επιπλέον, λαμβάνεται υπόψη η συμβατότητα των φορτιστών με το επιλεγμένο ηλεκτρικό όχημα, βάσει του τύπου πρίζας, καθώς και οι προτιμήσεις του χρήστη ως προς την προτεραιότητα σε οικολογική ενέργεια, μικρή απόκλιση από τη διαδρομή ή υψηλή διαθεσιμότητα.

Η εκτίμηση των φορτιστών βασίζεται σε έναν σύνθετο δείκτη βιωσιμότητας, τον EcoScore, ο οποίος υπολογίζεται δυναμικά λαμβάνοντας υπόψη το ποσοστό καθαρής ηλιακής ενέργειας, το κόστος παρέκκλισης και τη διαθεσιμότητα. Το αποτέλεσμα είναι μια προσωποποιημένη και οικολογικά βελτιστοποιημένη πρόταση προς τον οδηγό, ανάλογα με τις συνθήκες και τις επιλογές του.

Πέρα από τον αλγοριθμικό σχεδιασμό, η εργασία υλοποιεί ένα πλήρες πληροφοριακό σύστημα, βασισμένο σε Flask backend, SQLite βάση δεδομένων και Leaflet.js διεπαφή, προσφέροντας στους χρήστες δυναμική χαρτογραφική απεικόνιση των διαθέσιμων φορτιστών με χρωματική κωδικοποίηση και δυνατότητα επιλογής οχήματος και φίλτρων.

Με τον τρόπο αυτό, η εργασία συμβάλλει ουσιαστικά στην ανάπτυξη τεχνολογικών εργαλείων που υποστηρίζουν τη βιώσιμη κινητικότητα και την πράσινη μετάβαση στις μεταφορές, δίνοντας έμφαση στη συνδυαστική χρήση δεδομένων, αλγορίθμων και εμπειρίας χρήστη για την παροχή πληροφόρησης που προάγει οικολογικά υπεύθυνες αποφάσεις.

1.4 Συνοπτική περιγραφή εργασίας

Η παρούσα διπλωματική εργασία αποτελείται από επτά κεφάλαια.

Στο Κεφάλαιο 1 παρουσιάζονται το κίνητρο, το πρόβλημα και η συνεισφορά της εργασίας, καθώς και μια συνοπτική επισκόπηση της συνολικής προσέγγισης.

Στο Κεφάλαιο 2 καταγράφονται σχετικές υπηρεσίες και πλατφόρμες που αφορούν τη φόρτιση ηλεκτρικών οχημάτων, καθώς και επιστημονικές μελέτες που άπτονται θεμάτων ανανεώσιμης ενέργειας, πρόβλεψης ηλιακής ακτινοβολίας, μικροδικτύων και χωροχρονικών ερωτημάτων.

Στο Κεφάλαιο 3 γίνεται η διατύπωση του προβλήματος και η περιγραφή του μοντέλου του συστήματος EcoCharge+. Παρουσιάζεται η λειτουργικότητα, ο αλγόριθμος κατάταξης φορτιστών, καθώς και η αρχιτεκτονική του συστήματος.

Το Κεφάλαιο 4 περιλαμβάνει την αναλυτική υλοποίηση του συστήματος τόσο για το backend όσο και για το frontend. Η υλοποίηση διαρθρώνεται σε επίπεδο δεδομένων, υπολογιστικό επίπεδο και επίπεδο διεπαφής, ενώ παρατίθενται αναλυτικά οι τεχνολογίες που χρησιμοποιήθηκαν.

Στο Κεφάλαιο 5 αναλύεται το γραφικό περιβάλλον χρήστη του συστήματος, με παρουσίαση όλων των στοιχείων της διεπαφής.

Το Κεφάλαιο 6 επικεντρώνεται στη μεθοδολογία και τα πειράματα που πραγματοποιήθηκαν για την αξιολόγηση του συστήματος. Γίνονται πειράματα στο επίπεδο δεδομένων, στο υπολογιστικό επίπεδο, καθώς και στο επίπεδο διεπαφής χρήστη.

Τέλος, στο Κεφάλαιο 7 συνοψίζονται τα βασικά συμπεράσματα της εργασίας και προτείνονται μελλοντικές κατευθύνσεις εξέλιξης για την περαιτέρω ανάπτυξη του EcoCharge+.

Κεφάλαιο 2 Σχετική Εργασία

Στο κεφάλαιο αυτό παρουσιάζονται σύγχρονες πλατφόρμες και επιστημονικές εργασίες που σχετίζονται με τη φόρτιση ηλεκτρικών οχημάτων και τη λήψη αποφάσεων με βάση τεχνικά, γεωχωρικά και περιβαλλοντικά κριτήρια.

Αρχικά, αναλύονται υφιστάμενες υπηρεσίες που υποστηρίζουν τη δρομολόγηση ή την αναζήτηση σημείων φόρτισης. Στη συνέχεια, εξετάζονται σχετικές ερευνητικές προσεγγίσεις που αφορούν αλγορίθμους k πλησιέστερων γειτόνων, αξιολόγηση βιωσιμότητας, πρόβλεψη παραγωγής ενέργειας από ΑΠΕ, καθώς και προσωποποιημένες συστάσεις σε περιβάλλοντα μετακίνησης με ηλεκτρικά οχήματα.

2.1 Υπηρεσίες

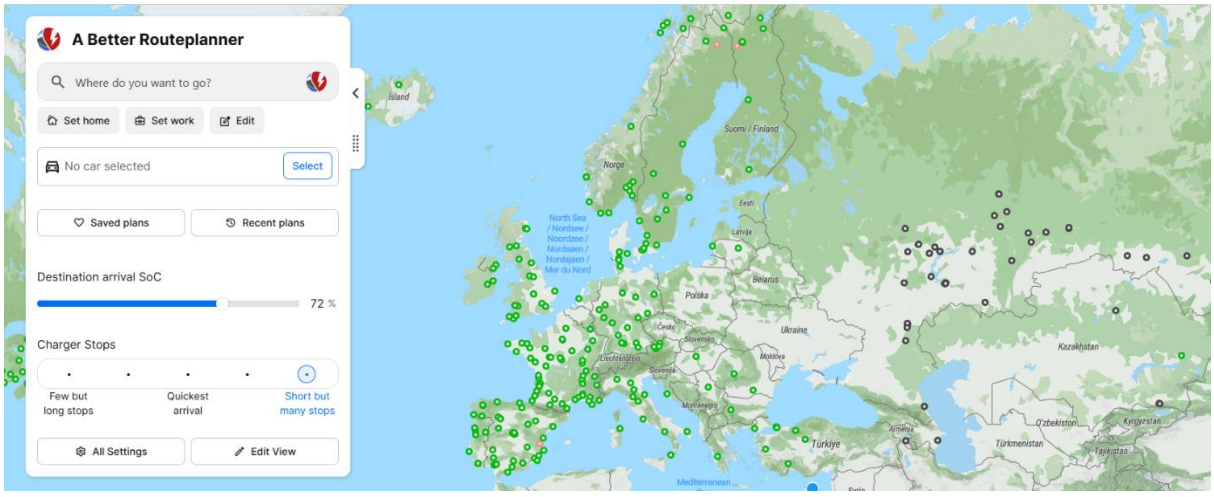
Στον τομέα της ηλεκτροκίνησης έχουν αναπτυχθεί διάφορες ψηφιακές πλατφόρμες που εξυπηρετούν τους οδηγούς στην αναζήτηση φορτιστών και στον σχεδιασμό διαδρομών με βάση τις ανάγκες των ηλεκτρικών οχημάτων. Η κάθε υπηρεσία προσφέρει διαφορετικά χαρακτηριστικά, βαθμό προσωποποίησης και πληροφόρηση σχετικά με τους σταθμούς φόρτισης. Στην ενότητα αυτή, παρουσιάζονται ενδεικτικά μερικές από τις πιο αντιπροσωπευτικές και ευρέως χρησιμοποιούμενες εφαρμογές.

2.1.1 A Better Routeplanner (ABRP)

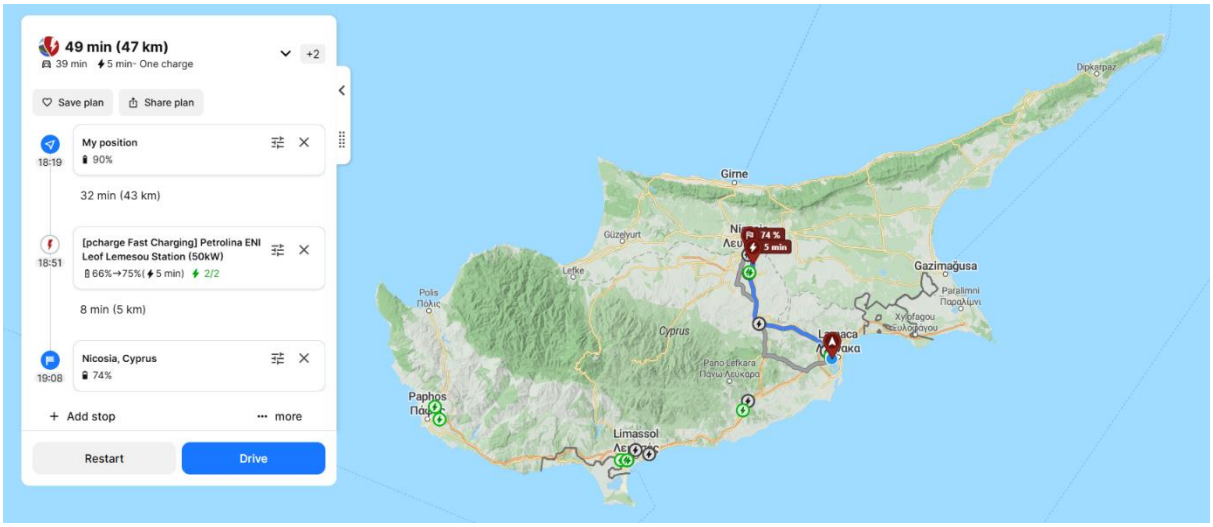
Το A Better Routeplanner (ABRP) [8] αποτελεί μία από τις πιο διαδεδομένες πλατφόρμες πλοήγησης για ηλεκτρικά οχήματα παγκοσμίως. Η εφαρμογή επιτρέπει στους οδηγούς να σχεδιάζουν τη διαδρομή τους λαμβάνοντας υπόψη παραμέτρους όπως η μπαταρία του οχήματος, η κατανάλωση ανά χιλιόμετρο, η ταχύτητα, οι υψομετρικές διαφορές και το διαθέσιμο δίκτυο φόρτισης κατά μήκος της διαδρομής. Ένα από τα βασικά χαρακτηριστικά του ABRP είναι η δυνατότητα προσωποποίησης ανάλογα με το προφίλ του οχήματος και του οδηγού. Ο χρήστης μπορεί να καθορίσει την αρχική και τελική κατάσταση φόρτισης, την ελάχιστη επιθυμητή φόρτιση κατά την άφιξη σε κάθε σταθμό, καθώς και το πώς θα κατανέμονται οι στάσεις κατά μήκος της διαδρομής για βέλτιστη απόδοση. Επιπλέον, η εφαρμογή επιτρέπει την επιλογή προτιμώμενων παρόχων φόρτισης και τύπων φορτιστών. Ωστόσο, παρά την προηγμένη πλοήγηση και τις δυνατότητες πρόβλεψης κατανάλωσης, το ABRP δεν λαμβάνει υπόψη τη βιωσιμότητα της ενέργειας που παρέχεται σε κάθε φορτιστή. Δεν παρέχει δηλαδή πληροφορίες σχετικά με την πηγή ενέργειας (ανανεώσιμη ή μη), ούτε ενσωματώνει δεδομένα πρόβλεψης καιρού ή ηλιακής ενέργειας.

Στην Εικόνα 2.1 παρουσιάζεται η αρχική οθόνη της εφαρμογής ABRP, μέσω της οποίας ο χρήστης μπορεί να εισάγει το όχημά του, το σημείο αναχώρησης και προορισμού, καθώς και τις επιθυμητές ρυθμίσεις φόρτισης.

Στην Εικόνα 2.2, φαίνεται το αποτέλεσμα της δρομολόγησης με σημεία στάσης για φόρτιση, την εκτιμώμενη κατάσταση της μπαταρίας και την προβλεπόμενη κατανάλωση σε κάθε τμήμα της διαδρομής.



Εικόνα 2.1 Αρχική οθόνη και προβολή σταθμών φόρτισης στην εφαρμογή ABRP.

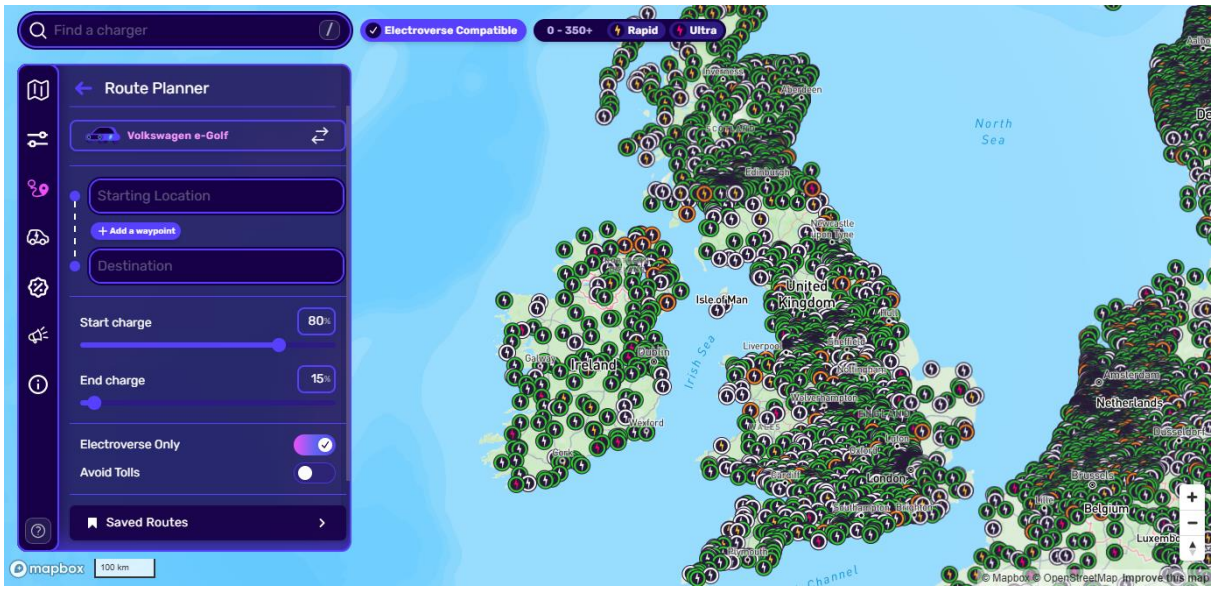


Εικόνα 2. 2 Υπολογισμένη διαδρομή με ενδιάμεση στάση φόρτισης στην Κύπρο.

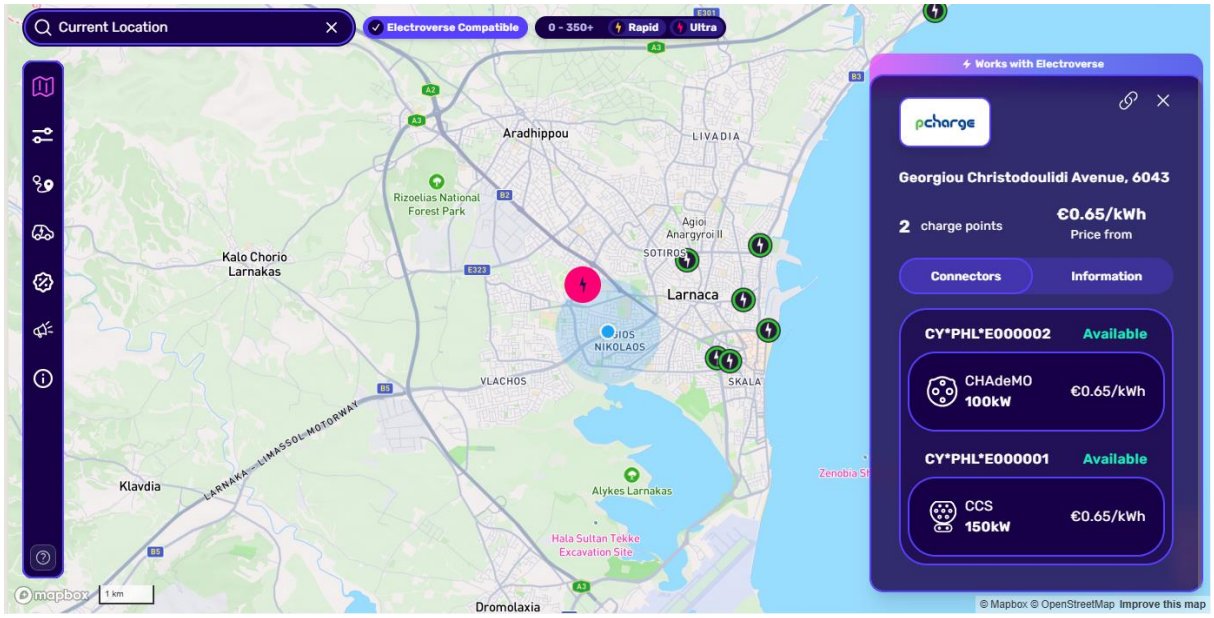
2.1.2 Octopus Electroverse

Η Octopus Electroverse αποτελεί μια σύγχρονη υπηρεσία φόρτισης ηλεκτρικών οχημάτων, η οποία δημιουργήθηκε από την βρετανική εταιρεία ενέργειας Octopus Energy. Στόχος της είναι η ενοποίηση του οικοσυστήματος των δημόσιων φορτιστών, προσφέροντας στους οδηγούς πρόσβαση σε χιλιάδες σημεία φόρτισης από διαφορετικούς παρόχους μέσω μιας ενιαίας εφαρμογής. Μία από τις βασικές καινοτομίες της Electroverse είναι η δυνατότητα «One card, one app» - μια προσέγγιση κατά την οποία ο χρήστης χρειάζεται μόνο μία κάρτα ή μια εφαρμογή για να χρησιμοποιεί όλους τους φορτιστές. Η πλατφόρμα υποστηρίζει τη δυνατότητα πλοήγησης, εμφάνισης της διαθεσιμότητας των φορτιστών σε πραγματικό χρόνο, και προβολής του κόστους φόρτισης, ενώ παρέχει και άμεση τιμολόγηση μετά από κάθε φόρτιση χωρίς την ανάγκη δημιουργίας λογαριασμού σε κάθε επιμέρους πάροχο. Η Octopus Electroverse βελτιώνει σημαντικά την εμπειρία χρήστη και διευκολύνει την πρόσβαση σε σταθμούς φόρτισης μέσω μιας ενοποιημένης και λειτουργικής διεπαφής, χωρίς όμως να εστιάζει στη βιωσιμότητα ή στην προσαρμογή βάσει περιβαλλοντικών κριτηρίων.

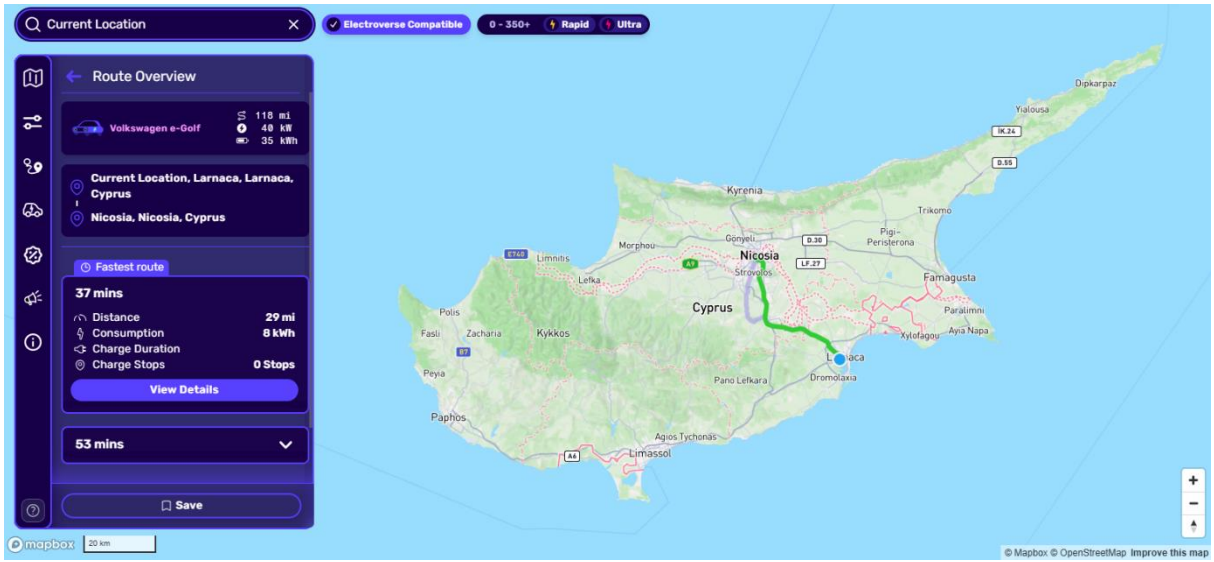
Στην Εικόνα 2.3 παρουσιάζεται η αρχική οθόνη της εφαρμογής Octopus Electroverse, όπου εμφανίζονται οι διαθέσιμοι σταθμοί φόρτισης στον χάρτη, με δυνατότητα φιλτραρίσματος.
Στην Εικόνα 2.4 φαίνεται η λεπτομερής προβολή ενός σταθμού φόρτισης, με πληροφορίες για τα διαθέσιμα σημεία φόρτισης, τους τύπους πριζών, τις τιμές ανά κιλοβατώρα και τη συμβατότητα με την Electroverse.
Στην Εικόνα 2.5 παρουσιάζεται το αποτέλεσμα μιας υπολογισμένης διαδρομής στην Κύπρο, όπου απεικονίζονται η κατανάλωση ενέργειας, η απόσταση και η διάρκεια της διαδρομής.



Εικόνα 2. 3 Αρχική οθόνη και προβολή σταθμών φόρτισης στην εφαρμογή Octopus Electroverse.



Εικόνα 2.4 Πληροφορίες σταθμού φόρτισης στην περιοχή της Λάρνακας μέσω της εφαρμογής Octopus Electroverse.



Εικόνα 2.5 Υπολογισμένη διαδρομή χωρίς ενδιάμεση στάση φόρτισης στην εφαρμογή Octopus Electroverse.

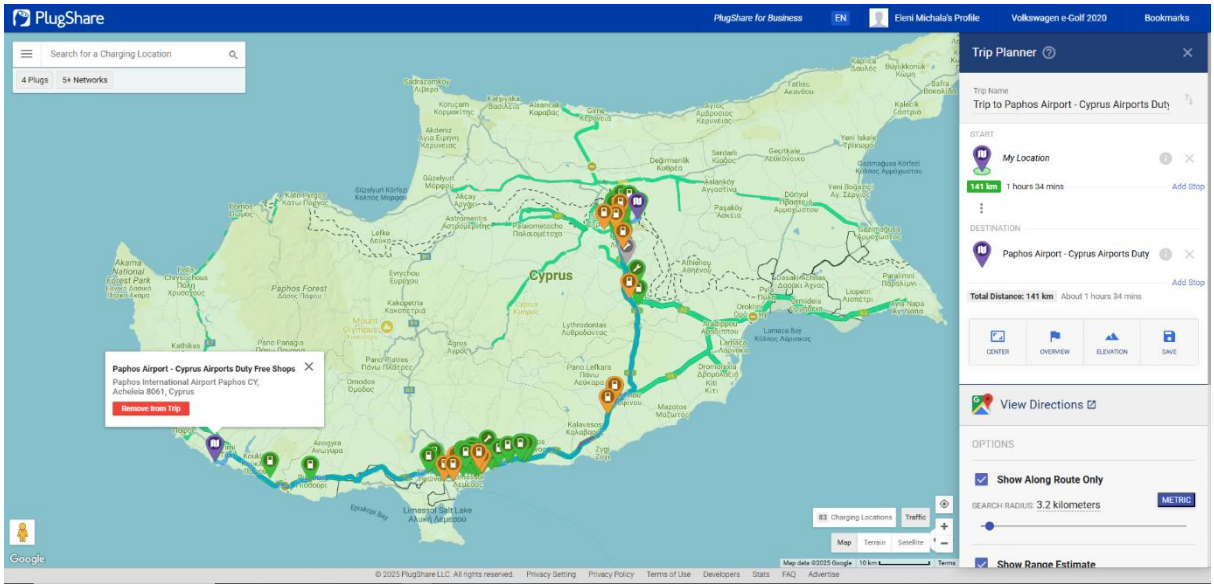
2.1.3 PlugShare

Το PlugShare αποτελεί μία από τις πιο ευρέως χρησιμοποιούμενες πλατφόρμες εντοπισμού σταθμών φόρτισης ηλεκτρικών οχημάτων σε παγκόσμιο επίπεδο. Βασισμένη σε μια προσέγγιση crowdsourcing, η εφαρμογή επιτρέπει στους ίδιους τους χρήστες να προσθέτουν νέους σταθμούς, να βαθμολογούν και να σχολιάζουν τις εμπειρίες τους από τη χρήση των φορτιστών. Μέσω της πλατφόρμας, οι οδηγοί μπορούν να αναζητούν σημεία φόρτισης με βάση φίλτρα όπως ο τύπος πρίζας, η ταχύτητα φόρτισης, ο πάροχος του σταθμού και η κατάσταση διαθεσιμότητας. Επιπλέον, το PlugShare παρέχει χάρτες, φωτογραφίες και οδηγίες πρόσβασης για κάθε σταθμό, ενώ δίνει τη δυνατότητα στους χρήστες να καταγράφουν και το κόστος φόρτισης όπου είναι διαθέσιμο.

Το PlugShare αποτελεί ένα πολύτιμο εργαλείο για την αναζήτηση και επιλογή σταθμών φόρτισης με βάση τη γεωγραφική τοποθεσία και την εμπειρία των χρηστών, αλλά δεν υποστηρίζει επιλογές με βάση περιβαλλοντική πληροφορία.

Στην Εικόνα 2.6 παρουσιάζεται η βασική χαρτογραφική προβολή της εφαρμογής PlugShare, όπου οι χρήστες μπορούν να δουν όλους τους διαθέσιμους σταθμούς φόρτισης σε παγκόσμια κλίμακα, να φιλτράρουν τα αποτελέσματα ανάλογα με τις ανάγκες τους και να επιλέξουν το κατάλληλο σημείο φόρτισης.
Στην Εικόνα 2.7 φαίνεται ένα παράδειγμα δρομολόγησης εντός Κύπρου, όπου η εφαρμογή παρέχει στους οδηγούς προτεινόμενες διαδρομές με ενσωματωμένους σταθμούς φόρτισης, λαμβάνοντας υπόψη τη θέση των σταθμών κατά μήκος της διαδρομής.

Εικόνα 2.6 Χαρτογραφική απεικόνιση σταθμών φόρτισης στην εφαρμογή PlugShare.



Εικόνα 2.7 Παράδειγμα δρομολόγησης με σημεία φόρτισης στην εφαρμογή PlugShare.

2.1.4 IONITY

Η IONITY [14] αποτελεί ένα από τα μεγαλύτερα και ταχύτερα αναπτυσσόμενα δίκτυα ταχυφόρτισης ηλεκτρικών οχημάτων στην Ευρώπη. Ιδρύθηκε το 2017 μέσω συνεργασίας μεγάλων αυτοκινητοβιομηχανιών (BMW Group, Ford Motor Company, Mercedes-Benz Group και Volkswagen Group), με στόχο την ανάπτυξη ενός εκτεταμένου ευρωπαϊκού δικτύου υποδομών φόρτισης υψηλής ισχύος. Το δίκτυο της IONITY επικεντρώνεται στην παροχή σταθμών ταχυφόρτισης με ισχύ έως και 350 kW, επιτρέποντας τη γρήγορη και αποδοτική φόρτιση των ηλεκτρικών οχημάτων κατά μήκος των κύριων ευρωπαϊκών αυτοκινητόδρομων. Η πλατφόρμα προσφέρει στους χρήστες τη δυνατότητα εντοπισμού σταθμών, ελέγχου

διαθεσιμότητας σε πραγματικό χρόνο και πλοήγησης μέσω της επίσημης εφαρμογής ή της ιστοσελίδας. Ένα ιδιαίτερο χαρακτηριστικό της IONITY είναι η δέσμευσή της στη βιωσιμότητα. Όπως αναφέρεται επίσημα από την εταιρεία, όλη η ηλεκτρική ενέργεια που χρησιμοποιείται στους σταθμούς της προέρχεται αποκλειστικά από ανανεώσιμες πηγές, όπως αιολική, ηλιακή και υδροηλεκτρική ενέργεια [15]. Η στρατηγική αυτή ενισχύει το οικολογικό προφίλ της πλατφόρμας και συμβάλλει ουσιαστικά στη μείωση των εκπομπών CO_2 που σχετίζονται με τη μετακίνηση.

Στην Εικόνα 2.8 παρουσιάζεται η χαρτογραφική προβολή της εφαρμογής IONITY, μέσω της οποίας οι χρήστες μπορούν να εντοπίσουν σταθμούς ταχυφόρτισης υψηλής ισχύος σε όλη την Ευρώπη. Η διεπαφή επιτρέπει την επιλογή οχήματος, τον καθορισμό επιπέδων μπαταρίας, καθώς και την προσαρμογή της δρομολόγησης σύμφωνα με τις καιρικές συνθήκες και την προτίμηση δικτύου φόρτισης.

Εικόνα 2.8 Προβολή σταθμών ταχυφόρτισης στην εφαρμογή IONITY.

2.1.5 ChargeMap

Το ChargeMap [16] είναι μια ευρωπαϊκή πλατφόρμα που προσφέρει πληροφορίες για σημεία φόρτισης ηλεκτρικών οχημάτων, με στόχο να διευκολύνει τους οδηγούς EV στον εντοπισμό διαθέσιμων φορτιστών κατά τη διάρκεια των μετακινήσεών τους. Ιδρύθηκε το 2011 και έχει εξελιχθεί σε έναν από τους βασικούς κόμβους πληροφόρησης για το δίκτυο φόρτισης στην

Ευρώπη. Η εφαρμογή ChargeMap επιτρέπει στους χρήστες να αναζητούν σταθμούς φόρτισης με βάση φίλτρα όπως ο τύπος πρίζας, η ισχύς φόρτισης, ο τύπος σταθμού (ταχείας ή κανονικής φόρτισης), ο πάροχος υπηρεσιών και η διαθεσιμότητα σε πραγματικό χρόνο. Η πλατφόρμα επίσης προσφέρει τη δυνατότητα χρήσης του ChargeMap Pass, μιας κάρτας RFID που επιτρέπει την εύκολη πρόσβαση σε διάφορα δίκτυα φόρτισης. Ένα από τα πλεονεκτήματα της ChargeMap είναι η κοινότητα χρηστών της, οι οποίοι συνεισφέρουν ενεργά φωτογραφίες, σχόλια και αξιολογήσεις σταθμών, βελτιώνοντας έτσι την ποιότητα και την αξιοπιστία των παρεχόμενων δεδομένων. Η εφαρμογή παρέχει επίσης πλοήγηση προς τον επιλεγμένο φορτιστή και εμφανίζει εκτιμήσεις για το κόστος φόρτισης, όπου αυτές είναι διαθέσιμες. Ωστόσο, όπως και σε άλλες παρόμοιες υπηρεσίες, το ChargeMap δεν παρέχει πληροφορίες σχετικά με την πηγή της ενέργειας που χρησιμοποιείται στους σταθμούς φόρτισης.

Στην Εικόνα 2.9 παρουσιάζεται η χαρτογραφική προβολή της εφαρμογής ChargeMap, η οποία επιτρέπει στους χρήστες να εντοπίζουν σταθμούς φόρτισης ηλεκτρικών οχημάτων σε όλη την Ευρώπη και άλλες περιοχές.

Εικόνα 2. 9 Χαρτογραφική απεικόνιση σταθμών φόρτισης στην εφαρμογή ChargeMap.

2.2 Επιστημονικές Μελέτες

Υπάρχουν πολυάριθμες επιστημονικές μελέτες που εστιάζουν στη φόρτιση ηλεκτρικών οχημάτων, τη χρήση ανανεώσιμων πηγών ενέργειας, τη διαχείριση μικροδικτύων, καθώς και στη χωροχρονική επεξεργασία δεδομένων που σχετίζονται με τη βιώσιμη μετακίνηση. Στην ενότητα αυτή, παρουσιάζονται οι βασικότερες σχετικές έρευνες που αποτέλεσαν θεωρητικό υπόβαθρο για την ανάπτυξη της παρούσας διπλωματικής εργασίας.

2.2.1 Φόρτιση Ηλεκτρικών Οχημάτων και Ανανεώσιμες Πηγές Ενέργειας

Η παγκόσμια στροφή προς τη βιώσιμη κινητικότητα έχει καταστήσει την ηλεκτροκίνηση έναν από τους βασικούς πυλώνες της ενεργειακής μετάβασης. Τα ηλεκτρικά οχήματα συμβάλλουν σημαντικά στη μείωση των εκπομπών διοξειδίου του άνθρακα σε σύγκριση με τα παραδοσιακά οχήματα εσωτερικής καύσης. Ωστόσο, το συνολικό περιβαλλοντικό όφελος εξαρτάται άμεσα από την πηγή ενέργειας που χρησιμοποιείται για τη φόρτισή τους. Η φόρτιση EVs μέσω παραδοσιακών δικτύων που βασίζονται σε ορυκτά καύσιμα μπορεί να μειώσει ή ακόμα και να αναιρέσει το θετικό περιβαλλοντικό αποτύπωμα της ηλεκτροκίνησης. Για τον λόγο αυτό, η ενσωμάτωση ανανεώσιμων πηγών ενέργειας στη φόρτιση ηλεκτρικών οχημάτων έχει καταστεί επιτακτική. Η δημιουργία μικροδικτύων που συνδυάζουν φωτοβολταϊκά συστήματα, αιολικές γεννήτριες και υποδομές αποθήκευσης ενέργειας προσφέρει μια ολοκληρωμένη λύση για την πράσινη φόρτιση ηλεκτρικών οχημάτων, μειώνοντας την εξάρτηση από το κεντρικό δίκτυο και προωθώντας την τοπική κατανάλωση καθαρής ενέργειας.

Σύμφωνα με τη μελέτη των Kumar, Chattopadhyay και Kar (2024) [17], η ανάπτυξη μικροδικτύων για σταθμούς φόρτισης ηλεκτρικών οχημάτων μπορεί να βελτιώσει την ενεργειακή αυτονομία, να μειώσει τις απώλειες μετάδοσης και να ελαχιστοποιήσει τις εκπομπές ρύπων. Το προτεινόμενο σύστημα συνδυάζει παραγωγή από φωτοβολταϊκά και αιολικά, αποθήκευση ενέργειας, και έναν δυναμικό μηχανισμό διαχείρισης ενέργειας (EMS), ο οποίος βελτιστοποιεί τη φόρτιση μέσω τεχνικών όπως το Model Predictive Control και το Dynamic Programming. Οι τεχνικές αυτές επιτρέπουν την πρόβλεψη της παραγωγής και της κατανάλωσης και τη λήψη αποφάσεων σε πραγματικό χρόνο. Συμπληρωματικά, οι Constantinou, Konstantinidis και Zeinalipour (2022) [19] εξετάζουν πράσινες στρατηγικές προγραμματισμού της κατανάλωσης ενέργειας με στόχο την αυτοκατανάλωση ανανεώσιμων πηγών. Η εργασία τους προτείνει ένα μοντέλο δρομολόγησης ενεργειακών ροών σε οικιακά μικροδίκτυα, με στόχο την ελαχιστοποίηση της εξάρτησης από το δίκτυο και τη μέγιστη

αξιοποίηση της τοπικής παραγωγής. Αν και η μελέτη εστιάζει σε οικιακά περιβάλλοντα, οι αρχές της μπορούν να επεκταθούν και σε υποδομές φόρτισης ηλεκτρικών οχημάτων, όπου η ευθυγράμμιση της ζήτησης με τη διαθεσιμότητα ανανεώσιμης ενέργειας παραμένει κρίσιμη.

Συνοψίζοντας, οι σύγχρονες επιστημονικές εργασίες συγκλίνουν στην ανάγκη ενσωμάτωσης ανανεώσιμων πηγών, δυναμικών στρατηγικών διαχείρισης ενέργειας και τεχνικών προσωποποιημένης πρόβλεψης για τη δημιουργία πραγματικά βιώσιμων υποδομών φόρτισης ηλεκτρικών οχημάτων.

2.2.2 Χωροχρονικά Ερωτήματα k-Πλησιέστερων Γειτόνων (kNN)

Ο αλγόριθμος k-Πλησιέστερων Γειτόνων αποτελεί θεμελιώδη μηχανισμό για την αναζήτηση χωροχρονικών δεδομένων, χρησιμοποιούμενος ευρέως σε εφαρμογές πλοήγησης, αναγνώρισης προτύπων και συστημάτων συστάσεων. Στο κλασικό του σενάριο, το ερώτημα kNN σε ένα στατικό περιβάλλον στοχεύει στον εντοπισμό των k αντικειμένων από ένα σύνολο P που ελαχιστοποιούν την απόσταση από ένα σημείο ενδιαφέροντος q, σύμφωνα με μια προκαθορισμένη μετρική απόστασης, συνήθως την ευκλείδεια απόσταση.

Έχουμε:
$P = \{p_1, p_2, ..., p_n\}$ ένα σύνολο δεδομένων σε R^d,

$q \in R^d$ ένα σημείο ενδιαφέροντος,

τότε το αποτέλεσμα του kNN(q, P) είναι το υποσύνολο $Nk(q) \subseteq P$ όπου:

$$\forall\, p_i \in Nk(q), \forall\, p_j \in P - Nk(q), d(q, p_i) \leq d(q, p_j)$$

δηλαδή όλα τα μέλη του αποτελέσματος είναι πιο κοντά στο q σε σχέση με οποιοδήποτε άλλο σημείο.

Σε δυναμικά περιβάλλοντα, όπως αυτά που προκύπτουν σε σενάρια φόρτισης ηλεκτρικών οχημάτων, όπου τόσο οι χρήστες όσο και οι διαθέσιμοι φορτιστές κινούνται ή αλλάζουν κατάσταση (διαθεσιμότητα, παραγωγή ηλιακής ενέργειας κ.λπ.), υπάρχει ανάγκη για συνεχή ερωτήματα kNN (Continuous kNN - CkNN). Σύμφωνα με την εργασία των Tao και Papadias (2003) [30], η υποστήριξη συνεχών ερωτημάτων βασίζεται στην έννοια των "ασφαλών περιοχών", δηλαδή γεωμετρικών περιοχών γύρω από το σημείο, μέσα στις οποίες το σύνολο

των k πλησιέστερων γειτόνων παραμένει αμετάβλητο και το αποτέλεσμα του kNN δεν χρειάζεται να ανανεωθεί, μειώνοντας δραστικά την υπολογιστική επιβάρυνση.

Παρόλο που αυτή η προσέγγιση επαρκεί για περιβάλλοντα όπου μόνο η γεωγραφική θέση μεταβάλλεται, σε πιο σύνθετα σενάρια απαιτούνται επιπλέον παράμετροι που επηρεάζουν την επιλογή γειτονικών κόμβων, οι οποίες δεν είναι στατικές και δεν μπορούν να μοντελοποιηθούν μόνο με βάση την απόσταση, γεγονός που αυξάνει σημαντικά την πολυπλοκότητα της διαδικασίας. Για να αντιμετωπίσουν τέτοια προβλήματα, οι Chatzimilioudis et al. (2016) [31] εισήγαγαν τεχνικές διανεμημένης επεξεργασίας όλων των kNN ερωτημάτων (all-kNN), βασιζόμενοι σε παράλληλη επεξεργασία σε κατανεμημένα περιβάλλοντα in-memory. Συγκεκριμένα, το All-kNN, για ένα σύνολο δεδομένων O, υπολογίζει για κάθε αντικείμενο $o \in O$ το σύνολο των k πλησιέστερων γειτόνων του μέσα στο ίδιο σύνολο. Η άμεση υλοποίηση του προβλήματος με πλήρη σύγκριση όλων των αντικειμένων οδηγεί σε πολυπλοκότητα $O(n^2)$, γεγονός που καθιστά αναγκαία την εφαρμογή τεχνικών βελτιστοποίησης. Μία προσέγγιση είναι η διαίρεση του χώρου σε πλέγμα, όπου κάθε σημείο ελέγχει μόνο γειτονικά κελιά, μειώνοντας τον υπολογιστικό φόρτο σε $O(n\sqrt{n})$ σε ιδανικές συνθήκες. Επιπλέον, σε περιβάλλοντα με μεγάλο όγκο δεδομένων, όπως τα δίκτυα φόρτισης ηλεκτρικών οχημάτων, προτείνονται διανεμημένες τεχνικές επεξεργασίας, όπου τα δεδομένα χωρίζονται σε partitions και κάθε κόμβος υπολογίζει τοπικά τα kNN ερωτήματα πριν ακολουθήσει φάση συγχώνευσης αποτελεσμάτων. Σε δυναμικά συστήματα όπου τα χαρακτηριστικά των σημείων μεταβάλλονται (π.χ., διαθεσιμότητα φορτιστών ή ηλιακή παραγωγή), το All-kNN απαιτεί συνεχείς ενημερώσεις, καθιστώντας κρίσιμη τη χρήση ευέλικτων δομών δεδομένων και τεχνικών προσαρμοστικού ελέγχου ασφαλών περιοχών για τη μείωση του κόστους ανανέωσης. Αυτή η προσέγγιση αποτρέπει τη συμφόρηση και υποστηρίζει την επεξεργασία μεγάλου αριθμού δυναμικών ερωτημάτων σε πραγματικό χρόνο, γεγονός κρίσιμο για μεγάλης κλίμακας δίκτυα, όπως υποδομές φόρτισης ηλεκτρικών οχημάτων σε πόλεις.

Η κλασική μορφή του kNN βασίζεται αποκλειστικά στη φυσική απόσταση, κάτι που δεν ισχύει σε περιπτώσεις όπου στο σύστημα συμμετέχουν δυναμικοί παράγοντες. Η επεξεργασία χωροχρονικών ερωτημάτων kNN σε δυναμικά περιβάλλοντα, όπου τόσο τα δεδομένα όσο και τα ερωτήματα μεταβάλλονται με το χρόνο, είναι απαραίτητη για εφαρμογές όπως η πλοήγηση ηλεκτρικών οχημάτων. Οι Constantinou, Costa, Konstantinidis, Mokbel και Zeinalipour-Yazti (2024) [26] παρουσίασαν ένα εξειδικευμένο πλαίσιο για την κατάταξη φορτιστών ηλεκτρικών οχημάτων χρησιμοποιώντας Continuous kNN με Estimated Components (CkNN-EC). Σε αντίθεση με τα παραδοσιακά kNN ερωτήματα που βασίζονται αποκλειστικά στη γεωγραφική

απόσταση, το CkNN-EC λαμβάνει υπόψη εκτιμώμενες παραμέτρους. Σε αυτό το μοντέλο, καθένας από τους υποψήφιους γείτονες (φορτιστές) συνοδεύεται από ένα σύνολο εκτιμώμενων παραμέτρων που αντικατοπτρίζουν την κατάστασή του. Οι εκτιμώμενες παράμετροι περιλαμβάνουν (i) το κόστος παρέκκλισης (απώλεια χρόνου ή απόστασης από τη διαδρομή), (ii) την εκτιμώμενη διαθεσιμότητα του φορτιστή (πιθανότητα να είναι ελεύθερος) και (iii) τη διαθεσιμότητα ανανεώσιμης ενέργειας βάσει ηλιακής πρόβλεψης. Η συνολική κατάταξη κάθε φορτιστή παράγεται μέσω μιας σταθμισμένης συνάρτησης κόστους, όπου ο χρήστης μπορεί να ρυθμίζει τα βάρη για να δώσει μεγαλύτερη σημασία, για παράδειγμα, στη χρήση καθαρής ενέργειας έναντι της ελάχιστης παρέκκλισης. Η επιλογή φορτιστών μέσω του CkNN-EC προσαρμόζεται δυναμικά στην κίνηση του οχήματος, στην εξέλιξη των καιρικών συνθηκών, στη μεταβολή της διαθεσιμότητας των σταθμών φόρτισης και στην προτίμηση και προτεραιοποίηση του ίδιου του χρήστη.

2.2.3 Πρόβλεψη Ηλιακής Ενέργειας για Υποδομές Φόρτισης

Η αξιοποίηση της ηλιακής ενέργειας για τη φόρτιση ηλεκτρικών οχημάτων αποτελεί σημαντικό παράγοντα της πράσινης μετάβασης στη βιώσιμη κινητικότητα. Ωστόσο, η αβεβαιότητα της ηλιακής παραγωγής, που επηρεάζεται από παράγοντες όπως η νεφοκάλυψη, η εποχικότητα και η ώρα της ημέρας, δημιουργεί την ανάγκη για αξιόπιστα μοντέλα πρόβλεψης παραγωγής ενέργειας σε υποδομές φόρτισης.

Η πρόβλεψη της ηλιακής παραγωγής σε σταθμούς φόρτισης βασίζεται συνήθως στη χρήση μετεωρολογικών δεδομένων πρόγνωσης και ιστορικών δεδομένων παραγωγής. Σύμφωνα με τη μελέτη των Zhang και Chen (2024) [20], η διαδικασία πρόβλεψης περιλαμβάνει την επεξεργασία μεταβλητών όπως η ολική ηλιακή ακτινοβολία (Global Horizontal Irradiance – GHI), η θερμοκρασία περιβάλλοντος και η νεφοκάλυψη.

Τυπικά, η πρόβλεψη της ηλιακής ενέργειας για έναν σταθμό φόρτισης στηρίζεται στην εξής γενική φόρμουλα:

$$P_{solar}(\mathrm{t}) = \mathrm{GHI}(\mathrm{t}) \times A_{PV} \times n_{PV}$$

όπου:

- $P_{solar}(\mathrm{t})$ είναι η εκτιμώμενη ισχύς εξόδου του φωτοβολταϊκού συστήματος τη χρονική στιγμή t,
- $\mathrm{GHI}(\mathrm{t})$ είναι η ολική ηλιακή ακτινοβολία σε W/m^2,
- A_{PV} είναι το ενεργό εμβαδόν του φωτοβολταϊκού πεδίου (m^2),
- n_{PV} είναι η αποδοτικότητα του φωτοβολταϊκού συστήματος.

Στα πλαίσια της διαχείρισης υποδομών φόρτισης, η ακριβής πρόβλεψη της ηλιακής παραγωγής επιτρέπει:

- Την δυναμική ενημέρωση της διαθεσιμότητας καθαρής ενέργειας σε κάθε σταθμό,
- Τη βελτιστοποίηση της ανάθεσης χρηστών σε φορτιστές με βάση την εκτιμώμενη διαθεσιμότητα πράσινης ενέργειας,
- Την ευθυγράμμιση της φόρτισης EVs με περιόδους υψηλής παραγωγής ηλιακής ενέργειας, μεγιστοποιώντας τη χρήση ανανεώσιμων πηγών.

Η ενσωμάτωση τέτοιων προβλέψεων στην υποδομή διαχείρισης φορτιστών καθιστά δυνατή τη δημιουργία ευφυών, περιβαλλοντικά ευαισθητοποιημένων συστημάτων δρομολόγησης και φόρτισης ηλεκτρικών οχημάτων.

2.2.4 Διαχείριση Μικροδικτύων και Βιώσιμη Ενέργεια

Η διαχείριση μικροδικτύων αποτελεί κεντρικό στοιχείο για την προώθηση της τοπικής ενεργειακής αυτονομίας και τη βιώσιμη φόρτιση ηλεκτρικών οχημάτων. Τα μικροδίκτυα χαρακτηρίζονται ως τοπικά, αυτόνομα ενεργειακά συστήματα που ενσωματώνουν παραγωγικές μονάδες ανανεώσιμης ενέργειας, όπως φωτοβολταϊκά και αιολικά συστήματα, σε συνδυασμό με συστήματα αποθήκευσης ενέργειας και ελεγχόμενα φορτία, συμπεριλαμβανομένων των σταθμών φόρτισης EVs.

Σε ένα τυπικό μικροδίκτυο φόρτισης ηλεκτρικών οχημάτων, η ενεργειακή διαχείριση πρέπει να λαμβάνει υπόψη τη μεταβλητότητα της παραγωγής ανανεώσιμων πηγών, τους περιορισμούς του inverter (όπως η μέγιστη επιτρεπόμενη ισχύς εξόδου), την ωριαία κατανομή της παραγωγής, καθώς και τη σταδιακή υποβάθμιση της αποδοτικότητας λόγω ηλικίας του συστήματος. Οι στρατηγικές ενεργειακής διαχείρισης επιδιώκουν την επίτευξη στόχων όπως η μεγιστοποίηση της χρήσης τοπικής ανανεώσιμης ενέργειας, η ελαχιστοποίηση της εξάρτησης από το κεντρικό δίκτυο, και η διατήρηση της σταθερότητας της τάσης και της συχνότητας.

Στις [27] και [28], οι συγγραφείς παρουσίασαν τα συστήματα Energy Planner (EP) και Green Planner (GP), τα οποία ενσωματώθηκαν στο ευρύτερο πλαίσιο του IMCF+. Και τα δύο αξιοποιούν αλγορίθμους τεχνητής νοημοσύνης, όπως το hill climbing και το simulated annealing, με έμφαση στον "μακροπρόθεσμο" ενεργειακό σχεδιασμό. Αυτό σημαίνει ότι επιδιώκουν να υπολογίσουν ένα συνολικό πλάνο για ολόκληρο το έτος, περιορίζοντας έτσι την ανάγκη για καθημερινούς περίπλοκους υπολογισμούς. Για παράδειγμα, το IMCF+ δημιουργεί

ένα ενεργειακό πλάνο για το νοικοκυριό, λαμβάνοντας υπόψη τον προκαθορισμένο ετήσιο προϋπολογισμό κατανάλωσης (π.χ. 11.500 kWh) και το Rule Automation Workflow (RAW). Ο βασικός στόχος είναι να εντοπιστούν οι κανόνες που πρέπει να τροποποιηθούν ή να απορριφθούν, ώστε η συνολική κατανάλωση να παραμένει εντός του επιθυμητού ορίου. Παράλληλα, ανέπτυξαν και το σύστημα GreenCap [29], το οποίο επικεντρώνεται στον "καθημερινό" σχεδιασμό. Το GreenCap αποσκοπεί στην εύρεση του βέλτιστου τρόπου κατανομής και χρονικής μετατόπισης της χρήσης οικιακών συσκευών μέσα στην ημέρα, με στόχο τη μείωση της ενέργειας που εισάγεται από το δίκτυο. Λαμβάνει υπόψη τόσο τις ώρες αιχμής όσο και τις περιόδους αυξημένης παραγωγής ενέργειας.

Σύμφωνα με τη μελέτη των Liu και Wan (2025) [22], η ολοκληρωμένη βελτιστοποίηση της διαχείρισης μικροδικτύων, μέσω της ευθυγράμμισης της παραγωγής ανανεώσιμων πηγών με την προγραμματισμένη φόρτιση EVs και τη χρήση στρατηγικών απόκρισης ζήτησης, μπορεί να μειώσει σημαντικά την ενεργειακή εξάρτηση από το κεντρικό δίκτυο κατά 29% και να αυξήσει την αποδοτικότητα χρήσης τοπικής ενέργειας. Η προσαρμοστική διαχείριση της φόρτισης σε συνδυασμό με την πρόβλεψη της παραγωγής ανανεώσιμης ενέργειας, επιτρέπει στα μικροδίκτυα να προσφέρουν αξιόπιστες και βιώσιμες λύσεις φόρτισης ηλεκτρικών οχημάτων, ενώ συμβάλλουν και στην επίτευξη των ευρύτερων στόχων της πράσινης μετάβασης.

2.2.5 Τεχνολογίες Vehicle-to-Grid (V2G) και Vehicle-to-Home (V2H)

Η τεχνολογία Vehicle-to-Grid (V2G) και η παραλλαγή της Vehicle-to-Home (V2H) αποτελούν βασικούς μηχανισμούς για τη δυναμική αξιοποίηση της αποθηκευμένης ενέργειας στα ηλεκτρικά οχήματα. Στα σενάρια V2G, τα ηλεκτρικά οχήματα λειτουργούν ως φορητές μονάδες αποθήκευσης, παρέχοντας ενέργεια πίσω στο κεντρικό δίκτυο κατά περιόδους υψηλής ζήτησης ή μειωμένης παραγωγής από ανανεώσιμες πηγές. Αντίστοιχα, με την τεχνολογία V2H, τα οχήματα τροφοδοτούν απευθείας ένα κτίριο ή κατοικία, προσφέροντας αυτονομία και ενεργειακή ευελιξία, ειδικά σε περιόδους διακοπών ρεύματος ή υψηλών τιμών ενέργειας.

Για την υλοποίηση των V2G και V2H απαιτείται αμφίδρομος φορτιστής, ο οποίος μπορεί να υποστηρίξει τόσο ροή ενέργειας από το δίκτυο προς το όχημα (G2V – Grid-to-Vehicle), όσο και αντίστροφα από το όχημα προς το δίκτυο ή το σπίτι (V2G/V2H). Παράλληλα, το όχημα πρέπει να είναι τεχνολογικά συμβατό με πρότυπα αμφίδρομης φόρτισης, όπως το CHAdeMO [24] (κυρίως σε ασιατικά EVs) ή τα νέα πρωτόκολλα ISO 15118 [25] Combined Charging

System (CCS), τα οποία επιτρέπουν έξυπνη επικοινωνία μεταξύ ηλεκτρικού οχήματος και δικτύου.

Σύμφωνα με τους Kumar et al. [23], η τεχνολογία V2G μπορεί να συμβάλει σημαντικά στη σταθερότητα του ηλεκτρικού δικτύου, παρέχοντας υπηρεσίες υποστήριξης συχνότητας (frequency regulation), εξομάλυνσης αιχμών (peak shaving) και εφεδρείας (backup reserves). Η σωστή διαχείριση της φόρτισης και εκφόρτισης είναι κρίσιμη για τη διατήρηση της υγείας των μπαταριών των ηλεκτρικών οχημάτων, καθώς η υπερβολική χρήση V2G μπορεί να επιταχύνει τη γήρανση της μπαταρίας. Έτσι, σύγχρονα συστήματα ενσωματώνουν μηχανισμούς περιορισμού των κύκλων εκφόρτισης και έξυπνους αλγορίθμους βελτιστοποίησης.

Συμπληρωματικά, η μελέτη των Arévalo et al. (2024) [21] αναλύει τη στρατηγική ενσωμάτωση V2G και V2H σε μικροδίκτυα. Μέσω της αμφίδρομης ροής ενέργειας, τα οχ λειτουργούν ως "ενεργοί συμμετέχοντες" στο μικροδίκτυο, προσφέροντας τοπική αποθήκευση ενέργειας, εξισορρόπηση της στοχαστικής παραγωγής από ΑΠΕ και μεγαλύτερη ευελιξία στην κατανάλωση και παροχή ισχύος. Η ενσωμάτωση τεχνολογιών όπως V2G και V2H μειώνει την ανάγκη για στατικές μονάδες αποθήκευσης ενέργειας, οδηγώντας σε οικονομικότερα και πιο βιώσιμα μικροδίκτυα. Παρά τις σημαντικές δυνατότητες, προκλήσεις όπως η τυποποίηση των πρωτοκόλλων επικοινωνίας, η επίπτωση στη διάρκεια ζωής των μπαταριών, και η ανάγκη για ευέλικτες πολιτικές τιμολόγησης ενέργειας, παραμένουν ανοιχτά ζητήματα προς επίλυση.

Κεφάλαιο 3 Αρχιτεκτονική του συστήματος

Κεφάλαιο 3 Αρχιτεκτονική του συστήματος

3.1 Μοντέλο του EcoCharge+

Το EcoCharge+ λειτουργεί πάνω σε ένα έξυπνο και γεωχωρικά κατανεμημένο μοντέλο φόρτισης ηλεκτρικών οχημάτων σε ένα αστικό περιβάλλον. Το οδικό δίκτυο του χρήστη αναπαρίσταται ως ένας κατευθυνόμενος γράφος $G = (V, E)$, όπου V είναι το σύνολο των κόμβων (σημεία ενδιαφέροντος, διασταυρώσεις), καθένας με γεωγραφικές συντεταγμένες, και E είναι το σύνολο των ακμών (δρόμοι), καθεμία με βάρος $w(u, v)$ που εκφράζει το κόστος μετάβασης.

Κάθε ηλεκτρικό όχημα m βρίσκεται σε συγκεκριμένη θέση loc_{m} και έχει συγκεκριμένες απαιτήσεις ενέργειας, ενώ στο δίκτυο υπάρχουν πολλαπλοί φορτιστές $\mathrm{b} \in \mathrm{B}$, ομαδοποιημένοι σε μικροδίκτυα (MicroGrids) $\mathrm{MG_k}$.

Το κάθε MicroGrid συνδέεται είτε με φυσική φωτοβολταϊκή μονάδα (τοπική παραγωγή) είτε με απομακρυσμένες μονάδες μέσω εικονικού net metering. Οι φορτιστές που ανήκουν σε ένα μικροδίκτυο μοιράζονται την ηλιακή ενέργεια που παράγεται κάθε ώρα t, η οποία εκτιμάται με βάση μετεωρολογικά δεδομένα από το OpenWeather API [5] και μοντέλα πρόβλεψης GHI (Global Horizontal Irradiance).

Η παραγόμενη ηλιακή ενέργεια ανά ώρα συμβολίζεται ως $L(t)$, ενώ η αντίστοιχη θεωρητική μέγιστη τιμή $L\_theo(t)$ προκύπτει από τον προφίλ παραγωγής του μήνα και την εγκατεστημένη ισχύ του μικροδικτύου. Η διαθεσιμότητα ενός φορτιστή b την ώρα t, δηλαδή το αν είναι ελεύθερος για χρήση, εκφράζεται ως $A(b, t)$ και εκτιμάται βάσει ιστορικών και πραγματικών δεδομένων.

Το σύστημα αξιολογεί τους φορτιστές βάσει ενός οικολογικού δείκτη (ecoScore), ο οποίος λαμβάνει υπόψη:

- $D(m, b)$: το κόστος παρέκκλισης από τη διαδρομή (σε km ή kWh),
- $S(b, t)$: το ποσοστό της ζήτησης του φορτιστή που μπορεί να καλυφθεί από πράσινη ενέργεια,
- $A(b, t)$: την εκτιμώμενη διαθεσιμότητα φορτιστή.

Το αποτέλεσμα είναι ένας δυναμικός πίνακας O, ο οποίος περιλαμβάνει τους κορυφαίους n φορτιστές, κατανεμημένους είτε σε κοντινή απόσταση είτε σε προτεινόμενα μικροδίκτυα. Ο πίνακας αυτός ανανεώνεται περιοδικά (π.χ. κάθε 5 λεπτά ή όταν το όχημα διανύει απόσταση > 5 km), αποφεύγοντας περιττούς επαναυπολογισμούς.

Η λειτουργία του EcoCharge+ βασίζεται σε παραμέτρους πρόβλεψης και πραγματικής μέτρησης, οι οποίες τροφοδοτούν δυναμικά τον αλγόριθμο απόφασης του συστήματος. Ακολουθεί αναλυτικό μοντέλο για κάθε βασική οντότητα:

1. Ημερήσια και ωριαία παραγωγή ηλιακής ενέργειας

Έστω:

- E_m: Η εκτιμώμενη μηνιαία παραγωγή (kWh) του MicroGrid για τον μήνα m.
- r_h: Η σχετική κατανομή παραγωγής για την ώρα h, σύμφωνα με στατιστικά προφίλ.
- n: Η απόδοση (efficiency) του συστήματος PV.

Η θεωρητική παραγωγή για την ώρα h δίνεται από:

$$L_{theo}(h) = \frac{E_m}{30} \times r_h \times n$$

2. Πρόβλεψη GHI και προσαρμογή στην πραγματικότητα

Το σύστημα υπολογίζει το Clear Sky GHI για κάθε ώρα h με βάση εμπειρική καμπύλη, και στη συνέχεια προσαρμόζει αυτόν τον δείκτη με βάση την νεφοκάλυψη C:

$$GHI_{est} = GHI_{clear}(h) \times (1 - 0.75 \times (\frac{C}{100})^3)$$

Ο παραγόμενος ενεργειακός όγκος (σε kW) προκύπτει από:

$$P_{PV}(h) = \frac{GHI_{est}}{1000} \times P_{inst} \times n$$

όπου P_{inst} είναι η εγκατεστημένη ισχύς (kWp).

3. Κατανομή στους φορτιστές

Κάθε μικροδίκτυο MG_k διαθέτει n φορτιστές. Έστω D_i η ζήτηση του φορτιστή i σε kW και D_{total} το άθροισμα των απαιτήσεων των φορτιστών του grid: $D_{total} = \sum_{i=1}^{n} D_i$

Το μερίδιο πράσινης ενέργειας που αντιστοιχεί σε κάθε φορτιστή δίνεται από:

$$P_i(h) = P_{PV}(h) \times \frac{D_i}{D_{total}}$$

Από αυτό, μπορούμε να υπολογίσουμε το ποσοστό κάλυψης από ηλιακή ενέργεια:

$$S_i(h) = min\left(\frac{P_i(h)}{D_i}, 1\right)$$

3.2 Διατύπωση προβλήματος

Σκοπός του συστήματος EcoCharge+ είναι να προτείνει στους οδηγούς ηλεκτρικών οχημάτων ένα σύνολο εναλλακτικών σημείων φόρτισης, τα οποία μεγιστοποιούν την περιβαλλοντική απόδοση της φόρτισης και ελαχιστοποιούν το κόστος παρέκκλισης από τη διαδρομή του οχήματος. Η επιλογή των φορτιστών βασίζεται στην πρόβλεψη ηλιακής παραγωγής, στη γεωγραφική εγγύτητα προς τον χρήστη, καθώς και στη διαθεσιμότητα του φορτιστή. Η κύρια ιδέα είναι ο υπολογισμός ενός πίνακα Ο υποψηφίων φορτιστών, με βάση την εκτιμώμενη διαθεσιμότητα πράσινης ενέργειας από το μικροδίκτυο στο οποίο ανήκει ο φορτιστής κατά την εκτιμώμενη ώρα άφιξης, την εκτιμώμενη διαθεσιμότητα του φορτιστή και το ενεργειακό και γεωγραφικό κόστος της παρέκκλισης από τη διαδρομή του οχήματος για να φτάσει στον φορτιστή. Παράλληλα, να υποστηρίζει προσωποποιημένα φίλτρα με βάση το επιλεγμένο ηλεκτρικό όχημα του χρήστη, τα χαρακτηριστικά του (όπως χωρητικότητα μπαταρίας, κατανάλωση) και τους υποστηριζόμενους τύπους πρίζας. Έτσι, από το σύνολο των διαθέσιμων φορτιστών, λαμβάνονται υπόψη μόνο όσοι είναι τεχνικά συμβατοί με το όχημα του χρήστη. Το σύστημα εφαρμόζει μία συνάρτηση κατάταξης που λαμβάνει υπόψη τις παραπάνω παραμέτρους με στάθμιση ανάλογα με τις προτιμήσεις του χρήστη (π.χ., έμφαση στην πράσινη φόρτιση έναντι της απόστασης).

Η πρόθεση της εφαρμογής είναι να υποστηρίξει επιλογές που μεγιστοποιούν τη χρήση ανανεώσιμων πηγών, ενώ παράλληλα παραμένουν ρεαλιστικές με βάση τη διαδρομή και τις ανάγκες του οχήματος. Ο στόχος είναι η εξισορρόπηση μεταξύ της περιβαλλοντικής αποδοτικότητας και της πρακτικής ευκολίας φόρτισης.

3.3 Λειτουργικότητα

Το EcoCharge+ έχει σχεδιαστεί ώστε να προσφέρει μια ολοκληρωμένη και δυναμικά προσαρμόσιμη εμπειρία στον χρήστη ηλεκτρικού οχήματος, εστιάζοντας τόσο στην περιβαλλοντική βιωσιμότητα όσο και στην εξατομίκευση της πρότασης φόρτισης. Η βασική λειτουργία του συστήματος είναι η εύρεση και η κατάταξη φορτιστών με βάση προβλεπόμενα δεδομένα ηλιακής παραγωγής, εκτιμώμενη διαθεσιμότητα και απόσταση από τον χρήστη, ενσωματώνοντας παράλληλα τις τεχνικές απαιτήσεις και προτιμήσεις του οχήματος.

Κατά την είσοδο στην εφαρμογή, ο χρήστης μπορεί να επιλέξει το όχημά του από προκαθορισμένη βάση δεδομένων κατασκευαστών και μοντέλων. Το όχημα αυτό φέρει χαρακτηριστικά όπως η χωρητικότητα μπαταρίας, η ενεργειακή κατανάλωση ανά χιλιόμετρο, και οι υποστηριζόμενοι τύποι πρίζας. Τα χαρακτηριστικά αυτά επηρεάζουν άμεσα την αναζήτηση φορτιστών, καθώς το σύστημα φιλτράρει αυτόματα μόνο όσους φορτιστές είναι τεχνικά συμβατοί. Ο χρήστης μπορεί επίσης να διαμορφώσει φίλτρα, όπως μέγιστη απόσταση από τη διαδρομή, ελάχιστο επίπεδο «πράσινης ενέργειας», ή προτεραιότητα σε γρήγορη φόρτιση.

Η εφαρμογή εντοπίζει την τοποθεσία του χρήστη και ανακτά μια προβλεπόμενη διαδρομή προς τον επιθυμητό προορισμό του, είτε μέσω API πλοήγησης είτε μέσω χειροκίνητης εισαγωγής σημείων. Για την προβλεπόμενη ώρα άφιξης σε κάθε πιθανό σημείο φόρτισης, το σύστημα αντλεί από το OpenWeather API την πρόβλεψη νεφοκάλυψης και την συνδυάζει με τα ωριαία και μηνιαία προφίλ παραγωγής κάθε μικροδικτύου. Χρησιμοποιώντας παραμέτρους όπως η απόδοση και η εγκατεστημένη ισχύς, το σύστημα εκτιμά την ηλιακή παραγωγή σε kWh για την αντίστοιχη χρονική στιγμή, παράγοντας έτσι ένα προβλεπόμενο ποσοστό «πράσινης ενέργειας» για κάθε φορτιστή.

Οι φορτιστές είναι οργανωμένοι σε μικροδίκτυα (MicroGrids), τα οποία αναπαριστούν φωτοβολταϊκά συστήματα που παρέχουν ενέργεια σε τοπικούς ή εικονικούς φορτιστές. Κάθε μικροδίκτυο περιέχει λεπτομερή τεχνικά στοιχεία και διαχειρίζεται από έναν διαχειριστή, δηλαδή έναν εξουσιοδοτημένο χρήστη με ρόλο διαχειριστή. Ένας διαχειριστής μπορεί μέσω του dashboard να δημιουργήσει νέα μικροδίκτυα, να καταχωρίσει τεχνικά χαρακτηριστικά και να συνδέσει φορτιστές στο δίκτυό του, ενισχύοντας τη δυνατότητα διαφανούς διαχείρισης τοπικών ενεργειακών κοινοτήτων.

Καθώς το σύστημα συγκεντρώνει όλα τα δεδομένα, υπολογίζει για κάθε φορτιστή ένα ecoScore βάσει τεσσάρων κύριων παραμέτρων: του ποσοστού ενέργειας από ΑΠΕ (L), της εκτιμώμενης διαθεσιμότητας του φορτιστή (A), του κόστους παρέκκλισης από τη διαδρομή (D), και των προτιμήσεων του χρήστη. Ο αλγόριθμος κατάταξης σταθμίζει δυναμικά αυτές τις παραμέτρους και επιστρέφει έναν πίνακα ταξινομημένων φορτιστών, είτε σε επίπεδο μεμονωμένων φορτιστών είτε συνολικά ανά μικροδίκτυο. Η κατάταξη ενημερώνεται δυναμικά ανάλογα με την κίνηση του οχήματος ή την αλλαγή χρονικής στιγμής στο ρυθμιστικό χρονικής πρόβλεψης (forecast slider) που διαθέτει η εφαρμογή. Ο χρήστης μπορεί να μετακινήσει τον δείκτη σε μελλοντικά χρονικά σημεία και να δει πώς μεταβάλλονται τα ποσοστά παραγωγής ηλιακής ενέργειας και, κατ' επέκταση, οι προτεινόμενοι φορτιστές.

Το τελικό αποτέλεσμα παρουσιάζεται στον χρήστη μέσω του διαδραστικού χάρτη. Κάθε προτεινόμενος φορτιστής συνοδεύεται από πληροφορίες όπως τύπος πρίζας, απόσταση, ETA, ισχύς φόρτισης, καθώς και ένδειξη «οικολογικής» απόδοσης. Ο χρήστης μπορεί να επιλέξει έναν φορτιστή και να λάβει οδηγίες πλοήγησης προς αυτόν, ολοκληρώνοντας έτσι έναν κύκλο βιώσιμης και προσωποποιημένης φόρτισης.

3.4 Μετρικές

Το σύστημα EcoCharge+ βασίζεται σε ένα σύνολο από μετρικές που επιτρέπουν την αξιολόγηση και κατάταξη των υποψήφιων φορτιστών με γνώμονα την οικολογική βιωσιμότητα, την τεχνική καταλληλότητα, και την πρακτικότητα της επιλογής. Οι μετρικές αυτές συνδυάζονται για την παραγωγή ενός σύνθετου οικολογικού σκορ (ecoScore) που καθορίζει την κατάταξη των φορτιστών ανά στιγμή.

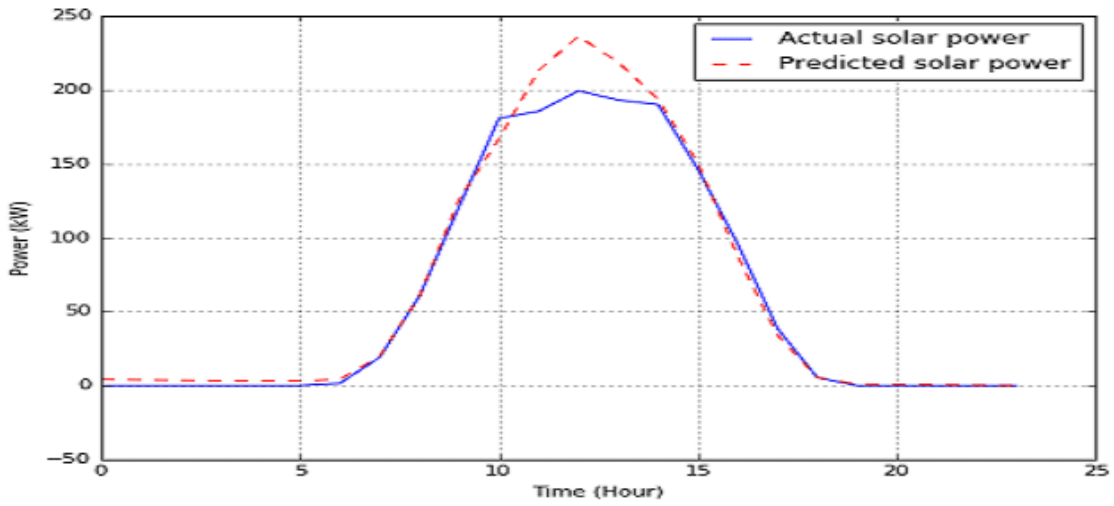
Η σημαντικότερη διάσταση του συστήματος είναι η χρονική: κάθε πρόταση φόρτισης λαμβάνει υπόψη την εκτιμώμενη ώρα άφιξης του οχήματος στον φορτιστή, και προσαρμόζει δυναμικά τις μετρικές με βάση δεδομένα πρόβλεψης έως και 24 ώρες στο μέλλον. Αυτό γίνεται μέσω ενός ρυθμιστικού στοιχείου (forecast slider) που επιτρέπει στον χρήστη να επιλέξει οποιαδήποτε χρονική στιγμή εντός του επόμενου 24ώρου. Για κάθε τέτοια χρονική στιγμή, το σύστημα επαναϋπολογίζει:

- L(b, t): την εκτιμώμενη διαθεσιμότητα πράσινης ενέργειας, όπως αυτή προκύπτει από την πρόβλεψη ηλιακής ακτινοβολίας, η οποία εκφράζεται μέσω της πρόβλεψης νέφωσης, την ώρα της άφιξης t, και το ωριαίο/μηνιαίο προφίλ παραγωγής του

μικροδικτύου (βλ. Εικόνα 3.1). Ο υπολογισμός αυτός λαμβάνει υπόψη και τεχνικά χαρακτηριστικά του μικροδικτύου, όπως απόδοση και εγκατεστημένη ισχύ.

- A(b, t): η εκτιμώμενη διαθεσιμότητα του φορτιστή, η οποία προκύπτει είτε από στατιστικά χρήσης, είτε από πραγματικό χρόνο, όπου είναι διαθέσιμος.
- D(m, b): το κόστος παρέκκλισης του οχήματος m από τη διαδρομή του για να φτάσει στον φορτιστή b, υπολογισμένο σε ενεργειακές μονάδες ή χιλιόμετρα, βάσει της τοποθεσίας και της κατανάλωσης του συγκεκριμένου μοντέλου EV.
- P(u): οι προσωποποιημένες προτιμήσεις του χρήστη u, οι οποίες εκφράζονται ως βάρη στις παραπάνω μετρικές. Για παράδειγμα, ένας χρήστης μπορεί να δώσει μεγαλύτερη έμφαση στην οικολογική καθαρότητα της ενέργειας (L) έναντι της ταχύτητας πρόσβασης (D).

Η πρόγνωση καιρού που περιλαμβάνει η συνιστώσα L(b, t) ανακτάται από cloud-based υπηρεσίες, όπως το OpenWeather, οι οποίες αξιοποιούν προγνωστικά μοντέλα μεγάλης ακρίβειας, όπως το Global Forecast System (GFS) και το European Centre for Medium-Range Weather Forecasts (ECMWF). Τα μοντέλα αυτά παρέχουν ακρίβεια της τάξεως του 95–96% για χρονικό ορίζοντα έως 12 ώρες, και 85–95% για προβλέψεις έως 3 ημέρες, καθιστώντας τη χρήση forecasted δεδομένων αξιόπιστη για τη λήψη αποφάσεων σε πραγματικό χρόνο.



Εικόνα 3. 1 Παράδειγμα παραγωγής ηλιακής ενέργειας ανά ώρα της ημέρας

Ο τελικός τύπος υπολογισμού του ecoScore μπορεί έχει τη μορφή:

$$ecoScore(b) = w_L \times L(b,t) - w_D \times D(m,b) + w_A \times A(b,t)$$

όπου τα w_L, w_D, w_A είναι βάρη που ορίζονται από τον χρήστη ή προεπιλεγμένα από την εφαρμογή.

3.5 Αλγόριθμος

Για να παρέχει δυναμικές και βιώσιμες προτάσεις φόρτισης κατά τη διάρκεια ενός ταξιδιού, ο αλγόριθμος του συστήματος βασίζεται στην ιδέα των **Continuous k-Nearest Neighbors (CkNN)**, προσαρμοσμένη στις ιδιαιτερότητες της πρόβλεψης ηλιακής ενέργειας και της ενεργειακής απόδοσης μικροδικτύων. Η υλοποίηση του αλγορίθμου επηρεάζεται άμεσα από τη μέθοδο **CkNN-EC** που έχει χρησιμοποιηθεί στο παρελθόν σε αντίστοιχα σενάρια βιώσιμης πλοήγησης.

Η βασική λογική είναι ο διαχωρισμός της προγραμματισμένης διαδρομής P σε επιμέρους τμήματα p_i, καθένα από τα οποία αντιπροσωπεύει ένα δυναμικό σημείο εκτίμησης των πλησιέστερων φορτιστών. Σε κάθε τέτοιο σημείο, εφαρμόζεται ο αλγόριθμος CkNN για την εύρεση των υποψήφιων φορτιστών, όπως φαίνεται στην Εικόνα 3.2. Η διαδικασία αξιολόγησης πραγματοποιείται σε δύο φάσεις:

- Στη φάση φιλτραρίσματος, απορρίπτονται όλοι οι φορτιστές που δεν είναι συμβατοί με το όχημα (π.χ. λάθος τύπος πρίζας), όσοι βρίσκονται εκτός της αποδεκτής απόστασης παρέκκλισης, ή όσοι έχουν μηδενική διαθεσιμότητα βάσει ιστορικών δεδομένων.
- Στη φάση βελτιστοποίησης, εκτελείται εκτίμηση για κάθε φορτιστή που πέρασε το φίλτρο με βάση το Eco Score, το οποίο αποτελεί συνάρτηση της προβλεπόμενης διαθεσιμότητας πράσινης ενέργειας, της τοπικής διαθεσιμότητας και του κόστους παρέκκλισης.

```
| Input:
B: σύνολο φορτιστών
P: προγραμματισμένη διαδρομή
m: όχημα χρήστη
weatherData: πρόβλεψη νεφοκάλυψης
| Output:
O: ταξινομημένος πίνακας βιώσιμων φορτιστών

1: ForecastAwareEcoCharging(B, P, m, weatherData)
2:     B′ ← filterCompatibleChargers(B, m.plugType)
3:     B′ ← filterByDistanceAndETA(B′, m.location, P, maxDeviation)
4:     for each b in B′ do
5:         ETA ← estimateETA(m.location, b.location)
6:         L ← forecastSolarProduction(b.microgrid, weatherData, ETA)
7:         A ← estimateAvailability(b, ETA)
8:         D ← estimateDeviationCost(m, b)
9:         SC(b) ← wL * L + wA * A - wD * D
10:    end for
11:    O ← sortDescending(B′, by=SC)
12:    return O
```

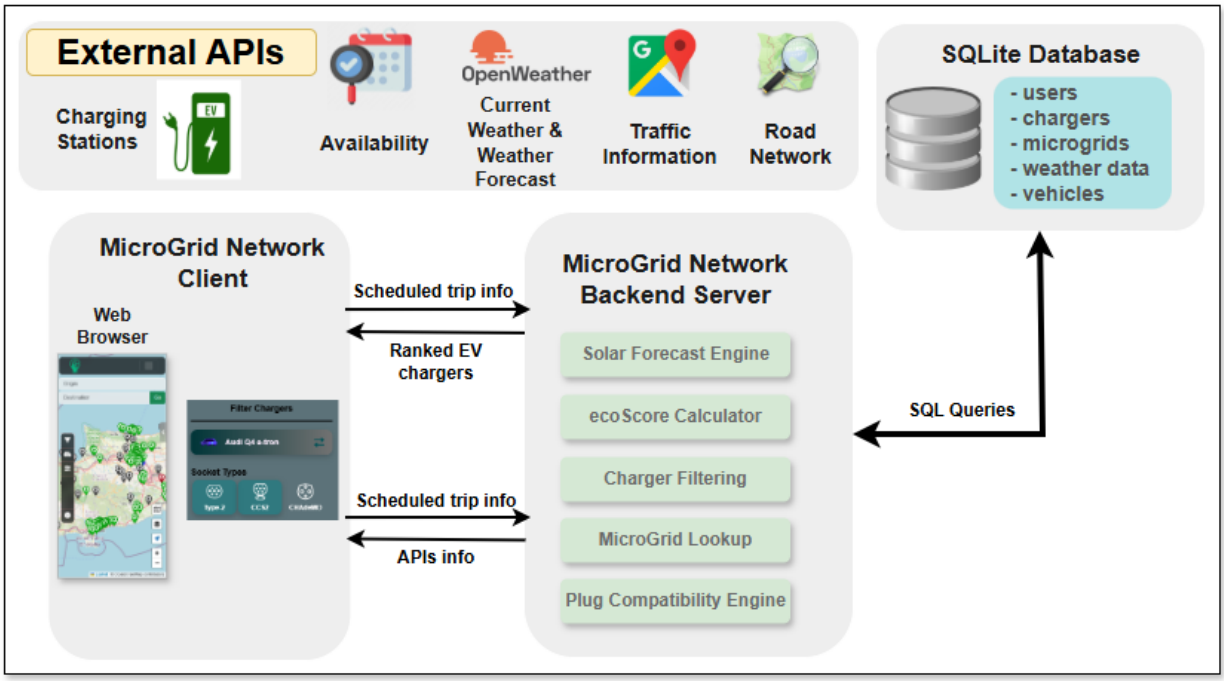
Εικόνα 3. 2 Αλγόριθμος του EcoCharge+

3.6 Αρχιτεκτονική

Η αρχιτεκτονική του συστήματος EcoCharge+ οργανώνεται σε τέσσερα διακριτά επίπεδα: Client, Backend Processing, Database, και External APIs (βλ Εικόνα 3.3). Το σύστημα είναι σχεδιασμένο ώστε να παρέχει σε πραγματικό χρόνο προτάσεις βιώσιμης φόρτισης, αξιοποιώντας προβλεπόμενα και στατικά δεδομένα, καθώς και δυναμικά φίλτρα εξατομίκευσης.

Στο επίπεδο Client, ο τελικός χρήστης αλληλεπιδρά μέσω ενός web-based περιβάλλοντος, όπου μπορεί να επιλέξει το όχημά του, να δηλώσει τις ενεργειακές του ανάγκες και να ρυθμίσει τις προτιμήσεις του για το επίπεδο πράσινης φόρτισης. Επιπλέον, το περιβάλλον περιλαμβάνει οπτικοποίηση φορτιστών στον χάρτη, δυνατότητα διαδραστικού φίλτρου (π.χ. επιλογή plug type, forecast slider), καθώς και επιλογή μικροδικτύου με δυνατότητα ανάθεσης σε Operator. Ο Backend Server, υλοποιημένος σε Flask, δέχεται αιτήματα από τον client και συντονίζει τη ροή δεδομένων. Πραγματοποιεί τον υπολογισμό του ecoScore για κάθε φορτιστή, εφαρμόζει τον αλγόριθμο ταξινόμησης, και διαχειρίζεται την αλληλεπίδραση με την τοπική βάση δεδομένων SQLite. Ο server είναι επίσης υπεύθυνος για τη συγκέντρωση δεδομένων πρόγνωσης καιρού από εξωτερικές υπηρεσίες, καθώς και για την καθημερινή αποθήκευση προβλεπόμενης ηλιακής παραγωγής. Η βάση δεδομένων του συστήματος αποτελείται από

πίνακες όπως: Chargers, MicroGrids, MicroGridHourlyProfile και MonthlyProduction (προφίλ παραγωγής), Users και Vehicles, καθώς και δυναμικά δεδομένα πρόβλεψης. Το επίπεδο των External APIs περιλαμβάνει το OpenWeather API, για πρόβλεψη νεφοκάλυψης με ακρίβεια έως και 95% για το άμεσο μέλλον, την GeoPy για γεωδαιτικό υπολογισμό αποστάσεων, και προαιρετικά APIs από πλατφόρμες όπως PlugShare ή OpenStreetMap για δεδομένα διαθεσιμότητας και χαρτογραφίας.



Εικόνα 3. 3 Αρχιτεκτονική συστήματος

Η αρχιτεκτονική του EcoCharge+ βασίζεται στα θεμέλια της πλατφόρμας EcoCharge [26], αλλά προχωρά σημαντικά βήματα παραπέρα, ενσωματώνοντας λειτουργίες που καθιστούν το σύστημα περισσότερο προσαρμοστικό, προσωποποιημένο και περιβαλλοντικά ευαισθητοποιημένο. Η πρώτη ειδοποιός διαφορά είναι η ενσωμάτωση πρόβλεψης ηλιακής παραγωγής για τις επόμενες 24 ώρες με βάση forecasted δεδομένα νεφοκάλυψης, σε συνδυασμό με το ωριαίο και μηνιαίο ενεργειακό προφίλ κάθε μικροδικτύου, ώστε να υπολογίζει το αναμενόμενο επίπεδο καθαρής ενέργειας για κάθε χρονικό παράθυρο. Προστέθηκε επίσης η έννοια της τοπικής ενεργειακής διαχείρισης μέσω των MicroGrid Operators — χρηστών με ρόλο διαχειριστή, που έχουν τη δυνατότητα να ορίζουν και να ενημερώνουν τις παραμέτρους των μικροδικτύων, να συνδέουν φορτιστές, και να παρακολουθούν την απόδοσή τους. Επιπλέον, η διαδραστική διεπαφή χρήστη περιλαμβάνει έναν ρυθμιστή πρόβλεψης (forecast slider), με τον οποίο ο χρήστης μπορεί να "μετακινηθεί"

χρονικά στο μέλλον και να δει σε ποια χρονική στιγμή προβλέπεται η μεγαλύτερη απόδοση πράσινης ενέργειας ανά φορτιστή ή μικροδίκτυο. Τέλος, το σύστημα ενσωματώνει ένα μοντέλο συμβατότητας οχημάτων, το οποίο επιτρέπει το αυτόματο φιλτράρισμα των φορτιστών με βάση τον τύπο πρίζας που υποστηρίζει το όχημα του χρήστη. Η λειτουργία αυτή εξασφαλίζει ότι οι προτάσεις φόρτισης είναι όχι μόνο βιώσιμες αλλά και τεχνικά υλοποιήσιμες στην πράξη.

Με τις παραπάνω επεκτάσεις, το EcoCharge+ διατηρεί τον βασικό στόχο της πλατφόρμας EcoCharge, τη βελτιστοποίηση της φόρτισης σε ένα ευρύ οδικό δίκτυο, ενώ προσθέτει κρίσιμα στοιχεία προβλεπτικής ανάλυσης, διαχειριστικής ευελιξίας και εξατομικευμένης εμπειρίας.

Κεφάλαιο 4 Υλοποίηση Συστήματος

Η υλοποίηση του EcoCharge+ βασίστηκε σε πολυεπίπεδη αρχιτεκτονική που διαχωρίζει τη διαχείριση δεδομένων, την υπολογιστική λογική και τη διεπαφή χρήστη. Η παρούσα ενότητα παρουσιάζει αναλυτικά την υλοποίηση καθενός από τα τρία επίπεδα.

4.1 Υλοποίηση για το backend

Η υλοποίηση για το backend περιλαμβάνει το επίπεδο δεδομένων και το υπολογιστικό επίπεδο.

4.1.1 Επίπεδο Δεδομένων

Το επίπεδο δεδομένων είναι υπεύθυνο για την αποθήκευση, ανάκτηση και οργάνωση όλων των απαραίτητων πληροφοριών που σχετίζονται με φορτιστές, μικροδίκτυα, προγνώσεις καιρού, παραγωγή ενέργειας και επιλογές χρηστών. Η υλοποίηση βασίζεται σε βάση δεδομένων SQLite.

4.1.1.1 Επιλογή Βάσης Δεδομένων: SQLite

Για την αποθήκευση των δεδομένων του συστήματος επιλέχθηκε η χρήση της SQLite [32], μιας ελαφριάς σχεσιακής βάσης δεδομένων ενσωματωμένης στη γλώσσα προγραμματισμού Python μέσω της βιβλιοθήκης sqlite3.

Η SQLite είναι ιδανική για εφαρμογές όπου δεν απαιτείται η διαχείριση πολλών ταυτόχρονων χρηστών ή η ανάπτυξη σε μεγάλο distributed περιβάλλον. Παρέχει τα πλεονεκτήματα μιας SQL βάσης (σχήμα με πίνακες, σχέσεις, JOIN, GROUP BY, FOREIGN KEYS) χωρίς την ανάγκη εγκατάστασης εξωτερικού server ή υπηρεσίας.

4.1.1.2 Σχεδίαση Σχήματος Βάσης Δεδομένων

Το σχήμα της βάσης δεδομένων σχεδιάστηκε με στόχο να υποστηρίξει πλήρως τις ανάγκες του συστήματος, τόσο σε επίπεδο αποθήκευσης στατικών δεδομένων, όπως τα στοιχεία φορτιστών και τα χαρακτηριστικά των μικροδικτύων, όσο και δυναμικών δεδομένων, όπως είναι η πρόβλεψη ηλιακής παραγωγής και τα καιρικά δεδομένα.

Βασικοί πίνακες:

- MicroGrids

Λειτουργεί ως οντότητα-κλειδί για ομαδοποίηση φορτιστών μέσα σε μικροδίκτυα.

- MicroGridDetails

Περιέχει:

- InstalledCapacity_kWp: εγκατεστημένη ισχύς σε kWp,
- Efficiency: απόδοση μικροδικτύου,
- InverterLimit_kW, DegradationRate, Latitude, Longitude.

- Chargers

Περιλαμβάνει δεδομένα φορτιστών όπως:

- Latitude, Longitude: τοποθεσία,
- Power, Stalls: τεχνικά χαρακτηριστικά,
- LocationName, Address: περιγραφικά στοιχεία τοποθεσίας

- ChargerPlugTypes

Πίνακας many-to-many μεταξύ φορτιστών και τύπων υποδοχής πρίζας. Επιτρέπει φιλτράρισμα βάσει συμβατότητας με το όχημα.

- WeatherForecast

Καταγράφει μετεωρολογικά δεδομένα ανά φορτιστή και χρονική στιγμή:

- Temperature, Clouds, Humidity, WindSpeed,
- timestamp: Ώρα καταγραφής

Χρησιμοποιείται για εκτίμηση παραγωγής.

- MicroGridMonthlyProduction

Αποθηκεύει την εκτιμώμενη μηνιαία παραγωγή ενέργειας (kWh) για κάθε MicroGrid, επιτρέποντας τη μοντελοποίηση της εποχιακής διακύμανσης παραγωγής

- MicroGridHourlyProfile

Ποσοστιαία κατανομή παραγωγής σε 24ωρη βάση (0–1), π.χ. υψηλή παραγωγή 12:00–15:00.

- MicroGridHourlyGeneration

Αληθινή παραγωγή ανά ώρα, ενημερώνεται αυτόματα.

- Vehicles

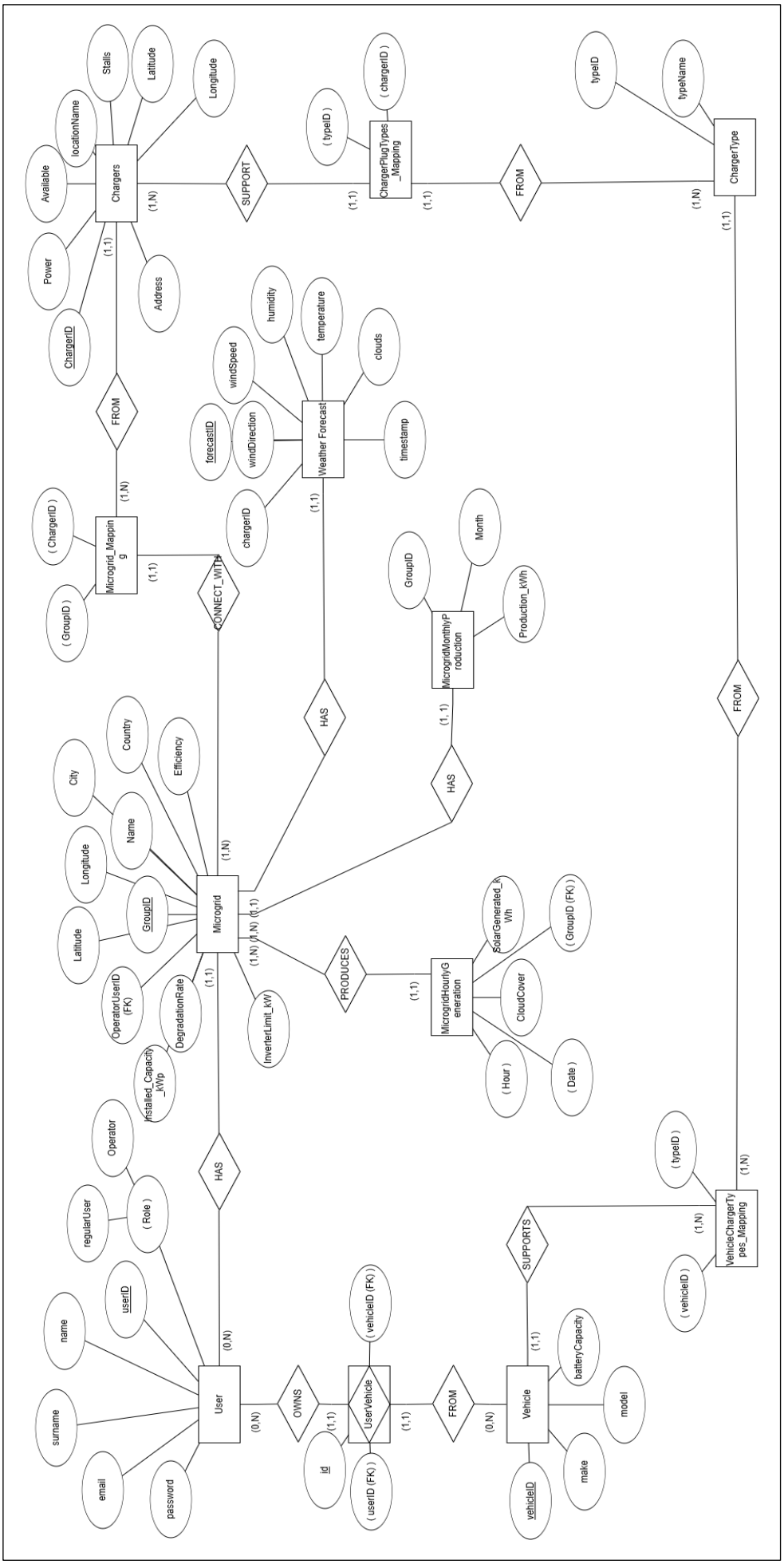
Αποθηκεύει πληροφορίες για ηλεκτρικά οχήματα:

- Make, Model, BatteryCapacity, ConsumptionRate
- Η επιλογή οχήματος επηρεάζει τους υποστηριζόμενους τύπους φορτιστών και τις προτιμήσεις φόρτισης

- userVehicle

Συσχετίζει κάθε χρήστη με το επιλεγμένο όχημά του, ώστε να προσαρμόζονται τα φίλτρα π.χ. υποδοχής.

Η βάση δεδομένων έχει σχεδιαστεί με τρόπο που να αποτυπώνει ρεαλιστικά τις μεταξύ των οντοτήτων σχέσεις. Υπάρχει σχέση 1-προς-πολλά (1-to-many) μεταξύ των πινάκων MicroGrids και Chargers, καθώς κάθε μικροδίκτυο μπορεί να περιλαμβάνει πολλούς φορτιστές. Αντίστοιχα, το κάθε μικροδίκτυο συσχετίζεται με πολλές εγγραφές στους πίνακες MicroGridHourlyGeneration και MicroGridMonthlyProduction, οι οποίοι καταγράφουν την ωριαία και μηνιαία παραγωγή αντίστοιχα, διαμορφώνοντας έτσι μια ακόμη σχέση 1-προς-πολλά. Αντίθετα, μεταξύ Chargers και WeatherForecast υπάρχει μια σχέση 1-προς-1, καθώς για κάθε φορτιστή διατηρείται μία ενεργή εγγραφή πρόβλεψης καιρού ανά χρονική στιγμή, η οποία ενημερώνεται περιοδικά (ανά ώρα). Τέλος, κάθε φορτιστής μπορεί να υποστηρίζει πολλαπλούς τύπους υποδοχών φόρτισης, γεγονός που αποτυπώνεται με μια σχέση 1-προς-πολλά μεταξύ των Chargers και του πίνακα ChargerPlugTypes. Για την καλύτερη κατανόηση της δομής και των μεταξύ τους σχέσεων, στην Εικόνα 4.1 ακολουθεί το Διάγραμμα Οντοτήτων-Συσχετίσεων (ER Diagram) της βάσης δεδομένων.



Εικόνα 4. 1 ER Διάγραμμα Βάσης Δεδομένων

4.1.2 Υπολογιστικό Επίπεδο

Η υπολογιστική λογική του EcoCharge+ υλοποιείται μέσω μιας σειράς τεχνολογιών που συνεργάζονται για την πρόβλεψη παραγωγής, την αξιολόγηση φορτιστών, και τη δυναμική ενημέρωση του backend με περιβαλλοντικά και ενεργειακά δεδομένα. Κάθε τεχνολογία αξιοποιείται για συγκεκριμένες λειτουργίες του συστήματος, όπως περιγράφεται παρακάτω.

4.1.2.1 Python 3

Η Python 3 [33] είναι μια υψηλού επιπέδου, δυναμικά τυποποιημένη γλώσσα προγραμματισμού που έχει καθιερωθεί για την ευκολία χρήσης, τη σαφή σύνταξη και την πλούσια βιβλιοθήκη εργαλείων. Χρησιμοποιείται ευρέως σε τομείς όπως η επιστήμη δεδομένων, η μηχανική μάθηση, ο αυτοματισμός και η ανάπτυξη web εφαρμογών. Στο πλαίσιο της παρούσας εργασίας, η Python 3 αποτέλεσε τη βασική τεχνολογία για την υλοποίηση των υπολογιστικών λειτουργιών του backend, καθώς προσφέρει ευελιξία, επεκτασιμότητα και δυνατότητα γρήγορης ανάπτυξης πρωτοτύπων.

4.1.2.2 sqlite3

Η sqlite3 [34] είναι μια ενσωματωμένη βιβλιοθήκη της Python που επιτρέπει την αλληλεπίδραση με βάσεις δεδομένων SQLite. Παρέχει λειτουργίες για εκτέλεση SQL ερωτημάτων, δημιουργία πινάκων και διαχείριση δεδομένων χωρίς την ανάγκη ξεχωριστού server βάσης.

4.1.2.3 GeoPy

Η βιβλιοθήκη GeoPy [35] είναι ένα ανοικτού κώδικα εργαλείο της Python που παρέχει λειτουργίες γεωχωρικού υπολογισμού και γεωκωδικοποίησης, καθιστώντας εφικτή τη διαχείριση γεωγραφικών συντεταγμένων και την εκτίμηση αποστάσεων μεταξύ σημείων. Χρησιμοποιείται ευρέως σε εφαρμογές πλοήγησης, ανάλυσης θέσης και διαδρομών. Η μέθοδος αυτή υπολογίζει την γεωδαιτική απόσταση λαμβάνοντας υπόψη την καμπυλότητα της Γης, προσφέροντας ακριβέστερα αποτελέσματα σε σχέση με την ευκλείδεια απόσταση.

4.1.2.4 Flask

Για την επικοινωνία μεταξύ backend και frontend χρησιμοποιήθηκε το Flask [36]. Το Flask είναι ένα ελαφρύ και ευέλικτο πλαίσιο εφαρμογών ιστού γραμμένο σε Python, το οποίο χρησιμοποιείται ευρέως για την ανάπτυξη δυναμικών ιστοσελίδων και RESTful υπηρεσιών. Αποτελεί ένα από τα πιο δημοφιλή μικρο-frameworks, καθώς δεν επιβάλλει αυστηρή αρχιτεκτονική και επιτρέπει στον προγραμματιστή να επιλέξει τα εργαλεία και τις βιβλιοθήκες που ταιριάζουν καλύτερα στις ανάγκες της εφαρμογής του. Παρά την απλότητά του, παρέχει όλες τις βασικές λειτουργίες για τη διαχείριση αιτημάτων (requests), προτύπων HTML (templates), συνεδρίες (sessions), καθώς και routing — δηλαδή τη χαρτογράφηση συγκεκριμένων URL σε καθορισμένες συναρτήσεις, επιτρέποντας στην εφαρμογή να ανταποκρίνεται κατάλληλα ανάλογα με τη διαδρομή που ζητά ο χρήστης. Η δημοτικότητά του οφείλεται κυρίως στην απλότητα χρήσης, την καθαρή σύνταξη και την ευκολία ενσωμάτωσης με άλλες τεχνολογίες.

4.1.2.5 flask_session

Η flask_session [37] είναι μια επέκταση του Flask που επιτρέπει τη διαχείριση sessions με αποθήκευση στον server (π.χ. αρχείο, Redis, βάση δεδομένων), προσφέροντας καλύτερο έλεγχο και ασφάλεια σε σύγκριση με την προεπιλεγμένη αποθήκευση cookies.

4.1.2.6 JSON

Η βιβλιοθήκη json [38] της Python επιτρέπει τη μετατροπή δεδομένων μεταξύ της μορφής Python (π.χ. λεξικά, λίστες) και της μορφής JSON (JavaScript Object Notation), που είναι μια ευρέως χρησιμοποιούμενη μορφή ανταλλαγής δεδομένων στο διαδίκτυο.

4.1.2.7 datetime

Η datetime [39] είναι μια ενσωματωμένη βιβλιοθήκη της Python για την αναπαράσταση και διαχείριση ημερομηνιών και χρόνου. Παρέχει αντικείμενα για ημερομηνίες (date), ώρες (time), χρονικές στιγμές (datetime) και χρονικές διαφορές (timedelta).

4.1.2.8 timezone

Η timezone [40] είναι ένα αντικείμενο της datetime που χρησιμοποιείται για να καθορίσει την χρονική ζώνη ενός αντικειμένου datetime, επιτρέποντας τον χειρισμό μετατροπών μεταξύ διαφορετικών ζωνών ώρας.

4.1.2.9 timedelta

Η timedelta [41] είναι μια κλάση της datetime που αναπαριστά μια χρονική διάρκεια. Χρησιμοποιείται για πράξεις προσθαφαίρεσης σε ημερομηνίες και ώρες, π.χ. “5 ώρες αργότερα” ή “2 ημέρες πριν”.

4.1.2.10 ZoneInfo

Η ZoneInfo (από το module zoneinfo) [42] εισήχθη στην Python 3.9 και προσφέρει υποστήριξη για IANA time zones (όπως "Europe/Nicosia"). Χρησιμοποιείται για να δημιουργήσει αντικείμενα timezone βασισμένα σε πραγματικές χρονικές ζώνες με υποστήριξη θερινής ώρας.

4.1.2.11 bcrypt

Η bcrypt [43] είναι μια βιβλιοθήκη κρυπτογράφησης που χρησιμοποιείται για την ασφαλή αποθήκευση και έλεγχο κωδικών πρόσβασης. Βασίζεται στον αλγόριθμο bcrypt, ο οποίος είναι ανθεκτικός σε επιθέσεις brute-force χάρη στη χρήση salt και επαναλήψεων (work factor). Η χρήση του bcrypt διασφαλίζει ότι ακόμα και αν διαρρεύσει η βάση δεδομένων, οι κωδικοί παραμένουν προστατευμένοι.

4.1.2.12 sickit-learn

Η βιβλιοθήκη Scikit-learn [44] είναι ένα από τα πιο διαδεδομένα εργαλεία μηχανικής μάθησης στην Python, παρέχοντας υλοποιήσεις αλγορίθμων για ταξινόμηση, παλινδρόμηση, ομαδοποίηση, μείωση διαστάσεων και προ-επεξεργασία δεδομένων. Ένας από τους αλγορίθμους ομαδοποίησης που περιλαμβάνει είναι ο DBSCAN (Density-Based Spatial Clustering of Applications with Noise), ο οποίος βασίζεται στην έννοια της πυκνότητας: εντοπίζει περιοχές με υψηλή συγκέντρωση σημείων και τις ομαδοποιεί, αγνοώντας σημεία που

βρίσκονται σε αραιές περιοχές (τα οποία χαρακτηρίζει ως θόρυβο). Είναι ιδανικός για χωρικά δεδομένα, καθώς δεν απαιτεί εκ των προτέρων τον αριθμό των ομάδων.

4.1.3 Υλοποίηση

Η διαχείριση δεδομένων στο σύστημα EcoCharge+ βασίζεται σε ένα συνδυασμό στατικών και δυναμικών πηγών. Η εισαγωγή αρχικών δεδομένων και η συνεχιζόμενη ενημέρωση υλοποιούνται με Python scripts, χρονοπρογραμματιστές, όπως cron, και σύνδεση με εξωτερικά APIs. Με στατική εισαγωγή εισήχθηκαν τα αρχικά δεδομένα που αφορούν τους φορτιστές, από JSON αρχείο της παρακάτω μορφής, και με τη χρήση βοηθητικού Python script.


```
{
        "access": 1,
        "address": "QFF5+JMC, Kyriakou Matsi, Konia 8300, Cyprus",
        "available_station_count": null,
        "coming_soon": false,
        "connector_types": [
            "CCS2"
        ],
        "icon": "https://assets.plugshare.com/icons/Y.png",
        "icon_type": "Y",
        "id": 602366,
        "in_use_station_count": null,
        "is_fast_charger": true,
        "latitude": 34.77385368598048,
        "longitude": 32.45897295035705,
        "name": "ENI Konia Station",
        "score": 9.0,
        "station_count": 1,
        "stations": [
            {
                "id": 1572726,
                "outlets": [
                    {
                        "connector": 20,
                        "id": 3572793,
                        "kilowatts": 50.0,
                        "power": 0,
                        "status": null
                    },
                    {
                        "connector": 20,
                        "id": 3572794,
                        "kilowatts": 50.0,
                        "power": 0,
                        "status": null
                    }
                ]
            }
        ],
        "under_repair": false,
        "url": "http://api.plugshare.com/view/location/602366"
},
```


Εικόνα 4.2 Απόσπασμα από το αρχείο "Cyprus.json "

Για την ταξινόμηση των φορτιστών σε ομάδες που αντιπροσωπεύουν τοπικά μικροδίκτυα, εφαρμόστηκε ο αλγόριθμος χωρικής ομαδοποίησης DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Ο DBSCAN επιλέχθηκε διότι επιτρέπει την ανίχνευση γεωγραφικά κοντινών ομάδων χωρίς να απαιτεί προκαθορισμένο αριθμό clusters, ενώ ταυτόχρονα εντοπίζει φορτιστές που δεν ανήκουν σε καμία ομάδα (θόρυβος).

Αρχικά, εξήχθησαν οι συντεταγμένες όλων των φορτιστών από τη βάση δεδομένων. Οι συντεταγμένες μετατράπηκαν σε πίνακα NumPy και μετασχηματίστηκαν σε ακτίνια για να χρησιμοποιηθεί η μετρική Haversine, η οποία υπολογίζει την απόσταση μεταξύ δύο σημείων στην επιφάνεια της σφαίρας (Γης). Η παράμετρος eps του DBSCAN ορίστηκε ώστε να αντιστοιχεί σε ακτίνα 3 χιλιομέτρων, επιτρέποντας την ένωση φορτιστών που βρίσκονται σε μικρή ακτίνα.

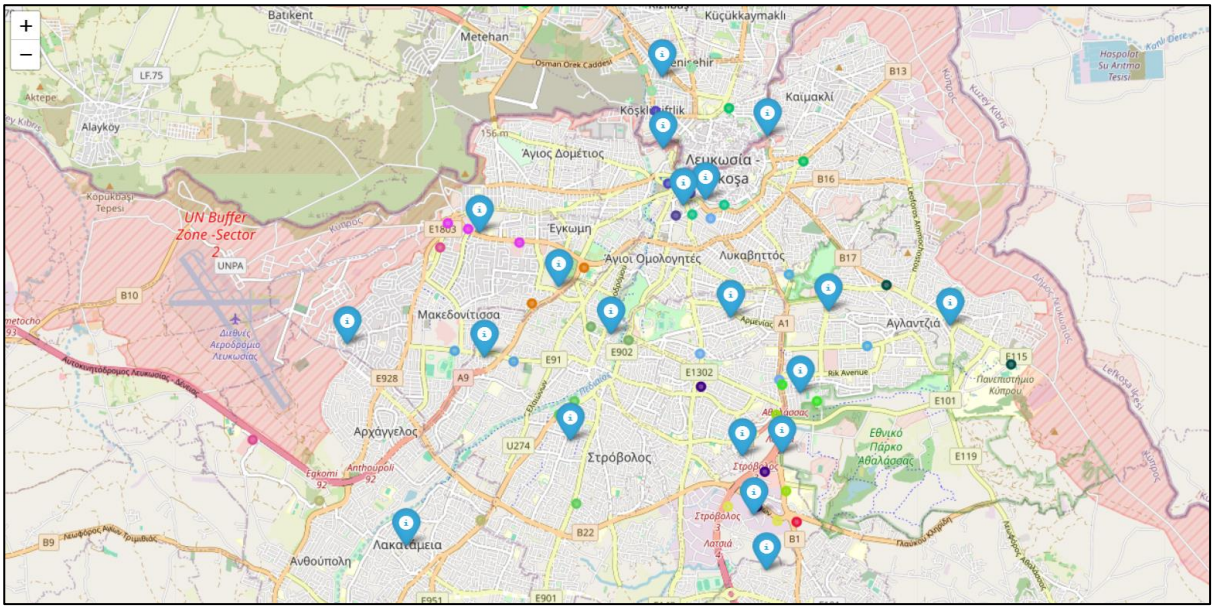
Κατά την εκτέλεση του αλγορίθμου, κάθε φορτιστής αποδόθηκε σε ένα GroupID (ταυτότητα MicroGrid) και η αντιστοίχιση αυτή αποθηκεύτηκε στον πίνακα Microgrids. Φορτιστές που χαρακτηρίστηκαν ως "θόρυβος" (cluster_id = -1) εξαιρέθηκαν από την ανάθεση. Ο παρακάτω πίνακας περιγράφει τη λειτουργία του αλγορίθμου:

```
# Convert to NumPy array for clustering
coords = np.array([[lat, lon] for _, lat, lon in chargers])

# Define DBSCAN clustering with ~3km radius
kms_per_radian = 6371.0088
epsilon = 3 / kms_per_radian

db = DBSCAN(eps=epsilon, min_samples=2, algorithm='ball_tree', metric='haversine')
clusters = db.fit(np.radians(coords))

# Insert results into Microgrids table
for (charger, cluster_id) in zip(chargers, clusters.labels_):
    if cluster_id == -1:
        continue  # Noise (unclustered)
    charger_id = charger[0]
    cursor.execute(
        "INSERT INTO Microgrids (GroupID, ChargerID) VALUES (?, ?)",
        (int(cluster_id), charger_id)  # cast to plain int
    )
```

Εικόνα 4.3 Αλγόριθμος χωρικής ομαδοποίησης DBSCAN και εισαγωγή στη βάση δεδομένων.

Στην Εικόνα 4.4 απεικονίζονται οι φορτιστές ηλεκτρικών οχημάτων στην περιοχή της Λευκωσίας, χρωματικά κωδικοποιημένοι ανάλογα με το GroupID - κοινόχρωμοι δείκτες αντιστοιχούν στο ίδιο μικροδίκτυο. Η ομαδοποίηση αυτή επιτρέπει την αντιστοίχιση κάθε

μικροδικτύου με κοινό ενεργειακό προφίλ, την εφαρμογή ωριαίων ή ημερήσιων προφίλ παραγωγής ανά ομάδα, και τη βελτιστοποίηση του διαμοιρασμού ηλιακής ενέργειας εντός του μικροδικτύου.



Εικόνα 4.4 Οπτικοποίηση των MicroGrids στην περιοχή της Λευκωσίας.

Οι φορτιστές που ταξινομήθηκαν ως θόρυβος από τον αλγόριθμο DBSCAN (δηλαδή δεν ανήκαν σε κάποια ομάδα) αντιστοιχήθηκαν σε ξεχωριστά μικροδίκτυα, καθένα από τα οποία περιλαμβάνει μόνο έναν φορτιστή.

Τα δεδομένα για τα ηλεκτρικά οχήματα που εισήχθησαν στον πίνακα Vehicle προήλθαν από τη δημόσια βάση δεδομένων της υπηρεσίας EV Database, η οποία παρέχει λεπτομερή τεχνικά χαρακτηριστικά για δεκάδες EVs που κυκλοφορούν στην Ευρώπη, όπως χωρητικότητα μπαταρίας, κατανάλωση, και υποστηριζόμενοι τύποι φόρτισης. Η επιλογή των οχημάτων έγινε με βάση τη διαθεσιμότητα στην κυπριακή και ευρωπαϊκή αγορά.

Η εκτίμηση της ηλιακής ενέργειας που παράγεται από κάθε μικροδίκτυο ανά μήνα και η παραγωγή δεδομένων για τον πίνακα MicroGridMonthlyProduction έγινε με βάση τη ζήτηση ισχύος σε κάθε μικροδίκτυο, δηλαδή το άθροισμα της ισχύος των φορτιστών που ανήκουν στο ίδιο μικροδίκτυο. Επιπλέον, η μηνιαία παραγωγή τροποποιείται με βάση εποχιακούς συντελεστές, π.χ. τον Ιούλιο αναμένεται υψηλότερη παραγωγή σε σχέση με τον Ιανουάριο λόγω αυξημένης ηλιοφάνειας.

Η ανάκτηση και επεξεργασία των δεδομένων του συστήματος πραγματοποιείται δυναμικά μέσω ερωτημάτων SQL που εκτελούνται μέσω Python, με στόχο την παροχή όλων των απαραίτητων πληροφοριών στο σύστημα. Συγκεκριμένα, χρησιμοποιούνται ερωτήματα JOIN και GROUP BY για τη δυναμική σύνδεση μεταξύ των πινάκων Chargers, Microgrids και MicroGridDetails, επιτρέποντας τον συσχετισμό κάθε φορτιστή με το αντίστοιχο μικροδίκτυο. Παράλληλα, από τον πίνακα WeatherForecast ανακτώνται οι πιο πρόσφατες μετεωρολογικές συνθήκες ανά φορτιστή. Με βάση αυτά τα δεδομένα, υπολογίζονται κρίσιμοι δείκτες όπως η εκτιμώμενη ηλιακή παραγωγή (kWh) για κάθε μικροδίκτυο, το ποσοστό καθαρής ενέργειας ανά φορτιστή, και το ποσοστό κάλυψης ζήτησης μέσω ηλιακής ενέργειας (solar percentage). Οι δείκτες αυτοί είναι απαραίτητοι για την κατάταξη των φορτιστών και τη δημιουργία προτάσεων βιώσιμης φόρτισης.

```
# Load latest weather data per charger
cursor.execute("""
    SELECT wf.ChargerID, wf.Temperature, wf.Clouds, wf.timestamp, c.power
    FROM WeatherForecast wf
    JOIN Chargers c ON wf.ChargerID = c.id
    WHERE wf.timestamp IN (
        SELECT MAX(timestamp) FROM WeatherForecast GROUP BY ChargerID
    )
    ORDER BY wf.ChargerID;
""")
weather_rows = cursor.fetchall()

# Map ChargerID → GroupID
cursor.execute("SELECT ChargerID, GroupID FROM Microgrids")
charger_to_group = {row[0]: row[1] for row in cursor.fetchall()}

# Map GroupID → MicroGridDetails
cursor.execute("SELECT GroupID, InstalledCapacity_kWp, Efficiency FROM MicroGridDetails")
grid_details = {row[0]: {"capacity": row[1], "efficiency": row[2]} for row in cursor.fetchall()}
```

Εικόνα 4.5 Ανάκτηση πρόσφατων μετεωρολογικών δεδομένων ανά φορτιστή και δυναμική συσχέτιση με τα αντίστοιχα μικροδίκτυα μέσω SQL ερωτημάτων και Python dictionaries.

Ο πίνακας WeatherForecast ενημερώνεται δυναμικά, τρέχοντας ένα python script (update_weather.py) κάθε μία ώρα, με χρήση cron job. Το script ενημερώνει τη βάση δεδομένων με δεδομένα καιρού από το OpenWeather One Call API, για κάθε τοποθεσία φορτιστή.

```python
def fetch_weather_data(lat, lon):
    """
    Fetch current weather data for a given latitude and longitude from OpenWeather API.
    """
    try:
        url = f'http://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API_KEY}&units=metric'
        print(f"Fetching weather data for coordinates: lat={lat}, lon={lon}...")
        response = requests.get(url)
        if response.status_code == 200:
            print("Weather data fetched successfully.")
            return response.json()
        else:
            print(f"Failed to fetch data. HTTP Status Code: {response.status_code}")
            return None
    except requests.RequestException as e:
        print(f"Error fetching weather data: {e}")
        return None
```

Εικόνα 4 6 Fetch weather data from OpenWeather API in 'update_weather.py'

Στην υλοποίηση του συστήματος, το Flask αποτελεί τη βάση του backend, διαχειρίζεται τα εισερχόμενα αιτήματα από το frontend και καλεί τις υπολογιστικές συναρτήσεις για την αξιολόγηση φορτιστών, την πρόβλεψη ηλιακής παραγωγής, καθώς και τη σύνδεση με δεδομένα από τη βάση. Η διασύνδεση επιτυγχάνεται με τη χρήση REST API endpoints.

Αφού εισάγουμε τις απαραίτητες βιβλιοθήκες της Flask (βλ. Εικόνα 4.7), δημιουργούμε ένα instance της εφαρμογής με την εντολή app = Flask(__name__), το οποίο αποτελεί τη βάση για την ανάπτυξη των endpoints του backend. Η Flask αξιοποιείται για την επεξεργασία αιτημάτων που αποστέλλονται από το frontend και την επιστροφή αποτελεσμάτων σε μορφή JSON.

```python
from flask import Flask, request, render_template, redirect, url_for, jsonify, session
from flask_session import Session
import sqlite3
import bcrypt
from datetime import datetime, timezone, timedelta
from geopy.distance import geodesic
from zoneinfo import ZoneInfo
```

Εικόνα 4.7 Εισαγωγή βιβλιοθηκών Flask

```python
# Create a Flask apllication instance named 'app'
app = Flask(__name__)
```

Εικόνα 4.8 Δημιουργία instance του Flask

Παρακάτω περιγράφεται η λειτουργία των βασικών endpoints που συμβάλλουν στην εκτέλεση του κύριου αλγορίθμου του συστήματος.

Με τη φόρτωση του συστήματος ενεργοποιείται το endpoint /chargers, το οποίο φορτώνει όλους τους φορτιστές και τα πεδία τους από τη βάση δεδομένων (βλ. Εικόνα 4.9), υπολογίζει την ηλιακή ενέργεια που παράγεται σε κάθε φορτιστή σε πραγματικό χρόνο καλώντας τη συνάρτηση get_all_weather_data() και την αντιστοιχεί σε κάθε φορτιστή (βλ. Εικόνα 4.10). Οπότε, επιστρέφει τους φορτιστές και τα ηλιακά τους δεδομένα σε μορφή JSON στο frontend, για αρχική απεικόνιση στον χάρτη.

```python
# API to fetch all chargers from DB
@app.route('/api/chargers', methods=['GET'])
def get_chargers():
    conn = connect_db()
    cursor = conn.cursor()

    cursor.execute("""
    SELECT
    c.id,
    c.LocationName AS name,
    GROUP_CONCAT(ct.TypeName) AS types,
    c.Power, c.Available, c.latitude, c.longitude, c.Address, c.stalls
    FROM Chargers c
    LEFT JOIN ChargerPlugTypes cp ON c.id = cp.ChargerID
    LEFT JOIN ChargerType ct ON cp.PlugType = ct.TypeID
    GROUP BY c.id;
    """)
```

Εικόνα 4.9 Απόσπασμα του endpoint /chargers

```python
# Fetch solar power data from the weather API function
solar_data = get_all_weather_data()

# Create a dictionary for quick lookup of solar data by charger_id
solar_map = {s["charger_id"]: s for s in solar_data}

# Merge solar power data with charger data
for charger in chargers_from_db:
    solar_info = solar_map.get(charger["charger_id"])
    if solar_info:
        charger["solar_power_estimate"] = solar_info["solar_power_estimate"]
        charger["grid_power_percentage"] = solar_info["grid_power_percentage"]
    else:
        charger["solar_power_estimate"] = 0
        charger["grid_power_percentage"] = "N/A"

return jsonify(chargers_from_db), 200
```

Εικόνα 4.10 Solar data mapping with chargers

Η συνάρτηση get_all_weather_data() αναλαμβάνει τον υπολογισμό της παραγωγής ηλιακής ενέργειας ανά φορτιστή και ανά μικροδίκτυο. Η παραγωγή κάθε φορτιστή κατανέμεται ανάλογα με το ποσοστό της ισχύος του σε σχέση με τη συνολική ζήτηση του μικροδικτύου, και βασίζεται σε συνδυασμό καιρικών δεδομένων, μηνιαίων εκτιμήσεων παραγωγής ανά μικροδίκτυο, και χρονικών χαρακτηριστικών (ώρα, μήνας). Η τελική παραγωγή υπολογίζεται σε kWh και χρησιμοποιείται για να εκτιμηθεί το ποσοστό κάλυψης της ζήτησης κάθε φορτιστή από ηλιακή ενέργεια. Παρακάτω φαίνεται απόσπασμα της συνάρτησης get_all_weather_data().

```python
for row in weather_rows:
    charger_id, temp, clouds, timestamp_utc, power_demand = row
    power_demand = power_demand or 0
    group_id = charger_to_group.get(charger_id)
    if not group_id:
        continue

    # Retrieve MicroGrid details
    monthly_kwh = monthly_prod.get(group_id, {}).get(current_month, 0)
    monthly_kwh *= derating_factor
    installed_capacity = grid_details.get(group_id, {}).get("capacity", 0)
    efficiency = grid_details.get(group_id, {}).get("efficiency", 1.0)
    hour_fraction = hourly_profile.get(current_hour, 0)

    # Estimate GHI based on cloud coverage and time
    cloud_fraction = min(max(clouds, 0), 100) / 100.0
    clear_sky_ghi = estimate_clear_sky_ghi(current_hour)
    ghi_estimated = estimate_ghi(cloud_fraction, clear_sky_ghi) * hour_fraction

    # Estimate normalized cloud reduction
    cloud_scaling = ghi_estimated / clear_sky_ghi if clear_sky_ghi > 0 else 0

    # Monthly + hourly expected energy (in kWh)
    daily_kwh = monthly_kwh / 30.0 if monthly_kwh else 0
    hourly_group_generation = daily_kwh * efficiency * cloud_scaling

    # Distribute solar energy proportionally to charger demand
    chargers_in_group = group_to_chargers.get(group_id, [])
    total_group_demand = sum(charger_demand_map.get(cid, 0) for cid in chargers_in_group)

    if total_group_demand > 0:
        demand_fraction = power_demand / total_group_demand
    else:
        demand_fraction = 1.0 / len(chargers_in_group) if chargers_in_group else 1.0

    # Actual energy assigned to this charger
    solar_power_now = hourly_group_generation * demand_fraction

    group_hourly_totals[group_id] += solar_power_now

    # Compute solar percentage for this charger
    solar_power_percentage = round(min(solar_power_now / power_demand, 1.0) * 100, 2) if power_demand else 0
```

Εικόνα 4.12 Estimate solar percentage per charger.

Στη συνάρτηση get_all_weather_data() καλούνται δύο άλλες κύριες συναρτήσεις, η estimate_clear_sky_ghi(), στην οποία υπολογίζεται θεωρητικά η μέγιστη ηλιακή ακτινοβολία για κάθε ώρα της ημέρας με βάση ημιτονοειδή μοντελοποίηση μεταξύ ανατολής (06:00) και δύσης (18:00), και η estimate_ghi() η οποία υπολογίζει την πραγματική ηλιακή ακτινοβολία (GHI) που εκτιμάται από τη θεωρητική τιμή, λαμβάνοντας υπόψη την κάλυψη νεφών.

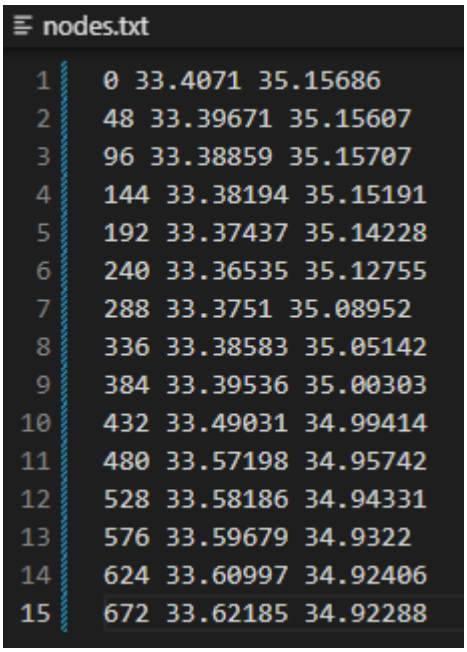
```python
def estimate_clear_sky_ghi(hour, latitude=35.0):
    # Approx: sunrise at 6, sunset at 19
    sunrise = 6
    sunset = 18
    if hour < sunrise or hour >= sunset:
        return 0

    # Normalize hour to [0, π] over the solar day
    hour_angle = (hour - sunrise) / (sunset - sunrise) * math.pi
    return 1000 * math.sin(hour_angle)
```

Εικόνα 4.13 Clear sky GHI estimation.

```python
def estimate_ghi(cloud_coverage_fraction, ghi_clear_sky):
    """
    Estimate GHI based on cloud coverage using empirical formula:
    GHI_estimated = GHI_clear_sky * (1 - 0.75 * (cloud_coverage_fraction)^3)
    """
    return ghi_clear_sky * (1 - 0.75 * (cloud_coverage_fraction) ** 3)
```

Εικόνα 4.14 Actual GHI estimation based on cloud coverage.

Όταν ο χρήστης εισάγει την αρχή και τον προορισμό του, το endpoint /nodes λαμβάνει από το frontend μια λίστα με γεωγραφικά σημεία (γεωγραφικό πλάτος και μήκος) που αντιπροσωπεύουν τη διαδρομή του οχήματος. Το backend διαβάζει τα δεδομένα JSON (π.χ. {"nodesList": [...]}), τα αποθηκεύει σε αρχείο nodes.txt και τα χρησιμοποιεί για να προσδιορίσει αν ο χρήστης παρεκκλίνει από τη βέλτιστη διαδρομή για φόρτιση. Το αρχείο αυτό ανανεώνεται κάθε φορά που ο χρήστης επιλέγει μια διαδρομή. Το αρχείο περιέχει κάθε σημείο της διαδρομής και το διαιρούμε ώστε να έχει περίπου 15 σημεία κατά μήκος της διαδρομής για γρηγορότερους υπολογισμούς. Στην Εικόνα 4.15 φαίνεται ένα παράδειγμα του αρχείου ‘nodes.txt’ για τη διαδρομή Λευκωσίας – Λάρνακας.



Εικόνα 4.15 Παράδειγμα αρχείου 'nodes.txt'

Το endpoint /api/forecast_solar_scores ενεργοποιείται όταν ο χρήστης σύρει το slider πρόβλεψης καιρού, στέλνοντας αιτήματα POST με σκοπό την πρόβλεψη της ποσοστιαίας συμμετοχής της ηλιακής ενέργειας ανά φορτιστή σε μελλοντικές χρονικές στιγμές. Το αίτημα περιέχει forecast_time: χρονική στιγμή ενδιαφέροντος (μορφή "%Y-%m-%d %H:%M:%S") και clouds: λεξικό με τιμές νεφοκάλυψης (cloud cover) ανά charger_id.

Τα δεδομένα πρόβλεψης δεν υπολογίζονται σε πραγματικό χρόνο αλλά έχουν ανακτηθεί εκ των προτέρων από την υπηρεσία 5 Day / 3 Hour Forecast του OpenWeatherMap, η οποία παρέχει μετεωρολογικές προβλέψεις ανά τρίωρο για τις επόμενες 5 ημέρες. Η ανάκτηση πραγματοποιείται μέσω του script daily_forecast_fetcher.py, το οποίο στέλνει HTTP αιτήματα για κάθε φορτιστή (βάσει γεωγραφικών συντεταγμένων). Το script αυτό καλείται περιοδικά (ανά μέρα) μέσω προγραμματισμένης εργασίας cron (cron job) και δημιουργεί ένα JSON αρχείο (daily_forecast.json) που αποθηκεύεται τοπικά στον φάκελο static/forecast. Αυτό το αρχείο διατίθεται από το endpoint /api/daily_forecast και περιέχει για κάθε φορτιστή τις γεωγραφικές του συντεταγμένες και τη λίστα των μελλοντικών ωριαίων προβλέψεων, όπως παρέχεται από το OpenWeather API. Στη συνέχεια, τα δεδομένα αυτά προωθούνται στο /api/forecast_solar_scores, το οποίο εφαρμόζει τον αλγόριθμο πρόβλεψης GHI της συνάρτησης get_all_weather_data(), με διαφορά ότι χρησιμοποιεί την επιλεγμένη μελλοντική ώρα, και όχι την τωρινή. Το αποτέλεσμα περιλαμβάνει, για κάθε φορτιστή, το εκτιμώμενο ποσοστό χρήσης ηλιακής ενέργειας στη μελλοντική του ζήτηση.

```python
@app.route('/api/daily_forecast')
def daily_forecast():
    with open('static/forecast/daily_forecast.json') as f:
        data = json.load(f)
    return jsonify(data)
```

Εικόνα 4.17 ‘ /api/daily_forecast’ endpoint

```python
@app.route('/api/forecast_solar_scores', methods=['POST'])
def forecast_solar_scores():
    try:
        data = request.get_json()
        forecast_time_str = data.get("forecast_time")
        forecast_clouds = data.get("clouds", {})  # {charger_id: clouds_value}

        if not forecast_time_str or not forecast_clouds:
            return jsonify({"error": "Missing forecast_time or clouds data"}), 400

        try:
            forecast_time = datetime.strptime(forecast_time_str, "%Y-%m-%d %H:%M:%S")
            cyprus_time = forecast_time.replace(tzinfo=ZoneInfo("Europe/Nicosia"))
            forecast_hour = cyprus_time.hour
            forecast_month = cyprus_time.month
        except Exception as e:
            return jsonify({"error": f"Invalid time format: {str(e)}"}), 400
```

Εικόνα 4.16 Απόσπασμα από το route για διαχείριση αιτημάτων POST προς το endpoint ’/api/forecast_solar_scores’.

```
{
  "generated_at": "2025-05-06T15:54:06.999644",
  "data": [
    {
      "charger_id": 1,
      "latitude": 34.823502,
      "longitude": 32.390793,
      "forecast": {
        "cod": "200",
        "message": 0,
        "cnt": 40,
        "list": [
          {
            "dt": 1746554400,
            "main": {
              "temp": 20.54,
              "feels_like": 20.66,
              "temp_min": 19.52,
              "temp_max": 20.54,
              "pressure": 1017,
              "sea_level": 1017,
              "grnd_level": 1010,
              "humidity": 77,
              "temp_kf": 1.02
            },
            "weather": [
              {
                "id": 801,
                "main": "Clouds",
                "description": "few clouds",
                "icon": "02n"
              }
            ],
            "clouds": {
              "all": 14
            },
            "wind": {
              "speed": 1.34,
              "deg": 286,
              "gust": 1.74
            },
```


Εικόνα 4.118 Απόσπασμα από το αρχείο 'daily_forecast.json'

Η υλοποίηση της εφαρμογής περιλαμβάνει συγκεκριμένα REST API endpoints τα οποία επιτρέπουν την ανάκτηση, εμφάνιση και ανάλυση των δεδομένων που σχετίζονται με τα MicroGrids.

Ένα βασικό endpoint είναι το /api/microgrids, το οποίο επιστρέφει τα βασικά πεδία όλων των μικροδικτύων, όπως το γεωγραφικό τους πλάτος και μήκος, και τους φορτιστές που ανήκουν σε αυτό. Η πληροφορία αυτή χρησιμοποιείται στην αρχική σελίδα ή στον πίνακα ελέγχου του διαχειριστή για την εμφάνιση συνοπτικών καρτών MicroGrids. Παράλληλα, το frontend έχει τη δυνατότητα να χρησιμοποιήσει τα δεδομένα αυτά για την απεικόνισή τους στον χάρτη.

```python
@app.route('/api/microgrids')
def get_microgrid_clusters():
    conn = sqlite3.connect("MicroGrid.db")
    cursor = conn.cursor()

    # Fetch all chargers and their group
    cursor.execute("""
        SELECT m.GroupID, c.latitude, c.longitude, c.id, c.LocationName
        FROM Microgrids m
        JOIN Chargers c ON c.id = m.ChargerID
    """)
    rows = cursor.fetchall()
    conn.close()

    # Organize by group
    clusters = {}
    for group_id, lat, lon, charger_id, name in rows:
        clusters.setdefault(group_id, []).append({
            "lat": lat,
            "lon": lon,
            "charger_id": charger_id,
            "name": name
        })

    return jsonify(clusters)
```

Εικόνα 4.19 Απόσπασμα από το route για διαχείριση αιτημάτων GET προς το endpoint '/api/microgrids'

Για την εμφάνιση πλήρους πληροφορίας ενός MicroGrid, υπάρχει το endpoint /api/microgrid_details/<group_id>, το οποίο επιστρέφει αναλυτικά στοιχεία του αντίστοιχου MicroGrid, όπως την εγκατεστημένη ισχύ (InstalledCapacity_kWp), τον βαθμό απόδοσης (Efficiency), την ωριαία και μηνιαία παραγωγή, καθώς και τους φορτιστές που είναι συνδεδεμένοι με το συγκεκριμένο group. Το συγκεκριμένο endpoint χρησιμοποιείται όταν ο χρήστης επιλέξει το κουμπί "View Details" από μία κάρτα MicroGrid, ώστε να μεταβεί σε μια σελίδα με πλήρη στατιστικά και τεχνικά δεδομένα του συστήματος.

Το endpoint /nearby_chargers (βλ. Εικόνα 4.20) αποτελεί ένα από τα σημαντικότερα REST API σημεία στο σύστημα, καθώς είναι υπεύθυνο για την εξυπηρέτηση του βασικού σκοπού της εφαρμογής: την εύρεση των οικολογικά βέλτιστων φορτιστών σε μια συγκεκριμένη γεωγραφική ακτίνα γύρω από τη θέση του οχήματος. Η κλήση προς το /nearby_chargers πραγματοποιείται μέσω HTTP POST αιτήματος και περιλαμβάνει στο σώμα του δεδομένα JSON, τα οποία περιέχουν τις συντεταγμένες του οχήματος (γεωγραφικό πλάτος και μήκος), το κόστος απομάκρυνσης από την αρχική πορεία (derouting_cost), τη διαθεσιμότητα φορτιστών, το επιθυμητό επίπεδο βιωσιμότητας (συνήθως ως ποσοστό ηλιακής κάλυψης) και την ακτίνα αναζήτησης (σε km). Όλα αυτά τα δεδομένα χρησιμοποιούνται για την προσωρινή φιλτραρισμένη αξιολόγηση φορτιστών. Το endpoint, μόλις λάβει το αίτημα, περνά τις παραμέτρους αυτές σε μία υπολογιστική συνάρτηση κατάταξης, η οποία υλοποιεί έναν αλγόριθμο αξιολόγησης. Ο αλγόριθμος αυτός συνυπολογίζει παράγοντες όπως η απόσταση κάθε φορτιστή από το όχημα (χρησιμοποιώντας γεωδαιτικές αποστάσεις μέσω της GeoPy), το ποσοστό ηλιακής ενέργειας που είναι διαθέσιμο ανά φορτιστή ή ανά μικροδίκτυο, τη ζήτηση ενέργειας του φορτιστή, τη γενική διαθεσιμότητα του σημείου φόρτισης, καθώς και τη συμβατότητα με το επιλεγόμενο τύπο φορτιστή. Το αποτέλεσμα του endpoint είναι μία λίστα με τους κορυφαίους 5 φορτιστές, οι οποίοι επιστρέφονται με φθίνουσα σειρά eco_score. Κάθε φορτιστής περιέχει επίσης συμπληρωματικά στοιχεία, όπως εκτιμώμενο ETA (χρόνος προσέγγισης) και ποσοστό χρήσης ηλιακής ενέργειας. Αυτά τα αποτελέσματα χρησιμοποιούνται για την άμεση ενημέρωση της διεπαφής του χρήστη.

```python
@app.route('/nearby_chargers', methods=['POST'])
def get_nearby_chargers():
    data = request.get_json()

    lat = float(data['lat'])
    lng = float(data['lng'])
    derouting_cost = float(data['derouting_cost'])
    charger_availability = float(data['charger_availability'])
    sustainable_charging_level = float(data['sustainable_charging_level'])
    radius = float(data['radius'])

    # Get charger and solar data
    chargers = get_chargers()[0].get_json()

    selected_types = set(data.get("plug_types", []))
    if selected_types:
        chargers = [c for c in chargers if selected_types & set(c.get("types", []))]

    origin = (lat, lng)
```

Εικόνα 4.20 Απόσπασμα από το route για διαχείριση αιτημάτων POST προς το endpoint '/nearby_chargers'

Κατά την αξιολόγηση φορτιστών στο endpoint /nearby_chargers, κάθε φορτιστής c αξιολογείται ως προς την απόστασή του από τον χρήστη u, μέσω της GeoPy:

```
dist = geodesic((user_lat, user_lon), (c["latitude"], c["longitude"])).km
```

Το endpoint /forecasted_ranked_chargers υλοποιεί την πρόβλεψη και κατάταξη φορτιστών ηλεκτρικών οχημάτων για μια μελλοντική χρονική στιγμή, με βάση δεδομένα πρόγνωσης ηλιακής παραγωγής. Ουσιαστικά, λειτουργεί με την ίδια λογική και υπολογιστικό αλγόριθμο όπως και το endpoint /nearby_chargers, με τη διαφορά ότι αντί για πραγματικά δεδομένα ηλιακής ενέργειας, χρησιμοποιεί προβλεπόμενα δεδομένα μεταγενέστερης ώρας για τον καθορισμό του EcoScore κάθε φορτιστή.

```python
@app.route('/forecasted_ranked_chargers', methods=['POST'])
def forecasted_ranked_chargers():
    try:
        data = request.get_json()
        lat = float(data['lat'])
        lng = float(data['lng'])
        derouting_cost = float(data['derouting_cost'])
        charger_availability = float(data['charger_availability'])
        sustainable_charging_level = float(data['sustainable_charging_level'])
        radius = float(data['radius'])
        forecast_time = data['forecast_time']
```

Εκτός από τα endpoints που σχετίζονται άμεσα με τον βασικό αλγόριθμο αξιολόγησης φορτιστών και πρόβλεψης παραγωγής, το σύστημα περιλαμβάνει και επιπλέον REST API endpoints που υποστηρίζουν τη γενικότερη λειτουργικότητα της εφαρμογής, όπως τη διαχείριση χρηστών (Authentication & Sessions), τη διαχείριση οχημάτων από τους χρήστες, και τη διαχείριση MicroGrids από τους υπεύθυνους διαχειριστές του συστήματος.

Η διαχείριση χρηστών περιλαμβάνει endpoints που επιτρέπουν την εγγραφή, είσοδο και αποσύνδεση των χρηστών, καθώς και την αποθήκευση και ανάκτηση στοιχείων ταυτότητας μέσω session μεταβλητών. Το /signup επιτρέπει τη δημιουργία νέου λογαριασμού είτε ως απλός χρήστης είτε ως operator. Το /signin υλοποιεί τον έλεγχο των credentials και δημιουργεί session για τον χρήστη, ενώ το /logout τερματίζει το session

Οι χρήστες μπορούν να αποθηκεύσουν και να διαχειριστούν τα οχήματά τους μέσω των endpoints /api/select_vehicle, /api/user_vehicle και /api/delete_vehicle. Το πρώτο χρησιμοποιείται για να αποθηκευτεί η επιλογή ενός οχήματος που πραγματοποιείται μέσα από

την εφαρμογή, ενώ το δεύτερο επιστρέφει όλα τα οχήματα που έχει αποθηκεύσει ο χρήστης, μαζί με πληροφορίες όπως χωρητικότητα μπαταρίας, κατανάλωση και υποστηριζόμενοι τύποι φορτιστών. Τέλος, μέσω του /api/delete_vehicle, ο χρήστης μπορεί να αφαιρέσει κάποιο όχημα από το προφίλ του.

Οι διαχειριστές (operators) μπορούν να δημιουργήσουν και να διαχειριστούν MicroGrids μέσω του endpoint /api/create_microgrid, το οποίο καταχωρεί ένα νέο μικροδίκτυο, προσθέτει σχετικούς φορτιστές και αποθηκεύει προφίλ παραγωγής και πρόγνωση καιρού. Τα δικά τους MicroGrids μπορούν να τα ανακτήσουν μέσω του /api/my_microgrids. Πιο αναλυτικά δεδομένα για κάθε MicroGrid επιστρέφονται μέσω του /api/microgrid_details/<group_id> (πλήρης προβολή σελίδας) και του /api/microgrid/<group_id> (συνοπτική έκδοση). Οπτικοποίηση ωριαίας παραγωγής για την τρέχουσα ημέρα προσφέρεται μέσω του /api/microgrid_hourly_today/<group_id>, ενώ το /api/update_microgrid επιτρέπει την ενημέρωση βασικών χαρακτηριστικών ενός ήδη καταχωρημένου MicroGrid.

Για να εκτελεστεί επιτυχώς το αρχείο app.py, πρέπει να έχουν εγκατασταθεί οι ακόλουθες βιβλιοθήκες Python:

```
sudo apt update
sudo apt install python3-pip
sudo pip3 install flask
sudo pip3 install flask-session
sudo pip3 install geopy
sudo pip3 install bcrypt
sudo pip3 install requests
```

Για την εκτέλεση:

```
python3 app.py
```

4.2 Υλοποίηση για το frontend

Ο κώδικας για το frontend βρίσκεται στο αρχείο "index.html" και είναι γραμμένος σε HTML, CSS και JavaScript.

4.2.1 HTML

Η HTML (HyperText Markup Language) [45] είναι η βασική γλώσσα σήμανσης που χρησιμοποιείται για τη δημιουργία και τη δομή ιστοσελίδων. Μέσω των HTML στοιχείων (tags), μπορούμε να οργανώσουμε το περιεχόμενο της σελίδας, όπως τίτλους, παραγράφους, εικόνες, πίνακες, φόρμες και συνδέσμους. Κάθε σελίδα στο διαδίκτυο βασίζεται σε HTML για να αποδώσει το περιεχόμενό της και να υποδείξει στον φυλλομετρητή πώς να το εμφανίσει.

4.2.2 CSS

Η CSS (Cascading Style Sheets) [46] είναι η γλώσσα που χρησιμοποιείται για τον καθορισμό της εμφάνισης και της διάταξης των HTML στοιχείων. Μέσω της CSS μπορούμε να ορίσουμε χρώματα, γραμματοσειρές, αποστάσεις, περιθώρια, θέσεις και άλλα οπτικά χαρακτηριστικά για κάθε στοιχείο της σελίδας. Η χρήση της CSS διαχωρίζει το περιεχόμενο από την παρουσίαση και επιτρέπει τη δημιουργία αισθητικά ευχάριστων και φιλικών προς τον χρήστη διεπαφών.

4.2.3 JavaScript

Η JavaScript [47] είναι μία γλώσσα προγραμματισμού που χρησιμοποιείται στον frontend για την προσθήκη διαδραστικών λειτουργιών σε ιστοσελίδες. Με τη χρήση JavaScript, μπορούμε να χειριστούμε δυναμικά στοιχεία της σελίδας, να ανταποκρινόμαστε σε ενέργειες του χρήστη, καθώς και να αποστέλλουμε και να λαμβάνουμε δεδομένα από τον server μέσω αιτημάτων HTTP (AJAX ή Fetch API), χωρίς την ανάγκη ανανέωσης της σελίδας.

4.2.4 Leaflet

Η Leaflet [48] είναι μία από τις πιο διαδεδομένες βιβλιοθήκες JavaScript ανοιχτού κώδικα για την ενσωμάτωση διαδραστικών χαρτών σε ιστοσελίδες και web εφαρμογές. Χρησιμοποιήθηκε στην παρούσα εφαρμογή για την απεικόνιση των σταθμών φόρτισης, των MicroGrids και τη δυναμική εμφάνιση διαδρομών με markers, layers και tooltips στον χάρτη. Αναπτύχθηκε από τον Volodymyr Agafonkin και είναι σχεδιασμένη με έμφαση στην απλότητα, την ευκολία χρήσης και την υψηλή απόδοση ακόμη και σε φορητές συσκευές. Παρόλο που έχει μέγεθος μικρότερο από 50 KB, παρέχει πληθώρα λειτουργιών χαρτογράφησης όπως προσθήκη

markers, πολυγώνων, υποστήριξη zoom, pan και interaction, υποστήριξη tile layers από παρόχους όπως OpenStreetMap και Mapbox, ευέλικτο API για επέκταση μέσω plugins.
Η ενσωμάτωσή της έγινε με την εισαγωγή των παρακάτω αρχείων στο “index.html”:

- `<link rel="stylesheet" ref="https://unpkg.com/leaflet/dist/leaflet.css"/>`
- `<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>`

4.2.5 Leaflet Routing Machine

Για τη δυνατότητα πλοήγησης και δημιουργίας διαδρομών στην εφαρμογή μας, αξιοποιήθηκε το plugin Leaflet Routing Machine [49] της βιβλιοθήκης Leaflet.
Το Leaflet Routing Machine είναι ένα ελαφρύ και εύχρηστο plugin που επιτρέπει τη δρομολόγηση πάνω σε Leaflet χάρτες. Υποστηρίζει καθορισμό σημείων εκκίνησης και προορισμού, απεικόνιση της βέλτιστης διαδρομής, και παρέχει πληροφορίες όπως η απόσταση και ο εκτιμώμενος χρόνος ταξιδιού. Η ενσωμάτωσή της στην εφαρμογή μας επιτρέπει στον χρήστη να βλέπει σε πραγματικό χρόνο την προτεινόμενη διαδρομή για να φτάσει στους κατάλληλους σταθμούς φόρτισης. Το plugin υποστηρίζει πληθώρα εξωτερικών παρόχων υπηρεσιών δρομολόγησης, όπως OSRM, Mapbox Directions API, GraphHopper, κ.ά. Στην παρούσα εργασία χρησιμοποιήθηκε ο ανοιχτού κώδικα OSRM (Open Source Routing Machine), ο οποίος παρέχει γρήγορη απόκριση και υποστήριξη για δρομολόγηση σε οδικά δίκτυα.

Η Leaflet Routing Machine παρέχει έτοιμες διεπαφές για εμφάνιση κουμπιών, markers και αλλαγή της διαδρομής με μετακίνηση των σημείων πάνω στον χάρτη. Υποστηρίζει επίσης callbacks για προσαρμοσμένη επεξεργασία και εμφάνιση των διαδρομών, κάτι που αξιοποιήθηκε στην εφαρμογή μας για την αλληλεπίδραση με τους σταθμούς φόρτισης.

Για την ενσωμάτωση του plugin προστέθηκαν τα ακόλουθα στο αρχείο “index.html”:

- `<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet-routing-machine@latest/dist/leaflet-routing-machine.css" />`
- `<script src="https://cdn.jsdelivr.net/npm/leaflet-routing-machine@latest/dist/leaflet-routing-machine.min.js"></script>`

4.2.6 Leaflet.LocateControl

Η Leaflet.LocateControl [50] είναι plugin της Leaflet που παρέχει δυνατότητα εντοπισμού της τρέχουσας τοποθεσίας του χρήστη μέσω της λειτουργίας geolocation του browser. Προσφέρει κουμπί στον χάρτη που μετακινεί την προβολή στο σημείο του χρήστη.
Για την ενσωμάτωση των αρχείων CSS και JavaScript Leaflet.Locate προστέθηκαν τα ακόλουθα στο αρχείο "index.html":

- `<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol@0.78.0/dist/L.Control.Locate.min.css" />`
- `<script src="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol@0.78.0/dist/L.Control.Locate.min.js"></script>`

4.2.7 Bootstrap

Για την εμφάνιση και τη διάταξη του περιεχομένου της εφαρμογής μας χρησιμοποιήθηκε το Bootstrap 5.3.0 [51]. Το Bootstrap είναι ένα από τα πιο διαδεδομένα CSS frameworks ανοικτού κώδικα, το οποίο παρέχει έτοιμα πρότυπα για τη δημιουργία responsive και αισθητικά ευχάριστων ιστοσελίδων. Αναπτύχθηκε αρχικά από προγραμματιστές του Twitter και κυκλοφόρησε το 2011, αποκτώντας ευρεία αποδοχή από την κοινότητα του web development. Περιλαμβάνει έτοιμες κλάσεις για διάταξη, κουμπιά, φόρμες και άλλα συστατικά της σελίδας.
Για την ενσωμάτωσή του προστέθηκαν τα ακόλουθα αρχεία στο "index.html":

- `<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"/>`
- `<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>`

4.2.8 jQuery

Η jQuery [52] είναι μια δημοφιλής βιβλιοθήκη JavaScript που απλοποιεί τον χειρισμό του DOM (Document Object Model), τα AJAX (Asynchronous JavaScript and XML) αιτήματα και τις αλληλεπιδράσεις με τη σελίδα. Χρησιμοποιείται για να κάνει τον κώδικα πιο σύντομο και ευανάγνωστο, ενώ προσφέρει ευκολία στη συμβατότητα μεταξύ browser.

Για την ενσωμάτωση του JavaScript αρχείου της jQuery προστέθηκε το ακόλουθο στο “index.html”:

- `<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>`

4.2.9 Χάρτες

Ο βασικός χάρτης της εφαρμογής προέρχεται από το OpenStreetMap [57], μία δωρεάν γεωγραφική βάση δεδομένων που διατηρείται από κοινότητα εθελοντών μέσω ανοικτής συνεργασίας. Το OpenStreetMap παρέχει ελεύθερα διαθέσιμους χάρτες και γεωχωρικά δεδομένα που μπορούν να χρησιμοποιηθούν για οποιαδήποτε εφαρμογή πλοήγησης ή χωρικής απεικόνισης.
Πέρα από το OpenStreetMap, η εφαρμογή ενσωματώνει πολλαπλά επίπεδα (tile layers) τα οποία ο χρήστης μπορεί να εναλλάξει ανάλογα με τις προτιμήσεις του. Συγκεκριμένα υποστηρίζονται:

- OpenTopoMap [53]: βασισμένος στο OpenStreetMap, δίνει έμφαση στην απεικόνιση της ανάγλυφης διαμόρφωσης του εδάφους.
- Esri World Street Map και Esri World Topo Map [54]: προσφέρονται από την Esri και προέρχονται από τη βάση του ArcGIS. Παρέχουν λεπτομερείς χάρτες δρόμων και τοπογραφίας.
- CARTO Light και Dark [53]: διανυσματικά layers που προσφέρουν καθαρό σχεδιασμό για εφαρμογές που θέλουν να τονίσουν τα δεδομένα πάνω από τον χάρτη, ιδανικά για night mode ή minimal UI.

Η υλοποίηση έγινε με χρήση της Leaflet βιβλιοθήκης. Η προσθήκη των layers έγινε μέσω της συνάρτησης L.tileLayer() της Leaflet, όπου δηλώνονται τα URL templates και τα attribution metadata. Ο χρήστης μπορεί να επιλέξει δυναμικά οποιοδήποτε από αυτά τα επίπεδα βάσης κατά τη χρήση του χάρτη.

4.2.10 Plotly.js

Η Plotly.js [56] είναι βιβλιοθήκη JavaScript για διαδραστική απεικόνιση δεδομένων μέσω γραφημάτων. Υποστηρίζει γραφήματα γραμμών, ράβδων, πίτας και πολλά άλλα, με δυνατότητα zoom, hover και φιλτραρίσματος. Είναι ιδανική για την παρουσίαση δεδομένων όπως ηλιακή παραγωγή και συγκρίσεις προβλέψεων.

Για την ενσωμάτωση του JavaScript αρχείου της Plotly προστέθηκε το ακόλουθο στο “index.html”:

- `<script src="https://cdn.plot.ly/plotly-2.27.0.min.js"></script>`

4.2.11 Υλοποίηση

Για να προβάλλουμε τα σημεία ενδιαφέροντος της εφαρμογής πάνω στον χάρτη, αρχικοποιούμε το Leaflet map με το L.map(), δίνοντας αρχικές συντεταγμένες για την Κύπρο και zoom 10 (βλ. Σχήμα 4.1.1). Ακολούθως, προσθέτουμε πλακίδια χάρτη (tiles) από το OpenStreetMap και ενσωματώνουμε το στοιχείο zoomControl στην κάτω δεξιά γωνία. Με αυτόν τον τρόπο, ο χρήστης μπορεί να αλληλεπιδράσει με τον χάρτη μετακινώντας και μεγεθύνοντάς τον. Στη συνέχεια, χρησιμοποιούμε την L.control.locate() για να προσθέσουμε κουμπί που εμφανίζει τη τρέχουσα θέση του χρήστη στον χάρτη. Η τοποθεσία επισημαίνεται με μπλε κυκλικό marker και σχετικό popup μήνυμα (βλ. Σχήμα 4.1.2).

```
map = L.map('map').setView([latitude, longitude], 10);
map.zoomControl.setPosition('bottomright');
L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '© OpenStreetMap contributors'
}).addTo(map);
const locateControl = L.control.locate({
    position: 'bottomright',
    flyTo: false,
    keepCurrentZoomLevel: true,
    drawCircle: true,
    markerStyle: {
        color: "blue",
        fillColor: "blue",
        fillOpacity: 0.8
    },
    strings: {
        title: "Show My Location",
        popup: "Your location is within {distance} meters",
    }
}).addTo(map);
```

Εικόνα 4.21 Αρχικοποίηση του Leaflet map και Προσθήκη OpenStreetMap tile layer και zoom controls

Προκειμένου να εμπλουτιστεί ο χάρτης με πληροφορίες σχετικές με τις καιρικές συνθήκες, προστέθηκαν τρεις θεματικές στρώσεις (layers) από το OpenWeatherMap: νεφοκάλυψη ,

άνεμος και θερμοκρασία. Κάθε στρώση δημιουργείται με τη χρήση της L.tileLayer() και βασίζεται σε πλακίδια PNG που παρέχει το OpenWeather API (βλ. Εικόνα 4.22).
Οι στρώσεις προστίθενται στο χάρτη ως επιλογές και μπορούν να ενεργοποιηθούν από τον χρήστη για την καλύτερη εκτίμηση της κατάστασης σε κάθε περιοχή.

```
cloudsLayer = L.tileLayer(`https://tile.openweathermap.org/map/clouds_new/{z}/{x}/{y}.png?appid=${apiKey}`, {
    attribution: 'Map data © <a href="https://openweathermap.org">OpenWeatherMap</a>',
    opacity: 1.0
});
const windLayer = L.tileLayer(`https://tile.openweathermap.org/map/wind_new/{z}/{x}/{y}.png?appid=${apiKey}`, {
    attribution: 'Map data © <a href="https://openweathermap.org">OpenWeatherMap</a>',
    opacity: 1.0
});
const tempLayer = L.tileLayer(`https://tile.openweathermap.org/map/temp_new/{z}/{x}/{y}.png?appid=${apiKey}`, {
    attribution: 'Map data © <a href="https://openweathermap.org">OpenWeatherMap</a>',
    opacity: 1.0
});
```

Εικόνα 4.22 Προσθήκη θεματικών στρώσεων για νεφοκάλυψη, άνεμο και θερμοκρασία από OpenWeatherMap

Για την καλύτερη οπτική εμπειρία του χρήστη, προστέθηκε η δυνατότητα εναλλαγής μεταξύ διαφορετικών βασικών χαρτών (base maps), όπως OpenStreetMap, OpenTopoMap, Esri, και CartoDB. Οι χάρτες αυτοί προσφέρονται από διάφορους παρόχους και περιλαμβάνουν τόσο φωτεινές όσο και σκοτεινές θεματικές απεικονίσεις. Η δημιουργία τους γίνεται με χρήση L.tileLayer() για κάθε περίπτωση (βλ. Εικόνα 4.23).

```
const openStreet = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '© OpenStreetMap contributors'
});
const topoMap = L.tileLayer('https://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {
    attribution: '© OpenTopoMap'
});
const esriStreet = L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer/tile/{z}/{y}/{x}', {
    attribution: 'Tiles © Esri — Source: Esri, DeLorme, NAVTEQ',
    maxZoom: 19
});
const esriTopo = L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_Topo_Map/MapServer/tile/{z}/{y}/{x}', {
    attribution: 'Tiles © Esri — Source: USGS, Esri, FAO, NOAA',
    maxZoom: 17
});
const cartoLight = L.tileLayer('https://{s}.basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}.png', {
    attribution: '&copy; <a href="https://carto.com/">CARTO</a>',
    subdomains: 'abcd'
});
const cartoDark = L.tileLayer('https://{s}.basemaps.cartocdn.com/dark_all/{z}/{x}/{y}{r}.png', {
    attribution: '&copy; <a href="https://carto.com/">CARTO</a>',
    subdomains: 'abcd'
});
```

Εικόνα 4.23 Δημιουργία διαφορετικών base map layers.

Οι δύο ξεχωριστοί πίνακες baseLayers και overlayMaps καταχωρούνται σε δύο L.control.layers, που τοποθετούνται στην κάτω δεξιά γωνία του χάρτη. Με αυτόν τον τρόπο, ο χρήστης μπορεί να ενεργοποιήσει ή να απενεργοποιήσει κάθε επίπεδο προβολής ανάλογα με τις ανάγκες του.

```
const overlayMaps = {
    "Cloud Coverage": cloudsLayer,
    "Temperature": tempLayer,
    "Wind": windLayer
};

const baseLayers = {
    "OpenStreetMap": openStreet,
    "OpenTopoMap": topoMap,
    "Esri_WorldStreetMap": esriStreet,
    "Esri_WorldTopoMap": esriTopo,
    "CartoDB.Positron": cartoLight,
    "CartoDB.DarkMatter": cartoDark
};

const overlayControl = L.control.layers(null, overlayMaps, { collapsed: true, position: 'bottomright' }).addTo(map);
const layerControl = L.control.layers(baseLayers, {}, { collapsed: true, position: 'bottomright' }).addTo(map);
```

Εικόνα 4.24 Προσθήκη θεματικών και βασικών στρώσεων στο layer control.

Κατά τη φόρτωση της εφαρμογής, γίνεται κλήση στο endpoint /api/user_vehicle ώστε να ανακτηθεί το αποθηκευμένο όχημα του χρήστη. Αν υπάρχει τουλάχιστον ένα όχημα, επιλέγεται αυτόματα το πρώτο με τη χρήση της συνάρτησης selectVehicle(), διαφορετικά, αποκρύπτεται το πλαίσιο εμφάνισης του επιλεγμένου οχήματος (βλ. Σχήμα 4.11.9). Η διαδικασία συνοδεύεται από fallback χειρισμό σφαλμάτων, εξασφαλίζοντας ότι σε περίπτωση αποτυχίας το UI παραμένει σε σταθερή κατάσταση.

```
function populateUserVehicle() {
    fetch("/api/user_vehicle")
        .then(response => {
            if (!response.ok) {
                throw new Error("Server error: " + response.status);
            }
            return response.json();
        })
        .then(vehicles => {
            console.log("Full vehicle array:", vehicles);
            renderVehiclePanel(vehicles);

            if (vehicles.length > 0) {
                selectVehicle(vehicles[0]); // Auto-select the first vehicle
            } else {
                hideSelectedCarBox();
            }
        })
        .catch(error => {
            console.error("Error fetching saved vehicles:", error);
            renderVehiclePanel([]);
            hideSelectedCarBox();
        });
}
```

Εικόνα 4.25 Ανάκτηση και αυτόματη επιλογή αποθηκευμένου οχήματος κατά την είσοδο στην εφαρμογή.

Κατά την προβολή των οχημάτων του χρήστη, δημιουργείται δυναμικά μια κάρτα για κάθε εγγεγραμμένο όχημα του χρήστη, η οποία περιλαμβάνει την μάρκα, μοντέλο, χωρητικότητα μπαταρίας, κατανάλωση και τους υποστηριζόμενους τύπους πρίζας σε μορφή (βλ. Εικόνα 4.26).

Η κάθε κάρτα εμφανίζεται με custom styling, ενώ προστίθεται event listener που ενεργοποιεί τη λειτουργία επιλογής του οχήματος. Επιπλέον, προστίθεται και κουμπί για πιθανή διαγραφή του οχήματος από το προφίλ του χρήστη.

```
vehicles.forEach((vehicle, index) => {
    const card = document.createElement("div");
    card.className = "selected-car-box vehicle-card mb-2 p-2";
    card.style.background = "linear-gradient(to right, #003538, #727272)";
    card.style.border = "2px solid #005c61d0";
    card.style.color = "white";
    card.style.borderRadius = "15px";
    card.style.cursor = "pointer";
    card.setAttribute("data-vehicle-id", vehicle.vehicle_id);

    let plugIconsHTML = (vehicle.plug_types || []).map(type => `<span class="badge bg-light text-dark mx-1">${type}</span>`).join(' ');

    card.innerHTML = `
        <h6 style="margin:0; font-weight: 800;">${vehicle.make} ${vehicle.model}</h6>
        <div style="font-size: 14px; font-weight: 500;">Battery: ${vehicle.battery_capacity} kWh</div>
        <div style="font-size: 14px; font-weight: 500;">Consumption: ${vehicle.consumption_rate} kWh/100 miles</div>
        <div class="d-flex justify-content-center align-items-center mt-1">${plugIconsHTML}</div>
        <button class="btn btn-danger btn-sm mt-2" style="border-radius: 12px;">Delete</button>
    `;

    card.addEventListener("click", () => {
        selectVehicle(vehicle);
    });
```

Εικόνα 4.26 Δυναμική δημιουργία καρτών για κάθε όχημα με τις σχετικές πληροφορίες και plug types.

Όταν ο χρήστης επιλέξει ένα όχημα, η συνάρτηση selectVehicle() ενεργοποιείται ώστε να ενημερωθεί η διεπαφή με βάση τις ιδιότητες του επιλεγμένου οχήματος (βλ. Σχήμα 4.27). Πιο συγκεκριμένα, ενημερώνεται το φίλτρο plug types καλώντας τη highlightPlugTypes() (βλ. Εικόνα 4.28). Η αντίστοιχη κάρτα επισημαίνεται οπτικά με χρωματιστό περίγραμμα, ενώ αφαιρείται από τις υπόλοιπες. Με αυτό τον τρόπο διασφαλίζεται ότι η επιλογή του οχήματος επηρεάζει τόσο τα οπτικά στοιχεία της διεπαφής όσο και τη λειτουργικότητα των φίλτρων.

```
function selectVehicle(vehicle) {
    selectedVehicle = vehicle;

    highlightPlugTypes(vehicle.plug_types || []);

    // Update selected car box in filter tab
    const box = document.getElementById("selectedCarBox");
    const nameElement = box.querySelector(".car-name");
    nameElement.textContent = `${vehicle.make} ${vehicle.model}`;
    box.style.display = "flex";

    // Highlight the selected card visually
    document.querySelectorAll('.vehicle-card').forEach(card => {
        if (parseInt(card.getAttribute('data-vehicle-id')) === vehicle.vehicle_id) {
            card.style.border = '2px solid #005c61d0';
        } else {
            card.style.border = 'none';
        }
    });
}
```

Εικόνα 4.27 Ενημέρωση διεπαφής και φίλτρων κατά την επιλογή ηλεκτρικού οχήματος.

```
function highlightPlugTypes(types) {
    if (typeof types === "string") {
        types = types.split(",").map(s => s.trim());
    }

    selectedPlugTypes = new Set(types);
    document.querySelectorAll('.plug-icon-box').forEach(elem => {
        const type = elem.getAttribute('data-plug-type');
        if (selectedPlugTypes.has(type)) {
            elem.classList.add('selected-plug');
        } else {
            elem.classList.remove('selected-plug');
        }
    });

    updatePlugTypeFiltering();
    const showAllToggle = document.getElementById("loadAllChargers");
    if (showAllToggle) toggleAllChargers(showAllToggle.checked);
}
```

Εικόνα 4.28 Εφαρμογή και οπτική ανάδειξη των plug types που υποστηρίζει το επιλεγμένο όχημα, με βάση τα δεδομένα που επιστρέφει η βάση.

Μετά την επιλογή plug types από τον χρήστη ή από κάποιο όχημα, η συνάρτηση updatePlugTypeFiltering() εφαρμόζει το αντίστοιχο φίλτρο στους φορτιστές του χάρτη. Για κάθε marker, ελέγχεται αν οι διαθέσιμοι τύποι του φορτιστή περιλαμβάνουν κάποιον από τους

επιλεγμένους. Αν υπάρχει συμβατότητα ή αν δεν έχει επιλεγεί κανένα φίλτρο, ο marker εμφανίζεται στον χάρτη. Διαφορετικά, αφαιρείται προσωρινά (βλ. Σχήμα 4.29). Με αυτόν τον τρόπο διασφαλίζεται η δυναμική και προσαρμόσιμη οπτικοποίηση των φορτιστών, ανάλογα με τις ανάγκες του χρήστη.

```
function updatePlugTypeFiltering() {
    allChargerMarkers.forEach(marker => {
        const charger = marker.chargerData;
        const chargerTypes = Array.isArray(charger.types) ? charger.types : [charger.types];

        console.log("Selected plug types:", selectedPlugTypes);
        console.log("This charger's types:", marker.chargerData.types);

        const matches = Array.from(selectedPlugTypes).some(type => chargerTypes.includes(type));
        if (selectedPlugTypes.size === 0 || matches) {
            map.addLayer(marker);
        } else {
            map.removeLayer(marker);
        }
    });
}
```

Εικόνα 4.29 Εφαρμογή δυναμικού φιλτραρίσματος στους markers βάσει επιλεγμένων plug types.

Για την αξιολόγηση των φορτιστών σύμφωνα με τα οικολογικά κριτήρια του χρήστη, συλλέγουμε δυναμικά τις τιμές των sliders που αντιστοιχούν στα βάρη των τριών βασικών προτιμήσεων: derouting cost (ec1), availability (ec2) και sustainable charging level (ec3). Οι τιμές κανονικοποιούνται διαιρούμενες με το 100 (βλ. Εικόνα 4.30). Επιπλέον, καταχωρείται η ακτίνα αναζήτησης (radius) καθώς και οι επιλεγμένοι τύποι πρίζας (selectedPlugTypes). Όλες αυτές οι πληροφορίες αποστέλλονται στο backend για να υπολογιστεί και να επιστραφεί η κατάλληλη κατάταξη φορτιστών.

```
document.getElementById("ec1").value / 100,
document.getElementById("ec2").value / 100,
document.getElementById("ec3").value / 100,
parseInt(document.getElementById("radius").value),
Array.from(selectedPlugTypes)
```

Εικόνα 4.30 Ανάγνωση οικολογικών προτιμήσεων και παραμέτρων αναζήτησης από τα στοιχεία του frontend.

Η λειτουργία πρόβλεψης βασίζεται σε έναν χρονικό slider που επιτρέπει στον χρήστη να επιλέξει διαφορετικές χρονικές στιγμές από το forecast API του OpenWeather. Κάθε φορά που μετακινείται το slider, ενεργοποιείται event listener τύπου input που ενημερώνει το label με

την αντίστοιχη ημερομηνία/ώρα και καλεί τη συνάρτηση forecastAndColorAllChargers() (βλ. Εικόνα 4.31) για να επικαιροποιήσει τα χρώματα των markers στον χάρτη (βλ. Εικόνα 4.32). Κατά την αρχική φόρτωση, το label ρυθμίζεται ώστε να εμφανίζει την ώρα του πρώτου διαθέσιμου forecast.

```
slider.addEventListener("input", function () {
    const index = parseInt(this.value);
    label.textContent = `Forecast for: ${forecastTimeList[index].dt_txt}`;
    if (!forecastInitialized) forecastInitialized = true;
    forecastAndColorAllChargers(index);
});

label.textContent = `Forecast for: ${forecastTimeList[0].dt_txt}`;
```

Εικόνα 4.31 Ενημέρωση του label και ενεργοποίηση πρόβλεψης φορτιστών μέσω forecast slider.

Κατά την αρχική εκκίνηση της εφαρμογής καλείται η ασύγχρονη συνάρτηση loadChargers(), η οποία φορτώνει όλους τους διαθέσιμους φορτιστές από το /api/chargers και τα δεδομένα πρόβλεψης ηλιακής απόδοσης από το /api/daily_forecast. Τα forecast δεδομένα αποθηκεύονται τοπικά σε chargerForecastCache ανά φορτιστή, για να μπορούν να χρησιμοποιηθούν άμεσα χωρίς νέα κλήση API (βλ. Σχήμα 4.11.17).
Η επιτυχημένη φόρτωση ενεργοποιεί την ενημέρωση του dropdown πρόβλεψης μέσω της populateForecastDropdown().

```
async function loadChargers() {
    try {
        const chargerResponse = await fetch('/api/chargers');
        allChargers = await chargerResponse.json();

        const forecastResponse = await fetch('/api/daily_forecast');
        const forecastJson = await forecastResponse.json();

        // Build forecast cache from local file
        forecastJson.data.forEach(entry => {
            chargerForecastCache[entry.charger_id] = entry.forecast.list;
        });

        console.log("Forecasts preloaded from local file:", chargerForecastCache);
        populateForecastDropdown();
    } catch (error) {
        console.error('Error loading chargers or forecasts:', error);
    }
}
```

Εικόνα 4.32 Φόρτωση φορτιστών και πρόβλεψης καιρού από το backend και κατασκευή τοπικής cache πρόβλεψης

Κατά την αλλαγή τιμής στον forecast slider, καλείται η ασύγχρονη συνάρτηση forecastAndColorAllChargers(), η οποία υπολογίζει την προβλεπόμενη ηλιακή απόδοση για κάθε φορτιστή την επιλεγμένη ώρα. Αρχικά αφαιρούνται όλοι οι markers από τον χάρτη, και δημιουργείται το body που περιλαμβάνει τη χρονική στιγμή και το ποσοστό νεφοκάλυψης ανά φορτιστή (βλ. Εικόνα 4.33).

Στη συνέχεια, γίνεται POST αίτημα στο endpoint /api/forecast_solar_scores, το οποίο επιστρέφει τις εκτιμώμενες τιμές ηλιακής ενέργειας για κάθε φορτιστή. Οι τιμές αυτές αξιοποιούνται για τον επανασχεδιασμό των markers, χρωματισμένων ανάλογα με την πρόβλεψη (Εικόνα 4.34).

```
async function forecastAndColorAllChargers(forecastIndex = 0) {

    allChargerMarkers.forEach(m => map.removeLayer(m));
    allChargerMarkers = [];

    const body = {
        forecast_time: null,
        clouds: {}
    };

    for (const charger of allChargers) {
        const forecastList = chargerForecastCache[charger.charger_id];
        if (!forecastList || !forecastList[forecastIndex]) continue;
        const entry = forecastList[forecastIndex];
        body.clouds[charger.charger_id] = entry.clouds.all;
        if (!body.forecast_time) body.forecast_time = entry.dt_txt.replace("T", " ").split(":").slice(0, 2).join(":") + ":00";
    }

    const forecastScores = await fetch("/api/forecast_solar_scores", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(body)
    }).then(r => r.json());

    let usedForecastTime = body.forecast_time;
```

Εικόνα 4.33 Αποστολή δεδομένων πρόβλεψης προς το backend και λήψη εκτιμώμενων τιμών ηλιακής απόδοσης για όλους τους φορτιστές.

```
for (const charger of allChargers) {
    if (!charger.latitude || !charger.longitude) continue;

    const solar_score = forecastScores[charger.charger_id] || 0;
    let iconUrl = "/static/darkRedCharger.png";
    if (solar_score >= 75) iconUrl = "/static/greenCharge.png";
    else if (solar_score >= 40) iconUrl = "/static/blackCharge.png";

    const chargerIcon = L.icon({
        iconUrl: iconUrl,
        iconSize: [20, 30],
        iconAnchor: [10, 30],
        popupAnchor: [0, -30],
        className: 'forecast-charger-marker'
    });
```

Εικόνα 4.34 Επιλογή εικονιδίου φορτιστή με βάση το ποσοστό προβλεπόμενης ηλιακής απόδοσης και δημιουργία Leaflet icon.

Η συνάρτηση useCurrentLocation() χρησιμοποιείται για την ανάκτηση της τρέχουσας γεωγραφικής θέσης του χρήστη μέσω του browser. Αν υποστηρίζεται η δυνατότητα γεωεντοπισμού, η θέση του χρήστη προσδιορίζεται και ο χάρτης μετακινείται (map.flyTo) σε αυτή με zoom 14. Ταυτόχρονα, το πεδίο "origin" ενημερώνεται με την ένδειξη "Current Location" (βλ. Σχήμα 4.35). Σε περίπτωση αποτυχίας, εμφανίζεται σχετικό μήνυμα σφάλματος, ενώ γίνεται και απόκρυψη του drop-down με την επιλογή προέλευσης.

```
function useCurrentLocation() {
    if (!navigator.geolocation) {
        alert("Geolocation is not supported by your browser.");
        return;
    }

    if (!map) {
        console.warn("Map is not initialized yet. Try again in a moment.");
        return;
    }

    console.log("Trying to get geolocation...");
    navigator.geolocation.getCurrentPosition(
        (position) => {
            const lat = position.coords.latitude;
            const lng = position.coords.longitude;

            userLocation = L.latLng(lat, lng);
            document.getElementById("origin").value = "Current Location";

            const dropdown = bootstrap.Dropdown.getInstance(document.getElementById("origin"));
            if (dropdown) dropdown.hide();

            map.flyTo(userLocation, 14);
        },
        () => alert("Unable to retrieve your location.")
    );
}
```

Εικόνα 4.35 Χρήση γεωεντοπισμού για μετακίνηση του χάρτη στην τρέχουσα τοποθεσία του χρήστη

Η συνάρτηση getCoordinates() αξιοποιεί την υπηρεσία Nominatim του OpenStreetMap για να μετατρέψει μια ονομασία τοποθεσίας (π.χ. πόλη ή διεύθυνση) σε γεωγραφικές συντεταγμένες (γεωκωδικοποίηση). Εκτελείται ασύγχρονα και αν εντοπιστεί έγκυρη απάντηση, επιστρέφεται αντικείμενο τύπου L.latLng με το latitude και longitude. Σε διαφορετική περίπτωση, εμφανίζεται προειδοποιητικό μήνυμα ότι η τοποθεσία δεν βρέθηκε (βλ. Εικόνα 4.36).

```
// Function to get geocoded coordinates from location name
async function getCoordinates(location) {
    let response = await fetch(`https://nominatim.openstreetmap.org/search?format=json&q=${location}`);
    let data = await response.json();
    if (data.length > 0) {
        return L.latLng(data[0].lat, data[0].lon);
    } else {
        alert("Location not found: " + location);
        return null;
    }
}
```

Εικόνα 4.36 Γεωκωδικοποίηση τοποθεσίας μέσω OpenStreetMap Nominatim API

Η συνάρτηση searchRoute() χρησιμοποιείται για να δημιουργηθεί διαδρομή από το σημείο εκκίνησης προς τον επιλεγμένο φορτιστή ή προορισμό. Χρησιμοποιεί τη βιβλιοθήκη leaflet-routing-machine με υποστήριξη OSRM (Open Source Routing Machine) και εμφανίζει γραφικά τη διαδρομή πάνω στον χάρτη με μπλε γραμμή (βλ. Εικόνα 4.37). Αρχικά καθαρίζονται προηγούμενες διαδρομές και markers, και έπειτα δημιουργείται νέο αντικείμενο L.Routing.control. Όταν ολοκληρωθεί η αναζήτηση, εμφανίζεται το αποτέλεσμα με λεπτομέρειες για τον χρόνο της διαδρομής σε λεπτά.

```
// Function to handle routing
async function searchRoute() {
    let originCoords = await getOriginCoordinates();
    let destinationCoords = await getCoordinates(document.getElementById("destination").value);
    if (!originCoords || !destinationCoords) return;
    // Remove previous route if it exists
    if (routingControl !== null) {
        map.removeControl(routingControl);
        routingControl = null;
    }
    // Only remove previous ranked charger markers (not all custom-marker class)
    nearbyChargerMarkers.forEach(marker => map.removeLayer(marker));
    nearbyChargerMarkers = [];
    // Add new route
    routingControl = L.Routing.control({
        waypoints: [originCoords, destinationCoords],
        routeWhileDragging: true,
        lineOptions: {
            styles: [{ color: "blue", opacity: 0.7, weight: 6 }]
        },
        router: L.routing.osrmv1({
        serviceUrl: 'https://router.project-osrm.org/route/v1'
        }),
        container: document.getElementById("routeInstructions")
    }).on('routesfound', function(e) {
        let route = e.routes[0];
        let travelTimeSeconds = route.summary.totalTime;
        let travelTimeMinutes = Math.round(travelTimeSeconds / 60);
        console.log("Route Object:", route);
        if (!route.summary) {
            console.error("No route summary found! Check OSRM response.");
            return;
        }
```

Εικόνα 4.37 Δημιουργία και εμφάνιση διαδρομής μέσω OSRM μεταξύ εκκίνησης και επιλεγμένου φορτιστή - προορισμού.

Μετά τη σχεδίαση διαδρομής, τοποθετείται ένας διαδραστικός marker σε μορφή ταξί (taxi-marker) στο σημείο εκκίνησης. Ο χρήστης μπορεί να σύρει το marker σε διαφορετική θέση στον χάρτη, ενεργοποιώντας αυτόματα τη συνάρτηση fetchNearbyChargers() με τα νέα συντεταγμένα. Η συνάρτηση καλείται με τις οικολογικές παραμέτρους (ec1, ec2, ec3), την ακτίνα αναζήτησης και τα επιλεγμένα plug types (βλ. Εικόνα 4.38).

```
taxiMarker = L.marker([originCoords.lat, originCoords.lng], {
    icon: L.icon({
        iconUrl: "/static/taxi.png",
        iconSize: [30, 40],
        iconAnchor: [15, 40],
        popupAnchor: [0, -40],
        className: 'taxi-marker'
    }),
    draggable: true
}).addTo(map);
taxiMarker.bindPopup("Drag the cab to find nearby chargers.").openPopup();
taxiMarker.on('dragend', function (event) {
    const marker = event.target;
    const position = marker.getLatLng();
    fetchNearbyChargers(
        position.lat,
        position.lng,
        document.getElementById("ec1").value / 100,
        document.getElementById("ec2").value / 100,
        document.getElementById("ec3").value / 100,
        parseInt(document.getElementById("radius").value),
        Array.from(selectedPlugTypes)
    );
});
```

Εικόνα 4.38 Τοποθέτηση draggable marker τύπου "ταξί" και ενεργοποίηση δυναμικής αναζήτησης φορτιστών με βάση τις οικολογικές προτιμήσεις

Πιο κάτω φαίνεται η συνάρτηση sendNodes() η οποία χρησιμοποιείται για την αποστολή της λίστας κόμβων μιας διαδρομής στο backend μέσω HTTP POST αιτήματος στη διαδρομή /nodes. Τα δεδομένα μετατρέπονται σε JSON μορφή και αποστέλλονται.

```
function sendNodes(nodesList) {
    fetch('/nodes', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ nodesList: nodesList }),
    })
    .then(response => response.json())
    .then(data => {
        console.log('Server response:', data);
    })
    .catch(error => {
        console.error('Error sending nodes:', error);
    });
}

document.addEventListener("DOMContentLoaded", function() {
    fetchSolarPowerData();
});
```

Εικόνα 4.39 Αποστολή λίστας κόμβων στο backend .

Η συνάρτηση fetchNearbyChargers() είναι υπεύθυνη για την αποστολή αιτήματος προς το backend με σκοπό την ανάκτηση των πλησιέστερων φορτιστών, βασισμένων στις προτιμήσεις του χρήστη. Το αίτημα αποστέλλεται μέσω HTTP POST προς το endpoint /nearby_chargers, με παραμέτρους όπως: συντεταγμένες, βάρη προτιμήσεων (derouting_cost, sustainable_charging_level), ακτίνα αναζήτησης και επιλεγμένοι τύποι πρίζας (βλ. Εικόνα 4.40). Αφού ληφθούν οι απαντήσεις από τον server, οι προηγούμενοι markers φορτιστών αφαιρούνται και οι νέοι προστίθενται δυναμικά στον χάρτη με τη χρήση της addMarkers().

```
function fetchNearbyChargers(lat, lng, derouting_cost, charger_availability, sustainable_charging_level, radius) {
    fetch('/nearby_chargers', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            lat: lat,
            lng: lng,
            derouting_cost: derouting_cost,
            charger_availability: charger_availability,
            sustainable_charging_level: sustainable_charging_level,
            radius: radius,
            plug_types: Array.from(selectedPlugTypes)
        }),
    })
    .then(response => response.json())
    .then(data => {
        console.log('Nearby Chargers:', data);
        // Only remove previous ranked charger markers (not all custom-marker class)
        nearbyChargerMarkers.forEach(marker => map.removeLayer(marker));
        nearbyChargerMarkers = [];
        // Add new nearby charger markers
        addMarkers(data.nearby_chargers);
    })
    .catch(error => console.error('Error fetching nearby chargers:', error));
}
```

Εικόνα 4.40 Συνάρτηση που αποστέλλει δεδομένα στο endpoint "/nearby_chargers"

Κατά το πάτημα σε κάποιο marker φορτιστή, ενεργοποιείται ένα side panel τύπου κάρτας (charger-card) το οποίο εμφανίζει αναλυτικές πληροφορίες για τον επιλεγμένο φορτιστή.

```
marker.on('click', function () {
    const panel = document.getElementById('charger-card');
    panel.style.display = 'block';
    setTimeout(() => {
        panel.classList.add('show');
    }, 10);

    document.getElementById('charger-title').textContent = charger.LocationName;
    document.getElementById('charger-address').textContent = charger.Address;
    document.getElementById('charger-status').textContent = charger.enabled ? "Enabled" : "Disabled";
    document.getElementById('charger-power').textContent = charger.power || "N/A";
    document.getElementById('charger-solar').textContent = charger.solar_power_estimate.toFixed(1);
    document.getElementById('charger-grid').textContent = (charger.grid_power_percentage || 0).toFixed(1);
    document.querySelector('.power strong').textContent = charger.power + ' kW';
```

Εικόνα 4.41 Ενεργοποίηση κάρτας πληροφοριών φορτιστή

Η συνάρτηση getUserID() χρησιμοποιείται για την αναγνώριση του συνδεδεμένου χρήστη μέσω κλήσης στο endpoint /api/get_user_id, αλλά και την εύρεση του ρόλου του (regular user ή operator). Αν ο χρήστης είναι συνδεδεμένος, αποθηκεύεται το userID και καλούνται οι συναρτήσεις renderVehiclePanel() και populateUserVehicle() για την προβολή των αντίστοιχων στοιχείων. Ακόμα και αν δεν υπάρχει χρήστης ή υπάρξει σφάλμα, η διεπαφή ανανεώνεται ώστε να επανέλθει σε default κατάσταση.

```
function getUserID() {
    fetch('/api/get_user_id')
        .then(response => response.json())
        .then(data => {
            if (data.userID) {
                userID = data.userID;
                renderVehiclePanel();
                populateUserVehicle();
                if (data.role === "Operator") {
                    const viewGridItem = document.getElementById("viewGridNavItem");
                    if (viewGridItem) viewGridItem.style.display = "block";
                }
            } else {
                userID = null;
                renderVehiclePanel();  // still refresh UI
            }
        })
        .catch(error => {
            console.error('Error fetching user ID:', error);
            userID = null;
            renderVehiclePanel();
        });
}
```

Εικόνα 4.42 Ανάκτηση ταυτότητας χρήστη και προσαρμογή της διεπαφής ανάλογα με το ρόλο

Μετά την επιβεβαίωση του ρόλου από τη συνάρτηση getUserID(), οι χρήστες με ρόλο Operator διαθέτουν επιπρόσθετες δυνατότητες διαχείρισης μέσω της διεπαφής. Ένας operator μπορεί να δει τα MicroGrids που έχει δημιουργήσει μέσω ειδικού navigation tab, το οποίο ενεργοποιείται μόνο για χρήστες με ρόλο "Operator".

Η συνάρτηση loadMyMicrogrids() καλείται όταν ο operator χρήστης ανοίγει την καρτέλα διαχείρισης των MicroGrids του. Μέσω κλήσης στο endpoint /api/my_microgrids, ανακτώνται από τη βάση δεδομένων όλα τα MicroGrids που έχει δημιουργήσει ο συγκεκριμένος χρήστης. Για κάθε MicroGrid δημιουργείται δυναμικά μία κάρτα προβολής, με πληροφορίες όπως όνομα, πόλη, χώρα, ισχύς εγκατάστασης και απόδοση (βλ Εικόνα 4.43). Αν δεν έχουν δημιουργηθεί ακόμα MicroGrids, εμφανίζεται κατάλληλο ενημερωτικό μήνυμα.

```
function loadMyMicrogrids() {
    fetch("/api/my_microgrids")
        .then(res => res.json())
        .then(microgrids => {
            const container = document.getElementById("operatorMicrogridsContainer");
            container.innerHTML = "";

            if (!microgrids || microgrids.length === 0) {
                container.innerHTML = "<p>No MicroGrids created yet.</p>";
                return;
            }

            microgrids.forEach(grid => {
                const div = document.createElement("div");
                div.classList.add("card", "mb-2", "p-3", "card_microgrid");
                div.innerHTML = `
                    <h5>${grid.name}</h5>
                    <p><strong>City:</strong> ${grid.city} | <strong>Country:</strong> ${grid.country}</p>
                    <p><strong>Capacity:</strong> ${grid.capacity_kwp} kWp | <strong>Efficiency:</strong> ${(grid.efficiency * 100).toFixed(0)}%</p>
                    <button class="btn btn-sm btn-outline-info" onclick="showMicrogridModal(${grid.group_id})" >View Details</button>
                `;
                container.appendChild(div);
            });
        });
}
```

Εικόνα 4.43 Δυναμική εμφάνιση MicroGrids του Operator

Κατά την επιλογή της λειτουργίας “View Details” σε μια κάρτα MicroGrid, καλείται η συνάρτηση showMicrogridModal() η οποία πραγματοποιεί κλήση στο /api/microgrid_details/{groupId} για να ανακτήσει αναλυτικά δεδομένα του συγκεκριμένου MicroGrid. Τα δεδομένα αποθηκεύονται προσωρινά και προβάλλονται σε μορφή πίνακα μέσα σε modal παράθυρο με περισσότερες πληροφορίες (βλ. Εικόνα 4.44). Σε αυτό το παράθυρο εμφανίζονται και οι φορτιστές που ανήκουν στο μικροδίκτυο, από το πεδίο data.chargers της απάντησης, καθώς και κουμπιά επεξεργασίας τους.

```
function showMicrogridModal(groupId) {
fetch(`/api/microgrid_details/${groupId}`)
    .then(res => res.json())
    .then(data => {
        console.log("Microgrid API Response:", data);
    cachedMicrogridData = data;
    const m = data.microgrid;
    const chargers = data.chargers;
    let html = `
        <h5 class="text-info mb-3"><i class="bi bi-lightning"></i> Microgrid Summary</h5>
        <table class="table rounded-table text-white">
            <tbody>
                <tr><td><strong>City:</strong></td><td>${m.city}</td></tr>
                <tr><td><strong>Country:</strong></td><td>${m.country}</td></tr>
                <tr><td><strong>Latitude:</strong></td><td>${m.latitude.toFixed(5)}</td></tr>
                <tr><td><strong>Longitude:</strong></td><td>${m.longitude.toFixed(5)}</td></tr>
                <tr><td><strong>Installed Capacity:</strong></td><td>${m.capacity_kwp} kWp</td></tr>
                <tr><td><strong>Efficiency:</strong></td><td>${m.efficiency}%</td></tr>
                <tr><td><strong>Inverter Limit:</strong></td><td>${m.inverter_limit_kw} kW</td></tr>
                <tr><td><strong>Degradation Rate:</strong></td><td>${m.degradation_rate}%</td></tr>
            </tbody>
        </table>
```

Εικόνα 4.44 Απόσπασμα προβολής λεπτομερειών ενός MicroGrid σε modal

Για την οπτικοποίηση της ενεργειακής παραγωγής ενός MicroGrid, χρησιμοποιείται η συνάρτηση renderProductionCharts(), η οποία δημιουργεί ραβδόγραμμα (bar chart) με τα εκτιμώμενα kWh παραγωγής ανά μήνα. Οι τιμές προέρχονται από το αντικείμενο production.monthly και προβάλλονται μέσω της βιβλιοθήκης Plotly.js (βλ. Σχήμα 4.45).

```
function renderProductionCharts(production, group_id) {
    // Monthly Chart
    const monthLabels = Object.keys(production.monthly);
    const monthValues = Object.values(production.monthly);

    Plotly.newPlot('monthlyChart', [{
        x: monthLabels,
        y: monthValues,
        type: 'bar',
        marker: { color: '#5bb0b4' }
    }], {
        title: 'Monthly Production (kWh)',
        paper_bgcolor: '#ffffff',
        plot_bgcolor: '#ffffff',
        font: { color: '#000' },
        height: 250,
        margin: {
            t: 50,
            b: 20,
            l: 60,
            r: 60
        }
    });
```

Εικόνα 4.45 Ραβδόγραμμα μηνιαίας παραγωγής ενός MicroGrid με χρήση της Plotly.js

Για την ανάλυση της ημερήσιας συμπεριφοράς παραγωγής ενός MicroGrid, πραγματοποιείται κλήση στο endpoint /api/microgrid_hourly_today/{group_id} και τα δεδομένα αποδίδονται σε διάγραμμα τύπου γραμμής με markers. Οι τιμές βασίζονται στην πραγματική παραγωγή ανά ώρα (actual)(βλ. Σχήμα 4.11.30).

```
// Hourly Chart
fetch(`/api/microgrid_hourly_today/${group_id}`)
    .then(res => res.json())
    .then(data => {
        const actualHours = Object.keys(data.actual)
            .map(h => parseInt(h))
            .sort((a, b) => a - b);
        const actualValues = actualHours.map(h => data.actual[h]);
        const cloudValues = actualHours.map(h => data.clouds[h] || "?");
        const theoreticalValues = actualHours.map(h => data.theoretical[h] ?? 0);
        const xLabels = actualHours.map(h => `${h}:00`);

        Plotly.newPlot('hourlyChart', [
        {
            x: xLabels,
            y: actualValues,
            name: "Actual Generation",
            type: 'scatter',
            mode: 'lines+markers',
            text: cloudValues.map(c => `Clouds: ${c}`),
            hovertemplate: 'Hour: %{x}<br>Actual: %{y:.2f} kWh<br>%{text}<extra></extra>',
            line: { color: '#007b81', width: 3 }
        },
        {
            x: xLabels,
            y: theoreticalValues,
            name: "Max Possible (No Clouds)",
            type: 'scatter',
            mode: 'lines+markers',
            hoverlabel: {
                font: { color: 'white' }
            },
            line: { color: 'grey', dash: 'dash' }
        }
```

Εικόνα 4.46 Γραφική απεικόνιση ημερήσιας παραγωγής MicroGrid και σύνδεση με ποσοστά νεφοκάλυψης ανά ώρα

Κεφάλαιο 5 Γραφικό περιβάλλον χρήστη

Σε αυτό το κεφάλαιο παρουσιάζεται το γραφικό περιβάλλον χρήστη του συστήματος.

5.1 Navbar

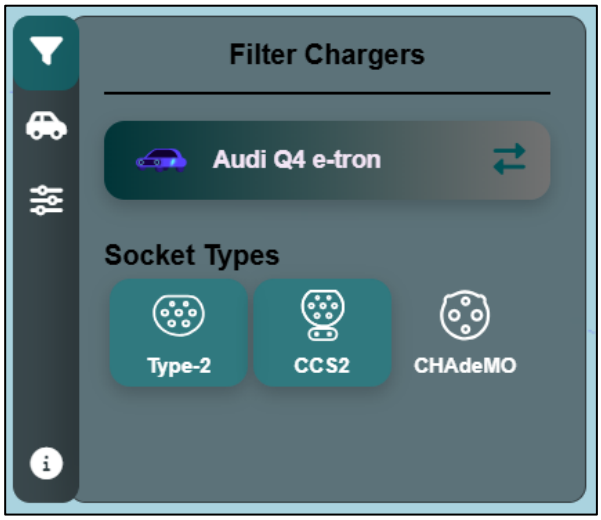
Στην εικόνα 5.1 φαίνεται η γραμμή πλοήγησης (navbar) επιτρέπει εναλλαγή ανάμεσα στις κύριες λειτουργίες: χάρτης, πρόβλεψη και διαχείριση μικροδικτύου. Περιλαμβάνει κουμπί για login ή logout ανάλογα αν ο χρήστης είναι συνδεδεμένος.

Εικόνα 5. 1 Navigation bar

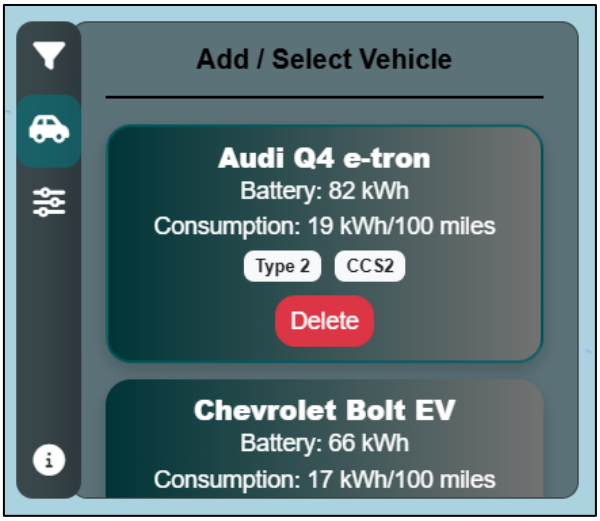
5.2 Sidebar

Στην Εικόνα 5.2 φαίνεται το πρώτο tab του sidebar, το οποίο εμφανίζει φίλτρα τύπου πρίζας φορτιστών με χρήση εικονιδίων, αλλά και το επιλεγμένο αυτοκίνητο του χρήστη. Το δεύτερο tab της Εικόνα 5.3 επιτρέπει την επιλογή ή την εισαγωγή νέου ηλεκτρικού οχήματος και φορτώνει αυτόματα πληροφορίες όπως μπαταρία και κατανάλωση. Οι προτιμήσεις του χρήστη ρυθμίζονται με sliders όπως φαίνεται στην Εικόνα 5.4, που ελέγχουν την ευαισθησία σε

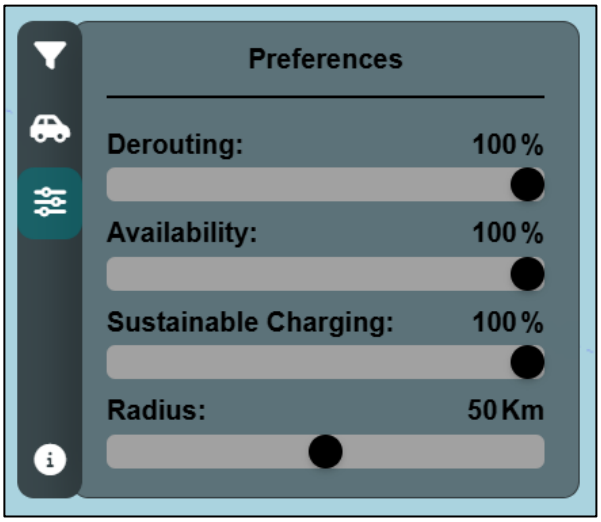
απόσταση, διαθεσιμότητα και βιωσιμότητα, καθώς και ένα range slider για την ακτίνα μέσα στην οποία ο αλγόριθμος θα προτείνει σταθμούς φόρτισης. Τέλος, το tab “Info” της Εικόνας 5.4 περιλαμβάνει υπόμνημα με σημασιολογία των χρωμάτων για solar efficiency φορτιστών και microgrids.



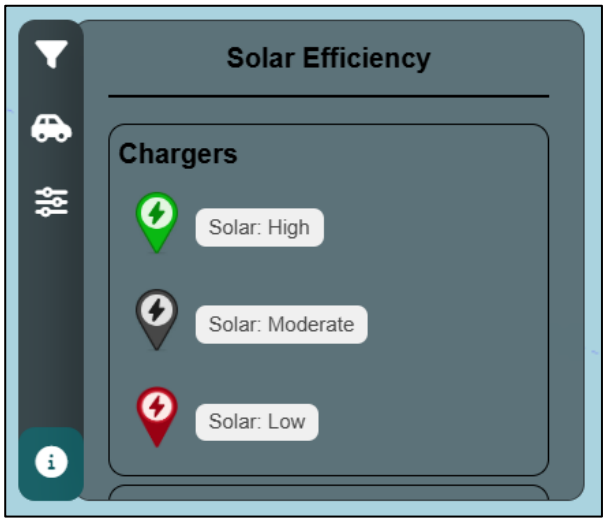
Εικόνα 5. 2 Filters tab



Εικόνα 5. 3 Vehicle tab



Εικόνα 5. 2 Estimated Components tab



Εικόνα 5. 5 Info tab

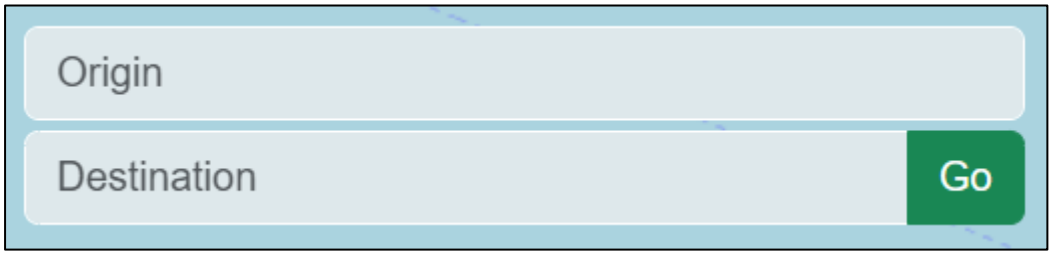
5.3 Εμφάνιση όλων των φορτιστών ή μικροδικτύων

Στην Εικόνα 5.6 φαίνονται οι διακόπτες Show Chargers και Show Microgrids που επιτρέπουν στον χρήστη να εμφανίσει ή να αποκρύψει δυναμικά τα markers των φορτιστών και τις γεωμετρικές περιοχές των MicroGrids στον χάρτη, αντίστοιχα.

Εικόνα 5.6 Show chargers or microgrids switch

5.4 Εισαγωγή διαδρομής

Στην πάνω αριστερή περιοχή του χάρτη εμφανίζονται τα πεδία της Εικόνας 5.7 στα οποία ο χρήστης μπορεί να εισάγει την αρχή και τον προορισμό της διαδρομής που επιθυμεί και να ξεκινήσει τη δρομολόγηση πατώντας το κουμπί Go, το οποίο ενεργοποιεί τον υπολογισμό διαδρομής και χρόνου άφιξης.



Εικόνα 5. 7 Route inputs

5.5 Επιλογή χάρτη και φίλτρων χάρτη

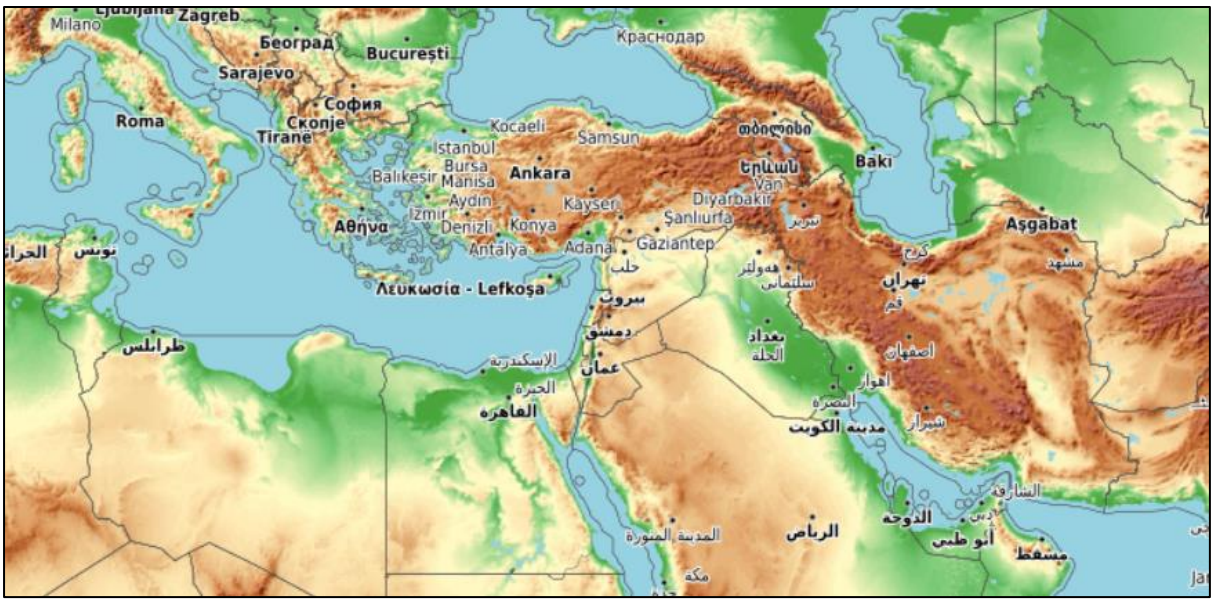
Στην κάτω δεξιά πλευρά του χάρτη εμφανίζεται ένα κατακόρυφο πάνελ, όπου ο χρήστης μπορεί να επιλέξει τον χάρτη που επιθυμεί από τους διαθέσιμους γεωγραφικούς και τοπογραφικούς χάρτες, αλλά και τα διαθέσιμα φίλτρα χάρτη.
Οι διαθέσιμοι χάρτες είναι οι ακόλουθοι:

- OpenStreetMap (βλ. Εικόνα 5.8)
- OpenTopoMap (βλ. Εικόνα 5.9)
- Esri_WorldStreetMap
- Esri_WorldTopotMap
- CartoDB.Positron
- CartoDB.DarkMatter

Τα διαθέσιμα φίλτρα χάρτη είναι τα ακόλουθα:

- Cloud coverage
- Temperature (βλ. Εικόνα 5.10)
- Wind

Εικόνα 5. 8 Γεωγραφικός χάρτης από OpenStreetMap

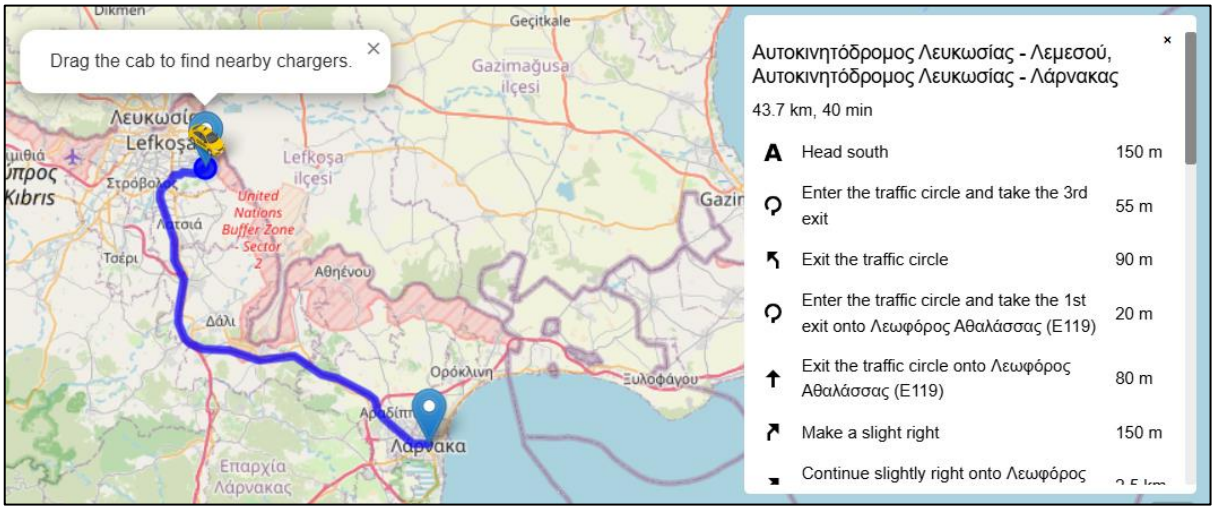


Εικόνα 5.9 Τοπογραφικός χάρτης από OpenTopoMap

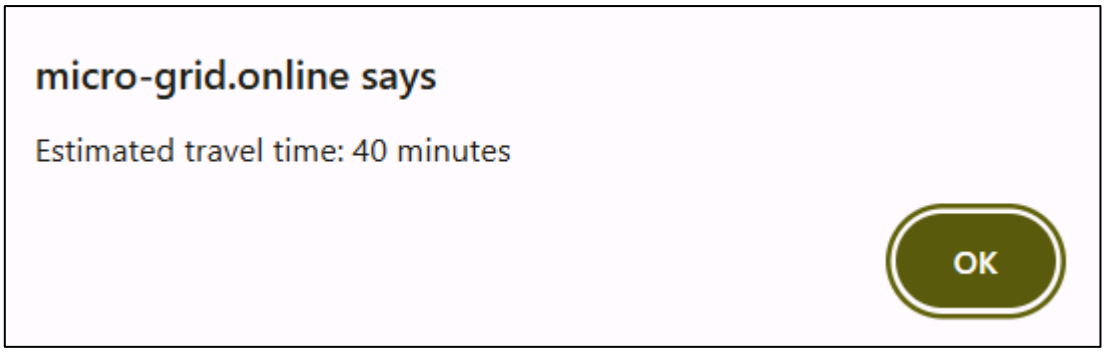
Εικόνα 5. 10 Temperature layer

5.6 Εμφάνιση διαδρομής

Όταν ο χρήστης εισάγει προέλευση και προορισμό, εμφανίζεται στον χάρτη μπλε γραμμή διαδρομής με αρχικό και τελικό σημείο, καθώς και εικονίδιο ταξί που μπορεί να μετακινηθεί (βλ. Εικόνα 5.11). Στα δεξιά προβάλλεται αναλυτική λίστα οδηγιών κατεύθυνσης, ενώ παράλληλα εμφανίζεται μήνυμα με τον εκτιμώμενο χρόνο ταξιδιού, όπως φαίνεται στην Εικόνα 5.12 .

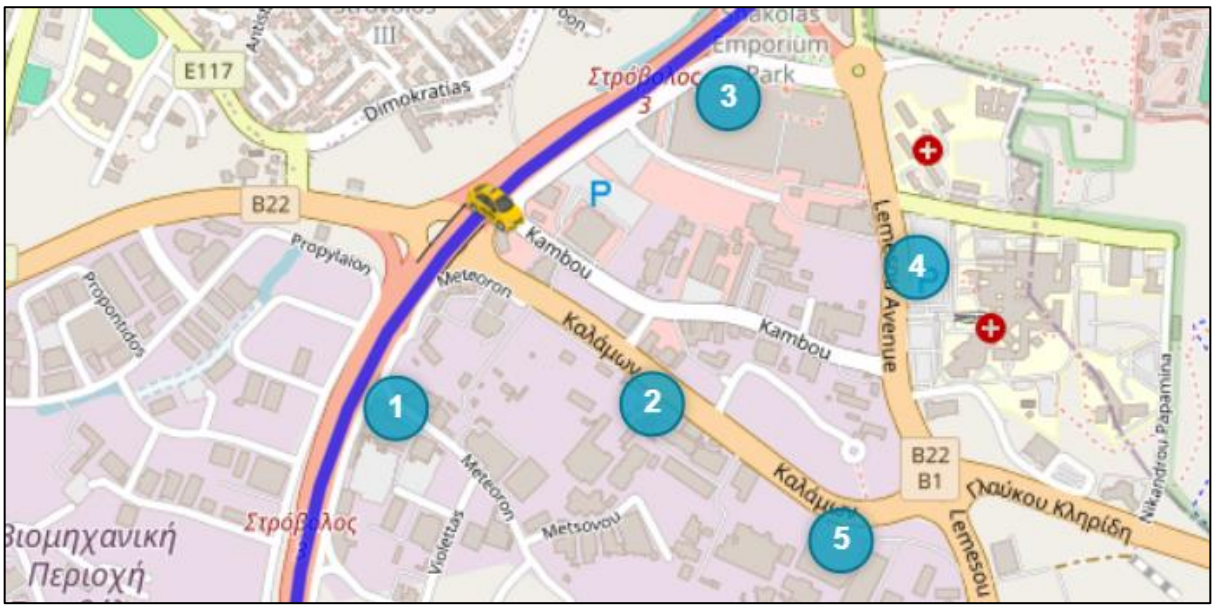


Εικόνα 5.11 Εμφάνιση επιλεγμένης διαδρομής



Εικόνα 5. 12 Μήνυμα εκτιμώμενου χρόνου ταξιδιού

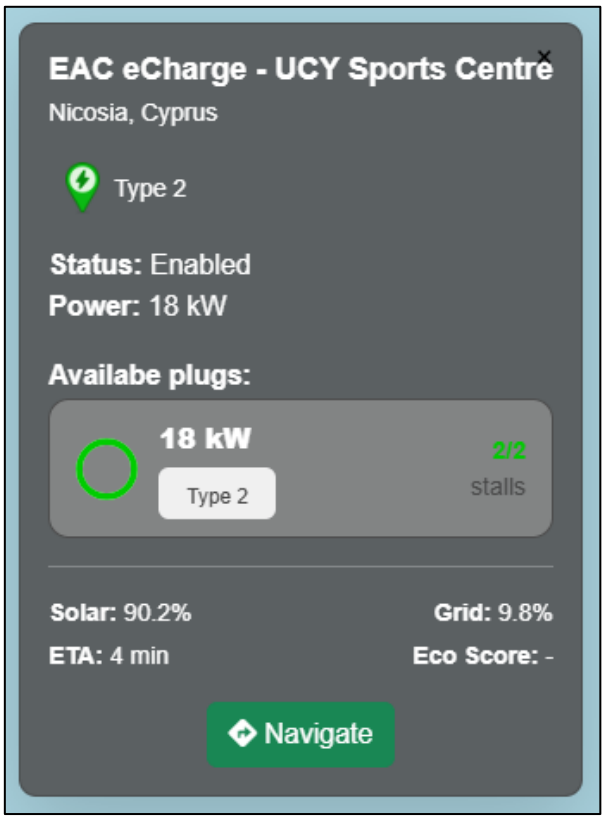
Η κατάταξη φορτιστών φαίνεται στην Εικόνα 5.13, όπου πάνω στον χάρτη εμφανίζονται πράσινοι κυκλικοί δείκτες με αριθμούς από 1 έως 5, που υποδεικνύουν τους κορυφαίους φορτιστές με βάση την οικολογική κατάταξη, σε συνδυασμό με τη διαδρομή και τη θέση του εικονιδίου ταξί.



Εικόνα 5. 13 Παράδειγμα κατάταξης φορτιστών

5.7 Κάρτα πληροφοριών φορτιστή

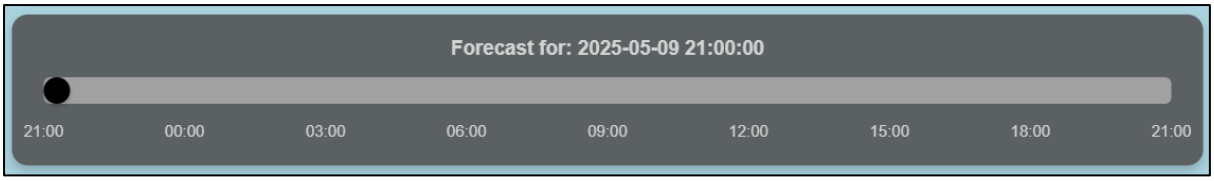
Στην Εικόνα 5.14 φαίνεται η κάρτα πληροφοριών που εμφανίζεται με το πάτημα πάνω σε έναν φορτιστή, η οποία περιέχει πληροφορίες όπως τοποθεσία, τύπος πρίζας, ισχύς σε kW, πλήθος διαθέσιμων θέσεων φόρτισης, ποσοστό ηλιακής και δικτυακής ενέργειας και χρόνος άφιξης (ETA), καθώς και κουμπί πλοήγησης για ευκολότερη δρομολόγηση σε αυτόν τον φορτιστή.



Εικόνα 5.14 Κάρτα πληροφοριών φορτιστή

5.8 Forecast slider

Στο Forecast tab εμφανίζεται στο κάτω μέρος του χάρτη μια οριζόντια μπάρα με slider, όπως στην Εικόνα 5.15, όπου ο χρήστης μπορεί να επιλέξει ώρα πρόβλεψης μέσα σε χρονικό διάστημα 24 ωρών. Πάνω από το slider προβάλλεται η ακριβής ημερομηνία και ώρα που αντιστοιχεί στην επιλεγμένη θέση, και με την αλλαγή της τιμής του slider ανανεώνονται αντίστοιχα και οι φορτιστές.

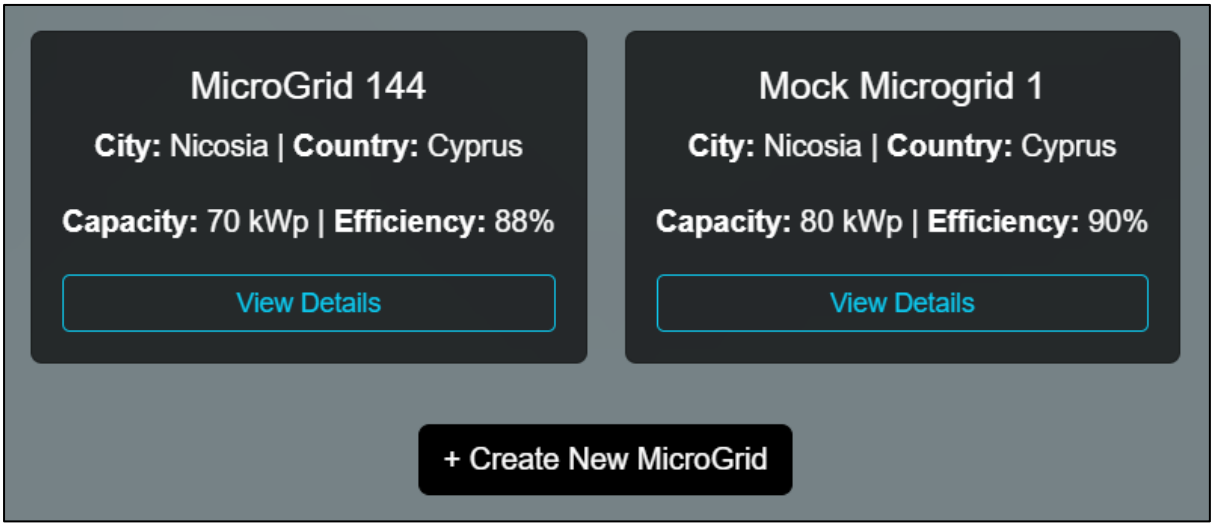


Εικόνα 5. 15 Forecast slider

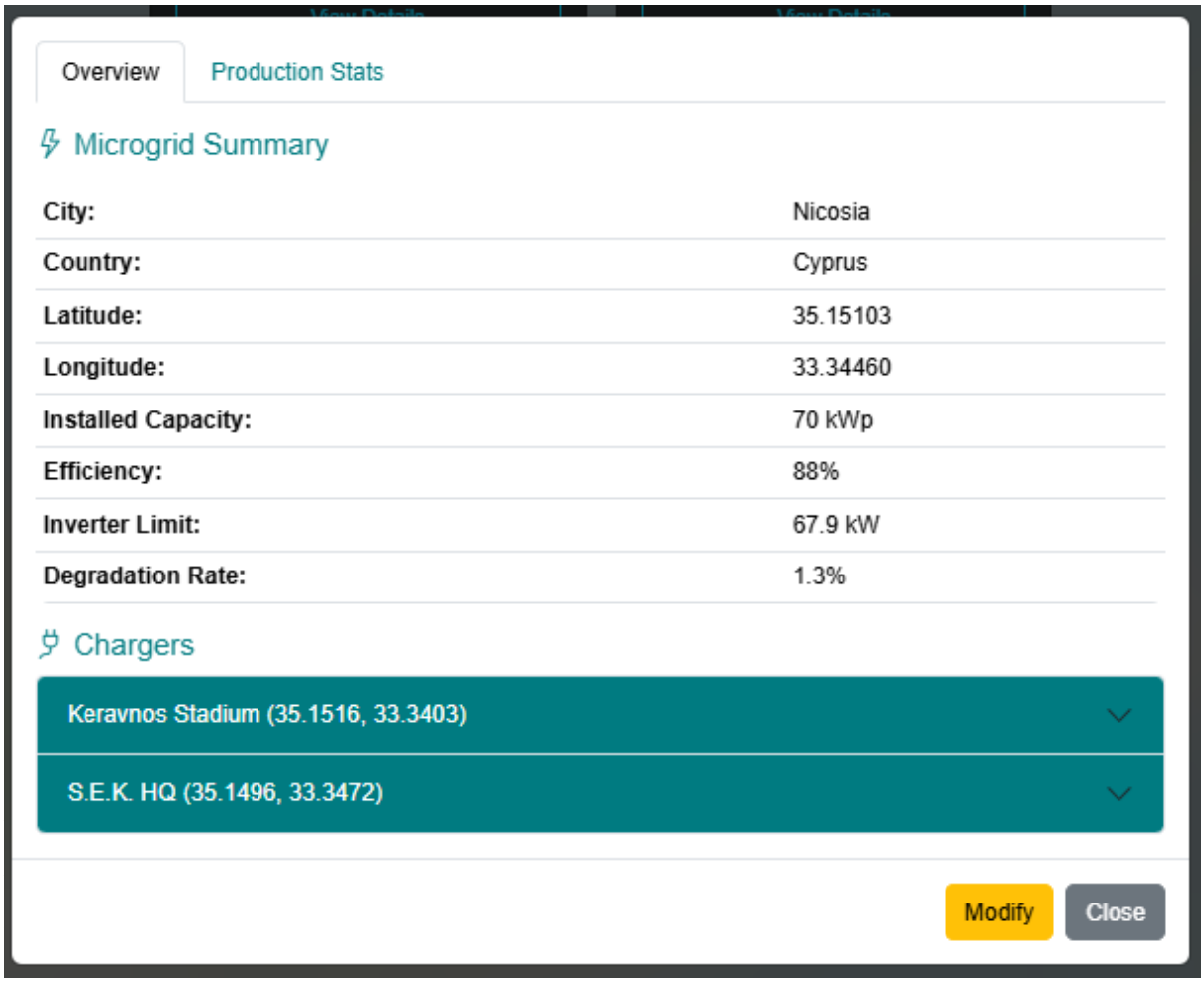
5.9 View My Grid tab

Στο tab View My Grid εμφανίζονται οι κάρτες των μικροδικτύων που ανήκουν σε έναν operator, κάθε μία με συνοπτικές πληροφορίες όπως το όνομα, την πόλη, τη χώρα, την εγκατεστημένη ισχύ (kWp) και την απόδοση, δίνοντας στον operator μια άμεση επισκόπηση

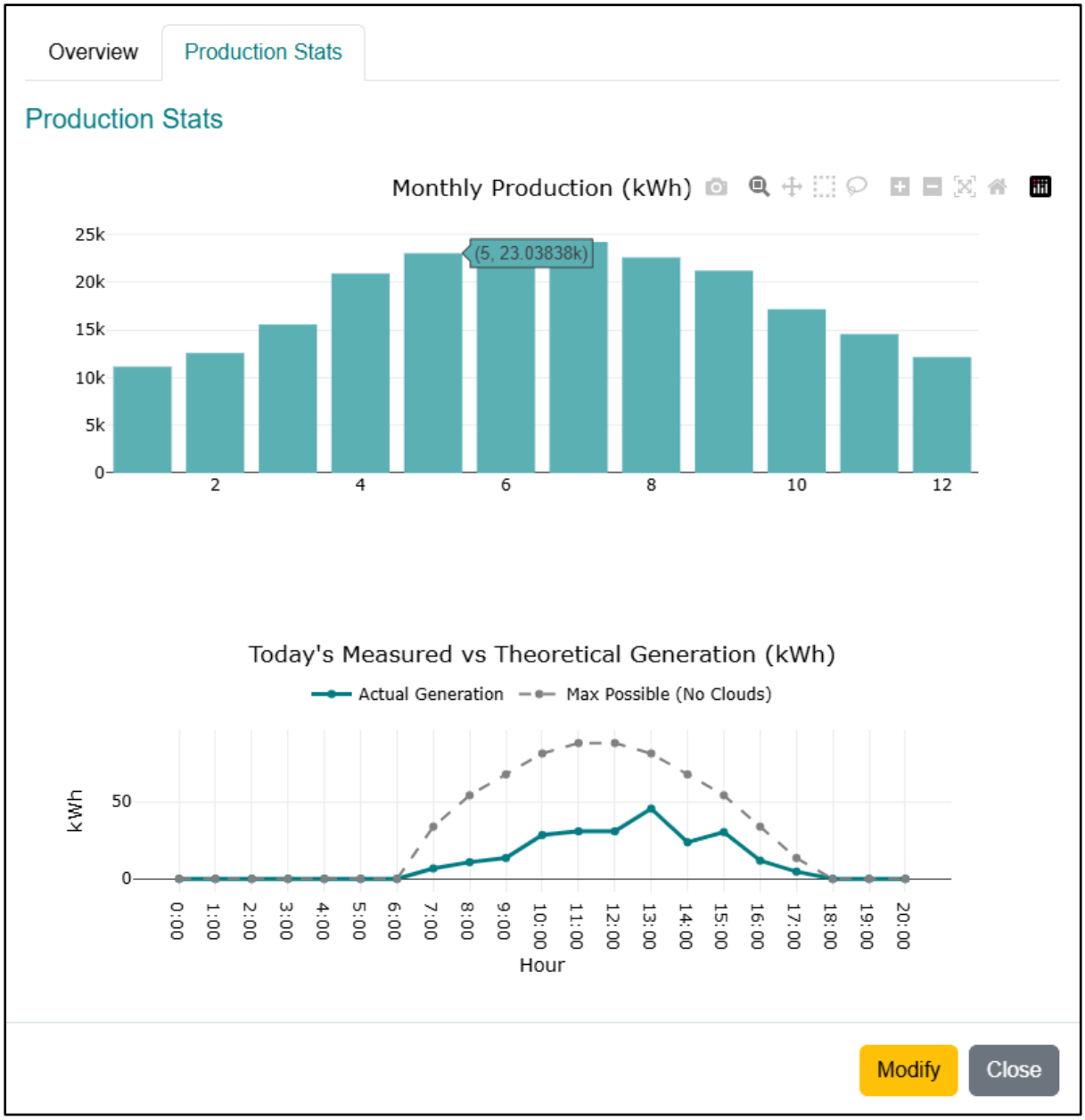
των μικροδικτύων που έχει δημιουργήσει. Υπάρχει επίσης το κουμπί View Details για περαιτέρω προβολή, όπου ανοίγει νέα καρτέλα, όπως φαίνεται στην Εικόνα 5.17 . Κάτω από τις κάρτες υπάρχει κουμπί Create New MicroGrid για την προσθήκη νέου μικροδικτύου.



Εικόνα 5. 16 Microgrid cards

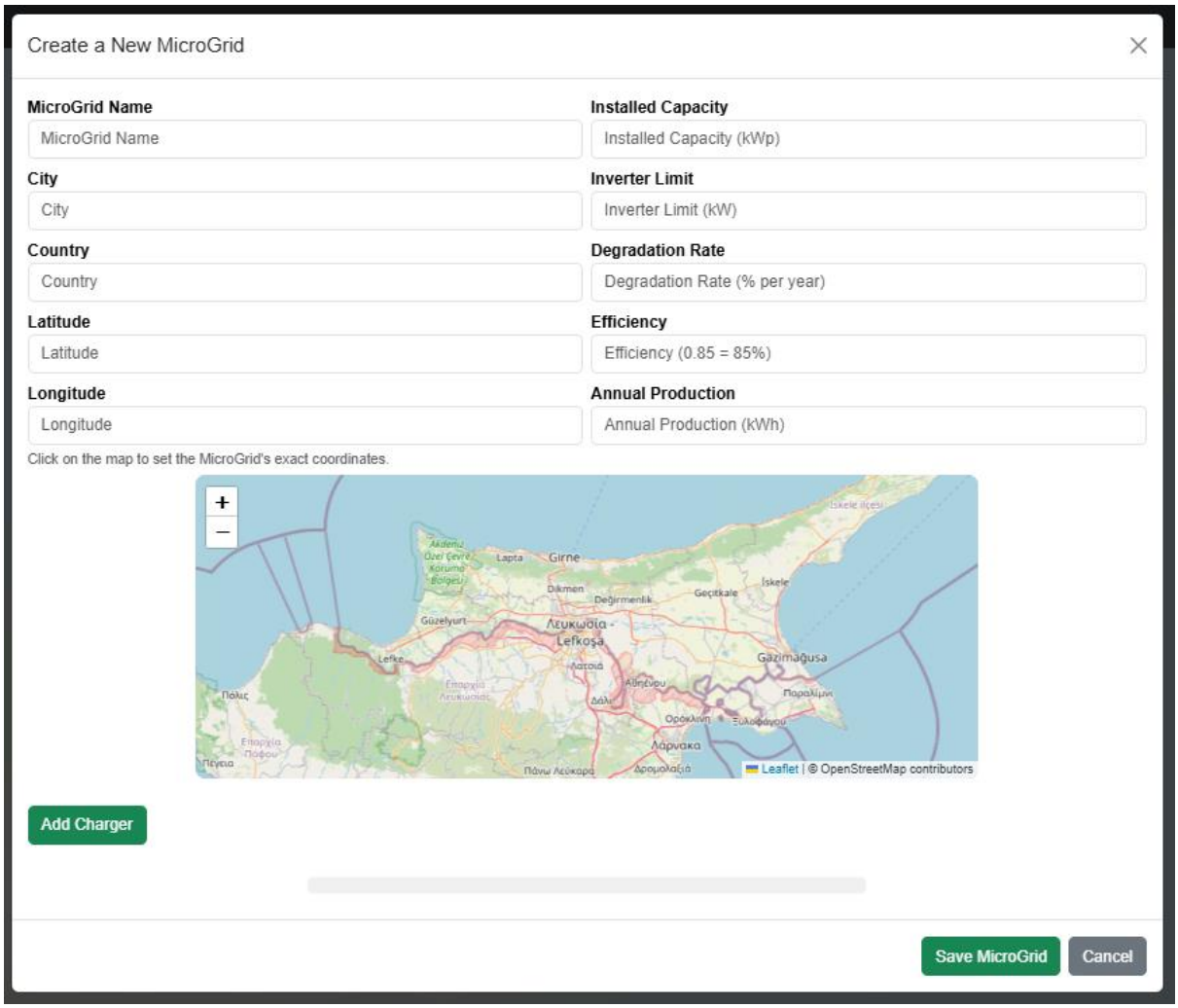


Εικόνα 5. 17 Microgrid details

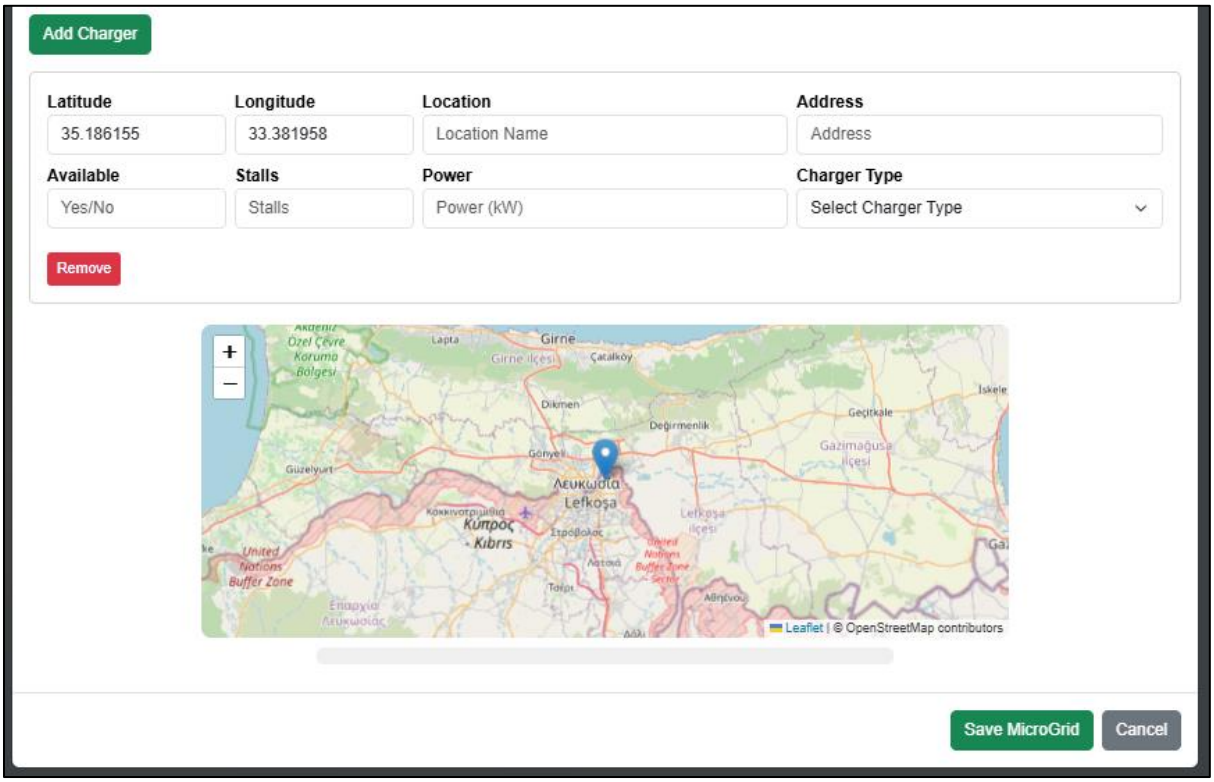


Εικόνα 5. 18 Microgrid production statistics

Κατά την επιλογή ‘Create New MicroGrid’, εμφανίζεται αναδυόμενο παράθυρο με πεδία συμπλήρωσης για τις λεπτομέρειες του μικροδικτύου, όπως αυτό στην Εικόνα 5.18. Στο κάτω μέρος υπάρχει διαδραστικός χάρτης για την επιλογή ακριβούς θέσης και δυνατότητα προσθήκης φορτιστών πριν την τελική αποθήκευση. Πατώντας το κουμπί ‘Add Charger’, το παράθυρο επεκτείνεται, όπως φαίνεται στην Εικόνα 5.19, και εμφανίζονται τα απαραίτητα πεδία για τη συμπλήρωση λεπτομερειών ενός φορτιστή, αλλά και ο χάρτης για την εύρεση της ακριβούς τοποθεσίας του φορτιστή.



Εικόνα 5. 18 Δημιουργία MicroGrid

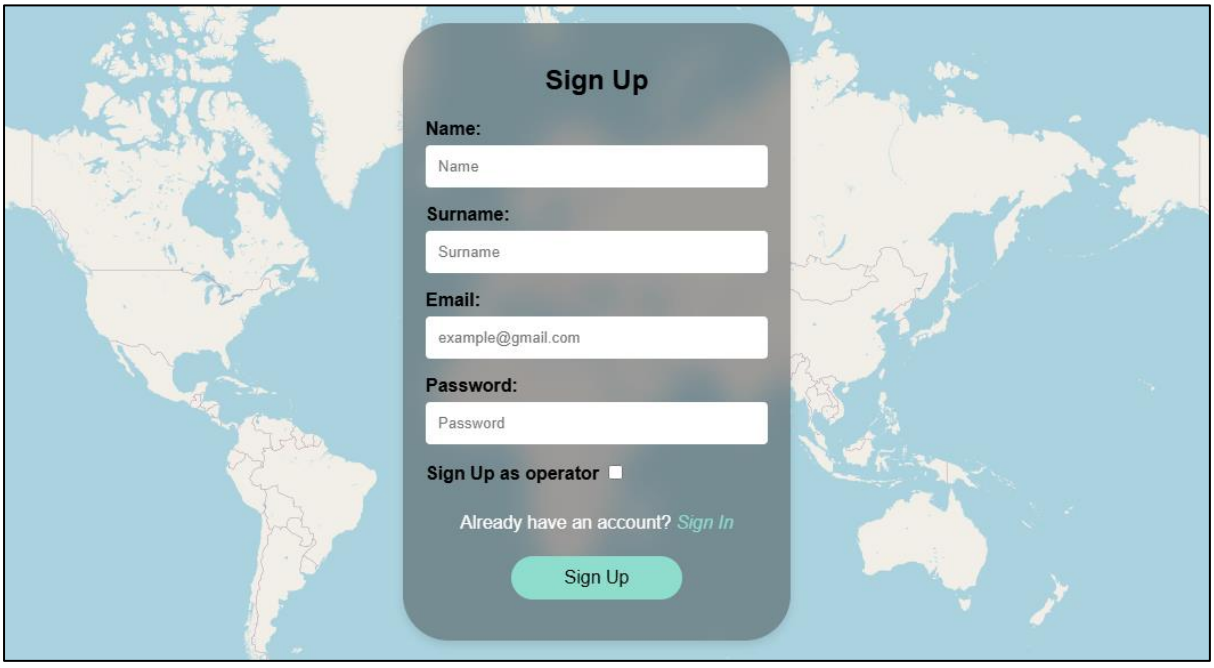


Εικόνα 5. 19 Προσθήκη φορτιστή

5.10 Σελίδες Αυθεντικοποίησης

Για την αυθεντικοποίηση του χρήστη υπάρχουν οι σελίδες για Sign In και Sign Up, οι οποίες περιέχουν τις κατάλληλες φόρμες και προβάλλονται στο κέντρο της οθόνης, μπροστά από υπόβαθρο παγκόσμιου χάρτη. Η φόρμα σύνδεσης περιλαμβάνει πεδία email και password, ενώ η φόρμα εγγραφής επιπλέον πεδία για όνομα, επώνυμο και επιλογή ρόλου ως operator.

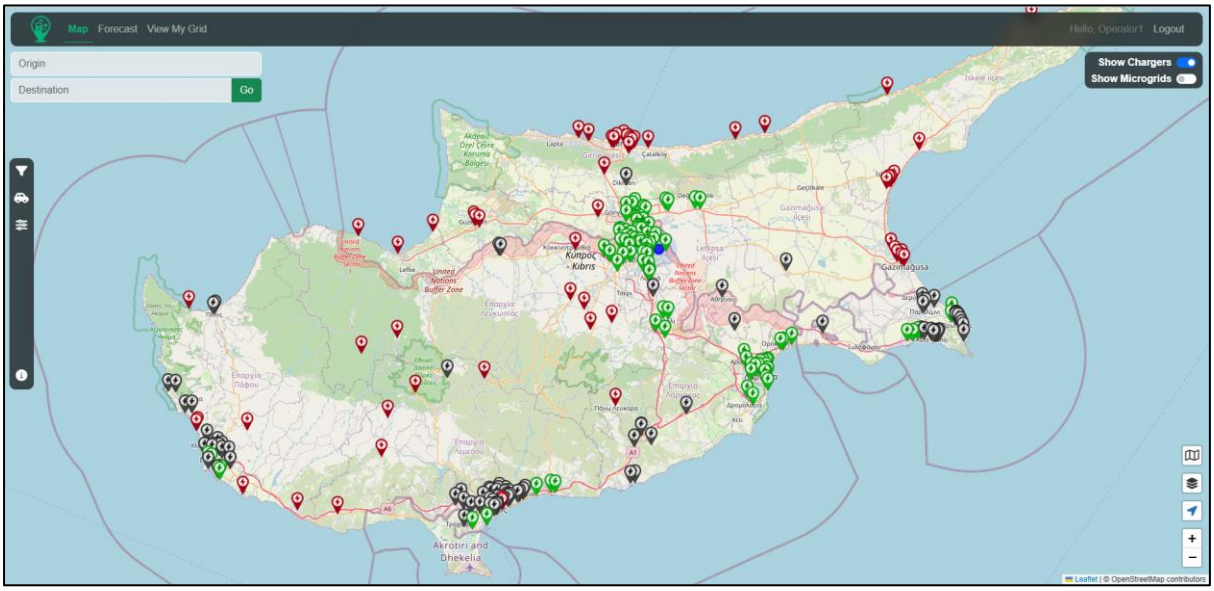
Εικόνα 5. 20 Sign Up form



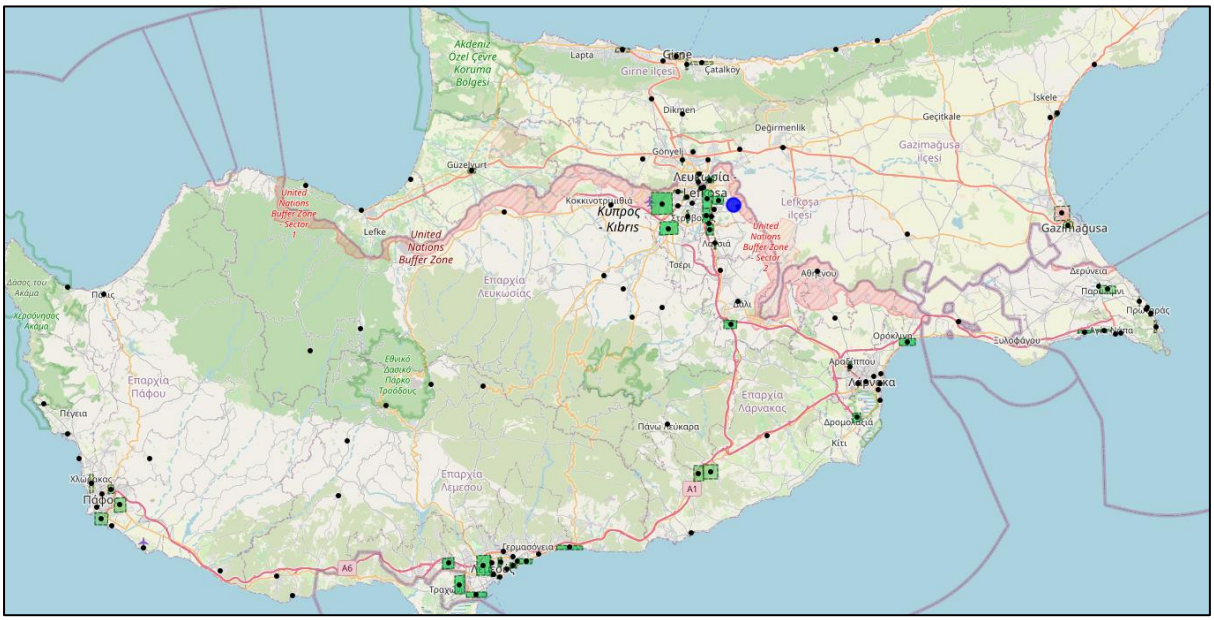
Εικόνα 5. 21 Sign In form

5.11 Βασική οθόνη

Η Εικόνα 5.22 απεικονίζει τη βασική οθόνη του συστήματος, στην οποία εμφανίζονται όλοι οι φορτιστές πάνω στον χάρτη. Κάθε φορτιστής απεικονίζεται με marker του οποίου το χρώμα αντανακλά το ποσοστό αξιοποίησης ηλιακής ενέργειας, όπου οι πράσινοι δείχνουν υψηλή χρήση ηλιακής παραγωγής, ενώ οι γκρι και κόκκινοι αντιστοιχούν σε περιορισμένη ή μηδενική αξιοποίηση. Στην Εικόνα 5.23 φαίνονται τα αντίστοιχα μικροδίκτυα, με το χρώμα τους επίσης να αντιστοιχεί στην ηλιακή τους παραγωγή.



Εικόνα 5. 22 Βασική οθόνη συστήματος



Εικόνα 5. 23 Παρουσίαση μικροδικτύων

Κεφάλαιο 6 Πειραματική μεθοδολογία και αποτίμηση

Σε αυτό το κεφάλαιο παρουσιάζεται η μεθοδολογία και η αποτίμηση των πειραμάτων που διεξαχθήκαν.

6.1 Χαρακτηριστικά εικονικής μηχανής

Τα χαρακτηριστικά της εικονικής μηχανή (Virtual Machine) που χρησιμοποιήθηκε για την διεξαγωγή των πειραμάτων είναι τα εξής:

- Αριθμός CPUs: 2
- Συχνότητα CPU: 2.10 GHz
- Αρχιτεκτονική CPU: x86_64
- Μοντέλο CPU: Intel® Xeon® Silver 4208 CPU @ 2.10 GHz
- Μνήμη: 2.05 GB
- Αποθήκευση: 98 GB (55 GB διαθέσιμα)
- Λειτουργικό σύστημα: Ubuntu 22.04.4 LTS
- Πυρήνας: Linux 6.8.0-58-generic
- Hypervisor: VMware

6.2 Πείραμα στο Επίπεδο Δεδομένων

6.2.1 Σύγκριση Ανάκτησης Δεδομένων: JSON File vs SQLite

Σε αυτό το πείραμα μελετάται η απόδοση της ανάκτησης δεδομένων φορτιστών από δύο διαφορετικές πηγές δεδομένων: τη βάση SQLite και ένα τοπικό αρχείο τύπου JSON. Η μέτρηση στοχεύει στο να διαπιστωθεί ποια μέθοδος είναι πιο αποτελεσματική σε χρόνο εκτέλεσης, υπό ισοδύναμο όγκο πληροφορίας.

Η πρώτη μέθοδος βασίζεται σε ερώτημα προς τη βάση MicroGrid.db, και συγκεκριμένα στον πίνακα Chargers. Το SQL ερώτημα που χρησιμοποιήθηκε είναι:

```
SELECT id, LocationName, Power, latitude, longitude, Address, Available, Stalls
FROM Chargers;
```

Η εκτέλεση του ερωτήματος πραγματοποιήθηκε μέσω Python, με χρήση της βιβλιοθήκης sqlite3, για 10 επαναλήψεις. Κάθε εκτέλεση μετρούσε τον ακριβή χρόνο απόκρισης με τη χρήση της time().

Η δεύτερη μέθοδος αξιολόγησε την απόδοση της ανάγνωσης των ίδιων δεδομένων σε μορφή .json, μέσω της Python και της βιβλιοθήκης json. Το αρχείο cyprus.json περιείχε όλα τα δεδομένα του πίνακα Chargers σε σειρά αντικειμένων JSON. Η διαδικασία εκτέλεσης περιλάμβανε 10 αναγνώσεις του αρχείου και μετατροπή του σε Python αντικείμενο.

Ο παρακάτω πίνακας συνοψίζει τους χρόνους απόκρισης ανά επανάληψη για τις δύο διαφορετικές τεχνικές ανάκτησης δεδομένων. Κάθε τιμή αντιστοιχεί στον χρόνο που απαιτήθηκε για την πλήρη φόρτωση των δεδομένων φορτιστών σε Python αντικείμενο. Η μέση τιμή προκύπτει από τις δέκα μετρήσεις και επιτρέπει τη συγκριτική αξιολόγηση των δύο μεθόδων.

	SQLite (ms)	JSON (ms)
	0.661	72.833
	0.313	7.835
	0.3	3.274
	0.288	3.424
	0.285	2.48
	0.3	2.707
	0.331	2.509
	0.295	2.448
	0.299	2.217
	0.317	2.298
Average	0.3389	10.2025

Πίνακας 6. 1 Χρόνοι ανάκτησης φορτιστών από Βάση Δεδομένων και .json αρχείο

Τα αποτελέσματα έδειξαν ότι η χρήση της SQLite ήταν αισθητά ταχύτερη. Συγκεκριμένα, ο μέσος χρόνος ανάκτησης από τη βάση ήταν μόλις 0.000339 δευτερόλεπτα, ενώ η αντίστοιχη ανάγνωση από JSON αρχείο είχε μέσο χρόνο 0.0102025 δευτερόλεπτα. Αξιοσημείωτο είναι επίσης ότι η πρώτη εκτέλεση ανάγνωσης από το JSON παρουσίασε καθυστέρηση ~0.07s λόγω αρχικής πρόσβασης στον δίσκο, ενώ οι επόμενες σταθεροποιήθηκαν.

Τα παραπάνω υποδεικνύουν ότι, παρότι τα αρχεία JSON προσφέρουν ευκολία και ευανάγνωστη μορφή για στατικά δεδομένα ή caching, η βάση δεδομένων SQLite αποτελεί σαφώς πιο αποτελεσματική επιλογή για επαναλαμβανόμενη και δυναμική πρόσβαση σε δεδομένα, ακόμα και σε μικρές βάσεις.

6.3 Πείραμα στο Υπολογιστικό Επίπεδο

6.3.1 Πειραματικές διαδρομές

Οι διαδρομή που χρησιμοποιήθηκε για αυτό το πείραμα είναι η ακόλουθη:

1. Αρχή: Nicosia, Nicosia Muncipality, Nicosia District, Cyprus
   Τέλος: Paphos Municipality, Paphos District, Cyprus

6.3.2 Μέτρηση Χρόνου Εκτέλεσης Οικολογικού Σκορ (EcoScore)

Η παρούσα δοκιμή αφορά τη μέτρηση του χρόνου υπολογισμού για την κατάταξη φορτιστών με βάση το οικολογικό σκορ (EcoScore), μέσω του backend της εφαρμογής. Ο υπολογισμός πραγματοποιείται από τη συνάρτηση get_ranking_chargers() η οποία ενεργοποιείται κάθε φορά που ο χρήστης ζητά προτεινόμενους φορτιστές μέσω του endpoint /nearby_chargers.

Για τις ανάγκες του πειράματος, επιλέχθηκε η Διαδρομή 1 και επαναλήφθηκε η διαδικασία υπολογισμού του σκορ για πέντε διαφορετικές ακτίνες αναζήτησης (2 km, 10 km, 25 km, 50 km, 75 km). Η μέτρηση του χρόνου απόκρισης έγινε μέσω των Developer Tools του browser, και συγκεκριμένα μέσω της καρτέλας "Timing", όπου καταγράφηκε ο χρόνος "Waiting for server response", δηλαδή ο καθαρός χρόνος επεξεργασίας του αιτήματος από τον server.

Ο παρακάτω πίνακας συνοψίζει τα αποτελέσματα:

Ακτίνα (km)	Χρόνος Εκτέλεσης (ms)
2	208.48
10	592.48
25	832.44
50	1230
75	2070

Πίνακας 6. 2 Χρόνοι εκτέλεσης οικολογικού σκορ σε πραγματικό χρόνο

Από τα αποτελέσματα παρατηρείται ότι ο χρόνος υπολογισμού αυξάνεται σχεδόν γραμμικά ως προς το πλήθος των φορτιστών που βρίσκονται εντός της καθορισμένης ακτίνας. Το γεγονός ότι η αύξηση του χρόνου είναι προβλέψιμη και ομαλή δείχνει ότι η συνάρτηση κατάταξης κλιμακώνεται ικανοποιητικά και παραμένει κατάλληλη για χρήση σε πραγματικό χρόνο, τουλάχιστον για ακτίνες έως 25–50 km.

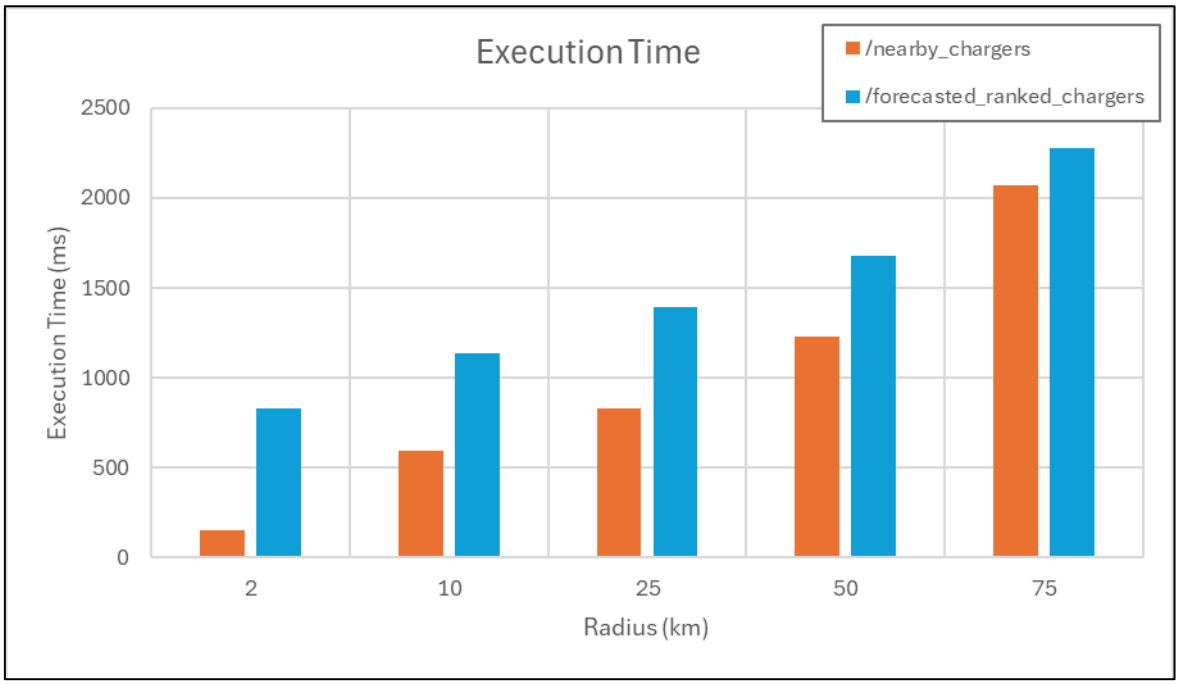
Πέρα από την κατάταξη φορτιστών με βάση τα τρέχοντα δεδομένα, η πλατφόρμα υποστηρίζει και κατάταξη με βάση προβλεπόμενα δεδομένα ηλιακής ενέργειας για την επόμενη ημέρα, μέσω του endpoint /forecasted_ranked_chargers. Στόχος του παρόντος πειράματος είναι να εξεταστεί αν ο υπολογισμός κατάταξης με προβλεπόμενα δεδομένα επηρεάζει τον χρόνο εκτέλεσης, δεδομένου ότι ενσωματώνει προβλέψεις καιρού και ηλιακής παραγωγής για διαφορετικές ώρες του επόμενου 24ώρου.

Κατά την εκτέλεση του πειράματος, χρησιμοποιήθηκαν οι ίδιες ακτίνες αναζήτησης (2–75 km), δηλαδή αυξανόμενο πλήθος φορτιστών κάθε φορά, όπως στο προηγούμενο σενάριο, για επόμενη χρονική στιγμή (π.χ. επόμενη ημέρα στις 12:00). Η λειτουργία αυτή ενεργοποιείται μέσω του endpoint /forecasted_ranked_chargers, το οποίο χρησιμοποιεί ως είσοδο δεδομένα πρόγνωσης που έχουν προηγουμένως ληφθεί από το OpenWeather API.

Ακτίνα (km)	Χρόνος Εκτέλεσης (ms)
2	827.09
10	1140
25	1390
50	1680
75	2280

Πίνακας 6. 3 Χρόνοι εκτέλεσης οικολογικού σκορ στο κοντινό μέλλον

Οι μετρήσεις έδειξαν ότι ο χρόνος απόκρισης αυξάνεται σχεδόν γραμμικά με την ακτίνα, γεγονός που υποδηλώνει ότι η υπολογιστική πολυπλοκότητα της μεθόδου είναι ανάλογη του αριθμού των φορτιστών που εξετάζονται. Για ακτίνα 2 km, ο μέσος χρόνος εκτέλεσης ήταν περίπου 827 ms, ενώ για τα 75 km έφτασε τα 2.28 δευτερόλεπτα, αποδεικνύοντας ότι η διαδικασία κατάταξης απαιτεί σημαντικούς υπολογιστικούς πόρους όταν αυξάνεται η ακτίνα αναζήτησης. Το Γράφημα 6.1 απεικονίζει τη σύγκριση του χρόνου εκτέλεσης μεταξύ των endpoints /nearby_chargers και /forecasted_ranked_chargers για διαφορετικές ακτίνες αναζήτησης.



Γράφημα 6. 1 Σύγκριση του χρόνου εκτέλεσης των endpoints /nearby_chargers και /forecasted_ranked_chargers για διαφορετικές ακτίνες αναζήτησης.

Παρατηρείται ότι και στις δύο περιπτώσεις, ο χρόνος αυξάνεται αναλογικά με την ακτίνα, καθώς αυξάνεται και το πλήθος των φορτιστών που επεξεργάζεται ο αλγόριθμος. Ωστόσο, οι μετρήσεις κατέδειξαν σαφή διαφορά στην απόδοση μεταξύ των δύο μεθόδων. Για παράδειγμα, σε ακτίνα 2 km, η κατάταξη με πραγματικά δεδομένα εκτελέστηκε σε περίπου 208 ms, ενώ η αντίστοιχη με προβλεπόμενα δεδομένα απαίτησε περίπου 827 ms, δηλαδή σχεδόν τετραπλάσιο χρόνο. Καθώς η ακτίνα αυξάνεται, η διαφορά παραμένει σταθερή: στα 75 km, η real-time κατάταξη απαιτεί περίπου 2.07 δευτερόλεπτα, ενώ η forecast κατάταξη ξεπερνά τα 2.28 δευτερόλεπτα. Αν και η αύξηση δεν είναι εκθετική, αποδεικνύει ότι η επεξεργασία των προβλεπόμενων δεδομένων αποτελεί υπολογιστικό βάρος για το backend.

Η διαφορά αυτή οφείλεται στο γεγονός ότι η forecast κατάταξη περιλαμβάνει επιπλέον επεξεργασία: για κάθε φορτιστή υπολογίζεται η αναμενόμενη ηλιακή παραγωγή για την επιλεγμένη ώρα, λαμβάνοντας υπόψη τη νεφοκάλυψη, την αποδοτικότητα του αντίστοιχου microgrid, και τη σχετική ωριαία κατανομή παραγωγής. Οι υπολογισμοί αυτοί γίνονται δυναμικά κατά την ώρα του αιτήματος, σε αντίθεση με τα δεδομένα πραγματικού χρόνου τα οποία έχουν ήδη προϋπολογιστεί και αποθηκευτεί στη βάση. Ως αποτέλεσμα, η forecast-based κατάταξη είναι σημαντικά πιο βαριά υπολογιστικά, ιδιαίτερα όταν περιλαμβάνει μεγάλο αριθμό φορτιστών.

6.3.3 Μέτρηση Οικολογικού Σκορ

Για την περαιτέρω αξιολόγηση της υπολογιστικής λογικής του συστήματος, πραγματοποιήθηκε σύγκριση μεταξύ της αθροιστικής οικολογικής βαθμολογίας (Eco Score) των πέντε κορυφαίων φορτιστών που προτείνονται σε κάθε ακτίνα, για τις δύο εκδόσεις της κατάταξης: σε πραγματικό χρόνο και με προβλεπόμενα δεδομένα. Οι μετρήσεις πραγματοποιήθηκαν για ακτίνες 2, 10, 25, 50 και 75 χιλιομέτρων. Τα δεδομένα σε πραγματικό χρόνο αφορούσαν χρονική στιγμή περίπου στις 12:00 το μεσημέρι, όπου η παραγωγή ηλιακής ενέργειας βρίσκεται στο μέγιστο της καμπύλης ημερήσιας παραγωγής. Αντίθετα, τα προβλεπόμενα δεδομένα βασίζονταν σε πρόγνωση για τις 15:00, χρονικό σημείο όπου η ηλιακή απόδοση αρχίζει σταδιακά να μειώνεται.

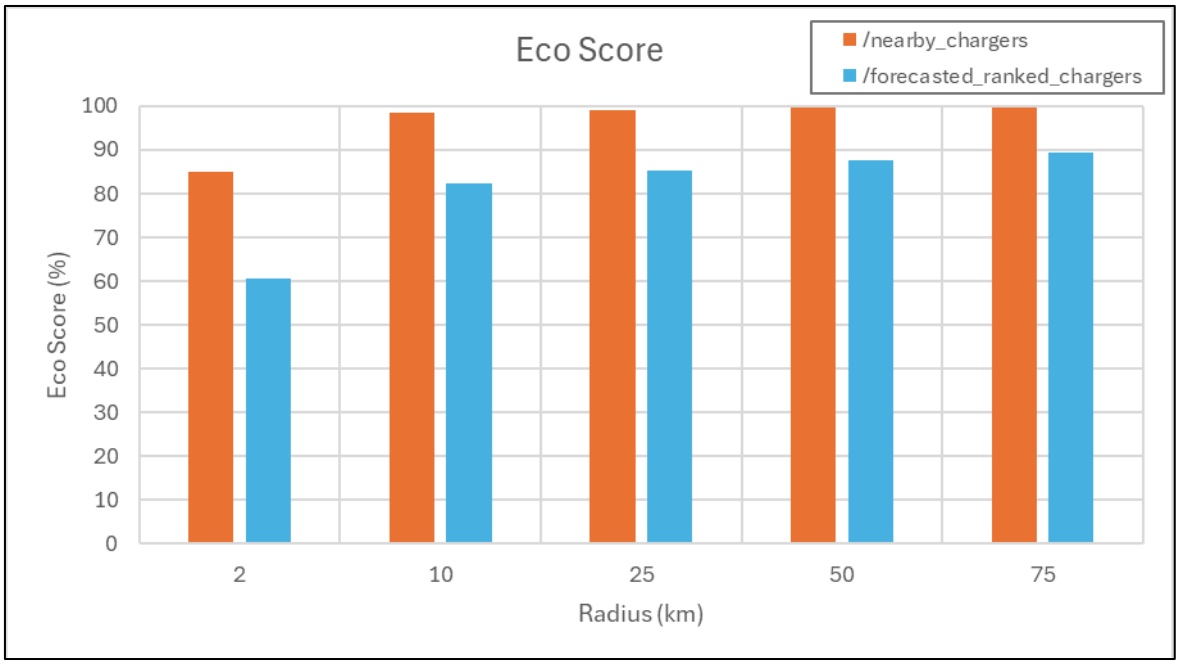
Ο Πίνακας 6.4 συνοψίζει τα αποτελέσματα των οικολογικών σκορ ανά ακτίνα:

Radius (km)	Real-Time Eco Scores	Forecasted Eco Scores
2	855.27	610.4
10	988.95	828.3
25	996.21	856.75
50	1001.34	880.9
75	1001.52	899.2

Πίνακας 6. 4 Eco Score for different radii

Η διαφορά στην ηλιακή παραγωγή μεταξύ των δύο χρονικών σημείων αποτυπώθηκε στις συνολικές οικολογικές βαθμολογίες. Σε όλες τις ακτίνες, οι αθροιστικές τιμές Eco Score της κατάταξης σε πραγματικό χρόνο ήταν υψηλότερες από τις αντίστοιχες της πρόβλεψης. Για παράδειγμα, στην ακτίνα των 2 km, το άθροισμα των Eco Scores ήταν 855.27 έναντι 610.40, ενώ στην ακτίνα των 75 km ήταν 1001.52 έναντι 899.20. Η διαφορά αυτή οφείλεται κυρίως στον παράγοντα της ηλιακής παραγωγής, ο οποίος αποτελεί βασικό συστατικό του υπολογισμού του οικολογικού σκορ και επηρεάζεται άμεσα από την ηλιοφάνεια και ηλιακή ακτινοβολία κάθε χρονικής στιγμής.

Η απεικόνιση του Γραφήματος 6.2.2 επιβεβαιώνει την επίδραση της χρονικής στιγμής στην αξιολόγηση των φορτιστών.



Γράφημα 6. 2 Eco Score for different times

Στις μικρές ακτίνες, η κατάταξη σε πραγματικό χρόνο εμφανίζει σαφώς υψηλότερες βαθμολογίες, με διαφορά που ξεπερνά τις 20 ποσοστιαίες μονάδες στα 2 km. Αυτό συμβαίνει διότι σε μικρότερες ακτίνες το πλήθος των διαθέσιμων φορτιστών είναι περιορισμένο, γεγονός που μειώνει τη δυνατότητα επιλογής φορτιστών με την υψηλότερη οικολογική απόδοση. Εάν οι κοντινοί φορτιστές διαθέτουν μέτρια χαρακτηριστικά, όπως χειρότερη διαθεσιμότητα ή λιγότερη ηλιακή κάλυψη, τότε το σύστημα υποχρεούται να τους κατατάξει, ακόμα κι αν οι οικολογικές βαθμολογίες τους είναι χαμηλές. Αντίθετα, σε μεγαλύτερες ακτίνες η αύξηση των υποψήφιων φορτιστών αυξάνει και τις πιθανότητες εντοπισμού φορτιστών με εξαιρετική βαθμολογία σε όλους τους επιμέρους δείκτες (διαθεσιμότητα, ηλιακή παραγωγή, απόσταση), οδηγώντας έτσι σε συνολικά υψηλότερα οικολογικά σκορ. Ωστόσο, σε κάθε περίπτωση η κατάταξη σε πραγματικό χρόνο υπερτερεί, γεγονός που αποδίδεται στη μέγιστη ηλιακή παραγωγή κατά την ώρα λήψης των δεδομένων (12:00), έναντι μειωμένης παραγωγής στην πρόβλεψη (15:00). Η γραφική σύγκριση καταδεικνύει την ανάγκη δυναμικής αξιολόγησης φορτιστών όχι μόνο χωρικά αλλά και χρονικά.

Η ανάλυση αυτή αναδεικνύει τη σημασία του χρονικού παράγοντα στον υπολογισμό της οικολογικότητας ενός φορτιστή. Το ίδιο γεωγραφικό σημείο και η ίδια εγκατάσταση μπορεί να αποδώσει διαφορετικά σκορ ανάλογα με την ώρα της ημέρας, γεγονός που υποστηρίζει την ανάγκη για δυναμική προσαρμογή των συστάσεων με βάση τη χρονική στιγμή που ο χρήστης σκοπεύει να φτάσει στον προτεινόμενο φορτιστή.

6.4 Πείραμα στο Επίπεδο Διεπαφής Χρήστη

6.4.1 Ανάλυση Απόκρισης Διεπαφής Χρήστη

Το παρόν πείραμα επικεντρώνεται στη μέτρηση του χρόνου απόκρισης της διεπαφής χρήστη του συστήματος κατά την εκτέλεση αλληλεπιδράσεων που προκαλούν οπτικές αλλαγές, όπως η ενημέρωση των φορτιστών στον χάρτη, η εμφάνιση καρτών με πληροφορίες ή η δυναμική επεξεργασία στοιχείων βάσει προτιμήσεων του χρήστη. Η μελέτη αυτή έχει ως στόχο να αποτιμήσει την εμπειρία χρήσης σε όρους ταχύτητας και αμεσότητας, όπως αντιλαμβάνεται ο τελικός χρήστης.

Για την αξιολόγηση του χρόνου απόκρισης της διεπαφής χρήστη πραγματοποιήθηκαν μετρήσεις με χρήση του εργαλείου Performance του προγράμματος περιήγησης (Google

Chrome DevTools). Συγκεκριμένα, καταγράφηκαν τα χρονικά διαστήματα μεταξύ της ενεργοποίησης μιας αλληλεπίδρασης και της πλήρους απόδοσης της οπτικής πληροφορίας στην οθόνη. Οι ενέργειες που αξιολογήθηκαν είναι οι εξής:

1. Εμφάνιση όλων των φορτιστών μέσω του κουμπιού Show All Chargers, το οποίο ενεργοποιεί την απόδοση όλων των διαθέσιμων markers στον χάρτη Leaflet.
2. Εμφάνιση λεπτομερειών μικροδικτύου με το πάτημα του κουμπιού View Details, το οποίο ενεργοποιεί την εμφάνιση ενός modal παραθύρου με αναλυτικά δεδομένα.
3. Αλλαγή τύπου πρίζας, η οποία ενεργοποιεί δυναμικό φιλτράρισμα των markers ώστε να εμφανίζονται μόνο οι συμβατοί φορτιστές.
4. Αλλαγή καρτέλας διεπαφής μεταξύ Map Tab, Forecast Tab και View My Grid Tab, η οποία επηρεάζει την απόδοση διαφορετικών επιφανειών της διεπαφής με διακριτό περιεχόμενο.

Σε κάθε περίπτωση, καταγράφηκε η χρονική διάρκεια μεταξύ του event (click, change, tab switch) και της ολοκλήρωσης των φάσεων scripting, rendering και painting. Η μέτρηση επαναλήφθηκε τρεις φορές ανά ενέργεια και υπολογίστηκε ο μέσος χρόνος απόκρισης.

Ο παρακάτω πίνακας συνοψίζει τους μέσους χρόνους απόκρισης UI για τις τέσσερις υπό εξέταση ενέργειες:

Action	UI Response Time (ms)
Show All Chargers	2200
Open Microgrid Details	1897
Change Plug Type Filter	1870
Switch Tab	640

Πίνακας 6. 5 Χρόνοι απόκρισης διεπαφής χρήστη

Η ενέργεια με τον μεγαλύτερο χρόνο απόκρισης ήταν η καθολική απόδοση φορτιστών στον χάρτη, λόγω του πλήθους markers και DOM updates που απαιτούνται. Αντίστοιχα υψηλό χρόνο παρουσίασε και η εμφάνιση modal με λεπτομέρειες μικροδικτύου, καθώς περιλαμβάνει animation, δυναμική φόρτωση περιεχομένου και επεξεργασία DOM. Η αλλαγή τύπου πρίζας είχε επίσης σημαντικό χρόνο, καθώς επηρεάζει την ορατότητα μεγάλου αριθμού φορτιστών στον χάρτη. Η ταχύτερη απόκριση παρατηρήθηκε στην απλή εναλλαγή tabs, καθώς βασίζεται κυρίως σε CSS visibility χωρίς ουσιαστική αναδόμηση περιεχομένου.

Τα αποτελέσματα δείχνουν ότι, ακόμη και για σύνθετες αλληλεπιδράσεις όπως η απόδοση όλων των φορτιστών ή η εμφάνιση ενός modal με λεπτομέρειες, η διεπαφή διατηρεί ικανοποιητική απόκριση κάτω από τα 2.5 δευτερόλεπτα. Η καθυστέρηση που παρατηρείται είναι αναμενόμενη, δεδομένου του όγκου περιεχομένου που αποδίδεται και του αριθμού των DOM μεταβολών. Οι ενέργειες που σχετίζονται με ελαφριές αλλαγές στο περιβάλλον χρήστη (όπως αλλαγή tab) εκτελούνται με σχεδόν ακαριαίο ρυθμό, γεγονός που ενισχύει τη συνολική αίσθηση ρευστότητας και διαδραστικότητας της εφαρμογής.

Κεφάλαιο 7 Συμπεράσματα

7.1 Συμπεράσματα

Η παγκόσμια στροφή προς τη βιώσιμη κινητικότητα και τη μείωση του περιβαλλοντικού αποτυπώματος έχει καταστήσει την ηλεκτροκίνηση έναν από τους σημαντικότερους άξονες πράσινης τεχνολογικής ανάπτυξης. Ωστόσο, η απλή μετάβαση σε ηλεκτρικά οχήματα δεν επαρκεί από μόνη της για την επίτευξη ουσιαστικής μείωσης των εκπομπών. Η προέλευση της ενέργειας που χρησιμοποιείται για τη φόρτιση παίζει καθοριστικό ρόλο στην περιβαλλοντική αξία της ηλεκτροκίνησης. Η ενσωμάτωση ανανεώσιμων πηγών ενέργειας –όπως η ηλιακή και η αιολική– στις υποδομές φόρτισης κρίνεται απαραίτητη για την πραγματική μετάβαση σε καθαρές μεταφορές. Στο πλαίσιο αυτό, καθίσταται αναγκαία η ανάπτυξη έξυπνων συστημάτων που αξιολογούν, κατατάσσουν και προτείνουν φορτιστές με βάση την οικολογική τους απόδοση, συνυπολογίζοντας παράγοντες όπως η τοπική ηλιακή παραγωγή, η απόσταση και η διαθεσιμότητα.

Η παρούσα διπλωματική εργασία πραγματεύεται την υλοποίηση ενός ολοκληρωμένου γεωγραφικού και υπολογιστικού συστήματος που στοχεύει στη βιώσιμη φόρτιση ηλεκτρικών οχημάτων μέσω ενός οικοσυστήματος μικροδικτύων. Το EcoCharge+ αποτελεί μια καινοτόμα προσέγγιση στην κατεύθυνση της περιβαλλοντικά ευαισθητοποιημένης ηλεκτροκίνησης, με έμφαση στην αξιοποίηση ανανεώσιμων πηγών ενέργειας και στη δυναμική προσαρμογή των προτεινόμενων διαδρομών βάσει πραγματικών και προβλεπόμενων δεδομένων παραγωγής ενέργειας.

Η μετάβαση προς ένα μέλλον με ηλεκτρικά οχήματα κρίνεται απαραίτητη για τη μείωση των εκπομπών αερίων του θερμοκηπίου. Ωστόσο, η επίτευξη ουσιαστικού περιβαλλοντικού οφέλους απαιτεί η ενέργεια που χρησιμοποιείται για τη φόρτιση των EVs να προέρχεται κατά προτεραιότητα από ανανεώσιμες πηγές όπως τα φωτοβολταϊκά. Το προτεινόμενο σύστημα λαμβάνει υπόψη αυτή την παράμετρο, ενσωματώνοντας χωρικά και χρονικά δεδομένα καθώς

και καιρικές συνθήκες ώστε να υπολογίσει ένα "οικολογικό σκορ" για κάθε φορτιστή ή μικροδίκτυο.

Το σύστημα περιλαμβάνει βάση δεδομένων με λεπτομερή πληροφόρηση για MicroGrids, φορτιστές, προφίλ παραγωγής ενέργειας, τύπους πριζών και οχήματα. Αξιοποιεί το OpenWeather API για την περιοδική λήψη μετεωρολογικών δεδομένων και πρόβλεψης νέφωσης., αλλά και υπολογιστικά εργαλεία που εκτιμούν την παρούσα και μελλοντική ηλιακή παραγωγή σε κάθε σημείο του χάρτη με βάση μηνιαία και ωριαία προφίλ παραγωγής. Τέλος, ενσωματώνει έναν ευφυή αλγόριθμο κατάταξης φορτιστών, ο οποίος υπολογίζει έναν οικολογικό δείκτη ανάλογα με την απόσταση, τη διαθεσιμότητα, και τη βιωσιμότητα κάθε σημείου φόρτισης.

Η εφαρμογή υλοποιήθηκε με σύγχρονες τεχνολογίες frontend (Leaflet.js, Bootstrap), backend (Flask, SQLite). Η αρχιτεκτονική του συστήματος διαρθρώθηκε σε τρία επίπεδα: επίπεδο δεδομένων, υπολογιστικό επίπεδο και διεπαφή χρήστη, επιτρέποντας ευελιξία, επεκτασιμότητα και ευχρηστία. Στο πλαίσιο της υλοποίησης, δόθηκε ιδιαίτερη έμφαση στην αποδοτική αναπαράσταση των μικροδικτύων, την πρόβλεψη ηλιακής παραγωγής και την ομαλή διασύνδεση χαρτογραφικής διεπαφής με το backend σύστημα λογικής. Η διεπαφή επιτρέπει στο χρήστη να ορίσει προτιμήσεις, να επιλέξει το όχημά του, να εφαρμόσει φίλτρα βάσει τύπου πρίζας και να δει σε πραγματικό χρόνο ποιοι φορτιστές είναι περισσότερο οικολογικοί. Ο χρήστης έχει επίσης τη δυνατότητα να σύρει έναν δείκτη-ταξί στο χάρτη και να δει τους πέντε κορυφαία καταταγμένους φορτιστές με βάση τα επιλεγμένα κριτήρια. Επιπλέον, η προβλεπτική λειτουργικότητα του συστήματος επιτρέπει στο χρήστη να βλέπει την εκτιμώμενη ηλιακή απόδοση σε διάφορα χρονικά διαστήματα του επόμενου 24ώρου, καθιστώντας το σύστημα κατάλληλο για μελλοντικό προγραμματισμό. Επιπλέον, η υποδομή υποστηρίζει τη δημιουργία μικροδικτύων από διαχειριστές, οι οποίοι μπορούν να καταχωρήσουν νέους φορτιστές, να καθορίσουν γεωγραφική θέση και τεχνικά χαρακτηριστικά του φωτοβολταϊκού συστήματος, και να παρακολουθούν την απόδοσή τους στο χάρτη.

Η λειτουργικότητα του συστήματος αξιολογήθηκε μέσα από μια σειρά πειραμάτων που κάλυψαν κάθε επίπεδο της αρχιτεκτονικής του: το επίπεδο δεδομένων, το υπολογιστικό επίπεδο και τη διεπαφή χρήστη. Στο επίπεδο των δεδομένων, μετρήθηκε η αποδοτικότητα της βάσης SQLite, καθώς και η ταχύτητα ανάκτησης δεδομένων από τοπικό JSON αρχείο, επιβεβαιώνοντας την υπεροχή της SQLite σε πραγματικές συνθήκες ανάκτησης και επεξεργασίας. Στο υπολογιστικό επίπεδο, το πείραμα επικεντρώθηκε στον υπολογισμό του οικολογικού σκορ για κάθε φορτιστή, λαμβάνοντας υπόψη αποστάσεις, διαθεσιμότητα και

ποσοστά ηλιακής παραγωγής, τόσο σε πραγματικό χρόνο όσο και με προβλεπόμενα δεδομένα. Τα αποτελέσματα έδειξαν ότι όσο αυξάνεται η ακτίνα αναζήτησης (από 2 έως 75 χιλιόμετρα), αυξάνονται και οι πιθανότητες εντοπισμού πιο οικολογικά αποδοτικών φορτιστών, γεγονός που αποτυπώθηκε σε υψηλότερες συνολικές και μέσες τιμές Eco Score. Η ακτίνα αναζήτησης επηρεάζει επίσης τη διάρκεια εκτέλεσης των υπολογισμών, χωρίς όμως να προκαλεί μη αποδεκτές καθυστερήσεις. Στο επίπεδο της διεπαφής χρήστη, μελετήθηκαν οι χρόνοι απόκρισης της διεπαφής για διαφορετικές ενέργειες, όπως η εμφάνιση όλων των φορτιστών, η αλλαγή φίλτρων τύπου πρίζας και η εναλλαγή καρτελών. Οι χρόνοι αυτοί κυμάνθηκαν εντός αποδεκτών ορίων, με μεγαλύτερες καθυστερήσεις να παρατηρούνται σε περιπτώσεις όπου το πλήθος των οπτικών στοιχείων ήταν αυξημένο. Συνολικά, τα πειραματικά αποτελέσματα υποδεικνύουν ότι το σύστημα είναι τεχνικά λειτουργικό, υπολογιστικά αποδοτικό και ικανό να προσφέρει ποιοτικές οικολογικές συστάσεις στον χρήστη σε πραγματικό ή προβλεπόμενο χρόνο.

7.2 Μελλοντική εργασία

Η παρούσα υλοποίηση αποτελεί ένα πρώτο βήμα προς την κατεύθυνση της οικολογικά ευφυούς επιλογής φορτιστών. Υπάρχουν όμως περιθώρια για περαιτέρω εξέλιξη σε διάφορα επίπεδα. Αρχικά, η υπολογιστική λογική μπορεί να επεκταθεί ώστε να λαμβάνει υπόψη περισσότερες παραμέτρους, όπως τη συμφόρηση στους δρόμους (real-time traffic), την προβλεπόμενη διάρκεια φόρτισης ή την προτίμηση του χρήστη για συγκεκριμένα σημεία ενδιαφέροντος. Επιπλέον, το οικολογικό σκορ θα μπορούσε να ενισχυθεί με δεδομένα εκπομπών CO_2 ανά τύπο παραγωγής ενέργειας ανά μικροδίκτυο, αν υπάρχει τέτοια πληροφορία διαθέσιμη. Σε επίπεδο διεπαφής, μπορούν να ενσωματωθούν λειτουργίες όπως ειδοποιήσεις για μελλοντική διαθεσιμότητα φορτιστών ή προσωποποιημένες διαδρομές φόρτισης βάσει συνηθειών του χρήστη. Επίσης, θα μπορούσε να δημιουργηθεί native mobile εφαρμογή για Android και iOS για μεγαλύτερη φορητότητα. Τέλος, σε μεγαλύτερη κλίμακα, το σύστημα μπορεί να διασυνδεθεί με πραγματικούς παρόχους φορτιστών ή να ενσωματωθεί σε υπάρχουσες εφαρμογές έξυπνων πόλεων, παρέχοντας προτάσεις φόρτισης με έμφαση στην περιβαλλοντική βιωσιμότητα. Επιπρόσθετα, προτείνεται η υλοποίηση μηνιαίου report που να καταγράφει για κάθε φορτιστή το συνολικό ποσό ενέργειας που φορτίστηκε, ώστε να γίνεται ανασκόπηση της χρήσης. Θα ήταν επίσης χρήσιμο να προστεθεί θεσμοθέτηση περιόδων φόρτισης για ανάλυση της δραστηριότητας ανά χρονικό πλαίσιο (π.χ. ώρες αιχμής). Τέλος, θα εξεταστεί η επίδραση περιβαλλοντικών παραγόντων όπως η σκόνη και η θερμοκρασία στην

ηλιακή απόδοση. Τα δεδομένα αυτά μπορούν να αξιοποιηθούν για τη βελτίωση των μοντέλων πρόβλεψης και της εκτίμησης παραγωγής ανανεώσιμης ενέργειας ανά μικροδίκτυο.

Βιβλιογραφία

[1] International Energy Agency (IEA). Global EV Outlook 2023. https://www.iea.org/reports/global-ev-outlook-2023

[2] Ghosh, A., Goswami, A. (2021). EV Charging Station Recommendation Using Renewable Energy Forecast, IEEE Trans. Smart Grid. https://doi.org/10.1109/TSG.2021.3050782

[3] US Energy Information Administration (EIA). Annual Energy Outlook 2021. https://www.eia.gov/outlooks/aeo

[4] "A Framework for Continuous kNN Ranking of EV Chargers with Estimated Components", Soteris Constantinou, Constantinos Costa, Andreas Konstantinidis, Mohamed F. Mokbel, Demetrios Zeinalipour Yazti, "The 40th IEEE International Conference on Data Engineering" (ICDE '24), IEEE Computer Society, ISBN:, Pages: 13 pages, May 13 - May 17, 2024, Utrecht, Netherlands,2024.

[5] OpenWeatherMap. Retrieved from https://openweathermap.org/

[6] Esri & Leaflet. (2023). *Building Web Maps with Leaflet*. https://leafletjs.com

[7] European Commission. (2019). *The European Green Deal*. COM(2019) 640 final.

[8] A Better Routeplanner. Retrieved from https://abetterrouteplanner.com/

[9] Octopus Electroverse. Retrieved from https://electroverse.com/

[13] PlugShare. Retrieved from https://www.plugshare.com/

[14] IONITY. Retrieved from https://ionity.eu/

[15] IONITY. (n.d.). *Emobility and Sustainability*. Retrieved from https://www.ionity.eu/stories/emobility-and-sustainability

[16] ChargeMap. (n.d.). *Charging stations for electric cars*. Retrieved from https://chargemap.com

[17] Kumar, A., Chattopadhyay, S., & Kar, S. (2024). *Microgrid system for electric vehicle charging stations integrated with renewable energy sources*. Frontiers in Energy Research. https://www.frontiersin.org/articles/10.3389/fenrg.2024.1492243/full

[18] Singh, R., Dubey, A., & Jain, S. (2023). *Optimized Renewable Energy Management in EV Charging Microgrids Using V2G and Smart Forecasting*. IEEE Access. https://ieeexplore.ieee.org/document/10096888

[19] S. Constantinou, A. Konstantinidis, and D. Zeinalipour-Yazti, "Green planning systems for self consumption of renewable energy," IEEE Internet Computing, pp. 1–1, 2022.

[20] Zhang, Y., & Chen, X. (2024). *Smart algorithms for power prediction in smart EV charging stations*. ScienceDirect.

https://www.sciencedirect.com/science/article/pii/S2307187723003334

[21] Arévalo, P., Ochoa-Correa, D., & Villa-Ávila, E. (2024). Systematic Review of the Effective Integration of Storage Systems and Electric Vehicles in Microgrid Networks: Innovative Approaches for Energy Management. *Vehicles*, *6*(4), 2075-2105. https://doi.org/10.3390/vehicles6040102

[22] Liu, Y., & Wan, F. (2025). Integrated Optimization of Microgrids with Renewable Energy, Electric Vehicles, and Adaptive Demand Response for Sustainable and Efficient Energy Management. *Smart Grids and Sustainable Energy*. https://doi.org/10.1007/s40866-025-00257-1

[23] Pulkit Kumar, Harpreet Kaur Channi, Raman Kumar, Asha Rajiv, Bharti Kumari, Gurpartap Singh, Sehijpal Singh, Issa Farhan Dyab, Jasmina Lozanović,
A comprehensive review of vehicle-to-grid integration in electric vehicles (2024)
https://doi.org/10.1016/j.ecmx.2024.100864

[24] CHAdeMO Association. (n.d.). *CHAdeMO Protocol Overview*. Retrieved from https://www.chademo.com/

[25] CharIN. (n.d.). *ISO 15118 – The Communication Standard*. Retrieved from https://www.charinev.org/

[26] "EcoCharge: A Framework for Sustainable Electric Vehicles Charging", Soteris Constantinou, Dimitris Papazachariou, Constantinos Costa, Andreas Konstantinidis, Mohamed

F. Mokbel, Demetrios Zeinalipour Yazti, "The 25th IEEE International Conference on Mobile Data Management" (MDM '24), IEEE Computer Society, ISBN:, Pages: 4 pages, June 24 - June 27, 2024, Brussels, Belgium, 2024.

[27] S. Constantinou, A. Konstantinidis, P. K. Chrysanthis, and D. Zeinalipour-Yazti, “Green planning of IOT home automation workflows in smart buildings,” ACM Trans. Internet Things, Jun 2022.

[28] S. Constantinou, A. Konstantinidis, D. Zeinalipour-Yazti, and P. K. Chrysanthis, “The IOT meta-control firewall,” in 37th IEEE ICDE, April 19 - April 22, 2021, Chania, Crete, 2021, conference, p. 12 pages.

[29] S. Constantinou, N. Polycarpou, C. Costa, A. Konstantinidis, P. K. Chrysanthis, and D. Zeinalipour Yazti, “An iot data system for solar selfconsumption,” in 24th IEEE International Conference on Mobile Data Management (MDM), July 3- July 6, 2023, Singapore, 2023, conference, p. 10 pages.

[30] Y. Tao and D. Papadias, “Continuous nearest neighbor search,” The VLDB Journal, vol. 12, no. 3, pp. 275–292, 2003.

[31] G. Chatzimilioudis, C. Costa, D. Zeinalipour-Yazti, W. C. Lee, and E. Pitoura, “Distributed in-memory processing of all k nearest neighbor queries,” IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 4, pp. 925–938, 2016.

[32] SQLite. Retrieved from https://www.sqlite.org/

[33] Python 3. Retrieved from https://www.python.org/

[34] sqlite3 DB-API 2.0 interface for SQLite databases. Retrieved from https://docs.python.org/3/library/sqlite3.html

[35] GeoPy. Retrieved from https://geopy.readthedocs.io/en/stable/

[36] Flask. Retrieved from https://flask.palletsprojects.com/en/3.0.x/

[37] Flask-Session. Retrieved from https://flask-session.readthedocs.io/

[38] json. Retrieved from https://docs.python.org/3/library/json.html

[39] Python datetime. Retrieved from https://docs.python.org/3/library/datetime.html

[40] Python timezone. Retrieved from https://docs.python.org/3/library/datetime.html#timezone-objects

[41] Python timedelta. Retrieved from https://docs.python.org/3/library/datetime.html#timedelta-objects

[42] Python ZoneInfo. Retrieved from https://docs.python.org/3/library/zoneinfo.html

[43] bcrypt. Retrieved from https://pypi.org/project/bcrypt/

[44] scikit-learn. Retrieved from https://scikit-learn.org/stable/

[45] HTML Living Standard. Retrieved from https://html.spec.whatwg.org/

[46] Cascading Style Sheets (CSS). Retrieved from https://www.w3.org/Style/CSS/

[47] JavaScript. Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript

[48] Leaflet. Retrieved from https://leafletjs.com/

[49] Leaflet Routing Machine. Retrieved from https://www.liedman.net/leaflet-routing-machine/

[50] Leaflet.Locate. Retrieved from https://github.com/domoritz/leaflet-locatecontrol

[51] Bootstrap. Retrieved from https://getbootstrap.com/

[52] jQuery. Retrieved from https://jquery.com/

[53] OpenTopoMap. Retrieved from https://opentopomap.org/

[54] Esri World Street Map. Retrieved from https://www.arcgis.com/home/item.html?id=10df2279f9684e4a9f6a7f08febac2a9

[55] CARTO Maps. Retrieved from https://carto.com/

[56] Plotly.js. Retrieved from https://plotly.com/javascript/

[57] OpenStreetMap. Retrieved from https://www.openstreetmap.org/