ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Εφαρμογή για Σταθμούς Φόρτισης Ηλεκτρικών Οχημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μάρκος Ν. Μπαράτσας

Επιβλέπων : Εμμανουήλ Βαρβαρίγος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Εφαρμογή για Σταθμούς Φόρτισης Ηλεκτρικών Οχημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

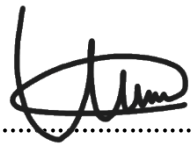
Μάρκος Ν. Μπαράτσας

Επιβλέπων : Εμμανουήλ Βαρβαρίγος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 8η Ιουνίου 2022.

.....	.....	.....
Εμμανουήλ Βαρβαρίγος	Θεοδώρα Βαρβαρίγου	Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2022



.....

Μάρκος Ν. Μπαράτσας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η ενεργειακή κρίση και τα περιβαλλοντικά προβλήματα έχουν ενθαρρύνει την υιοθέτηση ηλεκτρικών οχημάτων ως εναλλακτικό τρόπο μεταφοράς για την αντικατάσταση των συμβατικών οχημάτων εσωτερικής καύσης. Απαραίτητη για αυτήν τη μετάβαση, ωστόσο, είναι η εξόπλιση του συστήματος μεταφορών με σταθμούς φόρτισης ηλεκτρικών οχημάτων, έτσι ώστε να παρέχεται η δυνατότητα στα ηλεκτρικά οχήματα να κινούνται χωρίς περιορισμούς. Εκτός αυτού, είναι σημαντική η δημιουργία λογισμικού που να υλοποιεί έξυπνες λειτουργικότητες για την εξυπηρέτηση των αναγκών των σταθμών φόρτισης. Με βάση, λοιπόν, τα παραπάνω στο πλαίσιο αυτής της διπλωματικής υλοποιήθηκε μία web εφαρμογή για τους χειριστές (operators) των σταθμών φόρτισης με έμφαση στην χρήση στρατηγικών τιμολόγησης για την παραγωγή δυναμικών τιμών. Συγκεκριμένα, οι χειριστές μπορούν να διαλέξουν τη στρατηγική τιμολόγησης που εξυπηρετεί τα κίνητρα τους, έτσι ώστε να παράγονται δυναμικά οι τιμές πώλησης της ηλεκτρικής ενέργειας (€/KWh) με βάση διάφορα κριτήρια. Επίσης, σε αυτήν την εφαρμογή οι χειριστές μπορούν να δουν συγκεντρωτικά την κατάσταση του κάθε σταθμού, δηλαδή πληροφορίες για την πληρότητα του σταθμού, την κατάσταση των φορτιστών, τις κρατήσεις φόρτισης από τους ιδιοκτήτες οχημάτων και την κατάσταση των οχημάτων που φορτίζουν τη δεδομένη χρονική στιγμή. Παράλληλα, παράγονται στατιστικά για όλους τους σταθμούς, τα οποία οι χειριστές μπορούν να δουν εποπτικά, ώστε να λαμβάνουν εξυπνότερες αποφάσεις. Παρόλο που η εφαρμογή επικεντρώνεται στην εξυπηρέτηση των αναγκών των χειριστών του σταθμού, δημιουργήθηκε επιπλέον και λειτουργικότητα για τους ιδιοκτήτες των ηλεκτρικών οχημάτων, οι οποίοι μπορούν μέσω της εφαρμογής να δημιουργήσουν εύκολα κρατήσεις φόρτισης προς τους σταθμούς της επιλογής τους. Σε αυτήν τη διπλωματική, αρχικά αναλύονται κάποιες εισαγωγικές έννοιες όπως η αγορά ηλεκτρικής ενέργειας, τα ηλεκτρικά αυτοκίνητα και οι σταθμοί φόρτισης ηλεκτρικών οχημάτων. Στη συνέχεια, αναλύονται τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση αυτής της web εφαρμογής. Τα επόμενα κεφάλαια αφορούν το σχεδιασμό και την υλοποίηση της εφαρμογής, δηλαδή τη διαδικασία που εφαρμόστηκε ώστε η εφαρμογή από ιδέα να γίνει πραγματικότητα. Τέλος, το τελευταίο κεφάλαιο παρουσιάζει κάποια παραδείγματα χρήσης της web εφαρμογής, ώστε να φανεί στο σύνολο η λειτουργικότητά της.

Λέξεις-κλειδιά: αγορά ηλεκτρικής ενέργειας, ηλεκτρικά οχήματα, σταθμοί φόρτισης ηλεκτρικών οχημάτων, σταθμοί φόρτισης με έξυπνη λειτουργικότητα, δυναμική τιμολόγηση, web εφαρμογή για τους χειριστές σταθμών φόρτισης

Abstract

Energy crisis and environmental issues have encouraged the adoption of electric vehicles as an alternative transportation option to the conventional vehicles with internal combustion engines. However, to make this transition feasible, the transportation system should be properly equipped with charging stations, so that electric vehicles are able to travel without any restrictions. Apart from that, it is also important that charging stations be equipped with software that includes several smart capabilities, so that their needs are accommodated. Taking that into consideration, a web application for the charging stations' operators was implemented as part of this thesis, which focuses on the utilization of pricing strategies to dynamically produce the stations' energy prices. Specifically, charging station operators can choose the pricing strategy that better serves their motives, so that the energy prices (€/KWh) are dynamically produced based on several criteria. In addition, operators are able to easily view the state of each station (i.e. station occupancy, chargers' health status, charging reservations from EV owners, live status of charging vehicles). Meanwhile, statistics for all charging stations are automatically produced, which operators can view, so that they are able to make smart decisions. Despite the fact that this web application focuses on addressing the needs of station operators, it also includes functionality for the EV owners, who can use the app to easily make charging reservations to the stations of their choosing. This thesis initially analyzes some introductory concepts, such as the electricity market, electric vehicles and charging stations. Consequently, it analyzes the tools used for the implementation of this web application. The next chapters focus on the design and the actual implementation of the application; that is the steps followed to make the idea of this app come into reality. Finally, the last chapter focuses on presenting some use cases, so that the functionalities of the app are clearly projected.

Keywords: electricity market, electric vehicles, charging stations, charging station with smart charging capabilities, dynamic pricing, web application for charging station operators

Ευχαριστίες

Η διπλωματική αυτή εργασία σηματοδοτεί την ολοκλήρωση ενός σημαντικού κεφαλαίου στην ακαδημαϊκή και προσωπική μου πορεία. Επομένως, σε αυτό το σημείο θα ήθελα να ευχαριστήσω όλους τους ανθρώπους που με βοήθησαν να καταφέρω να φτάσω επιτυχώς μέχρι εδώ.

Αρχικά, θα ήθελα να ευχαριστήσω τον κύριο Βαρβαρίγο που ήταν ο επιβλέπων καθηγητής σε αυτήν τη διπλωματική και με καθοδήγησε από την αρχή μέχρι και το τέλος της. Παρόλο που τον Σεπτέμβριο του 2021 είχαμε συζητήσει σχετικά με την επιλογή ενός άλλου θέματος για τη διπλωματική μου, ο κύριος Βαρβαρίγος μου πρότεινε ως πιθανή εναλλακτική το θέμα που τελικά επέλεξα, το οποίο μου φάνηκε ιδιαίτερα ενδιαφέρον και είμαι βέβαιος ότι ήταν το κατάλληλο θέμα για να κάνω τη διπλωματική μου. Επιπλέον, θα ήθελα να ευχαριστήσω τον κύριο Αριστοτέλη Κρέτση και τον κύριο Γιώργο Τσαούσογλου, οι οποίοι ήταν πρόθυμοι να με βοηθήσουν σε οποιαδήποτε απορία είχα.

Ακόμα, θέλω να ευχαριστήσω όλους τους φίλους μου για την βοήθεια και τη συμπαράσταση κατά τη διάρκεια των 5 αυτών χρόνων, με τους οποίους περάσαμε πολύ ωραίες στιγμές.

Πάνω από όλα, θα ήθελα να ευχαριστήσω την οικογένεια μου. Αρχικά, θα ήθελα να ευχαριστήσω ιδιαίτερα τον αδερφό μου, Αλέξη, ο οποίος καθόλη τη διάρκεια της ακαδημαϊκής μου πορείας με συμβούλευε κατάλληλα ώστε να πετυχαίνω τους στόχους μου. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, Νίκο και Αριστέα, και την αδερφή μου, Μαριτίνα, για την υποστήριξη που μου προσέφεραν όλα αυτά τα χρόνια.

Περιεχόμενα

Περίληψη **5**
Abstract **7**
Ευχαριστίες **9**
Περιεχόμενα **11**
Κατάλογος εικόνων **13**
Δομή της διπλωματικής **15**
Κεφάλαιο 1: Εισαγωγικές έννοιες **17**
Αγορά ηλεκτρικής ενέργειας 18
Ηλεκτρικά οχήματα 20
Εισαγωγή *20*
Τύποι ηλεκτρικών οχημάτων *20*
Πλεονεκτήματα και προκλήσεις ηλεκτρικών οχημάτων *21*
Μπαταρίες ηλεκτρικών οχημάτων *22*
Σταθμοί Φόρτισης Ηλεκτρικών Οχημάτων 23
Εισαγωγή *23*
Vehicle-To-Grid *23*
Σταθμοί φόρτισης με δυνατότητα έξυπνης φόρτισης *24*
Προκλήσεις και μελλοντική έρευνα *24*
Κεφάλαιο 2: Εργαλεία που χρησιμοποιήθηκαν **27**
React JS Library 28
Python Django 29
Django REST Framework 30
Python Celery & Django Celery Beat 30
Swagger: API Documentation 31
Κεφάλαιο 3: Σχεδιασμός της εφαρμογής **33**
Στρατηγικές τιμολόγησης ενέργειας ανά KWh 34
Οργάνωση των φορτιστών *34*
Μέθοδοι τιμολόγησης ενέργειας *34*
Μέθοδος σταθερής τιμής (Fixed Price) *34*
Μέθοδος σταθερού κέρδους (Fixed Profit) *35*
Μέθοδος διαμόρφωσης τιμής αναλόγως της ζήτησης (Demand-centered Profit) *35*
Μέθοδος διαμόρφωσης τιμής αναλόγως των ανταγωνιστών (Competitor-centered Profit) *36*
Υπολογισμός κόστους φόρτισης 37
Υπολογισμός κόστους ηλεκτρικής ενέργειας από το δίκτυο 38
Αρχιτεκτονική του συστήματος 38
Σχεδιασμός ER διαγράμματος για το Back End 41
ER διάγραμμα για το users app *41*
ER διάγραμμα για το gridprice app *41*
ER διάγραμμα για το stations app *42*
ER διάγραμμα για το chargers app *43*

ER διάγραμμα για το reservations app ... *44*
Συγκεντρωτικό ER διάγραμμα ... *45*
Σχεδιασμός των API Endpoints του Back End ... 47
API endpoints για το users app ... *47*
API endpoints για το gridprice app ... *47*
API endpoints για το stations app ... *48*
API endpoints για το chargers app ... *48*
API endpoints για το reservations app ... *49*
API endpoints για το stats app ... *51*
Mockups για το Front End Environment ... 52
Κοινά Mockups ... *52*
Charging Operator Frontend Mockups ... *54*
EV Owner Frontend Mockups ... *60*
Κεφάλαιο 4: Υλοποίηση της εφαρμογής ... 63
Εισαγωγή ... 64
Υλοποίηση του Back End ... 64
Υλοποίηση του Grid Price Collector ... 66
Back End Testing ... 67
Documentation για τα Backend Endpoints ... 67
Υλοποίηση του Front End ... 69
Κεφάλαιο 5: Παραδείγματα χρήσης του Front End ... 71
Εισαγωγή ... 72
User Registration and Login ... 72
Charging Operator Frontend: Station Creation ... 76
Charging Operator Frontend: Station Add ... 81
Charging Operator Frontend: Station Overview ... 82
Charging Operator Frontend: Station Prices and Chargers ... 83
Charging Operator Frontend: Station Reservations ... 86
Charging Operator Frontend: Station Parking ... 90
Charging Operator Frontend: Stations Statistics ... 90
EV Owner Frontend: Dashboard ... 92
EV Owner Frontend: Create Reservation ... 93
Κεφάλαιο 6: Συμπεράσματα ... 95
Συμπεράσματα ... 96
Επόμενα βήματα ... 96
Πηγαίος κώδικας ... 97
Βιβλιογραφία ... 98

Κατάλογος εικόνων

Εικόνα 1: Αγορά ηλεκτρικής ενέργειας με μοντέλο μονοπωλίου (εικόνα από Hunt and Shuttleworth, 1996) 19
Εικόνα 2: Αγορά ηλεκτρικής ενέργειας με μοντέλο πλήρους ανταγωνισμού (εικόνα από Hunt and Shuttleworth, 1996) 19
Εικόνα 3: Δομή ενός django project 30
Εικόνα 4: Υπολογισμός ολικού κόστους φόρτισης 37
Εικόνα 5: Διάγραμμα αρχιτεκτονικής του συστήματος 39
Εικόνα 6: ER diagram για το users app 41
Εικόνα 7: ER diagram για το gridprice app 42
Εικόνα 8: ER diagram για το stations app 42
Εικόνα 9: ER diagram για το chargers app 43
Εικόνα 10: ER diagram για το reservations app 45
Εικόνα 11: Συγκεντρωτικό ER διάγραμμα για όλα τα apps του backend 46
Εικόνα 12: Mock-up για τη σελίδα εγγραφής 53
Εικόνα 13: Mock-up για τη σελίδα σύνδεσης 53
Εικόνα 14: Mock-up για τη σελίδα Dashboard των operators 54
Εικόνα 15: Mock-up για τη σελίδα δημιουργίας νέου σταθμού (βήμα 1) 55
Εικόνα 16: Mock-up για τη σελίδα δημιουργίας νέου σταθμού (βήμα 2) 55
Εικόνα 17: Mock-up για τη σελίδα δημιουργίας νέου σταθμού (βήμα 3) 55
Εικόνα 18: Mock-up για το Modal καθορισμού παραμέτρων των μεθόδων τιμολόγησης 56
Εικόνα 19: Mock-up για τη σελίδα προσθήκης ενός υπάρχοντος σταθμού 56
Εικόνα 20: Mock-up για τη σελίδα επισκόπησης ενός σταθμού 57
Εικόνα 21: Mock-up για τη σελίδα διαχείρισης Pricing Groups ενός σταθμού 58
Εικόνα 22: Mock-up για τη σελίδα διαχείρισης φορτιστών ενός σταθμού 58
Εικόνα 23: Mock-up για τη σελίδα κρατήσεων ενός σταθμού 59
Εικόνα 24: Mock-up για τη σελίδα Parking ενός σταθμού 59
Εικόνα 25: Mock-up για τη σελίδα κατάστασης ενός οχήματος που φορτίζει 60
Εικόνα 26: Mock-up για τη σελίδα Dashboard ενός ιδιοκτήτη οχήματος 60
Εικόνα 27: Mock-up για τη σελίδα δημιουργίας κράτησης από ιδιοκτήτη οχήματος 61
Εικόνα 28: Δομή του django project του backend 64
Εικόνα 29: Δομή ενός django application του backend 65
Εικόνα 30: Σελίδα Swagger Documentation για τα API endpoints του backend 68
Εικόνα 31: Συνάρτηση get_stations από το views.py του stations app 68
Εικόνα 32: Swagger Documentation για το endpoint της συνάρτησης get_stations 69
Εικόνα 33: Δομή του Frontend React Application 70
Εικόνα 34: Σελίδα εγγραφής χρήστη 72
Εικόνα 35: Βοηθητικές ενδείξεις κατά την πληκτρολόγηση (αριστερά), εμφάνιση σφάλματος (δεξιά) 73

Εικόνα 36: Μήνυμα επιτυχούς εγγραφής χρήστη 74
Εικόνα 37: Σελίδα σύνδεσης χρήστη. 74
Εικόνα 38: Μήνυμα σφάλματος στη σελίδα σύνδεσης χρήστη 75
Εικόνα 39: Σελίδα Dashboard ενός operator 75
Εικόνα 40: Σελίδα Dashboard ενός ιδιοκτήτη ηλεκτρικού οχήματος 76
Εικόνα 41: Σελίδα δημιουργία σταθμού (βήμα 1/4) 77
Εικόνα 42: Σελίδα δημιουργία σταθμού (βήμα 2/4) 77
Εικόνα 43: Modal τροποποίησης/διαγραφής φορτιστή 78
Εικόνα 44: Σελίδα δημιουργία σταθμού (βήμα 3/4) 78
Εικόνα 45: Modal για την επιλογή μεθόδου και παραμέτρων τιμολόγησης 79
Εικόνα 46: Σελίδα δημιουργία σταθμού (βήμα 4/4) 80
Εικόνα 47: Σελίδα προσθήκης υπάρχοντος σταθμού 81
Εικόνα 48: Σελίδα ρυθμίσεων που περιλαμβάνει τα StationRequests 82
Εικόνα 49: Σελίδα επισκόπησης ενός σταθμού 83
Εικόνα 50: Σελίδα διαχείρισης των Pricing Groups ενός σταθμού 84
Εικόνα 51: Σελίδα διαχείρισης φορτιστών ενός σταθμού 85
Εικόνα 52: Σελίδα διαχείρισης κρατήσεων ενός σταθμού 86
Εικόνα 53: Modal δημιουργίας κράτησης 87
Εικόνα 54: Δημιουργία δύο κρατήσεων στην ίδια χρονική περίοδο 87
Εικόνα 55: Modal για την επιβεβαίωση άφιξης ενός οχήματος 88
Εικόνα 56: Modal για τον τερματισμό μίας κράτησης 88
Εικόνα 57: Modal για την παρουσίαση των πληροφοριών μίας κράτησης 89
Εικόνα 58: Σελίδα διαχείρισης κόστους parking ενός σταθμού 90
Εικόνα 59: Σελίδα παρουσίασης στατιστικών 91
Εικόνα 60: Modal για την τροποποίηση των στοιχείων ενός ιδιοκτήτη οχήματος 92
Εικόνα 61: Modal για τη δημιουργία ενός νέου οχήματος 92
Εικόνα 62: Σελίδα για τη δημιουργία κράτησης ενός ιδιοκτήτη οχήματος 93

Δομή της διπλωματικής

Η δομή αυτής της διπλωματικής περιγράφεται παρακάτω:

- Κεφάλαιο 1: Εισαγωγικές Έννοιες. Στο κεφάλαιο αυτό αναλύονται κάποιες εισαγωγικές έννοιες σχετικά με (α) την αγορά ηλεκτρικής ενέργειας, (β) τα ηλεκτρικά αυτοκίνητα, και (γ) του σταθμούς φόρτισης ηλεκτρικών αυτοκινήτων. Στόχος αυτού του κεφαλαίου είναι να κατανοήσει ο αναγνώστης βασικές έννοιες, που είναι απαραίτητες για την κατανόηση της χρησιμότητας της εφαρμογής που υλοποιήθηκε στο πλαίσιο αυτής της διπλωματικής.
- Κεφάλαιο 2: Εργαλεία που χρησιμοποιήθηκαν. Στο κεφάλαιο αυτό περιγράφονται τα εξής εργαλεία: (α) React JS Library, (β) Python Django, (γ) Django REST Framework, (δ) Python Celery & Django Celery Beat, και (ε) Swagger. Στόχος αυτού του κεφαλαίου είναι η περιγραφή των εργαλείων που χρησιμοποιήθηκαν, ώστε ο αναγνώστης να καταλάβει τη χρησιμότητα του κάθε εργαλείου στην υλοποίηση του συστήματος.
- Κεφάλαιο 3: Σχεδιασμός της εφαρμογής. Αρχικά, στο κεφάλαιο αυτό περιγράφονται κάποια ιδιαίτερα χαρακτηριστικά που έπρεπε να χειριστεί η εφαρμογή, τα οποία έπρεπε να ληφθούν υπόψιν στο σχεδιασμό. Τα χαρακτηριστικά αυτά σχετίζονταν με τις στρατηγικές τιμολόγησης και τον υπολογισμό του κόστους φόρτισης. Η ανάλυση αυτών ήταν απαραίτητη να προηγηθεί του σχεδιασμού, αφού παραδείγματος χάριν το database σχεδιάστηκε με τέτοιον τρόπο ώστε να γίνεται σωστός χειρισμός αυτών των δεδομένων. Ακολούθως, παρουσιάζεται η αρχιτεκτονική του συστήματος, ο σχεδιασμός του ER διαγράμματος για το Back End, ο σχεδιασμός των API endpoints του Back End, και, τέλος, παρουσιάζονται κάποια mockups για το Front End. Στόχος αυτού του κεφαλαίου είναι να περιγραφεί αναλυτικά ο σχεδιασμός του συστήματος, ώστε να κατανοήσει ο αναγνώστης την αρχιτεκτονική του συστήματος.
- Κεφάλαιο 4: Υλοποίηση της εφαρμογής. Το κεφάλαιο αυτό περιγράφει την διαδικασία της υλοποίησης των επιμέρους στοιχείων του συστήματος, αναλύοντας ποια από τα εργαλεία χρησιμοποιήθηκαν για κάθε στοιχείο. Επίσης, γίνεται αναφορά στο Testing και στο Documentation που γράφηκαν για το σύστημα. Στόχος αυτού του κεφαλαίου, λοιπόν, είναι να αναλυθεί ο τρόπος με τον οποίο υλοποιήθηκε το σύστημα βάσει του σχεδιασμού που είχε προηγηθεί.
- Κεφάλαιο 5: Παραδείγματα Χρήσης του Front End. Το κεφάλαιο αυτό περιλαμβάνει δέκα παραδείγματα χρήσης της εφαρμογής. Ουσιαστικά, σε αυτό το κεφάλαιο φαίνεται αναλυτικά το Front End της εφαρμογής, μέσω στιγμιότυπων που λήφθηκαν. Στόχος του κεφαλαίου είναι να αναδειχθεί στον αναγνώστη πως αυτή η web εφαρμογή χρησιμοποιείται τόσο από τους χειριστές (operators) των σταθμών, όσο και από τους ιδιοκτήτες ηλεκτρικών οχημάτων.
- Κεφάλαιο 6: Συμπεράσματα. Το κεφάλαιο αυτό περιλαμβάνει συμπεράσματα, επόμενα βήματα καθώς και πληροφορίες για τον πηγαίο κώδικα του συστήματος που υλοποιήθηκε.

Κεφάλαιο 1: Εισαγωγικές έννοιες

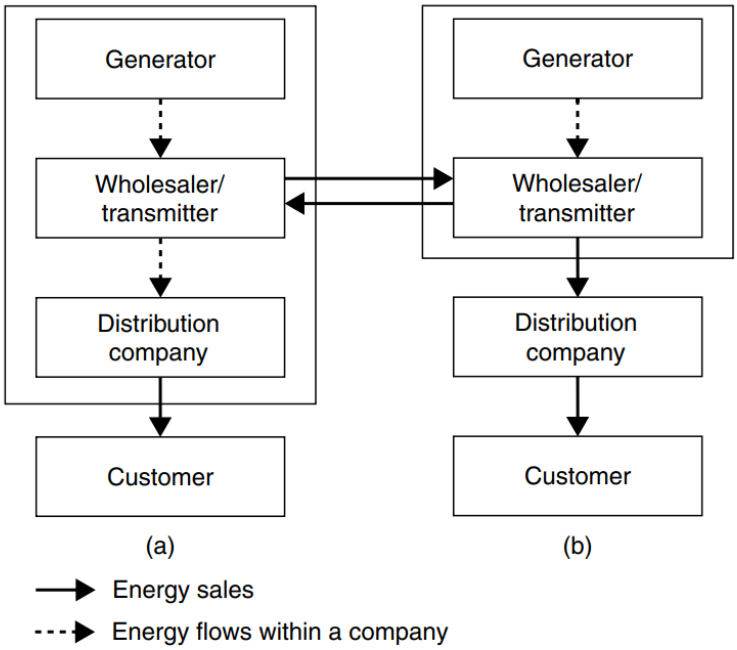
Αγορά ηλεκτρικής ενέργειας

Στο παρελθόν όταν οι καταναλωτές ήθελαν να αγοράσουν ηλεκτρική ενέργεια, η μόνη επιλογή που είχαν ήταν να την αγοράσουν από την υπηρεσία που κατείχε το μονοπώλιο για την παροχή της ηλεκτρικής ενέργειας, η οποία ενδεχομένως ήταν διαφορετική αναλόγως της περιοχής που έμεναν. Σε μερικές περιπτώσεις, οι υπηρεσίες αυτές ήταν υπεύθυνες για την παραγωγή, μετάδοση και διανομή της ηλεκτρικής ενέργειας. Σε άλλες περιπτώσεις, οι υπηρεσίες αυτές ήταν υπεύθυνες μόνο για την πώληση και τη διανομή της ηλεκτρικής ενέργειας, οι οποίες και πάλι θα αγόραζαν την ηλεκτρική ενέργεια από την υπηρεσία παραγωγής και μεταφοράς ηλεκτρικής ενέργειας που κατείχε το μονοπώλιο σε μία ευρύτερη γεωγραφική περιοχή. Οι υπηρεσίες αυτές ήταν άλλοτε ιδιωτικές εταιρίες, ενώ άλλοτε ήταν δημόσιες εταιρίες ή κρατικές υπηρεσίες. Ανεξαρτήτως της ιδιοκτησίας, ωστόσο, το χαρακτηριστικό σε όλους αυτούς τους φορείς ήταν ότι κατείχαν μονοπώλια σε συγκεκριμένες γεωγραφικές περιοχές. [1]

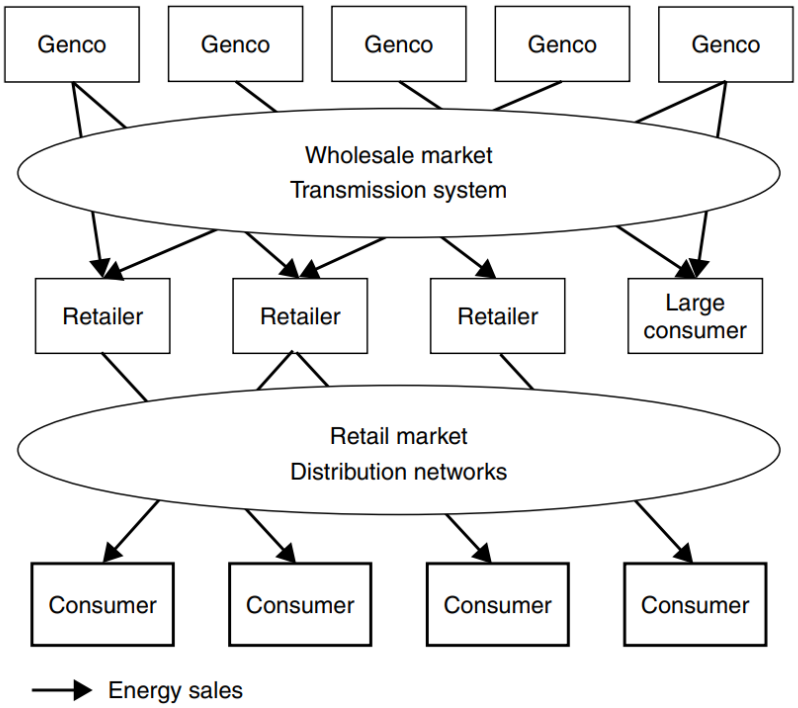
Στα τέλη του εικοστού αιώνα, κάποιοι οικονομολόγοι άρχισαν να υποστηρίζουν ότι το μοντέλο αυτό έπρεπε να αλλάξει, ισχυριζόμενοι ότι η κατοχή μονοπωλίου από αυτούς τους φορείς δεν παρείχε κίνητρο ώστε να λειτουργούν αποδοτικά και να αποφεύγουν αχρείαστες επενδύσεις. Εκτός αυτών, οι οικονομολόγοι υποστήριζαν ότι κάθε λάθος που έκαναν οι φορείς κατέληγε να επιβαρύνει τους καταναλωτές. Αντιθέτως, οι οικονομολόγοι αυτοί πίστευαν ότι εάν εδραιωνόταν μία ανταγωνιστική αγορά, η τιμή της ηλεκτρικής ενέργειας θα μειώνονταν και η κοινωνία στο σύνολό της θα επωφελούνταν. Αυτό επειδή εάν υπάρχουν εταιρίες που μπορούν να ανταγωνίζονται μεταξύ τους για την προώθηση των υπηρεσιών σχετικές με την ηλεκτρική ενέργεια, η αποδοτικότητα που προκύπτει από αυτήν την ανταγωνιστικότητα θα ωφελούσε τελικά τους καταναλωτές. [1]

Ωστόσο, η ηλεκτρική ενέργεια έχει κάποια ιδιαίτερα χαρακτηριστικά σε σύγκριση με άλλα απλά αγαθά, όπως παραδείγματος χάριν είναι τα τρόφιμα ή ακόμα και τα καύσιμα. Μία θεμελιώδης διαφορά, λοιπόν, μεταξύ της ηλεκτρικής ενέργειας και άλλων απλών αγαθών είναι ότι η ηλεκτρική ενέργεια είναι αναπόφευκτα συνδεδεμένη με ένα σύστημα που λειτουργεί πολύ πιο γρήγορα από κάθε άλλη αγορά. Σε αυτό το σύστημα, η προσφορά και η ζήτηση (η παραγωγή και η κατανάλωση) πρέπει να ισορροπούν κάθε δευτερόλεπτο. Αν αυτή η ισορροπία δεν διατηρηθεί, το σύστημα καταρρέει με καταστροφικές συνέπειες. Εκτός αυτού, μία άλλη επίσης σημαντική διαφορά είναι ότι η ενέργεια που παράγεται από μία γεννήτρια δεν μπορεί να παραδοθεί με αποκλειστικό τρόπο σε έναν καταναλωτή και ο καταναλωτής δεν μπορεί να καταναλώσει αποκλειστικά ηλεκτρική ενέργεια από μία συγκεκριμένη γεννήτρια. Τέλος, ένα τελευταίο χαρακτηριστικό της ηλεκτρικής ενέργειας ως προϊόντος είναι ότι παρουσιάζει ημερησία και εβδομαδιαία κυκλικότητα που είναι εύκολα προβλέψιμη. [1][3]

Λαμβάνοντας υπόψιν όλα τα παραπάνω γίνεται κατανοητό ότι η σχεδίαση μιας ανταγωνιστικής αγοράς για την ηλεκτρική ενέργεια δεν ήταν μία εύκολη διαδικασία, ή τουλάχιστον τόσο εύκολη όσο για αγορές άλλων υλικών αγαθών. Για να μεταβεί η αγορά ηλεκτρικής ενέργειας από μία αγορά μονοπωλίου (εικόνα 1) σε μία αγορά βασισμένη στον πλήρη ανταγωνισμό (εικόνα 2), χρειάστηκαν αρκετά χρόνια καθώς έπρεπε να γίνει με σωστό τρόπο ο σχεδιασμός του μοντέλου της αγοράς και η νομοθέτηση αυτής, ώστε οι επιχειρηματικές δραστηριότητες εντός της αγοράς να είναι όσο το δυνατόν πιο αποδοτικές και δίκαιες για όλα τα συμβαλλόμενα μέρη. Αυτή η εξέλιξη από μονοπωλιακά μοντέλα σε πλήρως ανταγωνιστικά μοντέλα έγινε με στόχο την τεχνολογική πρόοδο και τη μείωση κόστους της ηλεκτρικής ενέργειας για τον τελικό καταναλωτή.



Εικόνα 1: Αγορά ηλεκτρικής ενέργειας με μοντέλο μονοπωλίου (εικόνα από [3])



Εικόνα 2: Αγορά ηλεκτρικής ενέργειας με μοντέλο πλήρους ανταγωνισμού (εικόνα από [3])

Ηλεκτρικά οχήματα

Εισαγωγή

Τα ηλεκτρικά αυτοκίνητα, ή αλλιώς Electric Vehicles (EVs), έχουν αρχίσει να γίνονται ιδιαίτερα δημοφιλή τα τελευταία χρόνια λόγω ποικίλων παραγόντων. Μεταξύ αυτών είναι η εξέλιξη της τεχνολογίας, η μείωση της τιμής αγοράς, καθώς και η ευαισθητοποίηση σχετικά με το περιβάλλον και την κλιματική αλλαγή [2]. Στο υποκεφάλαιο αυτό θα γίνει μία σύντομη περιγραφή κάποιων βασικών εννοιών σχετικές με τα ηλεκτρικά αυτοκίνητα. Αρχικά, θα γίνει μία αναφορά στους διαφορετικούς τύπους των ηλεκτρικών οχημάτων. Στη συνέχεια, θα γίνει μία αναφορά των πλεονεκτημάτων που προκύπτουν από τη χρήση ηλεκτρικών οχημάτων και των προκλήσεων που τα ηλεκτρικά αυτοκίνητα καλούνται να αντιμετωπίσουν. Τέλος, λόγω του γεγονότος ότι η μπαταρία ενός αυτοκινήτου είναι ένα σημαντικό θέμα για τα ηλεκτρικά αυτοκίνητα, θα γίνει μία σύντομη περιγραφή κάποιων χαρακτηριστικών που σχετίζονται με τις μπαταρίες ηλεκτρικών οχημάτων.

Τύποι ηλεκτρικών οχημάτων

Οι διαφορετικοί τύποι ηλεκτρικών οχημάτων κατηγοριοποιούνται με βάση την τεχνολογία του κινητήρα. Έτσι, τα ηλεκτρικά οχήματα χωρίζονται στις εξής κατηγορίες, με βάση το πως περιγράφεται στο [2]:

- Battery Electric Vehicles (BEVs): Τα οχήματα αυτά κινούνται καταναλώνοντας αποκλειστικά ηλεκτρική ενέργεια. Συγκεκριμένα, δεν έχουν κάποια μηχανή εσωτερικής καύσης, και δεν χρησιμοποιούν κανενός είδους υγρό καύσιμο. Για να απεκτήσουν επαρκή αυτονομία τα οχήματα αυτά συνήθως χρησιμοποιούν μεγάλα σετ μπαταριών. Μία τυπική μπαταρία ενός BEV οχήματος είναι από 25 έως και 100 KWh.
- Plug-In Hybrid Electric Vehicles (PHEVs): Τα plug-in υβριδικά αυτοκίνητα κινούνται χρησιμοποιώντας κινητήρα εσωτερικής καύσης καθώς και ηλεκτρικό κινητήρα που φορτίζει συνδεόμενος με κάποια εξωτερική ηλεκτρική πηγή. Τα plug-in υβριδικά μπορούν να αποθηκεύσουν επαρκή ηλεκτρική ενέργεια για να μειώσουν σημαντικά την κατανάλωση τους υπό κανονικές συνθήκες οδήγησης. Τυπικά διαθέτουν μπαταρία που συνήθως τους δίνει μία αυτονομία κοντά στα 50 χλμ. Μία τυπική μπαταρία ενός plug-in υβριδικού οχήματος είναι 10 KWh.
- Hybrid Electric Vehicles (HEVs): Τα υβριδικά αυτοκίνητα, ομοίως με τα plug-in υβριδικά, κινούνται χρησιμοποιώντας κινητήρα εσωτερικής καύσης και ηλεκτρικό κινητήρα. Η διαφορά τους με τα plug-in υβριδικά είναι ότι δεν μπορούν να συνδεθούν σε εξωτερική ηλεκτρική πηγή. Έτσι, η μπαταρία που παρέχει ενέργεια στον ηλεκτρικό κινητήρα φορτίζεται μέσω της ισχύος που παράγεται από τον κινητήρα εσωτερικής καύσης. Μία τυπική μπαταρία ενός υβριδικού οχήματος είναι 2 KWh.
- Fuel Cell Electric Vehicles (FCEVs): Τα οχήματα αυτά κινούνται μέσω ηλεκτρικού κινητήρα που χρησιμοποιεί ένα συνδυασμό συμπιεσμένου υδρογόνου και οξυγόνου που προέρχεται από τον αέρα. Έτσι, τα οχήματα αυτά παράγουν μόνο νερό ως απόβλητο από την διαδικασία αυτή. Παρόλο που τα οχήματα αυτά παρουσιάζονται ότι έχουν μηδενικές εκπομπές, αξίζει να σημειωθεί ότι το υδρογόνο που καταναλώνουν μπορεί να έχει εξαχθεί από φυσικό αέριο.
- Extended-range EVs (ER-EVs): Τα οχήματα αυτά είναι αρκετά παρόμοια με τα BEV που παρουσιάστηκαν παραπάνω. Ωστόσο, τα ER-EVs περιέχουν έναν επιπλέον κινητήρα εσωτερικής καύσης ο οποίος φορτίζει τις μπαταρίες αν αυτό απαιτείται. Ο κινητήρας αυτός, σε αντίθεση με αυτόν των PHEVs και HEVs, χρησιμοποιείται μόνο για φόρτιση της μπαταρίας και δεν χρησιμοποιείται για την κίνηση των τροχών του αυτοκινήτου.

Πλεονεκτήματα και προκλήσεις ηλεκτρικών οχημάτων

Το συμβατικό σύστημα μεταφοράς θεμελιώνεται πάνω από την τεχνολογία μηχανών καύσης ορυκτών καυσίμων για την μετατροπή χημικής ενέργειας σε κινητική. Σύμφωνα με δεδομένα της Ευρωπαϊκής Ένωσης ο κλάδος των μεταφορών είναι υπεύθυνος για το 28% των συνολικών εκπομπών διοξειδίου του άνθρακα, με το 70% αυτών να προκύπτει από τις οδικές μεταφορές. Ταυτοχρόνως, η κλιματική αλλαγή είναι ένα θέμα που απασχολεί την παγκόσμια κοινότητα και γίνεται προσπάθεια για μετάβαση προς πιο βιώσιμες λύσεις. Υπό αυτές τις συνθήκες, η ηλεκτροκίνηση είναι ένα από τα σημαντικότερα πεδία έρευνας τα τελευταία χρόνια.

Τα ηλεκτρικά οχήματα παρουσιάζουν διάφορα πλεονεκτήματα σε σύγκριση με τα συμβατικά αυτοκίνητα μηχανών καύσης. Συγκεκριμένα, κάποια από αυτά τα πλεονεκτήματα είναι τα εξής (όπως περιγράφονται στο [2]):

- Μηδενικές εκπομπές: Τα ηλεκτρικά αυτοκίνητα δεν εκπέμπουν ούτε διοξείδιο του άνθρακα ούτε διοξείδιο του αζώτου. Εκτός αυτού οι κατασκευαστική διαδικασία τείνει να είναι πιο φιλική προς το περιβάλλον, ωστόσο, η κατασκευή μπαταριών επηρεάζει αρνητικά τα αποτυπώματα άνθρακα.
- Κόστος: Το κόστος συντήρησης ενός ηλεκτρικού αυτοκινήτου και το κόστος ηλεκτρικής ενέργειας είναι σημαντικά μικρότερο από το κόστος συντήρησης ενός αυτοκινήτου εσωτερικής καύσης και τα καύσιμα αυτού.
- Άνεση: Τα ταξίδια με ηλεκτρικά οχήματα προσφέρουν περισσότερη άνεση λόγω της έλλειψης δονήσεων και ήχων από τη μηχανή.
- Αποδοτικότητα: Τα ηλεκτρικά αυτοκίνητα είναι πιο αποδοτικά από τα παραδοσιακά οχήματα εσωτερικής καύσης. Ωστόσο, η ολική well to wheel (WTW) απόδοση εξαρτάται και από την απόδοση του εργοστασίου παραγωγής ηλεκτρικής ενέργειας. Παραδείγματος χάριν, η απόδοση WTW ενός βενζινοκίνητου οχήματος ανέρχεται κοντά στο 11 με 27% και ενός ντιζελοκίνητου οχήματος ανέρχεται στο 25 με 37%. Αντιθέτως, τα ηλεκτρικά οχήματα, όταν φορτίζουν με ρεύμα που παρήχθη σε εργοστάσιο παραγωγής ηλεκτρικής ενέργειας φυσικού αερίου, έχουν μία απόδοση 13 με 31%, ωστόσο, με φόρτιση από ανανεώσιμες πηγές ενέργειας η απόδοση φτάνει μέχρι και 70%.
- Προσβασιμότητα: Τα ηλεκτρικά αυτοκίνητα επιτρέπεται να κυκλοφορούν ελεύθερα σε περιοχές όπου οχήματα εσωτερικής έχουν περιορισμένη πρόσβαση (π.χ. δακτύλιος Αθηνών).

Ωστόσο, εξαιτίας του γεγονότος ότι για τη λειτουργία τους απαιτούνται μπαταρίες, προκύπτουν οι εξής προκλήσεις (όπως περιγράφονται στο [2]):

- Αυτονομία οδήγησης: Ο αριθμός των χιλιομέτρων που μπορεί να ταξιδέψει ένα ηλεκτρικό αυτοκίνητο είναι ένα θέμα που απασχολεί τους ιδιοκτήτες ηλεκτρικών οχημάτων. Η αυτονομία των ηλεκτρικών οχημάτων κυμαίνεται μεταξύ διακοσίων και τετρακοσίων χιλιομέτρων, όμως το εύρος αυτό συνεχώς βελτιώνεται. Παραδείγματος χάριν, το ηλεκτρικό αυτοκίνητο Nissan Leaf έχει αυτονομία κοντά στα 364 km, ενώ το Tesla Model S Plaid έχει αυτονομία έως και 650 km.
- Χρόνος φόρτισης: Μία πλήρης φόρτιση ενός ηλεκτρικού οχήματος διαρκεί 4 με 8 ώρες. Ωστόσο, σε φορτιστές ταχείας φόρτισης μία φόρτιση μέχρι το 80% μπορεί να διαρκεί 30 λεπτά.

- Υποδομές φόρτισης: Ένα άλλο θέμα που απασχολεί τους ιδιοκτήτες ηλεκτρικών οχημάτων είναι η έλλειψη σημείων φόρτισης. Αυτό το θέμα, ωστόσο, με την πάροδο του χρόνου τείνει να εξαλειφθεί, αφού τα ηλεκτρικά οχήματα έχουν αρχίσει να επικρατούν.
- Κόστος μπαταρίας: Ηλεκτρικά αυτοκίνητα που έχουν μεγαλύτερες μπαταρίες κοστίζουν περισσότερο.
- Μάζα και όγκος: Οι μπαταρίες ηλεκτρικών αυτοκινήτων είναι βαριές και καταλαμβάνουν ένα σημαντικό ποσοστό του χώρου του οχήματος. Μία τυπική μπαταρία ηλεκτρικού οχήματος ζυγίζει κοντά στα 200 κιλά, αλλά αυτό εξαρτάται και από την χωρητικότητα της μπαταρίας.

Μπαταρίες ηλεκτρικών οχημάτων

Όπως αναδείχθηκε παραπάνω οι μπαταρίες ηλεκτρικών οχημάτων είναι ένα θεμελιώδες θέμα που συζητείται σχετικά με τις προκλήσεις που πρέπει να αντιμετωπίσουν τα ηλεκτρικά οχήματα. Τα τελευταία χρόνια έχει γίνει αξιοσημείωτη πρόοδος σε τεχνολογίες που χρησιμοποιούνται για την κατασκευή μπαταριών. Επίσης, η παγκόσμια παραγωγή μπαταριών έχει αυξηθεί σημαντικά, το οποίο αδιαμφισβήτητα σχετίζεται με την αύξηση πωλήσεων ηλεκτρικών οχημάτων. Αυτή η αυξητική τάση φαίνεται να συνεχίζεται και στο μέλλον, με όλο και περισσότερους κατασκευαστές αυτοκινήτων να προσφέρουν περισσότερες επιλογές ηλεκτρικών οχημάτων. Παρακάτω θα αναλυθούν μερικά από τα κύρια χαρακτηριστικά των μπαταριών.

Δύο από τα πιο βασικά χαρακτηριστικά των μπαταριών είναι η χωρητικότητα της μπαταρίας και η ενεργειακή κατάσταση αυτής. Η δυσκολία και το κόστος αποθήκευσης ηλεκτρικής ενέργειας είναι ένα θεμελιώδες πρόβλημα, όπως αναλύθηκε και στο υποκεφάλαιο της αγοράς ηλεκτρικής ενέργειας. Για το λόγο αυτό, κάθε χρόνο επενδύονται μεγάλα χρηματικά ποσά για την αύξηση της απόδοσης και της αξιοπιστίας των μπαταριών. Η χωρητικότητα της μπαταρίας είναι η μέγιστη ποσότητα ενέργειας που μπορεί να εξαχθεί από αυτήν κάτω από συγκεκριμένες συνθήκες. Μετριέται σε αμπερώρες (Ah) ή κιλοβατώρες (KWh), αλλά οι KWh είναι το πιο συνηθισμένο. Συνήθως, τα σημερινά ηλεκτρικά αυτοκίνητα έχουν μπαταρίες που κυμαίνονται από 20 έως και 100 KWh. Όσον, αφορά την ενεργειακή κατάσταση της μπαταρίας, αυτός ο όρος αναφέρεται στο επίπεδο της μπαταρίας (ποσοστό επί της 100), σε σύγκριση με το 100% της χωρητικότητας αυτής.

Κάποια επίσης σημαντικά χαρακτηριστικά των μπαταριών είναι οι κύκλοι φόρτισης και η διάρκεια ζωής. Ένας κύκλος φόρτισης θεωρείται ότι ολοκληρώνεται όταν η μπαταρία έχει αποφορτίσει πλήρως ή όταν έχει φορτίσει στο 100%. Η διάρκεια ζωής μιας μπαταρίας μετριέται σε κύκλους φόρτισης που η μπαταρία αντέχει και είναι ένας σημαντικός παράγοντας αξιολόγησης μίας μπαταρίας. Ο στόχος είναι να κατασκευάζονται μπαταρίες που μπορούν να αντέξουν μεγάλο αριθμό από κύκλους φόρτισης και αποφόρτισης.

Σταθμοί Φόρτισης Ηλεκτρικών Οχημάτων

Εισαγωγή

Τα τελευταία χρόνια, πολλές χώρες έχουν θέσει συγκεκριμένους στόχους σχετικά με την σταδιακή αντικατάσταση των οχημάτων που χρησιμοποιούν ορυκτά καύσιμα με ηλεκτρικά αυτοκίνητα, εξαιτίας των αρνητικών επιδράσεων που επιφέρουν στο περιβάλλον. Η χρήση των ηλεκτρικών οχημάτων είναι βέβαιο ότι θα αυξηθεί τα επόμενα χρόνια. Αυτό με τη σειρά του δημιουργεί την ανάγκη για εξόπλιση του συστήματος μεταφορών με υποδομές φόρτισης και, συγκεκριμένα, σταθμούς φόρτισης ηλεκτρικών οχημάτων. Λόγω της αύξησης πωλήσεων ηλεκτρικών αυτοκινήτων υπάρχει μία αυξητική τάση επενδύσεων προς υποδομές φόρτισης αυτών.

Η μεγαλύτερη πιθανή μείωση εκπομπών αερίων του θερμοκηπίου με τη χρήση των ηλεκτρικών οχημάτων, έναντι συμβατικών οχημάτων, συμβαίνει κατά τη φόρτιση, ειδικά αν η ηλεκτρική ενέργεια που χρησιμοποιείται προκύπτει από ανανεώσιμες πηγές ενέργειας [4]. Εκτός αυτού ένας παράγοντας που θα μπορούσε να συνεισφέρει είναι η χρήση υποδομών φόρτισης που εμπεριέχουν κάποια έξυπνη λειτουργικότητα. Ως αποτέλεσμα αυτού έχει αναδειχθεί η ανάγκη για δημιουργία έξυπνων βιώσιμων επιχειρησιακών μοντέλων για σταθμούς φόρτισης, οι οποίοι θα έχουν διαμορφωθεί με προδιαγραφές οι οποίες θα επιτρέπουν λειτουργικότητες έξυπνης φόρτισης.

Στο υπόλοιπο αυτού του υποκεφαλαίου, λοιπόν, θα γίνει μία περιγραφή κάποιων βασικών εννοιών που σχετίζονται με σταθμούς φόρτισης που εμπεριέχουν έξυπνες λειτουργικότητες. Αρχικά, θα περιγραφεί η τεχνολογία Vehicle-to-Grid, η οποία συναντάται συχνά σε συστήματα που υλοποιούν έξυπνη φόρτιση μεταξύ οχημάτων και δικτύου. Ακολούθως, θα γίνει μία ανάλυση κάποιων χαρακτηριστικών που διαθέτουν σταθμοί με δυνατότητα έξυπνης φόρτισης. Τέλος, θα αναφερθούν κάποιες προκλήσεις σχετικές με τους σταθμούς φόρτισης και θα περιγραφούν κάποιες τεχνολογίες σχετικές με τους σταθμούς φόρτισης πάνω στις οποίες γίνεται έρευνα τα τελευταία χρόνια.

Vehicle-To-Grid

Η ενσωμάτωση των ηλεκτρικών οχημάτων στο σύστημα μεταφορών και στο δίκτυο ηλεκτρικής ενέργειας αναδύει κάποια αναπόφευκτα προβλήματα, τα οποία προκύπτουν κυρίως λόγω της αύξησης ηλεκτρικού φορτίου που απαιτείται κατά τη φόρτιση του στόλου των ηλεκτρικών οχημάτων. Ωστόσο, η εισαγωγή όλων αυτών των οχημάτων δίνει την ευκαιρία για εφαρμογή Vehicle-to-Grid.

Το Vehicle-to-Grid, ή αλλιώς και V2G, περιγράφει ένα σύστημα το οποίο plug-in ηλεκτρικά αυτοκίνητα (BEVs, PHEVs και ER-EVs που αναλύθηκαν στο υποκεφάλαιο των ηλεκτρικών οχημάτων) επικοινωνούν με το δίκτυο ηλεκτρικής ενέργειας για πώληση υπηρεσιών ανταπόκρισης στη ζήτηση [5]. Με άλλα λόγια, τα ηλεκτρικά αυτοκίνητα που συμμετέχουν σε V2G ανταποκρίνονται σε ανάγκες που έχει το δίκτυο ηλεκτρικής ενέργειας μέσω επιστροφής ηλεκτρικής ενέργειας στο δίκτυο ή μέσω μείωσης του ρυθμού που φορτίζουν.

Συγκεκριμένα, υπάρχουν τρία είδη διαφορετικών τεχνολογιών σχετικά με τεχνολογίες διασύνδεσης των ηλεκτρικών οχημάτων με το δίκτυο [5]. Πρώτον, υπάρχει η έννοια του Vehicle-to-Home (V2H), η οποία αναφέρεται στην ανταλλαγή ενέργειας μεταξύ του οχήματος και του δικτύου ηλεκτρικού ρεύματος ενός σπιτιού. Σε αυτήν την περίπτωση, η

μπαταρία του ηλεκτρικού οχήματος μπορεί να χρησιμοποιηθεί για αποθήκευση ενέργειας, η οποία μπορεί να χρησιμοποιηθεί σε μεταγενέστερη στιγμή στις ηλεκτρικές συσκευές του σπιτιού. Δεύτερον, υπάρχει η έννοια Vehicle-to-Vehicle (V2V), η οποία αναφέρεται στην ανταλλαγή ηλεκτρικής ενέργειας μεταξύ οχημάτων που βρίσκονται σε μία τοπική κοινωνία. Τρίτον, υπάρχει η έννοια V2G που αναφέρθηκε παραπάνω, η οποία ενδεχομένως εκμεταλλεύεται την ενέργεια από μία τοπική κοινότητα ηλεκτρικών αυτοκινήτων και ανταλλάσσει ενέργεια με το δίκτυο, με χρήση aggregators για τον έλεγχο και τη διαχείριση της ορθής λειτουργίας.

Σταθμοί φόρτισης με δυνατότητα έξυπνης φόρτισης

Σε σταθμούς φόρτισης ηλεκτρικών οχημάτων που ενσωματώνουν έξυπνη φόρτιση, υπάρχει ένα σύστημα διαχείρισης που προγραμματίζει και ελέγχει τη φόρτιση των EVs [4]. Σε ένα τέτοιο σύστημα, μία συντονιστική οντότητα χειρίζεται την είσοδο αυτοκινήτων στο σταθμό φόρτισης, αναθέτει τα ηλεκτρικά οχήματα σε κατάλληλα σημεία φόρτισης και ελέγχει την ενέργεια του κάθε ηλεκτρικού αυτοκινήτου. Οι αποφάσεις αυτές λαμβάνονται με στόχο τόσο την εξυπηρέτηση των σκοπών του κάθε ηλεκτρικού αυτοκινήτου, όσο και με την επίτευξη κάποιων στόχων του σταθμού φόρτισης, όπως είναι η μείωση του λειτουργικού κόστους του σταθμού φόρτισης, η επίτευξη της δίκαιης κατανομής των πόρων του σταθμού προς τα οχήματα, ή η συμμόρφωση με κάποιους περιορισμούς που σχετίζονται με την ασφαλή λειτουργία του υποκείμενου ηλεκτρικού δικτύου. Οι σταθμοί φόρτισης ηλεκτρικών οχημάτων με δυνατότητα έξυπνης φόρτισης μπορούν να εκμεταλλευτούν λειτουργικότητες όπως το V2G για την επίτευξη των αντικειμενικών στόχων που πρέπει να πετύχουν.

Για τη βέλτιστη λειτουργία ενός σταθμού με δυνατότητα έξυπνης φόρτισης, υπάρχουν τρία κύρια πεδία μελέτης [4]. Το πρώτο σχετίζεται με την τον σχεδιασμό και το μέγεθος των σταθμών φόρτισης. Η έρευνα σε αυτό το πεδίο ασχολείται με την τοποθεσία των σταθμών και το μέγεθος των σταθμών. Το δεύτερο πεδίο μελέτης επικεντρώνεται στον σχεδιασμό έξυπνων αλγορίθμων χρονοπρογραμματισμού για τη βελτιστοποίηση της λειτουργίας των σταθμών φόρτισης. Το κίνητρο σε αυτού του είδους την έρευνα είναι οικονομικά ή και τεχνικά πλεονεκτήματα που μπορούν να προκύψουν μέσω μίας βέλτιστης χρονοδρομολόγησης της φόρτισης των οχημάτων. Τέλος, το τρίτο πεδίο μελέτης έχει να κάνει με το μέγεθος των σταθμών φόρτισης, με στόχο την υψηλή ποιότητα εξυπηρέτησης (Quality of Service) των ιδιοκτητών ηλεκτρικών οχημάτων. Η ποιότητα εξυπηρέτησης είναι ένας εξαιρετικά σημαντικός δείκτης όσον αφορά την απόδοση ενός σταθμού φόρτισης, αφού αναδεικνύει την ορθή ή όχι λειτουργία του σταθμού και των υπηρεσιών αυτού, από την πλευρά των ιδιοκτητών ηλεκτρικών οχημάτων.

Προκλήσεις και μελλοντική έρευνα

Η διεξαγωγή έρευνας σχετική με τους σταθμούς φόρτισης ηλεκτρικών οχημάτων είναι απαραίτητη. Ωστόσο, θεωρείται ότι είναι ακόμα σε αρχικό στάδιο [6]. Αυτό συμβαίνει λόγω του γεγονότος ότι ο αριθμός των ηλεκτρικών οχημάτων είναι ακόμα ένα μικρό ποσοστό του συνολικού αριθμού των οχημάτων, καθώς επίσης και λόγω της έλλειψης δεδομένων σχετικά με ηλεκτρικά αυτοκίνητα και τους σταθμούς φόρτισης. Εκτός αυτού, η τεχνολογία στο πεδίο των EVs και των σταθμών φόρτισης συνεχώς εξελίσσεται.

Τα ηλεκτρικά αυτοκίνητα και οι σταθμοί φόρτισης, ωστόσο, έχουν να αντιμετωπίσουν και διάφορες προκλήσεις. Καταρχάς, εξαιτίας του γεγονότος ότι κάποιες χώρες είναι

περισσότερο εξαρτημένες σε ορυκτά καύσιμα είναι διστακτικές στην προώθηση των ηλεκτρικών αυτοκινήτων [8]. Επίσης, εφόσον τα ηλεκτρικά οχήματα καταλαμβάνουν μόνο ένα μικρό μέρος όλης της αγοράς οχημάτων, οι επενδύσεις προς σταθμούς φόρτισης ενδεχομένως να αργήσουν να επιφέρουν κέρδη στους πιθανούς επενδυτές [8]. Τέλος, ένα άλλο θέμα που έχει προκύψει είναι κάποιοι περιορισμοί σχετικά με το δίκτυο ηλεκτρικής ενέργειας, δηλαδή η πιθανή ανεπάρκεια του δικτύου να παρέχει ηλεκτρική ενέργεια προς όλα τα ηλεκτρικά οχήματα που αναμένεται να κυκλοφορούν τα επόμενα χρόνια [8].

Από τα παραπάνω γίνεται ξεκάθαρο ότι η προώθηση της ηλεκτροκίνησης είναι μονόδρομος εάν η κοινωνία στοχεύει να πετύχει τους στόχους της σχετικά με την κλιματική ουδετερότητα. Ταυτόχρονα, η δημιουργία σταθμών με δυνατότητα έξυπνης φόρτισης έχει να προσφέρει πολλά πλεονεκτήματα τόσο στους διαχειριστές των σταθμών φόρτισης, όσο και στους ιδιοκτήτες των ηλεκτρικών οχημάτων. Λαμβάνοντας υπόψιν τα παραπάνω αξίζει να σημειωθεί ότι το σύστημα που υλοποιήθηκε στο πλαίσιο αυτής της διπλωματικής, θα μπορούσε να ήταν κομμάτι ενός συστήματος που παρέχει υπηρεσίες έξυπνης φόρτισης, εφόσον υλοποιείται μέρος λειτουργικότητας που θα ήταν ιδιαίτερα χρήσιμο για ένα τέτοιο σύστημα (πχ κρατήσεις φόρτισης ηλεκτρικών οχημάτων). Στα επόμενα κεφάλαια θα γίνει αναλυτική περιγραφή του συστήματος για να αναδειχθούν οι χρήσιμες λειτουργικότητες αυτού.

Κεφάλαιο 2: Εργαλεία που χρησιμοποιήθηκαν

React JS Library

Η React JS, ή απλώς React, είναι μία open source βιβλιοθήκη JavaScript που χρησιμοποιείται για τη δημιουργία User Interfaces βασιζόμενη σε στοιχεία διεπαφής χρήστη (ή αλλιώς UI components) [9]. Είναι μία από τις δημοφιλέστερες βιβλιοθήκες για την ανάπτυξη User Interface εφαρμογών. Η React αναπτύχθηκε από την εταιρία Meta (πρώην Facebook) και συνεχίζει να συντηρείται από αυτήν καθώς και από μία μεγάλη κοινότητα από ανεξάρτητους developers και εταιρίες.

Η React βασίζεται σε μία declarative, component-based λογική, στοχεύοντας στο να διευκολύνει τη δημιουργία interactive UIs [9]. Λόγω της declarative (δηλωτικής) λογικής, οι προγραμματιστές μπορούν να σχεδιάσουν όψεις για κάθε πιθανή κατάσταση της εφαρμογής με την React να αναλαμβάνει το rendering του κατάλληλου αντικειμένου δεδομένης μίας κατάστασης, ενώ ταυτοχρόνως, κάνει updates στα components όταν τα δεδομένα που υπάρχουν σε αυτά έχουν αλλάξει. Λόγω της component-based λογικής, οι προγραμματιστές αναπτύσσουν components σχεδιασμένα έτσι ώστε να εμπεριέχουν τις δικές τους καταστάσεις (states), ενώ με σύνθεση αυτών των components δημιουργούνται πιο περίπλοκες εφαρμογές. Δεδομένου ότι η λογική ενός component είναι γραμμένη σε JavaScript αντί για απλά templates, είναι ιδιαίτερα εύκολο για τους προγραμματιστές να περάσουν πιο περίπλοκα δεδομένα μέσα στην εφαρμογή, διευκολύνοντας τους να δημιουργήσουν interactive UIs.

Ο κώδικας React αποτελείται από επαναχρησιμοποιούμενα components. Παρακάτω φαίνονται κάποια παραδείγματα function components.

```
function Station() {
    return <div>
        <h2>Station Name: <span>Example Station</span></h2>
        <h2>Station Operator: <Operator /> </h2>
    </div>
}

function Operator() {
    return <span>Operator Name</span>
}
```

Στο παράδειγμα αυτό έχουμε ένα function component που επικεντρώνεται στο rendering του σταθμού, το οποίο χρησιμοποιεί ένα component που ονομάζεται Operator και ασχολείται με το rendering των στοιχείων ενός operator. Στο συγκεκριμένο παράδειγμα οι συναρτήσεις επιστρέφουν στοιχεία JSX (JavaScript Syntax Extension), που είναι μία επέκταση της JavaScript που μοιάζει σε μεγάλο βαθμό με την HTML, η οποία χρησιμοποιείται για τη δημιουργία ιστοσελίδων.

Τέλος, ένα από τα θετικά της React JS είναι ότι επιτρέπει να χρησιμοποιηθούν μαζί με αυτήν άλλες βιβλιοθήκες και frameworks. Αυτό είναι ιδιαίτερα χρήσιμο, καθώς δίνει τη δυνατότητα να αναπτύσσονται πολύ πιο γρήγορα οι εφαρμογές, εφόσον μπορούν να χρησιμοποιούνται έτοιμες βιβλιοθήκες ή frameworks που έχουν δημιουργηθεί για κάποιες συγκεκριμένες λειτουργίες.

Η React JS χρησιμοποιήθηκε στο πλαίσιο αυτής της διπλωματικής για τη δημιουργία του front end environment, με στόχο η web εφαρμογή που θα χρησιμοποιούν οι operators των σταθμών και οι ιδιοκτήτες των οχημάτων να είναι διαδραστική και φιλική προς αυτούς.

Python Django

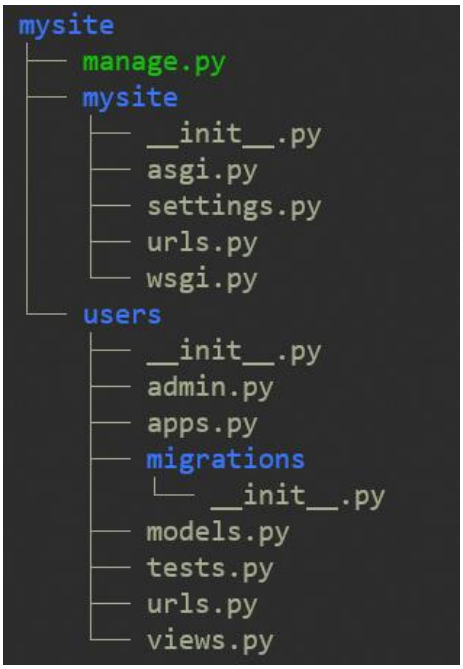
Το Django είναι ένα open source Python web framework που στοχεύει στην γρήγορη ανάπτυξη λογισμικού, ενώ ταυτοχρόνως επιδιώκει καθαρό και πρακτικό σχεδιασμό [10]. Το Django συντηρείται από το Django Software Foundation, ένα ανεξάρτητο οργανισμό που επικεντρώνεται στην ανάπτυξη και συντήρηση αυτού του ευρέως διαδεδομένου framework. Ακολουθεί αρχιτεκτονικά patterns της μορφής model-template-views (MTV).

Ο κύριος στόχος του Django είναι η διευκόλυνση και η επιτάχυνση της δημιουργίας περίπλοκων εφαρμογών που απαιτούν έντονη χρήση του database, ενώ παράλληλα βάζει σε προτεραιότητα την ασφάλεια και το scalability των web εφαρμογών [10].

Μετά την εγκατάσταση του django web framework είναι πολύ εύκολο να δημιουργηθεί μία web εφαρμογή με τη χρήση απλών εντολών που παρέχει το django, για δημιουργία projects και apps. Ένα django project περιέχει κώδικα που είναι συνήθως αυτόματα παραγόμενος από τις εντολές που προαναφέρθηκαν, ο οποίος περιέχει ρυθμίσεις σχετικά με το database configuration καθώς και άλλες σχετικά με το project. Εκτός αυτού, μέσα σε ένα django project περιέχονται apps, τα οποία υλοποιούν την κύρια λειτουργικότητα μίας εφαρμογής django. Κάθε app επικεντρώνεται στο χειρισμό κάποιων συγκεκριμένων λειτουργιών μιας εφαρμογής, κάτι το οποίο διευκολύνει τους προγραμματιστές να γράφουν πιο καθαρό κώδικα.

Στην εικόνα 3 φαίνεται ένα παράδειγμα του structure ενός django project για ένα web application που ονομάζεται mysite, καθώς και ένα django app που εμπεριέχεται σε αυτό το project και λέγεται users, που έχει ως στόχο να επικεντρωθεί στον χειρισμό των λειτουργικοτήτων των χρηστών αυτού του συστήματος.

Στην εικόνα 3 φαίνεται το structure που ακολουθούν τα django apps. Τα βασικά αρχεία κάθε django app είναι το models.py, το urls.py και το views.py. Το αρχείο models.py περιέχει όλα τα μοντέλα του app, τα οποία είναι κλάσεις που περιγράφουν τα database tables, τα πεδία του κάθε table και γενικότερα τους περιορισμούς προς το database. Τα μοντέλα αυτά είναι ο built-in τρόπος του django για σύνδεση μιας django εφαρμογής με databases, και διευκολύνουν σε μεγάλο βαθμό την επικοινωνία της εφαρμογής με το database, η οποία γίνεται μέσω του Django ORM. Ακολούθως, το urls.py αρχείο περιέχει όλα URLs, δηλαδή τις διευθύνσεις από τις ιστοσελίδες, τις εικόνες και γενικότερα τα αρχεία ενός django app. Στο αρχείο αυτό περιγράφεται το πως θα πρέπει η εφαρμογή να χειρίζεται requests προς συγκεκριμένα URLs, όπου συνήθως γίνεται παραπομπή προς κάποια συνάρτηση που βρίσκεται στο views.py. Το αρχείο views.py, με τη σειρά του, υλοποιεί τα views ενός django app, τα οποία είναι συνήθως συναρτήσεις που περιγράφουν το πως θα αλληλεπιδράσει η εφαρμογή με τον client.



Εικόνα 3: Δομή ενός django project

Το Django Framework χρησιμοποιήθηκε στο πλαίσιο αυτής της διπλωματικής για τη δημιουργία του back end, με στόχο να επιταχυνθεί η δημιουργία αυτού του component, λαμβάνοντας, ταυτοχρόνως, υπόψιν την ασφάλεια που απαιτείται σε μία τέτοια εφαρμογή.

Django REST Framework

Το Django REST framework είναι ένα lightweight framework σε γλώσσα Python που χρησιμοποιείται πάνω από το Django [11]. Είναι ένα ιδιαίτερα χρήσιμο εργαλείο για ανάπτυξη Web APIs.

Το Django REST framework χαρακτηρίζεται από έναν ισχυρό μηχανισμό σειριοποίησης (serialization), που είναι η διαδικασία μετάφρασης μίας δομής δεδομένων ή μίας κατάστασης αντικειμένου, σε μία μορφή που μπορεί να μεταδοθεί και να ανακατασκευαστεί (όπως πχ. είναι το JSON, το XML και άλλα) [12]. Επίσης, η σειριοποίηση υποστηρίζεται και προς κλάσεις του django ORM, κάτι το οποίο είναι ιδιαίτερα χρήσιμο, αφού διευκολύνεται η μετάδοση δεδομένων που υπάρχουν στο database προς κάποιο εξωτερικό σύστημα που θέλει να τα χρησιμοποιήσει.

Το Django REST framework χρησιμοποιήθηκε στο πλαίσιο της διπλωματικής για την ευκολότερη σειριοποίηση και μετάδοση δεδομένων από το back end προς το front end του συστήματος.

Python Celery & Django Celery Beat

Το Celery είναι μια opensource υλοποίηση μίας ουράς εργασιών για Python web εφαρμογές, που επιτρέπει την εκτέλεση εργασιών ασύγχρονα, εκτός του HTTP request-response κύκλου [13]. Το Celery είναι ιδιαίτερα χρήσιμο για εφαρμογές που πρέπει να τρέξουν στο παρασκήνιο και, ενδεχομένως, να επαναλαμβάνονται ανά τακτά χρονικά διαστήματα.

Παραδείγματα τέτοιων εφαρμογών είναι η μαζική αποστολή email σε πλήθος χρηστών, η επαναλαμβανόμενη συγκέντρωση δεδομένων από ένα εξωτερικό σύστημα ή επεξεργασία μεγάλων όγκων δεδομένων.

Μία θεμελιώδης έννοια στο Celery είναι ο διαχωρισμός του συστήματος που εκτελεί εργασίες, που ονομάζεται celery worker, και του συστήματος που δημιουργεί το χρονοδιάγραμμα των εργασιών, που ονομάζεται celerybeat.

Το Django Celery Beat μία επέκταση του Celery για το Django framework, που μπορεί να χρησιμοποιηθεί για το χειρισμό περιοδικών εργασιών και την αποθήκευση τους στο database.

Στο πλαίσιο αυτής της διπλωματικής χρησιμοποιήθηκε το Django Celery Beat για την περιοδική συγκέντρωση των τιμών της ηλεκτρικής ενέργειας από άλλα εξωτερικά συστήματα. Το Django Celery Beat ήταν ιδιαίτερα χρήσιμο για αυτήν την εργασία, καθώς οι τιμές της ηλεκτρικής ενέργειας αλλάζουν ανά τακτά χρονικά διαστήματα και απαιτείται περιοδική ενημέρωση των τιμών, ώστε τα δεδομένα που έχει το σύστημα να είναι όσο το δυνατόν πιο πρόσφατα.

Swagger: API Documentation

Το Swagger είναι μία σουίτα από εργαλεία ανάπτυξης λογισμικού για APIs, που προσφέρονται από την εταιρία SmartBear Software, ακολουθώντας προδιαγραφές OpenAPI Specification [14]. Το OpenAPI Specification προσδιορίζει μία standard, ανεξαρτήτου γλώσσας διεπαφή για APIs, που επιτρέπει σε ανθρώπους και υπολογιστές να ανακαλύψουν και να κατανοήσουν τις δυνατότητες μίας υπηρεσίας, χωρίς να έχουν πρόσβαση στον πηγαίο κώδικα ή το εσωτερικό documentation.

Το Swagger UI είναι μέρος του Swagger και είναι ένα open source εργαλείο που παράγει μία σελίδα με documentation για ένα API. Το Swagger UI είναι ιδιαίτερα δημοφιλές καθώς μπορεί να παραχθεί αυτόματα, για κάθε API που ακολουθεί το OpenAPI Specification.

Το Swagger UI χρησιμοποιήθηκε στο πλαίσιο αυτής της διπλωματικής για να παραχθεί αυτόματα API documentation για το back end API του συστήματος. Συγκεκριμένα, χρησιμοποιήθηκε μία υλοποίηση ενός Swagger γεννήτορα, που παρείχε συμβατότητα με το Django framework, ώστε να επιταχυνθεί η παραγωγή του API documentation για το back end σύστημα.

Κεφάλαιο 3: Σχεδιασμός της εφαρμογής

Στρατηγικές τιμολόγησης ενέργειας ανά KWh

Κατά την σχεδίαση αυτού του συστήματος αφιερώθηκε αρκετός χρόνος για την εύρεση μεθόδων τιμολόγησης της ενέργειας ανά KWh που θα αγοράζουν οι ιδιοκτήτες οχημάτων κατά την επίσκεψή τους στους σταθμούς φόρτισης. Συγκεκριμένα, δόθηκε ιδιαίτερη προσοχή στη δομή που πρέπει να έχουν οι μηχανισμοί τιμολόγησης, ώστε να είναι όσο το δυνατόν πιο εύκολο για έναν operator να ελέγχει με τον καλύτερο δυνατό τρόπο τις τιμές ανά KWh που προσφέρουν οι σταθμοί του.

Οργάνωση των φορτιστών

Λόγω του γεγονότος ότι οι operators πρέπει να ελέγχουν εύκολα και αποτελεσματικά τις τιμές των φορτιστών που βρίσκονται στους σταθμού τους, οι φορτιστές κάθε σταθμού οργανώθηκαν σε ομάδες φορτιστών, ή αλλιώς charger groups. Κάθε charger group έχει κάποια συγκεκριμένη τιμή χρέωσης ανά KWh και ακολουθεί κάποια προκαθορισμένη στρατηγική τιμολόγησης (pricing strategy). Έτσι, φορτιστές που ανήκουν στο ίδιο charger group, έχουν ίδια στρατηγική τιμολόγησης και χρέωση ανά KWh. Να σημειωθεί ότι τα charger groups ονομάζονται αλλιώς και Pricing Groups και στο εξής οι δύο αυτές έννοιες θα είναι συνώνυμες.

Μέθοδοι τιμολόγησης ενέργειας

Από τα παραπάνω γίνεται κατανοητό ότι η τιμή της ηλεκτρικής ενέργειας που θα προσφέρεται από τους σταθμούς στα οχήματα είναι ένα σημαντικό θέμα, καθώς από εκεί προκύπτει ένα μεγάλο ποσοστό των εσόδων των σταθμών φόρτισης ηλεκτρικών οχημάτων.

Έτσι, το σύστημα που αναπτύχθηκε στο πλαίσιο αυτής της διπλωματικής οφείλει να παρέχει κάποιες χρήσιμες στρατηγικές τιμολόγησης, τις οποίες θα μπορούν να χρησιμοποιούν έτοιμες οι operators, θέτοντας συγκεκριμένες παραμέτρους αναλόγως των στόχων που έχουν για κάποιο σταθμό.

Παρακάτω αναλύονται οι τέσσερις στρατηγικές τιμολόγησης που μπορούν να χρησιμοποιήσουν οι operators μέσω της εφαρμογής, ξεκινώντας από μία απλοϊκή μέθοδο και συνεχίζοντας με πιο «έξυπνες» μεθόδους.

Μέθοδος σταθερής τιμής (Fixed Price)

Αυτή είναι η πιο απλή μέθοδος τιμολόγησης που μπορεί να χρησιμοποιήσει ένας operator. Σε αυτή τη μέθοδο, ο operator θέτει μία σταθερή τιμή ανά KWh που θέλει να προσφέρει στα οχήματα που φορτίζουν στο σταθμό του. Έτσι, η τιμή (€/KWh) ενός charger group που χρησιμοποιεί αυτή τη μέθοδο είναι η εξής:

$$Price = c$$

Όπου c είναι μια σταθερά που θέτει ο χρήστης.

Μέθοδος σταθερού κέρδους (Fixed Profit)

Η μέθοδος σταθερού κέρδους στοχεύει στην επίτευξη σταθερού κέρδους ανά KWh ανεξαρτήτως της τιμής της ενέργειας που διατίθεται από την αγορά ηλεκτρικής ενέργειας ή άλλων εξόδων που ο operator θέλει να προσμετρήσει στην κοστολόγηση κάθε KWh. Επομένως, η τιμή (€/KWh) ενός charger group που χρησιμοποιεί αυτή τη μέθοδο προκύπτει από τον παρακάτω τύπο:

$$Price = all\_expenses + c$$

Όπου το $all\_expenses$ περιλαμβάνει όλα τα έξοδα που θέλει να κοστολογήσει σε κάθε KWh ο operator, δηλαδή (προαιρετικά) η τιμή της ενέργειας και (προαιρετικά) οποιοδήποτε άλλο κόστος. Η σταθερά c υποδηλώνει το κέρδος που θέλει να έχει ο operator σε κάθε KWh και παίρνει μη αρνητικές τιμές.

Να σημειωθεί ότι η τιμή της ενέργειας που προέρχεται από την αγορά ενέργειας (grid price), υπολογίζεται και προσμετράται αυτόματα από το σύστημα, ώστε να μην χρειάζεται ο operator να αλλάζει συνεχώς την τιμή που προσφέρει ο σταθμός, προσπαθώντας να επιτευχθεί σταθερό κέρδος.

Μέθοδος διαμόρφωσης τιμής αναλόγως της ζήτησης (Demand-centered Profit)

Είναι εξαιρετικά σύνηθες η τιμή ενός προϊόντος να αυξάνεται όταν η ζήτηση για αυτό το προϊόν αυξάνεται, όπως είναι γνωστό από το νόμος προσφοράς και ζήτησης. Η μέθοδος διαμόρφωσης τιμής αναλόγως της ζήτησης προσπαθεί να εκμεταλλευτεί αυτό το φαινόμενο, με στόχο τα παρακάτω:

- Καλύτερη διαχείριση της άφιξης των οχημάτων στο σταθμό, μέσω κινητροδότησης των οδηγών για φόρτιση στο σταθμό σε ώρες που η ζήτηση θα είναι χαμηλότερη
- Αποσυμφόρηση του σταθμού σε ώρες αιχμής
- Αποσυμφόρηση του δικτύου (grid) σε ώρες αιχμής
- Αύξηση κερδών του σταθμού ανά KWh που προσφέρεται, καθώς η ζήτηση πλησιάζει την μέγιστη διαθεσιμότητα των φορτιστών

Προφανώς, αυτή η μέθοδος τιμολόγησης μπορεί να έχει και κάποια αρνητικά, όπως για παράδειγμα το γεγονός ότι η τιμή θα αλλάζει συνεχώς κάθε φορά που θα μεταβάλλεται ο αριθμός των αυτοκινήτων που φορτίζουν σε αυτό το charger group. Ωστόσο, με σωστή ανάθεση των παραμέτρων που θα αναλυθούν παρακάτω, το μειονέκτημα αυτό μπορεί πρακτικά να εξαλειφθεί.

Έτσι, η τιμή (€/KWh) ενός charger group που χρησιμοποιεί αυτή τη μέθοδο προκύπτει από τον παρακάτω τύπο:

$$Price = all\_expenses + c1 + c2 * \frac{(group\_occupied\_chargers)^{\mathrm{n}}}{all\_group\_chargers}$$

Όπου:

- Το $all\_expenses$ είναι τα έξοδα (τιμή ενέργειας + επιπλέον έξοδα), όπως αναλύθηκαν στην μέθοδο σταθερού κέρδους.

- Το $c1$ είναι μία σταθερά που εκφράζει το ελάχιστο κέρδος ανά KWh και παίρνει μη αρνητικές τιμές.
- Το $c2$ είναι μία σταθερά που στοχεύει στον καθορισμό της μέγιστης τιμής ανά KWh που θα προσφέρεται από το charger group και παίρνει μη αρνητικές τιμές.
- Το n είναι μία σταθερά που εκφράζει το πως θα ανεβαίνει η τιμή σε σύγκριση με την αύξηση της ζήτησης και παίρνει ακέραιες θετικές τιμές. Για n μεγαλύτερο του 1, η τιμή ανά KWh αυξάνεται με εκθετικό τρόπο, καθώς οι φορτιστές γίνονται απασχολημένοι.
- Το $group\_occupied\_chargers$ είναι ο αριθμός των φορτιστών σε αυτό το charger group που είναι κατειλημμένοι, και υπολογίζεται αυτομάτως από το σύστημα (ο operator δεν αναθέτει τιμή άμεσα).
- Το $all\_group\_chargers$ είναι ο αριθμός όλων των φορτιστών σε αυτό το charger group (και πάλι, ο operator δεν αναθέτει τιμή άμεσα).

Μέθοδος διαμόρφωσης τιμής αναλόγως των ανταγωνιστών (Competitor-centered Profit)

Είναι εξαιρετικά σύνηθες οι operators των σταθμών να θέτουν την τιμή πώλησης ενέργειας των φορτιστών τους, αναλόγως της τιμής που προσφέρουν οι ανταγωνιστές. Επομένως, θα ήταν χρήσιμο να δημιουργηθεί μία μέθοδος που να εξυπηρετεί αυτό το φαινόμενο. Ωστόσο, για να είναι πλήρης αυτή η μέθοδος, θα πρέπει να περιλαμβάνει μία κατώτατη τιμή, δεδομένου ότι οι ανταγωνιστές μπορεί να προσφέρουν τόσο χαμηλές τιμές που είναι πλέον ασύμφορο για το σταθμό να της ακολουθήσει.

Λαμβάνοντας υπόψιν τα παραπάνω, η τιμή (€/KWh) ενός charger group που ακολουθεί μέθοδο διαμόρφωσης τιμής αναλόγως των ανταγωνιστών προκύπτει μέσω του παρακάτω τύπου:

$$Price = max(all\_expenses + c1, \\ min(competitors\_prices) + c2)$$

Όπου:

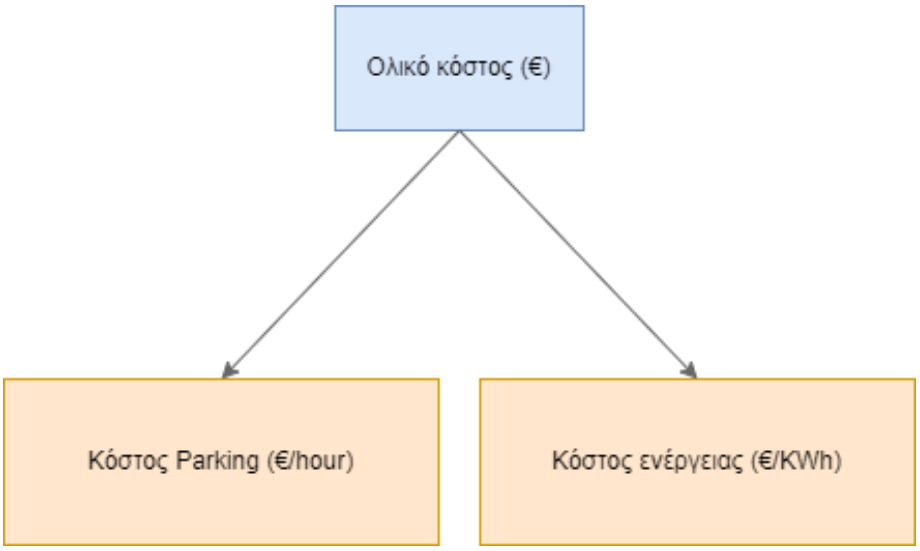
- Το $all\_expenses$ είναι τα έξοδα (τιμή ενέργειας + επιπλέον έξοδα), όπως αναλύθηκαν στην μέθοδο σταθερού κέρδους.
- Το $c1$ είναι μία σταθερά που εκφράζει το ελάχιστο κέρδος ανά KWh και παίρνει μη αρνητικές τιμές.
- Το $competitor\_prices$ είναι οι τιμές των ανταγωνιστών που έχει επιλέξει ο operator. Ως ανταγωνιστές ορίζονται οι σταθμοί φόρτισης και ως τιμή των ανταγωνιστών ορίζεται η ελάχιστη τιμή που προσφέρει ο σταθμός. Έτσι, $min(competitor\_prices)$ είναι η ελάχιστη τιμή από όλες τις τιμές των ανταγωνιστών.
- Το $c2$ είναι μία σταθερά που εκφράζει τη διαφορά της τιμής που θα έχει ο σταθμός σε σύγκριση με τις τιμές των ανταγωνιστών. Να σημειωθεί ότι το $c2$ μπορεί να πάρει και αρνητικές τιμές, ώστε το charger group να προσφέρει τιμή κάτω από όλους τους ανταγωνιστές.

Υπολογισμός κόστους φόρτισης

Όπως έγινε κατανοητό μέσω του προηγούμενου υποκεφαλαίου, κάθε φορτιστής ενός σταθμού θα ανήκει σε ένα Pricing Group που θα έχει μία από τις τέσσερις μεθόδους τιμολόγησης που αναλύθηκαν. Μέσω αυτών των μεθόδων τιμολόγησης (€/KWh), σε συνδυασμό με την ποσότητα ενέργειας (KWh) που μεταδόθηκε σε ένα όχημα, θα υπολογίζεται το κόστος (€) σχετικά με την κατανάλωση ενέργειας ενός οχήματος.

Ωστόσο, θα πρέπει εκτός του κόστους λόγω κατανάλωσης ενέργειας να υπάρχει και ένας παράγοντας που θα σχετίζεται με την διάρκεια παραμονής ενός αυτοκινήτου σε ένα σταθμό. Αυτό είναι απαραίτητο, καθώς ορισμένοι σταθμοί φόρτισης βρίσκονται σε περιοχές όπου η εύρεση parking για τα οχήματα είναι δύσκολη, οπότε οι ίδιοι οι σταθμοί, εκτός από υπηρεσίες φόρτισης παρέχουν και υπηρεσίες parking προς τα οχήματα που βρίσκονται στο σταθμό.

Επομένως, είναι επιθυμητό οι χρεώσεις που σχετίζονται με την επίσκεψη ενός οχήματος σε κάποιον σταθμό, να προκύπτουν από δύο παράγοντες, όπως φαίνεται στην εικόνα 4.



Εικόνα 4: Υπολογισμός ολικού κόστους φόρτισης

Προφανώς, το σύστημα πρέπει να δίνει τη δυνατότητα στον operator του σταθμού να θέτει το κόστος parking (€/ώρα) για κάθε σταθμό που του ανήκει. Έτσι, ο operator έχει τον πλήρη έλεγχο σχετικά με την κοστολόγηση μιας επίσκεψης ενός αυτοκινήτου σε ένα σταθμό φόρτισης.

Υπολογισμός κόστους ηλεκτρικής ενέργειας από το δίκτυο

Όπως φάνηκε στο υποκεφάλαιο με τις στρατηγικές τιμολόγησης, πολλές μέθοδοι απαιτούν να είναι γνωστή η τιμή της ηλεκτρικής ενέργειας (€/KWh) τη δεδομένη χρονική στιγμή, ώστε να παραχθεί η τιμή του Pricing Group (€/KWh). Για να συμβεί αυτό, το σύστημα επικοινωνεί με ένα εξωτερικό API, το οποίο επιστρέφει την τιμή της ενέργειας (€/MWh) για τις πιο πρόσφατες παρελθοντικές ώρες (έως και 1 ώρα πριν από την τρέχουσα ώρα). Περισσότερα σχετικά με αυτό θα αναλυθούν στο υποκεφάλαιο με την αρχιτεκτονική του συστήματος.

Επομένως, το σύστημα διαθέτει μόνο παρελθοντικές τιμές της ενέργειας. Ωστόσο, για να υπολογιστεί η τιμή του Pricing Group σε κάποια μελλοντική στιγμή (που είναι το πιο συνηθισμένο σενάριο) θα πρέπει να υπάρχουν διαθέσιμες και οι μελλοντικές τιμές της ενέργειας. Γι' αυτό το λόγο, υλοποιήθηκε ένας απλός αλγόριθμος πρόβλεψης της τιμής της ηλεκτρικής ενέργειας για μελλοντικές χρονικές στιγμές.

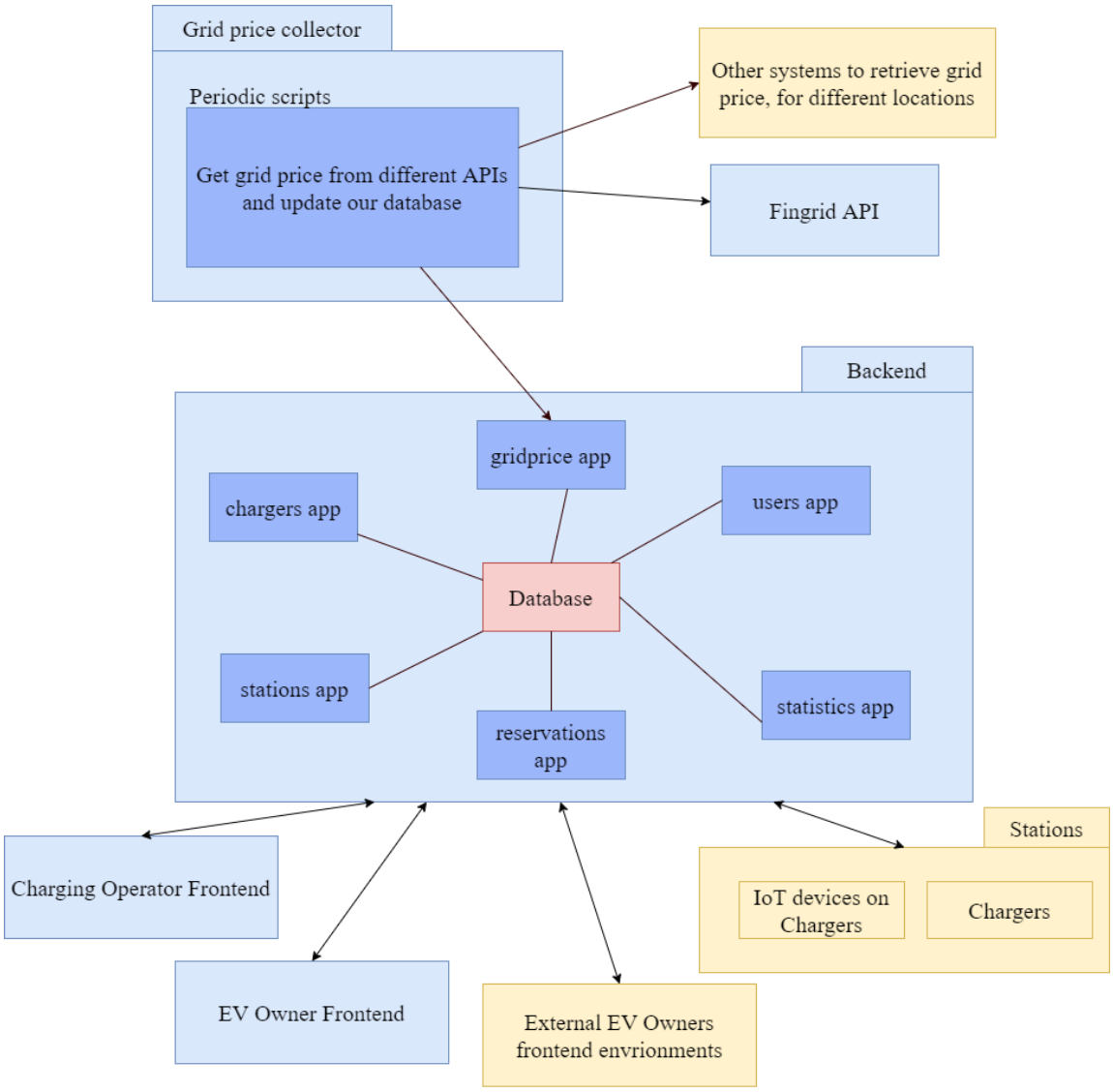
Στο πλαίσιο αυτής της διπλωματικής, δεν αναπτύχθηκε κάποιος περίπλοκος αλγόριθμος πρόβλεψης για την τιμή της ενέργειας. Η προσέγγιση που λήφθηκε ήταν αρκετά απλή, αφού η υλοποίηση αλγορίθμων πρόβλεψης είναι εκτός του πλαισίου αυτής της διπλωματικής. Έτσι, η λογική πίσω από τον αλγόριθμο ήταν η εξής: Όταν απαιτείται η εύρεση της τιμής της ενέργειας για κάποια ώρα Χ κάποιας ημέρας, η συνάρτηση πρόβλεψης της ώρας θα επιστρέψει το μέσο όρο από τις 2 πιο πρόσφατες τιμές ενέργειας για την ώρα Χ. Παραδείγματος χάριν, εάν απαιτείται η εύρεση της τιμής της ενέργειας για την αυριανή μέρα στις 14:00, ο αλγόριθμος θα βρει τις 2 πιο πρόσφατες τιμές της ενέργειας που διαθέτει για την ώρα 14:00, και θα επιστρέψει το μέσο όρο αυτών.

Προφανώς, ο αλγόριθμος αυτός δεν επιστρέφει τις καλύτερες δυνατές προβλέψεις, ωστόσο σε μερικές περιπτώσεις δίνει πολύ καλά αποτελέσματα. Εναλλακτικοί αλγόριθμοι για εξαγωγή καλύτερων προβλέψεων θα μπορούσαν εύκολα να αντικαταστήσουν αυτόν τον αλγόριθμο του συστήματος, με απλή αντικατάσταση της συνάρτησης εύρεσης μελλοντικών τιμών ενέργειας.

Αρχιτεκτονική του συστήματος

Η αρχιτεκτονική ενός συστήματος λογισμικού αφορά την οργάνωση των επιμέρους εξαρτημάτων του συστήματος, την αλληλεπίδραση που έχουν μεταξύ τους, καθώς και το περιβάλλον μέσα στο οποίο βρίσκονται. Ο κύριος στόχος κατά τη σχεδίαση της αρχιτεκτονικής ενός συστήματος λογισμικού είναι η εύρεση και καταγραφή της κύριας δομής και των διαφόρων απαιτήσεων του εν λόγω συστήματος, χωρίς ωστόσο να δίνεται σημασία σε λεπτομέρειες υλοποίησης. Κατά τη σχεδίαση της αρχιτεκτονικής, χρησιμοποιούνται διαγράμματα για την οπτικοποίηση των επιμέρους δομικών στοιχείων του συστήματος.

Στην εικόνα 5 φαίνεται η βασική δομή του συστήματος που υλοποιήθηκε στο πλαίσιο αυτής της διπλωματικής εργασίας. Κάθε ένα από τα πλαίσια που παρουσιάζονται στην εικόνα 5 θεωρείται ότι είναι ένα δομικό στοιχείο / Component / εξάρτημα του συστήματος.



Εικόνα 5: Διάγραμμα αρχιτεκτονικής του συστήματος

Ξεκινώντας, λοιπόν, την ανάλυση αυτού του διαγράμματος, ένα από τα βασικά Components του συστήματος είναι το Backend. Όπως, φαίνεται στο διάγραμμα, το Backend αποτελείται από 6 μικρότερα εξαρτήματα, τα οποία αποκαλούνται εφαρμογές. Κάθε μία από αυτές τις εφαρμογές έχει ως στόχο την ικανοποίηση ενός συνόλου αναγκών που σχετίζονται με αυτές. Έτσι, λοιπόν, η εφαρμογή users app επικεντρώνεται στην αυθεντικοποίηση και στο χειρισμό των δεδομένων των χρηστών του συστήματος, δηλαδή εν προκειμένω των operators του σταθμού και των ιδιοκτητών ηλεκτρικών οχημάτων που επιθυμούν να φορτίσουν στους σταθμούς. Η εφαρμογή gridprice app χειρίζεται την τιμή της ηλεκτρικής ενέργειας, λαμβάνοντας υπόψιν ότι διαφορετικές περιοχές μπορεί να έχουν διαφορετικές τιμές ηλεκτρικής ενέργειας. Στη συνέχεια, η εφαρμογή chargers app, επικεντρώνεται στο χειρισμό απαιτήσεων που έχουν να κάνουν με τους φορτιστές ενός σταθμού φόρτισης ηλεκτρικών οχημάτων και στην παραγωγή των δυναμικών τιμών βάσει των στρατηγικών τιμολόγησης που αναπτύχθηκαν σε προηγούμενο υποκεφάλαιο. Η εφαρμογή stations app επικεντρώνεται στη διαχείριση διαφόρων δεδομένων που σχετίζονται με τους σταθμούς φόρτισης ηλεκτρικών οχημάτων, όπως παραδείγματος χάριν είναι η τοποθεσία του σταθμού στο χάρτη, η διεύθυνση, το όνομα, το τηλέφωνο του σταθμού κτλ. Ακολούθως, η εφαρμογή reservations app επικεντρώνεται σε κρατήσεις που γίνονται προς φορτιστές ενός σταθμού. Τέλος, η εφαρμογή statistics app στοχεύει στο να παράγει χρήσιμα δεδομένα για τους operators ενός

σταθμού, που μπορούν να χρησιμοποιηθούν για παραγωγή γραφημάτων, με στόχο την καλύτερη κατανόηση από την πλευρά των operators της λειτουργίας, των εσόδων και πιθανών προβλημάτων που μπορεί να έχει ένας σταθμός φόρτισης. Επιπλέον ανάλυση σχετικά με το Backend θα γίνει στα επόμενα δύο υποκεφάλαια που επικεντρώνονται στο σχεδιασμό αυτού.

Το άνω αριστερά component, που ονομάζεται Grid Price Collector, χρησιμοποιείται για τη συλλογή δεδομένων σχετικά με τις τιμές της ηλεκτρικής ενέργειας σε διαφορετικές περιοχές. Στο πλαίσιο αυτής της διπλωματικής το component αυτό συγκεντρώνει δεδομένα κάνοντας κλήσεις REST API στο εξωτερικό σύστημα Fingrid API, που φαίνεται στα δεξιά του. Fingrid είναι ο εθνικός φορέας δικτύου μεταφοράς ηλεκτρικής ενέργειας της Φιλανδίας, και στο πλαίσιο της διπλωματικής χρησιμοποιήθηκε το ανοιχτό API που παρέχει, ώστε να λαμβάνεται η τιμή της ηλεκτρικής ενέργειας. Το πλαίσιο που βρίσκεται πάνω από το Fingrid API, που είναι χρωματισμένο με αχνό κίτρινο χρώμα, συμβολίζει άλλα εξωτερικά συστήματα από όπου το σύστημά θα μπορούσε να παίρνει δεδομένα σχετικά με τις τιμές της ηλεκτρικής ενέργειας. Ωστόσο, στο πλαίσιο αυτής της διπλωματικής δεν υλοποιήθηκε κάποια άλλη σύνδεση με εξωτερικό σύστημα πέρα από αυτή με το Fingrid.

Ακολούθως, ένα επίσης σημαντικό κομμάτι του συστήματος είναι το Charging Operator Frontend. Στην ουσία αυτό το component είναι το Frontend της εφαρμογής που χρησιμοποιούν οι Operators των σταθμών, δηλαδή το presentation layer του συστήματος. Το Charging Operator Frontend επικοινωνεί με το backend και ανταλλάσσουν δεδομένα μέσω REST API calls, μέσω συγκεκριμένων API endpoints που υπάρχουν στο backend. Οι operators των σταθμών φόρτισης, μπορούν να χρησιμοποιούν το Charging Operator Frontend, για να ελέγχουν όλους τους σταθμούς στους οποίους είναι αρμόδιοι, να θέτουν στρατηγικές τιμολόγησης για όλα τα Pricing Groups, να ελέγχουν την κατάσταση των φορτιστών, να βλέπουν κρατήσεις φόρτισης που έχουν γίνει προς τους σταθμούς τους και να βλέπουν στατιστικά. Πρακτικά, αυτό είναι το component που δίνει στους operators τη δυνατότητα να κάνουν όλες τις ενέργειες που προσφέρει το υπό ανάλυση σύστημα. Περαιτέρω ανάλυση του Charging Operator Frontend θα γίνει σε επόμενα υποκεφάλαια, που επικεντρώνονται σε αυτό.

Παρότι το σύστημα έχει επικεντρωθεί στους operators των σταθμών φόρτισης, υλοποιήθηκε και ένα περιβάλλον Frontend και για τους ιδιοκτήτες ηλεκτρικών οχημάτων. Το EV Owner Frontend, λοιπόν, είναι το presentation layer του συστήματος, που χρησιμοποιούν οι ιδιοκτήτες ηλεκτρικών οχημάτων για να κάνουν κρατήσεις φόρτισης προς τους σταθμούς και να ελέγχουν τις πληροφορίες τους και τα οχήματά τους. Το σύστημα θα μπορούσε, ωστόσο, να συνεργάζεται και με άλλα εξωτερικά συστήματα που επικεντρώνονται στους EV Owners, τα οποία στο διάγραμμα φαίνονται με αχνό κίτρινο χρώμα, επειδή δεν έχουν υλοποιηθεί στο πλαίσιο της διπλωματικής.

Τέλος, τα δομικά στοιχεία που έχουν επισημανθεί με αχνό κίτρινο χρώμα και βρίσκονται στο κάτω δεξιά μέρος της εικόνας, συμπληρώνουν το σύστημα, ώστε να γίνεται πλήρως λειτουργικό. Αυτό είναι το Stations component, το οποίο περιέχει IoT συσκευές που βρίσκονται στο σταθμό και από τους ίδιους τους φορτιστές. Αυτά, με βάση το σχεδιασμό που έγινε, θεωρούνται εξωτερικά συστήματα και θα μπορούσαν να χρησιμοποιηθούν ώστε να ενημερώνεται αυτόματα το σύστημα σχετικά με το health των φορτιστών, την ακριβή ώρα άφιξης και αναχώρησης αυτοκινήτων και άλλα. Στο πλαίσιο αυτής της διπλωματικής δεν υλοποιήθηκαν αυτά τα components. Ωστόσο, το backend παρέχει API endpoints όπου θα μπορούσαν αυτά τα components να χρησιμοποιούν για να είναι πλήρως λειτουργικά.

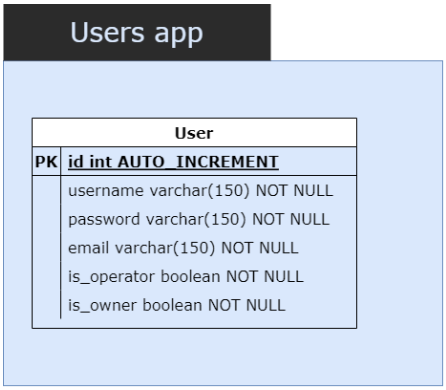
Σχεδιασμός ER διαγράμματος για το Back End

Το ER διάγραμμα (Entity Relationship Diagram), ή διάγραμμα Οντοτήτων-Συσχετίσεων στα ελληνικά, είναι ένας τύπος διαγράμματος που έχει ως στόχο να αναδείξει τον τρόπο που οντότητες, όπως άνθρωποι, αντικείμενα ή έννοιες συσχετίζονται μεταξύ τους σε ένα σύστημα. Συνήθως, τα ER διαγράμματα χρησιμοποιούνται για τη σχεδίαση σχεσιακών βάσεων δεδομένων σε συστήματα λογισμικού. Παρακάτω θα χρησιμοποιηθούν ER διαγράμματα για την περιγραφή της δομής του back end database.

Λαμβάνοντας υπόψιν το διάγραμμα της αρχιτεκτονικής του συστήματος που αναλύθηκε στο προηγούμενο υποκεφάλαιο, αρχικά, θα αναλυθούν τα ER διαγράμματα για κάθε εφαρμογή του back end ξεχωριστά και, στη συνέχεια, θα γίνει σύνδεση αυτών των υποδιαγραμμάτων σε ένα ενιαίο ER διάγραμμα που αφορά όλο το back end.

ER διάγραμμα για το users app

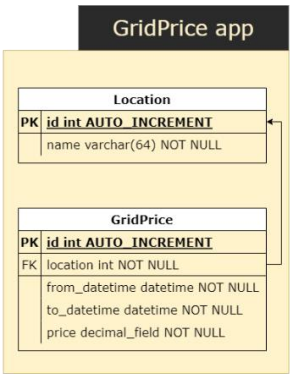
Ξεκινώντας, λοιπόν, με το users app, όπως φαίνεται στην εικόνα 6, για την εφαρμογή αυτή θα δημιουργηθεί ένα μόνο database table που θα επικεντρώνεται στην διαχείριση των χρηστών της εφαρμογής. Να σημειωθεί ότι εκτός των πεδίων id, username, password, email που είναι συνηθισμένα σε ένα table που διαχειρίζεται χρήστες, έχουν προστεθεί τα πεδία is_operator και is_owner που χρησιμοποιούνται για να επιδεικνύουν αν κάποιος χρήστης είναι χειριστής σταθμών (operator) ή ιδιοκτήτης οχήματος (owner). Αυτά τα πεδία είναι χρήσιμα, καθώς η εφαρμογή θα πρέπει να χειριστεί διαφορετικά αυτές τις δύο κατηγορίες χρηστών.



Εικόνα 6: ER diagram για το users app

ER διάγραμμα για το gridprice app

Στην εικόνα 7, φαίνεται το ER diagram για το gridprice app, που περιέχει δύο database tables. Το table που ονομάζεται Location έχει προκύψει λόγω του γεγονότος ότι η τιμή της ενέργειας σε διαφορετικές γεωγραφικά περιοχές είναι πιθανώς όχι η ίδια. Έτσι, με τη δημιουργία αυτού του table, γίνεται με σωστό τρόπο διαχείριση των τιμών της ενέργειας για διαφορετικές περιοχές. Το table που ονομάζεται GridPrice επικεντρώνεται με την διαχείριση των τιμών ηλεκτρικής ενέργειας και περιέχει πεδία τέτοια ώστε να αποθηκεύονται επιτυχώς τιμές για διαφορετικές χρονικές στιγμές (με πεδία που δηλώνουν από πότε μέχρι πότε χρεώνεται μία τιμή) και για διαφορετικές περιοχές (μέσω του πεδίου location που χρησιμοποιεί ως Foreign Key ένα Location).

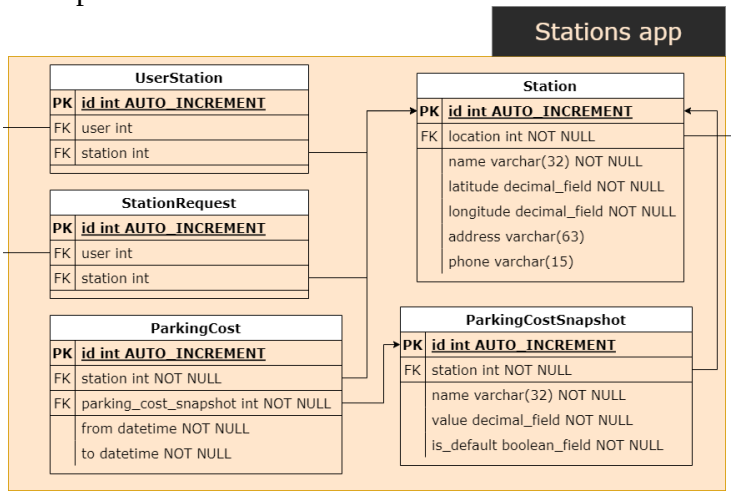


Εικόνα 7: ER diagram για το gridprice app

ER διάγραμμα για το stations app

Στη συνέχεια, ακολουθεί το ER διάγραμμα για το stations app, που φαίνεται στην εικόνα 8. Αυτό περιλαμβάνει τα εξής database tables:

- Station: Table για αποθήκευση των στοιχείων σχετικά με τους σταθμούς, όπως το όνομα, οι συντεταγμένες, η διεύθυνση και το τηλέφωνο. Επίσης, περιλαμβάνεται την τοποθεσία (location) του σταθμού, που δημιουργεί ένα Foreign Key προς το table Location του gridprice app.
- UserStation: Αντιπροσωπεύει μία σχέση Many to Many μεταξύ των User και Station tables και στοχεύει να δείξει ότι ένας σταθμός θα έχει κάποιους operators, και κάθε operator θα έχει κάποιους σταθμούς που θα διαχειρίζεται
- StationRequest: Αντιπροσωπεύει μία σχέση Many to Many μεταξύ των User και Station και χρησιμοποιείται όταν ένας operator ζητήσει να αποκτήσει πρόσβαση σε έναν άλλο σταθμό που ήδη υπάρχει.
- ParkingCostSnapshot και ParkingCost: Tables που σχετίζονται με το κόστος στάθμευσης ανά ώρα που θα χρεώνει κάποιος σταθμός προς όλα τα οχήματα που θα χρησιμοποιούν το χώρο του. Να σημειωθεί ότι έγινε διαχωρισμός σε δύο tables, ώστε το ένα να επικεντρώνει στην διαχείριση της τιμής και του default ή όχι κόστος parking ενός σταθμού (ParkingCostSnapshot), ενώ το άλλο να επικεντρώνει σε ποιες ημερομηνίες και ώρες θα ισχύει αυτό το κόστος. Αυτό έχει το πλεονέκτημα ότι όταν ένα κόστος parking κάποιου σταθμού πρέπει να επαναληφθεί σε διαφορετικές ημερομηνίες, θα είναι πιο εύκολο να επαναχρησιμοποιηθούν τα ίδια ParkingCostSnapshots.



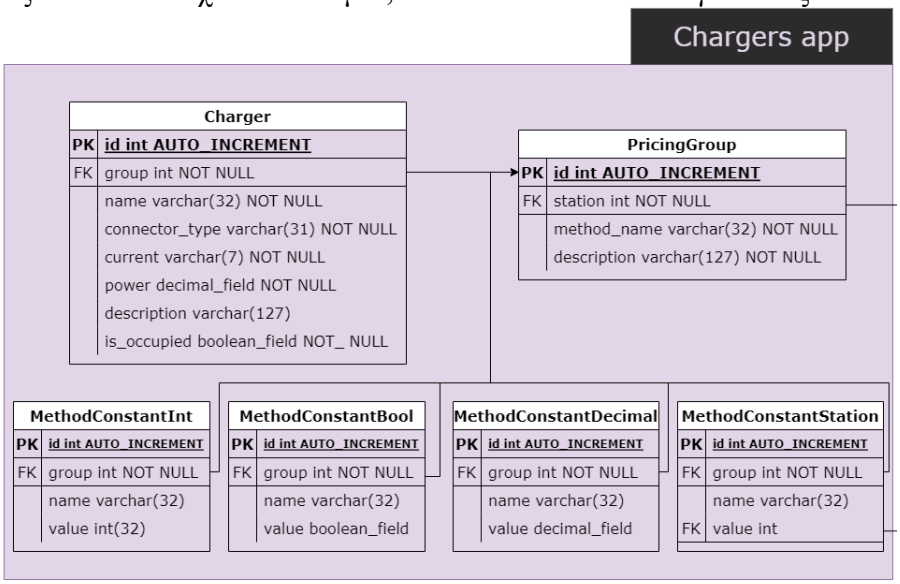
Εικόνα 8: ER diagram για το stations app

ER διάγραμμα για το chargers app

Όπως αναφέρθηκε και στο υποκεφάλαιο με τις στρατηγικές τιμολόγησης ενέργειας ανά KWh, για την καλύτερη διαχείριση των σταθμών, οι φορτιστές κάθε σταθμού έχουν οργανωθεί σε charger groups ή αλλιώς Pricing Groups, και κάθε τέτοια ομάδα φορτιστών πρέπει να ακολουθεί μία από τις τέσσερις μεθόδους τιμολόγησης που αναπτύχθηκαν.

Έτσι, το ER διάγραμμα του chargers app που περιλαμβάνει τους φορτιστές και τα Pricing Groups φαίνεται στην εικόνα 9. Αυτό περιλαμβάνει τα εξής database tables:

- Charger: Table για αποθήκευση των στοιχείων ενός φορτιστή όπως όνομα, τύπος σύνδεσης, τύπος ρεύματος (DC/AC), ισχύς φορτιστή, περιγραφή και αν είναι κατειλημμένος ή όχι. Εκτός αυτού περιέχει ένα Foreign Key προς το PricingGroup table που περιγράφει σε ποιο Pricing Group ανήκει ο φορτιστής.
- PricingGroup: Table για αποθήκευση των στοιχείων ενός Pricing Group, όπως το όνομα της μεθόδου τιμολόγησης, περιγραφή και σε ποιον σταθμό ανήκει.
- MethodConstantInt: Table για αποθήκευση των παραμέτρων Pricing Groups που είναι τύπου integer. Τέτοια παράμετρος είναι η παράμετρος n της μεθόδου Demand-centered Profit.
- MethodConstantBool: Table για αποθήκευση των παραμέτρων Pricing Groups που είναι τύπου boolean. Τέτοια παράμετρος είναι η παράμετρος grid price των μεθόδων Fixed Profit, Demand-centered Profit και Competitor-centered Profit, που δείχνει αν ο operator θέλει να προσμετράται η τιμή της ηλεκτρικής ενέργειας στον καθορισμό της παραμέτρου all_expenses.
- MethodConstantDecimal: Table για αποθήκευση των παραμέτρων Pricing Groups που είναι τύπου decimal/float. Τέτοιες παράμετροι είναι το c της μεθόδου Fixed Price, τα c και all_expenses της μεθόδου Fixed Profit, all_expenses, c1 και c2 της μεθόδου Demand-centered Profit και, τέλος, τα all_expenses, c1 και c2 της Competitor-centered Profit.
- MethodConstantStation: Table για αποθήκευση των παραμέτρων Pricing Groups που είναι τύπου Station, δηλαδή το πεδίο value του table είναι Foreign Key προς το table Station. Αυτό το table είναι χρήσιμο για τη μέθοδο Competitor-centered profit, που χρειάζεται να επιλεχθούν σταθμοί, οι οποίοι θα είναι ανταγωνιστές.



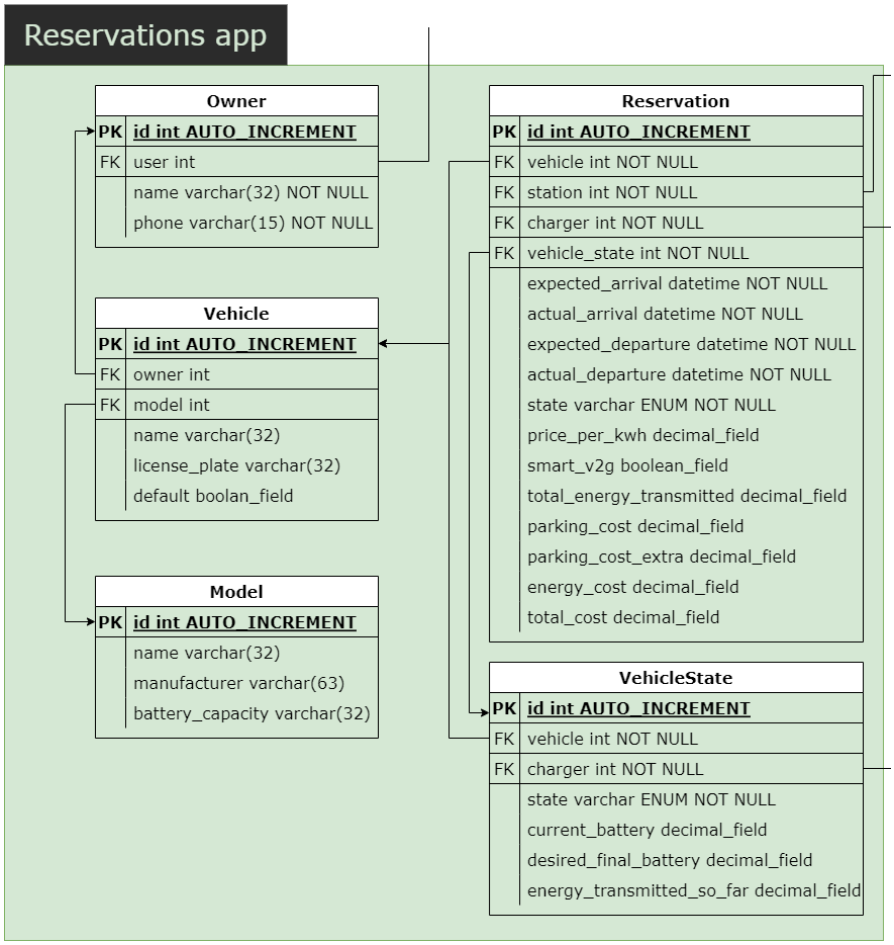
Εικόνα 9: ER diagram για το chargers app

ER διάγραμμα για το reservations app

Το reservations app χειρίζεται τις κρατήσεις που γίνονται από ιδιοκτήτες ηλεκτρικών οχημάτων προς στους σταθμούς φόρτισης. Τα tables που περιέχονται στο ER διάγραμμα αυτού του app (εικόνα 10) είναι τα εξής:

- Owner: Table για την αποθήκευση των στοιχείων των ιδιοκτητών οχημάτων. Καταγράφονται το όνομα, το τηλέφωνο καθώς, επίσης, και ένα Foreign Key προς το user table του users app.
- Model: Table για την αποθήκευση διαφορετικών μοντέλων αυτοκινήτων. Αυτό το table είναι χρήσιμο καθώς πολλοί ιδιοκτήτες έχουν ίδια μοντέλα αυτοκινήτων με ίδια χαρακτηριστικά, επομένως, θα ήταν χρήσιμο να δομηθεί με σωστό τρόπο αυτή η πληροφορία. Τα πεδία που περιέχει είναι το όνομα, ο κατασκευαστής και η χωρητικότητα μπαταρίας αυτού του μοντέλου.
- Vehicle: Table για την αποθήκευση πληροφοριών των οχημάτων. Περιλαμβάνει 2 Foreign keys προς τα Owner και Model tables, καθώς, επίσης, και τα πεδία για αποθήκευση του ονόματος, της πινακίδας και ένα boolean για τον έλεγχο αν το αυτοκίνητο αυτό είναι η προεπιλογή του owner ή όχι.
- VehicleState: Table για την αποθήκευση πληροφοριών σχετικά με την κατάσταση του οχήματος που φορτίζει στο σταθμό. Περιέχει τα εξής πεδία:
    - vehicle: Για ποιο αυτοκίνητο έχει δημιουργηθεί η κράτηση. Foreign Key προς το Vehicle table.
    - charger: Ο φορτιστής που έχει γίνει η κράτηση. Foreign Key προς το Charger table του chargers app.
    - state: Η κατάσταση του οχήματος. Τύπου ENUM, δηλαδή πρέπει να επιλεχθεί ένα από τα εξής: Charging, Canceled, Success, Failure
    - current_battery: Η μπαταρία του αυτοκινήτου τη δεδομένη χρονική στιγμή. Το πεδίο αυτό θα ενημερώνεται συνεχώς με νέες τιμές που θα στέλνονται από εξωτερικά συστήματα.
    - desired_final_battery: Η επιθυμητή τελική μπαταρία του οχήματος.
    - energy_transmitted_so_far: Η ενέργεια που έχει μεταφερθεί στο όχημα τη δεδομένη χρονική στιγμή. Ομοίως, με το πεδίο current_battery θα ενημερώνεται με νέες τιμές που θα στέλνονται από εξωτερικά συστήματα.
- Reservation: Table για την αποθήκευση πληροφοριών σχετικά με τις κρατήσεις που γίνονται προς κάποιους σταθμούς. Περιλαμβάνει τα εξής πεδία:
    - vehicle: Foreign Key προς το Vehicle table. Για ποιο αυτοκίνητο έχει δημιουργηθεί η κράτηση.
    - station: Foreign Key προς το Station table. Για ποιο σταθμό έχει δημιουργηθεί η κράτηση.
    - charger: Foreign Key προς το Charger table. Προς ποιον φορτιστή έχει γίνει η κράτηση.
    - vehicle_state: Foreign Key προς το VehicleState table. Σε τι κατάσταση βρίσκεται το αυτοκίνητο όταν φορτίζει.
    - expected_arrival: Προσδοκώμενη ώρα άφιξης του οχήματος στο σταθμό (datetime).
    - actual_arrival: Πραγματική ώρα άφιξης στο σταθμό (datetime).
    - expected_departure: Προσδοκώμενη ώρα άφιξης στο σταθμό (datetime).
    - actual_departure: Πραγματική ώρα αναχώρησης από το σταθμό (datetime).
    - state: Η κατάσταση της κράτησης. Τύπου ENUM, δηλαδή πρέπει να επιλεχθεί ένα από τα εξής: Charging, Canceled, Success, Failure, Reserved

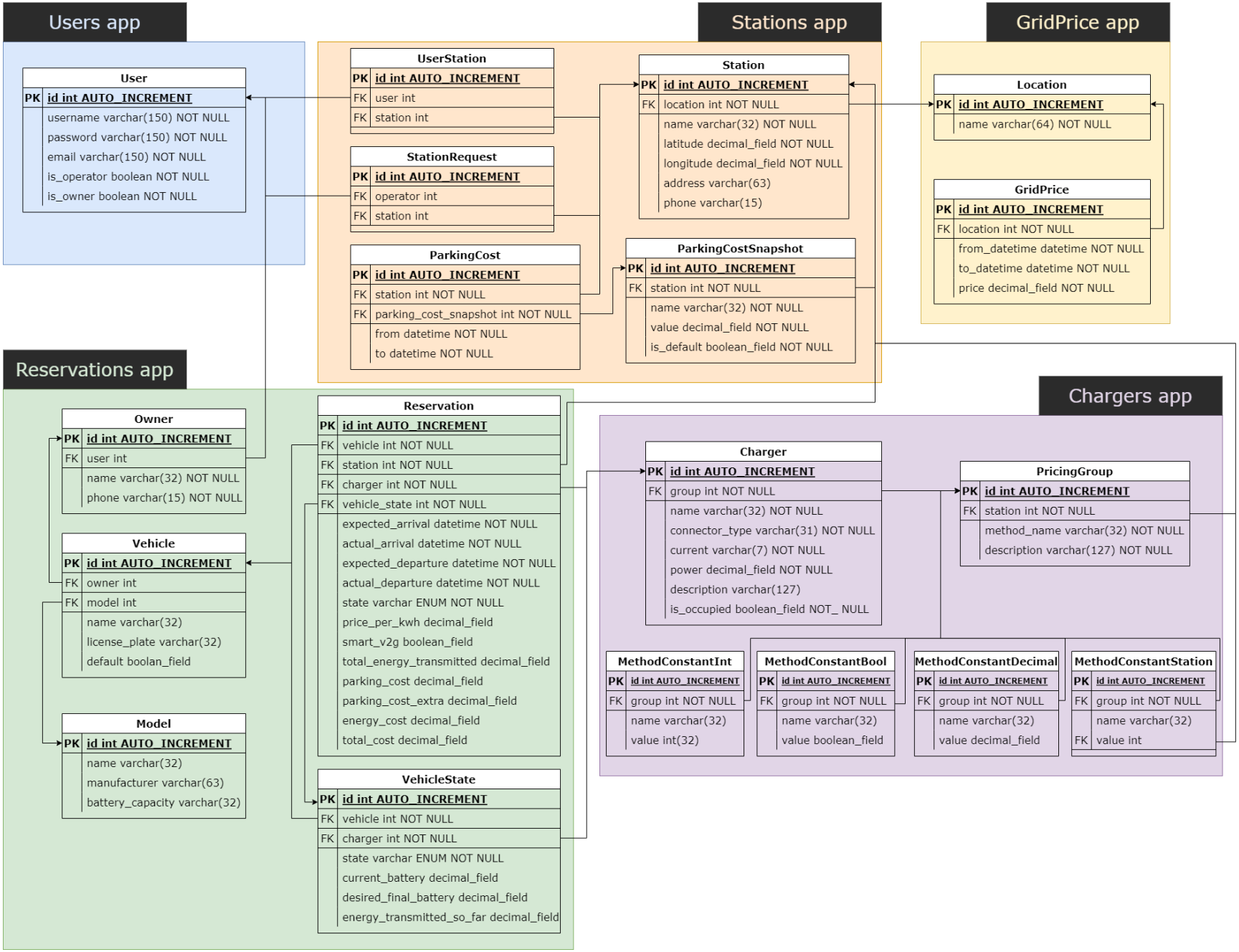
- price_per_kwh: Τιμή χρέωσης της ενέργειας (€/KWh) που ισχύει για την κράτηση.
- smart_v2g: Boolean που επιδεικνύει την πρόθεση ή όχι του ιδιοκτήτη να συμμετέχει σε Vehicle-2-Grid που ενδεχομένως γίνεται σε κάποιους σταθμούς.
- total_energy_transmitted: Ολική ενέργεια που δέχτηκε το όχημα καθόλη τη διάρκεια. Υπολογίζεται στο τέλος της φόρτισης.
- parking_cost: Κόστος parking που επιφέρεται για όλη τη διάρκεια που παρέμεινε το αυτοκίνητο στο σταθμό.
- parking_cost_extra: Επιπλέον κόστος parking που μπορεί να έχει επιβληθεί, για κάποιον λόγο.
- energy_cost: Κόστος ενέργειας που προκύπτει από τον τύπο:
  price_per_kwh * total_energy_transmitted
- total_cost: Ολικό κόστος της κράτησης που προκύπτει με πρόσθεση των προηγούμενων



Εικόνα 10: ER diagram για το reservations app

Συγκεντρωτικό ER διάγραμμα

Ενώνοντας τα προηγούμενα ER διαγράμματα όλων των apps σε ένα ενιαίο διάγραμμα, προκύπτει το διάγραμμα που φαίνεται στην εικόνα 11. Σε αυτήν την εικόνα φαίνονται και τα διάφορα Foreign Keys που υπάρχουν μεταξύ database tables διαφορετικών apps. Να σημειωθεί ότι το statistics app δεν έχει tables, αφού παράγει στατιστικά δεδομένα που συγκεντρώνονται από τα tables άλλων apps.



Εικόνα 11: Συγκεντρωτικό ER διάγραμμα για όλα τα apps του backend

Σχεδιασμός των API Endpoints του Back End

Στο προηγούμενο υποκεφάλαιο αναπτύχθηκαν τα ER διαγράμματα σχετικά με τα διάφορα apps του backend. Μέσω αυτών των διαγραμμάτων έγινε κατανοητό το τι πληροφορίες χειρίζεται και αποθηκεύει το backend του συστήματος. Ωστόσο, για να αποθηκεύεται οποιαδήποτε πληροφορία στο database του backend, πρέπει να δημιουργηθεί κάποια διεπαφή που θα μπορούν να χρησιμοποιούν άλλα components, όπως το Frontend, ώστε να στέλνουν αιτήματα για αλλαγές στα δεδομένα του backend database.

Συγκεκριμένα, πρέπει να γίνει η σχεδίαση κάποιας διεπαφής προγραμματισμού εφαρμογών, ή αλλιώς Application Programming Interface (API). Σε αυτήν πρέπει να οριστούν συγκεκριμένα άκρα επικοινωνίας (endpoints), τα οποία θα εκθέτει (expose) το backend προς άλλα συστήματα (πχ frontend) για να τα χρησιμοποιούν.

Παρακάτω θα αναλυθούν τα API endpoints που σχεδιάστηκαν για κάθε ένα από τα 6 apps του backend. Για κάθε ένα από αυτά δίνεται ένα URL, καθώς και μια περιγραφή για το τι δεδομένα απαιτούνται και θα επιστραφούν. Να σημειωθεί ότι σε όλα αυτά τα endpoints δεν γράφεται το BASE_URL, δηλαδή το σταθερό μέρος του URL, το οποίο είναι ίδιο σε όλα τα requests. Επίσης, να σημειωθεί ότι στόχος αυτού του υποκεφαλαίου δεν είναι τόσο το να αναλυθούν εκτενώς όλα τα endpoints και τι δεδομένα χρειάζονται για να συμπεριφέρονται σωστά, όσο το να γίνει μία σύντομη περιγραφή των endpoints που πρέπει να δημιουργηθούν ώστε να ανταλλάσσονται επιτυχώς δεδομένα μεταξύ του frontend και του backend. Έτσι, λεπτομέρειες υλοποίησης, όπως η HTTP μέθοδος, τα HTTP headers και η εκτενής περιγραφή του τι θα ζητείται και τι θα επιστρέφεται από κάθε αίτηση στα endpoints, έχουν παραλειφθεί σκοπίμως.

API endpoints για το users app

Το users app θα εκθέσει τα ακόλουθα endpoints για να χρησιμοποιηθούν από το Frontend ή, ενδεχομένως, άλλα συστήματα:

- */register/* | Εγγραφή ενός νέου χρήστη στο σύστημα.
- */login/* | Αυθεντικοποίηση ενός χρήστη στο σύστημα και επιστροφή ενός API token, το οποίο μπορεί να χρησιμοποιήσει ο χρήστης για δημιουργία requests που απαιτούν ταυτοποίηση.
- */logout/* | Αποσύνδεση ενός χρήστη από το σύστημα. Διαγραφή του API token από το database της εφαρμογής.
- */validate-token/* | Επαλήθευση του API token του χρήστη.

Από τα παραπάνω endpoints, τα login και register δεν απαιτούν να περιέχονται στοιχεία ταυτοποίησης στο request, ενώ τα logout και validate-token endpoints απαιτούν να περιέχεται ένα έγκυρο API token. Σε περίπτωση, που δεν περιέχεται ή το token δεν είναι έγκυρο, τότε τα endpoints αυτά επιστρέφουν μήνυμα ότι ο χρήστης δεν είναι εξουσιοδοτημένος (unauthorized).

API endpoints για το gridprice app

Το gridprice app θα εκθέσει τα ακόλουθα endpoints για να χρησιμοποιηθούν από το Frontend ή, ενδεχομένως, άλλα συστήματα:

- */gridprice/get/* | Επιστροφή κάποιων τιμών ηλεκτρικής ενέργειας για ένα συγκεκριμένο σταθμό ή μία περιοχή και μία συγκεκριμένη περίοδο, που δίνονται από το χρήστη.
- */gridprice/get-recent-prices /* | Επιστροφή των x πιο πρόσφατων τιμών ηλεκτρικής ενέργειας για ένα συγκεκριμένο σταθμό ή μία περιοχή. Το x, ο σταθμός και η περιοχή δίνονται από το χρήστη.
- */gridprice/locations/* | Επιστροφή όλων των διαθέσιμων περιοχών, δηλαδή των περιοχών για τις οποίες υπάρχουν τιμές ηλεκτρικής ενέργειας στο backend (αντικείμενα Location που υπάρχουν στο database)

Να σημειωθεί ότι όλα τα endpoints σε αυτό το app απαιτούν αυθεντικοποίηση χρήστη, δηλαδή να περιλαμβάνεται ένα έγκυρο API token στο request.

API endpoints για το stations app

Το stations app είναι μεγαλύτερο app σε σύγκριση με τα users και gridprice apps και, επομένως, θα εκθέσει περισσότερα endpoints. Συγκεκριμένα, τα endpoints που θα εκθέσει είναι τα εξής:

- */stations/* | Επιστροφή όλων των σταθμών ενός χρήστη, μαζί με κάποιες πληροφορίες όπως όνομα, συντεταγμένες και αριθμός φορτιστών του σταθμού.
- */stations/parking-costs/* | Επιστροφή των parking costs ενός σταθμού για μία συγκεκριμένη περίοδο.
- */stations/parking-cost/set_default/* | Καθορισμός του default parking cost για ένα σταθμό.
- */stations/markers/* | Επιστροφή των συντεταγμένων όλων των σταθμών.
- */stations/add-station/* | Αίτημα προσθήκης ενός operator στη διαχείριση ενός υπάρχοντος σταθμού (δημιουργία StationRequest).
- */stations/answer-request/* | Αποδοχή/απόρριψη ενός αιτήματος προσθήκης ενός operator σε κάποιον σταθμό, από κάποιον άλλον operator που είναι αρμόδιος για αυτόν τον σταθμό.
- */stations/requests/* | Επιστροφή όλων των StationRequest για όλους τους σταθμούς που ένας operator είναι αρμόδιος.
- */stations/personal-requests/* | Επιστροφή όλων των StationRequest που έχουν γίνει από έναν operator προς άλλους σταθμούς.
- */stations/create-station/* | Δημιουργία ενός νέου σταθμού. Το request πρέπει να περιέχει όλες τις πληροφορίες για τον νέο σταθμό, όπως το όνομα, η τοποθεσία, το τηλέφωνο και οι πληροφορίες για τους φορτιστές και τα Pricing Groups.
- */stations/get-station/* | Επιστροφή πληροφοριών για ένα συγκεκριμένο σταθμό που ζητείται από το χρήστη.

Να σημειωθεί ότι όλα τα endpoints του stations app απαιτούν να περιέχεται ένα έγκυρο API token, ώστε να είναι ταυτοποιημένος ο χρήστης που κάνει τις αιτήσεις. Επίσης, πρέπει να γίνεται έλεγχος αν ο χρήστης που κάνει τα αιτήματα (operator) είναι εξουσιοδοτημένος για την αλλαγή των δεδομένων ενός συγκεκριμένου σταθμού.

API endpoints για το chargers app

Το chargers app θα εκθέσει τα ακόλουθα endpoints:

- */chargers/pricing-groups/information/* | Επιστροφή κάποιων πληροφοριών σχετικά με όλα τα Pricing Groups ενός σταθμού που δίνεται από το χρήστη. Οι πληροφορίες αυτές περιλαμβάνουν το όνομα, τον αριθμό των κατειλημμένων φορτιστών, τον αριθμό όλων των υπαρχόντων φορτιστών και τη μέθοδο τιμολόγησης στο συγκεκριμένο Pricing Group.
- */chargers/pricing-groups/prices/* | Επιστροφή των τιμών τη δεδομένη χρονική στιγμή όλων των Pricing Groups ενός station που δίνεται από το χρήστη.
- */chargers/pricing-groups/* | Επιστροφή όλων των Pricing Groups ενός station που δίνεται από το χρήστη, μαζί με πληροφορίες για αυτά, όπως το όνομα, η μέθοδος, οι φορτιστές του κάθε group και η τιμή τη δεδομένη χρονική στιγμή.
- */chargers/pricing-group/create/* | Δημιουργία ενός Pricing Group. Το request πρέπει να περιέχει όλες τις πληροφορίες του Pricing Group, όπως η μέθοδος τιμολόγησης, και οι παράμετροι που τίθενται από το χρήστη.
- */chargers/pricing-group/update/* | Ενημέρωση των στοιχείων ενός υπάρχοντος Pricing Group. Το request πρέπει να περιέχει το id του Pricing Group που θα αλλάξει, καθώς και τη νέα μέθοδος τιμολόγησης μαζί με τις παραμέτρους αυτής, όπως τίθενται από το χρήστη.
- */chargers/pricing-group/delete/* | Διαγραφή ενός υπάρχοντος Pricing Group. Το request πρέπει να περιέχει το id του Pricing Group που θα διαγραφεί.
- */chargers/create/* | Δημιουργία ενός νέου φορτιστή για έναν συγκεκριμένο σταθμό.
- */chargers/update/* | Ενημέρωση των στοιχείων ενός υπάρχοντος φορτιστή. Το request πρέπει να περιέχει το id του φορτιστή που θα αλλάξει, καθώς και τα νέα στοιχεία του.
- */chargers/delete/* | Διαγραφή ενός υπάρχοντος φορτιστή. Το request πρέπει να περιέχει το id του φορτιστή που θα διαγραφεί.
- */chargers/get-not-healthy/* | Επιστροφή όλων των φορτιστών ενός σταθμού που η κατάστασή τους φαίνεται να μην είναι υγιείς.
- */chargers/not-healthy/* | Ορισμός ενός συγκεκριμένου φορτιστή ως μη υγιής..
- */chargers/healthy/* | Ορισμός ενός συγκεκριμένου φορτιστή ως υγιής.

Να σημειωθεί ότι όλα τα endpoints του chargers app απαιτούν να περιέχεται ένα έγκυρο API token, ώστε να είναι ταυτοποιημένος ο χρήστης που κάνει τις αιτήσεις. Επίσης, πρέπει να γίνεται έλεγχος αν ο χρήστης που κάνει τα αιτήματα (operator) είναι εξουσιοδοτημένος για την αλλαγή των στοιχείων συγκεκριμένων φορτιστών ή Pricing Groups. Παραδείγματος χάριν, ένας operator απαγορεύεται να αλλάζει τα στοιχεία ενός Pricing Group που ανήκει σε έναν σταθμό όπου δεν είναι αρμόδιος (έλεγχος μέσω του operators field του Station table).

API endpoints για το reservations app

Τα endpoints που θα εκθέσει το reservations app είναι τα εξής:

- */reservations/vehicle-states/* | Επιστροφή όλων των καταστάσεων των οχημάτων που φορτίζουν τη δεδομένη χρονική στιγμή σε έναν σταθμό.
- */reservations/vehicle-state/get/* | Επιστροφή της κατάστασης ενός συγκεκριμένου οχήματος.
- */reservations/* | Επιστροφή όλων των κρατήσεων ενός σταθμού για μια συγκεκριμένη χρονική περίοδο.
- */reservations/available-chargers/* | Επιστροφή όλων των διαθέσιμων φορτιστών ενός σταθμού για μία συγκεκριμένη περίοδο. Τέτοιοι φορτιστές είναι αυτοί που δεν υπάρχει κάποια κράτηση προς αυτούς για μία συγκεκριμένη περίοδο.

- */reservations/create/* | Δημιουργία μίας κράτησης. Το request πρέπει να περιέχει τις πληροφορίες της κράτησης, όπως ο σταθμός, ο φορτιστής, η διάρκεια φόρτισης, τα στοιχεία του οχήματος, τα στοιχεία του ιδιοκτήτη και η επιθυμία ή όχι του ιδιοκτήτη το όχημα να συμμετέχει στο έξυπνο V2G σύστημα.
- */reservations/update/* | Ενημέρωση των στοιχείων μίας κράτησης. Το request πρέπει να περιέχει το id και τις νέες πληροφορίες της κράτησης.
- */reservations/delete/* | Διαγραφή μίας συγκεκριμένης κράτησης που επιθυμεί ο χρήστης.
- */reservations/cancel/* | Ακύρωση μίας συγκεκριμένης κράτησης που επιθυμεί ο χρήστης.
- */reservations/vehicle-state/create/* | Δημιουργία ενός database VehicleState αντικειμένου, που σχετίζεται με την κατάσταση ενός οχήματος που φορτίζει στο σταθμό. Αυτό θα συμβαίνει όταν ένα όχημα φτάνει στο σταθμό και ξεκινάει τη φόρτιση. Το request πρέπει να περιέχει την μπαταρία του αυτοκινήτου κατά την άφιξη, την επιθυμητή τελική μπαταρία του αυτοκινήτου, τον χρόνο άφιξης και το id της κράτησης που έχει προηγηθεί.
- */reservations/end-reservation/* | Επιτυχής τερματισμός μίας κράτησης. Το request πρέπει να περιέχει το id της κράτησης, την ολική ενέργεια που μεταφέρθηκε στο όχημα (αριθμός KWh), την ώρα αναχώρησης και το επιπλέον κόστος parking που επιθυμεί να χρεώσει ο operator.
- */reservations/model/get-manufacturers/* | Επιστροφή όλων των ονομάτων των εταιριών κατασκευής ηλεκτρικών αυτοκινήτων.
- */reservations/model/get-models/* | Επιστροφή όλων των ονομάτων των μοντέλων μίας συγκεκριμένης εταιρίας κατασκευής ηλεκτρικών αυτοκινήτων. Το request πρέπει να περιέχει το όνομα της εταιρίας.
- */reservations/vehicles/create/* | Δημιουργία ενός νέου οχήματος για έναν EV Owner. Το request πρέπει να περιέχει τις πληροφορίες του οχήματος που θα δημιουργηθεί, όπως το μοντέλο, το όνομα και τις πινακίδες του αυτοκινήτου.
- */reservations/vehicles/delete/* | Διαγραφή ενός νέου οχήματος για έναν EV Owner. Το request πρέπει να περιέχει το id του οχήματος που πρέπει να διαγραφεί.
- */reservations/owner-create-reservation/* | Δημιουργία μίας κράτησης από την πλευρά ενός EV Owner. Το request πρέπει να περιέχει τις πληροφορίες της κράτησης, όπως το id του ιδιοκτήτη, το id του οχήματος, ο σταθμός, ο φορτιστής, η διάρκεια φόρτισης, και η επιθυμία ή όχι του ιδιοκτήτη το όχημα να συμμετέχει στο έξυπνο V2G σύστημα.
- */reservations/owner/* | Επιστροφή των πληροφοριών ενός EV Owner (του χρήστη που κάνει το συγκεκριμένο request).
- */reservations/owner/edit/* | Ενημέρωση των στοιχείων ενός EV Owner. Το request πρέπει να περιέχει πληροφορίες όπως το καινούριο όνομα και το καινούριο τηλέφωνο του EV Owner.
- */reservations/vehicles/* | Επιστροφή όλων των οχημάτων ενός EV Owner.
- */reservations/owner-reservations/* | Επιστροφή όλων των κρατήσεων που έχει δημιουργήσει ένας EV Owner.

Ομοίως με τα stations και chargers apps, τα endpoints του reservations app απαιτούν αυθεντικοποίηση του χρήστη, ώστε να είναι βέβαιο ότι ο χρήστης που κάνει τα αιτήματα έχει όντως πρόσβαση στα αντίστοιχα δεδομένα. Να σημειωθεί, επίσης, ότι τα τελευταία 9 endpoints είναι προσβάσιμα μόνο από χρήστες που είναι EV Owners (το πεδίο is_owner στο table User πρέπει είναι true), ενώ τα υπόλοιπα endpoints είναι προσβάσιμα μόνο από operators (το πεδίο is_operator στο table User πρέπει είναι true), εξαιρουμένου του endpoint

/reservations/available-chargers/, το οποίο είναι διαθέσιμο και από EV owners και από operators.

API endpoints για το statistics app

Το statistics app επικεντρώνεται στην εξαγωγή στατιστικών σχετικά με τους σταθμούς. Τα στατιστικά παράγονται σε μεγάλο βαθμό από τα reservations που γίνονται προς τους διάφορους σταθμούς. Γι' αυτό, αρκεί η δημιουργία ενός μόνο endpoint, το οποίο είναι το εξής:

- */statistics/reservations/* | Επιστροφή στατιστικών σχετικά με τις κρατήσεις ενός σταθμού, για ένα συγκεκριμένο χρονικό διάστημα που δίνεται από το χρήστη.

Το endpoint αυτό απαιτεί αυθεντικοποίηση για την ασφάλεια των δεδομένων διαφορετικών σταθμών.

Mockups για το Front End Environment

Αυτό το υποκεφάλαιο στοχεύει στην περιγραφή της σχεδίασης που έγινε για το presentation layer του συστήματος, δηλαδή τα Components Charging Operator Frontend και EV Owner Frontend, όπως περιεγράφηκαν στο υποκεφάλαιο με την αρχιτεκτονική του συστήματος.

Για το σχεδιασμό του Frontend αφιερώθηκε ένα μεγάλο ποσοστό του χρόνο στο σχεδιασμό mockups. Στην ανάπτυξη λογισμικού, mockup είναι ένας στατικός σχεδιασμός μίας σελίδας της εφαρμογής, ο οποίος αναδεικνύει τα τελικά στοιχεία που θα απαρτίζουν τη σελίδα, χωρίς όμως να είναι λειτουργική. Ο όρος στατικός σχεδιασμός σημαίνει ότι τα mockups δεν περιέχουν τη λειτουργικότητα που έχει μία πραγματική σελίδα κάποιας εφαρμογής. Παραδείγματος χάριν, τα mockups μπορεί να περιέχουν μία μπάρα πλοήγησης, ωστόσο, αν γίνει click πάνω σε κάποιο link της μπάρας, αυτό δεν θα προκαλέσει την ανακατεύθυνση σε κάποια άλλη σελίδα.

Η σχεδίαση των mockups έγινε με γνώμονα την δημιουργία μιας web εφαρμογής που θα είναι όσο το δυνατόν πιο φιλική προς τους operators των σταθμών και τους ιδιοκτήτες των οχημάτων. Με άλλα λόγια, έγινε μία προσπάθεια να κατανοηθούν οι ανάγκες αυτών των χρηστών, ώστε να δημιουργηθεί μία εφαρμογή που θα διευκολύνει τους χρήστες να αλληλοεπιδράσουν με αυτήν για να πετύχουν οποιοδήποτε στόχο έχουν. Για τη σχεδίαση των mockups χρησιμοποιήθηκε η δωρεάν web εφαρμογή Marvel App [16], που διαθέτει περιβάλλον για σχεδίαση εφαρμογών λογισμικού.

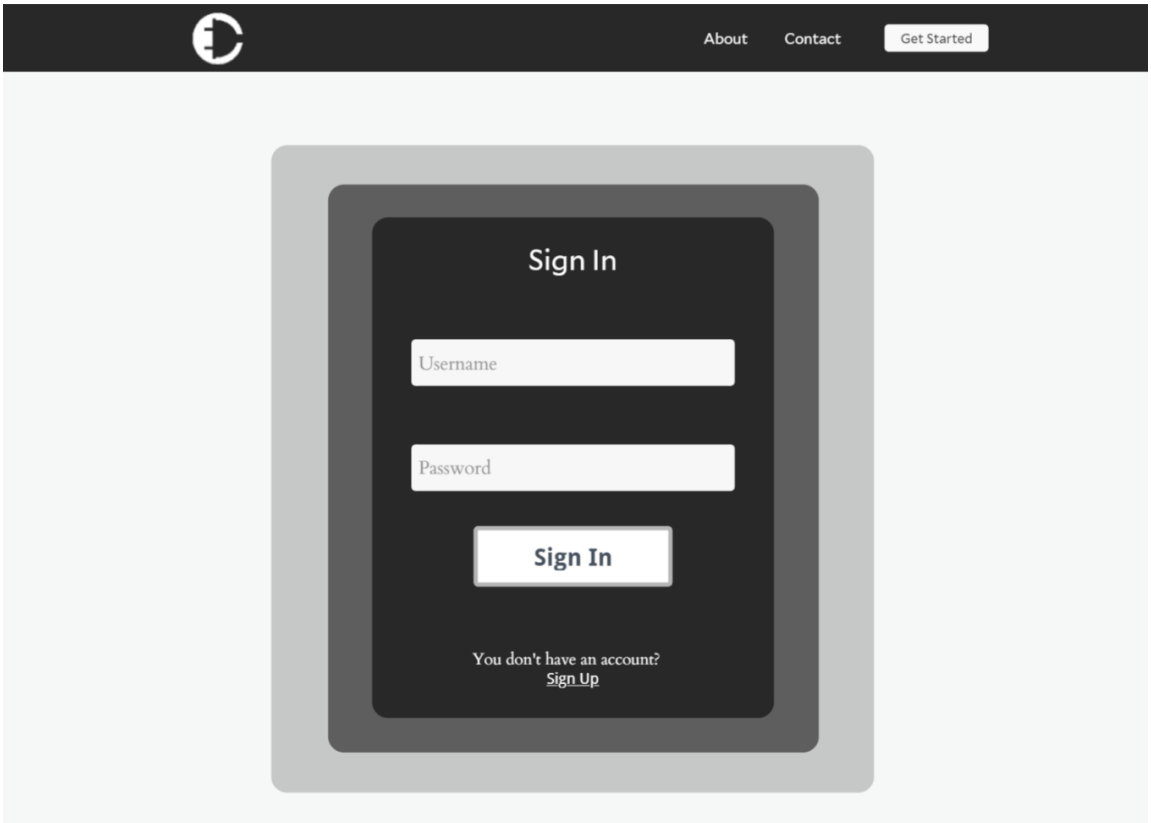
Παρακάτω θα παρουσιαστούν τα διάφορα mockups που σχεδιάστηκαν. Αρχικά, θα παρουσιαστούν mockups που είναι κοινά και για το Charging Operator Frontend και για το EV Owners Frontend, ενώ στη συνέχεια θα προβληθούν mockups των σελίδων που είναι αποκλειστικά για ένα από τα 2 περιβάλλοντα. Να σημειωθεί και πάλι ότι η διπλωματική αυτή επικεντρώνεται στην δημιουργία μίας web εφαρμογής για χειριστές (operators) σταθμών φόρτισης, και αυτός είναι ο λόγος που τα mockups για το Charging Operator Frontend είναι εμφανώς περισσότερα από αυτά του EV Owner Frontend.

Κοινά Mockups

Ξεκινώντας με τα mockups των σελίδων που είναι κοινά για όλους τους χρήστες της εφαρμογής (και operators και EV Owners), στην εικόνα 12 φαίνεται το mockup για τη σελίδα την οποία μπορούν να χρησιμοποιήσουν οι χρήστες (operators και EV Owners) για να κάνουν εγγραφή στην εφαρμογή.

Παρόμοιο φαίνεται και το mockup για τη σύνδεση των χρηστών στην εφαρμογή, που φαίνεται στην εικόνα 13.

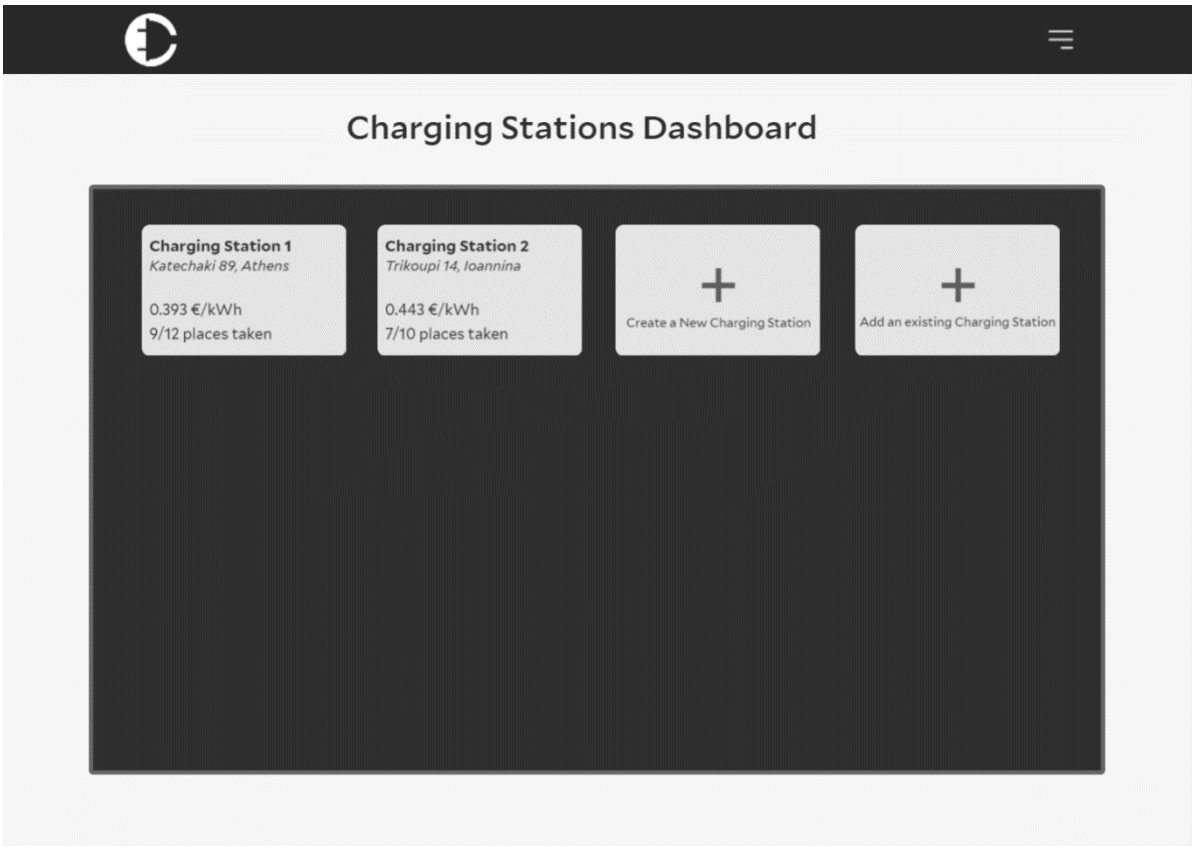
Εικόνα 12: Mock-up για τη σελίδα εγγραφής



Εικόνα 13: Mock-up για τη σελίδα σύνδεσης

Charging Operator Frontend Mockups

Στη συνέχεια, θα παρουσιαστούν τα mockups για επίσης σελίδες που θα χρησιμοποιούν οι operators των σταθμών. Στην εικόνα 14 φαίνεται η σελίδα που βλέπουν οι operators μετά τη σύνδεση τους στην εφαρμογή. Το Dashboard αυτό παρουσιάζει όλους τους σταθμούς στους οποίους είναι αρμόδιος ο συνδεδεμένος operator, καθώς επίσης μπορεί να δει και λίγες πληροφορίες για τους σταθμούς. Επίσης, μπορεί να δημιουργήσει έναν καινούριο σταθμό ή να κάνει αίτημα για να προσθέσει έναν υπάρχοντα σταθμό στη λίστα των σταθμών του.

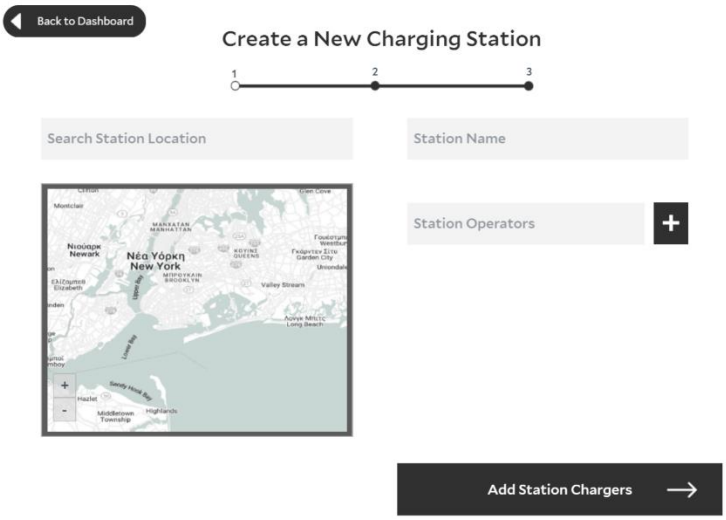


Εικόνα 14: Mock-up για τη σελίδα Dashboard των operators

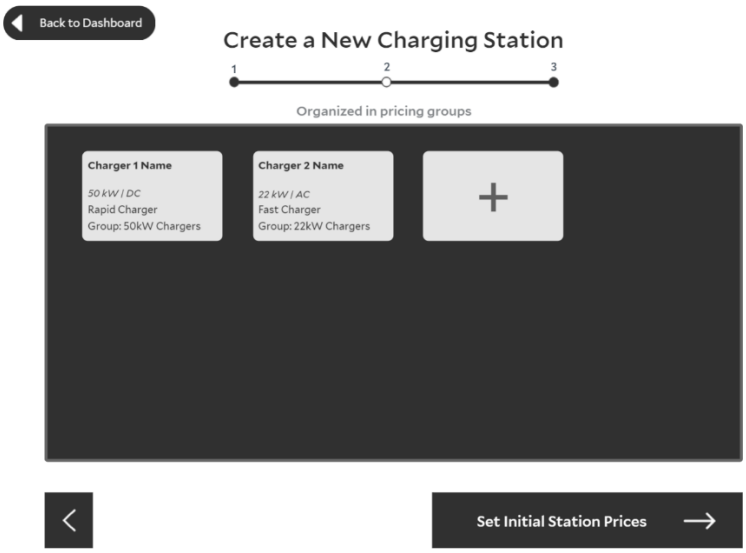
Στις εικόνες 15, 16, 17 φαίνονται τα mockups για τα 3 στάδια που πρέπει να ολοκληρώσει ένας operator για να δημιουργήσει έναν καινούριο σταθμό φόρτισης.

Το πρώτο στάδιο επικεντρώνεται στις πληροφορίες του σταθμού όπως η διεύθυνση, το όνομα, κτλ. Το δεύτερο στάδιο επικεντρώνεται στην δημιουργία φορτιστών με τη συμπλήρωση όλων των απαραίτητων στοιχείων που χρειάζονται για να περιγράψουν έναν φορτιστή, όπως το ρεύμα (AC/DC), η ισχύς, ο τύπος σύνδεσης κτλ. Το τρίτο στάδιο επικεντρώνεται στον καθορισμό των μεθόδων τιμολόγησης των Charger Groups, που αναπτύχθηκαν σε προηγούμενο υποκεφάλαιο.

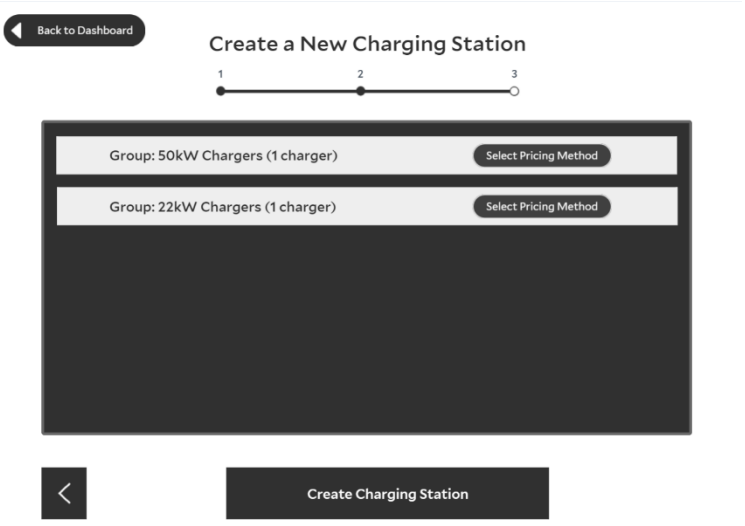
Συγκεκριμένα, για τον καθορισμό των παραμέτρων των μεθόδων τιμολόγησης, δημιουργήθηκε ένα διαφορετικό mockup που φαίνεται στην εικόνα 18.



Εικόνα 15: Mock-up για τη σελίδα δημιουργίας νέου σταθμού (βήμα 1)



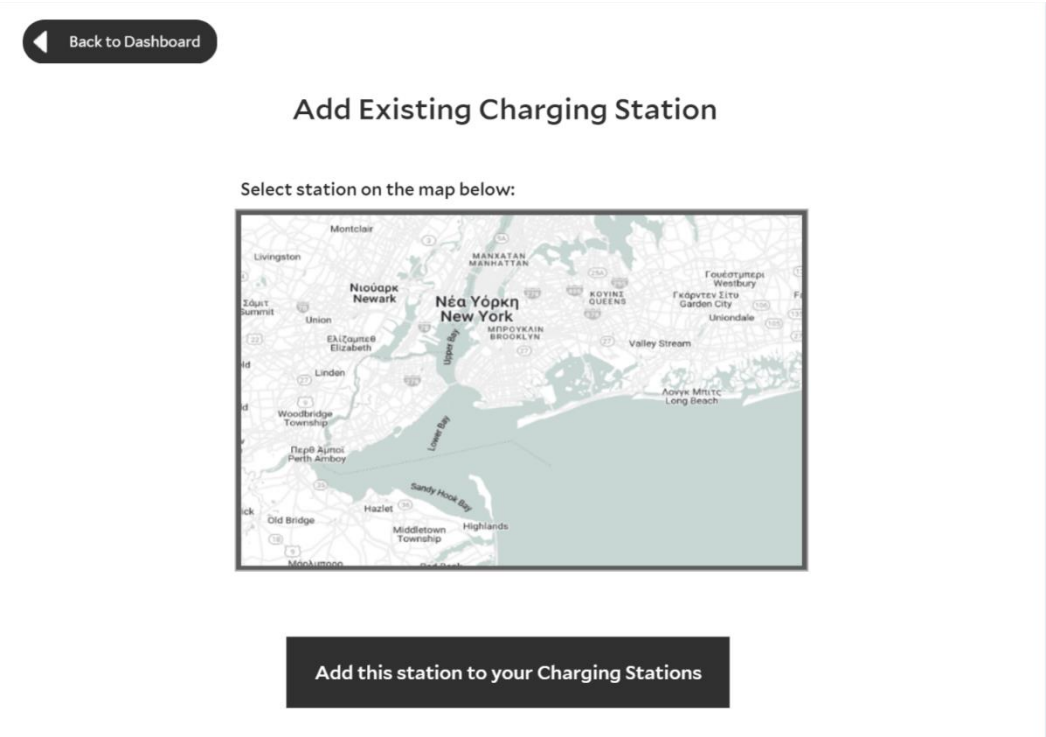
Εικόνα 16: Mock-up για τη σελίδα δημιουργίας νέου σταθμού (βήμα 2)



Εικόνα 17: Mock-up για τη σελίδα δημιουργίας νέου σταθμού (βήμα 3)

Εικόνα 18: Mock-up για το Modal καθορισμού παραμέτρων των μεθόδων τιμολόγησης

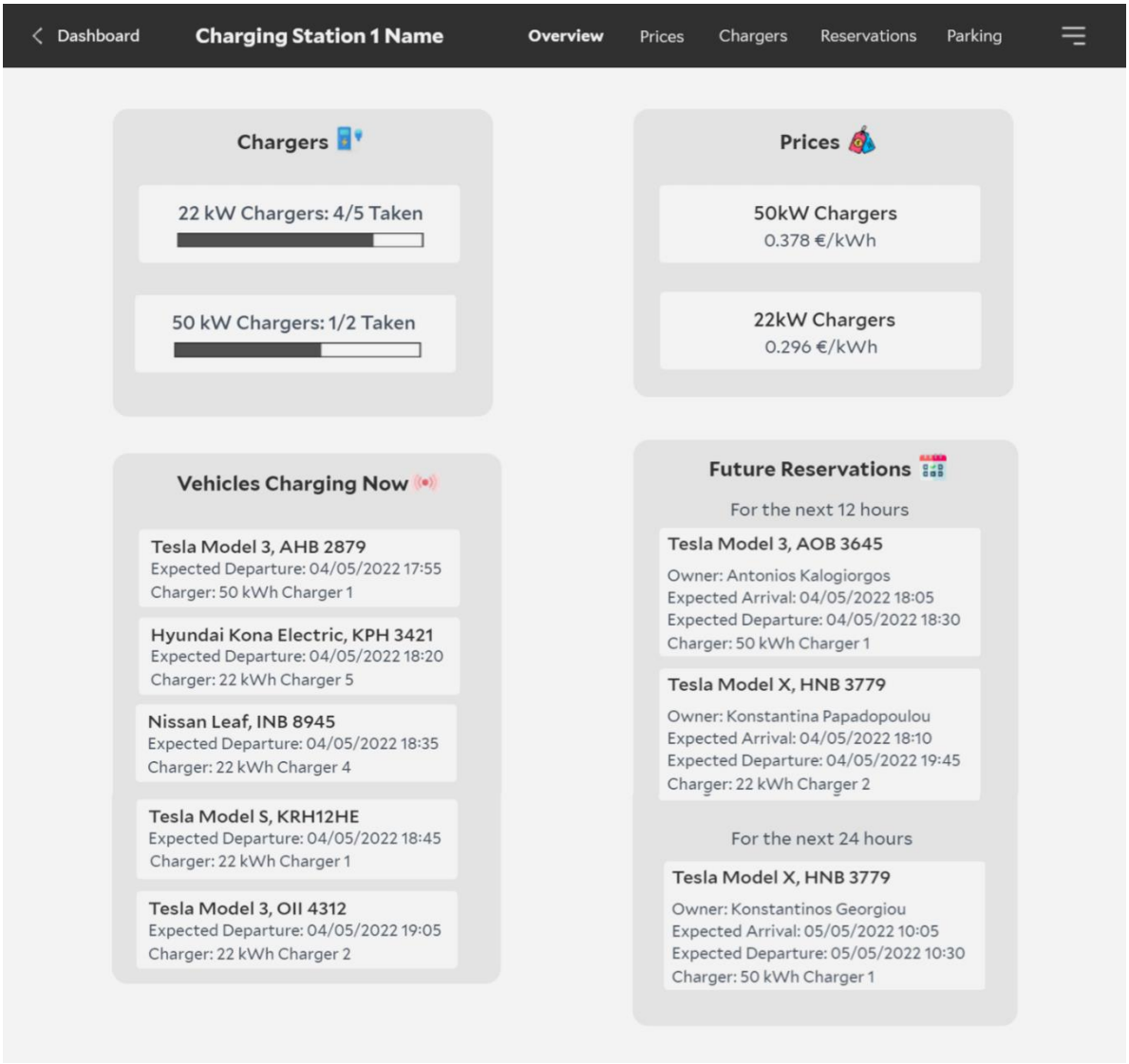
Ακολούθως, στην εικόνα 19 φαίνεται το mockup για της σελίδας αιτήματος προσθήκης υπάρχοντος σταθμού στη λίστα των σταθμών ενός Operator. Αυτή η λειτουργικότητα είναι χρήσιμη, καθώς μπορεί κάποιοι σταθμοί να έχουν παραπάνω από έναν operator.



Εικόνα 19: Mock-up για τη σελίδα προσθήκης ενός υπάρχοντος σταθμού

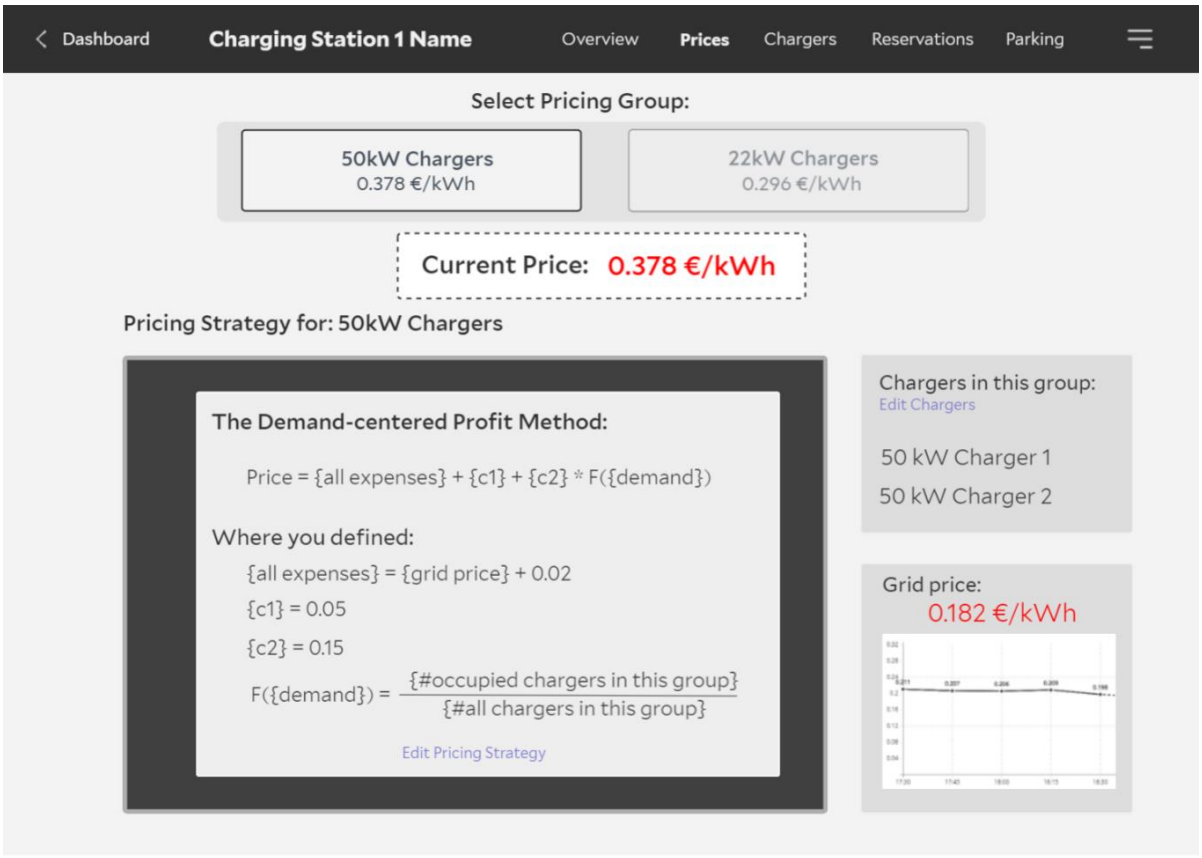
Μέχρι στιγμής παρουσιάστηκαν mockups σελίδων που δεν επιδεικνύουν σε μεγάλο βαθμό τη λειτουργικότητα της εφαρμογής. Στη συνέχεια, ωστόσο, θα προβληθούν mockups των σελίδων που σχετίζονται με την κύρια λειτουργικότητα της εφαρμογής για τους operators των σταθμών φόρτισης.

Έτσι, λοιπόν, στην εικόνα 20 φαίνεται το mockup της σελίδας επισκόπησης ενός σταθμού. Σε αυτή τη σελίδα, ο operator μπορεί να δει τα πιο σημαντικά στοιχεία για έναν σταθμό, όπως είναι ο αριθμός των γεμάτων και άδειων φορτιστών, οι τιμές πώλησης, τα αυτοκίνητα που βρίσκονται στο σταθμό και φορτίζουν, καθώς και μελλοντικές κρατήσεις φόρτισης που έχουν γίνει προς τον σταθμό.



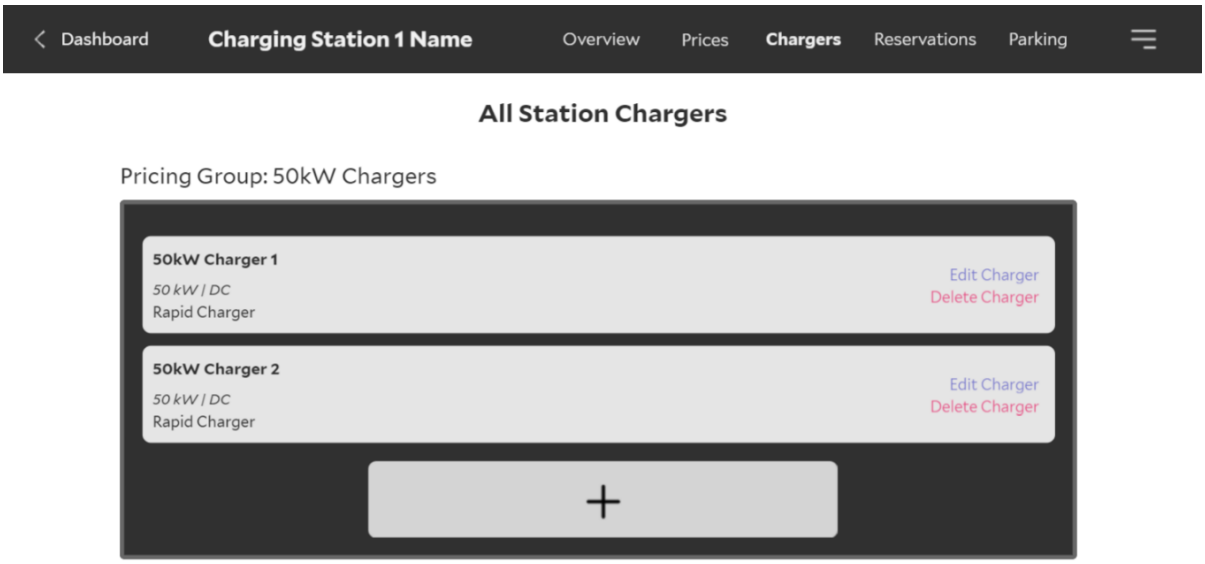
Εικόνα 20: Mock-up για τη σελίδα επισκόπησης ενός σταθμού

Στην εικόνα 21 φαίνεται το mockup της σελίδας Prices, που σχετίζεται με τις τιμές των Charger Groups, όπως καθορίζονται βάσει των παραμέτρων της επιλεγμένης μεθόδου τιμολόγησης. Επίσης, δίνει τη δυνατότητα στους operators να δουν και να αλλάξουν εύκολα τη μέθοδο τιμολόγησης, τις παραμέτρους της μεθόδου και τους φορτιστές του Charger Group.



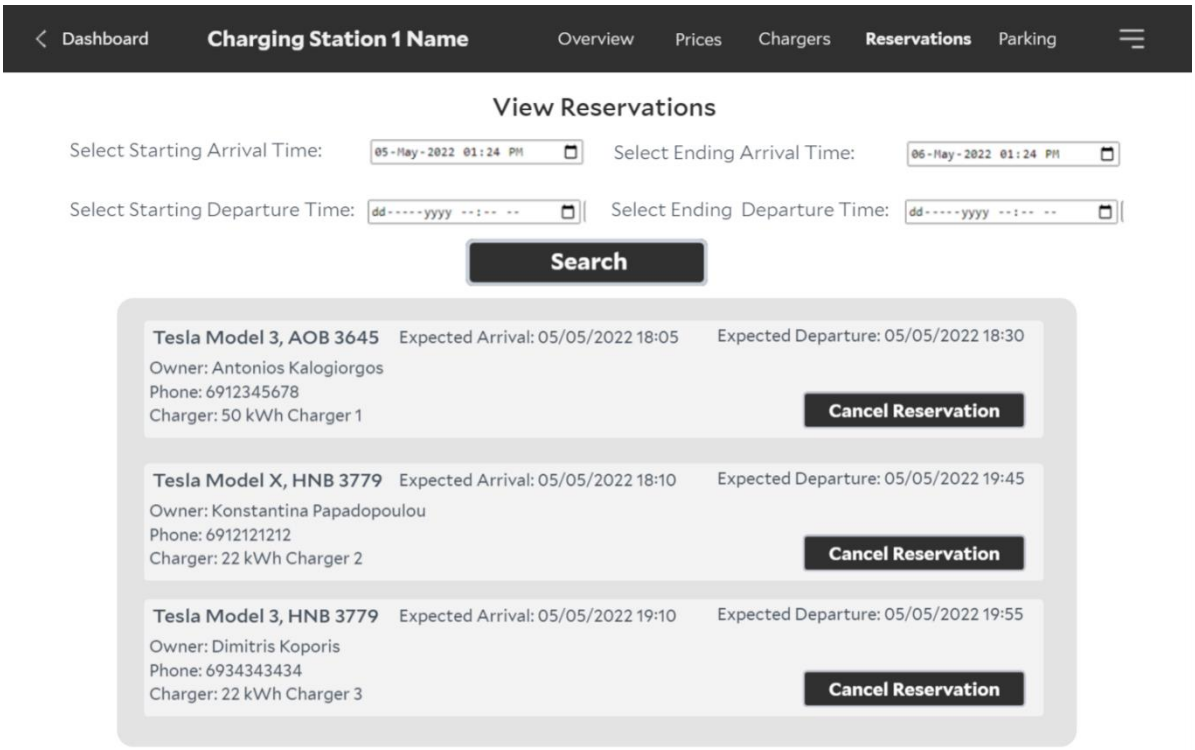
Εικόνα 21: Mock-up για τη σελίδα διαχείρισης Pricing Groups ενός σταθμού

Στην εικόνα 22 φαίνεται το mockup για τη σελίδα διαχείρισης των φορτιστών ενός σταθμού.



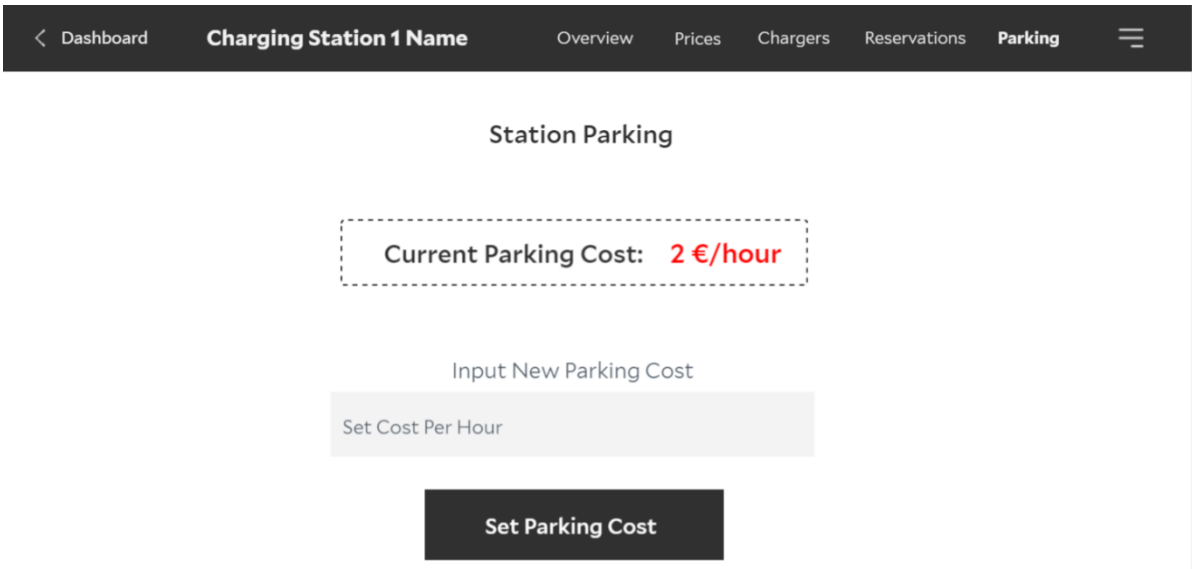
Εικόνα 22: Mock-up για τη σελίδα διαχείρισης φορτιστών ενός σταθμού

Στη συνέχεια, στην εικόνα 23 παρουσιάζεται το mockup για τη σελίδα διαχείρισης των κρατήσεων. Εκεί ο operator έχει τη δυνατότητα να κάνει σύνθετες αναζητήσεις για παρελθοντικές και μελλοντικές κρατήσεις, επίσης και να δημιουργήσει νέες κρατήσεις φόρτισης.



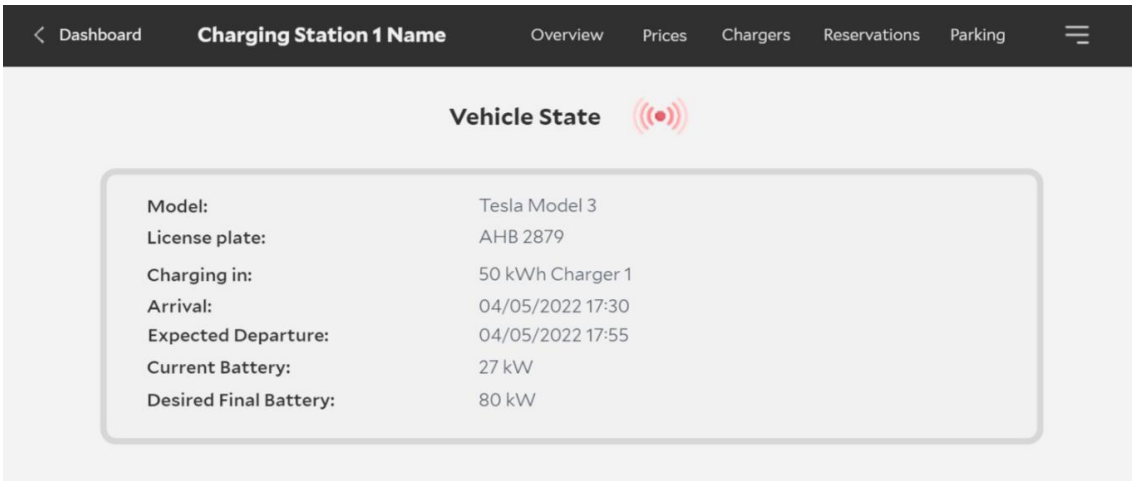
Εικόνα 23: Mock-up για τη σελίδα κρατήσεων ενός σταθμού

Προχωρώντας, ακολουθεί το mockup της σελίδας όπου ο operator ελέγχει την τιμή του parking του σταθμού.



Εικόνα 24: Mock-up για τη σελίδα Parking ενός σταθμού

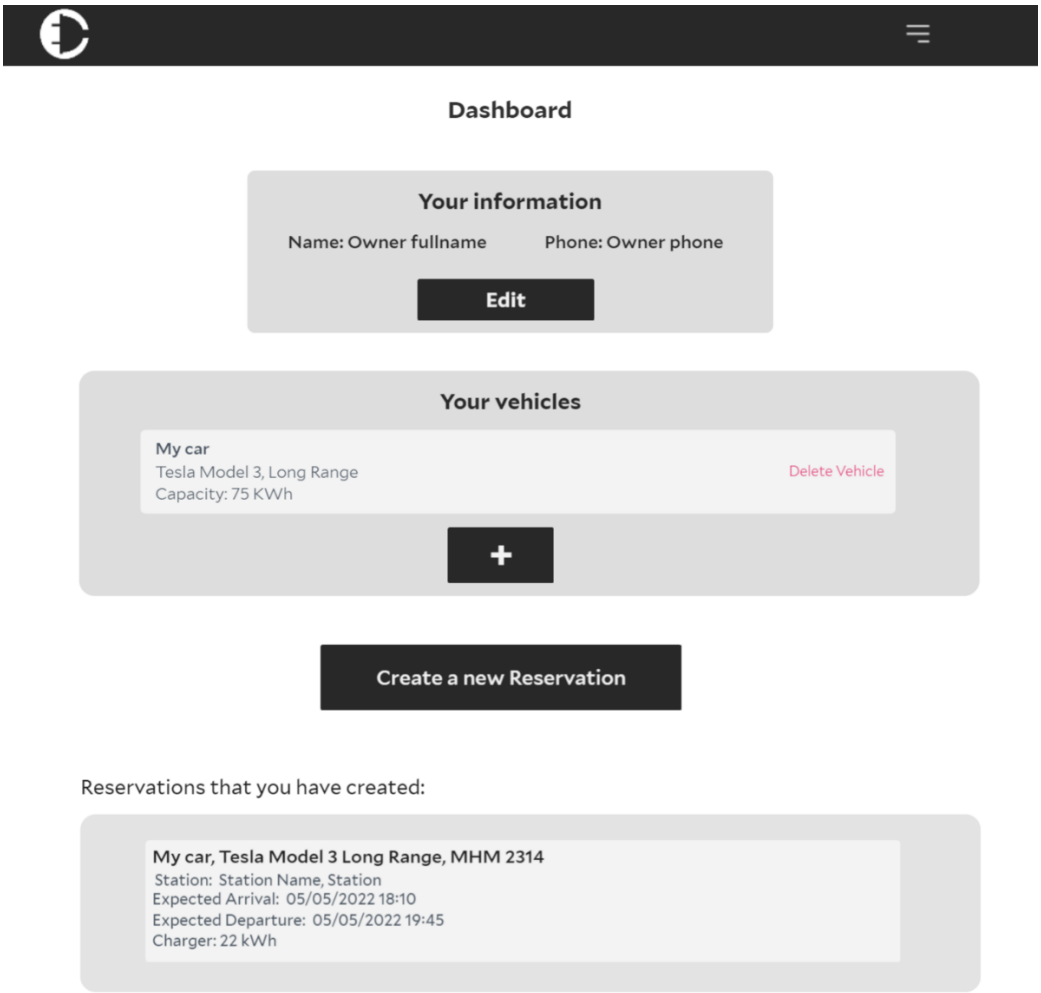
Τέλος, στην εικόνα 25 φαίνεται το mockup για τη σελίδα όπου επιδεικνύει την κατάσταση της φόρτισης ενός αυτοκινήτου. Σε αυτή τη σελίδα, μπορεί ο operator εύκολα να δει τα στοιχεία του αυτοκινήτου, σε ποιον φορτιστή γίνεται η φόρτιση, την ώρα άφιξης, την αναμενόμενη ώρα αναχώρησης, την τωρινή μπαταρία και την επιθυμητή τελική μπαταρία του οχήματος.



Εικόνα 25: Mock-up για τη σελίδα κατάστασης ενός οχήματος που φορτίζει

EV Owner Frontend Mockups

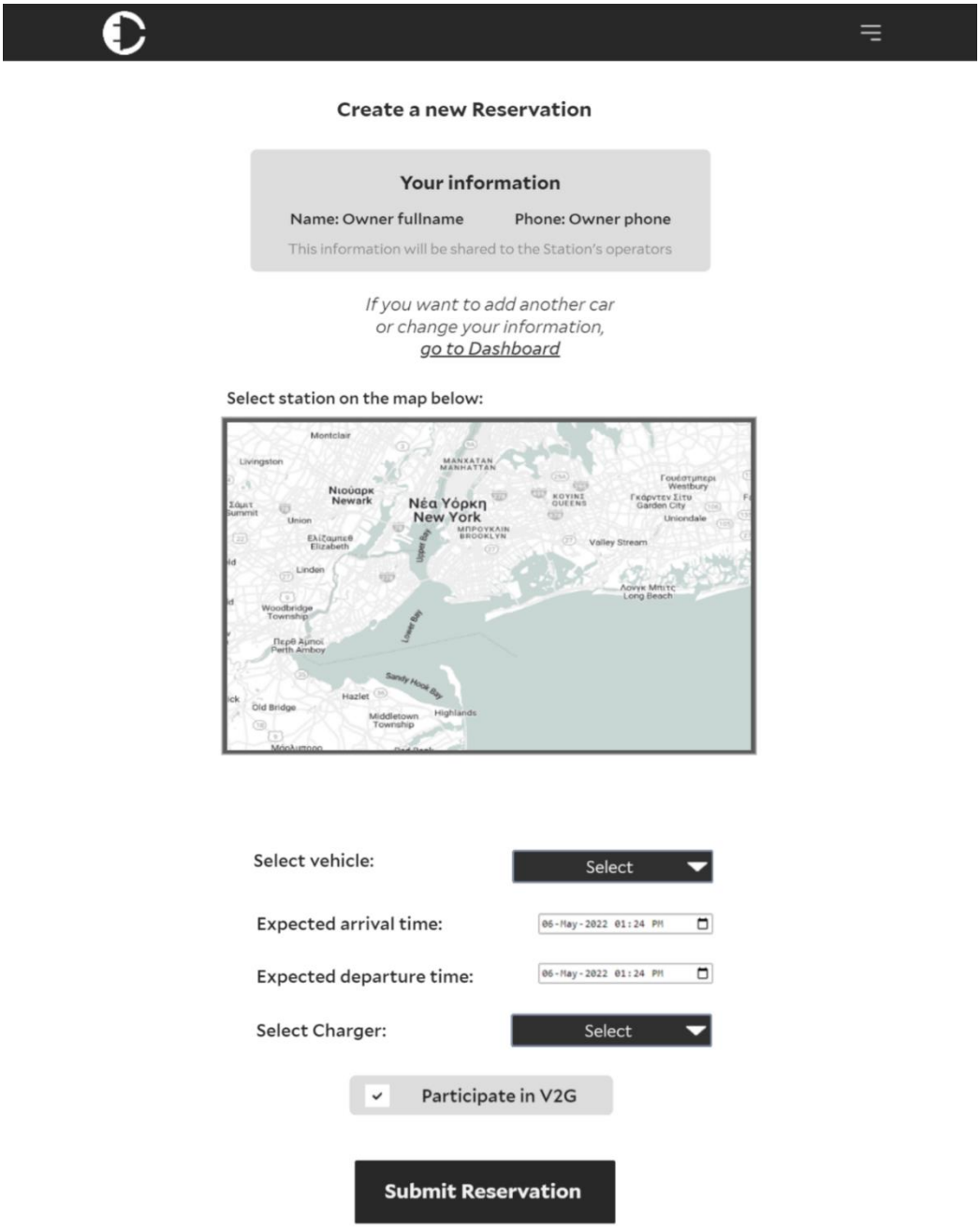
Στη συνέχεια, θα παρουσιαστούν τα mockups για τις σελίδες που θα χρησιμοποιούν οι EV owners. Το Frontend για τους ιδιοκτήτες ηλεκτρικών οχημάτων περιέχει δύο κύριες σελίδες, τα mockups των οποίων θα παρουσιαστούν παρακάτω.



Εικόνα 26: Mock-up για τη σελίδα Dashboard ενός ιδιοκτήτη οχήματος

Στην εικόνα 26 φαίνεται η σελίδα που βλέπουν οι ιδιοκτήτες ηλεκτρικών οχημάτων μετά τη σύνδεσή τους στην εφαρμογή. Στο Dashboard μπορεί ο EV Owner να αλλάξει τα στοιχεία του, να προσθέσει νέα οχήματα και να δει τις κρατήσεις φόρτισης που έχει δημιουργήσει (παρελθοντικές και μελλοντικές).

Τέλος, στην εικόνα 27 φαίνεται το mockup για τη σελίδα των κρατήσεων. Σε αυτή τη σελίδα οι ιδιοκτήτες οχημάτων, μπορούν να δημιουργήσουν κρατήσεις φόρτισης προς άλλους σταθμούς. Συγκεκριμένα, μπορούν να επιλέξουν στο χάρτη το σταθμό που επιθυμούν να κάνουν κράτηση, να επιλέξουν το όχημα που επιθυμούν, την αναμενόμενη ώρα άφιξης και αναχώρησης, το φορτιστή που επιθυμούν να φορτίσουν καθώς και το αν επιθυμούν να συμμετέχουν ή όχι σε διαδικασία έξυπνης Vehicle-to-Grid φόρτισης.



Εικόνα 27: Mock-up για τη σελίδα δημιουργίας κράτησης από ιδιοκτήτη οχήματος

Κεφάλαιο 4: Υλοποίηση της εφαρμογής

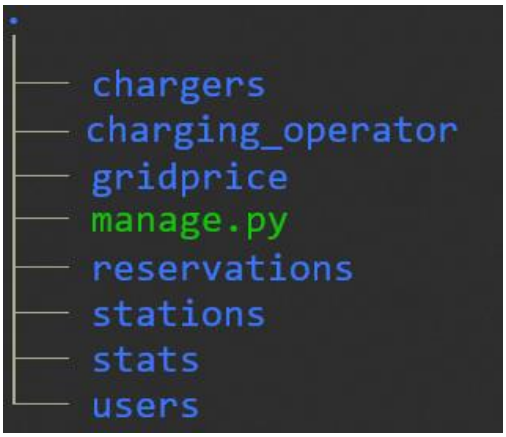
Εισαγωγή

Μετά την αρχική σχεδίαση του συστήματος, ακολουθεί η υλοποίηση του. Στο κεφάλαιο αυτό θα αναλυθούν οι μέθοδοι και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος που σχεδιάστηκε στο προηγούμενο κεφάλαιο. Να σημειωθεί ότι κατά τη διάρκεια της ανάπτυξης λογισμικού του συστήματος, πολλές φορές ήταν αναγκαίο να γίνουν διάφορες αλλαγές στο σχεδιασμό της εφαρμογής. Κάτι τέτοιο είναι συνηθισμένο κατά τη διάρκεια σχεδιασμού και ανάπτυξης συστημάτων λογισμικού, αφού είναι εξαιρετικά δύσκολο, έως αδύνατο, να γίνει από την πρώτη επανάληψη ένας «τέλειος» σχεδιασμός. Έτσι, παραδείγματος χάριν έγιναν διάφορες αλλαγές στα διαγράμματα ER και τροποποιήθηκαν πεδία ή δημιουργήθηκαν νέα tables. Επίσης, προστέθηκαν επιπλέον endpoints, όπου αυτό κρινόταν απαραίτητο. Επομένως, ο τελικός σχεδιασμός του συστήματος που αναλύθηκε στο προηγούμενο κεφάλαιο ήταν αποτέλεσμα πολλών επαναλήψεων μεταξύ σχεδιασμού και ανάπτυξης λογισμικού, ώστε τελικά να προκύψει ο «τέλειος» σχεδιασμός για το σύστημα, που ικανοποιεί όλες τις απαιτήσεις του συστήματος.

Υλοποίηση του Back End

Για την υλοποίηση του Back End component χρησιμοποιήθηκαν τα Python Django και Django REST Framework. Για κάθε ένα από τα 6 apps που αναλύθηκαν στο υποκεφάλαιο με το σχεδιασμό των διαγραμμάτων ER για το Back end, δημιουργήθηκε ένα διαφορετικό Django application.

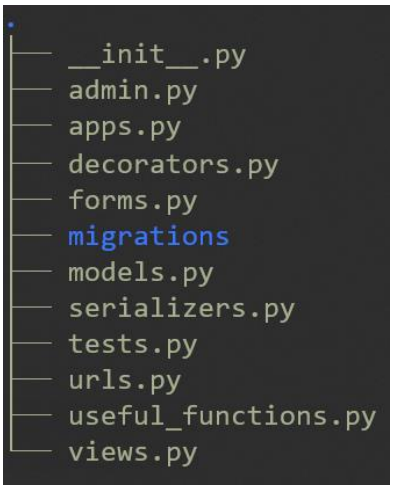
Έτσι, το Back End έχει τη δομή που φαίνεται στην εικόνα 28.



Εικόνα 28: Δομή του django project του backend

Επομένως, δημιουργήθηκε ένα Django project που ονομάζεται charging_operator, και περιέχει τα applications: chargers, gridprice, reservations, stations, stats (statistics) και users.

Κάθε ένα από τα django applications έχουν την δομή που φαίνεται στην εικόνα 29.



Εικόνα 29: Δομή ενός django application του backend

Τα κύρια αρχεία είναι τα εξής:

- models.py: Περιέχει τα tables του κάθε app. Παραδείγματος χάριν το gridprice app περιέχει τα Location και GridPrice tables, χρησιμοποιώντας το Django ORM για να περιγράφουν.
- urls.py: Περιέχει όλα τα endpoints του εκάστοτε app, όπως αναλύθηκαν στο υποκεφάλαιο με τη σχεδίαση των API endpoints του Back End.
- views.py: Περιέχει συναρτήσεις σχετικά με το πως να χειριστεί το Back End τα requests που γίνονται προς τα API endpoints του.
- serializers.py: Περιέχει κλάσεις για τη σειριοποίηση δεδομένων, χρησιμοποιώντας μεθόδους που δίνει το Django REST Framework.
- useful_functions.py: Περιέχει χρήσιμες συναρτήσεις που επαναχρησιμοποιούνται από πολλές μεθόδους του views.py. Παραδείγματος χάριν μία τέτοια συνάρτηση είναι η *get_user_station* που χρησιμοποιείται για να επαληθεύσει ότι ένας χρήστης (operator) έχει πρόσβαση στα δεδομένα ενός σταθμού. Άλλο ένα παράδειγμα είναι η *calculate_parking_cost*, που υπολογίζει το κόστος parking για μία κράτηση φόρτισης που διαρκεί μία συγκεκριμένη περίοδο.
- tests.py: Περιέχει functional tests για τις διάφορες συναρτήσεις που χρησιμοποιούνται σε κάθε app. Παραδείγματος χάριν για τη συνάρτηση *calculate_parking_cost* επαληθεύει ότι το κόστος που επιστρέφει είναι σωστό, δεδομένου της περιόδου, του σταθμού και του κόστους parking. Περισσότερα σχετικά με το testing στο back end θα αναλυθούν σε επόμενο υποκεφάλαιο.
- decorators.py: Περιέχει decorator συναρτήσεις Python, που χρησιμοποιούνται σε πολλά functions στο views.py. Παραδείγματος χάριν μία τέτοια decorator συνάρτηση είναι η *operator_required* που επαληθεύει ότι ο χρήστης που κάνει ένα request είναι operator (και όχι EV Owner).

Γενικότερα, δεν θα αναλυθεί περαιτέρω ο κώδικας που γράφτηκε για την κατασκευή όλων των endpoints, των tests, των models και των serializers, καθώς αυτός είναι μεγάλος σε όγκο (περισσότερες από 7000 σειρές κώδικα Python) και δεν θα προσφέρει επιπλέον αξία στο πλαίσιο της περιγραφής του συστήματος σε αυτήν την διπλωματική. Ωστόσο, ο πηγαίος κώδικας της εφαρμογής δίνεται στο τελευταίο κεφάλαιο της διπλωματικής.

Υλοποίηση του Grid Price Collector

Όπως αναφέρθηκε και στο σχεδιασμό της αρχιτεκτονικής του συστήματος, το component Grid Price Collector, χρησιμοποιείται για τη συλλογή δεδομένων σχετικά με τις τιμές της ηλεκτρικής ενέργειας σε διαφορετικές περιοχές. Ο Grid Price Collector ουσιαστικά αποτελείται από κάποιες συναρτήσεις που τρέχουν περιοδικά (πχ κάθε 5 λεπτά), συγκεντρώνοντας δεδομένα σχετικά με τις τιμές της ηλεκτρικής ενέργειας. Στο πλαίσιο αυτής της διπλωματικής περιέχει μόνο μία περιοδική συνάρτηση, που επικοινωνεί με το εξωτερικό σύστημα Fingrid API.

Όπως προαναφέρθηκε, Fingrid είναι ο εθνικός φορέας δικτύου μεταφοράς ηλεκτρικής ενέργειας της Φιλανδίας, και στο πλαίσιο της διπλωματικής χρησιμοποιήθηκε το ανοιχτό API που παρέχει, επειδή ήταν σχετικά εύκολο να αποκτηθεί πρόσβαση στο API. Συγκεκριμένα, για την απόκτηση πρόσβασης στο Fingrid API απαιτούνταν ένα API token, το οποίο δημιουργούνταν αυτόματα κατά την δημιουργία ενός λογαριασμού στην πλατφόρμα που παρέχει ο φορέας Fingrid. Έχοντας πρόσβαση στο API, χρησιμοποιήθηκε ένα συγκεκριμένο endpoint που δίνει την τιμή της ηλεκτρικής ενέργειας (€/MWh), το οποίο ανανεώνει τις τιμές κάθε ώρα.

Για να τρέχει με αυτόματο τρόπο η περιοδική συνάρτηση που συγκεντρώνει τα δεδομένα, χρησιμοποιήθηκαν τα Python Celery και Django Celery Beat. Συγκεκριμένα, μέσα στο κύριο directory του django project, δημιουργήθηκε ένα αρχείο που ονομάζεται celery.py. Εκεί μπορούν να οριστούν τα django celery beat schedules, δηλαδή διάφορα εργασίες (tasks) που θα τρέχουν περιοδικά. Ο κώδικας που υποδεικνύει το περιοδικό task φαίνεται παρακάτω:

```
app.conf.beat_schedule = {
    'get-fingrid-prices-5-minutes': {
        'task': 'get_fingrid_prices',
        'schedule': 300.0,
        'args': ()
    }
}
```

Συγκεκριμένα, φαίνεται ότι η συνάρτηση *get_fingrid_prices* πρέπει να τρέχει κάθε 300 δευτερόλεπτα (5 λεπτά). Έτσι, έχοντας ορίσει τις περιοδικές συναρτήσεις, ο celery beat scheduler ειδοποιεί αυτόματα τους celery workers (κάθε 5 λεπτά) να τρέξουν τη συνάρτηση *get_fingrid_prices*.

Να σημειωθεί ότι αντί για τα Python Celery και Django Celery Beat θα μπορούσαν να είχαν χρησιμοποιηθεί απλά cronjobs και Python scripts. Ωστόσο, αυτά δεν θα παρείχαν τη δυνατότητα ελέγχου των tasks και schedules μέσω του django admin, καθώς επίσης, δεν θα ήταν δυνατό να εκτελεστούν συγκεκριμένα functions, αλλά μόνο ολόκληρα scripts.

Back End Testing

Σημαντικό κομμάτι στον κύκλο ζωής της ανάπτυξης λογισμικού (software development life cycle) είναι το testing. Το software testing είναι η διαδικασία ελέγχου και επαλήθευσης ότι ένα κομμάτι λογισμικού λειτουργεί όπως πρέπει. Οι developers γράφοντας σωστά tests, μπορούν να εμποδίσουν την εμφάνιση bugs, να μειώσουν το κόστος ανάπτυξης και να βελτιώσουν τις επιδόσεις ενός συστήματος λογισμικού.

Στο πλαίσιο αυτής της διπλωματικής γράφτηκαν unit tests για κάποιες βασικές συναρτήσεις του backend. Unit tests είναι μία συγκεκριμένη κατηγορία από tests, που επικεντρώνονται στον έλεγχο και επαλήθευση της σωστής λειτουργίας μία μικρής μονάδας κώδικα, όπως είναι οι συναρτήσεις. Τα tests γράφτηκαν μέσα στο Django test suite, που παρέχει το django. Συγκεκριμένα, γράφτηκαν συνολικά 15 tests για τις πιο σημαντικές συναρτήσεις των 6 django applications του backend, ώστε να επαληθευτεί ότι το κύριο functionality του κάθε app είναι το επιθυμητό.

Αυτό ήταν ιδιαίτερα χρήσιμο, αφού οποιαδήποτε στιγμή απαιτούνταν να γίνει κάποια αλλαγή στο backend, έτρεχαν αυτοματοποιημένα τα διάφορα tests εκτελώντας την εντολή που φαίνεται παρακάτω στο κύριο directory του backend:

```
$ python3 manage.py test
```

Έτσι, όταν γινόταν κάποια αλλαγή στο backend ήταν ιδιαίτερα εύκολο να τρέξουν τα tests και να γίνει έλεγχος ότι οι αλλαγές που έγιναν δεν επηρεάζουν το ορθό functionality του backend Component.

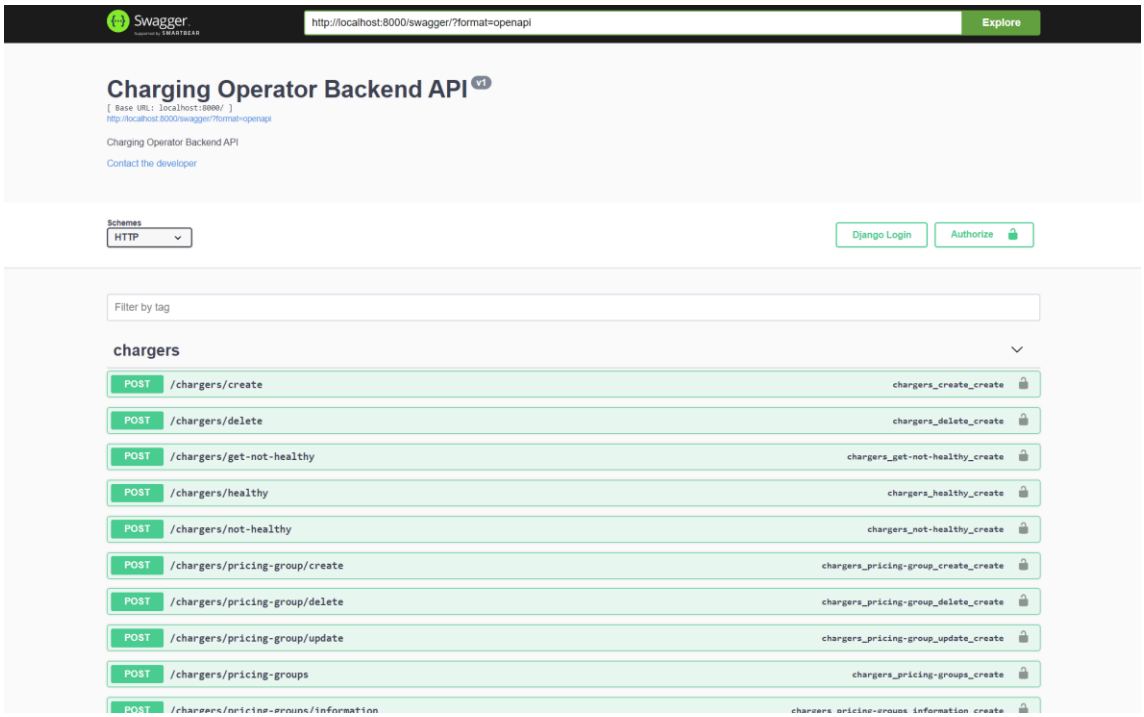
Documentation *για τα* Backend Endpoints

Documentation στην ανάπτυξη λογισμικού είναι κείμενο ή εικόνες που συνοδεύουν τον κώδικα ή είναι ενσωματωμένο στον κώδικα. Το documentation είναι ιδιαίτερα χρήσιμο, καθώς εξηγεί τον τρόπο με τον οποίο λειτουργεί το υπό εξέταση λογισμικό ή πως μπορεί να το χρησιμοποιήσει κάποιος τρίτος. Παράδειγμα documentation θα μπορούσε να είναι το μεγαλύτερο μέρος αυτής της διπλωματικής, αφού εξηγεί τον τρόπο που λειτουργεί και έχει σχεδιαστεί το σύστημα.

Στο συγκεκριμένο υποκεφάλαιο θα γίνει μία περιγραφή του τρόπου με τον οποίο έγινε το documentation για τα API endpoints του backend. Το documentation σε ένα σύστημα που εκθέτει (exposes) κάποια endpoints είναι σε πολλές περιπτώσεις απαραίτητο, καθώς διαφορετικοί ενδεχομένως χρήστες θα κληθούν να χρησιμοποιήσουν το API. Στην περίπτωση αυτής της διπλωματικής, ο λόγος που αναπτύχθηκε documentation για τα API endpoints του backend είναι ώστε το σύστημα που αναπτύχθηκε να είναι πιο ολοκληρωμένο και να είναι ξεκάθαρο το τι υλοποιεί το κάθε endpoint του backend.

Για να επιτευχθεί αυτό, λοιπόν, χρησιμοποιήθηκε μία συγκεκριμένη έκδοση ενός Swagger γεννήτορα, που παρείχε συμβατότητα με το Django framework, ώστε να επιταχυνθεί η παραγωγή του API documentation και να παραχθούν αυτόματα οι πληροφορίες για κάθε

endpoint του backend. Στην εικόνα 30 φαίνεται η σελίδα με το Swagger documentation που δημιουργήθηκε για το backend component του συστήματος.



Εικόνα 30: Σελίδα Swagger Documentation για τα API endpoints του backend

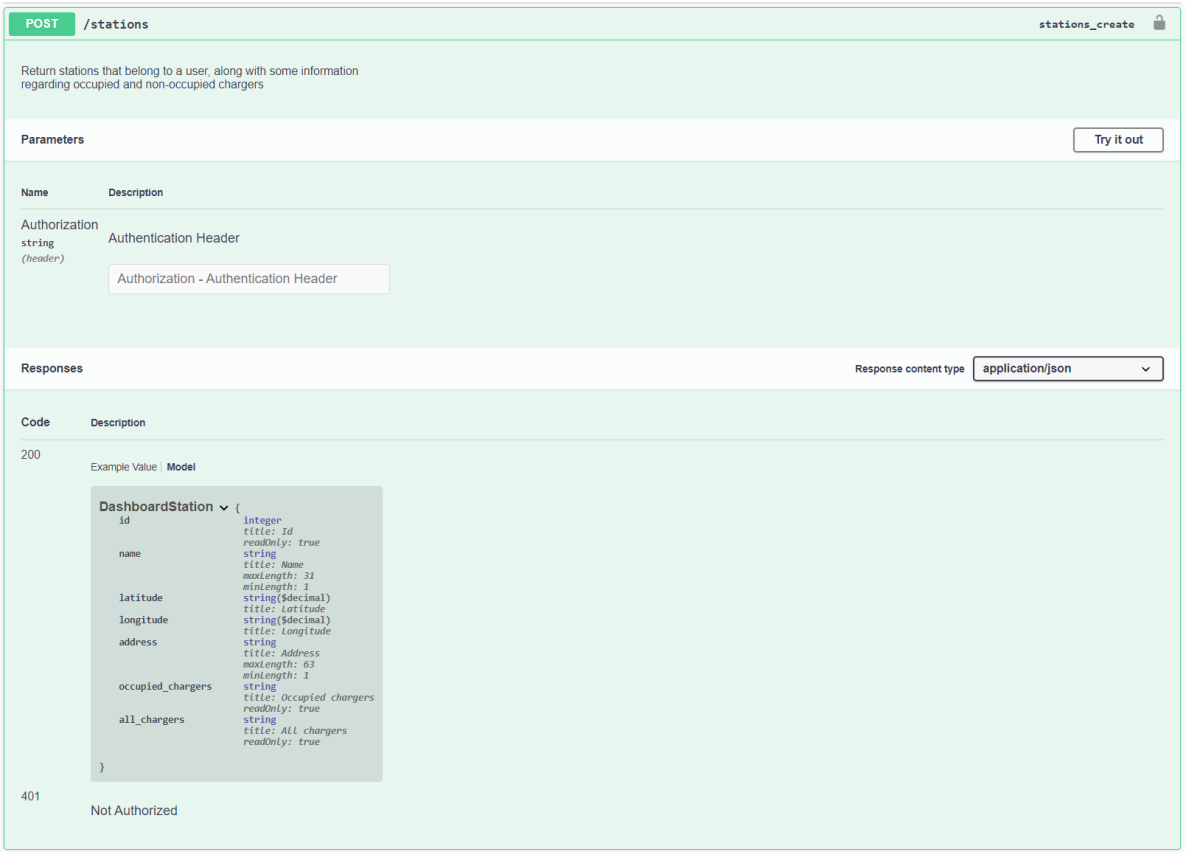
Με άλλα λόγια, το εργαλείο Swagger αναλαμβάνει να παράγει αυτόματα το documentation για όλα τα endpoints της εφαρμογής. Ωστόσο, για να συμβεί αυτό πρέπει κάθε συνάρτηση των αρχείων views.py να έχει τη μορφή που φαίνεται στην εικόνα 31.

```
22  @swagger_auto_schema(
23      methods=['POST'],
24      manual_parameters=[AUTHENTICATION_HEADER],
25      responses={
26          200: DashboardStationSerializer,
27          401: 'Not Authorized'
28      }
29  )
30  @api_view(['POST', ])
31  @permission_classes((IsAuthenticated,))
32  @operator_required
33  def get_stations(request):
34      """Return stations that belong to a user, along with some information
35      regarding occupied and non-occupied chargers
36
37      Returns:
38          data, status: if successful returns a list of stations, with an
39              HTTP_200_OK status
40      """
41      stations = Station.objects.filter(operators__id=request.user.id)
42      serializer = DashboardStationSerializer(stations, many=True)
43
44      return Response(serializer.data, status=status.HTTP_200_OK)
```

Εικόνα 31: Συνάρτηση get_stations από το views.py του stations app

Συγκεκριμένα, πρέπει να χρησιμοποιηθεί ο decorator *swagger_auto_schema* και να προσδιοριστούν μέσω αυτού οι μέθοδοι, τα headers, και το τι επιστρέφει η κάθε συνάρτηση στο views.py. Επίσης, πρέπει να γραφεί documentation υπό τη μορφή που φαίνεται στις σειρές 34 έως 40, που περιγράφει το τι κάνει αυτή η συνάρτηση-endpoint και τι επιστρέφει.

Στην εικόνα 32 φαίνεται το αποτέλεσμα του Swagger UI για την συνάρτηση της εικόνας 31.



Εικόνα 32: Swagger Documentation για το endpoint της συνάρτησης get_stations

Να σημειωθεί ότι δημιουργήθηκε τέτοιο documentation για όλα τα 51 endpoints του backend.

Υλοποίηση του Front End

Για την υλοποίηση των Charging Operator Frontend και EV Owner Frontend, χρησιμοποιήθηκε η βιβλιοθήκη React JS. Συγκεκριμένα δημιουργήθηκε ένα κοινό App και για τα δύο components. Για την επιτάχυνση της ανάπτυξης λογισμικού για το Frontend, χρησιμοποιήθηκαν και εξωτερικά libraries, όπως recharts για γραφήματα, react-alerts για alerts, bootstrap για modals, react-loading για εμφάνιση εικονιδίων loading και άλλα. Επίσης, είναι σημαντικό να αναφερθεί ότι έγινε integration με το Google Maps API, για την εμφάνιση χαρτών που απεικονίζονται οι σταθμοί φόρτισης. Αυτό είναι ιδιαίτερα χρήσιμο, καθώς μπορεί εύκολα να γίνει οπτικοποίηση των σταθμών, ώστε οι operators να βλέπουν άλλους σταθμούς κοντά στους σταθμούς τους, και οι EV Owners να μπορούν να δουν ποιοι σταθμοί τους εξυπηρετούν στο σημείο που βρίσκονται.

Όσον αφορά τη δομή του React App είναι αυτή που φαίνεται στην εικόνα 33.

Εικόνα 33: Δομή του Frontend React Application

Μέσα στο src/ directory του React App, τα αρχεία και οι φάκελοι που φαίνονται είναι τα εξής:

- App.js: Αρχείο που περιέχει τα routes της web εφαρμογής, δηλαδή για ποια urls θα εμφανίζονται οι διάφορες σελίδες που δημιουργήθηκαν.
- api: Φάκελος που βρίσκονται 2 αρχεία που περιέχουν συναρτήσεις για την επικοινωνία με το backend API. Οι συναρτήσεις που επικοινωνούν με το backend API γράφτηκαν σκόπιμα σε ένα directory, και όχι διάσπαρτα σε πολλά σημεία της εφαρμογής, ώστε να μπορούν εύκολα να επαναχρησιμοποιηθούν συναρτήσεις, και να είναι εύκολη η τροποποίηση αυτών σε περίπτωση που απαιτείται.
- bootstrap.css: Αρχείο που περιέχει CSS styles σχετικά με τα components της βιβλιοθήκης bootstrap.
- components: Directory που περιέχει components της εφαρμογής που χρησιμοποιούνται στις διάφορες σελίδες. Η React προωθεί την επαναχρησιμοποίηση κώδικα, και γι' αυτό έγινε μία προσπάθεια η πλειοψηφία των React Components που αναπτύχθηκαν να γραφούν με τέτοιο τρόπο ώστε να μπορούν να επαναχρησιμοποιηθούν όπου απαιτείται. Οπότε για παράδειγμα ένα στοιχείο Select, για επιλογή από το χρήστη, δημιουργήθηκε ως ξεχωριστό component και επαναχρησιμοποιήθηκε σε πολλές σελίδες.
- context: Φάκελος που περιέχει ένα αρχείο που διαχειρίζεται την αυθεντικοποίηση των χρηστών. Συγκεκριμένα, περιέχει τις συναρτήσεις *getAuth, setAuth, isAuthenticatedOperator, isAuthenticatedOwner* και *logout*.
- hooks: Περιέχει τα custom React hooks της εφαρμογής.
- pages: Περιέχει τις σελίδες της εφαρμογής. Πρακτικά, οι σελίδες είναι αυτές που αναλύθηκαν στο υποκεφάλαιο με τα mockups για το Frontend του προηγούμενου κεφαλαίου.
- standard.css: Αρχείο που περιέχει CSS styles για την εφαρμογή.
- utils: Φάκελος που περιέχει ένα αρχείο με διάφορες χρήσιμες συναρτήσεις που επαναχρησιμοποιούνται σε πολλά σημεία στην εφαρμογή. Παραδείγματος χάριν, μία τέτοια συνάρτηση είναι η *stringToDatetime*, που μετατρέπει μία ημερομηνία σε μορφή string όπως δίνεται από το backend, σε μορφή Javascript Date Object.

Ομοίως με το υποκεφάλαιο της υλοποίησης του backend, δεν θα αναλυθεί περαιτέρω ο κώδικας που γράφτηκε για την κατασκευή όλων των σελίδων, καθώς αυτός είναι μεγάλος σε όγκο (περισσότερες από 15000 σειρές κώδικα JavaScript και CSS) και δεν θα προσφέρει επιπλέον αξία στο πλαίσιο της περιγραφής του συστήματος σε αυτήν την διπλωματική. Ωστόσο, ο πηγαίος κώδικας της εφαρμογής δίνεται στο τελευταίο κεφάλαιο της διπλωματικής.

Κεφάλαιο 5: Παραδείγματα χρήσης του Front End

Εισαγωγή

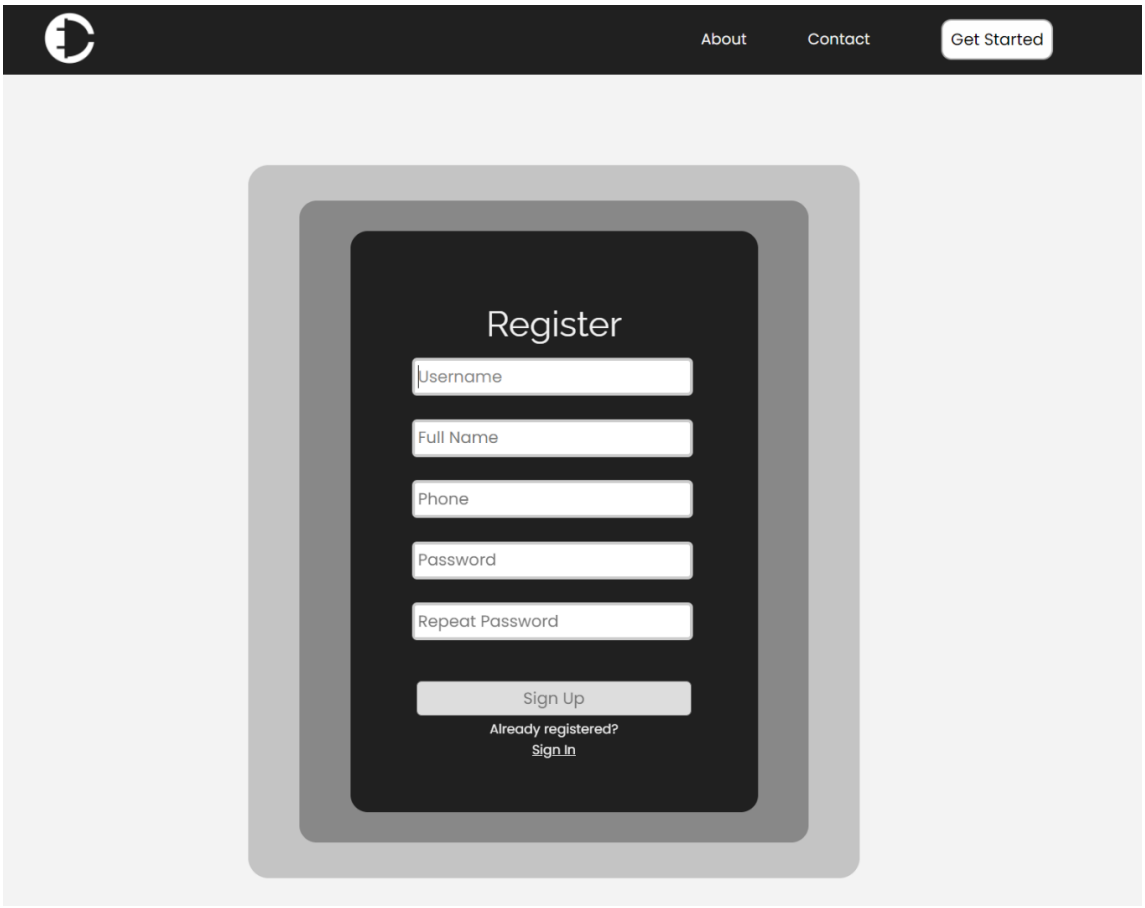
Στο κεφάλαιο αυτό θα παρουσιαστούν κάποια παραδείγματα χρήσης της εφαρμογής, δηλαδή θα περιγραφεί ένα μέρος της λειτουργικότητας του συστήματος που αναδεικνύεται ο τρόπος που οι χρήστες της εφαρμογής (operators και EV owners) μπορούν να την χρησιμοποιήσουν.

Σε αντίθεση με τα mockups του Frontend που παρουσιάστηκαν στο κεφάλαιο 3, σε αυτό το κεφάλαιο θα παρουσιαστούν στιγμιότυπα από την πραγματική εφαρμογή, που υλοποιήθηκε όπως περιεγράφηκε στο κεφάλαιο 4. Με άλλα λόγια, οι εικόνες που θα παρουσιαστούν σε αυτό το κεφάλαιο είναι στιγμιότυπα από τα πλήρως λειτουργικά Charging Operator Frontend και EV Owner Frontend, τα οποία ανταλλάσσουν πραγματικά δεδομένα με το backend, ακολουθώντας τις προδιαγραφές που αναλύθηκαν στα Κεφάλαια 3 και 4.

Ο στόχος αυτού του κεφαλαίου είναι να αναδειχθούν μέσα από παραδείγματα οι περιπτώσεις χρήσης του συστήματος που αναπτύχθηκε στο πλαίσιο αυτής της διπλωματικής.

User Registration and Login

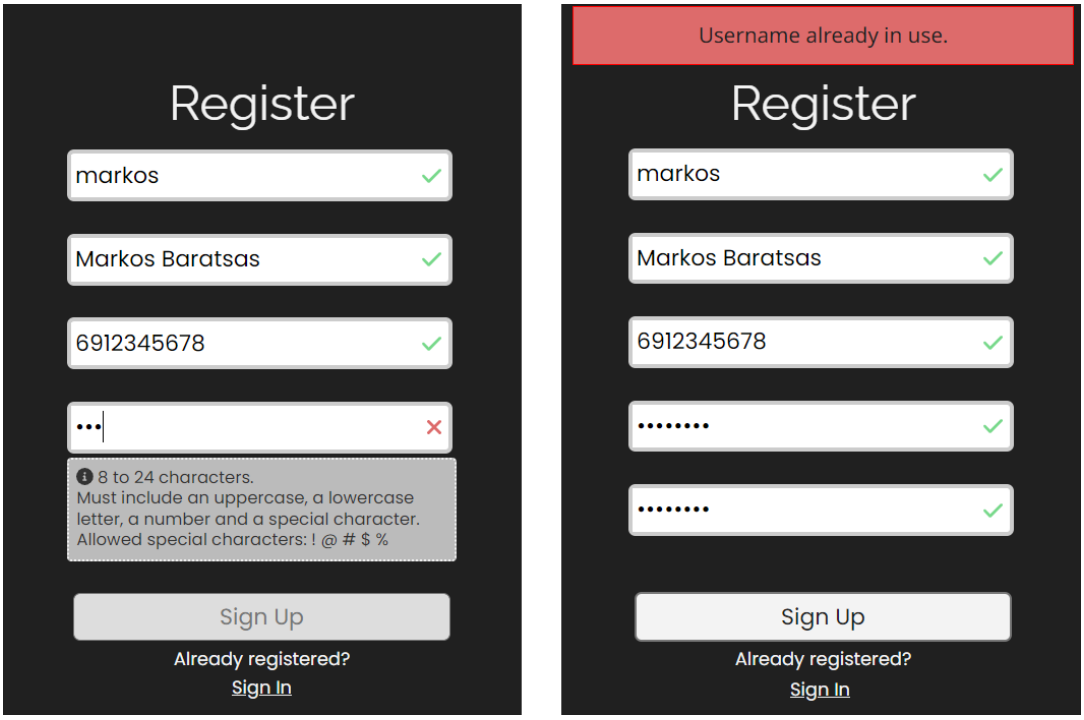
Στο κεφάλαιο αυτό θα παρουσιαστεί ο τρόπος που μπορεί ένας χρήστης (operator ή EV owner) να κάνει εγγραφή στην εφαρμογή (δημιουργία λογαριασμού) καθώς και ο τρόπος που μπορεί να συνδεθεί.



Εικόνα 34: Σελίδα εγγραφής χρήστη

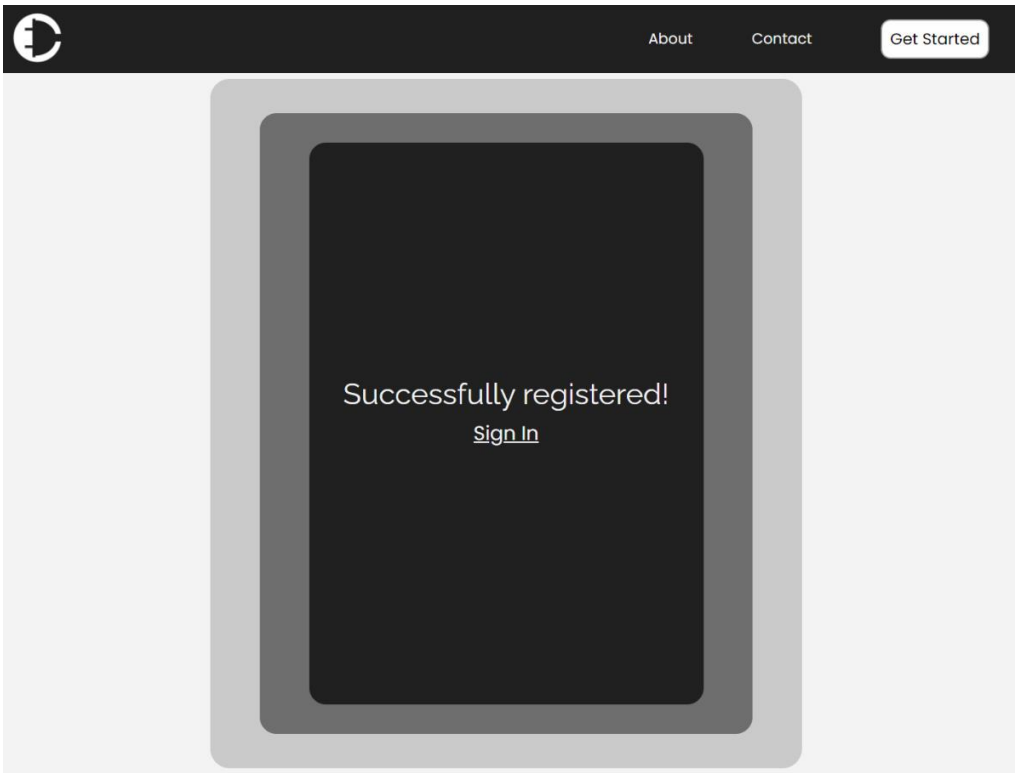
Στην εικόνα 34, φαίνεται ένα στιγμιότυπο από τη σελίδα εγγραφής (Register). Να σημειωθεί ότι γίνεται έλεγχος των πεδίων που συμπληρώνονται από το χρήστη, ώστε τα δεδομένα που θα σταλούν στο backend να έχουν φιλτραριστεί. Προφανώς, επιπλέον έλεγχοι γίνονται και στο backend.

Εκτός των ελέγχων που γίνονται όταν ο χρήστης πατήσει το κουμπί «Sign Up», γίνονται και άμεσοι έλεγχοι των χαρακτήρων που συμπληρώνει ο χρήστης ώστε να μπορεί εύκολα να κατανοήσει ποιοι είναι οι αποδεκτοί χαρακτήρες για κάθε πεδίο. Στην εικόνα 35 φαίνονται κάποια βοηθητικές ενδείξεις που εμφανίζονται αυτόματα, καθώς ο χρήστης πληκτρολογεί. Μετά την επιτυχής συμπλήρωση των πεδίων κατά την πίεση του κουμπιού «Sign up», γίνονται έλεγχοι από το backend σχετικά με το αν υπάρχει χρήστης που ήδη έχει αυτό το username, αν το username περιλαμβάνει μη επιτρεπτούς χαρακτήρες κτλ. Στην εικόνα 36 φαίνεται το μήνυμα που εμφανίζεται όταν ένας χρήστης είναι ήδη εγγεγραμμένος με αυτό το username.



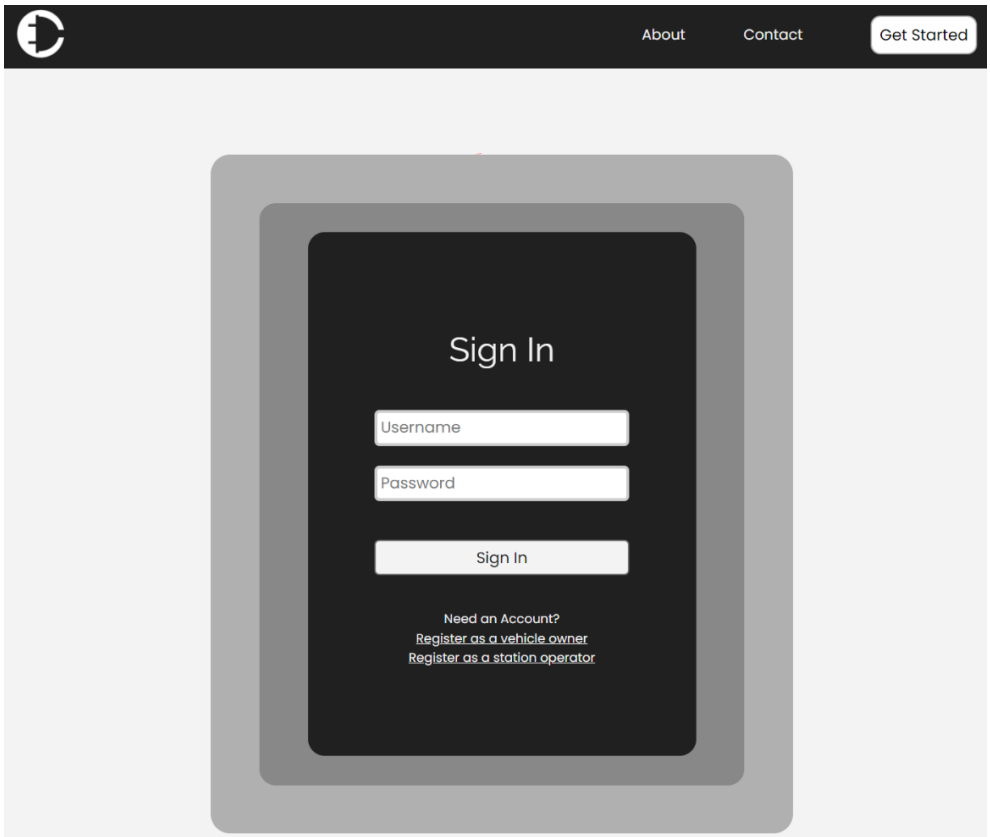
Εικόνα 35: Βοηθητικές ενδείξεις κατά την πληκτρολόγηση (αριστερά), εμφάνιση σφάλματος (δεξιά)

Σε περίπτωση που όλα τα στοιχεία συμπληρώθηκαν σωστά και δεν υπάρχει κάποιο μήνυμα σφάλματος από το backend (όπως πχ ότι το όνομα χρήστη είναι ήδη κατειλημμένο), τότε η εγγραφή χρήστη είναι επιτυχής και εμφανίζεται το μήνυμα της εικόνας 36.



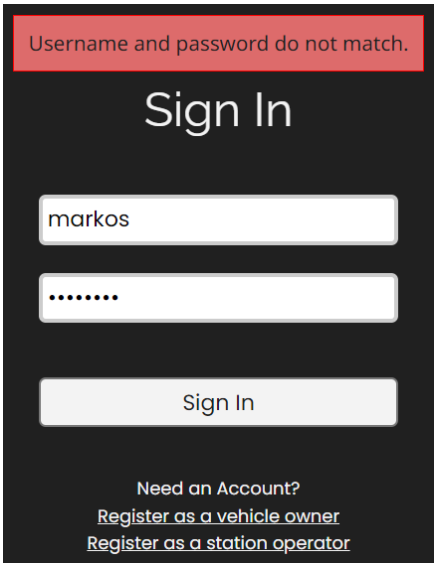
Εικόνα 36: Μήνυμα επιτυχούς εγγραφής χρήστη

Στη συνέχεια, εφόσον ο χρήστης (operator ή EV Owner) έχει δημιουργήσει λογαριασμό μπορεί να συνδεθεί σε αυτόν χρησιμοποιώντας τη σελίδα σύνδεσης (εικόνα 37).



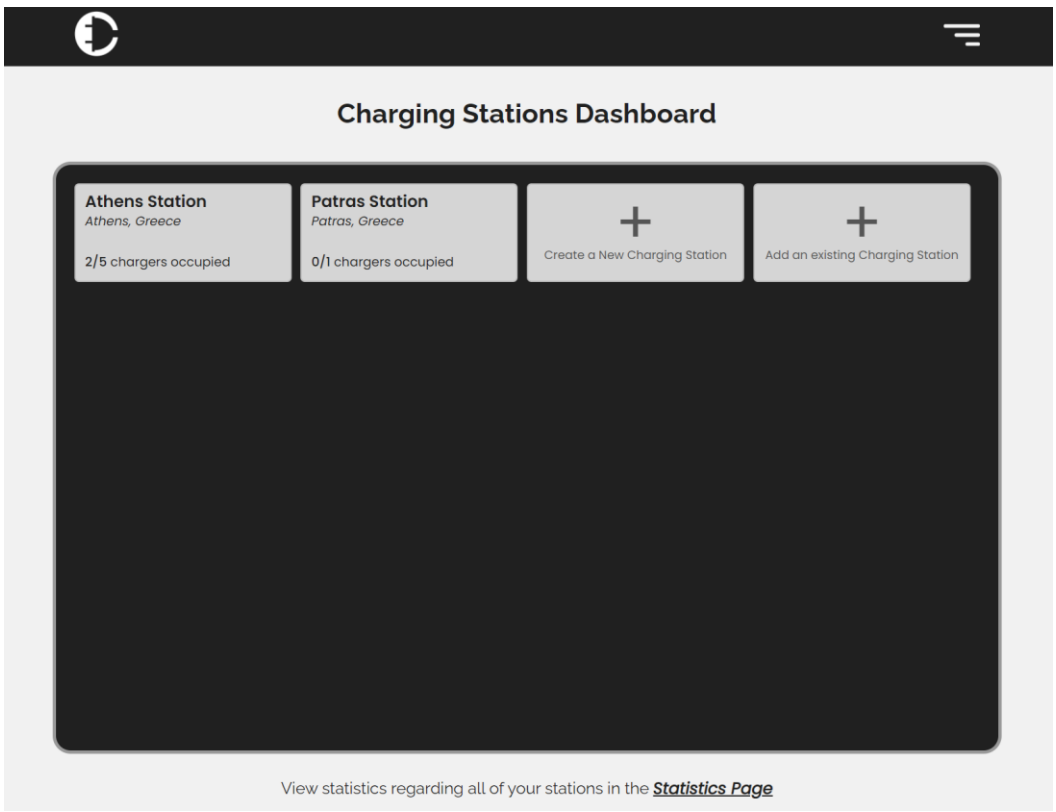
Εικόνα 37: Σελίδα σύνδεσης χρήστη.

Ομοίως με τη σελίδα εγγραφής χρήστη και στη σελίδα σύνδεσης εμφανίζονται κατάλληλα μηνύματα όταν ο χρήστης δίνει στοιχεία που δεν είναι σωστά.

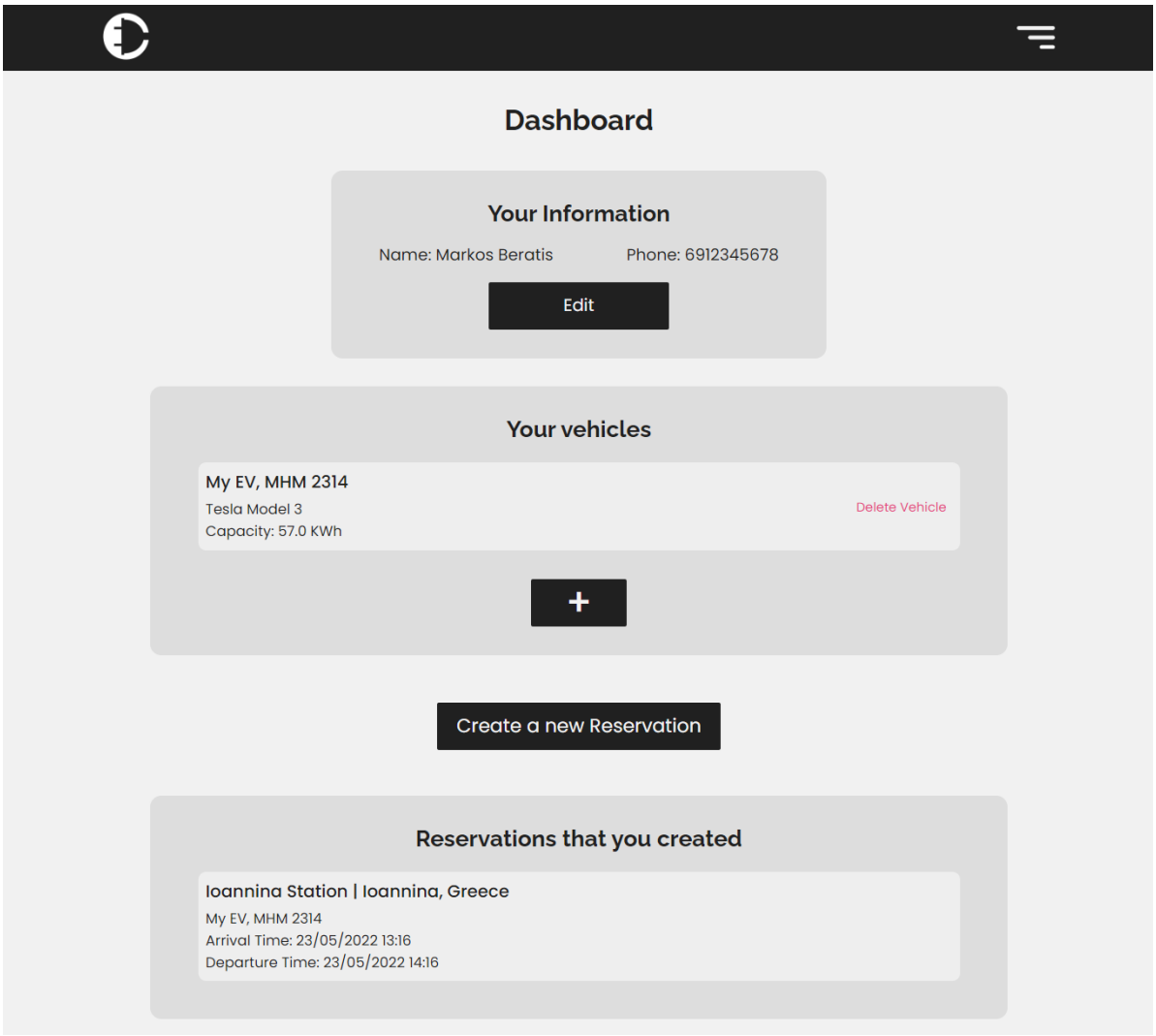


Εικόνα 38: Μήνυμα σφάλματος στη σελίδα σύνδεσης χρήστη

Σε περίπτωση που ο χρήστης δώσει σωστό συνδυασμό username και password, τότε συνδέεται και ανακατευθύνεται αυτόματα στη σελίδα Dashboard, η οποία προφανώς είναι διαφορετική για τους operators και για τους EV owners. Στην εικόνα 39 φαίνεται το Dashboard που εμφανίζεται στους operators των σταθμών φόρτισης, ενώ στη σελίδα 40 φαίνεται το Dashboard που εμφανίζεται στους EV owners.



Εικόνα 39: Σελίδα Dashboard ενός operator



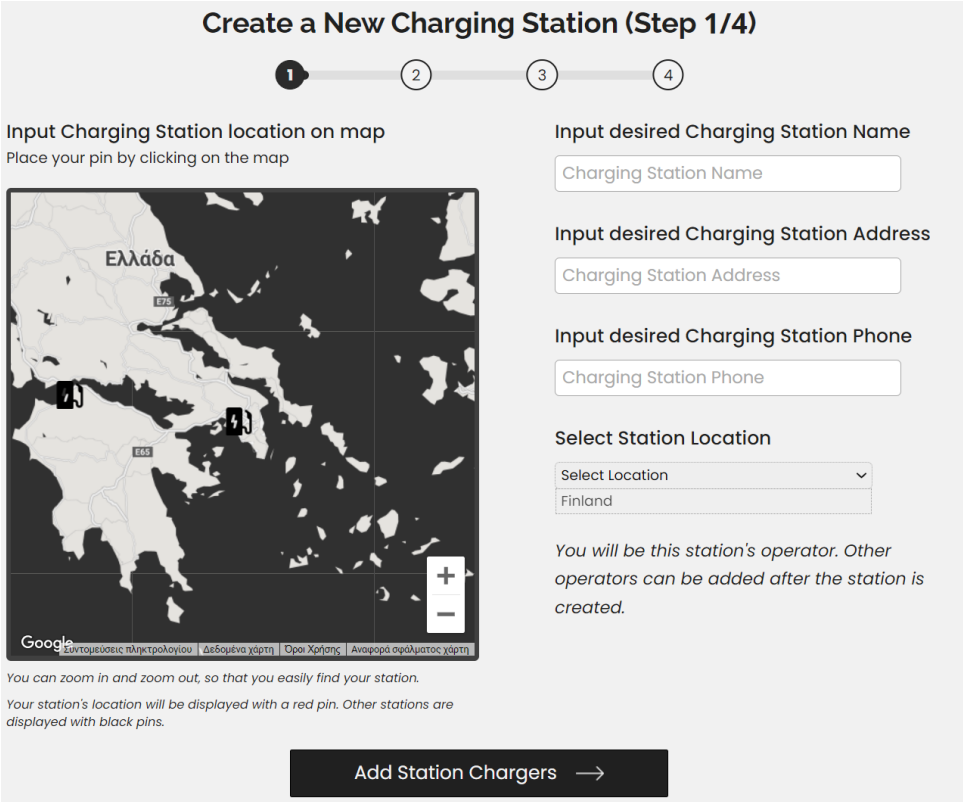
Εικόνα 40: Σελίδα Dashboard ενός ιδιοκτήτη ηλεκτρικού οχήματος

Περισσότερες πληροφορίες σχετικά με το Dashboard των EV Owners θα δοθούν σε επόμενο υποκεφάλαιο που επικεντρώνεται στην ανάλυση του EV Owner Dashboard.

Charging Operator Frontend: Station Creation

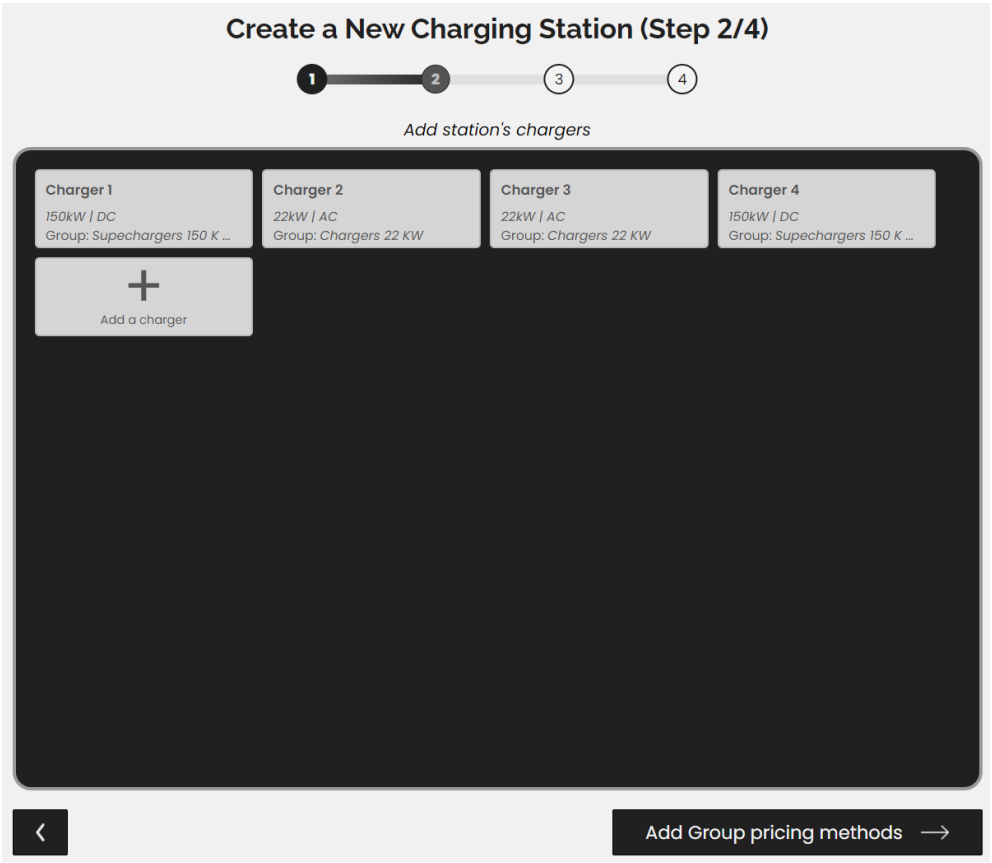
Κάνοντας click πάνω στο «Create a New Charging Station» της εικόνας 39, ένας operator μπορεί να δημιουργήσει ένα νέο σταθμό. Η δημιουργία ενός νέου σταθμού περιλαμβάνει 4 βήματα.

Στο πρώτο βήμα συμπληρώνονται πληροφορίες σχετικά με το σταθμό (εικόνα 41). Συγκεκριμένα, ο operator συμπληρώνει το όνομα, τη διεύθυνση, το τηλέφωνο και το Location του σταθμού (table Location από το gridprice app). Όπως φαίνεται το μόνο διαθέσιμο Location είναι η Φιλανδία, αφού λαμβάνονται δεδομένα σχετικά με την τιμή της ενέργειας μόνο από το Fingrid API.



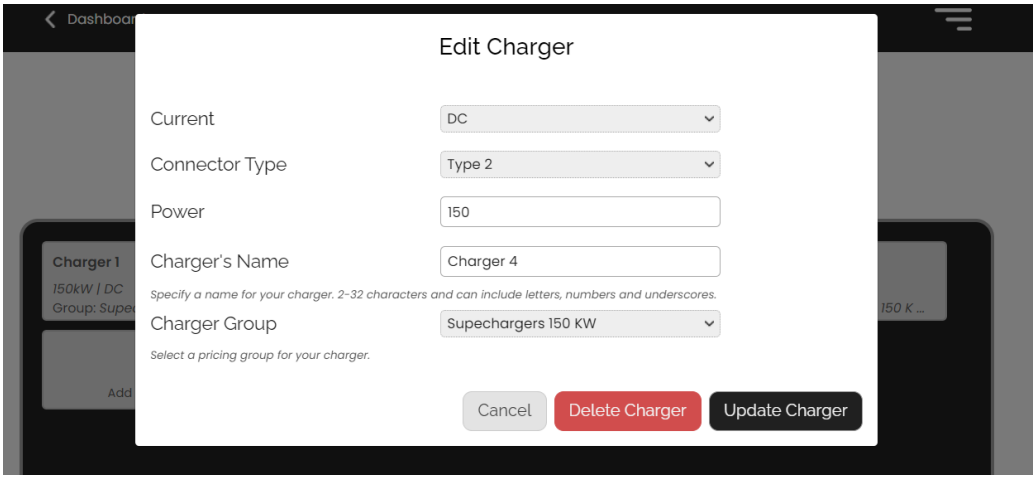
Εικόνα 41: Σελίδα δημιουργία σταθμού (βήμα 1/4)

Το δεύτερο βήμα περιλαμβάνει την προσθήκη των φορτιστών του σταθμού (εικόνα 42).



Εικόνα 42: Σελίδα δημιουργία σταθμού (βήμα 2/4)

Συγκεκριμένα, κάνοντας click στο «Add a Charger» ανοίγει ένα Modal που δίνει τη δυνατότητα προσθήκης πληροφοριών για έναν φορτιστή. Εκτός αυτού, παρέχεται η λειτουργικότητα αλλαγής των στοιχείων ή διαγραφής ενός φορτιστή, ώστε να μπορεί ο operator με εύκολο τρόπο να προσθέσει/τροποποιήσει/διαγράψει φορτιστές. Στην εικόνα 43 φαίνεται το Modal που ανοίγει για την τροποποίηση/διαγραφή ενός φορτιστή.

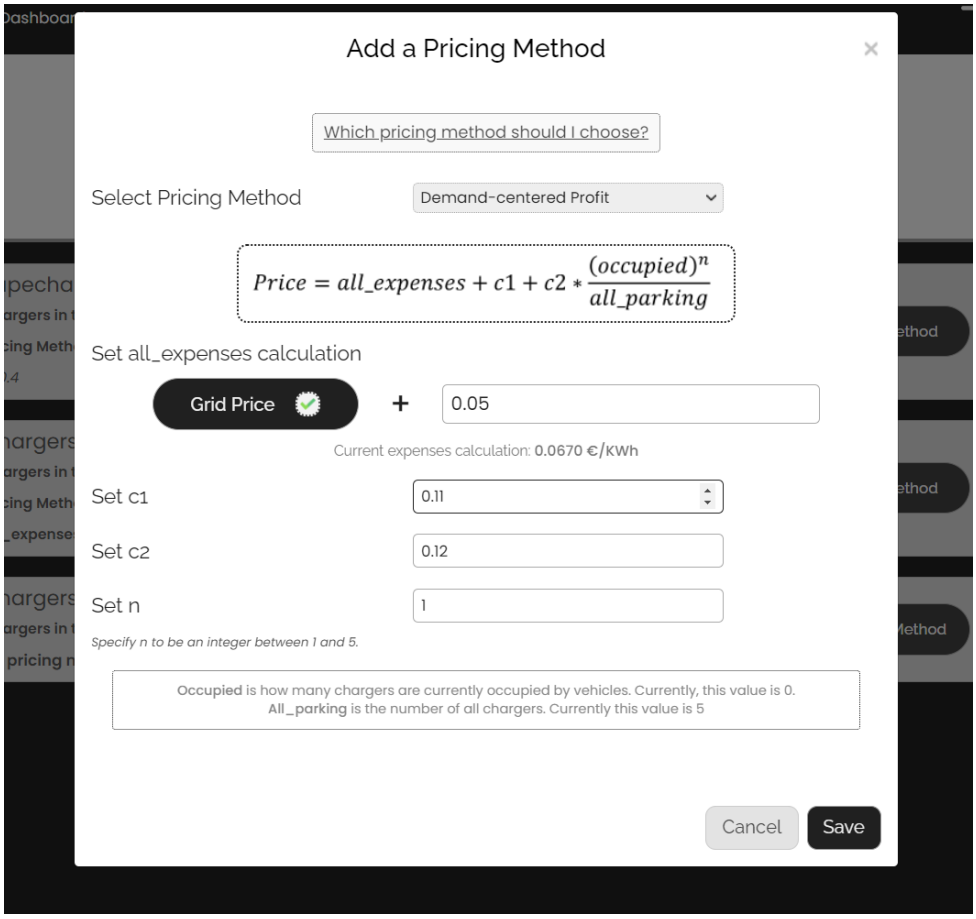


Εικόνα 43: Modal τροποποίησης/διαγραφής φορτιστή

Ακολούθως, στο βήμα 3 (εικόνα 44) παρουσιάζονται τα Charger Groups/Pricing Groups και δίνεται η δυνατότητα στον operator να επιλέξει μία μέθοδο τιμολόγησης από τις 4 διαθέσιμες.

Εικόνα 44: Σελίδα δημιουργία σταθμού (βήμα 3/4)

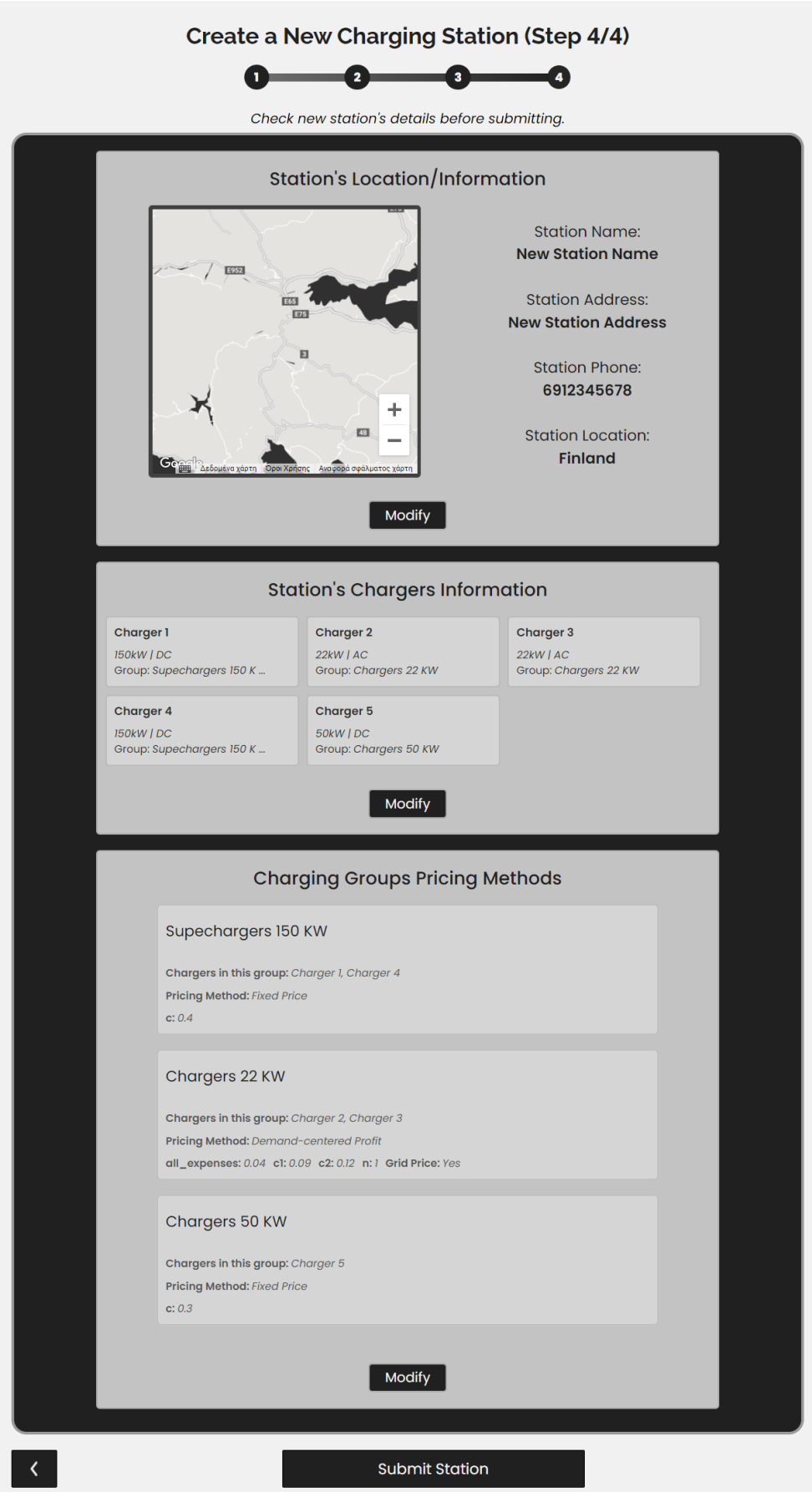
Στην εικόνα 45 φαίνεται το Modal που ανοίγει όταν γίνεται click στα «Select/Edit Pricing Method». Ο operator μπορεί να επιλέξει μία από τις 4 διαθέσιμες μεθόδους τιμολόγησης και να ορίσει τις παραμέτρους της. Εκτός αυτού, οι operators μπορούν να δουν αναλυτικά πληροφορίες για κάθε μία από τις διαθέσιμες μεθόδους, κάνοντας click στο «Which pricing method should I choose», το οποίο οδηγεί σε μία σελίδα που παρουσιάζονται αναλυτικά πληροφορίες σχετικά με τις μεθόδους τιμολόγησης.



Εικόνα 45: Modal για την επιλογή μεθόδου και παραμέτρων τιμολόγησης

Τέλος, στο τέταρτο βήμα γίνεται ένας τελικός έλεγχος των στοιχείων που προστέθηκαν (εικόνα 46) . Ο operator έχει τη δυνατότητα να δει αναλυτικά τις πληροφορίες που πρόσθεσε πριν πατήσει υποβολή, ώστε ο σταθμός να δημιουργηθεί. Να σημειωθεί ότι η εφαρμογή επιτρέπει την ενημέρωση των στοιχείων των φορτιστών και των Pricing Groups των σταθμών και σε μεταγενέστερη κατάσταση. Επομένως, ακόμα και αν έγινε κάποιο λάθος από την πλευρά του operator έχει τη δυνατότητα να το διορθώσει.

Μετά την υποβολή ο σταθμός και όλες οι πληροφορίες του καταχωρούνται στο database του backend και, στη συνέχεια, ο σταθμός εμφανίζεται στο Dashboard του operator (όπως είδαμε στην εικόνα 39).

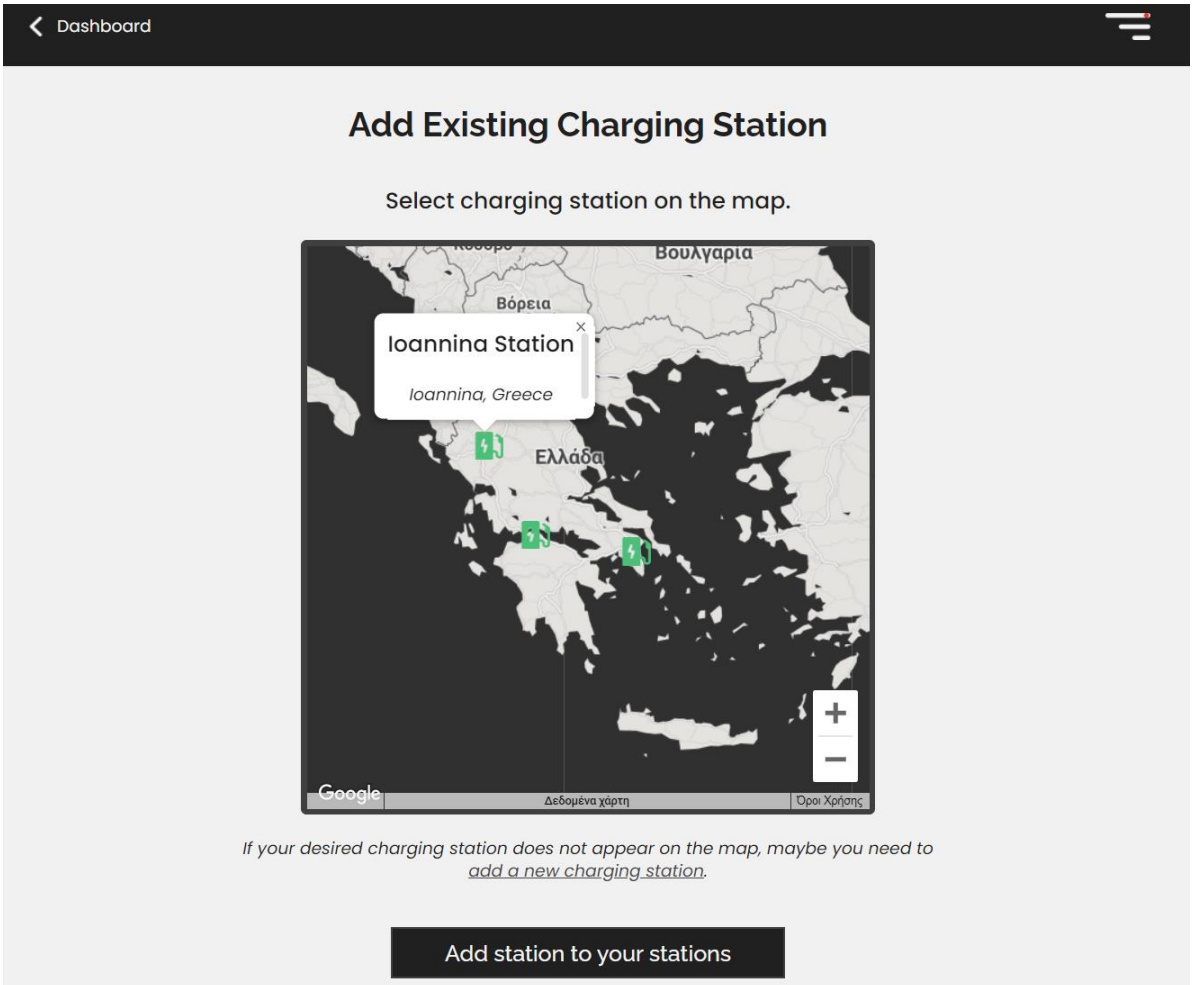


Εικόνα 46: Σελίδα δημιουργία σταθμού (βήμα 4/4)

Charging Operator Frontend: Station Add

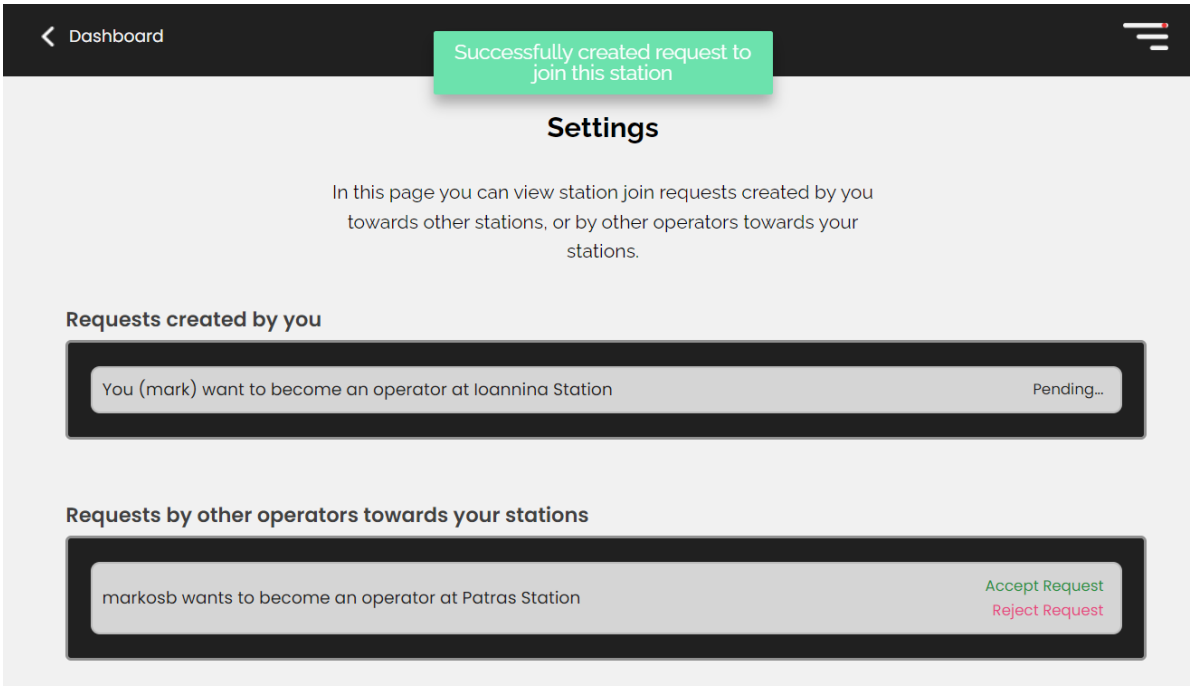
Ένα άλλο παράδειγμα χρήσεως είναι η προσθήκη ενός υπάρχοντος σταθμού στους διαθέσιμους σταθμούς ενός operators. Αυτό είναι απαραίτητο όταν σε ένα σταθμό φόρτισης υπάρχουν παραπάνω από ένας operators. Ο operator μπορεί να προσθέσει έναν υπάρχων σταθμό κάνοντας click στο «Add an existing Charging Station» του Dashboard (εικόνα 39).

Στην εικόνα 47 φαίνεται η σελίδα προσθήκης υπάρχοντος σταθμού. Συγκεκριμένα, ο operator μπορεί να επιλέξει το σταθμό που θέλει να προσθέσει, χρησιμοποιώντας τον ειδικά διαμορφωμένο χάρτη Google Maps που ενσωματώθηκε στην εφαρμογή και περιέχει όλους τους σταθμούς.



Εικόνα 47: Σελίδα προσθήκης υπάρχοντος σταθμού

Αφού επιλεγεί ο σταθμός και πατηθεί το κουμπί «Add stations to your stations», δημιουργείται ένα StationRequest object στο backend database και ο χρήστης ανακατευθύνεται στην σελίδα που φαίνεται στην εικόνα 48. Αυτό το StationRequest μπορούν να το δουν οι αρμόδιοι operators και να το αποδεχτούν/απορρίψουν. Έτσι, διασφαλίζεται ότι σε κάθε σταθμό έχουν πρόσβαση οι operators που πρέπει.



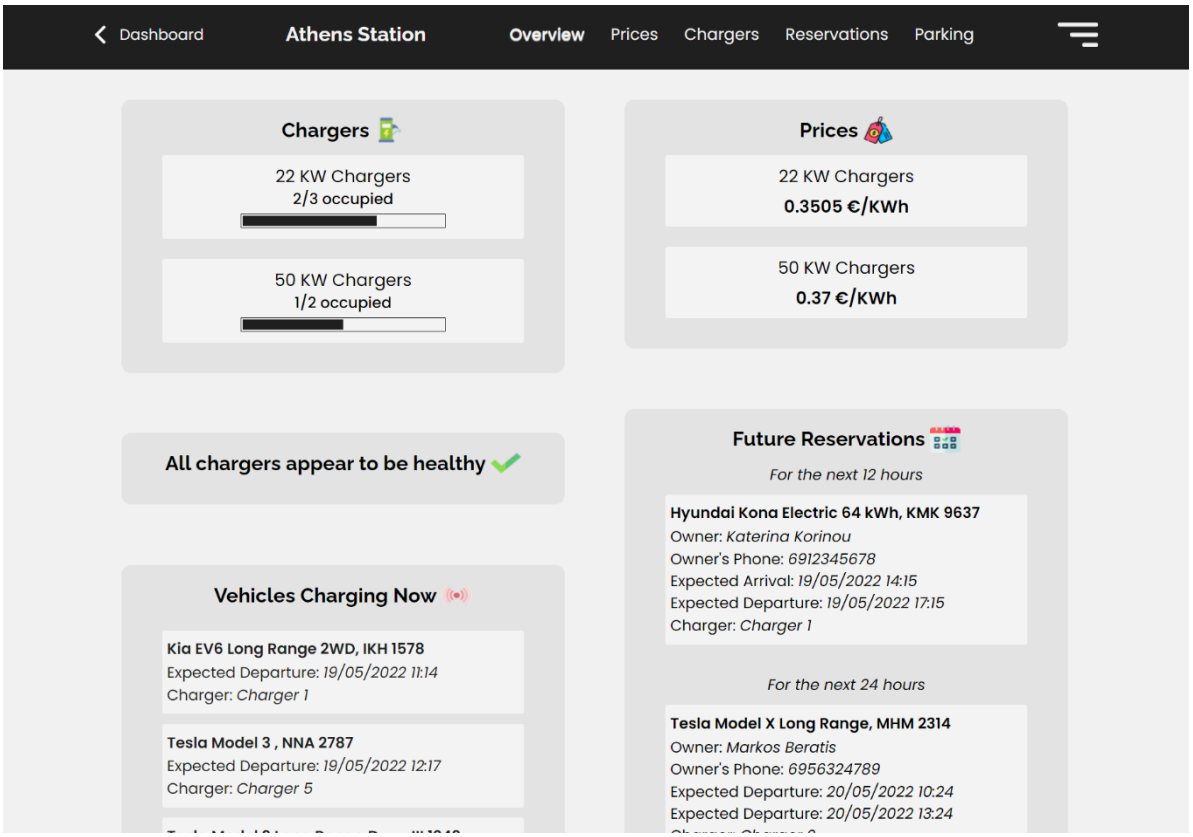
Εικόνα 48: Σελίδα ρυθμίσεων που περιλαμβάνει τα StationRequests

Charging Operator Frontend: Station Overview

Σε αυτό και στα επόμενα παραδείγματα χρήσης θα φανούν οι κύριες χρήσεις της εφαρμογής, καθώς στα προηγούμενα παραδείγματα χρήσης περιγράφονταν διαδικασίες που δεν θα επαναλαμβάνονται πολλαπλές φορές από τους operators (παραδείγματος χάριν η προσθήκη ενός σταθμού θα γίνει λίγες φορές). Σε αυτό το παράδειγμα χρήσης θα παρουσιαστεί η σελίδα επισκόπησης ενός σταθμού. Αυτή είναι η σελίδα που ο operator πιθανώς θα βλέπει πιο συχνά καθώς σε αυτήν παρουσιάζονται τα πιο σημαντικά στοιχεία για έναν σταθμό, όπως είναι ο αριθμός των γεμάτων και άδειων φορτιστών, οι τιμές πώλησης, το health των φορτιστών, τα αυτοκίνητα που βρίσκονται στο σταθμό και φορτίζουν, καθώς και μελλοντικές κρατήσεις φόρτισης που έχουν γίνει προς τον σταθμό.

Στην εικόνα 49 φαίνεται η επισκόπηση (Overview) του σταθμού που ονομάζεται «Athens Station».

Να σημειωθεί ότι η σελίδα αυτή κάνει αυτόματα ανανέωση των δεδομένων που εμφανίζονται κάθε 5 δευτερόλεπτα, επικοινωνώντας με το backend. Έτσι, οι operators βλέπουν πάντα τα πιο πρόσφατα δεδομένα και όχι δεδομένα που ενδεχομένως δεν ισχύουν πλέον.

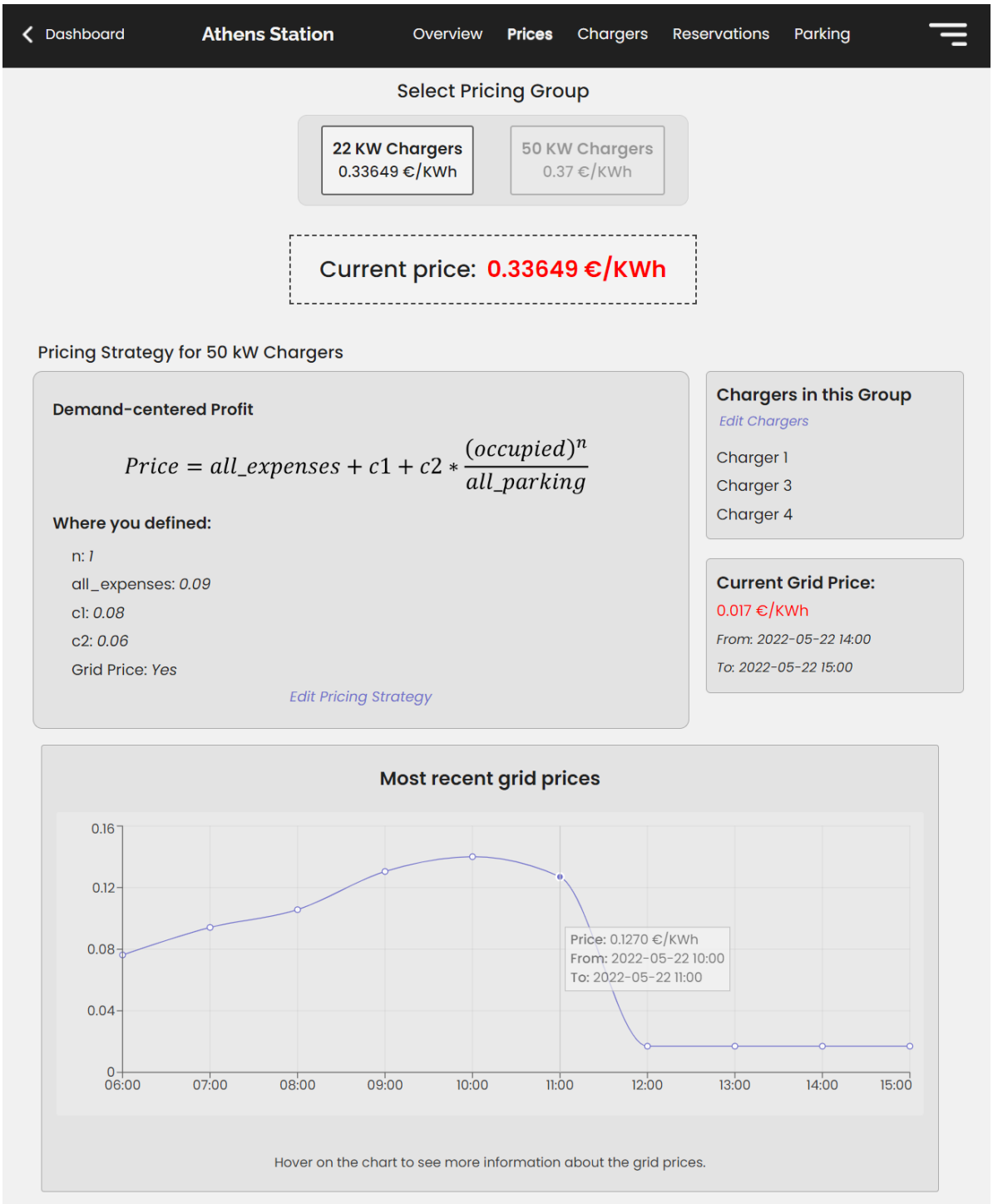


Εικόνα 49: Σελίδα επισκόπησης ενός σταθμού

Charging Operator Frontend: Station Prices and Chargers

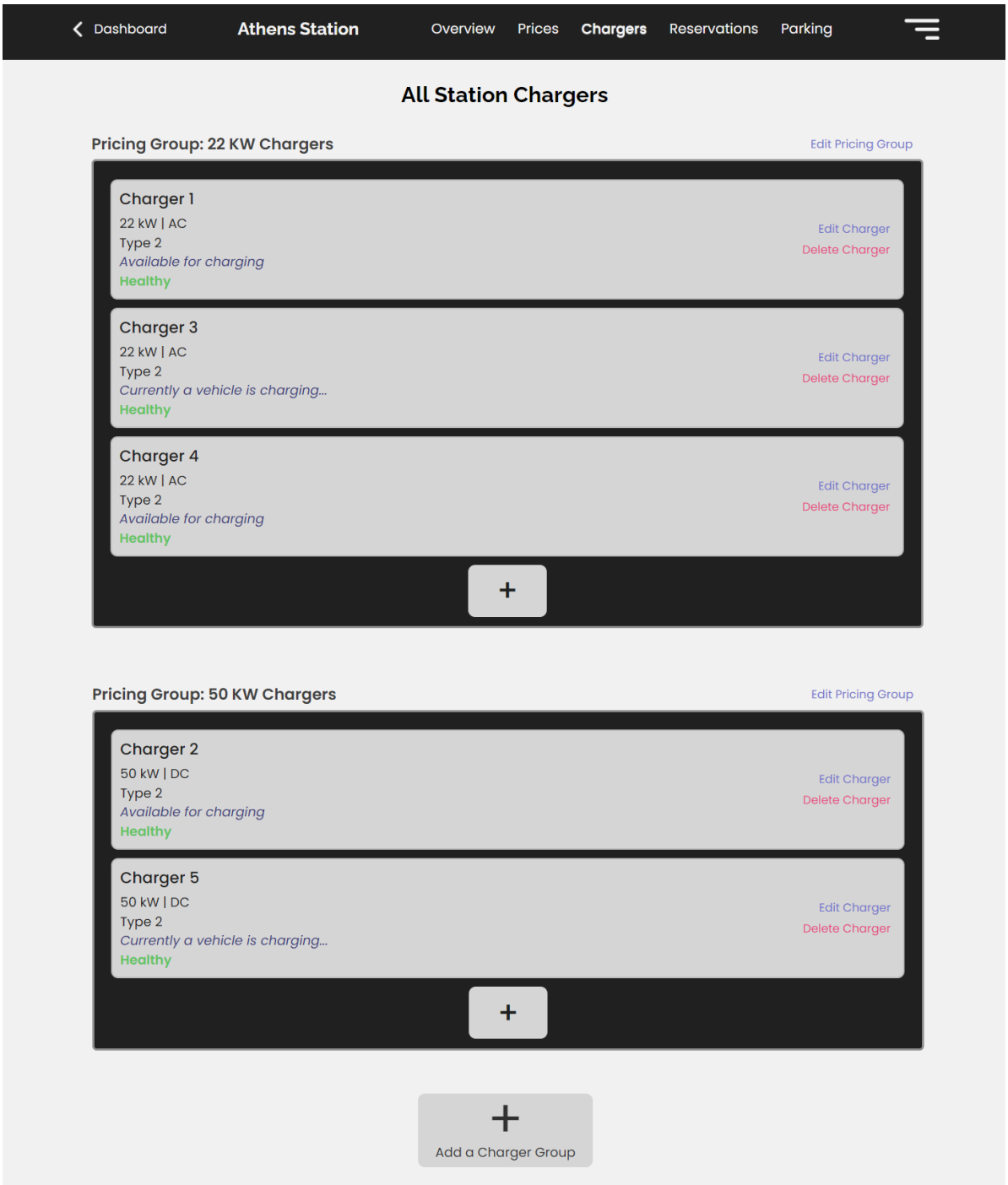
Σε αυτό το παράδειγμα χρήσης θα παρουσιαστούν οι σελίδες που εμφανίζουν τα Pricing Groups και τους φορτιστές του σταθμού.

Στην εικόνα 50 φαίνεται η σελίδα Prices του Charging Operator Frontend που παρουσιάζει τα Pricing Groups του σταθμού και τις τιμές που προσφέρουν για φόρτιση τη δεδομένη χρονική στιγμή. Ο operator μπορεί εύκολα να δει τις μεθόδους τιμολόγησης και τις παραμέτρους αυτών. Μπορεί να τροποποιήσει τις μεθόδους τιμολόγησης ενός Pricing Group κάνοντας click στο «Edit Pricing Method». Αυτό θα ανοίγει το Modal που παρουσιάστηκε στην εικόνα 45. Ο operator μπορεί να τροποποιήσει τους φορτιστές ενός Pricing Group κάνοντας click στο «Edit Chargers». Τέλος, ο operator μπορεί να δει ενός πιο πρόσφατες τιμές ηλεκτρικής ενέργειας, μέσω ενός διαγράμματος που παρουσιάζει την τιμή ανά KWh ως συνάρτηση του χρόνου. Οι τιμές αυτές λαμβάνονται από το Grid Price Collector του backend, που επικοινωνεί με το εξωτερικό σύστημα Fingrid API.



Εικόνα 50: Σελίδα διαχείρισης των Pricing Groups ενός σταθμού

Στη συνέχεια, η εικόνα 51 φαίνεται η σελίδα Chargers του Charging Operator Frontend που παρουσιάζει επίσης φορτιστές του σταθμού. Ο operator μπορεί να αλλάξει τα στοιχεία ενός Pricing Group ή να δημιουργήσει ένα νέο κάνοντας click στα «Edit Pricing Group» και «Add a Charger Group» αντίστοιχα. Αυτό θα ανοίξει και πάλι το Modal της εικόνας 45 (έγινε επαναχρησιμοποίηση του ίδιου Modal React component σε όλες αυτές τις περιπτώσεις). Επίσης, μπορεί να προσθέσει, να τροποποιήσει ή να διαγράψει φορτιστές, κάνοντας click στο «+», στο «Edit Charger» και στο «Delete Charger» αντίστοιχα.



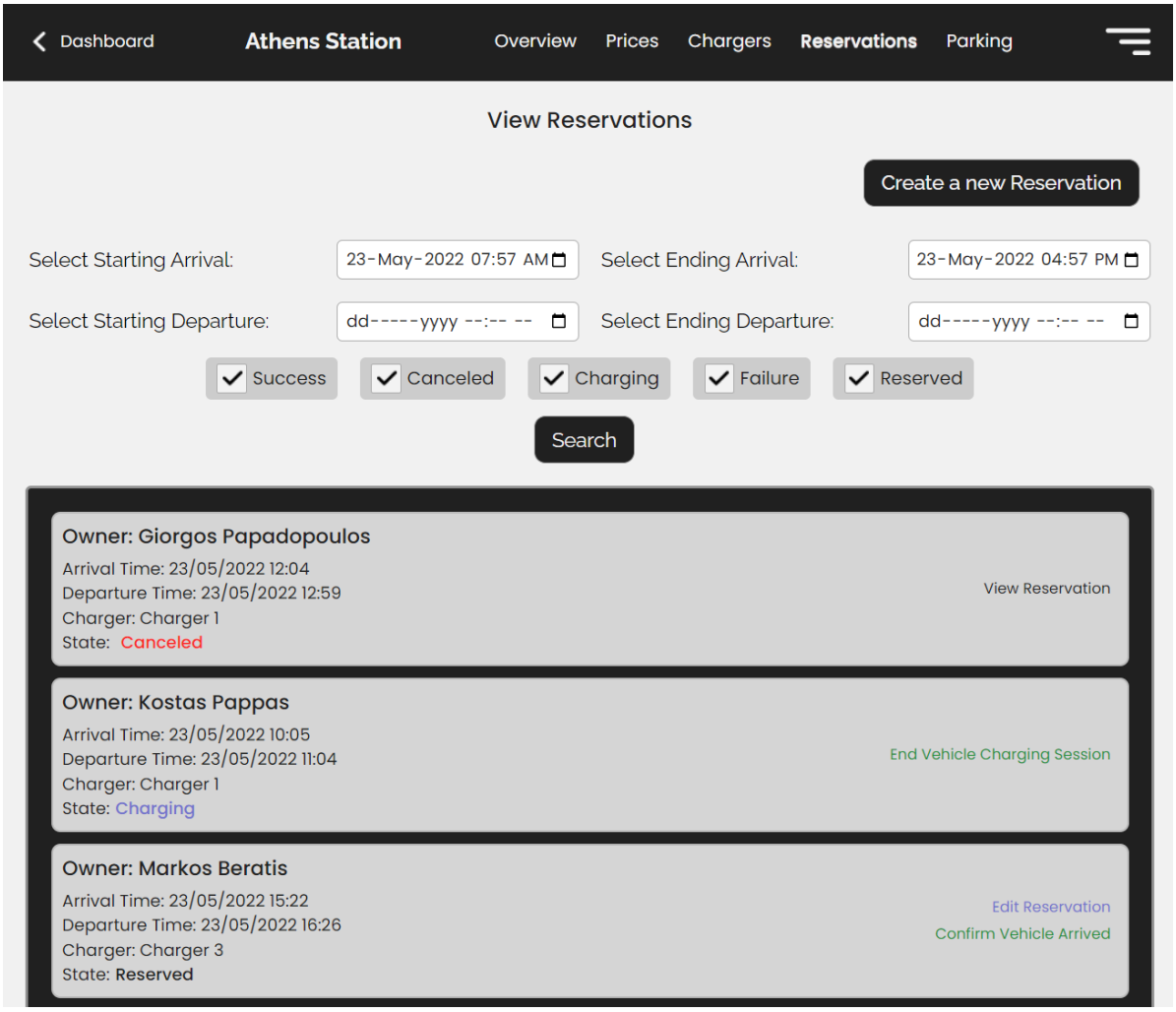
Εικόνα 51: Σελίδα διαχείρισης φορτιστών ενός σταθμού

Έτσι, ο operator έχει τον πλήρη έλεγχο των Pricing Groups και των φορτιστών του κάθε σταθμού του.

Charging Operator Frontend: Station Reservations

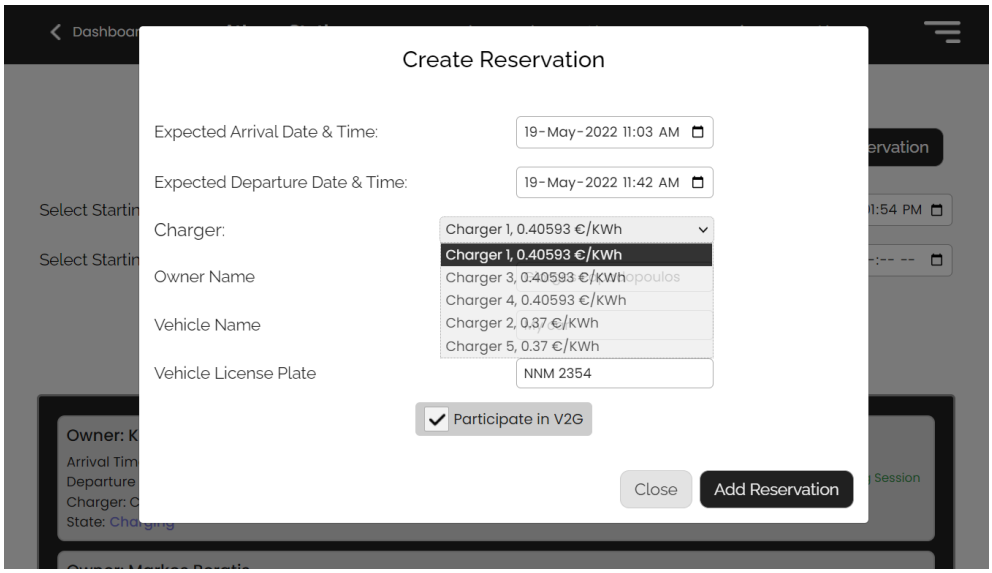
Σε αυτό το παράδειγμα χρήσης θα παρουσιαστεί η σελίδα που διαχειρίζεται τις κρατήσεις φόρτισης του σταθμού.

Στην εικόνα 52 φαίνεται η σελίδα που δείχνει τις κρατήσεις του σταθμού. Ο operator μπορεί να κάνει σύνθετες αναζητήσεις για διαφορετικές ημερομηνίες και καταστάσεις κράτησης.



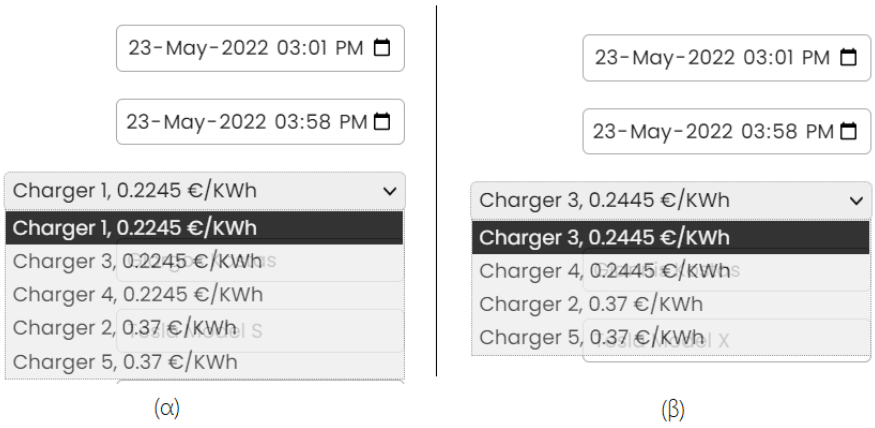
Εικόνα 52: Σελίδα διαχείρισης κρατήσεων ενός σταθμού

Εκτός σύνθετων αναζητήσεων, ο operator μπορεί να δημιουργήσει νέες κρατήσεις για το σταθμό. Κάνοντας click στο κουμπί «Create a new Reservation», ανοίγει το Modal που φαίνεται στην εικόνα 53. Σε αυτό το Modal, ο operator ορίζει την αναμενόμενη ώρα άφιξης και αναχώρησης ενός ιδιοκτήτη, και αυτομάτως ενημερώνονται οι διαθέσιμοι φορτιστές.



Εικόνα 53: Modal δημιουργίας κράτησης

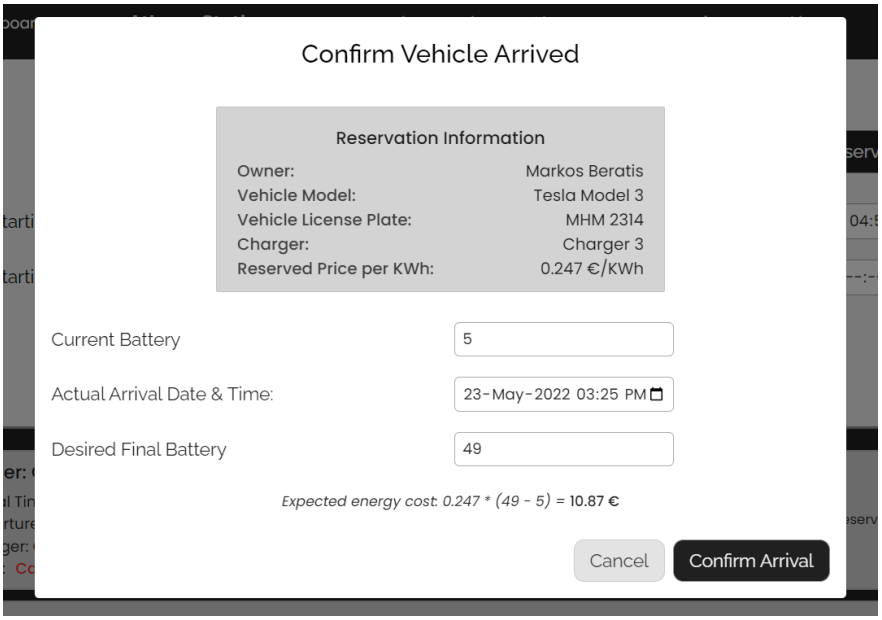
Ως έλεγχο για την ορθότητα των δεδομένων που εμφανίζονται στο συγκεκριμένο Modal (διαθέσιμοι φορτιστές και τιμές) θα γίνει ένα πείραμα όπως φαίνεται στην εικόνα 54. Αρχικά, γίνεται μία κράτηση για την ημερομηνία και ώρα που φαίνεται στην εικόνα 54(α). Στη συνέχεια, γίνεται δοκιμή να πραγματοποιηθεί μία κράτηση την ίδια ημερομηνία και ώρα, όπως φαίνεται στην εικόνα 54(β). Είναι εμφανές ότι ο «Charger 1» δεν εμφανίζεται πλέον ως διαθέσιμος, αφού υπάρχει άλλη κράτηση προς αυτόν. Επίσης, οι τιμές πώλησης αλλάζουν όταν πραγματοποιείται μία κράτηση, το οποίο είναι το αναμενόμενο, καθώς το Pricing Group που περιέχει τους φορτιστές «Charger 1», «Charger 3» και «Charger 4» έχει μέθοδο Demand-Centered Profit. Προφανώς, tests για την ορθότητα των δεδομένων που παράγονται από το backend και στη συνέχεια στέλνονται στο Frontend, έχουν γραφεί και αναλύθηκαν στο υποκεφάλαιο «Backend Testing» το κεφαλαίου 4.



Εικόνα 54: Δημιουργία δύο κρατήσεων στην ίδια χρονική περίοδο

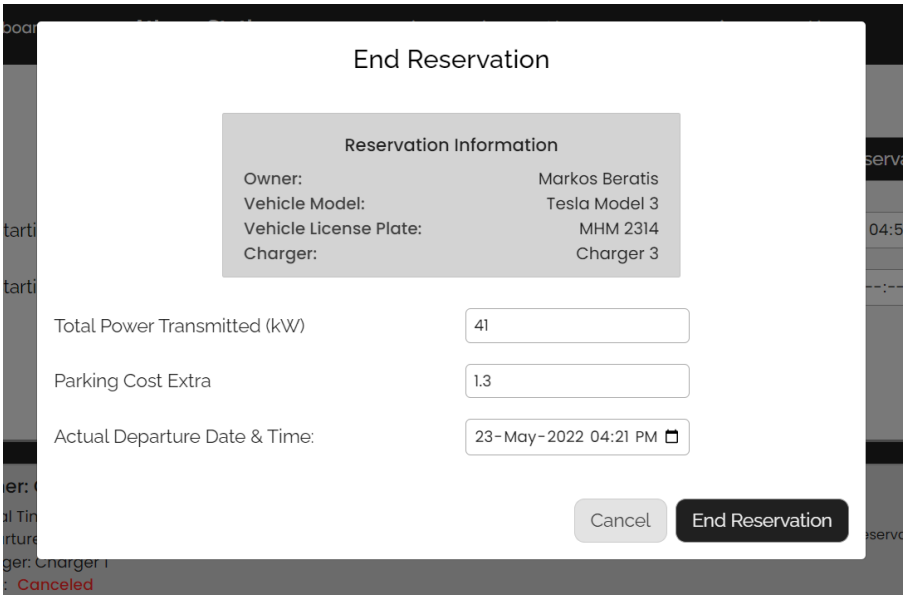
Ακολούθως, μετά τη δημιουργία μίας κράτησης, είναι δυνατό να ενημερωθούν τα στοιχεία της κράτησης ή να ακυρωθεί η κράτηση, πατώντας το κουμπί «Edit Reservation», όπως φαίνεται στην εικόνα 52.

Όταν το αυτοκίνητο φτάνει στο σταθμό και είναι έτοιμο να φορτίσει, τότε μπορεί να πατηθεί το κουμπί «Confirm Vehicle Arrived» της εικόνας 52, το οποίο θα ανοίξει το Modal της εικόνας 55. Εκεί ο operator ορίζει την τωρινή, την επιθυμητή τελική μπαταρία και την ημερομηνία άφιξης του οχήματος.



Εικόνα 55: Modal για την επιβεβαίωση άφιξης ενός οχήματος

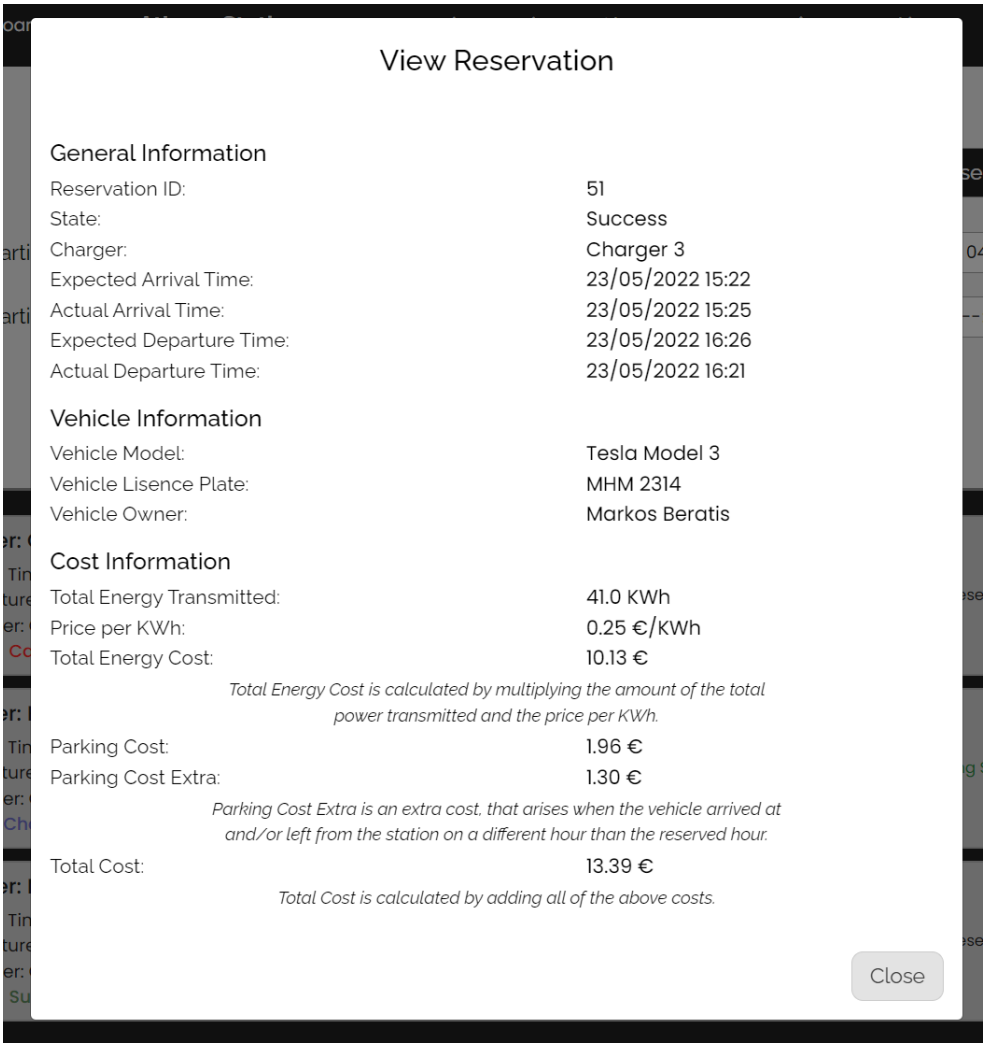
Επίσης, όταν ένα αυτοκίνητο τελειώσει τη φόρτιση του, ο operator μπορεί να πατήσει το κουμπί «End Vehicle Charging Session» της εικόνας 52, το οποίο ανοίγει το Modal της εικόνας 56. Εκεί ο operator ορίζει τη συνολική ενέργεια που μεταδόθηκε, (προαιρετικά) κάποιο επιπλέον κόστος parking, καθώς και την ώρα που αναχωρεί από το σταθμό το όχημα.



Εικόνα 56: Modal για τον τερματισμό μίας κράτησης

Να σημειωθεί ότι τα παραπάνω δύο Modals (εικόνες 55, 56), θα μπορούσαν να μην υπάρχουν αν όντως το backend ήταν συνδεδεμένο με ειδικούς αισθητήρες ή IoT συσκευές που θα υπήρχαν όντως στους σταθμούς φόρτισης. Έτσι, θα αυτοματοποιούταν ακόμα περισσότερο η διαδικασία των κρατήσεων.

Τέλος, κάνοντας click στο «View Reservation» της εικόνας 52 θα εμφανιστεί το Modal της εικόνας 57. Εκεί ο operator μπορεί να δει τις πληροφορίες μίας κράτησης, την κατάσταση, τον φορτιστή, τις ημερομηνίες άφιξης και αναχώρησης, πληροφορίες για τον ιδιοκτήτη και το όχημα, πληροφορίες σχετικά με την ενέργεια που καταναλώθηκε καθώς και αναλυτική περιγραφή της κοστολόγησης που έγινε προς τον ιδιοκτήτη.

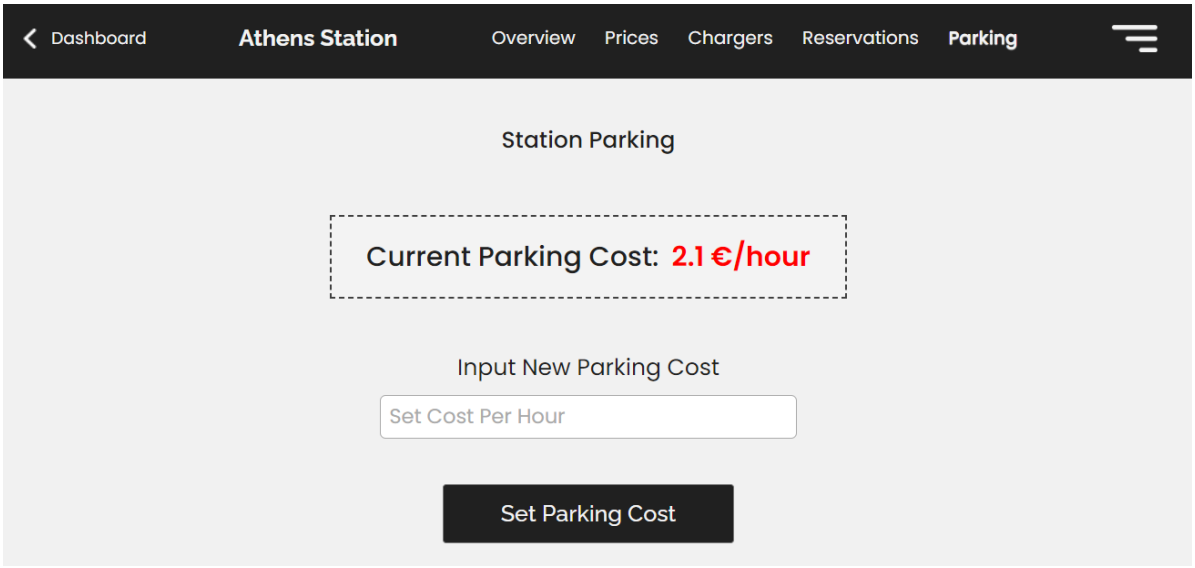


Εικόνα 57: Modal για την παρουσίαση των πληροφοριών μίας κράτησης

Από όλα τα παραπάνω έγινε κατανοητό ότι ο operator του σταθμού έχει τον πλήρη έλεγχο των κρατήσεων φόρτισης. Σε επόμενο υποκεφάλαιο, θα παρουσιαστεί και η δημιουργία κράτησης από την πλευρά του EV Owner.

Charging Operator Frontend: Station Parking

Σε αυτό το παράδειγμα χρήσης θα παρουσιαστεί η σελίδα που ο operator θέτει το κόστος parking του σταθμού. Στην εικόνα 58 φαίνεται η σελίδα Parking, όπου ο operator μπορεί να θέσει το κόστος Parking για το σταθμό.



Εικόνα 58: Σελίδα διαχείρισης κόστους parking ενός σταθμού

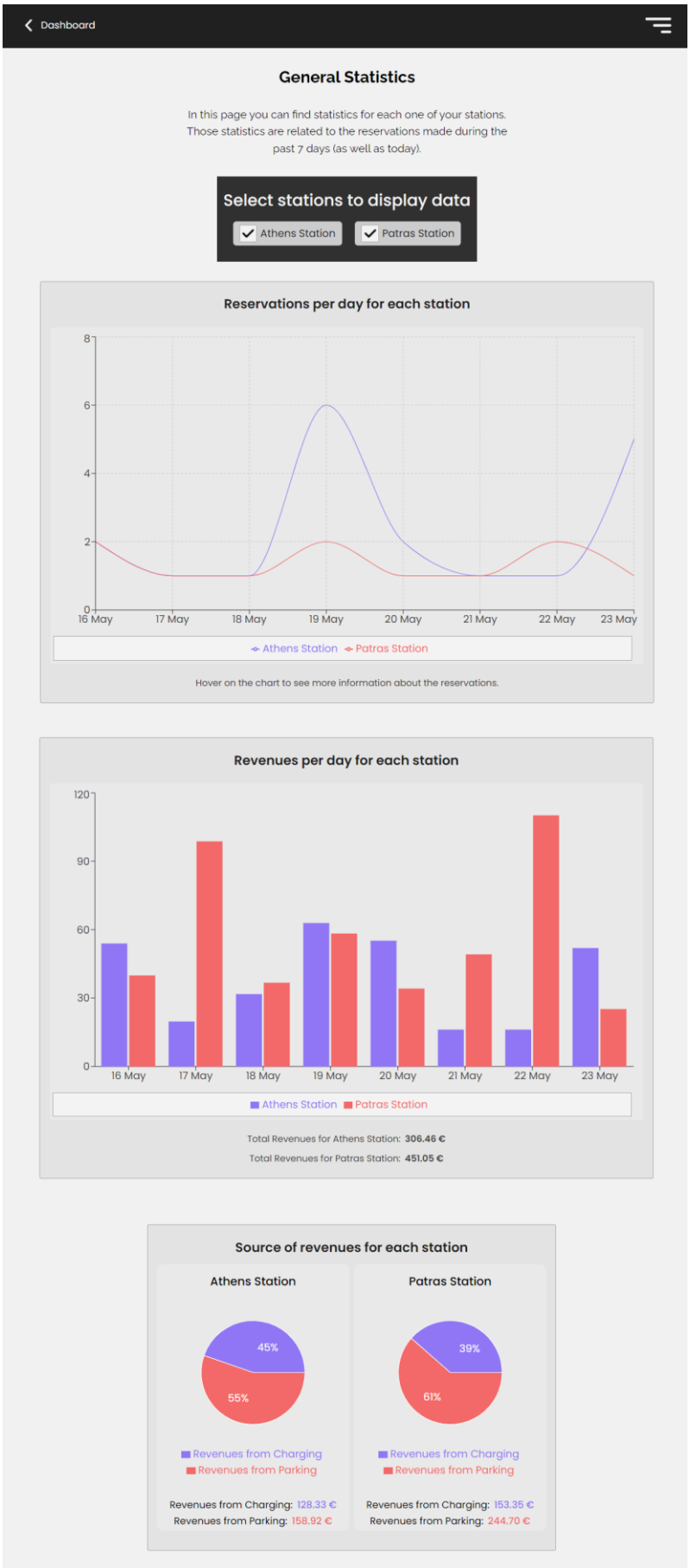
Να σημειωθεί ότι επιλέχθηκε για λόγους απλότητας ο operator να μπορεί να θέσει μόνο ένα κόστος parking. Ωστόσο, το database υποστηρίζει και πιο περίπλοκο μηχανισμό για κοστολόγηση του parking. Παραδείγματος χάριν, θα μπορούσε ο ιδιοκτήτης να θέσει διαφορετικές τιμές parking για διαφορετικές ώρες ή ημέρες ή μήνες. Κάτι τέτοιο δεν υλοποιήθηκε στην πλευρά του Frontend στο πλαίσιο αυτής της διπλωματικής.

Charging Operator Frontend: Stations Statistics

Μία επίσης σημαντική λειτουργικότητα της εφαρμογής είναι η εξαγωγή χρήσιμων στατιστικών για τους σταθμούς ενός operator. Έχοντας ένα τέτοιο εργαλείο ο operator θα βρίσκεται σε θέση να παίρνει πιο έξυπνες αποφάσεις σχετικά με τους σταθμούς του.

Συγκεκριμένα, υλοποιήθηκε η εξαγωγή στατιστικών σχετικά με τις κρατήσεις φόρτισης κάθε σταθμού, τα έσοδα ανά ημέρα και τον καταμερισμό των εσόδων μεταξύ κοστολόγησης φόρτισης και Parking. Έτσι ο operator γνωρίζει καλύτερα τι συμβαίνει σε κάθε σταθμό του και είναι σε θέση να συγκρίνει άμεσα πληροφορίες για διαφορετικούς σταθμούς.

Στην εικόνα 59, φαίνεται η σελίδα των στατιστικών που παρουσιάζει τα δεδομένα που περιεγράφηκαν για τους δύο σταθμούς ενός operator.

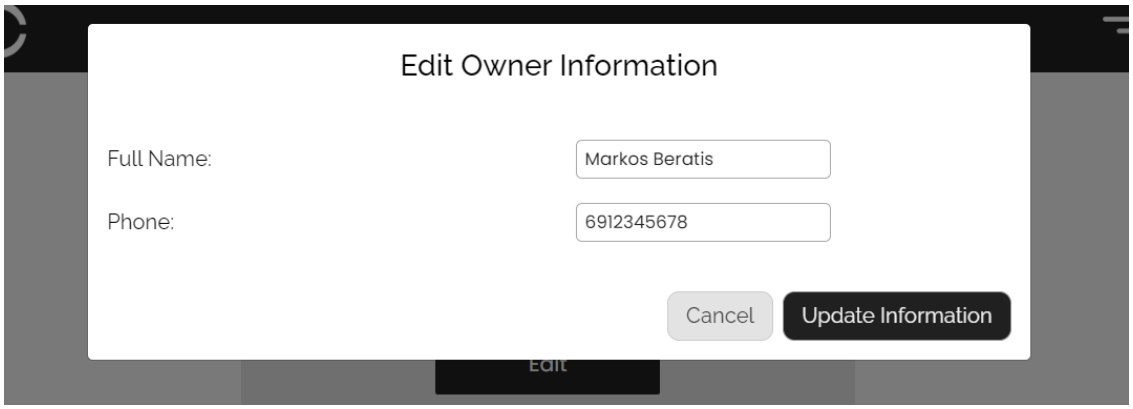


Εικόνα 59: Σελίδα παρουσίασης στατιστικών

EV Owner Frontend: Dashboard

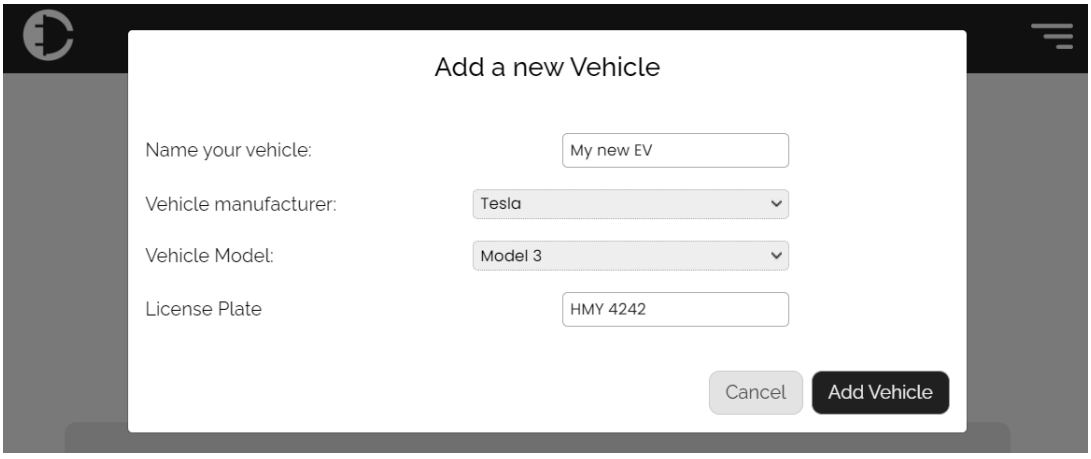
Σε αυτό το παράδειγμα χρήσης θα παρουσιαστεί η σελίδα Dashboard που εμφανίζεται στους EV Owners μετά τη σύνδεση στο λογαριασμό τους. Το dashboard αυτό παρουσιάστηκε στην εικόνα 40. Σε αυτή τη σελίδα ο ιδιοκτήτης ηλεκτρικού οχήματος μπορεί να τροποποιήσει τις πληροφορίες του, να διαχειριστεί τα οχήματά του και να δει κρατήσεις φόρτισης που έχει δημιουργήσει. Επίσης, μπορεί να μεταβεί στη σελίδα για τη δημιουργία μίας νέας κράτησης πατώντας το κουμπί «Create a new Reservation».

Για να τροποποιήσει τα στοιχεία του ο ιδιοκτήτης μπορεί να πατήσει το κουμπί «Edit» της εικόνας 40. Τότε θα εμφανιστεί το Modal της εικόνα 60. Εκεί ο ιδιοκτήτης μπορεί να αλλάξει το όνομα και το τηλέφωνό του σε περίπτωση που κάποιο από αυτά είναι λάθος.



Εικόνα 60: Modal για την τροποποίηση των στοιχείων ενός ιδιοκτήτη οχήματος

Επίσης, ο ιδιοκτήτης μπορεί να προσθέσει ένα νέο όχημα με όλες τις απαραίτητες πληροφορίες κάνοντας click στο κουμπί «+» της εικόνας 40. Τότε, θα εμφανιστεί το Modal της εικόνας 61.

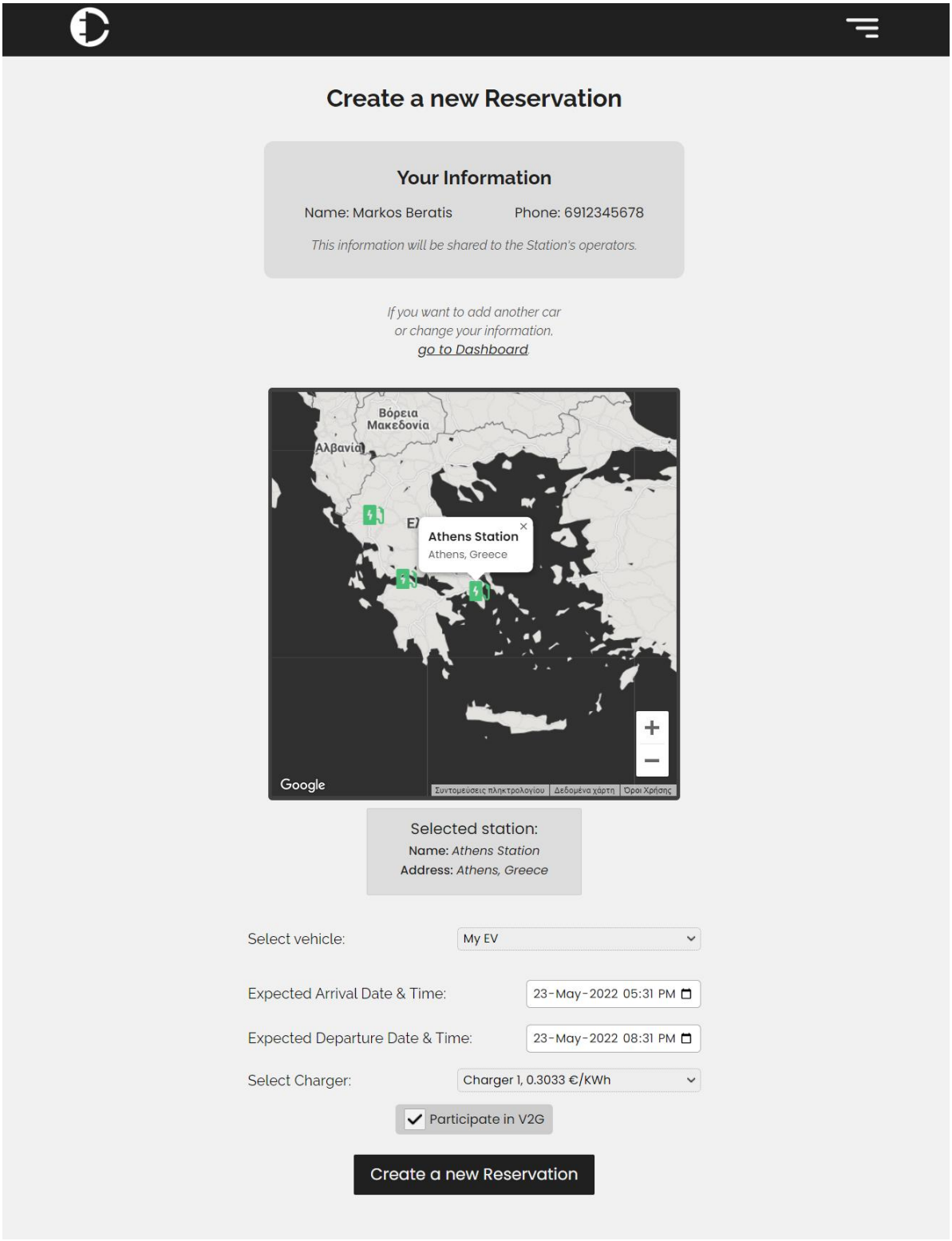


Εικόνα 61: Modal για τη δημιουργία ενός νέου οχήματος

Να σημειωθεί ότι οι κατασκευαστές των οχημάτων και τα μοντέλα του κάθε κατασκευαστή προήλθαν από ένα database ηλεκτρικών οχημάτων το οποίο έχει αναφερθεί στη βιβλιογραφία [15].

EV Owner Frontend: Create Reservation

Ως τελευταίο παράδειγμα χρήσης θα παρουσιαστεί η σελίδα Create a new Reservation των EV Owners. Στην εικόνα 62 φαίνεται αυτή η σελίδα.



Εικόνα 62: Σελίδα για τη δημιουργία κράτησης ενός ιδιοκτήτη οχήματος

Στη σελίδα αυτή οι ιδιοκτήτες μπορούν να δημιουργήσουν μία νέα κράτηση φόρτισης. Αρχικά, διαλέγουν το σταθμό που τους ενδιαφέρει. Στη συνέχεια, επιλέγουν το αυτοκίνητο και τις αναμενόμενες ημερομηνίες άφιξης και αναχώρησης. Ακολούθως, μπορούν να δουν τους διαθέσιμους φορτιστές αυτού του σταθμού και τις τιμές αυτών. Τέλος, μπορούν να επιλέξουν αν επιθυμούν να συμμετέχουν ή όχι σε έξυπνη Vehicle-to-Grid φόρτιση, εφόσον υποστηρίζεται από τον συγκεκριμένο σταθμό.

Επομένως, είναι εμφανές ότι με αυτή τη σελίδα οι ιδιοκτήτες οχημάτων μπορούν να δημιουργήσουν εύκολα κρατήσεις για τα ηλεκτρικά τους αυτοκίνητα, καθώς επίσης τους παρέχεται ευελιξία για να διαλέξουν οποιονδήποτε από τους διαθέσιμους σταθμούς, οι οποίοι ενδεχομένως προσφέρουν διαφορετικές τιμές φόρτισης.

Κεφάλαιο 6: Συμπεράσματα

Συμπεράσματα

Η δημιουργία του συστήματος που αναλύθηκε σε αυτήν τη διπλωματική περιείχε αρκετές προκλήσεις που εκ των προτέρων δεν ήταν προβλέψιμες. Το σύστημα που υλοποιήθηκε περιέχει πολλά components, τα οποία είναι απαραίτητα για την ορθή λειτουργία της εφαρμογής. Παραδείγματος χάριν, το component Grid Price Collector, για την συλλογή δεδομένων σχετικά με την τιμή της ηλεκτρικής ενέργειας, δεν είχε υπολογιστεί στον αρχικό σχεδιασμό του συστήματος. Ωστόσο, μέσω πολλαπλών επαναλήψεων στο σχεδιασμό του συστήματος, έγιναν διάφορες αλλαγές ώστε να επιτευχθεί ο σχεδιασμός που αναλύθηκε στο κεφάλαιο 3.

Παρά τις προκλήσεις που είχε ο σχεδιασμός και η υλοποίηση αυτού του συστήματος, η τελική web εφαρμογή που προέκυψε είναι ιδιαίτερα χρήσιμη για τους operators και για τους ιδιοκτήτες ηλεκτρικών οχημάτων. Χρησιμοποιώντας αυτήν την εφαρμογή οι operators έχουν τον πλήρη έλεγχο των σταθμών φόρτισης στους οποίους είναι αρμόδιοι. Ταυτοχρόνως, οι ιδιοκτήτες είναι σε θέση να κάνουν κρατήσεις φόρτισης με αρκετά εύκολο τρόπο. Επομένως, το σύστημα που σχεδιάστηκε παρέχει πλήρως τις βασικές λειτουργικότητες που ενδιαφέρουν τους operators και τους ιδιοκτήτες οχημάτων, ώστε η διαδικασία φόρτισης ενός ηλεκτρικού οχήματος σε ένα σταθμό φόρτισης να είναι όσο το δυνατόν πιο ομαλή.

Επόμενα βήματα

Στην παραπάνω σχεδίαση και υλοποίηση του συστήματος δεν ήταν εφικτό να επιτευχθεί η κάλυψη όλων των πιθανών πτυχών του συστήματος. Γι' αυτό παρακάτω παρατίθενται κάποια πιθανά επόμενα βήματα που θα μπορούσαν να συμπληρώσουν τη λειτουργικότητα της εφαρμογής:

- Περεταίρω επέκταση του EV Owner Frontend: Εξαιτίας του γεγονότος ότι ο στόχος της εφαρμογής που σχεδιάστηκε ήταν η εστίαση στην πλευρά των operators, το EV Owner Frontend, δηλαδή το Frontend της εφαρμογής που χρησιμοποιούν οι ιδιοκτήτες ηλεκτρικών οχημάτων, θα μπορούσε να επεκταθεί επιπλέον. Παραδείγματος χάριν θα μπορούσε να υπάρχει σελίδα με στατιστικά για τους ιδιοκτήτες των οχημάτων. Επίσης, θα μπορούσε να παρέχεται μία σελίδα όπου θα γίνεται πιο εύκολα σύγκριση τιμών μεταξύ σταθμών.
- Δυνατότητα καθορισμού πιο περίπλοκων station parking costs: Τα parking costs ενός σταθμού χρησιμοποιούνται για τον καθορισμό της τιμής parking (€/ώρα) που πρέπει να πληρώσει ένα όχημα για τη διάρκεια που βρίσκεται στο σταθμό. Στην τωρινή υλοποίηση, ο operator μπορεί να θέσει μόνο ένα συγκεκριμένο parking cost στο σταθμό. Αυτό που προτείνεται ως επόμενο βήμα είναι ο καθορισμός parking costs τα οποία είναι διαφορετικά ανά μήνα, ημέρα ή ακόμα και ώρα της ημέρας. Έτσι, πχ ημέρες όπου παρουσιάζεται αυξημένη κίνηση, ο operator μπορεί να θέτει υψηλότερη τιμή ώστε να μεγιστοποιήσει το κέρδος του σταθμού.
- Υλοποίηση καλύτερων αλγορίθμων για τον υπολογισμό μελλοντική τιμής της ηλεκτρικής ενέργειας: Όπως αναλύθηκε στο κεφάλαιο 3, ο υπολογισμός κόστους της ηλεκτρικής ενέργειας που παράγεται για μελλοντικές τιμές της ενέργειας γίνεται με την υλοποίηση ενός απλού αλγορίθμου. Σαν επόμενο βήμα, θα μπορούσαν να

υλοποιηθούν πιο σύνθετοι και πιο ακριβείς αλγόριθμοι πρόβλεψης τιμής ηλεκτρικής ενέργειας.

- Απεικόνιση επιπλέον στατιστικών για τους operators: Παρόλο που δημιουργήθηκε σελίδα με στατιστικά για τους operators, θα μπορούσε ως επόμενο βήμα να παρέχονται στους operators επιπλέον στατιστικά από αυτά που τώρα απεικονίζονται. Επίσης, θα μπορούσε να δίνεται η δυνατότητα στον operator να επιλέγει το εύρος των ημερομηνιών για το οποίο θέλει να δει στατιστικά.
- Επιπλέον επιλογές για τη διαχείριση κρατήσεων από τον operator: Παρόλο που στη σελίδα Reservations ο operator είναι σε θέση να διαχειριστεί σε μεγάλο βαθμό κάθε κράτηση και κάθε φόρτιση στο σταθμό, θα μπορούσε να γίνει επιπλέον επέκταση της λειτουργικότητας. Παραδείγματος χάριν, θα μπορούσε να δίνεται η δυνατότητα στον operator να αναφέρει τους λόγους για τους οποίους μία κράτηση έγινε Canceled. Επίσης, θα μπορούσε να του δίνεται η δυνατότητα να δηλώσει ότι μία κράτηση απέτυχε, και για ποιους λόγους απέτυχε. Έτσι, ο operator θα είναι σε θέση να γνωρίζει περισσότερες πληροφορίες σχετικά με τις κρατήσεις στο σταθμό του.

Πηγαίος κώδικας

Όλος ο πηγαίος κώδικας είναι ανεβασμένος στο GitHub σε repository που είναι διαθέσιμο δημόσια. Το link για το GitHub repository είναι το εξής:

https://github.com/markosbaratsas/charging-operator

Εκεί περιέχονται οι απαραίτητες πληροφορίες για την εγκατάσταση του συστήματος σε οποιοδήποτε μηχάνημα, αφού περιέχονται τα αρχεία requirements.txt (για το backend), και package.json/package-lock.json (για το frontend).

Βιβλιογραφία

[1] Daniel Kirschen, Goran Strbac (2004) *Fundamentals of Power System Economics*

[2] Julio A. Sanguesa, Vicente Torres-Sanz, Piedad Garrido, Francisco J. Martinez, Johann M. Marquez-Barja (2021) *A Review on Electric Vehicles: Technologies and Challenges*

[3] Hunt S, Shuttleworth G (1996) *Competition and Choice in Electricity*

[4] Alireza Khaksari, Georgios Tsaousoglou, Prodromos Makris, Konstantinos Steriotis, Nikolaos Efthymiopoulos, Emmanouel Varvarigos (2021) *Sizing of electric vehicle charging stations with smart charging capabilities and quality of service requirements*

[5] Kang Miao Tan, Vigna K. Ramachandaramurthy, Jia Ying Yong (2016) *Integration of electric vehicles in smart grid: A review on vehicle to grid technologies and optimization techniques*

[6] Fareed Ahmad, Atif Iqbal, Imtiaz Ashraf, Mousa Marzband, Irfan khan (2022) *Optimal location of electric vehicle charging station and its impact on distribution network: A review*

[7] Mohammad shafiei, Ali Ghasemi-Marzbali (2022) *Fast-charging station for electric vehicles, challenges and issues: A comprehensive review*

[8] Zhenya Jia, Xueliang Huanga (2018) *Plug-in electric vehicle charging infrastructure deployment of China towards 2020: Policies, methodologies, and challenges*

[9] Meta Platforms, Inc (2022) *https://reactjs.org/*

[10] Django Software Foundation (2005-2022) *https://www.djangoproject.com/*

[11] *https://www.django-rest-framework.org/*

[12] *https://code.google.com/archive/p/implementing-rest/wikis/DjangoRESTframework.wiki*

[13] *https://docs.celeryq.dev/en/stable/*

[14] *https://swagger.io/*

[15] EV Database (2022) *https://ev-database.org/*

[16] *https://marvelapp.com/*